Distinguishing Bounded Reducibilities by Sparse Sets

(Extended Abstract)

Ker-I Ko
Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11794

1. Introduction

The notion of resource-bounded reducibilities plays an important role in complexity theory. Different ways of making queries to oracles by reduction machines define different types of reducibilities. Proofs that distinguish different types of reducibilities better our understanding of how oracle machines behave under different restrictions. Two kinds of restrictions on the power of oracle machines have arisen when considering resource-bounded reducibilities. The first kind is to restrict the underlying machine model used for reduction. These restrictions include the number of queries allowed on each input (e.g., bounded-query reducibilities), the way the answers to the queries are used (e.g., conjunctive and disjunctive reducibilities), and the dependency between queries (e.g., adaptive and nonadaptive reducibilities). Comparison among reducibilities of this kind of restrictions reveals the differences among these oracle machine models.

The second kind of restrictions on oracle machines is to restrict the resources used by the machine, usually in terms of runtime or workspace. For a fixed type of reducibility ≤r, we may consider the log-space reducibility ≤ls, the polynomial-time reducibility ≤P, the NP reducibility ≤NP, and the strong NP reducibility ≤SNP. This formulation of restricted reducibilities and their comparison provide a more general framework for the basic questions in complexity theory such as determinism versus nondeterminism, and time versus space. For example, the separation result that there exist sets A and B such that A ≤P B but A ≤NP B has now become one of the fundamental result in the theory of relativization (Baker, Gill and Solovay [3]).

In this paper, we will study the reducibilities defined by the first kind of restrictions, and the only resource bound on oracle machines is the polynomial time bound. The comparison of polynomial-time bounded reducibilities can be made in many different approaches. The simplest one is to consider each type of reducibility as a binary relation on sets of finite strings and a separation result involves the proof that two types of reducibility define two different relation. More precisely, for any two types of reducibility ≤P and ≤NP, we distinguish them by finding sets A and B such that B ≤P A but B ≤NP A.

Ladner, Lynch and Selman [19] are among the first to show such simple separation results on some of the best known reducibilities: ≤P, ≤P, ≤P, ≤NP, ≤NP, ≤NP, and ≤NP (See Section 2 for the definitions of these reducibilities.) Most of these simple separations can be done by sets computable in exponential time. An interesting question arises asking whether sets in NP or PSPACE can distinguish these reducibilities, assuming that P ≠ NP or P ≠ PSPACE, respectively. In particular, can we distinguish sets ≤P-complete in NP from sets ≤NP-complete in NP? Selman [21] has considered this question and found sets A, B in NP such that A ≤NP B but A ≤NP B (assuming EXP ≠ NEXP), but the type of sets he used, called p-selective sets, are too simple to distinguish ≤P-completeness from ≤NP-completeness in NP, unless the polynomial time hierarchy collapses [14].

This last question motivates the second approach to the comparison of polynomial-time reducibilities, namely, to compare the complete sets in some complexity class C defined by different types of reducibility. As this is difficult for the classes NP and PSPACE, the natural candidate is the class EXP of all sets computable in time 2O(n). For each type of reducibility ≤P, let Cm(EXP) denote the class of complete sets for EXP under the ≤P-reducibility. Ko and Moore [17] first showed that Cm(EXP) ≠ Cm(EXP). Finer separation results in this direction have been obtained by Watanabe [23]. For example, he has shown that for each k > 1,

\[ C_m(EXP) \neq C_{k+1}(EXP) \neq C_{k+2}(EXP) \leq C_k(EXP) \]

Recently Book and Ko [8] considered a third type of separation results on polynomial-time reducibilities by comparing the reduction classes defined by sparse oracles. In general, let C be a class of sets defined by some complexity bound or some structural restriction. Then, we may consider, for each type of reducibility ≤P, the reduction class \( P(C) \) of all sets A such that A ≤P B for some
set $B \in C$. A stronger separation result (with respect to the class $C$) on two reducibilities of separation results are stronger than simple separation results of Ladner, Lynch and Selman. Intuitively, the information stored in set $A$ can be found by a $\leq^P_T$-type oracle machine from some set $B \in C$ but cannot be found by a weaker $\leq^P_T$-type oracle machine, even if one is allowed to reorganize the information into a different set $C$ in $C$. The study of reduction classes defined by the class $\text{SPARSE}$ of all sparse sets was basically motivated by recent interesting characterizations of the class $P/poly$ of sets computable by polynomial-size circuits (Karp and Lipton [13]) and the characterization of the class of sets $\leq^P_T$-equivalent to tally sets by generalized Kolmogorov complexity (Balcazar and Book [4]). Their main results may be summarized as follows:

(a) $P_{k,t}(\text{SPARSE}) \neq P_{k+1,t}(\text{SPARSE})$, for all $k > 0$.
(b) $P_{k,t}(\text{SPARSE}) \neq P_{\ell}(\text{SPARSE}) = P_T(\text{SPARSE}) = P^{/\text{poly}}$.

In this paper, we continue the investigation of polynomial-time reducibilities along the direction of Book and Ko [8]. We compare the reduction classes defined by sparse oracles under polynomial-time bounded reducibilities, including $\leq^P_{ktu}$, $\leq^P_{tr}$, $\leq^P_{dtu}$, $\leq^P_{dtu}$, and $\leq^P_{du}$. Recently bounded query machines have been widely studied. Amir and Gasarch [2], Beigel [5] and Goldsmith, Joseph and Young [10] 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). Köbler, Schoning and Wagner [18] have shown that the classes $P_{k,t}(NP)$, $k > 0$, define a truth-table hierarchy in $\Delta^P_4$ which is nicely intertwined with the boolean hierarchy in $\Delta^P_3$. Kadin [12] showed that this hierarchy is a properly infinite hierarchy unless the Meyer-Stockmeyer polynomial-time hierarchy collapses. In regard to the comparison between $\leq^P_{kt}$ and $\leq^P_{k+1}$-reducibilities within $NP$, Beigel [6] has shown that for every $k \geq 1$, $P_{k,t}(NP) = P_{k+1,t}(NP)$.

The main results of this paper are the strong separation results for $P_{k,t}(\text{SPARSE})$ versus $P_{k+1,t}(\text{SPARSE})$ and for $P_{k,t}(\text{SPARSE})$ versus $P_{k+1,t}(\text{SPARSE})$. Figure 1 summarizes the relations between $k$-truth-table and $k$-Turing reducibilities. These results represent a new approach to the more general question of adaptive versus nonadaptive computation. We discuss them in Section 3. Figure 2 summarizes the relations among bounded truth-table, conjunctive and disjunctive reducibilities. These separation results will be discussed more carefully in Section 4.

