SIAM J. COMPUT.
Vol. 1, No. 4, December 1972

ON LANGUAGES ACCEPTED IN POLYNOMIAL TIME*

RONALD V. BOOK†

Abstract. The family NP (P) of languages accepted by nondeterministic (deterministic) Turing machines operating in polynomial time is distinct from many well-known families of languages defined by tape-bounded or time-bounded Turing machines. In particular, it is shown that NP (P) is not equal to the family of context-sensitive languages, the family of languages accepted by deterministic linear bounded automata, or the family of languages accepted by deterministic Turing machines operating within exponential time.

Key words. Automata-based complexity, complexity classes of formal languages, polynomial time, time-bounded machines, tape-bounded machines.

Introduction. In automata-based complexity much effort has been expended in attempting to answer questions regarding time–tape trade-offs, deterministic simulation of nondeterministic machines, etc. Recently nondeterministic Turing machines operating in polynomial time have been studied in order to classify the "relative complexity" of certain (nonautomata theoretic) problems [6], [12]. The relationship between the family NP (P) of languages accepted by nondeterministic (deterministic) Turing machines operating in polynomial time and other complexity classes of languages is of interest in order to further classify these problems.

It has been shown [6] that any language in NP is "polynomially reducible" to various languages accepted by deterministic linear bounded automata and hence by deterministic Turing machines operating in exponential time. Here we compare P and NP to many well-known families of languages defined by tape-bounded or time-bounded Turing machines and show that these families are distinct.

In § 1 we compare P and NP with certain tape-bounded classes and in § 2 with certain time-bounded classes. In § 3 we give quite different proofs of some results from § 1 and 2, and in § 4 we provide an outline of another approach to these results.

1. We begin by defining the families of languages under investigation. The functions f used to bound the amount of time or tape used in a Turing machine's computation are such that for all $x, y \geqq 0, f(x) \geqq x$ and $f(x) + f(y) \leqq f(x + y)$. Such functions are nondecreasing. Further, most functions used are "self-computable" in the sense that there is a Turing machine M_1 which upon input w runs for precisely $f(|w|)$ steps and halts, and a machine M_2 which upon input w marks precisely $f(|w|)$ consecutive tape squares and halts.[1] (See [2].)

DEFINITION. Let f be a bounding function. For a Turing acceptor M, $L(M)$ is a set of strings accepted by M.

* Received by the editors May 23, 1972.

† Center for Research in Computing Technology, Division of Engineering and Applied Physics, Harvard University, Cambridge, Massachusetts 02138. This research was supported in part by the National Aeronautics and Space Administration under Grant NGR-22-007-176 and by the National Science Foundation under Grant GJ-30409.

[1] For a string w, $|w|$ is the length of w.

(i) A multitape Turing acceptor M operates within time bound f if for each input string w accepted by M, every accepting computation of M on w has no more than $\max(|w|, f(|w|))$ steps.

(ii) Define $\text{NTIME}(f) = \{L(M) | M$ is a nondeterministic multitape Turing acceptor which operates within time bound $f\}$ and $\text{DTIME}(f) = \{L(M) | M$ is a deterministic multitape Turing acceptor which operates within time bound $f\}$.

(iii) A multitape Turing acceptor M operates within tape bound f if for each input string w accepted by M, every accepting computation of M on w visits no more than $\max(|w|, f(|w|))$ tape squares on any one of its storage tapes.

(iv) Define $\text{NTAPE}(f) = \{L(M) | M$ is a nondeterministic Turing acceptor which operates within tape bound $f\}$ and $\text{DTAPE}(f) = \{L(M) | M$ is a deterministic Turing acceptor which operates within tape bound $f\}$.

The specific families of languages which we study in this paper can be formally defined using the above notation.

DEFINITION. The family of languages accepted by nondeterministic Turing machines which operate in polynomial time is $\text{NP} = \bigcup_{k=1}^{\infty} \text{NTIME}(x^k)$. The family of languages accepted by deterministic Turing machines which operate in polynomial time is $\text{P} = \bigcup_{k=1}^{\infty} \text{DTIME}(x^k)$.

In [6] the family P is referred to as $\mathscr{L}_*$ and the family NP as $\mathscr{L}_*^+$.

There are several other families of languages to which we frequently refer.

DEFINITION. Let i be the identity function, $i(x) = x$.

(i) The family of context-sensitive languages is the family $\text{CS} = \text{NTAPE}(i)$.

