Word Problems Requiring Exponential Time[†]: Preliminary Report

L.J. Stockmeyer and A.R. Meyer
Massachusetts Institute of Technology

1. INTRODUCTION

The equivalence problem for Kleene's regular expressions has several effective solutions, all of which are computationally inefficient. In [1], we showed that this inefficiency is an inherent property of the problem by showing that the problem of membership in any arbitrary context-sensitive language was easily reducible to the equivalence problem for regular expressions. We also showed that with a squaring abbreviation (writing $(E)^2$ for $E \cdot E$) the equivalence problem for expressions required computing space exponential in the size of the expressions.

In this paper we consider a number of similar decidable word problems from automata theory and logic whose inherent computational complexity can be precisely characterized in terms of time or space requirements on deterministic or nondeterministic Turing machines. The definitions of the word problems and a table summarizing their complexity appears in the next section. More detailed comments and an outline of some of the proofs follows in the remaining sections. Complete proofs will appear in the forthcoming papers [9, 10, 13]. In the final section we describe some open problems.

2. WORD PROBLEMS AND REDUCIBILITIES

We consider word problems involving (1) regular-like expressions for subsets of Σ^*, where Σ is a finite set of letters, (2) similar expressions for subsets of the nonnegative integers N, and (3) certain closed formulas related to the predicate calculus.

Regular-like expressions over Σ are well-formed parenthesized expressions involving constants $\sigma \in \Sigma$, and the empty string λ, binary operations $\cdot$ (concatenation), $\cup$ (union), and unary operations $*$(Kleene star), $\neg$(complement relative to Σ^*), and ^{2}(squaring). For any regular-like expression E, the set $L(E) \subset \Sigma^*$ described by E is defined inductively in the obvious way, e.g.,

$$L(\sigma) = \{\sigma\} \text{ for } \sigma \in \Sigma,$$
$$L((E_1 \cdot E_2)) = L(E_1) \cdot L(E_2),$$
$$L((E)^2) = L(E) \cdot L(E),$$
$$L(\neg(E)) = \Sigma^* - L(E), \text{ etc.}$$

For any set Σ of letters and set of operations $\varphi \subset \{\cdot, \cup, *, \neg, ^2\}$, define

$\underline{\text{MEMBER}}(\Sigma, \varphi) = \{(x, E) \mid x \in \Sigma^*$, E is a regular-like expression over Σ containing only operations in φ, and $x \in L(E)\}$,

$\underline{\text{INEQ}}(\Sigma, \varphi) = \{(E_1, E_2) \mid E_1$ and E_2 are regular-like expressions over Σ containing only operations in φ and $L(E_1) \neq L(E_2)\}$.

Integer expressions are well-formed parenthesized expressions involving nonegative integer constants written in radix notation (say base two), binary operations + (addition), $\cup$ (union), and the unary operation $\neg$ (complement relative to N). For any integer expression E, the set $L(E) \subset N$ described by E is defined inductively as follows:

$$L(m) = \{m\} \text{ for } m \in N,$$
$$L((E_1 \cup E_2)) = L(E_1) \cup L(E_2),$$
$$L((E_1 + E_2)) = \{m+n \mid m \in L(E_1) \text{ and } n \in L(E_2)\},$$
$$L(\neg(E)) = N - L(E).$$

For any set of operations $\varphi \subset \{+, \cup, \neg\}$, define

$\underline{\text{N-MEMBER}}(\varphi) = \{(x, E) \mid x$ is the binary representation of some integer n, E is an integer expression containing only operations in φ, and $n \in L(E).\}$,

$\underline{\text{N-INEQ}}(\varphi) = \{(E_1, E_2) \mid E_1$ and E_2 are integer expressions containing only operations in φ and $L(E_1) \neq L(E_2)\}$.

[†]Work reported here was supported in part by Project MAC, an MIT research program sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract Number N00014-70-A-0362-0006 and the National Science Foundation under contract number GH00-4327. Reproduction in whole or in part is permitted for any purpose of the United States Government.

Finally, we consider Boolean expressions involving doubly subscripted Boolean variables $x_{i,j}$ with $i,j \geq 1$ written in base two notation, and the usual Boolean operations, $\wedge$ (and), $\vee$ (or), $\neg$ (negation), $\equiv$ (equivalence), $\supset$ (implication), and Boolean constants 0, 1. In describing Boolean expressions we write $\bar{X}_i$ for the sequence of variables $x_{i,1}, x_{i,2}, x_{i,3}, \ldots$; $\exists\bar{X}_i$ for $\exists x_{i,1}\exists x_{i,2}\exists x_{i,3}\ldots$; etc. These abbreviations do not appear in the Boolean expressions themselves. We use the notation $A(\bar{X}_1,\ldots,\bar{X}_k)$ to indicate a Boolean expression containing no variable $x_{i,j}$ such that $i > k$.

Define for $k \geq 1$,

$$B_k = \{A(\bar{X}_1,\ldots,\bar{X}_k) \mid \exists\bar{X}_1\,\forall\bar{X}_2\,\exists\bar{X}_3\ldots Q_k\bar{X}_k[A(\bar{X}_1,\ldots,\bar{X}_k) = 1]\}$$

where $Q_k = \exists$ if k is odd and $Q_k = \forall$ if k is even. For example, B_1 corresponds to the satisfiable formulas of propositional calculus. Define

$$B_\omega = \bigcup_{k=1}^{\infty} B_k.$$

Let LEQ denote the set of valid sentences in the first order theory of equality.

We shall classify the complexity of sets of words in terms of two binary relations $\leq_{\log}$ and $\leq_{\text{log-lin}}$, called log-space reducibility, and log-linear reducibility, respectively.

