Arithmetic Circuits and the Hadamard Product of Polynomials

V. Arvind, Pushkar S. Joglekar, Srikanth Srinivasan

Institute of Mathematical Sciences
C.I.T Campus,Chennai 600 113, India
{arvind,pushkar,srikanth}@imsc.res.in

ABSTRACT. Motivated by the Hadamard product of matrices we define the Hadamard product of multivariate polynomials and study its arithmetic circuit and branching program complexity. We also give applications and connections to polynomial identity testing. Our main results are the following.

- We show that noncommutative polynomial identity testing for algebraic branching programs over rationals is complete for the logspace counting class $C_{=}L$, and over fields of characteristic p the problem is in $\text{Mod}_pL/\text{poly}$.
- We show an exponential lower bound for expressing the Raz-Yehudayoff polynomial as the Hadamard product of two monotone multilinear polynomials. In contrast the Permanent can be expressed as the Hadamard product of two monotone multilinear formulas of quadratic size.

1 Introduction

In this paper we define the *Hadamard product* of two polynomials f and g in $\mathbb{F}\langle X\rangle$ and study its expressive power and applications to the complexity of arithmetic circuits and algebraic branching programs. We also apply it to give a fairly tight characterization of polynomial identity testing for algebraic branching programs over the field of rationals.

Suppose $X = \{x_1, x_2, \cdots, x_n\}$ is a set of n noncommuting variables. The free monoid X^* consists of all words over these variables. For a field $\mathbb{F}$ let $\mathbb{F}\langle x_1, x_2, \cdots, x_n\rangle$ denote the free noncommutative polynomial ring over $\mathbb{F}$ generated by the variables in X. Thus, the polynomials in this ring are $\mathbb{F}$-linear combinations of words over X. For a given polynomial $f \in \mathbb{F}\langle X\rangle$, let $\text{mon}(f) = \{m \in X^* \mid m \text{ is a nonzero monomial in } f\}$. If $X = \{x_1, x_2, \cdots, x_n\}$ is a set of n *commuting variables* then $\mathbb{F}[X]$ denotes the commutative polynomial ring with coefficients from $\mathbb{F}$.

Motivated by the well-known Hadamard product of matrices (see e.g. [6]) we define the Hadamard product of polynomials.

Foundations of Software Technology and Theoretical Computer Science (Kanpur) 2009.
Editors: Ravi Kannan and K. Narayan Kumar; pp 25–36
Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany.
Digital Object Identifier: 10.4230/LIPIcs.FSTTCS.2009.2304

DEFINITION 1. *Let $f, g \in \mathbb{F}\langle X \rangle$ where $X = \{x_1, x_2, \cdots, x_n\}$. The* Hadamard product *of f and g, denoted $f \circ g$, is the polynomial $f \circ g = \sum_m a_m b_m m$, where $f = \sum_m a_m m$ and $g = \sum_m b_m m$, where the sums index over monomials m.*

Complexity theory preliminaries We recall some definitions of logspace counting classes from [3]. Let L denote the class of languages accepted by deterministic logspace machines.

GapL is the class of functions $f : \Sigma^* \to \mathbb{Z}$, for which there is a logspace bounded NDTM M such that for each input $x \in \Sigma^*$, we have $f(x) = acc_M(x) - rej_M(x)$, where $acc_M(x)$ and $rej_M(x)$ are the number of accepting and rejecting paths of M on input x, respectively.

A language L is in $\mathrm{C_{=}L}$ if there exists a function $f \in$ GapL such that $x \in L$ if and only if $f(x) = 0$. For a prime p, a language L is in the complexity class $\mathrm{Mod}_p\mathrm{L}$ if there exists a function $f \in$ GapL such that $x \in L$ if and only if $f(x) = 0 (\mathrm{mod}\ p)$.

It is shown in [3] that checking if an integer matrix is singular is complete for $\mathrm{C_{=}L}$ with respect to logspace many-one reductions. The same problem is known to be complete for $\mathrm{Mod}_p\mathrm{L}$ over a field of characteristic p. It is useful to recall that both $\mathrm{C_{=}L}$ and $\mathrm{Mod}_p\mathrm{L}$ are contained in TC^1 (which, in turn, is contained in NC^2).

