

Infeasibility of Instance Compression and Succinct PCPs for NP*

Lance Fortnow
Rahul Santhanam

March 19, 2010

Abstract

The OR-SAT problem asks, given Boolean formulae ϕ_1, \dots, ϕ_m each of size at most n , whether at least one of the ϕ_i 's is satisfiable.

We show that there is no reduction from OR-SAT to any set A where the length of the output is bounded by a polynomial in n , unless $\text{NP} \subseteq \text{coNP}/\text{poly}$, and the Polynomial-Time Hierarchy collapses. This result settles an open problem proposed by Bodlaender et. al. [BDFH08] and Harnik and Naor [HN06b] and has a number of implications.

- A number of parametric NP problems, including Satisfiability, Clique, Dominating Set and Integer Programming, are not *instance compressible* or *polynomially kernelizable* unless $\text{NP} \subseteq \text{coNP}/\text{poly}$.
- Satisfiability does not have PCPs of size polynomial in the number of variables unless $\text{NP} \subseteq \text{coNP}/\text{poly}$.
- An approach of Harnik and Naor to constructing collision-resistant hash functions from one-way functions is unlikely to be viable in its present form.
- (Buhrman-Hitchcock) There are no subexponential-size hard sets for NP unless NP is in co-NP/poly.

We also study *probabilistic* variants of compression, and show various results about and connections between these variants. To this end, we introduce a new strong derandomization hypothesis, the Oracle Derandomization Hypothesis, and discuss how it relates to traditional derandomization assumptions.

1 Introduction

Bodlaender et al. [BDFH08] and Harnik and Naor [HN06b] raise the following question, which has relevance to a wide variety of areas, including parameterized complexity, cryptography, probabilistically checkable proofs and structural complexity. This question asks whether the OR-SAT problem (given a list of formulae, is at least one satisfiable) is compressible.

Question 1.1 *Is there a function f that, given as input m Boolean formula ϕ_1, \dots, ϕ_m where each ϕ_i has length at most n (possibly much less than m), outputs a Boolean formula, and has the following properties?*

*An earlier version of this paper appeared in the Proceedings of the 40th Symposium on the Theory of Computing [FS08].

- f is computable in time polynomial in m and n ,
- $f(\phi_1, \dots, \phi_m)$ is satisfiable if and only if at least one of the ϕ_i is satisfiable, and
- $|f(\phi_1, \dots, \phi_m)|$ is bounded by a polynomial in n .

We essentially settle this question in the negative by showing that a positive answer to Question 1.1 implies that the polynomial-time hierarchy collapses. We actually show the following stronger statement, in which f is allowed to map to an arbitrary set.

Theorem 1.2 *If NP is not contained in coNP/poly then there is no set A and function f such that given m Boolean formula ϕ_1, \dots, ϕ_m where each ϕ_i has length at most n , f has the following properties*

- f is computable in time polynomial in m and n ,
- $f(\phi_1, \dots, \phi_m) \in A$ if and only if at least one of the ϕ_i is satisfiable, and
- $|f(\phi_1, \dots, \phi_m)|$ is bounded by a polynomial in n .

Bodlaender et al. [BDFH08] arrive at Question 1.1 from the perspective of parameterized complexity. Parameterized complexity asks about the complexity of NP problems based on some inherent parameter, like the number of variables in a formula or clique size in a graph. In parameterized complexity, feasibility is identified with *fixed-parameter tractability*. A problem is fixed-parameter tractable (FPT) if it has an algorithm running in time $f(k)n^{O(1)}$ where n is the input size and $f(k)$ is an arbitrary function of the parameter k . One technique to show fixed-parameter tractability is *kernelization*, where one reduces the solution of the given instance to the solution of an instance of size depending only on the parameter; this new instance is called a “problem kernel”. Chen et al. [CCDF97] show that a problem is FPT if and only if it has a kernelization. However, in general, the kernel can be arbitrarily long as a function of the parameter.

It is a fundamental question in parameterized complexity as to which problems have polynomial-size kernels [FG06, GN07]. Bodlaender et al. [BDFH08] develop a theory of polynomial kernelizability, and define a notion of *strong distillation* which is useful in this context. A problem L has a strong distillation function if there is a polynomial-time computable function f that takes inputs x_1, \dots, x_m and outputs a y with $|y|$ bounded by a polynomial in $\max_i |x_i|$, such that y is in L if and only if at least one of the x_i ’s are in L . Question 1.1 is equivalent to asking if SAT has a strong distillation. Bodlaender et al. conjectured that the answer to Question 1.1 is negative, and under that conjecture showed that the parameterized versions of several NP-complete problems do not have polynomial kernelizations, including k -Path and k -Cycle. Theorem 1.1 confirms the conjecture of Bodlaender et al. modulo a widely-believed complexity-theoretic assumption, a rare connection between parameterized complexity and a traditional complexity-theoretic hypothesis.

Harnik and Naor [HN06b] arrived at essentially Question 1.1 with a very different motivation, *cryptographic* in nature. An NP language L is *instance compressible* if there is some polynomial-time computable function f and a set A in NP such that x is in L if and only if $(f(x), 1^{|x|})$ is in A , and $|f(x)|$ is bounded by a polynomial in the length of a *witness* for x .¹ They showed

¹Harnik and Naor actually allow $|f(x)|$ to be bounded by a polynomial in both the length of the witness and $\log|x|$. This variation is actually equivalent to our formulation, as discussed in Section 2

that if the Satisfiability problem is compressible then collision resistant hash functions can be constructed from one-way functions. If an even stronger compressibility assumption holds, then oblivious transfer protocols can be constructed from one-way functions. This would imply that public-key cryptography can be based on one-way functions, solving one of the outstanding open problems in theoretical cryptography.

The cryptographic reductions of Harnik and Naor also follow from a weaker assumption than compressibility of SAT, namely the compressibility of the OR-SAT problem. In the OR-SAT problem, a list of m formulae $\phi_1 \dots \phi_m$ each of size at most n is given as input, and a “yes” instance is one in which at least one of these formulae is satisfiable. Note that the size of a witness for OR-SAT is bounded above by n , hence compressibility of OR-SAT implies a positive answer to Question 1.1. Harnik and Naor describe a hierarchy of problems including Satisfiability, Clique, Dominating Set and Integer Programming, the compression of any of which would imply the compression of the OR-SAT problem. Thus Theorem 1.2 shows that none of these problems are compressible unless the polynomial-time hierarchy collapses. From a cryptographic point of view, our result indicates that the approach of Harnik and Naor may not be viable in its current form. But rather than considering this as a purely negative result, we hope it stimulates research into whether weaker and more plausible conditions already suffice for their constructions to work.

Theorem 1.2 also has applications to probabilistically checkable proofs (PCPs), and to structural complexity.

The result is directly relevant to the question of whether there are succinct PCPs for NP, which has been raised recently by Kalai and Raz [KR07]. A succinct PCP for an NP language L is a probabilistically checkable proof for L where the size of the proof is polynomial in the witness size n rather than in the instance size m . Current proofs of the PCP theorem [AS98, ALM⁺98, Din07] do not yield such PCPs. Kalai and Raz state that the existence of succinct PCPs “would have important applications in complexity theory and cryptography, while a negative result would be extremely interesting from a theoretical point of view”. We show such a negative result: unless $\text{NP} \subseteq \text{coNP}/\text{poly}$ and the Polynomial Hierarchy collapses, SAT does not have succinct PCPs, nor do problems like Clique and DominatingSet. On the other hand, polynomially kernelizable problems such as VertexCover do have succinct PCPs.

Buhrman and Hitchcock [BH08] use Theorem 1.2 to show implications of NP-complete problems reducing to subexponential-size sets. Mahaney [Mah82] showed that if there are NP-complete sparse sets (polynomial number of strings at every length) then $\text{P}=\text{NP}$. Karp and Lipton [KL82] show that if NP has a Turing-reduction to sparse sets then NP is in P/poly . But we had no known polynomial-time consequences of reductions to larger sets. Buhrman and Hitchcock use our result to show that there are no NP-complete sets of size $2^{n^{\sigma(1)}}$ unless NP is in coNP/poly . More generally they show that if NP is not in coNP/poly then for every set A of size $2^{n^{\sigma(1)}}$, there is no reduction from Satisfiability to A using $n^{1-\epsilon}$ adaptive queries to A for any fixed $\epsilon > 0$.

All the results mentioned so far concern compressibility using an f that is computable in deterministic polynomial time or in probabilistic polynomial time with very small error. We also study the case of general probabilistic compression. Here we do not have strong negative results, but we do have negative results for restricted notions as well as connections between different notions. Under a strong derandomization assumption that we call the Oracle Derandomization Hypothesis, we extend our infeasibility result for succinct PCPs to the more general case of gap amplification without blowing up the parameter by more than a polynomial factor. This has applications to the theory of hardness of approximation in the context of parameterized complexity.

In Section 2 we formally define the model and the problems. In Section 3 we prove our main result Theorem 3.1. In Section 4 we describe some applications of our main result and techniques for succinct PCPs, kernelization and cryptography. In Section 5 we give results about probabilistic compression. We discuss the feasibility of the Oracle Derandomization Hypothesis in Section 6.

