ON THE TIME AND TAPE COMPLEXITY OF LANGUAGES*

Harry Bowen Hunt III

TR 73-182

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University for the Degree of

Doctor of Philosophy

August 1973

Department of Computer Science
Cornell University
Ithaca, New York 14850

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

ON THE TIME AND TAPE COMPLEXITY OF LANGUAGES

Harry Bowen Hunt 111, Ph.D.

Cornell University, 1973

ABSTRACT

We investigate the following:

(1) the relationship between the classes of languages accepted by deterministic and nondeterministic polynomial time bounded Turing machines;

(2) the relationship between the classes of languages accepted by deterministic and nondeterministic linear bounded automata;

(3) sufficient conditions for undecidability of linguistic predicates; and

(4) the time and space complexity of several predicates on the regular sets.

We show that the set $\{R \mid R$ is a $(\cup, \cdot, *, \cap)$ regular expression and $L(R) = \{0,1\}^*\}$ is not recognizable by any polynomial space bounded Turing machine. We also find conditions which guarantee that any predicate on the regular sets satisfying them is as hard to decide as emptiness or equivalence.

BIOGRAPHICAL SKETCH

Harry Bowen Hunt III was born in Sharon, Pennsylvania on July 1, 1949. He was graduated from Case Western Reserve University in June, 1971, with the degrees of Bachelor and Master of Science (both in Mathematics.) He entered the graduate school at Cornell University in July, 1971.

to my parents and grandparents

ACKNOWLEDGEMENTS

First, I would like to thank my thesis advisor Professor John E. Hopcroft for his help, encouragement, and time. I would also like to thank Professor Juris Hartmanis. Several of the results of Sections 2.0 and 2.1 were arrived at jointly with him. Thanks are also due to Professors Michael Morley and John Dennis for serving on my special committee.

Finally, this research was supported by a National Science Foundation Graduate Fellowship in Computer Science.

TABLE OF CONTENTS

Chapter 1 INTRODUCTION

1.0 Background

Recently there has been considerable activity in studying the computational complexity of combinatorial problems. Cook [71] and Karp [72] have introduced the concept of a p-reducibility and using it have shown that a variety of combinatorial problems, such as finding if a Boolean form is a tautology, finding if an undirected graph has a Hamilton circuit, and finding if an undirected graph has a clique of size k, have polynomial time bounded deterministic algorithms if and only if the classes of languages accepted by deterministic and nondeterministic polynomial time bounded Turing machines are the same. This result is of interest since it is generally agreed that a problem is tractable on a digital computer only if it has a deterministic polynomial time bounded algorithm. Thus, these results strongly suggest but do not prove that these combinatorial problems are not tractable.

Meyer and Stockmeyer [72] have shown that the equivalence problem for regular expressions has a deterministic polynomial time bounded algorithm if and only if the classes of languages accepted by deterministic polynomial time bounded and nondeterministic polynomial tape bounded Turing machines are the same. They have also shown that the equivalence problem for extended regular expressions (regular expressions with both complements and intersections added) is not elementary recursive. That is there is no algorithm for this problem of time complexity 2^n, 2^{2^n}, $2^{2^{2^n}}$, etc.

We investigate the complexity of the emptiness and equivalence problems for several other varieties of regular and extended regular expressions. The equivalence problem for extended regular expressions, without complements but with intersections, is shown to have no polynomial space bounded algorithm. Several natural conditions on predicates on the regular sets, which imply that a predicate is as hard to decide as emptiness or equivalence, are found. We also study the relationships between the classes of languages accepted by deterministic and nondeterministic polynomial time bounded Turing machines and between the classes of languages accepted by deterministic and nondeterministic linear bounded automata.

1.1 Basic Facts and Definitions

We assume the reader is familiar with both deterministic and nondeterministic time and tape bounded multi-tape Turing machines (abbreviated henceforth by Tms.) There are two varieties of these machines studied in the literature. In the first the input tape is ready only; in the second all tapes are read-write. Our machines are always of the second kind. Thus, a single tape Tm in this thesis is a Turing machine with exactly 1 tape, which is used both for input and work space. Similarly, a 2 tape Tm is a Turing machine with 2 read-write tapes, one of which is used for input. We also assume the reader is familiar with the regular sets, regular grammars, context-free languages and grammars (abbreviated by cfls and cfgs), and both deterministic and nondeterministic linear bounded automata (abbreviated by lba.) Details concerning any of the above may be found in Hopcroft and Ullman [69].

Definition 1.1.0: N denotes the set of nonnegative integers.

Definition 1.1.1: PTIME is the class of all languages (over some countably infinite alphabet $\overline{\Sigma}$) recognized by some deterministic polynomial time bounded Tm.

Definition 1.1.2: NPTIME is the class of all languages (over some countably infinite alphabet $\overline{\Sigma}$) recognized by some nondeterministic polynomial time bounded Tm.

<u>Definition 1.1.3</u>: <u>PTAPE</u> is the class of all languages (over some countably infinite alphabet $\bar{\Sigma}$) recognized by some deterministic or nondeterministic polynomial tape bounded Tm. Savitch [70] has shown that the classes of languages accepted by deterministic and nondeterministic polynomial tape bounded Tms are the same.

<u>Definition 1.1.4</u>: <u>Dtime (T(n))</u> is the class of all languages (over some countably infinite alphabet $\bar{\Sigma}$) recognized by some deterministic T(n) time bounded Tm. Ndtime (T(n)) is defined analogously.

<u>Definition 1.1.5</u>: Let $L(n) \geq n$. <u>Dtape (L(n))</u> is the class of all languages (over some countably infinite alphabet $\bar{\Sigma}$) recognized by some deterministic L(n) tape bounded Tm. <u>Ndtape (L(n))</u> is defined similarly.

<u>Definition 1.1.6</u>: <u>DCSL</u> is the class of all languages (over some countably infinite alphabet $\bar{\Sigma}$) recognized by some deterministic lba. <u>NDCSL</u> is defined analogously.

<u>Definition 1.1.7</u>: Let $\Sigma = \{0,1\}$. Let π be the class of functions from Σ^* into Σ^* computable by some deterministic polynomial time bounded Tm. Let L and M be languages. L is <u>p-reducible</u> to M, written $L \underset{\text{Ptime}}{\leq} M$, if $\exists f \in \pi$ such that $x \in L$ iff $f(x) \in M$. We leave it to the reader to extend this definition to countably infinite alphabets $\overline{\Sigma}$.

<u>Definition 1.1.8</u>: The $(\cup,\cdot,*)$ regular expressions over $\{0,1\}$ are defined recursively as follows:

(a) λ, ϕ, 0, 1 are $(\cup,\cdot,*)$ regular expressions over $\{0,1\}$,

(b) If A and B are $(\cup,\cdot,*)$ regular expressions over $\{0,1\}$, then so are $(A)\cup(B)$, $(A)\cdot(B)$, and $(A)^*$, and

(c) A is a $(\cup,\cdot,*)$ regular expression over $\{0,1\}$ iff it satisfies (a) or (b). $(\cup,\cdot)$, $(\cup,\cdot,\cap)$, $(\cup,\cdot,*,\cap)$, and $(\cup,\cdot,*,\sim)$ regular expressions over $\{0,1\}$ are defined similarly.

<u>Definition 1.1.9</u>: Inequiv $(\cup,\cdot,*) = \{(R_1,R_2) \mid R_1$ and R_2 are $(\cup,\cdot,*)$ regular expressions over $\{0,1\}$ and $L(R_1) \neq L(R_2)\}$. Notempty $(\cup,\cdot,*) = \{R \mid R$ is a $(\cup,\cdot,*)$ regular expression over $\{0,1\}$ and

$L(R) \neq \phi\}$. Equiv, Inequiv, Empty and Notempty are defined analogously for $(\cup,\cdot)$, $(\cup,\cdot,\cap)$, $(\cup,\cdot,*,\cap)$, and $(\cup,\cdot,*,\sim)$ regular expressions over $\{0,1\}$.

Finally, in several places in this thesis we discuss structures such as cfgs over arbitrarily large alphabets. Since we are interested in the computational complexity of languages over fixed size finite alphabets, we must encode such structures. Thus, we talk about the set $G = \{\gamma \mid \gamma$ is the code of a cfg with terminal alphabet $\{0,1\}\}$. We generally encode such objects using binary integers in the standard manner.

1.2 Outline of Results

We feel that a viable theory of algorithms must talk about the complexity of classes of problems not just individual ones. We introduce the intuitive concept of a complexity core, that is, a finite set of conditions such that any problem satisfying them has a complexity bound (lower or upper) derivable from the conditions of the core alone. Thus, a complexity core characterizes the complexity of the class of problems which satisfies its conditions. The problem is to find cores which are both natural and describe the complexity of a broad class of problems. In Chapter 5 we present one such core, which characterizes the minimal complexity of a broad class of predicates on the regular sets, including almost all such predicates studied in the literature.

Finally, we briefly outline the contents of this thesis. In Chapter 2 we find several necessary and sufficient conditions for PTIME to equal NPTIME and for DCSL to equal NDCSL. We find several hardest languages in each nondeterministic class. In Chapter 3, we extend the undecidability theorems appearing in Greibach [68].

In Chapter 4 we characterize the computational complexity of the emptiness and equivalence predicates for $(\cup,\cdot,*)$, $(\cup,\cdot,*,\cap)$, and $(\cup,\cdot,*,\sim)$ regular expressions, and for 2-way deterministic finite automata. In 5 we find conditions which guarantee that a predicate on the regular sets is as hard to decide as emptiness or equivalence. We show that almost all predicates studied in the literature satisfy these conditions. We also find several natural hardest NDCSLs. Chapter 6 is a short conclusion.

Chapter 2 THE CLASSES PTIME, NPTIME, DCSL, AND NDSCL

2.0 Preliminaries

We study the relationships between the classes PTIME, NPTIME, DCSL, and NDCSL. We show that there are hardest languages in both nondeterministic classes. These languages need be nothing more complex than universal time or tape bounded Tms. We show that the relationships between PTIME and NPTIME, and DCSL and NDCSL are strongly reflected in the relationships between Dtime (T(n)) and Ndtime (T(n)), and between Dtape (L(n)) and Ndtape (L(n)) for all functions T(n) and L(n) $\geq$ n.

Definition 2.0.1: L_1 is p-hard if $L_o \in$NPTIME implies $L_o \underset{Ptime}{\leq} L_1$. L_1 is p-complete if it is both p-hard and an element of NPTIME.

Definition 2.0.2: L_1 is a Savitch CSL iff $L_1 \in$NDCSL; and $L_1 \in$DCSL implies DCSL = NDCSL. Savitch languages are named for W.J. Savitch, who in Savitch [70] presented a hardest nondeterministic logn tape recognizable language.

First, we prove that there is a simple p-complete language, which is of linear nondeterministic time complexity on some 2 tape Tm and of nlogn nondeterministic time complexity on some single tape Tm.

To prove these results we need two technical results:

Lemma 2.0.3: $(\forall k \geq 1)$ ($\exists$ a 2-tape Tm M_k) $(\forall w \in \Sigma^+)$

$[L(M_k) = \{x \mid x = w ¢ 1^{|w|^{2^k}}\$$ with $¢, \$ \notin \Sigma\}$.

Moreover, M_k operates within linear time.]

Lemma 2.0.4: $(\forall k \geq 1)$ ($\exists$ a single-tape Tm M_k) $(\forall w \in \Sigma^+)$

$[L(M_k) = \{x \mid x = w ¢ 1^{|w|^{2^k}}\$$ with $¢, \$ \notin \Sigma\}$.

Moreover, M_k operates within time $O(|x| \log |x|)$.]

The proofs of Lemmas 2.0.3 and 2.0.4 are standard.

Proof of 2.0.3: The lemma states that for any $k \geq 1$, there is a 2-tape linear time Tm M_k whose accepted language

$$L(M_k) = \{x \mid x = w ¢ 1^{|w|^{2^k}}\$ \text{ with } ¢, \$, \notin \Sigma\}.$$

Only the details will be sketched.

The proof is by induction on k. Let k=1. The first tape of M_1 contains the input string x. The second tape has 2 tracks; a counter of length $|w|$ is kept on each track. The $|w|^2$ 1's are divided into $|w|$ sets; each set contains $|w|$ 1's. The bottom counter records which set the machine is checking off on the first tape. The top records which 1 in a given set the machine is checking off on the first tape. The time used is $O(|x|)$.

Assume the lemma holds for k=i-1. The proof for k =i is similar to that for k=1 with $|w|$ replaced by $|w|^{2^{i-1}}$.

Proof of 2.0.4: The method of proof follows that of Lemma 2.0.3 with the following modifications:

1) the counters are on the same tape as the input,

2) they are kept in base 2 and hence have maximum size $O(\log |x|)$, and

3) they are moved one cell to the right each time a new symbol is checked. Again the time required is $O(\log |x|)$. Hence the total time required is $O(|x| \log |x|)$.

Lemma 2.0.5: 1) There are polynomial complete problems of nondeterministic time complexity $O(n \log n)$ on single tape Tms.

2) There are polynomial complete problems of linear nondeterministic time complexity on 2-tape Tms.

Proof: We only sketch the proof of 1. Both 1 and 2 follow from the fact that we are able to deterministically pad inputs within the allowed time bounds.

Using well-known results concerning encodings, we can encode arbitrary single-tape Tms as strings over a fixed finite alphabet. We can do this in such a way that the following hold:

A. The encoding process is in PTIME;

B. There is a single-tape nondeterministic Tm M, which given the code for the ith Tm M_i together with the code of an input x for M_i, simulates the action of M_i on x; and

C. The time required for M to simulate n moves of M_i on x is bounded by a polynomial in the sum of n and the lengths of the encodings of M_i and of x.

We modify M by adding a clock to shut M off if M_i computes for too long ($> |x|^2$) and by padding its inputs to cover the costs of C and the bookkeeping required for running the clock. The total time required is $0(n \log n)$.

Next, we find a simple Savitch CSL using the same techniques as above; mainly our language is a universal NDCSL.

<u>Lemma 2.0.6</u>: Let L = {M_i # code $(X_1,\ldots,X_n)$ # | M_i is the code of an LBA using the symbols {(,),',0,1,+,-}, $X_1 \ldots X_n \in L(M_i)$ and the lengths of the codes of $X_1,\ldots,X_n$ are all equal to the cardinality of the tape alphabet of M_i + 1}.[1] Then the L $\in$ NDCSL, and L $\in$ DCSL iff DCSL = NDCSL.

<u>Proof</u>: First, L $\in$ NDCSL. This follows since we can check in space | M_i # code $(X_1,\ldots,X_n)$ # | that M_i and $X = X_1\ldots X_n$ are properly encoded. Secondly, the space required to simulate M_i is at most K (the cardinality of the tape alphabet of M_i) times $|x|$. But $K \cdot n \leq |$ code $(X_1,\ldots,X_n)$ |.

Suppose L $\in$ DCSL. Let M_i be some LBA. Define $M_{\sigma(i)}$ as follows:

1) Given an input $X = X_1\ldots X_n$, $M_{\sigma(i)}$ writes out M_i# code $(X_1,\ldots,X_n)$#.
2) $M_{\sigma(i)}$ applies M_{i_o} as a subroutine to M_i # code $(X_1,\ldots,X_n)$#, where M_{i_o} is some deterministic LBA whose accepted language is L. Then $M_{\sigma(i)}$ requires deterministic space O(n),

1. Let the input alphabet to M be $\sigma_1,\ldots,\sigma_i$. Then the code of σ_1 is 10^K, where K is the cardinality of the

tape alphabet of M_i. Similarly, the code of σ_2 is 110^{K-1}, etc. Thus, the code of a symbol <u>varies</u> with M_i.

