Resource-Bounded Kolmogorov Complexity Revisited[*]

Harry Buhrman[**1] and Lance Fortnow[***2]

[1] CWI, PO Box 94079, 1090 GB Amsterdam, The Netherlands
[2] CWI & University of Chicago, Department of Computer Science, 1100 E. 58th St., Chicago, IL 60637

Abstract. We take a fresh look at **CD** complexity, where $\mathbf{CD}^t(x)$ is the smallest program that distinguishes x from all other strings in time $t(|x|)$. We also look at a **CND** complexity, a new nondeterministic variant of **CD** 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 avoiding the random string needed by Sipser. Also we give a new simpler proof of Sipser's lemma.
- 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 a satisfying assignment can be found in output polynomial time if and only if a unique assignment can be found quickly. This answers an open question of Papadimitriou.
- New Kolmogorov-based constructions of the following relativized worlds:
  - There exists an infinite set in **P** with no sparse infinite subsets in **NP**.
  - **EXP** = **NEXP** but there exists a nondeterministic exponential time Turing machine whose accepting paths cannot be found in exponential time.
  - Satisfying assignment cannot be found with nonadaptive queries to **SAT**.

[*] At http://www.cs.uchicago.edu/~fortnow/papers/ a full version of this paper including complete proofs can be found.
[**] 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 and SION project 612-34-002, and by the European Union through NeuroCOLT ESPRIT Working Group Nr. 8556, and HC&M grant nr. ERB4050PL93-0516.
[***] Email: fortnow@cs.uchicago.edu. Supported in part by NSF grant CCR 92-53582 and the Fulbright scholar program and NWO.

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 [LV93, 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 the given string and no others. Sipser showed that one can approximate the size of sets using **CD** complexity with random advice.

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 random advice required of Sipser's Lemma [Sip83].

We also give a new simpler proof of Sipser's Lemma and show how it implies the important Valiant-Vazirani lemma [VV86] 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 result 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 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 nondeterministic exponential time Turing machine whose accepting paths cannot be found in polynomial 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 collide. We give a precise characterization of when we have equality of these classes, 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 [LV93]. We use $|x|$ to represent the length of a string x and $\|A\|$ to represent the number of elements in the set A. All of the logarithms are base 2. **EXP** is defined as DTIME(2^{poly}) and **NEXP** is defined as NTIME(2^{poly}).

Formally, we define the Kolmogorov complexity function $\mathrm{C}(x|y)$ by $\mathrm{C}(x|y) = \min_p\{|p| : U(p,y) = x\}$ where U is some fixed universal deterministic Turing machine. We define unconditional Kolmogorov complexity by $\mathrm{C}(x) = \mathrm{C}(x|\epsilon)$.

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, $\mathrm{C}(x) \leq |x| + c$ for every x.
- For every n and every y, there is an x such that $|x| = n$ and $\mathrm{C}(x|y) \geq n$.

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

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

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

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

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

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

Once again we let $\mathbf{CND}^t(x) = \mathbf{CND}^t(x|\epsilon)$.

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

Since one can distinguish a string by generating it we have

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

where c is a constant. Likewise, since every deterministic computation is also a nondeterministic computation we get

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

In Section 6 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.

Lemma 3. *Let $S = \{x_1, \dots, x_d\} \subseteq \{0, \dots, 2^n - 1\}$. For all $x_i \in S$ and at least half of the primes $p \le 4dn^2$, $x_i \not\equiv x_j \bmod p$ 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. 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 (see for example [Ing32]) 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 $\mathbf{CD}^{p,A^{=n}}(x) \le 2\log(\|A\|) + O(\log n)$ 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 $\mathbf{CD}^{poly}$ 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 the above Lemma also works for $\mathbf{CND}^{p}$ complexity for p some polynomial.

Corollary 5. *Let A be a set in $\mathbf{P}$. For each string $x \in A$ it holds that: $\mathbf{CD}^{p}(x) \le 2\log(\|A^{=n}\|) + O(\log(n))$ 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. □

Corollary 6. *1. A set S is sparse if and only if for all $x \in S$, $\mathbf{CD}^{p,S}(x) \leq O(\log(|x|))$, for some polynomial p.*

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

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

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 7 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 $x \in A^{=n}$,*

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

Proof: For each k, $1 \leq k \leq n$, 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 $2^{k-1} < 4dn^2 \leq 2^k$. Consider one of the numbers y listed in r_k. By the Prime Number Theorem [Ing32], 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 can get 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 7, we are able to eliminate the random string required by Sipser at the cost of an additional $\log |A^{=n}|$ bits.