Let Σ, Δ be finite sets of letters and $f: \Sigma^* \to \Delta^*$ a function. We say that f is log-space computable iff there is a deterministic Turing machine with a two-way read-only input tape, a one-way output tape and one two-way read-write working tape, which started with any $x \in \Sigma^*$ on its input tape will halt having written $f(x)$ on its output tape and having visited at most $\log_2|x|$ tape squares on its work tape, where $|x|$ is the length of x.

Let $A \subset \Sigma^*$, $B \subset \Delta^*$ be sets of words.

Definition. $A \leq_{\log} B$ iff there is a log-space computable function f such that $(\forall x \in \Sigma^*)[x \in A \Leftrightarrow f(x) \in B]$. If in addition there is a constant $c > 0$ such that $|f(x)| \leq c\cdot|x|$ for all $x \in \Sigma^*$, then $A \leq_{\text{log-lin}} B$.

We assume the reader is familiar with the notion of nondeterministic Turing machines [cf. 2]. Briefly, the time required for a nondeterministic machine to accept an input x is the length (number of steps) in the shortest accepting computation; a set A of inputs words is said to be accepted in time $\leq t$ by a nondeterministic Turing machine $\mathfrak{M}$, where $t: N \to N$, iff for all input words x, (1) $x \in A \Leftrightarrow$ there is an accepting computation of $\mathfrak{M}$ started on input x, and (2) $x \in A \Rightarrow$ there is an accepting computation of $\mathfrak{M}$ on x of length $\leq t(|x|)$. Similar definitions apply for space.

The main properties of these reducibilities are stated in the next two lemmas.

Lemma 2.1. $\leq_{\log}$ and $\leq_{\text{log-lin}}$ are transitive relations.

Lemma 2.2. If $A \leq_{\log} B$ ($A \leq_{\text{log-lin}} B$) and there is a $\begin{Bmatrix}\text{deterministic}\\ \text{nondeterministic}\end{Bmatrix}$ Turing machine accepting B in time $\leq t$ and space $\leq s$, then there is a polynomial p (and constant $c > 0$) such that A is accepted by a $\begin{Bmatrix}\text{deterministic}\\ \text{nondeterministic}\end{Bmatrix}$ Turing machine in time $\leq t'$ where

$$t'(n) = p(n)\cdot\max\{t(j) \mid j \leq p(n)\}$$

$$(t'(n) = p(n)\cdot\max\{t(j) \mid j \leq c\cdot n\})$$

and space $\leq s'$ where

$$s'(n) = \log_2 n + \max\{s(j) \mid j \leq p(n)\}$$

$$(s'(n) = \log_2 n + \max\{s(j) \mid j \leq c\cdot n\}).$$

To outline the proof of Lemma 2.1, suppose $A \subset \Sigma^*$, $B \subset \Delta^*$, $C \subset \Gamma^*$ and $A \leq_{\text{log-lin}} B$ via $f: \Sigma^* \to \Delta^*$ and $B \leq_{\text{log-lin}} C$ via $g: \Delta^* \to \Gamma^*$. Clearly $x \in A \Leftrightarrow g(f(x)) \in \Gamma^*$, so one need only show that $g \circ f$ is log-space computable. The difficulty is that the obvious Turing machine which given x on its input tape prints $g(f(x))$ on its output tape must write $f(x)$ on its work tape and so may use more than $\log_2|x|$ tape squares. However, instead of writing $f(x)$ on its work-tape, a machine with input x can simulate the computation of g at argument $f(x)$ by recording on its work tape an instantaneous description of the computation of g and the position j in $f(x)$ which the input head would occupy if the input were actually $f(x)$. Since $j \leq |f(x)| \leq c\cdot|x|$, only $\log_2|x| + \log_2 c$ extra squares on the work tape are required to record j. To simulate another step in the computation of $g(f(x))$, the machine with input x computes the j^{th} digit of $f(x)$ using only $\log_2|x|$ work tape squares, which is possible since f can be computed in log space. Hence, from x on the input tape, $f(g(x))$ can be printed on the output tape using proportional to $\log_2|x|$ work tape squares. The computation can then actually be done using only $\log_2|x|$ work-tape squares using a larger set of symbols on the work-tape (cf. [2]).

Note that if f is log-space computable, then f is necessarily polynomial time computable. So in particular, $|f(x)| \leq p(|x|)$ for some polynomial p. With this observation, the preceding argument may also be used to show that $\leq_{\log}$ is transitive.

The proof of Lemma 2.2 is similar, and is omitted.

We remark that $A \leq_{\log} B \Rightarrow A \; \alpha \; B$ where α is polynomial time reducibility defined by Karp [4]. We believe that all of the particular polynomial time reductions described in [4] are actually log space reducibilities, although it would be surprising if log space and polynomial time were the same in general.

Let $\mathcal{F}$ be a family of sets of words, and $\leq$ a transitive relation on all sets of words. We say that $\mathcal{F} \leq C$ iff $B \leq C$ for all $B \in \mathcal{F}$. C is $\leq$-complete in $\mathcal{F}$ iff $\mathcal{F} \leq C$ and $C \in \mathcal{F}$. For example, let $\mathcal{E}$ be the family of sets recognizable in nondeterministic exponential time, i.e., $A \in \mathcal{E}$ iff there is a constant $a > 1$ and a nondeterministic Turing machine which accepts A in time $\leq a^n$; let C be a set such that $\mathcal{E} \leq_{\text{log-lin}} C$. Standard diagonal arguments imply the existence of a set $A \in \mathcal{E}$ such that any deterministic Turing machine accepting A requires time $\geq 2^{|x|}$ for infinitely many inputs x. The contrapositive of Lemma 2 immediately implies that there is a constant $b > 1$ such that any deterministic Turing machine accepting C requires time $\geq b^{|x|}$ for infinitely many inputs $|x|$. In fact, Seiferas [5], extending methods of Ibarra [6] and Cook [7], has shown that for $1 \leq a_1 < a_2$, there exists a set B recognizable in nondeterministic time $\leq a_2^n$ but not in nondeterministic time $\leq a_1^n$, so that C requires time $\geq b^n$ for some constant $b > 1$ and infinitely many n even on nondeterministic machines.

