Theoretical Computer Science 3 (1977) 23–33.

COMPLETE SETS AND THE POLYNOMIAL-TIME HIERARCHY*

Celia WRATHALL[1]
Department of Computer Science, Yale University, New Haven, CT 06520, U.S.A.

Communicated by Albert Meyer
Received March 1975
Revised December 1975

New proofs of two properties of the polynomial-time hierarchy are given. The classes in the hierarchy are characterized using polynomially bounded quantifiers. Using this result, a sequence of complete sets for the hierarchy is exhibited.

The subject of this paper is the internal structure of the "polynomial-time hierarchy" ($\mathcal{P}$-hierarchy). The $\mathcal{P}$-hierarchy was defined in [9] as a structure analogous to the arithmetic hierarchy: its definition extends upward the analogy between the recursive sets (those sets for which it is possible to decide membership) and the class of sets recognizable deterministically in polynomial time (those sets for which it is "practical" to decide membership [4, 7]).

This hierarchy can be used for the classification of problems whose solution is not known to require more than polynomial time, but for which no polynomial-time algorithm (even nondeterministic) is known to exist. The reader should consult [11] for examples of such classification and for a more complete discussion of the properties of the $\mathcal{P}$-hierarchy.

The definition of the $\mathcal{P}$-hierarchy uses a restricted Turing reducibility. Here a useful characterization of the classes is given, which shows that they can also be defined by means of polynomially bounded quantifiers. The proof of this characterization relies on a simple representation of the classes in the $\mathcal{P}$-hierarchy using homomorphisms that perform bounded erasing. Using this characterization, a new, simpler proof of Theorem 4.2 of [12] is presented: each class of the hierarchy contains a set B_k of encoded propositional formulas that is complete (for that class) with respect to reductions that can be performed in $\log(n)$ space. This reducibility is the most restrictive for which such a completeness result can hold, in the sense that transformations computed in time some fixed polynomial will not suffice to reduce any language in some class of the hierarchy to a single language. A completeness

* This research was supported in part by the National Science Foundation under Grants GJ-30409 and DCR-75-15945.

[1] Current address: Department of Mathematics, University of California, Santa Barbara, CA 93106, U.S.A.

result from [11] gives some insight into the relationship between the $\mathcal{P}$-hierarchy and the class of sets that can be accepted in polynomial space. The latter class contains all of the polynomial-time hierarchy; whether the containment is proper is not known, but it must be proper if the hierarchy is infinite.

The reader is assumed to be familiar with basic notions of Turing machines and formal languages. For the most part, the notation and conventions of [11] will be followed. If S is a finite set of symbols (an alphabet), then S^* denotes the set of strings of symbols in S, including the empty string λ. A language is a subset of $S^+ = S^* - \{\lambda\}$; i.e., we consider only languages that do not contain the empty string. The length of a string x is denoted by $|x|$. For a class of languages $\mathcal{C}$,

$$\text{co-}\mathcal{C} = \{\bar{L} = S^+ - L : L \subseteq S^+, L \in \mathcal{C}\}.$$

The model for Turing acceptor used here has a two-way read-only input tape and multiple work tapes. It may be deterministic or nondeterministic. Suppose f is a function from $\mathbf{N}$ (the nonnegative integers) to $\mathbf{N}$. A language L is accepted by a Turing machine M in time $f(n)$ if for $x \in L$, there is an accepting computation of M on x of at most $f(|x|)$ steps, and for $x \notin L$, there is no accepting computation of M on x. Similarly, M accepts L in space $f(n)$ if for $x \in L$, there is an accepting computation of M on x in which no more than $f(|x|)$ squares are visited on any work tape, and for $x \notin L$, there is no accepting computation of M on x. The class of languages accepted by deterministic (nondeterministic) Turing machines in time $f(n)$ is denoted by DTIME($f(n)$) (NTIME($f(n)$)), and the class of languages accepted by deterministic (nondeterministic) Turing machines in space $f(n)$ is denoted by DSPACE($f(n)$) (NSPACE($f(n)$)). Also, let

$$\mathcal{P} = \bigcup \{\text{DTIME}(p(n)) : p(n) \text{ a polynomial}\},$$

$$\mathcal{NP} = \bigcup \{\text{NTIME}(p(n)) : p(n) \text{ a polynomial}\},$$

$$\mathcal{P}\text{-SPACE} = \bigcup \{\text{DSPACE}(p(n)) : p(n) \text{ a polynomial}\}.$$

The definition of the polynomial-time hierarchy uses polynomial analogs of the relations "r.e. in" and "recursive in" used to define the arithmetic hierarchy. The definition of these "polynomial in" relations is based on the following model for relative computation.