2.1 Upwards Reducibilities and Effective Translations

In this section we show that the relationship between PTIME and NPTIME is strongly reflected in the relationships between Ndtime (f(n)) and Dtime (f(n)) for all $f(n) \geq n$. We also show almost identical results concerning the relationship between NDCSL and DCSL.

Theorem 2.1.1: The following are equivalent:

1) PTIME = NPTIME;
2) all linear time nondeterministic 2-tape Tm languages are elements of PTIME;
3) all n log n time nondeterministic single tape Tm languages are elements of PTIME; and
4) there exists a positive integer k and a recursive function σ: $N \rightarrow N$ such that for all functions $T(n) \geq n$, if M_i is a nondeterministic T(n) time bounded Tm, then $M_{\sigma(i)}$ is an equivalent $[T(n)]^k$ deterministic time bounded Tm.

Proof: The equivalence of 1 and 2 and of 1 and 3 follow from lemmas 2.0.3 and 2.0.4 respectively.

Let L_o be some fixed language of single tape nondeterministic time complexity n^{Ko}. Then

$L_o = \{w ¢ 1^{|w| 2^{j_o}} \$^{|w|} \mid w \in L_o;$ ¢, \$ $\notin$ the tape alphabet of L_o, and $2^{j_o} \geq K_o$ } is of nondeterministic linear time complexity on some 2 tape Tm. L_o is of nondeterministic n log n time complexity on some 1 tape Tm.

Clearly 4 implies 1. As is easily seen and we shall prove in Theorem 2.2.3, the set $L = \{(R_1,R_2) \mid R_1$ and R_2 are $(\cup,\cdot)$ regular expressions over $\{0,1\}$ and $L(R_1) \neq L(R_2)\} \in$ NPTIME. We now show how, given a single tape Tm M with state set S, tape alphabet T, set of accepting states F, start state q_o and an input $x = x_1 \ldots x_m$ to M, to construct $(\cup,\cdot)$ regular expressions $\beta_x(n)$ and $\gamma_x(n)$ such that

1) $L(\beta_x) = (\Sigma \cup \lambda)^{4n^2+2}$ if and only if M does not accept x in time n, where $\Sigma = \{\#\} \cup T \cup (S \times T)$;
2) $|\beta_x(n)| \leq C_M \cdot n^3$, where C_M depends only upon M not X or n;
3) $L(\gamma_x(n)) = (\Sigma \cup \lambda)^{4n^2+2}$ if and only if all computations of M on X halt in less than or equal to n steps;
4) $|\gamma_x(n)| \leq C_M \cdot n^3$, where C_M depends only upon M not X or n;

and 5) the deterministic time required for the construction of $\beta_x(n)$ and $\gamma_x(n)$ given M, x, and n is bounded by a polynomial in n (we assume $n \geq m$).

We only sketch the constructions of $\beta_x(n)$ and $\gamma_x(n)$ here. A more detailed discussion of similar constructions appears in Chapter 4.

β_x is the union of the following:

1) strings in which one instantaneous description does not follow the preceeding one by 1 application of a move rule of M.

$$\bigcup_{\sigma_1,\sigma_2,\sigma_3,\in\Sigma} (\Sigma\cup\{\lambda\})^{2n^2-n-1}\sigma_1,\sigma_2,\sigma_3\ \Sigma^{2n-2}\ (\Sigma^3-f_M(\sigma_1,\sigma_2,\sigma_3))\ (\Sigma\cup\{\lambda\})^{2n^2-n-1}\cup\lambda;$$

2) strings of length $\geq 2n^2 + n + 2$ and of (There are n i.d.'s each of length 2n + (n+1) "#'s".)

length $\leq 4n^2 + 2$ --

$$\Sigma^{2n^2+n+2}\cdot(\Sigma\cup\{\lambda\})^{2n^2-n};$$

3) strings of length $< 2n^2 + n + 1$ -- $\lambda \cup \Sigma \cdot (\Sigma \cup \{\lambda\})^{2n^2 + n - 1}$;

4) strings that do not begin with $\# \cdot ฿^n \cdot (q_0, X_1) \cdot X_2 \cdot \ldots \cdot X_m \cdot ฿^{n-m} \cdot \#$ of length $\leq 4n^2 + 2$ --
$[(\Sigma - \{\#\}) \cdot (\Sigma \cup \lambda)^{2n+1} \cup \Sigma \cdot (\Sigma - \{฿\}) \cdot (\Sigma \cup \lambda)^{2n} \cup \ldots \cup \Sigma^{n+1} \cdot (\Sigma - \{(q_0, X_1)\}) \cdot (\Sigma \cup \lambda)^n \cup \ldots \cup \Sigma^{2n+1} \cdot (\Sigma - \{\#\})] \cdot (\Sigma \cup \lambda)^{4n^2 - 2n}$

5) strings of length $2n^2 + n + 1$ that do not end in $\#$ -- $\Sigma^{2n^2 + n} \cdot (\Sigma - \{\#\})$;

and 6) strings that do not accept --
$[\Sigma - (\bigcup_{q_f \in F} \{q_f\} \times T)]^{2n^2 + n + 1}$.

We assume $n \geq m$ and n is sufficiently large so that all exponents in the above strings are positive. $|\beta_x(n)| \leq 0(n^3)$.

Similarly, $\gamma_x(n)$ is the union of (1), (2), (3), (4) and (5) above, plus (6') all strings of length $2n^2 + n + 1$ which are in terminal configurations, that is those in which there is no next move which M can legally apply --
(6') $\Sigma^{2n^2 - n} [\alpha \cdot \Sigma^{2n} \cup \Sigma \cdot \alpha \cdot \Sigma^{2n-1} \cup \ldots \cup \Sigma^{2n-1} \cdot \alpha \cdot \Sigma]$, where α is the set of all

state-symbol pairs with no next move out of them. Again $|\gamma_x(n)| \le 0\ (n^3)$.

Let M_i be a nondeterministic $T(n)$ time bounded single tape Tm. Thus if x is an input to M_i of length n, all computations of M_i on x require time $\le T(n)$. Suppose PTIME = NPTIME, then Inequiv $(\cup,\cdot)$ and Equiv $(\cup,\cdot)$ $\in$ PTIME. $M_{\sigma(i)}$ executes the following algorithm:

For $j \leftarrow |x|$ step 1 Do

Begin

Write out $\beta_x(j)$;

If $L(\beta_x(j)) \neq (\Sigma\cup\lambda)^{4j^2+2}$ then

Begin Accept x; Halt; End;

Write out $\gamma_x(j)$,

If $L(\gamma_x(j)) = (\Sigma\cup\lambda)^{4j^2+2}$ then

Begin Reject x; Halt; End; End;

Since M_i is $T(n)$ time bounded, the loop is executed at most $(T(n) + 1)$ times. Each execution of the loop requires time $\le P_k(T^3(n))$, where P_k is the time required to determine if $L(\beta_x(j))$ and $L(\gamma_x(j)) = (\Sigma\cup\lambda)^{4j^2+2}$.

Theorem 2.1.2: (1) NDCSL = DCSL if and only if (2) there exists a recursive function $s: N \to N$ such that for all functions $L(n) \geq n$, if M_i is a non-deterministic $L(n)$ tape bounded Tm, then $M_{s(i)}$ is an equivalent $L(n)$ deterministic tape bounded Tm.

Proof: Clearly 2 implies 1. As we shall prove in Chapter 5, the set $L = \{(R_1,R_2) \mid R_1$ and R_2 are $(\cup,\cdot,*)$ regular expressions over $\{0,1\}$ and $L(R_1) \neq L(R_2)\} \in$ NDCSL. We now show how, given a single tape Tm M with state set S, tape alphabet T, set of accepting states F, start state q_0 and input $X = X_1 \ldots X_m$ to M, to construct $(\cup,\cdot,*)$ regular expressions $\beta_x(n)$ and $\gamma_x(n)$ such that

1) $L(\beta_x(n)) = (0\cup 1)^*$ if and only if M does not accept x in tape n;
2) $|\beta_x(n)| \leq C_M \cdot n$, where C_M depends only upon M and not x or n;
3) $L(\gamma_x(n)) = (0\cup 1)^*$ if and only if all computations of M on x require less than or equal to n tape cells;

4) $|\gamma_x(n)| \leq C_M \cdot n$, where C_M depends only upon M not x or n; and

5) the deterministic space required for the construction of $\beta_x(n)$ and $\gamma_x(n)$ given M, x, and n is linear in n; we assume $n \geq m$.

Again we only sketch the constructions of $\beta_x(n)$ and $\gamma_x(n)$. A more detailed discussion of similar constructions appears in Chapter 4.

Let $\int = T \cup (S \times T) \cup \{\#\}$. $\beta_x(n)$ is the union of the following:

(1) all strings in which one instantaneous description doesnot follow the proceding one by 1 application of a move rule of M-

$\cup_{a,b,c \in \int} [\int^* abc \int^{2n-2} (\int^3 - f_m(a,b,c)) \int^*]$;

all strings that donot begin with $\# b\!\!/^n (q_0, x_1) x_2 \ldots x_m b\!\!/^{n-m} \#$-

$[(\int - \#) \cup \# [(\int - b\!\!/) \cup b\!\!/ [\ldots b\!\!/ [(\int - (q_0, x_1) \cup \ldots b\!\!/ [(\int - \#)] \ldots] \ldots]]]$;

(3) all strings that do not end in #-

$\int^* (\int - \#)$; and

(4) all strings without an accepting state-

$[\int - (\cup_{q \in F} \{ q \} \times T)]^*$.

Similarly, $\gamma_x(n)$ is defined as the union of

(1), (2), and (3) above plus

(4') all strings in which the last instantaneous description does not require more space-

$\int^{*}\#[\ \alpha\int^{2n-1} \cup \int^{2n-1}\beta\ ]\#$, where α is the set of all state-symbol pairs with no next move which moves the tape head left and β is the set of all state-symbol pairs with no next move which moves the tape head right. Again $|\gamma_x(n)| \leq 0(n)$.

Let M_i be a nondeterministic $L(n)$ tape bounded single tape Tm, where $L(n) \geq n$. Thus if $|x| = n$, then all computations of M_i on x require $\leq L(n)$ tape cells. Suppose NDCSL = DCSL, then both Inequiv($\cup, \cdot, *$) and Equiv($\cup, \cdot, *$) $\in$ DCSL. $M_{s(i)}$ executes the following algorithm:

For j ← |x| step 1 Do

Begin

Write out $\beta_x(j)$;

If $L(\beta_x(j)) \neq \{0,1\}^*$ then begin Accept(x);

Halt; end;

Write out $\gamma_x(j)$;

If $L(\gamma_x(n)) = \{0,1\}^*$ then begin Reject(x);

Halt; end;

end;

Since M_i is $L(n)$ tape bounded, the loop is executed at most $(L(n) + 1)$ times. Each execution of the loop requires tape $\leq 0(L(n))$.

Finally for proofs of Theorems 2.1.1 and 2.1.2 not using regular expressions the reader should see Hartmanis and Hunt[73]. Several of the results of this section have been discovered independently by R. Fagin(the equivalence of 1 and a weaker version of 4 in 2.1.1) and R.V. Book[72] and [73] (the equivalence of 1 and 2 of 2.1.1 and 2.1.2.)

2.2 P-complete Languages and Operations

In this section we show that the equivalence and emptiness problemsfor several types of restricted regular expressions are p-complete. We also investigate nondeterministic polynomial time bounded operations that generate NPTIME when applied to elements of PTIME.

Theorem 2.2.1: The following are p-complete:

(1) Inequiv($\cup,\cdot$);

(2) { (M, M') | M and M' are nondeterministic finite automata (abbreviated nfa) which accept finite sets and $L(M) \neq L(M')$ };

(3) [Hopcroft and Hunt] Inequiv($\cup,\cdot,\cap$); and

(4) Notempty($\cup,\cdot,\cap$).

Proof: Cook[71] has shown that D_3-Tautology is p-complete. (D_3-Tautology is the set of nontautological Boolean forms in Disjunctive normal form with at most 3 literals per clause.) Let $f = \bigvee_{i=1}^{m} c_i$ be a D_3-Boolean form. Then each c_i is the Boolean product of at most three literals. For each c_i let $\beta_i = \beta_{i1}\beta_{i2}\cdots\beta_{in}$. (The number of variables appearing in f is n.)

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

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. Clearly the construction of β given f is deterministic polynomial in $|f|$. Thus D_3-Tautology $\leq_{ptime}$ Inequiv$(\cup,\cdot)$. (Usually D_3-Tautology is the set of D_3- tautologies. But using this definition it is not known whether this set is p-complete. In fact it is p-complete iff NPTIME is closed under complementation.)

Next we show that Inequiv$(\cup,\cdot) \in$ 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 finite alphabet $\{ \phi, \lambda, 0, 1 , \cup, \cdot, (,)\}$) . Hence given two $(\cup,\cdot)$ regular expressions α and β, to verify that $L(\alpha) \neq L(\beta)$, we need only guess a string x of length $\leq$ max($|\alpha|$, $|\beta|$) such that $x \in L(\alpha) \cap \overline{L(\beta)}$ or $x \in L(\alpha) \cap \overline{L(\beta)}$.

(2) Given a $(\cup, \cdot)$ regular expression R, we can construct an nfa M in deterministic polynomial time in $|R|$ such that $L(M) = L(R)$. Thus the fact that the set $\{ (M, M') \mid M$ and M' are nfa which accept finite sets and $L(M) \neq L(M') \}$ is p-hard follows from 1. Since M an nfa which accepts a finite set and $x \in L(M)$ implies that $|x| \leq$ the number of states of M, the set $\{ (M, M') \mid M$ and M' are nfa which accept finite sets and $L(M) \neq L(M') \}$ is easily seen to be an element of NPTIME.

(3) From 1 Inequiv$(\cup, \cdot, \cap)$ is p-hard. We show that it is an element of NPTIME. As in the proof of 1, α, a $(\cup, \cdot, \cap)$ regular expression implies for all $x \in L(\alpha)$, $|x| \leq |\alpha|$. Thus the proof of 1 would also prove 3 provided given x with $|x| \leq \max(|\alpha|, |\beta|)$ we can in deterministic polynomial time in $|x|$ decide if $x \in L(\alpha) \cap \overline{L(\beta)}$ or $x \in \overline{L(\alpha)} \cap L(\beta)$.

Let x be a string that differentiates between α and β. $|x| = n \leq \max(|\alpha|, |\beta|)$. W.l.g. we assume $x \in L(\alpha) \cap \overline{L(\beta)}$. There are at most $0(\frac{n(n+1)}{2})$ proper substrings of x. There are at most $(|\alpha| + |\beta|)$ regular subexpressions 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 regular subexpressions of $\alpha \leq |\alpha'| + |\alpha''| + 1$.)

For each regular subexpression of α and for each of β, we construct an array that indicates which proper substrings of x are in the regular set denoted by the subexpression and which are not. The i,jth element of the array corresponds to that proper substring of x beginning with the ith and ending with the j-1st character of x, i.e., $x_i x_{i+1} \ldots x_{j-1}$. It is 1 if $x_i x_{i+1} \ldots x_{j-1}$ is an element of the set denoted by the subexpression and 0 otherwise. $A_{ij} = 0$ if $j < i + 1$.

For each subexpression R we must know if $\lambda \in L(R)$. Clearly, given R and S if we know if $\lambda \in L(R)$ or not and if we know if $\lambda \in L(S)$ or not, then we know if $\lambda \in L(R \cup S)$, $L(R \cap S)$, or $L(R \cdot S)$.