In general, from the fact that $\mathcal{F}$ is reducible to C by a computationally efficient reducibility, one can immediately deduce that the computational complexity of C must be approximately as large as the computational complexity of any member of $\mathcal{F}$. Additional properties about the complexity of C, for example that any machine recognizing C can be "sped-up" effectively on infinitely many inputs in the sense of Blum [8], can also be proved in most cases. A full treatment of these properties of efficient reducibilities will appear in subsequent papers.

The following table summarizes our main results on the word problems defined above. For each set C and family $\mathcal{F}$ in the table, $\mathcal{F}$ is reducible to C by the indicated reducibility, and an upper bound on the complexity of C appears in the final column.

Complexity of Word Problems

C	$\mathcal{F}$ reducible to C	$\mathcal{F}$	Upper Bound on C
INEQ({0,1}, {U,•,¬})	log-lin	space $2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}\}\log_2 n$	space $2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}\}n$
INEQ({0,1}, {U,•,^{2},*})	log-lin	exponential space	complete
INEQ({0,1}, {U,•,^{2}})	log-lin	nondeterministic exponential time	complete
INEQ({0,1}, {U,•,*})	log-lin	nondeterministic linear space	complete
INEQ({0,1},{U,•})	log	$\mathcal{NP}$	complete
MEMBER({0,1}, {U,•,^{2},*,¬})	-	—	$\mathcal{P}$
B_ω	log	polynomial space	complete
1EQ	log	polynomial space	complete
B_k	log	Σ_k^p	complete
N-INEQ({U,+,¬})	log	polynomial space	complete
N-MEMBER({U,+,¬})	log	polynomial space	complete
N-INEQ({U,+})	log	Σ_2^p	complete
N-MEMBER({U,+})	log	$\mathcal{NP}$	complete
INEQ({0}, {U,•,^{2},¬})	log	polynomial space	complete
INEQ({0}, {U,•,*})	log	$\mathcal{NP}$	complete
INEQ({0}, {U,•,¬})	-	——	$\mathcal{P}$

Following Cook [3] and Karp [4], we let $\mathcal{P}$ ($\mathcal{NP}$) denote the family of sets of words A such that A is accepted by a deterministic (nondeterministic) Turing machine in time bounded by a polynomial. The classes Σ_k^p for $k \geq 0$ are defined in [1] and discussed in section 4 below.

3. REGULAR-LIKE EXPRESSIONS OVER {0,1}

The proofs that INEQ({0,1}, $\{\cup,\cdot,^2,^*\}$) is log-linear complete in exponential space, and that INEQ({0,1}, $\{\cup,\cdot,^*\}$) is log-linear complete in nondeterministic linear space have appeared in [1]. Complete proofs of all the results concerning regular-like expressions over {0,1} (i.e. the first six table entries) will appear in [9] and [10]. Here we shall indicate the proof of only the following

Theorem 3.1. The family of sets of words recognizable in nondeterministic exponential time is log-linear reducible to INEQ({0,1}, $\{\cup,\cdot,^2\}$).

Note that regular-like expressions involving only the operations $\cup,\cdot,^2$ can define only finite sets of words, so that the decidability of INEQ({0, 1}, $\{\cup,\cdot,^2\}$) is trivial. To describe large finite sets, one can use the identities $A^{n+1} = A^n \cdot A$ and $A^{2n} = (A^n)^2$ to write a regular-like expression F such that $L(F) = L(E)^n$, $|F|$ is bounded by a constant times $|E| \cdot \log_2 n$, and F involves only $\cup,\cdot,^2$ and the operations in E for any regular-like expression E and $n \geq 0$.

Let $\mathfrak{M}$ be a nondeterministic Turing machine which accepts a set $A \subset \{0,1\}^*$ in time $\leq 2^n$. It will be sufficient to show that $A \leq_{\text{log-lin}}$ INEQ(Σ, $\{\cup,\cdot,^2\}$), where Σ is a finite set of letters including symbols for the states and symbols of $\mathfrak{M}$, Σ contains a symbol β denoting a blank tape square, and Σ contains a symbol # which serves as an end marker.

As in [1], let $\text{Comp}_{\mathfrak{M}}(x)$ be the set of accepting computations of $\mathfrak{M}$ on x, that is, $\text{Comp}_{\mathfrak{M}}(x)$ consists of words of the form $\#I_1\#I_2\#\ldots\#I_k\#$ where I_1 is the initial instantaneous description (i.d.) of $\mathfrak{M}$ on input x, I_k contains an accepting state, and I_{j+1} is a possible next i.d. following in one step from I_j for $1 \leq j < k \leq 2^{|x|}$. No i.d. of $\mathfrak{M}$ need be longer than $2 \cdot 2^n + 1$, so there is no loss of generality in adding the technical requirement that each of the i.d.'s be surrounded by blanks to be of exactly length $2 \cdot 2^n + 1$.

Thus, the initial i.d. for x would be $\beta^{2^n} \cdot q_0 \cdot x \cdot \beta^{2^n - n}$ where q_0 is the start state of $\mathfrak{M}$ and $n = |x|$.

Note that no word in $\text{Comp}_{\mathfrak{M}}(x)$ is longer than $a(n) = 4^{n+2}$. We shall show how to construct from x a regular-like expression E_x over Σ involving only $\cup,\cdot,^2$ such that $L(E_x) = (\Sigma \cup \lambda)^{b(n)} - \text{Comp}_{\mathfrak{M}}(x)$ for a certain integer $b(n) \geq a(n)$. Hence,

$$x \in A \Leftrightarrow \text{Comp}_{\mathfrak{M}}(x) \neq \emptyset \Leftrightarrow L(E_x) \neq (\Sigma \cup \lambda)^{b(n)}.$$

Moreover, it will be apparent from the construction that the function mapping x to the pair (E_x, expression for $(\Sigma \cup \lambda)^{b(n)}$) is log-space computable and that the length of these expressions is proportional to n. Hence $A \leq_{\text{log-lin}}$ INEQ(Σ, $\{\cup,\cdot,^2\}$).