In addition to sparse oracles, we also consider tally sets as oracles. Recall that a tally set is a set over a singleton alphabet $\{0\}$. Let $\text{TALLY}$ be the class of all tally sets. We compare the reduction classes $P_r(\text{TALLY})$ for the reducibility types $r = m, ctt, dtt, tt$, and $T$. The results here are quite different from those for classes $P_r(\text{SPARSE})$. We summarize them in Figure 3. Comparing Figures 1 and 2 with Figure 3, we see much difference between strong separation results by sparse sets and those by tally sets. This discrepancy suggests us to examine the relations between sparse sets and tally sets more closely using the above reducibilities. We further discuss this issue in Section 5.

Recently still another interesting approach toward the strong separation of different reducibilities has been taken by Tang and Book [22] and Allender and Watanabe [1]. Instead of considering the classes $P_r(\text{SPARSE})$ (or $P_r(\text{TALLY})$), Tang and Book [22] considered the equivalence degrees $E^P_T(\text{SPARSE})$ (or, $E^P_T(\text{TALLY})$) of sets $\leq^P_T$-equivalent to some sparse (or, respectively, tally) sets. Some different separation results are found in this study. For instance, it is shown that $E^P_T(\text{TALLY}) \neq E^P_T(\text{SPARSE})$, and hence the adaptive and nonadaptive reducibilities are different with respect to the notion of equivalence to tally sets, in contrast to the well known result that $P_T(\text{TALLY}) = P_{dtt}(\text{TALLY}) = P^{/\text{poly}}$ (cf. [8]). Furthermore, Allender and Watanabe [1] showed that the question of whether $E^P_T(\text{TALLY}) = E^P_{k+1,t}(\text{TALLY})$ depends on the existence of certain strong one-way functions. These results, together with results in this paper, indicate that the relations between different reducibilities depend much on the formulations of the question. In other words, the exact relations depend on the information-theoretic complexity of the oracle sets as well as the different restrictions to oracle machines. Continued research on finding simple characterizations for these reduction classes and equivalence degrees, using notions other than oracle machines, is important to getting a complete picture of the relationship between reducibilities.

2. Definitions

In this paper we will use 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 $X$ and an integer $n$, let $X^\leq_n = \{x \in X \mid |x| \leq n\}$, $\chi_X$ be the characteristic function of $X$, and $\bar{X} = \Sigma^* - X$. The empty string is denoted by $\lambda$.

We assume that the reader is familiar with the notion of Turing machines, oracle Turing machines and their
time complexity. We are interested in polynomial-time reducibilities, in particular, various types of polynomial-time truth-table reducibilities and Turing reducibilities. Recall the following definitions (see [19] for the details and formal definitions):

(i) \( A \leq^p_\mu B \): if there is a polynomial-time computable function \( f \) such that for all \( x, z \in A \) if \( f(x) \in B \).

(ii) \( A \leq^p_\mu B \): if there exists a polynomial-time oracle Turing machine \( M \) such that for each input \( x, z \in A \) if \( M^B \) accepts \( z \).

(iii) \( A \leq^p_\mu B \): if there exist polynomial-time computable functions \( f \) and \( g \) such that for all \( x, f(x) \) is a list of strings, \( g(x) \) is a truth-table with the number of variables being equal to the number of strings in the list \( f(x) \), and \( z \in A \) if the truth-table \( g(x) \) evaluates to \( \text{true} \) on the \( k \)-tuple \( (x_B(1), \ldots , x_B(y_k)) \) where \( f(x) = (y_1, \ldots , y_k) \).

(iv) \( A \leq^p_\nu B \): if \( A \leq^p_\mu B \) by the functions \( f \) and \( g \) such that \( g(x) \) evaluates to \( \text{true} \) on the \( k \)-tuple \( (b_1, \ldots , b_k) \) if and only if all \( b_i \)'s are true.

(v) \( A \leq^p_\nu B \): if \( A \leq^p_\mu B \) by the functions \( f \) and \( g \) such that \( g(x) \) evaluates to \( \text{true} \) on the \( k \)-tuple \( (b_1, \ldots , b_k) \) if and only if at least one of \( b_i \)'s is true.

(vi) For every \( k > 0 \) and every reducibility \( \leq^p_\nu \) defined in (i)–(v), \( A \leq^p_\nu B \): if there is an integer \( k \) such that \( A \leq^p_\nu B \).

For any reducibility \( \leq^p_\nu \) computed in polynomial time and any class \( C \) of sets, let \( P_r(C) = \{ A \mid \text{there exists } C \subseteq C \text{ such that } A \leq^p_\nu C \} \). Recall that a set \( S \) is sparse if there is a polynomial \( q \) such that for all \( n \), \( \| S^{\leq q(n)} \| \leq q(n) \). Let \( \text{SPARSE} \) denote the class of all sparse sets. A set \( T \) is a tally set if \( A \subseteq \{0\}^\ast \). Let \( \text{TALLY} \) denote the class of all tally sets. We assume a fixed enumeration \( \{ p_n \}_{n=1}^\infty \) of polynomials; namely, \( p_n(n) = n^k + n \).

### 3. Adaptive versus Nonadaptive Reducibilities

The comparison between \( \leq^p_\nu \)-reducibility and \( \leq^p_\mu \)-reducibility is a complexity-theoretic formulation of the more general question of comparison between adaptive versus nonadaptive computation, which occurs in many different forms in different branches of computer science. It seems that there does not exist a general consensus about whether adaptive computation is strictly stronger than nonadaptive computation, as both positive and negative answers have been obtained in different formulations of the question.

In the form of truth-table and Turing reducibilities, the simplest comparison of these two reducibilities is due to Ladner, Lynch and Selman [19]: there exist sets \( A \) and \( B \) such that \( A \leq^p_\mu B \) but \( A \not\leq^p_\nu B \). In this section we are concerned with bounded reducibilities. For the simple separation, it is clear that for each \( k \geq 1 \), a \( k \)-query adaptive machine can be simulated by a \( \{2^k - 1\} \)-query nonadaptive machine. Therefore, we have, for all sets \( A \) and \( B \),

\[ A \leq^p_{k-1} B \Rightarrow A \leq^p_{k-1} B \Rightarrow A \leq^p_{2^k - 1} B. \]

By straightforward diagonalizations, we can show that the above relations are optimal. (Similar results have been reported by Beigel, Gasarch, Gill and Owings [7] in the recursion-theoretic setting.)

**Theorem 3.1.** For every \( k \geq 1 \),

(a) there exist sets \( A \) and \( B \) such that \( A \leq^p_{k-1} B \) but \( A \not\leq^p_{2^k - 1} B \), and

(b) there exist sets \( C \) and \( D \) such that \( C \leq^p_{k+1} D \) but \( C \not\leq^p_{k-1} D \).

Next we consider the reduction classes defined by sparse sets under bounded reducibilities. First we recall that Book and Ko [8] have proved that the classes \( P_{k-1}(\text{SPARSE}) \), \( k > 0 \), form a proper infinite hierarchy; that is, for every \( k > 0 \), \( P_{k-1}(\text{SPARSE}) \) is a different form in different branches of computer science.