2 Preliminaries

2.1 Basic Complexity Notions

For the definitions of basic complexity classes such as NP, P, E, BPP and AM, we refer the reader to the Complexity Zoo². The work of Garey and Johnson [GJ76] is an excellent compendium of NP-complete problems. There are a number of good reference works on parameterized complexity [DF99, FG06, Nie06].

We use $\text{SIZE}(s)$ to refer to the class of languages accepted by Boolean circuits of size $O(s)$, and $\text{NSIZE}(s)$ to refer to the class of languages accepted by non-deterministic Boolean circuits of size $O(s)$. Given a complexity class C , *i.o.* C is the class of languages L for which there is some $L' \in C$ such that L' agrees with L on infinitely many input lengths.

2.2 Instance Compression

Motivated by cryptographic applications, Harnik and Naor [HN06b] introduced the notion of instance compression for languages in NP. Informally, a language $L \in \text{NP}$ is instance compressible if there is a polynomial time algorithm that, for each instance x , produces an instance $C(x)$ of length polynomial in the *witness size* for x , such that $C(x) \in L$ iff $x \in L$. When the witness size is significantly smaller than the instance size, this gives a significant *compression* of the original instance x with respect to membership in L .

Harnik and Naor actually allow the size of the compressed instance to be polynomial in the logarithm of the instance size m as well as in the witness size n , but this does not give rise to a more relaxed notion, for the following reason. There are two cases: either $n > \log(m)$, or $n \leq \log(m)$. In the first case, being polynomial in $(n + \log(m))$ is equivalent to being polynomial in n . In the second case, the instance can be decided in polynomial time in m just by brute-force search over witnesses. Given a compression algorithm f according to the Harnik-Naor notion, we can define a compression algorithm f' according to our notion which checks which case holds and supposing the second case holds, solves its instance in polynomial time and correspondingly outputs a constant-sized “yes” or “no” instance instead of the instance produced by f .

Harnik and Naor [HN06b] define instance compression relative to languages. Since an NP language could correspond to different relations with varying witness lengths, this notion is not robust when considering compression of languages. Hence we choose to define instance compression for *parametric* problems - problems in which the instance is given with an additional parameter, and the length of the compressed instance is required to be polynomial in that parameter. Note that for most problems of interest in NP, such as Vertex Cover and Clique, the traditional instance specification includes the relevant parameter (respectively, the size of the vertex cover and size of the clique). For other problems, such as Satisfiability, the relevant parameter is implicit in the

²http://qwiki.caltech.edu/wiki/Complexity_Zoo

instance, eg., the description of a formula ϕ also reveals the number of variables in ϕ , and hence the witness length.

Our formulation has three other advantages. First, it enables us to pose the question of whether a given problem has small witnesses or not. Second, the parameter could potentially be used to represent quantities other than the witness length. Third, the formulation enables us to study compression of languages that are not in NP. In this paper, though, we'll focus on natural parametric versions of NP problems, where the parameter corresponds to the witness length.

Definition 2.1 *A parametric problem is a subset of $\{\langle x, 1^n \rangle \mid x \in \{0, 1\}^*, n \in \mathbb{N}\}$.*

Definition 2.2 *Let L be a parametric problem and $A \subseteq \{0, 1\}^*$. L is said to be compressible within A if there is a polynomial $p(\cdot)$, and a polynomial-time computable function f , such that for each $x \in \{0, 1\}^*$ and $n \in \mathbb{N}$, $|f(\langle x, 1^n \rangle)| \leq p(n)$ and $\langle x, 1^n \rangle \in L$ iff $f(\langle x, 1^n \rangle) \in A$. L is compressible if there is some A for which L is compressible within A . L is self-compressible if L is compressible within L .*

Harnik and Naor also consider variants where the compressed size need not be polynomially bounded in n and $\log(m)$ but is constrained to be at most some larger non-trivial function in n and $\log(m)$. Our negative results on compressibility scale smoothly in terms of the strength of the assumption required as the constraint on the compressed size becomes weaker.

We will also be interested in *probabilistic* compression.

Definition 2.3 *Let L be a parametric problem and $A \subseteq \{0, 1\}^*$. L is said to be probabilistically compressible with error $\epsilon(n)$ within A if there is a probabilistic polynomial-time computable function f such that for each $x \in \{0, 1\}^*$ and $n \in \mathbb{N}$, with probability at least $1 - \epsilon(|x|)$ we have:*

1. $|f(\langle x, 1^n \rangle)| \leq \text{poly}(n)$
2. $f(\langle x, 1^n \rangle) \in A$ iff $x \in L$

L is probabilistically compressible if there is an A such that L is probabilistically compressible within A with error $1/3$. L is errorless compressible if there is an A such that L is probabilistically compressible within A with error 0.

We say that a probabilistic compression function f has randomness complexity R if it uses at most R random bits.

Note that errorless compression is a distinct notion from deterministic compression.

We also define, informally but unambiguously, *non-uniform* and *average-case* versions of compression.

Definition 2.4 *A parametric problem L is said to be compressible with advice $s(\cdot, \cdot)$ if the compression function is computable in deterministic polynomial time when given access to an advice string of size $s(|x|, n)$ which depends only on $|x|$ and n but not on x . L is non-uniformly compressible if s is polynomially bounded in m and n .*

Definition 2.5 *A parametric problem L is $a(\cdot, \cdot)$ -compressible on average if the compression function works correctly for at least a fraction $a(m, n)$ of instances $\langle x, 1^n \rangle$ where $|x| = m$.*

Compression on average with advice is defined by combining the two notions in the obvious way; similarly probabilistic compression with advice is defined by combining the notion of probabilistic compression with the notion of non-uniform compression. Note that deterministic compression is 1-compression on average using 0 bits of advice.

We will mainly be discussing compressibility of parametric problems in NP. We next define some of the problems we will be studying.

Definition 2.6 $SAT = \{ \langle \phi, 1^n \rangle \mid \phi \text{ is a satisfiable formula, and } n \text{ is at least the number of variables in } \phi \}$.

Definition 2.7 $VC = \{ \langle G, 1^{k \log(m)} \rangle \mid G \text{ has a vertex cover of size at most } k, \text{ where } m = |G| \}$.

Definition 2.8 $Clique = \{ \langle G, 1^{k \log(m)} \rangle \mid G \text{ has a clique of size at least } k, \text{ where } m = |G| \}$.

Note that our representation of the inputs to VC and Clique is slightly non-standard in that we specify $k \log(m)$ rather than just the size k of the object for which we're searching. This is because we would like the parameter to accurately reflect the *witness* size.

Definition 2.9 $OR-SAT = \{ \langle \{ \phi_i \}, 1^n \rangle \mid \text{At least one } \phi_i \text{ is satisfiable, and each } \phi_i \text{ has size at most } n \}$.

Similarly we define the parametric problems DominatingSet and IntegerProgramming in the natural way.

One imagines it would be useful to have a structure theory relating compressibility of these various problems, analogous to the theory of NP-completeness for decidability. Harnik and Naor initiated just such a theory in their paper, by using the notion of “W-reduction” [HN06b] to define a hierarchy $VC_0 \subseteq VC_{OR} \subseteq VC_1 \subseteq VC_2 \dots$ of problems in NP, closely related to similar hierarchies in the theory of parameterized complexity. Their notion is not rigorous as defined, but becomes rigorous when each class VC_i is defined as a class of parametric problems. In this paper, we are not concerned with the hierarchy itself as much as with the basic notion of a W-reduction, which allows us to translate infeasibility of compression results from one parametric problem to another.

Definition 2.10 *Given parametric problems L_1 and L_2 , L_1 W-reduces to L_2 (denoted $L_1 \leq_W L_2$) if there is a polynomial-time computable function f and polynomials p_1 and p_2 such that:*

1. $f(\langle x, 1^{n_1} \rangle)$ is of the form $\langle y, 1^{n_2} \rangle$ where $|y| \leq p_1(n_1 + |x|)$ and $n_2 \leq p_2(n_1)$.
2. $f(\langle x, 1^{n_1} \rangle) \in L_2$ iff $\langle x, 1^{n_1} \rangle \in L_1$.

The semantics of a W-reduction is that if L_1 W-reduces to L_2 , it's as hard to compress L_2 as it is to compress L_1 .

Proposition 2.11 *If $L_1 \leq_W L_2$ and L_2 is compressible, then L_1 is compressible.*

Armed with this notion, we can investigate the relative compressibility of the parametric problems we defined earlier.

First, we ask: are there any natural parametric versions of NP-complete problems which can be shown to be compressible? The answer is yes, for problems for which the parameter is polynomially related to the input length. For instance, in the problem 3-SAT, the size of the formula is polynomially related to the number of variables (after deleting repeated clauses), hence any non-redundant version of the formula is itself a compressed instance, by our definition.

There still remains the question of whether there are less trivial compression algorithms for natural parametric problems. Using a close relationship between compressibility and the technique of kernelization in parameterized complexity, Harnik and Naor show the following:

Theorem 2.12 [DF99, HN06b] *VC is self-compressible.*

As for the remaining problems, all that was previously known were reductions between them.

Proposition 2.13 [HN06b] *OR-SAT \leq_W Clique \leq_W SAT \leq_W DominatingSet. Also SAT \leq_W IntegerProgramming.*

2.3 Chernoff bounds

In some of our proofs, we will need the following Chernoff bounds.

