Short PCPs with Projection Queries

Eli Ben-Sasson* and Emanuele Viola**

Technion — Israel Institute of Technology and Northeastern University
eli@cs.technion.ac.il, viola@ccs.neu.edu

Abstract. We construct a PCP for NTIME(2^n) with constant soundness, 2^npoly(n) proof length, and poly(n) queries where the verifier's computation is simple: the queries are a projection of the input randomness, and the computation on the prover's answers is a 3CNF. The previous upper bound for these two computations was polynomial-size circuits. Composing this verifier with a proof oracle increases the circuit-depth of the latter by 2. Our PCP is a simple variant of the PCP by Ben-Sasson, Goldreich, Harsha, Sudan, and Vadhan (CCC 2005). We also give a more modular exposition of the latter, separating the combinatorial from the algebraic arguments.

If our PCP is taken as a black box, we obtain a more direct proof of the result by Williams, later with Santhanam (CCC 2013) that derandomizing circuits on n bits from a class C in time $2^n/n^{\omega(1)}$ yields that NEXP is not in a related circuit class C'. Our proof yields a tighter connection: C is an And-Or of circuits from C'. Along the way we show that the same lower bound follows if the satisfiability of the And of any 3 circuits from C' can be solved in time $2^n/n^{\omega(1)}$.

1 Introduction

It has long been known that solving satisfiability of circuits, or derandomizing probabilistic circuits implies new circuit lower bounds (for various exponential-time classes), see e.g. [KL80,IKW02]. In [Wil10] Williams gives an interesting instance of this phenomenon, where a non-trivial lower bound against a circuit class C follows from a satisfiability or derandomization algorithm for circuits of a related class C' that runs in time $2^n/n^{\omega(1)}$, where n is the number of variables of circuits in C'. It is an interesting question whether the approach based on satisfiability or the one based on derandomization should be pursed to obtain new circuit lower bounds.

The satisfiability approach – not the derandomization approach – has given non-trivial lower bounds [Wil11]. Moreover this approach has been tightened, by making C' closer to C, in [SW13,JMV13,Oli13], making it plausible that more lower bounds will be obtained. In fact, we will tighten it a bit more in

* The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 240258.

** Supported by NSF grants CCF-0845003, CCF-1319206

J. Esparza et al. (Eds.): ICALP 2014, Part I, LNCS 8572, pp. 163–173, 2014.

this work. However, it is not clear how much this approach can be pushed. Do we believe that the satisfiability of (unrestricted) polynomial-size circuits can be solved faster than brute-force search? Even for seemingly simple problems such as MAX3SAT, no satisfiability algorithm better than brute-force search is known, despite attempts since a decade ago [Wil05]. Note that the MAX3SAT problem – given a 3CNF and an integer ℓ, is there an assignment that satisfies $\geq \ell$ clauses? – corresponds to the restricted class of depth-2 circuits known as MAJ $\circ$ AND$_3$: a Majority on And's on three variables. The lack of progress on MAX3SAT is an obstacle for obtaining new lower bounds from satisfiability.

A priori, the approach based on derandomization should apply more broadly, because most researchers indeed believe that derandomization is possible (and a long line of research has shown that indeed derandomization is possible based on lower bounds). Also, for several classes we have nontrivial derandomization algorithms but not satisfiability ones. For example, for the class mentioned above of MAJ $\circ$ AND$_3$ circuits a derandomization is given in [LVW93,Vio07]. Even when both types of algorithms are available, the speed of the derandomization one often outperforms that of the satisfiability one. For example, the running time for the derandomization of CNF, see [GMR13] for the latest, vastly outperforms that of satisfiability algorithms, cf. [Her11]. For another example, consider the class of poly-size, constant-depth circuits with Or, Not, and Parity gates (AC^0 with parity gates). To our knowledge, the best satisfiability algorithm is the one in [Wil11] which has running time $2^{n-n^{\epsilon}}$. By contrast, [FSUV13] derandomize these circuits in time $2^{n-n/\mathrm{poly}\log n}$ (building on available lower bounds).

