Theoretical Computer Science 82 (1991) 51-69
Elsevier

On adaptive versus nonadaptive bounded query machines*

Ker-I Ko

Department of Computer Science, State University of New York at Stony Brook, Stony Brook, NY 11794, USA

Communicated by M.S. Paterson
Received August 1987
Revised January 1989

Abstract

Ko, K.-I., On adaptive versus nonadaptive bounded query machines, Theoretical Computer Science 82 (1991) 51-69.

The polynomial-time adaptive (Turing) and nonadaptive (truth-table) bounded query machines are compared with respect to sparse oracles. A k-query adaptive machine has been found which, relative to a sparse oracle, cannot be simulated by any (2^k-2)-query nonadaptive machine, even with a different sparse oracle. Conversely, there is a $(3\cdot 2^{k-2})$-query nonadaptive machine which, relative to a sparse oracle, cannot be simulated by any k-query adaptive machine, with any sparse oracle.

1. Introduction

The comparison of adaptive and nonadaptive computation has been investigated in many different forms. It is often observed that adaptive computation is more powerful than nonadaptive computation, if certain restrictions are put on the computational models. However, when we lift these restrictions, the comparison becomes more difficult. Consider, for instance, the simple example of binary versus linear search. Assume that we want to compute a function $f(n)$ which has the property that $1 \leq f(n) \leq n$. Also assume that an oracle is available to answer questions of the type "Is $f(n) < c$?" for any constant c. Then, an adaptive binary search can find $f(n)$ by making only $\lceil \log n \rceil$ queries but a nonadaptive search has to make $n-1$ queries to the oracle to find $f(n)$. However, if the oracle is more powerful and

* Research supported in part by the NSF Grants CCR-8696135 and CCR-8801575.

can answer questions of the type "Is $f(n) \in S$?" then a more clever nonadaptive search can also find $f(n)$ in $\lceil \log n \rceil$ queries. (For instance, if $n = 16$, then a nonadaptive search can be done by the following four queries: "$f(16) \in \{1, 2, \ldots, 8\}$?", "$f(16) \in \{1, \ldots, 4, 9, \ldots, 12\}$?", "$f(16) \in \{1, 2, 5, 6, 9, 10, 13, 14\}$?" and "$f(16) \in \{1, 3, 5, \ldots, 15\}$?".)

In complexity theory, adaptive computation using a query type device is formally modeled by polynomial-time Turing reducibility ($\leqslant_T^P$-reducibility) and its counterpart for nonadaptive computation is polynomial-time truth-table reducibility ($\leqslant_{tt}^P$-reducibility). There have been extensive studies on these reducibilities. We review some of the recent results. Ladner, Lynch and Selman [11] first separated $\leqslant_T^P$-reducibility from $\leqslant_{tt}^P$-reducibility in the simplest form: there exist sets A and B such that $A \leqslant_T^P B$ but $A \not\leqslant_{tt}^P B$. In addition, sets A and B are constructed to run in deterministic time $2^{O(n)}$ (i.e., $A, B \in \mathit{EXPTIME}$).

A more general comparison is to consider the reduction classes defined by sparse sets under various reducibilities. Let $r = \mathrm{T}$ or tt, and $\mathscr{C}$ a class of sets. Define $P_r(\mathscr{C})$ to be the class of sets A which are $\leqslant_r^P$-reducible to sets in $\mathscr{C}$. Let *TALLY* be the class of all tally sets and *SPARSE* the class of all sparse sets. The class $P_T(\mathit{SPARSE})$, which denotes the class of sets $\leqslant_T^P$-reducible to sparse sets, has played an important role in recent studies of structural complexity theory. Among other results, Karp and Lipton [8] provided an interesting characterization of $P_T(\mathit{SPARSE})$: it is equivalent to the class of all sets which are computable by circuits of polynomially many gates, or the class P/poly. Book and Ko [5] pointed out that this class is equivalent to the class of sets $\leqslant_{tt}^P$-reducible to sparse sets: $P/\mathit{poly} = P_T(\mathit{SPARSE}) = P_{tt}(\mathit{SPARSE})$. Moreover, they are also equivalent to the classes $P_T(\mathit{TALLY})$ and $P_{tt}(\mathit{TALLY})$. Therefore, $\leqslant_T^P$-reducibility is no more powerful than $\leqslant_{tt}^P$-reducibility as long as oracles are restricted to be sparse.

In a more recent paper, Tang and Book [12] considered the equivalence classes, instead of reduction classes, defined by sparse sets. For $r = \mathrm{T}$ or tt, and any class $\mathscr{C}$, let $E_r^P(\mathscr{C})$ be the class of sets A for which there exists a set $S \in \mathscr{C}$ such that $A \leqslant_r^P S$ and $S \leqslant_r^P A$. Tang and Book compared the classes $E_T^P(\mathit{TALLY})$ with $E_{tt}^P(\mathit{TALLY})$, and showed that $E_{tt}^P(\mathit{TALLY}) \subsetneq E_T^P(\mathit{TALLY})$. (The question of whether $E_{tt}^P(\mathit{SPARSE}) = E_T^P(\mathit{SPARSE})$ remained open.) This result, together with the result of Book and Ko [5], showed that the relative power of adaptive versus nonadaptive query machines depends very much on the specific formulation of the question.

In addition to the unbounded reducibilities $\leqslant_T^P$ and $\leqslant_{tt}^P$, the bounded versions of these reducibilities have also been studied extensively. Let r be either T or tt. For each integer $k \geqslant 1$, we say that A is $\leqslant_{k\text{-}r}^P$-reducible to B, and write $A \leqslant_{k\text{-}r}^P B$, if A is $\leqslant_r^P$-reducible to B by an r-type query machine which, on any input, only makes k queries. It is natural to compare the power of these reducibilities. First, for the simple separation results, Ladner et al. [11] have shown that there exist sets A and B such that $A \leqslant_{(k+1)\text{-}tt}^P B$ but $A \not\leqslant_{k\text{-}tt}^P B$. Thus, in nonadaptive oracle computation, asking more queries yields more power of computation. This idea is further explored by Amir and Gasarch [1], Beigel [2] and Goldsmith et al. [6], who considered the structure of sets relative to which a k-query machine is stronger than a $(k-1)$-query machine (called *terse* sets), and sets relative to which a 2^k-query machine can be replaced by a k-query machine (called *cheatable* sets).

Regarding the relationship between $\leqslant^{\mathrm{P}}_{k\text{-tt}}$-reducibility and $\leqslant^{\mathrm{P}}_{k\text{-T}}$-reducibility, it is clear that for each $k \geqslant 1$, a k-query adaptive machine can be simulated by a (2^k-1)-query nonadaptive machine. Therefore, we have, for all sets A and B,

$$A \leqslant^{\mathrm{P}}_{k\text{-tt}} B \Rightarrow A \leqslant^{\mathrm{P}}_{k\text{-T}} B \Rightarrow A \leqslant^{\mathrm{P}}_{(2^k-1)\text{-tt}} B.$$

By straightforward diagonalizations, we can show that the above relations are optimal. That is, for every $k \geqslant 1$, (a) there exist sets A and B such that $A \leqslant^{\mathrm{P}}_{k\text{-T}} B$ but $A \not\leqslant^{\mathrm{P}}_{(2^k-2)\text{-tt}} B$, and (b) there exist sets C and D such that $C \leqslant^{\mathrm{P}}_{(k+1)\text{-tt}} D$ but $C \not\leqslant^{\mathrm{P}}_{k\text{-T}} D$ (see Corollary 3.3 and Theorem 4.1). The result (b) above also implies that the intuition that asking more queries yields more computation power is true in adaptive oracle computation. (Similar results have been reported in [4] in the recursion-theoretic setting.)

In addition to the simple separation results, many researchers have studied reduction classes with respect to bounded reducibilities, in particular the reduction classes defined by sparse oracles and oracles in NP. For any complexity class $\mathscr{C}$, we let $P_{k\text{-r}}(\mathscr{C})$ be the class of sets which are $\leqslant^{\mathrm{P}}_{k\text{-r}}$-reducible to some set $B \in \mathscr{C}$. Köbler et al. [10] have shown that the classes $P_{k\text{-tt}}(\mathrm{NP})$, $k > 0$, define a truth-table hierarchy in Δ^{P}_2 which is nicely interwined with the boolean hierarchy in Δ^{P}_2. Kadin [7] showed that this hierarchy is a properly infinite hierarchy unless the Meyer–Stockmeyer polynomial-time hierarchy collapses. Beigel [3] has shown that for every $k \geqslant 1$, $P_{k\text{-T}}(\mathrm{NP}) = P_{(2^k-1)\text{-tt}}(\mathrm{NP})$.

For the reduction classes defined by sparse oracles, Book and Ko [5] showed that for each $k \geqslant 1$, $P_{k\text{-tt}}(\mathit{SPARSE}) \subsetneq P_{(k+1)\text{-tt}}(\mathit{SPARSE})$. In other words, there exists a set which is computable by a polynomial-time $(k+1)$-query nonadaptive machine relative to a sparse oracle but it is not computable, relative to *any* sparse set, by any polynomial-time k-query nonadaptive machine. This result indicates, in a stronger form, that $k+1$ queries provide more information than k queries in nonadaptive oracle computation. A natural question then asks whether this is true if the queries are made in an adaptive form, and, in an even stronger sense, whether this is true if $k+1$ queries are made in a nonadaptive form but k queries are allowed to be made in an adaptive form. These are the questions to be investigated in the current paper.

From the basic relations between $\leqslant^{\mathrm{P}}_{k\text{-tt}}$-reducibility and $\leqslant^{\mathrm{P}}_{k\text{-T}}$-reducibility mentioned above, it follows immediately that

$$P_{k\text{-tt}}(\mathit{SPARSE}) \subseteq P_{k\text{-T}}(\mathit{SPARSE}) \subseteq P_{(2^k-1)\text{-tt}}(\mathit{SPARSE}).$$

The main results of this paper concern with the questions of whether the above inclusive relations are proper. For the second inclusive relation, we show that it is optimal. That is, for every $k \geq 2$,

$$P_{k\text{-T}}(SPARSE) \not\subseteq P_{(2^k-2)\text{-tt}}(SPARSE).$$

This result immediately implies that for every $k \geq 1$,

$$P_{(k+1)\text{-T}}(SPARSE) \neq P_{k\text{-T}}(SPARSE).$$

For the first inclusive relation, we can only give a weak result that for every $k \geq 2$,

$$P_{l_k\text{-tt}}(SPARSE) \not\subseteq P_{k\text{-T}}(SPARSE),$$

where $l_k = 3 \cdot 2^{k-2}$. In other words, the relations between $P_{k\text{-T}}(SPARSE)$ and $P_{h\text{-tt}}(SPARSE)$ for $h, k > 1$ (they are obviously equal when $h = k = 1$) are as follows:

(1) $P_{k\text{-T}}(SPARSE) \subsetneq P_{h\text{-tt}}(SPARSE)$ for all $h \geq 2^k - 1$; and $P_{k\text{-T}}(SPARSE) \not\subseteq P_{h\text{-tt}}(SPARSE)$ for all $h \leq 2^k - 2$.

