Sparse Sets and Collapse of Complexity Classes[1]

Vladimír Glasnák

APP Czech, B.B. Centrum, Vyskočilova 1A/1422, 140 00 Prague 4, Czech Republic
E-mail: vglasnak@appg.com

Received August 4, 1998

In this paper a new upward separation technique is developed. It is applied to prove that for a class of functions $\mathcal{F}$ a separation $\mathbf{DTIME}(\mathcal{F}) \neq \mathbf{NTIME}(\mathcal{F})$ can be characterized by the existence of (not only polynomially) sparse sets in certain complexity classes. As a consequence, a solution of an open question of J. Hartmanis (1983, *Inform. Process. Lett.* **16,** 55–60) is obtained: There is an $n^{O(\log n)}$–sparse set in $\mathbf{NP} - \mathbf{P}$ iff

$$\bigcup_{c>1} \mathbf{NTIME}\,(2^{c\sqrt{n}}) \neq \bigcup_{c>1} \mathbf{DTIME}\,(2^{c\sqrt{n}}).$$

Further we prove that there is an $n^{O(\log n)}$–sparse set in $\mathbf{NP} - co\mathbf{NP}$ iff

$$\bigcup_{c>1} \mathbf{NTIME}\,(2^{c\sqrt{n}}) \neq \bigcup_{c>1} co\mathbf{NTIME}\,(2^{c\sqrt{n}}).$$

The technique also allows us to characterize the existence of sets of different densities in $\mathbf{NP} - \mathbf{P}$ by the existence of slightly denser sets in $\mathbf{NTIME}(\mathcal{F}) - \mathbf{DTIME}(\mathcal{F})$ for certain classes of functions $\mathcal{F}$. For example, there is an $n^{O(\log n)}$–sparse set in $\mathbf{NP} - \mathbf{P}$ iff there is an $n^{O((\log n)^3)}$–sparse set in

$$\bigcup_{c>1} \mathbf{NTIME}\,(n^{c\log n}) - \bigcup_{c>1} \mathbf{DTIME}\,(n^{c\log n}).$$

The end of the paper is devoted to limitations of the technique.

1. INTRODUCTION

The most important questions in structural complexity are questions of a collapse of classes ($\mathbf{NP} \stackrel{?}{=} \mathbf{P}$, $\mathbf{NEXT} \stackrel{?}{=} \mathbf{DEXT}$, . . .). It seems very difficult to prove a collapse or a separation of a deterministic and a nondeterministic class with the same time bound. Therefore many distinct characteristics of collapses are investigated in the literature. The aim of this paper is to characterize a collapse in terms of the existence of special sets of a distinct density in certain complexity classes. The first result of this type was proved by Book [5]: $\mathbf{NEXT} \neq \mathbf{DEXT}$ iff $\mathbf{NP} - \mathbf{P}$ contains a tally set. This result was strengthened by the upward separation technique due to Hartmanis [12]. He proved that a separation of the exponential classes is equivalent to the existence of a polynomially sparse set in $\mathbf{NP} - \mathbf{P}$. This result not only characterizes the collapse $\mathbf{NEXT} = \mathbf{DEXT}$ but also shows a structural property of polynomially sparse sets in **NP**. It implies that the existence of a sparse set in $\mathbf{NP} - \mathbf{P}$ is equivalent to the existence of a tally set in $\mathbf{NP} - \mathbf{P}$.

There are many results derived by the original upward separation technique, for example Hartmanis and Yesha [15] proved that $\mathbf{NEXT} = co\mathbf{NEXT}$ iff every sparse set in **NP** has its complement also in **NP**. Hartmanis *et al.* [13] proved that there are sparse sets in $\mathbf{PSPACE} - \mathbf{NP}$ iff $\mathbf{ESPACE} \neq \mathbf{NEXT}$.

All these results show that the upward separation technique is a powerful and very useful tool for studying structural properties of complexity classes. Note that all described results are concerned with polynomial and exponential classes (and polynomially sparse sets). The papers [12, 13] also studied the existence of a set of nonpolynomial density in $\mathbf{NP} - \mathbf{P}$. It is natural that the existence of a set in $\mathbf{NP} - \mathbf{P}$ with density greater than polynomial should imply a separation of classes of a complexity less than exponential and, on the other side, the existence of a set in $\mathbf{NP} - \mathbf{P}$ with density less than polynomial should imply a separation of classes with complexity greater than exponential. The technique of [12] is

[1]The author gratefully acknowledge the support of the Grant Agency of the Czech Republic under Grant 201/98/1451.

0890-5401/01 $35.00

not strong enough to prove such results. Although a result on log–sparse sets is stated in [12], its proof is incorrect (see [1]).

Sets with small density were studied by Allender [1]. He constructed an oracle A such that $\mathbf{NP}^A - \mathbf{P}^A$ contains a set of very low density (for example, the set contains at most $\log n$ words of length at most n) but

$$\mathbf{NTIME}^A\left(2^{O(2^n)}\right) = \mathbf{DTIME}^A\left(2^{O(2^n)}\right).$$

Since the technique of [12] is relativisable, it cannot prove a result of type "if $\mathbf{NP} - \mathbf{P}$ contains a set of very low density, then there are classes greater than exponential ones which do not collapse."

Allender proved such a result under an additional assumption of low generalized Kolmogorov complexity of the very sparse set (the meaning of $K[k \log n + k, n^k + k]$ is explained later):

THEOREM 1.1 [1]. *For every l, $\mathbf{NTIME}(2^{O(2^{n/l})}) = \mathbf{DTIME}(2^{O(2^{n/l})})$ if and only if for every k, every subset of $K[k \log n + k, n^k + k]$ of polylogarithmic density in* **NP** *is in* **P**.

A result on sets of greater than polynomial density was proved by Hartmanis *et al.* [13]. They proved that $\mathbf{NP} - \mathbf{P}$ contains an $n^{O(\log n)}$–sparse uniformly distributed set iff $\mathbf{NTIME}(2^{O(\sqrt{n})}) \neq \mathbf{DTIME}(2^{O(\sqrt{n})})$. Analogously as for very sparse sets, it might seem that the assumption of the uniform distribution or another similar assumption is necessary (at least for a relativisation).

The main result of this paper shows that this assumption can be omitted. For a set of functions $\mathcal{F}$ and a function g between $n+1$ and 2^n fulfilling some suitable conditions we prove:

If $\mathbf{NTIME}(\mathcal{F}) - \mathbf{DTIME}(\mathcal{F})$ contains a g–sparse set, then $\mathbf{NTIME}(\mathcal{G}) \neq \mathbf{DTIME}(\mathcal{G})$, where $\mathcal{G} = \{\lambda n.\, f(g^{-1}(2^n)) : \ f \in \mathcal{F}\}$.

One of its corollaries solves the open question of Hartmanis [12].

THEOREM 1.2. *There is a $2^{O(\lceil \log n \rceil^2)}$–sparse set in* $\mathbf{NP} - \mathbf{P}$ *iff*

$$\bigcup_{c>1} \mathbf{NTIME}(2^{c\lceil \sqrt{n} \rceil}) \neq \bigcup_{c>1} \mathbf{DTIME}(2^{c\lceil \sqrt{n} \rceil}).$$

The main result uses a new method of the upward separation appropriate for sets of density between n and 2^n. The method is close to the techniques of Buhrman *et al.* [7], Saluja [17], and Schöning [18] for proofs that every sparse set is conjunctively truth–table reducible to a tally set [7, 17] and that every sparse set is randomly m–reducible to a tally set [18].

All these techniques work with polynomially sparse sets; therefore it is possible to search a universe which is polynomially related to density of a sparse set, in polynomial time (we simply try every member of the universe). Such a search cannot be done for a larger universe (and thus for slightly denser sets) in polynomial time. Our method exploits hashing functions with small descriptions to make such a search in polynomial time.

The following two theorems also follows from the technique.

THEOREM 1.3. *Let g be a nondecreasing function satisfying the following conditions.*

(1) $n + 1 \leq g(n) \leq 2^{2^{\lceil \sqrt{\log n} \rceil - 1}}$ *a.e.*;

(2) $h_0(n) = 2^{\lceil \sqrt{\log g^{-1}(2^n)} \rceil}$ *and* $h_1(n) = g^{-1}(2^n)$ *are time constructible* (*i.e., there is a Turing machine working in time $O(h_0(n))$ producing $1^{h_0(n)}$ on every input 1^n and similarly for h_1*);

(3) *there exists some b such that $g^{-1}(n^2) \leq (g^{-1}(n))^b$ a.e.*

Then $\mathbf{NP} - \mathbf{P}$ *contains a g–sparse set, iff*

$$\mathbf{NTIME}\left(2^{O(\lceil \log n \rceil^2)}\right) - \mathbf{DTIME}\left(2^{O(\lceil \log n \rceil^2)}\right)$$

contains a $g(2^{\lceil \log n \rceil^2})$–sparse set.

Theorem 1.3 relates the existence of a g–sparse set in $\mathbf{NP} - \mathbf{P}$ with the existence of a slightly denser set between classes above **NP** and **P**. For example, $\mathbf{NP} - \mathbf{P}$ contains a $2^{O(\lceil \log n \rceil^2)}$–sparse set iff $\mathbf{NTIME}(2^{O(\lceil \log n \rceil^2)}) - \mathbf{DTIME}(2^{O(\lceil \log n \rceil^2)})$ contains a $2^{O(\lceil \log n \rceil^4)}$–sparse set.

THEOREM 1.4. *Let $c > 0$ be a number. Then every $2^{O(\lceil \log n \rceil^c)}$–sparse set in* **NP** *has its complement also in* **NP** *iff*

$$\mathbf{NTIME}\big(2^{O(\lceil \sqrt[c]{n} \rceil)}\big) = co\mathbf{NTIME}\big(2^{O(\lceil \sqrt[c]{n} \rceil)}\big).$$

Theorem 1.4 is a generalization of a result of Hartmanis and Yesha [15] characterizing the separation $co\mathbf{NEXT} \neq \mathbf{NEXT}$.

At the end of the paper we discuss limitations of the technique. We show a limitation that does not use relativisation.

This paper is a full version of the paper [11].

2. PRELIMINARIES

Let f be a function on natural numbers. We use the following notation:

$$O(f) = \{g : \text{there exists a constant } c \text{ such that } g(n) \leq cf(n) + c \text{ for every } n\},$$
$$o(f) = \{g : \text{for every } c,\ cg(n) \leq f(n) \text{ holds for almost every } n\},$$
$$\Omega(f) = \{g : \text{there exists a constant } c \text{ such that } cg(n) + c \geq f(n) \text{ for every } n\}.$$

Our computation model is a multi-tape Turing machine with the alphabet $\{0, 1\}$. We say that a function f is time constructible if there is a Turing machine M working in time $O(f)$ such that M on every input 1^n produces the output $1^{f(n)}$. The class of sets recognized deterministically (or nondeterministically) in time $O(f)$ is denoted by $\mathbf{DTIME}(f)$ (or $\mathbf{NTIME}(f)$). Let us define (see [8]):

$$\mathbf{P} = \bigcup_i \mathbf{DTIME}(n^i), \qquad \mathbf{NP} = \bigcup_i \mathbf{NTIME}(n^i),$$
$$\mathbf{DEXT} = \bigcup_i \mathbf{DTIME}(2^{in}), \qquad \mathbf{NEXT} = \bigcup_i \mathbf{NTIME}(2^{in}),$$
$$\mathbf{EXPTIME} = \bigcup_i \mathbf{DTIME}\big(2^{n^i}\big), \qquad \mathbf{NEXPTIME} = \bigcup_i \mathbf{NTIME}\big(2^{n^i}\big).$$

The length of a word x is denoted by $|x|$. The binary expansion of a natural number n is denoted by $\mathrm{bin}(n)$. For a word $x \in 1\{0, 1\}^* \cup \{0\}$, let $\mathrm{nn}(x)$ denote the natural number with $\mathrm{bin}(\mathrm{nn}(x)) = x$.

Let s be a real number. The natural number n with $n \leq s < n + 1$ is denoted by $\lfloor s \rfloor$ and the natural number n with $n - 1 < s \leq n$ is denoted by $\lceil s \rceil$.

LEMMA 2.1. *Length of the binary expansion of a natural number $n > 0$ is $\lfloor \log n \rfloor + 1$.*

The proof of Lemma 2.1 is obvious.

For a set $A \subseteq \{0, 1\}^*$ denote $\overline{A} = \{0, 1\}^* - A$, $A_{\leq n} = \{x \in A : |x| \leq n\}$ and $A_{=n} = \{x \in A : |x| = n\}$.

Let $h : \{0, 1, \#\}^* \to \{0, 1\}^*$ be a homomorphism such that $h(0) = 00$, $h(1) = 11$, and $h(\#) = 10$. Then $\langle x_1, x_2, \ldots x_n \rangle$ is defined by

$$\langle x_1, x_2, \ldots x_n \rangle = \begin{cases} x_1, & \text{if } n = 1; \\ h(x_1 \# x_2 \# \ldots \# x_n), & \text{if } n > 1. \end{cases}$$

For $n > 1$ we have $|\langle x_1, x_2, \ldots x_n \rangle| = 2|x_1| + 2|x_2| + \cdots + 2|x_n| + 2n - 2$.

Let $r : \{0, 1\}^* \to \mathrm{N}$ be a partial function. Define

$$[x_1, x_2, \ldots x_n]_r = \langle x_1, x_2, \ldots x_n \rangle 01^{r(x_1) - |\langle x_1, x_2, \ldots x_n \rangle| - 1},$$

if $r(x_1)$ is defined and $r(x_1) - |\langle x_1, x_2, \ldots x_n \rangle| \geq 1$;

$$[x_1, x_2, \ldots x_n]_r = \langle x_1, x_2, \ldots x_n \rangle 0, \text{ otherwise.}$$

Thus if r grows quickly and x_1 compared with $\langle x_1, x_2, \ldots x_n\rangle$ has sufficient length, then $|[x_1, x_2, \ldots x_n]_r| = r(x_1)$.

