Resource-Bounded Kolmogorov Complexity Revisited

Harry Buhrman*
CWI
PO Box 94079
1090 GB Amsterdam
The Netherlands

Lance Fortnow†
University of Chicago
Department of Computer Science
1100 E. 58th St.
Chicago, IL 60637 USA

Sophie Laplante‡
LRI
Université Paris-Sud, Bâtiment 490
91405 Orsay, France

May 25, 2001

Abstract

We take a fresh look at **CD** complexity, where $\mathbf{CD}^t(x)$ is the size of the smallest program that distinguishes x from all other strings in time $t(|x|)$. We also look at **CND** complexity, a new nondeterministic variant of **CD** complexity, and time-bounded Kolmogorov complexity, denoted by **C** complexity.

We show several results relating time-bounded **C**, **CD** and **CND** complexity and their applications to a variety of questions in computational complexity theory, including:

- Showing how to approximate the size of a set using **CD** complexity without using the random string as needed in Sipser's earlier proof of a similar result. Also we give a new simpler proof of this result of Sipser's.
- Improving these bounds for almost all strings, using extractors.
- A proof of the Valiant-Vazirani lemma directly from Sipser's earlier **CD** lemma.
- A relativized lower bound for **CND** complexity.
- Exact characterizations of equivalences between **C**, **CD** and **CND** complexity.
- Showing that satisfying assignments of a satisfiable Boolean formula can be enumerated in time polynomial in the size of the output if and only if a unique assignment can be found quickly. This answers an open question of Papadimitriou.
- A new Kolmogorov complexity based proof that $\mathbf{BPP} \subseteq \Sigma_2^p$.
- New Kolmogorov complexity based constructions of the following relativized worlds:
  - There exists an infinite set in **P** with no sparse infinite **NP** subsets.
  - **EXP** = **NEXP** but there exists a **NEXP** machine whose accepting paths cannot be found in exponential time.
  - Satisfying assignments cannot be found with nonadaptive queries to **SAT**.

---

*E-mail: buhrman@cwi.nl.

Part of this research was done while visiting The University of Chicago. Partially supported by the Dutch foundation for scientific research (NWO) through NFI Project ALADDIN, under contract number NF 62-376.

†Email: fortnow@cs.uchicago.edu.

Supported in part by NSF grant CCR 92-53582.

‡Email: laplante@lri.fr. Work done while at the University of Chicago.

1 Introduction

Originally designed to measure the randomness of strings, Kolmogorov complexity has become an important tool in computability and complexity theory. A simple lower bound showing that there exist random strings of every length has had several important applications (see [LV97, Chapter 6]).

Early in the history of computational complexity theory, many people naturally looked at resource-bounded versions of Kolmogorov complexity. This line of research was initially fruitful and led to some interesting results. In particular, Sipser [Sip83] invented a new variation of resource-bounded complexity, **CD** complexity, where one considers the size of the smallest program that accepts one specific string and no others. Sipser used **CD** complexity for the first proof that **BPP** is contained in the polynomial-time hierarchy.

Complexity theory has marched on for the past two decades, but resource-bounded Kolmogorov complexity has seen little interest. Now that computational complexity theory has matured a bit, we ought to look back at resource-bounded Kolmogorov complexity and see what new results and applications we can draw from it.

First, we use algebraic techniques to give a new upper bound lemma for **CD** complexity without the additional advice required of Sipser's lemma [Sip83]. With this lemma, we can approximately measure the size of a set using **CD** complexity.

We obtain better bounds on **CD** complexity using extractor graphs. These graphs are usually used for derandomization. However these improved bounds only apply to most of the strings.

We also give a new simpler proof of Sipser's Lemma and show how it implies the important Valiant-Vazirani lemma [VV85] that randomly isolates satisfying assignments. Surprisingly, Sipser's paper predates the result of Valiant and Vazirani.

We define **CND** complexity, a variation of **CD** complexity where we allow nondeterministic computation. We prove a lower bound for **CND** complexity where we show that there exists an infinite set A such that every string in A has high **CND** complexity even if we allow access to A as an oracle. We use this lemma to prove some negative results on nondeterministic search vs. deterministic decision.

Once we have these tools in place, we use them to unify several important theorems in complexity theory. We answer an open question of Papadimitriou [Pap96] characterizing exactly when the set of satisfying assignments of a formula can be enumerated in output polynomial time. We also give straightforward proofs that **BPP** is in Σ_2^p (first proven by Gács (see [Sip83])) and create relativized worlds where assignments to **SAT** cannot be found with non adaptive queries to **SAT** (first proven by Buhrman and Thierauf [BT96]), and where **EXP** = **NEXP** but there exists a **NEXP** machine whose accepting paths cannot be found in exponential time (first proven by Impagliazzo and Tardos [IT89]).

These results in their original form require a great deal of time to fully understand the proof because either the ideas and/or technical details are quite complex. We show that by understanding resource-bounded Kolmogorov complexity, one can see full and complete proofs of these results without much additional effort. We also look at when polynomial-time **C**, **CD** and **CND** complexity coincide. We give a precise characterization of when we have equality of these measures, and some interesting consequences thereof.

2 Preliminaries

We use basic concepts and notation from computational complexity theory texts like Balcázar, Díaz, and Gabarró [BDG88] and Kolmogorov complexity from the excellent book by Li and Vitányi [LV97]. We use $|x|$ to represent the length of a string x and $\|A\|$ to represent the number of elements in the set A. $A^{=n}$ is the set of strings in A of length n. $[N]$ denotes the set of integers between 1 and N. All of the logarithms are base 2.

Formally, we define the Kolmogorov complexity function $\mathbf{C}_\phi(x|y)$ by $\mathbf{C}_\phi(x|y) = \min_p\{|p| : \phi(p,y) = x\}$. The exists a universal machine U such that for all ϕ there is a constant c such that for all x and y, $\mathbf{C}_U(x|y) \leq \mathbf{C}_\phi(x|y) + c$. We fix such a U and let $\mathbf{C}(x|y) = \mathbf{C}_U(x|y)$. We define unconditional Kolmogorov complexity by $\mathbf{C}(x) = \mathbf{C}(x|\epsilon)$.

A few basic facts about Kolmogorov complexity:

- The choice of U affects the Kolmogorov complexity by at most an additive constant.
- For some constant c, $\mathbf{C}(x) \leq |x| + c$ for every x.
- For every n and every y, there is an x such that $|x| = n$ and $\mathbf{C}(x|y) \geq n$.

We will also use time-bounded Kolmogorov complexity. Fix a fully time-constructible function $t(n) \geq n$. We define the $\mathbf{C}^t(x|y)$ complexity function as

$$\mathbf{C}^t(x|y) = \min_p\{|p| : U(p,y) = x \text{ and } U(p) \text{ runs in at most } t(|x|+|y|) \text{ steps}\}.$$

As before we let $\mathbf{C}^t(x) = \mathbf{C}^t(x|\epsilon)$. A different universal U may affect the complexity by at most a constant additive term and the time by a $\log(t)$ factor.

While the usual Kolmogorov complexity asks about the smallest program to *produce* a given string, we may also want to know about the smallest program to *distinguish* a string. While this difference affects the unbounded Kolmogorov complexity by only a constant it can make a difference for the time-bounded case. Sipser [Sip83] defined the distinguishing complexity $\mathbf{CD}^t$ by

$$\mathbf{CD}^t(x|y) = \min_p \left\{ |p| \quad : \quad \begin{array}{l} (1)\ U(p,x,y) \text{ accepts.} \\ (2)\ U(p,z,y) \text{ rejects for all } z \neq x. \\ (3)\ U(p,z,y) \text{ runs in at most } t(|z|+|y|) \text{ steps for all } z \in \Sigma^*. \end{array} \right\}$$

When the auxiliary input string y is the empty string, we write $\mathbf{CD}^t(x)$.

Fix a universal nondeterministic Turing machine U_n. We define the nondeterministic distinguishing complexity $\mathbf{CND}^t$ by

$$\mathbf{CND}^t(x|y) = \min_p \left\{ |p| \quad : \quad \begin{array}{l} (1)\ U_n(p,x,y) \text{ accepts.} \\ (2)\ U_n(p,z,y) \text{ rejects for all } z \neq x. \\ (3)\ U_n(p,z,y) \text{ runs in at most } t(|z|+|y|) \text{ steps for all } z \in \Sigma^*. \end{array} \right\}$$

In this definition, we mean that the nondeterministic Turing machine accepts or rejects in the usual sense of nondeterministic computation. Once again we let $\mathbf{CND}^t(x) = \mathbf{CND}^t(x|\epsilon)$.

We can also allow for relativized Kolmogorov complexity. For example, $\mathbf{CD}^{t,A}(x|y)$ is defined as above except that the universal machine U has access to A as an oracle.

One can distinguish a string by generating it then comparing it with the input, as stated in the following lemma.