(2) $P_{h\text{-tt}}(SPARSE) \subsetneq P_{k\text{-T}}(SPARSE)$ for all $h \leq k$; and $P_{h\text{-tt}}(SPARSE) \not\subseteq P_{k\text{-T}}(SPARSE)$ for all $h \geq \frac{3}{4} \cdot 2^k$.

(3) $P_{h\text{-tt}}(SPARSE)$ and $P_{k\text{-T}}(SPARSE)$ are incomparable for all h and k such that $\frac{3}{4} \cdot 2^k \leq h \leq 2^k - 2$.

(4) Whether or not $P_{h\text{-tt}}(SPARSE)$ is a subclass of $P_{k\text{-T}}(SPARSE)$ is unknown for h such that $k < h < \frac{3}{4} \cdot 2^k$.

The proofs for these results are by diagonalization. The main technique is an inductive construction together with a careful use of the pigeonhole principle. This technique has been used in [5] and [9] to obtain similar separation results. All of the sets witnessing these separation results can be constructed to be in *EXPTIME*. The inductive construction for the result (2) involves a complicated case analysis. It seems that any big improvement over result (2) on the bound of $\frac{3}{4} \cdot 2^k$ would require a different proof technique.

Finally we remark that it has been proved in [5] that for every $k \geq 2$, $P_{k\text{-tt}}(TALLY) = P_{1\text{-tt}}(TALLY)$, because for any tally set T we can embed all necessary information about a $\leq^{\mathrm{P}}_{k\text{-tt}}$-reduction to set T in the following tally set $T' = \{0^{\langle i_1, \ldots, i_k, w_t \rangle} \mid w_t$ is a 2^k-bit string encoding a k-tt-condition t, and $t(\chi_T(0^{i_1}), \ldots, \chi_T(0^{i_k})) = 1\}$. Therefore, the classes $P_{k\text{-tt}}(TALLY)$ and $P_{k\text{-T}}(TALLY)$ are equivalent for all $k \geq 1$. In fact, they are all equivalent to $P_{1\text{-tt}}(TALLY)$.

2. Definitions

In this paper we will consider the alphabet $\Sigma = \{0, 1\}$. We denote by $|x|$ the length of a string x and by $\|X\|$ the cardinality of a set X. For a set X, χ_X denotes the characteristic function of X, and $\bar{X} = \Sigma^* - X$. The empty string is denoted by λ.

It is convenient to identify strings in Σ^* with nonnegative integers. More precisely, define a function $\iota : N \to \Sigma^*$ by $\iota(0) = \lambda$, $\iota(1) = 0$, $\iota(2) = 1, \ldots, \iota(n) =$ the nth string in Σ^* under the lexicographic order. In particular, for every $k \geq 1$, $\iota(2^k - 2) = 1^{k-1}$. We will often drop the function name ι^{-1} and write s to denote the number $\iota^{-1}(s)$, and $s+1$ to denote either the number $\iota^{-1}(s)+1$ or the string $\iota(\iota^{-1}(s)+1)$.

We assume that the reader is familiar with Turing machines, oracle Turing machines and their time complexity. In the following, we define polynomial-time k-tt-reducibility and k-T-reducibility. For any integer $k \geq 1$, we say that set A is polynomial-time k-tt-reducible to set B, and write $A \leq^{\mathrm{P}}_{k\text{-tt}} B$, if there exist polynomial-time computable functions f and g such that for all x, $f(x)$ is a list of k strings, $g(x)$ is a truth-table with k variables (i.e., a table specifying a boolean function on k inputs), and $x \in A$ iff the truth-table $g(x)$ evaluates to **true** on the k-tuple $\langle \chi_B(x_1), \ldots, \chi_B(x_k) \rangle$ where $f(x) = \langle x_1, \ldots, x_k \rangle$.

To describe $\leq^{\mathrm{P}}_{k\text{-T}}$-reducibility, we first establish some new notation on binary trees of height k. Let T be a complete binary tree of height k. For each $s \in \Sigma^*$ of length $\leq k-1$, we write $T(s)$ to denote the sth (or, more precisely, the $\iota^{-1}(s)$th) node under the breadth-first ordering. In other words, $T(\lambda)$ is the root, and for each s of length $k-1$, $T(s)$ is a leaf, and for each s of length $< k-1$, $T(s)$ has two children: left child $T(s0)$ and right child $T(s1)$.

For any $k \geq 1$, we say a function f on input x generates a tree of height k if $f(x)$ is a list of 2^{k-1} strings $\langle x_\lambda, x_0, x_1, x_{00}, \ldots, x_{1^{k-1}} \rangle$. We will interpret the output of $f(x)$ as a tree T_x of height k such that $T_x(s) = x_s$ for all s of length $\leq k-1$. For any integer $k \geq 1$, we say that set A is polynomial-time k-T-reducible to set B, and write $A \leq^{\mathrm{P}}_{k\text{-T}} B$, if there exist polynomial-time computable functions f and g such that for all x, $f(x)$ generates a tree of height k, $g(x)$ is a truth-table with k variables, and $x \in A$ iff the truth-table $g(x)$ evaluates to **true** on the k-tuple $\langle \chi_B(x_\lambda), \chi_B(x_{s_1}), \ldots, \chi_B(x_{s_1 \ldots s_{k-1}}) \rangle$, where each bit s_i, $1 \leq i \leq k-1$, is defined by $s_1 = \chi_B(x_\lambda)$ and $s_i = \chi_B(x_{s_1 \ldots s_{i-1}})$, for $i > 1$.

For any complexity class $\mathscr{C}$, we let $P_{k\text{-r}}(\mathscr{C}) = \{A \mid \text{there exists a set } C \in \mathscr{C} \text{ such that } A \leq^{\mathrm{P}}_{k\text{-r}} C\}$, where r is either tt or T. Recall that a set S is *sparse* if there is a polynomial q such that for all n, $\|\{x \in S \mid |x| \leq n\}\| \leq q(n)$. Let *SPARSE* denote the class of all sparse sets. We assume a fixed enumeration $\{p_h\}_{h=1}^{\infty}$ of polynomials with nonnegative integer coefficients. For each $h > 0$, let $SPARSE_h$ denote the class of sets S such that for all n, $\|\{x \in S \mid |x| \leq n\}\| \leq p_h(n)$. A set A is a *tally* set if $A \subseteq \{0\}^*$. Let *TALLY* denote the class of all tally sets.

In the proof of the main results, we will need to enumerate all $\leq^{\mathrm{P}}_{k\text{-tt}}$-reduction machines and all $\leq^{\mathrm{P}}_{k\text{-T}}$-reduction machines. Let $\{f_i^{(k)}\}$ be an enumeration of all polynomial-time computable functions that for each input x yield a list of k strings, and $\{g_i^{(k)}\}$ be an enumeration of all polynomial-time computable k-tt-truth-tables. Then, we can enumerate all $\leq^{\mathrm{P}}_{k\text{-tt}}$-reduction machines as $\{M_{i,j}^{(k)}\}$, where each $M_{i,j}^{(k)}$ is defined by the ith k-tt-condition generator $f_i^{(k)}$ and the jth k-tt-condition evaluator $g_j^{(k)}$ as described above. We write $L(M_{i,j}^{(k)}, A)$ to denote the set of strings accepted by $M_{i,j}^{(k)}$ relative to oracle A. We can also enumerate all $\leq^{\mathrm{P}}_{k\text{-T}}$-reduction machines as $\{N_{i,j}^{(k)}\}$, where each $N_{i,j}^{(k)}$ is defined by the ith k-T-tree generator (i.e., the ith (2^{k-1})-tt-condition generator) $f_i^{(2^{k-1})}$ and the k-tt-condition evaluator $g_j^{(k)}$ as described above. We write $L(N_{i,j}^{(k)}, A)$ to denote the set of strings accepted by $N_{i,j}^{(k)}$ relative to oracle A.

3. A strong adaptive bounded query machine

In this section, we show that there exists an adaptive query machine which makes only k queries to a sparse oracle A such that the set computed by this machine relative to set A is not computable, relative to any sparse oracle S, by any nonadaptive query machine which makes at most 2^k-2 queries. This result is optimal, as it is easy to see that a (2^k-1)-query nonadaptive machine can simulate a k-query adaptive machine.

Theorem 3.1. *For every $k \geqslant 2$, $P_{k\text{-T}}(SPARSE) \not\subseteq P_{m_k\text{-tt}}(SPARSE)$, where $m_k = 2^k-2$.*

Proof. We let $k \geqslant 2$ be fixed and let $m_k = 2^k-2$. Recall that a k-query adaptive machine is defined by two polynomial-time functions f and g, where for each x, $f(x)$ generates a binary tree of height k and $g(x)$ outputs a k-tt-condition. We define a specific k-query adaptive machine N as follows. For any string x of length $m_k n$ for some $n \geqslant 1$, write x as $u_0u_1u_{00}\ldots u_{11}\ldots u_{1^{k-1}}$, where each u_s, $1 \leqslant |s| \leqslant k-1$, is of length n. The machine N on input x produces a tree T_x of height k, in which each node $T_x(s)$, $1 \leqslant |s| \leqslant k-1$, is attached with the string su_s, and the node $T_x(\lambda)$ is attached with the string 0^n. We write $T_x(s) = su_s$ and $T_x(\lambda) = 0^n$. The machine N on input x evaluates the tree T_x, relative to an oracle B, as follows: first N recursively computes the sequence $\langle s_1, s_2, \ldots, s_k\rangle$ by $s_1 = \chi_B(T_x(\lambda)) = \chi_B(0^n)$, and $s_{i+1} = \chi_B(T_x(s_1\ldots s_i)) = \chi_B(s_1\ldots s_i u_{s_1\ldots s_i})$, $1 \leqslant i \leqslant k-1$; and $N^B(x)$ accepts iff the number of 1's in the sequence $\langle s_1, \ldots, s_k\rangle$ is odd. In other words, if N queries oracle B about string $T_x(s)$ and receives an answer "no" then the next query is $T_x(s0)$, otherwise the next query is $T_x(s1)$; and N accepts x iff the number of "yes" answers is odd.