If $|R| = 1$ then the number of nonzero elements of the corresponding array $\leq n$. (It is easy in in deterministic polynomial time in n to compute this matrix.) If $|R| \geq 2$ then $R = R_1 \cup R_2$, $R_1 \cap R_2$, or $R_1 \cdot R_2$. Let A, A', and A'' be the corresponding matrices.

Case 1. If $R = R_1 \cup R_2$ then $A = A' \vee A''$ and $\lambda \in$

$L(R)$ iff $\lambda \in L(R_1)$ or $\lambda \in L(R_2)$, i.e., $A_{ij} = A'_{ij} \vee A''_{ij}$.

Case 2. If $R = R_1 \cap R_2$ then $A_{ij} = A'_{ij} \wedge A''_{ij}$ and $\lambda \in L(R)$ iff $\lambda \in L(R_2)$ and $\lambda \in L(R_1)$, i.e., $A_{ij} = A'_{ij} \wedge A''_{ij}$.

Case 3. If $R = R_1 \cdot R_2$ then λ is an element of both $L(R_1)$ and $L(R_2)$.

$$A_{ij} = \begin{cases} \bigvee_{i<k<j} A'_{ik} \times A''_{kj} & \text{if } \lambda \notin L(R_1) \text{ and } \lambda \notin L(R_2), \\ A''_{ij} \vee \bigvee_{i<k<j} A'_{ik} \times A''_{kj} & \text{if } \lambda \in L(R_1) \text{ and } \lambda \notin L(R_2), \\ A'_{ij} \vee \bigvee_{i<k<j} A'_{ik} \times A''_{kj} & \text{if } \lambda \notin L(R_1) \text{ and } \lambda \in L(R_2), \\ A'_{ij} \vee A''_{ij} \vee \bigvee_{i<k<j} A'_{ik} \times A''_{kj} & \text{if } \lambda \in L(R_1) \text{ and } \lambda \in L(R_2). \end{cases}$$

The operation $\vee$ is Boolean or. "x" denotes and.

(4) D_3-tautology is the same as C_3-satisfiability, where C_3-satisfiability is the set of satisfiable Boolean forms in conjunctive normal form with at most three literals per clause. Let the variables appearing in f be $\{x_1, \ldots, x_n\}$. Then f is a D_3-Boolean form iff (by definition) $f = \bigvee_{i=1}^{m} c_i$, where each term is the product(and) of at most three literals.

W.l.g. we assume no pair of literals x_j, $\overline{x_j}$ appears in any term of f. Then f is a tautology iff $\sim f = \bigwedge_{i=1}^{m} (\sim y_{i1} \vee \sim y_{i2} \vee \sim y_{i3})$ is not satisfiable. (Here $c_i = y_{i1} \wedge y_{i2} \wedge y_{i3}$.) Let $\beta = \beta_1 \cap \beta_2 \cap \ldots \cap \beta_m$, where $\beta_i = \beta_{i1} \cup \beta_{i2} \cup \beta_{i3}$ and $\beta_{ij} = \beta_{ij}^{1} \cdot \beta_{ij}^{2} \cdot \beta_{ij}^{3} \cdot \ldots \cdot \beta_{ij}^{n}$ and

$$\beta_{ij}^{k} = \begin{cases} (0 \cup 1) & \text{if } y_{ij} \neq x_k \text{ and } y_{ij} \neq \overline{x_k}, \\ (1) & \text{if } y_{ij} = \overline{x_k}, \text{ and} \\ (0) & \text{if } y_{ij} = x_k. \end{cases}$$

Let $x = t_1 \ldots t_n \in \{0,1\}^n$. Then $x \in L(\beta)$ iff $\sim f(t_1,\ldots,t_n) = \text{true}$, i.e., $\sim f$ is satisfiable. Finally, Notempty$(\cup,\cdot,\cap) \in$ NPTIME since we may guess a string x and verify that $x \in L(\beta)$ in deterministic time a polynomial in the length of x as shown in the proof of 3.

Informally a p-complete problem is a generator of NPTIME by deterministic polynomial time bounded operations. We next investigate the related questions:

(1) Is NPTIME generated by elements of PTIME by non-deterministic polynomial time bounded operations? and

(2) Are there complete operations?

We answer both questions affirmatively.

<u>Theorem(Book and Greibach[70])</u>: A language L is recognized by a linear time nondeterministic multi-tape Tm

if and only if L is the epsilon-free homomorphic image of the intersection of three context-free languages.

Since we have already presented a p-complete problem of linear nondeterministic time complexity, PTIME is closed under intersection, and all cfls are elements of PTIME, we have the following immediate corollaries-

Corollary 2.2.2: There are three cfls L_0, L_1, and L_2 and an epsilon-free homomorphism h such that h($L_0 \cap L_1 \cap L_2$) is p-complete; and

Corollary 2.2.3: PTIME = NPTIME iff PTIME is closed under epsilon-free homomorphism.

(Book[72] has independently proved 2.2.3.) Thus, Corollaries 2.2.2 and 2.2.3 answer the first question above affirmatively. Finally we present a fixed epsilon-free homomorphism **h and** a fixed language L $\in$ PTIME such that h(L) is p-complete.

Theorem 2.2.4: Let L = { fcx | f is a D_3-Boolean form expressed as a string in some finite alphabet S. $0,1 \notin S$. $x = x_1 \cdot \ldots \cdot x_n \in \{0,1\}^n$ for some n > 0. f is a function of n variables $t_1, \ldots, t_n$ and $f(x_1, \ldots, x_n)$ is false }. Let h be defined as follows:

(1) h is an epsilon-free homomorphism from [S $\cup$

$\{ c,0,1 \}$ into $[S \cup \{ c,0,1 \}]^*$,

(2) for all $s \in S \cup \{c\}$, $h(s) = s$,

(3) $h(0) = 0$, and

(4) $h(1) = 0$.

Then $L \in$ PTIME and $h(L)$ is p-complete.

Proof: Let f be a D_3-Boolean form of n variables. Then $fc0^n \in h(L)$ if and only if f is not a tautology.

(a) $fc0^n \in h(L)$ implies that there exists an $x = x_1 \cdot \ldots \cdot x_n \in \{0,1\}^n$ such that $fcx \in L$. But this implies that $f(x_1,\ldots,x_n)$ is false. Hence f is not a tautology.

(b) f is not a tautology implies that there exists an assignment of values $(t_1,\ldots,t_n)$ to the n variables of f such that $f(t_1,\ldots,t_n)$ is false. Then $fct_1\ldots t_n \in L$ and $fc0^n \in h(L)$.

Thus h is a complete operation and answers the second question above affirmatively.

Chapter 3 GENERAL UNDECIDABILITY THEOREMS

We state and prove new extensions of the undecidability theorems appearing in Greibach[68]. These extensions lead, naturally, to analogous theorems which give nontrivial lower bounds on the minimal deterministic time needed to decide many properties of the regular sets.

We state without proof Greibach's two undecidability theorems. Next we state and prove our extensions of these theorems. Finally, several examples are given where our theorems apply and those of Greibach do not.

Definition 3.0: $L/x = \{ y \mid yx \in L \}$. $x\backslash L = \{ y \mid xy \in L \}$.

We call L/x ($x\backslash L$) the right (left) quotient of L with the set $\{x\}$.

Definition 3.1: An effective family of languages is a quintuple (Σ,F,f_1,f_2,M) where

(1) Σ is a countably infinite vocabulary and M is a total recursive function such that for any finite subset Σ_1 of Σ, $M(\Sigma_1)$ is an element of $\Sigma-\Sigma_1$;

(2) F is a family of languages over Σ;

(3) f_1 is a function from N onto F such that the mapping g defined on $N\times\Sigma^*$ by

$$g(n,w)=\begin{cases}1 \text{ if } w \in f_1(n)\\ \text{undefined, otherwise}\end{cases}$$

is partial recursive; and

(4) f_2 is a total recursive function from N into the finite subsets of Σ such that $f_1(n)\subseteq[f_2(n)]^*$.

Definition 3.2: (Σ,F,f_1,f_2,M) is effectively closed under a binary operation δ on F, if there exists a total recursive function $\underline{\delta}:N\times N\to N$ such that $f_1(\underline{\delta}(n,m))=\delta(f_1(n),f_1(m))$. (Σ,F,f_1,f_2,M) is effectively closed under a binary operation ρ on F and the regular sets over Σ(denoted by R_Σ), if there exists a total recursive function $\underline{\rho}$: $N\times R_\Sigma \to N$ such that $f_1(\underline{\rho}(n,R))=\rho(f_1(n),R)$.

Several words of explanation concerning the intuitive meanings of 3.1 and 3.2 are in order:

(1) M is used to find a marker not in tape alphabet Σ_1:

(2) f_1 enumerates the languages L_i in F in such a way that each L_i is recursively enumerable;

(3) $f_2(i)$ is the terminal alphabet of the language L_i.

Similarly 3.2 means that given the indices i and j, we can effectively find an index k such that $L_k = \delta(L_i,L_j)$.

<u>Theorem(Greibach[68])</u>: Let $\Phi = (\Sigma,F,f_1,f_2,M)$ be an effective family of languages, which is effectively closed under union and under concatenation by regular sets. Let "$f_1(n)=[f_2(n)]^*$" be undecidable for $L_n=f_1(n)$ in F. If P is any property defined on F such that

(a) there is an epsilon-free language L_2 in F which does not have property P;

(b) all sets of the form $f_2(n)^* c_{nm} f_2(m)^*$ have property P, where $c_{nm}=M(f_2(n)\cup f_2(m))$; and

(c) if L has property P, R is regular and y is a string, then $L\cap R$ and $y\backslash L=\{ w \mid yw\in L\}$ have property P,

then P is undecidable for Φ.

Theorem(Greibach[68]): Let $\Phi=(\Sigma, F, f_1, f_2, M)$ be an effective family of languages, which is effectively closed under concatenation. Let "$f_1(n)= \phi$" be undecidable for Φ. If P is any property which

(a) is false for some epsilon-free L_2 in F;

(b) is true for ϕ; and

(c) is preserved by inverse gsm and intersection with regular sets,

then P is undecidable for Φ.

Theorem3.3: Let $\Phi=(\Sigma, F, f_1, f_2, M)$ be an effective family of languages, which is effectively closed under union and under concatenation by regular sets. Let "$f_1(n)=[f_2(n)]^*$" be undecidable on Φ. If P is any property on F such that

(a) $\exists L_0 \in F - \cup_{x\in\Sigma^*} \{ x\backslash L \mid P(L)= \text{true} \}$ [or $\exists L_0 \in F - \cup_{x\in\Sigma^*} \{ L/x \mid P(L)=\text{true} \}$] and

(b) P is true for all regular sets of the form $f_2(n)^* c_{nm} f_2(m)^*$, where $c_{nm} = M(f_2(n) \cup f_2(m))$.

Then P is undecidable on Φ.

Proof: Let $P_L = \cup_{x\in\Sigma^*} \{ x\backslash L \mid P(L)=\text{true} \}$ and let $P_R= \cup_{x\in\Sigma^*} \{ L/x \mid P(L)=\text{true} \}$. Let $L_i=f_1(i)$ and let $\Sigma_i=f_2(i)$.

Assume $F-P_L \neq \emptyset$.

Given any i we can effectively find an index j such that $L_j = L_i c_{i\,k} \Sigma_k^* \cup \Sigma_i^* c_{i\,k} L_k$, where L_k is an element of $F-P_L$. Then $P(L_j)$ is true iff $L_i = \Sigma_i^*$.

<u>Case1</u>. $L_i = \Sigma_i^*$. Then $L_j = \Sigma_i^* c_{i\,k} \Sigma_k^*$.

<u>Case2</u>. $L_i \neq \Sigma_i^*$. Then $\exists x \in \Sigma_i^* - L_i$. Therefore, $x c_{i\,k} \backslash L_j = L_k$. But $P(L_j)$ true implies $L_k \in P_L$, which is a contradiction. Hence $P(L_j)$ is false.

Thus, $P(L_j)$ is true iff $L_i = \Sigma_i^*$. The case when $P-P_R \neq \emptyset$ is left to the reader.

<u>Theorem 3.4</u>: Let $\Phi = (\Sigma, F, f_1, f_2, M)$ be an effective family of languages that is effectively closed under concatenation. Let "$=\emptyset$" be undecidable in Φ. Let P be any property that is

(a) true for $\emptyset$ and

(b) such that $F-P_L$ [or $F-P_R$] $\neq \emptyset$,

then P is undecidable for Φ.

<u>Proof</u>: Let $L_k \in F-P_L$. Let $L_j = L_i c_{i\,k} L_k$. Then $L_i = \emptyset$ implies that $L_j = \emptyset$ and that $P(L_j)$ is true. $L_i \neq \emptyset$ implies that there exists a string $x \in L_i$. But this implies that $x c_{i\,k} \backslash L_j = L_k$. Thus $P(L_j)$ is false. Thus $P(L_j)$ is true iff $L_i = \emptyset$.

<u>Definition 3.5</u>: An effective family of languages over $\{0,1\}$ is an ordered pair (F,f), where

(a) F is a family of languages over $\{0,1\}$ and

(b) f is a function from N onto F such that the mapping g defined on $N \times \{0,1\}^*$ by

$$g(n,w) = \begin{cases} 1 & \text{if } w \in f(n) \\ \text{undefined, otherwise} & \end{cases}$$

is partial recursive.

<u>Theorem 3.6</u>: Let $\Phi = (F,f)$ be an effective family of languages over $\{0,1\}$, which is effectively closed under concatenation with regular sets, union, application of the epsilon-free homomorphism h defined by $h(0) = 00$ and $h(1) = 01$, and such that $(00+01)^*[\lambda+0+1+11(0+1)^*] \in F$. Let P be any predicate on F such that

(a) $P(\ (0+1)^*)$ is true and

(b) $F-P_L$ [or $F-P_R$] $\neq \emptyset$.

Then P is undecidable on Φ.

<u>Proof</u>: Let $F-P_L \neq \emptyset$. Let $L_k \in F-P_L$. Then given an index i we can effectively find an index j such that $L_j = h(L_i)10(0+1)^* + (00+01)^*10L_k + \sim[(00+01)^*10(0+1)^*]$. $L_j \in F$, since $\sim[(00+01)^*10(0+1)^*] = (00+01)^*[\lambda+0+1+11(0+1)^*]$ and F is closed under h, union, and concatenation with regular sets.

$L_i = (0+1)^*$ implies that $L_j = (0+1)^*$ and $P(L_j)$ is true. $L_i \neq (0+1)^*$ implies that there exists $x \in (0+1)^* - L_i$. This implies that $h(x) \in (00+01)^* - h(L_i)$ and $h(x)10 \backslash L_j = L_k$. Hence $P(L_j)$ is false. Thus $P(L_j)$ is true iff $L_i = (0+1)^*$. [Note that in the statements and proofs of this theorem and of the results in Chapter 5, + means union and $\sim$ means complementation.]

<u>Corollary 3.7</u>: Let Φ be an effective family of languages that is effectively closed under concatenation. Let "$=\emptyset$" be undecidable in Φ. If F contains both finite and infinite languages, then finiteness is undecidable for Φ.

<u>Proof</u>: Finiteness is preserved by quotient.

<u>Corollary 3.8</u>: Let R be any subset of the deterministic context-free languages over $\{0,1\}$ such that $(0+1)^* \in R$. For the cfgs over $\{0,1\}$, the predicate "$L(G_i) \in R$" is undecidable.

<u>Proof</u>: Immediate from 3.6 since the deterministic cfls are closed under quotient with regular sets on the right.

<u>Corollary 3.9</u>: The predicate "$L=[L]^{rev}$" satisfies the conditions of Theorems 3.6.

<u>Proof</u>: See the proof of Theorem 5.2.1.