Lemma 1 $\forall t\ \exists c\ \forall x, y : \mathbf{CD}^{ct\log t}(x \mid y) \leq \mathbf{C}^t(x \mid y) + c$

where c is a constant. Likewise, every deterministic computation is also a nondeterministic computation, hence the lemma that follows.

Lemma 2 $\forall t\ \exists c\ \forall x, y : \mathbf{CND}^{ct\log t}(x \mid y) \leq \mathbf{CD}^{t}(x \mid y) + c.$

In Section 7 we examine the consequences of the converses of these lemmas.

3 Approximating Sets with Distinguishing Complexity

In this section we derive a lemma that enables one to deterministically approximate the density of a set, using polynomial-time distinguishing complexity. The technique we use of considering values modulo a prime is reminiscent of the hashing via the division method (see [Knu98, p. 515]).

Lemma 3 *Let $S = \{x_1, \ldots, x_d\} \subseteq \{0, \ldots, 2^n - 1\}$. For all $x_i \in S$ and at least half of the primes $p \leq 4dn^2$, $x_i \not\equiv x_j \bmod p$ for all $j \neq i$.*

Proof: For each $x_i, x_j \in S$, $i \neq j$, it holds that for at most n different prime numbers p, $x_i \equiv x_j \bmod p$ by the Chinese Remainder Theorem. (Alternatively, $|x_i - x_j| < 2^n$, so it can not have more than n prime factors.) For x_i there are at most dn primes p such that $x_i \equiv x_j \bmod p$ for some $x_j \in S$. The Prime Number Theorem [Ing32] (see also [HW79]) states that for any m there are approximately $m/\ln(m) > m/\log(m)$ primes less than m. There are at least $4dn^2/\log(4dn^2) > 2dn$ primes less than $4dn^2$. So at least half of these primes p must have $x_i \not\equiv x_j \bmod p$ for all $j \neq i$. □

Lemma 4 *Let A be any set. For all strings $x \in A^{=n}$ it holds that $\mathbf{CD}^{p,A^{=n}}(x) \leq 2\log(\|A^{=n}\|) + O(\log(n))$ for some polynomial p.*

Proof: Fix n and let $S = A^{=n}$. Fix $x \in S$ and a prime p_x fulfilling the conditions of Lemma 3 for x.

The $\mathbf{CD}^{poly,A}$ program for x works as follows:

input y
If $y \notin A^{=n}$ then REJECT
else if $y \bmod p_x = x \bmod p_x$ then ACCEPT
else REJECT

The size of the above program is $|p_x| + |x \bmod p_x| + O(1)$. This is $2\log(\|A\|) + O(\log(n))$. It is clear that the program runs in polynomial time, and only accepts x. □

We note that Lemma 4 also works for $\mathbf{CND}^{p}$ complexity for p some polynomial.

Buhrman, Laplante and Miltersen [BLM00] show that Lemma 4 is tight.

Theorem 5 (Buhrman-Laplante-Miltersen) *For every polynomial p and sufficiently large n there exists a set of strings $A \subseteq \{0,1\}^n$ containing more than $2^{n/50}$ strings such that there is an x in A with*

$$\mathbf{CD}^{p,A}(x) \geq 2\log(\|A^{=n}\|) - O(1)$$

Corollary 6 *Let A be a set in $\mathbf{P}$. For each string $x \in A$ it holds that: $\mathbf{CD}^{p}(x) \leq 2\log(\|A^{=n}\|) + O(\log(n))$ for some polynomial p.*

Proof: We will use the same scheme as in Lemma 4, now using that $A \in \mathbf{P}$ and specifying the length of x, yielding an extra $\log(n)$ term for $|x|$ plus an additional $2\log\log(n)$ penalty for concatenating the strings. □

Theorem 5 also gives a relativized tightness result for Corollary 6.

Corollary 7 *1. A set S is sparse if and only if for all $x \in S$, $\mathbf{CD}^{p,S}(x) \le O(\log(|x|))$, for some polynomial p.*

2. A set $S \in \mathbf{P}$ is sparse if and only if for all $x \in S$, $\mathbf{CD}^{p}(x) \le O(\log(|x|))$, for some polynomial p.

3. A set $S \in \mathbf{NP}$ is sparse if and only if for all $x \in S$, $\mathbf{CND}^{p}(x) \le O(\log(|x|))$, for some polynomial p.

Proof: Lemma 4 yields that all strings in a sparse set have $O(\log(n))$ $\mathbf{CD}^p$ complexity. On the other hand simple counting shows that for any set A there must be a string $x \in A$ such that $\mathbf{CND}^A(x) \ge \log(\|A^{=|x|}\|)$. □

3.1 Sipser's Lemma

We can also use Lemma 3 to give a simple proof of the following important result due to Sipser [Sip83].

Lemma 8 *(Sipser) For every polynomial-time computable set A there exists a polynomial p and constant c such that for every n, for most r in $\Sigma^{p(n)}$ and every $x \in A^{=n}$,*

$$\mathbf{CD}^{p,A^{=n}}(x|r) \le \log(\|A^{=n}\|) + c\log(n)$$

Proof: For each k, $1 \le k \le n$, let r_k be a list of $4k(n+1)$ randomly chosen numbers less than 2^k. Let r be the concatenation of all of the r_k.

Fix $x \in A^{=n}$. Let $d = \|A^{=n}\|$. Fix k such that $2^{k-1} < 4dn^2 \le 2^k$. Consider one of the numbers y listed in r_k. By the Prime Number Theorem [HW79], the probability that y is prime and less than $4dn^2$ is at least $\frac{1}{2\log(4dn^2)}$. The probability that y fulfills the conditions of Lemma 3 for x is at least $\frac{1}{4\log(4dn^2)} > \frac{1}{4k}$. With probability about $1 - 1/e^{n+1} > 1 - 1/2^{n+1}$ we have that some y in r_k fulfills the condition of Lemma 3.

With probability at least 1/2, for every $x \in A$ there is some y listed in r_k fulfilling the conditions of Lemma 3 for x.

We can now describe x by $x \bmod y$ and the pointer to y in r. □

Note: Sipser's original proof gives a tighter bound than $c\log(n)$ but for most applications the additional $O(\log(n))$ additive factor makes no substantial difference.

Comparing our Lemma 4 with Sipser's lemma (Lemma 8,) we are able to eliminate the random string required by Sipser at the cost of an additional $\log(\|A^{=n}\|)$ bits.

4 Approximating sets with Extractors

By using extractors, we can obtain nearly the bound of Sipser's lemma 8 without the random string it requires. However, our result only works for most strings in A.

Theorem 9 *For any set A, any function $\varepsilon(n)$, there is a polynomial p such that for all n and for all but a $2\varepsilon(n)$ fraction of the $x \in A^{=n}$, $\mathbf{CD}^{p,A^{=n}}(x) \le \log\|A^{=n}\| + \log^{O(1)}(n/\varepsilon(n))$.*

We give a nondeterministic version of this result and give a bound on **CND** complexity. We also give a randomized version of these theorems, stating that the shorter string can be chosen at random and the probability of getting a short string which encodes as much information as the original string is bounded away from 1/2.

4.1 Extractors

An extractor can be thought of as a bipartite graph, whose first color class is larger than the second color class. By convention we think of the first color class as being on the left, and the second on the right. The vertices on the left side are all the strings of length n, so the first color class can be equated with the set $[N]$, where $N = 2^n$. Likewise, the vertices on the right side of the graph are labeled by strings of length $m \leq n$, so we let $M = 2^m$ and $[M]$ is equated with the vertices in the second color class.

4.1.1 Distributions

We will be choosing a node on the left side of the graph at random according to a distribution X. The result of choosing a neighbor uniformly at random in the graph will produce a distribution Y on vertices on the right.

The *min-entropy* of a distribution X over $[N]$ is defined as $\min\{-\log_2(X(x)) | x \in [N]\}$. The min-entropy of X can be thought of as a measure of the randomness present in a string x chosen according to X.

A distribution Y is said to be ε-close to Z if both distributions are over the same space $[M]$, and that for any $S \subseteq [M]$, $|Y(S) - Z(S)| \leq \varepsilon$.

4.1.2 Definition of extractors

A bipartite graph G with (independent) vertex sets $[N]$ and $[M]$, $N = 2^n, M = 2^m$, and for which the degree of all the vertices in the first color class is bounded by $D = 2^d$ is an $(n, k, d, m, \varepsilon)$ *extractor* if given any distribution X on the N vertices whose min-entropy is at least k, the result of choosing an x according to this distribution and a random neighbor y of x in the graph is ε-close to the uniform distribution over $[M]$. In our setting, the distribution X will be the uniform distribution over a subset $A \subseteq [N]$, so k will be $\log(\|A\|)$.

$\Gamma(x)$ denotes the set of neighbors of x in G when x is a vertex on the left side of the graph. The number of edges originating at some vertex x on the left side of the graph is called the *outdegree* of x, whereas the number $w(y)$ of edges adjacent with a vertex y on the right side of the graph is called the *indegree* (or weight) of y. $G(x, r)$ represents the rth neighbor of x in the graph, where multiple edges are allowed. When y is a vertex on the right side of the graph, $\Gamma_A^{-1}(y)$ is the subset of preimages of y which lie in A. The notation extends to sets in the natural way.

