MATHEMATICAL SYSTEMS THEORY **10**, 33–52 (1976)

On the Complexity of Finite, Pushdown, and Stack Automata

by

H. B. HUNT, III*

Harvard University
Cambridge, Massachusetts 02138

ABSTRACT

The complexity of predicates on several classes of formal languages is studied. For finite automata, pushdown automata, and several classes of stack automata, every nontrivial predicate on the languages accepted by the 1-way devices requires as much time and space as the recognition problem for any language accepted by the corresponding 2-way devices. Moreover there are nontrivial predicates $\mathscr{P}$ on the languages accepted by the 1-way devices such that $\{M|\mathscr{P}(L(M))$ is true$\}$ is the accepted language of some corresponding one or two head 2-way device. Thus our lower bounds are fairly tight.

1. Introduction. Recently considerable activity has been devoted to studying the computational complexity of decidable problems in combinatorics, logic, and automata theory. Cook [1] and Karp [2] among others have studied a variety of combinatorial problems. Meyer [3] and Fischer and Rabin [4] have studied several decidable theories of mathematical logic. Stockmeyer and Meyer [5], Hunt and Rosenkrantz [6], and Hunt, Rosenkrantz, and Szymanski [7] have studied formal language theory emphasizing the regular and context-free languages. Here we consider the complexity of decidable problems for several classes of formal languages including the languages accepted by multiple-head finite automata, pushdown automata, and stack automata.

Our research is motivated by several questions concerning computational complexity and the theory of algorithms.

1. Can techniques be found to prove uniform nonlinear lower time or space complexity bounds for nontrivial classes of problems?
2. What is the relationship between time and space complexity of languages in P?
3. Can sufficient conditions be found for a nontrivial class of problems in P to require "as much space" as any problem in P?
4. What is the relationship between the classes of languages accepted by deterministic and nondeterministic $L(n)$ space-bounded Turing machines?
5. What is the time and space complexity of the various decidable problems of the natural extensions of the context-free languages intermediate between the context-free languages and the context-sensitive languages?

* This research was sponsored in part by an NSF graduate fellowship in computer science at Cornell University, NSF grant GJ-35570 at Princeton University, and by United States Army Contract No. DA-31-124-ARO-D-462 at the University of Wisconsin.

We provide insights into each of these questions. We provide definite answers to Questions 3 and 5. We provide partial or tentative answers to Questions 1 and 2.

Let $\mathscr{F}$ be a family of languages. A general technique for efficiently reducing the membership problem for any language in $\mathscr{F}$ to any nontrivial predicate on $\mathscr{F}$ is presented. Informally if some language L in $\mathscr{F}$ is "hard" to recognize, then all nontrivial predicates on $\mathscr{F}$ are "hard" to decide. Conversely if the complexity of some nontrivial predicate $\mathscr{P}$ on $\mathscr{F}$ is known, then all languages in $\mathscr{F}$ are no "harder" to recognize than the decision problem for $\mathscr{P}$. We formalize these ideas in Sections 3, 4, and 5. We use this technique to relate the work of Savitch [8] and Hartmanis and Hunt [9] on Question 4 above to that of Jones [10] and Sudborough [11] on log-reducibility and multi-head finite automata, respectively. Similarly we use this technique to relate our work to that of Jones and Laaser [12], Cook [13], and Cook and Sethi [14] on the space complexity of languages in P. Finally this technique is also used to show that all nontrivial predicates on several of the natural classes of languages intermediate between the context-free and context-sensitive languages, such as the indexed languages, require nonpolynomial time.

The remainder of this paper is divided into five sections. Section 2 consists of definitions and preliminaries. Section 3 consists of a detailed study of the complexity of predicates on the 1-way stack automaton languages. In Section 4 we study the time and space complexity of all nontrivial predicates on the regular and context-free languages. In Section 5 the results of Sections 3 and 4 are generalized. We also relate our results to Rice's Theorem for the recursively enumerable sets. Section 6 is a short conclusion.

2. Definitions. We assume that the reader is familiar with the definitions of finite automata, pushdown automata, and stack automata, abbreviated by *FA*, *PDA*, and *SA*, respectively. Formal definitions and a discussion of PDA and SA can be found in [15] and [16]. *Nested stack automata*, abbreviated by *nSA*, are a still more powerful class of devices introduced by Aho [15], whose class of accepted languages equals the class of indexed languages introduced by Aho [17]. In Sections 3–5 we consider several subclasses of the stack automata especially the *nonerasing stack automata*, abbreviated *NESA*, the *checking stack automata*, abbreviated *CSA*, and the *reading pushdown automata*, abbreviated *RPDA*. An NESA is a stack automaton that never erases its storage tape. A CSA is an NESA that, once its stack head enters its stack, can never write upon its stack thereafter. An RPDA is a stack automaton whose storage tape head can move upwards, only with a reset move that puts its storage tape head on top of its stack. Related grammatical concepts include the *indexed grammars* of [17] and the *macro grammars* of [18] or [19].

Formally the various kinds of automata mentioned above are defined as k-tuples for different values of k. The coordinates of these k-tuples include a finite nonempty set of states Q, a finite nonempty input tape alphabet Σ, and a finite nonempty storage tape alphabet Γ. Also included in the formal definition is the start state q_0, the set of accepting states $F \subseteq Q$, and the next state function δ. We are interested in the time and space complexity of various decidable problems of the classes of automata mentioned above. To discuss time and space complexity

meaningfully, we need fixed formal representations of these classes of automata as strings over some finite alphabet. Our complexity results will assume this particular formal representation of FA, PDA, and SA.

Our representation of an automaton is a finite sequence of move-rules as described informally below.

(1) For a 1-way nondeterministic FA M, a move-rule or move is a 3-tuple (q, a, r) with $q, r \in Q$ and $a \in \Sigma \cup \{\lambda\}$.[2] The interpretation of (q, a, r) is that M, when in state q and scanning symbol $a \in \Sigma$ (or any symbol in Σ if $a = \lambda$), can in one move change state to r and move its input head one tape square to the right if $a \in \Sigma$ (or keep its input head stationary if $a = \lambda$.)

(2) For a 1-way nondeterministic PDA M, a move-rule or move is a 5-tuple (q, a, b, r, w) with $q, r \in Q$, $a \in \Sigma \cup \{\lambda\}$, $b \in \Gamma$ and $w \in \Gamma^*$. The interpretation of (q, a, b, r, w) is that M, when in state q scanning input tape symbol $a \neq \lambda$ (or any input symbol if $a = \lambda$) with b as its top pushdown store symbol, can in one move change state to r, replace b on the top of its pushdown store by w, and move its input tape head one tape square to the right if $a \neq \lambda$ (or keep its input tape head stationary if $a = \lambda$.)

(3) For a 1-way SA M there are two distinct types of move-rules or moves. If M is in the stack reading mode, then a move-rule is a 6-tuple (q, a, b, r, d_1, d_2) with $q, r \in Q$, $a \in \Sigma$, $b \in \Gamma$, $d_1 \in \{S, R\}$, and $d_2 \in \{L, R, S\}$. The interpretation of (q, a, b, r, d_1, d_2) is that M, when in state q scanning input tape symbol a and scanning stack tape symbol b, can in one move—
   (a) change to r;
   (b) move its input head one tape square to the right or leave its input tape head stationary, if $d_1 = R$ or S, respectively; and
   (c) move its stack tape head one tape square to the left, right, or leave its stack tape head stationary, if $d_2 = L$, R, or S, respectively.[3]

The representation of move-rules of M, when M is in the stack writing mode, is similar and is left to the reader.

We assume that the input tape alphabet and the pushdown store or stack alphabet of each automaton equal $\{0, 1\}$ and Γ, respectively. Γ is some fixed finite nonempty alphabet of cardinality ≥ 2. States are denoted by the binary numerals for the nonnegative integers $\{0, 1, \cdots, |Q|-1\}$. The start state is always denoted by the numeral 0. The accepting states are denoted in some simple uniform fashion. Thus each automata M is represented by a string over some finite alphabet Δ invariant of M. We define $|M|$ to be the length of this representation of M. Henceforth M will refer both to the automaton and to its representation.

We use the following abbreviations—

1. *cfl* for context-free language;
2. *cfg* for context-free grammar;
3. *csl* for context-sensitive language;
4. *Tm* for Turing machine;
5. *re* for recursively enumerable;
6. *i.o.* for infinitely often;

[2] λ denotes the empty word. Thus our nondeterministic FA have epsilon-rules.