For a function $f : \mathrm{N} \to \mathrm{N}$ and a word x define $[x]_f = [x]_r$ where $r(x) = f(|x|)$. For a set A define $A_f = \{[x]_f : x \in A\}$. We say that A_f is the set A coded by f. The complement of A_f is denoted by $\overline{A_f}$ and the complement of A coded by f is denoted by $(A)_f$.

We work with nondecreasing functions on natural numbers. For a nondecreasing unbounded function f define $f^{-1}(n) = \min\{t : f(t) \geq n\}$. The notion "$^{-1}$" is used here in this sense only. The standard inverse function of f is denoted f^{inv} in this paper. Observe that f^{-1} coincides with f^{inv} for any increasing function f on all n where $f^{\mathrm{inv}}(n)$ is defined.

LEMMA 2.2. *Let $f : \mathrm{N} \to \mathrm{N}$ be a nondecreasing unbounded function. Then f^{-1} is a nondecreasing unbounded function and*

$$f^{-1}(f(n)) \leq n < f^{-1}(f(n)+1), \quad n \leq f(f^{-1}(n)), \quad \text{for all } n.$$

Proof. All inequalities follow just from the definition of f^{-1}. ■

When we speak about functions on natural numbers, we always denote the argument of the function by n. For example, $f^{-1}(2^n + m)$ denotes the function $\lambda n.\, f^{-1}(2^n + m)$.

In one section hierarchy theorems are applied. We refer to the following hierarchy theorems proved by Hartmanis and Stearns [14] and by Seiferas *et al.* [19].

THEOREM 2.1 [14]. *If t_2 is a time constructible function, then the following set difference contains a set over $\{0, 1\}$:*

$$\mathbf{DTIME}(t_2) - \bigcup\{\mathbf{DTIME}(t_1) : t_2 \notin O(t_1 \log t_1)\}.$$

THEOREM 2.2 [19]. *If t_2 is a time constructible function, then the following set difference contains a set over $\{0, 1\}$:*

$$\mathbf{NTIME}(t_2) - \bigcup\{\mathbf{NTIME}(t_1) : t_1(n+1) \in o(t_2(n))\}.$$

We also apply the following theorem which was independently proved by Fredman *et al.* [10] and Mehlhorn [16].

THEOREM 2.3 [10, 16]. *There is a constant c such that for every number N and every set $S \subseteq \{0, 1, \ldots N-1\}$ of cardinality $|S| = m$, there exists a prime $p \leq cm^2 \log N + c$ such that the hashing function $h(x) = x \bmod p$ is perfect for S (i.e., $h(x) \neq h(y)$ for every two different $x, y \in S$).*

Theorem 2.3 has many applications not only in this paper. Its variants were used to prove that $\mathbf{BPP} \subseteq \Sigma_2$ (by Buhrman and Fortnow [6]) and in a derandomization process (by Allender *et al.* [3]).

Allender and Rubinstein [2] proved that sets polynomially isomorphic to tally sets have a small generalized Kolmogorov complexity. Exactly, let U be a fixed universal Turing machine that simulates deterministic Turing machines with logarithmic slowness. For functions f, g, define

$$K[f, g] = \{x : (\exists y)(|y| \leq f(|x|) \text{ and } U \text{ on } y \text{ outputs } x \text{ within } g(|x|) \text{ steps})\}.$$

Further define

$$\mathit{SETS}\text{–}K[\log, \mathrm{poly}] = \{A : (\exists k)(A \subseteq K[k \log n + k, n^k + k])\}.$$

Two sets A, B are polynomially isomorphic if there exists a bijection $h : \{0, 1\}^* \to \{0, 1\}^*$ such that h and h^{inv} are computable in polynomial time and $h(A) = B$.

THEOREM 2.4 [2]. *The class $\mathit{SETS}\text{–}K[\log, \mathrm{poly}]$ is equal to the class of the sets polynomially isomorphic to a tally set.*

3. THE BASE

DEFINITION 3.1. Let $\mathcal{F} = \{f_i\}$ be a sequence of nondecreasing time constructible functions such that $n \leq f_i(n) \leq f_{i+1}(n)$ for all natural numbers i, n. The classes $\mathbf{D}(\mathcal{F})$, $\mathbf{N}(\mathcal{F})$ are defined by:

$$\mathbf{D}(\mathcal{F}) = \bigcup_i \mathbf{DTIME}(f_i), \quad \mathbf{N}(\mathcal{F}) = \bigcup_i \mathbf{NTIME}(f_i).$$

We say that $\mathcal{F}$ is a base of the classes $\mathbf{D}(\mathcal{F})$, $\mathbf{N}(\mathcal{F})$.

Let $\mathcal{F} = \{f_i\}$, $\mathcal{G} = \{g_i\}$ be bases. We say that $\mathcal{F}$ is

- closed under composition if for any natural numbers i, j, there is a natural number k such that $f_i \circ f_j \in O(f_k)$;
- closed under right composition with $\mathcal{G}$ if for any natural numbers i, j, there is a natural number k such that $f_i \circ g_j \in O(f_k)$;
- closed under product if for any natural numbers i, j, there is a natural number k such that $f_i \cdot f_j \in O(f_k)$;
- closed under product with $\mathcal{G}$ if for any natural numbers i, j, there is a natural number k such that $f_i \cdot g_j \in O(f_k)$.

A base $\mathcal{F}$ collapses if $\mathbf{N}(\mathcal{F}) = \mathbf{D}(\mathcal{F})$.

Let $\mathcal{F} = \{f_i\}$ and $\mathcal{G} = \{g_i\}$ be bases. The composition of the bases $\mathcal{F}$ and $\mathcal{G}$ is the base $\mathcal{F} \circ \mathcal{G} = \{f_i \circ g_i\}$.

If f is a nondecreasing time constructible function with $f(n) \geq n$, then it defines a base $\mathcal{F}_f = \{f\}$. We use f instead of $\mathcal{F}_f$ if there is no possible confusion. For example, $\mathcal{G} \circ f$ denotes $\mathcal{G} \circ \mathcal{F}_f$.

To avoid confusion when using functions instead of bases, we mostly use $f(g(n))$ for the composition of functions f and g, while $f \circ g$ in most cases denotes a base.

Let $\mathcal{F} = \{f_i\}$ be a sequence of functions (not necessarily a base). We define the following classes:

$$O(\mathcal{F}) = \{\mathcal{G} : \text{ for every } g \in \mathcal{G} \text{ there exists a function } f \in \mathcal{F} \text{ such that } g \in O(f)\},$$

$$\Omega(\mathcal{F}) = \{\mathcal{G} : \mathcal{F} \in O(\mathcal{G})\},$$

$$\Theta(\mathcal{F}) = \Omega(\mathcal{F}) \cap O(\mathcal{F}).$$

We say that a Turing machine works in $\mathbf{D}(\mathcal{F})$–time if it works deterministically in time $g \in O(\mathcal{F})$. Analogously, we use the notion of $\mathbf{N}(\mathcal{F})$–time, $\mathbf{D}(\mathcal{F})$–algorithm, etc.

Remark. Classes of sets with bases $\mathcal{F}, \mathcal{G}$ such that $\mathcal{F} \in \Theta(\mathcal{G})$ coincide. Note that the definition of bases composition is correct because $n \leq f_i(g_i(n)) \leq f_{i+1}(g_{i+1}(n))$. If $\mathcal{F}$ is closed under composition then it is clear that $\mathcal{G} \circ \mathcal{F}$ is closed under right composition with $\mathcal{F}$. If a base $\mathcal{F}$ is closed under product, then it is also closed under product with polynomials. Note that "big O, Ω, and Θ" allows us to speak about some properties of bases alternatively. For example $\mathcal{F}$ is closed under right composition with $\mathcal{G}$ iff $\mathcal{F} \circ \mathcal{G} \in O(\mathcal{F})$.

Finally, we give a few examples of standard bases of some complexity classes.

DEFINITION 3.2. Let $k > 0$. Define

$$\mathcal{LIN} = \{in\}_{i>0}, \qquad \mathcal{P} = \{n^i\}_{i>0},$$

$$\mathcal{EXT} = \{2^{in}\}_{i>0}, \qquad \mathcal{EXP} = \left\{2^{n^i}\right\}_{i>0},$$

$$\mathcal{EL}_k = \left\{2^{i\lceil \log n \rceil^k}\right\}_{i>0}, \quad \mathcal{EPL} = \left\{2^{\lceil \log n \rceil^i}\right\}_{i>0},$$

$$\mathcal{ER}_k = \left\{2^{i\lceil \sqrt[k]{n} \rceil}\right\}_{i>0}.$$

The classes with the bases $\mathcal{P}$ or $\mathcal{EXT}$ are **NP**, **P,** or **NEXT**, **DEXT**.

It is easy to show that the base $\mathcal{EPL}$ is closed under product and under composition, the base $\mathcal{EL}_k$ is closed under product and $\mathcal{P} \in \Theta(\mathcal{EL}_1)$. Similarly, $\mathcal{EXT} \in \Theta(\mathcal{ER}_1)$.

PROPOSITION. *The following statements hold for any* $k > 0$.

(1) $\mathbf{D}(\mathcal{EL}_k) \subsetneq \mathbf{D}(\mathcal{EL}_{k+1})$ *and* $\mathbf{N}(\mathcal{EL}_k) \subsetneq \mathbf{N}(\mathcal{EL}_{k+1})$;

(2) $\mathbf{D}(\mathcal{EPL}) = \cup_i \mathbf{D}(\mathcal{EL}_i)$ *and* $\mathbf{N}(\mathcal{EPL}) = \cup_i \mathbf{N}(\mathcal{EL}_i)$;

(3) $\mathbf{D}(\mathcal{EPL}) \subsetneq \mathbf{D}(\mathcal{ER}_k)$ *and* $\mathbf{N}(\mathcal{EPL}) \subsetneq \mathbf{N}(\mathcal{ER}_k)$;

(4) $\mathbf{D}(\mathcal{ER}_{k+1}) \subsetneq \mathbf{D}(\mathcal{ER}_k)$ *and* $\mathbf{N}(\mathcal{ER}_{k+1}) \subsetneq \mathbf{N}(\mathcal{ER}_k)$.

Proof. (1), (3), (4) follow from Theorems 2.1 and 2.2. (2) is obvious. ■

4. SPARSE SETS AND THE PADDING TECHNIQUE

The result of [12] relates the existence of a polynomially sparse set in $\mathbf{NP} - \mathbf{P}$ to the existence of a tally set in $\mathbf{NP} - \mathbf{P}$. We extend the result for sets of densities between n and 2^n. Instead of tally sets we consider their analogy for greater densities—padded sets.

DEFINITION 4.1. For a set S define its census function by $c_S(n) = |S_{\le n}|$. Let $f : \mathrm{N} \to \mathrm{N}$ be a function. We say that a set S is f–sparse if $c_S \in O(f)$.

Let $\mathcal{F}$ be a base. A set S is $\mathcal{F}$–sparse if there exists a function $f \in \mathcal{F}$ such that S is f–sparse.

Let f be a nondecreasing time constructible function with $f(n) > n$ almost everywhere. A set A is called f–padded if $A = B_f$ for some set B.

Let F be a class of functions. A set T is F–padded if there exists a nondecreasing time constructible function $f \in F$ with $f(n) > n$ a.e. such that T is f–padded.

Remark. The mentioned class F will be mostly $\Omega(f)$ for a function f.

Every padded set contains "a long tail" word; hence it is not surprising that padded sets may have small density. This is shown in the following lemma.

LEMMA 4.1. *Let f be a nondecreasing time constructible function with $f(n) > n$ a.e. Then the following statements hold*:

(1) *every f–padded set is $2^{f^{-1}(n+1)}$–sparse*;

(2) *if f is increasing, then every f–padded set is $2^{f^{-1}(n)}$–sparse*;

(3) *if $f \in \Omega(g)$ for some nondecreasing function g, then every f–padded set is $2^{g^{-1}(cn)}$–sparse, where c is a constant depending on f.*

Proof. Every f–padded set is a subset of $(\{0,1\}^*)_f$; therefore it suffices to prove that $(\{0,1\}^*)_f$ is sufficiently sparse. Since $f(n) > n$ almost everywhere, there are only finitely many x with $|[x]_f| \neq f(|x|)$. Let $d = |\{x : |[x]_f| \neq f(|x|)\}|$. For every n, we have

$$|((\{0,1\}^*)_f)_{\le n}| \le |\{x \in \{0,1\}^* : f(|x|) \le n\}| + d$$
$$= d + \sum_{i \in \{k:\, f(k) \le n\}} 2^i = d + 2^0 + 2^1 + \cdots + 2^{\max\{k:\, f(k) \le n\}} \le d + 2^{\max\{k:\, f(k) \le n\}+1} - 1.$$

Now we continue separately for (1), (2), and (3).

(1) If f is nondecreasing, then, by Lemma 2.4, $k < f^{-1}(f(k)+1)$; hence

$$2^{\max\{k:\, f(k) \le n\}+1} \le 2^{\max\{k:\, k < f^{-1}(n+1)\}+1} = 2^{f^{-1}(n+1)}.$$

Therefore $(\{0,1\}^*)_f$ is $2^{f^{-1}(n+1)}$–sparse.