To construct E_x we wish to describe a finite set of words containing all words in Σ^* of length $\leq a(n)$ except those in $\text{Comp}_{\mathfrak{M}}(x)$. Let $\tilde{\sigma} = \Sigma - \{\sigma\}$ for any $\sigma \in \Sigma$ and say $x = x_1 x_2 \ldots x_n$ where $x_i \in \{0,1\}$.

Now words may fail to be in $\text{Comp}_{\mathfrak{M}}(x)$ because they "start wrong". These words can be described as follows:

"too short": $(\Sigma \cup \lambda)^{2 \cdot 2^n + 2}$

"doesn't start with #": $\tilde{\#} \cdot (\Sigma \cup \lambda)^{a(n)}$,

"doesn't start with enough blanks":
$\#\cdot(\beta \cup \lambda)^{2^n - 1} \cdot \tilde{\beta} \cdot (\Sigma \cup \lambda)^{a(n)}$,

"doesn't start with $\#\beta^{2^n} q_0 x$":

$$\#\cdot\beta^{2^n}\cdot(\tilde{q}_0 \cup q_0\cdot(\tilde{x}_1 \cup x_1\cdot(\tilde{x}_2 \cup x_2\cdot(\ldots(\tilde{x}_{n-1} \cup x_{n-1}\cdot\tilde{x}_n)\ldots))))\cdot(\Sigma \cup \lambda)^{a(n)},$$

"not enough blanks following x":

$$\Sigma^{2^n+n+2}\cdot(\beta \cup \lambda)^{2^n-n-1}\cdot\tilde{\beta}\cdot(\Sigma \cup \lambda)^{a(n)},$$

"missing the second #": $\Sigma^{2 \cdot 2^n + 2} \cdot \tilde{\#} \cdot (\Sigma \cup \lambda)^{a(n)}$.

A word may also fail to be in $\text{Comp}_{\mathfrak{M}}(x)$ because it "ends wrong". These words can be described as follows:

"doesn't halt in the accepting state q_a":

$$(\Sigma \cup \lambda)^{a(n)} \cdot \# \cdot ((\Sigma \cup \lambda) - \{\#, q_a\})^{2 \cdot 2^n + 1} \cdot \#,$$

"doesn't end with #": $(\Sigma \cup \lambda)^{a(n)} \cdot \tilde{\#}$.

Finally, a word may fail to be in $\text{Comp}_{\mathfrak{M}}(x)$ because the consecutive i.d.'s do not conform to the possible moves of $\mathfrak{M}$. In particular, the definition of $\mathfrak{M}$ determines for every three symbols σ_1, σ_2, $\sigma_3 \in \Sigma$ a set $N(\sigma_1\sigma_2\sigma_3)$ of words of length three such that if $\sigma_1\sigma_2\sigma_3$ appears in a word in $\text{Comp}_{\mathfrak{M}}(x)$,

then only words in $N(\sigma_1\sigma_2\sigma_3)$ may appear one i.d. length to the right of $\sigma_1\sigma_2\sigma_3$. The words not in $\mathrm{Comp}_{\mathfrak{M}}(x)$ because they "move wrong" can now be described as follows:

$$(\Sigma \cup \lambda)^{a(n)} \cdot U \cdot (\Sigma \cup \lambda)^{a(n)}$$

where

$$U = \bigcup_{\substack{\sigma_1,\sigma_2,\sigma_3 \\ \in \Sigma}} \sigma_1 \cdot \sigma_2 \cdot \sigma_3 \cdot \Sigma^{2 \cdot 2^n - 1} \cdot (\Sigma^3 - N(\sigma_1\sigma_2\sigma_3))$$

The union of the sets given by the expressions above contains all words over Σ of length $\leq a(n)$ except those in $\mathrm{Comp}_{\mathfrak{M}}(x)$. It also contains certain longer words, none of which however is longer than $b(n) = 2 \cdot a(n) + 2 \cdot 2^n + 5$. Thus, the words which are "too long" are

$$\Sigma^{a(n)+1} \cdot (\Sigma \cup \lambda)^{b(n)-a(n)-1}.$$

The regular-like expression E_x is thus simply the union of the regular-like expressions corresponding to the words which "start wrong", "end wrong", "move wrong", or are "too long".†

4. POLYNOMIAL TIME QUANTIFIERS

In [1] we defined an analogue to the arithmetic hierarchy in which $\mathcal{P}$ plays the role of the recursive sets, $\Sigma_1^{\mathcal{P}}$ was defined to be $\mathcal{NP}$ and $\Sigma_{k+1}^{\mathcal{P}}$ was defined as the family of sets of words accepted in nondeterministic polynomial time by Turing machines with oracles for sets in $\Sigma_k^{\mathcal{P}}$. The analogy to the arithmetic hierarchy is made more explicit in the next theorem, which is stated without proof.

Let $P(x_1,\ldots,x_k)$ be a predicate on words in Σ^* for some Σ. We say that P is polynomial time computable if $\{x_1\#x_2\#\ \ldots\ \#x_k \mid P(x_1,\ldots,x_k)\}$ is a set of words recognizable in deterministic polynomial time where # is a symbol not in Σ.

Theorem 4.1. For $k \geq 1$, a set of words A is in $\Sigma_k^{\mathcal{P}}$ iff there is a deterministic polynomial time computable predicate $P(x,y_1,y_2,\ldots,y_k)$ and a polynomial p such that

$$A = \{x \mid \exists y_1\, \forall y_2\, \exists y_3 \ldots Q_k y_k [P(x,y_1,\ldots,y_k)]\}$$

where the quantifiers range over $y_i \in \Sigma^*$ such that $|y_i| \leq p(|x|)$.