4.1.3 Best known explicit constructions

The results we state are subject to improvement if better explicit extractor constructions are found. We have stated our results in general terms so that new results on extractors will be immediately applicable.

The current best known explicit constructions for extractors are due to Ta-Shma, Zuckerman, Trevisan, and Raz, Reingold and Vadhan [Ta-96, Zuc96, Tre99, RRV99]. The extractors best suited for our purposes are the ones which can be constructed for any k, and with $m = k$, with the smallest amount of additional randomness. We illustrate our results with the parameters obtained from Ta-Shma's construction.

Theorem 10 (Ta-Shma) *There is an explicit construction that for every n and for any function $\varepsilon(n)$ and every $m = m(n) \leq n$ yields an extractor with parameters $(n, m, \log^{O(1)}(n/\varepsilon(n)), m, \varepsilon(n))$.*

It is useful to compare this construction to the current lower bound on extractors, due to Nisan and Zuckerman [NZ93].

Theorem 11 (Nisan-Zuckerman) *There is a constant c such that for all $n, m, k \leq n-1, \varepsilon < 1/2$, if there is an extractor whose parameters are $(n, k, d, m, \varepsilon)$, then it must be the case that $d \geq \min\{m, c\log(n/\varepsilon)\}$.*

This lower bound also gives a good indication as to the limits of the techniques described in this paper.

4.2 Extracting CD complexity

Theorem 9 follows immediately from the following result, using the explicit extractor construction of Ta-Shma. For this theorem we assume that there is an explicit extractor construction whose parameters are $(n, k, d, m, \varepsilon)$, and we write $M = 2^m$.

Theorem 12 *Fix a set A, a polynomial $q(n)$ and $\varepsilon = \varepsilon(n)$. Then there is a polynomial $p(n)$ such that for all n and for all but a 2ε fraction of the $x \in A^{=n}$, there is a y such that*

1. $|y| = m$
2. $\mathbf{C}^p(y|x) \leq d + O(1)$
3. $\mathbf{CD}^{p,A^{=n}}(x) \leq \mathbf{C}^q(y) + 3d + 2\log(\|A^{=n}\|/M) + c\log(n + d + \log(\|A^{=n}\|/M)) + O(1)$,

where the underlying extractor's parameters are determined by n and $k = \log(\|A^{=n}\|)$, and c is a small absolute constant.

For the remainder of this section, we fix n and we let $S = \|A^{=n}\|$. In our setting, we will think of the set $A^{=n}$ as defining a distribution of min-entropy $k = \log(S)$. The string x represents an element of $A^{=n}$ and y is one of its neighbors in the graph G. Hence y has length m; computing y from x requires knowing only a short ("random") string of length d; and as we will see, y together with some short additional distinguishing information will suffice to distinguish the string x (in the sense of **CD** complexity).

The following lemmas are at the heart of the argument. They allow us to upper-bound the number of "bad" elements in $A^{=n}$, where "bad" means the strings x for which Theorem 12 will not apply.

In order to get a short description for x, we need to find a string y in its range which has small indegree (counting only those edges originating in $A^{=n}$.)

In Lemma 13, we use the properties of the extractor to obtain an upper bound on the number of y which have large indegree. In the statement of the lemma, we use the variable w_0 to represent the threshold on degree: any vertex with degree larger than w_0 has large degree. A typical value for w_0 is twice the average degree of the graph.

Lemma 14 gives an upper bound on the number of x on the left side of the extractor whose neighbors all lie within a small subset of the right side of the graph. When the small subset is the set of vertices with large indegree, these x are the "bad" x to which the theorem will not apply.

Lemma 13 *Consider the restriction of the extractor to the set of edges originating in $A^{=n}$. Recall that the degree of the graph is bounded by $D = 2^d$. In this restricted graph, let w_0 be an indegree threshold, $\frac{DS}{M} < w_0 \leq DS$, and Y be a subset of vertices on the right hand side of the extractor graph. If $\forall y \in Y, w(y) > w_0$, then $\|Y\| \leq \varepsilon \left(\frac{w_0}{DS} - \frac{1}{M}\right)^{-1}$.*

Proof: Let Y be the set of vertices whose indegree (in the restricted graph) exceeds w_0. Because the graph is an extractor, it must be the case that $\varepsilon \geq \frac{w(Y)}{\Gamma(A^{=n})} - \frac{\|Y\|}{M} \geq \frac{w(Y)}{DS} - \frac{\|Y\|}{M}$. Since $w(Y) \geq w_0\|Y\|$, we get $\|Y\| \leq \varepsilon \left(\frac{w_0}{DS} - \frac{1}{M}\right)^{-1}$ as claimed. □

Lemma 14 *In the restricted graph, if Y is a set on the right side of the graph, then*

$$\|\{x \in A^{=n} : \Gamma(x) \subseteq Y\}\| \leq \left(\varepsilon + \frac{\|Y\|}{M}\right) S.$$

Proof: Let $X = \{x \in A^{=n} : \Gamma(x) \subseteq Y\}$. The distribution which consists of picking a random element of $A^{=n}$ and then choosing a random neighbor gives measure at least $\|X\|/S$ to the set Y. Because of the extractor property, $\frac{\|X\|}{S} - \frac{\|Y\|}{M} \leq \varepsilon$. □

To conclude, we give the proof of Theorem 12.

Proof: Let A be a set and ε, n be given as in the statement of the theorem. By Lemma 13, applied with $w_0 = 2DS/M$ $(D = 2^d)$ and Lemma 14 with Y as in the hypothesis of Lemma 13, the size of the subset $B \subseteq A^{=n}$ such that $\forall x \in B$, $\forall y \in \Gamma(x)$, y has indegree at least w_0 can have size at most $2\varepsilon S$. Therefore for all but $2\varepsilon S$ of the x in $A^{=n}$, there is a y in its range whose indegree is at most $2DS/M$. For each such x, let r_x be the label of one of the edges in G which connects x to such a y. We need to verify 3 properties for each of these pairs x, y.

1. $|y| = m$: This is by choice of the extractor G.

2. $\mathbf{C}^p(y|x) \leq d + O(1)$: $y = G(x, r_x)$ for some $r_x \in \Sigma^d$, so the algorithm to print y will contain an encoding of r_x, and on input x computes $G(x, r_x)$ and outputs the result.

3. $\mathbf{CD}^{p,A^{=n}}(x) \leq \mathbf{C}^q(y) + 3d + 2\log(S/M) + c\log(n) + O(1)$: The program to recognize x will contain an encoding for an r_x and y for which $G(x, r_x) = y$ and the indegree of y is at most $2DS/M$. It must also contain a distinguishing program p_x which recognizes x among the $2DS/M$ vertices on the left originating in A that are adjacent to y. (The encoding of r_x is required to test that x is adjacent to y, but may be omitted if the degree of the graph is polynomial. This is not the case in the current explicit, efficient extractors, whose degree is on the order of $2^{\log^{O(1)}(n)}$.) The length of p_x is bounded by $2\log(2DS/M) + O(\log(n + \log(2DS/M)))$, by Lemma 4. (An additional logarithmic term is needed to encode the lengths of the various components of the encoding, but this is bundled in the O notation.)

   The algorithm follows:

   input z
   If $z \notin A^{=n}$ then REJECT
   else if $G(z, r_x) \neq y$ then REJECT
   else if $p_x(z) = 1$ then ACCEPT
   else REJECT

So the program requires an encoding of y, r, and the distinguishing program p_x, for a total length of $\mathbf{C}^q(y) + d + 2\log(2DS/M) + c\log(n + \log(2DS/M)) + O(1)$.

$\square$

4.3 Extracting CND complexity

A statement analogous to Theorem 12 can be made for **CND** complexity. Using a slight variant of the proof of Theorem 12, we can get a bound which is smaller by a a term of d. Also in the upper bound, $\mathbf{CD}^q(y)$ is used instead of possibly larger term $\mathbf{C}^q(y)$.

Theorem 15 *Fix a set A in* **NP**, *a polynomial $q(n)$, and $\varepsilon = \varepsilon(n)$. Then there is a polynomial $p(n)$ such that for every n and for all but a 2ε fraction of the $x \in A^{=n}$, there is a y such that*

1. $|y| = \log(\|A^{=n}\|)$
2. $\mathbf{C}^p(y|x) \leq d + O(1)$
3. $\mathbf{CND}^p(x) \leq \mathbf{CD}^q(y) + 2d + c\log(n+d) + O(1)$.

The proof is essentially the same as that of Theorem 12. To simplify the notation we make the assumption that the extractor used achieves $k = m$, as does Ta-Shma's construction. To obtain property 3, we need only guess y, and verify our guess using a distinguishing program for y whose length is bounded by $\mathbf{CD}^q(y)$. Likewise, we can simply guess r and omit its encoding, and use the distinguishing program p to verify our guess for r.

