Electronic Colloquium on Computational Complexity, Revision 3 of Report No. 49 (2008)

Derandomizing the Isolation Lemma and Lower Bounds for Circuit Size

V. Arvind and Partha Mukhopadhyay
Institute of Mathematical Sciences
C.I.T Campus,Chennai 600 113, India
{arvind,partham}@imsc.res.in

Abstract

The isolation lemma of Mulmuley et al [MVV87] is an important tool in the design of randomized algorithms and has played an important role in several nontrivial complexity upper bounds. On the other hand, polynomial identity testing is a well-studied algorithmic problem with efficient randomized algorithms and the problem of obtaining efficient *deterministic* identity tests has received a lot of attention recently. The goal of this paper is to compare the isolation lemma with polynomial identity testing:

1. We show that derandomizing reasonably restricted versions of the isolation lemma implies circuit size lower bounds. We derive the circuit lower bounds by examining the connection between the isolation lemma and polynomial identity testing. We give a randomized polynomial-time identity test for noncommutative circuits of polynomial degree based on the isolation lemma. Using this result, we show that derandomizing the isolation lemma implies noncommutative circuit size lower bounds. For the commutative case, a stronger derandomization hypothesis allows us to construct an explicit multilinear polynomial that does not have subexponential size commutative circuits. The restricted versions of the isolation lemma we consider are natural and would suffice for the standard applications of the isolation lemma.
2. From the result of Klivans-Spielman [KS01] we observe that there is a randomized polynomial-time identity test for commutative circuits of polynomial degree, also based on a more general isolation lemma for linear forms. Consequently, derandomization of (a suitable version of) this isolation lemma implies that either NEXP $\not\subset$ P/poly or the Permanent over $\mathbb{Z}$ does not have polynomial-size arithmetic circuits.

1 Introduction

We recall the Isolation Lemma [MVV87]. Let $[n]$ denote the set $\{1, 2, \cdots, n\}$. Let U be a set of size n and $\mathcal{F} \subseteq 2^U$ be any family of subsets of U. Let $w : U \to \mathbb{Z}^+$ be a weight function that assigns positive integer weights to the elements of U. For $T \subseteq U$, define its weight $w(T)$ as $w(T) = \sum_{u \in T} w(u)$. Then Isolation Lemma guarantees that for any family of subsets $\mathcal{F}$ of U and for any random weight assignment $w : U \to [2n]$, with high probability there will be a unique minimum weight set in $\mathcal{F}$.

Lemma 1.1 (Isolation Lemma) [MVV87] Let U be an universe of size n and $\mathcal{F}$ be any family of subsets of U. Let $w : U \to [2n]$ denote a weight assignment function to elements of U. Then,

$$\text{Prob}_w[\text{ There exists a unique minimum weight set in } \mathcal{F}] \geq \frac{1}{2},$$

where the weight function w is picked uniformly at random.

ISSN 1433-8092

In the seminal paper [MVV87] Mulmuley et al apply the isolation lemma to give a randomized NC algorithm for computing maximum cardinality matchings for general graphs (also see [ARZ99]). Since then the isolation lemma has found several other applications. For example, it is crucially used in the proof of the result that NL $\subset$ UL/poly [AR00] and in designing randomized NC algorithms for linear representable matroid problems [NSV94]. It is also known that the isolation lemma can be used to prove the Valiant-Vazirani lemma that SAT is many-one reducible via randomized reductions to USAT.

Whether the matching problem is in deterministic NC, and whether NL $\subseteq$ UL are outstanding open problems. Thus, the question whether the isolation lemma can be derandomized is clearly important.

As noted in [Agr07], it is easy to see by a counting argument that the isolation lemma can not be derandomized, in general, because there are 2^{2^n} set systems $\mathcal{F}$. More formally, the following is observed in [Agr07].

Observation 1.2 [Agr07] *The Isolation Lemma can not be fully derandomized if we allow weight functions $w : U \to [n^c]$ for a constant c (i.e. weight functions with a polynomial range). More precisely, for any polynomially bounded collection of weight assignments $\{w_i\}_{i\in[n^{c_1}]}$ with weight range $[n^c]$, there exists a family $\mathcal{F}$ of $[n]$ such that for all $j \in [n^{c_1}]$, there exists two minimal weight subsets with respect to w_j.*

However that does not rule out the derandomization of any special usage of the isolation lemma. Indeed, for all applications of the isolation lemma (mentioned above, for instance) we are interested only in exponentially many set systems $\mathcal{F} \subseteq 2^U$.

We make the setting more precise by giving a general framework. Fix the universe $U = [n]$ and consider an n-input boolean circuit C where $size(C) = m$. The set 2^U of all subsets of U is in a natural 1-1 correspondence with the length n-binary strings $\{0,1\}^n$: each subset $S \subseteq U$ corresponds to its characteristic binary string $\chi_S \in \{0,1\}^n$ whose i^{th} bit is 1 iff $i \in S$. Thus the n-input boolean circuit C implicitly defines the set system

$$\mathcal{F}_C = \{S \subseteq [n] \mid C(\chi_S) = 1\}.$$

As an easy consequence of Lemma 1.1 we have the following.

Lemma 1.3 *Let U be an universe of size n and C be an n-input boolean circuit of size m. Let $\mathcal{F}_C \subseteq 2^U$ be the family of subsets of U defined by circuit C. Let $w : U \to [2n]$ denote a weight assignment function to elements of U. Then,*

$$\text{Prob}_w[\text{ There exists a unique minimum weight set in } \mathcal{F}_C] \geq \frac{1}{2},$$

where the weight function w is picked uniformly at random. Furthermore, there is a collection of weight functions $\{w_i\}_{1\leq i\leq p(m,n)}$, where $p(m,n)$ is a fixed polynomial, such that for each $\mathcal{F}_C$ there is a weight function w_i w.r.t. which there is a unique minimum weight set in $\mathcal{F}_C$.