[3] Move-rules for 2-way FA, PDA, and SA are represented analogously.

7. *a.e.* for almost everywhere, when applied to the nonnegative integers almost everywhere means except for some finite set of nonnegative integers;
8. *FA* for finite automaton;
9. *NDFA* for nondeterministic finite automaton;
10. *PDA* for pushdown automaton;
11. *SA* for stack automaton;
12. *DSA* for deterministic stack automaton;
13. *nSA* for nested stack automaton;
14. *nDSA* for nested deterministic stack automaton;
15. *NESA* for nonerasing stack automaton;
16. *NEDSA* for nonerasing deterministic stack automaton;
17. *CSA* for checking stack automaton;
18. *RPDA* for reading pushdown automaton; and
19. *RAM* for random access machine.

We also use the following definitions.

Definition 2.1. P is the class of all languages over $\{0, 1\}$ recognizable by some deterministic polynomially time-bounded Tm. NP is the class of all languages over $\{0, 1\}$ recognizable by some nondeterministic polynomially time-bounded Tm. *PSPACE* is the class of all languages over $\{0, 1\}$ recognizable by some polynomially tape-bounded Tm. ■

Definition 2.2. *Dtape* $[f(n)]$ (*Ndtape* $[f(n)]$) is the class of all languages over $\{0, 1\}$ recognizable by some $f(n)$ tape-bounded deterministic (nondeterministic) Tm. ■

The next three definitions define the different kinds of efficient reducibilities considered in this paper. In each definition Σ and Δ denote finite nonempty alphabets.

Definition 2.3. Let $L \subseteq \Sigma^*$ and $M \subseteq \Delta^*$. Let $\Pi(\Sigma, \Delta)$ denote the class of all functions from Σ^* into Δ^* computable by some deterministic polynomially time-bounded Tm. We say that L is *p-reducible* to M (written $L \leq_{\text{ptime}} M$) if there exists $f \in \Pi(\Sigma, \Delta)$ such that for all $x \in \Sigma^*$, $x \in L$ if and only if $f(x) \in M$. A language L_0 is said to be *NP-hard* if for all $L \in NP$, $L \leq_{\text{ptime}} L_0$. A language L_0 is said to be *NP-complete* if L_0 is NP-hard and is the accepted language of some nondeterministic polynomially time-bounded Tm. Analogously L_0 is said to be *PSPACE-hard* if for all $L \in$ PSPACE, $L \leq_{\text{ptime}} L_0$. L_0 is said to be *PSPACE-complete* if L_0 is PSPACE-hard and is the accepted language of some polynomially tape-bounded Tm. ■

Definition 2.4. A *log-space transducer* M is a deterministic Tm with a 2-way read-only input tape, a 1-way output tape, and several 2-way read-write work tapes such that M, given input x, always halts with some string y on its output tape and such that M never uses more than $o(\log |x|)$ tape cells on its work tapes. A function f from Σ^* into Δ^* is said to be *log-space computable* if there exists a log-space transducer M such that M, when given input $x \in \Sigma^*$, eventually halts with

output $f(x)$. For $L \subseteq \Sigma^*$ and $N \subseteq \Delta^*$ we say that L is *log-space reducible* to N (written $L \leq_{\log} N$) if there exists a log-space computable function f such that for all $x \in \Sigma^*$, $x \in L$ if and only if $f(x) \in N$. If in addition $|f(x)| \leq c\,|x| \log(|x|)$ and some log-space transducer that computes f is $o(|x| \log(|x|))$ time-bounded, we denote this by

$$L \underset{\substack{\log \\ \text{nlogn(space+time)}}}{\leq} N.$$

A language L_0 is said to be *log-space complete in Ndtape* (logn) if $L_0 \in Ndtape$ (logn) and for all $L \in Ndtape(\log n)$, $L \leq_{\log} L_0$. L_0 is said to be *log-space complete in* P if $L_0 \in P$ and for all $L \in P$, $L \leq_{\log} L_0$. □

Definition 2.5. We say that $L \subseteq \Sigma^*$ is *effectively reducible* to $M \subseteq \Delta^*$ (written $L \leq_{\text{eff}} M$) if there exists a recursive function f: $\Sigma^* \to \Delta^*$ such that for all $x \in \Sigma^*$, $x \in L$ if and only if $f(x) \in M$. □[4]

Definition 2.6. A predicate $\mathscr{P}$ is a function from a set Δ into {True, False}, $\mathscr{P}$ is said to be *nontrivial* if there exists $a, b \in \Delta$ such that $\mathscr{P}(a) =$ True and $\mathscr{P}(b) =$ False. ■

Definition 2.7. A function T(n) is said to be *countable* if there is some single-tape Tm M that is T(n) time bounded and marks off a block of length T(n) on its tape. ■

PROPOSITION 2.8.

(1) *$Dtape(\log n) = Ndtape(\log n)$ if and only if there exists a language $L_0 \in Dtape(\log n)$ such that for all $L \in Ndtape(\log n)$, $L \leq_{\log} L_0$.*

(2) *Let k be any nonnegative integer. $P \subseteq Dtape([\log n]^k)$ if and only if there exists a language $L \in Dtape([\log n]^k)$ such that for all $L \in P$, $L \leq_{\log} L_0$.* ■

A proof of (1) can be found in [10]. A proof of (2) can be found in [12].

Finally, we use $L_1 \leq_{\text{ptime}} L_2$ and $L_2 \geq_{\text{ptime}} L_1$, $L_1 \leq_{\log} L_2$ and $L_2 \geq_{\log} L_1$, $L_1 \leq_{\text{eff}} L_2$ and $L_2 \geq_{\text{eff}} L_1$, and

$$L_1 \underset{\text{nlogn(space+time)}}{\leq_{\log}} L_2 \text{ and } L_2 \underset{\text{nlogn(space+time)}}{\geq_{\log}} L_1,$$

interchangeably.

3. Stack Languages and an Exponential Time Gap. In this section the membership problems for several kinds of 2-way stack automata are efficiently reduced to many predicates on the corresponding 1-way devices, e.g. emptiness or finiteness. Since the time complexities of the membership problems for the 2-way devices are known to be exponential, exponential lower time bounds are derived for the predicates on the 1-way devices. This shows that there is an exponential complexity gap between the time complexity of many predicates on the cfl's and the corresponding predicates on the indexed or 1-way stack languages. Moreover, there are nontrivial predicates $\mathscr{P}$ on the various classes of 1-way SA languages

[4] This is the many-one reducibility in Rogers [20].

such that $\{M|\mathcal{P}(L(M))$ is true$\}$ is the accepted language of some two head 2-way device of the same type. This can be used to show that our lower bounds are fairly tight. Finally we investigate the time complexity of the recognition problems for various languages accepted by 1-way nondeterministic SA. Extending results in Rounds [21], we show that the simplest kind of stack automata, the checking stack automata, accept NP-complete languages.

The following theorem summarizes some of the known results about 2-way stack automata.

THEOREM 3.1 ([22], [23], AND [24]). *For all integers $k \geq 1$.*

(1) *a language L is accepted by a 2-way k-head SA if and only if it is accepted by some $2^{cn^{2k}}$ deterministic time bounded Tm, where c is a constant;*
(2) *a language L is accepted by a 2-way k-head DSA if and only if it is accepted by some $2^{cn^k \log n}$ deterministic time bounded Tm, where c is a constant;*
(3) *a language L is accepted by a 2-way k-head NESA if and only if it is accepted by some n^{2k} nondeterministic tape bounded Tm;*
(4) *a language L is accepted by a 2-way k-head NEDSA if and only if it is accepted by some $n^k \log n$ deterministic tape bounded Tm; and*
(5) *a language L is accepted by a 2-way k-head CSA if and only if it is accepted by some n^k nondeterministic tape bounded Tm.* ■

Here we consider only one and two head devices for a proof of Theorem 3.1, see Ibarra [24].

We also use the following two well-known results. The first is from Hennie and Stearns [25]; the second is from Ibarra [26].

PROPOSITION 3.2 [25]. *If $T_1(n)$ is a real-time countable function, then there is a set A of strings, which is accepted by some deterministic multi-tape Tm within time $T_1(n)$ but by no such machine within time $T_2(n)$ for any function $T_2(n)$ satisfying*

$$\liminf_{n \to \infty} \frac{T_2(n) \log (T_2(n))}{T_1(n)} = 0. \quad \blacksquare$$

PROPOSITION 3.3 [26]. *Let m, p, and q be integers with $m \geq 0$ and $p, q \geq 1$. Let $L_1(n) = m^{m + p/q}$ and let $L_2(n) = n^{m + p/(q+1)}$. Then there is a language A accepted by some nondeterministic $L_1(n)$ tape bounded Tm, that is not accepted by any nondeterministic $L_2(n)$ tape bounded Tm.* ■

Our first major result is the following.