4 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 8. *There exists an infinite set A such that for every polynomial p, $\mathrm{CD}^{p,A}(x) \geq |x|/5$ for almost all $x \in A$.*

We extend and strengthen their result for CND complexity:

Theorem 9. *There exists an infinite set A such that $\mathrm{CND}^{2^{\sqrt{|x|}},A}(x) \geq |x|/4$ for all $x \in A$.*

The proof of Fortnow and Kummer of Theorem 8 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 Corollary 10 of Goldsmith, Hemachandra and Kunen [GHK92]. In Section 5, we will also describe a rough equivalence between this result and an "X-search" theorem of Impagliazzo and Tardos [IT89].

Using Theorem 9 we get the following corollary first proved by Goldsmith, Hemachandra and Kunen [GHK92].

Corollary 10 Goldsmith-Hemachandra-Kunen. *Relative to some oracle, there exists an infinite set in* **P** *with no infinite sparse subsets in* **NP**.

Proof: Let A from Theorem 9 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 6(3) it follows that $\mathrm{CND}^{A,p}(x) \leq O(\log(n))$. This contradicts the fact that $x \in S \subseteq A$ and Theorem 9. □

The above argument shows actually something stronger:

Corollary 11. *Relative to some oracle, there exists an infinite polynomial-time computable set with no infinite subset in* **NP** *of density less than $2^{n/9}$.*

5 Search vs. Decision in Exponential-Time

If **P** = **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 **EXP** = **NEXP** then one should find a witness of a nondeterministic exponential-time 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 12 [IT89]. *There exists a relativized world where* **EXP** = **NEXP** *but there exists a nondeterministic exponential-time Turing machine whose accepting paths cannot be found in exponential time.*

We can give a short proof of this theorem using Theorem 9.

Proof of Theorem 12: Let A be from Theorem 9.

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 $M^{A\oplus T}(w)$ 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 nondeterministic exponential time Turing machine with access to B whose accepting paths cannot be found in time exponential relative to B.

Consider the nondeterministic 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 9, 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 13. $\mathbf{CND}^{2^{\log^k n},A}(x|w_1,\ldots,w_{\log^k n}) \leq \log n + O(1)$.

Assuming Lemma 13, Theorem 12 follows since for each i, $|w_i| \leq i+1$. We thus have our contradiction with Equation (1).

Proof of Lemma 13: 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. □

Impagliazzo and Tardos [IT89] prove Theorem 12 using an "X-search" problem. We can also relate this problem to **CND** complexity and Theorem 9.

Definition 14. 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 12.

Theorem 15 [IT89]. *If the type of the queries is 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 12 to prove a similar bound for Theorem 15. One needs just to apply Theorem 9 relative to the strategy of the player.

One can also use Theorem 15 to prove a variant of Theorem 9. Suppose Theorem 9 fails. For any A and for every x in A there exists a small program that nondeterministically distinguishes x. For some x suppose we know p. We can find x by asking a **DNF** question based on p about the ith bit of x.

We do not in general know p but there are not too many possibilities. We can use an additional round of queries to try all programs and test all the answers in parallel. This will give us a general strategy for the X-search problem contradicting Theorem 15.

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

Theorem 17. *The following are equivalent:*

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

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

Definition 18. 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.

Corollary 19. *If* $\forall p_2\, \exists p_1, c\, \forall x, y : \mathbf{CD}^{p_1}(x \mid y) \leq \mathbf{CND}^{p_2}(x \mid y) + c\log(|x|)$ *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 $\phi_1, \ldots, \phi_m$ of n variables each such that at least one is satisfiable, there exists a i such that ϕ_i is satisfiable and*

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

Proof: The consequences in the corollary follow from the assumption that $\mathbf{FP}^{\mathbf{NP}[\log(n)]} = \mathbf{FP}^{\mathbf{NP}}_{tt}$ [JT95]. $\mathbf{FP}^{\mathbf{NP}[\log(n)]} = \mathbf{FP}^{\mathbf{NP}}_{tt}$ follows from Theorem 17. □

We can use Corollary 19 to get a complete collapse if there is only a constant difference between **CD** and **CND** complexity.