For each set A, let $L_k(A)$ be the set of all strings of length $m_k n$ for some n which are accepted by machine N relative to set A. It is obvious that $L_k(A) \leqslant_{k\text{-T}}^{\text{P}} A$. We will construct a sparse set A such that for all sparse sets S, it is not the case that $L_k(A) \leqslant_{m_k\text{-tt}}^{\text{P}} S$. This will allow us to conclude that $P_{k\text{-T}}(SPARSE) \not\subseteq P_{m_k\text{-tt}}(SPARSE)$.

The construction is done by stages. For each pair $\langle i, j\rangle$, let $M_{i,j} = M_{i,j}^{(m_k)}$. Recall that $\{M_{i,j}\}$ is an enumeration of all $\leqslant_{m_k\text{-tt}}^{\text{P}}$-reduction machines, where each $M_{i,j}$ is defined by the ith polynomial-time m_k-tt-condition generator $f_i^{(m_k)}$ and the jth polynomial-time m_k-tt-condition evaluator $g_j^{(m_k)}$. Also recall that $\{p_h\}$ is an enumeration of polynomial functions. At stage $e = \langle i, j, h\rangle$, we will find an integer $n = n_e$ and construct A_e such that $L_k(A_e) \cap \Sigma^{m_k n} \neq L(M_{i,j}, S) \cap \Sigma^{m_k n}$, for all $S \in SPARSE_h$.

Prior to stage 1, we let $\delta = 1/(4m_k)$, $A_0 = \emptyset$ and $n_0 = 1$.

Stage $e = \langle i, j, h\rangle$. Let n be an integer satisfying $n > m_k n_{e-1}$ (so that the conditions established in earlier stages are not affected by the construction of the current stage), $\delta n > 2^{m_k}$ and $m_k \cdot p_h(q(m_k n)) < 2^{\delta n}$, where q is a polynomial bounding the runtime of machine $M_{i,j}$ (so that the number of strings in $S \in SPARSE_h$ which are queried by $M_{i,j}$ on some string of length $m_k n$ is bounded by $2^{\delta n}/m_k$). Let $n_e = n$. For each x, assume that $f_i^{(m_k)}(x) = \langle x_1, \ldots, x_{m_k}\rangle$. Consider the m_k-tt-conditions $g_j^{(m_k)}(x)$, for all $x \in \Sigma^{m_k n}$. Since there are only $2^{2^{m_k}}$ different m_k-tt-conditions, there exists an m_k-tt-condition t such that $G_t = \{x \in \Sigma^{m_k n} \mid g_j^{(m_k)}(x) = t\}$ has size $\geqslant 2^{m_k n - 2^{m_k}}$. Fix such an m_k-tt-condition t and let $G = G_t$. Note that $\|G\| \geqslant 2^{(m_k - \delta)n}$. Now we consider two cases.

Case 1. The function $f_i^{(m_k)}$ is not one-to-one on G.

Let x and y be two strings in G such that $f_i^{(m_k)}(x) = f_i^{(m_k)}(y)$. Write $x = u_0 u_1 u_{00} \ldots u_{1^{k-1}}$ and $y = v_0 v_1 v_{00} \ldots v_{1^{k-1}}$, with each u_s and each v_s of length n. Then, $x \neq y$ implies that there exists $s \leqslant 1^{k-1}$ such that $u_s \neq v_s$. Let s be the smallest such index s, and assume that $s = s_1 s_2 \ldots s_l$, where each s_r is a bit 0 or 1. We assign $T_x(\lambda) = 0^n$ to B iff $s_1 = 1$, and $T_x(s_1 \ldots s_r) = s_1 \ldots s_r u_{s_1 \ldots s_r}$ to B iff $s_{r+1} = 1$, for all r such that $1 \leqslant r \leqslant l-1$. Then we assign $T_x(s) = s u_s$ to B (and leave $T_y(s) = s v_s$ to $\bar{B}$). Note that y agrees with x on the first $(s-1)n$ bits, and so the computation of $M^B(y)$ is exactly the same as that of $M^B(x)$ until it asks the query $T_y(s)$, for which it receives a different answer from the query $T_x(s)$. (Note that we have assigned values to B in such a way that $T_y(s)$ will be the $(l+1)$st query to B.) Since all later queries $T_x(s')$ and $T_y(s'')$ with $s' > s$ and $s'' > s$ receive answer "no", we have $x \in L_k(B) \Leftrightarrow y \notin L_k(B)$. However, $f_i^{(m_k)}(x) = f_i^{(m_k)}(y)$ implies that for any set S, x and y are both in $L(M_{i,j}, S)$ or both in $\overline{L(M_{i,j}, S)}$. Thus, either $z = x$ or $z = y$ satisfies the requirement $z \in L_k(B) \Leftrightarrow z \notin L(M_{i,j}, S)$. Let $A_e = A_{e-1} \cup B$.

Case 2. The function $f_i^{(m_k)}$ is one-to-one on G.

We consider two more subcases.

Subcase 2.1. There exist an integer r, $1 \leqslant r \leqslant m_k$, and a string z, such that the set $H_r(z) = \{x \in G \mid x_r = z\}$ has size $\|H_r(z)\| \geqslant 2^{(m_k - 2\delta)n}$.

Then, there exist two strings $v, w \in \Sigma^n$ such that $\|H_r(z) \cap \{v\}\Sigma^{(m_k-1)n}\| \geqslant 2^{(m_k - 1 - 3\delta)n}$ and $\|H_r(z) \cap \{w\}\Sigma^{(m_k-1)n}\| \geqslant 2^{(m_k-1-3\delta)n}$. (Let v be the string in Σ^n such that $\|H_r(z) \cap \{v\}\Sigma^{(m_k-1)n}\|$ is maximized, and w be the string such that $\|H_r(z) \cap \{w\}\Sigma^{(m_k-1)n}\|$ is the second largest. Then,

$$\|H_r(z) \cap \{v\}\Sigma^{(m_k-1)n}\| \geqslant \|H_r(z)\| \cdot 2^{-n} = 2^{(m_k-1-2\delta)n},$$

and

$$\|H_r(z) \cap \{w\}\Sigma^{(m_k-1)n}\| \geqslant \|H_r(z) - \{v\}\Sigma^{(m_k-1)n}\| \cdot 2^{-n}$$
$$\geqslant (2^{(m_k-2\delta)n} - 2^{(m_k-1)n}) \cdot 2^{-n} \geqslant 2^{(m_k-1-3\delta)n}.)$$

We will prove the theorem in this case by induction. The induction statement is a little stronger than the intended result. We state it in a separate lemma as follows.

Lemma 3.2. *The following holds for all j, $1 \leq j \leq m_k - 1$. Assume that there exist a function f which yields, on input x, a list of j strings $\langle x_1, \ldots, x_j\rangle$, and a set $G \subseteq \{v_1 \ldots v_{m_k-j-1}\}\Sigma^{(j+1)n}$ for some strings $v_1, \ldots, v_{m_k-j-1} \in \Sigma^n$, satisfying the following properties*:

(1) *f is one-to-one on G, and for all $x \in G$ and all r, $1 \leq r \leq j$, $|x_r| \leq q(m_k n)$; and*

(2) *there exist strings v_{m_k-j} and w_{m_k-j} such that $\|G \cap V_j\| \geq 2^{(j-(2m_k-2j+1)\delta)n}$ and $\|G \cap W_j\| \geq 2^{(j-(2m_k-2j+1)\delta)n}$, where $V_j = \{v_1 \ldots v_{m_k-j-1} v_{m_k-j}\}\Sigma^{jn}$ and $W_j = \{v_1 \ldots v_{m_k-j-1} w_{m_k-j}\}\Sigma^{jn}$.*

Then, there exists a set C such that

(3) *for each i, $n \leq i \leq n+k-1$, $\|C \cap \Sigma^i\| \leq 1$, and for each i, $i < n$ or $n+k \leq i$, $C \cap \Sigma^i = \emptyset$, and*

(4) *for all $S \in SPARSE_h$, and all j-tt-conditions t, there exists an $x \in G$ such that $x \in L_k(C) \Leftrightarrow t(\chi_S(x_1), \ldots, \chi_S(x_j)) = 0$.*

To apply Lemma 3.2 to Subcase 2.1, let $j = m_k - 1$ and $f(x) = \langle x_1, \ldots, x_{r-1}, x_{r+1}, \ldots, x_{m_k}\rangle$. Then, the function f and set $H_r(z)$ satisfy the assumptions of Lemma 3.2. (Note that $x_r = z$ for all $x \in H_r(z)$, and therefore, f must be one-to-one on $H_r(z)$; otherwise, $f_i^{(m_k)}$ would not be one-to-one on G.) So, we obtain from Lemma 3.2 a set C satisfying conditions (3) and (4). We verify that for any $S \in SPARSE_h$ and any m_k-tt-condition t, there exists a string $x \in G$ such that $x \in L_k(C) \Leftrightarrow x \notin L(M_{i,j}, S)$. Let t_0 and t_1 be two $(m_k - 1)$-tt-conditions defined as follows: $t_b(b_1, \ldots, b_{r-1}, b_{r+1}, \ldots, b_{m_k}) = t(b_1, \ldots, b_{r-1}, b, b_{r+1}, \ldots, b_{m_k})$. Then, let $b = \chi_S(z)$, and consider t_b. The set C satisfies condition (4) with respect to set S and $(m_k - 1)$-tt-condition t_b. It means that there exists a string $x \in H_r(z)$ such that $x \in L_k(C) \Leftrightarrow t_b(\chi_S(x_1), \ldots, \chi_S(x_{r-1}), \chi_S(x_{r+1}), \ldots, \chi_S(x_{m_k})) = 0$. By the definition of t_b, we have $x \in L_k(C) \Leftrightarrow t(\chi_S(x_1), \ldots, \chi_S(x_{m_k})) = 0 \Leftrightarrow x \notin L(M_{i,j}, S)$.

Let $A_e = A_{e-1} \cup C$.

Subcase 2.2. The condition specifying Subcase 2.1 does not hold.

Then, for every set $S \in SPARSE_h$, there are at most

$$\sum_{r=1}^{m_k} \sum_{z \in S} \|H_r(z)\| < m_k \cdot p_h(q(m_k n)) \cdot 2^{(m_k-2\delta)n} < 2^{\delta n} \cdot 2^{(m_k-2\delta)n} = 2^{(m_k-\delta)n}$$

strings in G having at least one x_r in S. Therefore, at least one x in G has all x_r in $\bar{S}$. Let 0^n be in set B iff $t(0, \ldots, 0) = 0$. Then, for any sparse set $S \in SPARSE_h$, at least one $x \in G$ has the property that $\chi_S(x_1) = \chi_S(x_2) = \cdots = \chi_S(x_{m_k}) = 0$, and so $x \in L_k(B) \Leftrightarrow t(\chi_S(x_1), \ldots, \chi_S(x_{m_k})) = 0 \Leftrightarrow x \notin L(M_{i,j}, S)$. Let $A_e = A_{e-1} \cup B$. This completes stage e of the construction.