THEOREM 3.5.

(1) *For all nontrivial predicates $\mathcal{P}$ on the 1-way SA languages, to decide $\mathcal{P}(L(M))$, where M is a 1-way SA, requires at least $2^{cn^2/[\log n]^2}$ time i.o. on any deterministic multi-tape Tm, where c is a constant greater than 0;*
(2) *For all nontrivial predicates $\mathcal{P}$ on the indexed language, to decide $\mathcal{P}(L(M))$, where M is a 1-way nSA, requires at least $2^{cn^2/[\log n]^2}$ time i.o. on any deterministic multi-tape Tm, where c is a constant greater than 0;*
(3) *For all nontrivial predicates $\mathcal{P}$ on the 1-way DSA languages, to decide $\mathcal{P}(L(M))$, where M is a 1-way DSA, requires at least 2^{cn} time i.o. on any deterministic multi-tape Tm, where c is a constant greater than 0;*

(4) *For all nontrivial predicates $\mathscr{P}$ on the 1-way nDSA languages, to decide $\mathscr{P}(L(M))$, where M is a 1-way nDSA, requires at least 2^{cn} time i.o. on any deterministic multi-tape Tm, where c is a constant greater than 0;*

(5) *For all nontrivial predicates $\mathscr{P}$ on the 1-way NESA languages, to decide $\mathscr{P}(L(M))$, where M is a 1-way NESA, requires tape $> n^r$ for all $r < 2$, i.o. on any nondeterministic Tm;*

(6) *For all nontrivial predicates $\mathscr{P}$ on the 1-way NEDSA languages, to decide $\mathscr{P}(L(M))$, where M is a 1-way NEDSA, requires tape $> n^r$ for all $r < 1$, i.o. on any deterministic Tm; and*

(7) *there exists $r > 0$ such that for all nontrivial predicates $\mathscr{P}$ on the indexed languages to decide $\mathscr{P}(L(G))$, where G is an indexed or OI-macro grammar, requires at least 2^{n^r} time i.o. on any deterministic Tm.* ■

Proof. (1) Let $\mathscr{P}$ be any nontrivial predicate on the 1-way SA languages. Without loss of generality we assume that $\mathscr{P}(\phi)$ is false. Since $\mathscr{P}$ is nontrivial there exists a 1-way SA M_0 such that $\mathscr{P}(L(M_0))$ is true. Let $L_0 = L(M_0)$, Let M_i be any arbitrary 2-way SA. For each x, a 1-way SA $M_{i,x}$ can be constructed such that

$$\mathrm{L}(\mathrm{M}_{i,x}) = \begin{cases} \phi, & \text{if } x \notin L(M_i), \\ L_0, & \text{otherwise.} \end{cases}$$

For each input $x = x_1 x_2 \cdots x_n$ to M_i, $M_{i,x}$ is constructed as follows.

(a) All input tape configurations of M_i on x are embedded in $M_{i,x}$'s finite state control.

(b) $M_{i,x}$ uses its stack to simulate M_i's stack directly.

(c) $|M_{i,x}| \leq k_i|x|\log|x|$, where k_i depends only upon M_i and M_0 not on x.

(d) $M_{i,x}$ simulates M_i on x. If M_i accepts x, then $M_{i,x}$ simulates M_0 on its ($M_{i,x}$'s) input. If M_i rejects x or fails to halt on x, then $M_{i,x}$ fails to halt for all of its inputs.

The reader should note that $M_{i,x}$'s simulation of M_i on input x only involves epsilon moves. $M_{i,x}$'s state set includes states of the form (p, v), where p denotes a state of M_i and v is a binary numeral for a nonnegative integer $[v]$ with $[v] \leq |x| = n$. State (p, v) signifies that M_i is in state p and that M_i's input tape head is scanning the $[v]^{th}$ character of x.

We illustrate the construction of $\mathrm{M}_{i,x}$ (actually the representation of $M_{i,x}$) from M_i, M_0, and x. We note that moves or move-rules of 2-way SA can also be denoted as described in Section 2. Let $\mu = (p, a, b, q, L, R)$ with p, q binary numerals and a, b letters in the 2-way SA's input tape and stack tape alphabets, respectively. The interpretation of μ is that the 2-way SA can in one move, when in the state denoted by p, scanning a on its input tape and b on its stack tape, change state to that state denoted by q, move its input tape head one tape square to the left, and move its stack tape pointer one tape square to the right. Suppose M_i contains rule μ. Then $M_{i,x}$ simulates the application of μ on x by all 6-tuples of the form

$$((p, v_1), c, b, (q, v_2), S, R), \text{ where}$$

(i) v_1 and v_2 are binary numerals for the positive integers $[v_1]$ and $[v_2]$, respectively, with $[v_1], [v_2] \leq |x| = n$ and $[v_1] - [v_2] = 1$;

(ii) the $[v_1]^{th}$ symbol of x is a; and

(iii) c is any element of M_i's tape alphabet.

The amount of tape required to write out each such move-rule is $o(\log n)$. There are $o(n)$ such move-rules. Thus $|M_{i,x}| \leq k_i n \log n$, where k_i depends only upon M_i and M_0 not x. Clearly the move-rules of $M_{i,x}$ can be constructed deterministically in time bounded by a polynomial π_i in $|x|$; and the intermediate storage required to perform this construction is $o(\log n)$.

But $\mathscr{P}(L(M_{i,x}))$ is true if and only if $x \in L(M_i)$. This follows since $x \notin L(M_i)$ implies that $L(M_{i,x}) = \varnothing$ and $\mathscr{P}(L(M_{i,x}))$ is false by assumption. Similarly if $x \in L(M_i)$, then $L(M_{i,x}) = L_0$ and $\mathscr{P}(L(M_{i,x}))$ is true by assumption. Let $T(m)$ be the time required by some deterministic multi-tape Tm to decide $\mathscr{P}$. Then for all constants $c > 0$, all languages $\mathscr{L}$ accepted by 2^{cn^2} time-bounded deterministic multi-tape Tm's are accepted deterministically in time $\leq \pi_j(n) + T(|M_{j,x}|)$ a.e. for some j (where M_j is a 2-way SA and $\mathscr{L} = L(M_j)$) by some deterministic multi-tape Tm.

From 3.2 there exists a 2-way SA language $\hat{L}$ and constants s, $t > 0$ with $s - t > 0$ such that $\hat{L}$ is the accepted language of some 2^{sn^2} deterministic time-bounded Tm, and $\hat{L}$ is not the accepted language of any 2^{tn^2} deterministic time-bounded Tm. Let M_j be a 2-way SA for which $\hat{L}$ equals $L(M_j)$. Then $\pi_j(n) + T(|M_{j,x}|) > 2^{tn^2}$ i.o. Let $c = t/2k_j^2$. Then letting $m = |M_{j,x}|$, $T(m) > 2^{(t/2k_j^2)m^2/[\log m]^2}$ i.o.

We prove this by contradiction. Suppose that $T(m) \leq 2^{(t/2k_j^2)m^2/[\log m]^2}$ a.e. For almost all inputs x to M_j, $n \log n \leq m \leq k_j \cdot n \cdot \log n$. Thus

$$\pi_j(n) + T(m) \leq \pi_j(n) + 2^{(t/2k_j^2)} \cdot \frac{k_j^2 n^2 [\log n]^2}{[\log(n \log n)]^2} \text{ a.e. } \leq 2^{tn^2} \text{ a.e.}$$

(2) The proof of (2) follows immediately from that of (1), since every 2-way SA is also a 2-way nSA.

(3) The proof of (3) is almost identical to that of (1), except that in this case $M_{i,x}$ is a 1-way DSA. Let $T(n)$ be the time required for some deterministic multi-tape Tm to decide $\mathscr{P}$. Then for all constants $c > 0$, all languages accepted by some $2^{cn \log n}$ time-bounded deterministic multi-tape Tm are accepted deterministically in time $\leq \pi_i(n) + T(|M_{i,x}|)$ a.e., where $n = |x|$, by some multi-tape Tm.