Definition. An *oracle machine* M is a multitape Turing acceptor with a distinguished *oracle tape* and distinguished states $q_?$ (for "query"), "yes", and "no". Computations of such a machine are defined relative to an *oracle set* A, which may be any language over the tape alphabet of M. The computation proceeds as if M were a Turing machine until M enters the query state in order to make an oracle call; the next state is "yes" if the (nonblank) contents of the oracle tape is in A and "no" otherwise. (M can make transitions from the query state only in this way.) We make the convention that the oracle tape is reset to the blank tape after such a call

on the oracle. As with Turing acceptors, an oracle machine is deterministic if there is at most one move possible at each step and nondeterministic otherwise. The *language accepted by oracle machine M with oracle set A* will be denoted by $M(A)$. An oracle machine is said to *operate in a time bound* $t(n)$ if for any input x every computation (relative to any oracle set) of the machine on x halts within $t(|x|)$ steps. (An oracle call takes one step.)

This model of oracle machines is similar to that of [8]. Except for the convention that the oracle tape is erased after an oracle call, these machines are essentially the query machines of [2, 5]. Note that this restriction on their mode of operation does not affect their power when arbitrary polynomial time bounds are allowed.

Definition. Let A be a language. Then

$$\mathcal{P}(A) = \{M(A)\text{: } M \text{ is a deterministic oracle machine that operates in time } p(n) \text{ where } p \text{ is some polynomial}\},$$

$$\mathcal{NP}(A) = \{M(A)\text{: } M \text{ is a nondeterministic oracle machine that operates in time } p(n) \text{ where } p \text{ is some polynomial}\}.$$

For a class of languages $\mathcal{C}$,

$$\mathcal{P}(\mathcal{C}) = \bigcup \{\mathcal{P}(A)\text{: } A \in \mathcal{C}\},$$

$$\mathcal{NP}(\mathcal{C}) = \bigcup \{\mathcal{NP}(A)\text{: } A \in \mathcal{C}\}.$$

Thus $\mathcal{P}(A)$ and $\mathcal{NP}(A)$ are classes of languages whose membership problems are decidable relative to A in polynomial time.

In this paper, Turing machines and oracle machines will be viewed as operating on tuples of strings for input; to achieve this it is assumed that some encoding $\langle x_1, \ldots, x_k \rangle$ of k-tuples, $k \geq 1$, has been chosen. One candidate for such an encoding is the "sequential" encoding suggested in [11]: if $R \subseteq S^+ \times \ldots \times S^+$ (k times) is a k-ary string relation then

$$\{x_1 \# x_2 \# \ldots \# x_k\text{: } (x_1, \ldots, x_k) \in R\}$$

is its encoding, where $\#$ is a new symbol. Another possibility is the parallel encoding used in [13]. We assume that the chosen encoding can be manipulated (e.g., the components can be determined) in linear time and that $|\langle x_1, \ldots, x_k \rangle|$ is linear in $\max\{|x_i|\text{: } 1 \leq i \leq k\}$.

The following definition of the polynomial-time hierarchy is that of [11].

Definition. The *$\mathcal{P}$-hierarchy* is the collection of classes $\{\Sigma_k^p, \Pi_k^p, \Delta_k^p\text{: } k \geq 0\}$, where

$$\Sigma_0^p = \Pi_0^p = \Delta_0^p = \mathcal{P}$$

and for $k \geq 0$,

$$\Sigma^P_{k+1} = \mathcal{NP}(\Sigma^P_k),$$

$$\Pi^P_{k+1} = \text{co-}\mathcal{NP}(\Sigma^P_k),$$

$$\Delta^P_{k+1} = \mathcal{P}(\Sigma^P_k).$$

Also define $\mathcal{PH} = \bigcup \{\Sigma^P_k : k \geq 0\}$.

Note that $\Delta^P_1 = \mathcal{P}$ and $\Sigma^P_1 = \mathcal{NP}$. Since for any set A, A and its complement are in $\mathcal{P}(A) \subseteq \mathcal{NP}(A)$, we see that for any $k \geq 0$, $\Delta^P_k \subseteq \Sigma^P_k \cap \Pi^P_k$ and $\Sigma^P_k \cup \Pi^P_k \subseteq \Delta^P_{k+1}$. Thus the $\mathcal{P}$-hierarchy is a nested structure of classes of languages. It is not known, however, whether any of the inclusions $\Sigma^P_k \subseteq \Sigma^P_{k+1}$ for $k \geq 0$ is proper, i.e., whether the hierarchy is finite or infinite.

