JOURNAL OF COMPUTER AND SYSTEM SCIENCES 27, 14–28 (1983)

A Low and a High Hierarchy within *NP**

UWE SCHÖNING

Institut für Informatik, Universität Stuttgart, D-7000 Stuttgart, Federal Republic of Germany

Received October 4, 1981; revised September 13, 1982

A low and a high hierarchy within *NP* are defined. The definition is similar to the jump hierarchies below the degree of the halting problem. For this purpose a complexity theoretic counterpart of the jump operator in recursion theory is defined. Some elementary properties of these hierarchies are investigated. The high hierarchy is, in some sense, a hierarchy of generalized *NP*-completeness notions.

1. INTRODUCTION

In recent literature about degrees below $\varnothing'$, the degree of the halting problem, there appears the definition of a low and a high hierarchy of sets (or of degrees) below the degree of the halting problem [11, 20, 27]. A set A belongs to the nth class in the low hierarchy (high hierarchy) iff the nth jump of A, $A^{(n)}$, is in the degree of $\varnothing^{(n)}$ (or of $\varnothing^{(n+1)}$, respectively). We translate this definition into the context of *NP* complexity. To do this, we first define a complexity theoretic counterpart of the jump operator in recursion theory. This is the K-operator which is defined in Section 3. An interesting connection between the K-operator and strong nondeterministic Turing reducibility is shown.

In Section 4 we give the definition of a low and a high hierarchy within *NP*. Some elementary properties concerning the disjointness of these hierarchies and the existence of sets "between" the low and the high hierarchy are investigated.

In Section 5 it is shown that the bottom two levels of the low hierarchy are P and $NP \cap \text{co-}NP$, so that there is a connection between the question whether $P = NP \cap \text{co-}NP$ and the existence of the low hierarchy. We argue in Section 6 that the high hierarchy is, in some sense, a hierarchy of *NP*-completeness notions, in that the first two stages of the high hierarchy coincide with known formulations of *NP*-completeness. We finish this paper by exhibiting a list of open problems concerning these hierarchies.

* This work was supported by Deutsche Forschungsgemeinschaft.

0022-0000/83 $3.00

2. Notation

We suppose all sets to be languages over some fixed alphabet Γ such that $|\Gamma| \geqslant 2$. The complement of $L \subseteq \Gamma^*$ is $\bar{L} = \Gamma^* - L$. For each class of sets C let co-C denote the set of complements of the sets in C, co-$C = \{L | \bar{L} \in C\}$. In some cases we systematically "ignore" the sets $\emptyset$ and Γ^*, hence for each class of sets C we define $C^- = C\text{-}\{\emptyset, \Gamma^*\}$.

A Turing machine M is t time-bounded for some function t on natural numbers iff for all $x \in \Gamma^*$ M makes at most $t(|x|)$ steps before accepting or rejecting. Further, M is polynomial time-bounded iff M is p time-bounded for some polynomial p. Let $L(M)$ denote the set accepted by Turing machine M. The class of sets accepted by deterministic (nondeterministic) polynomial time-bounded Turing machines is denoted by $P(NP)$. Let $L(M, A)$ denote the set accepted by oracle machine M using oracle A. The t time-bounded and polynomial time-bounded oracle machines are defined as above, whereby the time bound is required to hold uniformly for each oracle A. Define $P(A) = \{L(M, A) |\ M$ is a deterministic, polynomial time-bounded oracle machine$\}$ and $NP(A) = \{L(M, A) | M$ is a nondeterministic, polynomial time-bounded oracle machine$\}$. For a class of sets C let

$$P(C) = \bigcup_{A \in C} P(A)$$

and

$$NP(C) = \bigcup_{A \in C} NP(A).$$

For each $A \subseteq \Gamma^*$ let $\sum_0^p(A) = P(A)$ and $\sum_k^p(A) = NP(\sum_{k-1}^p(A))$ for $k \geqslant 1$. The $\sum$ classes of the polynomial hierarchy [28, 30] are defined as follows:

For all $k \geqslant 0$, $\sum_k^p = \sum_k^p(\emptyset)$. Furthermore, let $PH = \bigcup_{k \geqslant 0} \sum_k^p$. It is not known whether the polynomial hierarchy is a proper hierarchy, i.e., whether the inclusion $\sum_k^p \subseteq \sum_{k+1}^p$ is proper for all $k \geqslant 0$. But if $\sum_k^p \neq \sum_{k+1}^p$ holds for some k, then it follows $\sum_i^p \neq \sum_{i+1}^p$ for all $i \leqslant k$.

The following definitions are polynomial time analogs of m-reducibility and T-reducibility in recursion theory: A set A is p-m-reducible to a set B (in symbols, $A \leqslant_m^p B$) iff there is a function f which can be computed by a polynomial time-bounded Turing machine (with output tape) such that for all $x \in \Gamma^*$, $x \in A \Leftrightarrow f(x) \in B$. A set A is p-T-reducible to a set B (in symbols, $A \leqslant_T^p B$) iff $A \in P(B)$.

In the following definitions nondeterministic machines are allowed to produce outputs on a special output tape. Such a machine M defines a binary relation

$$R_M = \{(x, y) |\ \text{there is a computation of } M \text{ on input } x \text{ which produces output } y\}.$$

The following reducibility is from Adleman and Manders [2]; they call it "γ-reducibility." We adopt here the more systematic notation of Long [18] and call it

"strong nondeterministic polynomial time many-one reducibility." A set A is *sn-m*-reducible to a set B (in symbols, $A \leqslant_m^{sn} B$) iff there is a nondeterministic, polynomial time-bounded Turing machine M with output device such that for all $x \in \Gamma^*$:

(i) $\exists y \in \Gamma^*$ $((x, y) \in R_M)$,

(ii) $\forall y \in \Gamma^*$ $((x, y) \in R_M \Rightarrow (x \in A \Leftrightarrow y \in B))$.