4.4 Randomly extracting CD complexity

Another variant that saves a $d = \log(D)$ term is to choose a counterpart y to a string x in a set in **P** at random. We will only require that for most x, at least half of the edges from x map to a "good" y. Although this comes at the cost of only applying to "most" strings x, this improves upon the result of Sipser [Sip83] by reducing the length of the random string from $n^{O(1)}$ to $\log^{O(1)}(n/\varepsilon)$. The proof is similar to that of Theorem 12; it requires only a slight modification to the counting argument.

Theorem 16 *Fix a set A in* **P**, *a polynomial $q(n)$, and a function $\varepsilon(n)$. Then there is a polynomial $p(n)$ such that for every n and for all but a $4\varepsilon(n)$ fraction of the $x \in A^{=n}$, and at least half of the strings r of length d, there is a y such that:*

1. $|y| = \log(\|A^{=n}\|)$
2. $\mathbf{C}^p(y|x, r) \leq O(1)$
3. $\mathbf{CD}^p(x|r) \leq \mathbf{C}^q(y) + 2d + 2\log(n+d) + O(1)$.

5 Extracting random strings

In the previous section, we used the fact that the strings examined were in a small set of bounded complexity, and we showed the existence of strings for which the mutual information was roughly the **CND** complexity of the original string. Here we use extractor techniques to a achieve a slightly different goal. We obtain an *incompressible* string whose length is close to the **CD** complexity of x and which can be computed from x using only $\log(n/\varepsilon)$ bits.

In the case of unbounded Kolmogorov complexity, it is easy to see that the following proposition is true.

Proposition 17 [LV97, Ex. 2.1.5, p. 102] *For any string x of length n, there is a y such that:*

1. $|y| = \mathbf{C}(x)$
2. $\mathbf{C}(y|x) \leq \log(n)$
3. $\mathbf{C}(y) > |y| - O(1)$.

Namely, y is a minimal-length program for x, and can be obtained from x by dovetailing, given the value of $\mathbf{C}(x)$. In the time-bounded setting however, this argument fails, since dovetailing would take too much time. Our use of extractors is far afield from the above approach, yet it yields results surprisingly close to Proposition 17. (Non-explicit extractors actually allow us to give an alternate proof of Proposition 17, although this is more an artifact than a useful new proof.)

Theorem 18 *For any polynomial $q(n)$ and function $\varepsilon(n)$, then there exists a polynomial $p(n)$ such that for any string x of length n, there is a string y such that:*

1. $|y| = \mathbf{CND}^p(x)/2 - c_1 \log(n)$
2. $\mathbf{C}^p(y|x) \leq \log^{c_2}(n/\varepsilon(n))$
3. $\mathbf{C}^q(y) > |y| - c_\varepsilon$,

where c_1, c_2 are absolute constants, and c_ε depends only on ε.

Instead of giving the proof of Theorem 18, we prove the result in the following more general form, which may be improved as explicit extractor constructions are improved.

Theorem 19 *For any polynomial $q(n)$ and $\varepsilon = \varepsilon(n)$, there exists a polynomial $p(n)$ such that for any string x, there is a string y such that:*

1. $|y| = m$
2. $\mathbf{C}^p(y|x) \leq d + c_1$
3. $\mathbf{C}^q(y) > |y| - c_\varepsilon$,

where c_1 is an absolute constant, c_ε is a constant depending only on ε, $k = \frac{1}{2}(\mathbf{CND}^{2^d \cdot p}(x) - 2\log(n) - c_1 - 1)$ and $(n, k, d, m, \varepsilon)$ are the parameters of an explicit extractor.

Theorem 18 follows by applying Theorem 19 with parameters obtained from Ta-Shma's extractor [Ta-96].

Proof: (Sketch) Consider a family of extractors with parameters $n, k, m(k)$. Fix any n, k and let $G = G_{n,k,m}$, $m = m(k)$, be the extractor with parameters n, m, k. (Later we will fix k to be a specific value.) Let $A_{n,m} = \{x | \Gamma(x) \subseteq \mathbf{C}[q(n), m - c_\varepsilon]\}$, where $\mathbf{C}[t, l] = \{z | \mathbf{C}^t(z) \leq l\}$, and c_ε is chosen so that $c_\varepsilon > \log(\frac{1}{1-\varepsilon})$ for large enough n.

The fact that G is an extractor prohibits the set $A_{n,m}$ from being large, as we see now. If $\|A_{n,m}\| > 2^k$, then by the properties of the extractor,

$$1 - \varepsilon \leq \frac{\|\mathbf{C}[q(n), m - c_\varepsilon]\|}{2^m}.$$

But $\|\mathbf{C}[q(n), m - c_\varepsilon]\| \leq 2^{m - c_\varepsilon}$, and we have chosen $c_\varepsilon > \log(\frac{1}{1-\varepsilon})$ in order to get a contradiction. Hence we must conclude that $\|A_{n,m}\| \leq 2^k$.

Now we may apply Lemma 4 for **CND** to conclude that all $x \in A_{n,m}$ must have small **CND** complexity. First notice that verifying membership in $A_{n,m}$ is in NTIME$[2^d \cdot p]$ for some polynomial p, since it suffices to guess, for each neighbor y of x in $G_{n,m}$ a program of length $m - c_\varepsilon$ which prints out y. Hence, there exists a constant c_1 such that for every $x \in A_{n,m}$, $\mathbf{CND}^{2^d \cdot p}(x) \leq 2\log(\|A_{n,m}\|) + 2\log(n) + c_1$.

Now consider x with respect to the extractor $G_{n,\hat{k},m(\hat{k})}$, where $\hat{k} = \frac{1}{2}(\mathbf{CND}^{2^d \cdot p}(x) - 2\log(n) - c_1 - 1)$ and m is maximal for this k. By the observation above, it must be the case that $x \notin A_{n,m}$. Therefore there must be a y not in $\mathbf{C}[q(n), m - c_\varepsilon]$ to which x is mapped under $G_{n,k,m}$. It is easy to verify that y satisfies the properties claimed in the statement of the theorem. □

6 Lower Bounds

In this section we show that there exists an infinite set A such that every string in A has high **CND** complexity, even relative to A.

Fortnow and Kummer [FK96] prove the following result about relativized **CD** complexity:

Theorem 20 *There is an infinite set A such that for every polynomial p, $\mathbf{CD}^{p,A}(x) \geq |x|/5$ for almost all $x \in A$.*

We extend and strengthen their result for **CND** complexity:

Theorem 21 *There is an infinite set A such that $\mathbf{CND}^{2^{\sqrt{|x|}},A}(x) \geq |x|/4$ for all $x \in A$.*

The proof of Fortnow and Kummer of Theorem 20 uses the fact that one can start with a large set A of strings of the same length such that any polynomial-time algorithm on an input x in A cannot query any other y in A. However, a nondeterministic machine may query every string of a given length. Thus we need a more careful proof.

This proof is based on the proof of a result due to Goldsmith, Hemachandra and Kunen [GHK92] which we obtain as Corollary 22 below. In Section 9, we will also describe a rough equivalence between this result and an "X-search" theorem of Impagliazzo and Tardos [IT89].

Proof of Theorem 21:

We create our set A in stages. In stage k, we pick a large n and add to A a nonempty set of strings B of length n such that for all nondeterministic programs p running in time $2^{\sqrt{n}}$ such that $|p| < n/4$, $p^{B \cup A}$ accepts either zero or more than one strings in A. We first create a B that makes as many programs as possible accept zero strings in B. After that we carefully remove some strings from B to guarantee that the rest of the programs accept at least two strings.

Let P be the set of nondeterministic programs of size less than $n/4$. We have $\|P\| < 2^{n/4}$. We will clock all of these programs so that they will reject if they take time more than $2^{\sqrt{n}}$. We also assume that on every program p in P, input x and oracle O, $p^O(x)$ queries x.

Let $v = 2^{\sqrt{n}+1}\|P\|$ and $w = \|P\|v2^{\sqrt{n}}$. Pick sets $\Delta \subseteq P$ and $H \subseteq \Sigma^n$ that maximizes $\|\Delta\| + \|H\|$ such that $\|H\| \leq w\|\Delta\|$, and for all $X \subseteq \Sigma^n - H$ and $p \in \Delta$, $X \cap p^{A\cup X} = \emptyset$.

Note that $H \neq \Sigma^n$ since $\|H\| \leq w\|\Delta\| \leq w\|P\| \leq 2^{2\sqrt{n}+1}2^{3n/4} < 2^n$. Since some small program p always accepts we have that $\Delta \neq P$.

