Theoretical Computer Science 34 (1984) 17-32
North-Holland

COMPUTATION TIMES OF NP SETS OF DIFFERENT DENSITIES

J. HARTMANIS* and Y. YESHA**

Department of Computer Science, Cornell University, Ithaca, NY 14853, U.S.A.

Abstract. In this paper we study the computational complexity of sets of different densities in NP. We show that the deterministic computation time for sets in NP can depend on their density if and only if there is a collapse or partial collapse of the corresponding higher nondeterministic and deterministic time bounded complexity classes. We show also that for NP sets of different densities there exist complete sets of the corresponding density under polynomial time Turing reductions. Finally, we show that these results can be interpreted as results about the complexity of theorem proving and proof presentation in axiomatized mathematical systems. This interpretation relates fundamental questions about the complexity of our intellectual tools to basic structural problems about P, NP, CoNP and PSPACE, discussed in this paper.

Introduction

The general motivation for this work is the need and desire to understand what makes the solution of NP problems hard, provided $P \neq NP$. The fundamental question is whether the deterministic computation time required to solve NP problems could depend on the density of the set of problems under consideration. In other words, is the problem of finding satisfying assignments for Boolean formulas in conjunctive normal form, SAT, computationally hard because there are exponentially many formulas up to size n and that no one single method can solve them all easily? Or is the satisfiability problem still hard if we consider only 'thinned out' sets of formulas whose density is much lower than exponential?

It has been shown recently that the structural properties of lower density sets in NP are directly determined by the relations between the corresponding higher deterministic and nondeterministic time bounded complexity classes. We cite one such result next [8, 9].

A set S is said to be *sparse* if S contains only polynomially many elements up to size n, i.e., $|S \cap (\varepsilon + \Sigma)^n| \leq n^k + k$. Let

$$\text{NEXPTIME} = \bigcup_{c \geq 1} \text{NTIME}[2^{cn}], \qquad \text{EXPTIME} = \bigcup_{c \geq 1} \text{TIME}[2^{cn}]$$

* The work of this author was supported in part by National Science Foundation Grants MCS78-00418 and MCS83-01766.

** The work of this author was supported in part by a Dr. Chaim Weizmann Post-Doctoral Fellowship for Scientific Research.

and

$$\text{EXPSPACE} = \bigcup_{c>1} \text{SPACE}(2^{cn}).$$

Theorem A. *There exist sparse sets in* NP − P *if and only if* NEXPTIME ≠ EXPTIME.

For related results about tally sets, see [5].

In the present paper we continue this study and show that the deterministic computation speed of sets in NP can depend on their density if and only if the corresponding higher deterministic and nondeterministic complexity classes have collapsed or partially collapsed.

We first show that there are sets of prescribed densities in NP and PSPACE which are complete under polynomial time Turing reductions for all other sets of the same density in NP and PSPACE, respectively. We cite one such result.

Theorem B. *There exists a sparse set* S_0 *in* NP *such that all other sparse sets in* NP *are in* P^{S_0}.

This completeness result contrasts with the well-known results by Mahaney [11] and Karp and Lipton [10]. The first result asserts that if there exists a sparse, many-one complete set for NP, then P = NP. The Karp-Lipton result shows that if there exists a sparse set S such that $\text{NP} \subseteq \text{P}^S$, then the polynomial time hierarchy collapses to Σ_2^{P}. Our results show that as long as we restrict ourselves to sparse sets in NP, there exist sparse complete sets. At the same time it is interesting to note that the same results do not seem to hold for CoNP, or at least they do not hold for relativized CoNP computations whereas the above results also hold for relativized NP computations [9]. We also show that there are relativized computations for which there do not exist sparse sets in NP which are complete for all other sparse sets in NP under many-one polynomial time reductions.

From Theorem B we immediately obtain a proof of the previously known Theorem A as well as new results about the relation between partial collapse of higher deterministic and nondeterministic computations and the recognition speed of sparse sets.

Theorem C. *All sparse sets in* NP *are in*

$$\bigcup_{c>1} \text{TIME}[n^{c \log n}]$$

if and only if

$$\text{NEXPTIME} \subseteq \bigcup_{c>1} \text{TIME}[2^{cn}].$$

Related results are derived for sets of other densities and computation times as well as for PSPACE versus NP and PSPACE versus P.

From all these results we see that the deterministic time complexity of sets in NP can depend on their density if and only if the corresponding higher deterministic and nondeterministic time classes have suffered a collapse or partial collapse. Since it is our thesis that the density of sets in NP and PSPACE cannot affect their computation time, we are lead to the *generalized complexity hypothesis.* This conjecture asserts for NP (i.e., the *generalized* NP *hypothesis*) that SAT requires roughly deterministic exponential time and that the deterministic recognition time of sets in NP does not depend on their density. This clearly implies, because of our results, that the higher deterministic and nondeterministic time classes have not even partially collapsed. For example, we conjecture that there exist sets in NEXPTIME which require roughly doubly exponential deterministic recognition time. The *generalized* PSPACE *hypothesis* versus P as well as NP is formulated similarly.

Intuitively, the generalized NP hypothesis asserts that the computational difficulty of finding assignments for Boolean formulas in SAT does not stem from the existence of the aggregate of such formulas, but that the difficulty is inherent even in very sparse subsets of SAT.