Adleman and Manders [3] call the following reducibility "unfaithful γ-reducibility." We say, A is strong nondeterministic polynomial time unfaithfully many-one reducible to B (short, *sn-um*-reducible, in symbols, $A \leqslant_{um}^{sn} B$) iff there is a nondeterministic, polynomial time-bounded Turing machine M with output device such that for all $x \in \Gamma^*$:

(i) $x \in A \Rightarrow \forall y \in \Gamma^*$ $((x, y) \in R_M \Rightarrow y \in B)$,

(ii) $x \notin A \Rightarrow \exists y \in \Gamma^* ((x, y) \in R_M \wedge y \notin B)$.

The *sn-um*-reducibility is weaker than *sn-m*-reducibility: if $x \in A$, then M is allowed to produce no output at all, and if $x \notin A$, then M could produce outputs y such that $y \in B$.

The following definition is from Long [18]: A is *sn-T*-reducible to B (in symbols, $A \leqslant_T^{sn} B$) iff there is a nondeterministic oracle machine M and a polynomial p such that for all $x \in \Gamma^*$:

(i) If $x \in A$, then there exists a computation of M on x using oracle B that goes into the accepting state in at most $p(|x|)$ steps, and no computation of M on x using oracle B goes into the rejecting state.

(ii) If $x \notin A$, then there exists a computation of M on x using oracle B that goes into the rejecting state in at most $p(|x|)$ steps, and no computation of M on x using oracle B goes into the accepting state.

Let C be a class of sets. We call a set A $p-m$ ($p-T$, $sn-m$, $sn-um$, $sn-T$)-complete for C iff $A \in C$ and for all $B \in C$, B is $p-m$ ($p-T$, $sn-m$, $sn-um$, $sn-T$)-reducible to A.

We assume the existence of a polynomial time computable pairing function $\langle \cdot, \cdot \rangle$ (on natural numbers and on Γ^*) whose inverses π_1 and π_2 are also computable in polynomial time. Furthermore, we extend $\langle \cdot, \cdot \rangle$ to n-tuples in the usual way.

3. The K-Operator

Our aim in Section 4 is to define a low and a high hierarchy within NP similar to the jump hierarchies that are defined in [11, 20, 27]. To perform this we first have to define a complexity theoretic analog to the jump operator in recursion theory (see [21]). The jump of a set A, A', is the halting problem relativized to A, i.e.,

$$A' = \{x \mid \text{the oracle machine with representation } x \text{ halts on input } x \text{ using oracle } A\}.$$

The halting problem is an *m*-complete set for the class of recursively enumerable sets. Thus, to define a complexity theoretic jump operator, we could take some *p-m*-complete set for *NP* and relativize it.

The set

$$K = \{\langle M, x, 1^t\rangle | \text{ the nondeterministic Turing machine with representation } M \text{ accepts } x \text{ in } t \text{ or fewer steps}\}$$

is *p-m*-complete for *NP* (cf. [17, Theorem 7.5.2]). Its relativization is

$$K(A) = \{\langle M, x, 1^t\rangle | \text{ the nondeterministic oracle machine with representation } M \text{ accepts } x \text{ in } t \text{ or fewer steps using oracle } A\}.$$

This idea appears also in [5, Lemma 1]. Another approach is to relativize the satisfiability problem for Boolean formulas, *SAT*. This is done in [24].

The important property of the *K*-operator (which parallels with its recursion theoretic counterpart, the jump operator) is formulated in

THEOREM 3.1. *For each set A, K(A) is p-m-complete for* $NP(A) = \sum_1^p(A)$.

Proof (See also [5, Lemma 1]). We describe a nondeterministic oracle machine Q such that $L(Q, A) = K(A)$. On input $z = \langle M, x, 1^t\rangle$, Q guesses nondeterministically a sequence of configurations $C = (C_0, \ldots, C_s)$ such that $s \leqslant t$ and $|C_i| = O(t)$. Then Q verifies that C is an accepting computation of M on input x using A as oracle. (To verify this Q possibly has to query its own oracle A.) Since $|C| = O(t^2) = O(|z|^2)$, Q is polynomial time-bounded, hence $K(A) \in NP(A)$.

Now, let $L \in NP(A)$ be arbitrary. We have to show that $L \leqslant_m^p K(A)$. Let M be a nondeterministic, p time-bounded oracle machine such that $L = L(M, A)$, where p is a polynomial. Then $x \mapsto \langle M, x, 1^{p(|x|)}\rangle$ is the desired *p-m*-reduction:

$$\begin{aligned} x \in L &\Leftrightarrow x \in L(M, A), \\ &\Leftrightarrow M \text{ accepts } x \text{ in } p(|x|) \text{ or fewer steps using oracle } A, \\ &\Leftrightarrow \langle M, x, 1^{p(|x|)}\rangle \in K(A). \quad \blacksquare \end{aligned}$$

Defining $K^0(A) = A$ and $K^n(A) = K(K^{n-1}(A))$ for $n \geqslant 1$, Corollary 3.2 follows easily by an induction argument.

COROLLARY 3.2. (a) *For all natural numbers n and k, if A is p-m-complete in* $\sum_k^p$, *then* $K^n(A)$ *is p-m-complete in* $\sum_{k+n}^p$.

(b) *For all sets L,* $K^n(L)$ *is p-m-complete in* $\sum_n^p(L)$.

THEOREM 3.3. *For each set A,* $A \leqslant_m^p K(A)$.