An *Algebraic Branching Program* (ABP) [13, 14] over a field $\mathbb{F}$ and variables $x_1, x_2, \cdots, x_n$ is a *layered* directed acyclic graph with one *source* vertex of indegree zero and one *sink* vertex of outdegree zero. Let the layers be numbered $0, 1, \cdots, d$. The source and sink are the unique layer 0 and layer d vertices, respectively. Edges only go from layer i to $i + 1$ for each i. Each edge in the ABP is labeled with a linear form over $\mathbb{F}$ in the input variables. Each source to sink path in the ABP computes the product of the linear forms labelling the edges on the path, and the sum of these polynomials over all source to sink paths is the polynomial computed by the ABP. The size of the ABP is the number of vertices.

Main results. We show that the *noncommutative* branching program complexity of the Hadamard product $f \circ g$ is upper bounded by the product of the branching program sizes for f and g.This upper bound is natural because we know from Nisan's seminal work [13] that the algebraic branching program (ABP) complexity $B(f)$ is well characterized by the ranks of its "communication" matrices $M_k(f)$, and the rank of Hadamard product $A \circ B$ of two matrices A and B is upper bounded by the product of their ranks. Our proof is constructive: we give a deterministic logspace algorithm for computing an ABP for $f \circ g$.

We then apply this result to polynomial identity testing. It is shown by Raz and Shpilka [14] that polynomial identity testing of noncommutative ABPs can be done in deterministic polynomial time. A simple divide and conquer algorithm can be easily designed to show that the problem is in deterministic NC^3. What then is the precise complexity of polynomial identity testing for noncommutative ABPs? For noncommutative ABPs over *rationals* we give a tight characterization by showing that the problem is $\mathrm{C_{=}L}$-complete. We prove this result using the result on Hadamard product of ABPs explained above.

For noncommutative ABPs over a finite field of characteristic p, we show that identity testing is in the nonuniform class $\mathrm{Mod}_p\mathrm{L}/\mathrm{poly}$ (more precisely, in randomized $\mathrm{Mod}_p\mathrm{L}$). Furthermore, the problem turns out to be hard (w.r.t. logspace many-one reductions) for both NL and $\mathrm{Mod}_p\mathrm{L}$. Hence, it is not likely to be easy to improve this upper bound unconditionally to $\mathrm{Mod}_p\mathrm{L}$ (it would imply that NL is contained in $\mathrm{Mod}_p\mathrm{L}$). However, under a hardness assumption we can apply standard arguments [4, 12] to derandomize this algorithm and put the problem in $\mathrm{Mod}_p\mathrm{L}$.

In Section 4 we consider the Hadamard product for commutative polynomials. We show an exponential lower bound for expressing the Raz-Yehudayoff polynomial [15] as the Hadamard product of two monotone multilinear polynomials. In contrast the Permanent can be expressed as the Hadamard product of two monotone multilinear formulas of quadratic size.

2 The Hadamard Product

Let $f, g \in \mathbb{F}\langle X\rangle$ where $X = \{x_1, x_2, \cdots, x_n\}$. Clearly, $\mathrm{mon}(f \circ g) = \mathrm{mon}(f) \cap \mathrm{mon}(g)$. Thus, the Hadamard product can be seen as an algebraic version of the intersection of formal languages. Our definition of the Hadamard product of polynomials is actually motivated by the well-known Hadamard product $A \circ B$ of two $m \times n$ matrices A and B. We recall the following well-known bound for the rank of the Hadamard product.