From 3.2 there is a 2-way DSA language $\hat{L}$ and constants s, $t > 0$ with $s - t > 0$ such that $\hat{L}$ is the accepted language of some $2^{sn \log n}$ deterministic time-bounded Tm, and $\hat{L}$ is not the accepted language of any $2^{tn \log n}$ deterministic time-bounded Tm. Let M_j be a 2-way DSA for which $\hat{L}$ equals $L(M_j)$. Then $\pi_j(n) + T(|M_{j,x}|) > 2^{tn \log n}$ i.o. Thus letting $m = |M_{j,x}|$, $T(m) > 2^{(t/2k_j)m}$ i.o. The remainder of the proof is analogous to that of (1) and is left to the reader.

(4) The proof of (4) follows immediately from that of (3), since every 2-way DSA is also a 2-way nDSA.

(5) $M_{i,x}$ is a 1-way NESA. Let $L(n)$ be the space required for some nondeterministic multi-tape Tm to recognize the set $\{M | M$ is a 1-way NESA and $\mathscr{P}(L(M))$ is true$\}$. Then all languages accepted by some n^2 tape bounded nondeterministic Tm are accepted in nondeterministic space

$\leq o(L(|M_{i,x}|)+k_i n\log n)$ a.e. Suppose some nondeterministic Tm that recognizes the set $\{M|M$ is a 1-way NESA and $\mathscr{P}(L(M))$ is true$\}$ uses space $L(n) < n^{2-\varepsilon}$ a.e. for some $\varepsilon > 0$. Since $|M_{i,x}| \leq k_i n\log n$, this implies that there exists an $\varepsilon' > 0$ such that all languages recognized by n^2 tape bounded nondeterministic Tm are also recognized within space $n^{2-\varepsilon'}$. This contradicts 3.3.

(6) $M_{i,x}$ is a 1-way NEDSA. We note that for fixed 2-way NEDSA M_i, the construction of $M_{i,x}$ from M_i and x can be effected by a log$n$ tape bounded deterministic transducer. The proof of (6) now follows by an argument similar to that of (5) and is left to the reader.

(7) In [15] Aho gives an algorithm for converting an arbitrary 1-way nSA M into an equivalent indexed grammar G. This algorithm can be seen to be executable deterministically in time bounded by a polynomial in $|M|$. In [18] Fischer gives an algorithm for converting an arbitrary indexed grammar G into an equivalent OI-macro grammar G'. Again this algorithm can be seen to be executable deterministically in time bounded by a polynomial in $|G|$.

We note that the constructions, used in the proof of this theorem and that of Proposition 3.9 below, are closely related to the construction used in [16] to show that all 2-way SA languages are recursive. □

We present several immediate consequences of 3.5. Let M and N denote arbitrary 1-way stack automata, indexed grammars, or OI-macro grammars. If $\mathscr{P}$ is any nontrivial predicate on the 1-way stack or indexed languages, respectively, such that $L(M) = L(N)$ implies $\mathscr{P}(M) = \mathscr{P}(N)$, then $\mathscr{P}$ satisfies the conditions of 3.5. Thus such predicates as "$L(M)$ is empty;" "$L(M)$ is finite;" "$L(M)$ is bounded;" "$x \in L(M)$," where x is any fixed string in $\{0, 1\}^*$, all require exponential time for the 1-way stack automata, indexed grammars, and OI-macro grammars. Moreover this implies—

COROLLARY 3.6. *$\{(x, M)|x \in L(M)$ and M is a 1-way DSA, 1-way SA, indexed grammar, or OI-macro grammar$\} \notin P$, i.e. there is no deterministic polynomial algorithm uniform in both M and x for the membership problem for the 1-way DSA, 1-way SA, indexed grammars, or OI-macro grammars.* ■

Next we present upper time or space bounds for several of the easiest predicates on the various types of 1-way devices mentioned in 3.5. In each case the simplest nontrivial predicates are recognizable by 2-way 2-head devices of the same type. This together with 3.1 shows that the lower bounds of 3.5 are reasonably tight. The emptiness problems for the 1-way devices are shown to be recognizable by 2-way 2-head nondeterministic devices of the same type. Finally the emptiness problems for the 1-way deterministic devices are shown to be as hard as the emptiness problems for the 1-way nondeterministic devices of the same type.

PROPOSITION 3.7. *For all $x \in \{0, 1\}^*$,*

(1) *the set $\{M|M$ is a 1-way SA and $x \in L(M)\}$ is recognizable by some 2^{cn^4} deterministic time bounded multi-tape Tm, where c is a constant;*

(2) *the set $\{M|M$ is a 1-way DSA and $x \in L(M)\}$ is recognizable by some $2^{cn^2\log n}$ deterministic time bounded multi-tape Tm, where c is a constant;*

(3) *the set* $\{M|M$ *is a* 1*-way NESA and* $x \in L(M)\}$ *is recognizable by some* n^4 *nondeterministic tape bounded Tm; and*

(4) *the set* $\{M|M$ *is a* 1*-way NEDSA and* $x \in L(M)\}$ *is recognizable by some* $n^2 \log n$ *deterministic tape bounded Tm.* ■

Proof. This proposition follows from Theorem 3.1 once the sets in statements (1) through (4) are shown to be recognizable by 2-way 2-head stack devices of the same type. Thus, for example, to prove (1) it suffices to show that the set $L = \{M|M$ is a 1-way SA and $x \in L(M)\}$ is recognizable by a 2-way 2-head SA. We only sketch the proof of (1) for $x = \lambda$, the empty string.

We describe the operation of a 2-way 2-head SA **A** that accepts L. **A**, given input y, first verifies that y is a 1-way SA, that is y consists of a finite set of 1-way SA move-rules with designated start and accepting states. If not **A** halts without accepting. If so **A** directly simulates the operation of y on λ, using its stack to simulate y's stack. The only point that requires verification is that **A**, when simulating y on λ, can apply any allowable next move-rule of y and can only apply allowable next move-rules of y.

A accomplishes this as follows. Suppose that, after k steps of some computation of y on λ, y is in state q with stack contents w. **A**'s stack contents is also w; and **A**'s first input tape head points to some move-rule μ of y such that $\mu = (q, -, -, -, -, -)$. Next **A** guesses some move-rule $\nu = (q', b, a, q'', s, -)$ of y as an allowable $k+1^{th}$ move of some computation of y on λ. Using both of its input tape heads **A** verifies that $q' = q$ and that a is the top symbol of its (**A**'s) stack. If not **A** halts without accepting y, if so **A** moves its first input tape head to point to some move-rule $\mu' = (r, -, -, -, -, -)$ of y for which $r = q''$. **A** uses both of its input tape heads to verify that $r = q''$. Finally **A** updates its stack according to ν. □

Thus the lower bounds of Theorem 3.5 are fairly tight.

A slight modification of the ideas sketched in the proof of Proposition 3.7 shows the following.

PROPOSITION 3.8. (1) $L = \{M|M$ *is a* 1*-way SA and* $L(M) \neq \varnothing\}$ *is the accepted language of a* 2*-way* 2*-head SA; and*

(2) $L = \{M|M$ *is a* 1*-way NESA and* $L(M) \neq \varnothing\}$ *is the accepted language of a* 2*-way* 2*-head NESA.* ■

Proof. (1) The automaton **A** that accepts L behaves as follows. **A**, given input y, first verifies that y is a 1-way SA. If not **A** halts without accepting. If so **A** guesses a string x one character at a time such that $x \in L(M)$. **A** verifies that $x \in L(M)$ as described in the proof of 3.7. The proof of (2) is similar. □ Thus our lower bounds are fairly tight for some interesting predicates on the 1-way devices as well.

Next we note that the lower bounds for the complexity of the emptiness problem for the 1-way nondeterministic stack devices also hold for the emptiness problem for the 1-way deterministic stack devices of the same type.

PROPOSITION 3.9.

(1) *The emptiness problem for* 1*-way DSA requires at least* $2^{cn^2/[\log n]^2}$ *time i.o. on any deterministic multi-tape Tm, where c is a constant greater that* 0;

(2) *The emptiness problem for* 1*-way nDSA requires at least* $2^{cn^2/[\log n]^2}$ *time i.o. on any deterministic multi-tape Tm, where c is a constant greater than* 0; *and*

(3) *The emptiness problem for* 1*-way NEDSA requires tape* $> n^r$ *for all* $r < 2$, *i.o. on any nondeterministic Tm.* ■

Proof. We sketch the proof of (1).