The first main result of this paper is that the second inclusion in (a) is optimal. (The proofs of Theorems 3.2 and 3.4 are included in the Appendix.)

**Theorem 3.2.** For every \( k \geq 2 \), \( P_{k-1}(\text{SPARSE}) \subseteq P_{2^k - 1}(\text{SPARSE}) \).

This separation result has an immediate corollary that the bounded adaptive machines define a properly infinite hierarchy, because \( P_{k-1}(\text{SPARSE}) \subseteq P_{2^k - 1}(\text{SPARSE}) \) but \( P_{k+1-1}(\text{SPARSE}) \not\subseteq P_{2^k - 1}(\text{SPARSE}) \).

**Corollary 3.3.** For every \( k \geq 1 \), \( P_{k-1}(\text{SPARSE}) \not\subseteq P_{k+1-1}(\text{SPARSE}) \).

We also give a partial answer to the question of the optimality of the first inclusion of (a). For every integer \( k \geq 2 \), we demonstrate a \( \{2^k-1\} \)-query nonadaptive machine which cannot be simulated by any \( k \)-query adaptive machine, even if different sparse oracles are allowed. Whether the bound \( \{2^k-1\} \) can be reduced to \( k+1 \) is an interesting open question.
4. Conjunctive and Disjunctive Reducibilities

Conjunctive and disjunctive reducibilities are among the simplest types of reducibilities. The comparison of their powers with that of more general truth-table reducibilities tells us how severe it is to limit the ability of the reduction machines to further process the answers from the oracles. Due to the simplicity of these reduction machines, one may suspect that they are much weaker than general truth-table reduction machines. However, our comparisons of these reducibilities by sparse oracles demonstrate that the relations between these reduction classes are quite complicated.

We begin with the bounded reducibilities. Observe that the cross product $S \times \cdots \times S$ of $k$ copies of a sparse set $S$, for any fixed $k > 0$, is still a sparse set. This fact shows immediately that the hierarchy of $P_k(\text{SPARSE})$, $k > 0$, collapses.

**Theorem 4.1.** For all $k \geq 1$, $P_k(\text{SPARSE}) = P_k(\text{SPARSE}) = P_k(\text{SPARSE}) = P_k(\text{SPARSE})$.

On the other hand, the behavior of $\leq_k$- reducibilities is different and the classes $P_k(\text{SPARSE})$, $k > 0$, form a properly infinite hierarchy. In fact, we can prove that with only one additional query, a $\mu$-reduction machine could be stronger than a $\mu$-reduction machine.

**Theorem 4.2.** For any $k \geq 1$, $P_{k+1}(\text{SPARSE}) \supseteq P_k(\text{SPARSE})$.

**Corollary 4.3.** For any $k \geq 1$, $P_{k+1}(\text{SPARSE}) \supseteq P_k(\text{SPARSE})$.

**Corollary 4.4.** $P_{\mu}(\text{SPARSE}) \supseteq P_{\mu}(\text{SPARSE})$.

Next we consider the converse of Theorem 4.2. First we define the class $P(\text{CLOSE})$ as the class of sets $A$ for which there exists a set $L \in P$ such that the difference set $A \Delta L$ is sparse. It is easy to see that $P(\text{CLOSE})$ is the class $P(\text{CLOSE})$ for every $k > 0$. The following result shows that this inclusion cannot be generalized to the classes $P_k(\text{SPARSE})$.

**Theorem 4.5.** For all $k \geq 1$, $P(\text{CLOSE}) \supseteq P_k(\text{SPARSE})$.

**Theorem 4.5** has the following corollaries immediately.

**Corollary 4.6.** For all $h, k \geq 1$, $P_{k+1}(\text{SPARSE}) \supseteq P_{k+1}(\text{SPARSE})$.

**Corollary 4.7.** For all $h, k \geq 1$, $P_{k+1}(\text{SPARSE}) \supseteq P_{k+1}(\text{SPARSE})$.

While the hierarchy of reduction classes defined by bounded conjunctive reducibilities collapses, the class $P_{\mu}(\text{SPARSE})$ does not collapse to this hierarchy. In fact, it is not included in $P_{\mu}(\text{SPARSE})$ for any $k > 0$.

**Theorem 4.8.** For all $k \geq 1$, $P_{\mu}(\text{SPARSE}) \supseteq P_{\mu}(\text{SPARSE})$.

**Corollary 4.9.** For all $k \geq 1$, $P_{\mu}(\text{SPARSE}) \supseteq P_{\mu}(\text{SPARSE})$.

**Corollary 4.10.** $P_{\mu}(\text{SPARSE}) \supseteq P_{\mu}(\text{SPARSE})$.

The next result shows that the converse of Theorem 4.8 holds.

**Theorem 4.11.** $P_{\mu}(\text{SPARSE}) \supseteq P_{\mu}(\text{SPARSE})$.

**Corollary 4.12.** $P_{\mu}(\text{SPARSE}) \supseteq P_{\mu}(\text{SPARSE})$.

Finally we compare the class $P_{\mu}(\text{SPARSE})$ with the hierarchy $P_k(\text{SPARSE})$, $k > 0$. From Theorem 4.8 and the fact that $P_k(\text{SPARSE}) \subseteq P_{\mu}(\text{SPARSE})$, we obtain the following relation.

**Corollary 4.13.** For all $k > 0$, $P_{\mu}(\text{SPARSE}) \supseteq P_{\mu}(\text{SPARSE})$.

In addition, we can prove that

**Theorem 4.14.** $P_{\mu}(\text{SPARSE}) \supseteq P_{\mu}(\text{SPARSE})$.

All of the above results are proved by an inductive application of the pigeonhole principle in the form similar to that of Theorem 3.2 in the Appendix. In particular, the proofs for Theorems 4.1, 4.5 and 4.8 are very similar to that proof. The proofs for Theorems 4.10 and 4.14 are a little different using different forms of the pigeonhole principle. Due to the space limit, we omit all these proofs. The reader is referred to Ko [15] for complete proofs.

There are a number of questions left open concerning the relationship between the reduction classes defined by conjunctive, disjunctive and bounded truth-table reducibilities. It appears that the technique of counting arguments used in our proofs in this section is not strong
enough to solve these questions. We list them in the following.

(1) Is it true that for each $k > 1$, $P_{\text{tt}}(\text{SPARSE}) \not\subseteq P_{\text{tt}}(\text{SPARSE})$?

(2) Is it true that for each $k > 1$, $P_{\text{tt}}(\text{SPARSE}) \not\subseteq P_{\text{tt}}(\text{SPARSE})$?

(3) Is it true that $P_{\text{tt}}(\text{SPARSE}) \not\subseteq P_{\text{tt}}(\text{SPARSE})$?