(ii) The family of languages accepted by deterministic linear bounded automata is $\text{DLBA} = \text{DTAPE}(i)$.

(iii) The family of languages accepted by deterministic Turing machines which operate in polynomial storage is

$$\mathscr{T} = \bigcup_{k=1}^{\infty} \text{DTAPE}(x^k).$$

(iv) The family of quasi-realtime languages is the family $\text{Q} = \text{NTIME}(i)$ [1].

The family DLBA is the family of languages whose characteristic functions are in $\mathscr{E}^2$ (where $\mathscr{E}^2$ is the subclass of primitive recursive functions defined by Grzegorczyk). Sometimes this family is referred to as the family of deterministic $\mathscr{E}^2$ relations or $\mathscr{E}_*^2$. Then the family CS is referred to as the family of nondeterministic $\mathscr{E}^2$ relations.

Recall that for any function f, $\text{NTAPE}(f) \subseteq \text{DTAPE}(f^2)$ [13]; thus,

$$\mathscr{T} = \bigcup_{k=1}^{\infty} \text{NTAPE}(x^k).$$

Further, for any real numbers, $1 \leqq r < s$, $\text{DTAPE}(x^r) \subsetneq \text{DTAPE}(x^s)$ [14]. Hence, there is no real number t such that $\mathscr{T} = \text{DTAPE}(x^t)$ or $\mathscr{T} = \text{NTAPE}(x^t)$. These facts are helpful in establishing our results comparing P and NP to various families defined by tape-bounded machines.

THEOREM 1. *If there exists a real number* $r \geqq 1$ *such that* $\text{DTAPE}(x^r) \subseteq P$, *then* $\mathscr{T} = \text{P} = \text{NP}$. *Similarly, if there exists a real number* $r \geqq 1$ *such that* $\text{DTAPE}(x^r) \subseteq \text{NP}$, *then* $\mathscr{T} = \text{NP}$.

Proof. We give the proof for the first part, the proof for the second part being identical.

First, note that $P \subseteq NP \subseteq \mathscr{T}$, since a machine can use no more tape than it does time. Second, suppose $r \geqq 1$ is such that $DTAPE(x^r) \subseteq P$. Without loss of generality, assume that r is rational, say $r = p/q$, where $p \geqq q \geqq 1$ are integers. To show that $\mathscr{T} \subseteq P$ it is sufficient to show that $DTAPE(x^s) \subseteq P$ for $s = 2qr = 2p$.

Let M_1 be a deterministic Turing machine which operates within tape bound x^s. Let $L_1 = L(M_1)$. If Σ is a finite alphabet such that $L_1 \subseteq \Sigma^*$, let c be a new symbol, $c \notin \Sigma$. Let $L_2 = \{wc^m | w \in L_1, |wc^m| = |w|^s\}$. Since M_1 operates within tape bound x^s, it is clear that one can construct a deterministic linear bounded automaton M_2 from M_1 such that $L(M_2) = L_2$. Thus, $L_2 \in DLBA \subseteq DTAPE(x^r) \subseteq P$. But if $L_2 \in P$, then $L_1 \in P$ since the difference (with respect to time) between recognizing L_1 and L_2 is simply the computation of a polynomial. Hence, $DTAPE(x^s) \subseteq P$.

Since for any real number $r \geqq 1$, $DTAPE(x^r) \subseteq NTAPE(x^r)$, the result of Theorem 1 will hold if $NTAPE(x^r)$ is substituted for $DTAPE(x^r)$.

THEOREM 2. *There is no pair (r, s) of real numbers, $1 \leqq r \leqq s$, such that* $DTAPE(x^r) \subseteq P \subseteq DTAPE(x^s)$ *or* $NTAPE(x^r) \subseteq P \subseteq NTAPE(x^s)$. *Similarly, there is no pair (r, s) of real numbers, $1 \leqq r \leqq s$, such that* $DTAPE(x^r) \subseteq NP \subseteq DTAPE(x^s)$ *or* $NTAPE(x^r) \subseteq NP \subseteq NTAPE(x^s)$.