We give an interpretation of these results in terms of the computational complexity of doing mathematics. We assume that we are using Peano Arithmetic, F. Let

$$L_1 = \{\textbf{THEOREM: ``Statement of result''. PROOF: } b^k\square \mid \text{There is a proof of length } k \text{ or less of the stated theorem in } F\}.$$

It is easily seen that L_1 is an NP complete set.

Similarly the set

$$L_2 = \{\textbf{THEOREM: ``Statement of result''. PRESENTATION OF PROOF: } b^k\square \mid \text{There is a proof of the stated theorem in } F \text{ which can be presented on tape of length } k\}$$

is PSPACE complete. By presentation of proof we mean a formal writing down of the proof so that a simple proof checker can guarantee that the theorem has a proof, but we can erase any part of the proof not needed later. Thus, when the presentation is completed, the verifier knows that a proof exists, but there may not be a complete proof written down.

Clearly, PSPACE $\neq$ NP if and only if $L_2 \notin$ NP and this happens if and only if in Peano Arithmetic there are infinitely many theorems for which the difference in the length of the shortest proof and the space needed to present a proof is not polynomially bounded.

Similarly, the same relationship will hold for sparse subsets of L_2 which are in PSPACE, even if we are allowed to design specialized proof systems for these restricted subsets, if and only if EXPSPACE $\neq$ NEXPTIME.

Corollary D. *There exist sparse sets in* PSPACE $-$ NP *if and only if* EXPSPACE $\neq$ NEXPTIME, *and the quantitative difference between proof length and length of proof presentation depends on the quantitative difference between* EXPSPACE *and* NEXPTIME.

Furthermore, we observe that the existence of sparse subsets of tautologies in CoNP − NP implies that for these sparse subsets we cannot design special proof rules to prove in polynomial length that they are tautologies. This is so because if a sparse subset of TAUT is not in NP, then we know that there cannot exist a proof system which proves these formulas to be tautologies with polynomially long proofs.

We prove the following result.

Theorem E. *There exists a sparse set S in* P *such that*

$$S \cap \text{TAUT} \in \text{CoNP} - \text{NP}$$

if and only if

$$\text{CoNEXPTIME} \neq \text{NEXPTIME}.$$

Thus, if and only if CoNEXPTIME ≠ NEXPTIME can we find a syntactically restricted sparse subset (a sparse set S in P) of Boolean formulas for which we cannot find a good proof system that would yield polynomially long proofs for formulas in $S \cap$ TAUT. Furthermore, the actual length of the possible (not polynomially bounded) proofs for $S \cap$ TAUT is given by the disparity between CoNEXPTIME and NEXPTIME.

Now, let PSIZE denote the class of languages which have polynomial size circuits (see, for instance, [4]). Using similar methods we show that EXPSPACE ≠ EXPTIME if and only if there exists a language in PSPACE which is not in P, but has polynomial size circuits. Formally, we can state the following theorem.

Theorem F

$$\text{EXPSPACE} \neq \text{EXPTIME} \Leftrightarrow \text{PSPACE} \cap \text{PSIZE} \neq \text{P}.$$

As a corollary we get an 'upward separation' result as follows.

Corollary G. *Let C be a class of languages such that* $\text{P} \subset C \subset \text{PSPACE}$ *and* $C \subset \text{PSIZE}$. *Then,*

$$C \neq \text{P} \Rightarrow \text{EXPSPACE} \neq \text{EXPTIME}.$$

As an application, we could have $C = R$ or $C =$ BPP. (R and BPP are probabilistic polynomial time classes. They are defined in [7]. There, R is called VPP.)

From the above comments we see that the classic problems about $\text{P} \stackrel{?}{=} \text{NP} \stackrel{?}{=} \text{PTAPE}$, $\text{NP} \stackrel{?}{=} \text{CoNP}$ etc., are really questions about the complexity of our intellectual tools, namely mathematics. Correspondingly, our work tries to address the fundamental question of what makes these problems hard and whether restricting them to subsets of lower density can make them simpler to compute. Our results show that the lower density problems can become computationally easier than the unrestricted problems

if and only if there is a partial collapse of the differences between the corresponding higher complexity classes.

1. Sparse complete sets and the structure of NP

In this section we show that the computational complexity of sets of different densities in NP and PSPACE are completely determined by the relations between the corresponding higher complexity classes.

The main tool in this study will be the existence of sparse sets in NP which are complete for all other sparse sets in NP. Let $\leq_T^P$ denote polynomial time Turing reducibility.

Theorem 1.1. *There exists a tally set S_0 in* NP, $S_0 \subseteq 1^*$, *such that all sparse sets in* NP *are $\leq_T^P$ reducible to S_0, i.e.*

$$\{S \mid S \text{ sparse and in NP}\} \subseteq P^{S_0}.$$

Proof. Let A be a complete set of NEXPTIME under many-one linear time reductions and let

$$S_0 = \text{TALLY}(A) = \{1^n \mid n \in 1A\}.$$

Note that $S_0 \in$ NP.