(4) Is it true that $P_{\text{tt}}(\text{SPARSE}) \neq P_{\text{tt}}(\text{SPARSE})$?

As we mentioned above, most results in this section use the same proof technique involving an inductive use of the pigeonhole principle. This proof technique does not seem to be strong enough for the above questions. Rather, some stronger form of the pigeonhole principle seems in need. We discuss in the Appendix one such case involving question (2) above.

5. Sets Reducible to Tally Sets

Recall that a set $T$ is tally if $T \subseteq \{0\}^*$, and the class TALLY is the class of all tally sets. It has been shown in Book and Ko [8] that $P_{\text{tt}}(\text{TALLY}) = P_{\text{tt}}(\text{TALLY})$, and hence $P_{\text{tt}}(\text{TALLY})$ is properly included in $P_{\text{tt}}(\text{SPARSE})$. This result indicates that tally sets are too simple to be used to distinguish $(k + 1)$-tt-reducibilities from $k$-tt-reducibilities. In this section, we show that $P_{\text{tt}}(\text{TALLY})$ is properly included in $P_{\text{tt}}(\text{TALLY})$ and $P_{\text{tt}}(\text{TALLY})$ and $P_{\text{tt}}(\text{TALLY})$ are properly included in $P_{\text{tt}}(\text{TALLY})$.

First, we observe that $P_{\text{tt}}(\text{TALLY}) \subseteq P_{\text{tt}}(\text{TALLY}) \subseteq P_{\text{tt}}(\text{TALLY})$ and that $P_{\text{tt}}(\text{TALLY}) \subseteq P_{\text{tt}}(\text{TALLY}) \subseteq P_{\text{tt}}(\text{TALLY})$. Therefore, by the result $P_{\text{tt}}(\text{TALLY}) = P_{\text{tt}}(\text{TALLY})$ of [8], we know that all these classes are equivalent.

**Proposition 5.1.** $P_{\text{tt}}(\text{TALLY}) = P_{\text{tt}}(\text{TALLY}) = P_{\text{tt}}(\text{TALLY}) = P_{\text{tt}}(\text{TALLY})$.

Next, we consider classes $P_{\text{tt}}(\text{TALLY})$ and $P_{\text{tt}}(\text{TALLY})$. Note that, for any sets $A$ and $B$, $A \leq^P_\text{tt} B$ if and only if $A \leq^P_\text{tt} B$. If $B$ is a tally set, then we can replace $B$ by $B' = (0^*)^* + B$ and get $A \leq^P_\text{tt} B$ if and only if $A \leq^P_\text{tt} B'$. Therefore, we have the following result.

**Proposition 5.2.** For any set $A$, $A \leq^P_\text{tt} P_{\text{tt}}(\text{TALLY})$ if and only if $A \leq^P_\text{tt} P_{\text{tt}}(\text{TALLY})$.

Thus, to distinguish the classes $P_{\text{tt}}(\text{TALLY})$ and $P_{\text{tt}}(\text{TALLY})$, we need only to show that the class $P_{\text{tt}}(\text{TALLY})$ is not closed under complement. Furthermore, this would imply that $P_{\text{tt}}(\text{TALLY})$ is not equivalent to either $P_{\text{tt}}(\text{TALLY})$ or $P_{\text{tt}}(\text{TALLY})$, since the latter two classes are both closed under complement.

We first note that sparse sets of a special type are $\leq^P_\text{tt}$-reducible to tally sets.

**Lemma 5.3.** Assume that $A$ is a sparse set such that for each $n > 0$, $\|A \cap \Sigma^n\| \leq 1$. Then, $A \leq^P_\text{tt} P_{\text{tt}}(\text{TALLY})$.

**Proof.** Let $T = \{0^{n \cdot k}b\}^*$ there exists an $x \in A \cap \Sigma^n$ such that the $i$th bit of $x$ equals to $b$. Define a function $f$ that maps each $x \in \Sigma^*$ to the set $\{0^{n \cdot k}x\}^* \leq_i \leq n$; and $x_i$ is the $i$th bit of $x$. Note that $f$ is polynomial-time computable and $x \in A \Rightarrow f(x) \subseteq T$. Thus, $f \leq^P_\text{tt}$-reduces $A$ to $T$. □

From the above lemma and the observation of a stronger form of Theorem 4.10 that there exists a sparse set $A$ such that for each $n$, $\|A \cap \Sigma^n\| \leq 1$ and $A \not\subseteq \text{SPARSE}$ for any sparse set $S$ (this is indeed the form proved in Ko [15]), we find a sparse set $A$ such that $A \leq^P_\text{tt} P_{\text{tt}}(\text{TALLY})$ and $A \not\subseteq \text{SPARSE}$ for any tally set $T$. So, we get the following relations.

**Corollary 5.4.** (a) $P_{\text{tt}}(\text{TALLY}) \not\subseteq P_{\text{tt}}(\text{TALLY})$; $P_{\text{tt}}(\text{TALLY}) \not\subseteq P_{\text{tt}}(\text{TALLY})$.

(b) $P_{\text{tt}}(\text{TALLY}) \not\subseteq P_{\text{tt}}(\text{TALLY}) \subseteq P_{\text{tt}}(\text{TALLY})$.

(c) $P_{\text{tt}}(\text{TALLY}) \not\subseteq P_{\text{tt}}(\text{TALLY}) \subseteq P_{\text{tt}}(\text{TALLY})$.

The above relations among the reduction classes defined by tally sets are quite different from that among the reduction classes defined by sparse sets. To understand why this happens, we reexamine the reducibility relations between sparse sets and tally sets. The best equivalence result between sparse sets and tally sets is found by Hartmanis [11], who proved that for every sparse set $S$, there is a tally set $T$ such that $S$ is polynomial-time nondeterministic Turing equivalent to $T$ ($S \equiv^P_{\text{ND}} T$). Long [20] showed that this is probably the strongest type of reducibility with which an equivalence result holds: there is a sparse set $S$ such that for all tally sets $T$, $S$ is not polynomial-time strong nondeterministic equivalent to $T$ ($S \not\equiv^P_{\text{ND}} T$). We also observe that the proof for Hartmanis's result does not work for polynomial-time nondeterministic truth-table equivalence (in the sense of Book, Long, and Selman [9]). For the one-way reducibility, Book and Ko [8] showed that each sparse set is $\leq^P_\text{tt}$-reducible to a tally set ($P_{\text{tt}}(\text{SPARSE}) = P_{\text{tt}}(\text{TALLY})$) but there exists a sparse set which is not $\leq^P_\text{tt}$-reducible to any tally set ($\text{SPARSE} \not\subseteq P_{\text{tt}}(\text{TALLY})$). Note that Lemma 5.3 and Corollary 5.4 above actually proved another negative relation between sparse sets and tally sets. Namely, there exists a sparse set which is not $\leq^P_\text{tt}$-reducible to any tally set. Let co-SPARSE be the class of all co-sparse sets.