Clearly Corollaries 3.8 and 3.9 do not satisfy the conditions of the undecidability theorems mentioned above due to Greibach. Thus, the extensions of Greibach's theorems in this chapter include the following:

(1) P need not be preserved by intersection with regular sets;

(2) P need not be closed under quotient with single string; and

(3) If L is effectively closed under the epsilon-free homomorphism h of 3.6 and $(00+01)^*[\lambda+0+1+11(0+1)^*]$ is an element of F, then P need only be true for $(0+1)^*$.

The power and importance of these extensions (especially 2) will become apparent in Chapter 5.

CHAPTER 4 THE EQUIVALENCE AND EMPTINESS PROBLEMS FOR REGULAR SETS

4.0 Introduction

Using the techniques of Meyer and Stockmeyer [12], we investigate the complexity of the equivalence and emptiness problems for various kinds of regular set descriptions (e.g., $(\cup,\cdot,*)$, $(\cup,\cdot,*,\cap)$, and $(\cup,\cdot,*,\neg)$ regular expressions.)

Definition 4.0.1: Let L_o be a language. By PTAPE $\underset{\text{PTIME}}{\leq}$ L_o we mean that every language in PTAPE is p-reducible to L_o, that is $L \in$ PTAPE implies $L \underset{\text{PTIME}}{\leq} L_o$.

Lemma 4.0.2: Let L be a language over $\{0,1\}$. If $\exists r > 0$ such that DTAPE $(n^r) \underset{\text{PTIME}}{\leq}$ L, then PTAPE $\underset{\text{PTIME}}{\leq}$ L. In particular if DCSL $\underset{\text{PTIME}}{\leq}$ L, then PTAPE $\underset{\text{PTIME}}{\leq}$ L.

Proof: Left for the reader.

Finally, in each of the remaining sections of this chapter we consider regular set descriptors over arbitrarily large alphabets Σ. It is left for the reader to verify (see Lemma 5.1.1) that in each case, give a descriptor R, we can effectively find a descriptor S over $\{0,1\}$ such that $L(R) = \emptyset$ iff $L(S) = \emptyset$ or $L(R) = \Sigma^*$ iff $L(S) = \{0,1\}^*$. Furthermore, this encoding process is linear in space and deterministic polynomial in time.

4.1 Equiv $(\cup,\cdot,*)$ and Empty $(\cup,\cdot,*,\cap)$

We show that PTAPE $\underset{\text{PTIME}}{\leq}$ Inequiv $(\cup,\cdot,*)$ and that PTAPE $\underset{\text{PTIME}}{\leq}$ Notempty $(\cup,\cdot,*,\cap)$. We do this by efficiently embedding arbitrary deterministic lba computations into $(\cup,\cdot,*)$ and $(\cup,\cdot,*,\cap)$ regular expressions.

Definition 4.1.1: Let M be a deterministic lba with tape symbols T and states S. Assume $0,1,\not b \in T$, where $\not b$ denotes the blank tape square. An instantaneous description (i.d.) of M is a word in $T^* \cdot (S \times T) \cdot T^*$.

Definition 4.1.2: Given any i.d. $x = y \cdot (s \times t) \cdot z$ for $y,z, \in T^*$, the next i.d., $\text{Next}_M(x)$ 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 $\text{Next}_M(x)$ is

1) $y\cdot(s' \times t')\cdot z$ if M does not shift its head,
2) $y\cdot t'\cdot(s' \times u)\cdot w$ if M shifts its head right and $z = u\cdot w$ for $u \in T$ and $w \in T^*$,
3) $w\cdot(s' \times u)\cdot t'\cdot z$ if M' shifts its head left and $y = w\cdot u$ for $u \in T$ and $w \in T^*$,

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

<u>Definition 4.1.3</u>: $\text{Next}_M(x,0) = x$ if x is an i.d. and is undefined otherwise.

$\text{Next}_M(x,n+1) = \text{Next}_M(\text{Next}_M(x,n))$.

<u>Definition 4.1.4</u>: Let $\#$ be a symbol not in $T \cup (s \times T)$.

The <u>computation</u> $C_M(x)$ of M from x is the following word in $(\{\#\} \cup T \cup (S \times T))^*$:

$C_M(x) = \# \cdot \text{Next}_M(x,0) \cdot \# \cdot \text{Next}_M(x,1) \cdot \# \cdot \ldots \cdot$
$\# \cdot \text{Next}_M(x,n) \cdot \#$.

Here, n is the least positive integer such that $(q_f \times t)$ occurs in $\text{Next}_M(x,n)$ 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 $= \{\#\} \cup T \cup (S \times T)$. 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 $\text{Next}_M(y)$ is determined uniquely by the $i\text{-}1^{\underline{st}}$, $i^{\underline{th}}$ and $i+1^{\underline{st}}$ symbols of y.

Theorem 4.1.5:

(1) (Meyer and Stockmeyer [72])

Inequiv $(\cup, \cdot, *) \underset{\text{PTIME}}{\geq}$ PTAPE.

(2) (Hunt[73b]) Notempty $(\cup, \cdot, *, \cap) \underset{\text{PTIME}}{\geq}$ PTAPE.

Proof: From 4.0.2 we need only simulate all deterministic linear bounded automata. Let M be a deterministic linear bounded automata with states S, tape alphabet T, designated halting state $q_f \epsilon S$, and designated start state $q_o \epsilon S$. We assume q_f is final. Let $\Sigma = \{\#\} \cup T \cup (S \times T)$. Let $x = x_1 \cdot \ldots \cdot x_n$ be a given input to M.

(1) We construct a $(\cup, \cdot, *)$ regular expression β_x such that $L(\beta_x) = \Sigma^*$ iff $X \notin L(M)$. $L(\beta_x)$ is the set of invalid computations of M on x.

$\therefore L(\beta_x) = \Sigma^* - C_M(x)$. β_x is characterized as follows: β_x is the union of

a) all word that do not begin with

$$\# \cdot (q_o, x_1) \cdot x_2 \cdot \ldots \cdot x_n \cdot \#;$$

b) all words not of the form

$$\# \cdot [(\Sigma - \#)^n \cdot \#]^+;$$

c) all words that do not contain a halt state;

and

d) all words that violate the next move requirement of M.

$$\therefore\ \beta_x = \lceil(\Sigma-\#) \cup \# \,[(\Sigma-(q_o,x_1)) \quad \cup(q_o,x_1)$$
$$[(\Sigma-x_2) \cup X_2 \,[\ \ldots\ (\Sigma-\#)]\ldots]]]\cdot \Sigma^* \qquad \text{(a)}$$

$$\cup$$

$$\Sigma^* \cdot(\Sigma-\#) \cup (\Sigma-\#)\cdot\Sigma^* \cup \Sigma^* \# (\Sigma\cup\lambda)^{n-1}\# \Sigma^*$$
$$\cup \Sigma^* \# (\Sigma-\#)^{n+1} \Sigma^* \qquad \text{(b)}$$

$$\cup$$

$$(\Sigma-[\bigcup_{t\in T}(\{q_f\} \times T)])^* \qquad \text{(c)}$$

$$\bigcup_{\sigma_1,\sigma_2,\sigma_3 \in \Sigma} \Sigma^* \sigma_1\sigma_2\sigma_3\ \Sigma^{n-1}\ f_M(\sigma_1,\sigma_2,\sigma_3)\ \Sigma^*. \qquad \text{(d)}$$

Clearly if a string $\gamma\in\{0,1\}^*$ is not a valid computation of M on x then it satisfies (a), (b), (c), or (d). Similarly, if γ satisfies (a), (b), (c), or (d) then γ is not a valid computation of M on x. Finally, note that the construction of β_x given x requires space linear in $|x|$ and time deterministic polynomial in $|x|$.

(2) We construct a $(\cup,\cdot,*,\cap)$ regular expression β_x such that $L(\beta_x) = C_M(x)$. Let M be as in (1). β_x is characterized as follows: β_x is the intersection of

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

f) all words that contain exactly one q_f,

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

h) and i) all words that do not violate the next-move requirement of M.

$$\beta_x \underline{\underline{\text{def}}}.\ \# \cdot (q_0, x_1) \cdot x_2 \cdot \ldots \cdot x_n \cdot \# \cdot \Sigma^* \qquad \text{(e)}$$

$$(\Sigma - [\bigcup_{t \in T} (q_f, t)])^* \cdot [\bigcup_{t \in T} (q_f, t)\rceil \cdot \qquad \text{(f)}$$

$$(\Sigma - \lceil \bigcup_{t \in T} (q_f, t)\rceil)^* \qquad \text{(g)}$$

$$\cap$$

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

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

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

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

Clearly from (e), (f), and (g) a string $\alpha \epsilon L(\beta_x)$ only if $\alpha \epsilon \# [(\Sigma - \#)^n \#]^k$ 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 (h) 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 (i) 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

h h h

$[\# \; id_1][\# \; id_2 \;][\# \; id_3][\# \; id_4 \;][\# \; id_5][\# \; id_6] \cdots$

i i i

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

<u>Proposition 4.1.6</u>: Inequiv $(\cup, \cdot, *) \in$ DCSL $\Rightarrow$ NDSCL = DCSL.

<u>Proof</u>: Let M be as in 4.1.5 except that M may now be nondeterministic. We would like to simulate the proof of 4.1.5. However, we must modify the definitions of f_M, $Next_M(x)$, and $C_M(x)$.

Let $\Sigma = \{\#\} \cup T \cup (S \times T)$.

1) f_M: $\Sigma^3 \to 2^{\Sigma^3}$ defined as follows:

$$\sigma_1\sigma_2\sigma_3 \in \Sigma^3 \quad f_M(\sigma_1\sigma_2\sigma_3) = \begin{cases} \Sigma\sigma_2\Sigma \text{ if } \sigma_1\sigma_2\sigma_3 \notin SxT \text{ or } \sigma_2 = \#; \\ A(\sigma_1\sigma_2\sigma_3) \text{ if } \sigma_1,\sigma_3 \in T\cup\{\#\} \text{ and } \sigma_2 \in SxT; \\ \Sigma^3, \text{ otherwise.} \end{cases}$$

$A(\sigma_1\sigma_2\sigma_3)$ is defined as follows. Let $\beta = \beta_1\beta_2\beta_3 \in \Sigma^3$. Then $\beta \in A(\sigma_1\sigma_2\sigma_3)$ iff

(a1) $\beta_1 = \sigma_1$, $\beta_2 = s' \times t'$, $\beta_3 = \sigma_3$ and (s,t,s',t',o) is a move of M where $\sigma_2 = s \times t$, or

(a2) $\beta_1 = (s' \times \sigma_1)$, $\beta_2 = t'$, $\beta_3 = \sigma_3$ and $(s,t,s',t',-1)$ is a move of M, where $\sigma_1 \neq \#$ and $\sigma_2 = (s \times t)$, or

(a3) $\beta_1 = \sigma_1$, $\beta_2 = t'$, $\beta_3 = (s' \times \sigma_3)$ and $(s,t,s',t',+1)$ is a move of M, where $\sigma_3 \neq \#$

and $\sigma_2 = (s\ x\ t)$, or

(a3) $\beta_1 = \sigma_1$, $\beta_2 = t'$, $\beta_3 = (s'\ x\ \sigma_3)$ and $(s,t,s',t',+1)$ is a move of M, where $\sigma_3 \neq \#$ and $\sigma_2 = (s\ x\ t)$.[1]

f_M guarantees that consecutive sets of three symbols correspond to the <u>same</u> allowed move rule. (f_M as defined in the proof of 4.1.5 does <u>not</u> do this.)

2) $Next_M(x)$ = if x is an i.d. then $\{y | y$ is an i.d. and y follows from x by exactly 1 move of M$\}$ else undefined.

3) $C_M(x)$ = if x is an i.d. then $\{\#x_o\#x_1\# \ldots \#x_t\# |$ where $x_o = x$, each $x_i \epsilon$ Next (X_{i-1}), and q_f x t is a substring of x_t, where q_f is an accepting state and $t \epsilon T\}$ else undefined.

Then β_x is the union of:

β_1, the set of strings that do <u>not</u> begin with

$\#(q_o,x_1)\ x_2\ \ldots\ x_n\#$;

β_2, the set of strings that do not contain a symbol (q_f,t), where $q_f \epsilon F$;

1. $(s,t,s',t',-1)$ is interpreted as follows: M when in state s and scanning symbol t can change t to t', changes state from s to s', and move its tape head 1 tape unit to the left. For a more detailed discussion of Turing machines see Hopcroft and Ullman [69].

β_3, the set of strings not of the form

$\# \cdot [(\Sigma-\#)^n \#]^+$; and

β_4, the set of strings that make a mistake between one i.d. and the next.

But $\beta_1 = [(\Sigma-\#) \cup \# \cdot [(\Sigma-(q_o,x_1)) \cup (q_o,x_1) \cdot [(\Sigma-x_2) \cup x_2 [\ \ldots \cup x_n [(\Sigma-\#)] \ldots]]]] \cdot \Sigma^*$.

To understand β_1 the reader should notice the similarity of the above to Horner's rule for evaluating polynomials, i.e.,

$a_0 + a_1 x + \ldots + a_n x^n = a_0 + x [a_1 + x [a_2 + \ldots + x [a_n] \ldots]]$.

$\beta_2 = [\Sigma - (\bigcup_{q_f \in f} \{q_f\} \times T)]^*$.

$\beta_3 = (\Sigma-\#) \Sigma^* \cup \Sigma^* (\Sigma-\#) \cup \Sigma^* \# (\Sigma \cup \lambda)^{n-1} \# \Sigma^*$
$\quad \cup \Sigma^* \# (\Sigma-\#)^{n+1} \Sigma^*$.

And $\beta_4 = \bigcup_{\sigma_1\sigma_2\sigma_3 \in \Sigma^3} \Sigma^* \sigma_1\sigma_2\sigma_3 \Sigma^{n-2} (\Sigma^3 - f_M(\sigma_1\sigma_2\sigma_3)) \Sigma^*$.

Clearly $|\beta_x| \leq C_M \cdot |x|$, where C_M depends only upon M not x.

4.2 Equiv(∪,˙,*,∩)

In this section we show that Equiv(∪,˙,*,∩) is not an element of PTAPE. We show that we can in space 0(kn) embed all n^k- deterministic (or nondeterministic) tape bounded Tm computations into (∪,˙,*,∩) regular expressions. We show that using intersection we can in space 0(kn) write a language $L_k(n)$, such that $L_k(n)$ is essentially S^{n^k}, for some finite alphabet S. We illustrate this with several examples.

Example 1: Let $\# \notin S$. Then $(S^n\#)^* \cap (S^*\#)^n = (S^n\#)^n$. But the length of the (∪,˙,*,∩) regular expression $(S^n\#)^* \cap (S^*\#)^n$ is 0(n). Since the set $(S^n\#)^n$ is finite, the length of any (∪,˙,*) regular expression denoting it must be at least $0(n^2)$. (We note that the expression $(S^n\#)^* \cap (S^*\#)^n$ does not actually satisfy the definition of (∪,˙,*, ∩) regular expressions given in 1.1.8. However as the reader can easily verify, it can be converted into an equivalent legal (∪,˙,*,∩) of length 0(n) by inserting parentheses.)

Example 2: Let $\#, \$ \notin S$. Then $(S^n\{\#,\$\})^* \cap ((S^*\#)^n S^*\$)^* \cap ((S \cup \{\#\})^* S^*\$)^n = ((S^n\#)^n S^n\$)^n$. Again using intersection the length of the expression is 0(n).

In Lemmas 4.2.3, 4.2.4, and 4.2.5 we define $L_k(n)$, determine the length of any string in $L_k(n)$, and determine several other properties of $L_k(n)$. We also show that in space 0(kn) using intersection we can define Cycle($L_k(n)$) =

$\{ x\$y \mid yx\$ \in L_k(n) \}$, which is roughly the set of cyclic permutations of elements of $L_k(n)$.

