Comparing Complexity Classes*

RONALD V. BOOK†

Center for Research in Computing Technology, Harvard University,
Cambridge, Massachusetts 02138

Received January 20, 1973

Complexity classes defined by time-bounded and space-bounded Turing acceptors are studied in order to learn more about the cost of deterministic simulation of nondeterministic processes and about time-space tradeoffs. Here complexity classes are compared by means of reducibilities and class-complete sets. The classes studied are defined by bounds of the order \( n, n^k, 2^n, 2^{nk} \). The results do not establish the existence of possible relationships between these classes; rather, they show the consequences of such relationships, in some cases offering circumstantial evidence that these relationships do not hold and that certain pairs of classes are set-theoretically incomparable.

INTRODUCTION

Certain long-standing open questions in automata-based complexity have resurfaced recently due to the work by Cook [9] and Karp [17] on efficient reducibilities among combinatorial problems. In particular, questions regarding time-space tradeoffs and the cost of deterministic simulation of nondeterministic machines have received renewed attention. The purpose of this paper is to study relationships between certain classes of languages accepted by time- and space-bounded Turing machines in order to learn more about these questions.

The questions of time-space tradeoffs and deterministic simulation of nondeterministic processes can be studied on an ad hoc basis, e.g., a particular problem can be solved via a nondeterministic process and then an efficient deterministic process might be shown to realize the result. If the problem is “complete” for a class, then one may obtain information regarding the mass problem for that class. But results concerning a complete problem must be interpreted in terms of the reduction functions before they can be applied to the mass problem. For example, the completeness of the satisfiability problem for CNF formulas in the class of nondeterministic polynomial

* This research was supported in part by the National Science Foundation under Grant NSF GJ-30409.
† Current address: Department of Computer Science, Yale University, New Haven, CT 06520.
time problems and the fact that satisfiability can be determined in linear space do not imply that all nondeterministic polynomial time problems can be solved in linear space, only that they can be solved in polynomial space.

In this paper certain classes are compared by means of reducibilities, “class-complete” sets, and techniques of automata and formal language theory. The classes studied are specified by time- and space-bounded multitape Turing acceptors which operate deterministically or nondeterministically within bounds that are subelementary, specifically time bounds of the form $n^k$, $2^{cn}$, and $2^{n^k}$, and space bounds of the form $n^k$. The principal results show the existence of complete sets for certain classes with respect to easy-to-compute reducibilities. The implications of one class containing a set which is complete for another class are then developed.

The results do not show that certain classes are equal or that one is included in another. However, in some cases, it is shown that two classes are not equal. Some of these inequalities and some of these implications should be taken as circumstantial evidence for the nonexistence of certain relationships between complexity classes, and for the set-theoretic incomparability of certain pairs of classes.

Let us note that the motivation for this study is not only from the long-standing open questions of automata-based computational complexity but also from the fact that many of the sets shown in [9] and [17] to be “polynomial complete” happen to be sets accepted by deterministic Turing machines which operate within linear space bounds.

In Section 1 we state basic definitions and establish notation. Known results are summarized in Section 2. The principal results on complete sets are established in Section 3 while certain translation results are given in Section 4.

In this paper a familiarity with concepts from automata and formal language theory and the basic questions of computational complexity are assumed. We do not define specific models in detail because the results are essentially independent of the many minor variations in the definition of a Turing machine found in the literature.

1. Preliminaries

Unless specified to the contrary, the functions $f$ used to bound the amount of time or tape used in a (multitape, deterministic or nondeterministic) Turing machine’s computation are such that for all $n, m \geq 0$, $f(n) + f(m) \leq f(n + m)$. Such functions are nondecreasing. Further, the functions are “self-computable” (“linearly honest”) in the sense that there is a deterministic Turing machine $M_1$ which upon input $w$ runs for precisely $f(|w|)$ steps and halts, and a deterministic machine $M_2$ which upon input $w$ marks precisely $f(|w|)$ consecutive tape squares and halts.\(^1\)

\(^1\) For a string $w$, $|w|$ is the length of $w$. 

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

(i) A multitape Turing acceptor $M$ operates within time bound $f$ if for each input string $w$ accepted by $M$, there exists an accepting computation of $M$ on $w$ which has no more than $f(|w|)$ steps. Define $\text{NTIME}(f) = \{L(M) \mid M$ is a nondeterministic multitape Turing acceptor which operates within time bound $f\}$ and $\text{DTIME}(f) = \{L(M) \mid M$ is a deterministic multitape Turing acceptor which operates within time bound $f\}$.

(ii) A multitape Turing acceptor $M$ operates within space bound $f$ if for each input string $w$ accepted by $M$, there exists an accepting computation of $M$ on $w$ which visits no more than $f(|w|)$ tape squares on any one of its storage tapes. Define $\text{NSPACE}(f) = \{L(M) \mid M$ is a nondeterministic Turing acceptor which operates within space bound $f$ and $\text{DSPACE}(f) = \{L(M) \mid M$ is a deterministic Turing acceptor which operates within space bound $f\}$.

Notation. For any bounding function $f$ and any $k \geq 1$, the function $k^f(n)$ is defined for all $n$ by $k^f(n) = k^f(n)$, and the function $f^k$ is defined for all $n$ by $f^k(n) = (f(n))^k$.

Some of the classes we consider are defined by taking a union of complexity classes where the union is taken over a class of bounding functions. We adopt a simple notation for the most frequently studied classes. This particular notation is chosen with hopes of making uniform the entire notational scheme.²

Notation.

(i) Let $\text{DTIME}(\text{poly}) = \bigcup_{k=1}^{\infty} \text{DTIME}(n^k)$, so that $\text{DTIME}(\text{poly})$ is the class of sets accepted by deterministic Turing acceptors which operate in polynomial time.

(ii) Let $\text{NTIME}(\text{poly}) = \bigcup_{k=1}^{\infty} \text{NTIME}(n^k)$, so that $\text{NTIME}(\text{poly})$ is the class of sets accepted by nondeterministic Turing acceptors which operate in polynomial time.

(iii) Let $\text{DTIME}(2^{\text{lin}}) = \bigcup_{k=1}^{\infty} \text{DTIME}(k^n) = \bigcup_{c>0} \text{DTIME}(2^{cn})$.