**Corollary 5.5.** (a) $\text{SPARSE} \not\subseteq P_{\text{tt}}(\text{TALLY})$.

(b) $\text{co-SPARSE} \not\subseteq P_{\text{tt}}(\text{TALLY})$.

It seems an interesting question to determine whether there exists a sparse set which is not $\leq^P_\text{tt}$-reducible to any
A tally set.

Acknowledgment

The author thanks Professors Ronald Book and Richard Beigel for their suggestions and encouragement.

References

6. R. Beigel, Bounded queries to SAT and the boolean hierarchy, preprint.

Appendix

In this appendix, we give sketches of the proofs for Theorems 3.2 and 3.4. In addition, we make a combinatorial conjecture which would imply an affirmative answer to question (2) of Section 4.

Notation. Before we give the proofs, we need some more notation about bounded query machines. First consider binary trees of height $k$. Let $T$ be a complete binary tree of height $k$. For each $s \in \Sigma^k$ of length $k < k - 1$, we write $T(s)$ to denote the $s$th (under the lexicographic order in $\Sigma^k$) 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 $k < k - 1$, $T(s)$ has two children: left child $T(s0)$ and right child $T(s1)$. Now a $\leq^p_T$-reduction machine, or a $k$-query adaptive machine, may be 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. More precisely, the function $f$ on $x$ outputs a list of $2^k$-1 strings $(x_1, x_0, x_1, x_0, \ldots, x_{k-1})$, with the interpretation that these $2^k-1$ strings form a binary tree $T_x$ of height $k$ such that $T_x(s) = x_s$ for all $s$ of length $k < k - 1$. The function $g$ on $x$ defines a truth-table with $k$ variables with the interpretation that the query machine accepts $x$ relative to oracle $B$ if the truth-table $g(x)$ evaluates to true on the $k$-tuple $(\chi_B(x_1), \chi_B(x_0), \ldots, \chi_B(x_{k-1}))$, where each bit $s_i$, $1 \leq i \leq k - 1$, is defined by $s_i = \chi_g(x_i)$ and $s_i = \chi_B(x_{i-1})$, for $1 < i < k - 1$.

Proof of Theorem 3.2. The proof technique of the theorem is a refinement of that of the separation result $P_{k+1}(\text{SPARSE}) \neq P_{k+1}^p(\text{SPARSE})$ of Book and Ko [8]. We only present a sketch.

Let $k \geq 2$ be fixed and let $m_k = 2^k-2$. We will define, for each set $A$, a set $L_k(A)$ such that $L_k(A) \leq_{\text{tt}} P_{k+1}$ $A$. Then, we will define a sparse set $A$ by a stage construction such that $L_k(A) \leq_{\text{tt}} P_{k+1}^p$ $A$ for all sparse sets $S$.

In order to describe the set $L_k(A)$, we define a specific $k$-query adaptive machine $N$ as follows. For any string $x$ of length $m_k$ for some $n \geq 1$, write $x$ as $a_{k-1}a_{k-2} \ldots a_0$, where each $a_i$, $1 \leq |s| \leq k - 1$, is of length $n$. The machine $N$ on input $x$ produces a tree $T_x$ of height $k$, where the query string of each node $T_x(s)$ is defined to be $T_x(s) = s_{i-1}$, if $1 \leq |s| \leq k - 1$, and
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 \( A \). From the definition we know that \( L_k(A) \subseteq L_k^N A \). We now construct a sparse set \( A \) such that for all sparse sets \( S \), it is not the case that \( L_k(A) \subseteq L_k^N S \).

The construction can be done by stages. First, we consider a fixed enumeration \( \{ M_j \} \) of all \( \leq m_k n \)-reduction machines. We consider two more subcases.

Case 1. The function \( f_1 \) is one-to-one on \( G \).

Then, we have strings \( x \neq y \) in \( G \) such that \( f_1(x) = f_1(y) \) and \( g_1(x) = g_1(y) \). So, \( x \) and \( y \) are both in \( L(M_{j_0}, S) \) or both in \( L(M_{j_1}, S) \). However, since \( x \neq y \), it is not hard to design a set \( B \) such that the trees \( T_x \) and \( T_y \) evaluate differently relative to \( B \) and \( x, y \in L_k(B) \Rightarrow y \notin L_k(B) \). This completes the construction for Case 1.

Case 2. The function \( f_1 \) is one-to-one on \( G \).

We consider two more subcases.

Subcase 2.1. There exist an integer \( r \), \( 1 \leq r \leq m_k \), and a string \( z \) such that the set \( H_r(z) = \{ x \in G \mid x \models z \} \) has size \( \| H_r(z) \| \geq 2^{m_k - 26n} \).

Then, it is not hard to see that there exist two strings \( u, w \in 2^n \) such that \( \| H_r(z) \cap \{ u \} \Sigma^m \| \geq 2^{m_k - 10m - 35n} \) and \( \| H_r(z) \cap \{ w \} \Sigma^m \| \geq 2^{m_k - 10m - 35n} \). 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 A.1.** The following holds for all \( j, 1 \leq j \leq m_k - 1 \): Assume that there exist a function \( f \) which yields, on input \( z \), a list of \( j \) strings \( \{ z_1, \ldots, z_j \} \), and a set \( G \subseteq \{ \} \Sigma^{m_k - j} \) for some string \( u \) of length \( (m_k - j - 1)n \), satisfying the following properties:

1. \( f \) is one-to-one on \( G \), and for all \( x \in G \) and all \( r, 1 \leq r \leq j, |x| \leq q(m_k n) \); and
2. there exist strings \( v \) and \( w \) of length \( n \) such that \( \| G \cap V_j \| \geq 2^{j - (m_k - j - 1)n} \) and \( \| G \cap W_j \| \geq 2^{j - (m_k - j - 1)n} \), where \( V_j = \{ v \} \Sigma^n \) and \( W_j = \{ w \} \Sigma^n \).

Then, there exists a set \( C \) such that

3. \( f \) is one-to-one on \( C \), and for all \( x \in C \) and all \( j \)-tt-conditions \( t \), there exists an \( m_k \)-tt-condition \( t \) such that \( \| C \cap G \cap \{ z \} \Sigma^m \| > 0 \) and \( t(x_{z1}, \ldots, x_{zmk}) = 0 \).

By the definition of \( t_{z1}, \ldots, t_{zmk} \), we have \( t_{z1}(x_{z1}, \ldots, x_{zmk}) = 0 \). Therefore, at least one \( x_{zk} \) in \( G \). We can define the set \( C = A \cup \{ \} \Sigma^m \).

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

Then, for every set \( S \subseteq \{ \} \Sigma^m \), there are at most \( \sum_{r=1}^{m_k} \sum_{x \in G} \| H_r(z) \| \cdot p_k(q(m_k n)) \cdot 2^{m_k - 26n} \) strings in \( G \) having at least one \( x_{zk} \) in \( S \).