Proof. If $DTAPE(x^r) \subseteq P$, then by Theorem 1, $\mathscr{T} = P$. Hence, if $P \subseteq DTAPE(x^s)$, then $\mathscr{T} = DTAPE(x^s)$. But as noted above $DTAPE(x^s) \subsetneq \mathscr{T}$. Thus, either $DTAPE(x^r) \not\subseteq P$ or $P \not\subseteq DTAPE(x^s)$. The proof for NP is identical. Since $DTAPE(x^r) \subseteq NTAPE(x^r)$ and $NTAPE(x^s) \subseteq DTAPE(x^{2s})$, if $NTAPE(\cdot)$ is substituted for $DTAPE(\cdot)$ throughout, then the results still hold.

COROLLARY. *For any real number $r \geqq 1$,*

(i) $P \neq DTAPE(x^r)$,

(ii) $P \neq NTAPE(x^r)$,

(iii) $NP \neq DTAPE(x^r)$,

(iv) $NP \neq NTAPE(x^r)$.

In particular, $P \neq DLBA$, $P \neq CS$, $NP \neq DLBA$, *and* $NP \neq CS$.

It is not known whether $DLBA \subseteq P$ or $DLBA \subseteq NP$. From Theorem 2, we see that if for some real number s, $P \subseteq DTAPE(x^s)$ or $NP \subseteq DTAPE(x^s)$, then there is no real number $r \geqq 1$ such that $DTAPE(x^r) \subseteq P$ or $DTAPE(x^r) \subseteq NP$ (or $NTAPE(x^r) \subseteq P$ or $NTAPE(x^r) \subseteq NP$).

2. We turn to comparing P and NP with families defined by time-bounded machines.

The first result is analogous to Theorem 1.

THEOREM 3. *If there exists a real number $p > 1$ such that* $DTIME(p^x) \subseteq NP$, *then*

$$\bigcup_{k=1}^{\infty} DTIME(k^x) \subsetneq NP = \mathscr{T} = \bigcup_{j=1}^{\infty} DTIME(2^{x^j}).$$

COROLLARY. *There is no real number $p > 1$ such that* $DTIME(p^x) = NP$. *Further,* $\bigcup_{k=1}^{\infty} DTIME(k^x) \neq NP$.

Note that $\bigcup_{k=1}^{\infty} DTIME(k^x)$ is the family of languages accepted by deterministic Turing machines operating in exponential time. In [5] it is shown that $\bigcup_{k=1}^{\infty} DTIME(k^x)$ is precisely the family of languages accepted by (deterministic or nondeterministic) auxiliary pushdown machines which operate within tape

bound $i(x) = x$. Thus, $CS \subseteq \bigcup_{k=1}^{\infty} DTIME(k^x)$. It is not known whether $NP \subseteq \bigcup_{k=1}^{\infty} DTIME(k^x)$.

To obtain Theorem 3 we establish two lemmas. In both cases the proof is similar to that of Theorem 1.

LEMMA 1. *If there exists a real number* $p > 1$ *such that* $DTIME(p^x) \subseteq NP$, *then* $DTIME((p^2)^x) \subseteq NP$.

Proof. Suppose such a p exists. Let $L_1 \in DTIME((p^2)^x)$. Let M_1 be a deterministic Turing machine such that $L(M_1) = L_1$ and M_1 operates within time bound $(p^2)^x$. Let Σ be a finite alphabet such that $L_1 \subseteq \Sigma^*$ and let c be a new symbol, $c \notin \Sigma$.

If $L_2 = \{wc^{|w|} | w \in L_1\}$, then one can construct a deterministic Turing machine M_2 from M_1 such that $L(M_2) = L_2$ and such that the running time of M_2 is bounded by $(p^2)^{|w|}$. Since $|wc^{|w|}| = 2|w|$, this means that M_2 operates within time bound p^x. Hence, $L_2 = L(M_2) \in DTIME(p^x)$. Since $DTIME(p^x) \subseteq NP$, this means that $L_2 \in NP$. But if $L_2 \in NP$, then $L_1 \in NP$ since the difference (with respect to time) between recognizing L_1 and L_2 is simply the computation of a polynomial. Hence, $DTIME((p^2)^x) \subseteq NP$.

LEMMA 2. *If* $\bigcup_{k=1}^{\infty} DTIME(k^x) \subseteq NP$, *then* $\bigcup_{j=1}^{\infty} DTIME(2^{x^j}) \subseteq NP$.

Proof. If $L_1 \in \bigcup_{j=1}^{\infty} DTIME(2^{x^j})$, then there is a deterministic Turing machine M_1 and a constant j such that $L(M_1) = L_1$ and M_1 operates within time bound 2^{x^j}. Let Σ be a finite alphabet such that $L_1 \subseteq \Sigma^*$ and let c be a new symbol, $c \notin \Sigma$.