One advantage of satisfiability over derandomization is that the corresponding connection to lower bounds is simpler and incurs less overhead. To obtain lower bounds from derandomization one relies on probabilistically checkable proofs (PCP), specifically the somewhat intricate work by Ben-Sasson, Goldreich, Harsha, Sudan, and Vadhan [BGH$^+$05]. The intricacy of this work reflects on two aspects of the approach. First, to make it apply to restricted circuit classes such as ACC^0 or TC^0, previous to this work one needed a roundabout argument, provided by Santhanam and Williams [SW13], which actually relies on a subsequent PCP by Mie [Mie09] combining [BGH$^+$05] with Dinur's gap amplification [Din07]. Second, the indirect aspect of the argument is reflected in the overhead in the reduction. For example, to obtain a lower bound against circuits of depth d, one needed a derandomization algorithm for circuits of depth cd for a constant $c > 1$.

1.1 Our Results

In this work we provide a variant of the PCP [BGH$^+$05] where the computation of the verifier is quite modest: Given randomness, the verifier computes its queries just by taking projections of the randomness, and the computation on the prover's answers is a 3CNF. The previous upper bound for these two computations was polynomial-size circuits.

Theorem 1 (Short PCPs with Projection Queries). *Let M be an algorithm running in time $T = T(n) \geq n$ on inputs of the form (x, y) where*

$|x| = n$. Given $x \in \{0,1\}^n$ one can output in time $\mathrm{poly}(n, \log T)$ *circuits* $\mathsf{Query} : \{0,1\}^r \to [2^r]^t$ *for* $t = \mathrm{poly}(r)$ *and* $\mathsf{Dec} : \{0,1\}^t \to \{0,1\}$ *such that:*

- **Proof length.** $2^r \leq T \cdot \mathrm{poly} \log T$,
- **Completeness.** *If there exists y such that $M(x,y)$ accepts then there exists a map $\pi : [2^r] \to \{0,1\}$ such that for any $z \in \{0,1\}^r$ we have $\mathsf{Dec}(\pi(q_1), \ldots, \pi(q_t)) = 1$ where $(q_1, \ldots, q_t) = \mathsf{Query}(z)$,*
- **Soundness.** *If no y causes $M(x,y)$ to accept, then for every map $\pi : [2^r] \to \{0,1\}$, at most $1/n^{10}$ fraction of the $z \in \{0,1\}^r$ have $\mathsf{Dec}(\pi(q_1), \ldots, \pi(q_t)) = 1$ where $(q_1, \ldots, q_t) = \mathsf{Query}(z)$,*
- **Complexity.** Query *is a projection (a.k.a. 1-local), i.e., each output bit of* Query *is one input bit, the negation of an input bit, or a constant;* Dec *is a 3CNF.*

The polynomial in the soundness item in Theorem 1 can be traded with the number t of queries.

There is a substantial literature that develops PCPs with optimized parameters. One focus of this literature has been to optimize the complexity of Dec. However typically these works do not produce PCPs of length quasilinear in T, and the complexity of Query is not optimized. Both these aspects are critical for our applications.

Remark 1 (Number of queries vs. Query *complexity).* Relaxing the complexity of Query to be a $\mathrm{poly}(r)$-computation allows to reduce the number of queries made to the oracle to a constant, while obtaining constant soundness [Mie09]. It is an interesting open problem to find the lowest complexity obtainable for Query in a PCP statement with proof length quasilinear in T, polylogarithmic verifier running time, and where soundness, alphabet, and number of queries are all constant. In particular, it is not clear to us if it is possible in such a case to have Query be a projection.

Along the way we give a more accessible presentation of [BGH$^+$05]. Our presentation is modular and separates the combinatorial steps (given in Theorem 5) from the algebraic ones.

Taking Theorem 1 as a black box, we eliminate the roundabout argument mentioned before from the result in [SW13] that derandomizing TC^0 circuits on n bits in time $2^n/n^{\omega(1)}$ implies that NEXP is not in TC^0. Also, Theorem 1 is a small variant on [BGH$^+$05], whereas as we mentioned [SW13] needs Mie's PCP [Mie09]. Finally, we also obtain the following alternative argument, which only uses the PCP result in [BGH$^+$05] as a black-box.