(iv) Let $\text{NTIME}(2^{\text{lin}}) = \bigcup_{k=1}^{\infty} \text{NTIME}(k^n) = \bigcup_{c>0} \text{NTIME}(2^{cn})$.

(v) Let $\text{DSPACE}(\text{poly}) = \bigcup_{k=1}^{\infty} \text{DSPACE}(n^k)$ so that $\text{DSPACE}(\text{poly})$ is the class of sets accepted by deterministic Turing acceptors which use polynomial space.

There are several specific classes which occur frequently. The class of context-sensitive languages is the class $\text{NSPACE}(n)$. These sets are generated by the context-sensitive grammars and are accepted by nondeterministic linear bounded automata.

² This notation was suggested by Patrick C. Fischer.
The class of sets accepted by deterministic linear bounded automata is the class DSPACE(n). This class, sometimes denoted by $\mathcal{D}^2_{\text{lin}}$, is the class of sets whose characteristic functions are in $\mathcal{D}^2$ (where $\mathcal{D}^2$ is the subclass of primitive recursive functions defined by Grzegorczyk).

The class of sets accepted by deterministic Turing acceptors which operate in polynomial time is DTIME(poly). Cobham [7] discussed the importance of the class of functions which can be computed in polynomial time; the subclass of characteristic function corresponds to DTIME(poly). (In [1, 9, 17] this class is referred to as $P$.) The class of sets accepted by nondeterministic Turing acceptors which operate in polynomial time is NTIME(poly). Recently Cook [9] and Karp [17] have shown the importance of the class NTIME(poly) in the study of concrete computational complexity. (In [1, 9, 17] this class is referred to as $NP$.)

The class NTIME(2lin) of sets accepted in exponential time by nondeterministic Turing acceptors was characterized in [16] as the class of spectra of formulae of first-order logic with equality (the spectrum of a formula is the set of cardinalities of its finite models). This class was also studied in [11]. The class DTIME(2lin) of sets accepted in exponential time by deterministic Turing acceptors was characterized in [8] as the class of sets accepted by deterministic or nondeterministic auxiliary pushdown acceptors which operate within space bound $n$.

As noted in [6, 7, 8, 9, 17], for any of the space bounded classes studied here as well as the classes DTIME(poly), NTIME(poly), DTIME(2lin), NTIME(2lin), one gains or loses nothing by specifying random access machines or general recursive programs instead of Turing machines. (This is not true if we restrict attention to multicounter acceptors. For example, the set $\{ww^Rw \mid w \in \{a, b\}^*\}$ is in DTIME($n$) but is not accepted by any nondeterministic online multicounter acceptor which operates in polynomial time.) These classes are specified by means of Turing acceptors in order to take advantage of the conceptual simplicity offered by this model.

2. Basic Results

We are interested in comparing time- and space-bounded classes specified by deterministic and nondeterministic acceptors where the bounds are of the form $n$, $n^k$, $2^n$, $2^{n^k}$. In this section we review what is known with respect to "deterministic simulation of nondeterministic processes" and "time-space tradeoffs".

In the case of the deterministic simulation of nondeterministic time-bounded processes, only the "naive" bounds are known: $\text{DTIME}(f) \subseteq \text{NTIME}(f) \subseteq \bigcup_{c>0} \text{DTIME}(2^{cf})$. Only in a few special cases is more known. It is known that $\text{DTIME}(n) \neq \text{NTIME}(n)$ [2], however it is not known whether there is a polynomial $g$ such that $\text{NTIME}(n) \subseteq \text{DTIME}(g)$ or whether $\text{NTIME}(n) \subseteq \text{DTIME}(\text{poly})$ or whether $\text{NTIME}(n) = \text{DTIME}(2^{\text{lin}})$. If one considers only very restricted Turing
Comparing Complexity Classes

217

machine models, then a few more results are known, e.g., the class of languages accepted by nondeterministic pushdown store acceptors which operate in real time is the class of all context-free languages, while the class of languages accepted by deterministic pushdown store acceptors which operate in linear time is the class of deterministic context-free languages, a proper subclass of the class of context-free languages. The question of whether DTIME(poly) equals NTIME(poly) draws a great deal of attention today [9, 17] due to its importance in the study of the inherent complexity of combinatorial problems. One of the goals of this paper is to explore the connections between questions such as whether DTIME(poly) equals NTIME(poly) and whether DTIME(2^{lin}) equals NTIME(2^{lin}).

In the case of the deterministic simulation of nondeterministic space-bounded processes, one need not use the naive bound. Savitch [21] has shown that NSPACE(f) \subseteq DSPACE(f^2). Thus, DSPACE(poly) = \bigcup \{NSPACE(g) \mid g \text{ is a polynomial}\}, and for any function f, \bigcup_{g=1}^{\infty} NSPACE(f^g) = \bigcup_{g=1}^{\infty} DSPACE(f^g). It is not known whether there exists an \( \epsilon > 0 \) such that NSPACE(f) equals DSPACE(f^{1+\epsilon}), that is, whether Savitch’s result can be “tightened” to yield an equality. The question of whether DSPACE(n) equals NSPACE(n) is the “LBA problem”. See [13] for a discussion of the role of this question in the theory of computation.

To discuss “time-space tradeoffs”, one compares classes specified by the measures time and space while varying the nature of the operation of the Turing acceptors between deterministic and nondeterministic. Recalling that one cannot use more space than time in a single computation, it is clear that for any function f, DTIME(f) \subseteq DSPACE(f) and NTIME(f) \subseteq NSPACE(f). It is known that for any function f, NTIME(f) \subseteq DSPACE(f) [3] and that DSPACE(f) \subseteq NSPACE(f) \subseteq \bigcup_{c}^{\infty} DTIME(2^{c^f}) [8]. In general, finer distinctions are not known, e.g., it is not known how NTIME(2^{lin}) and DSPACE(poly) compare, and it is not known if any of the above weak inclusions are actually equalities, e.g., whether NSPACE(f) equals \bigcup_{c} DTIME(2^{c^f}).