We use essentially the same definitions of instantaneous descriptions, $\mathrm{Next}_M(x)$, and $C_m(x)$ as those in Section 4.1. However, symbols now are triples corresponding to the symbol triples of Section 4.1. Thus, every tape cell's left and right immediate neighbors are embeddeded in the symbol denoting it.

Definition 4.2.1: Let M be a deterministic Tm with state set $\int$, tape alphabet T, start state q_0, and accepting state q_f. Let $\#_1,\ldots,\#_k \in T \cup (\int \times T)$. Then

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

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

(First, we could do this derectly for nondeterministic Tms but this would only complicate the proofs. Second, S's seemingly odd definition is due to the fact that the symbols $\{ \#_1,\ldots,\#_k \}$ are only used as hash marks. They are not used as computation space. Thus the regular expressions describing the instantaneous descriptions must allow for passing information about change of state and of direction of the tape-head across them.)

Let $s = (a,b,c) \in S$. Then $p_1(s)$, $p_2(s)$, and $p_3(s)$ are the first , second, and third projection functions, respectively, on S^3. Thus, $p_1(s) = a$, $p_2(s) = b$, and $p_3(s) = c$. Finally, for all $s \in T \cup (\int \times T)$, $<s> = \{ t \mid t \in S$ and $pr_2(t) = s \}$. Similarly for all $s \in \{ \#_1, \ldots, \#_k \}$, $<s> = \{ t \mid t \in S$ and $pr_2(t) = s$ or $pr_2(t) \in ((\int \times T) \times s) \}$.

<u>Definition 4.2.2</u>: As in Section 4.1 $C_m(x) = <\#_k> \cdot <i.d._1> <\#_k> \cdot \ldots \cdot <\#_k> \cdot <i.d._n> \cdot <\#_k>$ where

(1) each $i.d._j$ is a valid instantaneous description of the Tm M,

(2) $i.d._1 \in <(q_0, x_1)> <x_2> \ldots <x_n><\#_1>(<\not b>^n <\#_1>)^{n-2} \ldots$,

(3) $i.d._j$ follows from $i.d._{j-1}$ by one application of a move of M, and

(4) $i.d._n$ is the first instantaneous description in $C_M(x)$ in which an accepting state appears.

However, there are several differences.

(a) Each character in $C_M(x)$ is an ordered triple.

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

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

[(a) and (b) guarantee that the left and right contexts of a symbol are compatible with its first and second projections. Thus knowing (a,b,c) we can calculate the middle coordinate of the corresponding element of the next instantaneous description. The reader should note, however, that a and b do not determine the first and third coordinates of the corresponding element of the next instantaneous description.]

(4) The tape-head of the Tm is allowed to move left or right over $\#_1,\ldots,\#_{k-1}$ but not over $\#_k$.

The definition of $C_M(x)$ will become clearer after the statments and proofs of Lemmas 4.2.3, 4.2.4, and 4.2.5.

Lemma 4.2.3: Let k be a positive integer ≥ 4. We define $L_k(n)'$ to be the intersection of the following sets:

(1) $\langle\#_k\rangle \cdot [\ (\ S - (\ \langle\#_{k-1}\rangle, \langle\#_k\rangle\ )\ )^* \cdot \langle\#_{k-1}\rangle \cdot (\ S - (\ \langle\#_{k-1}\rangle,\langle\#_k\rangle\ )\ )^*\ ]^{n-1} \cdot \langle\#_k\rangle,$

(2) $\langle\#_k\rangle \cdot [[(S - (\ \langle\#_{k-2}\rangle,\langle\#_{k-1}\rangle,\langle\#_k\rangle\ )\ )^* \cdot \langle\#_{k-2}\rangle \cdot (\ S - (\ \langle\#_{k-2}\rangle,\langle\#_{k-1}\rangle,\langle\#_k\rangle\ )\ )^*\ ]^{n-1} \cdot (\ \langle\#_{k-1}\rangle, \langle\#_k\rangle\ )\ ]^+,$

...

(k-1) $\langle\#_k\rangle \cdot [\ [\ (\ S - (\ \langle\#_1\rangle,\langle\#_2\rangle,\ldots,\langle\#_k\rangle\ )\ )^* \cdot$

$<\#_1> \cdot$ (S - ($<\#_1>,\ldots,<\#_k>$))* $]^{n-1}$.

($<\#_2>,\ldots,<\#_k>$) $]^+$, and

(k) $<\#_k> \cdot$ [(S - ($<\#_1>,\ldots,<\#_k>$) $)^n \cdot (<\#_1>,\ldots,$
$<\#_k>)]^+$.

Then (a) $\#_k$ occurs exactly twice;

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

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

...

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

<u>Proof</u>: First (1) is the set of all strings β over S such that the first and last characters of β are $\#_k$'s, no other $\#_k$'s appear in β, and exactly n-1 $\#_{k-1}$'s appear in β. (2) is the set of all strings β over S that begin with $\#_k$ and in which exactly n-1 $\#_{k-2}$'s occur between consecutive $\#_k$'s and $\#_{k-1}$'s. Similarly, (k-1) is the set of all strings β over S that begin with $\#_k$ and in which exactly n-1 $\#_1$'s occur between consecutive $\#_2,\ldots,\#_k$'s. Clearly (1) and (2) are true. Therefore, let j > 2. There are n-1 $\#_j$'s for each $\#_i$ such that $j<i\leq k$ except for the first $\#_k$. Therefore the number of $\#_j$'s = $(n-1)\times[(\sum_{i=j+1}^{k}$ (the number of $\#_i$'s)) - 1]. But this equals $(n-1)\times[1+(n-1)+n(n-1)+\ldots+n^{k-(j+2)}(n-1)] = n(n-1)[1+(n-1)\sum_{i=0}^{k-(j+3)} n^i =$

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

$L_k(n) = <\#_k>\backslash L_k(n)'$. Thus the number of occurences of the various levels of markers and nonmarkers in $L_k(n)$ is the same as that of $L_k(n)'$ except that only 1 $\#_k$ appears in $L_k(n)$.

Lemma 4.2.4: All strings β in $L_k(x)$ have the same length. Furthermore $|\beta| > n^k$.

Proof: For each marker there are n different nonmarkers. Thus the number of nonmarkers appearing in β = $n[1+(n-1)+\ldots+n^{k-2}(n-1)] = n^k$.

Lemma 4.2.5: Cycle($L_k(n)$) equals the intersection of the following sets:

(1) $[\ S^{n+1} \cap (\ S - (\ <\#_1>,\ldots,<\#_k>\ )\ )^* \cdot (\ <\#_1>,\ldots,<\#_k>\ ) \cdot (\ S - (\ <\#_1>,\ldots,<\#_k>\ )\ )^* .]^+$;

(2) $(\ S - (\ <\#_1>,\ldots,<\#_k>\ )\ )^* \cdot (\ <\#_1>,\ldots,<\#_k>\ )$ $[\ (\ S - (\ <\#_1>,\ldots,\ <\#_k>\ )\ )^n \cdot (\ <\#_1>,\ldots,<\#_k>\ )\ ]^*$ $\cdot\ (\ S - (\ <\#_1>,\ldots,<\#_k>\ )\ )^*$;

(3) $(\ S - <\#_k>\ )^* \cdot <\#_k> \cdot (\ S - <\#_k>\ )^*$;

(4) $[\ (\ S - <\#_{k-1}>\ )^* \cdot <\#_{k-1}> \cdot (\ S - <\#_{k-1}>\ )^*]^{n-1}$;

(5,1) $(\ S - (\ <\#_{k-1}>,<\#_k>\ )\ )^*\ \ (\ <\#_{k-1}>,<\#_k>\ ) \cdot$ $[\ \{\ (\ S - (\ <\#_{k-2}>,<\#_{k-1}>,<\#_k>\ )\ )^* \cdot <\#_{k-2}> \cdot$ $(\ S - (\ <\#_{k-2}>,<\#_{k-1}>,<\#_k>\ )\ )^*\ \}^{n-1} \cdot (\ <\#_{k-1}>,$ $<\#_k>\ )\ ]^* \cdot (\ S - (\ <\#_{k-1}>,<\#_k>\ )\ )^*$;

(6,1) $[\ (\ S - (\ \langle\#_{k-2}\rangle, \langle\#_{k-1}\rangle, \langle\#_k\rangle\ )\ )^* \cdot (\ \langle\#_{k-2}\rangle, \lambda\ ) \cdot (\ S - (\ \langle\#_{k-2}\rangle, \langle\#_{k-1}\rangle, \langle\#_k\rangle\ )\ )^*\ ]^{n-1} \cdot (\ \langle\#_{k-1}\rangle, \langle\#_k\rangle\ ) \cdot S^* \cdot (\ \langle\#_{k-1}\rangle, \langle\#_k\rangle\ ) \cdot [\ (\ S - (\ \langle\#_{k-2}\rangle, \langle\#_{k-1}\rangle, \langle\#_k\rangle\ )\ )^* \cdot (\ \langle\#_{k-2}\rangle, \lambda\ ) \cdot (\ S - (\ \langle\#_{k-2}\rangle, \langle\#_{k-1}\rangle, \langle\#_k\rangle)\ )^*\ ]^{n-1}$;

(7,1) $[\ [\ (\ S - \langle\#_{k-2}\rangle\ )^* \cdot \langle\#_{k-2}\rangle \cdot (\ S - \langle\#_{k-2}\rangle\ )\ ]^n\ ]^+$;

...

(5,k-2) $(\ S - (\ \langle\#_2\rangle, \ldots, \langle\#_k\rangle\ )\ )^* \cdot (\ \langle\#_2\rangle, \ldots, \langle\#_k\rangle\ ) \cdot [\ \{\ (\ S - (\ \langle\#_1\rangle, \ldots, \langle\#_k\rangle\ )\ )^* \cdot \langle\#_1\rangle \cdot (\ S - (\ \langle\#_1\rangle, \ldots, \langle\#_k\rangle\ )\ )^*\ \}^{n-1} \cdot (\ \langle\#_2\rangle, \ldots, \langle\#_k\rangle\ )\ ]^* \cdot (\ S - (\ \langle\#_2\rangle, \ldots, \langle\#_k\rangle\ )\ )^*$;

(6,k-2) $[\ (\ S - (\ \langle\#_1\rangle, \ldots, \langle\#_k\rangle\ )\ )^* \cdot (\ \langle\#_1\rangle, \lambda\ ) \cdot (\ S - (\ \langle\#_1\rangle, \ldots, \langle\#_k\rangle\ )\ )^*\ ]^{n-1} \cdot (\ \langle\#_2\rangle, \ldots, \langle\#_k\rangle\ ) \cdot S^* \cdot (\ \langle\#_2\rangle, \ldots, \langle\#_k\rangle\ ) \cdot [\ (\ S - (\ \langle\#_1\rangle, \ldots, \langle\#_k\rangle))^* \cdot (\ \langle\#_1\rangle, \lambda\ ) \cdot (\ S - (\ \langle\#_1\rangle, \ldots, \langle\#_k\rangle\ )\ )^*\ ]^{n-1}$;

(7,k-2) $[\ [\ (\ S - \langle\#_1\rangle\ )^* \cdot \langle\#_1\rangle \cdot (\ S - \langle\#_1\rangle\ )^*\ ]^{n-1}\ ]^+$.

<u>Proof</u>: (1) is the set of all strings β over S of length $\geq$ n+1 such that $\beta = \beta_1 \cdot \ldots \cdot \beta_m$, $|\beta_1| = \ldots = |\beta_m| = n+1$, and exactly 1 marker appears in each β_i. [The reader should note that the proof of 4.2.4 shows that $\alpha \in L_k(n) \rightarrow |\alpha| = n^k + n^{k-1} = n^{k-1}(n+1)$. Thus the length of every string in Cycle($L_k(n)$) is

divisible by n+1.] (2) is the set of strings β over S such that between every 2 consecutive markers occuring in β there are exactly n nonmarkers.

(3) is the set of all strings β over S containing exactly 1 occurrence of $<\#_k>$. Similarly, (4) is the set of all strings β over S containing exactly n-1 occurrences of $<\#_{k-1}>$. (5,1) is the set of all strings β over S containing at least 1 $<\#_{k-1}>$ or $<\#_k>$ such that between every 2 consecutive occurrences of $<\#_{k-1}>$ or $<\#_k>$, there are exactly n-1 $<\#_{k-2}>$'s. (6,1) is the set of all strings β over S containing at least 2 $<\#_{k-1}>$ or $<\#_k>$'s such that at most n-1 $<\#_{k-2}>$'s occur, before the first $<\#_{k-1}>$ or $<\#_k>$ in β, and at most n-1 $<\#_{k-2}>$'s occur after the last $<\#_{k-1}>$ or $<\#_k>$ in β. (7,1) is the set of all strings β over S containing at least n $<\#_{k-2}>$'s such that the number of $<\#_{k-2}>$'s in β is divisible n. The remaining sets are defined analogously.

$\beta \in (1) \cap (2)$ implies that β is of the form $[\, \alpha_1\mu_1\beta_1 \,]\cdot[\, \alpha_2\mu_2\beta_2 \,]\cdot\ldots\cdot[\, \alpha_m\mu_m\beta_m \,]$, where $\mu_1,\ldots,\mu_m$ are markers, $|\alpha_1| = \ldots = |\alpha_m|$, $|\beta_1| = \ldots = |\beta_m|$, $|\beta_1\alpha_2| = \ldots = |\beta_m\alpha_1| = n$, and the only markers appearing in β are $\mu_1,\ldots,\mu_m$.

Let $\beta \in (1) \cap \ldots \cap (7,k-2)$. We show that $\beta \in \mathrm{Cycle}(\mathbf{L}_k(n))$. To do this we show that for any j, $1 \leq j \leq k$, the number and distribution of $<\#_j>$'s in β are the same as those of some string in $\mathrm{Cycle}(\mathbf{L}_k(n))$. From (3) and (4) exactly 1 $<\#_k>$ and exactly n-1 $<\#_{k-1}>$'s appear in β.

Assume that for all i, $j < i \leq k$, the $<\#_i>$'s appearing in β have both the right number and distribution. By (5,j-2) between any 2 consecutive markers $<\#_{i1}>,<\#_{i2}>$ in β with i1,i2 > 2, there are exactly n-1 $<\#_j>$'s. Hence, the number of $<\#_j>$'s between the first and last occurrence of a $<\#_{j+1}>, \ldots, <\#_k>$ in β = $(n-1)\times[\ (n-1) \sum_{i=0}^{k-(j+2)} n^i\ ] = (n-1)\times[\ n^{k-(j+1)} - 1\ ]$, which is exactly n-1 fewer $<\#_j>$'s than the number in any string in $\mathrm{Cycle}(L_k(n))$.

Let r be the number of $<\#_j>$'s before the first $<\#_{j+1}>, \ldots,$ or $<\#_k>$ in β. Let s be the number of $<\#_j>$'s after the last $<\#_{j+1}>, \ldots,$ or $<\#_k>$ in β. By (6,j-2) $r,s \leq n-1$. By (7,j-2) the number of $<\#_j>$'s in β is divisible by n. Hence, if there are too few or too many $<\#_j>$'s in β, there are at least n too few or at least n too many. But there are at most n-1 too few $<\#_j>$'s in β as shown above. Hence, there are not too few $<\#_j>$'s in β. If there are too many $<\#_j>$'s in β then there are at least n too many, which implies that $r+s \geq n+(n-1)$. But $r+s \leq 2(n-1) = 2n-2$. Hence the number of $<\#_j>$'s in β is correct. It is clear that the distribution of $<\#_j>$'s in β is correct also. Thus if $\beta \in (1) \cap \ldots \cap (7,k-2)$ and $\beta \notin \mathrm{Cycle}(L_k(n))$, it is not because of the number or distribution of any of the markers appearing in β. Finally as shown above, the number and distribution of the nonmarkers in β is correct. Thus, $\beta \in (1) \cap \ldots \cap (7,k-2) \rightarrow \beta \in \mathrm{Cycle}(L_k(n))$.