Therefore, at least one \( x \) in \( G \) has all \( x_{zk} \) in \( S \). Let \( 0^m \) be in set \( B \) if \( t(0, \ldots, 0) = 0 \). Without loss of generality, assume that \( t(0, \ldots, 0) = 0 \). Then, for any sparse set \( S \subseteq \{ \} \Sigma^m \),

\[ \]
all \( x \) in \( G \) are in \( L_k(B) \), but at least one \( x \in G \) has the property that \( (\chi_S(x_1), \ldots, \chi_S(x_m)) = t(0, \ldots, 0) = 0 \) and hence \( e \notin L(M_{ij}, S) \). Let \( A_4 = A_{i-1} \cup B \). This completes stage \( e \) of the construction.

It remains to prove Lemma A.1. The lemma can be proved by induction on \( j \). When \( j = 1 \), \( f(x) = x_1 \) is one-to-one on \( G \). Therefore, the sizes of sets \( G \cap V_1 \) and \( G \cap W_1 \) are bigger than that of \( S \)'s \( m \), and there exist \( \phi = xw' \in G \cap V_1 \) and \( \psi = wy' \in G \cap W_1 \), both in \( S \). On the other hand, by choosing \( C \) to contain exactly one of \( sv \) and \( sw \) (for some appropriate string \( s \)), we can make \( z \in L_k(C) \) \( \leftrightarrow y \notin L_k(C) \). So, one of \( z \) and \( y \) satisfies the condition (4) of the lemma.

The inductive step is proved by considering subcases depending on whether sets of the form \( H_e(x) = \{ x \in G \mid x_e = x \} \) have large size or not. The proof is similar to the Subcases 2.1 and 2.2 above. We omit it here. \( \square \)

**Proof of Theorem 3.4.** For each set \( A \) and each \( m \geq 1 \), let
\[
L_m(A) = \{ u_{t+1} \cdots u_m \mid u_{t+1} = \cdots = u_m = \eta; \text{ the number of strings in the following list that are in } A \text{ is odd: } f^{m+1}(u_{t+1}), u_1 \cdots \eta \}.
\]
Then it is clear that \( L_m(A) \subseteq \mathbb{P} \). For each \( k \geq 2 \), let \( t_k = 3 \cdot 2^{k^4 - 2} \). We will construct a sparse set \( A \) such that for every sparse set \( S \), it is not the case that \( L_{t_k}(A) \subseteq L_k(S) \). This will prove the theorem.

The construction proceeds by stages. Recall that each \( \leq \mathbb{P} \)-reduction machine is defined by two polynomial-time functions \( f \) and \( g \), with the interpretation that \( f(x) \) generates a binary tree of height \( k \) and \( g(x) \) is a \( k \)-tt-condition evaluator. Assume standard enumerations \( \{ f_i \} \) and \( \{ g_j \} \) for such functions. Then we can enumerate \( \leq \mathbb{P} \)-reduction machines as \( \{ N_{ij} \} \), where \( N_{ij} \) is defined by the \( i \)th polynomial-time height-\( k \) tree generator \( f_i \) and the \( j \)th polynomial-time \( k \)-tt-condition evaluator \( g_j \). We fix the integer \( k \geq 2 \). In stage \( (i, j, h) \), we will find an integer \( n \) such that \( L_{t_k}(A) \not\subseteq \mathbb{P} \). We choose a constant \( \delta = \frac{1}{8t_k} \), and let \( A_0 = \emptyset \), \( n_0 = 1 \).

**Stage** \( e = (i, j, h) \). Assume that the runtime of the machine \( N_{ij} \) is bounded by a polynomial \( p \). We choose an integer \( n \) such that \( n > t_k n_{e-1} \) (so that the conditions established in earlier stages are not affected by the construction in the current stage), \( \delta n > 2t_k \), and \( p_n(p^k((t_k - 1)n)) < 2^{k^4} \) (so that the number of strings \( y \) in \( S \in \mathbb{P} \) that are queried by \( N_{ij} \) on some input of length \( t_k n \) is at most \( 2^{k^4} \)). Let \( n = n_0 \).

Consider the \( k \)-tt-conditions \( g_i(x) \) for all \( x \in \mathbb{P} \). There are only \( 2^{k^4} \) different \( k \)-tt-conditions. So there must be at least one \( k \)-tt-condition \( t \) such that \( G_t = \{ x \in \mathbb{P} \mid g_i(x) = t \} \) has size \( \geq 2^{(k^4 - 1)n} \). We fix such a \( k \)-tt-condition \( t \), and let \( G = G_t \).

Next, we will construct set \( A_e \) by an inductive construction. We state this inductive construction in a separate lemma.

For each \( j \geq 2 \), define \( \beta_2 \) 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 \( \delta = 1/(8t_k) < 1/(2\beta_k) \).

**Lemma A.2.** The following holds for all \( j, 2 \leq j \leq k \). Assume that there exist a \( \leq \mathbb{P} \)-reduction machine \( N_{j+1} \), defined by a height-\( j \) tree generator \( f \) and a fixed \( j \)-tt-condition \( t \), and a set \( G \subseteq \{ u \cdots u_m \} \Sigma^k \) for some \( u_1 \cdots u_m \in \Sigma^k \) \( (0 \leq m \leq t_k - 1) \), such that
1. the runtime of machine \( N_{j+1} \) is bounded by polynomial \( p \), and
2. \( ||G|| \geq 2(\beta_j + 1) \cdot 3^j \) for some \( \gamma \geq 0 \) and some \( \alpha \), \( 0 \leq \alpha < 2 \) and \( \beta_j + 1 \).

Then there exists a set \( C \) such that
3. \( C \) is in \( \mathbb{P} \) of the form \( u_1 \cdots u_m v_{m+1} \cdots v_{m+j} 0^{(k^4 - 1)n}(n+1) \), where \( 0 \leq i \leq t_k - 1 \) and \( v_{m+1} \cdots v_{m+j} \in \Sigma^k \),
4. \( ||C|| \leq 2^j \), and
5. for every set \( S \in \mathbb{P} \), the set \( D = \{ w \in G \mid w \notin L(N_{j+1}, S) \} \) has size \( ||D|| \geq 2^{k^4 - 1} \). We observe that machine \( N_{j+1} \) and set \( G \) satisfy the assumptions of Lemma A.2. Note that here we have \( m = 0, \gamma = 0 \) and \( \alpha = 1 \). So, we may apply Lemma A.2 to obtain a set \( C \) of strings of the form \( v_1 \cdots v_m 0^{(k^4 - 1)n}(n+1) \) such that \( ||C|| \leq 2^j \) and such that for every set \( S \in \mathbb{P} \), there exists a string \( w \in G \) such that \( w \notin L(N_{j+1}, S) \). Since \( w \in G \) implies that \( w \notin L(N_{j+1}, S) \) \( \iff w \notin L(N_{j+1}, S) \) for every set \( S \in \mathbb{P} \), we have a witness to the requirement \( L(N_{j+1}, S) \not\subseteq L(N_{j+1}, S) \). We let \( A_e = A_{e-1} \cup C \). This completes Stage \( e \).