It is useful to consider one more notational convention. It is easy to see that for any polynomial f, \bigcup_{g=1}^{\infty} DTIME(2^{g^f}) = \bigcup_{g=1}^{\infty} DTIME(2^{g^f}) = \bigcup \{DTIME(2^{g^f}) \mid g \text{ is a polynomial}\}. This class will be denoted here by DTIME(2^{poly}). Similarly, \bigcup \{NTIME(2^{g^f}) \mid g \text{ is a polynomial}\} will be denoted by NTIME(2^{poly}) and \bigcup \{DSPACE(2^{g^f}) \mid g \text{ is a polynomial}\} will be denoted by DSPACE(2^{poly}). From the discussion above it should be clear that DSPACE(poly) \subseteq DTIME(2^{poly}) and that NSPACE(2^{poly}) = DSPACE(2^{poly}).

3. Reducibilities and Complete Sets

Cook [9] and Karp [17] have shown that there are questions in logic, in combinatorial mathematics, and in operations research which, when suitably encoded as sets of strings, are “polynomial complete.” A language \( L_0 \) is “polynomial complete”
if $L_0 \in \text{NTIME}(\text{poly})$ and for each $L \in \text{NTIME}(\text{poly})$ there is a "polynomial translation" $f$ (a function which can be computed by a deterministic Turing machine operating in polynomial time) with the property that for any string $w$, $w \in L$ if and only if $f(w) \in L_0$. That is, a decision procedure for $L$ can be obtained from a decision procedure for $L_0$ by means of a polynomial translation. Now $\text{DTIME}(\text{poly}) = \text{NTIME}(\text{poly})$ if and only if there is a polynomial complete language $L$ such that $L$ is in $\text{DTIME}(\text{poly})$ if and only if every polynomial complete language is in $\text{DTIME}(\text{poly})$.

Here we investigate the existence of sets that are "class complete" with respect to certain reducibilities. This allows us to compare various complexity classes defined by time- or space-bounded Turing acceptors discussed in Sections 1 and 2, not just $\text{DTIME}(\text{poly})$ and $\text{NTIME}(\text{poly})$.

**Definition.** Let $f : \Sigma^* \rightarrow \Delta^*$ be a function. A set $L_1 \subseteq \Sigma^*$ is $f$-reducible to $L_2 \subseteq \Delta^*$ if for every $w \in \Sigma^*$, $w \in L_1$ if and only if $f(w) \in L_2$. Let $\mathcal{C}$ be a class of functions (on strings) and let $\mathcal{L}$ be a class of languages. A language $L_0$ is $\mathcal{C}$-complete for $\mathcal{L}$ if $L_0 \in \mathcal{L}$ and for each $L \in \mathcal{L}$, there is a function $f \in \mathcal{C}$ such that $L$ is $f$-reducible to $L_0$.

The type of reducibility used here is a restriction, similar to that used by Karp [17], of the notion of many-one reducibility studied in recursive function theory [19]. If $\mathcal{C}$ is a class of functions which contains the identity function and is closed under composition, then the relation $\leq_{\mathcal{C}}$ defined by $L_1 \leq_{\mathcal{C}} L_2$ if and only if there is a function $f \in \mathcal{C}$ such that $L_1$ is $f$-reducible to $L_2$ is both reflexive and transitive.

There are two specific classes of functions (on strings) used to perform the reducibilities studied here.

**Notation.**

(i) Let $\Pi$ be the class of functions (on strings) computed by deterministic Turing machines which operate in polynomial time.

(ii) Let $\mathcal{F}$ be the class of functions (on strings) computed by $e$-limited one-way finite-state translators, i.e., $e$-limited gsm's.4

The following result will be very useful in comparing complexity classes.

**Lemma 3.1.** Let $\mathcal{L}_1$ be any one of the classes $\text{DTIME}(\text{poly})$, $\text{NTIME}(\text{poly})$,
DTIME(2|lin), NTIME(2|lin), DTIME(2|poly), NTIME(2|poly), DSPACE(poly), or DTIME(g), NTIME(g), DSPACE(g), or NSPACE(g) where g is a polynomial. If \( L_1 \) is an arbitrary language, \( L_2 \in \mathcal{L} \), and for some \( f \in \mathcal{F} \), \( L_1 \) is \( f \)-reducible to \( L_2 \), then \( L_1 \in \mathcal{L} \).

Thus, if \( \mathcal{L} \) is any class of languages such that there exists a language \( L_0 \) which is \( \mathcal{F} \)-complete for \( \mathcal{L} \), then \( \mathcal{L} \subseteq \mathcal{L} \) if and only if \( L_0 \in \mathcal{L} \).

**Proof.** For \( i = 1, 2 \), let \( \Sigma_i \) be a finite alphabet such that \( L_i \subseteq \Sigma_i^* \). Since \( L_2 \in \mathcal{L} \), there is a Turing machine \( M_2 \) such that \( L(M_2) = L_2 \) and such that \( M_2 \) operates in a manner specified by choice of \( \mathcal{L} \) (i.e., within a specified time bound or a specified space bound as indicated by the possible choices of \( \mathcal{L} \)). From a gsm which specifies \( f \), one can modify \( M_2 \) to produce a Turing machine \( M_1 \) which upon input string \( w \in \Sigma_1^* \) simulates \( M_2 \)'s operation on \( f(w) \). Since \( w \in L_1 \) if and only if \( f(w) \in L_2 \), \( L(M_1) = L_1 \). Further, \( M_1 \) operates in a manner specified by choice of \( \mathcal{L} \). Hence, \( L_1 \in \mathcal{L} \).

If \( L_0 \) is \( \mathcal{F} \)-complete for \( \mathcal{L} \), then for every \( L \in \mathcal{L} \) there is a function \( f \in \mathcal{F} \) such that \( L \) is \( f \)-reducible to \( L_0 \). Thus, if \( L_0 \in \mathcal{L} \), then every \( L \in \mathcal{L} \) is in \( \mathcal{L} \). Conversely, if \( \mathcal{L} \subseteq \mathcal{L} \), then \( L_0 \in \mathcal{L} \) since being \( \mathcal{F} \)-complete for \( \mathcal{L} \) implies \( L_0 \in \mathcal{L} \).