Proposition 2.14 *Let $X_1, X_2 \dots X_n$ be independent $\{0,1\}$ random variables such that $E(\sum X_i) = \mu$. Then, for any $t > 0$, $\Pr(|\sum X_i - \mu| > t) < 2e^{-2t^2/n}$.*

This formulation follows immediately from known bounds (e.g. [AS08, Theorem A.1.16]).

3 Infeasibility of Deterministic Compression

In this section, we show that deterministic compression of SAT implies that the Polynomial Hierarchy collapses. In fact, the conclusion holds even under the milder assumption that OR-SAT is deterministically compressible. Theorem 3.1 below is just Theorem 1.2 re-phrased using the terminology of compression.

Theorem 3.1 *If OR-SAT is compressible, then $\text{coNP} \subseteq \text{NP}/\text{poly}$, and hence PH collapses.*

Proof. Let ϕ be any formula of size at most m consisting of the disjunction of formulae each of size at most n . By the assumption on compressibility of OR-SAT, there is a language A and a function f computable in deterministic time $\text{poly}(m)$ such that $|f(\phi, 1^n)| \leq O(\text{poly}(n, \log(m)))$, and ϕ is satisfiable iff $f(\phi, 1^n) \in A$. Let c be a constant such that the length of compressed instances on OR-SAT formulae of size at most m and parameter at most n is at most $k = (n + \log(m))^c$.

Now let S be the set of unsatisfiable formulae of size at most n and T be the set of strings in \bar{A} of length at most k . The function f induces a map $g : S^{m/n} \rightarrow T$, since a tuple of m/n formulae of size n can be represented in size m in a natural encoding scheme, and the correctness of the reduction implies that a disjunction of m/n unsatisfiable formulae maps to a string in \bar{A} of length at most k .

Our strategy will be as follows: we will attempt to find a $\text{poly}(n)$ size set C of strings in T , such that any formula in S is contained in at least one tuple that maps to a string in C under the mapping g . If such a set C exists, then we have a *proof with advice* of unsatisfiability of a formula z of size n , by guessing a tuple of m/n formulae of size at most n such that z belongs to the tuple, and then checking if the tuple maps to a string in C . The check whether the tuple maps to a string in C can be done with polynomial advice, by enumerating the strings in C in the advice string. Any unsatisfiable formula will have such a proof with advice, just by the definition of C . Conversely, any tuple containing a satisfiable formula will map to a string in A and hence

to a string in \bar{C} , implying that no satisfiable formula will have such a proof. If $m = \text{poly}(n)$, then the proof is polynomial-size, and since the advice is polynomial-size as well by assumption on C , we get that $\text{coNP} \subseteq \text{NP}/\text{poly}$.

Thus the proof reduces to showing the existence of a set C with the desired properties. The proof is via a purely combinatorial argument. We employ a greedy strategy, trying to “cover” as many strings in S as possible with each string we pick in C . We prove that such a greedy strategy terminates after picking polynomially many strings.

We pick the set C in stages, with one string picked in each stage. Let C_i be the set of strings picked at or before stage i , $|C_i| = i$. Let S_i denote the set of strings y in S , such that y is not part of a tuple that maps to a string in C_i under g . Let $X = S^{m/n}$, and $X_i \subseteq X$ be the set of tuples that do not belong to the pre-image set of C_i (under the mapping g). Note that $X_i = S_i^{m/n}$, since a tuple belongs to X_i iff every element of the tuple belongs to S_i .

At stage 0, C_i is the empty set, $S_i = S$ and $X_i = X$. We proceed iteratively as follows. If S_i is empty, we stop. Otherwise, at stage i , we pick the string y in T with the maximum number of pre-images in X_{i-1} , and set $C_i = C_{i-1} \cup \{y\}$.

We show that if m is picked appropriately as a function of n , then this process concludes within $\text{poly}(n)$ stages. It is enough to show that the size of S_i decreases by at least a constant factor in each stage. Since $|S| \leq 2^{n+1}$, this implies that the process concludes after $O(n)$ stages.

Now we analyze the decrease in the size of S_i . Recall that at stage i , we add to C_i the string y in T with the maximum number of pre-images in X_{i-1} . Let Y denote the set of pre-images of y under the mapping g within X_{i-1} . Since $|T| \leq 2^{k+1}$, by the pigeonhole principle, $|Y|$ is at least $|X_{i-1}|/2^{k+1}$, i.e., $|Y| \geq |S_{i-1}|^{m/n}/2^{k+1}$.

Now, every element of a tuple in Y is in S_{i-1} because $Y \subseteq X_{i-1}$. On the other hand, S_i is formed from S_{i-1} by removing every member of S_{i-1} which is in a tuple in Y . Thus, Y is contained in the set of all m/n -tuples with elements in $S_{i-1} - S_i$, and so $|Y| \leq |S_{i-1} - S_i|^{n/m}$. Combining this with the inequality at the end of the previous para, we have that $|S_{i-1} - S_i| \geq |S_{i-1}|/2^{(k+1)n/m}$. Since $k \leq (\log(m) + n)^c$ for some constant c , we can pick a constant $c' > c$ large enough so that $(k+1)n < m$ when $m = n^{c'}$ and n is large enough. For this choice of m , we have that $|S_{i-1} - S_i| \geq |S_{i-1}|/2$ for large enough n , and therefore that $|S_i| \leq |S_{i-1}|/2$.

Thus, for this choice of m , we have that the set C has size $O(n)$ and that m is polynomially bounded in n . By the argument given earlier, this gives polynomial-sized proofs with polynomial advice for unsatisfiable formulae, and implies that $\text{coNP} \subseteq \text{NP}/\text{poly}$. From a result of Yap [Yap83], it follows that PH collapses to the third level. \square

Since OR-SAT W-reduces to SAT by Proposition 2.13, our infeasibility result also applied to compression of general satisfiable formulae with small witnesses. From Proposition 2.13, we also get consequences for other natural parametric problems.

Corollary 3.2 *If SAT is compressible, then $\text{coNP} \subseteq \text{NP}/\text{poly}$, and PH collapses. The same conclusion holds if Clique, DominatingSet or IntegerProgramming are compressible.*

We next extend our infeasibility result to errorless compression and compression with very small error. This extension uses “Adleman’s trick” [Adl78] to embed a “good” random string in the advice, and then applies the argument for deterministic compression.

Theorem 3.3 *If OR-SAT is compressible with error $< 2^{-m}$, where m is the instance size, then $\text{NP} \subseteq \text{coNP}/\text{poly}$ and PH collapses.*

Proof. The key observation is that compression with error $< 2^{-m}$ implies *non-uniform* compression. This is because, by a union bound, there must be some choice of randomness that yields a correct compressed instance for *each* instance of length m . Let z be such a random string. z is of size at most $\text{poly}(m)$, since the compression algorithm runs in probabilistic polynomial time. Now we just use the same argument as in the proof of Theorem 3.1 except that we also include z in the advice string when defining the proof system with advice for unsatisfiable formulae. In the earlier argument, the mapping from a tuple to a string could be performed deterministically; in the present case, we just perform it using z as the random string for the probabilistic compression function. \square

Our infeasibility result also extends to non-uniform compression, using the same proof as for Theorem 3.1.

Corollary 3.4 *if OR-SAT is non-uniformly compressible, then $\text{NP} \subseteq \text{coNP}/\text{poly}$ and PH collapses.*

4 Applications

In this section, we discuss the implications of our infeasibility result in various other research areas. First, we show a connection to succinct PCPs, which have attracted interest recently in the context of the study of short zero-knowledge proofs/arguments for NP. Second, we show that our result connects the theory of parameterized complexity to classical complexity theory, specifically with regard to the “kernelization” technique in parameterized complexity. Third, we discuss some implications for the classical cryptographic question of basing public-key cryptography for one-way functions. Here our results are mainly negative, showing the infeasibility of certain approaches to this problem which motivated Harnik and Naor to define the notion of instance compression.

4.1 Succinct PCPs

The PCP theorem [AS98, ALM⁺98] is one of the landmark achievements in complexity theory. It states that any NP-complete language has polynomial-size proofs that can be verified probabilistically by reading only a *constant* number of bits of the proof.

Here “polynomial-size” means that the proof size is polynomial in the size of the input. It’s natural to ask whether the proof size can be made polynomial in the size of the *witness* instead, giving more efficient proofs. The witness constitutes a proof which can be verified by reading all the bits of the proof - perhaps the number of bits read can be reduced to a constant by just blowing up the size of the proof polynomially? The known proofs of the PCP theorem [AS98, Din07] do not achieve this, since the size of the proof obtained is polynomial in the input size and not just in the witness size. But a priori, a more efficient argument is conceivable.

Kalai and Raz [KR07] raise this question in their paper on “interactive PCPs”, and state that either a positive answer or a negative answer would be interesting. Here we give strong evidence for a negative answer by showing a strong connection between succinct PCPs and compressibility, and then invoking the results in the previous section.

We begin by defining “succinct PCPs”, which are intuitively PCPs where the proof size depends polynomially on the witness length rather than the input length. In our framework of parametric problems, this corresponds to the proof size depending polynomially on the parameter rather than the input length.