Proof. Let M be the following oracle machine: on input x, M queries its oracle for x. If the answer is "yes," then M accepts, otherwise M rejects. Now for each set

A, $A = L(M, A)$. Furthermore, M is q time-bounded for some polynomial q. Then $x \mapsto \langle M, x, 1^{q(|x|)} \rangle$ is the desired p-m-reduction from A to $K(A)$. ∎

Considering the converse of Theorem 3.3, we have

THEOREM 3.4. *There exist recursive sets A, B such that*

$$K(A) \leqslant_T^p A \qquad \text{and} \qquad K(B) \not\leqslant_T^p B.$$

Proof. There exist recursive sets A, B such that $P(A) = NP(A)$ and $P(B) \neq NP(B)$ [5]. These sets have the desired properties. ∎

An interesting relation between the K-operator, p-m-reductions, sn-T-reductions, and relativized NP classes is given in

THEOREM 3.5. *The following statements are equivalent:*

(i) $K(A) \leqslant_m^p K(B)$,

(ii) $A \leqslant_T^{sn} B$,

(iii) $NP(A) \subseteq NP(B)$,

(iv) $A \in NP(B)$ *and* $\bar{A} \in NP(B)$.

Proof. (Note that equivalence (iii) ⇔ (iv) is also proved in [23, Theorem 13], and (ii) ⇔ (iv) is proved in [18, Proposition 3.7].)

(i) ⇒ (ii) Let M_0, M_1 be the (trivial) oracle machines such that $L(M_0, A) = A$ and $L(M_1, A) = \bar{A}$ for each set A; M_0, M_1 are polynomial time-bounded. Let p be an appropriate polynomial. Let $K(A) \leqslant_m^p K(B)$ via some polynomial time computable function f. Now we describe a nondeterministic oracle machine M which performs the desired sn-T-reduction from A to B. On input x,

(1) M computes $z = f(\langle M_0, x, 1^{p(|x|)} \rangle)$. Let $M' = \pi_1(z)$, $x' = \pi_2(z)$, and $t' = |\pi_3(z)|$. (We suppose that z has the correct form, otherwise M rejects.) Then M simulates (nondeterministically) the oracle machine M' on input x' for t' steps using oracle B. From any simulated computation of M' which accepts, M accepts. From any simulated computation of M' which does not accept, M proceeds to (2).

(2) M computes $w = f(\langle M_1, x, 1^{p(|x|)} \rangle)$. Then M simulates as in (1) $\pi_1(w)$ on input $\pi_2(w)$ for $|\pi_3(w)|$ steps using oracle B. From any computation of $\pi_1(w)$ which accepts, M rejects, otherwise M does not stop.

Now one verifies that M is a polynomial time-bounded oracle machine of the special form which is required in the definition of sn-T-reducibility, and M witnesses $A \leqslant_T^{sn} B$.

(ii) ⇒ (iii) Let $L \in NP(A)$. Then there exists a nondeterministic polynomial time-bounded oracle machine M such that $L = L(M, A)$. Let $A \leqslant_T^{sn} B$ via M'. An obvious combination of M and M' yields a nondeterministic polynomial time-bounded oracle machine M'' such that $L = L(M'', B)$. Hence, $L \in NP(B)$.

(iii) $\Rightarrow$ (iv) Trivial, because $A \in NP(A) \subseteq NP(B)$ and $\bar{A} \in NP(A) \subseteq NP(B)$.

(iv) $\Rightarrow$ (i) Let $A = L(Q, B)$ and $\bar{A} = L(\bar{Q}, B)$, where Q, $\bar{Q}$ are nondeterministic polynomial time-bounded oracle machines. We have to construct a polynomial time computable function $\langle M, x, 1^t \rangle \mapsto \langle M', x', 1^{t'} \rangle$ such that

M accepts x in t or fewer steps using oracle A if and only if

M' accepts x' in t' or fewer steps using oracle B.

This can be done as follows: M' is constructed from M, Q, and $\bar{Q}$: M' behaves like M. If M queries its oracle for a word w, then M' guesses whether the answer is "yes" or "no." If the guess is "yes," then M' verifies its guess by simulating Q on input w using oracle B. Otherwise, M verifies its guess "no" by simulating $\bar{Q}$ on input w using oracle B. If this verification is successful, then M' returns to the simulation of M in the yes- or no-state, respectively, otherwise M' rejects.

For reasonable representations of nondeterministic oracle machines, computing the representation of M' from the representation of M can be done in time polynomial in the length of the representation of M.

Let q be a monotone increasing polynomial bounding the running time of Q. The longest string which can appear on the oracle tape of M when M is limited to at most t steps has length at most t. Thus, M accepts x in t or fewer steps using oracle A if and only if M' accepts x in $tq(t)$ or fewer steps using oracle B. This implies that $\langle M, x, 1^t \rangle \mapsto \langle M', x, 1^{tq(t)} \rangle$ is the desired p-m-reduction from $K(A)$ to $K(B)$. ∎

Equivalence (i) $\Leftrightarrow$ (ii) of Theorem 3.5 shows that the K-operator provides an isomorphism from the ordering of sn-T-degrees into the ordering of p-m-degrees.

COROLLARY 3.6. (i) *If* $A \leqslant_T^p B$, *then* $K(A) \leqslant_m^p K(B)$.

(ii) *If* $A \leqslant_m^p B$, *then* $K(A) \leqslant_m^p K(B)$.

Proof. (ii) follows from (i), and (i) follows from equivalence (i) $\Leftrightarrow$ (ii) in Theorem 3.5, and the fact that p-T-reducibility implies sn-T-reducibility. ∎

Corollary 3.7 shows that neither $A \leqslant_T^p B$ nor $A \leqslant_m^p B$ are sufficient for $K(A) \leqslant_m^p K(B)$.