Lemma 1.3 allows us to formulate two natural and reasonable derandomization hypotheses for the isolation lemma.

Hypothesis 1. There is a deterministic algorithm $\mathcal{A}_1$ that takes as input (C, n), where C is an n-input boolean circuit, and outputs a collection of weight functions $w_1, w_2, \cdots, w_t$ such that $w_i : [n] \to [2n]$, with the property that for some w_i there is a unique minimum weight set in the set system $\mathcal{F}_C$. Furthermore, $\mathcal{A}_1$ runs in time subexponential in $size(C)$.

Hypothesis 2. There is a deterministic algorithm $\mathcal{A}_2$ that takes as input (m, n) in unary and outputs a collection of weight functions $w_1, w_2, \cdots, w_t$ such that $w_i : [n] \to [2n]$, with the property that for each size m boolean circuit C with n inputs there is some weight function w_i w.r.t. which $\mathcal{F}_C$ has a unique minimum weight set. Furthermore, $\mathcal{A}_2$ runs in time polynomial in m.

Clearly, Hypothesis 2 is stronger than Hypothesis 1. It demands a "black-box" derandomization in the sense that $\mathcal{A}_2$ efficiently computes a collection of weight functions that will work for *any* set system in 2^U specified by a boolean circuit of size m.

Notice that a random collection $w_1, \cdots, w_t$ of weight functions will fulfil the required property of either hypotheses with high probability. Thus, the derandomization hypotheses are plausible. Indeed, it is not hard to see that suitable standard hardness assumptions that yield pseudorandom generators for derandomizing BPP would imply these hypotheses. We do not elaborate on this here. In this paper we show the following consequences of Hypotheses 1 and 2.

1. Hypothesis 1 implies that either NEXP $\not\subset$ P/poly or the Permanent does not have polynomial size noncommutative arithmetic circuits.

2. Hypothesis 2 implies that for almost all n there is an explicit multilinear polynomial $f_n(x_1, x_2, \cdots, x_n) \in \mathbb{F}[x_1, x_2, \cdots, x_n]$ in *commuting* variables x_i (where by explicit we mean that the coefficients of the polynomial f_n are computable by a uniform algorithm in time exponential in n) that does not have commutative arithmetic circuits of size $2^{o(n)}$ (where the field $\mathbb{F}$ is either the rationals or a finite field).

The first result is a consequence of an identity testing algorithm for noncommutative circuits that is based on the isolation lemma. This algorithm is based on ideas from [AMS08] where we used automata theory to pick matrices from a suitable matrix ring and evaluate the given arithmetic circuit on these matrices. In the next section, we describe the background and then give the identity test in the following section.

Remark 1.4 *Notice that derandomizing the isolation lemma in specific applications like the* RNC *algorithm for matchings* [MVV87] *and the containment* NL $\subseteq$ UL/poly [AR00] *might still be possible without implying such circuit size lower bounds.*

Noncommutative circuits

Noncommutative polynomial identity testing has been the focus of recent research [RS05, BW05, AMS08]. One reason to believe that it could be easier than the commutative case to derandomize is because lower bounds are somewhat easier to prove in the noncommutative setting as shown by Nisan [N91]. Using a rank argument Nisan has shown exponential size lower bounds for noncommutative formulas (and noncommutative algebraic branching programs) that compute the noncommutative permanent or determinant polynomials in the ring $\mathbb{F}\{x_1, \cdots, x_n\}$ where x_i are noncommuting variables. In [CS07], Chien and Sinclair further extend Nisan's idea to prove exponential size lower bounds for noncommutative formulas computing noncommutative permanent or determinant polynomial over matrix algebra, quaternion algebra and group algebra. However, no superpolynomial lower bounds are known for the size of noncommutative circuits for explicit polynomials.

Our result in this paper is similar in flavour to the Impagliazzo-Kabanets result [KI03], where for *commutative* polynomial identity testing they show that derandomizing polynomial identity testing implies circuit

lower bounds. Specifically, it implies that either NEXP $\not\subset$ P/poly or the integer Permanent does not have polynomial-size arithmetic circuits.

In [AMS08] we have observed that an analogous result also holds in the noncommutative setting. I.e., if noncommutative PIT has a deterministic polynomial-time algorithm then either NEXP $\not\subset$ P/poly or the *noncommutative* Permanent function does not have polynomial-size noncommutative circuits.

The connection that we show here between derandomizing the isolation lemma and noncommutative circuit size lower bounds is based on the above observation and our noncommutative polynomial identity test based on the isolation lemma.

Commutative circuits

As a consequence of Hypothesis 2 we are able to show that for almost all n there is an explicit multilinear polynomial $f_n(x_1, x_2, \cdots, x_n) \in \mathbb{F}[x_1, x_2, \cdots, x_n]$ in *commuting* variables x_i (where by explicit we mean that the coefficients of the polynomial f_n are computable by a uniform algorithm in time exponential in n) that does not have commutative arithmetic circuits of size $2^{o(n)}$ (where the field $\mathbb{F}$ is either the rationals or a finite field). This is a fairly easy consequence of the univariate substitution idea and the observation that for arithmetic circuits computing multilinear polynomials, we can efficiently test if a monomial has nonzero coefficient (Lemma 2.6).

Klivans and Spielman [KS01] apply a more general form of the isolation lemma to obtain a polynomial identity test (in the commutative) case. This lemma is stated below.