By the above discussion, it is clear that set $A = \bigcup_{e=1}^{\infty} A_e$ satisfies our requirement. It remains to prove Lemma 3.2.

Proof of Lemma 3.2. The lemma will be proved by induction on $j = 1, \ldots, m_k - 1$. First assume that $j = 1$. Then $f(x)$ outputs only one string x_1. For any $S \in SPARSE_h$, since f is one-to-one on G, there are at most $p_h(q(m_k n)) < 2^{\delta n}$ strings x in G having

$x_1 \in S$. Therefore, there exist strings $x \in G \cap V_1$ and $y \in G \cap W_1$ such that $x_1 \notin S$ and $y_1 \notin S$. Hence, for any 1-tt-condition t, we have $t(\chi_S(x_1)) = t(0) = t(\chi_S(y_1))$.

Recall that a string in V_1 must have the form $v_1 \ldots v_{m_k-2} v_{m_k-1} v'$ for some $v' \in \Sigma^n$, and a string in W_1 must have the form $v_1 \ldots v_{m_k-2} w_{m_k-1} w'$ for some $w' \in \Sigma^n$. For convenience, let us rename each string v_l as $v_{\iota(l)}$ (and hence a string in V_1 has the form $v_0 v_1 v_{00} \ldots v_{1^{k-2}} v_{1^{k-2}0} v'$ for some $v' \in \Sigma^n$). We assign strings 0^n, $1v_1$, $11v_{11}, \ldots, 1^{k-3}v_{1^{k-3}}$ and $1^{k-2}0v_{1^{k-2}0}$ to set C (but let $1^{k-2}v_{1^{k-2}}$ and $1^{k-2}0w_{1^{k-2}0}$ be in $\bar{C}$). The above assignments force $G \cap V_1 \subseteq L_k(C)$ and $G \cap W_1 \subseteq \overline{L_k(C)}$ if k is even, and $G \cap V_1 \subseteq \overline{L_k(C)}$ and $G \cap W_1 \subseteq L_k(C)$ if k is odd. Therefore, $x \in L_k(C) \Leftrightarrow y \notin L_k(C)$. Together with the property that for every $S \in SPARSE_h$ and every 1-tt-condition t, $t(\chi_S(x_1)) = t(\chi_S(y_1))$, we see that either x or y is a witness for condition (4). This completes the proof for the initial case of $j = 1$.

For the inductive step, let $j > 1$. We consider two cases. (These two cases are similar to Subcases 2.1 and 2.2 in the proof of Theorem 3.1.).

Case 1. There exist an integer r, $1 \leqslant r \leqslant j$, and a string z such that the set $H_r(z) = \{x \in G \mid x_r = z\}$ satisfying $\|H_r(z) \cap V_j\| \geqslant 2^{(j-(2m_k-2j+2)\delta)n}$ or $\|H_r(z) \cap W_j\| \geqslant 2^{(j-(2m_k-2j+2)\delta)n}$.

Without loss of generality, assume that $\|H_r(z) \cap V_j\| \geqslant 2^{(j-(2m_k-2j+2)\delta)n}$. Let $f'(x) = \langle x_1, \ldots, x_{r-1}, x_{r+1}, \ldots, x_j \rangle$ and $G' = H_r(z) \cap V_j$. Then, similarly to Subcase 1.1 in the proof of Theorem 3.1, there exist two strings $v_{m_k-j+1}, w_{m_k-j+1} \in \Sigma^n$ such that $\|G \cap V_{j-1}\| \geqslant 2^{(j-1-(2m_k-2j+3)\delta)n}$ and $\|G \cap W_{j-1}\| \geqslant 2^{(j-1-(2m_k-2j+3)\delta)n}$, where $V_{j-1} = \{v_1 \ldots v_{m_k-j} v_{m_k-j+1}\}\Sigma^{(j-1)n}$ and $W_{j-1} = \{v_1 \ldots v_{m_k-j} w_{m_k-j+1}\}\Sigma^{(j-1)n}$.

Thus, function f' and G' satisfy the assumptions of the inductive hypothesis. By induction, we can find a set C satisfying conditions (3) and (4) (with parameter $j-1$).

Now consider a j-tt-condition t. We note that $x_r = z$ for all $x \in H_r(z)$, and so for any set S, $t(\chi_S(x_1), \ldots, \chi_S(x_j))$ is equal to $t'(\chi_S(x_1), \ldots, \chi_S(x_{r-1}), \chi_S(x_{r-1}), \ldots, \chi_S(x_j))$, where t' is defined by $t'(b_1, \ldots, b_{r-1}, b_{r+1}, \ldots, b_j) = t(b_1, \ldots, b_{r-1}, \chi_S(z), b_{r+1}, \ldots, b_j)$. By condition (4), for any $S \in SPARSE_h$ and any j-tt-condition t, there exists an $x \in H_r(z) \cap V_j$ such that

$$x \in L_k(C) \Leftrightarrow t'(\chi_S(x_1), \ldots, \chi_S(x_{r-1}), \chi_S(x_{r+1}), \ldots, \chi_S(x_j)) = 0$$

$$\Leftrightarrow t(\chi_S(x_1), \ldots, \chi_S(x_j)) = 0.$$

This completes the proof for Case 1.

Case 2. The condition specifying Case 1 does not hold.

For every string z and every r, $1 \leqslant r \leqslant j$, $\|H_r(z) \cap V_j\| < 2^{(j-(2m_k-2j+2)\delta)n}$. Therefore, for any $S \in SPARSE_h$, there are at most

$$\sum_{r=1}^{j} \sum_{z \in S} \|H_r(z) \cap V_j\| < 2^{(j-(2m_k-2j+2)\delta)n} \cdot 2^{\delta n} = 2^{(j-(2m_k-2j+1)\delta)n}$$

strings x in V_j having at least one x_r in S. It follows that there exists at least one $x \in G \cap V_j$ having $t(\chi_S(x_1), \ldots, \chi_S(x_j)) = t(0, \ldots, 0)$ for any j-tt-condition t. Similarly, there exists a string $y \in G \cap W_j$ having $t(\chi_S(y_1), \ldots, \chi_S(y_j)) = t(0, \ldots, 0)$.

Recall that $V_j = \{v_1 \ldots v_{m_k - j}\}\Sigma^{jn}$. Let $s = \iota(m_k - j)$ and write $s = s_1 s_2 \ldots s_l$, where each s_i is a bit 0 or 1. We assign 0^n to C iff $s_1 = 1$, and $s_1 \ldots s_i v_{s_1 \ldots s_i}$ to C iff $s_{i+1} = 1$, for $i = 1, \ldots, l-1$. Finally, we assign $s_1 \ldots s_l v_{s_1 \ldots s_l}$ to C (but $s_1 \ldots s_l w_{s_1 \ldots s_l}$ to $\bar{C}$). It follows that $V_j \subseteq L_k(C)$ iff $W_j \subseteq \overline{L_k(C)}$ (both are true if s contains an even number of 1's, and both are false if s contains an odd number of 1's).

Now, for any $S \in SPARSE_h$ and any j-tt-condition t, consider strings $x \in G \cap V_j$ and $y \in G \cap W_j$ with $t(\chi_S(x_1), \ldots, \chi_S(x_j)) = t(0, \ldots, 0) = t(\chi_S(y_1), \ldots, \chi_S(y_j))$. We have constructed set C such that $x \in L_k(C) \Leftrightarrow y \notin L_k(C)$. Thus either x or y is a witness for condition (4). This completes the proof of Lemma 3.2. □

Corollary 3.3. *For every $k \geq 2$, there exist sets A and B such that $A \leq^{\mathrm{P}}_{k\text{-T}} B$ but $A \not\leq^{\mathrm{P}}_{m_k\text{-tt}} B$, where $m_k = 2^k - 2$.*

Corollary 3.4. *For every $k \geq 1$, $P_{(k+1)\text{-T}}(SPARSE) \not\subseteq P_{k\text{-T}}(SPARSE)$.*

4. A strong nonadaptive bounded query machine

To study the ability of adaptive bounded query machines to simulate nonadaptive bounded query machines, we first prove that there exist sets A and B such that A is $\leq^{\mathrm{P}}_{\mathrm{tt}}$-reducible to B by a $(k+1)$-query nonadaptive machine but is not $\leq^{\mathrm{P}}_{\mathrm{T}}$-reducible to B by any k-query adaptive machine.

Theorem 4.1. *For any $k \geq 1$, there exist sets A and B such that $A \leq^{\mathrm{P}}_{(k+1)\text{-tt}} B$ but $A \not\leq^{\mathrm{P}}_{k\text{-T}} B$.*

Proof. Let $k \geq 1$ be fixed, and for each pair $\langle i, j\rangle$, let $N_{i,j} = N^{(k)}_{i,j}$. For any set B, let $L(B) = \{0^n \mid$ the number of strings in the following list that are in B is odd: $\langle 0^n, 0^n1, 0^n1^2, \ldots, 0^n1^k\rangle\}$. Then, it is obvious that $L(B) \leq^{\mathrm{P}}_{(k+1)\text{-tt}} B$. We will construct a set B such that for every $\leq^{\mathrm{P}}_{k\text{-T}}$-reduction machine $N_{i,j}$, there exists an integer n such that $0^n \in L(B) \Leftrightarrow 0^n \notin L(N_{i,j}, B)$. This will allow us to conclude that $L(B) \not\leq^{\mathrm{P}}_{k\text{-T}} B$.

We construct set B by stages. Assume that the runtime of machine $N_{i,j}$ is bounded by a polynomial q_e, where $e = \langle i, j\rangle$. Let $B_0 = \emptyset$, and $n_0 = 1$.

In stage $e = \langle i, j\rangle$, we choose an integer n such that $n > q_{e-1}(n_{e-1})$ and $n > kn_{e-1}$ (so that the conditions established in earlier stages are not affected by the construction of the current stage). Let $n_e = n$. Then, we simulate $N_{i,j}$ on input 0^n using oracle B_{e-1}. Note that among the $k+1$ strings $0^n, 0^n1, \ldots, 0^n1^k$, there are at most k strings being queried in this computation. Let 0^n1^i be the shortest string in this list which is not queried, and let $B_e = B_{e-1} \cup \{0^n1^i\}$ if the computation of $N^{B_{e-1}}_{i,j}(0^n)$ rejects, and let $B_e = B_{e-1}$ if the computation of $N^{B_{e-1}}_{i,j}(0^n)$ accepts. Since 0^n1^i is not queried by $N^{B_{e-1}}_{i,j}$ on 0^n, the computation of $N^{B_e}_{i,j}(0^n)$ is the same as that of $N^{B_{e-1}}_{i,j}(0^n)$. It follows that $N^{B_e}_{i,j}(0^n)$ accepts iff $B_e \cap \{0^n, 0^n1, \ldots, 0^n1^k\} = \emptyset$ iff $0^n \notin L(B_e)$.