Our final B will be a subset of $\Sigma^n - H$ which guarantees that for all $p \in \Delta$, $p^{A\cup B}$ will not accept any strings in B. We will create B such that for all $p \in P - \Delta$, $p^{A\cup B}$ accepts at least two strings in B. Initially let $B = \Sigma^n - H$. For each $p \in P - \Delta$ and for each integer i, $1 \leq i \leq v$ do the following:

> Pick a minimal $X \subseteq B$ such that for some $y \in X$, $p^{A\cup X}(y)$ accepts. Fix an accepting path and let Q_{p_i} be all the queries made on that path. Let $y_{p,i} = y$, $X_{p,i} = X$ and $B = B - Q_{p,i}$.

Note that $\|Q_{p,i}\| \leq 2^{\sqrt{n}}$. We remove no more than $\|P\|v2^{\sqrt{n}} \leq w$ strings in total. So if we cannot find an appropriate X, we have violated the maximality of $\|\Delta\| + \|H\|$. Note that $y_{p,i} \in X_{p,i} \subseteq Q_{p,i}$ and all of the $X_{p,i}$ are disjoint.

Initially set all of the $X_{p,i}$ as unmarked. For each $p \in P - \Delta$ do the following twice:

> Pick an unmarked $X_{p,i}$. Mark all $X_{q,j}$ such that $X_{q,j} \cap Q_{p,i} \neq \emptyset$. Let $B = B \cup X_{p,i}$.

We have that $y_{p,i} \in B$ and $p^{A\cup B}(y_{p,i})$ accepts for every $X_{p,i}$ processed.

At most $2 \cdot 2^{\sqrt{n}}\|P\| - 1 < v$ of the $X_{q,j}$'s get marked before we have finished, we always can find an unmarked $X_{p,i}$.

Finally note that $B \subseteq \Sigma^n - H$ and for every $p \in P - \Delta$ we have at least two $y \in B$, such that $p^{A\cup B}(y)$ accepts. Since $P - \Delta \neq \emptyset$ this also guarantees that $B \neq \emptyset$. Thus we have fulfilled the requirements for stage k. □

Using Theorem 21 we get the following corollary first proved by Goldsmith, Hemachandra and Kunen [GHK92].

Corollary 22 (Goldsmith-Hemachandra-Kunen) *Relative to some oracle, there exists an infinite polynomial-time computable set with no infinite sparse* **NP** *subsets.*

Proof: Let A from Theorem 21 be both the oracle and the set in P^A. Suppose A has an infinite sparse subset S in NP^A. Pick a large x such that $x \in S$. Applying Corollary 7(3) it follows that $\mathbf{CND}^{A,p}(x) \leq O(\log(n))$. This contradicts the fact that $x \in S \subseteq A$ and Theorem 21. □

The above argument shows actually something stronger:

Corollary 23 *Relative to some oracle, there exists an infinite polynomial-time computable set with no infinite subset in* **NP** *of density less than* $2^{n/9}$.

It remains open whether Corollary 23 holds for $2^{\delta n}$ for $\frac{1}{9} < \delta < 1$.

7 CD vs. C and CND

This section deals with the consequences of the assumption that one of the complexity measures **C**, **CD**, and **CND** coincide for polynomial time. We will see that these assumptions are equivalent to well-studied complexity-theoretic assumptions. This allows us to apply the machinery developed in the previous sections. We will use the following function classes:

Definition 24 *1. The class $\mathbf{FP}^{\mathbf{NP}[\log(n)]}$ is the class of functions computable in polynomial time that can adaptively access an oracle in* **NP** *at most $c\log(n)$ times, for some c.*

2. The class $\mathbf{FP}^{\mathbf{NP}}_{tt}$ is the class of functions computable in polynomial time that can non-adaptively access an oracle in **NP**.

Theorem 25 *The following are equivalent:*

1. $\forall p_2\ \exists p_1, c\ \forall x, y : \mathbf{C}^{p_1}(x \mid y) \leq \mathbf{CND}^{p_2}(x \mid y) + c\log(|x| + |y|)$.
2. $\forall p_2\ \exists p_1, c\ \forall x, y : \mathbf{CD}^{p_1}(x \mid y) \leq \mathbf{CND}^{p_2}(x \mid y) + c\log(|x| + |y|)$.
3. $\mathbf{FP}^{\mathbf{NP}[\log(n)]} = \mathbf{FP}^{\mathbf{NP}}_{tt}$.

Proof: $(1 \Rightarrow 2)$ is trivial.

$(2 \Rightarrow 3)$ We will first need the following lemma due to Lozano (see [JT95, pp. 184–185]).

Lemma 26 $\mathbf{FP}^{\mathbf{NP}[\log(n)]} = \mathbf{FP}^{\mathbf{NP}}_{tt}$ *if and only if for every f in $\mathbf{FP}^{\mathbf{NP}}_{tt}$ there exists a function $g \in \mathbf{FP}$, that generates a polynomial-size set such that $f(x) \in g(x)$.*

In the following let $f \in \mathbf{FP}^{\mathbf{NP}}_{tt}$. Let $f(x) = y$. We will see that there exists a p and c such that $\mathbf{CND}^p(y \mid x) \leq c\log(|x|)$. We can assume that the machine computing $f(x)$ produces a list of queries $Q = \{q_1, \ldots, q_l\}$ to **SAT**. Let w be the exact number of queries in Q that are in **SAT**. Thus $w = \|Q \bigcap \mathbf{SAT}\|$. Consider the following $\mathbf{CND}^p$ program given x:

input z
use $f(x)$ to generate Q.
guess $q^1, \ldots, q^w \in Q$ that are in **SAT**
guess satisfying assignments for $q^1, \ldots, q^w$.
REJECT if not all $q^1, \ldots, q^w$ are satisfiable.
compute $f(x)$ with $q^1, \ldots, q^w$ answered YES and $q_j \in Q \setminus \{q^1, \ldots, q^w\}$ answered NO
ACCEPT if and only if $f(x) = z$

The size of the above program is $c\log(|x|)$, accepts only y, and runs in time p, for some polynomial p and constant c depending only on f. It follows that also all the prefixes of y have $\mathbf{CND}^p$ complexity bounded by $c\log(|x|) + \log(|x|) + O(1)$. By assumption there exists a polynomial p' and constant d such that $\mathbf{CD}^{p'}(z \mid x) \leq d\log(|x|)$ for z a prefix of y. For each of these z there is some program r such that

1. $|r| \leq d\log(|x|)$,
2. $U(r, z, x)$ accepts,
3. $U(r, u, x)$ rejects for each $u \neq z$ and

4. $U(r, u, x)$ runs in time at most $p'(|x|)$ for each $|u| \leq |x|$.

We can use the following procedure to enumerate possibilities for y.

Let $S_0 = \{\epsilon\}$.
For $m = 1$ to $|y|$.
Let S'_m consist of all strings u of length m such that u extends some string in S_{m-1}.
Let S_m consist of all strings u in S'_m such that there is some r, $|r| \leq d\log(|x|)$,
such that $U(r, u, x)$ accepts in $p'(|x|)$ steps and
for all $v \in S'_m - \{u\}$, $U(r, v, x)$ does not accept in $p'(|x|)$ steps.

Note for all m, S_m and S'_m will have size bounded by $2|x|^d$ so the above algorithm runs in polynomial time. By our discussion, y will belong to $S_{|y|}$ so it follows using Lemma 26 that $\mathbf{FP}^{\mathbf{NP}[\log(n)]} = \mathbf{FP}^{\mathbf{NP}}_{tt}$.
$(3 \Rightarrow 1)$ Let y be a string such that $\mathbf{CND}^{p'}(y \mid x) = k$. Let e be the program of length k that witnesses this. Consider the following function:

$$f(\langle e, 0^l, 0^m, x\rangle) = w_1 w_2 \dots w_m$$

where

$$w_i = \begin{cases} 1 & \text{if there is a } z \text{ of length } m \text{ with the } i^{th} \text{ bit equal to one} \\ & \text{such that } U_n(e, z, x) \text{ nondeterministically accepts in } l \text{ steps.} \\ 0 & \text{otherwise.} \end{cases}$$

Note that if e is a **CND** program, that runs in l steps, then it accepts exactly one string, w of length m. Hence $f(\langle r, 0^{p'(|y|+|x|)}, 0^{|y|}, x\rangle) = y$. It is not hard to see that in general f is in $\mathbf{FP}^{\mathbf{NP}}_{tt}$ and by assumption in $\mathbf{FP}^{\mathbf{NP}[\log(n)]}$ via machine M. Next given $e = r$, $m = |y|$, $l = p'(|y| + |x|)$, x and the $c\log(|y| + |x|)$ answers to the **NP** oracle that M makes we can generate y in polynomial time. We have that $\mathbf{C}^p(y \mid x) \leq \mathbf{CND}^{p'}(y \mid x) + c\log(|y| + |x|)$ □

For the next corollary we will use some results from [JT95]. We will use the following class of limited nondeterminism defined in [DT90].