Finally it is clear that $\mathrm{Cycle}(L_k(n)) \subset (1) \cap \ldots \cap (7,k-2)$, since every string in $\mathrm{Cycle}(L_k(n))$ satisfies (1), (2), ... , and (7,k-2).

Theorem 4.2.5: $\mathrm{Equiv}(\cup,\cdot,*,\cap) \notin$ PTAPE.

Proof: Let M be the deterministic Tm described in Definition 4.2.1. Let M be n^k tape bounded. We construct a $(\cup,\cdot,*,\cap)$ regular expression Γ such that $L(\Gamma)$ is the set of invalid computations of M on x, i.e., $L(\Gamma) = S^* - C_M(x)$, and $|\Gamma|$ is $O(k|x|)$.

$\lambda \in S^*$ is not a valid computation iff

(1) $\lambda \notin <\#_k> \cdot L_k(n)^*$, where $n = |x|$, or

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

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

(4) λ doesnot start with the right initial instantaneous description, or

(5) λ doesnot have an accepting state, or

(6) λ makes an error between one instantaneous description and the next.

We can easily write out a $(\cup, \cdot, *, \cap)$ expression for (2), (3), and (5). If $\lambda \notin <\#_k> \cdot L_k(n)^*$, then

(i.0) λ is the empty word , or

(i.1) λ is of length at least two and doesnot begin and end with $<\#_k>$, or

(i.2) between 2 consecutive $<\#_k>$'s in λ there are fewer or more than n-1 $<\#_{k-1}>$'s, or

(i.3) between 2 consecutive $<\#_{k-1}>$'s or $<\#_k>$'s there are fewer or more than n-1 $<\#_{k-2}>$'s, or... or

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

(i.k+1) between 2 consecutive markers there are fewer or more than n nonmarkers.

We leave it for the reader to construct $(\cup, \cdot, *)$ regular expressions for (i.0),...,(i.k+1).

(4) Since we know that all strings with improper marker-nonmarker structure are covered by the union of the expressions for (i.0),..., and (i.k+1), the only possibilities for an invalid initial instantaneous description that need be covered are

(a) λ does not begin with $<\#_k> \cdot <(q_0,x_1)> \cdot \ldots \cdot <x_n> \cdot <\#_1>$ and

(b) there is a nonmarker nonblank to the right of the first $<\#_1>$ and to the left of the second $<\#_k>$ in λ. Again the expression is easily constructed and is left to the reader.

(6) Finally if λ contains an error then $\lambda \in \bigcup_{a \in S} S^* \; a \;\; \mathrm{Cycle}(L_k(n)) \left[\bigcup_{\{b \mid p_1(b) \neq f_M(a)\}} {}^{<b>} \right] S^*$. If we choose that string in $L_k(n)$ whose marker-nonmarker structure alligns with that of S^*a then we find any errors that occur. If we choose $\beta \in \mathrm{Cycle}(L_k(n))$ such that the marker-nonmarker structure of β doesnot allign with that of S^*a , then $\lambda \in S^*a \cup <b>S^*$ implies $\lambda \notin <\#_k> \cdot L_k(n)^*$ and therefore is not a valid computation.

4.3 Empty($\cup,\cdot,*,\sim$)

Definition 4.3.1: We define $g(i,j)\colon N\times N \to N$ as follows:

$\forall j\in N,\ g(0,j) = j,$

$\forall i \geq 1,\ g(i,j) = 2^{g(i-1,j)}.$

A language L_0 is elementary recursive if there exists i_0 such that $L_0 \in$ Ndtape($g(i_0,n)$), i.e., L_0 is recognized by some $\left.2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}}\right\} i_0$ 2's non-deterministic tape bounded Tm.

We will only sketch the details of a proof that Empty($\cup,\cdot,*,\sim$) is not elementary recursive. This result and the basic idea of the proof prsented here are due to A.R. Meyer and L.J. Stockmeyer.

As in Section 4.2 we want ($\cup,\cdot,*,\sim$) regular expressions $R_k(n)$ of relatively short length such that roughly $L(R_k(n)) = \Sigma^{g(k,n)}$. Again we use $R_k(n)$ as a yardstick to detect errors between consecutive i.d.'s of length $g(k,n)$. Again we use the set Cycle($x\#$) = $\{x_2\#x_1 \mid x_1x_2 = x\}$. We change the definition of valid computation slightly from that in 4.2.

Let M be a Tm with state set S, tape alphabet T, designated start state $q_0 \in S$, and designated accepting state q_f. Let q_f be final. Let $y = y_1 \ldots y_n$ be an input to M. Let $\Sigma_1 = [T \cup (S \times T)]^3 \cup \{\#,\$\}$. Let $\Sigma \cap \Sigma_1 = \phi$. Let $x \in \Sigma^*$. Let $\Sigma_2 = \Sigma \cup \{\#,\$\}$. Then a valid computation of M on y with yardstick Cycle(x#) is a string in $(\Sigma_1 \times \Sigma_2)^*$ of the form

$$\begin{array}{c} \$\ C_0\ \#\ C_1\ \#\ \ldots\ \#\ C_n\ \# \\ \$\ \ x\ \#\ \ x\ \#\ \ldots\ \#\ \ x\ \# \end{array}, \text{ where } \$\ C_0\ \#\ \ldots\ \#\ C_n\#$$

is a valid computation of M on y.

<u>Lemma 4.3.2</u>: Let $h: T^* \to \Delta^*$ be a length preserving homomorphism. Let R be a $(\cup,\cdot,*,\sim)$ regular expression over T^* such that $L(R) \subset \Delta^*$. Then there exists a polynomial time bounded deterministic linear bounded automaton M such that M, when given R as input, outputs a $(\cup,\cdot,*,\sim)$ regular expression S such that $L(S)=h^{-1}(L(R))$.

<u>Proof</u>: Since h is a length preserving homomorphism we have $h^{-1}(A \cup B) = h^{-1}(A) \cup h^{-1}(B) \in h^{-1}(A \cdot B) = h^{-1}(A) \cdot h^{-1}(B)$, $h^{-1}(A^*) = (h^{-1}(A))^*$, and $h^{-1}(\sim A) = \sim(h^{-1}(A))$.

<u>Lemma 4.3.2</u>: Let R be a $(\cup, \cdot, *, \sim)$ regular expression denoting cycle $(x\#)$. Then there exists a deterministic polynomial time bounded Tm μ such that μ, given R as input, outputs a $(\cup, \cdot, *, \sim)$ regular expression S, which denotes the complement of the set of cyclic permutations of the valid computations of M with yardstick cycle $(x\#)$.

<u>Proof</u>: We sketch some of the more important details. A string γ is <u>not</u> a cyclic permutation of a valid computation of M with yardstick cycle $(x\#)$ iff

1) γ's lower track is not a cyclic permutation of $\$ (x\#)^*$;

2) γ's lower track is a cyclic permutation of $\$ (x\#)^*$ but its upper track is not a cyclic permutation of a valid computation of M on y;

or 3) the upper and lower tracks do not correspond to the same cyclic permutation.

The strings satisfying (3) are those which do not satisfy 1 or 2 in which the "\$"'s appearing in the lower and upper tracks do not coincide.

Let h_o: $\Sigma^* \to \Sigma^*$ be the length preserving homomorphism defined by $h_o(\#) = h_o(\$) = \#$ and $\delta \epsilon \Sigma_2$, $h_o(\sigma) = \sigma$. Then $h_o^{-1}(L(R)) =$ $\{x_1 \# x_2, x_1 \$ x_2 \mid x_1 \# x_2 \epsilon \text{ Cycle } (x\#)\}$. Let R' be the $(\cup, \cdot, *, \sim)$ regular expression output by the Tm of 4.3.2, when given input R. Then $L(R') = h_o^{-1}(L(R))$ and $|R'| \le C \cdot |R|$.

Let h_1 and h_2: $(\Sigma_1 \times \Sigma_2)^* \to \Sigma_1$ and Σ_2 be the length preserving homomorphisms defined by

$$h_1((a \times b)) = a \text{ and } h_2((a \times b)) = b.$$

All strings satisfying (1) are given by

$$\sim(\overline{U}_1 \cap \overline{U}_2 \cap \overline{U}_3), \text{ where}$$

$$\overline{U}_1 = h_2^{-1} [\Sigma^* \cdot (\# \cup \$) \cdot (R' \cap (\Sigma^* \# \cup \Sigma^* \$))^* \Sigma^*];$$

$$\overline{U}_2 = h_2^{-1} [(\Sigma_2 \cup \#)^* \$ (\Sigma_2 \cup \#)^*]; \text{ and}$$

$$\overline{U}_3 = h_2^{-1}[(R')^*].$$

This follows since

(a) $\gamma \epsilon \overline{U}_2 \Rightarrow$ exactly 1 "\$" occurs in γ's lower track;

(b) $\gamma \epsilon \overline{U}_2 \cap \overline{U}_3 \Rightarrow \gamma$'s lower track is of the form $y_1 \# y_2 \; y_3 \# y_4 \; y_5 \# \ldots y_i \$ y_{i+1} \# \ldots y_j \# y_{j+1}$, where $y_1\#y_2,\ y_3\#y_4,\ y_5\#y_6, \ldots, y_j\#y_{j+1} \in$ $\{x_1\#x_2,\ x_1\$x_2 \mid x_1\#x_2 \in \text{Cycle}\ (x\#)\}$;

and c) $\gamma \in \overline{U}_1 \rightarrow y_2y_3 = y_4y_5 = \ldots = y_{j-1}y_j = x.$

The set of strings satisfying (2) is harder to describe. We only sketch the most important details. Suppose β does not satisfy (2), then β is of the form

$$\begin{pmatrix} y_2 \# C_j \# C_{j+1} \# \ldots \# C_k \$ C_o \# \ldots \# C_{j-2} \# y_1 \\ x_2 \# x_1\ x_2 \# x_1\ x_2 \# \ldots \# x_1\ x_2 \# x_1\ x_2 \# \ldots \# x_1 x_2 \# x_1 \end{pmatrix}$$

where

a) $C_{j-1} = y_1y_2$

b) C_o is an initial i.d. and C_k is a final i.d.;

c) for each i $C_{i+1} \in \text{Next}_M\ (C_i)$;

d) $C_j = y_1'y_2'$, with $|y_1'| = |y_1|$ and $|y_2'| = |y_2|$;

e) $C_{j-2} = y_1''\ y_2''$, with $|y_1''|=|y_1|$ and $|y_2''|=|y_2|$; and

f) $x_1x_2 = x$, with $|x_1|=|y_1|$ and $|x_2|=|y_2|$

It is easy to find any errors between consecutive i.d.'s to the left or the right of the "\$".

$$\bigcup_{a \in \Sigma_1 - \$} (\Sigma_1 - \$ \times \Sigma_2 - \$)^*\ h_1^{-1}(a)\ [h_2^{-1}(R_1) \cap ((\Sigma_1 - \$) \times (\Sigma_2 - \$))^*]$$

$$h_1^{-1}(\Sigma_1 - f_M(a))\ (\Sigma_1 \times \Sigma_2)^* \quad \cup$$

$$\bigcup_{a\in\Sigma_1-\$} (Z_1 x \Sigma_2)^* h_1^{-1}(a)[h_2^{-1}(R_1)\cap((\Sigma_1-\$)x(\Sigma_2-\$))^*\rceil$$

$$h_1^{-1}(\Sigma_1-f_M(a)) \cdot (\Sigma_1-\$,\ \Sigma_2-\$)^*.$$

This separation is necessary since the first i.d. need not follow from the last one.

We must also, however, catch errors that occur between y_1 and y_1', and between y_2 and y_2'' as pictured below.

$$y_2 \ \# \ y_1' \ y_2' \ \# \ \ldots \ \# \ y_1'' \ y_2'' \ \# \ y_1$$

But this can be accomplished as follows:

$$\bigcup_{b\neq f_M(a)} (\Sigma_1-(\#,\$),\ \Sigma_2-(\#,\$))^* \cdot \{(\#,\#) \cup (\$,\$)\} \cdot$$

$$(\Sigma_1-(\#,\$),\ \Sigma_2-(\#,\$))^* \ h_1^{-1}(b)\ h_2^{-1}\lceil\text{cycle } (\$(x\#)^*)\rceil$$

$$h_1^{-1}(a)\ (\Sigma_1-(\#,\$),\ \Sigma_2-(\#,\$))^*, \text{ etc.}$$

(Note, y_1 is characterized by the fact that no "#" or "\$" appears to its right.) The remaining constructions are intricate but straight forward given the above.

<u>Theorem 4.3.4 (Meyer and Stockmeyer)</u>: Empty $(\cup, \cdot, *, \sim)$ is not elementary recursive.

<u>Proof</u>: The theorem follows from repeated applications of Lemma 4.3.3.

<u>Note</u>: The details of this proof where worked out with Professor John Hopcroft and also appear in Aho, Hopcroft, and Ullman [73].

4.4 Two Way Finite Automata

We show that the emptiness problem for 2-way deterministic finite automata is as hard as the equivalence problem for regular expressions.

Definition 4.4.1 (Hopcroft and Ullman [69]): A 2-way deterministic finite automation M over Σ is a 5-tuple (k,Σ,δ,q_o,F), where K is a finite set of states written as binary integers, δ: K x $\Sigma \rightarrow$ K x $\{-1,0,1\}$, $q_o \in K$ is the start state, and $F \subseteq K$ is the set of final states $\delta(q,a) = (p,D)$, $p \in K$ and $D \in \{-1,0,1\}$ is interpreted to mean that M, in state q, scanning the input symbol a, will move its input head one cell to the left, to the right, or not move its input head at all, depending on whether D equals -1,1,0, respectively. L(M), the accepted language of M, is defined in the standard manner. (For more details see Hopcroft and Ullman [69].)

Definition 4.4.2: Notempty (2dfa) = $\{M | M$ is a 2-way deterministic finite automation with tape alphabet $\{0,1\}$ and $L(M) \neq \emptyset\}$. Inequiv (2dfa) is defined analogously.

<u>Theorem 4.4.3</u>: PTAPE $\leq_{\text{PTIME}}$ Notempty (2dfa).

Proof: There exists a deterministic polynomial time bounded Tm $T(\Delta)$ such that $T(\Delta)$ given input (M,x), where M is a deterministic lba with tape alphabet $T \subseteq \Delta$, state set S, start state q_o and accepting state q_f, and $x \in T^*$, outputs a 2-way deterministic finite automation $A_{M,x}$ such that $L(A_{M,x}) = C_M(x)$. Thus, $L(A_{M,x}) = \emptyset$ iff $x \notin L(M)$. Given an input y $A_{M,x}$ operates as follows:

1) It checks that y begins with

$$\# \; (q_o,x_1) \; \ldots y_n \; \#;$$

2) It checks that $y \in \# \cdot [(\Sigma-\#)^n \; \#]^+$;

3) It checks that $y \in \Sigma^* [\bigcup_{t \in T} (q_f \times t)] \cdot \Sigma^*$;

and 4) If y satisfies 1), 2), and 3) then $A_{M,x}$ checks that y contains no errors. To do this $A_{M,x}$ uses its ability to move both ways on its input tape. $A_{M,x}$ stores consecutive triples $(\sigma_1,\sigma_2,\sigma_3)$ of characters of y in its finite control, calculates $f_m(\sigma_1,\sigma_2,\sigma_3)$ counts n+1 tape cells to the right and compares the contents of this tape cell with $f_M(\sigma_1,\sigma_2,\sigma_3)$. If they are not equal A_M,x rejects y. Otherwise, it counts n squares to the left and repeats the process.