The Alternative Argument. Given as a black-box a PCP such as [BGH$^+$05], i.e., with the parameters as in Theorem 1 but where the complexity is replaced by polynomial-size circuits, we can construct a PCP where the verifier has low-complexity but makes adaptive queries to the proof. Specifically, we will rely on the prover to obtain the indices of our queries, and later query the prover at those indices and also verify the prover's computation, again with the help of the

prover. This latter verification, as well as the computation Dec on the prover's answers, can be done by a 3CNF via a simple use of the Cook-Levin theorem – cf. Lemma 1.

Again, this alternative argument is sufficient to recover the TC^0 result in [SW13]. However, with Theorem 1 we obtain better parameters. Indeed, we seek very tight connections in the hope they will lead to progress on various challenges in computational lower bounds such as those mapped in [Vio13].

The concurrent work [Wil14] shows that the ability to count the number of satisfying assignments to circuits faster than brute-force search yields lower bounds against related circuits. This connection is used to obtain some new lower bounds. By our work the same lower bounds can be obtained from a satisfiability algorithm (using Theorem 3) or even a derandomization algorithm (using Theorem 2).

Next we state the tighter connections we obtain between derandomization and lower bounds. First we make a definition.

Definition 1. *Let C_n be a set of functions from $\{0,1\}^n$ to $\{0,1\}$. We say that C_n is* efficiently closed under projections *if functions in C_n have a* poly(n)*-size description and given (the description of) a function $f \in C_n$, indexes $i, j \leq n$, and a bit b, we can compute in time* poly(n) *the functions* not$f$*, $f(x_1, \ldots, x_{i-1}, b \text{ XOR } x_j, x_{i+1}, \ldots, x_n)$, and $f(x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_n)$, all of which are in C_n.*

Most of the standard classes have this property. For the theorem, the two occurrences of "poly(n)" above can be relaxed. We also use the notation $\wedge_{\mathrm{poly}(n)} \vee_3 C_{n+O(\log n)}$ to indicates the And of poly(n) Or of 3 functions from $C_{n+O(\log n)}$, all on the same n input bits.

Theorem 2 (Derandomization Implies Lower Bounds, Tightly). *Let C_n be efficiently closed under projections.*

If the acceptance probability of functions of the form $h = \wedge_{\mathrm{poly}(n)} \vee_3 C_{n+O(\log n)}$ can be distinguished from being $= 1$ or $\leq 1/n^{10}$ in Time$(2^n/n^{\omega(1)})$, then there is a function f in E^{NP} such that $f_n \notin C_n$.

One can place f in NEXP if we replace $C_{n+O(\log n)}$ with $C_{\mathrm{poly}(n)}$ and reason as in [IKW01,Wil13,Wil11].

The first step of our more modular exposition of [BGH+05] is a reduction to 3SAT that builds on [JMV13] (cf. [BCGT13a]) but achieves incomparable guarantees (Theorem 5). Using that, we can obtain the following connection between satisfiability algorithms and lower bounds.

Theorem 3 (Satisfiability Implies Lower Bounds, Tightly). *Let C_n be efficiently closed under projections.*

If the satisfiability of functions $h = g_1 \wedge g_2 \wedge g_3$, where $g_i \in C_{n+O(\log n)}$ is in Time$(2^n/n^{\omega(1)})$, then there is a function f in E^{NP} such that $f_n \notin C_n$.

The overhead to go from a satisfiability algorithm to a lower bound is evident from the theorem. The loss in size is a multiplicative factor $3 + o(1)$.

Previous losses were polynomial [Wil10], or multiplicative by a larger constant [JMV13]. The loss in depth is 2 for circuits with fan-in 2. For unbounded fan-in (or even fan-in 3) circuits with And gates (or threshold) the depth loss is 1. Previous losses were 2 [JMV13,Oli13].

Recall that the best lower bound for an explicit function on n bits is $3n - o(n)$ (non-input) gates [Blu84] (cf. [DK11]). This seems to be the best known even for functions in E^{NP} (note the number of circuits of size $3n$ is superlinear, so one cannot easily diagonalize against them in E^{NP}). By Theorem 3, to obtain a function in E^{NP} of circuit complexity $3n$ one would need to solve satisfiability of a circuit with $3(3n)$ non-input gates and n inputs – ignoring lower-order terms. The Cook-Levin theorem reduces this to a 3SAT instance on $9n + n = 10n$ variables. So one would need to solve 3SAT in deterministic time c^n for any $c < 2^{1/10} = 1.07\ldots$ The current record is $c = 1.33\ldots$ [MTY11], cf. [Her11].