Lemma 1.5 [KS01, Lemma 4] *Let L be any collection of linear forms over variables $z_1, z_2, \cdots, z_n$ with integer coefficients in the range $\{0, 1, \cdots, K\}$. If each z_i is picked independently and uniformly at random from $\{0, 1, \cdots, 2Kn\}$ then with probability at least $1/2$ there is a unique linear form from C that attains minimum value at $(z_1, \cdots, z_n)$.*

We can formulate a restricted version of this lemma similar to Lemma 1.3 that will apply only to sets of linear forms L accepted by a boolean circuit C. More precisely, an integer vector $(\alpha_1, \cdots, \alpha_n)$ such that $\alpha_i \in \{0, \cdots, K\}$ is in L if and only if $(\alpha_1, \cdots, \alpha_n)$ is accepted by the boolean circuit C.

Thus, for this form of the isolation lemma we can formulate another derandomization hypothesis analogous to Hypothesis 2 as follows.

Hypothesis 3. There is a deterministic algorithm $\mathcal{A}_3$ that takes as input (m, n, K) and outputs a collection of weight functions $w_1, w_2, \cdots, w_t$ such that $w_i : [n] \to [2Kn]$, with the property that for any size m boolean circuit C that takes as input $(\alpha_1, \cdots, \alpha_n)$ with $\alpha_i \in \{0, \cdots, K\}$ there is some weight vector w_i for which there is a *unique* linear form $(\alpha_1, \cdots, \alpha_n)$ accepted by C which attains the minimum value $\sum_{j=1}^{n} w_i(j)\alpha_j$. Furthermore, $\mathcal{A}_3$ runs in time subexponential in $size(C)$.

We show that Hypothesis 3 yields a lower bound consequence for the integer permanent.

2 Automata Theory background

We recall some standard automata theory [HU78]. Fix a finite automaton $A = (Q, \Sigma, \delta, q_0, q_f)$ which takes inputs in Σ^*, Σ is the alphabet, Q is the set of states, $\delta : Q \times \Sigma \to Q$ is the transition function, and q_0 and q_f are the initial and final states respectively (we only consider automata with unique accepting states). For each $b \in \Sigma$, let $\delta_b : Q \to Q$ be defined by: $\delta_b(q) = \delta(q, b)$. These functions generate a submonoid of the

monoid of all functions from Q to Q. This is the transition monoid of the automaton A and is well-studied in automata theory [Str94, page 55]. We now define the 0-1 matrix $M_b \in \mathbb{F}^{|Q|\times|Q|}$ as follows:

$$M_b(q,q') = \begin{cases} 1 & \text{if } \delta_b(q) = q', \\ 0 & \text{otherwise.} \end{cases}$$

The matrix M_b is the adjacency matrix of the graph of δ_b. As M_b is a 0-1 matrix, we can consider it as a matrix over any field $\mathbb{F}$.

For a string $w = w_1 w_2 \cdots w_k \in \Sigma^*$ we define M_w to be the matrix product $M_{w_1} M_{w_2} \cdots M_{w_k}$. If w is the empty string, define M_w to be the identity matrix of dimension $|Q| \times |Q|$. Let δ_w denote the natural extension of the transition function to w; if w is the empty string, δ_w is simply the identity function. We have

$$M_w(q,q') = \begin{cases} 1 & \text{if } \delta_w(q) = q', \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Thus, M_w is also a matrix of zeros and ones for any string w. Also, $M_w(q_0, q_f) = 1$ if and only if w is accepted by the automaton A.

2.1 Noncommutative arithmetic circuits and automata

This subsection is reproduced from [AMS08] to make this paper self-contained.

Consider the ring $\mathbb{F}\{x_1, \cdots, x_n\}$ of polynomials with noncommuting variables $x_1, \cdots, x_n$ over a field $\mathbb{F}$. Let C be a noncommutative arithmetic circuit computing a polynomial $f \in \mathbb{F}\{x_1, \cdots, x_n\}$. Let d be an upper bound on the degree of f. We can consider monomials over $x_1, \cdots, x_n$ as strings over the alphabet $\Sigma = \{x_1, x_2, \cdots, x_n\}$.

Let $A = (Q, \Sigma, \delta, q_0, q_f)$ be a finite automaton over the alphabet $\Sigma = \{x_1, x_2, \cdots, x_n\}$. We have matrices $M_{x_i} \in \mathbb{F}^{|Q|\times|Q|}$ as defined in Section 2. We are interested in the output matrix obtained when the inputs x_i to the circuit C are replaced by the matrices M_{x_i}. This output matrix is defined in the obvious way: the inputs are $|Q| \times |Q|$ matrices and we do matrix addition and matrix multiplication at each addition gate (respectively, multiplication gate) of the circuit C. We define the *output of C on the automaton A* to be this output matrix M_{out}. Clearly, given circuit C and automaton A, the matrix M_{out} can be computed in time poly$(|C|, |A|, n)$.

We observe the following property: the matrix output M_{out} of C on A is determined completely by the polynomial f computed by C; the structure of the circuit C is otherwise irrelevant. This is important for us, since we are only interested in f. In particular, the output is always 0 when $f \equiv 0$.

More specifically, consider what happens when C computes a polynomial with a single term, say $f(x_1, \cdots, x_n) = c x_{j_1} \cdots x_{j_k}$, with a non-zero coefficient $c \in \mathbb{F}$. In this case, the output matrix M_{out} is clearly the matrix $cM_{x_{j_1}} \cdots M_{x_{j_k}} = cM_w$, where $w = x_{j_1} \cdots x_{j_k}$. Thus, by Equation 1 above, we see that the entry $M_{out}(q_0, q_f)$ is 0 when A rejects w, and c when A accepts w. In general, suppose C computes a polynomial $f = \sum_{i=1}^{t} c_i m_i$ with t nonzero terms, where $c_i \in \mathbb{F} \setminus \{0\}$ and $m_i = \prod_{j=1}^{d_i} x_{i_j}$, where $d_i \leq d$. Let w_i denotes the string representing monomial m_i. Finally, let $S_A^f = \{i \in \{1, \cdots, t\} \mid A \text{ accepts } w_i\}$.