Definition 27 *Let $f(n)$ be a function from $\mathbb{N} \mapsto \mathbb{N}$. The class $\mathbf{NP}[f(n)]$ denotes that class of languages that are accepted by polynomial-time bounded nondeterministic machines that on inputs of length n make at most $f(n)$ nondeterministic moves.*

Jenner and Torán [JT95] show a series of consequences of the assumption $\mathbf{FP}^{\mathbf{NP}[\log(n)]} = \mathbf{FP}^{\mathbf{NP}}_{tt}$. By Theorem 25 we also get these consequences from a collapse of **CD** and **CND** complexity.

Corollary 28 *If $\forall p_2\ \exists p_1, c\ \forall x, y : \mathbf{CD}^{p_1}(x \mid y) \leq \mathbf{CND}^{p_2}(x \mid y) + c\log(|x| + |y|)$ then for any k:*

1. *$\mathbf{NP}[\log^k(n)]$ is included in $\mathbf{P}$.*

2. *$\mathbf{SAT} \in \mathbf{NP}[\frac{n}{\log^k(n)}]$.*

3. *$\mathbf{SAT} \in \mathbf{DTIME}(2^{n^{O(1/\log\log(n))}})$.*

4. *There exists a polynomial q such that for every m formulae $\phi_1, \dots, \phi_m$ of n variables each such that at least one is satisfiable, there exists an i such that ϕ_i is satisfiable and*

$$\mathbf{CND}^q(\phi_i | \langle\phi_1, \dots, \phi_m\rangle) \leq O(\log\log(n + m))$$

The last consequence simply restates one of the Jenner-Torán results in the notation of this paper.

We can use Corollary 28 to get a complete collapse if there is only a constant difference between **CD** and **CND** complexity.

Theorem 29 *The following are equivalent:*

1. $\forall p_2\ \exists p_1, c\ \forall x, y : \mathbf{C}^{p_1}(x \mid y) \leq \mathbf{CND}^{p_2}(x \mid y) + c.$
2. $\forall p_2\ \exists p_1, c\ \forall x, y : \mathbf{CD}^{p_1}(x \mid y) \leq \mathbf{CND}^{p_2}(x \mid y) + c.$
3. $\mathbf{P} = \mathbf{NP}.$

Proof Sketch: $(1 \Rightarrow 2)$ and $(3 \Rightarrow 1)$ are easy.

$(2 \Rightarrow 3)$ By Corollary 28(4) combined with the the assumption we have for any formulae $\phi_1, \ldots, \phi_m$ where at least one is satisfiable that

$$\mathbf{CD}^{p_1}(\phi_i | \langle \phi_1, \ldots, \phi_m \rangle) \leq c \log\log(n+m)$$

for some satisfiable ϕ_i. We can enumerate all the programs p of length at most $c \log\log(n+m)$ and find all the formula ϕ_i such that $p(\phi_i, \langle \phi_1, \ldots, \phi_m \rangle) = 1$ and $p(\phi_j, \langle \phi_1, \ldots, \phi_m \rangle) = 0$ for $j \neq i$.

Thus given $\phi_1, \ldots, \phi_m$ we can in polynomial-time create a subset of size $\log^c(n+m)$ that contains a satisfiable formula if the original list did. We then apply a standard tree-pruning algorithm to find the satisfying assignment of any satisfiable formula. □

A simple modification of the proof shows that Theorem 29 holds if we replace the constant c with $a\log(n)$ for any $a < 1$.

For the next corollary we will need the following definition (see [ESY84]).

Definition 30 *A promise problem is a pair of sets (Q, R). A set L is called a solution to the promise problem (Q, R) if $\forall x (x \in Q \Rightarrow (x \in L \Leftrightarrow x \in R))$. For any function f, f**SAT** denotes the set of Boolean formulas with at most $f(n)$ satisfying assignments for formulae of length n.*

The next theorem states that nondeterministic computations that have few accepting computations can be "compressed" to nondeterministic computations that have few nondeterministic moves if and only if $\mathbf{C}^{poly} \leq \mathbf{CD}^{poly}$.

Theorem 31 *The following are equivalent:*

1. $\forall p_2\ \exists p_1, c\ \forall x, y : \mathbf{C}^{p_1}(x \mid y) \leq \mathbf{CD}^{p_2}(x \mid y) + c.$
2. $(1\mathbf{SAT}, \mathbf{SAT})$ *has a solution in* $\mathbf{P}$.
3. *For all time constructible f,* $(f\mathbf{SAT}, \mathbf{SAT})$ *has a solution in* $\mathbf{NP}[2\log(f(n)) + O(\log(n))]$.

Proof: $(1 \iff 2)$ This was proven in [FK96].
$(3 \Rightarrow 2)$ Take $f(n) = 1$ and the fact [DT90] that $\mathbf{NP}[O(\log(n))] = \mathbf{P}$.
$(2 \Rightarrow 3)$ Let ϕ be a formula with at most $f(|\phi|)$ satisfying assignments. Lemma 4 yields that for every satisfying assignment a to ϕ, there exists a polynomial p such that $\mathbf{CD}^p(a \mid \phi) \leq 2\log(f(|\phi|)) + O(\log(|\phi|))$. Hence (using that $1 \iff 2$) it follows that $\mathbf{C}^{p'}(a \mid \phi) \leq 2\log(f(|\phi|)) + c\log(|\phi|)$, for some constant c and polynomial p'. The limited nondeterministic machine now guesses a $\mathbf{C}^{p'}$ program program e of size at most $2\log(f(|\phi|)) + c\log(|\phi|)$, and runs it (relative to ϕ) and accepts iff the generated string is a satisfying assignment to ϕ. □

Corollary 32 $\mathbf{FP}^{\mathbf{NP}[\log(n)]} = \mathbf{FP}^{\mathbf{NP}}_{tt}$ *implies the following:*

1. *For any k the promise problem* $(2^{\log^k(n)}\mathbf{SAT}, \mathbf{SAT})$ *has a solution in* $\mathbf{P}$.
2. *For any k, the class of languages that is accepted by nondeterministic machines that have at most $2^{\log^k(n)}$ accepting paths on inputs of length n is included in* $\mathbf{P}$.

Proof: This follows from Theorem 25, Theorem 31, and Corollary 28. □

8 Satisfying Assignments

We show several connections between **CD** complexity and finding satisfying assignments of Boolean formulae. By Cook's Theorem [Coo71], finding satisfying assignments is equivalent to finding accepting computation paths of any **NP** computation.

8.1 Enumerating Satisfying Assignments

Papadimitriou [Pap96] mentioned the following proposition:

Proposition 33 *There exists a Turing machine that given a formula ϕ will output the set A of satisfying assignments of ϕ in time polynomial in $|\phi|$ and $\|A\|$.*

We can use **CD** complexity to show the following.

Theorem 34 *Proposition 33 is equivalent to* $(1\mathbf{SAT}, \mathbf{SAT})$ *has a solution in* $\mathbf{P}$.

In Proposition 33, we do not require the machine to halt after printing out the assignments. If the machine is required to halt in time polynomial in ϕ and $\|A\|$ we have that Proposition 33 is equivalent to $\mathbf{P} = \mathbf{NP}$.

Proof of Theorem 34: The implication of $(1\mathbf{SAT}, \mathbf{SAT})$ having a solution in $\mathbf{P}$ is straightforward. We concentrate on the other direction.

Let $d = \|A\|$. By Lemma 4 and Theorem 31 we have that for every element x of A, $\mathbf{C}^q(x|\phi) \leq 2\log(d) + c\log(n)$ for some polynomial q and constant c. We simply now try every program p in length increasing order and enumerate $p(\phi)$ if it is a satisfying assignment of ϕ. □

8.2 Computing Satisfying Assignments

In this section we turn our attention to the question of the complexity of generating a satisfying assignment for a satisfiable formula [WT93, HNOS96, Ogi96, BKT94]. It is well known [Kre88] that one can generate (the leftmost) satisfying assignment in $\mathbf{FP}^{\mathbf{NP}}$. A tantalizing open question is whether one can compute some (not necessary the leftmost) satisfying assignment in $\mathbf{FP}^{\mathbf{NP}}_{tt}$. Formalizing this question, define the function class $\mathbf{F}_{sat}$ by $f \in \mathbf{F}_{sat}$ if when $\varphi \in \mathbf{SAT}$ then $f(\varphi)$ is a satisfying assignment of φ.

The question now becomes $\mathbf{F}_{sat} \bigcap \mathbf{FP}^{\mathbf{NP}}_{tt} = \emptyset$? Translating this to a **CND** setting we have the following.

Lemma 35 $\mathbf{F}_{sat} \bigcap \mathbf{FP}^{\mathbf{NP}}_{tt} \neq \emptyset$ *if and only if for all $\phi \in \mathbf{SAT}$ there exists a satisfying assignment a of ϕ such that $\mathbf{CND}^p(a \mid \phi) \leq c\log(|\phi|)$ for some polynomial p and constant c.*

Lemma 35 relativizes where we consider a relativized version of $\mathbf{SAT}^A$ [GJ93] by adding a series of extra predicates $A_0, A_1, A_2, \ldots$ such that $A_n(x_1, \ldots, x_n)$ is true if $x_1 \ldots x_n$ is in A.