COROLLARY 3.7. (a) *There exist recursive sets* A *and* B *such that* $A \not\leqslant_m^p B$ *and* $K(A) \leqslant_m^p K(B)$.

(b) *There exist recursive sets* A *and* B *such that* $A \not\leqslant_T^p B$ *and* $K(A) \leqslant_m^p K(B)$.

(c) *There exist recursive sets* A *and* B *such that* $A \not\leqslant_T^p B$, $K(A) \not\leqslant_T^p K(B)$ *and* $K^2(A) \leqslant_m^p K^2(B)$.

Proof. (a) follows from (b), and for (b) choose a recursive set B such that $P(B) \neq NP(B) = \text{co} - NP(B) = \sum_2^p(B)$ (see [5]). Then for $A = K(B)$, $A \not\leqslant_T^p B$, and $K(A) \leqslant_m^p K(B)$. In [18] it is shown that there exists a recursive set B such that $P(B) \neq NP(B) \neq \sum_2^p(B) = \sum_3^p(B)$. Then for $A = K(B)$, $A \not\leqslant_T^p B$, $K(A) \not\leqslant_T^p K(B)$, and $K^2(A) \leqslant_m^p K^2(B)$. This proves (c). ∎

4. The Hierarchies

Now we give a definition of a low and a high hierarchy within NP similar to the jump hierarchies that are defined in [11, 20, 27]. Note that our definition is not exactly analogous to the recursion theoretic definition, but it seems to be better for the purposes in NP complexity.

Definition 4.1. (i) The low hierarchy: For $n \geqslant 0$ define

$$L_n^p = \{L \in NP^- \mid K^n(L) \text{ is } p\text{-}m\text{-complete for } \textstyle\sum_n^p\}.$$

(ii) The high hierarchy: For $n \geqslant 0$ define

$$H_n^p = \{L \in NP^- \mid K^n(L) \text{ is } p\text{-}m\text{-complete for } \textstyle\sum_{n+1}^p\}.$$

(iii) We call a set low iff it is in L_1^p, and high iff it is in H_1^p.

(iv) Define $LH = \bigcup_{n \geqslant 0} L_n^p$ and $HH = \bigcup_{n \geqslant 0} H_n^p$.

Observe that we exclude the sets $\varnothing$ and Γ^* in our definition, because these sets cause some messy (but trivial) special cases. The following inclusions are immediate from Definition 4.1 and Corollary 3.2:

$$P^- = L_0^p \subseteq L_1^p \subseteq L_2^p \subseteq \cdots \subseteq LH \subseteq NP^-$$

and

$$\{p\text{-}m\text{-complete sets for } NP\} = H_0^p \subseteq H_1^p \subseteq H_2^p \subseteq \cdots \subseteq HH \subseteq NP^-.$$

The question whether these inclusions are proper is open.

Next we give an equivalent definition of the low and the high hierarchy.

Theorem 4.2. (i) *For all* $n \geqslant 0$, $L_n^p = \{L \in NP^- \mid \sum_n^p(L) \subseteq \sum_n^p\}$.

(ii) *For all* $n \geqslant 1$, $H_n^p = \{L \in NP^- \mid \sum_{n+1}^p \subseteq \sum_n^p(L)\}$.

Proof. (i) Let $L \in L_n^p$ for some $n \geqslant 0$. Then $K^n(L)$ is p-m-complete in $\sum_n^p$. But $K^n(L)$ is also p-m-complete in $\sum_n^p(L)$, so it follows that $\sum_n^p(L) \subseteq \sum_n^p$ since $\sum_n^p$ is closed under p-m-reductions. Thus, $L \in \{L \in NP^- \mid \sum_n^p(L) \subseteq \sum_n^p\}$.

Now assume that $\sum_n^p(L) \subseteq \sum_n^p$. Since $\sum_n^p \subseteq \sum_n^p(L)$ for any set L, it follows that $\sum_n^p(L) = \sum_n^p$. But $K^n(L)$ is p-m-complete in $\sum_n^p(L)$, so $K^n(L)$ is now p-m-complete in $\sum_n^p$ and $L \in L_n^p$.

(ii) Let $L \in H_n^p$ for some $n \geqslant 1$. Then $K^n(L)$ is p-m-complete in $\sum_{n+1}^p$. But $K^n(L)$ is also p-m-complete in $\sum_n^p(L)$, so it follows that $\sum_{n+1}^p \subseteq \sum_n^p(L)$ since $\sum_n^p(L)$ is closed under p-m-reductions.

Conversely, assume that $L \in NP^-$ and $\sum_{n+1}^p \subseteq \sum_n^p(L)$. Since for any set $L \in NP^-$, $\sum_n^p(L) \subseteq \sum_{n+1}^p$ (here we have to exclude $n = 0$), it follows that $\sum_n^p(L) = \sum_{n+1}^p$. But $K^n(L)$ is p-m-complete in $\sum_n^p(L)$, so $K^n(L)$ is now p-m-complete in $\sum_{n+1}^p$, hence $L \in H_n^p$. ∎

Observe that the characterization of H_n^p in Theorem 4.2(ii) for $n=0$ yields the class of p-T-complete sets in NP.

We now relate questions concerning whether the low and high hierarchies are infinite and disjoint to similar, well-known questions concerning the polynomial-time hierarchy.

THEOREM 4.3. *For all* $n \geqslant 0$,

(i) $L_n^p = H_n^p$ *or* $L_n^p \cap H_n^p = \emptyset$,

(ii) $\sum_n^p = \sum_{n+1}^p \Rightarrow NP^- = L_n^p = \bigcup_{i \leqslant n} L_i^p = H_n^p = \bigcup_{i \geqslant n} H_i^p$,

(iii) $\sum_n^p \neq \sum_{n+1}^p \Rightarrow L_n^p \cap H_n^p = \emptyset$.