(2) If f is increasing, then $f(k) \le n$ implies $k \le f^{-1}(n)$; hence

$$2^{\max\{k:\, f(k) \le n\}+1} \le 2^{\max\{k:\, k \le f^{-1}(n)\}+1} = 2 \cdot 2^{f^{-1}(n)}.$$

Therefore $(\{0,1\}^*)_f$ is $2^{f^{-1}(n)}$–sparse.

(3) Since $f \in \Omega(g)$, there exists a constant c such that $cf(n) \geq g(n)+1$ almost everywhere. By Lemma 2.1 for almost every n,

$$2^{\max\{k:\, f(k)\leq n\}+1} = 2^{\max\{k:\, cf(k)\leq cn\}+1} \leq 2^{\max\{k:\, g(k)+1\leq cn\}+1} = 2^{\max\{k:\, k<g^{-1}(cn)\}+1} = 2^{g^{-1}(cn)}.$$

Therefore $(\{0,1\}^*)_f$ is $2^{g^{-1}(cn)}$–sparse. ■

Now we show that padded sets are a really good analogy for tally sets.

THEOREM 4.1. *Every $\Omega(2^n)$–padded set is polynomially isomorphic to a tally set and every tally set is polynomially isomorphic to a 2^n–padded set.*

Proof. We will prove that any $\Omega(2^n)$–padded set belongs to *SETS–K*[log, poly]. Then, by Theorem 2.4, every $\Omega(2^n)$–padded set is polynomially isomorphic to a tally set.

Let $f \in \Omega(2^n)$ be a nondecreasing time constructible function with $f(n) > n$ a.e. There exists a Turing machine M working in time $O(f)$ that for every input y outputs $y01^{f(|y|)-|y|-1}$.

Now, for every $x = y01^{f(|y|)-|y|-1}$ for some y of sufficient length we have:

(1) $c|x| \geq 2^{|y|}$, where c is a constant depending only on f. Thus

$$\log c + \log|x| \geq |y|.$$

Since x can be described by y and a code of M, there exists a sufficiently small description of x.

(2) An effective simulation of M on y produces x in time polynomial in $|x|$.

Since membership in *SETS–K*[log, poly] is not affected by finitely many words $x = y0$ with $f(|y|) \leq |y|$, every $\Omega(2^n)$–padded set belongs to *SETS–K*[log, poly], and the first statement follows from Theorem 2.4.

Now we prove that every tally set is polynomially isomorphic to a 2^n–padded set. The technique of the proof is the same as in [2].

Let $NUM = \{\text{bin}(n) : n \in \mathrm{N}\}$ and $f(n) = 2^n$. Define a function $u : \{1\}^* \to NUM_f$ by

$$u(1^n) = \text{bin}(n)01^{2^{|\text{bin}(n)|}-|\text{bin}(n)|-1}.$$

Note that u is bijective and computable in polynomial time. Similarly also u^{inv} is computable in polynomial time. Besides, both $\{1\}^*$ and NUM_f are in **P**.

Further, it is possible to extend u to an isomorphism $h : \{0,1\}^* \to \{0,1\}^*$ by mapping the ith word of $\{0,1\}^* - \{1\}^*$ onto the ith word of $\{0,1\}^* - NUM_f$. See [2] or [9] for details of such a construction.

Finally, every tally set is a subset of $\{1\}^*$; thus h maps it onto a subset of NUM_f which is 2^n–padded. ■

The oldest method relating the existence of a certain set (a padded set) with the collapse of complexity classes is called downward separation. It is based on a padding argument.

Recall that $\overline{A_g}$ denotes the complement of the set A_g and the complement of A coded by g is denoted by $(\bar{A})_g$.

THEOREM 4.2 (downward separation). *Let $\mathcal{F} = \{f_i\}$ be a base and let g be a nondecreasing time constructible function such that $g(n) > n$ for almost every n. Then, for every set A, the following equivalences hold*:

(1) $A_g \in \mathbf{N}(\mathcal{F}) - \mathbf{D}(\mathcal{F})$ *iff* $A \in \mathbf{N}(\mathcal{F} \circ g) - \mathbf{D}(\mathcal{F} \circ g)$;

(2) $\overline{A_g} \in \mathbf{N}(\mathcal{F})$ *iff* $\bar{A} \in \mathbf{N}(\mathcal{F} \circ g)$.

Proof. Since g is time constructible, a verification whether any given y is equal to $[x]_g$ for some x runs in $O(|y|)$ and a computation of $[z]_g$ for any z runs in $O(g(|z|))$. Therefore if $A \in \mathbf{DTIME}(f_i \circ g)$, then a decision whether $y = [x]_g \in A_g$ can be made in time $O(|y| + f_i(g(|x|))) = O(f_i(|y|))$. If

$A_g \in \mathbf{DTIME}(f_i)$, then a decision whether $x \in A$ runs in time $O(g(|x|) + f_i(g(|x|))) = O(f_i(|[x]_g|))$. By the same reasoning we obtain the analogous statement for nondeterministic time.

Thus $A_g \in \mathbf{N}(\mathcal{F}) - \mathbf{D}(\mathcal{F})$ iff $A \in \mathbf{N}(\mathcal{F} \circ g) - \mathbf{D}(\mathcal{F} \circ g)$.

Similarly, $\overline{A} \in \mathbf{N}(\mathcal{F} \circ g)$ iff $(\overline{A})_g \in \mathbf{N}(\mathcal{F})$. Moreover, $(\overline{A})_g \in \mathbf{N}(\mathcal{F})$ iff $\overline{A_g} \in \mathbf{N}(\mathcal{F})$ because $\overline{A_g}$ contains only elements of $(A)_g$ and words which are not correctly coded by $[x]_f$ (these are recognized in linear time). ■

This result relates the existence of a certain padded set with a separation of complexity classes. By Theorem 4.2 and Lemma 4.4, a separation of base classes implies the existence of a sparse set between smaller base classes. Therefore Theorem 4.2 is called downward separation. The opposite implication—whether the existence of a sparse set between certain classes implies a separation of some higher classes—is investigated in the next section.

5. SPARSE SETS AND COLLAPSE OF CLASSES

The first upward separation technique was developed by Hartmanis [12]. He proved that there is a polynomially sparse set in $\mathbf{NP} - \mathbf{P}$ iff $\mathbf{NEXT} \neq \mathbf{DEXT}$. While the implication from the right to the left is simple (and follows, for example, from Theorem 4.2 and Lemma 4.4), the implication from the left to the right was proved by a clever method. We briefly describe it here:

Assume that $\mathbf{NEXT} = \mathbf{DEXT}$ and let S be a polynomially sparse set in $\mathbf{NP}$. We will show that $S \in \mathbf{P}$. The main idea is "to uniquely code" the elements of S into a set $C \in \mathbf{NEXT} = \mathbf{DEXT}$ such that every member of S can be decoded from C. Define

$$C = \{\langle \mathrm{bin}(n), \mathrm{bin}(i), \mathrm{bin}(j), \mathrm{bin}(k), d\rangle : \exists x_1 < x_2 < \cdots < x_i \in S_{=n} \text{ and the } k\text{th bit of } x_j \text{ is } d\}.$$

For $i_n = |S_{=n}|$ there is exactly one sequence $x_1 < x_2 < \cdots < x_{i_n}$ of elements of $S_{=n}$; thus $\langle \mathrm{bin}(n), \mathrm{bin}(i_n), \mathrm{bin}(j), \mathrm{bin}(k), d\rangle \in S$ iff the kth bit of x_j is d. Hence knowing i_n and the membership in C we can construct every element of $S_{=n}$ from its order in $S_{=n}$ (bit by bit).

There is a natural nondeterministic algorithm for C: On the input nondeterministically guess x_1, $x_2, \ldots, x_i$, verify whether they belong to $S_{=n}$ by an **NP**–algorithm for S, and deterministically verify the other conditions. This algorithm runs in polynomial time with respect to n but its input has length in $O(\log n)$; therefore it is a **NEXT**–algorithm. Since we assume $\mathbf{NEXT} = \mathbf{DEXT}$, we have $C \in \mathbf{DEXT}$.

A polynomial time algorithm for S consist of two steps. Let x of length n be the input. In the first step, we compute i_n. In the second step, we construct all members of $S_{=n}$ and verify whether x belongs to this set.

The first step: Note that i_n is the smallest i such that $\langle \mathrm{bin}(n), \mathrm{bin}(i+1), 1, 1, d\rangle \notin C$ for every $d \in \{0, 1\}$. Since i_n is polynomially bounded, to compute i_n we need at most polynomially many verifications whether a word of length in $O(\log n)$ belongs to C. Since $C \in \mathbf{DEXT}$, i_n is computed in polynomial time.

The second step: For every j, $1 \leq j \leq i_n$ we construct the jth word of $S_{=n}$, by verifying whether $\langle \mathrm{bin}(n), \mathrm{bin}(i_n), \mathrm{bin}(j), \mathrm{bin}(k), 1\rangle \in C$ for every k, $1 \leq k \leq n$. Since j is polynomially bounded and since every verified tuple has length in $O(\log n)$, this computation runs in polynomial time. At the end we verify whether x is among the constructed words.

The described algorithm is the **P**–algorithm for S. For more details of the technique see [1, 12, 13, 15].

There are two important limitations of the technique extension for sets with different than polynomial density. The first one concerns sets of low density. For example, it is stated in [12] that there are log–sparse sets in $\mathbf{NP} - \mathbf{P}$ iff $\mathbf{NTIME}(2^{O(2^n)}) \neq \mathbf{DTIME}(2^{O(2^n)})$ but the proof is incorrect (see [1]). Let **NEE** and **DEE** denote these two classes. To prove this result we should redefine the set C such that every member of a log–sparse set S can be constructed from S and $C \in \mathbf{NEE}$. When we want to construct x of length n bit by bit, then we need for every bit also its position in x. A description of a position in x has length $O(\log n)$. Therefore any member of C describing x has length $O(\log n)$. Thus if $C \in \mathbf{DEE}$, it cannot help us to construct a polynomial (but only exponential) time algorithm for S. This problem was investigated by Allender [1]. He proved that no relativisable technique can be used to prove that there are log–sparse sets in $\mathbf{NP} - \mathbf{P}$ iff $\mathbf{NEE} \neq \mathbf{DEE}$. Although this result is negative, Allender

also proved that adding an assumption of low Kolmogorov complexity of the sparse set enables to use a variant of the original upward separation technique [12] to prove upward separation results (recall Theorem 1.1).

The second limitation concerns denser sets. If we want to prove Theorem 1.2 by the technique of [12], then we should redefine C such that $C \in \mathbf{NTIME}(2^{O(\sqrt{n})})$. Since a natural algorithm for C has to nondeterministically guess all members of $S_{=n}$, we obtain $C \in \mathbf{NEXT}$. Therefore an extension of the technique concerning denser sets in [13] needs an additional assumption of uniform distribution of the sparse set.

Here we briefly mention how to avoid this problem. The key idea is to apply perfect hashing. Note that the set C is defined such that every member x of $S_{=n}$ is uniquely given by n, $i_n = |S_{=n}|$ and its position j in the lexicographic order. These three numbers allows us to save all bits of x. Two of these three numbers (i_n and j) depend on all other members of $S_{=n}$; hence to compute any of them, we should guess all members of $S_{=n}$. This is the problem because $S_{=n}$ may have a cardinality that is too big. Therefore we give another way of unique identification of $x \in S_{=n}$.

Let us define $h_b(x) = \mathrm{nn}(1x) \bmod b$ for a number b. Viewing the set $S_{=n}$ as a small cardinality subset of $\{2^n, 2^n+1, \ldots 2^{m+1}\}$ and applying Theorem 2.3, we obtain a small number b such that h_b is a perfect hashing function for $S_{=n}$. It means that every $x \in S_{=n}$ is uniquely identified by n, b, and $h_b(x)$. Now note the difference:

- To find an $x \in S_{=n}$ which is the jth element in the lexicographic order we should guess all members of $S_{=n}$ and find the jth one.
- To find an $x \in S_{=n}$ with $h_b(x) = t$ we need to guess only x and compute $h_b(x)$.

However, in the second case we need b describing the perfect hashing function. Fortunately, it is possible to compute b. In the following this idea is unfolded more exactly.

Our technique is divided into some lemmas, which allows us to prove all main results at once. The assumption under which our technique works is summarized below. We refer to it as to the assumption (X) in the following. In every lemma in this section we assume that (X) holds without mentioning it.

Let $\mathcal{F} = \{f_i\}$ be a base closed under product with $\mathcal{P}$. Let g be a nondecreasing function such that $n+1 \le g(n) \le 2^n$ a.e. and the function $g^{-1}(2^n)$ is time constructible. Let us assume that there is a function $g_0 : \mathrm{R} \to \mathrm{N}$, a nondecreasing function $g_1 : \mathrm{N} \to \mathrm{N}$, and a constant $\alpha > 21$ such that:

(a) the function $g^{-1}(g_0(2^{\frac{n}{\alpha}}))$ is time constructible;
(b) $g_0(n) \ge 2n$ for almost every $n \in \mathrm{N}$;
(c) $g^{-1}(g_0(n)) \le g_1(g^{-1}(n))$ for almost every $n \in \mathrm{N}$;
(d) the base $\mathcal{F}$ is closed under right composition with g_1.

Define $\mathcal{G} = \{f_i(g^{-1}(g_0(2^{\frac{n}{\alpha}})))\}$ and $r : 1\{0,1\}^* \to \mathrm{N}$ by $r(\mathrm{bin}(n)) = \alpha\lfloor \log(g(n))\rfloor$.

The function r defined at the end of (X) will be used to code tuples by $[\ldots]_r$.