Let $B = \bigcup_{e=1}^{\infty} B_e$. By the choice of the integer n_e, it is clear that $L(B) \neq L(N_{i,j}, B)$ for all $\leq_{k\text{-T}}^{\text{P}}$-reduction machine $N_{i,j}$. □

Next we compare the reduction classes defined by sparse sets. For every integer $k \geq 2$, let $l_k = 3 \cdot 2^{k-2}$. We will prove that there exist a polynomial-time nonadaptive query machine which makes l_k queries to oracles, and a sparse oracle A, so that the set computed by this machine relative to A is not computable, relative to any sparse oracle, by any polynomial-time adaptive query machine which makes only k queries.

Theorem 4.2. *For any $k \geq 2$, $P_{l_k\text{-tt}}(SPARSE) \not\subseteq P_{k\text{-T}}(SPARSE)$.*

Proof. For each set A and each $m \geq 1$, let

$$L_{m+1}(A) = \{u_1 \ldots u_m \,|\, |u_1| = \cdots = |u_m| = n;\ \text{the number of strings in the following list that are in } A \text{ is odd: } 0^{m(n+1)}, u_1 0^{(m-1)(n+1)}, \ldots, u_1 \ldots u_m\}.$$

Then it is clear that $L_m(A) \leq_{m\text{-tt}}^{\text{P}} A$. For each $k \geq 2$, we will construct a sparse set A such that for every sparse set S, it is not the case that $L_{l_k}(A) \leq_{k\text{-T}}^{\text{P}} S$. This will prove the theorem.

The proof proceeds by stages. Recall that the $\langle i, j\rangle$th $\leq_{k\text{-T}}^{\text{P}}$-reduction machine $\{N_{i,j}^{(k)}\}$ is defined by the ith polynomial-time height-k tree generator $f_i^{(2^k-1)}$ and the jth polynomial-time k-tt-condition evaluator $g_j^{(k)}$. We fix the integer $k \geq 2$, and let $N_{i,j} = N_{i,j}^{(k)}$. In stage $\langle i, j, h\rangle$, we will find an integer n such that $L_{l_k}(A) \cap \Sigma^{(l_k-1)n} \neq L(N_{i,j}, S) \cap \Sigma^{(l_k-1)n}$ for all sparse sets $S \in SPARSE_h$. We choose a constant $\delta = 1/(8l_k)$, and let $A_0 = \emptyset$, $n_0 = 1$.

Stage $e = \langle i, j, h\rangle$. Assume that the runtime of the machine $N_{i,j}$ is bounded by a polynomial p'. We choose an integer n such that $n > l_k n_{e-1}$ (so that the conditions established in earlier stages are not affected by the construction in the current stage), $\delta n > 2l_k$, and $p_h(p'((l_k - 1)n)) < 2^{\delta n}$ (so that the number of strings y in $S \in SPARSE_h$ which are queried by $N_{i,j}$ on some input of length $l_k n$ is at most $2^{\delta n}$). Let $n_e = n$.

Consider the k-tt-conditions $g_j^{(k)}(x)$ for all $x \in \Sigma^{(l_k-1)n}$. There are only 2^{2^k} different k-tt-conditions. So there must be at least one k-tt-condition t such that $G_t = \{x \in \Sigma^{(l_k-1)n} \,|\, g_j^{(k)}(x) = t\}$ has size $\geq 2^{(l_k-1)n-2^k} \geq 2^{(l_k-1-\delta)n}$. We fix such a k-tt-condition t, and let $G = G_t$. Next, we will construct set A_e by an inductive construction which involves some complicated case analysis. We state this inductive construction in a separate lemma.

For each $j \geq 2$, define β_j recursively as follows: $\beta_2 = 8$, $\beta_{j+1} = 2\beta_j + 3$; that is, $\beta_j = 11 \cdot 2^{j-2} - 3$. Note that $\delta = 1/(8l_k) < 1/(2\beta_k)$.

Lemma 4.3. *The following holds for all j, $2 \leq j \leq k$. Assume that there exist a $\leq_{j\text{-T}}^{\text{P}}$-reduction machine $N_{f,t}$, defined by a height-j tree generator f and a fixed j-tt-condition t, and a set $G \subseteq \{u_1 \ldots u_m\}\Sigma^{(l_k-m-1)n}$ for some $u_1, \ldots, u_m$ in Σ^n $(0 \leq m \leq l_k - 1)$, such that*

(1) *the runtime of machine $N_{f,t}$ is bounded by polynomial p', and*

(2) $\|G\| \geq 2^{(l_j+\gamma-1-\alpha\delta)n}$ *for some* $\gamma \geq 0$ *and some* α, $0 \leq \alpha \leq \beta_k - \beta_j + 1$.
Then there exists a set C such that

(3) *strings in C are of the form* $u_1 \ldots u_m v_{m+1} \ldots v_{m+i} 0^{(l_k-m-i-1)(n+1)}$, *where* $0 \leq i \leq l_k - m - 1$ *and* $v_{m+1}, \ldots, v_{m+i} \in \Sigma^n$,

(4) $\|C\| \leq 2^i$, *and*

(5) *for every set* $S \in SPARSE_h$, *the set* $D = \{w \in G \mid w \in L(N_{f,t}, S) \Leftrightarrow w \notin L_{l_k}(C)\}$ *has size* $\|D\| \geq 2^{(\gamma-(\alpha+\beta_j)\delta)n}$ *if* $\gamma > 0$, *and has size* $\|D\| > 0$ *if* $\gamma = 0$.

Let $f = f_i^{(2^k-1)}$, and t the k-tt-condition with $\|G_t\| \geq 2^{(l_k-1-\delta)n}$. We observe that machine $N_{f,t}$ and set G satisfy the assumptions of Lemma 4.3. Note that here we have $m=0$, $\gamma=0$ and $\alpha=1$. So, we may apply Lemma 4.3 to obtain a set C of strings of the form $v_1 \ldots v_q 0^{(l_k-q-1)(n+1)}$ such that $\|C\| \leq 2^k$ and such that for every set $S \in SPARSE_h$, there exists a string $w \in G$ such that $w \in L_{l_k}(C) \Leftrightarrow w \notin L(N_{f,t}, S)$. Since $w \in G$ implies that $w \in L(N_{f,t}, S) \Leftrightarrow w \in L(N_{i,j}, S)$ for every set $S \in SPARSE_h$, w is a witness to the requirement $L_{l_k}(C) \neq L(N_{i,j}, S)$. We let $A_e = A_{e-1} \cup C$. This completes Stage e.

It is quite clear from the choice of n_e that if we let $A = \bigcup_{e=1}^{\infty} A_e$ then for every $\leq_{k\text{-T}}^{P}$-reduction machine $N_{i,j}$ and every $S \in SPARSE_h$, $L_{l_k}(A) \cap \Sigma^{(l_k-1)n} \neq L(N_{i,j}, S) \cap \Sigma^{(l_k-1)n}$. So the theorem is proven. It remains to prove Lemma 4.3.

Proof of Lemma 4.3. We prove the lemma by induction on $j = 2, \ldots, k$. First assume that $j = 2$. Then, $l_j = 3$. For each $x \in \Sigma^{(l_k-1)n}$, assume that the tree T_x generated by $f(x)$ has nodes $T_x(\lambda) = x_\lambda$, $T_x(0) = x_0$, and $T_x(1) = x_1$. For each z and each string $s = \lambda, 0, 1$, let $H_s(z) = \{x \in G \mid x_s = z\}$. Let $U_m = \{u_1 \ldots u_m\}\Sigma^{(l_k-m-1)n}$. Then, $G \subseteq U_m$ and $\|G\| \geq 2^{(2+\gamma-\alpha\delta)n}$. Now consider the following three cases.

Case 1. There exists a string z_λ such that $\|H_\lambda(z_\lambda)\| \geq 2^{(2+\gamma-(\alpha+2)\delta)n}$.

Then, we claim that there exist strings $u_{m+1}, \ldots, u_{m+q-1}, u_{m+q}, v_{m+q} \in \Sigma^n$, $1 \leq q \leq l_k - m - \gamma - 2$, such that

$$2^{(1+\gamma-(\alpha+3)\delta)n} \leq \|H_\lambda(z_\lambda) \cap E\| \leq 2^{(2+\gamma-(\alpha+2)\delta)n-2}$$

holds for both $E = U_{m+q}$ and $E = V_{m+q}$, where $U_{m+q} = \{u_1 \ldots u_{m+q-1} u_{m+q}\}\Sigma^{(l_k-m-q-1)n}$ and $V_{m+q} = \{u_1 \ldots u_{m+q-1} v_{m+q}\}\Sigma^{(l_k-m-q-1)n}$.

Proof of Claim. For convenience, let $H = H_\lambda(z_\lambda)$. For $i = 1, \ldots, l_k - m - \gamma - 2$, recursively define strings u_{m+i}, v_{m+i} and sets U_{m+i} and V_{m+i} as follows: let u_{m+i} be the string in Σ^n such that $\|H \cap \{u_1 \ldots u_{m+i-1} u_{m+i}\}\Sigma^{(l_k-m-i-1)n}\|$ is the largest among all choices of u_{m+i} in Σ^n, and let v_{m+i} be the string in Σ^n such that $\|H \cap \{u_1 \ldots u_{m+i-1} v_{m+i}\}\Sigma^{(l_k-m-i-1)n}\|$ is the second largest. Let $U_{m+i} = \{u_1 \ldots u_{m+i-1} u_{m+i}\}\Sigma^{(l_k-m-i-1)n}$ and $V_{m+i} = \{u_1 \ldots u_{m+i-1} v_{m+i}\}\Sigma^{(l_k-m-i-1)n}$.

Now let q be the smallest i, $1 \leq i \leq l_k - m - \gamma - 2$, such that

$$2^{(1+\gamma-(\alpha+3)\delta)n} \leq \|H \cap V_{m+i}\| \leq \|H \cap U_{m+i}\| \leq 2^{(2+\gamma-(\alpha+2)\delta)n-2}.$$

Then, the strings $u_{m+1}, \ldots, u_{m+q-1}, u_{m+q}, v_{m+q}$ are what we want.