It remains to prove Lemma A.2. We prove the lemma by induction on \( j = 2, \ldots, k \). The initial case of \( j = 2 \) involves a tedious case analysis, we omit the details (see [16] for the complete proof). For the inductive step, we assume that \( k > 2 \) and \( 2 < j \leq k \). Note that \( \ell_{j-1} = \ell_j/2 \). Recall that for each \( x, f \), there is a list of \( 2^{k^4} \) strings \( \{ x_1 \cdots x_{2^{k^4}} \} \), in particular, let \( x_1 \) be the root of the tree. Define \( H(x) = \{ x \in G \mid x_1 = x \} \). We consider two cases.

**Case 1.** There exists a string \( x \) such that \( ||H(x)|| \geq 2 \ell_j \cdot 2 \ell_j \cdot 2 \ell_j \).

By a simple induction, we can find strings \( u_{m+1} \),

188
,..., \mu_{m+q-1}, \mu_{m+q}, and \mu_{m+q} in \Sigma^m, 1 \leq q \leq t_k - m - \gamma - \ell_j/2, such that both \( H(z) \cap U_{m+q} \) and \( H(z) \cap V_{m+q} \) have size \( \geq 2^{(t_k/2-1-\gamma-\ell_j-1)j} \), where \( U_{m+q} = \{ \{ w_1 \cdots \mu_{m+q-1} \mu_{m+q} \} \Sigma^{(t_k-\gamma-1)} \} \) and \( V_{m+q} = \{ \{ w_1 \cdots \mu_{m+q-1} \mu_{m+q} \} \Sigma^{(t_k-\gamma-1)} \} \).

**Step A.** Define \( f_0 \) to be the function which maps each string \( x \) to the left subtree of \( f(x) \) and \( q_0 \) to be the \((j-1)-it\)-condition defined by \( q_0(b_1, \ldots, b_{j-1}) = t_l(0, b_1, \ldots, b_{j-1}) \). Without loss of generality, we assume that the machine \( N_{f_0,q_0} \) defined by \( f_0 \) and \( q_0 \) has runtime bounded by \( p \) as well. Also let \( G_0 = H(x) \cap U_{m+q} \). Note that \( \| G_0 \| \geq 2^{(t_k-\gamma-1-\alpha-\beta_j-1)j} \) implies \( \alpha + \beta_j + 1 \leq \beta_k - \beta_j + 2 \), and since \( \beta_k - \beta_j + 2 \) is a particular condition (5) states that for each \( z \in \Sigma^m \), \( 0 \leq q \leq t_k - m - \gamma - \ell_j/2 \), and \( \beta \leq 0 \), we can apply the inductive hypothesis to machine \( N_{f_0,q_0} \) and set \( G_0 \). That is, we can find a set \( C_0 \) satisfying conditions (3)-(5), with respect to strings \( u_1, \ldots, u_{m+q} \), and parameter \( j \). In particular, condition (5) states that for each \( S \in \text{SPARSE}_k \), \( \| D_0 \| \geq 2^{(t_k-\gamma-1-\alpha-\beta_j-1)j} \) if \( \gamma \geq 1 \), and \( D_0 \neq \emptyset \) if \( \gamma = 0 \), where \( D_0 = \{ w \in G_0 \mid w \in L(N_{f_0,q_0}, S) \} \). (Note that \( \beta_j = \beta_k - 3 + 2 \).

**Step B.** Define \( f_1, f_l, f_2 \) and \( G \) similarly: \( f_1 \) is the right subtree of \( f(x) \), \( t_l(b_1, \ldots, b_{j-1}) = t_l(0, b_1, \ldots, b_{j-1}) \), and \( G_l = H(z) \cap V_{m+q} \). By the inductive hypothesis, we find a set \( C_1 \) satisfying conditions (3)-(5), with respect to strings \( u_1, \ldots, u_{m+q} \) and parameter \( j \). In particular, condition (5) states that for each \( S \in \text{SPARSE}_k \), \( \| D_1 \| \geq 2^{(t_k-\gamma-1-\alpha-\beta_j-1)j} \) if \( \gamma \geq 1 \), and \( D_1 \neq \emptyset \) if \( \gamma = 0 \), where \( D_1 = \{ w \in G_l \mid w \in L(N_{f_1,q_1}, S) \} \).

Let \( C = C_0 \cup C_1 \) and observe that \( C \) satisfies conditions (3) and (4) (with respect to strings \( u_1, \ldots, u_m \) and parameter \( j \)). Finally, we prove condition (5).

Let \( S \in \text{SPARSE}_k \), and recall that \( D = \{ w \in G \mid w \in L(N_{f_1,q_1}, S) \} \) and \( D_k = \{ w \in G_l \mid w \in L(N_{f_1,q_1}, S) \} \). First consider the case that \( \gamma \geq 1 \) and \( \emptyset \neq S \). Then, by the definition of \( C_0 \), \( \| D_0 \| \geq 2^{(t_k-\gamma-1-\alpha-\beta_j-1)j} \) where \( \sum_{x \in C_0} = \{ w \in G_0 \mid w \in L(N_{f_1,q_1}, S) \} \). Note that for all \( x \in \Sigma \), \( H(x) \cap U_{m+q} \in \Sigma^{(t_k-\gamma-1)} \) and \( \sum_{x \in C_0} = \{ w \in G_0 \mid w \in L(N_{f_1,q_1}, S) \} \) (because all strings in \( C_0 \) begin with different prefixes from those in \( C_0 \), and \( \sum_{x \in C_0} \neq \emptyset \)).

Let \( C = C_0 \cup C_1 \) and observe that \( C \) satisfies conditions (3) and (5) (with respect to strings \( u_1, \ldots, u_{m+q} \) and parameter \( j \)).

Then, we define function \( f_2 \) and \((j-1)-it\)-condition \( q_2 \) as in Case 1. Let \( \gamma_0 = \gamma + \ell_j/2 \), and \( \gamma_0 = \gamma + \ell_j/2 \). Then, by the inductive hypothesis, there exists a set \( C_2 \) 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 \( S \in \text{SPARSE}_k \), \( \| D_2 \| \geq 2^{(t_k-\gamma_0-1-\alpha-\beta_j-1)j} \), where \( D_2 = \{ w \in G \mid w \in L(N_{f_2,q_2}, S) \} \).

Let \( C = C_0 \cup C_1 \cup C_2 \) satisfies the require-
A Combinatorial Conjecture Related to Question (2) of Section 4.