Let L be any fixed 2-way SA language. We efficiently reduce the membership problem for L to the emptiness problem for 1-way DSA.

Let M be some 2-way SA that accepts L. Without loss of generality we assume that for each state, tape symbol, and stack symbol triple (p, s, t), M has at most two distinct applicable move-rules, i.e. M when in state p, scanning tape symbol s and stack symbol t can make at most two distinct moves. For every arbitrary 2-way SA, an equivalent 2-way SA with this property can be constructed effectively by adding more states. For each input $x = x_1 \cdots x_n$ to M, a 1-way DSA M_x is constructed as described below—

(a) All input tape configurations of M on x are embedded in M_x's finite state control.

(b) M_x uses its stack directly to simulate M_i's stack.

(c) $|M_x| \leq k \cdot n\log n$, where k depends only upon M not on x.

(d) M_x simulates M on the fixed string x. M_x uses its input to determine which sequence of moves to apply to x.

The reader should note that unlike $M_{i,x}$ in the proof of (1) of Theorem 3.5, M_x has no epsilon-rules. We illustrate the moves of M_x with an example. Suppose $m_1 = (p, a, b, q, L, R)$ and $m_2 = (p, a, b, q', R, S)$ are the two distinct moves of M for the state, input tape symbol, stack symbol triple (p, a, b). Suppose the i^{th} character of x equals a, then $m_1' = ((p, v_1), 0, b, (q, v_2), R, R)$ and $m_2' = ((p, v_1), 1, b, (q', v_3), R, S)$ are moves of M_x. Here v_1, v_2, and v_3 are the binary numerals for the nonnegative integers i, $i-1$, and $i+1$, respectively. The input symbol 0 causes M_x to execute m_1' and thus to simulate rule m_1. Similarly the input symbol 1 causes M_x to execute m_2' and thus to simulate rule m_2. Since for each state, input tape symbol, stack symbol triple, there are at most two distinct applicable moves, 0 and 1 suffice to determine M_x's moves.

Thus an input string $y = y_1 \cdots y_m$ to M_x causes M_x to simulate M on x using the rules determined by $y_1, \cdots, y_m$. If this sequence of rules is not applicable (M_x ends in a trap state) or does not correspond to an accepting computation of M on x, M_x halts without accepting. If the sequences of moves of M on x does result in M's accepting x, then M_x halts and accepts y. Thus $L(M_x) \neq \varnothing$ if and only if $x \in L(M)$. Since the construction of M_x from x requires time $\leq K \cdot n\log n$, where K depends only upon M not on x, $K \cdot n\log n + T(|M_x|) \geq$ time required to determine if $x \in L$, a.e. Here T is the time required by some multi-tape deterministic Tm to determine if $L(M_x) = \varnothing$. The lower bound of (1) now follows from an argument exactly analogous to that of the proof of (1) of 3.5.□

Finally, we consider the time complexity of the recognition problem for fixed 1-way SA languages. Rounds [21] has shown that—

(i) there exist NP-complete languages that are recognizable by 1-way SA and

(ii) all indexed languages, hence all 1-way SA languages, belong to NP.

We show that (i) holds even for the most restricted kind of stack automata studied in the literature.

PROPOSITION 3.10. *There exists an NP-complete language L that is recognizable by a 1-way CSA that is also an RPDA.* ■

Proof. L is the set of nontautological D_3-Boolean forms with all variable subscripts in unary. From [1] L is NP-complete. One 1-way CSA M that accepts L behaves as follows: Given a string x on its input tape,

(1) M verifies that x is a D_3-Boolean form f with all variable subscripts in unary. M does this in its finite control.
(2) M guesses a string $y = y_n \cdots y_1$ for which $f(y_1, \cdots, y_n)$ is false and writes y on its stack tape.
(3) M verifies that $f(y)$ is false. To do this it uses the unary subscripts of f as counters.

Since M need only read its stack tape in one direction, M is also an RPDA. □

Finally, we observe that the results of this section show that there is a provably exponential gap between the time complexity of many predicates on the deterministic cfl's and on the 1-way DSA languages. These results also show that there is a provably exponential gap between the time complexity of many predicates on the cfl's and on the 1-way SA languages, and on the cfg's and the indexed grammars. These results also suggest that such an exponential gap exists between the time complexity of the recognition problems for the cfl's and the index languages as well.

4. Finite and Pushdown Automata. Exact analogues of the results in Section 3 hold for the multi-head finite and pushdown automata as well.

THEOREM 4.1.

(1) *Let $\mathscr{P}$ be any nontrivial predicate on the regular sets over $\{0, 1\}$ such that $\mathscr{P}(\varnothing)$ is false. Then for all $L \in$ Ndtape (logn), $L \leq_{log} \{M | M$ is an NDFA with epsilon moves and $\mathscr{P}(L(M))$ is true$\}$.*
(2) *Let $\mathscr{P}$ be any nontrivial predicate on the deterministic cfl's over $\{0, 1\}$ such that $\mathscr{P}(\varnothing)$ is false. Then for all $L \in P$, $L \leq_{log} \{M | M$ is a deterministic PDA and $\mathscr{P}(L(M))$ is true$\}$.*
(3) *Let $\mathscr{P}$ be any nontrivial predicate on the cfl's over $\{0, 1\}$ such that $\mathscr{P}(\varnothing)$ is false. Then for all $L \in P$, $L \leq_{log} \{M | M$ is a PDA [or cfg] and $\mathscr{P}(L(M))$ is true$\}$.*
(4) *Let $\mathscr{P}$ be any nontrivial predicate on the cfl's over $\{0, 1\}$ such that $\mathscr{P}(\varnothing)$ is false. Then for all 2-way PDA languages L,*

$$L \underset{\substack{\log \\ n\log n(\text{space}+\text{time})}}{\leq} \{M | M \text{ is a PDA and } \mathscr{P}(L(M)) \text{ is true}\}.$$ ■

Proof. (1) It is well-known (see [9]) that the classes of languages accepted by 2-way multi-head deterministic and 2-way multi-head nondeterministic FA equal Dtape (log$n$) and Ndtape(log$n$), respectively. Noting this the proof is almost identical to that of (1) of Theorem 3.5 and can be found in [39].

(2) Cook [23] has shown that the class of languages accepted by 2-way multi-head deterministic PDA equals P. The proof now follows that of (1) of 3.5 and (1) of this theorem and is left to the reader.

(3) Cook [23] has also shown that the class of languages accepted by 2-way multi-head nondeterministic PDA equals P.

The theorem holds for the cfg's as well as the *PDA* since there exists a deterministic log$n$ space-bounded transducer M such that M, when given a *PDA* as input, outputs an equivalent cfg G. The standard construction in [16] can be modified so that it requires only $o(\log |M|)$ intermediate space, when executed on a two-way transducer.

(4) For a fixed one head 2-way *PDA* M_i, $M_{i,x}$ can be constructed from M_i using only $o(\log |x|)$ intermediate storage and $o(|x| \log |x|)$ time and space. □

Theorem 4.1 provides whole new classes of log-space complete problems for Ndtape(log$n$) and P. Thus it extends the results in Jones [10], Jones and Laaser [12], Cook [13], and Cook and Sethi [14]. Moreover (4) of Theorem 4.1 strongly suggests that every nontrivial predicate on the cfl's, when applied to the *PDA*, requires nonlinear time. Otherwise every 2-way *PDA* language is recognizable in $o(n\log n)$ space and time on a multi-tape deterministic Tm. The best known algorithms for recognizing arbitrary 2-way *PDA* languages require $o(n^3)$ time and $o(n^2)$ space on random access machines, not to mention Turing machines.

One immediate corollary of Theorem 4.1 follows.

COROLLARY 4.2.

(1) *If there exists a nontrivial predicate $\mathscr{P}$ on the regular sets over $\{0, 1\}$ such that $\{M|M$ is an NDFA with epsilon moves and $\mathscr{P}(L(M))$ is true$\} \in$ Dtape(log$n$), then* Dtape(log$n$) = Ndtape(log$n$).