If

$$L_2 = \{wc^m | w \in L_1, |wc^m| = |w|^j\},$$

then one can construct a deterministic Turing machine M_2 such that $L(M_2) = L_2$ and such that the running time of M_2 is bounded by $2^{|w|^j}$. Since $|wc^m| = |w|^j$, this means that M_2 operates within time bound 2^x. Hence,

$$L_2 = L(M_2) \in DTIME(2^x) \subseteq \bigcup_{k=1}^{\infty} DTIME(k^x).$$

Since $\bigcup_{k=1}^{\infty} DTIME(k^x) \subseteq NP$, this means that $L_2 \in NP$. But if $L_2 \in NP$, then $L_1 \in NP$ since the difference between recognizing L_1 and L_2 is simply the computation of a polynomial. Hence $\bigcup_{j=1}^{\infty} DTIME(2^{x^j}) \subseteq NP$.

Proof of Theorem 3. Suppose there exists a real number $p > 1$ such that $DTIME(p^x) \subseteq NP$. By use of Lemma 1, for any integer $q > 1$, $DTIME((p^q)^x) \subseteq NP$. Thus,

$$\bigcup_{k=1}^{\infty} DTIME(k^x) \subseteq NP.$$

By Lemma 2, this means that $\bigcup_{j=1}^{\infty} DTIME(2^{x^j}) \subseteq NP$. As noted in § 1, $NP \subseteq \mathscr{T}$. It is clear that

$$\mathscr{T} \subseteq \bigcup_{k=1}^{\infty} \bigcup_{j=1}^{\infty} DTIME(k^{x^j}).$$

But

$$\bigcup_{k=1}^{\infty} \bigcup_{j=1}^{\infty} DTIME(k^{x^j}) = \bigcup_{j=1}^{\infty} DTIME(2^{x^j}),$$

so we have

$$\text{NP} = \mathscr{T} = \bigcup_{j=1}^{\infty} \text{DTIME}(2^{x^j}).$$

From [9], we have

$$\bigcup_{k=1}^{\infty} \text{DTIME}(k^x) \subsetneq \bigcup_{j=1}^{\infty} \text{DTIME}(2^{x^j}).$$

Our other result with respect to time is a statement of two necessary and sufficient conditions for P and NP to be the same.

THEOREM 4. *The following are equivalent*:

(a) P = NP;

(b) P *is closed under nonerasing homomorphic mappings*;

(c) Q $\subseteq$ P.

Proof. It is clear that the family NP is closed under nonerasing homomorphic mappings so that (a) implies (b). In [1] it is shown that $L \in \text{Q}$ if and only if there exist $L_2 \in \text{DTIME(i)} \subset \text{P}$ and a nonerasing homomorphism h such that $L = \{h(w) | w \in L_2\}$. Thus, (b) implies (c).

To see that (c) implies (a), again we use an argument similar to the proof of Theorem 1. If $L_1 \in \text{NP} = \bigcup_{j=1}^{\infty} \text{NTIME}(x^j)$, then there exist an integer $k \geqq 1$ and a nondeterministic Turing machine M_1 such that $L(M_1) = L_1$ and M_1 operates within time bound x^k. Let Σ be a finite alphabet such that $L_1 \subseteq \Sigma^*$ and let c be a new symbol, $c \notin \Sigma$. Let

$$L_2 = \{wc^m | w \in L_1, |wc^m| = |w|^k\}.$$

Since M_1 operates within time bound x^k, one can construct a machine M_2 from M_1 such that $L(M_2) = L_2$ and M_2 operates within time bound $i(x) = x$. Thus, $L_2 \in \text{Q}$ so $\text{Q} \subseteq \text{P}$ implies $L_2 \in \text{P}$. But if $L_2 \in \text{P}$, then clearly $L_1 \in \text{P}$. Hence, NP $\subseteq$ P. Since P $\subseteq$ NP, we have NP = P.

Those familiar with formal language theory may note that condition (b) is equivalent to the assertion that P is an abstract family of languages (AFL).

3. Now we give a totally different proof of part of the corollary to Theorem 2 and the corollary to Theorem 3.

PROPOSITION 1. *For any rational number* $r \geqq 1$, $\text{NP} \neq \text{DTAPE}(x^r)$, $\text{NP} \neq \text{NTAPE}(x^r)$, *and* $\text{NP} \neq \bigcup_{k=1}^{\infty} \text{DTIME}(k^{x^r})$.