1.2 Techniques

Ideas behind the proof of Theorem 1. We start with the PCP in [BGH+05] and follow its proof closely. There are two computations of the verifier in this PCP that we need to optimize. The first — Query — is taking the input randomness to the queries, which we call *preprocess*. The second — Dec — is the computation on the prover's answers, which we call *postprocess*. We discuss them separately.

Postprocess: It is a common experience in theoretical computer science research, to study at length an intricate proof in the reckless hope of optimizing parameters, only to be surprised by the late realization that a trivial, sweeping argument takes the complex proof as a black-box and gets a pretty good parameter optimization, too.

Lemma 1 (Making Dec a 3CNF). *Suppose Theorem 1 holds except that* Dec *is an unrestricted circuit of size* $\mathrm{poly}(r)$*. Then Theorem 1 holds as stated.*

Proof. By the Cook-Levin theorem we reduce Dec to a 3CNF with $\mathrm{poly}(r)$ variables and terms. The verifier will ask the prover for an additional $\mathrm{poly}(r)$ queries to obtain the satisfying assignments for this 3CNF corresponding to the input randomness. On input z, these queries are of the form (z, i), where i is an $O(\log r)$-bit index to a variable in the 3CNF. The proof contains the values of the variables in the 3CNF that verify the computation on the outputs of the queries that are made by the verifier on input z.

This general technique shows that we can always make the postprocess a 3CNF as long as we allow for $\mathrm{poly}(r)$ queries. Using it, there is no benefit in reducing the number of queries to a constant.

Preprocess: This is in turn comprised of two parts, acting in parallel, known as "algebraic constraint satisfaction" and "low-degree testing", and in this work we offer a clear separation between the two. In the first part we reduce the succinct constraint satisfaction problem (CSP) associated with verifying the M accepts x in T steps, to an algebraic CSP (ACSP) problem, one stated as a question about equality of polynomials. We offer a definition of ACSP that is algebraically

cleaner than [BS08] and following works (e.g., [BGH+05,BCGT13b]). In particular, previous definitions included degree bounds on the "assignment polynomial" and involved a "zero-testing" problem. In contrast, we define a satisfying assignment as one that causes a polynomial to vanish, and degree-bounds are dealt with by the separate low-degree testing part, discussed later. We now elaborate on how we obtain efficient preprocessing in each of the two parts.

In the ACSP part, our verifier simply selects a random field element α, generates $\text{poly}(r)$ queries to the prover where the ith query is $\alpha + \sigma_i$ where σ_i is fixed and independent of α. Each such query can be verified to be a projection. To reach this simple form of preprocessing we use a modular reduction from the combinatorial succinct CSP captured by Theorem 4 to the succinct ACSP. The mid-point between the combinatorial and algebraic settings is given in Theorem 5. In it we reach a 3CNF formula with $\approx T$ clauses where each clause (i.e., the three variables of the clause and their polarities) can be computed by a simple XOR operation. Since XOR is addition in fields of characteristic 2, irrespective of the basis chosen for them, we get a simpler ACSP than [BS08,BGH+05] albeit one that has a super-constant number of variables.

Turning to the second part, low-degree testing, we use auxiliary information in form of a PCP of Proximity (PCPP) [BSGH+06,DR06]. This part is essentially from [BS08] and regrettably remains an intricate step of the proof. The answers to queries of the verifier in [BS08] can be seen as arranged in the nodes of a tree. The query at each node is indeed a projection. However, the verifier uses part of its input randomness to select a path in this tree, reaching a leaf, then possibly redirects the query to a node higher up in the tree. This computation is more complicated than just a projection. Here we use the following simple, key idea. The path from root to leaf is determined by only $O(\log r)$ of the verifier's r input bits. Additionally, the process of redirecting a query from a leaf to a node elsewhere in the tree is also determined by only $O(\log r)$ input bits. Instead of following the path, we let the verifier query every possible endpoint. This multiplies the number of queries by a factor $\text{poly}(r)$, which we can afford. We delegate the task of picking the right query to the postprocess.

One more complication is that each of the two parts needs to be combined with a randomness-efficient hitter to achieve constant soundness. Using e.g. Cayley expanders built from small-bias sets, this step is again just a projection.