The assumption that $\alpha > 21$ depends on the coding of tuples. At certain points of the proof it is necessary to have α sufficiently large to obtain a bound for length of some code of a tuple. Precisely, in Lemma 5.4 we need $\alpha > 11$, in Lemma 5.5 we need $\alpha > 19$, and in Lemma 5.7 we need $\alpha > 21$.

These conditions are technical and as we will see later, "common" bases and functions satisfy them.

We begin with a lemma showing a construction of $\lfloor \log g(n)\rfloor$.

LEMMA 5.1. *There exists an algorithm working in time $O(n^2)$ that on every input 1^n outputs $1^{\lfloor \log g(n)\rfloor}$ if $g(n) > 0$ and 0 if $g(n) = 0$.*

Proof. Since g is nondecreasing and $n+1 \le g(n)$ a.e., there exists n_0 such that $g(n) = 0$ for $n < n_0$ and $g(n) > 0$ for $n \ge n_0$.

By the properties of an inverse function (Lemma 2.4) we obtain

$$g^{-1}\left(2^{\lfloor \log g(n)\rfloor}\right) \le g^{-1}(g(n)) \le n < g^{-1}(g(n)+1) \le g^{-1}\left(2^{\lfloor \log g(n)\rfloor+1}\right).$$

We apply this inequality to construct an algorithm computing $1^{\lfloor \log g(n)\rfloor}$. Since $g^{-1}(2^n)$ is time constructible, there is an algorithm M and a constant c such that M on every input 1^t works in time $cg^{-1}(2^t)+c$ and outputs $1^{g^{-1}(2^t)}$.

ALGORITHM 1. 1 [computing $1^{\lfloor \log g(n)\rfloor}$]

1. Input 1^n;
2. If $n < n_0$ output 0 and stop;
3. Let $t = 1$;
4. Simulate the first $cn + c$ steps of M on the input 1^t;
5. If M did not stop or M stopped and $n < g^{-1}(2^t)$, then output 1^{t-1} and stop;
6. Else let $t = t + 1$ and go to step 4.

Let 1^n be an input with $n \ge n_0$. Note that if M does not stop in step 4, then $cn + c < cg^{-1}(2^t) + c$ because M on every input 1^t works in time at most $cg^{-1}(2^t) + c$. Therefore the algorithm stops in step 5 iff t is the smallest number with $n < g^{-1}(2^t)$. Thus the algorithm is correct.

For one fixed t the complexity of each step is bounded by $O(n)$. The complexity of the algorithm depends on the number of repetitions of the step 4. Since $g^{-1}(2^n)$ is time constructible, $g^{-1}(2^n) \in \Omega(n)$; therefore step 4 repeats at most $t = dn + d$ times for some d. Thus, the complexity of the algorithm is $O(n^2)$. ■

LEMMA 5.2. *Define* $t(n) = \alpha\lfloor \log g(n)\rfloor$. *Then*

$$n < g^{-1}\left(g_0\left(2^{\frac{t(n)}{\alpha}}\right)\right) \text{a.e.} \quad \textit{and} \quad g^{-1}\left(g_0\left(2^{\frac{t(n)}{\alpha}}\right)\right) \le g_1(n) \textit{ a.e.}$$

Proof.

$$\begin{aligned} n &< g^{-1}(g(n)+1) && \text{by Lemma 2.2,}\\ &\le g^{-1}\left(2^{\lfloor \log g(n)\rfloor+1}\right) && \text{because } g(n) < 2^{\lfloor \log g(n)\rfloor+1},\\ &\le g^{-1}\left(g_0\left(2^{\frac{t(n)}{\alpha}}\right)\right) \text{ a.e.} && \text{since } 2n \le g_0(n) \text{ a.e.} \end{aligned}$$

This proves the first inequality.

$$\begin{aligned} g^{-1}\left(g_0\left(2^{\frac{t(n)}{\alpha}}\right)\right) &\le g_1\left(g^{-1}\left(2^{\lfloor \log g(n)\rfloor}\right)\right) \text{ a.e. since } g^{-1}(g_0(n)) \le g_1(g^{-1}(n)) \text{ a.e.,}\\ &\le g_1(g^{-1}(g(n))) \text{ a.e.}\\ &\le g_1(n) \text{ a.e. by Lemma 2.2 and since } g_1 \text{ is nondecreasing.} \end{aligned}$$

This proves the second inequality. ■

Remark. Lemma 5.2 allows us to compute a complexity of an algorithm in the following situation: All lengths of members of a set A are in the range of function r. Assume that there is an algorithm M for membership in A and a constant c such that the time complexity of M for every input of length $n = r(\text{bin}(m))$ is $c(f_i(m)) + c$. It is important that the length of the input is n while the complexity depends on m (not n). To obtain a "real" complexity (depending on n) we use Lemma 5.2:

$$f_i(m) \le f_i\left(g^{-1}\left(g_0\left(2^{r(\text{bin}(m))/\alpha}\right)\right)\right) = f_i\left(g^{-1}\left(g_0\left(2^{\frac{n}{\alpha}}\right)\right)\right) \text{ a.e.}$$

Hence M works on every input of length in the range of r in $\mathcal{G}$–time.

The following lemma shows a quick verification whether a word is a code $[\dots]_r$ of a tuple.

LEMMA 5.3. *Let $k > 1$ be a natural number. Define a set A by*

$$A = \{[\mathrm{bin}(n), x_1, x_2, \ldots, x_k]_r : n \in \mathrm{N} - \{0\},\ x_2, x_3, \ldots, x_k \in \Sigma^*\}.$$

Then $A \in \mathbf{D}(\mathcal{G})$.

Proof. Consider the following algorithm:

ALGORITHM 2.

1. Input z;
2. Verify whether $z = \langle \mathrm{bin}(n), x_2, x_3, \ldots, x_k\rangle 01^m$ for words $x_2, x_3, \ldots, x_k$ and numbers n, m (otherwise REJECT);
3. Let $b = g^{-1}(2^{\lfloor |z|/\alpha \rfloor + 1})$;
4. If $m = 0$ and $n < b$, then ACCEPT;
5. If $m > 0$ and $g^{-1}(2^{\lceil |z|/\alpha \rceil}) \leq n < b$, then ACCEPT;
6. Otherwise REJECT.

By Lemma 2.2,

$$g^{-1}\left(2^{\lfloor \log g(n) \rfloor}\right) \leq n < g^{-1}\left(2^{\lfloor \log g(n) \rfloor + 1}\right). \tag{1}$$

Note that $z \in A$ iff $z = \langle \mathrm{bin}(n), x_2, x_3, \ldots, x_k\rangle 01^m$ for numbers n, m and words $x_2, x_3, \ldots, x_k$ and either $m = 0$ and $|z| \geq \alpha \lfloor \log g(n) \rfloor$ or $m > 0$ and $|z| = \alpha \lfloor \log g(n) \rfloor$. It follows from (1) and from the fact that g^{-1} is nondecreasing that

$$|z| \geq \alpha \lfloor \log g(n) \rfloor \quad \text{iff} \quad \left\lfloor \frac{|z|}{\alpha} \right\rfloor \geq \lfloor \log g(n) \rfloor \quad \text{iff} \quad g^{-1}(2^{\lfloor |z|/\alpha \rfloor + 1}) > n.$$

Similarly, we obtain

$$|z| \leq \alpha \lfloor \log g(n) \rfloor \quad \text{iff} \quad \left\lceil \frac{|z|}{\alpha} \right\rceil \leq \lfloor \log g(n) \rfloor \quad \text{iff} \quad g^{-1}(2^{\lceil |z|/\alpha \rceil}) \leq n.$$

Therefore $z \in A$ if and only if $z = \langle \mathrm{bin}(n), x_2, x_3, \ldots x_k\rangle 01^m$ for numbers n, m and words $x_2, x_3, \ldots, x_k$ and either $m = 0$ and $g^{-1}(2^{\lfloor |z|/\alpha \rfloor + 1}) > n$ or $m > 0$ and $g^{-1}(2^{\lceil |z|/\alpha \rceil}) \leq n < g^{-1}(2^{\lfloor |z|/\alpha \rfloor + 1})$. Hence the algorithm is correct.

Further we compute its time complexity. The complexity of step 2 is linear in $|z|$ because it is possible to decode members of a tuple in linear time. Since $g^{-1}(2^n)$ is time constructible, steps 3–5 run in $O(g^{-1}(2^{\lfloor |z|/\alpha \rfloor + 1}))$. The whole algorithm runs in time

$$O(g^{-1}(2^{\lfloor |z|/\alpha \rfloor + 1})) \subseteq O(g^{-1}(g_0(2^{|z|/\alpha}))),$$

hence it is a $\mathbf{D}(\mathcal{G})$–algorithm. ■

For every number b and a word x define $h_b(x) = \mathrm{nn}(1x) \bmod b$. Let S be a given g–sparse set in $\mathbf{N}(\mathcal{F})$. According to Theorem 2.3, there exists a constant c_0 such that for every n, there exists a prime p_n satisfying $p_n \leq c_0(n+1)g^2(n) + c_0$ and h_{p_n} is a perfect hashing function for $S_{=n}$. For every n, define $q_n = 2^{4\lfloor \log g(n) \rfloor}$. Let n_0 be an integer such that for every $n \geq n_0$ the following conditions hold:

(1) $c_0(n+1)g^2(n) + c_0 < q_n$;

(2) $\lfloor \log g(n) \rfloor > 14$; (Similarly to the constant 21, this constant is applied to compute an upper bound. See Lemmas 5.4, 5.5, and 5.7.)

(3) $n + 1 \leq g(n) \leq 2^n$;

(4) $g_0(n) \geq 2n$;

(5) $g^{-1}(g_0(n)) \leq g_1(g^{-1}(n))$.

Now we have $1 < p_n < q_n$; i.e., the set $\{2, 3, \ldots, q_n - 1\}$ contains a number determining a perfect hashing function for $S_{=n}$. Note that every number in the set $\{2, 3, \ldots, q_n - 1\}$ has the binary expansion of length at most $4\lfloor \log g(n) \rfloor$ (for $n \geq n_0$).

Later, we will need to verify whether a number b is less than q_n. This can be done by comparing length of the binary expansion of b with $4\lfloor \log g(n) \rfloor$. By Lemma 5.1, this takes time $O(n^2)$. Moreover, a computation of $h_b(x)$ can be made polynomial time in $|\text{bin}(b)|$ and $|x|$. Since in our case we use only $b < q_{|x|}$, time of the computation of $h_b(x)$ is a polynomial in $|x|$.

First, we show how to find perfect hashing functions for S. Define

$$A_S = \{[\text{bin}(n), \text{bin}(a)]_r : n_0 \leq n \,\&\, a < q_n \,\&\, h_a(x) = h_a(y) \text{ for some distinct } x, y \in S_{=n}\}.$$

Note that this set describes hashing functions that are not perfect.

LEMMA 5.4. *A_S belongs to $\mathbf{N}(\mathcal{G})$.*

Proof. Note that if $[\text{bin}(n), \text{bin}(a)]_r \in A_S$, then

$$\begin{aligned} |\langle \text{bin}(n), \text{bin}(a) \rangle| &\leq 2((\lfloor \log n \rfloor + 1) + 4\lfloor \log g(n) \rfloor) + 2 \quad \text{by Lemma 2.1,} \\ &\leq 10\lfloor \log g(n) \rfloor + 4 \\ &\leq 11\lfloor \log g(n) \rfloor \quad \text{because } \lfloor \log g(n) \rfloor > 14 > 4, \\ &< r(\text{bin}(n)) \quad \text{because } \alpha > 21 > 11. \end{aligned}$$

Thus every member $[\text{bin}(n), \text{bin}(a)]_r$ of A_S has length $r(\text{bin}(n))$.

A nondeterministic algorithm for membership in A_S verifies whether its input z is a correct code of a 2–tuple, decodes $\text{bin}(n)$ and $\text{bin}(a)$, verifies whether $n_0 \leq n$ and $a < q_n$, guesses x, y of length n, verifies whether $h_a(x) = h_a(y)$, and verifies their membership in S (by an $\mathbf{N}(\mathcal{F})$–algorithm for S).

By Lemma 5.3, a verification whether z is a correct code of a tuple takes $\mathbf{D}(\mathcal{G})$–time. The other steps work in $\mathbf{N}(\mathcal{F})$–time with respect to n. Since $|z| = r(\text{bin}(n))$ the algorithm works in $\mathbf{N}(\mathcal{G})$–time by Lemma 5.2. ■

Define

$$\begin{aligned} B_S = \{[\text{bin}(n), \text{bin}(a), \text{bin}(b)]_r : {} & n_0 \leq n \,\&\, a < b < q_n \,\&\, \text{there exists } k \text{ with } a < k \leq b \\ & \text{such that } [\text{bin}(n), \text{bin}(k)]_r \notin A_S\}. \end{aligned}$$

Note that this set describes intervals containing a perfect hashing function for $S_{=n}$ (for a given $n \geq n_0$).

LEMMA 5.5. *Let $\hat{f}$ be a function. If $\overline{A_S} \in \mathbf{NTIME}(\hat{f})$, then $B_S \in \mathbf{NTIME}(\hat{g} + \hat{f})$, for some $\hat{g} \in \mathcal{G}$.*

Proof. Note that if $[\text{bin}(n), \text{bin}(a), \text{bin}(b)]_r \in B_S$, then

$$\begin{aligned} |\langle \text{bin}(n), \text{bin}(a), \text{bin}(b) \rangle| &\leq 2((\lfloor \log n \rfloor + 1) + 8\lfloor \log g(n) \rfloor) + 4 \quad \text{by Lemma 2.1,} \\ &\leq 18\lfloor \log g(n) \rfloor + 6 \\ &\leq 19\lfloor \log g(n) \rfloor \quad \text{because } \lfloor \log g(n) \rfloor > 14 > 6, \\ &< r(\text{bin}(n)) \quad \text{because } \alpha > 21 > 19. \end{aligned}$$

Thus every member $[\text{bin}(n), \ldots]_r$ of B_S has length $r(\text{bin}(n))$.