It is clear from the description of $A_{M,x}$ above that

a) M exists and b) the construction of M is deterministic polynomial time bounded.

<u>Corollary 4.4.4</u>: PTAPE $\leq_{PTIME}$ {M | M is a 2-way deterministic finite automation with tape alphabet {0,1} and $L(M) \neq \{0,1\}^*$}.

<u>Proof</u>: We can easily modify $A_{M,x}$ in the proof of 4.4.2 so that $L(A_{M,x}) = \Sigma^* - C_M(x)$.

Chapter 5 HARD LANGUAGES

5.0 Introduction

In this chapter we investigate the computational complexity (both time and space) of many decidable predicates in formal language and automata theory. We find natural sufficient conditions for predicates on the regular sets to be as hard to decide as the emptiness or equivalence problems for several different kinds of regular set descriptors (e.g., $(\cup,\cdot,*)$, $(\cup,\cdot,*,\cap)$, and $(\cup,\cdot,*,\sim)$ regular expressions.) These conditions are identical to those of Theorems 3.3, 3.4, and 3.6. We also show that structural equivalence (defined below) for cfgs is as hard to decide as is the equivalence problem for $(\cup,\cdot,*)$ regular expressions.

Definition 5.0.1: Let $L \subset \{0,1\}^*$ and let $x \in \{0,1\}^*$. Then

$$x\backslash L = \{ y \mid xy \in L \} \text{ and } L/x = \{ y \mid yx \in L \}.$$

Definition 5.0.2: Let $\mathbf{L} = \{ L_1, L_2, \ldots , L_n, \ldots \}$, where each $L_i \subset \{0,1\}^*$. Then $x\backslash\mathbf{L} = \{ x\backslash L_1, x\backslash L_2, \ldots , x\backslash L_n, \ldots \}$ and $\mathbf{L}/x = \{ L_1/x, L_2/x, \ldots , L_n/x, \ldots\}$.

5.1 Predicates on the Regular Sets

In Chapter 4 we saw the following:

(1) $\{ R | R$ is a $(\cup, \cdot, *)$ regular expression over $\{0,1\}$ and $L(R) \neq \{0,1\}^* \} \underset{ptime}{\geq}$ PTAPE and is a Savitch CSL;

(2) $\{ R | R$ is a $(\cup, \cdot, *, \cap)$ regular expression over $\{0,1\}$ and $L(R) \neq \emptyset \} \underset{ptime}{\geq}$ PTAPE;

(3) $\{ R | R$ is a $(\cup, \cdot, *, \cap)$ regular expression over $\{0,1\}$ and $L(R) \neq \{0,1\}^* \}$ is <u>not</u> an element of PTAPE;

(4) $\{ M | M$ is a 2-way deterministic finite automaton (abbreviated 2dfa) with tape alphabet $\{0,1\}$ and $L(M) \neq \emptyset$ [or $L(M) \neq \{0,1\}^*$] $\} \underset{ptime}{\geq}$ PTAPE; and

(5) $\{ R | R$ is a $(\cup, \cdot, *, \sim)$ regular expression over $\{0,1\}$ and $L(R) \neq \{0,1\}^* \}$ is not elementary recursive.

Using these results we classify the time and tape complexity of many predicates on the regular sets. Before stating and proving our main theorems we need one technical result.

<u>Lemma 5.1.1</u>: Let $h: \{0,1\}^* \rightarrow \{0,1\}^*$ be the epsilon-free homomorphism defined by $h(0) = 00$ and $h(1) = 01$. Then there exists a deterministic polynomial time bounded lba M such that M, when given a $(\cup, \cdot, *)$, $(\cup, \cdot, *, \cap)$, or $(\cup, \cdot, *, \sim)$ regular expression R over $\{0,1\}$ as input, outputs a $(\cup, \cdot, *)$, $(\cup, \cdot, *, \cap)$, or $(\cup, \cdot, *, \sim)$ regular expression S over $\{0,1\}$ such that

$h(L(R)) = L(S)$.

Proof: $h(A \cup B) = h(A) \cup h(B)$, $h(A \cdot B) = h(A) \cdot h(B)$, $h(A^*) = [h(A)]^*$, $h(A \cap B) = h(A) \cap h(B)$, and $h(\sim A) = \sim h(A) \cap (00+01)^*$. M applies these rules recursively on R.

Theorem 5.1.2: Let P be any predicate on the regular sets over $\{0,1\}$ such that

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

(b) $P_L = \bigcup_{x \in \{0,1\}^*} \{ x \backslash L \mid P(L) \text{ is true} \}$ or

$P_R = \bigcup_{x \in \{0,1\}^*} \{ L / x \mid P(L) \text{ is true} \}$

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

Then (1) $\{ R \mid R$ is a $(\cup, \cdot, *)$ regular expression over $\{0,1\}$ and $P(L(R))$ is false $\} \geq_{ptime}$ PTAPE and is an element of DCSL only if DCSL = NDCSL;

(2) $\{ R \mid R$ is a $(\cup, \cdot, *, \cap)$ regular expression over $\{0,1\}$ and $P(L(R))$ is false $\}$ is not an element of PTAPE;

(3) $\{ R \mid R$ is a $(\cup, \cdot, *, \sim)$ regular expression over $\{0,1\}$ and $P(L(R))$ is false $\}$ is not elementary recursive; and

(4) $\{ M \mid M$ is a 2dfa with tape alphabet $\{0,1\}$ and $P(L(M))$ is false $\} \geq_{ptime}$ PTAPE.

Proof: From Chapter 4 we already know that (1), (2), (3), and (4) are true when P is "$L(R)$ equals $\{0,1\}^*$" or P is

"$L(M)$ equals $\{0,1\}^*$". Let L_0 be a regular set over $\{0,1\}$ such that $L_0 \notin P_L$. Let h be the epsilon-free homomorphism defined in Lemma 5.1.1. Then applying 5.1.1 given R, a $(\cup,\cdot,*)$, $(\cup,\cdot,*,\cap)$, or $(\cup,\cdot,*,\sim)$ regular expression over $\{0,1\}$, we can effectively find in linear space and deterministic polynomial time in $|R|$ a $(\cup,\cdot,*)$, $(\cup,\cdot,*,\cap)$, or $(\cup,\cdot,*,\sim)$ regular expression S such that

$$\begin{aligned} L(S) &= h(L(R))10(0+1)^* + (00+01)^*10L_0 + \\ &\quad \sim[(00+01)^*10(0+1)^*] \\ &= h(L(R))10(0+1)^* + (00+01)^*10L_0 + \\ &\quad (00+01)^*[\lambda+0+1+11(0+1)^*]. \end{aligned}$$

<u>Case 1</u>: $L(R) = (0+1)^*$. Then $h(L(R)) = (00+01)^*$ and $L(S) = (0+1)^*$. Hence, $P(L(S))$ is true.

<u>Case 2</u>: $L(R) \neq (0+1)^*$. Then $\exists x \in (0+1)^* - L(R)$. Hence $h(x) \in (00+01)^* - h(L(R))$. But $P(L(S))$ true implies $h(x)10\backslash L(S) \in P_L$. Hence, $P(L(S))$ is false. [$h(x)10\backslash L(S) = L_0$.]

Therefore, $P(L(S))$ is true iff $L(R) = (0+1)^*$. (1), (2), and (3) follow immediately since the construction of S above, given R, requires at most polynomial time on some deterministic lba. (4) follows since given M, we can in deterministic polynomial time in $|M|$, construct a 2dfa M' such that $L(M') = h(L(M))10(0+1)^* + (00+01)^*10L_0 + \sim[(00+01)^*10(0+1)^*]$. M' behaves as follows- M' given input $x \in \{0,1\}^*$ first checks if $x \in (00+01)^*10L_0 + \sim[(00+01)^*10(0+1)^*]$. If x is an element, then M' accepts it. Otherwise, M' applies M as a subroutine to x.

The proof when $P_R \subsetneq$ regular sets over $\{0,1\}$ is left to the reader.

<u>Theorem 5.1.3</u>: Let P be any predicate on the regular sets over $\{0,1\}$ such that

(a) $P(\emptyset)$ is true and

(b) $P_L = \cup_{x \in \{0,1\}^*} \{ x\backslash L \mid P(L) \text{ is true} \}$

or $P_R = \cup_{x \in \{0,1\}^*} \{ L/x \mid P(L) \text{ is true} \}$

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

Then (1) $\{ R \mid R$ is a $(\cup,\cdot,*,\cap)$ regular expression over $\{0,1\}$ and $P(L(R))$ is false $\} \underset{\text{ptime}}{\geq}$ PTAPE;

(2) $\{ R \mid R$ is a $(\cup,\cdot,*,\sim)$ regular expression over $\{0,1\}$ and $P(L(R))$ is false $\}$ is <u>not</u> elementary recursive; and

(3) $\{ M \mid M$ is a 2dfa with tape alphabet $\{0,1\}$ and $P(L(M))$ is false $\} \underset{\text{ptime}}{\geq}$ PTAPE.

<u>Proof</u>: Assume $P_L \subsetneq$ regular sets over $\{0,1\}$. Let L_0 be a regular set over $\{0,1\}$ not in P_L. The proof is almost identical to that of 5.1.2 except that

$L(S) = h(L(R))10L_0$ and $L(M') = h(L(M))10L_0$.

If $L(R) = \emptyset$ then $h(L(R)) = \emptyset$. Hence $L(S) = \emptyset$ and $P(L(S))$ is true. If $L(R) \neq \emptyset$ then $\exists x \in L(R)$. But $P(L(S))$ true implies that $h(x)10\backslash L(S) = L_0 \in P_L$. Hence, $P(L(S))$ is true iff $L(R) = \emptyset$. Similarly, $P(L(M))$ is true iff $L(M) = \emptyset$. The case when $P_R \subsetneq$ regular sets over $\{0,1\}$ is left for the reader.

5.2 A Natural Complexity Core

The surprising fact is that almost all predicates on the regular sets studied in the literature satisfy the conditions of Theorems 5.1.2 and 5.1.3.

Theorem 5.2.1: The following predicates satisfy the conditions of Theorem5.1.2:

(1) $R = \{0,1\}^*$;

(2) R is a star event, i.e., $R = R^*$;

(3) R is a code event, i.e., $\exists w_1, \ldots, w_n$ such that $R = (w_1 \cup \ldots \cup w_n)^*$;

(4) R is an ultimate, reverse ultimate, or generalized ultimate definite event;

(5) R is a comet, reverse comet, or generalized comet event;

(6) $R = \gamma(R)$, where $\gamma(R) = \{ y \mid \exists x \in R \text{ and } |y| = |x| \}$;

(7) $R = \mathrm{Init}(R)$, where $\mathrm{Init}(R) = \{ y \mid \exists x \text{ and } yx \in R \}$;

(8) $R = \mathrm{Final}(R)$, where $\mathrm{Final}(R) = \{ y \mid \exists x \text{ and } xy \in R \}$;

(9) $R = \mathrm{Subword}(R)$, where $\mathrm{Subword}(R) = \{ y \mid \exists x, z \text{ and } xyz \in R \}$;

(10) R is cofinite;

(11) For all $k \geq 1$, R is a k-definite, k-reverse

definite, or k-generalized definite event;

(12) R is a definte, reverse definite, or generalized definite event;

(13) For all $k \geq 1$, R is k-testable in the strict sense;

(14) For all $k \geq 1$, R is k-testable;

(15) R is locally testable in the strict sense;

(16) R is locally testable;

(17) R is a star-free, noncounting, or group-free event;

(18) R is of restricted star height 1;

(19) R is accepted by some strongly connected deterministic finite automaton; and

(20) $R = [R]^{rev}$,

(The definition of 4 may be found in Paz and Peleg[65]; that of 5 may be found in Brzozowski and Cohen[69]. The definitions of 11 through 17 may be found in McNaughton and Papert [71]; that of 18 may be found in McNaughton[69]; and that of 19 may be found in Hartmanis and Stearns[66].)

<u>Proof</u>: 1, 2, 3, 4, and 5 are proved similarly. Here, we prove only 5. A regular set R is a comet event if there are regular sets R_1 and R_2 such that $R = R_1 \cdot R_2$, $R_1 = R_1^*$, and $R_1 \neq \{\lambda\}$. Let P be the predicate "R is a comet event." Then $P_R \subseteq \{\phi\} \cup$ the infinite

regular sets over $\{0,1\}$. This follows since for all $x \in \{0,1\}^*$ either $R/x = \emptyset$ or $\exists y = y_1 \cdot y_2 \in R_1 \cdot R_2$ such that $y/x = w$. But $R_1 = R_1^*$, $R_1 \neq \{\lambda\}$ and $R/x \neq \emptyset$ imply there exists a string $z \in R_1$ such that $|z| \geq 1$. Then for all $k \geq 0$ $z^k y \in R$ and $z^k y /x = z^k w$. Hence, R/x is infinite.

6. $R = \gamma(R) \rightarrow$ for all $x \in \{0,1\}^*$ $R/x = \gamma(R/x)$. Let $y \in R/x$, then $yx \in R$. But $R = \gamma(R)$ implies for all srings z such that $|z| = |yx|$, $z \in R$. Hence, for all strings z' such that $|z'| = |y|$, $z'x \in R$. Hence, for all strings z' such that $|z'| = |y|$, $z' \in R/x$.

7, 8, and 9 are proved similarly. Note that 9 follows immediately from 7 since $\{ R \mid R$ is a regular set over $\{0,1\}$ and $R = \text{Subword}(R) \} \subseteq \{ R \mid R$ is a regular set over $\{0,1\}$ and $R = \text{Init}(R)\}$. We prove only 7.

$R = \text{Init}(R)$ implies for all $x \in \{0,1\}^*$ $x\backslash R = \text{Init}(x\backslash R)$. This follows since $y \in x\backslash R$ implies $x \cdot \text{Init}(y) \subset \text{Init}(R)$. Thus, $\text{Init}(y) \subseteq x\backslash\text{Init}(R) = x\backslash R$.

10. R is cofinite implies for all $x \in \{0,1\}^*$

$x\backslash R$ is cofinite. Since all but a finite number of strings over $\{0,1\}$ are elements of R, all but a finite number of strings over $\{0,1\}$ beginning with x are elements of R.

11, 12, 13, 14, 15, 16, and 17 all follow from 17, since a regular set satisfies 11 through 16 only if it satisfies 17. McNaughton and Papert[71] show that the star-free, noncounting and group-free events are the same.

By definition a regular set R is a noncounting event over $\{0,1\}$ if for some $n \geq 1$ and for all words U, V, W over $\{0,1\}$, $UV^{n+x}W \in R$ iff $UV^{n}W \in R$, for all positive integers x. R is a noncounting event $\rightarrow$ for all $x \in \{0,1\}^*$ $x\backslash R$ is a noncounting event. R is a noncounting event implies $\exists\, n \geq 1$ such that for all U, V, W $\in \{0,1\}^*$ and positive integers y, $UV^{n+y}W \in R$ iff $UV^{n}W \in R$. Hence $\exists n \geq 1$ such that for all U, V, W $\in \{0,1\}^*$ and positive integers y, $x \cdot UV^{n+y}W \in R$ iff $x \cdot UV^{n}W \in R$. (An alternative proof of 17 may be found in Hunt[73a].)

18 follows since quotient with single string does not raise the restricted star height of a regular set; and there are regular sets of arbitrary restricted star height.