Ideas Behind the Proof of Theorem 3. A natural idea is to improve the previous constant-locality result [JMV13] to locality 1. But this may not be possible. Instead, we show how to reduce arbitrary computation to a polynomial number of 3CNF formulae, each of which has locality 1. By enumerating over these 3CNF, and running the satisfiability algorithm on each of them, we get the result. This idea is similar to the one described above to make [BS08] a projection: after reading a logarithmic number of bits, the rest of the computation becomes just a projection.

Open Problems. Improve Theorem 2 to have the same overhead as Theorem 3.

Organization. In §2 we give a variant of the reduction of non-deterministic time to 3SAT given in [JMV13]. Using that and Theorem 1 as a black-box, in §3 we give the proofs of theorems 2 and 3. Due to space restrictions we refer to the full version for the proof of our main Theorem 1.

2 A Combinatorial Reduction to 3SAT

Our starting point is the following result from [JMV13].

Theorem 4 ([JMV13]). *Let M be an algorithm running in time $T = T(n) \geq n$ on inputs of the form (x, y) where $|x| = n$. Given $x \in \{0,1\}^n$ one can output a circuit $D : \{0,1\}^\ell \to \{0,1\}^{3v+3}$ in time $\mathrm{poly}(n, \log T)$ mapping an index to a clause of a 3CNF ϕ in v-bit variables, for $v = \Theta(\ell)$, such that*

1. *ϕ is satisfiable iff there is $y \in \{0,1\}^T$ such that $M(x, y)$ accepts, and*
2. *For any $r \leq n$ we can have $\ell = \max(\log T, n/r) + O(\log n) + O(\log\log T)$ and each output bit of D is a decision tree of depth $O(\log r)$.*

Note that for $T = 2^n$ and $r = O(1)$ this gives a 3CNF with $T\mathrm{poly}\log T$ clauses such that each clause can be computed from its index by a function with constant locality.

We need an incomparable variant of the latter. We enlarge the locality to $O(\log n)$, but at the same time there are only $O(\log n)$ input bits that affect more than 1 bit. If we fix these bits, the rest of the computation is just a bit-wise xor.

Theorem 5. *Let M be an algorithm running in time $T = T(n) \geq n$ on inputs of the form (x, y) where $|x| = n$. Let $\ell_1 = \log T$. For some $\ell_2 = O(\log\log T) + O(\log n)$ the following is true. Given $x \in \{0,1\}^n$ one can output in time $\mathrm{poly}(n, \log T)$ six circuits (of size $\mathrm{poly}(n, \log T)$): $S_i : \{0,1\}^{\ell_2} \to \{0,1\}^{\ell_1+\ell_2}, b_i : \{0,1\}^{\ell_2} \to \{0,1\}$, for $i = 1, 2, 3$, such that:*

Let ϕ_x be the 3CNF with $2^{\ell_1+\ell_2} = T\mathrm{poly}(n, \log T)$ clauses (and variables) whose $(\alpha, \beta) \in \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2}$ clause contains variables $V_i = (\alpha, \beta) \oplus S_i(\beta)$ and corresponding sign bits $b_i = b_i(\beta)$, where $i = 1, 2, 3$ and $\oplus$ is bit-wise xor. Then ϕ_x is satisfiable iff there is $y \in \{0,1\}^T$ such that $M(x, y)$ accepts.

Note that in the case $T = 2^{O(n)}$ each output bit of D depends only on $|\beta|+1 = O(\log n)$ bits of the input.

The next proof heavily builds on previous works. We give a sketch that highlights the tiny changes from previous proofs, and to work out parameters. The closest previous proof is [JMV13], to which we also refer for a discussion of other related works.

Proof (Proof sketch). Without loss of generality the algorithm is implemented by a random-access Turing machine running in time $T' = T\mathrm{poly}\log n$ and only using memory cells at indexes $\leq \mathrm{poly}(T)$ (see e.g. [NEU12] for details).

Consider a circuit-sat instance where the circuit first guesses a computation trace consisting of T' configurations of size $O(\log T)$ each; and then the circuit

checks its validity and acceptance. The validity check consists of two separate checks. The first is the check of the consistency of the transition function of the machine, assuming that memory reads are correct. The second is the check of the consistency of the memory reads and writes. The trace is valid if and only if both checks pass. These two checks are implemented in a similar fashion; we only describe the second.