Proof. If $n=0$, the theorem is trivial.

Now suppose that $n \geqslant 1$.

Case 1. $\sum_n^p = \sum_{n+1}^p$. Let $L \in NP^-$. Then it holds $\sum_n^p \subseteq \sum_n^p(L) \subseteq \sum_{n+1}^p$. From the hypothesis $\sum_n^p = \sum_{n+1}^p$ and Theorem 4.2(i) and (ii) it follows that $L \in L_n^p$ and $L \in H_n^p$, hence we have

$$NP^- = L_n^p = \bigcup_{i \geqslant n} L_i^p = H_n^p = \bigcup_{i \geqslant n} H_i^p.$$

Case 2. $\sum_n^p \neq \sum_{n+1}^p$. Suppose, by way of a contradiction, that $L \in L_n^p \cap H_n^p$. Using Theorem 4.2(i) and (ii) it follows $\sum_{n+1}^p \subseteq \sum_n^p(L) \subseteq \sum_n^p$, a contradiction to the case hypothesis. Hence, L_n^p and H_n^p must be disjoint. ∎

COROLLARY 4.4. *The polynomial hierarchy is infinite if and only if* $LH \cap HH = \emptyset$.

Proof. Immediate from Theorem 4.3. ∎

If the polynomial hierarchy is a proper hierarchy up to $\sum_{n+1}^p$, then Theorem 4.3(iii) says that L_n^p and H_n^p are disjoint. But do there exist sets in NP^- which are in neither L_n^p nor H_n^p? Similarly, if the polynomial hierarchy is infinite, then LH and HH are disjoint (Corollary 4.4). But do there exist sets in NP^- which are in neither LH nor HH? To answer these questions we need some additional framework.

DEFINITION 4.5. A class of sets C is recursively presentable iff there is a recursively enumerable sequence of Turing machines $M_0, M_1, M_2, \ldots$ such that $C = \{L(M_i) | i \geqslant 0\}$. Furthermore we require that each Turing machine in such an enumeration halts on each input.

Observe that Definition 4.5 makes sense only for classes of recursive sets.

DEFINITION 4.6. A class of sets C is closed under finite variations iff for all sets A, B: if $A \in C$ and the symmetric difference of A and B, $A \Delta B$, is finite, then $B \in C$.

To answer the questions above we can use Theorem 4.7 which is proved in [25, 26].

THEOREM 4.7. *Let A_1, A_2 be recursive sets and C_1, C_2 be classes of recursive sets with the following properties*:

(a) $A_1 \notin C_1, A_2 \notin C_2$,

(b) C_1, C_2 *are recursively presentable,*

(c) C_1, C_2 *are closed under finite variations.*

Then there exists a recursive set A such that

(d) $A \notin C_1, A \notin C_2$.

(e) *If $A_1 \in P$ and $A_2 \notin \{\emptyset, \Gamma^*\}$, then $A \leqslant_m^p A_2$.*

We can apply Theorem 4.7 in the following manner: Suppose that $\sum_n^p \neq \sum_{n+1}^p$; choose $A_1 = \emptyset$, $A_2 = SAT$, $C_1 = H_n^p$, and $C_2 = L_n^p \cup \{\emptyset, \Gamma^*\}$. (We have to include $\emptyset$ and Γ^* here because of the requirement that the classes must be closed under finite variations.) Then, from the theorem follows the existence of a recursive set A, not in H_n^p, not in $L_n^p \cup \{\emptyset, \Gamma^*\}$, but p-m-reducible to SAT, hence $A \in NP$. In Lemma 4.8 we prove the only missing detail, that the classes H_n^p and $L_n^p \cup \{\emptyset, \Gamma^*\}$ are recursively presentable. To yield a set A which answers the second question, choose in Theorem 4.7 $A_1 = \emptyset$, $A_2 = SAT$, $C_1 = HH$, and $C_2 = LH \cup \{\emptyset, \Gamma^*\}$. What remains to show is that HH and $LH \cup \{\emptyset, \Gamma^*\}$ are recursively presentable.

LEMMA 4.8. (i) *For each $n \geqslant 0$, $L_n^p \cup \{\emptyset, \Gamma^*\}$ is recursively presentable.*

(ii) *$LH \cup \{\emptyset, \Gamma^*\}$ is recursively presentable.*

(iii) *For each $n \geqslant 0$, H_n^p is recursively presentable.*

(iv) *HH is recursively presentable.*

Proof. Let $\{M_j | j \geqslant 0\}$ be an effective enumeration of nondeterministic, polynomial time-bounded Turing machines. It is important to note that the set $\{\langle x, i, n\rangle | x \in K^n(L(M_i))\}$ is recursive. (This is because each M_i halts on each input.) Let $\{T_k | k \geqslant 0\}$ be an effective enumeration of the deterministic, polynomial time-bounded Turing machine transducers.

(i) Let $n \geqslant 0$. We have to construct a recursively enumerable sequence of Turing machines $\{P_i | i \geqslant 0\}$ which each halt on each input, such that $\{L(P_i) | i \geqslant 0\} = L_n^p \cup \{\emptyset, \Gamma^*\}$.

By the characterization of the low hierarchy given in Theorem 4.2(i), $L \in L_n^p$ iff some p-m-complete set in $\sum_n^p(L)$ is p-m-reducible to some p-m-complete set in $\sum_n^p$. A p-m-complete set in $\sum_n^p(L)$ is $K^n(L)$, and for each $A \in P^-$, $K^n(A)$ is p-m-complete in $\sum_n^p$. Thus let $A \in P^-$ be arbitrary. Define the machine P_i, letting $i = \langle j, k\rangle$, as follows:

$P_i(x)$: Test for all y such that $|y| < |x|$, whether

$$y \in K^n(L(M_j)) \Leftrightarrow T_k(y) \in K^n(A).$$

If this test is true for all such y, then
(accept x iff $x \in L(M_j)$) else (accept x iff i is even).