We need to show that such a q must exist. First note that if $i = l_k - m - \gamma - 2$, then

$$\|H \cap U_{m+i}\| \leq \|U_{m+i}\| = 2^{(l_k-m-i-1)n} = 2^{(1+\gamma)n} < 2^{(2+\gamma-(\alpha+2)\delta)n-2}.$$

So, we can let q' be the smallest i such that $\|H \cap U_{m+i}\| \leq 2^{(2+\gamma-(\alpha+2)\delta)n-2}$. Note that $\|H \cap U_{m+q'-1}\| > 2^{(2+\gamma-(\alpha+2)\delta)n-2}$. Next assume, by way of contradiction, that

$$\|H \cap V_{m+i}\| < 2^{(1+\gamma-(\alpha+3)\delta)n}$$

for all i, $q' \leq i \leq l_k - m - \gamma - 2$. Then, we claim that $\|H \cap U_{m+i}\| \geq 2^{(2+\gamma-(\alpha+2)\delta)n-(i-q'+3)}$ for all i, $q' \leq i \leq l_k - m - \gamma - 2$. In particular, when $i = l_k - m - \gamma - 2$, we have $\|H \cap U_{m+i}\| \geq 2^{(2+\gamma-(\alpha+2)\delta)n-(l_k+3)} > 2^{(1+\gamma)n}$, which contradicts with the fact that $\|U_{m+i}\| = 2^{(1+\gamma)n}$. This proves the existence of such a q.

It remains to check the claim, which can be done by a simple induction. We observe that if $\|H \cap U_{m+i-1}\| \geq 2^{(2+\gamma-(\alpha+2)\delta)n-(i-q'+2)}$ and $\|H \cap V_{m+i}\| < 2^{(1+\gamma-(\alpha+3)\delta)n}$, then for all $v \in \Sigma^n$, $v \neq u_{m+i}$, we have $\|H \cap \{u_1 \ldots u_{m+i-1}v\}\Sigma^{(l_k-m-i-1)n}\| < 2^{(1+\gamma-(\alpha+3)\delta)n}$, and hence

$$\begin{aligned}\|H \cap U_{m+i}\| &\geq \|H \cap U_{m+i-1}\| - 2^n \cdot 2^{(1+\gamma-(\alpha+3)\delta)n} \\ &\geq 2^{(2+\gamma-(\alpha+2)\delta)n-(i-q'+2)} - 2^{(2+\gamma-(\alpha+3)\delta)n} \\ &\geq 2^{(2+\gamma-(\alpha+2)\delta)n-(i-q'+3)}.\end{aligned}$$

This completes the proof of the claim. □

Now we consider the following subcases.

Subcase 1.1. There exists a string z_0 such that $\|H_\lambda(z_\lambda) \cap H_0(z_0) \cap E\| \geq 2^{(1+\gamma-(\alpha+5)\delta)n}$ for $E = U_{m+q}$ or $E = V_{m+q}$.

Without loss of generality, assume that the above holds with $E = U_{m+q}$. We further consider two sub-subcases.

Subcase 1.1.1. There exists a string z_1 such that $\|H_\lambda(z_\lambda) \cap H_0(z_0) \cap H_1(z_1) \cap U_{m+q}\| \geq 2^{(1+\gamma-(\alpha+7)\delta)n}$.

To define set C, we separate the case of $\gamma > 0$ from the case of $\gamma = 0$. First assume that $\gamma > 0$. Then, similarly to the argument in Case 1 above, we can find strings $u_{m+q+1}, \ldots, u_{m+q+r-1}, u_{m+q+r}, v_{m+q+r} \in \Sigma^n$, with $m + q + r \leq l_k - \gamma - 1$, such that $2^{(\gamma-(\alpha+8)\delta)n} \leq \|H \cap E\| \leq 2^{(1+\gamma-(\alpha+7)\delta)n-2}$ holds for both $E = U_{m+q+r}$ and $E = V_{m+q+r}$, where $H = H_\lambda(z_\lambda) \cap H_0(z_0) \cap H_1(z_1)$, $U_{m+q+r} = \{u_1 \ldots u_{m+q+r-1}u_{m+q+r}\}\Sigma^{cn}$, $V_{m+q+r} = \{u_1 \ldots u_{m+q+r-1}v_{m+q+r}\}\Sigma^{cn}$ and $c = l_k - m - q - r - 1$. We assign the string $u_1 \ldots u_{m+q+r-1}u_{m+q+r}0^{c(n+1)}$ to set C and the string $u_1 \ldots u_{m+q+r-1}v_{m+q+r}0^{c(n+1)}$ to $\bar{C}$.

If $\gamma = 0$, then we can find strings $u_{m+q+1}, \ldots, u_{m+q+r-1}, u_{m+q+r}, v_{m+q+r} \in \Sigma^n$, with $r = l_k - m - q - 1$, such that both $w_1 = u_1 \ldots u_{m+q+r-1}u_{m+q+r}$ and $w_2 = u_1 \ldots u_{m+q+r-1}v_{m+q+r}$ are in $H_\lambda(z_\lambda) \cap H_0(z_0) \cap H_1(z_1)$. We assign w_1 to C and w_2 to $\bar{C}$.

Subcase 1.1.2. The condition specifying Subcase 1.1.1 does not hold.

In this case, we will do the same thing as in Subcase 1.1.1 with respect to the set $H_\lambda(z_\lambda) \cap H_0(z_0)$ (instead of the set $H_\lambda(z_\lambda) \cap H_0(z_0) \cap H_1(z_1)$). In addition, we assign the string $u_1 \ldots u_{m+q}0^{(l_k-m-q-1)(n+1)}$ to C iff $t(1, 0) = 0$.

Subcase 1.2. The condition specifying Subcase 1.1 does not hold, and there exists a string z_1 such that

$$\|H_\lambda(z_\lambda)\cap H_1(z_1)\cap E\|\geq 2^{(1+\gamma-(\alpha+5)\delta)n}$$

for $E=U_{m+q}$ or $E=V_{m+q}$.

This case is symmetric to Subcase 1.1. We do similar assignments to C and to $\bar{C}$.

Subcase 1.3. The conditions specifying Subcases 1.1 and 1.2 do not hold.

We assign the string $u_1\ldots u_{m+q-1}u_{m+q}0^{(l_k-m-q-1)(n+1)}$ to C and $u_1\ldots u_{m+q-1}v_{m+q}0^{(l_k-m-q-1)(n+1)}$ to $\bar{C}$.

The above completes the construction of C in Case 1. Before we continue the construction for Cases 2 and 3, we first show that the set C constructed above satisfies, when Case 1 holds, the requirements of the lemma.

It is clear that the strings in C have the specified form and that the size of C is bounded by $2<4=2^j$. To check condition (5), let z_λ be the string such that $\|H_\lambda(z_\lambda)\|\geq 2^{(2+\gamma-(\alpha+2)\delta)n}$. We consider the subcases described above.

Lemma 4.4. *If Subcase* 1.1.1 *holds, then set C satisfies condition* (5).

Proof. Let $S\in SPARSE_h$ be given. First let us assume that $z_\lambda\notin S$ and $\gamma>0$. Let z_0 and z_1 be strings such that $\|H\cap U_{m+q}\|\geq 2^{(1+\gamma-(\alpha+7)\delta)n}$, where $H=H_\lambda(z_\lambda)\cap H_0(z_0)\cap H_1(z_1)$. Then, either $H\cap U_{m+q}\subseteq L(N_{f,t},S)$ (if $t(0,\chi_S(z_0))=1$), or $H\cap U_{m+q}\subseteq \overline{L(N_{f,t},S)}$ (if $t(0,\chi_S(z_0))=0$).

On the other hand, we have assigned string $u_1\ldots u_{m+q+r-1}u_{m+q+r}0^{c(n+1)}$ to C and string $u_1\ldots u_{m+q+r-1}v_{m+q+r}0^{c(n+1)}$ to $\bar{C}$. As a consequence, $H\cap U_{m+q+r}\subseteq L_{l_k}(C)$ and $H\cap V_{m+q+r}\subseteq \overline{L_{l_k}(C)}$. So, either $H\cap U_{m+q+r}$ or $H\cap V_{m+q+r}$ has the property that all its strings x satisfy

$$x\in L(N_{f,t},S)\Leftrightarrow x\notin L_{l_k}(C).$$

That is, $\|D\|\geq 2^{(\gamma-(\alpha+8)\delta)n}$.

Note that when $\gamma=0$, the same argument proves that either $x=w_1$ or $x=w_2$ has the property that $x\in L(N_{f,t},S)\Leftrightarrow x\notin L_{l_k}(C)$. Therefore, $D\neq\emptyset$.

For the case when $z_\lambda\notin S$, it is not hard to see that a symmetric argument can be used because the condition specifies Subcase 1.1.1 is symmetric. □

Lemma 4.5. *If Subcase* 1.1.2 *holds, then set C satisfies condition* (5).

Proof. Let $S\in SPARSE_h$ be given. For Subcase 1.1.2, we note that if $z_\lambda\notin S$ then the situation is similar to Subcase 1.1.1 and a similar proof exists. We assume that $z_\lambda\in S$.

We note that $z_\lambda\in S$ implies that for all strings $x\in H_\lambda(z_\lambda)$, with $x_1\notin S$, $x\in L(N_{f,t},S)\Leftrightarrow t(1,0)=1$. On the other hand, we assigned string $u_1\ldots u_{m+q}0^{(l_k-m-q-1)(n+1)}$ to C iff $t(1,0)=0$, and assigned one of strings $u_1\ldots u_{m+q+r-1}u_{m+q+r}0^{c(n+1)}$ and $u_1\ldots u_{m+q+r-1}v_{m+q+r}0^{c(n+1)}$ to C. Therefore, for each string $x \in H_\lambda(z_\lambda) \cap H_0(z_0) \cap U_{m+q}$ with $x_1 \notin S$ and $x \notin U_{m+q+r} \cup V_{m+q+r}$, it has the property that $x \in L(N_{f,t}, S) \Leftrightarrow x \notin L_{l_k}(C)$. Note that we have

$$\begin{aligned}
&\|\{x \in H_\lambda(z_\lambda) \cap H_0(z_0) \cap U_{m+q} \mid x_1 \in S\}\| \\
&\quad \leq \sum_{z \in S} \|H_\lambda(z_\lambda) \cap H_0(z_0) \cap H_1(z) \cap U_{m+q}\| \\
&\quad < 2^{(1+\gamma-(\alpha+7)\delta)n} \cdot 2^{\delta n} = 2^{(1+\gamma-(\alpha+6)\delta)n}, \\
&\|H_\lambda(z_\lambda) \cap H_0(z_0) \cap U_{m+q+r}\| < 2^{(1+\gamma-(\alpha+5)\delta)n-2} \\
&\|H_\lambda(z_\lambda) \cap H_0(z_0) \cap V_{m+q+r}\| < 2^{(1+\gamma-(\alpha+5)\delta)n-2}.
\end{aligned}$$

So,

$$\begin{aligned}
\|D\| &\geq \|H_\lambda(z_\lambda) \cap H_0(z_0) \cap (U_{m+q} - U_{m+q+r} - V_{m+q+r}) \cap \{x \mid x_1 \notin S\}\| \\
&\geq 2^{(1+\gamma-(\alpha+5)\delta)n} - 2 \cdot 2^{(1+\gamma-(\alpha+5)\delta)n-2} - 2^{(1+\gamma-(\alpha+6)\delta)n} \\
&\geq 2^{(1+\gamma-(\alpha+6)\delta)n}.
\end{aligned}$$

This completes the proof of Lemma 4.5. (Both cases of $\gamma > 0$ and $\gamma = 0$ are proved by the same argument above.) □

Lemma 4.6. *If Subcase* 1.2 *holds, then set C satisfies condition* (5).