Let S be a sparse set in NP, say

$$|S \cap (\varepsilon + \Sigma)^n| \leq n^{k_0} + k_0.$$

Then the set

$$B = \{(n, r) \mid |S \cap (\varepsilon + \Sigma)^n| \geq r\} \quad \text{is in NEXPTIME,}$$

since for n and r represented in binary one has enough nondeterministic time to guess r strings in S, $r \leq n^{k_0} + k_0$, and verify that they are in S. Hence, B is many one linear time reducible to A, and the corresponding set

$$B' = \{(1^n, 1^r) \mid |S \cap (\varepsilon + \Sigma)^n| \geq r\}$$

is polynomial time many-one reducible to S_0. Hence, $B' \in P^{S_0}$. Since P^{S_0} is closed under complement we see that

$$B'' = \{(1^n, 1^{r_n}) \mid |S \cap (\varepsilon + \Sigma)^n| = r_n\} \quad \text{is in } P^{S_0}.$$

Thus in P^{S_0} we can compute the exact number of elements in S up to size n, namely r_n.

Furthermore, the set

$$C = \{(n, i, j, k, d) \mid (\exists x_1 < x_2 < \cdots < x_i = x < y_1 < y_2 < \cdots < y_j)$$
$$[|y_j| \leq n \text{ and } x_r, y_t \in S \text{ for } 1 \leq r \leq i \text{ and } 1 \leq t \leq j$$
$$\text{and } |x| = n \text{ and the } k\text{th digit of } x \text{ is } d]\}$$

is in NEXPTIME since in nondeterministic time 2^{cn} one can guess the appropriate strings, verify that they satisfy the required conditions and are in S. But then the corresponding set C' obtained by replacing (n, i, j, k, d) by $(1^n, 1^i, 1^j, 1^k, d)$ is in P^{S_0}, by the same argument used to show that $B' \in P^{S_0}$. Since B'' is in P^{S_0}, for any x such that $|x| = n$ we can compute r_n and then, using C', check for $1 \leq i \leq r_n$, $1 \leq j \leq r_n$ such that $i + j = r_n$, whether $x = x_i$ for x_i in S. Therefore we conclude that

$$S \in P^{S_0},$$

as was to be shown. □

Later in this paper we will investigate the possibility that there exists an $S_0 \subseteq 1^*$ which is many-one complete for all sparse sets in NP, and show that there exist relativized computations for which this is not true (though Theorem 1.1 holds for relativized computations).

From the first theorem we immediately obtain a known result about the collapse of higher deterministic and nondeterministic time bounded complexity classes [8, 9], as well as a set of new results about partial collapse of these classes.

Corollary 1.2. EXPTIME = NEXPTIME *if and only if there are no sparse sets in* NP − P.

Proof. If EXPTIME = NEXPTIME, then a complete set A of NEXPTIME is in EXPTIME and therefore TALLY$(A) = S_0$ is in P. But then all sparse sets in NP are in P.

Conversely, if a sparse set S is in NP − P, then S_0 is not in P hence $A \notin$ EXPTIME and therefore EXPTIME $\neq$ NEXPTIME. □

We say that a set S is P-*printable* if and only if for input 1^n in polynomial time we can print all the elements of S up to size n. Clearly, every P-printable set is sparse and in P.

Similarly, we define a set S to be NP-*printable* if and only if there exists a nondeterministic polynomial time machine such that for input 1^n there exists a computation which prints exactly all the elements of S of length at most n, and every computation either prints exactly those elements or halts with indication of failure to print. Clearly, every NP-printable set is sparse and in NP ∩ CoNP.

The proofs of the previous results yield the following.

Corollary 1.3. EXPTIME = NEXPTIME *if and only if every sparse set in* NP *is* P-*printable.*

Proof. If every sparse set in NP is P-printable, then clearly every such set is in P. Hence EXPTIME = NEXPTIME [9]. Conversely, suppose that EXPTIME = NEXPTIME, and let S be a sparse set in NP. Then $S \in$ P [9]. Let

$$S' = \{(1^n, 1^r) \mid |S \cap (\Sigma + \Sigma)^n| \geq r\}.$$

S' is a sparse set in NP, hence, by [9], $S' \in$ P. Since P is closed under complement, we see that

$$S'' = \{(1^n, 1^r) \mid |S \cap (\varepsilon + \Sigma)^n| = r\}$$

is in P.

Now let

$$S''' = \{(1^n, 1^i, 1^k, 1^d) \mid \text{the } i\text{th (in lexicographic order) element of } S \text{ has length at most } n, \text{ and its } k\text{th digit is } d\}.$$

S''' is clearly sparse. S''' is also in NP: Using S'' we can in deterministic polynomial time compute $r = |S \cap (\varepsilon + \Sigma)^n|$. Then we can guess distinct $x_1, x_2, \ldots, x_r$ of length at most n ($x_1 < x_2 < \cdots < x_r$), and verify that $x_j \in S$ ($1 \leq j \leq r$). Thus we can check whether x_i has kth digit d. Since NEXPTIME = EXPTIME, $S''' \in$ P by [9]. Using S''' it is clear that we can print $S \cap (\varepsilon + \Sigma)^n$ in time polynomial in n. □

Next we show that the upward separation method yields necessary and sufficient conditions also for NP-printability.

Theorem 1.4. NEXPTIME = CONEXPTIME *if and only if every sparse set in* NP *is* NP-*printable*.