19. R is accepted by some strongly connected deterministic finite automaton implies for all $x \in \{0,1\}^*$ $x\backslash R$ is empty or infinite. Let $M = (K,\{0,1\},\delta,q_0,F)$ be a strongly connected deterministic finite automaton which accepts R. Then $x\backslash L \neq \emptyset \rightarrow \exists y$ in $\{0,1\}^*$ such that $\delta(q_0,xy) = q_f \in F$. But M strongly connected implies $\exists z \in \{0,1\}^+$ such that $\delta(q_f,z) = q_0$. Hence, for all $k \geq 0$ $xy(zxy)^k \in R$. Hence, $x\backslash R$ is infinite.

(20) results from the following two facts:

Fact 1. If for all $x \in \{0,1\}^*$ $L/x \neq [\ L/x\ ]^{rev}$, then there is no regular set L_0 and no string $y \in \{0,1\}^*$ such that $L_0 = [\ L_0\ ]^{rev}$ and $y\backslash L_0 = L$.

Fact 2. For all x in $\{0,1\}^*$ $1(0+1)^*/x \neq [\ 1(0+1)^*/x\ ]^{rev}$.

We prove Facts 1 and 2. Suppose for all x in $\{0,1\}^*$, $L/x \neq [\ L/x\ ]^{rev}$ but $\exists$ a regular set L_0 and a string y such that $L_0 = [\ L_0\ ]^{rev}$ and $y\backslash L_0 = L$. But $(y\backslash L_0)/y^{rev} = [\ (y\backslash L_0)/y^{rev}]^{rev}$. Hence $L/y^{rev} = [\ L/y^{rev}\ ]^{rev}$. Finally for all $x, 1(0+1)^*/x \subseteq \{\lambda\} \cup 1(0+1)^*$. But for all x in $\{0,1\}^*$ $10 \in 1(0+1)^*/x$. Hence, $1(0+1)^* \notin P_L$.

(The fact that quotient with single string doesnot raise restricted star height was pointed out to the author by J. Brzozowski. Fact 2 in 20 resulted from discussions with P. M. Lewis 11, D.J. Rosenkrantz, and R.E. Stearns.)

<u>Definition 5.2.2</u>: A dot-free $(\cup, \cdot, *, \sim)$ regular expression over $\{0,1\}$ is a $(\cup, \cdot, *, \sim)$ regular expression over $\{0,1\}$ with no occurrence of "$\cdot$" . Let DF be the set of regular sets that can be represented by a dot-free $(\cup, \cdot, *, \sim)$ regular expression over $\{0,1\}$.

<u>Lemma 5.2.3</u>: $R \in DF \rightarrow R = [R]^{rev}$.

<u>Proof</u>: Any $R \in DF$ can be built up from the finite sets $\emptyset$, $\{\lambda\}$, $\{0\}$, and $\{1\}$ by finitely many applications of $\cup$, $\sim$, and $*$. If $R \in \{ \emptyset, \{\lambda\}, \{0\}, \{1\} \}$ then $R = [R]^{rev}$. Let $A = [A]^{rev}$ and $B = [B]^{rev}$. If $C = A \cup B$, A^*, or $\sim A$, then $C = [C]^{rev}$.

<u>Corollary 5.2.4</u>: $L = \{ R \mid R$ is a $(\cup, \cdot, *, \sim)$ regular expression over $\{0,1\}$ and $L(R)) \in DF \}$ is not elementary recursive.

<u>Proof</u>: Immediate from 20 of 5.2.1 and 5.2.3.

t <u>Theorem 5.2.5</u>: The following satisfy the conditions of 5.1.3:

(1) $R = \emptyset$;

(2) R is finite;

(3) R is of restricted star height 0;

(4) 2, 4-9, 11-17, 19 and 20 of 5.2.1; and

(5) R is bounded, i.e., $\exists x_1, x_2, \ldots, x_n \in \{0,1\}^+$ such that $R \subseteq x_1^* \ldots x_n^*$.

Proof: The proofs of 1, 2, and 3 are obvious. 4 follows since $\emptyset$ satisfies predicates 4-9, 11-17, 19 and 20 of 5.2.1. 5 follows since R bounded implies both $0\backslash R$ and $1\backslash R$ are bounded. By definition R is bounded if there exist nonempty words $x_1, \ldots, x_n$ such that $R \subseteq x_1^* \ldots x_n^*$. But $0\backslash R \subseteq (0\backslash x_1^*)\, x_1^*\, x_2^* \ldots x_n^* \cup (0\backslash x_2^*) \ldots x_n^* \cup \ldots \cup (0\backslash x_n^*)\, x_n^*$. Hence $R \subseteq (0\backslash x_1^*)\, x_1^*\, (0\backslash x_2^*)\, x_2^* \ldots (0\backslash x_n^*)\, x_n^*$.

Informally as we briefly mentioned in Chapter 1, a complexity core is a structure such that all problems containing it are hard. The structure of 5.1.2 and 5.1.3 is a natural candidate for satisfying this intuitive definition.

5.3 Some Related Hard Problems

In this section we discuss several poblems related to those of Section 5.2. First, the construction in the proof of 5.1.2 preserves the restricted star height of L_0 provided L_0's restricted star height ≥ 1. Thus, if there exists a language $L_0 \in$ regular sets over $\{0,1\}$ - P_L of restricted star height 1, we may replace (1) of Theorem 5.1.2 by

$\{ R \mid R$ is a $(\cup, \cdot, *)$ over $\{0,1\}$ of restricted star height L and $P(L(R))$ is false $\} \underset{\text{ptime}}{\geq}$ PTAPE and is an element of DCSL only if DCSL = NDCSL.

Secondly, we present a deterministic finite automaton analogue of Theorem 5.1.2.

Definition 5.3.1: Let P be a predicate on the regular sets over $\{0,1\}$. P is preserved by quotient with single string on the left , if P(L) true implies that both $P(0\backslash L)$ and $P(1\backslash L)$ are true.

Let φ be a set of deterministic finite automata encoded in some finite alphabet, where the input alphabet of each automaton is $\{0,1\}$. Thus $\varphi = \{ M_1, \ldots , M_n, \ldots\}$, where $M_i = (Q_i, \{0,1\}, \delta_i, q_0^{\,i}, F_i)$, $F_i \subset Q_i$, $q_0^{\,i} \in Q_i$, and $\delta_i : Q_i \times \{0,1\} \to Q_i$. Let $\Lambda = \{ L_i \mid L_i = L(M_i)$ where $M_i \in \varphi\}$. Let M_i be some arbitrary element of φ. Then given M_i we show how to find two deterministic finite automata $M_i(0)$

and $M_i(1)$ such that $L(M_i(0)) = 0\backslash L(M_i)$ and $L(M_i(1)) = 1\backslash L(M_i)$.

There are two distinct cases to consider. If $\delta_i(q_0^{\,i},0) = q_0^{\,i}$ then $M_i(0) = M_i$, otherwise $M_i(0) = (Q_i, \{0,1\}, \delta_i, \delta_i(q_0^{\,i},0), F_i)$, where $M_i = (Q_i, \{0,1\}, \delta_i, q_0^{\,i}, F_i)$. $M_i(1)$ is defined analogously with 1 replacing 0. Thus if $M_i \in \varphi$ implies both $M_i(0)$ and $M_i(1)$ are elements of φ, then the predicate "$L_i \in \Lambda$" , i.e., $\exists M_j \in \varphi$ such that $L_i = L(M_j)$, is preserved by quotient with single string on the left.

<u>Theorem 5.3.2</u>: Let φ be a set of deterministic finite automata as discussed above such that

(a) There exists $M_j \in \varphi$ such that $L(M_j) = \{0,1\}^*$;

(b) $M_i \in \varphi$ implies that both $M_i(0)$ and $M_i(1)$ are elements of φ; and

(c) $\Lambda = \{ L \mid \exists M_i \in \varphi \text{ and } L = L(M_i) \} \nsubseteq$ regular sets over $\{0,1\}$. Then the predicate "$R \in \Lambda$" satisfies the conditions of Theorem 5.1.2.

We note that similar results hold for quotient on the right with single string. Also the set of strongly connected finite automata satisfies the conditions of 5.3.2.

Next we prove that structural equivalence for cfgs is as hard to decide as equivalence for regular expressions.

<u>Definition 5.3.3</u>: Two cfgs G_1 and G_2 are said to be

<u>structurally equivalent</u> if they generate the same strings and their parse trees are the same except for labels.

<u>Proposition 5.3.4</u>: { (G_1, G_2) | G_1 and G_2 are representations over some fixed finite alphabet of cfgs Γ_1 and Γ_2 , respectively, and Γ_1 and Γ_2 are not structurally equivalent } $\underset{\text{ptime}}{\geq}$ PTAPE.

<u>Proof</u>: Given a regular expression R, we can convert it into a regular grammar G such that L(G) = L(R). We can do this in deterministic polynomial time in $|R|$. But G is a cfg and for regular grammars equivalence and structural equivalence are the same.

Finally, we note that L = { (R_1, R_2) | R_1 and R_2 are regular expressions over {0,1} and $L(R_1) \neq L(R_2)$ } $\in$ NDCSL. To see this note that to verify that $L(R_1) \neq L(R_2)$ nondeterministicly, we need only guess a string x, one character at a time, and verify that $x \in [\, L(R_1) \cap \overline{L(R_2)}\,] \cup [\, \overline{L(R_1)} \cap L(R_2)\,]$. But to do this we need keep only 1 character of x at a time on our machine tape.

<u>Corollary 5.3.5</u>: Let Reg be the set of ($\cup$,$\cdot$,*) regular expressions over {0,1}. Then the following are Savitch CSLs:

(1) { (R_1, R_2) | R_1 and R_2 are elements of Reg

and $L(R_1) \neq L(R_2)$ };

(2) $\{ R \mid R \in \text{Reg and } L(R) \neq \{0,1\}^* \}$;

(3) $\{ R \mid R \in \text{Reg and } L(R) \neq [L(R)]^{rev} \}$;

(4) $\{ R \mid R \in \text{Reg and } L(R) \neq L(R)^* \}$;

(5) $\{ R \mid R \in \text{Reg and } L(R) \neq \gamma[L(R)] \}$;

(6) For all $k \geq 1$, $\{ R \mid R \in$ Reg and $L(R)$ is not k-definite or k-reverse definite }; and

(7) $\{ R \mid R \in$ Reg and $L(R)$ is coinfinite }.

<u>Proof</u>: 1, 2, 3, 4, and 5 follow from the observation made above that the set $\{ (R_1, R_2) \mid R_1, R_2 \in \text{Reg and } L(R_1) \neq L(R_2) \} \in$ NDCSL. 6 follows from the fact that for any fixed k there are only a finite number of k-definite events. 7 follows since $\overline{L(R)}$ is infinite iff $\exists z \in \overline{L(R)}$ such that $|z| \geq 2^{2|R|}$.

Chapter 6 CONCLUSION

There are five main results in this thesis. First, there are natural and powerful complexity cores. These cores can characterize the complexity of classes of problems not just individual ones. They can be found. Second, the interaction between formal language theory and computational complexity as in Chapters 3 and 5 can provide insights in both areas. Third, there are natural problems which are provably hard. We can no longer say that any reasonable language is context-sensitive. Fourth, the regular expressions have far more descriptive power than has been thought. Finally, a small but nontrivial area of computational complexity, the predicates on regular sets, has been solved.

6.1 Open Problems

There are several results bearing on the subject of this thesis which we have not included due to space and time considerations:

(1) Both the emptiness and equivalence problems for 2dfa are decidable by polynomial space bounded Tms;

(2) Empty($\cup, \cdot, *, \cap$) is an element of PTAPE;

(3) Equiv($\cup, \cdot, *, \cap$) requires space $O(c^{\sqrt{n}})$ for some $c>1$;

(4) For all rational $r\geq 1$, Ndtape(n^r) has a hardest tape and time language(See Hartmanis and Hunt[73].)

Finally, we mention several open problems suggested by the results of this thesis.

(1) Is structural equivalence for cfgs decidable in polynomial space?

(2) Are there natural hardest tape languages for Ndtape(n^2), Ndtape(n^3), etc.?

(3) Is the equivalence problem for 2ndfsa decidable in polynomial space?

(4) Is PTIME equal to NPTIME?

(5) Is PTIME or NPTIME equal to PTAPE?

(6) Is DCSL equal to NDCSL?

(7) Find other P-complete operations. In particular note that the language L_0 of Theorem 2.2.5 is recognizable in Dtape(Log(n)). Thus Dtape(Log(n)) is an AFL implies that

PTIME equals NPTIME.

(8) Find sufficient conditions on predicates on graphs such that any predicate satisfying them can be decided in PTIME.

(9) Find sufficient conditions on predicates on graphs such that any predicate satisfying them is P-complete.

(10) Find sufficient conditions on predicates on cfgs (e.g., "G_i is LR(1)", "G_i is LL(1)", etc.) such that any predicate satisfying them is decidable or is undecidable.

BIBLIOGRAPHY

Aho, A.V., J.E. Hopcroft, and J.D. Ullman, [1973]. Preliminary notes for a text on the theory of algorithms.

Book, R.V., [1972]. "On languages accepted in polynomial time," Harvard University technical report.

———, [1973]. "Comparing complexity classes," Harvard University technical report.

Book, R.V., and S.A. Greibach, [1970]. "Quasi-realtime languages," Mathematical Systems Theory, 4, 97-111.

Brzozowski, J.A., and R. Cohen, [1969]. "On the decomposition of regular events," JACM, 18:1, 4-18.

Cook, S.A., [1971]. "The complexity of theorem-proving procedures," Proceedings of Third Annual ACM Symposium on Theory of Computing, Shaker Heights, Ohio.

Greibach, S.A., [1968]. "A note on undecidable properties of formal languages." Mathematical Systems Theory, 2:1,

Hartmanis, J., and H.B. Hunt111, [1973]. "The lba problem and its importance in the theory of computing," Amer. Math. Soc. Symp. on the Complexity of Real Computation, New York.

Hartmanis, J., and R.E. Stearns, [1966]. Algebraic Structure Theory of Sequential Machines, Prentice-Hall, Englewood Cliffs, N. J.

Hopcroft, J.E., and J.D. Ullman, [1969]. Formal Languages and Their Relation to Automata, Addison-Wesley, Reading, Mass.

Hunt, H.B. 111, [1973a] . "On the time and tape complexity oflanguages, 1," Cornell University Department of Computer Science technical report no. 73-156.

———, [1973b]. "The equivalence problem for regular expressions with intersection is not polynomial in tape," Cornell University Department of Computer Science technical report no. 73-161.

Karp, R.M., [1972]. "Reducibility among combinatorial problems," in Complexity of Computer Computations, R.E. Miller and J.W. Thatcher, ed., Plenum Press, New York, 86-104.

McNaughton, R., [1967]. "Parenthesis grammers," JACM, 14:3, 490-500.

———, [1969]. "The loop complexity of regular events," Information Sciences, 1, 305-328.

McNaughton, R., and S. Papert, [1971]. Counter-Free Automata, MIT Press, Cambridge, Mass.

Meyer, A.R., and L.J. Stockmeyer, [1972]. "The equivalence problem for regular expressions with squaring requires exponential space," IEEE Conference Record of 13th Annual Symposium on Switching and Automata Theory, College Park, Maryland, 125-129.

Paull, M., and S.H. Unger, [1968]. "Structural equivalence of context-free grammars," JCSS, 2:1, 427-463.

Paz, A., and B. Peleg, [1965]. "Ultimate-definite and symmetric-definite events and automata," JACM, 12:3, 399-410.

Savitch, W., [1970]. "Relationships between nondeterministic and deterministic tape complexities," JCSS, 4:2, 177-192.