If \( \mathcal{F} \) is replaced by \( \Pi \), then Lemma 3.1 does not hold for \( \mathcal{L} \) equal to DTIME(2|lin), NTIME(2|lin), or DTIME(g), NTIME(g), DSPACE(g), or NSPACE(g) where g is a polynomial. The reason for this is the fact that none of these classes is closed under "polynomial translation." For example, if g is a polynomial, \( L_1 \in \text{DSPACE}(g) \), and \( L_2 \) is \( f \)-reducible to \( L_1 \) for some \( f \in \Pi \), then one can conclude that \( L_2 \in \text{DSPACE}(g) \), but in general \( L_2 \notin \text{DSPACE}(g) \). However, if \( \mathcal{F} \) is replaced by \( \Pi \), then Lemma 3.1 does hold for \( \mathcal{L} \) equal to DTIME(poly), NTIME(poly), DTIME(2|poly), and DSPACE(poly) (as well as for many other classes). This is true because in each case the order of the class of bounds is preserved under composition with polynomials and, hence, under composition with reducibilities in \( \Pi \). Thus, we state the following result without proof.

**Lemma 3.2.** Let \( \mathcal{L} \) be any of the classes DTIME(poly), NTIME(poly), DTIME(2|poly), NTIME(2|poly), or DSPACE(poly). If \( L_1 \) is an arbitrary language, \( L_2 \in \mathcal{L} \), and for some \( f \in \Pi \), \( L_1 \) is \( f \)-reducible to \( L_2 \), then \( L_1 \in \mathcal{L} \). Thus, if \( \mathcal{L} \) is any class of languages such that there exists a language \( L_0 \) which is \( \Pi \)-complete for \( \mathcal{L} \), then \( \mathcal{L} \subseteq \mathcal{L} \) if and only if \( L_0 \in \mathcal{L} \).

We shall show the existence of \( \mathcal{F} \)-complete languages for a variety of complexity classes. This yields the existence of \( \Pi \)-complete languages for other classes as shown by the next result.

**Lemma 3.3.**

(i) For any polynomial g, if \( L \) is a language which is \( \mathcal{F} \)-complete for DSPACE(g) (NSPACE(g)), then \( L \) is \( \Pi \)-complete for DSPACE(poly).
(ii) For any polynomial $g$, if $L$ is a language which is $\mathcal{F}$-complete for $\text{NTIME}(g)$, then $L$ is $\Pi$-complete for $\text{NTIME}(\text{poly})$.

(iii) If $L_0$ is a language which is $\mathcal{F}$-complete for $\text{DTIME}(2^{\text{lin}})$ (respectively, $\text{NTIME}(2^{\text{lin}})$), then $L_0$ is $\Pi$-complete for $\text{DTIME}(2^{\text{poly}})$ (respectively, $\text{NTIME}(2^{\text{poly}})$).

**Proof.** We give the proof for part (i), the proofs for parts (ii) and (iii) being essentially the same.

Let $k > 0$ be an integer and let $L_0$ be a language which is $\mathcal{F}$-complete for $\text{DSPACE}(n^k)$. If $L_1 \in \text{DSPACE}(\text{poly})$, then there is an integer $j > 0$ such that $L_1 \in \text{DSPACE}(n^j)$. If $j \leq k$, then $\text{DSPACE}(n^j) \subseteq \text{DSPACE}(n^k)$, so there exists a function $f \in \mathcal{F}$ such that $L_1$ is $f$-reducible to $L_0$. If $j > k$, then let $M_1$ be a deterministic Turing acceptor such that $L(M_1) = L_1$ and $M_1$ operates within space bound $n^j$. Let $\Sigma$ be the input alphabet of $M_1$ so that $L_1 \subseteq \Sigma^*$, and let $c$ be a new symbol, $c \notin \Sigma$. Let $g : \Sigma^* \to (\Sigma \cup \{c\})^*$ be the function defined by $g(w) = wc^m$ where $|wc^m| = |w|^j$. Clearly, $g$ is computed by a deterministic Turing machine which operates in time bound $n^j$, so that $g \in \Pi$. Let $L_2 = g(L_1) = \{g(w) \mid w \in L_1\} = \{wc^m \mid w \in L_1, |wc^m| = |w|^j\}$, so that for all $w \in \Sigma^*$, $w \in L_1$ if and only if $g(w) \in L_2$.

From $M_1$, construct a deterministic Turing machine $M_2$ which considers input strings of the form $wc^m$, $w \in \Sigma^*$, and operates by checking whether $|wc^m| = |w|^j$, and, if so, imitating $M_1$ on $w$ to determine whether or not $w \in L_1$. Since $M_1$ operates within space bound $n^j$, it is clear that $M_2$ can be made to operate in space $n$. Clearly, $L(M_2) = L_2$ so that $L_2 \in \text{DSPACE}(n)$. Now, $n \leq n^k$ so $\text{DSPACE}(n) \subseteq \text{DSPACE}(n^k)$.

Since $L_2 \in \text{DSPACE}(n)$ and $L_0$ is $\mathcal{F}$-complete for $\text{DSPACE}(n^k)$, there is a function $f \in \mathcal{F}$ such that $L_2$ is $f$-reducible to $L_0$. Thus, for all $y \in (\Sigma \cup \{c\})^*$, $y \in L_2$ if and only if $f(y) \in L_0$, so for all $w \in \Sigma^*$, $w \in L_1$ if and only if $g(w) \in L_2$ if and only if $f(g(w)) \in L_0$. Thus, $L_1$ is $f \cdot g$-reducible to $L_0$. But $f \in \mathcal{F}$ and $g \in \Pi$, so $f \cdot g$ is in $\Pi$. Since $L_1$ was chosen arbitrarily in $\text{DSPACE}(\text{poly})$, and $L_0 \in \text{DSPACE}(n^k) \subseteq \text{DSPACE}(\text{poly})$, this shows that $L_0$ is $\Pi$-complete for $\text{DSPACE}(\text{poly})$.

To show the existence of sets which are $\mathcal{F}$-complete for some of these classes, we rely on results established in [3, 4, 24] showing these classes to be principal abstract families of languages. We restrict attention to the classes needed for the theorem.

**Lemma 3.4.**

(i) For any polynomial $g$, there exists a language $L_0$ which is $\mathcal{F}$-complete for $\text{DSPACE}(g)$ (i.e., $\text{NSPACE}(g)$) and hence $\Pi$-complete for $\text{DSPACE}(\text{poly})$.

(ii) For any polynomial $g$, there exists a language $L_0$ which is $\mathcal{F}$-complete for $\text{NTIME}(g)$ and hence $\Pi$-complete for $\text{NTIME}(\text{poly})$.