It is technically useful to represent languages in Σ^P_k in terms of languages in Π^P_{k-1}. Clearly $\Sigma^P_k = \mathcal{NP}(\Pi^P_{k-1})$; however, a simpler representation, using a type of bounded-erasing homomorphism, also holds.

Proposition 1. *For any $k \geq 1$, a language $L \subseteq S^+$ is in Σ^P_k if and only if there exist a homomorphism $h: S^* \to T^*$, a language $L' \subseteq T^+$ in Π^P_{k-1} and a polynomial $p(n)$ such that $L = h(L')$ and for any $x \in L'$, $|x| \leq p(|h(x)|)$. That is, $\Sigma^P_k = \{h(L') : L' \in \Pi^P_{k-1}$, h a homomorphism that performs polynomial-bounded erasing on $L'\}$.*

Proof. For $k \geq 1$, let $\mathcal{H}_k$ denote the class of languages of the form $h(L')$ with $L' \in \Pi^P_{k-1}$ and h a homomorphism such that for some polynomial p, for all $x \in L'$, $|x| \leq p(|h(x)|)$. Using a straightforward machine construction, it is easily seen that each Σ^P_k is closed under application of homomorphisms that perform polynomial-bounded erasing; hence $\mathcal{H}_k \subseteq \Sigma^P_k$ for all k. Conversely, the statement is known to hold for $k = 1$, since $\Pi^P_0 = \mathcal{P}$ and $\Sigma^P_1 = \mathcal{NP}$ (see [3]). The proof proceeds by induction on k. For $k > 1$, suppose that $L = M(A)$ is a language in Σ^P_k, with $A \in \Sigma^P_{k-1}$, M a nondeterministic polynomial-time oracle machine. Then

(*) there exist a homomorphism h_1 and languages $L_1 \in \mathcal{P}$, $L_2 \in \Sigma^P_{k-1}$, and $L_3 \in \Pi^P_{k-1}$ such that $L = h_1(L_1 \cap L_2 \cap L_3)$ and for some polynomial p_1, $|x| \leq p_1(|h(x)|)$ for any $x \in L_1$ (see [13]).

Strings in L_1 have the form $z = \langle x, y, u, v \rangle$ where y encodes a sequence of transitions that cause M (on input x) to query its oracle about the strings in $u = u_1 \# \ldots \# u_m \#$ and $v = v_1 \# \ldots \# v_r \#$ and that cause M to accept x if the answers are "yes" for the strings in u and "no" for the strings in v. ("#" is a new symbol.) Since M operates in polynomial time, if $z \in L_1$ then $|y|$, $|u|$, $|v|$ are bounded by a polynomial in the length of x. The language L_2 (L_3) then consists of strings $\langle x, y, u, v \rangle$ such that $u \in (A \#)^*$ (resp., $v \in (\bar{A} \#)^*$). Now $\mathcal{H}_k$ can be shown to be closed under intersection and under application of homomorphisms that perform polynomial-bounded erasing. By definition, $\mathcal{P} \subseteq \Pi^P_{k-1} \subseteq \mathcal{H}_k$ and (since

$\Pi_{k-2}^p \subseteq \mathcal{H}_k$) by the induction hypothesis, $\Sigma_{k-1}^p \subseteq \mathcal{H}_k$. Hence any language of the form described at (*) is in $\mathcal{H}_k$, so $\Sigma_k^p \subseteq \mathcal{H}_k$. □

Notation. Suppose L is a language and q a polynomial. Let languages L_1 and L_2 be defined by:

$$x \in L_1 \Leftrightarrow \exists y\ [|y| \leq q(|x|) \text{ and } \langle x, y\rangle \in L],$$

$$x \in L_2 \Leftrightarrow \forall y\ [\text{if } |y| \leq q(|x|) \text{ then } \langle x, y\rangle \in L].$$

Then L_1 (L_2) will be said to be defined from L by *polynomial-bounded existential (universal) quantification.* The expression defining L_1 will also be written $(\exists y)_q\ [\langle x, y\rangle \in L]$, and that for L_2, $(\forall y)_q\ [\langle x, y\rangle \in L]$. In the case of multiple quantifiers, the subscripting polynomials will all refer to bounds in terms of x; e.g., $(\exists y)_q\ (\forall y')_{q'}\ [\langle x, y, y'\rangle \in L]$ denotes

$$(\exists y)(\forall y')[|y| \leq q(|x|) \quad \text{and if} \quad |y'| \leq q'(|x|) \quad \text{then} \quad \langle x, y, y'\rangle \in L].$$

Proposition 2. *For each $k \geq 1$, Σ_k^p is closed under polynomial-bounded existential quantification and Π_k^p is closed under polynomial-bounded universal quantification.*

The desired characterization of the classes Σ_k^p and Π_k^p can now be stated; it was in this form — definition by the number of alternations of bounded quantifiers — that the polynomial hierarchy was suggested by Karp [7].