PROPOSITION 2. *Let A and B be $m \times n$ matrices over a field $\mathbb{F}$. Then*

$$\mathrm{rank}(A \circ B) \leq \mathrm{rank}(A)\mathrm{rank}(B)$$

It is known from Nisan's work [13] that the ABP complexity $B(f)$ of a polynomial $f \in \mathbb{F}\langle X\rangle$ is closely connected with the ranks of the communication matrices $M_k(f)$, where $M_k(f)$ has its rows indexed by degree k monomials and columns by degree $d-k$ monomials and the $(m, m')^{th}$ entry of $M_k(f)$ is the coefficient of mm' in f. Nisan showed that $B(f) = \sum_k \mathrm{rank}(M_k(f))$. Indeed, Nisan's result and the above proposition easily imply the following bound on the ABP complexity of $f \circ g$.

LEMMA 3. *For $f, g \in \mathbb{F}\langle X\rangle$ we have $B(f \circ g) \leq B(f)B(g)$.*

Proof. By Nisan's result $B(f \circ g) = \sum_k \mathrm{rank}(M_k(f \circ g))$. The above proposition implies

$$\sum_k \mathrm{rank}(M_k(f \circ g)) \leq \sum_k \mathrm{rank}(M_k(f))\mathrm{rank}(M_k(g)) \leq (\sum_k \mathrm{rank}(M_k(f))(\sum_k \mathrm{rank}(M_k(g))),$$

and the claim follows. ■

We now show an algorithmic version of this upper bound.

THEOREM 4. *Let P and Q be two given ABP's computing polynomials f and g in $\mathbb{F}\langle x_1, x_2, \ldots, x_n\rangle$, respectively. Then there is a deterministic polynomial-time algorithm that will output an ABP R for the polynomial $f \circ g$ such that the size of R is a constant multiple of the product of the sizes of P and Q. (Indeed, R can be computed in deterministic logspace.)*

Proof. Let f_i and g_i denote the i^{th} homogeneous parts of f and g respectively. Then $f = \sum_{i=0}^{d} f_i$ and $g = \sum_{i=0}^{d} g_i$. Since the Hadamard product is distributive over addition and $f_i \circ g_j = 0$ for $i \neq j$ we have $f \circ g = \sum_{i=0}^{d} f_i \circ g_i$. Thus, we can assume that both P and Q are homogeneous ABP's of degree d. Otherwise, we can easily construct an ABP to compute $f_i \circ g_i$ separately for each i and put them together. Note that we can easily compute ABPs for f_i and g_i in logspace given as input the ABPs for f and g.

By allowing parallel edges between nodes of P and Q we can assume that the labels associated with each edge in an ABP is either 0 or αx_i for some variable x_i and scalar $\alpha \in \mathbb{F}$. Let s_1 and s_2 bound the number of nodes in each layer of P and Q respectively. Denote the j^{th} node in layer i by $\langle i, j\rangle$ for ABPs P and Q. Now we describe the construction of the ABP R for computing the polynomial $f \circ g$. Each layer i, $1 \leq i \leq d$ of R will have $s_1 \cdot s_2$ nodes, with node labeled $\langle i, a, b\rangle$ corresponding to the node $\langle i, a\rangle$ of P and the node $\langle i, b\rangle$ of Q. We can assume there is an edge from every node in layer i to every node in layer $i + 1$ for both ABPs. For, if there is no such edge we can always include it with label 0.

In the new ABP R we put an edge from $\langle i, a, b\rangle$ to $\langle i + 1, c, e\rangle$ with label $\alpha\beta x_t$ if and only if there is an edge from node $\langle i, a\rangle$ to $\langle i + 1, c\rangle$ with label αx_t in P and an edge from $\langle i, b\rangle$ to $\langle i + 1, e\rangle$ with label βx_t in ABP Q. Let $\langle 0, a, b\rangle$ and $\langle d, c, e\rangle$ denote the source and the sink nodes of ABP R, where $\langle 0, a\rangle, \langle 0, b\rangle$ are the source nodes of P and Q, and $\langle d, c\rangle, \langle d, e\rangle$ are the sink nodes of P and Q respectively. It is easy to see that ABP R can be computed in deterministic logspace. Let $h_{\langle i,a,b\rangle}$ denote the polynomial computed at node $\langle i, a, b\rangle$ of ABP R. Similarly, let $f_{\langle i,a\rangle}$ and $g_{\langle i,b\rangle}$ denote the polynomials computed at node $\langle i, a\rangle$ of P and node $\langle i, b\rangle$ of Q. We can easily check that $h_{\langle i,a,b\rangle} = f_{\langle i,a\rangle} \circ g_{\langle i,b\rangle}$ by an induction argument on the number of layers in the ABPs. It follows from this inductive argument that the ABP R computes the polynomial $f \circ g$ at its sink node. The bound on the size of R also follows easily. ∎