(iii) There exists a language $L_0$ which is $\mathcal{F}$-complete for $\text{DTIME}(2^{\text{lin}})$ (respectively, $\text{NTIME}(2^{\text{lin}})$) and hence $\Pi$-complete for $\text{DTIME}(2^{\text{poly}})$ (respectively, $\text{NTIME}(2^{\text{poly}})$).
Proof. We give the proof for part (i), the proofs of parts (ii) and (iii) being the same. In [3, 4] it was shown that for any polynomial \( g \) there exists a language \( L_0 \in \text{DSPACE}(g) \) such that for every language \( L \in \text{DSPACE}(g) \) there exist a nonerasing homomorphism \( h_1 \), a homomorphism \( h_2 \), and a regular set \( R \) such that \( L = h_1(h_2^{-1}(L_0) \cap R) \). The language \( L_0 \) is the set of all strings of the form \( \bar{a}_1\bar{M}\bar{a}_2\bar{M} \cdots \bar{a}_n\bar{M} \) where \( \bar{M} \) is the encoding of a deterministic Turing acceptor which operates within space bound \( g \), each \( \bar{a}_i \) is the encoding of a symbol in the input alphabet of \( M \), and \( a_1 \cdots a_n \in L(M) \). Thus, for any deterministic Turing acceptor \( M \) which operates within space bound \( g \) there is a function \( f_M \in \mathcal{F} \) (in fact, a nonerasing homomorphism) such that \( a_1 \cdots a_n \in L(M) \) if and only if \( f_M(a_1, \ldots, a_n) = \bar{a}_1\bar{M} \cdots \bar{a}_n\bar{M} \in L_0 \). Thus, \( L_0 \) is \( \mathcal{F} \)-complete for \( \text{DTAPE}(g) \), and hence, by Lemma 3.4, \( L_0 \) is \( \Pi \)-complete for \( \text{DSPACE}(\text{poly}) \).

In [9, 17] quite different \( \Pi \)-complete sets for \( \text{NTIME}(\text{poly}) \) are studied. By using reducibilities computed by machines which operate in polynomial time and linear space, it is shown in [18] that the complexity of problems such as the equivalence of regular expressions can be related to classes such as \( \text{NSPACE}(n) \) and \( \text{DSPACE}(\text{poly}) \). A survey of such results can be found in [13].

There is no language which is \( \mathcal{F} \)-complete for \( \text{NTIME}(\text{poly}) \). For suppose \( L_0 \) is \( \mathcal{F} \)-complete for \( \text{NTIME}(\text{poly}) \). Then \( L_0 \in \text{NTIME}(\text{poly}) \) so there is some integer \( k > 0 \) such that \( L_0 \in \text{NTIME}(n^k) \). By Lemma 3.1, this implies that \( \text{NTIME}(\text{poly}) \subseteq \text{NTIME}(n^k) \), contradicting the facts that \( \text{NTIME}(\text{poly}) = \bigcup_{j=1}^{\infty} \text{NTIME}(n^j) \) and for every \( j \), \( \text{NTIME}(n^j) \subseteq \text{NTIME}(n^{j+1}) \) [10]. Similarly, there is no language which is \( \mathcal{F} \)-complete for \( \text{DSPACE}(\text{poly}) \) or for \( \text{DTIME}(2^{\text{poly}}) \) or \( \text{NTIME}(2^{\text{poly}}) \).

One should note that any study of complete sets does not show that certain problems are simple. That is, a complete element "encodes" all the information about the class.

Now we turn to the principal results of this section. We use the fact that some classes have \( \mathcal{F} \)-complete sets and apply Lemma 3.1 to obtain comparisons between classes.

**Theorem 3.5.** For any polynomial \( g \), there exists a language \( L_0 \in \text{NSPACE}(g) \) such that each of the following hold.

1. \( L_0 \in \text{DTIME}(\text{poly}) \) (respectively, \( \text{NTIME}(\text{poly}) \)) if and only if \( \text{NSPACE}(g) \subseteq \text{DTIME}(\text{poly}) \) (respectively, \( \text{NTIME}(\text{poly}) \)) if and only if \( \text{DTIME}(\text{poly}) = \text{DSPACE}(\text{poly}) \) (respectively, \( \text{NTIME}(\text{poly}) = \text{DSPACE}(\text{poly}) \));
2. \( L_0 \in \text{DTIME}(2^{\text{lin}}) \) (respectively, \( \text{NTIME}(2^{\text{lin}}) \)) if and only if \( \text{NSPACE}(g) \subseteq \text{DTIME}(2^{\text{lin}}) \) (respectively, \( \text{NSPACE}(g) \subseteq \text{NTIME}(2^{\text{lin}}) \));
3. \( L_0 \in \text{DSPACE}(g) \) if and only if \( \text{NSPACE}(g) = \text{DSPACE}(g) \).

Proof. In each case we take \( L_0 \) to be any language which is \( \mathcal{F} \)-complete for \( \text{NSPACE}(g) \). From Lemma 3.4 we know that such an \( L_0 \) exists. Applying Lemma 3.1
we see that NSPACE\(g\) \(\subseteq\) DTIME(poly) if and only if \(L_0 \in\) DTIME(poly). Since \(L_0\) is \(\mathcal{F}\)-complete for NSPACE\(g\), \(L_0\) is \(\Pi\)-complete for DSPACE(poly) (Lemma 3.3). Thus, by Lemma 3.2 DSPACE(poly) \(\subseteq\) DTIME(poly) if and only if \(L_0 \in\) DTIME(poly). Since DTIME(poly) \(\subseteq\) NTIME(poly) \(\subseteq\) DSPACE(poly), we have (i). The proofs of (ii) and (iii) are the same, noting that DSPACE\(g\) \(\subseteq\) NSPACE\(g\). 

Part (i) of Theorem 3.5 says that (deterministic or nondeterministic) polynomial time has the same computational power as polynomial space if and only if any language which is \(\mathcal{F}\)-complete for a class specified by any fixed polynomial space bound can be recognized in polynomial time. We conjecture that NTIME(poly) \(\neq\) DSPACE(poly) so that such complete sets must take more than polynomial time to recognize.

Part (iii) of Theorem 3.5 may be interpreted as the "LBA-problem translated upward to space bound \(g\)."

If we consider deterministic machines operating within space bounds, then similar results can be obtained. We state these results in the next theorem, the proof being essentially the same as that of Theorem 3.5.