Toda and Watanabe [WT93] showed that relative to a random oracle $\mathbf{F}_{sat} \bigcap \mathbf{FP}_{tt}^{\mathbf{NP}} \neq \emptyset$. On the other hand Buhrman and Thierauf [BT96] showed that there exists an oracle where $\mathbf{F}_{sat} \bigcap \mathbf{FP}_{tt}^{\mathbf{NP}} = \emptyset$. Their result also holds relative to the set constructed in Theorem 21.

Theorem 36 *Relative to the set A constructed in Theorem 21, $\mathbf{F}_{sat} \bigcap \mathbf{FP}_{tt}^{\mathbf{NP}} = \emptyset$.*

Proof: For some n, let ϕ be the formula on n variables such that $\phi(x)$ is true if and only if $x \in A$. Suppose $\mathbf{F}_{sat} \bigcap \mathbf{FP}_{tt}^{\mathbf{NP}} \neq \emptyset$. It now follows by Lemma 35 that there exists an $x \in A$ such that $\mathbf{CND}^{p,A}(x) \leq O(\log(|x|))$ for some polynomial p, contradicting the fact that for all $x \in A$, $\mathbf{CND}^{2^{\sqrt{|x|}},A}(x) \geq |x|/4$. $\square$

8.3 Isolating Satisfying Assignments

In this section we take a Kolmogorov complexity view of the statement and proof of the famous Valiant-Vazirani lemma [VV85]. The Valiant-Vazirani lemma gives a randomized reduction from a satisfiable formula to another formula that with a non negligible probability has exactly one satisfying assignment.

We state the lemma in terms of Kolmogorov complexity.

Lemma 37 *There is some polynomial p such that for all ϕ in* **SAT** *and all r such that $|r| = p(|\phi|)$ and $\mathbf{C}(r) \geq |r|$, there is some satisfying assignment a of ϕ such that $\mathbf{CD}^p(a|\langle\phi, r\rangle) \leq O(\log(|\phi|))$.*

The usual Valiant-Vazirani lemma follows from the statement of Lemma 37 by choosing r and the $O(\log(|\phi|))$-size program randomly.

We show how to derive the Valiant-Vazirani Lemma from Sipser's Lemma (Lemma 8). Note that Sipser's result predates Valiant-Vazirani by a couple of years.

Proof of Lemma 37: Let $n = |\phi|$.

Consider the set A of satisfying assignments of ϕ. We can apply Lemma 8 conditioned on ϕ using part of r as the random strings. Let $d = \lfloor \log(\|A\|) \rfloor$. We get that every element of A has a **CD** program of length bounded by $d + c\log(n)$ for some constant c. Since two different elements from A must have different programs, we have at least $1/n^c$ of the strings of length $d + c\log(n)$ must distinguish some assignment in A.

We use the rest of r to list n^{2c} different strings of length $d + c\log(n)$. Since r is random, one of these strings w must be a program that distinguishes some assignment a in A. We can give a **CD** program for a in $O(\log(n))$ bits by giving d and a pointer to w in r. $\square$

9 Search vs. Decision in Exponential-Time

If $\mathbf{P} = \mathbf{NP}$ then given a satisfiable formula, one can use binary search to find the assignment.

One might expect a similar result for exponential-time computation, i.e., if $\mathbf{EXP} = \mathbf{NEXP}$ then one should find a witness of a **NEXP** computation in exponential time. However, the proof for polynomial-time breaks down because as one does the binary search the input questions get too long. Impagliazzo and Tardos [IT89] give relativized evidence that this problem is indeed hard.

Theorem 38 ([IT89]) *There exists a relativized world where* $\mathbf{EXP} = \mathbf{NEXP}$ *but there exists a* **NEXP** *machine whose accepting paths cannot be found in exponential time.*

We can give a short proof of this theorem using Theorem 21.

Proof of Theorem 38: Let A be from Theorem 21.

We will encode a tally set T such that $\mathbf{EXP}^{A\oplus T} = \mathbf{NEXP}^{A\oplus T}$. Let M be a nondeterministic oracle machine such that M runs in time 2^n and for all B, M^B is $\mathbf{NEXP}^B$-complete.

Initially let $T = \emptyset$. For every string w in lexicographic order, put 1^{2w} into T if $M^{A\oplus T}(x)$ accepts.

Let $B = A \oplus T$ at the end of the construction. Since $M(w)$ could only query strings with length at most $2^{|w|} \leq w$, this construction will give us $\mathbf{EXP}^B = \mathbf{NEXP}^B$.

We will show that there exists a $\mathbf{NEXP}^B$ machine whose accepting paths cannot be found in time exponential relative to B.

Consider the $\mathbf{NEXP}^B$ machine M that on input n guesses a string y of length n and accepts if y is in A. Note that M runs in time $2^{|n|} \leq n$.

Suppose accepting computations of M^B can be found in time $2^{|n|^k} = 2^{\log^k(n)}$ relative to B. By Theorem 21, we can fix some large n such that $A^{=n} \neq \emptyset$ and for all $x \in A^{=n}$,

$$\mathbf{CND}^{2^{\log^k(n)},A}(x) \geq n/4. \tag{1}$$

Let $w_i = \|\{1^m \mid 1^m \in T \text{ and } 2^i < m \leq 2^{i+1}\}\|$. We will show the following lemma.

Lemma 39

$$\mathbf{CND}^{2^{\log^k(n)},A}(x|w_1, \ldots, w_{\log^k(n)}) \leq \log(n) + O(1)$$

Assuming Lemma 39, Theorem 38 follows since for each i, $|w_i| \leq i+1$. We thus have our contradiction with Equation (1).

Proof of Lemma 39: We will construct a program p^A to nondeterministically distinguish x. We use $\log(n)$ bits to encode n. First p will reconstruct T using the w_i's.

Suppose we have reconstructed T up to length 2^i. By our construction of T, strings of T of length at most 2^{i+1} can only depend on oracle strings of length at most $2^{i+1}/2 = 2^i$. We guess w_i strings of the form 1^m for $2^i < m \leq 2^{i+1}$ and nondeterministically verify that these are the strings in T. Once we have T, we also have $B = A \oplus T$ so in time $2^{\log^k(n)}$ we can find x. $\Box$

Impagliazzo and Tardos [IT89] prove Theorem 38 using an "X-search" problem. We can also relate this problem to **CND** complexity and Theorem 21.

Definition 40 *The X-search problem has a player who given N input variables not all zero, wants to find a one. The player can ask r rounds of l parallel queries of a certain type each and wins if the player discovers a one.*

Impagliazzo and Tardos use the following result about the X-search problem to prove Theorem 38.

Theorem 41 *([IT89]) If the queries are restricted to k-DNFs and $N > 2(klr)^2(l+1)^r$ then the player will lose on some non-zero setting of the variables.*

One can use a proof similar to that of Theorem 38 to prove a similar bound for Theorem 41. One needs just to apply Theorem 21 relative to the strategy of the player.

One can also use Theorem 41 to prove a weaker version of Theorem 21. Pick a large n and a time bound t. Let $N = 2^n$ and suppose for all $B \subseteq \Sigma^n$ there is an x in B, $\mathbf{CND}^{t,B}(x) \leq w$. Let $N = 2^n$.

For a fixed B and x, let p be the **CND** program that distinguishes x. Nondeterministically we can find the ith bit of x using t queries to B by guessing x and the accepting computation of $U_n(p,x)$. We can express this computation as the complement of a t-**DNF** question.

We now build a strategy for X-search: Try all p and i, $|p| \leq 2^{|p|}$ and $1 \leq i \leq n$ in the first round using the t-**DNF** described above. This gives us a list of $2^{|p|}$ possible x's. We just try them all.

This solves the X-search problem using $k = t$, $l = n2^{|w|}$ and $r = 2$. By Theorem 41 we have $N \leq 2(tn2^w 2)^2(n2^w)^2$. Taking logarithms we get $n \leq 2(\log t + \log n + w + \log 2) + 2(\log n + w)$. Thus we have a contradiction whenever $w = an$ and $t = 2^{bn}$ for $4a + 2b < 1$. In particular this gives us an infinite set A such that $\mathbf{CND}^{2^{n^{1/6}},A}(x) \geq |x|/7$ for all x in A.

10 BPP in the second level of the polynomial hierarchy

One of the applications of Sipser's [Sip83] randomized version of Lemma 4 is the proof that **BPP** is in Σ_2^p. We will show that the approach taken in Lemma 4 yields a new proof of this result. We will first prove the following variation of Lemma 3.