A nondeterministic algorithm for membership in B_S verifies whether its input z is a correct code of a 3–tuple, decodes $\text{bin}(n)$, $\text{bin}(a)$ and $\text{bin}(b)$, verifies whether $n_0 \leq n$ and $a < b < q_n$, guesses k, verifies whether $a < k \leq b$, and verifies whether $[\text{bin}(n), \text{bin}(k)]_r \in \overline{A_S}$ (by a nondeterministic algorithm for $\overline{A_S}$).

By Lemma 5.3, a verification whether z is a correct code of a tuple takes $\mathbf{D}(\mathcal{G})$–time. Decoding 3–tuple and guessing k is in linear time. Deciding membership in $\overline{A_S}$ runs in $O(\hat{f})$; thus the described algorithm works nondeterministically in time $O(\hat{g} + \hat{f})$ for some $\hat{g} \in \mathcal{G}$. ■

The set B_S can be used to construct an algorithm Find_S computing a description of a perfect hashing function for $S_{=n}$. It is a simple binary search navigated by B_S.

ALGORITHM 3 (Find_S).

1. Input 1^n;
2. Let $L = 1$, $R = q_n - 1$;
3. Let $H = \lceil \frac{L+R}{2} \rceil$;
4. If $[\mathrm{bin}(n), \mathrm{bin}(L), \mathrm{bin}(H)]_r \in B_S$, then let $R = H$;
5. Else if $[\mathrm{bin}(n), \mathrm{bin}(H), \mathrm{bin}(R)]_r \in B_S$, then let $L = H$;
6. Else REJECT;
7. If $L + 1 = R$, then h_R is a perfect hashing function for $S_{=n}$;
8. Else go to 3.

To decide membership in B_S, a deterministic or nondeterministic algorithm may be applied. In the deterministic case, Find_S is a deterministic algorithm and no confusion is possible. If membership in B_S is checked by a nondeterministic algorithm, then there are many computations of Find_S, some of them reject and some of them declare that their output is a perfect hashing function for $S_{=n}$. This is analyzed in the following lemma.

LEMMA 5.6. *The following statements hold*:

(1) *If* $B_S \in \mathbf{D}(\mathcal{G})$ *and* Find_S *uses a* $\mathbf{D}(\mathcal{G})$*–algorithm for* B_S, *then* Find_S *runs in* $\mathbf{D}(\mathcal{F})$*–time and for every input* 1^n *with* $n \geq n_0$, *it produces a perfect hashing function for* $S_{=n}$.

(2) *Let* $\mathcal{H} \in \Omega(\mathcal{F})$ *be a base closed under product with* $\mathcal{P}$ *and under right composition with* g_1. *If* $B_S \in \mathbf{N}(\mathcal{H} \circ g^{-1}(g_0(2^{\frac{n}{\alpha}})))$ *and* Find_S *uses an* $\mathbf{N}(\mathcal{H} \circ g^{-1}(g_0(2^{\frac{n}{\alpha}})))$*–algorithm for* B_S, *then* Find_S *runs in* $\mathbf{N}(\mathcal{H})$*–time and for every input* 1^n *with* $n \geq n_0$, *every accepting computation of* Find_S *produces a perfect hashing function for* $S_{=n}$. *Moreover*, *there exists an accepting computation.*

Proof. (1) Let $f_i \in \mathcal{F}$ be a function such that $B_S \in \mathbf{DTIME}(f_i(g^{-1}(g_0(2^{\frac{n}{\alpha}}))))$. Define $t(n) = \alpha \lfloor \log g(n) \rfloor$. Since the search begins with the interval containing a perfect hashing function, in the deterministic case the algorithm always keeps an interval containing a perfect hashing function. Therefore it finds a perfect hashing function.

The complexity of step 2 is $O(n^2)$ by Lemma 5.1 and steps 3–8 run in

$$O\left(f_i\left(g^{-1}\left(g_0\left(2^{\frac{t(n)}{\alpha}}\right)\right)\right)\right).$$

Since they repeat at most $\log q_n = 4\lfloor \log g(n) \rfloor$ times, the algorithm works in time

$$O\left(n^2 + \log g(n)\, f_i\left(g^{-1}\left(g_0\left(2^{\frac{t(n)}{\alpha}}\right)\right)\right)\right) \subseteq O\left(n f_i\left(g^{-1}\left(g_0\left(2^{\frac{t(n)}{\alpha}}\right)\right)\right)\right).$$

By Lemma 5.2 and since $\mathcal{F}$ is closed under product with $\mathcal{P}$ and right composition with g_1, we obtain

$$O\left(n f_i\left(g^{-1}\left(g_0\left(2^{\frac{t(n)}{\alpha}}\right)\right)\right)\right) \subseteq O(n f_i(g_1(n))) \subseteq O(f_j(n)),$$

for some j. It is $\mathbf{D}(\mathcal{F})$–time and the first statement is proved.

(2) If Find_S uses a nondeterministic algorithm for B_S the time complexity of any accepting computation can be computed similarly to the deterministic case and we obtain that the algorithm runs in $\mathbf{N}(\mathcal{H})$–time.

If a computation outputs h_R then it is a perfect hashing function because

$$[\mathrm{bin}(n), \mathrm{bin}(R-1), \mathrm{bin}(R)]_r \in B_S.$$

Therefore every accepting computation outputs a perfect hashing function.

Since we begin with the interval containing at least one perfect hashing function, there exists an accepting computation. ■

Remark. In the beginning of this section we mentioned that our technique avoided guessing all words of $S_{=n}$. In fact, we need work with all words in $S_{=n}$ but in a different way. The set B_S can be defined as

$$B_S = \{[\text{bin}(n), \text{bin}(a), \text{bin}(b)]_r : n_0 \le n \,\&\, a < b < q_n \,\&\, (\exists k)(a < k \le b \text{ and } (\forall x, y \in S_{=n})(h_k(x) \ne h_k(y)))\}.$$

Hence the guessing of all words of $S_{=n}$ is replaced by the universal quantificator.

Hartmanis and Yesha [15] used the original upward separation technique to prove that there is a polynomially sparse set T in **NP** such that all polynomially sparse sets in **NP** are Turing reducible to T. Since in our case B_S need not to belong to $\mathbf{N}(\mathcal{G})$ (generally, it belongs to "$\exists\forall\mathbf{N}(\mathcal{G})$") our technique cannot be directly used to prove that there is an $n^{O(\log n)}$–sparse set D in **NP** such that every $n^{O(\log n)}$–sparse set in **NP** is Turing reducible to D. The question whether such a set exists remains open.

Define

$$C_S = \{[\text{bin}(n), \text{bin}(t), \text{bin}(a), \text{bin}(i), d]_r : n_0 \le n \,\&\, t < a < q_n \,\&\, 1 \le i \le n \,\&\, \text{there exists } x \in S_{=n} \text{ such that } h_a(x) = t \text{ and the } i\text{–th digit of } x \text{ is } d\}.$$

LEMMA 5.7. *C_S belongs to $\mathbf{N}(\mathcal{G})$.*

Proof. If $[\text{bin}(n), \text{bin}(t), \text{bin}(a), \text{bin}(i), d]_r \in C_S$, then

$$\begin{aligned} |\langle \text{bin}(n), \text{bin}(t), \text{bin}(a), \text{bin}(i), d\rangle| &\le 2((2\lfloor \log n\rfloor + 2) \\ &\quad + 8\lfloor \log g(n)\rfloor + 1) + 8 \quad \text{by Lemma 2.1,} \\ &\le 20\lfloor \log g(n)\rfloor + 14 \\ &\le 21\lfloor \log g(n)\rfloor \quad \text{because } \lfloor \log g(n)\rfloor > 14, \\ &< r(\text{bin}(n)) \quad \text{because } \alpha > 21. \end{aligned}$$

Thus every member $[\text{bin}(n), \dots]_r$ of C_S has length $r(\text{bin}(n))$.

A nondeterministic algorithm for membership in C_S verifies whether its input z is a correct code of a 5–tuple, decodes $\text{bin}(n)$, $\text{bin}(a)$, $\text{bin}(t)$, $\text{bin}(i)$ and d, verifies the inequalities $n_0 \le n$, $1 \le i \le n$, $t < a < q_n$, guesses x of length n, verifies whether $h_a(x) = t$, verifies membership of x in S (by an $\mathbf{N}(\mathcal{F})$–algorithm for S), and verifies whether the ith digit of x is d.

By Lemma 5.3, a verification whether z is a correct code of a tuple takes $\mathbf{D}(\mathcal{G})$–time. The other steps work in $\mathbf{N}(\mathcal{F})$–time with respect to n. Since $|z| = r(\text{bin}(n))$ the algorithm works in $\mathbf{N}(\mathcal{G})$–time by Lemma 5.2. ■

Let x be a word and we want to decide whether $x \in S$. The function h_R obtained by Find_S on the input $1^{|x|}$ is a perfect hashing function for $S_{=|x|}$. Together with the set C_S we can use it to construct an algorithm determining whether x of length at least n_0 belongs to S.

ALGORITHM 4 (Test_S).

1. Input x;
2. Let R be the output of Find_S on the input $1^{|x|}$;
3. Let $t = \text{bin}(h_R(x))$, $n = |x|$;
4. For $i = 1, 2, \dots, n$ do
5. Let d be the i–th digit of x;
6. If $[\text{bin}(n), t, \text{bin}(R), i, d]_r \in \overline{C_S}$, REJECT;
7. ACCEPT.

Let Test'$_S$ be Test$_S$ with exchanged accepting and rejecting states.

LEMMA 5.8. *The following statements hold*:

(1) *If* Find$_S$ *is a* $\mathbf{D}(\mathcal{F})$*–algorithm and* $\overline{C_S} \in \mathbf{D}(\mathcal{G})$ *and* Test$_S$ *uses a* $\mathbf{D}(\mathcal{G})$*–time algorithm for* $\overline{C_S}$*, then it works in* $\mathbf{D}(\mathcal{F})$*–time and it accepts* x *with* $|x| \geq n_0$ *iff* $x \in S$*;*

(2) *Let* $\mathcal{H} \in \Omega(\mathcal{F})$ *be a base closed under product with* $\mathcal{P}$ *and under right composition with* g_1*. If* Find$_S$ *is an* $\mathbf{N}(\mathcal{H})$*–algorithm and* $\overline{C_S} \in \mathbf{N}(\mathcal{H} \circ g^{-1}(g_0(2^{\frac{n}{\alpha}})))$ *and* Test'$_S$ *uses an* $\mathbf{N}(\mathcal{H} \circ g^{-1}(g_0(2^{\frac{n}{\alpha}})))$*–algorithm for* $\overline{C_S}$ *then every computation works in* $\mathbf{N}(\mathcal{H})$*–time. Moreover, there is an accepting computation for* x *with* $|x| \geq n_0$ *iff* $x \in \overline{S}$*.*

Proof. First, we prove that the algorithms Test$_S$ and Test'$_S$ are correct. Let x be a word of length at least n_0. Let x_i denote the ith bit of x. Assume that $x \in S$. By the definition of C_S and since h_R is perfect, for every i with $1 \leq i \leq |x|$ we have

$$[\text{bin}(|x|), \text{bin}(h_R(x)), \text{bin}(R), i, d]_r \in C_S \text{ if and only if } d = x_i.$$

Therefore Test$_S$ accepts every member of S of length at least n_0 while Test'$_S$ rejects all these words.

If $x \notin S$, we distinguish two cases:

Case 1. There is no word $y \in S_{=|x|}$ with $h_R(y) = h_R(x)$. Then, by the definition of C_S, $[\text{bin}(|x|), \text{bin}(h_R(x)), \text{bin}(R), i, x_i]_r \notin C_S$, for every i with $1 \leq i \leq |x|$. Therefore x is rejected by Test$_S$ and accepted by Test'$_S$.

Case 2. There is a word $y \in S_{=|x|}$ with $h_R(y) = h_R(x)$. Since h_R is perfect, $h_R(z) \neq h_R(x)$ for every $z \in S_{=|x|} - \{y\}$. It follows from the definition of C_S that for every $d \in \{0, 1\}$, $[\text{bin}(|x|), \text{bin}(h_R(x)), \text{bin}(R), i, d]_r \in C_S$ iff the ith bit of y is d. Since there is some j such that x_j differs from the jth bit of y, $[\text{bin}(|x|), \text{bin}(h_R(x)), \text{bin}(R), j, x_j]_r \notin C_S$. Hence x is rejected by Test$_S$ and accepted by Test'$_S$.

Thus x of length at least n_0 is accepted by Test$_S$ iff $x \in S$ and x is accepted by Test'$_S$ iff $x \notin S$. The algorithms are correct.

Finally we compute the complexity of Test$_S$. The complexity of Test'$_S$ can be computed similarly. Let $f_i \in \mathcal{F}$ be a function such that

$$\overline{C_S} \in \mathbf{DTIME}(f_i(g^{-1}(g_0(2^{n/\alpha})))).$$

Define $t(n) = \alpha \lfloor \log g(n) \rfloor$. Since Find$_S$ is a $\mathbf{D}(\mathcal{F})$–algorithm, the complexity of step 2 is $O(f_j)$ for some j. The complexity of steps 3–7 is bounded by

$$O\left(n f_i\left(g^{-1}\left(g_0\left(2^{\frac{t(n)}{\alpha}}\right)\right)\right)\right)$$

because it is at most n times repetition of deciding membership in $\overline{C_S}$. Since $\mathcal{F}$ is closed under product with $\mathcal{P}$ and right composition with g_1 and by Lemma 5.2, Test$_S$ is a $\mathbf{D}(\mathcal{F})$–algorithm. ■

In the next section we show results which can be obtained by this upward separation technique.