Definition 4.1 Let L be a parametric problem. L is said to have succinct PCPs with completeness c , soundness s , proof size S and query complexity q if there is a probabilistic polynomial-time oracle machine V such that the following holds for any instance $\langle x, 1^n \rangle$:

1. If $\langle x, 1^n \rangle \in L$, then there is a proof y of size $S(n)$ such that on input $\langle x, 1^n \rangle$ and with oracle access to y , V makes at most q queries to y on any computation path, and accepts with probability at least c .
2. If $\langle x, 1^n \rangle \notin L$, then there for any string y of size $S(n)$, on input $\langle x, 1^n \rangle$ and with oracle access to y , V makes at most q queries to y on any computation path, and accepts with probability at most s .

L is said to have succinct PCPs if it has succinct PCPs with completeness 1, soundness $1/2$, proof size $\text{poly}(n)$ and query complexity $O(1)$.

For standard NP-complete problems such as SAT, CLIQUE and VERTEX-COVER, we abuse notation and say the problem has succinct PCPs if the natural parametric version of it does.

We show a close connection - a near-equivalence - between compressibility and the existence of succinct PCPs. In one direction, the existence of succinct PCPs implies compressibility with small error.

Theorem 4.2 *If SAT has succinct PCPs, then SAT is self-compressible with error less than 2^{-m} .*

We informally describe the basic plan of the proof, and then give the details.

Let us first assume, for the sake of simplicity, that the verifier only uses a logarithmic amount of randomness, say $r \log(n)$ for some constant r . Suppose that the verifier makes at most q queries on any computation path, for q a constant. Note that the size of the proof is effectively at most qn^r , since at most this many proof bits can be read by the verifier.

Given an input formula of size m with n variables, we use the hypothesis that succinct PCPs exist to find an equivalent formula of size $O(n^r)$. The variables of the formula correspond to proof bits, and there is a clause in the formula corresponding to each computation path of the verifier. The clause corresponding to a computation path encodes whether the proof bits read on that path cause the verifier to accept on that path. Note that once the input formula is fixed, this is just a function of the proof bits read on that path, and hence can be expressed as a CNF formula of size at most $O(q2^q)$ on the variables corresponding to the proof bits read on that path. The equivalent formula we construct is just the conjunction of the formulae corresponding to each computation path. Note that the size of this formula is at most $O(q2^q)n^r$, which is polynomial in n .

To argue correctness, we use the hypothesis that the verifier corresponds to a PCP. If the input formula is satisfiable, there is some setting of the proof bits such that the verifier accepts on each computation path, and hence some setting of the variables in the compressed formula such that the formula is satisfiable. If the input formula is unsatisfiable, for any setting of the proof bits, at least one computation path rejects (in fact, at least a $1/2$ fraction of them reject) and hence the compressed formula is unsatisfiable.

The above argument works when the randomness complexity of the verifier is low. In general, this may not be the case, and we do not have enough time to compute a formula for each computation path. However, in this case, we can use the fact that the proof size and query complexity are small together with a sampling argument to prove that there is probabilistic compression with very low error. We give details below.

Proof of Theorem 4.2. Assume that SAT has succinct PCPs with proof size n^C for some constant C , query size q (where q is a constant) and randomness complexity R . Since the verifier runs in probabilistic polynomial time, we have that $R = O(\text{poly}(m))$.

Our self-compression algorithm works as follows. It samples independently m^2 random strings $r_1, r_2 \dots r_{m^2}$ each of length R . For each r_i , running the verifier for the succinct PCP with random string r_i corresponds to a function f_{r_i} on q bits of the proof such that the verifier accepts if and only if the function evaluates to 1 on those bits. Moreover, given the verifier, a canonical CNF formula of size $O(q2^q)$ for this function can be computed explicitly in time $\text{poly}(m)$. Note that since the proof has size at most n^C , there are at most n^{Cq2^q} such functions. Here, the input formula is fixed, and hence the size of the input formula does not factor into the bound. The self-compression algorithm computes a canonical CNF formula of size at $O(q2^q)$ for each $f_{r_i}, i = 1 \dots m^2$, and outputs the conjunction of these formulae, omitting duplicate formulae. This is a SAT formula, and we need to show that the formula has size at most $\text{poly}(n)$, and that with error $< 2^{-m}$, the compressed formula is satisfiable if and only if the input formula is satisfiable.

The size bound follows from the upper bound on the total number of functions on q bits of the proof, together with the fact that we remove duplicates. For the error bound, it follows from the definition of PCPs that if the input formula is satisfiable, then the compressed formula is also satisfiable, since the verifier accepts with probability 1 on a correct proof. If the input formula is not satisfiable, then for any proof, the verifier accepts with probability at most $1/2$. In this case, the compressed formula we output may not be the conjunction of *all* possible formula corresponding to runs of the verifier, but we will argue that with very high probability, the fraction of strings r such that f_r is omitted is less than $1/2$, and hence that the compressed formula is still unsatisfiable even with the formulae corresponding to these f_r omitted from the conjunction. Indeed, if the fraction of such r were greater than or equal to $1/2$, the probability that no such r is sampled as some r_i is at most $1/2^{m^2}$, since the sampling is done at random. Hence, with probability at least $1 - 1/2^{m^2}$, the compressed formula is unsatisfiable if the original formula is unsatisfiable. Thus the error is at most $1/2^{m^2}$. \square

Using a more refined approach, we can show the infeasibility of PCPs with non-negligible gap between soundness and completeness, not just of PCPs with completeness 1. We provide a sketch of the proof indicating in what respects the proof differs from the proof of Theorem 4.2, since a formal description would be very technical.

Theorem 4.3 *If SAT has succinct PCPs with completeness c , soundness s , proof size $\text{poly}(n)$ and query complexity $O(1)$, where $c + s$ is computable in time $\text{poly}(m)$ and $c - s \geq 1/\text{poly}(n)$, then SAT is self-compressible.*

Proof Sketch. We follow the basic framework of the proof of Theorem 4.2. We compute different functions corresponding to independent runs of the verifier, and then take the conjunction of constraints derived from these functions in order to produce a compressed instance. The compressed instance will not directly be a SAT instance, however it will be an instance of an NP language, hence we can reduce it to a SAT instance of polynomial size.

The fact that the verifier does not have completeness 1 does complicate the argument. Let us call a function on $O(1)$ bits of the proof corresponding to some run of the verifier a “test function”. Now we also need a notion of the “weight” of a test function, which is essentially the measure of random strings on which the verifier run corresponds to the function. Our compressed formula will consist of the conjunction of constraints derived from test functions. Each constraint is on $O(1)$

variables, and we pick an arbitrary representation for these constraints (say, a list consisting of the names of variables involved, together with the truth table of the test function). The constraints will be present with multiplicity, where the multiplicity of a constraint is approximately the weight of its test function. This is designed to ensure that in the case of satisfiable input formulae, approximately a fraction c of constraints are satisfiable, and in the case of unsatisfiable ones, approximately a fraction s of them are satisfiable. We sample enough test functions and choose multiplicities with enough granularity (while still keeping the multiplicity of any one constraint polynomial in n) so that with probability more than $1 - 2^{-m}$, at least $(c + s)/2$ constraints are satisfiable if the input formula is satisfiable, and fewer than $(c + s)/2$ are satisfiable if the input formula is not satisfiable. This is argued with Chernoff bounds, using the facts that there is a non-negligible separation between c and s and that we sample enough. When we sample test functions, we also update estimates of their weights, sampling enough to ensure that the estimates are very accurate. The estimates are normalized and truncated to their $O(\log(n))$ high-order bits to ensure that multiplicities are at most polynomial. The compressed instance produced in this way is still of size polynomial in n , and is an instance of a general constraint satisfaction problem in NP , where we are given a set of constraints explicitly together with a parameter k and asked whether at least k of those constraints are satisfiable. Here the parameter $k = (c + s)/2$, which by our assumption can be computed explicitly in time $\text{poly}(m)$. This already establishes compressibility of SAT with error $< 2^{-m}$ from the assumption on succinct PCPs; in order to get self-compressibility, we reduce the constraint satisfaction instance to a SAT instance using the Cook-Levin reduction [Coo71], which preserves the instance size up to a polynomial factor.

Note that it is actually sufficient to obtain a good approximation (i.e., an approximation to within an arbitrary $1/\text{poly}(n)$ additive term) of $c + s$ or even of just one of c and s , in order to obtain our result. But we do not see how to carry the compression through without access to *any* information about c or s apart from the fact that they are non-negligibly separated. \square

Using Corollary 3.2, we get the following.

Corollary 4.4 *SAT does not have succinct PCPs unless $\text{NP} \subseteq \text{coNP}/\text{poly}$ and PH collapses.*

In the other direction, self-compressibility implies the existence of succinct PCPs.

Theorem 4.5 *If SAT is self-compressible, then SAT has succinct PCPs.*

Proof. Assume SAT is self-compressible via a compression function f . Given a formula ϕ of size m on n variables, $f(\phi)$ is a formula of size $\text{poly}(n)$ such that $f(\phi)$ is satisfiable if and only if ϕ is satisfiable. The PCP theorem states that there are PCPs for SAT with polynomial proof size, completeness 1, soundness $1/2$ and $O(1)$ queries.