The following theorem was stated and proved in slightly weaker form in [1].

Theorem 4.2. For $k \geq 1$, the set B_k is $\leq_{\log}$-complete in $\Sigma_k^{\mathcal{P}}$.

In fact, the construction given in [11] for converting an arbitrary propositional formula to disjunctive form in polynomial time while preserving validity may be extended to show that the set of disjunctive (conjunctive) form formulas in B_k is log-space complete when k is even (odd).

It remains an open question whether the containment of $\Sigma_k^{\mathcal{P}}$ in $\Sigma_{k+1}^{\mathcal{P}}$ is proper. Nevertheless, we believe the "hierarchy" of $\Sigma_k^{\mathcal{P}}$ classes is technically useful for classifying the complexity of problems. In the next section we describe a word problem which is complete in $\Sigma_2^{\mathcal{P}}$.

The set $B_\omega = \bigcup_{k=1}^{\infty} B_k$ is the natural analogue to the ω-jump of the empty set in recursive function theory. The following theorem reveals a surprising relationship between the polynomial time hierarchy and polynomial space.

Theorem 4.3. B_ω is $\leq_{\log}$-complete in polynomial space.

Proof. To show that polynomial space is $\leq_{\log} B_\omega$, let $\mathfrak{M}$ be a Turing machine which accepts some set $L \subset \{0,1\}^*$ in space $\leq p(n)$ for some polynomial p. In the computation of $\mathfrak{M}$ on input x, no instantaneous description is longer than $1 + p(|x|)$. We choose an encoding of the states and symbols of $\mathfrak{M}$ into words in $\{0,1\}^*$, so that any instantaneous description of $\mathfrak{M}$ in its computation on input x will be a word $y \in \{0,1\}^*$ such that $|y| = q(|x|)$ for some polynomial q depending on $\mathfrak{M}$, p, and the encoding, but not depending on x.

As in [1], [3], one can construct a Boolean formula $A_{\mathfrak{M},n}(\bar{U},\bar{V})$ where $\bar{U} = u_1, u_2,\ldots,u_n$ and $\bar{V} = v_1,v_2,\ldots,v_n$ are sequences of Boolean variables and $n = q(|x|)$ such that

$A_{\mathfrak{M},n}(\bar{U},\bar{V}) \Leftrightarrow$ $u_1u_2\ldots u_n$ and $v_1v_2\ldots v_n$ are the encodings of i.d.'s of $\mathfrak{M}$ and $v_1v_2\ldots v_n$ follows from $u_1u_2\ldots u_n$ in at most one step of $\mathfrak{M}$.

Moreover, for fixed $\mathfrak{M}$, the length of $A_{\mathfrak{M},n}$ is bounded by a polynomial in n.

†Theorem 1 was stimulated by John Brzozowski's remark that our use of Kleene * in [1] was very restricted and might therefore be removable. We are also grateful to Harry Hunt who pointed out that the function N used in [1] should actually have been a mapping from words to sets of words as in the construction given here.

By quantifying over some of the variables, one can now construct formulas $A_{k,n}(\bar{U},\bar{V})$ for $k \geq 0$ such that

$$A_{k,n}(\bar{U},\bar{V}) \Leftrightarrow u_1u_2\ldots u_n \text{ and } v_1v_2\ldots v_n \text{ are the encodings of i.d.'s of } \mathfrak{M} \text{ and } v_1v_2\ldots v_n \text{ follows from } u_1u_2\ldots u_n \text{ in} \leq 2^k \text{ steps of } \mathfrak{M}.$$

Moreover, the length of $A_{k,n}(\bar{U},\bar{V})$ is bounded by k+1 times a polynomial in n. This follows by induction from the equivalence

$$A_{k+1,n}(\bar{U},\bar{V}) \Leftrightarrow (\exists\bar{W})(\forall\bar{Y})(\forall\bar{Z})[((\bar{Y}=\bar{Y} \wedge \bar{W} = \bar{Z}) \vee (\bar{W} = \bar{Y} \wedge \bar{V} = \bar{Z})) \Rightarrow A_{k,n}(\bar{Y},\bar{Z})],$$

where we have used $\bar{U} = \bar{Y}$ as an abbreviation for the formula $(u_1 \equiv y_1) \wedge (u_2 \equiv y_2) \wedge \ldots \wedge (u_n \equiv y_n)$. (We note the similarity of this construction to Savitch's proof that nondeterministic space n is contained in deterministic space n^2 [12].)

Now since $\mathfrak{M}$ requires space $\leq p$, there is a constant $c > 1$ such that $\mathfrak{M}$ accepts an input word x iff $\mathfrak{M}$ accepts x in time $\leq c^{p(|x|)}$. Let $I_x(\bar{U})$ be a Boolean formula which equals one iff $u_1u_2\ldots u_n$ is the encoding of initial i.d. of $\mathfrak{M}$ on x, and let $F(\bar{V})$ be the Boolean formula which equals one iff $v_1v_2\ldots v_n$ is the encoding of an accepting i.d. of $\mathfrak{M}$. Then, $x \in L \Leftrightarrow \mathfrak{M}$ accepts x in $\leq c^{p(|x|)}$ steps

$$\Leftrightarrow (\exists\bar{U})(\exists\bar{V})[I_x(\bar{U}) \wedge F(\bar{V}) \wedge A_{\log_2 c \cdot p(|x|),n}(\bar{U},\bar{V})].$$

The Boolean formula on the righthand side of the above equivalence is of length bounded by a polynomial in $|x|$, and can be rewritten as a formula E_x such that $x \in L \Leftrightarrow E_x \in B_\omega$ by renaming the variables appropriately. We shall leave it to the reader to convince himself that the function mapping x to E_x is log-space computable. Hence $L \leq_{\log} B_\omega$. It is also not hard to show that B_ω is recognizable in deterministic linear space, which completes the proof.