Proof. Assume NEXPTIME = CONEXPTIME, let S be a sparse set in NP and define

$$L = \{(n, i) \mid |S \cap (\varepsilon + \Sigma)^n| \geq i\},$$

where n and i are represented in binary. Clearly, for any (n, i) in nondeterministic exponential time a machine can guess i different strings up to size n and verify that they are in S. Therefore, L is in NEXPTIME and since NEXPTIME = CONEXPTIME we can use a nondeterministic exponential time machine to check if (n, i) is in $\bar{L}$. Clearly $i_n = |S \cap (\varepsilon + \Sigma)^n|$ is given by $(n, i_n) \in L$ and $(n, i_n + 1) \in \bar{L}$. Thus we see that

$$L' = \{(n, i_n) \mid |S \cap (\varepsilon + \Sigma)^n| = i_n\} \in \text{NEXPTIME}$$

and therefore

$$L'' = \{(1^n, i_n) \mid |S \cap (\varepsilon + \Sigma)^n| = i_n\} \in \text{NP}.$$

But then a nondeterministic polynomial time machine for input 1^n can print

$$x_1 < x_2 < \cdots < x_j < \cdots < x_{i_n} \quad \text{for } 1 \leq j \leq i_n, |x_j| \leq n, x_j \in S,$$

by first guessing i_n and verifying that it is a correct guess and then guessing i_n distinct strings of S of length at most n and printing them if the guess is verified (if not the machine fails to print). Thus S is NP-printable.

Assume that every sparse set in NP is NP-printable and let A be a set in NEXPTIME. Then TALLY(A) = $\{1^n \mid n \in 1A\}$ is a sparse set in NP and therefore NP-printable, but then $\overline{\text{TALLY}(A)}$ is also in NP and we see that $\bar{A}$ is in NEXPTIME. But then

CoNEXPTIME $\subseteq$ NEXPTIME and therefore

$$\text{CoNEXPTIME} = \text{NEXPTIME}. \quad \square$$

From Theorem 1.4 we can obtain a further characterization of the NEXPTIME = CoNEXPTIME collapse.

Corollary 1.5. NEXPTIME = CoNEXPTIME *if and only if, for all sparse set* S *in* NP, $\bar{S}$ *is in* NP.

Proof. Since NEXPTIME = CoNEXPTIME implies that S in NP is NP-printable by Theorem 1.4, we immediately see that $\bar{S}$ is in NP.

Conversely, if, for every sparse set T in NP, $\bar{T}$ is also in NP, then we see that for any sparse set S in NP the set

$$L'' = \{(1^n, i_n) \mid |S \cap (\varepsilon + \Sigma)^n| = i_n\} \quad \text{is in NP}$$

and therefore S is NP-printable. Therefore,

$$\text{NEXPTIME} = \text{CoNEXPTIME}$$

by Theorem 1.4. $\square$

Next we show that a partial collapse of the higher deterministic and nondeterministic complexity classes directly determines the computation time of the lower density sets in NP and PSPACE. We first prove, as an example, a special case of our general result.

Theorem 1.6. *For all* $k > 1$, $\text{NEXPTIME} \subseteq \bigcup_{c \geq 1} \text{TIME}[2^{cn^k}]$ *if and only if all sparse sets in* NP *are in*

$$\bigcup_{c \geq 1} \text{TIME}[n^{c(\log n)^{k-1}}].$$

Proof. If $\text{NEXPTIME} \subseteq \bigcup_{c \geq 1} \text{TIME}[2^{cn^k}]$, then, for a complete set A of NEXPTIME,

$$\text{TALLY}(A) = S_0 \quad \text{is in } \text{TIME}[2^{d(\log n)^k}] = \text{TIME}[n^{d(\log n)^{k-1}}].$$

But then by Theorem 1.1 every sparse set S in NP is in P^{S_0} and

$$S \in \text{P}^{S_0} \subseteq \bigcup_{c \geq 1} \text{TIME}[n^{c(\log n)^{k-1}}].$$

Conversely, if every sparse set of NP is in

$$\bigcup_{c \geq 1} \text{TIME}[n^{c(\log n)^{k-1}}],$$

then so is S_0 and we see that

$$A \in \text{TIME}[2^{rn^k}]$$

for some r. But then

$$\text{NEXPTIME} \subseteq \bigcup_{c \geq 1} \text{TIME}[2^{cn^k}]. \qquad \square$$

Related results can easily be derived for PSPACE versus NP and PSPACE versus P. More specifically, for all $k > 1$, $\text{EXPSPACE} \subseteq \bigcup_{c \geq 1} \text{NTIME}[2^{cn^k}]$ if and only if all sparse sets in PSPACE are in $\bigcup_{c \geq 1} \text{NTIME}[n^{c(\log n)^{k-1}}]$. $\text{EXPSPACE} \subseteq \bigcup_{c \geq 1} \text{TIME}[2^{cn^k}]$ if and only if all sparse sets in PSPACE are in $\bigcup_{c \geq 1} \text{TIME}[n^{c(\log n)^{k-1}}]$.

The above results can easily be generalized to any well behaved computation times.