Theorem 3. *Let $L \subseteq S^+$ be a language. For any $k \geq 1$, $L \in \Sigma_k^p$ if and only if there exist polynomials $p_1, \ldots, p_k$ and a language $L' \in \mathcal{P}$ such that for all $x \in S^+$,*

$$x \in L \quad \text{iff} \quad (\exists y_1)_{p_1} (\forall y_2)_{p_2} \ldots (\mathsf{Q} y_k)_{p_k}\ [\langle x, y_1, \ldots, y_k\rangle \in L'].$$

Dually, $L \in \Pi_k^p$ if and only if

$$x \in L \quad \text{iff} \quad (\forall y_1)_{p_1} \ldots (\mathsf{Q}' y_k)_{p_k}\ [\langle x, y_1, \ldots, y_k\rangle \in L']$$

for some $L' \in \mathcal{P}$ and polynomials $p_1, \ldots, p_k$.

The quantifiers in these expressions alternate, so that Q is $\exists$ if k is odd and $\forall$ if k is even. Q will be used in this way consistently, and Q′ for its dual; Q′ is $\forall$ if k is odd and $\exists$ if k is even. This result was announced, without proof, in [12].

Proof. The proof is by induction on k. For ease in the discussion, let $\mathcal{E}_k$ (respectively, $\mathcal{A}_k$) be the class of languages that can be defined from languages in $\mathcal{P}$ by k alternations of polynomial-bounded quantifiers beginning with an existential (respectively, universal) quantifier. The theorem can then be restated as $\mathcal{E}_k = \Sigma_k^p$ and $\mathcal{A}_k = \Pi_k^p$ for $k \geq 1$. Notice that for all k, $\mathcal{A}_k = \text{co-}\mathcal{E}_k$, so if for some k, $\mathcal{E}_k = \Sigma_k^p$ then also $\mathcal{A}_k = \Pi_k^p$.

For the case $k = 1$, it is known [7] that $\mathcal{E}_1 = \mathcal{NP}$. For $k \geq 1$, it can be seen that any

language in $\mathscr{C}_{k+1}$ can be defined by polynomial-bounded existential quantification from a language in $\mathscr{A}_k$. Since (from the previous proposition) Σ^p_{k+1} is closed under this operation, if $\mathscr{A}_k = \Pi^p_k$ then $\mathscr{C}_{k+1} \subseteq \Sigma^p_{k+1}$. On the other hand, if L is a language in Σ^p_{k+1} then from Proposition 1, $L = h(L')$ where $L' \in \Pi^p_k = \mathscr{A}_k$ and h is a homomorphism that performs polynomial-bounded erasing on L'. From the bounded-quantifier formula defining $L' \in \mathscr{A}_k$ as in the statement of the theorem, it is possible to construct a bounded-quantifier formula demonstrating that $h(L') \in \mathscr{C}_{k+1}$; hence $L \in \mathscr{C}_{k+1}$ and $\Sigma^p_{k+1} \subseteq \mathscr{C}_{k+1}$.

The formula for $h(L')$ is constructed as follows. Suppose for convenience that k is even and

$$x \in L' \quad \textit{iff} \quad (\forall y_1)_{p_1} \ldots (\exists y_k)_{p_k} [\langle x, y_1, \ldots, y_k \rangle \in L_1]$$

where $L_1 \in \mathscr{P}$ and $p_1, \ldots, p_k$ are polynomials. Define a language $L_0 \in \mathscr{P}$ by $\langle x, y_0, y_1, \ldots, y_k \rangle \in L_0$ if and only if

(1) $h(y_0) = x$;
(2) for $1 \leqslant i \leqslant \frac{1}{2}k$, $|y_{2i}| \leqslant p(|y_0|)$; and
(3) if for $0 \leqslant i \leqslant \frac{1}{2}k - 1$, $|y_{2i+1}| \leqslant p(|y_0|)$, then $\langle y_0, y_1, \ldots, y_k \rangle \in L_1$.