Essentially the same construction may be used to prove that polynomial space is $\leq_{\log}$ 1EQ. A polynomial space upper bound on 1EQ follows from the well-known fact that a first-order formula with n quantifiers and no predicates other than equality is valid iff it is valid for domains of all cardinalities between 1 and n.

Corollary 4.4. 1EQ is $\leq_{\log}$-complete in polynomial space.

5. INTEGER EXPRESSIONS

We shall prove in this section that N-MEMBER($\{U,+\}$) is $\leq_{\log}$-complete in $\mathcal{NP}$, and that N-INEQ($\{U,+\}$) is $\leq_{\log}$-complete in Σ_2^p. The latter is an example of a reasonably natural decision problem for which the Σ_k^p classes provide a precise complexity characterization.

The proofs that N-INEQ($\{U,+,\neg\}$) and N-MEMBER($\{U,+,\neg\}$) are $\leq_{\log}$-complete in polynomial space involve a combination of the techniques used in Theorem 5.2 below and Theorem 4.2, but are too long to present here. They will appear in [13].

For simplicity we shall identify nonnegative integers with their binary representations, e.g., $|x|$ for $x \in N$ means the number of digits in the binary representation of x.

Lemma 5.1. N-MEMBER($\{U,+\}$) $\in \mathcal{NP}$.

Proof. Let x be a nonnegative integer, and let E be an integer expression. Define a "proof" that $(x,E) \in$ N-MEMBER($\{U,+\}$) recursively as follows: (x,x) is a proof of (x,x); if P_1 is a proof of (x_1,E_1) and P_2 is a proof of (x_2,E_2), then (P_1,P_2) is a proof of $(x_1+x_2,(E_1+E_2))$; if P_1 is a proof of (x,E), then P_1 is also a proof of $(x,(E \cup F))$ and of $(x,(F \cup E))$ for any integer expression F. Let $Q(x,E,P)$ be the predicate which is true iff P is a proof of (x,E). It is not hard to see that Q is computable in deterministic polynomial time, and that if $Q(x,E,P)$, then $|P|$ is bounded by a polynomial in $|x| + |E|$. It follows that $x \in L(E)$ iff $\exists P[Q(x,E,P)$ and $|P| \leq$ polynomial $(|x| + |E|)]$, so by Theorem 4.1, N-MEMBER($\{U,+\}$) $\in \Sigma_1^p = \mathcal{NP}$.

Theorem 5.1. N-MEMBER($\{U,+\}$) is $\leq_{\log}$-complete in $\mathcal{NP}$.

Proof. Karp [4] has shown that KNAPSACK $\leq_{\log}$-complete in $\mathcal{NP}$, where KNAPSACK $= \{(a_1,a_2,\ldots,a_n,b) \mid n > 0, a_i \in N$ for $1 \leq i \leq n$, $b \in N$, and $\sum_{i=1}^{n} a_i z_i = b$ for some integers $z_i \in \{0,1\}$ for $1 \leq i \leq n\}$.

But $(a_1,a_2,\ldots,a_n,b) \in$ KNAPSACK $\Leftrightarrow$ $(b,(a_1 \cup 0)+(a_2 \cup 0)+\ldots+(a_n \cup 0)) \in$ N-MEMBER($\{U,+\}$).

Hence, KNAPSACK $\leq_{\log}$ N-MEMBER($\{U,+\}$), which completes the proof.

Theorem 5.2. N-INEQ($\{U,+\}$) is $\leq_{\log}$-complete in Σ_2^p.

Proof. A simple induction on the length of integer expressions implies that if $L(E) \neq L(F)$, then there is a $z \in N$ such that $z \in L(E) \Leftrightarrow z \notin L(F)$, and $|z| \leq |E| + |F|$. Hence, $L(E) \neq L(F) \Leftrightarrow \exists z[\exists P_1[Q(z,E,P_1)] \Leftrightarrow \neg\exists P_2[Q(z,F,P_2)]]$ where Q is the predicate defined in

Lemma 5.1, and all quantifiers range over words whose length is bounded by a polynomial in $|E| + |F|$. Thus, Theorem 4.1 and standard manipulation of quantifiers implies that N-INEQ($\{U,+\}$) $\in \Sigma_2^p$.

To complete the proof, we shall show that $B_2 \leq_{\log}$ N-INEQ($\{U,+\}$).

Let $A(\bar{X}_1,\bar{X}_2)$ be a Boolean formula, which by the remark following Theorem 4.2 may be assumed to be in disjunctive form.

For simplicity assume the variables in A are $x_{i,j}$ for $i = 1,2$ and $1 \leq j \leq n$. Let D_k be the set of literals (variables or their negations) in the k^{th} disjunct of A for $1 \leq k \leq m$. Let $[\alpha \in D_k] = 1$ if $\alpha \in D_k$; $[\alpha \in D_k] = 0$ if $\alpha \notin D_k$. For each literal α, define $I(\alpha) \in N$ to be

$$I(\alpha) = \sum_{k=1}^{m} [\alpha \in D_k] b^k \text{ where } b = 2^{2+\lceil \log_2 n \rceil}.$$

Let $a = \sum_{k=1}^{m} b^k$, and for $1 \leq i \leq m$ let F_i be the integer expression

$$F_i = (b^i \cup 0) + (b^i \cup 0) + \ldots + (b^i \cup 0)$$

where the term $(b^i \cup 0)$ occurs $2n-1$ times. Finally, define integer expressions

$$E_1 = na + \sum_{k=1}^{n} ((a\text{-}I(x_{1,k})) \cup (a\text{-}I(\neg x_{1,k}))),$$

$$E_2 = (\sum_{k=1}^{n} (I(x_{2,k}) \cup I(\neg x_{2,k}))) + (\sum_{i=1}^{m} F_i).$$

Note that if $y \in L(E_1)$ then $y = \sum_{k=1}^{m} a_k b^k$ where $n \leq a_k \leq 2n$, and there is an assignment to the Boolean variables $\bar{X}_1$ in A such that for $1 \leq k \leq m$, $a_k = 2n$ iff no $\bar{X}_1$ literal in D_k is assigned the value 0. Similarly, for any assignment to the Boolean variables $\bar{X}_2$ in A, there is a $y \in L(E_2)$ such that $y = \sum_{k=1}^{m} a_k b^k$ where $a_k = 2n$ if some $\bar{X}_2$ literal in D_k is assigned value 0, and a_k may have any value $< 2n$ otherwise.