Consider a matrix of $r \times T'$ configurations, where $r = \operatorname{poly}\log T$. We use α to index a column in this matrix. (This actually gives $|\alpha| = \ell_1 = \log T' = \log T + O(\log\log T)$, but the low-order summand can be swallowed in ℓ_2.) We use β to index a row, and the gates within the subcircuits discussed next.

The first row is the computation trace mentioned above that the circuit guessed. For every $t = 1, \ldots, T$ we have a $\operatorname{poly}(n, \log T)$-size subcircuit which checks the pair of configurations (C, C') at positions $(1, t)$ and (r, t) in the matrix, i.e., in the same column but at antipodal rows. This subcircuit verifies that either C' accesses the same memory cell of C and has the timestamp of C plus one, or C' accesses a memory cell with index greater than that of the cell accessed by C, or – the wrap-around condition – it does nothing if C' is the configuration with timestamp 0. If all these checks pass then for every t the configuration at position (t, r) is the one that comes next the configuration at position $(t, 1)$ in the order given by memory location accessed breaking ties by timestamp. The subcircuit then verifies consistency of the memory read and write in C and C'. In particular it verifies that cells read for the first time are blank, and that the cells $1, \ldots, n$ read for the first time contain the input x. Note that the latter is possible because our circuits have size $\geq n$ and are built with knowledge of x.

Observe that these subcircuits operate independently on each column, and are identical across columns. Their connections depend only on the row index and an index to one of their gates. By including these two indexes inside β, these connections can be computed in the required format.

It remains to discuss connections across columns, which are needed to move configurations around to put them in the right order. For this we use routing networks such as Beneš', which are a simple composition of butterfly networks. The index of a neighbor in column α is obtained by xoring α with a string (of Hamming weight 1) which only depends on the row, which in turn is part of β. This leads to the desired format for V_i. The implementation of the routing network also needs a simple gadget to swap two configurations depending on a nondeterministic bit. This gadget is the same for every column and row. From this it follows that the V_i and b_i are in the desired format.

3 Lower Bounds from Fast Algorithms

In this section we prove theorems 2 and 3. First we restate the theorem.

Theorem 3 (Satisfiability Implies Lower Bounds, Tightly). *Let C_n be efficiently closed under projections.*

If the satisfiability of functions $h = g_1 \wedge g_2 \wedge g_3$, where $g_i \in C_{n+O(\log n)}$ is in Time($2^n/n^{\omega(1)}$), then there is a function f in E^{NP} such that $f_n \notin C_n$.

For the proof we use Theorem 5, and we *enumerate* over the β in its statement. The key observation is that for any fixed β, the reduction is only computing xor which can be hardwired with no loss in resources. This enumeration is feasible because $|\beta| = O(\log n)$. To get better constants we also work with unary languages.

Proof (Proof of Theorem 3). Suppose that every function in E^{NP} belongs to C_n when restricted to inputs of length n. Let L be a unary language in $\mathrm{NTime}(2^n) \setminus \mathrm{NTime}(o(2^n))$ [Coo73,SFM78,Zák83]. Consider the E^{NP} algorithm that on input $x' \in \{0,1\}^{O(\log n)}$ and $i \leq 2^n \mathrm{poly}(n)$ computes $x = 1^{x'}$, the 3CNF ϕ_x corresponding to L through Theorem 5, computes its first satisfying assignment if one exists, and outputs its ith bit. By assumption, on inputs of length $m = n + O(\log n)$ this function is in C_m. Also, by assumption, if we hardwire x' the resulting function still belongs to C_m. Call this function g_x.

We contradict the assumption on L by showing how to decide it in $\mathrm{Ntime}(o(2^n))$. Let D, S_i, α, β, V_i and b_i be as in Theorem 5. Consider the algorithm that on input $x = 1^n$ guesses g_x. Then it constructs the function g'_x that operates as follows. The input is that of D. Then it connects three copies of g_x to the output variables V_i. Further, the output of the ith copy is negated and then xored with b_i. And finally an And is taken. Call g'_x this new function (which may not belong to any C_n). Note that $g'_x(i) = 1$ iff the ith clause is not satisfied (by satisfying assignment g_x). So by determining the satisfiability of g'_x we can determine if $x \in L$ or not.