Theorem 1.7. *Let $f(n) \geq n$ be nondecreasing and fully-time-constructible. Then:*

(1) $\text{NEXPTIME} \subseteq \bigcup_{d \geq 1} \text{TIME}[2^{d(f(dn+d))}]$ *if and only if every sparse set in* NP *is in* $\bigcup_{d \geq 1} \text{TIME}[2^{d(f(d \log n + d))}]$.

(2) $\text{CoNEXPTIME} \subseteq \bigcup_{d \geq 1} \text{NTIME}[2^{d(f(dn+d))}]$ *if and only if the complement of every sparse set in* NP *is in* $\bigcup_{d \geq 1} \text{NTIME}[2^{d(f(d \log n+d))}]$.

Results about sets of higher than polynomial density are correspondingly related to higher complexity classes below exponential time.

We say that a set S has density $\sigma(n)$ if

$$|S \cap (\varepsilon + \Sigma)^n| \leq \sigma(n).$$

Theorem 1.8. *There are no $\sigma(n) = n^{\log n}$ dense sets in*

$$\text{PSPACE} - \text{NP}$$

if and only if

$$\bigcup_{c \geq 1} \text{SPACE}[2^{c\sqrt{n}}] = \bigcup_{c \geq 1} \text{NTIME}[2^{c\sqrt{n}}].$$

We can derive similar results for NP if we assume that our lower density sets are *uniformly distributed*. This property was introduced in [9]: A set A of density $\delta(n)$ is called *uniformly distributed* if and only if every interval of length $2^n/\delta(n)$ contains at most polynomially many elements of A of length at most n, where an *interval* is any set of strings consecutive in the lexicographic ordering of Σ^*.

Theorem 1.9. *There are no $\sigma(n) = n^{\log n}$ uniformly dense sets in* NP − P *if and only if*

$$\bigcup_{c \geq 1} \text{TIME}[2^{c\sqrt{n}}] = \bigcup_{c \geq 1} \text{NTIME}[2^{c\sqrt{n}}],$$

and in this case

$$\text{SAT} \in \bigcup_{c \geq 1} \text{TIME}[2^{c\sqrt{n}}].$$

Finally, we list an illustrative result about partial collapse of subexponential complexity classes.

Theorem 1.10. $\bigcup_{c>1} \text{SPACE}[2^{cn}] \subseteq \bigcup_{c>1} \text{TIME}[2^{cn^k}]$ *if and only if all* $\sigma(n) = n^{\log n}$ *dense sets of* PSPACE *are in*

$$\bigcup_{c>1} \text{TIME}[n^{c(\log n)^{k-1}}].$$

2. On many-one complete sparse sets

The existence of a tally set S_0 in NP such that all other sparse sets in NP are in P^{S_0} raises the question whether there exists a tally set which is many-one polynomial time complete for all sparse sets in NP.

Our results show that there exist relativized computations for which no tally set S_0 can be complete for all sparse sets in NP under many-one reductions. At the same time, it is easily seen that Theorem 1.1 holds for relativized computations and therefore for any oracle A there exists a sparse set complete for all other sparse sets in NP^A under Turing reducibility.

Let $\leq_m^P$ denote polynomial time many-one reducibility. Let Σ_i^E denote the Σ-levels of the exponential hierarchy, i.e.,

$$\Sigma_0^E = \text{EXPTIME}, \qquad \Sigma_1^E = \text{NEXPTIME},$$

$$\Sigma_2^E = \text{NEXPTIME}^{\Sigma_1^P} = \text{NEXPTIME}^{\text{SAT}}, \quad \text{etc.}$$

We first prove a technical result which shows that for some oracle A there do not exist tally sets which are $\leq_m^P$-complete for all sparse sets in NP^A.

Lemma 2.1. *Let* $S_0 \subseteq 1^*$ *and assume that for all sparse sets* S *in* NP *we have* $S \leq_m^P S_0$. *Then* $\text{NEXPTIME} = \Sigma_2^E$ *implies that*

$$\text{NEXPTIME} = \text{EXPTIME}.$$

Proof. We first observe that it is sufficient to show that every set S of the form $S = T \cap \text{SAT}$, where T is a sparse set in P, must be in P, since then, by [9], $\text{NEXPTIME} = \text{EXPTIME}$. The assumption

$$\Sigma_2^E = \Sigma_1^E = \text{NEXPTIME}$$

implies that for any sparse set $S = T \cap \text{SAT}$, T sparse and in P, the set

$$S_1 = \{(F_i, x_i) \mid F_i \in S \text{ and } x_i \text{ is the minimal solution of } F_i\}$$

is also in NP. (We assume that $S = \{F_1, F_2, \ldots, F_n, \ldots\}$ in lexicographic order.) This is shown as follows. Note that S_1 is a sparse set in Σ_2^P.