Theorem 2.1 [AMS08] *Given any arithmetic circuit C computing polynomial $f \in \mathbb{F}\{x_1, \cdots, x_n\}$ and any finite automaton $A = (Q, \Sigma, \delta, q_0, q_f)$, then the output M_{out} of C on A is such that $M_{out}(q_0, q_f) = \sum_{i \in S_A^f} c_i$.*

Proof. The proof is an easy consequence of the definitions and the properties of the matrices M_w stated in Section 2. Note that $M_{out} = f(M_{x_1}, \cdots, M_{x_n})$. But $f(M_{x_1}, \cdots, M_{x_n}) = \sum_{i=1}^{s} c_i M_{w_i}$, where w_i is the string representing monomial m_i. By Equation 1, we know that $M_{w_i}(q_0, q_f)$ is 1 if w_i is accepted by A, and 0 otherwise. Adding up, we obtain the result. ■

We now explain the role of the automaton A in testing if the polynomial f computed by C is identically zero. Our basic idea is to design an automaton A that accepts exactly one word among all the words that correspond to the nonzero terms in f. This would ensure that $M_{out}(q_0, q_f)$ is the nonzero coefficient of the monomial filtered out. More precisely, we will use the above theorem primarily in the following form, which we state as a corollary.

Corollary 2.2 [AMS08] *Given any arithmetic circuit C computing polynomial $f \in \mathbb{F}\{x_1, \cdots, x_n\}$ and any finite automaton $A = (Q, \Sigma, \delta, q_0, q_f)$, then the output M_{out} of C on A satisfies:*

(1) If A rejects every string corresponding to a monomial in f, then $M_{out}(q_0, q_f) = 0$.

(2) If A accepts exactly one string corresponding to a monomial in f, then $M_{out}(q_0, q_f)$ is the nonzero coefficient of that monomial in f.

Moreover, M_{out} can be computed in time poly$(|C|, |A|, n)$.

Proof. Both points (1) and (2) are immediate consequences of the above theorem. The complexity of computing M_{out} easily follows from its definition. ■

Another interesting corollary to the above theorem is the following.

Corollary 2.3 [AMS08] *Given any arithmetic circuit C over $\mathbb{F}\{x_1, \cdots, x_n\}$, and any monomial m of degree d_m, we can compute the coefficient of m in C in time* poly$(|C|, d_m, n)$.

Proof. Apply Corollary 2.2 with A being any standard automaton that accepts the string corresponding to monomial m and rejects every other string. Clearly, A can be chosen so that A has a unique accepting state and $|A| = O(nd_m)$. ■

Infact corollary 2.3 says that, given an arithmetic circuit C and a monomial m, there is an uniform way to generate a polynomial-size boolean circuit C' such that C' can decide whether m is a nonzero monomial in the polynomial computed by C. The boolean circuit C' is simply the description of the algorithm described in the proof of corollary 2.3.

Corollary 2.4 *Given an arithmetic circuit C over $\mathbb{F}\{x_1, \cdots, x_n\}$ and a monomial m of degree d, there is an uniform way to generate a* poly$(|C|, d, n)$ *size boolean circuit C' that decides whether m is a nonzero monomial in C.*

Remark 2.5 *Corollary 2.3 is very unlikely to hold in the commutative ring $\mathbb{F}[x_1, \cdots, x_n]$. For, it is easy to see that in the commutative case computing the coefficient of the monomial $\prod_{i=1}^{n} x_i$ in even a product of linear forms $\Pi_i \ell_i$ is at least as hard as computing the permanent over $\mathbb{F}$, which is #*P*-complete when $\mathbb{F} = \mathbb{Q}$. However, we can show the following for commutative circuits computing multilinear polynomials.*

Corollary 2.6 *Given a commutative arithmetic circuit $\hat{C}$ over $\mathbb{F}[x_1, \cdots, x_n]$, with the promise that $\hat{C}$ computes a* multilinear *polynomial, and any monomial $m = \prod_{i \in S} x_i$ where $S \subseteq [n]$, we can compute the coefficient of m in C in time* poly$(|\hat{C}|, n)$. *Furthermore, there is an uniform way to generate a boolean circuit C' of size* poly$(|C|, n)$ *such that C' takes as input description of C and m and decides whether m is a nonzero monomial in C.*

Proof. Let $m = \prod_{i \in S} x_i$ be the given monomial. The algorithm will simply substitute y (a new variable) for each x_i such that $i \in S$ and 0 for each x_i such that $i \notin S$ and evaluate the circuit $\hat{C}$ to find the coefficient of the highest degree of y. The boolean circuit C' is simply the description of the above algorithm. It is clear that C' can be uniformly generated. ■

3 Noncommutative identity test based on isolation lemma

We now describe a new identity test for noncommutative circuits based on the isolation lemma. It is directly based on the results from [AMS08]. This is conceptually quite different from the randomized identity test of Bogdanov and Wee [BW05].