Note that each P_i halts on each input. We verify that $\{L(P_i) | i \geqslant 0\} = L_n^p \cup \{\varnothing, \Gamma^*\}$. Suppose $L = L(P_i)$ for some $i \geqslant 0$, $i = \langle j, k \rangle$. By virtue of the way in which P_i operates, either $L(P_i)$ is a finite or cofinite set, or $L(P_i) = L(M_j)$. In the first case L is trivially in L_n^p, and in the second case $K^n(L) \leqslant_m^p K^n(A)$ via T_k, hence by the above remarks, $L \in L_n^p$. Conversely, let $L \in L_n^p$. Then $L \in NP^-$, hence there exists a j such that $L = L(M_j)$. Furthermore, there exists a k such that $K^n(L) \leqslant_m^p K^n(A)$ via T_k. Now we have $L = L(P_{\langle j,k \rangle})$.

Observe further that among the finite and cofinite sets which are accepted by the machines P_i there is also $\varnothing$ and Γ^*.

(ii) The construction is very similar. The only modification is that we let $i = \langle j, k, n \rangle$, instead of $i = \langle j, k \rangle$.

(iii) Analogous to (i) using the characterization of H_n^p given in Theorem 4.2(iii). From Theorem 4.2(ii) it follows that $L \in H_n^p$ iff some p-m-complete set in $\sum_{n+1}^p$ is p-m-reducible to some p-m-complete set in $\sum_n^p(L)$. (This characterization also holds for $n = 0$.) We only state the suitable machine P_i (letting $i = \langle j, k \rangle$):

$P_i(x)$: Test for all y such that $|y| < |x|$, whether

$$y \in K^{n+1}(A) \Leftrightarrow T_k(y) \in K^n(L(M_j)).$$

If this test is true for all such y, then
(accept x iff $x \in L(M_j)$) else (accept x iff $x \in SAT$).

Now, either $L(P_i) = SAT$ a.e., a p-m-complete set in NP, or $L(P_i) = L(M_j)$, and in this case $L(P_i)$ is in H_n^p by virtue of the way in which P_i operates.

(iv) Let in (iii) $i = \langle j, k, n \rangle$, instead of $i = \langle j, k \rangle$. ∎

COROLLARY 4.9. (i) *For each $n \geqslant 0$, there exist sets in NP^- which are in neither L_n^p nor H_n^p if and only if $\sum_n^p \neq \sum_{n+1}^p$.*

(ii) *There are sets in NP^- which are in neither LH nor HH if and only if the polynomial hierarchy is infinite.*

Proof. The "if" part follows from the discussion above, and the "only if" from Theorem 4.3. ∎

Note that Corollary 4.9(i) with $n = 0$ coincides with Ladner's result [15] that if $P \neq NP$, then there exist sets in $NP - P$ which are not p-m-complete for NP.

5. Low Sets

Low sets are the sets in L_1^p. The next result connects the question whether $P = NP \cap \text{co-}NP$ with the question whether $L_0^p = L_1^p$.

THEOREM 5.1. *The low sets are exactly the sets in* $NP^- \cap \text{co-}NP^-$, i.e., $L_1^p = NP^- \cap \text{co-}NP^-$.

Proof. Let $L \in L_1^p$, then L is in NP^- and $K(L)$ is p-m-complete for NP. It remains to show that $L \in \text{co-}NP^-$. Let M be the following oracle machine: On input x, M queries its oracle for x. If the answer is "no," then M accepts, otherwise M rejects. Here M is q time-bounded for some polynomial q. Now we have

$$\begin{aligned} x \in \bar{L} &\Leftrightarrow x \in L(M, L) \\ &\Leftrightarrow M \text{ accepts } x \text{ in } q(|x|) \text{ or fewer steps} \\ &\Leftrightarrow \langle M, x, 1^{q(|x|)} \rangle \in K(L). \end{aligned}$$

This means that $x \mapsto \langle M, x, 1^{q(|x|)} \rangle$ is a p-m-reduction from $\bar{L}$ to $K(L)$. Since $K(L)$ is p-m-complete for NP, it follows $L \in \text{co-}NP^-$.

Conversely, let $L \in NP^- \cap \text{co-}NP^-$. Then there exist nondeterministic, polynomial time-bounded Turing machines M and $\bar{M}$ such that $L(M) = L$ and $L(\bar{M}) = \bar{L}$. We have to show that $K(L)$ is p-m-complete for NP. For each set $A \in P^-$, $A \leqslant_m^p L$. So it follows (Corollary 3.6(ii)) $K(A) \leqslant_m^p K(L)$. Since $K(A)$ is p-m-complete for NP (Corollary 3.2), each set in NP is p-m-reducible to $K(L)$. It remains to show that $K(L) \in NP$. Let Q be the following nondeterministic Turing machine:

$Q(y)$: Test whether the input y has the form $y = \langle R, x, 1^t \rangle$ for some nondeterministic oracle machine R.
If not, reject y.
Simulate R on input x for t steps.
Accept y if R accepts x within t steps, otherwise reject y.
If R queries its oracle for some word w during this simulation, then guess nondeterministically whether the answer is "yes" or "no."
If the guess is "yes," then start M on input w. Only if M accepts w, continue the simulation of R in the yes state, otherwise reject y.
If the guess is "no," then start $\bar{M}$ on input w. Only if $\bar{M}$ accepts w, continue the simulation of R in the no state, otherwise reject y.