We modify the verifier V in the PCP theorem to a verifier V' that does the following: It first applies the compression function f to the input formula ϕ and obtains a formula $f(\phi)$ which is satisfiable if and only if ϕ is satisfiable. Then it treats the proof as a proof that $f(\phi)$ is satisfiable and runs V with input $f(\phi)$ and oracle access to the proof. The correctness of V' follows from the self-compressibility property, and V' inherits its parameters from V . The proof size is polynomial in the size of $f(\phi)$, i.e., of size $\text{poly}(n)$. Thus the PCP with verifier V' is a succinct PCP. \square

It's straightforward to prove a more general version of Theorem 4.5. First, the analogue of Theorem 4.5 holds for any parametric problem in NP , with basically the same proof and using the NP -completeness of SAT. Second, we don't even need the problem to be self-compressible - it suffices if the problem is compressible to a problem in NP .

Theorem 4.6 *Let L be any parametric problem in NP. If L is compressible within NP, then L has succinct PCPs.*

Using Theorem 2.12, we derive the following corollary.

Corollary 4.7 *VC has succinct PCPs.*

To sum up, a refined understanding of which parametric NP problems are compressible and which are not would also lead to a better understanding of which problems have efficient proofs.

4.2 Kernelization

The notion of compression is very closely related to the notion of kernelization in parameterized complexity. Indeed, a parametric problem has a polynomial-size kernel if and only if it is self-compressible. Thus, Theorem 2.12 is just a reflection of the well-known fact that VertexCover has a polynomial-size kernel, in fact a linear-size kernel. This raises the question of whether the parameterized versions of other NP-complete problems such as SAT and Clique are polynomially kernelizable. The question of whether the parametric problems SAT has a polynomial kernelization is explicitly posed as an open problem by Flum and Grohe [FG06]. In their survey on kernelization, Guo and Niedermeier [GN07] state, “In terms of computational complexity theory, the derivation of lower bounds is of the utmost importance.”

Bodlaender et al. [BDFH08] develop a framework for studying polynomial kernelizability of natural parametric problems. The central question in their framework is Question 1.1, and they show that a negative answer to this question implies that various natural parametric problems are not polynomially kernelizable. Proposition 2.13 also gives that a negative answer to Question 1.1 implies that SAT is not polynomially kernelizable. Thus, from Theorem 3.1, we immediately get:

Corollary 4.8 *SAT is not polynomially kernelizable unless PH collapses.*

Proposition 2.13 also implies that various other parametric problems are unlikely to be polynomially kernelizable.

Corollary 4.9 *The following parametric problems are not polynomially kernelizable unless PH collapses: Clique, DominatingSet, IntegerProgramming.*

Note that as per our definitions, all the problems considered above are in fact fixed-parameter tractable, since they can be solved in time $O(2^n)\text{poly}(m)$. However, except for Corollary 4.8, this is true for a parameterization of the problem that is different from the standard parameterization. Corollary 4.9 is of philosophical interest, showing a novel link between a classical complexity assumption and a parameterized complexity assumption. Bodlaender et al. [BDFH08] show separations between fixed-parameter tractability and polynomial kernelizability for various parametric problems more widely considered in practice.

4.3 Cryptographic Reductions

The main motivation of Harnik and Naor [HN06b] for studying instance compressibility was the wealth of cryptographic applications. They showed that compressibility of OR-SAT would imply

constructions of collision-resistant hash functions (CRHs) from one-way functions (OWFs), which would solve a long-standing open problem in cryptography. They also showed that the compressibility assumption would have interesting implications for the active research area of “everlasting security” in the bounded storage model - any hybrid scheme could be broken. Harnik and Naor also discuss a stronger compressibility assumption, *witness-retrievable compressibility*, which implies construction of oblivious transfer protocols (OT) based on OWFs, but they also give evidence against their assumption, so we do not discuss it here.

Independently, Dubrov and Ishai used strong *incompressibility* assumptions to construct “non-Boolean” pseudo-random generators (PRGs), namely pseudo-random generators that fool non-Boolean tests. They show how to use such assumptions to reduce the randomness complexity of sampling algorithms, so that the randomness used is only marginally larger than the output length of the sampler.

We now discuss what implications our results have for the feasibility of the above constructions. Our discussion is mostly informal, and we do not define the underlying cryptographic notions formally here. Consult the respective papers [HN06b, DI06] for definitions.

Construction of CRHs from OWFs: Constructing CRHs from OWFs is a classic problem in theoretical cryptography, and it is known this cannot be done using black-box reductions [Sim98]. Harnik and Naor suggest a non-black-box way to approach this problem, using the compressibility assumption for SAT. They show that probabilistic compression of OR-SAT with error $< 2^{-m}$ implies that CRHs can be constructed from OWFs. But as stated in Theorem 3.3, this compression assumption does not hold unless PH collapses, and hence this approach to constructing CRHs is unlikely to yield dividends. It is possible, though, that CRHs can be constructed from even weaker assumptions on compressibility that might be more plausible - it’s an interesting open problem to come up with such assumptions.

Everlasting Security of the Hybrid Bounded Storage model: The bounded storage model, defined by Maurer [Mau92], constrains the space used by adversaries rather than their running time. In this model, all honest and dishonest parties are given access to a shared random string. It has been shown that two parties that have previously shared a secret key can securely communicate using a small amount of local storage, even if the adversary has a much higher storage capability [AR99, DR02, Vad04]. These schemes have the property of *everlasting security*, meaning that the protocol remains secure if the private key is revealed to the adversary after the completion of the protocol. In the setting where the two honest parties do not exchange a secret key in advance, only quite weak results are possible (in terms of the local storage required to communicate securely). In an attempt to derive stronger results in a setting that is still quite realistic, a hybrid version of the Bounded Storage Model has been defined [HN06a]. Harnik and Naor [HN06b] show that if SAT is compressible, then no hybrid scheme can have the property of everlasting security. By giving evidence that SAT is not compressible, Theorem 3.1 holds out hope for achieving everlasting security using some protocol in the Hybrid Bounded Storage model.

Construction of non-Boolean PRGs: This is an application of incompressibility rather than compressibility, and since Theorem 3.1 provides a natural complexity-theoretic condition that implies incompressibility of NP, namely that $\text{NP} \not\subseteq \text{coNP}/\text{poly}$, one might imagine that our techniques would be more directly relevant here. Dubrov and Ishai construct different kinds (cryptographic and complexity-theoretic) of non-Boolean PRGs based on the assumptions, respectively, that there is a one-way permutation with an incompressible (on average) hard-core bit, and that for each

fixed k , there is a language in P not compressible (on average) to n^k bits³. These assumptions are both implied by the assumption that there are exponentially strong one-way permutations, but it would be interesting to derive the corresponding conclusions from a natural complexity-theoretic assumption, such as the assumption that NP does not have sub-exponential size co-nondeterministic circuits. Our results do not apply directly here for two reasons: (1) The incompressibility assumptions refer to incompressibility *on the average*, rather than in the worst-case. We consider such average-case assumptions in the next section, since they are related to probabilistic compression, but we do not know how to apply our techniques to say something of interest about them. (2) The incompressibility assumptions refer to incompressibility of problems computable in *polynomial time* rather than NP problems. Nevertheless, Theorem 3.1 can be considered as progress towards the goal of deriving non-Boolean PRGs from standard complexity-theoretic assumptions.

5 Probabilistic Compression and Parametric Hardness of Approximation

In this section, we study probabilistic compression. We are unable to show that probabilistic compression of NP -complete problems is unlikely under a standard complexity-theoretic assumption, but we do have results in settings where the number of random bits used is restricted. We also have negative results for the *implicit* case where the compression algorithm operates in time $\text{poly}(n)$ when given random access to its input, but these negative results are under a somewhat non-traditional strong derandomization assumption. In the next section, we discuss the derandomization assumption we use in more detail. The implicit case is relevant to the question of parametric hardness of approximation, where we ask if the classical machinery for hardness-of-approximation results can be extended to the parameterized setting.

5.1 Probabilistic Compression

We consider probabilistic compression of parametric problems. Theorem 3.3 applies to the case where the error is less than 2^{-m} , so here we will be concerned with the case where the error is larger. The techniques of Theorem 3.1 and 3.3 do not seem to apply here. But we do obtain various reductions between different versions of the problem, as well as results for restricted settings.

First, we note that the correctness probability can be amplified to $1 - 2^{-\text{poly}(n)}$ using standard techniques.

Proposition 5.1 *If L is probabilistically compressible with error $1/3$, then L is probabilistically compressible with error $2^{-\text{poly}(n)}$.*

Proof. Let A be a language such that L is probabilistically compressible within A . Let f be the compression function compressing inputs with length m and parameter n with high probability to length at most $t(n)$ which is polynomial in n . We fix a polynomial $p(\cdot)$ and define a new language A' and compression function f' such that L is probabilistically compressible within A' via f' . An input $\langle x_1, x_2, \dots, x_k \rangle$ is in A' iff the majority of x_i 's are in A . All inputs not of this form are excluded from A' . The probabilistic compression function f' , given input x of length m , simulates

³Here we are referring to compressibility of languages, not parametric problems, and the strength of the compression is expressed in terms of input length

f independently $p(n)$ times on x to obtain $x_1 \dots x_{p(n)}$. It discards all strings of length $> t(n)$, re-indexes the strings to obtain $x_1, x_2 \dots x_k$ and then outputs $\langle x_1, x_2 \dots x_k \rangle$. Using Chernoff bounds, one gets that if f has error at most $1/3$, then f' has error at most $2^{-\Omega(p(n))}$. \square

Note that the new error is not sufficiently small to use Adleman's trick as in the proof of Theorem 3.3, since the instance size is m .