Theorem 3.1 *Let $f \in \mathbb{F}\{x_1, x_2, \cdots, x_n\}$ be a polynomial given by an arithmetic circuit C of size m. Let d be an upper bound on the degree of f. Then there is a randomized algorithm which runs in time* poly(n, m, d) *and can test whether $f \equiv 0$.*

Proof. Let $[d] = \{1, 2, \cdots, d\}$ and $[n] = \{1, 2, \cdots, n\}$. Consider the set of tuples $U = [d] \times [n]$. Let $v = x_{i_1} x_{i_2} \cdots x_{i_t}$ be a nonzero monomial of f. Then the monomial can be identified with the following subset S_v of U :

$$S_v = \{(1, i_1), (2, i_2), \cdots, (t, i_t)\}$$

Let $\mathcal{F}$ denotes the family of subsets of U corresponding to the nonzero monomials of f i.e,

$$\mathcal{F} = \{S_v \mid v \text{ is a nonzero monomial in } f\}$$

By the Isolation Lemma we know that if we assign random weights from $[2dn]$ to the elements of U, with probability at least $1/2$, there is a unique minimum weight set in $\mathcal{F}$. Our aim will be to construct a family of small size automatons which are indexed by weights $w \in [2nd^2]$ and $t \in [d]$, such that the automata $A_{w,t}$ will precisely accept all the strings (corresponding to the monomials) v of length t, such that the weight of S_v is w. Then from the isolation lemma we will argue that the automata corresponding to the minimum weight will precisely accept only one string (monomial). Now for $w \in [2nd^2]$, and $t \in [d]$, we describe the construction of the automaton $A_{w,t} = (Q, \Sigma, \delta, q_0, F)$ as follows: $Q = [d] \times [2nd^2] \cup \{(0, 0)\}$, $\Sigma = \{x_1, x_2, \cdots, x_n\}$, $q_0 = \{(0, 0)\}$ and $F = \{(t, w)\}$. We define the transition function $\delta : Q \times \Sigma \to Q$,

$$\delta((i, V), x_j) = (i + 1, V + W),$$

where W is the random weight assign to $(i + 1, j)$. Our automata family $\mathcal{A}$ is simply,

$$\mathcal{A} = \{A_{w,t} \mid w \in [2nd^2], t \in [d]\}.$$

Now for each of the automaton $A_{w,t} \in \mathcal{A}$, we mimic the run of the automaton $A_{w,t}$ on the circuit C as described in Section 2. If the output matrix corresponding to any of the automaton is nonzero, our algorithm declares $f \neq 0$, otherwise declares $f \equiv 0$.

The correctness of the algorithm follows easily from the Isolation Lemma. By the Isolation Lemma we know, on random assignment, a unique set S in $\mathcal{F}$ gets the minimum weight $w_{\min}$ with probability at least $1/2$. Let S corresponds to the monomial $x_{i_1}x_{i_2}\cdots x_{i_\ell}$. Then the automaton $A_{w_{\min},\ell}$ accepts the string (monomial) $x_{i_1}x_{i_2}\cdots x_{i_\ell}$. Furthermore, as no other set in $\mathcal{F}$ get the same minimum weight, $A_{w_{min},\ell}$ rejects all the other monomials. So the (q_0, q_f) entry of the output matrix M_o, that we get in running $A_{w_{\min},\ell}$ on C is nonzero. Hence with probability at least $1/2$, our algorithm correctly decide that f is nonzero. The success probability can be boosted to any constant by standard independent repetition of the same algorithm. Finally, it is trivial to see that the algorithm always decides correctly if $f \equiv 0$. ■

4 Noncommutative identity testing and circuit lower bounds

For commutative circuits, Impagliazzo and Kabanets [KI03] have shown that derandomizing PIT implies circuit lower bounds. It implies that either NEXP $\not\subset$ P/poly or the integer Permanent does not have polynomial-size arithmetic circuits.

In [AMS08] we have observed that this also holds in the noncommutative setting. I.e., if noncommutative PIT has a deterministic polynomial-time algorithm then either NEXP $\not\subset$ P/poly or the *noncommutative* Permanent function does not have polynomial-size noncommutative circuits. We note here that noncommutative circuit lower bounds are sometimes easier to prove than for commutative circuits. E.g. Nisan [N91], Chien and Sinclair [CS07] have shown exponential-size lower bounds for noncommutative formula size and further results are known for pure noncommutative circuits [N91, RS05]. However, proving superpolynomial size lower bounds for general noncommutative circuits computing the Permanent has remained an open problem.

To keep this paper self contained, we briefly recall the discussion from [AMS08].

The noncommutative Permanent function over integer $Perm(x_1, \cdots, x_n) \in \mathbb{Z}\{x_1, \cdots, x_n\}$ ($\mathbb{Z}$ is the set of integer) is defined as:

$$Perm(x_1, \cdots, x_n) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} x_{i,\sigma(i)}.$$

Let SUBEXP denote $\cap_{\epsilon>0}\mathrm{DTIME}(2^{n^\epsilon})$ and NSUBEXP denote $\cap_{\epsilon>0}\mathrm{NTIME}(2^{n^\epsilon})$.

Theorem 4.1 [AMS08] *If* PIT *for noncommutative circuits of polynomial degree $C(x_1, \cdots, x_n) \in \mathbb{Z}\{x_1, \cdots, x_n\}$ is in* SUBEXP*, then either* NEXP $\not\subset$ P/poly *or the* noncommutative *Permanent function does not have polynomial-size noncommutative circuits.*

Proof. Suppose NEXP $\subset$ P/poly. Then, by the main result of [IKW02] we have NEXP = MA. Furthermore, by Toda's theorem MA $\subseteq$ P$^{Perm_{\mathbb{Z}}}$, where the oracle computes the integer permanent. Now, assuming PIT for noncommutative circuits of polynomial degree is in deterministic subexponential-time, we will show that the (noncommutative) Permanent function does not have polynomial-size noncommutative circuits. Suppose to the contrary that it does have polynomial-size noncommutative circuits. Clearly, we can use it to compute the integer permanent as well. Furthermore, as in [KI03] we notice that the noncommutative $n \times n$ Permanent is also uniquely characterized by the identities $p_1(x) \equiv x$ and $p_i(X) = \sum_{j=1}^{i} x_{1j}p_{i-1}(X_j)$ for $1 < i \leq n$, where X is a matrix of i^2 noncommuting variables and X_j is its j-th minor w.r.t. the first row. I.e. the polynomials $p_i, 1 \leq i \leq n$ satisfy these n identities over *noncommuting* variables $x_{ij}, 1 \leq i, j \leq n$ if and only if p_i computes the $i \times i$ permanent of noncommuting variables. The rest of the proof is exactly as in Impagliazzo-Kabanets [KI03]. We can easily describe an NP machine to simulate a P$^{Perm_{\mathbb{Z}}}$ computation.