If p is the polynomial such that $|x| \leqslant p(|h(x)|)$ for $x \in L'$, let $q_0(n) = p(n)$ and for $1 \leqslant i \leqslant k$, let $q_i(n) = p_i(p(n))$. Then

$$x \in L \quad \textit{iff} \quad (\exists y_0)_{q_0} (\forall y_1)_{q_1} \ldots (\exists y_k)_{q_k} [\langle x, y_0, \ldots, y_k \rangle \in L_0]. \ \square$$

The polynomial hierarchy was originally presented as a structure for classifying the complexity of problems (encoded as languages). A useful tool for such classification is the notion of "efficient reduction" of one problem to another and, hence, the concept of a complete problem. The reducibility (or transformability) relation considered here is that determined by functions computed in $\log(n)$ space.

Definition. Suppose $L_1 \subseteq S^+$, $L_2 \subseteq T^+$ are languages and $\mathscr{C}$ is a class of languages.

(i) $L_1 \leqslant_{\log} L_2$ if there is a function $f : S^+ \to T^+$ such that f can be computed in $\log(n)$ space and for any $x \in S^+$, $x \in L_1$ if and only if $f(x) \in L_2$.

(ii) L_1 is *log-complete* in $\mathscr{C}$ if $L_1 \in \mathscr{C}$ and for any $L \in \mathscr{C}$, $L \leqslant_{\log} L_1$.

The model for the computation of these functions is a Turing machine with two-way input, multiple work tapes to which the space bound applies, and a one-way output tape. The class of functions computable in this way in $\log(n)$ space is closed under composition [6, 12] and clearly contains the identity function, so $\leqslant_{\log}$ is a transitive and reflexive relation. $\leqslant_{\log}$ is a restriction of $\leqslant^p_m$, the reducibility defined by the class of functions that can be computed in polynomial time.

The reducibilities $\leqslant_{\log}$ and $\leqslant^p_m$ are appropriate for use with the classes in the polynomial hierarchy because of the following fact, which says that each of the classes Σ^p_k (hence also Π^p_k) and Δ^p_k is "closed under" polynomial time reductions.

Proposition 4. *Suppose* $A \leqslant_m^p B$ *and* $B \in \mathcal{NP}(C)$ *(respectively,* $\mathcal{P}(C)$*). Then also* $A \in \mathcal{NP}(C)$ $(\mathcal{P}(C))$.

The first proof of the existence of a complete set in $\mathcal{NP}$ is given in [5]. The explosion that followed in the number of known "polynomial-complete" problems indicates that classification using reducibilities can be more easily done when complete sets are already known. The proofs in [1,5] show that the set of (encodings of) satisfiable propositional formulas in conjunctive normal form is log-complete in $\mathcal{NP}$; extensions of this set are shown here to be complete in the other classes of the polynomial hierarchy. Another approach through which a log-complete set in each class can be found is based on the construction of a "universal" linear-time oracle machine [13].

Definition. (i) Let E be a map taking propositional formulas into strings, which leaves fixed the connective symbols and parentheses, as well as the constants 0 and 1 ("false", "true"), and writes the variable names over a finite alphabet. Assume for definiteness that the formulas contain variables from the set $\{x[i,j]: i,j \geqslant 1\}$ and that $E(x[i,j])$ is the string $x\beta(i) \# \beta(j) \in \{x,0,1,\#\}^+$ where $\beta(m)$ is the binary representation of m.

(ii) For $k \geqslant 1$, let C_k be the set of formulas that is defined by: if F is a formula in which, for $1 \leqslant i \leqslant k$, the variables $X^i = \{x[i,j]: 1 \leqslant j \leqslant n_i\}$ occur (k is fixed but $n_1, \ldots, n_k \geqslant 1$ depend on F) then $F \in C_k$ if and only if

$$(\exists X^1)(\forall X^2)\ldots(QX^k)\ [F(X^1,\ldots,X^k) \quad \textit{is true}].$$

Thus C_1 is the set of satisfiable formulas; and, for example, if $F = (x[1,1] \wedge x[1,2]) \vee x[2,1]$, then $F \in C_2$.

(iii) Let $B_0 = \emptyset$ and for $k \geqslant 1$, let

$$B_k = \{E(F): F \text{ is a formula in } C_k\}.$$

Suppose a formula F has variables $x[i,j]$ for $1 \leqslant j \leqslant n_i$, $1 \leqslant i \leqslant k$, with $k, n_1, \ldots, n_k \geqslant 1$. If $E(F)$ and strings $y_1, \ldots, y_k \in \{0,1\}^+$ are given, then it can be tested in polynomial time whether $|y_i| = n_i$ for $1 \leqslant i \leqslant k$, and if so, whether F is true under the assignment $x[i,j] =$ the jth symbol of y_i. Also, for $1 \leqslant i \leqslant k$, $n_i < |E(F)|$ if the variables $x[i,1], \ldots, x[i,n_i]$ occur in F, so strings y_i of length greater than $|E(F)|$ can be ignored. Therefore B_k can be defined from a language in $\mathcal{P}$ by k alternations of quantifiers that are bounded by the polynomial $i(n) = n$.