Proof. The proof is symmetric to that for Lemmas 4.4 and 4.5. □

Lemma 4.7. *If Subcase* 1.3 *holds, then set C satisfies condition* (5).

Proof. In Subcase 1.3, we have assigned strings $u_1 \ldots u_{m+q-1} u_{m+q} 0^{(l_k-m-q-1)(n+1)}$ and $u_1 \ldots u_{m+q-1} v_{m+q} 0^{(l_k-m-q-1)(n+1)}$ to C and $\bar{C}$, respectively, so that $U_{m+q} \subseteq L_{l_k}(C)$ and $V_{m+q} \subseteq \overline{L_{l_k}(C)}$.

On the other hand, note that all strings x in $H_\lambda(z_\lambda)$ with $x_0 \notin S$ and $x_1 \notin S$ must have the same membership in $L(N_{f,t}, S)$, depending upon whether $t(\chi_S(z_\lambda), 0) = 1$ or not. Since for any string z, $\|H_\lambda(z_\lambda) \cap H_0(z) \cap U_{m+q}\|$ and $\|H_\lambda(z_\lambda) \cap H_1(z) \cap U_{m+q}\|$ are bounded by $2^{(1+\gamma-(\alpha+5)\delta)n}$, there are at most

$$2^{(1+\gamma-(\alpha+5)\delta)n} \cdot 2^{\delta n} \cdot 2 \leq 2^{(1+\gamma-(\alpha+4)\delta)n}$$

strings x in $H_\lambda(z_\lambda) \cap U_{m+q}$ having either $x_0 \in S$ or $x_1 \in S$. The same property holds for set $H_\lambda(z_\lambda) \cap V_{m+q}$. So, one of the sets $H_\lambda(z_\lambda) \cap U_{m+q}$ and $H_\lambda(z_\lambda) \cap V_{m+q}$ has at least

$$2^{(1+\gamma-(\alpha+3)\delta)n} - 2^{(1+\gamma-(\alpha+4)\delta)n} \geq 2^{(1+\gamma-(\alpha+4)\delta)n}$$

strings x satisfying

$$x \in L(N_{f,t}, S) \Leftrightarrow x \notin L_{l_k}(C).$$

That is, $\|D\| \geq 2^{(1+\gamma-(\alpha+4)\delta)n}$. □

The above four lemmas complete the proof for Case 1. Now we proceed to Cases 2 and 3.

Case 2. The condition for Case 1 does not hold, and there exists a string z_0 such that $\|H_0(z_0)\| \geq 2^{(2+\gamma-(\alpha+2)\delta)n}$.

Similarly to Case 1, we can find strings $u_{m+1}, \ldots, u_{m+q-1}, u_{m+q}, v_{m+q} \in \Sigma^n$, $1 \leq q \leq l_k - m - \gamma - 2$, such that

$$2^{(1+\gamma-(\alpha+3)\delta)n} \leq \|H_0(z_0) \cap E\| \leq 2^{(2+\gamma-(\alpha+2)\delta)n-2}$$

holds for both $E = U_{m+q}$ and $E = V_{m+q}$. Then, we consider the following subcases.

Subcase 2.1. There exists a string z_λ such that $\|H_\lambda(z_\lambda) \cap H_0(z_0) \cap E\| \geq 2^{(1+\gamma-(\alpha+5)\delta)n}$ for $E = U_{m+q}$ or $E = V_{m+q}$.

This case is exactly the same as Subcase 1.1. (Note that in the proofs of Lemmas 4.4 and 4.5, we did not use the fact that $\|H_\lambda(z_\lambda) \cap U_{m+q}\| \geq 2^{(1+\gamma-(\alpha+3)\delta)n}$, which is the only difference between Subcase 1.1 and Subcase 2.1.) We construct C as in that case.

Subcase 2.2. The condition specifying Subcase 2.1 does not hold.

We assign the string $u_1 \ldots u_{m+q-1} u_{m+q} 0^{c'(n+1)}$ to C and the string $u_1 \ldots u_{m+q-1} v_{m+q} 0^{c'(n+1)}$ to $\bar{C}$, where $c' = l_k - m - q - 1$.

This completes the construction of C for Case 2. We check that, in this case, the set C constructed above satisfies the requirements of the lemma. Note that all strings in C have the specified form and that the size of C is bounded by $2 < 4 = 2^j$. Furthermore, since Subcase 2.1 is exactly the same as Subcase 1.1, condition (5) is satisfied by C in that case. We need consider only Subcase 2.2.

Lemma 4.8. *If Subcase* 2.2 *holds, then set C satisfies condition* (5).

Proof. This case is similar to Subcase 1.3. We have assigned strings $u_1 \ldots u_{m+q-1} u_{m+q} 0^{(l_k-m-q-1)(n+1)}$ and $u_1 \ldots u_{m+q-1} v_{m+q} 0^{(l_k-m-q-1)(n+1)}$ to C and $\bar{C}$, respectively, so that $U_{m+q} \subseteq L_{l_k}(C)$ and $V_{m+q} \subseteq \overline{L_{l_k}(C)}$.

On the other hand, for every set $S \in SPARSE_h$, note that all strings x in $H_0(z_0)$ with $x_\lambda \notin S$ must have the same membership in $L(N_{f,t}, S)$, depending upon whether $t(0, \chi_S(z_0)) = 1$ or not. Since for any string z, $\|H_\lambda(z) \cap H_0(z_0) \cap U_{m+q}\|$ is bounded by $2^{(1+\gamma-(\alpha+5)\delta)n}$, there are at most

$$2^{(1+\gamma-(\alpha+5)\delta)n} \cdot 2^{\delta n} \cdot 2 \leq 2^{(1+\gamma-(\alpha+4)\delta)n}$$

strings x in $H_0(z_0) \cap U_{m+q}$ having $x_\lambda \in S$. The same property holds for set $H_0(z_0) \cap V_{m+q}$. So, one of the sets $H_0(z_0) \cap U_{m+q}$ and $H_0(z_0) \cap V_{m+q}$ contains more than $2^{(1+\gamma-(\alpha+4)\delta)n}$ strings x such that

$$x \in L(N_{f,t}, S) \Leftrightarrow x \notin L_{l_k}(C).$$

This shows that $\|D\| \geq 2^{(1+\gamma-(\alpha+4)\delta)n}$, and the lemma is proven. □

Case 3. The conditions specifying Cases 1 and 2 do not hold. We assign string $u_1 \ldots u_m 0^{(l_k-m-1)(n+1)}$ to C iff $t(0,0) = 0$.

Lemma 4.9. *If Case 3 holds, then set C satisfies condition* (5).

Proof. Note that in this case, all x in G but at most $2^{(2+\gamma-(\alpha+2)\delta)n}\cdot 2^{\delta n}\cdot 2 = 2^{(2+\gamma-(\alpha+1)\delta)n+1}$ many have $x_\lambda \notin S$ and $x_0 \notin S$. For any x having this property, we observe that $x \in L_{l_k}(C) \Leftrightarrow u_1 \dots u_m 0^{(l_k-m-1)(n+1)} \in C \Leftrightarrow t(0,0)=0 \Leftrightarrow x \notin L(N_{f,t}, S)$. Thus,

$$\begin{aligned}\|D\| &\geq \|\{x \in G \mid x_\lambda \notin S \text{ and } x_0 \notin S\}\| \\ &\geq 2^{(2+\gamma-\alpha\delta)n} - 2^{(2+\gamma-(\alpha+1)\delta)n+1} \geq 2^{(2+\gamma-(\alpha+1)\delta)n}.\end{aligned}$$

This completes the proof of Lemma 4.9, as well as the initial step of the induction proof. □

For the inductive step, we assume that $k>2$ and $2<j\leq k$. Note that $l_{j-1}=l_j/2$. Again, we consider two cases.

Case 1. There exists a string z_λ such that $\|H_\lambda(z_\lambda)\| \geq 2^{(l_j/2+\gamma-(\alpha+\beta_{j-1}+2)\delta)n}$.

Similar to the argument in Case 1 of the initial step (when $j=2$), we can find strings $u_{m+1}, \dots, u_{m+q-1}, u_{m+q}, v_{m+q} \in \Sigma^n$, $1 \leq q \leq l_k - m - \gamma - l_j/2$, such that

$$\|H_\lambda(z_\lambda) \cap E\| \geq 2^{(l_j/2+\gamma-1-(\alpha+\beta_{j-1}+3)\delta)n}$$

holds for both $E=U_{m+q}$ and $E=V_{m+q}$, where $U_{m+q} = \{u_1 \dots u_{m+q-1}u_{m+q}\}\Sigma^{(l_k-m-q-1)n}$ and $V_{m+q} = \{u_1 \dots u_{m+q-1}v_{m+q}\}\Sigma^{(l_k-m-q-1)n}$.

Step A. Define f_0 to be the function which maps each string x to the left subtree of $f(x)$ and t_0 to be the $(j-1)$-tt-condition defined by $t_0(b_1, \dots, b_{j-1}) = t(0, b_1, \dots, b_{j-1})$. Without loss of generality, we assume that the machine N_{f_0,t_0}, defined by f_0 and t_0, has runtime bounded by p' also. Also let $G_0 = H_\lambda(z_\lambda) \cap U_{m+q}$. Note that

$$\|G_0\| \geq 2^{(l_{j-1}+\gamma-1-(\alpha+\beta_{j-1}+3)\delta)n}.$$

Since $\alpha \leq \beta_k - \beta_j + 1$ implies $\alpha+\beta_{j-1}+3 \leq \beta_k - \beta_{j-1}+1$, we can apply the inductive hypothesis to machine N_{f_0,t_0} and set G. That is, we can find a set C_0 satisfying conditions (3)-(5), with respect to strings $u_1, \dots, u_{m+q}$ and parameter $j-1$. In particular, condition (5) states that for each $S \in SPARSE_h$, $\|D_0\| \geq 2^{(\gamma-(\alpha+\beta_j)\delta)n}$ if $\gamma \geq 1$, and $D_0 \neq \emptyset$ if $\gamma=0$, where $D_0 = \{w \in G_0 \mid w \in L(N_{f_0,t_0}, S) \Leftrightarrow w \notin L_{l_k}(C_0)\}$. (Note that $\beta_j = 2\beta_{j-1}+3$.)