The satisfiability algorithm enumerates over all $\mathrm{poly}(n)$ choices for β. For each fixed β, the b_i are determined, and the remaining computation to obtain the V_i is an xor by $S_i(\beta)$. All this can be hardwired into g'_x in time $\mathrm{poly}(m)$, because C_m is efficiently closed under projections. For every i this gives a new function $g_i \in C_m$. There remains to solve the satisfiability of $g_1 \wedge g_2 \wedge g_3$. The latter can be done in time $2^m/m^{\omega(1)}$ by assumption. So overall the running time is $\mathrm{poly}(n,m)2^m/m^{\omega(1)} = 2^n/n^{\omega(1)} = o(2^n)$.

Theorem 2 (Derandomization Implies Lower Bounds, Tightly). *Let C_n be efficiently closed under projections.*

If the acceptance probability of functions of the form $h = \wedge_{\mathrm{poly}(n)} \vee_3 C_{n+O(\log n)}$ can be distinguished from being $= 1$ or $\leq 1/n^{10}$ in Time$(2^n/n^{\omega(1)})$, then there is a function f in E^{NP} such that $f_n \notin C_n$.

Proof (Proof Sketch). Proceed as the proof of Theorem 3, but let ϕ_x be instead of the 3CNF produced by Theorem 5 the constraint satisfaction problem corresponding to our main theorem, 1. As before, we obtain a function g'_x that on input i determines if the ith constraint is satisfied. (To show that the complexity of this function is as desired, we merge the Not gates of the 3CNF corresponding to Dec with the circuits in $C_{n+O(\log n)}$, using the closure of the class.) Thus, approximately determining how many constraints are satisfied amounts to approximately determinining the number of satisfying assignments to g'_x.

Acknowledgements. The first co-author is grateful to the SCIPR-lab (www.scipr-lab.org) members — Alessandro Chiesa, Daniel Genkin, Lior Greenblatt, Shaul Kfir, Michael Riabzev, Gil Timnat, Eran Tromer and Madars Virza — for helpful discussions. The second co-author thanks Ramamohan Paturi for a conversation on MAX3SAT.

References

AGHP92. Alon, N., Goldreich, O., Håstad, J., Peralta, R.: Simple constructions of almost k-wise independent random variables. Random Structures & Algorithms 3(3), 289–304 (1992)

ASE92. Alon, N., Spencer, J.H., Erdős, P.: The Probabilistic Method. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley and Sons, Inc. (1992)

BCGT13a. Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E.: Fast reductions from RAMs to delegatable succinct constraint satisfaction problems. In: ACM Innovations in Theoretical Computer Science Conf. (ITCS), pp. 401–414 (2013)

BCGT13b. Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E.: On the concrete efficiency of probabilistically-checkable proofs. In: ACM Symp. on the Theory of Computing (STOC), pp. 585–594 (2013)

BGH$^+$05. Ben-Sasson, E., Goldreich, O., Harsha, P., Sudan, M., Vadhan, S.P.: Short PCPs verifiable in polylogarithmic time. In: IEEE Conf. on Computational Complexity (CCC), pp. 120–134 (2005)

Blu84. Blum, N.: A boolean function requiring 3n network size. Theoretical Computer Science 28, 337–345 (1984)

BS08. Ben-Sasson, E., Sudan, M.: Short PCPs with polylog query complexity. SIAM J. on Computing 38(2), 551–607 (2008)

BSGH$^+$06. Ben-Sasson, E., Goldreich, O., Harsha, P., Sudan, M., Vadhan, S.P.: Robust PCPs of proximity, shorter PCPs, and applications to coding. SIAM J. on Computing 36(4), 889–974 (2006)

Coo73. Cook, S.A.: A hierarchy for nondeterministic time complexity. J. of Computer and System Sciences 7(4), 343–353 (1973)

Din07. Dinur, I.: The PCP theorem by gap amplification. J. of the ACM 54(3), 12 (2007)

DK11. Demenkov, E., Kulikov, A.S.: An elementary proof of a $3n - o(n)$ lower bound on the circuit complexity of affine dispersers. In: Symp. on Math. Foundations of Computer Science (MFCS), pp. 256–265 (2011)