A sketch of a proof that for each $k \geqslant 1$, B_k is log-complete in Σ_k^p is given in [12]; it is essentially a relativization to oracle machines of Cook's original proof for the case $k = 1$. The characterization of the classes of the polynomial-time hierarchy given in Theorem 3 allows a simpler proof for $k > 1$. The desired result will come as a corollary to Theorem 5.

Definition. (i) Consider functions ρ that take positive integers into propositional formulas and have the following property: there exist $k \geq 1$, $r \geq 2$ and non-zero polynomials $p_1, \ldots, p_k$ (all depending on ρ) such that for all $n \geq 1$ the variables in $\rho(n)$ are

$$X^m = \{x[m, j]: 1 \leq j \leq p_m(n)\}, \qquad 1 \leq m \leq k,$$

$$I = \{x[k + i, j]: 1 \leq i \leq r, 1 \leq j \leq n\}.$$

Given such a function, let $S = \{s_1, \ldots, s_r\}$ be an alphabet and $w = s_{i_1} s_{i_2} \ldots s_{i_n}$ be a string of length $n \geq 1$ over S. Then $\rho(n, w)$ denotes the formula resulting from $\rho(n)$ by the following assignment of the variables in the set I: for $1 \leq j \leq n$, $x[k + i_j, j] = 1$ and $x[k + i, j] = 0$ for $i \neq i_j$ (i.e., $x[k + i, j] = 1$ iff the jth symbol of w is s_i),

(ii) A function ρ as in (i) will be said to *represent* a language $L \subseteq S^+$ if:

(1) (a) $L \in \Sigma_k^p$,

(b) if k is even (odd) then for all n, $\rho(n)$ is in disjunctive (conjunctive) normal form, and

(c) for any $w \in S^+$, $w \in L$ if and only if $(\exists X^1)(\forall X^2)\ldots(QX^k)[\rho(|w|, w)$ *is true*] if and only if $\rho(|w|, w) \in C_k$;

or

(2) (a) $L \in \Pi_k^p$,

(b) if k is even (odd) then for all n, $\rho(n)$ is in conjunctive (disjunctive) normal form, and

(c) for any $w \in S^+$, $w \in L$ if and only if $(\forall X^1)\ldots(Q'X^k)[\rho(|w|, w)$ *is true*].

Note that for $k = 1$, a function ρ represents $L \in \Sigma_1^p = \mathcal{NP}$ if and only if $\rho(n)$ is in conjunctive normal form for each n and for any string w, $w \in L$ if and only if $E(\rho(|w|, w)) \in B_1$, the set of encodings of satisfiable formulas.

Theorem 5. *For any $k \geq 1$ and language L in Σ_k^p or Π_k^p, there exists a function ρ that represents L and is such that $E(\rho(n))$ can be computed from the binary representation of n in linear space.*

Proof. The formula $\rho(n)$ can be interpreted as describing the conditions under which the oracle machine associated with L can accept an input of length n. Attention is restricted to a particular input w by the assignment to the input variables I, which yields $\rho(|w|, w)$. The polynomials p_i associated with ρ depend on the time and space used by the machine, and r depends on its alphabet. However, because of Theorem 3, oracle machines are not used explicitly in the following.

The proof is by induction on k. The proof in [1, 5] of the completeness of B_1 in $\mathcal{NP}$ can be revised to yield the basis of the induction. (As pointed out in [6] that proof uses a $\log(n)$ space reduction.) The major difference is that the variables that

describe the first n squares of the tape before the first step are not "bound" to the input but rather are used as the variables I. With this change, only the length of the input is needed to construct the formula, so it may be given in binary to the machine which performs the construction; the amount of tape used is just that needed to write numbers that are polynomial in n, so their lengths are linear in the length of n in binary.

Since $\Pi_k^p = \text{co-}\Sigma_k^p$ for all $k \geq 1$, if the theorem holds for any Σ_k^p, then it also holds for Π_k^p. The induction step is simplified by the following corollary to Theorem 3:

Claim. *Let S be an alphabet, $\#(S) \geq 2$, $¢ \notin S$ and $T = S \cup \{¢\}$. For $k \geq 2$, if $L_1 \subseteq S^+$ is in Σ_k^p then there exist $L_2 \subseteq T^+$ in Π_{k-1}^p and a nondecreasing polynomial p such that for all $x \in T^+$, $x \in L_1$ if and only if $\exists y\,[|y| = p(|x|)$ and $xy \in L_2]$.*