Let

$$S_2 = \{(1^n, 1^j, 1^{k_1}, d, 1^{k_2}, d') \mid \text{the } k_1\text{th binary digit of } F_j \text{ is } d \text{ and the } k_2\text{th binary digit of } x_j \text{ is } d', \text{ and } |F_j| = n \text{ and } k_1 \leq n\}.$$

Clearly S_2 is sparse. Also $S_2 \in \Sigma_2^P$: A Σ_2 machine can in polynomial time, given $(1^n, 1^j, 1^{k_1}, d, 1^{k_2}, d')$, existentially guess formulas $G_1, G_2, \ldots, G_r$ $(r \geq j)$ of length at most n $(G_1 < G_2 < \cdots < G_r)$ such that the k_1th digit of G_j is d and $|G_j| = n$, and verify that all of them are in S. Then guess a solution x_j of G_j whose k_2th digit is d' and verify that it is a solution. Then universally verify that every formula G of length at most n such that $G \notin \{G_1, G_2, \ldots, G_r\}$ is not in S (since $S \in$ NP, the predicate $G \notin S$ is in CoNP and can be verified by a Π_1^P machine). Thus it is established that $G_j = F_j$. Then universally verify that, for every $x < x_j$, x is not a solution of F_j.

Thus indeed S_2 is in Σ_2^P. Let A_2 be obtained from S_2 by replacing $(1^n, 1^j, 1^{k_1}, d, 1^{k_2}, d')$ by (n, j, k_1, d, k_2, d'). Then $A_2 \in \Sigma_2^E$. Given (n, j, k_1, d, k_2, d'), compute $(1^n, 1^j, 1^{k_1}, d, 1^{k_2}, d')$ in deterministic exponential time. Then check whether $(1^n, 1^j, 1^{k_1}, d, 1^{k_2}, d')$ is in S_2 by a Σ_1^P oracle machine with SAT as an oracle (since $S_2 \in \Sigma_2^P = \Sigma_1^{P(SAT)}$). Since the length of $(1^n, 1^j, 1^{k_1}, d, 1^{k_2}, d')$ is bounded by an exponential in the length of (n, j, k_1, d, k_2, d') we immediately see that A_2 is accepted by a Σ_1^E oracle machine with oracle SAT. Now, since we have assumed NEXPTIME^{SAT} = NEXPTIME, we conclude that $A_2 \in$ NEXPTIME, hence $S_2 \in$ NP. Hence also $S_1 \in$ NP: given (F, z) where $|F| = n$, guess a $j, j \leq n^w + w$ (where $|S \cap (\varepsilon + \Sigma^n)| \leq n^w + w$). Then for $1 \leq k_1 \leq |F|$, $1 \leq k_2 \leq |x_j|$ verify that $(1^n, 1^j, 1^{k_1}, d_{k_1}, 1^{k_2}, d'_{k_2}) \in S_2$, where d_{k_1} is the k_1th digit of F and d'_{k_2} is the k_2th digit of x. This proves that $F = F_j$ and $x = x_j$. Thus indeed $S_1 \in$ NP as was to be shown. For F_i in S let F_i^k denote F_i with its first k variables, $0 \leq k \leq |x_i|$, filled in with the values of its minimal solution x_i (we choose our syntax so that $|F_i^k| = |F_i|$). Then

$$S' = \{F_i^k \mid F_i \in S \text{ and } 0 \leq k \leq |x_i|\}$$

is seen to be a sparse set in NP, since S_1 is in NP.

We now use a modification of Berman's tree search method [2] to decide S in polynomial time using $S' \leq_M^P S_0 \subseteq 1^*$. There exists a function g, computable in polynomial time such that $(\forall x \in \Sigma^*)(x \in S' \Leftrightarrow g(x) \in S_0)$. Also

$$(F, z_1 z_2 \ldots z_r) \in S' \Leftrightarrow [(F, z_1 z_2 \ldots z_r 0) \in S' \text{ or } (F, z_1 z_2 \ldots z_r 1) \in S']. \qquad (*)$$

Given a formula F, first check whether $F \in T$ in polynomial time. Reject if $F \notin T$. Otherwise, clearly $F \in S \Leftrightarrow F \in$ SAT. Perform a depth-first-search on the binary tree of self reductions [2, 11] of F which is defined as follows: The root is F. The left son of F is $(F, 0)$ and the right son of F is $(F, 1)$. Inductively, given a node $(F, z_1 z_2 \ldots z_r)$ of the tree where $r \leq m$, m being the number of variables in the formula F, then $(F, z_1 z_2 \ldots z_r)$ is a leaf if $r = m$. Otherwise, if $r < m$, its left son is $(F, z_1 z_2 \ldots z_r 0)$ and its right son is $(F, z_1 z_2 \ldots z_r 1)$.

In our search, a left son of a node is always searched first. The tree is not constructed in advance. New nodes are constructed as they are searched. Each time a leaf (F, x) is encountered, we check whether x is a solution of F. Since the leaves are encountered in lexicographic order, the first solution encountered must be the minimal solution of F. Hence by (*) above, *any modification of the search which will avoid searching some subtrees whose root is not in S' will still arrive at this minimal solution if $F \in S$.* Hence we indeed use such a modification in order to complete the search in polynomial time.

We add the following rules:

(R1) Every searched leaf (F, x) such that x is not a solution of F is marked by U (meaning that $(F, x) \notin S'$).

(R2) If the two sons of a node are marked U, mark this node by U.

(R3) For each searched node (F, x) compute $y = g((F, x))$. Mark the node by U if $y \notin 1^*$ or if $y = g((F', x'))$ where (F', x') is already marked by U.

(R4) After a node is marked by U, never search below this node.