Next, we show some results where the randomness complexity of the compression function is restricted.

Lemma 5.2 *If L is probabilistically compressible with randomness complexity $O(\log(n))$, then L is deterministically compressible.*

Proof. The proof is similar to the proof of Proposition 5.1. Let L be probabilistically compressible within A via a compression function f with randomness complexity $O(\log(n))$. Let c be a constant such that for any x , f compresses $\langle x, 1^n \rangle$ to a string of length at most n^c , with high probability. We define a new compression function f' computable in deterministic polynomial time, and a set A' , such that L is compressible within A' via f' . Given input x , f' simulates f on x with every possible random string of length $O(\log(n))$ to obtain $x_1, x_2 \dots x_t$, where $t = \text{poly}(n)$. It discards all strings with length $> n^c$ and then re-indexes the strings to obtain $x_1 \dots x_k$. It outputs $\langle x_1, x_2 \dots x_k \rangle$. The set A' is just the set of inputs of the form $\langle x_1, x_2 \dots x_k \rangle$ such that the majority of x_i 's are in A . \square

As a corollary, we can extend the negative result in Theorem 3.1 to the case of probabilistic compression with small randomness complexity.

Corollary 5.3 *If OR-SAT is probabilistically compressible with randomness complexity $O(\log(n))$, then PH collapses.*

We next show that if we could extend the negative result slightly to probabilistic compression with randomness complexity $O(\log(m))$ (using advice), then it would also rule out probabilistic compression in the general case.

Theorem 5.4 *If L is probabilistically compressible, then L is probabilistically compressible with randomness complexity $O(\log(m))$ and $\text{poly}(m)$ advice.*

Proof Sketch. The idea is to try to derandomize the probabilistic reduction f for L by using a discrepancy set of size $\text{poly}(m)$. The probabilistic reduction can be viewed as a deterministic function taking x and the random string as input and yielding a string of size $\text{poly}(n)$ which is in L iff $x \in L$, with high probability over the choice of random string. We can get by using only polynomially many pseudo-random strings rather than all possible random strings if the "test" that the output string is in L iff $x \in L$ is satisfied with approximately the same probability (say, with absolute difference at most $1/12$ between the probabilities) over pseudo-random strings as over random strings. The test can be encoded by a polynomial-size Boolean circuit with oracle access to L , hence a set of $\text{poly}(m)$ strings chosen at random will "fool" all such circuits with high probability. Thus there must exist such a set S of strings - we specify S explicitly in the advice string using $\text{poly}(m)$ bits. The new compression function f' for L with randomness complexity $O(\log(m))$ just uses its random string as an index into the discrepancy set encoded by the advice string, and simulates f using the corresponding element of the discrepancy set as its "random

choice” rather than using a purely random string. The defining property of the discrepancy set ensured that this is still a valid probabilistic compression algorithm for L . \square

Probabilistic compression also reduces to non-uniform average case compression, by a simple averaging argument.

Proposition 5.5 *if L is probabilistically compressible, then L is non-uniformly $(1 - 2^{-\text{poly}(n)})$ -compressible on average.*

Proof. By Proposition 5.1, we can assume that the compression algorithm has error at most $2^{-\text{poly}(n)}$. Now consider the random string used by the compression algorithm. By averaging, there is some choice r of the random string such the compression algorithm is correct on at least a $1 - 2^{-\text{poly}(n)}$ fraction of inputs when run with r as the random string. We encode r into the advice, yielding a non-uniform compression algorithm for L which works on a $1 - 2^{-\text{poly}(n)}$ fraction of inputs. \square

5.2 Parametric Hardness of Approximation and Implicit Compression

Here we study the question of whether parametric problems are as hard to approximate as they are to decide. The notion of *implicit* probabilistic compression, where the compression function is required to be computable in time polynomial in the size of the parameter, turns out to be useful in this regard.

A problem is hard to approximate if its “gap” version is hard to decide. In the classical unparameterized world, reducing SAT to its gap version is equivalent to the existence of PCPs. However, in the unparameterized setting, reduction of the parameterized problem SAT to its gap version is not equivalent to the existence of succinct PCPs - we know that succinct PCPs imply a reduction to the gap version, but not the converse. For the one-sided gap version, in a follow-up work, Chen, Flum and Müller [CFM09] rule out such a reduction under the assumption that PH does not collapse by applying the technique of Theorem 3.1 to probabilistic compression with *one-sided* error. Here we are concerned with the two-sided version.

Gap versions of parametric NP problems are modelled naturally as parametric promise problems - parametric problems where the “yes” and “no” instances are mutually exclusive but not exhaustive.

Definition 5.6 *A parametric promise problem is a pair (Y, N) where $Y, N \subseteq \{ \langle x, 1^n \rangle \mid x \in \{0, 1\}^* \}$, $n \in \mathbb{N}$, and $Y \cap N = \emptyset$.*

The various definitions of compression extend naturally to parametric promise problems - we require the compression to work correctly only for instances in $Y \cup N$ and allow it to behave arbitrarily on other instances. Similarly the definition of “W-reduction” also extends naturally to reductions between parametric promise problems and reductions from a parametric promise problem to a parametric problem or vice versa.

We define the natural parametric promise version of SAT.

Definition 5.7 *c, s -GapSAT is the parametric promise problem (Y, N) where $\langle x, 1^n \rangle \in Y$ if n is the number of variables in the formula x and at least c fraction of clauses in x are satisfiable, and $\langle x, 1^n \rangle \in N$ if n is the number of variables in x and at most s fraction of clauses in x are satisfiable.*

The natural question here is whether SAT W-reduces to c, s -GapSAT, where $c - s = \Omega(1)$. If so, then the parametric problem SAT is as hard to approximate within a factor s/c as it is to solve, in analogy with hardness of approximation results in the unparameterized case that follow from the PCP theorem.

An interesting property of the parameterized version of GapSAT is that it is *implicitly* probabilistically compressible. We now define this notion.

Definition 5.8 *A parametric promise problem S is implicitly probabilistically compressible if it is probabilistically compressible via a compression function f that, given an input $\langle x, 1^n \rangle$, can be computed in time $\text{poly}(n)$ with oracle access to the input.*

Dieter van Melkebeek and Matt Anderson [vM09] observed that decision problems such as OR-SAT and SAT are trivially not implicitly compressible, because an implicit compression is not sensitive to small changes in its input, such as changing one of the formulas in a NO input to OR-SAT from unsatisfiable to satisfiable. However, the notion is non-trivial for promise problems. Based on a sampling technique of Harnik and Naor [HN06b] (presented informally in Section 2.9 of their paper), we can derive an implicit probabilistic compression algorithm for c, s -GapSAT, even when $c - s = \Omega(1/\text{poly}(n))$.

Lemma 5.9 *Let $c(n)$ and $s(n)$ be functions computable in time $\text{poly}(n)$, such that $c - s = \Omega(1/\text{poly}(n))$. Then c, s -GapSAT is implicitly probabilistically compressible within SAT.*

Proof Sketch. Given a formula $\langle x, 1^n \rangle$, our implicit compression algorithm samples $\text{poly}(n)$ clauses from x and produces a formula corresponding to the question of whether at least $(c + s)/2$ fraction of these clauses are simultaneously satisfiable. Using Chernoff bounds, we can argue that if we sample enough (but still polynomially many) clauses, then if at least a c fraction clauses were satisfiable in x , close to c fraction of clauses will be satisfiable in the compressed formula, with probability very close to 1. The argument is a little more delicate for the soundness case. Here, if we fix an assignment, then the random sampling ensures (again using Chernoff bounds) that with probability $1 - 2^{-\text{poly}(n)}$, not much more than c fraction of clauses in the compressed formula will be satisfied by that assignment. Now, if we take a union bound over all 2^n assignments, the probability that not much more than c fraction of clauses are simultaneously satisfiable remains close to 1.

Since we randomly sample $\text{poly}(n)$ clauses from x and then spend $\text{poly}(n)$ time processing these samples, the algorithm can be implemented in probabilistic time $\text{poly}(n)$ with random access to x . \square

We show that under a certain strong derandomization hypothesis, implicit probabilistic compression can be *derandomized*. As a corollary, assuming in addition that PH does not collapse, there cannot be a W-reduction from parameterized SAT to its gap version.

We do not have a strong belief about the *truth* of our derandomization assumption, but we do believe it is hard to *refute*, thus our result provides evidence that implicit probabilistic compression may be hard to obtain using known techniques.

The assumption we use states that given a string x , we can compute from x in polynomial time the truth table of a function f on $\lfloor \log(x) \rfloor$ bits such that f requires exponential-size non-deterministic circuits even when the circuits have oracle access to x . The assumption can be interpreted as a strong diagonalization assumption - given an oracle in explicit form, the assumption

states that it's possible to compute efficiently a function that diagonalizes against small circuits accessing that oracle.