Clearly Q is polynomial time-bounded. Furthermore $L(Q) = K(L)$. Hence $K(L) \in NP$. ∎

Because of the connection between K-operator and sn-T-reducibility (Theorem 3.5), this result is very similar to the observation made in [18, 23] that the 0-degree of sn-T-reducibility is $NP \cap \text{co-}NP$.

6. *NP*-Completeness and the High Hierarchy

The *p-m*-complete sets for *NP* (which are usually simply called "*NP*-complete") have the property that $P = NP$ iff a *p-m*-complete set is in *P*. This has led to some generalizations of the *NP*-completeness notion: Some authors [4, 22] define an *NP* set to be "NP-complete" iff the existence of a polynomial time algorithm for it implies $P = NP$. The notion of "inclusion–complete" sets in [8] is a generalization of this idea.

Another way of generalization is suggested in [3]: What is the weakest known hypothesis which suffices to conclude that a certain class of sets is not included in *P*? For the $p-m$ and *p-T*-complete sets for *NP* this hypothesis is "$P \neq NP$," and for the $sn-m$, $sn-um$, and *sn–T*-complete sets for *NP* this is "$NP \neq$ co-*NP*" (see [2, 3, 18]). The weaker the hypothesis that suffices to conclude that some class of sets is intractable, the stronger is the evidence for intractability of this class.

Now, from Theorem 4.3 we see that the sets in H_n^p are intractable (i.e., not in *P*) if $\sum_n^p \neq \sum_{n+1}^p$. So the high hierarchy generalizes this concept of "hypothesis" and "evidence for intractability" to an infinite hierarchy of hypotheses $\sum_n^p \neq \sum_{n+1}^p$, where the examples "$P \neq NP$" and "$NP \neq$ co-*NP*" are only the first two steps in this hierarchy. (Note that $NP \neq$ co-*NP* is equivalent to $\sum_1^p \neq \sum_2^p$ [6].) The higher a set is in the high hierarchy the weaker is the evidence for intractability of this set. In fact, we already observed that H_0^p is identical to the *p-m*-complete sets for *NP*, and from equivalence (i) $\Leftrightarrow$ (ii) in Theorem 3.5 we immediately get

THEOREM 6.1. *H_1^p is exactly the set of sn-T-complete sets for NP.*

LEMMA 6.2. *If $A \in NP$, then $A \leqslant_{um}^{sn} B$ implies $A \leqslant_T^{sn} B$.*

Proof. Let M_1 be a nondeterministic, p_1 time-bounded Turing machine that accepts *A*, where p_1 is a polynomial. Suppose $A \leqslant_{um}^{sn} B$ via nondeterministic Turing machine M_2 (with output device) and polynomial p_2. We construct a nondeterministic, polynomial time-bounded oracle machine M_3 which *sn-T*-reduces *A* to *B*: M_3 on input *x* first guesses nondeterministically whether $x \in A$ or $x \notin A$, then M_3 verifies its guess as described below. If this verification succeeds, then M_3 accepts or rejects, respectively, otherwise M_3 does not stop.

Verification of "$x \in A$": Run M_1 on input *x*: If M_1 accepts, the verification is successful.

Verification of "$x \notin A$:" Run M_2 on input *x*. Suppose M_2 produces output *y*. Query the oracle *B* for *y*. If $y \notin B$, then the verification is successful. In every other case (either M_2 produces no output or $y \in B$) the verification does not succeed.

COROLLARY 6.3. *The sn − m, sn − um, and sn–T-complete sets for NP are high.*

Proof. This is immediate for the $sn - m$ and sn–T-complete sets for NP. Using Lemma 6.2 we have that the sn–um-complete sets for NP are high. ∎

7. OPEN PROBLEMS

1. Express L_2^p in terms of more common complexity classes (P, NP,...).

2. From Theorem 4.3 it follows that, if $SAT \in L_2^p$, then the polynomial hierarchy collapses to $\sum_2^p$. This is an interesting analogy to the following result: If SAT has polynomial size circuits (or equivalently, if SAT is p-T-reducible to a sparse set [7]), then the polynomial hierarchy collapses to $\sum_2^p$ [14, 12]. Is there any direct connection between L_2^p and the polynomial size circuit sets (or R, the class of randomly decidable sets, which has been shown to have polynomial size circuits [1, 12])?

3. Characterize the graph isomorphism problem in terms of the low or high hierarchy. Is graph isomorphism in L_2^p?

4. Find necessary and/or sufficient conditions for $L_n^p \neq L_{n+1}^p$ (for $H_n^p \neq H_{n+1}^p$). There are some known necessary conditions for $P \neq NP \cap \text{co-}NP$ (hence $L_0^p \neq L_1^p$) (see [29, Proposition 5; 9]) and some sufficient conditions (see [10; 13, Exercise 13.24; 1]). Generalize these conditions to the whole low hierarchy.

5. $\forall n \geqslant 0\ (L_n^p = L_{n+1}^p \Leftrightarrow H_n^p = H_{n+1}^p)$?

6. Long [18, 19] shows that if sn–m-reducibility implies p–m-reducibility then $P = NP \cap \text{co-}NP$ (hence $L_0^p = L_1^p$). Can we use his proof to show that $H_0^p = H_1^p$ implies $L_0^p = L_1^p$? (Remember the connection between sn–m-complete sets for NP and H_1^p.)

7. Do the classes L_n^p for $n \geqslant 1$ possess complete sets? (Cf. the discussion, whether $NP \cap \text{co-}NP$ possesses complete sets in [1].)