(2) *If there exists a nontrivial predicate $\mathscr{P}$ on the deterministic cfl's over $\{0, 1\}$ and a positive integer k such that $\{M|M$ is a deterministic PDA and $\mathscr{P}(L(M))$ is true$\} \in$ Dtape($[\log n]^k$), then $P \subseteq$ Dtape($[\log n]^k$).*

(3) *If there exists a nontrivial predicate $\mathscr{P}$ on the cfl's over $\{0, 1\}$ and a positive integer k such that $\{M|M$ is a PDA [or cfg] and $\mathscr{P}(L(M))$ is true$\} \in$ Dtape($[\log n]^k$), then $P \subseteq$ Dtape($[\log n]^k$).* ■

Proof. From Theorem 4.1 for any such predicate $\mathscr{P}$ and all $L \in$ Ndtape(log$n$), either $L \leq_{\log} \{M|M$ is an NDFA with epsilon moves and $\mathscr{P}(L(M))$ is true$\}$ or $L \leq_{\log} \{M|M$ is an NDFA with epsilon moves and $\mathscr{P}(L(M))$ is false$\}$. Since Dtape(log$n$) is closed under complementation, (1) follows from (1) of Proposition 2.8. The proofs of (2) and (3) follow from (2) of Proposition 2.8 by a similar argument. □

Moreover analogues of Propositions 3.7–3.9 hold for the finite and pushdown automata.

PROPOSITION 4.3. *The set $L_0 = \{M|M$ is an NDFA with epsilon moves and $L(M) \neq \varnothing\}$ is the accepted language of some 2-way 2-head nondeterministic FA. Thus L_0 is the accepted language of some 2-way multi-head deterministic FA if and only if* Dtape(log$n$) = Ndtape(log$n$). ■

Proof. Theorem 4.1 implies that for all $L \in$ Ndtape(log$n$), $L \leq_{\log} L_0$. The proof that L_0 is the accepted language of some 2-way 2-head NDFA exactly parallels that of 3.7 and 3.8 and is left to the reader. The proposition now follows from (1) of Corollary 4.2. □

This provides a simple direct proof of the result in Sudborough [11] that there exists a 2-way 2-head nondeterministic FA M such that $L(M)$ is the accepted

language of some 2-way multi-head deterministic FA for any number of heads if and only if Dtape(log$n$) = Ndtape(log$n$).

PROPOSITION 4.4 [10]. *The set* $L_0 = \{M|M$ *is a deterministic FA and* $L(M) \neq \varnothing\}$ *is log-space complete in Ndtape(log*n*).* ■

Proof. That $L_0 \in$ Ndtape(log$n$) follows immediately from 4.3 since L_0 is the accepted language of some 2-way 2-head NDFA. The fact that for all $L \in$ Ndtape(log$n$), $L \leq_{\log} L_0$ follows from an argument analogous to that of 3.9 and is left to the reader. □

Stronger analogues of Propositions 3.7 and 3.8 hold for the PDA.

PROPOSITION 4.5.

(1) $L = \{M|M$ *is a deterministic PDA and* $\lambda \in L(M)\}$ *is the accepted language of some 2-way one-head deterministic PDA. Thus L is log-space complete in P.*

(2) $L' = \{M|M$ *is a nondeterministic PDA and* $L(M) \neq \varnothing\}$ *is the accepted language of some 2-way one-head nondeterministic PDA.* ■

Proof. A direct simulation of the constructions in the proofs of 3.8 and 3.9 shows that L and L' are recognizable by a 2-way 2-head deterministic *PDA* and a 2-way 2-head nondeterministic *PDA*, respectively. One of these two input tape heads is used only to verify that each consecutive pair of 1-way *PDA* moves (μ_1, μ_2) simulated is compatible, i.e. if $\mu_1 = (p, a, b, q, w_1)$ and $\mu_2 = (p', a', b', q', w_2)$, then $p' = q$. This head can be eliminated by writing the binary numeral q, which denotes the state of the automaton after executing μ_1, on the top of the 2-way device's pushdown store. Before simulating μ_2 the 2-way device verifies that p' equals q by popping its pushdown store. If $p' \neq q$ it rewrites q on top of its pushdown store employing the ability of its input tape head to move both ways. □[5]

Thus since

$$L_0 = \{M|M \text{ is a } PDA \text{ and } L(M) \neq \varnothing\} \underset{\substack{\log \\ n\log n(\text{space}+\text{time})}}{\geq}$$

all 2-way PDA languages by Theorem 4.1, L_0 is a hardest time and space 2-way PDA language analogous to the hardest time and space cfl in [31] and the hardest time and space csl's in [9].

5. On Rice's Theorem. The results of Sections 3 and 4 suggest a close relationship to Rice's Theorem for the re sets. As will be shown below, the results of Sections 3 and 4, Rice's Theorem for the re sets, and several other related results follow because for classes $\mathscr{C}$ of sufficiently powerful automata, grammars, etc., the membership problem for any language recognized or generated by elements of $\mathscr{C}$ is efficiently reducible to any nontrivial predicate on the languages accepted or generated by the elements of $\mathscr{C}$. First we state one version of Rice's Theorem for the re sets.

[5] A related result appears in [35].

THEOREM 5.1. *Let $\mathscr{P}$ be any nontrivial predicate on the re sets. Then $\{M|M$ is a Tm and $\mathscr{P}(L(M))$ is true$\}$ is not recursive.* ■

Definition 5.2. A *family of languages* $\hat{F}$ *over* $\{0, 1\}$ is a quadruple (Σ, C, f, F), where

(1) Σ is a finite alphabet;
(2) $C \subseteq \Sigma^*$, and
(3) $f\colon \Sigma^* \to 2^{\{0,1\}^*}$ such that
   (a) there exists a unique $L_0 \in 2^{\{0,1\}^*}$ such that $y \notin C$ implies $f(y) = L_0$ and
   (b) $F = f(C) \cup \{L_0\}$.

L_0 is frequently taken to be $\varnothing$. C corresponds to those strings over Σ satisfying the syntactic definition of a grammar or automaton.

$\hat{F}$ is recursive if the total function $g\colon \Sigma^* \times \{0, 1\}^* \to \{0, 1\}$ defined by

$$g(x, y) = \begin{cases} 1 & \text{if } y \in L_x \\ 0 & \text{if } y \notin L_x \end{cases}$$

is recursive. F is effective if the partial function $g\colon \Sigma^* \times \{0, 1\}^* \to \{0, 1\}$ defined by

$$g(x, y) = \begin{cases} 1 & \text{if } y \in L_x \\ 0 \text{ or undefined} & \text{if } y \notin L_x \end{cases}$$

is partial recursive. ■

Definition 5.3. A family of languages over $\{0, 1\}$ $\hat{F} = (\Sigma, C, f, F)$ is *effectively, p-effectively, log-effectively closed under intersection with single strings* if for all $y \in \Sigma^*$, there exists a Tm M_y, a deterministic polynominally time-bounded Tm M_y, or a deterministic log-space transducer M_y, respectively, such that M_y, given input $x \in \{0, 1\}^*$, outputs $z \in \{0, 1\}^*$, where $f(y) \cap \{x\} = f(z)$. Other effective, p-effective, and log-effective closures are defined analogously. ■

THEOREM 5.4.

(1) *Let $\hat{F} = (\Sigma, C, f, F)$ be a recursive or effective family of languages over $\{0, 1\}$ such that $\hat{F}$ is (a) effectively, (b) p-effectively, or (c) log-effectively closed under intersection with single strings, quotient with single strings on the left (or right) and concatenation. Let $\mathscr{P}$ be any nontrivial predicate on F such that $\mathscr{P}(\varnothing)$ is false. Then for all $y \in \Sigma^*$, $\{w|w \in \{0, 1\}^*$ and $w \in L_y = f(y)\} \leq_{eff} \{x|x \in \Sigma^*$ and $\mathscr{P}(L_x)$ is true$\}$, $\{w|w \in \{0, 1\}^*$ and $w \in L_y = f(y)\} \leq_{ptime} \{x|x \in \Sigma^*$ and $\mathscr{P}(L_x)$ is true$\}$, or $\{w|w \in \{0, 1\}^*$ and $w \in L_y = f(y)\} \leq_{log} \{x|x \in \Sigma^*$ and $\mathscr{P}(L_x)$ is true$\}$, respectively.*

(2) *Let $\hat{F} = (\Sigma, C, f, F)$ be a recursive or effective family of languages over $\{0, 1\}$ such that $\hat{F}$ is (a′) effectively, (b′) p-effectively, or (c′) log-effectively closed under the homomorphism $h\colon \{0, 1\}^* \to \{0, 1\}^*$ defined by $h(0) = 00$ and $h(1) = 01$, concatenation with 10, concatenation, and quotient with single string on the left (or right). Let $\mathscr{P}$ be any nontrivial predicate on F such that $\mathscr{P}(\varnothing)$ is false. Then for all $y \in \Sigma^*$, $\{w|w \in \{0, 1\}^*$ and $w \in L_y = f(y)\} \leq_{eff} \{x|x \in \Sigma^*$ and $\mathscr{P}(L_x)$ is true$\}$, $\{w|w \in \{0, 1\}^*$ and $w \in L_y = f(y)\} \leq_{ptime} \{x|x \in \Sigma^*$ and $\mathscr{P}(L_x)$ is true$\}$, or $\{w|w \in \{0, 1\}^*$ and $w \in L_y = f(y)\} \leq_{log} \{x|x \in \Sigma^*$ and $\mathscr{P}(L_x)\}$ is true, respectively.* ■

Proof. (1) Since $\mathscr{P}$ is nontrivial, there exists $L_0 \in F$ such that $\mathscr{P}(L_0)$ is true. Let $L_y = f(y) \in F$. For all $x \in \{0, 1\}^*$, let $L_x = L_y \cap \{x\}$, $L_x'' = x \backslash L_x$, and $L_x'' = L_x' \cdot L_0$.