Step B. Define f_1, t_1 and G_1 similarly: $f_1(x)$ is the right subtree of $f(x)$, $t_1(b_1, \dots, b_{j-1}) = t(1, b_1, \dots, b_{j-1})$, and $G_1 = H_\lambda(z_\lambda) \cap V_{m+q}$. By the inductive hypothesis, we find a set C_1 satisfying conditions (3)-(5). In particular, condition (5) states that for each $S \in SPARSE_h$, $\|D_1\| \geq 2^{(\gamma-(\alpha+\beta_j)\delta)n}$ if $\gamma \geq 1$, and $D_1 \neq \emptyset$ if $\gamma=0$, where $D_1 = \{w \in G_1 \mid w \in L(N_{f_1,t_1}, S) \Leftrightarrow w \notin L_{l_k}(C_1)\}$.

Let $C = C_0 \cup C_1$ and claim that C satisfies conditions (3)-(5) (with respect to strings $u_1, \dots, u_m$ and parameter j). First note that $\|C\| \leq \|C_0\| + \|C_1\| \leq 2^{j-1}+2^{j-1} = 2^j$. Also note that all strings in C_0 begin with prefix $u_1 \dots u_{m+q-1}u_{m+q}$ and all strings in C_1 begin with prefix $u_1 \dots u_{m+q-1}v_{m+q}$. This implies that all strings in C have the specified form. Finally, we prove condition (5).

Lemma 4.10. *If Case* 1 *holds, then set C constructed above satisfies condition* (5).

Proof. Let $S \in SPARSE_h$ and recall that $D = \{w \in G \mid w \in L(N_{f,t}, S) \Leftrightarrow w \notin L_{l_k}(C)\}$, and $D_0 = \{w \in G_0 \mid w \in L(N_{f_0,t_0}, S) \Leftrightarrow w \notin L_{l_k}(C_0)\}$. First consider the case that $\gamma > 0$ and $z_\lambda \notin S$. Then, by the definition of C_0, $\|D_0\| \geqslant 2^{(\gamma-(\alpha+\beta_j)\delta)n}$. Note that for all x in $G_0 = H_\lambda(z_\lambda) \cap U_{m+q}$, $x \in L_{l_k}(C) \Leftrightarrow x \in L_{l_k}(C_0)$ (because all strings in C_1 begin with different prefixes from those in C_0), and $x \in L(N_{f,t}, S) \Leftrightarrow x \in L(N_{f_0,t_0}, S)$ (because $x_\lambda = z_\lambda \notin S$). So, $D_0 \subseteq D$ and $\|D\| \geqslant 2^{(\gamma-(\alpha+\beta_j)\delta)n}$. Similar results can be proved for the cases when $\gamma = 0$ and/or when $z_\lambda \in S$. We leave the detail to the reader. □

Case 2. The condition specifying Case 1 does not hold.

Then, we define function f_0 and $(j-1)$-tt-condition t_0 as in Case 1. Let $\gamma_0 = \gamma + l_j/2$, and $G_0 = G$. Then, by the inductive hypothesis, there exists a set C_0 satisfying conditions (3)-(5) with respect to strings $u_1, \ldots, u_{m+q}$ and parameter $j-1$. In particular, condition (5) states that for each $S \in SPARSE_h$, $\|D_0\| \geqslant 2^{(\gamma_0-(\alpha+\beta_{j-1})\delta)n}$, where $D_0 = \{w \in G_0 \mid w \in L(N_{f_0,t_0}, S) \Leftrightarrow w \notin L_{l_k}(C_0)\}$ (because $\gamma_0 > 0$).

Let $C = C_0$. Clearly strings in C have the specified form and $\|C\| \leqslant 2^{j-1} < 2^j$.

Lemma 4.11. *If Case* 2 *holds, then set C constructed above satisfies condition* (5).

Proof. Let $S \in SPARSE_h$. Similar to Case 1, for those $x \in G_0$ with $x_\lambda \notin S$, $x \in L(N_{f,t}, S) \Leftrightarrow x \in L(N_{f_0,t_0}, S)$. That is,

$$
\begin{aligned}
\|D\| &\geqslant \|D_0 \cap \{x \mid x_\lambda \notin S\}\| \geqslant \|D_0\| - \|\{x \in G_0 \mid x_\lambda \in S\}\| \\
&> 2^{(l_{j-1}+\gamma-(\alpha+\beta_{j-1})\delta)n} - 2^{(l_{j-1}+\gamma-(\alpha+\beta_{j-1}+2)\delta)n} \cdot 2^{\delta n} \\
&\geqslant 2^{(l_{j-1}+\gamma-(\alpha+\beta_j)\delta)n}.
\end{aligned}
$$

Therefore, $\|D\| > 2^{(\gamma-(\alpha+\beta_j)\delta)n}$ if $\gamma > 0$, and $D \neq \emptyset$ if $\gamma = 0$. □

This completes the proof for the inductive step, as well as the proof of Lemma 4.3. □

Example 4.12. To help the reader understand the role of parameters γ and β_j in the above proof, let us look at an example. Let $k = 4$. Then $l_k = 12$, and assume that we begin with a set $G \subseteq \Sigma^{11n}$ having $\|G\| \geqslant 2^{(11-\delta)n}$ (i.e., $\gamma = 0$ and $\alpha = 1$).

We first ask whether there is a string z_λ such that $\|H_\lambda(z_\lambda)\| \geqslant 2^{(6-22\delta)n}$ ($\beta_3 = 19$). Assume that the answer is "no". Then, we ask whether there is a string z_0 such that $\|H_0(z_0)\| \geqslant 2^{(9-11\delta)n}$ (γ has been changed to 6 and $\beta_2 = 8$). Assume that the answer is "yes". Then, we find strings $u_1, \ldots, u_{q-1}$, u_q, v_q in Σ^n such that $\|H_0(z_0) \cap \{u_1 \ldots u_{q-1} u_q\}\Sigma^{(11-q)n}\| \geqslant 2^{(8-12\delta)n}$ and $\|H_0(z_0) \cap \{u_1 \ldots u_{q-1} v_q\}\Sigma^{(11-q)n}\| \geqslant 2^{(8-12\delta)n}$.

Let $G_0 = H_0(z_0) \cap \{u_1 \ldots u_{q-1}u_q\}\Sigma^{(11-q)n}$, and consider the leftmost height-2 subtree of $f(x)$. By a rather ad hoc procedure (i.e., the procedure for the initial step of the above induction proof), we find a set C_0 such that $D = \{x \in G_0 \mid x \in L_{l_k}(C_0) \Leftrightarrow x \notin L(N_{f_0,t_0}, S)\}$ has size $\|D\| \geq 2^{(6-20\delta)n}$, where N_{f_0,t_0} is the $\leq^{\mathrm{P}}_{2\text{-T}}$-reduction machine defined by the leftmost height-2 subtree $f_0(x)$ (i.e., the subtree with the root x_{00}) and the corresponding 2-tt-conditions t_0.

Now observe that among all x in $H_0(z_0)$, at most $2^{(6-22\delta)n} \cdot 2^{\delta n} = 2^{(6-21\delta)n}$ many may have $x_\lambda \in S$ (this follows from the assumption that there is no z_λ such that $\|H_\lambda(z_\lambda)\| \geq 2^{(6-22\delta)n}$). So, at least one string x in D has $x_\lambda \notin S$. If $z_0 \notin S$, then this x witnesses that $x \in L_{l_k}(C_0) \Leftrightarrow x \notin L(N_{f,t}, S)$.

Next let $G_1 = H_0(z_0) \cap \{u_1 \ldots u_{q-1}v_q\}\Sigma^{(11-q)n}$ and repeat the above construction on G_1 with respect to the second height-2 subtree of the tree $f(x)$ (i.e., the subtree with the root x_{01}). This will show that if $z_0 \in S$ then there exists a witness x such that $x \in L_{l_k}(C_1) \Leftrightarrow x \notin L(N_{f,t}, S)$. Since strings in C_0 and strings in C_1 have different prefixes, the above requirements do not conflict with each other. So, set $C = C_0 \cup C_1$ satisfies the requirements of the lemma.

References

[1] A. Amir and W. Gasarch, Polynomial terse sets, in: *Proc. 2nd Conf. on Structure in Complexity Theory* (1987) 22-27.

[2] R. Beigel, A structure theorem that depends quantitatively on the complexity of SAT, in: *Proc. 2nd Conf. on Structure in Complexity Theory* (1987) 28-32.

[3] R. Beigel, Bounded queries to SAT and the boolean hierarchy, Theoret. Comput. Sci., to appear.

[4] R. Beigel, W. Gasarch, J. Gill and J. Owings, Terse, superterse and verbose sets, Tech. Report TR-1806, Department of Computer Science, University of Maryland, 1987.

[5] R.V. Book and K. Ko, On sets truth-table reducible to sparse sets, *SIAM J. Comput.* **17** (1988) 903-919.

[6] J. Goldsmith, D. Joseph and P. Young, Self-reducible, p-selective, near-testable, and p-cheatable sets: the effect of internal structure on the complexity of a set, in: *Proc. 2nd IEEE Conf. on Structure in Complexity Theory* (1987) 50-59.

[7] J. Kadin, The polynomial time hierarchy collapses if the boolean hierarchy collapses, in: *Proc. 3rd IEEE Conf. on Structure in Complexity Theory* (1988) 278-292.

[8] R. Karp and R. Lipton, Some connections between nonuniform and uniform complexity classes, in: *Proc. 12th ACM Symp. Theory of Computing* (1980) 302-309.

[9] K. Ko, Distinguishing conjunctive and disjunctive reducibilities by sparse sets, *Inform. and Comput.* **81** (1989) 62-87.

[10] J. Köbler, U. Schöning and K. Wagner The difference and truth-table hierarchies for NP, *RAIRO Théor. Inform.* **21** (1987) 419-435.

[11] R. Ladner, N. Lynch and A. Selman, A comparison of polynomial-time reducibilities, *Theoret. Comput. Sci.* **1** (1975) 103-123.

[12] S. Tang and R. Book, Separating polynomial-time Turing and truth-table degrees of tally sets, in: *Proc. 15th Internat. Coll. on Automata, Languages and Programming*, Lecture Notes in Computer Science **317** (Springer, Berlin, 1988) 591-599.