8. In [2, 3] there are introduced some sets which are known to be sn–m- or sn–um-complete for NP, and which are not known to be p–m-complete for NP. These sets are also examples of sets being in H_1^p, and which are not known to be in H_0^p. Find "natural" sets in H_k^p, $k \geqslant 2$, which are possibly not in H_{k-1}^p.

ACKNOWLEDGMENT

The author thanks the referee for helpful comments and corrections.

Note added in proof. A solution to problem 2 has been found meanwhile by Ker-I Ko.

REFERENCES

1. L. ADLEMAN, Two theorems on random polynomial time, *in* "Proceedings, 19th Symposium on Foundations of Computer Science," pp. 75–83, 1978.
2. L. ADLEMAN AND K. MANDERS, Reducibility, randomness, and intractability, *in* "Proceedings, 9th Symposium on Theory of Computing," pp. 151–163, 1977.
3. L. ADLEMAN AND K. MANDERS, Reductions that lie, *in* "Proceedings, 20th Symposium on Foundations of Computer Science," pp. 397–410, 1979.
4. A. V. AHO, J. E. HOPCROFT, AND J. D. ULLMAN, "The Design and Analysis of Computer Algorithms," Addison–Wesley, Reading, Mass., 1974.
5. T. BAKER, J. GILL, AND R. SOLOVAY, Relativizations of the $P = ?NP$ question, *SIAM J. Comput.* **4** (1975), 431–442.
6. T. BAKER AND A. L. SELMAN, A second step toward the polynomial hierarchy, *Theoret. Comput. Sci.* **8** (1979), 177–187.
7. L. BERMAN AND J. HARTMANIS, On isomorphism and density of *NP* and other complete sets, *SIAM J. Comput.* **6** (1977), 305–322.
8. R. V. BOOK, C. WRATHALL, A. L. SELMAN, AND D. DOBKIN, Inclusion complete tally languages and the Hartmanis–Berman conjecture, *Math. Systems Theory* **11** (1977), 1–8.
9. A. BORODIN AND A. DEMERS, "Some Comments on Functional Self-Reducibility and the *NP*-Hierarchy," Tech. Rep. TR-76-284, Department of Computer Science, Cornell University, Ithaca, N.Y., 1976.
10. G. BRASSARD, S. FORTUNE, AND J. E. HOPCROFT, "A Note on Cryptography and $NP \cap \text{co}NP$-P," Tech. Rep. TR78-338, Department of Computer Science, Cornell University, Ithaca, N.Y., 1978.
11. R. E. EPSTEIN, "Degrees of Unsolvability: Structure and Theory," Lecture Notes in Mathematics, No. 759, Springer-Verlag, Berlin/New York, 1979.
12. J. E. HOPCROFT, Recent directions in algorithmic research, *in* "5th Conference on Theoretical Computer Science," pp. 123–134, Lecture Notes in Computer Science, No. 104, Springer-Verlag Berlin/New York, 1981.
13. J. E. HOPCROFT AND J. D. ULLMAN, "Introduction to Automata Theory, Languages, and Computation," Addison–Wesley, Reading, Mass., 1979.
14. R. M. KARP AND R. J. LIPTON, Some connections between nonuniform and uniform complexity classes, *in* "Proceedings, 12th Symposium on Theory of Computing," pp. 302–309, 1980.
15. R. E. LADNER, On the structure of polynomial time reducibility, *J. Assoc. Comput. Mach.* **22** (1975), 155–171.
16. R. E. LADNER, N. A. LYNCH, AND A. L. SELMAN, A comparison of polynomial time reducibilities, *Theoret. Comput. Sci.* **1** (1975), 103–123.
17. H. R. LEWIS AND C. H. PAPADIMITRIOU, "Elements of the Theory of Computation," Prentice–Hall, Englewood Cliffs, N. J., 1981.
18. T. J. LONG, "On Some Polynomial Time Reducibilities," Ph.D. Dissertation, Purdue University, Lafayette, Ind., 1978.
19. T. J. LONG, On γ-reducibility versus polynomial time many-one reducibility, *in* "Proceedings, 11th Symposium on Theory of Computing," pp. 278–287, 1979.
20. D. B. POSNER, A Survey of non-r.e. degrees $\leqslant 0'$, *in* "Recursion Theory: Its Generalisations and Applications," pp. 52–109, Cambridge Univ. Press, London/New York, 1980.
21. H. ROGERS, "Theory of Recursive Functions and Effective Computability," McGraw–Hill, New York, 1967.
22. S. SAHNI, Computationally related problems, *SIAM J. Comput.* **3** (1974), 262–279.
23. A. L. SELMAN, Polynomial time enumeration reducibility, *SIAM J. Comput.* **7** (1978), 440–457.
24. U. SCHÖNING, A note on complete sets for the polynomial time hierarchy, *SIGACT News* **13** (1981), 30–34.
25. U. SCHÖNING, "Untersuchungen zur Struktur von *NP* und verwandten Komplexitätsklassen mit Hilfe verschiedener polynomieller Reduktionen," Dissertation, Stuttgart, West Germany, 1981.

26. U. SCHÖNING, A uniform approach to obtain diagonal sets in complexity classes, *Theoret. Comput. Sci.* **18** (1982), 95–103.
27. R. I. SOARE, Recursively enumerable sets and degrees, *Bull. Amer. Math. Soc.* **84** (1978), 1149–1181.
28. L. J. STOCKMEYER, The polynomial-time hierarchy, *Theoret. Comput. Sci.* **3** (1977), 1–22.
29. L. G. VALIANT, Relative complexity of checking and evaluating, *Inform. Process. Lett.* **5** (2) (1976), 20–23.
30. C. WRATHALL, Complete sets and the polynomial-time hierarchy, *Theoret. Comput. Sci.* **3** (1977), 23–33.