6. MAIN RESULTS

THEOREM 6.1. *Let* (X) *hold. If* $\mathbf{N}(\mathcal{G}) = \mathbf{D}(\mathcal{G})$*, then* $\mathbf{N}(\mathcal{F}) - \mathbf{D}(\mathcal{F})$ *contains no* g*–sparse set.*

Proof. Let us assume that $\mathbf{N}(\mathcal{G}) = \mathbf{D}(\mathcal{G})$ and let $S \in \mathbf{N}(\mathcal{F})$ be a g–sparse set. We will prove that $S \in \mathbf{D}(\mathcal{F})$.

By Lemma 5.4, the set A_S is in $\mathbf{N}(\mathcal{G}) = \mathbf{D}(\mathcal{G})$. By Lemma 5.5, the set B_S is in $\mathbf{N}(\mathcal{G}) = \mathbf{D}(\mathcal{G})$. By Lemma 5.6, Find$_S$ using an $\mathbf{D}(\mathcal{G})$–algorithm for B_S finds a perfect hashing function for $S_{=n}$ in $\mathbf{D}(\mathcal{F})$–time. Therefore, for a given x, it is possible to find the perfect hashing function h_R in $\mathbf{D}(\mathcal{F})$–time.

By Lemma 5.7, the set C_S is in $\mathbf{N}(\mathcal{G}) = \mathbf{D}(\mathcal{G})$; thus $\overline{C_S} \in \mathbf{D}(\mathcal{G})$. If a $\mathbf{D}(\mathcal{G})$–algorithm for $\overline{C_S}$ is used in Test$_S$, then Test$_S$ is a $\mathbf{D}(\mathcal{F})$–algorithm, by Lemma 5.8.

Thus we constructed a $\mathbf{D}(\mathcal{F})$–algorithm deciding membership in S but only for words of length at least n_0. This algorithm can be easily modified to recognize all members of the set S with a constant time loss. ■

THEOREM 6.2. *Let* (X) *hold. Let* $\mathcal{H} \in \Omega(\mathcal{F})$ *be a base closed under product with* $\mathcal{P}$ *and right composition with* g_1. *If* $co\mathbf{N}(\mathcal{G}) \subseteq \mathbf{N}(\mathcal{H} \circ g^{-1}(g_0(2^{n/\alpha})))$, *then the complement of every g–sparse set in* $\mathbf{N}(\mathcal{F})$ *is in* $\mathbf{N}(\mathcal{H})$.

Proof. Let us assume that $co\mathbf{N}(\mathcal{G}) \subseteq \mathbf{N}(\mathcal{H} \circ g^{-1}(g_0(2^{n/\alpha})))$ and let S be a g–sparse set in $\mathbf{N}(\mathcal{F})$.

By Lemma 5.4, A_S belongs to $\mathbf{N}(\mathcal{G})$. Hence $\overline{A_S} \in co\mathbf{N}(\mathcal{G}) \subseteq \mathbf{N}(\mathcal{H} \circ g^{-1}(g_0(2^{n/\alpha})))$. Since $\mathcal{H} \in \Omega(\mathcal{F})$, we have $\mathbf{N}(\mathcal{G}) \subseteq \mathbf{N}(\mathcal{H} \circ g^{-1}(g_0(2^{n/\alpha})))$. By Lemma 5.5, the set B_S is in $\mathbf{N}(\mathcal{H} \circ g^{-1}(g_0(2^{n/\alpha})))$; thus there exists an $\mathbf{N}(\mathcal{H} \circ g^{-1}(g_0(2^{n/\alpha})))$–algorithm for B_S. Using this algorithm in Find_S, an $\mathbf{N}(\mathcal{H})$–algorithm computing a perfect hashing function is obtained by Lemma 5.6.

By Lemma 5.7, the set C_S is in $\mathbf{N}(\mathcal{G})$; therefore $\overline{C_S}$ is in $\mathbf{N}(\mathcal{H} \circ g^{-1}(g_0(2^{n/\alpha})))$. Using an $\mathbf{N}(\mathcal{H} \circ g^{-1}(g_0(2^{n/\alpha})))$–algorithm for $\overline{C_S}$ in Test'$_S$, by Lemma 5.8, we obtain an $\mathbf{N}(\mathcal{H})$–algorithm for $\overline{S}$ working on inputs of lengths at least n_0. The algorithm can be easily modified to accept S with a constant time loss. ■

COROLLARY 6.1. *Let* (X) *hold. If* $co\mathbf{N}(\mathcal{G}) = \mathbf{N}(\mathcal{G})$ *then every g–sparse set in* $\mathbf{N}(\mathcal{F})$ *has its complement also in* $\mathbf{N}(\mathcal{F})$.

Proof. In Theorem 6.2 we set $\mathcal{H} = \mathcal{F}$. ■

LEMMA 6.1. *Let* $b, c \geq 1$ *be numbers. Define* $\mathcal{F} = \mathcal{P}$, $g(n) = 2^{b\lceil \log n \rceil^c}$, $g_0(n) = n^{22}$ *and* $g_1(n) = n^{\lceil \sqrt[c]{22} \rceil + 1}$. *Then* $\mathcal{F}, g, g_0, g_1$ *satisfy* (X) *with* $\alpha = 22$.

Proof. Clearly, $\mathcal{F}$ is closed under product with $\mathcal{P}$ and $n+1 \leq g(n) \leq 2^n$ a.e. Further we determine $g^{-1}(n)$:

$$g^{-1}(n) = \min\{t : n \leq 2^{b\lceil \log t \rceil^c}\} = \min\left\{t : \sqrt[c]{\frac{\log n}{b}} \leq \lceil \log t \rceil\right\}$$
$$= \min\left\{t : \left\lceil \sqrt[c]{\frac{\log n}{b}} \right\rceil \leq \lceil \log t \rceil\right\} = 2^{\lceil \sqrt[c]{\frac{\log n}{b}} \rceil - 1} + 1.$$

We have to verify that

(a) $g^{-1}(g_0(2^{n/\alpha}))$ is time constructible. This is true because

$$g^{-1}\left(g_0\left(2^{\frac{n}{22}}\right)\right) = g^{-1}(2^n) = 2^{\lceil \sqrt[c]{\frac{n}{b}} \rceil - 1} + 1,$$

and this is a time constructible function.

(b) $g_0(n) \geq 2n$ a.e. This is apparent.

(c) $g^{-1}(g_0(n)) \leq g_1(g^{-1}(n))$ a.e. This is true because

$$g^{-1}(g_0(n)) = 2^{\lceil \sqrt[c]{\frac{22 \log n}{b}} \rceil - 1} + 1 \leq 2^{\lceil \sqrt[c]{22} \rceil \lceil \sqrt[c]{\frac{\log n}{b}} \rceil - 1} + 1$$
$$\leq \left(2^{\lceil \sqrt[c]{\frac{\log n}{b}} \rceil - 1} + 1\right)^{\lceil \sqrt[c]{22} \rceil + 1} = g_1(g^{-1}(n))$$

for every sufficiently large n.

(d) $\mathcal{F}$ is closed under right composition with g_1. This is true because $\mathcal{P}$ is closed under composition and g_1 is a polynomial.

Thus (X) is fulfilled. ■

The following theorem solves the open question of Hartmanis [12] (for $c = 2$).

THEOREM 6.3. *Let* $c > 1$ *be a natural number. Then the following three conditions are equivalent*:

(1) $\mathbf{NP} - \mathbf{P}$ *contains an* $\Omega(2^{\sqrt[c]{n}})$*–padded set*;

(2) $\mathbf{NP} - \mathbf{P}$ *contains an* $\mathcal{EL}_c$*–sparse set*;

(3) $\mathbf{N}(\mathcal{ER}_c) \neq \mathbf{D}(\mathcal{ER}_c)$.

Proof. (1) $\Rightarrow$ (2): By Lemma 4.1, every $\Omega(2^{\sqrt[c]{n}})$–padded set is $2^{\lceil \log^c dn \rceil}$–sparse for some constant d.
(2) $\Rightarrow$ (3): Let $g(n) = 2^{\lceil \log n \rceil^c}$ for some constant b. Let $\alpha, \mathcal{F}, g_0, g_1$ be defined as in Lemma 6.1. Then they satisfy (X). By Theorem 6.1, if there is a g–sparse set in $\mathbf{NP} - \mathbf{P}$, then the base

$$\mathcal{G} = \mathcal{F} \circ g^{-1}(g_0(2^{n/\alpha})) = \left\{ \left(2^{\lceil \sqrt[c]{\frac{n}{b}} \rceil - 1} + 1 \right)^i \right\}_i$$

does not collapse. Note that this base is in $\Theta(\mathcal{ER}_c)$. Therefore the existence of an $\mathcal{EL}_c$–sparse set in $\mathbf{NP} - \mathbf{P}$ implies that $\mathcal{ER}_c$ does not collapse.
(3) $\Rightarrow$ (1): Since $f(n) = 2^{\lceil \sqrt[c]{n} \rceil}$ is time constructible and $\mathcal{ER}_c \in \Theta(\mathcal{P} \circ f)$, this implication follows from Theorem 4.2. ■

Theorem 1.2 from the Introduction is a direct corollary of Theorem 6.3 for $c = 2$.

THEOREM 6.4. *Let c be a number. Then every $\mathcal{EL}_c$–sparse set in **NP** has its complement also in **NP** iff $\mathbf{N}(\mathcal{ER}_c) = co\mathbf{N}(\mathcal{ER}_c)$.*

Proof. Let $g(n) = 2^{\lceil \log n \rceil^c}$ for some constant b. Let $\alpha, \mathcal{F}, g_0, g_1$ be defined as in Lemma 6.1. Then they satisfy (X). By Corollary 6.1, if $\mathbf{N}(\mathcal{ER}_c) = co\mathbf{N}(\mathcal{ER}_c)$, then every $\mathcal{EL}_c$–sparse set in **NP** has its complement also in **NP**.
Assume that every $\mathcal{EL}_c$–sparse set in **NP** has its complement also in **NP**. Let $A \in \mathbf{N}(\mathcal{ER}_c)$ be a set and $f(n) = 2^{\lceil \sqrt[c]{n} \rceil}$. Then $A_f \in \mathbf{NP}$ by Theorem 4.2. By Lemma 4.1, A_f is $\mathcal{EL}_c$–sparse; hence $\overline{A_f} \in \mathbf{NP}$. By Theorem 4.2, $A \in \mathbf{N}(\mathcal{ER}_c)$. ■

Theorem 1.4 from the Introduction is a direct corollary of Theorem 6.4 for $c = 2$.
Theorems 4.2 and 6.1 can be also used to transfer the existence of a set of certain density in $\mathbf{N}(\mathcal{F}) - \mathbf{D}(\mathcal{F})$ to the existence of a set of another density in $\mathbf{N}(\mathcal{G}) - \mathbf{D}(\mathcal{G})$.

THEOREM 6.5. *Let $c > 1$ be a natural number and let g be a nondecreasing function satisfying these conditions*:

(1) $n + 1 \leq g(n) \leq 2^{2^{\lceil \sqrt[c]{\log n} \rceil - 1}}$ *a.e.*;
(2) $h_0(n) = 2^{\lceil \sqrt[c]{\log g^{-1}(2^n)} \rceil}$ *and* $h_1(n) = g^{-1}(2^n)$ *are time constructible functions*;
(3) *there exists some b such that $g^{-1}(n^2) \leq (g^{-1}(n))^b$ a.e.*

Then the following conditions are equivalent.

(i) $\mathbf{NP} - \mathbf{P}$ *contains a g–sparse set*;
(ii) $\mathbf{NP} - \mathbf{P}$ *contains a $g^{-1}(2^n)$–padded set*;
(iii) $\mathbf{N}(\mathcal{EL}_c) - \mathbf{D}(\mathcal{EL}_c)$ *contains a $g(2^{\lceil \log n \rceil^c})$–sparse set*;
(iv) $\mathbf{N}(\mathcal{EL}_c) - \mathbf{D}(\mathcal{EL}_c)$ *contains a $2^{\lceil \sqrt[c]{\log g^{-1}(2^n)} \rceil}$–padded set*;
(v) *the base $\mathcal{G} = \{(g^{-1}(2^n))^i\}_{i \in \mathbb{N}}$ does not collapse.*

Proof. Without loss of generality assume that $b \geq 2$.
(i)$\Rightarrow$(v). We prove that if $\mathcal{G}$ collapses, then, by Theorem 6.1, there is no g–sparse set in $\mathbf{NP} - \mathbf{P}$. First of all, we verify (X):
Let $\alpha = 22$, $\mathcal{F} = \mathcal{P}$, $g_0(n) = n^\alpha$, and $g_1(n) = n^{b^{\lceil \log \alpha \rceil}}$. By (1), $n + 1 \leq g(n) \leq 2^n$ a.e. Apparently, $\mathcal{F}$ is closed under product. We have to verify that

(a) $g^{-1}(g_0(2^{n/\alpha}))$ is time constructible. This is true because $g^{-1}(g_0(2^{n/\alpha})) = h_1(n)$.
(b) $g_0(n) \geq 2n$ a.e. This is clear.
(c) $g^{-1}(g_0(n)) \leq g_1(g^{-1}(n))$ a.e. This is true because the following inequalities hold for almost

every n:

$$g^{-1}(g_0(n)) \le g^{-1}\big(n^{2^{\lceil \log \alpha \rceil}}\big) \le g^{-1}\Big(\big(n^{2^{\lceil \log \alpha \rceil - 1}}\big)^2\Big) \tag{1}$$

$$\le \Big(g^{-1}\big(n^{2^{\lceil \log \alpha \rceil - 1}}\big)\Big)^b \le \Big(g^{-1}\Big(\big(n^{2^{\lceil \log \alpha \rceil - 2}}\big)^2\Big)\Big)^b \tag{2}$$

$$\le \Big(g^{-1}\big(n^{2^{\lceil \log \alpha \rceil - 2}}\big)\Big)^{b^2} \le \cdots \le (g^{-1}(n))^{b^{\lceil \log \alpha \rceil}} = g_1(g^{-1}(n)). \tag{3}$$

(d) $\mathcal{F}$ is closed under right composition with g_1. This is true because $\mathcal{P}$ is closed under composition and g_1 is a polynomial.