Hypothesis 5.10 (*Oracle Derandomization Hypothesis*) *Let m be polynomially bounded as a function of n , such that $m(n)$ is both computable and invertible in polynomial time. There is a family of functions $G = G_n$, $G_n : \{0,1\}^m \rightarrow \{0,1\}^n$ computable in time $\text{poly}(m)$ and a constant $\epsilon > 0$ independent of m such that for any x , $G(x)$, when interpreted as the truth table of a function on $\lceil \log(x) \rceil$ bits, requires non-deterministic circuits of size n^ϵ even when the circuits are given oracle access to x .*

We say that the Oracle Derandomization Hypothesis holds on $A \subseteq \{0,1\}^$ if the above condition holds for all $x \in A$.*

We use Hypothesis 5.10 by combining it with a pseudo-random generator of Shaltiel and Umans [SU01] to derandomize certain kinds of sampling algorithms using a smaller seed size than we would require if we carried out the derandomization naively. The obvious point of reference is Theorem 5.4, where we reduced the randomness to $O(\log(m))$ bits by embedding an appropriate discrepancy set in the advice string. We could not completely derandomize Theorem 5.4, i.e., convert it to a deterministic compression algorithm, because running over all possible random strings of length $O(\log(m))$ and defining the deterministic compressed string to correspond to the majority value of the probabilistic compressed strings would blow the size of the compressed string up to $\text{poly}(m)$, which makes the resulting deterministic compression trivial. Thus it is essential for us that the randomness is reduced to size $O(\log(n))$. If the probabilistic compression is implicit, we can use Hypothesis 5.10 to achieve this. We first describe what properties we require of the Shaltiel-Umans generator.

Theorem 5.11 (Shaltiel-Umans) *Fix any constant $d > 0$. There is a constant $e(d) > 0$ depending on d , and a family of functions $F = F_n$, where $F_n : \{0,1\}^n \times \{0,1\}^{O(\log(n))} \rightarrow \{0,1\}^{n^e}$ is computable in polynomial time, such that for any oracle A and oracle non-deterministic circuit C of size n^e with oracle access to A , if $y \in \{0,1\}^n$ has non-deterministic circuit complexity at least n^d with respect to circuits that have oracle access to A , then $|\Pr_{z \in \{0,1\}^{n^e}}(C(z) = 1) - \Pr_{z \in \{0,1\}^{O(\log(n))}}(C(F_n(y, z) = 1))| < 1/n^e$.*

Next we describe what kind of derandomization results from using Theorem 5.11 along with Hypothesis 5.10.

Lemma 5.12 *Let m and n be parameters such that m is polynomially bounded in n . Assume the Oracle Derandomization Hypothesis holds. Let A be a string of length m represented as an oracle function on $\lceil \log(m) \rceil$ bits, and let C be a non-deterministic circuit of size at most n^ϵ with oracle access to A , where $e(\epsilon) > 0$ is the constant in the statement of Theorem 5.11 corresponding to the constant ϵ in the statement of Hypothesis 5.10. Then there is a function $h : \{0,1\}^{O(\log(n))} \rightarrow \{0,1\}^n$ computable in $\text{poly}(n)$ time such that $|\Pr_{z \in \{0,1\}^{n^e}}(C(z) = 1) - \Pr_{z \in \{0,1\}^{O(\log(n))}}(C(h(z)) = 1)| < 1/n^e$.*

Proof. We define $h(\cdot) = F_n(G(A), \cdot)$. By Hypothesis 5.10, $G(A) \in \{0,1\}^n$ has circuit complexity at least n^ϵ with respect to circuits that have oracle access to A . Hence by theorem 5.11, $h(\cdot) = F_n(G(A), \cdot)$ is a pseudo-random generator that “fools” any circuit of size at most n^ϵ with oracle

access to A , in the sense that the acceptance probability of the circuit changes by at most an additive term of $1/n^e$ when it is run with outputs of h rather than with purely random circuits. \square

We now apply Lemma 5.12 to the case of implicit probabilistic compression, motivated by the question of whether there is a W-reduction from SAT to GapSAT.

Theorem 5.13 *If there is a W-reduction from the parametric problem SAT to c, s -GapSAT, where $c - s = \Omega(1/\text{poly}(n))$ and c and s are computable in time $\text{poly}(n)$, then either PH collapses or the Oracle Derandomization Hypothesis fails.*

Proof. We use Lemma 5.12 to derandomize the implicit probabilistic compression of Lemma 5.9 under the Oracle Derandomization Hypothesis, and then Theorem 3.1 to obtain a collapse of PH if the assumed W-reduction exists.

Assume that Hypothesis 5.10 holds. For the purpose of this lemma, we use “ k ” to denote the parameter size and reserve “ n ” for the application of Lemma 5.12.

Let the promise problem S be c, s -GapSAT, where c and s are as in the statement of the theorem. By Lemma 5.9, $S = (Y, N)$ is implicitly probabilistically compressible to SAT. Let f be the implicit probabilistic compression function for S . Let d be a constant such that on input $\langle x, 1^k \rangle$, f is computable in probabilistic time k^d with oracle access to the input. It follows that the length of the compressed string is at most k^d , since the probabilistic algorithm needs at least one unit of time per output bit it writes. We first show that the test whether the compressed string corresponding to a particular probabilistic run of f is “correct” can be performed by a non-deterministic circuit C of size $\text{poly}(k)$.

For a fixed input $\langle x, 1^k \rangle$ to the compression algorithm, this test takes as input the random string r used by the implicit probabilistic compression algorithm. The circuit C for the test first applies f to $\langle x, 1^k \rangle$ using r as the random choices for running the algorithm for f . It thus obtains a compressed formula y . It then checks if $y \in \text{SAT}$. Note that $\langle x, 1^k \rangle$ is *fixed* with respect to C . We cannot hard-code x into C since x is of length m which might be too large. Instead, we make use of the fact that the computation of f is implicit, and hence C can be represented as a non-deterministic circuit which has oracle access to x . The size of the circuit corresponds to the running time of the implicit algorithm for f together with the time required to check if the compressed formula is satisfiable, which is altogether $\text{poly}(k)$.

There are two cases - either x is in Y , or x is in N (we do not care what the implicit compression does on $x \notin Y \cup N$). In the first case, C tests if its random input leads to a satisfiable formula being output and in the second case, whether it leads to an unsatisfiable formula. Note that these two cases are symmetric with respect to the approximation of the acceptance probability of C .

Now we set $n = |C|^{1/e}$ where e is the constant in the statement of Lemma 5.12. We set m to be any large enough polynomial in n . With these parameters, the conditions of Lemma 5.12 hold and hence there is a function $h : \{0, 1\}^{O(\log(n))} \rightarrow \{0, 1\}^n$ computable in time $\text{poly}(n)$ such that the output of h fools C . We define a new deterministic compression algorithm f' which runs the algorithm for f with all possible outputs of h used as random string to obtain compressed inputs $x_1 \dots x_{\text{poly}(n)}$, each x_i of length k . Now, since Majority is a monotone function, the question whether the majority of x_i 's are satisfiable can be expressed as a SAT formula of size at most $\text{poly}(n)$.

Thus we get a deterministic compression algorithm for S . If there is a W-reduction from SAT to S , we get a deterministic compression algorithm for SAT which works for any instance length m

that is polynomially bounded in the parameter. But, using the proof of Theorem 3.1, this implies that PH collapses. \square

Theorem 5.13 suggests that some of the obstacles to finding good approximation algorithms in the classical setting are absent in the parameterized setting, and hence that we might hope for better approximations. Approximation algorithms in the context of parameterized complexity is an active field, with many open questions [Már06]. Hopefully Theorem 5.13 and the techniques used here will stimulate further work in this field.

6 The Oracle Derandomization Hypothesis

In this section we discuss the Oracle Derandomization Hypothesis used in the previous section. We comment on how this hypothesis relates to traditional derandomization hypotheses, and show that *disproving* it is as hard as separating P and NP.

In our opinion, quite apart from its relevance to compressibility, the Oracle Derandomization Hypothesis is interesting in its own right because it tests our intuitions of which kinds of derandomization are plausible and which are not. The hypothesis that E requires linear exponential size circuits, which was used by Impagliazzo and Wigderson [IW97] to completely derandomize BPP, now seems widely accepted by complexity theorists, so too the assumption that E requires linear exponential size *non-deterministic* circuits, used by Klivans and van Melkebeek [KvM02] to derandomize AM. Klivans and van Melkebeek [KvM02] use even stronger assumptions for other results on derandomization in a relativized context, such as the result that $\text{PH} \subseteq \text{P}^{\oplus \text{P}}$ if E does not have sub-exponential size circuits with oracle access to $\oplus \text{P}$. First we make some observations about how the Hypothesis generalizes traditional derandomization hypotheses.

Proposition 6.1 *There is an $\epsilon > 0$ such that $\text{E} \not\subseteq \text{i.o.NSIZE}(2^{\epsilon n})$ iff the Oracle Derandomization Hypothesis holds on 0^* .*

Proof. We prove the “if” direction first. If the Hypothesis holds on 0^* , then we define a function f in E which doesn’t have non-deterministic circuits of size $2^{\epsilon n}$ for some $\epsilon > 0$. On an input x of length n , we do the following to compute $f(x)$: we simulate G_{2^n} on $0^{m(2^n)}$ to obtain a string X_n of length 2^n . X_n will be interpreted as the truth table of f on inputs of length n . $f(x)$ is computed by just reading off the appropriate value in the bit string X_n .

To prove the hardness of f , let ϵ be the constant in the statement of Hypothesis 5.10. Assume contrariwise that f has non-deterministic circuits of size 2^ϵ infinitely often. Then the same circuits also contradict the Oracle Derandomization Hypothesis on 0^* .