One can now show that

$$\begin{aligned} A(\bar{X}_1,\bar{X}_2) \notin B_2 &\Leftrightarrow \forall \bar{X}_1 \exists \bar{X}_2 [A(\bar{X}_1,\bar{X}_2) = 0] \\ &\Leftrightarrow L(E_1) \subset L(E_2) \\ &\Leftrightarrow ((E_1 \cup E_2),E_2) \notin \text{N-INEQ}(\{U,+\}). \end{aligned}$$

We note that the function mapping $A(\bar{X}_1,\bar{X}_2)$ to $((E_1 \cup E_2),E_2)$ is log-space computable, which completes the proof.

6. REGULAR-LIKE EXPRESSIONS OVER {0}

Regular-like expressions over {0} resemble integer expressions in that if we define for $A \subset 0^*$ the set $N(A) \stackrel{def}{=} \{|x| \mid x \in A\} \subset N$, then $N(A_1 \cdot A_2) = N(A) + N(B)$, $N(\neg A) = \mathbf{N}\text{-}N(A)$, etc. Using squaring and concatenation, one can construct for any $n \in N$ a regular-like expression E over {0} such that $N(L(E)) = \{n\}$ and $|E|$ is proportional to $\log_2 n$, which yields a log-linear reduction from integer expression problems to regular-like expression problems. For example, the proof that INEQ($\{0\}$, $\{U,\cdot,^2,\neg\}$) is $\leq_{\log}$-complete in polynomial space is a simple corollary of the fact that N-INEQ($\{U,+,\neg\}$) is $\leq_{\log}$-complete in polynomial space. A detailed proof will appear in [13].

The contrast between regular-like expressions over {0} and those over {0,1} is best illustrated by the case involving the operations U, $\cdot$, and $\neg$. Inequivalence over {0,1} with these operations is enormously hard to decide, while inequivalence over {0} with these operations is trivial from our point of view, i.e., is decidable in deterministic polynomial time, because a regular-like expression of length n over U, $\cdot$, $\neg$ defines a finite set of words or the complement of a finite set of words of length at most n. Details will appear in [13].

When the squaring operation is not used, word problems for regular-like expressions over {0} seem to require somewhat different methods from either integer expressions or regular-like expressions over {0,1}. The following theorem is an example.

Theorem 6.1. INEQ($\{0\}$, $\{U,\cdot,*\}$) is $\leq_{\log}$-complete in $\mathcal{NP}$.

Proof. By "expression" in this proof we shall mean regular-like expression over {0} involving only the operations $\cdot$, U, $*$.

To test whether $L(E) \neq L(F)$ in nondeterministic polynomial time, construct from expressions E and F nondeterministic finite automata A_E with at most $|E|$ states and A_F with at most $|F|$ states. If $L(E) \neq L(F)$, then standard results in automata theory imply that there is an $n \leq 2^{|E|} + 2^{|F|}$ such that $0^n \in L(E) \Leftrightarrow 0^n \notin L(F)$. Nondeterministically "guess" the binary representation of n, and test whether there is a path in the transition graph of A_E and A_F of length n to accepting states. This latter test can be carried out deterministically in time bounded by a polynomial in $|n|$ by successively squaring and multiplying the connection matrices for the transition graphs of A_E and A_F. The entire procedure thus can be carried out in nondeterministic time bounded by a polynomial in $|E| + |F|$.

To prove completeness, let SAT be the set of satisfiable Boolean formulas in conjunctive form with exactly three literals per conjunct. Cook [3] has shown that $\mathcal{NP} \leq_{\log}$ SAT, and we shall show

that SAT $\leq_{\log}$ INEQ($\{0\}$, $\{\cup,\cdot,*\}$).

Let $A(\bar{X}_1)$ be a Boolean formula in conjunctive form with three literals per conjunct. Let C_k be the set of literals in the k^{th} conjunct, $1 \leq k \leq m$. Say that A has n distinct variables, so that an assignment to the variables can be represented as a binary vector of length n. Let $p_1, p_2, \ldots, p_n$ be the first n primes. If $z \in N$ is congruent to 0 or 1 modulo p_i for $1 \leq i \leq n$, we shall say z satisfies A if the assignment ($z \bmod p_1$, $z \bmod p_2, \ldots z \bmod p_n$) satisfies A.

Let E_0 be the expression such that $L(E_0) = \{0^z \mid z \in N$ does not encode an assignment$\} = \{0^z \mid (\exists k \leq n)[z \not\equiv 0 (\bmod p_k)$ and $z \not\equiv 1 \bmod (p_k)]\}$.

$$E_0 = \bigcup_{k=1}^{n} \bigcup_{j=2}^{p_k-1} [0^j \cdot (0^{p_k})^*]$$

For each conjunct C_k, we construct an expression E_k such that if $0^z \in L(E_k)$ and z is an assignment then z does not assign the value 1 to any literal in C_k. For example, if $C_k = \{x_{1,r}, \neg x_{1,s}, x_{1,t}\}$ for $1 \leq r,s,t \leq n$ and r,s,t distinct, let z_k be the unique integer such that $0 \leq z_k < p_r p_s p_t$, $z_k \equiv 0 (\bmod p_r)$, $z_k \equiv 1 (\bmod p_s)$, and $z_k \equiv 0 (\bmod p_t)$. Then

$$E_k = 0^{z_k} \cdot (0^{p_r p_s p_t})^*.$$

Now it is not hard to show that $A(\bar{X}_1)$ is satisfiable $\Leftrightarrow (\exists z \in N)$[z encodes an assignment to A and $0^z \notin L(E_k)$ for $1 \leq k \leq m$] $\Leftrightarrow L(E_0 \cup \bigcup_{k=1}^{m} E_k) \neq 0^*$.