Thus (X) is satisfied. By Theorem 6.1, if there is a g–sparse set in $\mathbf{NP} - \mathbf{P}$, then the base $\mathcal{G}$ does not collapse.

(v)⇒(ii). Since $g^{-1}(2^n)$ is time constructible, this follows from Theorem 4.2.

(ii)⇒(i). Note that $\mathbf{NP} - \mathbf{P}$ contains an h_1–padded set iff it contains an $h_2(n) = h_1(n) + 1$–padded set. By Lemma 4.1, every $h_2(n)$–padded set is $2^{h_2^{-1}(n+1)}$–sparse. It follows from the definition of an inverse function that for every function h_3, if $h_2(h_3(n)) \ge n$ a.e., then $h_2^{-1}(n) \le h_3(n)$ a.e. Since for almost every n,

$$h_2(\lceil \log g(n-1) \rceil + 1)$$
$$= g^{-1}\big(2^{\lceil \log g(n-1) \rceil + 1}\big) + 1 \ge g^{-1}(2g(n-1)) + 1 \ge g^{-1}(g(n-1)+1) + 1 > n - 1 + 1,$$

we obtain $h_2^{-1}(n+1) \le \lceil \log g(n) \rceil + 1$ a.e. Hence every h_2–padded set is g–sparse.

(iii)⇒(v). Let $\hat{g}(n) = g(2^{\lceil \log n \rceil^c})$. We prove that if the base $\mathcal{G}$ collapses, then, by Theorem 6.1, there is no $\hat{g}$–sparse set in $\mathbf{N}(\mathcal{EL}_c) - \mathbf{D}(\mathcal{EL}_c)$. First, we verify (X):

Let $\alpha = 22$, $\mathcal{F} = \mathcal{EL}_c$, $g_0(n) = n^\alpha$, and $g_1(n) = n^{b^{\lceil \log \alpha \rceil}+1}$. The base $\mathcal{F}$ is closed under product. We have $n + 1 \le \hat{g}(n) \le 2^n$ a.e. by (1). Define

$$h_4(n) = 2^{\lceil \sqrt[c]{\log g^{-1}(n)} \rceil - 1} + 1.$$

We will prove that $h_4(n) = \hat{g}^{-1}(n)$ almost everywhere. For almost every n we have:

$$\hat{g}(h_4(n)) = \hat{g}\big(2^{\lceil \sqrt[c]{\log g^{-1}(n)} \rceil - 1} + 1\big)$$
$$= g\big(2^{\lceil \log(2^{\lceil \sqrt[c]{\log g^{-1}(n)} \rceil - 1} + 1) \rceil^c}\big) \ge g\big(2^{\lceil \sqrt[c]{\log g^{-1}(n)} \rceil^c}\big) \ge g\big(2^{\log g^{-1}(n)}\big) \ge g(g^{-1}(n)) \ge n.$$

Hence $h_4(n) \ge \hat{g}^{-1}(n)$ a.e. Further, for almost every n,

$$\hat{g}(h_4(n) - 1) = \hat{g}\big(2^{\lceil \sqrt[c]{\log g^{-1}(n)} \rceil - 1}\big) = g\big(2^{\lceil \log(2^{\lceil \sqrt[c]{\log g^{-1}(n)} \rceil - 1}) \rceil^c}\big)$$
$$= g\big(2^{\lceil \sqrt[c]{\log g^{-1}(n)} - 1 \rceil^c}\big) \le g\big(2^{\log g^{-1}(n)} - 1\big) \le g(g^{-1}(n) - 1) < n.$$

Hence $h_4(n) \le \hat{g}^{-1}(n)$ a.e.

We have to verify that

(a) $\hat{g}^{-1}(g_0(2^{\frac{n}{\alpha}}))$ is time constructible. This is true because

$$\hat{g}^{-1}\big(g_0\big(2^{\frac{n}{\alpha}}\big)\big) = 2^{\lceil \sqrt[c]{\log g^{-1}(2^n)} \rceil - 1} + 1 \text{ a.e.}$$

and this is a time constructible function because h_0 is time constructible.

(b) $g_0(n) \ge 2n$ a.e. This is clear.

(c) $\hat{g}^{-1}(g_0(n)) \le g_1(\hat{g}^{-1}(n))$ a.e. This is true because

$$\hat{g}^{-1}(g_0(n)) = 2^{\lceil \sqrt[c]{\log g^{-1}(n^\alpha)}\rceil - 1} + 1 \le 2^{\lceil \sqrt[c]{\log (g^{-1}(n))^{b^{\lceil \log \alpha\rceil}}}\rceil - 1} + 1$$
$$\le 2^{\lceil \sqrt[c]{b^{\lceil \log \alpha\rceil}}\rceil \lceil \sqrt[c]{\log g^{-1}(n)}\rceil - 1} + 1 \le g_1(\hat{g}^{-1}(n)),$$

for every sufficiently large n.

(d) $\mathcal{F}$ is closed under right composition with g_1. This is true because $\mathcal{EL}_c$ is closed under right composition with polynomials.

Thus (X) is satisfied. By Theorem 6.1, if there is a $\hat{g}$–sparse set in $\mathbf{N}(\mathcal{EL}_c) - \mathbf{D}(\mathcal{EL}_c)$, then the base $\mathcal{EL}_c \circ \hat{g}^{-1}(g_0(2^{n/\alpha})) = \mathcal{EL}_c \circ \hat{g}^{-1}(2^n)$ does not collapse. Since

$$\mathcal{EL}_c \circ \hat{g}^{-1}(2^n) = \left\{2^{i\lceil \log(2^{\lceil \sqrt[c]{\log g^{-1}(2^n)}\rceil - 1} + 1)\rceil^c}\right\}_i,$$

it is easy to see that this base is in $\Theta(\mathcal{G})$.

(v)⇒(iv). Since $\mathcal{G} \in \Theta(\mathcal{EL}_c \circ \hat{g}^{-1}(2^n)) = \Theta(\mathcal{EL}_c \circ h_0)$, this implication is a direct application of Theorem 4.2.

(iv)⇒(iii). Let $h_5(n) = g(2^{\lceil \log n\rceil^c})$. Note that $\mathbf{N}(\mathcal{EL}_c) - \mathbf{D}(\mathcal{EL}_c)$ contains an h_0–padded set iff it contains an $h_6(n) = h_0(n) + 1$–padded set. We will prove that $h_6^{-1}(n) \le \lceil \log h_5(n-1)\rceil + 1$ a.e. For almost every n, we have

$$h_6(\lceil \log h_5(n-1)\rceil + 1) = 2^{\lceil \sqrt[c]{\log g^{-1}\left(2^{\lceil \log h_5(n-1)\rceil + 1}\right)}\rceil} + 1$$
$$\ge 2^{\lceil \sqrt[c]{\log g^{-1}(h_5(n-1)+1)}\rceil} + 1 \ge 2^{\lceil \sqrt[c]{\log 2^{\lceil \log (n-1)\rceil^c}}\rceil} + 1 \ge 2^{\lceil \log (n-1)\rceil} + 1 \ge n.$$

Therefore $h_6^{-1}(n) \le \lceil \log h_5(n-1)\rceil + 1$ a.e. By Lemma 4.1, every h_6–padded set is $2^{h_6^{-1}(n+1)}$–sparse and hence also $h_5(n)$–sparse. ■

If we set $c = 2$ in Theorem 6.5, we obtain Theorem 1.3 from the Introduction.

COROLLARY 6.2. *There is an $\mathcal{EL}_2$–sparse set in $\mathbf{NP} - \mathbf{P}$ iff there is an $\mathcal{EL}_4$–sparse set in $\mathbf{N}(\mathcal{EL}_2) - \mathbf{D}(\mathcal{EL}_2)$.*

Proof. Let $d \ge 1$ be any number. Define $g_d(n) = 2^{d\lceil \log (n+1)\rceil^2}$. Note that $g_d(n) > n$ almost everywhere and $g_d(n) \le 2^{2^{\lceil \sqrt{\log n}\rceil - 1}}$ almost everywhere. Moreover,

$$g_d^{-1}(n) = \min\{t : g_d(t) \ge n\} = \min\{t : \lceil \log (t+1)\rceil \ge \sqrt{(\log n)/d}\} = 2^{\lceil \sqrt{(\log n)/d}\rceil - 1}.$$

Define

$$h_0(n) = 2^{\lceil \sqrt{\log g_d^{-1}(2^n)}\rceil} = 2^{\lceil \sqrt{\lceil \sqrt{n/d}\rceil - 1}\rceil} \quad \text{and} \quad h_1(n) = g_d^{-1}(2^n) = 2^{\lceil \sqrt{n/d}\rceil - 1}.$$

Apparently, both h_0 and h_1 are time constructible.

Further, for every sufficiently large n,

$$g_d^{-1}(n^2) = 2^{\lceil \sqrt{(\log n^2)/d}\rceil - 1} \le \left(2^{\lceil \sqrt{(\log n)/d}\rceil - 1}\right)^2;$$

hence by Theorem 6.5 with $c = 2$, $\mathbf{NP} - \mathbf{P}$ contains a g_d–sparse set iff $\mathbf{N}(\mathcal{EL}_2) - \mathbf{D}(\mathcal{EL}_2)$ contains a $g_d(2^{\lceil \log n\rceil^2})$–sparse set.

Finally note that every set is $\mathcal{EL}_2$–sparse iff it is g_d–sparse for some d and every set is $\mathcal{EL}_4$–sparse iff it is $g_d(2^{\lceil \log n\rceil^2})$–sparse for some d. ■

Our technique is more general than the technique of [12]. Therefore we obtain the result of [12] as a consequence of Theorem 6.1.

THEOREM 6.6 [12]. *The following three conditions are equivalent*:

(1) $\mathbf{NP} - \mathbf{P}$ *contains a tally set*;

(2) $\mathbf{NP} - \mathbf{P}$ *contains a* $\mathcal{P}$*–sparse set*;

(3) $\mathbf{NEXT} \neq \mathbf{DEXT}$.

The importance of Theorem 6.6 is in the connection of tally sets and $\mathcal{P}$–sparse sets. The upward separation technique allows us to relate tally sets to nonpolynomially sparse sets but in another class difference:

THEOREM 6.7. *The following three conditions are equivalent*:

(1) $\mathbf{N}(\mathcal{EPL}) - \mathbf{D}(\mathcal{EPL})$ *contains a tally set*;

(2) $\mathbf{N}(\mathcal{EPL}) - \mathbf{D}(\mathcal{EPL})$ *contains an* $\mathcal{EPL}$*–sparse set*;

(3) $\mathbf{NEXPTIME} \neq \mathbf{EXPTIME}$.

Proof. (3) $\Rightarrow$ (1). If $A \in \mathbf{NEXPTIME} - \mathbf{EXPTIME}$, then its exponentially padded version is in $\mathbf{N}(\mathcal{EPL}) - \mathbf{D}(\mathcal{EPL})$, by Theorem 4.2. Since exponentially padded sets are polynomial time isomorphic to tally sets (Theorem 4.1) there is a tally set in $\mathbf{N}(\mathcal{EPL}) - \mathbf{D}(\mathcal{EPL})$.

(1) $\Rightarrow$ (2). Every tally set is n–sparse and hence also $\mathcal{EPL}$–sparse.

(2) $\Rightarrow$ (3). Let $g(n) = 2^{\lceil \log n \rceil^c}$ for some constant $c \geq 2$. We prove that if $\mathcal{EXP}$ collapses, then, by Theorem 6.1, there is no g–sparse set in $\mathbf{N}(\mathcal{EPL}) - \mathbf{D}(\mathcal{EPL})$. First of all, we verify (X):

Let $\alpha = 22$, $\mathcal{F} = \mathcal{EPL}$, and $g_0(n) = g_1(n) = n^\alpha$. Now $\mathcal{F}$ is closed under product. We have $n + 1 \leq g(n) \leq 2^n$ a.e. and

$$g^{-1}(n) = \min\{t : 2^{\lceil \log t \rceil^c} \geq n\} = \min\{t : \lceil \log t \rceil \geq \sqrt[c]{\log n}\} = 2^{\lceil \sqrt[c]{\log n} \rceil - 1} + 1.$$

We have to verify that

(a) $g^{-1}(g_0(2^{n/\alpha}))$ is time constructible. This is true because

$$g^{-1}(g_0(2^{n/\alpha})) = 2^{\lceil \sqrt[c]{n} \rceil - 1} + 1.$$

(b) $g_0(n) \geq 2n$ a.e. This is clear.