Lemma 42 *Let $S = \{x_1, \ldots, x_d\} \subseteq \{0, \ldots, n-1\}$. There exists a prime number p such that for all $x_i, x_j \in S$ $(i \neq j) : x_i \not\equiv x_j \bmod p$, such that $p \leq 2d^2 \log(n)$.*

Proof: We consider only prime numbers between c and $2c$. For $x_i, x_j \in S$ it holds that for at most $\log_c(n) = \frac{\log(n)}{\log(c)}$ different prime number p $x_i \equiv x_j \bmod p$. Moreover there are at most $d(d-1)$ different pairs of strings in S, so there exists a prime number p among the first $d(d-1)\frac{\log(n)}{\log(c)} + 1$ prime numbers such that for all $x_i, x_j \in S(i \neq j)$ it holds that $x_i \not\equiv x_j \bmod p$. Applying again the prime number Theorem [HW79] it follows that if we take $c > d(d-1)\log(n)$, $p \leq 2d^2 \log(n)$. □

The idea is to use Lemma 42 as a way to approximate the number of accepting paths of a **BPP** machine M. Note that the set of accepting paths $\mathbf{ACCEPT}_{M(x)}$ of M on x is in **P**. If this set is "small" then there exists a prime number satisfying Lemma 42. On the other hand if the set is "big" no such prime number exists. This can be verified in Σ_2^p: There exists a number p such that for all pairs of accepting paths x_i, x_j of M, $x_i \not\equiv x_j \bmod p$. In order to apply this idea we need the gap between the number of accepting paths when x is in the set and when it is not to be a square: if x is not in the set then $\|\mathbf{ACCEPT}_{M(x)}\| \leq k(|x|)$ and if x is in the set $\|\mathbf{ACCEPT}_{M(x)}\| > k^2(|x|)$. We will apply Zuckerman's [Zuc96] oblivious sampler construction to obtain this gap.

Theorem 43 *Let M be a probabilistic machine that witnesses that some set A is in* **BPP**. *Assume that $M(x)$ uses m random bits. There exists a machine M' that uses $3m + 9m^{1-\frac{1}{2\log^*(m)}}$ random bits such that if $x \in A$ then $\Pr[M'(x)$ accepts$] > 1 - \frac{1}{2^{2m}}$ and if $x \notin A$ then $\Pr[M'(x)$accepts$] < \frac{1}{2^{2m}}$.*

Proof: Use the sampler in [Zuc96] with $\epsilon < 1/6$, $\gamma = \frac{1}{2^{2m}}$, and $\alpha = 3m^{-\frac{1}{2\log^*(m)}}$.□

Let $A \in$ **BPP** witnessed by probabilistic machine M. Apply Theorem 43 to obtain M'. The Σ_2^p algorithm for $\overline{A}$ works as follows:

input x

Guess $p \leq 2^{2m+18m^{1-\frac{1}{2\log^*(m)}}}(3m + 9m^{1-\frac{1}{2\log^*(m)}})$

If for all $u, v \in \mathbf{ACCEPT}_{M'(x)}$ $u \not\equiv v \bmod p$ ACCEPT else REJECT

If $x \in \overline{A}$ then $\|\mathbf{ACCEPT}_{M'(x)}\| \leq 2^{m+9m^{1-\frac{1}{2\log^*(m)}}}$, and Lemma 42 guarantees that the above program accepts. On the other hand if $x \in A$ then $\|\mathbf{ACCEPT}_{M'(x)}\| > 2^{3m+9m^{1-\frac{1}{2\log^*(m)}}-1}$ and for every prime number $p \leq 2^{2m+18m^{1-\frac{1}{2\log^*(m)}}}(3m + 9m^{1-\frac{1}{2\log^*(m)}})$ there will be a pair of strings in $\mathbf{ACCEPT}_{M'(x)}$ that are not congruent modulo p. This follows because for every number $p \leq 2^{2m+18m^{1-\frac{1}{2\log^*(m)}}}(3m+9m^{1-\frac{1}{2\log^*(m)}})$ at most $2^{2m+18m^{1-\frac{1}{2\log^*(m)}}}(3m+9m^{1-\frac{1}{2\log^*(m)}})$ different u and v it holds that $u \not\equiv v \bmod p$. $\square$

Goldreich and Zuckerman [GZ97] independently used Zuckerman's sampler [Zuc96] to give another proof that **BPP** is in Σ_2^p.

Acknowledgments

We would like to thank José Balcázar, Leen Torenvliet and David Zuckerman for their comments on this subject. We thank John Tromp for the current presentation of the proof of Lemma 4. We thank Richard Beigel, Bill Gasarch, Stuart Kurtz, Amber Settle, Leen Torenvliet and the two anonymous referees for comments on earlier drafts.

References

[BDG88] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* Springer-Verlag, 1988.

[BKT94] H. Buhrman, J. Kadin, and T. Thierauf. On functions computable with nonadaptive queries to NP. In *Proceedings of the 9th IEEE Structure in Complexity Theory Conference*, pages 43–52. IEEE computer society press, 1994.

[BLM00] H. Buhrman, S. Laplante, and P. Miltersen. New bounds for the language compression problem. In *Proceedings of the 15th IEEE Conference on Computational Complexity*, pages 126–130. IEEE Computer Society, Los Alamitos, 2000.

[BT96] H. Buhrman and T. Thierauf. The complexity of generating and checking proofs of membership. In C. Pueach and R. Reischuk, editors, *13th Annual Symposium on Theoretical Aspects of Computer Science*, number 1046 in Lecture Notes in Computer Science, pages 75–86. Springer, 1996.

[Coo71] S. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd ACM Symposium on the Theory of Computing*, pages 151–158, Shaker Heights, Ohio, 1971.

[DT90] J. Díaz and J. Torán. Classes of bounded nondeterminism. *Math. Systems Theory*, 23:21–32, 1990.

[ESY84] S. Even, A. L. Selman, and Y. Yacobi. The complexity of promise problems with applications to public-key cryptography. *Information and Control*, 61(2):159–173, May 1984.

[FK96] L. Fortnow and M. Kummer. Resource-bounded instance complexity. *Theoretical Computer Science A*, 161:123–140, 1996.

[GHK92] J. Goldsmith, L. Hemachandra, and K. Kunen. Polynomial-time compression. *Computational Complexity*, 2(1):18–39, 1992.

[GJ93] J. Goldsmith and D. Joseph. Relativized isomorphisms of NP-complete sets. *Computational Complexity*, 3:186–205, 1993.

[GZ97] O. Goldreich and D. Zuckerman. Another proof that BPP subseteq PH (and more). *Electronic Colloquium on Computational Complexity*, 97(045), 1997.

[HNOS96] L. Hemaspaandra, A. Naik, M. Ogihara, and A. Selman. Computing solutions uniquely collapses the polynomial hierarchy. *SIAM J. Comput.*, 25(4):697–708, 1996.

[HW79] G. Hardy and E. Wright. *An introduction to the theory of numbers.* Oxford University Press, London, 5th edition, 1979.

[Ing32] A. Ingham. *The Distribution of Prime Numbers.* Cambridge Tracts in Mathematics and Mathematical Physics. Cambridge University Press, 1932.

[IT89] R. Impagliazzo and G. Tardos. Decision versus search problems in super-polynomial time. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, pages 222–227, 1989.

[JT95] B. Jenner and J. Torán. Computing functions with parallel queries to NP. *Theoretical Computer Science*, 141, 1995.

[Knu98] D. Knuth. *Sorting and Searching*, volume 3 of *The art of computer programming.* Addison-Wesley, second edition, 1998.

[Kre88] M. Krentel. The complexity of optimization problem. *J. Computer and System Sciences*, 36:490–509, 1988.

[LV97] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications.* Springer-Verlag, second edition, 1997.

[NZ93] N. Nisan and D. Zuckerman. More deterministic simulation in logspace. In *Proceedings of the 25th ACM Symposium on the Theory of Computing*, pages 330–335, 1993.

[Ogi96] M. Ogihara. Functions computable with limited access to NP. *Information Processing Letters*, 58:35–38, 1996.

[Pap96] C. Papadimitriou. The complexity of knowledge representation. Invited Presentation at the Eleventh Annual IEEE Conference on Computational Complexity, May 1996.

[RRV99] R. Raz, O. Reingold, and S. Vadhan. Extracting all the randomness and reducing the error in trevisan's extractors. In *Proceedings of the 29th ACM Symposium on the Theory of Computing*, pages 149–158, 1999.

[Sip83] M. Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th ACM Symposium on the Theory of Computing*, pages 330–335, 1983.

[Ta-96] A. Ta-Shma. On extracting randomness from weak random sources (extended abstract). In *Proceedings of the 28th ACM Symposium on the Theory of Computing*, pages 276–285, 1996.

[Tre99] L. Trevisan. Construction of extractors using pseudo-random generators. In *Proceedings of the 29th ACM Symposium on the Theory of Computing*, pages 141–148, 1999.

[VV85] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. In *Proceedings of the 17th ACM Symposium on the Theory of Computing*, pages 458–463, 1985.

[WT93] O. Watanabe and S. Toda. Structural analysis on the complexity of inverse functions. *Mathematical Systems Theory*, 26:203–214, 1993.

[Zuc96] D. Zuckerman. Randomness-optimal sampling, extractors, and constructive leader election. In *Proceedings of the 28th ACM Symposium on the Theory of Computing*, pages 286–295, 1996.