The above rules guarantee that the search will be completed in polynomial time [2, 11], either yielding the minimal solution of F, or determining that $F \notin S$. □

Corollary 2.2. *There exists an oracle A such that no tally set can be $\leq^{P}_{M}$-complete for all sparse sets in* NP^A.

Proof. Since there exists an oracle A such that [12]

$$\text{EXPSPACE}^A = \cdots = \Sigma_2^{E(A)} = \Sigma_1^{E(A)} \neq \Sigma_0^{E(A)},$$

a relativized version of the previous lemma implies that there cannot exist a tally set $\leq^{P}_{M}$-complete for all sparse sets of NP^A. □

Furthermore, from [9, Theorem 12] it follows that there exists an oracle A such that no tally set can be even $\leq^{P}_{T}$-complete for all sparse sets in $\mathcal{C}\ NP^A$.

3. The computational complexity of mathematics

It is well known that the sets of provable theorems of sufficiently rich, axiomatized mathematical systems form complete sets for the recursively enumerable sets under recursive reductions. Thus, intuitively, we can say that the provable theorems in Peano Arithmetic form a set which is computationally as hard as any recursively enumerable set. Unfortunately, this interpretation does not yield any real insight about the computational complexity of doing mathematics.

We believe that the proper formulation for the study of the computational complexity of mathematics and therefore the study of the computational complexity of our intellectual tools in general, is by investigating the difficulty of proving theorems by bounding the length of the desired proof. If we do this then, as will

be shown below, the questions about the computational complexity of the process of doing mathematics—finding proofs and presenting proofs—become questions about P, NP and PSPACE.

Assume that we have an axiomatized formal system F, which could be Peano Arithmetic, and that we have given a 'natural' definition for the length of proofs and related concepts.

Then it is easily seen that the set

$$L_1 = \{\text{THEOREM: "Statement of result". PROOF: } b^k\square \mid \text{There is a proof of length } k \text{ or less of the stated theorem in } F\}$$

is NP-complete.

Similarly the set

$$L_2 = \{\text{THEOREM: "Statement of result". PRESENTATION OF PROOF: } b^k\square \mid \text{There is a proof of the stated theorem in } F \text{ which can be presented on tape of length } k\}$$

is PSPACE-complete. By *presentation of proof* we mean a formal writing down of the proof so that a simple (polynomial time) proof checker can guarantee that the theorem has a proof, but we can erase any part of the proof not needed later. Thus, when the presentation is completed, the verifier knows that a proof exists, but there may not be a complete proof written down.

Clearly, PSPACE $\neq$ NP if and only if L_2 is not in NP and this happens if and only if in Peano Arithmetic there are infinitely many theorems for which the difference in the length of the shortest proof and the space needed to present a proof is not polynomially bounded.

The fundamental question is whether finding proofs of theorems in mathematics is hard because of the existence of the aggregate of all provable theorems so that no one method can prove them all easily or it is because 'individual' theorems are hard to prove. Since we cannot give precise mathematical meaning to 'computational complexity' of finding proofs for individual theorems, we preplace this question by questions about sparse or supersparse subsets of the sets L_1 and L_2. Clearly this brings us right back to the main topic of this paper and shows that questions about sparse subsets of NP − P, PSPACE − NP and PSPACE − P are actually fundamental questions about the nature of mathematics. For example, we easily obtain the following result.

Corollary 3.1. *There exists a sparse set S in* P *such that $L_2 \cap S \notin$* NP *if and only if*

EXPSPACE $\neq$ NEXPTIME.

In the study of proof techniques special attention has been given to proving a Boolean formula a tautology. Let

$$\text{TAUT} = \{F \mid F \text{ Boolean formula in DNF such that } (\forall x)(F(x) = 1)\}.$$

Clearly, TAUT is a complete set for CoNP. We now prove the following lemma.

Lemma 3.2. *The following conditions are equivalent:*

(1) *Some decision problem in* CoNP *is not in* NP *when restricted to some sparse domain S in* P.

(2) *For some sparse set* S_1 *in* P *we cannot design special proof rules which in polynomial length will prove for any tautology in* S_1 *that it is indeed a tautology.*

Proof. It is clear that (1), (2) are equivalent to (1'), (2') respectively:

(1') There exist $A \in$ CoNP and a sparse set $S \in$ P such that $A \cap S \notin$ NP.

(2') There exists a sparse set $S_1 \in$ P such that TAUT $\cap S_1 \notin$ NP.

Clearly (2') implies (1'). Now suppose (1') holds. Let $S_2 = \bar{A} \cap S$. S_2 is sparse. Since $\bar{A} \in$ NP and $S \in$ P, $S_2 \in$ NP. Now we claim that $\bar{S}_2 \notin$ NP. This is shown as follows: Suppose $\bar{S}_2 \in$ NP. Then also $\bar{S}_2 \cap S \in$ NP. But

$$\bar{S}_2 \cap S = (A \cup \bar{S}) \cap S = A \cap S.$$

So $A \cap S \in$ NP—a contradiction. So S_2 is a sparse set in NP such that $\bar{S}_2 \notin$ NP. By Corollary 1.5 this implies that NEXPTIME $\neq$ CoNEXPTIME. Hence clearly there exists a tally set T ($T \subseteq 1^*$) in CoNP − NP. By [4] there exists a 1-1 length increasing polynomial time computable function g reducing T to TAUT.