(c) $g^{-1}(g_0(n)) \leq g_1(g^{-1}(n))$ a.e. This is true because

$$g^{-1}(g_0(n)) = 2^{\lceil \sqrt[c]{\log n^\alpha} \rceil - 1} + 1 \leq 2^{\lceil \sqrt[c]{\alpha} \rceil \lceil \sqrt[c]{\log n} \rceil - 1} + 1 \leq g_1(g^{-1}(n))$$

for every sufficiently large n.

(d) $\mathcal{F}$ is closed under right composition with g_1. This is true because $\mathcal{EPL}$ is closed under right composition with $\mathcal{P}$ and g_1 is a polynomial.

Thus (X) is satisfied. By Theorem 6.1, if there is a g–sparse set in $\mathbf{N}(\mathcal{EPL}) - \mathbf{D}(\mathcal{EPL})$, then the base

$$\mathcal{G} = \mathcal{EPL} \circ g^{-1}(g_0(2^{n/\alpha})) = \mathcal{EPL} \circ g^{-1}(2^n) = \left\{2^{\lceil \log(2^{\lceil \sqrt[c]{n} \rceil - 1} + 1) \rceil^i}\right\}_i$$

does not collapse. Note that $\mathcal{G} \in \Theta(\mathcal{EXP})$. ■

Similar theorems can be shown for other bases. It is interesting that conditions (3) in both theorems are concerned with tally sets but conditions (2) are concerned with sets with different density ($\mathcal{P}$ and $\mathcal{EPL}$–sparse). For higher than polynomial base classes, the existence of sets of a greater density is still equivalent to the existence of tally sets.

Remark. The previous two results also follow from the original upward separation technique and its generalization in [1].

7. LIMITATIONS

In this section limitations of the technique are studied. The usual way to prove limitations is to find some "bad" relativisation. It is easy to see that our technique is relativisable. Thus, to find a limitation, it suffices to construct a "bad" oracle. There are many papers where such an oracle is constructed. One of them was made by Hartmanis, Immerman, and Sewelson:

THEOREM 7.1 [13]. *There is an oracle A such that $\mathbf{NP}^A - \mathbf{P}^A$ contains a polynomially sparse set (hence $\mathbf{NEXT}^A \neq \mathbf{DEXT}^A$) but $\mathbf{N}(\mathcal{EPL})^A = \mathbf{D}(\mathcal{EPL})^A$.*

Theorem 7.1 implies that no relativisable technique can connect the existence of a sparse set in $\mathbf{NP} - \mathbf{P}$ with the existence of any set in $\mathbf{N}(\mathcal{EPL}) - \mathbf{D}(\mathcal{EPL})$. Since our upward separation technique is relativisable it cannot be used to prove a result in this direction.

Using bases we give an example of a similar limitation. It is based on the following theorem.

THEOREM 7.2. *Let $\{f_i\}$ be a base and h be a function such that $f_i \in o(h)$ for all i. Then the sequence of functions $\{f_i \circ h\}$ is not closed under composition.*

Proof. Note that $f \in o(h)$ iff for every c, $cf(n)+1 \leq h(n)$ almost everywhere. Recall that elements of every base are nondecreasing functions. Assume that $\{f_i \circ h\}$ is closed under composition. Then for any i and j there exist k, c such that

$$f_i(h(f_j(h(n)))) \leq cf_k(h(n)) \text{ a.e.} \tag{2}$$

Fix such i, j, k, and c. Now we have

$$\begin{aligned}
f_i(cf_k(f_j(h(n))) + 1) &\leq f_i(h(f_j(h(n)))) \text{ a.e.} && \text{because } f_k \in o(h);\\
&\leq cf_k(h(n)) \text{ a.e.} && \text{by (2);}\\
&< cf_k(f_j(h(n))) + 1 \text{ a.e.} && \text{since } f_k \text{ is nondecreasing;}\\
&\leq f_i(cf_k(f_j(h(n))) + 1) \text{ a.e.} && \text{since } f_i \text{ is nondecreasing.}
\end{aligned}$$

This is a contradiction. ■

Generally, Theorem 6.1 says that for a base $\mathcal{F}$ and functions g, g_0 satisfying (X), if $\mathbf{N}(\mathcal{F} \circ h) \neq \mathbf{D}(\mathcal{F} \circ h)$, for $h(n) = g^{-1}(g_0(2^{n/\alpha}))$, then $\mathbf{N}(\mathcal{F}) - \mathbf{D}(\mathcal{F})$ contains a g–sparse set. Theorem 7.2 says that if h is a function significantly greater than any function in $\mathcal{F}$, then $\mathcal{F} \circ h$ is not closed under composition. Hence, if $\mathcal{F} \in \Theta(\mathcal{P})$, there is no h such that $\mathcal{F} \circ h \in \Theta(\mathcal{EPL})$. Therefore Theorem 6.1 cannot connect the existence of a sparse set in $\mathbf{NP} - \mathbf{P}$ with the existence of any set in $\mathbf{N}(\mathcal{EPL}) - \mathbf{D}(\mathcal{EPL})$. Although this limitation is similar to the first one, it works on different techniques. Theorem 7.1 gives a limitation on relativisable techniques while Theorem 7.2 limits techniques (even nonrelativisable) where the higher base is $\mathcal{F} \circ h$ and the lower base is $\mathcal{F}$.

Another interesting limitation consists in the area of applications of the main theorems and Theorem 4.2. The following two theorems show the difference.

THEOREM 7.3. *Define*

$$\mathcal{F} = \left\{2^{i2^n}\right\}_{i>0}.$$

Then $\mathbf{NEXT} - \mathbf{DEXT}$ *contains a tally set iff* $\mathbf{N}(\mathcal{F}) \neq \mathbf{D}(\mathcal{F})$.

Proof. By Theorem 4.2, there is an exponentially padded set in $\mathbf{NEXT} - \mathbf{DEXT}$ iff $\mathbf{N}(\mathcal{F}) \neq \mathbf{D}(\mathcal{F})$. By Theorem 4.1, there is a tally set in $\mathbf{NEXT} - \mathbf{DEXT}$ iff $\mathbf{N}(\mathcal{F}) \neq \mathbf{D}(\mathcal{F})$. ■

THEOREM 7.4. *For every $j > 0$ define*

$$\mathcal{G}_j = \left\{2^{i2^{\lceil \frac{n}{j} \rceil}}\right\}.$$

Then **NEXT** $-$ **DEXT** *contains a $\mathcal{P}$–sparse set iff there exists some j such that* $\mathbf{N}(\mathcal{G}_j) \neq \mathbf{D}(\mathcal{G}_j)$.

Proof. Let $\alpha = 22$ and $g(n) = 2^{c\lceil \log n \rceil}$ for some $c \geq 1$. We will prove that if $\mathcal{G}_{\alpha c}$ collapses, then, by Theorem 6.1 there is no g–sparse set in **NEXT** $-$ **DEXT**. First of all, we verify (X):

Let $\mathcal{F} = \mathcal{EXT}$ and $g_0(n) = 2^{\lceil \log n \rceil + 1}$ and $g_1(n) = 2n$. Now $\mathcal{F}$ is closed under product. We also have $n + 1 \leq g(n) \leq 2^n$ a.e. and

$$g^{-1}(n) = \min\{t : 2^{c\lceil \log t \rceil} \geq n\} = \min\{t : \lceil \log t \rceil \geq (\log n)/c\} = 2^{\lceil \frac{\log n}{c} \rceil - 1} + 1.$$

Note that $g^{-1}(2^n)$ is time constructible. We have to verify that

(a) $g^{-1}(g_0(2^{n/\alpha}))$ is time constructible. This is true because

$$g^{-1}\big(g_0\big(2^{\frac{n}{\alpha}}\big)\big) = 2^{\lceil (\lceil n/\alpha \rceil + 1)/c \rceil - 1} + 1$$

and this is a time constructible function.

(b) $g_0(n) \geq 2n$ a.e. This is clear.

(c) $g^{-1}(g_0(n)) \leq g_1(g^{-1}(n))$ a.e. This is true because

$$g^{-1}(g_0(n)) = 2^{\lceil \frac{\lceil \log n \rceil + 1}{c} \rceil - 1} + 1 \leq 2^{\lceil \frac{\lceil \log n \rceil}{c} + \frac{1}{c} \rceil - 1} + 1 \leq 2^{\lceil \frac{\lceil \log n \rceil}{c} \rceil} + 1 \leq 2g^{-1}(n) = g_1(g^{-1}(n)).$$

(d) $\mathcal{F}$ is closed under right composition with g_1. This is true because $\mathcal{EXT}$ is closed under right composition with $\mathcal{LIN}$ and g_1 is a linear function.

Thus (X) is satisfied. By Theorem 6.1, if there is a g–sparse set in **NEXT** $-$ **DEXT**, then the base

$$\mathcal{G} = \mathcal{EXT} \circ g^{-1}(g_0(2^{n/\alpha})) = \left\{2^{i2^{\lceil (\lceil n/\alpha \rceil + 1)/c \rceil - 1} + 1}\right\} \in \Theta(\mathcal{G}_{\alpha c})$$

does not collapse. Thus if **NEXT** $-$ **DEXT** contains a $\mathcal{P}$–sparse set, then there exists a j such that $\mathcal{G}_j$ does not collapse.

Now, suppose that $\mathbf{N}(\mathcal{G}_j) \neq \mathbf{D}(\mathcal{G}_j)$ for some j. By Theorem 4.2, there is a $2^{\lceil n/j \rceil}$–padded set in **NEXT** $-$ **DEXT**. By Lemma 4.1, this set is polynomially sparse. ■

What is a difference between these two theorems? Let $\mathcal{F}$ be the base from Theorem 7.3 and $\mathcal{G}_j$ bases from Theorem 7.4. Note that for every j we have $\mathcal{G}_j \in O(\mathcal{F})$; hence the existence of a $\mathcal{P}$–sparse set in **NEXT** $-$ **DEXT** gives us a weaker result (a separation of a base less than $\mathcal{F}$) than the existence of a tally set in **NEXT** $-$ **DEXT**. Thus, today, known results differ on $\mathcal{P}$–sparse sets and tally sets.

COROLLARY 7.1. *There is a polynomially sparse set in* **NEXT** $-$ **DEXT** *if and only if* **NP** $-$ **P** *contains a set from SETS–K*[log, poly] *of polylogarithmic density.*

Proof. It follows directly from Theorems 1.1 and 7.4. ■

REFERENCES

1. Allender E. (1991), Limitations of the upward separation technique, *Math. Systems Theory* **24**, 53–67.
2. Allender, E., and Rubinstein, R. (1988), P-printable sets, *SIAM J. Comput.* **17**, 1193–1202.
3. Allender, E., Reinhardt, K., and Zhou S. (1999), Isolation, matching and counting: Uniform and nonuniform upper bounds, *J. Comput. System. Sci.* **59**, 164–181.
4. Allender, E., and Wilson, C. (1990), Downward translations of equality, *Theoret. Comput. Sci.* **75**, 335–346.
5. Book, R. (1974), Tally languages and complexity classes, *Inform. and Control* **26**, 186–193.
6. Buhrman, H., and Fortnow, L. (1997), Resource-bounded Kolmogorov complexity revisited, *in* "Proc. of the 14th Symposium on Theoretical Aspects of Computer Science," Lecture Notes in Computer Science, Vol. 1200, pp. 105–116, Springer-Verlag, Berlin/New York.
7. Buhrman, H., Longpré, L., and Spaan E. (1993), SPARSE reduces conjunctively to TALLY, *in* "Proc. of the 8th IEEE Conf. Structure in Complexity Theory," pp. 208–214.
8. Balcázar, J. L., Díaz, J., and Gabarró J. (1995), "Structural Complexity I," EATCS Monographs on Theoretical Computer Science, Vol. 11, Springer-Verlag, Berlin.

9. Balcázar, J. L., Díaz, J., and Gabarró, J. (1990), "Structural Complexity II," EATCS Monographs on Theoretical Computer Science, Vol. 22, Springer-Verlag, Berlin.
10. Fredman, M. L., Komlós, J., and Szemerédi, E. (1984), Storing a sparse table with O(1) worst case access time, *J. Assoc. Comput. Mach.* **31**, 538–544.
11. Glasnák, V. (1997), On f-sparse sets in NP-P, *in* "Proc. of the 24th SOFSEM," Lecture Notes in Computer Science, Vol. 1338, pp. 415–422, Springer-Verlag, Berlin.
12. Hartmanis, J. (1983), On sparse sets in NP-P, *Inform. Process. Lett.* **16**, 55–60.
13. Hartmanis, J., Immerman, N., and Sewelson, V. (1985), Sparse sets in NP-P: EXPTIME versus NEXPTIME, *Inform. and Control* **65**, 158–181.
14. Hartmanis, J., and Stearns, R. E. (1965), On the computational complexity of algorithms, *Trans. Amer. Math. Soc.* **117**, 285–306.
15. Hartmanis, J., and Yesha, Y. (1984), Computation times of NP sets of different densities, *Theoret. Comput. Sci.* **34**, 17–32.
16. Mehlhorn, K. (1982), On the program size of perfect and universal hash functions, *in* "Proc. of the 23rd Annual IEEE Symposium on Foundations of Computer Science," pp. 170–175.
17. Saluja, S. (1993), Relativized limitations of left set technique and closure classes of sparse sets, *in* "Proc. of the 8th IEEE Conf. Structure in Complexity Theory," pp. 215–222.
18. Schöning, U. (1993), On random reductions from sparse to tally sets, *Inform. Process. Lett.* **46**, 239–241.
19. Seiferas, J. I., Fischer, M. J., and Meyer, A. (1978), Separating nondeterministic time complexity classes, *J. Assoc. Comput. Mach.* **25**, 146–167.