But,

$$L_x'' = \begin{cases} \varnothing & \text{if } x \notin L_y \\ L_0 & \text{if } x \in L_y \end{cases}.$$

Thus $\mathscr{P}(L_x'')$ is true if and only if $x \in L_y$. But since $\hat{F}$ is effectively, p-effectively or log-effectively closed under intersection with single strings, quotient with single strings on the left, and concatenation, and these reducibilities are transitive, given a fixed y, any index for L_0 and x, an index y' for L_x'' can be calculated such that $L_y' = L_x''$ and the lower bounds of (1) hold.

(2) Similar to that of (1) with $L_x'' = h(x) \cdot 10 \backslash h(L_y) \cdot 10 \cdot L_0$. □

The standard encoding of Tm's satisfies the conditions of (1) and (2) of Theorem 5.4. Thus 5.1 is a corollary of 5.4. Similarly various classes of Tm's with time or space clocks satisfy the conditions of Theorem 5.4. Moreover, several classes of 1-way stack automata also satisfy 5.4. Since there are NP-complete languages that are recognizable by 1-way nondeterministic CSA, $RPDA$, $NESA$, etc., all nontrivial predicates on these language classes are NP-hard. The stronger results in Sections 3 and 4 are due to the ability to simulate the membership problem for 2-way devices with 1-way devices efficiently. We conjecture that 5.4 is also applicable to many other classes of languages intermediate between the cfl's and csl's such as the context-free programmed grammar languages of Rosenkrantz [36] and various matrix grammars. Theorem 5.4 also applies to the scattered context grammars of Greibach and Hopcroft [37], since there are languages recognizable by linear time bounded non-deterministic Tm's that are NP-complete [32] and all languages recognizable by linear time bounded nondeterministic Tm's are generated by scattered context grammars [37].

One corollary of Theorem 5.4 for the re sets follows.

COROLLARY 5.5. *Let $\mathscr{P}$ be any nontrivial predicate on the re sets over $\{0, 1\}$ such that $\mathscr{P}(\varnothing)$ is true, then $\{M | M$ is a Tm and $\mathscr{P}(L(M))$ is true$\}$ is not recursively enumerable.* ■

Proof. From the proof of 5.4, $\{M | M$ is a Tm and M halts on empty input$\} \leq_{\text{eff}} \{M | M$ is a Tm and $\mathscr{P}(L(M))$ is false$\}$. Thus $L = \{M | M$ is a Tm and M does not halt on empty input$\} \leq_{\text{eff}} \{M | M$ is a Tm and $\mathscr{P}(L(M))$ is true$\}$. But L is well-known not to be re. □

Again as in Sections 3 and 4 partial converses hold for 5.4. One such converse is presented below.

PROPOSITION 5.6.

(1) *Let $\hat{F} = (\Sigma, C, f, F)$ satisfy the conditions of (a) or (a') of Theorem 5.4. If there exists some nontrivial predicate $\mathscr{P}$ on F such that $\mathscr{P}$ is decidable, i.e., the set $\{x | x \in \Sigma^*$ and $\mathscr{P}(L_x)$ is true$\}$ is recursive, then $F \subseteq$ recursive sets over $\{0, 1\}$.*

(2) *Let* $\hat{F} = (\Sigma, C, f, F)$ *satisfy the conditions of* (*b*) *or* (*b'*) *of Theorem* 5.4. *If there exists some nontrivial predicate* $\mathscr{P}$ *on* F *such that* $\{x | x \in \Sigma^*$ *and* $\mathscr{P}(L_x)$ *is true*$\} \in P$, *then* $F \subseteq P$.

(3) *If* $\hat{F} = (\Sigma, C, f, F)$ *is effectively closed under intersection with single string and the set* $\{x | x \in \Sigma^*$ *and* $L_x \neq \varnothing\}$ *is recursive, then* $F \subseteq$ *recursive sets over* $\{0, 1\}$.

(4) *If* $\hat{F} = (\Sigma, C, f, F)$ *is p-effectively closed under intersection with single string and the set* $\{x | x \in \Sigma^*$ *and* $L_x \neq \varnothing\}$ *is in* P, *then* $F \subseteq P$. ■

Proof. (1) and (2) follow immediately from Theorem 5.4.

(3) Let $\hat{F}$ be effectively closed under intersection with single string. Let $y \in \Sigma^*$. For all $x \in \{0, 1\}^*$ $x \in f(y)$ if and only if $f(z) = f(y) \cap \{x\} \neq \varnothing$. But z is effectively calculable from y and x. Thus $f(y)$ and hence $F \leq_{\text{eff}} \{x | x \in \Sigma^*$ and $f(x) \neq \varnothing\}$.

(4) If $\hat{F}$ is p-effectively closed under intersection with single strings, then $F \leq_{\text{ptime}} \{x | x \in \Sigma^*$ and $f(x) \neq \varnothing\}$. □

We present one interesting corollary of Proposition 5.6.

COROLLARY 5.7. *The IO-macro languages* $\subseteq P$. ■

Proof. The emptiness problem for IO-macro grammars is easily seen to be decidable by some deterministic polynomially time bounded Tm. (See Fischer [18] or [19].) Moreover the IO-macro grammars are p-effectively closed under intersection with single string. Thus by (3) of 5.6, the IO-macro languages $\subseteq P$. The proof that the IO-macro grammars are p-effectively closed under intersection with single string follows from a detailed analysis of the proof of the Factor Theorem (Theorem 3.2.7, pp. 3–25) in [18] and will not be presented here. The key ideas are for a *fixed* IO-macro grammar G and any arbitrary string $w \in \Sigma^*$,

(i) the congruence relation $\equiv_w$ on Σ^* defined by $x \equiv_w y$ if and only if for all $u, v \in \Sigma^*$, $u \cdot x \cdot v = w$ if and only if $u \cdot y \cdot v = w$, has only $o(|w|^2)$ distinct equivalence classes;

(ii) the partition of $\Sigma^* \mathscr{E}_w$ induced by $\equiv_w$ is calculable deterministically in time bounded by a polynomial in $|w|$;

(iii) letting V be the set of variable symbols of G, the cardinality of the set of all functions from V into $\mathscr{E}_w$ is $o(|w|^{2|V|})$, and thus is polynomial in $|w|$; and

(iv) the cardinality of the set $\{(F, f_0; f_1, \cdots, f_{\rho(F)}) | F$ is a function symbol of G of arity $\rho(F)$ and $f_0, \cdots, f_{\rho(F)} \in \mathscr{E}_w\}$ is $o(|w|^{K_G})$, where K_G is a positive integer depending only upon G not w. □

5.7 is of interest since the OI-macro grammars are known to generate *NP*-complete languages.

Finally to further illustrate the utility of the results in this section, we present a new $o(n^3 \pi(\log n))$ time bounded algorithm for arbitrary 2-way *PDA* language recognition on a logarithmic cost RAM. Here π is a polynomial.

Algorithm 5.8. *Let* L *be a fixed* 2*-way PDA language. Let* M *be a fixed* 2*-way PDA such that* $L(M) = L$. *Let* $x = x_1 \cdots x_n$ *be an input to* M. *To test if* $x \in L(M) = L$ *the following steps suffice:*

(*i*) *Construct a* 1*-way PDA* M_x, *as described in the proofs of* 3.5 *and* 4.1, *such that* $L(M_x) \neq \varnothing$ *if and only if* $x \in L$.