Proof. By results in [2], [3], $\text{DTAPE}(x^r)$, $\text{NTAPE}(x^r)$, and $\bigcup_{k=1}^{\infty} \text{DTIME}(k^{x^r})$ are principal AFLs.[2] We show that NP is not a principal AFL, thus obtaining the result.

In [7] it is shown that for any real numbers r, s, if $1 \leqq r < s$, then $\text{NTIME}(x^r) \subsetneq \text{NTIME}(x^s)$. Hence $\text{NTIME}(x)$, $\text{NTIME}(x^2)$, $\text{NTIME}(x^3) \cdots$ forms an infinite heirarchy of families of languages. By results in [2], each $\text{NTIME}(x^k)$ is a principal AFL. Thus by results in [8], $\text{NP} = \bigcup_{k=1}^{\infty} \text{NTIME}(x^k)$ is not a principal AFL.

[2] An abstract family of languages (AFL) is a family of languages closed under the following operations: union, concatenation, Kleene +, intersection with regular sets, nonerasing homomorphic mappings (a homomorphism $h: \Sigma^* \to \Delta^*$ is nonerasing if $h(w) = e$ implies $w = e$), and inverse homomorphic mappings. A family of languages is a principal AFL if it is the smallest AFL containing some given language [8].

COROLLARY. $\mathrm{NP} \subsetneq \bigcup_{k=1}^{\infty} \mathrm{NTIME}(k^x)$.

Proof. By results in [2], $\bigcup_{k=1}^{\infty} \mathrm{NTIME}(k^x)$ is a principal AFL. As shown above, NP is not a principal AFL so $\mathrm{NP} \neq \bigcup_{k=1}^{\infty} \mathrm{NTIME}(k^x)$.

The family $\bigcup_{k=1}^{\infty} \mathrm{NTIME}(k^x)$ of languages accepted by nondeterministic Turing machines operating in exponential time is the class of spectra of formulas of first order logic with equality [11].

Results in [2], [3], [8] show that NP cannot be the same as many families which have been studied in automata and formal language theory since NP is not a principal AFL.

4. The results established in §§ 1–3 were discovered by examining the application of "bounded erasing operators" to the families CS, DLBA, P, NP, etc. In particular, we have shown that NP is closed under "polynomial erasing" where neither CS nor DLBA has this property, the closure of either CS or DLBA under polynomial erasing being $\mathscr{T}$. In this section we define some of these notions and rephrase some of the results of [2], [4] in terms of these families. It is hoped that these techniques are applicable to other questions concerning complexity classes of languages, and are helpful in suggesting possible results to others.

We begin by defining the notion of a bounded erasing operator [2].

DEFINITION. If $h: \Sigma^* \to \Delta^*$ is a homomorphism, $L \subseteq \Sigma^*$, and f is a function such that for some $k > 0$ and all $w \in L$, $|w| \leqq kf(|h(w)|)$, then h is f-bounded on L. For any family $\mathscr{L}$ of languages[3] and any function f, the *image of* $\mathscr{L}$ *under* f*-bounded erasing* is $H_f[\mathscr{L}] = \{h(L) \mid L \in \mathscr{L}$ and h is a homomorphism which is f-bounded on $L\}$.

The first use of the bounded erasing operators is shown in the following "representation" results from [2].

PROPOSITION 2. *For any function* f,

$$\mathrm{NTIME}(f) = H_f[\mathrm{Q}], \quad \mathrm{NTAPE}(f) = H_f[\mathrm{CS}], \text{ and } \mathrm{DTAPE}(f) = H_f[\mathrm{DLBA}].$$

Hence, $\mathrm{NTIME}(f) \subseteq \mathrm{DTAPE}(f)$.

In [4] the composition of operators H_f and H_g is studied. Sufficient conditions on f, g, and $\mathscr{L}$ such that $H_g[H_f[\mathscr{L}]] = H_{f \circ g}[\mathscr{L}]$ (where $(f \circ g)(x) = f(g(x))$) are established. Applied to the families of languages and bounding functions studied here, the following results become important.