The reader can verify that the mapping from $A(\bar{X}_1)$ to $(E_0 \cup \bigcup_{k=1}^{m} E_k, 0^*)$ is log-space computable, which completes the proof.

7. CONCLUSIONS AND OPEN PROBLEMS

We have demonstrated that the inherent computational complexity of a large selection of word problems from automata theory, logic and arithmetic can be characterized precisely. Our results to date are summarized in the table in section 2.

We believe that the methods used here will have wide applicability in computational mathematics. Our results already imply that previous efforts to find efficient procedures for testing equivalence of regular expressions or minimizing nondeterministic finite automata (cf. [14], [15], [16]) were foredoomed. Recent studies by ourselves and coworkers of decision procedures for logical theories show that our methods are applicable to nearly all of the classical decidability results in logic, and that moreover with the exception of the propositional calculus and some theories resembling the first order theory of equality, all these decidable theories can be proved to require exponential or greater time. Although certain of the word problems considered in this paper are somewhat arbitrarily constructed, we have studied them in the hope that the methods of proof will extend to algebra, topology and other areas where decision procedures arise, and will curtail wasted effort in searching for efficient procedures when none exist.

One can easily generate several dozen word problems which are variants of those considered in this paper by considering different subsets of the operations we have defined or inventing similar ones. We hesitate to recommend this entire class of problems as an interesting research topic, but two problems we are interested in are

Open Problem: Characterize the computational complexity of INEQ($\{0\}$, $\{\cup,\cdot,*,^2,\neg\}$) and INEQ($\{0\}$, $\{\cup,\cdot,*,\neg\}$). In particular are they recognizable in time $\leq 2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}\}k}$ for any fixed k? We conjecture an affirmative answer to the latter question by reducing these problems to Presburger arithmetic (cf. 17).

Define polynomial expressions over finite sets of integers recursively as follows: $(x_1, x_2, \ldots, x_n)$ is an expression and $L((x_1,\ldots,x_n)) = \{x_1,\ldots,x_n\}$ where $n \geq 1$ and $x_1,\ldots,x_n$ are nonnegative integers expressed in binary notation; if E and F are expressions and $n \in N$ is expressed in binary notation, then (E + F), (E-F), (E×F), and $(E)^n$ are expressions and $L((E+F)) = \{x+y \mid x \in L(E)$ and $y \in L(F)\}$, $L((E-F)) = \{x-y \mid x \in L(E)$ and $y \in L(F)\}$, $L((E \times F)) = \{xy \mid x \in L(E)$ and $y \in L(F)\}$, and $L((E)^n) = \{x^n \mid x \in L(E)\}$.

Open problem: Characterize the computational complexity of the equivalence problem for polynomial expressions over finite sets of integers.

REFERENCES

1. Meyer, A.R. and L.J. Stockmeyer. The Equivalence Problem for Regular Expressions with Squaring Requires Exponential Space, 13th Annual IEEE Symp. on Switching and Automata Theory, Oct., 1972, 125-129.

2. Hopcroft, J. and J. Ullman. Formal Languages and their Relation to Automata, Addison-Wesley, c. 1969, 242pp.

3. Cook, S. The Complexity of Theorem-Proving Procedures, Conf. Rec. 3rd ACM Symp. on Theory of Computing, 1970, 151-158.

4. Karp, R. Reducibility Among Combinatorial Problems, in Complexity of Computer Computations, R.E. Miller and J.W. Thatcher, ed., Plenum Press, N.Y., 85-104.

5. Seiferas, J. Translations in the Nondeterministic Tape and Time Hierarchies, in preparation.

6. Ibarra, O. A Note Concerning Nondeterministic Tape Complexities, JACM, Vol. 19, 1972, 608-612.

7. Cook, S. A Hierarchy for Nondeterministic Time Complexity, Conf. Rec. 4th ACM Symp. on Theory of Computing, 1972, 187-192.

8. Blum, M. On Effective Procedures for Speeding Up Algorithms, JACM Vol. 18, 1971, 290-305.

9. Meyer, A. and Stockmeyer, L. Some Inherently Difficult Word Problems, Part I, to appear in Journal of Computer and System Sciences.

10. Meyer, A. and Stockmeyer, L. Nonelementary Word Problems in Automata and Logic, to be presented at the Amer. Math. Soc. Symp. on the Complexity of Computation, New York, April 1973.

11. Bauer, M., D. Brand, M. Fischer, A. Meyer, and M. Paterson. A Note on Disjunctive Form Tautologies, SIGACT NEWS, 1973, to appear.

12. Savitch, W. Relationships Between Nondeterministic and Deterministic Tape Complexities, J. Computer and System Sciences Vol. 4, 1970, 177-192.

13. Stockmeyer, L. Some Inherently Difficult Word Problems, Part II, in preparation.

14. Brzozowski, J.A. Derivatives of Regular Expressions, JACM Vol. 11, 1964, 481-494.

15. Wiener, P. and T. Kameda. On the Reduction of Non-Deterministic Automata, Tech. Report No. 57, Dept. of Electrical Engineering, Princeton University, 1968.

16. Ginzburg, A. A Procedure for Checking Equality of Regular Expressions, JACM, Vol 14, 355-362.

17. Oppen, D.C. Elementary Bounds for Presburger Arithmetic, this volume.