DR06. Dinur, I., Reingold, O.: Assignment testers: Towards a combinatorial proof of the PCP theorem. SIAM J. on Computing 36(4), 975–1024 (2006)

FSUV13. Fefferman, B., Shaltiel, R., Umans, C., Viola, E.: On beating the hybrid argument. Theory of Computing 9, 809–843 (2013)

GMR13. Gopalan, P., Meka, R., Reingold, O.: DNF sparsification and a faster deterministic counting algorithm. Computational Complexity 22(2), 275–310 (2013)

Her11. Hertli, T.: 3-SAT faster and simpler - unique-SAT bounds for PPSZ hold in general. In: IEEE Symp. on Foundations of Computer Science (FOCS), pp. 277–284 (2011)

IKW01. Impagliazzo, R., Kabanets, V., Wigderson, A.: In search of an easy witness: Exponential time vs. probabilistic polynomial time. In: IEEE Conf. on Computational Complexity (CCC) (2001)

IKW02. Impagliazzo, R., Kabanets, V., Wigderson, A.: In search of an easy witness: exponential time vs. probabilistic polynomial time. J. of Computer and System Sciences 65(4), 672–694 (2002)

JMV13. Jahanjou, H., Miles, E., Viola, E.: Local reductions (2013), http://www.ccs.neu.edu/home/viola/

KL80. Karp, R.M., Lipton, R.J.: Some connections between nonuniform and uniform complexity classes. In: ACM Symp. on the Theory of Computing (STOC), pp. 302–309 (1980)

LVW93. Luby, M., Veličković, B., Wigderson, A.: Deterministic approximate counting of depth-2 circuits. In: 2nd Israeli Symposium on Theoretical Computer Science (ISTCS), pp. 18–24 (1993)

Mie09. Mie, T.: Short pcpps verifiable in polylogarithmic time with $o(1)$ queries. Ann. Math. Artif. Intell. 56(3-4), 313–338 (2009)

MTY11. Makino, K., Tamaki, S., Yamamoto, M.: Derandomizing HSSW algorithm for 3-SAT. CoRR, abs/1102.3766 (2011)

NEU12. NEU. From RAM to SAT (2012), http://www.ccs.neu.edu/home/viola/

NN90. Naor, J., Naor, M.: Small-bias probability spaces: efficient constructions and applications. In: 22nd ACM Symp. on the Theory of Computing (STOC), pp. 213–223. ACM (1990)

Oli13. Oliveira, I.C.: Algorithms versus circuit lower bounds. CoRR, abs/1309.0249 (2013)

SFM78. Seiferas, J.I., Fischer, M.J., Meyer, A.R.: Separating nondeterministic time complexity classes. J. of the ACM 25(1), 146–167 (1978)

SW13. Santhanam, R., Williams, R.: On medium-uniformity and circuit lower bounds. In: IEEE Conf. on Computational Complexity (CCC) (2013)

Vio07. Viola, E.: Pseudorandom bits for constant-depth circuits with few arbitrary symmetric gates. SIAM J. on Computing 36(5), 1387–1403 (2007)

Vio13. Viola, E.: Challenges in computational lower bounds (2013), http://www.ccs.neu.edu/home/viola/

Wil05. Williams, R.: A new algorithm for optimal 2-constraint satisfaction and its implications. Theoretical Computer Science 348(2-3), 357–365 (2005)

Wil10. Williams, R.: Improving exhaustive search implies superpolynomial lower bounds. In: 42nd ACM Symp. on the Theory of Computing (STOC), pp. 231–240 (2010)

Wil11. Williams, R.: Non-uniform ACC circuit lower bounds. In: IEEE Conf. on Computational Complexity (CCC), pp. 115–125 (2011)

Wil13. Williams, R.: Improving exhaustive search implies superpolynomial lower bounds. SIAM J. on Computing 42(3), 1218–1244 (2013)

Wil14. Williams, R.: New algorithms and lower bounds for circuits with linear threshold gates. In: ACM Symp. on the Theory of Computing, STOC (2014)

Zák83. Zák, S.: A turing machine time hierarchy. Theoretical Computer Science 26, 327–333 (1983)