PROPOSITION 3. *For rational numbers* $r, s \geqq 1$, *if* $f(x) = x^r$ *and* $g(x) = x^s$ *are bounding functions, then*

(i) $H_g[H_f[\mathrm{Q}]] = H_g[\mathrm{NTIME}(f)] = \mathrm{NTIME}(f \circ g)$;

(ii) $H_g[H_f[\mathrm{CS}]] = H_g[\mathrm{NTAPE}(f)] = \mathrm{NTAPE}(f \circ g)$;

(iii) $H_g[H_f[\mathrm{DLBA}]] = H_g[\mathrm{DTAPE}(f)] = \mathrm{DTAPE}(f \circ g)$.

A family $\mathscr{L}$ is closed under "polynomial erasing" if for any real number $r > 1$, if $f(x) = x^r$, then $H_f[\mathscr{L}] \subseteq \mathscr{L}$. As shown in [4], there is no real number $s \geqq 1$ such that $\mathrm{NTAPE}(x^s)$ or $\mathrm{DTAPE}(x^s)$ is closed under H_f, i.e., $\mathscr{L} \subsetneq H_f[\mathscr{L}]$. On the other hand, the proof of Theorem 1 shows that NP is closed under polynomial erasing. The proof of Theorem 3 shows that $\bigcup_{k=1}^{\infty} \mathrm{DTIME}(k^x)$ is not closed under polynomial erasing.

[3] We assume that (i) if $L \in \mathscr{L}$, then there is a finite alphabet Σ such that $L \subseteq \Sigma^*$, and (ii) there exists $L \in \mathscr{L}$ such that $L \neq \emptyset$.

In [4] the results on bounded erasing are used to show the following result announced in [10]: for any pair of real numbers (r, s) such that $1 \leqq r < s$, $\mathrm{NTAPE}(x^r) \subsetneq \mathrm{NTAPE}(x^s)$.

The results in § 2 show that NP is the smallest AFL which is closed under polynomial erasing and contains P. If NP is simply the smallest AFL containing P, then one can extend results in § 2 and § 3 to show that $\mathrm{NP} \subsetneq \bigcup_{k=1}^{\infty} \mathrm{DTIME}(k^x)$.

The results stated in this section are "representation" and "translation" results. In particular, Proposition 3 is similar to the translation results of [3], [4], [7], [10], [13].

The bounded-erasing operators are "algebraic" and not "measure dependent." It would be interesting to know whether the type of "translation" results obtained here can be found in abstract complexity theory.

Acknowledgment. It is a pleasure to thank Stephen Cook for his criticism of a previous version of this paper.

REFERENCES

[1] R. Book and S. Greibach, *Quasi-realtime languages*, Math. Systems Theory, 4 (1970), pp. 97–111.

[2] R. Book, S. Greibach and B. Wegbreit, *Time- and tape-bounded Turing acceptors and AFLs*, J. Comput. System Sci., 4 (1970), pp. 606–621.

[3] R. Book, S. Greibach, O. Ibarra and B. Wegbreit, *Tape-bounded Turing acceptors and principal AFLs*, Ibid., 4 (1970), pp. 622–625.

[4] R. Book and B. Wegbreit, *A note on AFLs and bounding erasing*, Information and Control, 19 (1971), pp. 18–29.

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

[6] ———, *The complexity of theorem-proving procedures*, Proc. Third ACM Symposium on Theory of Computing, 1971, pp. 151–158.

[7] ———, *A hierarchy for nondeterministic time complexity*, Proc. Fourth ACM Symposium on Theory of Computing, 1972, pp. 187–192.

[8] S. Ginsburg and S. Greibach, *Principal AFL*, J. Comput. System Sci., 4 (1970), pp. 308–338.

[9] J. Hartmanis and R. Stearns, *On the computational complexity of algorithms*, Trans. Amer. Math. Soc., 117 (1965), pp. 285–306.

[10] O. Ibarra, *A note concerning nondeterministic tape complexities*, submitted for publication. Also, *Abstract* 71T-C22, Notices Amer. Math. Soc., 18 (1971), p. 165.

[11] N. Jones and A. Selman, *Turing machines and the spectra of first-order formulas with equality*, Proc. Fourth ACM Symposium on Theory of Computing, 1972, pp. 157–167.

[12] R. Karp, *Reducibility among combinatorial problems*, Proc. IBM Symposium on Computational Complexity, to appear.

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

[14] R. Stearns, J. Hartmanis and P. Lewis, *Hierarchies of memory limited computations*, Conf. Record IEEE Sixth Annual Symposium on Switching Circuit Theory and Logical Design, 1965, pp. 179–190.