Now suppose the theorem holds for some $k \geq 1$ and let $L_1 \subseteq S^+$ be a language in Σ_{k+1}^p. Let $T = S \cup \{¢\}$ and $\#(T) = r$. Using the claim, let $L_2 \in \Pi_k^p$ and polynomial p be such that $x \in L_1$ if and only if $(\exists y)[|y| = p(|x|)$ *and* $xy \in L_2]$ for any $x \in T^+$. Using the induction hypothesis let ρ_2 be a function that represents L_2, with associated polynomials $q_1, \ldots, q_k$. Then for any $x \in T^+$, $x \in L_1$ if and only if

$$\exists y\,[|y| = p(|x|) \text{ and } (\forall X^1)\ldots(\mathrm{Q}'X^k)\{\rho_2(|x| + p(|x|), xy) \text{ is true}]].$$

The input variables in $\rho_2(n + p(n))$ are

$$I_1 = \{x[k + i, j]\colon 1 \leq i \leq r, 1 \leq j \leq n\}$$

for an initial substring of length n of the input, and

$$I_2 = \{x[k + i, j]\colon 1 \leq i \leq r,\ n + 1 \leq j \leq n + p(n)\}$$

for the remainder of the string. We can construct $\rho_1(n)$ to represent L_1 from $\rho_2(n + p(n))$ by retaining the variables in I_1 as input variables, converting the variables in I_2 to

$$\{x[1, j]\colon 1 \leq j \leq (p(n) - n)r\}$$

(which will be existentially quantified for ρ_1), and changing X^1 to X^2, X^2 to X^3, etc.

More formally, let τ rename the variables in $\rho_2(n + p(n))$ as follows:

for $x[k + i, j] \in I_1$,	$\tau(x[k + i, j]) = x[k + i + 1, j]$;
for $x[k + i, j] \in I_2$,	$\tau(x[k + i, j]) = x[1, (j - n - 1)r + i]$;
and for $1 \leq i \leq k$, $1 \leq j \leq q_i(n + p(n))$,	$\tau(x[i, j]) = x[i + 1, j]$.

Let $\rho_1(n) = \tau(\rho_2(n + p(n)))$. Then the polynomials associated with ρ_1 are $p_1(n) = r \cdot p(n)$ and, for $2 \leq i \leq k + 1$, $p_i(n) = q_{i-1}(n + p(n))$. Since $\rho_1(n)$ is in the same form as $\rho_2(n + p(n))$, $\rho_1(n)$ will be in conjuctive or disjunctive normal form for all n if, for all m, $\rho_2(m)$ is in that form.

Further, for any string $x \in T^+$ of length n, $x \in L_1$ if and only if there is an

assignment to the variables I_2 in $\rho_2(n + p(n))$ such that if the variables in I_1 are assigned to describe x, then

$$(\forall X^2 = \tau(X^1))(\exists X^2)\ldots(Q'X^{k+1})\,[\tau(\rho_2(n + p(n))) \textit{ with those assignments made is true}].$$

Thus, $x \in L_1$ if and only if

$$(\exists X^1)(\forall X^2)\ldots(Q'X^{k+1})\,[\rho_1(n, x) \textit{ is true}];$$

hence ρ_1 represents L_1.

To construct $E(\rho_1(n))$ given n in binary, it is sufficient to calculate $p(n)$ and write out $E(\tau(\rho_2(n + p(n))))$. This can be done in space linear in the length of $n + p(n)$, hence linear in the length of n. □

Corollary 6. *For each $k \geq 1$, B_k is log-complete in Σ_k^p.*

Proof. From Theorem 3 and previous remarks, for each $k \geq 1$, $B_k \in \Sigma_k^p$. Suppose, then that L is a language in Σ_k^p for some $k \geq 1$. From the theorem, let ρ be a function that represents L and let $A(w) = E(\rho(|w|, w))$. Then the mapping A reduces L to B_k, since $w \in L$ if and only if $\rho(|w|, w) \in C_k$ if and only if $A(w) \in B_k$. Also, $A(w)$ can be computed from w in $\log(|w|)$ space, by computing $E(\rho(|w|))$ while making the appropriate substitution of constants for the input variables.