**THEOREM 3.6.** For any polynomial \(g\), there exists a language \(L_0 \in\) DSPACE\(g\) such that each of the following holds.

(i) \(L_0 \in\) DTIME(poly) (respectively, NTIME(poly)) if and only if DTIME(poly) = NTIME(poly) = DSPACE(poly) (respectively, NTIME(poly) = DSPACE(poly));

(ii) \(L_0 \in\) DTIME(\(2^{\text{lin}}\)) (respectively, NTIME(\(2^{\text{lin}}\))) if and only if DSPACE\(g\) \(\subseteq\) DTIME(\(2^{\text{lin}}\)) (respectively, NTIME(\(2^{\text{lin}}\))).

**Corollary.** There exists a language \(L_0 \in\) DSPACE(poly) such that \(L_0 \in\) DTIME(poly) (respectively, NTIME(poly)) if and only if DTIME(poly) = NTIME(poly) = DSPACE(poly) (respectively, NTIME(poly) = DSPACE(poly)).

Now we use languages which are complete for time classes.

**THEOREM 3.7.** For any polynomial \(g\), there exists a language \(L_0 \in\) NTIME\(g\) such that:

(i) \(L_0 \in\) DTIME(poly) if and only if NTIME\(g\) \(\subseteq\) DTIME(poly) if and only if NTIME(poly) = DTIME(poly);

(ii) \(L_0 \in\) DTIME(\(2^{\text{lin}}\)) if and only if NTIME\(g\) \(\subseteq\) DTIME(\(2^{\text{lin}}\)).

**Proof.** By Lemma 3.4 there is a language \(L_0\) which is \(\mathcal{F}\)-complete for NTIME\(g\) and \(\Pi\)-complete for NTIME(poly). By Lemma 3.1, \(L_0 \in\) DTIME(poly) if and only if NTIME\(g\) \(\subseteq\) DTIME(poly). By Lemma 3.2, \(L_0 \in\) DTIME(poly) if and only if NTIME(poly) \(\subseteq\) DTIME(poly). By Lemma 3.1, \(L_0 \in\) DTIME(\(2^{\text{lin}}\)) if and only if NTIME\(g\) \(\subseteq\) DTIME(\(2^{\text{lin}}\)).
COROLLARY. For any polynomial $g$, $\text{NTIME}(g) \neq \text{DTIME}(\text{poly})$.

Proof. If $\text{NTIME}(g) \subseteq \text{DTIME}(\text{poly})$, then $\text{NTIME}(\text{poly}) = \text{DTIME}(\text{poly})$. But $\text{NTIME}(g) \nsubseteq \text{NTIME}(\text{poly})$. [10].

Part (i) of Theorem 3.7 generalizes the result in [1] that $\text{NTIME}(\text{poly}) = \text{DTIME}(\text{poly})$ if and only if $\text{NTIME}(n) \subseteq \text{DTIME}(\text{poly})$. It should be noted that the only class $\text{NTIME}(g)$, $g$ a polynomial, known to be included in $\text{DTIME}(2^{\text{lin}})$ is the class $\text{NTIME}(n)$.

THEOREM 3.8. There exists a language $L_0 \in \text{DTIME}(2^{\text{lin}})$ such that each of the following hold.

(i) $L_0 \in \text{NTIME}(\text{poly})$ if and only if $\text{DTIME}(2^{\text{lin}}) \subseteq \text{NTIME}(\text{poly})$ if and only if $\text{DTIME}(2^{\text{poly}}) = \text{NTIME}(\text{poly}) = \text{DSPACE}(\text{poly})$;

(ii) For any polynomial $g$, $L_0 \in \text{DSPACE}(g)$ (respectively, $\text{NSPACE}(g)$) if and only if $\text{DTIME}(2^{\text{lin}}) \subseteq \text{DSPACE}(g)$ (respectively, $\text{NSPACE}(g)$) if and only if $\text{DTIME}(2^{\text{poly}}) = \text{DSPACE}(\text{poly})$.

Proof. By Lemma 3.4 there is a language $L_0$ which is $\mathcal{F}$-complete for $\text{DTIME}(2^{\text{lin}})$ and II-complete for $\text{DTIME}(2^{\text{poly}})$. By Lemma 3.1, $L_0 \in \text{NTIME}(\text{poly})$ if and only if $\text{DTIME}(2^{\text{lin}}) \subseteq \text{NTIME}(\text{poly})$, and for any polynomial $g$, $L_0 \in \text{DSPACE}(g)$ if and only if $\text{DTIME}(2^{\text{lin}}) \subseteq \text{DSPACE}(g)$. By Lemma 3.2, $L_0 \in \text{NTIME}(\text{poly})$ if and only if $\text{DTIME}(2^{\text{poly}}) \subseteq \text{NTIME}(\text{poly})$, and $L_0 \in \text{DSPACE}(\text{poly})$ if and only if $\text{DTIME}(2^{\text{poly}}) \subseteq \text{DSPACE}(\text{poly})$. Since

$$\text{NTIME}(\text{poly}) \subseteq \text{DSPACE}(\text{poly}) \subseteq \text{DTIME}(2^{\text{poly}}),$$

the equalities result.

It is shown in [1] that $\text{DTIME}(2^{\text{lin}}) \neq \text{NTIME}(\text{poly})$ and that for all $k > 1$, $\text{DTIME}(k^n) \neq \text{NTIME}(\text{poly})$. Similarly, from Theorem 3.8 we have the following result.

COROLLARY. $\text{DTIME}(2^{\text{lin}}) \neq \text{DSPACE}(\text{poly})$.

Proof. If $\text{DTIME}(2^{\text{lin}}) \subseteq \text{DSPACE}(\text{poly})$, then $\text{DTIME}(2^{\text{poly}}) = \text{DSPACE}(\text{poly})$. But $\text{DTIME}(2^{\text{lin}}) \nsubseteq \text{DTIME}(2^{\text{poly}})$ [14].

If we consider the counterpart of Theorem 3.8 for the case $\text{NTIME}(2^{\text{lin}})$, then we obtain a necessary and sufficient condition for $\text{DTIME}(2^{\text{lin}})$ to be equal to $\text{NTIME}(2^{\text{lin}})$. Once again the proof involves simple application of Lemmas 3.1–3.4 and is omitted.