The NP machine guesses a polynomial-size noncommutative circuit for $Perm$ on $m \times m$ matrices, where m is a polynomial bound on the matrix size of the queries made in the computation of the $\mathrm{P}^{Perm_{\mathbb{Z}}}$ machine. Then the NP machine verifies that the circuit computes the permanent by checking the m *noncommutative* identities it must satisfy. This can be done in SUBEXP by assumption. Finally, the NP machines uses the circuit to answer all the integer permanent queries that are made in the computation of $\mathrm{P}^{Perm_{\mathbb{Z}}}$ machine. Putting it together, we get NEXP $\subseteq$ NSUBEXP which contradicts the nondeterministic time hierarchy theorem. ■

5 The Results

We are now ready to prove our first result. Suppose the derandomization Hypothesis 1 holds (as stated in the introduction): i.e. suppose there is a deterministic algorithm $\mathcal{A}_1$ that takes as input (C, n) where C is an n-input boolean circuit and in subexponential time computes a set of weight functions $w_1, w_2, \cdots, w_t$, $w_i : [n] \rightarrow [2n]$ such that the set system $\mathcal{F}_C$ defined by the circuit C has a unique minimum weight set w.r.t. at least one of the weight functions w_i.

Let $C'(x_1, x_2, \cdots, x_n)$ be a noncommutative arithmetic circuit of degree d bounded by a polynomial in $size(C')$. By Corollary 2.3, there is a deterministic polynomial-time algorithm that takes as input C' and a monomial m of degree at most d and accepts if and only if the monomial m has nonzero coefficient in the polynomial computed by C'. Moreover by corollary 2.4, we have a uniformly generated boolean circuit C of size polynomial in $size(C')$ that accepts only the (binary encodings of) monomials $x_{i_1}x_{i_2}\cdots x_{i_k}$, $k \leq d$ that have nonzero coefficients in the polynomial computed by C'. Now, as a consequence of Theorem 3.1 and its proof we have a *deterministic* subexponential algorithm for checking if $C' \equiv 0$, assuming algorithm $\mathcal{A}_1$ exists. Namely, we compute the boolean circuit C from C' in polynomial time. Then, invoking algorithm $\mathcal{A}_1$ with C as input we compute at most subexponentially many weight functions $w_1, \cdots, w_t$. Then, following the proof of Theorem 3.1 we construct the automata corresponding to these weight functions and evaluate C' on the matrices that each of these automata define in the prescribed manner. By assumption about algorithm $\mathcal{A}_1$, if $C' \not\equiv 0$ then one of these w_i will give matrix inputs for the variables $x_j, 1 \leq j \leq n$ on which C' evaluates to a nonzero matrix. We can now show the following theorem.

Theorem 5.1 *If the subexponential time algorithm $\mathcal{A}_1$ satisfying Hypothesis 1 exists then noncommutative identity testing is in* SUBEXP *which implies that either* NEXP $\not\subset$ P/poly *or the Permanent does not have polynomial size noncommutative circuits.*

Proof. The result is a direct consequence of the discussion preceding the theorem statement and Theorem 4.1. ■

Commutative circuits

We now turn to the result under the *stronger* derandomization Hypothesis 2 (stated in the introduction). More precisely, suppose there is a deterministic algorithm $\mathcal{A}_2$ that takes as input (m, n) and in time polynomial in m computes a set of weight functions $w_1, w_2, \cdots, w_t$, $w_i : [n] \rightarrow [2n]$ such that for *each* n-input boolean circuit C of size m, the set system $\mathcal{F}_C$ defined by the circuit C has a unique minimum weight set w.r.t. at least one of the weight functions w_i. We show that there is an *explicit* polynomial[1] $f(x_1, \cdots, x_n)$ in commuting

[1]By explicit we mean that the coefficients of f are computable in time exponential in n.

variables x_i that does not have subexponential size *commutative* circuits. The following theorem is similar in flavour to the Agrawal's result that a black-box derandomization of PIT for a class of arithmetic circuit via pseudorandom generator will show similar lower bound (Lemma 5.1 of [Agr05]).

Theorem 5.2 *Suppose there is a polynomial-time algorithm $\mathcal{A}_2$ satisfying Hypothesis 2. Then for all but finitely many n there is an explicit multilinear polynomial (where by explicit we mean that the coefficients of the polynomial f_n are computable by a uniform algorithm in time exponential in n) $f(x_1, \cdots, x_n) \in \mathbb{F}[x_1, x_2, \cdots, x_n]$ (where $\mathbb{F}$ is either $\mathbb{Q}$ or a finite field) that is computable in $2^{n^{O(1)}}$ time (by a uniform algorithm) and does not have arithmetic circuits of size $2^{o(n)}$.*

Proof. We will pick an appropriate multilinear polynomial $f \in \mathbb{F}[x_1, x_2, \cdots, x_n]$ where:

$$f(x_1, x_2, \cdots, x_n) = \sum_{S \subseteq [n]} c_S \prod_{i \in S} x_i,$$

where the coefficients $c_S \in \mathbb{F}$ will be determined appropriately so that the polynomial f has the claimed property.