The formula encoded by $A(w)$ is in the same form as $\rho(|w|)$; hence if k is odd then the conjunctive-normal-form formulas in B_k form a log-complete set, and if k is even the disjunctive-normal-form formulas are complete. Note also that the basis step of the theorem can be proved with functions ρ representing languages in $\Sigma_1^p = \mathcal{NP}$ such that $\rho(n)$ is in conjunctive normal form with at most three literals per clause [5], and that in the induction step the structure of the formulas is not changed, only the labelling of the variables. Thus, e.g., for k odd, the set of conjunctive-normal-form formulas in B_k with at most three literals per clause is log-complete in Σ_k^p. □

This corollary is in the strongest possible form. If the time allowed for reductions is restricted to a fixed polynomial, then Σ_k^p cannot have a complete set with respect to the resulting reducibility. By "relativizing" the proof of the nondeterministic time-separation theorem given in [10], it is possible to prove that for any polynomial p and any language A, $\mathcal{NP}(A)$ properly contains the class $\{M(A)\colon M$ a nondeterministic oracle machine that operates in time $p(n)\}$ (see [13]). Using Corollary 6, an inductive proof can be given to shown that $\Sigma_k^p = \mathcal{NP}(B_{k-1})$ for each $k \geq 1$. Combining these two facts, if for some $L_0 \in \Sigma_k^p$ and polynomial $q(n)$, it was the case that any language in Σ_k^p could be reduced to L_0 in time $q(n)$, then Σ_k^p would be properly contained in itself.

Thus since each Σ_k^p can be decomposed into a certain infinite hierarchy of classes, they cannot possess some types of complete sets. If the $\mathcal{P}$-hierarchy is in fact infinite, then the same argument can be applied to $\mathcal{PH}$ to show that it cannot possess a log-complete set. This fact has the following consequence for the question of the relationship of the $\mathcal{P}$-hierarchy to the class $\mathcal{P}$-space.

Proposition 7. *If for all k,* $\Sigma_k^p \subsetneq \Sigma_{k+1}^p$ *then* $\mathcal{PH} \subsetneq \mathcal{P}$-SPACE.

Proof. Recall that $\mathcal{PH} \subseteq \mathcal{P}$-SPACE [11]. Let $B_\omega = \bigcup_{k=1}^{\infty} B_k$; it is proved in [11] that B_ω is log-complete in $\mathcal{P}$-SPACE. If $\mathcal{P}$-SPACE $\subseteq \mathcal{PH}$ then for some j, $B_\omega \in \Sigma_j^p$. Since (Proposition 4) Σ_j^p is closed under log(n)-space reductions, $B_\omega \in \Sigma_j^p$ implies that $\mathcal{P}$-SPACE $\subseteq \Sigma_j^p$ and then $\Sigma_{j+1}^p \subseteq \Sigma_j^p$. □

The contrapositive of this proposition states that if $\mathcal{P}$-SPACE is contained in the $\mathcal{P}$-hierarchy then there is a constant j such that any language in $\mathcal{P}$-SPACE can be defined by a formula using at most j alternations of polynomial-bounded quantifiers.

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms* (Addison-Wesley, Reading, Mass., 1974).

[2] T. Baker, J. Gill and R. Solovay, Relativizations of the $\mathcal{P} = ?\mathcal{NP}$ question, *SIAM J. Comput.*, 4 (1975) 431–442.

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

[4] A. Cobham, The intrinsic difficulty of functions, *Proc. 1964 Congress for Logic, Methodology and Philosophy of Science* (North-Holland, Amsterdam, 1964) 24–30.

[5] S.A. Cook, The complexity of theorem proving procedures, *Proc. Third Annual ACM Symposium on Theory of Computing* (1971) 151–158.

[6] N.D. Jones, Space-bounded reducibility among combinatorial problems. *J. Comput. System Sci.* 11 (1975) 68–85.

[7] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller and J.W. Thatcher, eds., *Complexity of Computer Computations* (Plenum Press, New York, 1972) 85–104.

[8] N.A. Lynch, Relativization of the theory of computational complexity, Doctoral Thesis, Report TR–99, M.I.T., Project MAC, Cambridge, Mass. (1972).

[9] A.R. Meyer and L.J. Stockmeyer, The equivalence problem for regular expressions with squaring requires exponential space, *Proc. Thirteenth Annual IEEE Symposium on Switching and Automata Theory* (1972) 125–129.

[10] J.I. Seiferas, Nondeterministic time and space complexity classes, Doctoral Thesis, Report TR–137, M.I.T., Project MAC, Cambridge, Mass. (1974).

[11] L.J. Stockmeyer, The polynomial-time hierarchy, *Theoret. Comput. Sci.* 3 (1976).

[12] L.J. Stockmeyer and A.R. Meyer, Word problems requiring exponential time: preliminary report, *Proc. Fifth Annual Symposium on Theory of Computing* (1973) 1–9.

[13] C. Wrathall, Subrecursive predicates and automata, Report 56, Department of Computer Science, Yale Univ., New Haven, Conn. (1975).