Let $M$ be an $m \times n$ 0-1 matrix. We say $M$ is $k$-column dependent if there exist $j_0, j_1, \ldots, j_k$, $1 \leq j_r \leq n$ for all $r = 0, 1, \ldots, k$, such that for every $i$, $1 \leq i \leq m$, if $M(i, j_0) = 1$ then $M(i, j_r) = 1$ for some $r = 1, \ldots, k$.

Conjecture. For every integer $k$, there exists an integer $n$ such that every $n^k \times 2^n$ 0-1 matrix $M$ is $n$-column dependent.

Theorem A.3. If the above conjecture holds, then $P_{k\text{-det}}(\text{SPARSE}) \subseteq P_{k\text{-det}}(\text{SPARSE})$. We let $\text{SPARSE}_{k\text{-det}}$ be the class of all sparse sets $S$ such that $|\{x \in S \mid |x| \leq n\}| \leq n^k$. We construct $A$ in stages. At stage $(i, h)$, we satisfy condition (*) for function $f_i$ and for all $S \in \text{SPARSE}_{k\text{-det}}$.

We get $D(A) \cap \Sigma^n = \{x_1, x_2, \ldots, x_n\}$. However, $x_j \notin D(A)$. Suppose, by way of contradiction, that condition (*) does not hold for some set $S \in \text{SPARSE}_{k\text{-det}}$. Then, $x_1, x_2, \ldots, x_n \notin D(A)$ implies $\cup_{i=1}^n f_i(x_i) \subseteq S$ and so $f_i(x_j) \subseteq \cup_{i=1}^n f_i(x_i) \subseteq S$ and $x_j \notin D(A)$.

Note that the above construction of $A$ is nonrecursive because Cases 1 and 2 cannot be distinguished by a recursive property. However, if the conjecture has a stronger recursive form in the sense that the integer $n$ can be found as a recursive image of integer $k$ (i.e., $n \leq f(k)$ for some recursive $f$) then $A$ can be constructed recursively. We need only, in stage $(i, h)$, search for integer $n > n_{i-1} + 1$ which satisfies Case 1 up to $f(1, h)$.

Fig. 1. Relations between classes $P_{k\text{-det}}(\text{SPARSE})$ and $P_{k\text{-det}}(\text{SPARSE})$. In the above, $A \rightarrow B$ means that $A \subseteq B$, $A \rightarrow | \rightarrow B$ means that $A \not\subseteq B$. 

We consider two cases.

Case 1. There exists an integer $n > n_{i-1} + 1$ such that $\|\cup_{|x|=n} f_i(x)\| > n^{k_A}$.

Then, we let $n = n_a$ and let $0^{n_a+1} \in A$. Note that this makes $D(A) \cap \Sigma^n = \Sigma^n$. But there must be at least one $x \in \Sigma^n$ such that $f_i(x) \not\subseteq S$ (otherwise $\cup_{|x|=n_a} f_i(x) \subseteq S \cap \Sigma^n$ and has size at most $n^{k_A}$).

Case 2. For all integers $n > n_{i-1} + 1$ the size of $\cup_{|x|=n} f_i(x)$ is $\leq n^{k_A}$.

From the conjecture, we choose an integer $n > n_{i-1} + 1$ such that every 0-1 matrix of dimension $n^{k_A} \times 2^n$ is $n$-column dependent. Let $n = n_a$. We note that $\|\cup_{|x|=n_a} f_i(x)\| \leq n^{k_A}$. Fix an enumeration of the set $\cup_{|x|=n_a} f_i(x)$: $y_1, y_2, \ldots, y_{n^{k_A}}$, and let $y_j, 1 \leq j \leq 2^n$, be the $j$th element in $\Sigma^n$. Define a particular $n^{k_A} \times 2^n$ matrix $M$ as follows: $M(m, j) = 1$ if $y_j \in f_i(x_j)$. Then there exist $j_0, j_1, \ldots, j_n$ such that for every $i, 1 \leq i \leq n^{k_A}$, if $M(i, j_0) = 1$ then $M(i, j_r) = 1$ for some $r = 1, \ldots, n$. We let $x_{j_1}, x_{j_2}, \ldots, x_{j_n}$ be in $A$. 

\[ (*) \]

This would allow us to conclude that $P_{k\text{-det}}(\text{SPARSE}) \not\subseteq P_{k\text{-det}}(\text{SPARSE})$. We let $\text{SPARSE}_{k\text{-det}}$ be the class of all sparse sets $S$ such that $|\{x \in S \mid |x| \leq n\}| \leq n^k$. We construct $A$ in stages. At stage $(i, h)$, we satisfy condition (*) for function $f_i$ and for all $S \in \text{SPARSE}_{k\text{-det}}$.

At stage $(i, h)$, assume that $n_{i-1}$ is chosen by stage $(i-1)$. Let $f_i$ be computable in time $n^k$. We consider two cases.

Case 1. There exists an integer $n > n_{i-1} + 1$ such that $\|\cup_{|x|=n} f_i(x)\| > n^{k_A}$.

Then, we let $n = n_a$ and let $0^{n_a+1} \in A$. Note that this makes $D(A) \cap \Sigma^n = \Sigma^n$. But there must be at least one $x \in \Sigma^n$ such that $f_i(x) \not\subseteq S$ (otherwise $\cup_{|x|=n_a} f_i(x) \subseteq S \cap \Sigma^n$ and has size at most $n^{k_A}$).

Case 2. For all integers $n > n_{i-1} + 1$ the size of $\cup_{|x|=n} f_i(x)$ is $\leq n^{k_A}$.

From the conjecture, we choose an integer $n > n_{i-1} + 1$ such that every 0-1 matrix of dimension $n^{k_A} \times 2^n$ is $n$-column dependent. Let $n = n_a$. We note that $\|\cup_{|x|=n_a} f_i(x)\| \leq n^{k_A}$. Fix an enumeration of the set $\cup_{|x|=n_a} f_i(x)$: $y_1, y_2, \ldots, y_{n^{k_A}}$, and let $y_j, 1 \leq j \leq 2^n$, be the $j$th element in $\Sigma^n$. Define a particular $n^{k_A} \times 2^n$ matrix $M$ as follows: $M(m, j) = 1$ if $y_j \in f_i(x_j)$. Then there exist $j_0, j_1, \ldots, j_n$ such that for every $i, 1 \leq i \leq n^{k_A}$, if $M(i, j_0) = 1$ then $M(i, j_r) = 1$ for some $r = 1, \ldots, n$. We let $x_{j_1}, x_{j_2}, \ldots, x_{j_n}$ be in $A$. 

\[ (*) \]
Fig. 2. Relations between classes $P_k(t_{ttt}(\text{SPARSE}))$, $P_k(t_{ttt}(\text{SPARSE}))$, and $P_{m(t_{ttt}(\text{SPARSE}))}$.

Fig. 3. Relations between classes $P_{r(TALLY)}$.