THEOREM 3.9. There exists a language $L_0 \in \text{NTIME}(2^{\text{lin}})$ such that each of the following holds.
(i) $L_0 \in \text{DTIME}(2^{\text{lin}})$ if and only if $\text{NTIME}(2^{\text{lin}}) = \text{DTIME}(2^{\text{lin}})$;
(ii) $L_0 \in \text{DTIME}(2^{\text{poly}})$ if and only if $\text{NTIME}(2^{\text{lin}}) \subseteq \text{DTIME}(2^{\text{poly}})$ if and only if $\text{NTIME}(2^{\text{poly}}) = \text{DTIME}(2^{\text{poly}})$;
(iii) For any polynomial $g$, $L_0 \in \text{DSPACE}(g)$ (respectively, $\text{NSPACE}(g)$) if and only if $\text{NTIME}(2^{\text{lin}}) \subseteq \text{DSPACE}(g)$ (respectively, $\text{NSPACE}(g)$) if and only if $\text{NTIME}(2^{\text{poly}}) = \text{DTIME}(2^{\text{poly}}) = \text{DSPACE}(\text{poly})$.

**Corollary.** $\text{NTIME}(2^{\text{lin}}) \neq \text{DSPACE}(\text{poly})$.

From results in [22], one can conclude that $\text{NTIME}(2^{\text{lin}}) \neq \text{NTIME}(2^{\text{poly}})$. Thus we have the following result.

**Corollary.** $\text{NTIME}(2^{\text{lin}}) \neq \text{DTIME}(2^{\text{poly}})$.

In [1] it is shown that $\text{NTIME}(\text{poly}) \neq \text{NTIME}(2^{\text{lin}})$. The proof uses results in formal language theory. A quite different proof using mathematical logic appears in [11].

### 4. Upward Translation Results

Suppose that $\text{DTIME}(\text{poly}) = \text{NTIME}(\text{poly})$ or that $\text{DTIME}(2^{\text{lin}}) = \text{NTIME}(2^{\text{lin}})$. What are the consequences? The consequences of $\text{DTIME}(\text{poly}) = \text{NTIME}(\text{poly})$ in terms of the complexity of many nonautomata-theoretic problems are explored in [9, 17]. A consequence of $\text{DTIME}(2^{\text{lin}}) = \text{NTIME}(2^{\text{lin}})$ in mathematical logic is discussed in [11, 16]. Here consequences with respect to other complexity classes are developed by applying a simple "translation" technique which has been used in several recent papers [5, 10, 15, 20, 21].

**Theorem 4.1.** If $\text{DTIME}(\text{poly}) = \text{NTIME}(\text{poly})$, then:

(i) for any function $f$, $\cup_{c > 0} \text{DTIME}(2^{cf}) = \text{NTIME}(2^{cf})$,
(ii) for any function $f$, $\cup_{j=1}^{\infty} \text{DTIME}(f^j) = \text{NTIME}(f^j)$,
(iii) $\text{DTIME}(2^{\text{poly}}) = \text{NTIME}(2^{\text{poly}})$.

**Proof.** The proof of (i) is given here, the proofs of (ii) and (iii) being similar. Recall that we consider only self-computable bounding functions.

For any function $f$, $\cup_{c > 0} \text{DTIME}(2^{cf}) \subseteq \cup_{c > 0} \text{NTIME}(2^{cf})$. For any $L_1 \in \cup_{c > 0} \text{NTIME}(2^{cf})$, there is some $j$ such that $L_1 \in \text{NTIME}(j^f)$ so that there is a nondeterministic Turing machine $M_1$ which accepts $L_1$ and which operates within time bound $j^f$. Let $\Sigma$ be a finite alphabet such that $L_1 \subseteq \Sigma^*$, let $c$ be a new symbol,
c \notin \Sigma$, and let $L_2 = \{wc^m \mid w \in L_1 \text{ and } |wc^m| = j^l(|w|)\}$. From $M_1$ one can construct a nondeterministic Turing machine $M_2$ such that $M_2$ accepts $L_2$ and $M_2$ operates within time bound $n$. Thus, $L_2 = L(M_2) \in \text{NTIME}(n)$. Now, if

$$\text{DTIME}(\text{poly}) = \text{NTIME}(\text{poly}),$$

then there is some integer $t \geq 1$ such that $L_2 \in \text{DTIME}(n^t)$. This means that there is a deterministic Turing machine $M_3$ such that $M_3$ accepts $L_2$ and $M_3$ operates within time bound $n^t$. But from $M_3$ one can construct a deterministic Turing machine $M_4$ which accepts $L_1$ and on input $w$ uses the same number of steps as $M_3$ uses on input $wc^m$ where $|wc^m| = j^l(|w|)$, that is, on input $w$, $M_4$ uses $(|wc^m|)^t = (j^l(|w|))^t = (j^t)^l(|w|)$ steps. Thus, $L_1 = L(M_4) \in \text{DTIME}(j^t)^l \subseteq \bigcup_{c>0} \text{DTIME}(2^{ct})$. But $L_1$ was chosen as an arbitrary language in $\bigcup_{c>0} \text{NTIME}(2^{ct})$. Hence, $\bigcup_{c>0} \text{NTIME}(2^{ct}) = \bigcup_{c>0} \text{DTIME}(2^{ct})$. 

In [11] quite different techniques are used to show that if $\text{DTIME}(\text{poly}) = \text{NTIME}(\text{poly})$, then $\text{DTIME}(2^{lin}) = \text{NTIME}(2^{lin})$.

The proof of the next result is very similar to that of Theorem 2.1 and is omitted.

**Theorem 4.2.** If $\text{DTIME}(2^{lin}) = \text{NTIME}(2^{lin})$, then

(i) for any function $f$, $\bigcup_{c>0} \text{NTIME}(2^{cf}) = \bigcup_{c>0} \text{DTIME}(2^{ct})$, and

(ii) $\text{DTIME}(2^{poly}) = \text{NTIME}(2^{poly})$.

The following results explore the consequences of assuming that $\text{NTIME}(\text{poly}) = \text{DSpace}(\text{poly})$ or that $\text{DTIME}(2^{lin})$ (or $\text{NTIME}(2^{lin})$) and $\text{DSpace}(\text{poly})$ are comparable. The proof of Theorem 4.3 follows the same tack as that of Theorem 4.1. However, since a time class is being compared to a tape class, the proof of one part is presented.