Let

$$S_1 = g(1^*).$$

Then S_1 is a sparse set in P which is, in fact, P-printable. Also g reduces T to TAUT $\cap S_1$. Hence TAUT $\cap S_1 \notin$ NP. □

The following theorem summarizes the connection between conditions (1) and (2) of Lemma 3.2 and the possibility of closure of NEXPTIME under complement.

Theorem 3.3. *The following conditions are equivalent:*

(1) CoNEXPTIME $\neq$ NEXPTIME.

(2) *For some set L in* CoNP *and some sparse set S in* P, $L \cap S \in$ CoNP − NP.

(3) *For some* P*-printable set S,* TAUT $\cap S \in$ CoNP − NP.

Proof. Clearly (3) implies (2). By the proof of Lemma 3.2, (2) implies (1) and also (1) implies (3). □

The above theorem can be generalized to any well-behaved computation times (rather than NEXPTIME and NP) in a fashion similar to Theorem 1.7. We omit the details.

4. PSPACE sets with polynomial size circuits

We prove the following separation result.

Theorem 4.1. *Let C be a class of languages such that $C \subset$ PSIZE and $P \subset C \subset$ PSPACE. Then, $C \neq P$ implies* EXPSPACE $\neq$ EXPTIME.

Before we prove this theorem, we need a lemma. In the proof of the lemma we use a result due to Meyer (see [4]), that

$$\text{PSIZE} = \bigcup_{S \text{ sparse}} P^S.$$

Lemma 4.2. EXPSPACE $\neq$ EXPTIME *if and only if* PSPACE $\cap$ PSIZE $\neq$ P.

Proof. If EXPSPACE $\neq$ EXPTIME, then there exists a tally set $T \in$ PSPACE $-$ P. Clearly $T \in$ PSIZE. Conversely, let $L \in$ PSPACE $-$ P be in PSIZE. Then there exists an integer w and a family of circuits $\{C'_n\}$ accepting L such that, for all n, $|C'_n| \leq n^w + w$. Now, for each n, let C_n be the (lexicographically) minimal circuit accepting $L \cap \Sigma^n$. Clearly, $|C_n| \leq n^w + w$. Also, given 1^n we can in space polynomial in n construct C_n. This is done by trying all possible circuits with n inputs and length at most $n^w + w$, in lexicographic order. For each such circuit we check for all strings x in Σ^n whether x is in L if and only if x is accepted by the circuit. This can be done in polynomial space since $L \in$ PSPACE. The first such circuit which accepts exactly $L \cap \Sigma^n$ is the required C_n. By Meyer's technique of converting polynomial size circuits into a sparse oracle we then obtain a sparse set S in PSPACE such that $L \in P^S$. Hence $S \in$ PSPACE $-$ P. By [9], EXPSPACE $\neq$ EXPTIME follows. □

Theorem 4.1 follows from Lemma 4.2.

As an application of Theorem 4.1, we could have $C = R$ or $C =$ BPP. From [7] we know that $P \subseteq R \subseteq$ BPP $\subseteq$ PSPACE. In [1] it is proved that $R \subset$ PSIZE. This is strengthened by Bennet and Gill [3] to BPP $\subset$ PSIZE. Hence indeed if C is either R or BPP, C satisfies the assumptions of Theorem 4.1.

References

[1] L. Adleman, Two theorems on random polynomial time, in: *Proc. 19th IEEE Foundations of Computer Science Symp.* (1978) pp. 75-83.

[2] P. Berman, Relationship between density and deterministic complexity of NP-complete languages, *5th ICALP*, Lecture Notes in Computer Science **62** (Springer, Berlin, 1978) pp. 63-71.

[3] C. Bennet and J. Gill, Relative to a random oracle $P^A \neq NP^A \neq CoNP^A$ with probability 1, *SIAM J. Comput.* **10** (1981) 96-113.

[4] L. Berman and J. Hartmanis, On isomorphism and density of NP and other complete sets, *SIAM J. Comput.* **6** (1977) 305-322.

[5] R.V. Book, Tally languages and complexity classes, *Information and Control* **26** (1974) 186-193.

[6] R. Book, C. Wilson and M. Xu, Relativizing time and space, *Proc. 21st IEEE Foundations of Computer Science Symp.* (1981) pp. 254-259.

[7] J. Gill, Computational complexity of probabilistic Turing machines, *SIAM J. Comput.* **6** (1977) 675-695.

[8] J. Hartmanis, On sparse sets in NP−P, *Inform. Process. Lett.* **16** (2) (1983) 55-60.
[9] J. Hartmanis, N. Immerman and V. Sewelson, Sparse sets in NP−P: EXPTIME vs. NEXPTIME, *Proc. 15th ACM Symp. Theor. Comput.* (1983) pp. 382-391.
[10] R.M. Karp and R.J. Lipton, Some connections between nonuniform and uniform complexity classes, *Proc. 12th ACM Symp. Theor. Comput.* (1980) pp. 302-309.
[11] S. Mahaney, Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis, *Proc. 21st IEEE Foundations of Computer Science Symp.* (1980) pp. 42-49.
[12] V. Sewelson, Private communication.