Theorem 20. *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. $\mathbf{P} = \mathbf{NP}.$

In fact Theorem 20 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 21. 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, fSAT 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 22. *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. *(1SAT,SAT) has a solution in* $\mathbf{P}$.
3. *For all time constructible f, (fSAT,SAT) has a solution in* $\mathbf{NP}[2\log(f(n)) + O(\log(n))]$.

Corollary 23. $\mathbf{FP}^{\mathbf{NP}[\log(n)]} = \mathbf{FP}^{\mathbf{NP}}_{tt}$ *implies the following:*

1. *For any k the promise problem ($2^{\log^k(n)}SAT$,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 17, Theorem 22, and Corollary 19. □

7 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 nondeterministic polynomial-time computation.

7.1 Enumerating Satisfying Assignments

Papadimitriou [Pap96] mentioned the following proposition:

Proposition 24. *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 25. *Proposition 24 is equivalent to $(1SAT, SAT)$ has a solution in* **P**.

In Proposition 24, 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 24 is equivalent to $\mathbf{P} = \mathbf{NP}$.

Proof of Theorem 25: The implication of $(1SAT, SAT)$ having a solution in **P** is straightforward. We concentrate on the other direction.

Let $d = \|A\|$. By Lemma 4 and Theorem 22 we have that for every element x of A, $\mathrm{C}^q(x|\phi) \leq 2\log d + c\log n$ 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. □

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

Toda and Watanabe [WT93] showed that $\mathbf{F}_{sat} \bigcap \mathbf{FP}^{\mathbf{NP}}_{tt} \neq \emptyset$ relative to a random oracle. On the other hand Buhrman and Thierauf [BT96] showed that there exists an oracle where $\mathbf{F}_{sat} \bigcap \mathbf{FP}^{\mathbf{NP}}_{tt} = \emptyset$. Their result also holds relative to the set constructed in Theorem 9.

Theorem 27. *Relative to the set A constructed in Theorem 9, $\mathbf{F}_{sat} \bigcap \mathbf{FP}_{tt}^{\mathbf{NP}} = \emptyset$.*

Proof: For some n, let ϕ be the formula on n variables such that $\phi(x) = \top$ if and only if $x \in A$. Suppose $\mathbf{F}_{sat} \bigcap \mathbf{FP}_{tt}^{\mathbf{NP}} \neq \emptyset$. It now follows by Lemma 26 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 $x \in A$, $\mathbf{CND}^{2^{\sqrt{|x|}},A}(x) \geq |x|/4$. □

7.3 Isolating Satisfying Assignments

In this section we take a Kolmogorov complexity view of the statement and proof of the famous Valiant-Vazirani lemma [VV86]. 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 28. *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 28 by choosing r and the $O(\log|\phi|)$ program randomly.

We show how to derive the Valiant-Vazirani Lemma from Sipser's Lemma (Lemma 7). Note Sipser's result predates Valiant-Vazirani by a couple of years.

Proof of Lemma 28: Let $n = |\phi|$.

Consider the set A of satisfying assignments of ϕ. We can apply Lemma 7 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. □

Acknowledgments

We would like to thank José Balcázar and Leen Torenvliet for their comments on this subject. We thank John Tromp for the current presentation of the proof of Lemma 4. We also thank Sophie Laplante for her important contributions to Section 6. We thank Richard Beigel, Bill Gasarch and Leen Torenvliet for comments on earlier drafts. We thank the anonymous reviewers for helpful comments.

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 *Proc. Structure in Complexity Theory 9th Annual Conference*, pages 43-52. IEEE computer society press, 1994.

[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 *Proc. 3rd ACM Symposium 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.

[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.

[Ing32] A.E. 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 *Proc. 30th IEEE Symposium on Foundations of Computer Science*, pages 222-227, 1989.

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

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

[LV93] Ming Li and P.M.B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications.* Springer-Verlag, 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.

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

[VV86] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.

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