**Theorem 4.3.** If $\text{NTIME}(\text{poly}) = \text{DSpace}(\text{poly})$, then:

(i) for any function $f$, $\bigcup_{c>0} \text{NTIME}(2^{cf}) = \bigcup_{c>0} \text{DSpace}(2^{ct})$,

(ii) for any function $f$, $\bigcup_{j=1}^\infty \text{NTIME}(f^j) = \bigcup_{j=1}^\infty \text{DSpace}(f^j)$.

(iii) $\text{NTIME}(2^{poly}) = \text{DSpace}(2^{poly})$.

**Proof of (ii).** As noted above, for any function $f$,

$$\bigcup_{j=1}^\infty \text{NTIME}(f^j) \subseteq \bigcup_{j=1}^\infty \text{NSpace}(f^j) = \bigcup_{j=1}^\infty \text{DSpace}(f^j).$$

For any $L_1 \in \bigcup_{j=1}^\infty \text{DSpace}(f^j)$, there is some $k$ such that $L_1 \in \text{DSpace}(f^k)$ so that there is a deterministic Turing machine $M_1$ which accepts $L_1$ and which operates
within space bound $f^k$. Let $\Sigma$ be a finite alphabet such that $L_1 \subseteq \Sigma^*$, let $c$ be a new symbol, $c \notin \Sigma$, and let $L_2 = \{wc^m | w \in L_1 \text{ and } |wc^m| = f(|w|)\}$. From $M_1$ one can construct a deterministic Turing machine $M_2$ such that $M_2$ accepts $L_2$ and $M_2$ operates within tape bound $g(n) = n^k$. Thus, $L_2 = L(M_2) \in \text{DSpace}(g)$. Since $g$ is a polynomial, $\text{DSpace}(g) \subseteq \text{DSpace}(\text{poly})$ so by hypothesis, there is some integer $t \geq 1$ such that $L_2 \in \text{NTIME}(n^t)$. This means that there is a nondeterministic Turing machine $M_3$ such that $M_3$ accepts $L_2$ and $M_3$ operates within time bound $n^t$. But from $M_3$ one can construct a nondeterministic Turing machine $M_4$ which accepts $L_1$ and on input $w$ uses the same number of steps as $M_3$ uses on input $wc^m$ where $|wc^m| = f(|w|)$, that is, on input $w$, $M_4$ uses $|wc^m|^t = (f(|w|))^t = f^t(|w|)$ steps. Thus, $L_1 \in \text{NTIME}(f^t) \subseteq \bigcup_{j=1}^{\infty} \text{NTIME}(f^j)$. But $L_1$ was chosen as an arbitrary language in $\bigcup_{j=1}^{\infty} \text{DSpace}(f^j)$. Hence, $\bigcup_{j=1}^{\infty} \text{DSpace}(f^j) \subseteq \bigcup_{j=1}^{\infty} \text{NTIME}(f^j)$. 

If $\text{DTIME}(\text{poly}) = \text{DSpace}(\text{poly})$, then $\text{DTIME}(\text{poly}) = \text{NTIME}(\text{poly})$ since $\text{DTIME}(\text{poly}) \subseteq \text{NTIME}(\text{poly}) \subseteq \text{DSpace}(\text{poly})$. Thus, Theorems 4.1 and 4.3 yield the following corollary.

**Corollary.** If $\text{DTIME}(\text{poly}) = \text{DSpace}(\text{poly})$, then

(i) for any function $f$, $\bigcup_{c>0} \text{DTIME}(2^{ac}) = \bigcup_{c>0} \text{NTIME}(2^{ac}) = \bigcup_{c>0} \text{DSpace}(2^{ac})$,

(ii) for any function $f$, $\bigcup_{j=1}^{\infty} \text{DTIME}(f^j) = \bigcup_{j=1}^{\infty} \text{NTIME}(f^j) = \bigcup_{j=1}^{\infty} \text{DSpace}(f^j)$,

(iii) $\text{DTIME}(2^\text{poly}) = \text{NTIME}(2^\text{poly}) = \text{DSpace}(2^\text{poly})$.

The results in this section are "upward" translation results. One might well consider the possibility of "downward" translation results. For example, if $\text{DTIME}(2^{n \ln n}) = \text{NTIME}(2^{n \ln n})$, is it true that $\text{DTIME}(\text{poly}) = \text{NTIME}(\text{poly})$? That is, does a partial converse to Theorem 4.1 hold? No results of this type are known and it is not clear whether any are possible for time-bounded or space-bounded computations.

5. Conclusion

In this paper a number of results on reducibilities, class complete sets, and translations have been established. There are no results such as $\text{DTIME}(\text{poly}) = \text{NTIME}(\text{poly})$ or $\text{DTIME}(\text{poly}) \neq \text{NTIME}(\text{poly})$. However, implications of some possible relationships have been investigated. Some of these results can be interpreted as circumstantial evidence that some of these relationships do not hold and that some pairs of classes are set-theoretically incomparable. This is particularly true of pairs where one class is defined by a time bound and the other by a space bound. The existence of complete sets with respect to easy-to-compute reducibilities gives some
FIG. 1. $L_1 \subseteq L_2$; $L_1 \subseteq L_2$. 

FIG. 2. $L_1 \neq L_2$ indicates that it is not known whether $L_1 = L_2$ or $L_1 \neq L_2$. 
indication of the structure of these classes and allows one to show that certain classes are not equal to one another.

Figure 1 shows some of the known inclusion relationships between the classes studied here, and Fig. 2 shows some of the inequalities.

The methods used here apply to a much wider range of complexity classes than considered in this paper (see [8, 11, 13, 18]). We have not considered classes defined by arbitrary bounding functions but have restricted our attention to bounds of the form $n, n^k, 2^n, 2^{n^k}$. It appears that reducibilities from $\mathcal{F}$ and $\mathcal{II}$ have most application here and the questions of deterministic simulation of nondeterministic processes and time-space tradeoffs are most significant in the case of subelementary bounds.

**References**

18. A. Meyer and L. Stockmeyer, The equivalence problem for regular expressions with
squaring requires exponential space, Conf. Record IEEE Thirteenth Annual Symposium on Switching and Automata Theory (1972), 125–129.