(*ii*) *Convert* M_x *into an equivalent context-free grammar* G_x.

(iii) Test G_x for emptiness.

(iv) If $L(G_x) \neq \varnothing$, then $x \in L$. Otherwise $x \notin L$. ■

The time required to execute step 1 is $o(n\log n)$. Similarly the time required to convert M_x into an equivalent *CFG* G_x is $\leq K_M \cdot n^3 \log n$, where K_M is a constant depending only upon M not x. Finally the time to test G_x for emptiness is well-known to be $o(n\log n)$ on a logarithmic cost *RAM*. (See [16] for the details of the construction of G_x from M_x. Note that we may assume that M hence M_x for all x in Σ^* pushes at most two symbols on its stack at one time.) A slightly more efficient algorithm appears in [38]. Our point for presenting Algorithm 5.8 is that the ideas of this section can yield reasonably efficient algorithms as well as insights into what causes problems to be hard.

6. Conclusion. We feel that there are two especially significant results in this paper. First, one simple combinatorial idea underlies most of the recent work in the relationships of time and space complexity classes. No longer need the results in [8], [9], [10], [11], [12], [13], [14], [21], [32], [38], etc. be viewed as being either isolated or unrelated. Nor need the existence of hardest time or space languages for Ndtape ($\log n$), P, the 2-way PDA languages, etc. be viewed as combinatorial accidents that just happen to be true. Second, we have presented uniform nontrivial lower time and space complexity bounds for several nontrival classes of problems such as problems about stack or indexed languages. We have also presented strong evidence for the nonlinearity of every nontrivial predicate on the cfl's, when applied to the pushdown automata.

7. Acknowledgment. I wish to thank L. H. Landweber for a careful reading of portions of this paper. I also wish to thank the referees for numerous suggestions on how to improve the presentation of these results.

REFERENCES

[1] S. A. COOK, The complexity of theorem-proving procedures. *Proc. 3rd Annual ACM Symp. on Theory of Computing*, May, 1971.

[2] R. M. KARP, Reducibility among combinatorial problems, in *Complexity of Computer Computations*, (R. E. Miller and J. W. Thatcher, eds.), Plenum Press, New York, 1972.

[3] A. R. MEYER, Weak monadic second order of successor is not elementary recursive-preliminary report (unpublished).

[4] M. J. FISCHER and M. O. RABIN, Super-exponential complexity of Presburger arithmetic, *SIAM-AMS Proc., vol. 7*, Amer. Math. Soc., Providence, R.I., 1974.

[5] L. J. STOCKMEYER and A. R. MEYER, World problems requiring exponential time: preliminary report, *Proc. 5th Annual ACM Symp. on Theory of Computing*, May, 1973.

[6] H. B. HUNT, III and D. J. ROSENKRANTZ, Computational parallels between the regular and context-free languages, *Proc. 6th Annual ACM Symposium on Theory of Computing*, May, 1974.

[7] H. B. HUNT, III, D. J. ROSENKRANTZ, and T. G. SZYMANSKI, On the equivalence, containment, and covering problems for the regular and context-free languages (submitted for publication).

[8] W. J. SAVITCH, Relationships between nondeterministic and deterministic tape complexities, *J. Comput. System Sci.* 4 (1970), 177–192.

[9] J. HARTMANIS and H. B. HUNT, III, The lba problem and its importance in the theory of computing, *SIAM-AMS Proc.*, vol. 7, Amer. Math. Soc. Providence, R.I., 1974.

[10] N. JONES, Preliminary report: reducibility among combinatorial problems in log*n* space. *Proc. 7th Annual Princeton Conference on Information Sciences and Systems,* March, 1973.

[11] I. H. SUDBOROUGH, On tape-bounded complexity classes and multi-head finite automata, *Proc. 14th Annual IEEE Symp. on Switching and Automata Theory,* October, 1973.

[12] N. D. JONES and W. T. LAASER, Complete problems for deterministic polynomial time, *Proc. 6th Annual ACM Symp. on Theory of Computing,* May, 1974.

[13] S. A. COOK, An observation on time-storage tradeoff, *Proc. 5th Annual ACM Symp. on Theory of Computing,* May, 1973.

[14] S. A. COOK and R. SETHI, Storage requirements for deterministic polynomial time recognizable languages, *Proc. 6th Annual ACM Symp. on Theory of Computing,* May, 1974.

[15] A. V. AHO, Nested stack automata, *J. Assoc. Comput. Mach.* **16** (1969), 383–406.

[16] J. E. HOPCROFT and J. D. ULLMAN, *Formal Languages and Their Relation to Automata,* Addison-Wesley, Reading, Mass., 1969.

[17] A. V. AHO, Indexed grammars—an extension of context-free grammars, *J. Assoc. Comput. Mach.* **15** (1968), 647–671.

[18] M. J. FISCHER, "Grammars with Macro-like Productions," Ph.D. Dissertation, Harvard University, 1968.

[19] M. J. FISCHER, Grammars with macro-like productions, *Proc. 9th Annual IEEE Symp. on Switching and Automata Theory,* October, 1968.

[20] H. ROGERS, Jr., *Theory of Recursive Functions and Effective Computability,* McGraw-Hill, New York, 1967.

[21] W. C. ROUNDS, Complexity of recognition in intermediate-level languages, *Proc. 14th Annual IEEE Symp. on Switching and Automata Theory,* October, 1973.

[22] J. E. HOPCROFT and J. D. ULLMAN, Nonerasing stack automata, *J. Comput. System Sci.* **1** (1967), 166–186.

[23] S. A. COOK, Characterizations of pushdown machines in terms of time-bounded computers, *J. Assoc. Comput. Mach.* **18** (1971), 4–18.

[24] O. H. IBARRA, Characterizations of some tape and time complexity classes of Turing machines in terms of multi-head and auxiliary stack automata, *J. Comput. System Sci.* **5** (1971), 88–117.

[25] F. C. HENNIE and R. E. STEARNS, Two-tape simulation of multi-tape Turing macnines, *J. Assoc. Comput. Mach,* **13** (1966), 533–546.

[26] O. H. IBARRA, A note concerning nondeterministic tape complexities. *J. Assoc. Comput. Mach.* **19** (1972), 608–612.

[27] J. I. SEIFERAS, M. J. FISCHER, and A. R. MEYER, Refinements of the nondeterministic time and space hierarchies, *Proc. 14th Annual IEEE Symp. on Switching and Automata Theory,* October, 1973.

[28] W. C. ROUNDS, Tree-oriented proofs of some theorems on context-free and indexed languages, *Proc. 2nd Annual ACM Symp. on Theory of Computing,* May, 1970.

[29] W. C. ROUNDS, Mappings and grammars on trees, *Math. Systems Theory* **4** (1970), 257–287.

[30] V. RAJLICH, Absolutely parallel grammars and two-way deterministic finite-state transducers, *Proc. 3rd Annual ACM Symp. on Theory of Computing,* May 1971.

[31] S. A. GREIBACH, The hardest context-free language, *SIAM J. Comput.* **2** (1973), 304–310.

[32] H. B. HUNT, III, On the time and tape complexity of languages I, *Proc. 5th Annual ACM Symp. on Theory of Computing,* May, 1973.

[33] H. B. HUNT, III, "On the Time and Tape Complexity of Languages," Ph.D. Dissertation, Cornell University, 1973.

[34] R. BOOK, On languages accepted in polynomial time, *SIAM J. Comput.* **1** (1972).

[35] Z. GALIL, Two-way deterministic pushdown automaton languages and some open problems in the theory of computation, *Proc. 15th Annual IEEE Symp. on Switching and Automata Theory,* October, 1974.

[36] D. J. ROSENKRANTZ, Programmed grammars and classes of formal languages, *J. Assoc. Comput. Mach.* **16** (1969), 107–131.

[37] S. A. GREIBACH and J. E. HOPCROFT, Scattered context grammars, *J. Comput. System Sci.* **3** (1969), 233–247.

[38] A. V. AHO, J. E. HOPCROFT, and J. D. ULLMAN, Time and tape complexity of pushdown automaton languages, *Information and Control* **13** (1968), 186–206.

[39] H. B. HUNT, III, "On the Complexity of Finite, Pushdown, and Stack Automata," MRC Technical Summary Report No. 1504, Mathematics Research Center, University of Wisconsin-Madison, 1975.

(Received 28 January 1974, and in revised form 10 February 1975 and 8 August 1975)