For the reverse direction, let f be a function without non-deterministic circuits of size $2^{\epsilon n}$. We define G as follows: on input $0^{m(n)}$, it outputs the truth table of f on $\lfloor \log(n) \rfloor$ bits, followed by a string of 0’s. Now suppose the Oracle Derandomization Hypothesis were false on 0^* . Then, for infinitely many n , $G(0^{m(n)})$ has non-deterministic circuits of size $2^{\epsilon n}$ with oracle access to $0^{m(n)}$. Now, if we replace each oracle gate with the input “0”, these are non-deterministic circuits of size $2^{\epsilon n}$ deciding f , contradicting the assumption on f . \square

Basically the same proof gives that the condition on the set A on which the Hypothesis holds can be made a little weaker.

Proposition 6.2 *let $\delta > 0$ be any constant, and $m(n)$ be a polynomially bounded function as in the statement of Hypothesis 5.10. Let A_m be the set of strings in $\{0, 1\}^{m(n)}$ which, when interpreted as the truth table of a function on $\lfloor \log(m) \rfloor$ bits, have circuits of size $2^{n^{1-\delta}}$. Let A be the union of $A_{m(n)}$ over all n . Then there exists an $\epsilon > 0$ such that $E \not\subseteq \text{NSIZE}(2^{\epsilon n})$ iff the Oracle Derandomization Hypothesis holds on A .*

Proposition 6.2 states that the traditional derandomization hypothesis used to derandomize AM is equivalent to the Oracle Derandomization Hypothesis holding on all “succinct” strings, where “succinct” means that the string can be described by a circuit of size sub-polynomial in the length of the string.

Next, we ask the question, is there any heuristic evidence the Oracle Derandomization Hypothesis is true? Often, if there is a probabilistic process which generates an object of a certain type with high probability, then it is reasonable to conjecture that there is a deterministic process of similar complexity producing an object of that type. For instance, a function with specified input and output lengths which is chosen uniformly at random is usually a pseudo-random generator with strong properties. However, in our case, a function G chosen uniformly at random does not satisfy the required condition with high probability, but only with non-zero probability. This is not necessarily evidence *against* Hypothesis 5.10. First, there might be some non-trivial probabilistic process sampling G 's that satisfy the condition in Hypothesis 5.10 with high probability and it might be reasonable to conjecture that this non-trivial process can be derandomized. Second, there are natural examples of probabilistic constructions (proved to be correct using the Lovasz Local Lemma) which are only known to work with non-zero probability, and can still be derandomized.

We turn the question on its head and ask if it might be possible to *unconditionally* disprove the Oracle Derandomization Hypothesis. This seems hard, since it would imply $P \neq \text{NP}$.

Theorem 6.3 *If the Oracle Derandomization Hypothesis is false, then $P \neq \text{NP}$.*

Proof. Assuming $P = \text{NP}$, we construct a G satisfying the required condition. Fix some ϵ such that $0 < \epsilon < 1$. We observe that for any x of length m , there *does* exist a string of length n , which when interpreted as a function on $\lfloor \log(n) \rfloor$ bits, doesn't have non-deterministic circuits of size n^ϵ with oracle access to x , just by a counting argument. Indeed, we can find the lexicographically first such string $y(x)$ in $P^{\Sigma_3^P}$. By assumption, $P = \text{NP}$ and hence $P^{\Sigma_3^P} = P$, thus we can carry out the search in polynomial time and output y . \square

In Section 5.2, we showed that the Oracle Derandomization Hypothesis can be used to derandomize *implicit* probabilistic compression. In follow-up work, Anderson and van Melkebeek show [vM09] that the Hypothesis is in fact more powerful - it can be used to derandomize probabilistic compression, even when the compression algorithm is not implicit.

7 Questions

We highlight the two main technical questions that arise from this work.

The first is whether some negative results can be shown under a standard assumption for the probabilistic or closely-related average-case version of compression. Such results would have relevance to parametric hardness of approximation, and provide further evidence for the infeasibility of certain approaches to cryptographic constructions.

The second is the general question of characterizing for which functions f , the compressibility of f -SAT implies collapse of PH. Here f -SAT is the natural generalization of the OR-SAT problem to Boolean functions other than OR. This question basically reduces to the question of whether AND-SAT is compressible, since the compression of f -SAT for non-monotone f directly implies $\text{NP} \subseteq \text{coNP/poly}$, and every monotone f that depends on all its m variables embeds either a \sqrt{m} sized OR or AND. Thus if we can show that (assuming NP not in co-NP/poly) AND-SAT is not compressible, then under that same assumption f -SAT is not compressible for any f that has no useless variables.

Acknowledgements

We thank Hans Bodlaender, Rod Downey, Mike Fellows and Danny Hermelin for bringing Question 1.1 to our attention and sending us an early version of their manuscript [BDFH08]. The second author would like to thank Valentine Kabanets for his support and for interesting discussions. We would also like to thank Harry Buhrman for interesting discussions and for sending us an early version of his paper with John Hitchcock [BH08].

References

- [Adl78] Leonard Adleman. Two theorems on random polynomial time. In *Proceedings of the 20th Annual IEEE Symposium on the Foundations of Computer Science*, pages 75–83, 1978.
- [ALM⁺98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.
- [AR99] Yonatan Aumann and Michael Rabin. Information theoretically secure communication in the limited storage space model. In *Proceedings of CRYPTO '99*, pages 65–79, 1999.
- [AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.
- [AS08] Noga Alon and Joel Spencer. *The Probabilistic Method*. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley, Third edition, 2008.
- [BDFH08] Hans Bodlaender, Rod Downey, Michael Fellows, and Danny Hermelin. On problems without polynomial kernels. In *Proceedings of 35th International Colloquium on Automata, Languages and Programming*, pages 563–574, 2008.
- [BH08] Harry Buhrman and John Hitchcock. NP-complete sets are exponentially dense unless $\text{NP} \subseteq \text{co-NP/poly}$. In *Proceedings of 23rd Annual IEEE Conference on Computational Complexity*, 2008. To appear.
- [CCDF97] Liming Cai, Jianer Chen, Rodney Downey, and Michael Fellows. Advice classes of parameterized tractability. *Annals of Pure and Applied Logic*, 84(1):119–138, 1997.

- [CFM09] Yijia Chen, Jörg Flum, and Moritz Müller. Lower bounds for kernelizations and other preprocessing procedures. In *Proceedings of Computability in Europe 2009*, Lecture Notes in Computer Science. Springer-Verlag, 2009. To appear.
- [Coo71] Stephen Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.
- [DF99] Rodney Downey and Michael Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.
- [DI06] Bella Dubrov and Yuval Ishai. On the randomness complexity of efficient sampling. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 711–720, 2006.
- [Din07] Irit Dinur. The PCP theorem by gap amplification. *Journal of the ACM*, 54(3), 2007.
- [DR02] Yan Zong Ding and Michael Rabin. Hyper-encryption and everlasting security. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 1–26, 2002.
- [FG06] Jorg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006.
- [FS08] L. Fortnow and R. Santhanam. Infeasibility of instance compression and succinct PCPs for NP. In *Proceedings of the 40th ACM Symposium on the Theory of Computing*, pages 133–142. ACM, New York, 2008.
- [GJ76] Michael Garey and David Johnson. *Approximation Algorithms for Combinatorial Problems: An Annotated Bibliography*, pages 41–52. Academic Press, New York, 1976.
- [GN07] Jiong Guo and Rolf Niedermeier. Invitation to data reduction and problem kernelization. *ACM SIGACT News*, 38(1):31–45, 2007.
- [HN06a] Danny Harnik and Moni Naor. On everlasting security in the hybrid bounded storage model. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming*, pages 192–203, 2006.
- [HN06b] Danny Harnik and Moni Naor. On the compressibility of NP instances and cryptographic applications. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 719–728, 2006.
- [IW97] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing*, pages 220–229, 1997.
- [KL82] Richard Karp and Richard Lipton. Turing machines that take advice. *L'Enseignement Mathématique*, 28(2):191–209, 1982.
- [KR07] Yael Tauman Kalai and Ran Raz. Interactive PCP. *Electronic Colloquium on Computational Complexity*, 7(31), 2007.
- [KvM02] Adam Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial hierarchy collapses. *SIAM Journal of Computing*, 31(5):1501–1526, 2002.

- [Mah82] Stephen Mahaney. Sparse complete sets for NP: Solution of a conjecture of berman and hartmanis. *Journal of Computer and System Sciences*, 25(2):130–143, 1982.
- [Már06] Daniel Márx. Parameterized complexity and approximation algorithms, 2006. To appear in *The Computer Journal*.
- [Mau92] Ueli Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 5(1):53–66, 1992.
- [Nie06] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.
- [Sim98] Daniel Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? *Advances in Cryptology, EUROCRYPT 1998, Lecture Notes in Computer Science*, 1403:334–345, 1998.
- [SU01] Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudo-random generator. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, 2001.
- [Vad04] Salil Vadhan. Constructing locally computable extractors and cryptosystems in the bounded storage model. *Journal of Cryptology*, 17(1):43–77, 2004.
- [vM09] Dieter van Melkebeek. Personal communication. 2009.
- [Yap83] Chee-Keng Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical Computer Science*, 26:287–300, 1983.