Suppose $\mathcal{A}_2$ runs in time m^c for constant $c > 0$, where m denotes the size bound of the boolean circuit C defining set system $\mathcal{F}_C$. Notice that the number t of weight functions output by $\mathcal{A}_2$ is bounded by m^c.

The total number of coefficients c_S of f is 2^n. For each weight function w_i let $(w_{i,1}, \cdots, w_{i,n})$ be the assignments to the variables x_i. For each weight function $w_i, 1 \leq i \leq t$ we write down the following equations

$$f(y^{w_{i,1}}, y^{w_{i,2}}, \cdots, y^{w_{i,n}}) = 0.$$

Since f is of degree at most n, and the weights $w_{i,j}$ are bounded by $2n$, $f(y^{w_{i,1}}, y^{w_{i,2}}, \cdots, y^{w_{i,n}})$ is a univariate polynomial of degree at most $2n^2$ in y. Thus, each of the above equations will give rise to at most $2n^2$ linear equations in the unknowns c_S.

In all, this will actually give us a system of at most $2n^2 m^c$ linear equations over $\mathbb{F}$ in the unknown scalars c_S. Since the total number of distinct monomials is 2^n, and 2^n asymptotically exceeds m^c for $m = 2^{o(n)}$, the system of linear equations has a *nontrivial* solution in the c_S provided $m = 2^{o(n)}$. Furthermore, a nontrivial solution for c_S can be computed using Gaussian elimination in time exponential in n.

We claim that f does not have commutative circuits of size $2^{o(n)}$ over $\mathbb{F}$. Assume to the contrary that $\hat{C}(x_1, \cdots, x_n)$ is a circuit for $f(x_1, \cdots, x_n)$ of size $2^{o(n)}$. By Lemma 2.6 notice that we can uniformly construct a boolean circuit C of size $m = 2^{o(n)}$ that will take as input a monomial $\prod_{i \in S} x_i$ (encoded as an n bit boolean string representing S as a subset of $[n]$) and test if it is nonzero in $\hat{C}$ and hence in $f(x_1, \cdots, x_n)$.

Assuming Hypothesis 2, let $w_1, \cdots, w_t$ be the weight functions output by $\mathcal{A}_2$ for input (m, n). By Hypothesis 2, for some weight function w_i there is a unique monomial $\prod_{j \in S} x_j$ such that $\sum_{j \in S} w_{i,j}$ takes the minimum value. Clearly, the commutative circuit $\hat{C}$ must be nonzero on substituting $y^{w_{i,j}}$ for x_j (the coefficient of $y^{\sum_{j \in S} w_{i,j}}$ will be nonzero). However, f evaluates to zero on the integer assignments prescribed by all the weight functions $w_1, \cdots, w_t$. This is a contradiction to the assumption and it completes the proof. ∎

Remark 5.3 *We note that Hypothesis 2 also implies the existence of an explicit polynomial in noncommuting variables that does not have noncommutative circuits of subexponential size (we can obtain it as an easy consequence of the above proof).*

We now show that under the derandomization Hypothesis 3 yields a different consequence (about the integer permanent rather than some explicit function).

Theorem 5.4 *If a subexponential-time algorithm $\mathcal{A}_3$ satisfying Hypothesis 3 exists then identity testing over $\mathbb{Z}$ is in* SUBEXP *which implies that either* NEXP $\not\subset$ P/poly *or the integer Permanent does not have polynomial size arithmetic circuits.*

Proof. Using Lemma 1.5 it is shown in [KS01, Theorem 5] that there is a randomized identity test for small degree polynomials in $\mathbb{Q}[x_1, \cdots, x_n]$, where the polynomial is given by an arithmetic circuit $\hat{C}$ of polynomially bounded degree d. The idea is to pick a random weight vector $w : [n] \rightarrow [2nd]$ and replace the indeterminate x_i by $y^{w(i)}$, where d is the total degree of the input polynomial. As the circuit $\hat{C}$ has small degree, after this univariate substitution the circuit can be evaluated in deterministic polynomial time to explicitly find the polynomial in y. By Lemma 1.5 it will be nonzero with probability $1/2$ if $\hat{C}$ computes a nonzero polynomial.

Coming to the proof of this theorem, if NEXP $\not\subset$ P/poly then we are done. So, suppose NEXP $\subset$ P/poly. Notice that given any monomial $x_1^{d_1} \cdots x_n^{d_n}$ of total degree bounded by d we can test if it is a nonzero monomial of $\hat{C}$ in exponential time (explicitly listing down the monomials of the polynomial computed by $\hat{C}$). Therefore, since NEXP $\subset$ P/poly there is a polynomial-size boolean circuit C that accepts the vector $(d_1, \cdots, d_n)$ iff $x_1^{d_1} \cdots x_n^{d_n}$ is a nonzero monomial in the given polynomial (as required for application of Hypothesis 3).

Now, we invoke the derandomization Hypothesis 3. We can apply the Klivans-Spielman polynomial identity test, explained above, to the arithmetic circuit $\hat{C}$ for each of the t weight vectors $w_1, \cdots, w_t$ generated by algorithm $\mathcal{A}_3$ to obtain a subexponential deterministic identity test for the circuit $\hat{C}$ by the properties of $\mathcal{A}_3$. Now, following the argument of Impagliazzo-Kabanets [KI03] it is easy to derive that the integer Permanent does not have polynomial size arithmetic circuits. ■

Remark 5.5 *Although the permanent is a multilinear polynomial, notice that Hypothesis 2 does not seem strong enough to prove the above theorem. The reason is that the arithmetic circuit for the permanent that is nondeterministically guessed may not be computing multilinear polynomial and hence the application of Lemma 2.6 is not possible. There does not appear any easy way of testing if the guessed circuit computes a multilinear polynomial.*

Remark 5.6 *We can formulate both Hypothesis 1 and Hypothesis 2 more generally by letting the running time of algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ be a function $t(m, n)$. We will then obtain suitably quantified circuit lower bound results as consequence.*

6 Discussion

An interesting open question is whether derandomizing similar restricted versions of the Valiant-Vazirani lemma also implies circuit lower bounds. We recall the Valiant-Vazirani lemma as stated in the original paper [VV86].

Lemma 6.1 *Let $S \subseteq \{0,1\}^t$. Suppose $w_i, 1 \leq i \leq t$ are picked uniformly at random from $\{0,1\}^t$. For each i, let $S_i = \{v \in S \mid v.w_j = 0, 1 \leq j \leq i\}$ and let $p_t(S)$ be the probability that $|S_i| = 1$ for some i. Then $p_t(S) \geq 1/4$.*

Analogous to our discussion in Section 1, here too we can consider the restricted version where we consider $S_C \subseteq \{0,1\}^n$ to be the set of n-bit vectors accepted by a boolean circuit C of size m. We can similarly formulate derandomization hypotheses similar to Hypotheses 1 and 2.

We do not know if there is another randomized polynomial identity test for noncommutative arithmetic circuits based on the Valiant-Vazirani lemma. The automata-theoretic technique of Section 3 does not appear to work. Specifically, given a matrix $h : \mathbb{F}_2^n \to \mathbb{F}_2^k$, there is no deterministic finite automaton of size $\text{poly}(n,k)$ that accepts $x \in \mathbb{F}_2^n$ if and only if $h(x) = 0$.

Acknowledgements. We are grateful to Manindra Agrawal for interesting discussions and his suggestion that Theorem 5.2 can be obtained from the stronger hypothesis. We also thank Srikanth Srinivasan for interesting discussions.

References

[Agr05] M. AGRAWAL. Proving Lower Bounds Via Pseudo-random Generators. *FSTTCS,* Springer LNCS 3821: 92-105, 2005.

[Agr07] M. AGRAWAL. Rings and Integer Lattices in Computer Science. *Barbados Workshop on Computational Complexity,* Lecture no 9, 2007.

[AR00] KLAUS REINHARDT AND ERIC ALLENDER. Making Nondeterminism Unambiguous. *SIAM J. Comput.* 29(4): 1118-1131 (2000).

[ARZ99] ERIC ALLENDER, KLAUS REINHARDT, AND SHIYU ZHOU. Isolation, matching and counting uniform and nonuniform upper bounds. *Journal of Computer and System Sciences,* 59(2):164–181, 1999.

[AMS08] V. ARVIND, P. MUKHOPADHYAY, S. SRINIVASAN. New results on Noncommutative and Commutative Polynomial Identity Testing. *In Proceedings of the 23rd IEEE Conference on Computational Complexity,* June 2008, to appear. Technical report version in *ECCC report TR08-025,* 2008.

[BW05] A. BOGDANOV AND H. WEE. More on Noncommutative Polynomial Identity Testing . *In Proc. of the 20th Annual Conference on Computational Complexity,* pp. 92-99, 2005.

[CS07] S. CHIEN AND A. SINCLAIR. Algebras with Polynomial Identities and Computing the Determinants. *SIAM J. of Comput.* 37(1): 252-266, 2007.

[HU78] J.E. HOPCROFT AND J.D. ULLMAN. Introduction to Automata Theory, Languages and Computation. *Addison-Wesley*, 1979.

[IKW02] R. IMPAGLIAZZO, V. KABANETS AND A. WIGDERSON. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences 65(4).,* pages 672-694, 2002.

[KI03] V. KABANETS AND R. IMPAGLIAZZO. Derandomization of polynomial identity tests means proving circuit lower bounds. *In Proc. of the thirty-fifth annual ACM Sym. on Theory of computing.,* pages 355-364, 2003.

[KS01] ADAM R. KLIVANS, DANIEL A. SPIELMAN. Randomness Efficient Identity Testing. *Proceedings of the 33rd Symposium on Theory of Computing (STOC),* 216-223, 2001.

[MVV87] K. MULMULEY, U. VAZIRANI AND V. VAZIRANI. Matching is as easy as matrix inversion. *In Proc. of the nineteenth annual ACM conference on Theory of Computing,* pages 345-354. ACM Press, 1987.

[N91] N. NISAN. Lower bounds for non-commutative computation. *In Proc. of the 23rd annual ACM Sym. on Theory of computing,* pages 410-418, 1991.

[NSV94] H. NARAYANAN, HUZUR SARAN, V. V. VAZIRANI. Randomized Parallel Algorithms for Matroid Union and Intersection, With Applications to Arboresences and Edge-Disjoint Spanning Trees. *SIAM J. Comput.* 23(2): 387-397 (1994).

[RS05] R. RAZ AND A. SHPILKA. Deterministic polynomial identity testing in non commutative models. *Computational Complexity.,* 14(1):1-19, 2005.

[Sch80] JACOB T. SCHWARTZ. Fast Probabilistic algorithm for verification of polynomial identities. *J. ACM.,* 27(4), pages 701-717, 1980.

[Str94] HOWARD STRAUBING. Finite automata, formal logic, and circuit complexity. *Progress in Theoretical Computer Science.* Birkhuser Boston Inc., Boston, MA, 1994.

[VV86] L. G. VALIANT AND V. V. VAZIRANI. NP is as Easy as Detecting Unique Solutions. *Theor. Comput. Sci.* 47(3): 85-93 (1986).

[Zip79] R. ZIPPEL. Probabilistic algorithms for sparse polynomials. *In Proc. of the Int. Sym. on Symbolic and Algebraic Computation.,* pages 216-226, 1979.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de/