Applying the above theorem we can give a *tight* complexity theoretic upper bound for identity testing of noncommutative ABPs over rationals.

THEOREM 5. *The problem of polynomial identity testing for noncommutative algebraic branching programs over $\mathbb{Q}$ is in NC^2. More precisely, it complete for the logspace counting class $\mathrm{C_{=}L}$ under logspace reductions.*

Proof. Let P be the given ABP computing $f \in \mathbb{Q}\langle X\rangle$. We apply the construction of Theorem 4 to compute a polynomial sized ABP R for the Hadamard product $f \circ f$ (i.e. of f with itself). Notice that $f \circ f$ is nonzero iff f is nonzero. Now, we crucially use the fact that $f \circ f$ is a polynomial whose nonzero coefficients are all *positive*. Hence, $f \circ f$ is nonzero iff it evaluates to nonzero on the all 1′s input. The problem thus boils down to checking if R evaluates to nonzero on the all 1′s input.

By Theorem 4, the ABP R for polynomial $f \circ f$ is computable in deterministic logspace, given as input an ABP for f. Furthermore, evaluating the ABP R on the all 1′s input can be easily converted to iterated integer matrix multiplication (one matrix for each layer of the ABP), and checking if R evaluates to nonzero can be done by checking if a specific entry of the product matrix is nonzero. It is well known that checking if a specific entry of an iterated integer matrix product is zero is in the logspace counting class $\mathrm{C_{=}L}$ (e.g. see [3, 1]). However, $\mathrm{C_{=}L}$ is contained in NC^2, in fact in TC^1.

We now argue the hardness of this problem for $\mathrm{C_{=}L}$. The problem of checking if an integer matrix A is singular is well known to be complete for $\mathrm{C_{=}L}$ under deterministic logspace reductions. The standard GapL algorithm for computing $\det(A)$ [16] can be converted to an ABP P_A which will compute $\det(A)$.* Hence the ABP P_A computes the identically zero polynomial iff A is singular. Putting it all together, it follows that identity testing of noncommutative ABPs over rationals is complete for the class $\mathrm{C_{=}L}$. ■

An iterative matrix product problem Suppose B is a noncommutative ABP computing a homogeneous polynomial in $\mathbb{F}\langle X\rangle$ of degree d, where each edge of the ABP is labeled by a homogeneous linear form in variables from X.

Let n_ℓ denote the number of nodes of B in layer ℓ, $0 \le \ell \le d$. For each x_i and layer ℓ, we associate an $n_\ell \times n_{\ell+1}$ matrix $A_{i,\ell}$ where the $(k,j)^{th}$ entry of matrix $A_{i,\ell}$ is the coefficient of x_i in the linear form associated with the (v_k, u_j) edge in the ABP B. Here v_k is the k^{th} node in layer ℓ and u_j the j^{th} node in the layer $\ell+1$. The following claim is easy to see and relates these matrices to the ABP B.

CLAIM 6. *The coefficient of any degree d monomial $x_{i_1}x_{i_2}\cdots x_{i_d}$ in the polynomial computed by the ABP B is the matrix product $A_{i_1,0}A_{i_2,1}\cdots A_{i_d,d-1}$ (which is a scalar since $A_{i_1,0}$ is a row and $A_{i_d,d-1}$ is a column).*

Let i and j be any two nodes in the ABP B. We denote by $B(i,j)$ the algebraic branching program obtained from the ABP B by designating node i in B as the source node and node j as the sink node. Clearly, $B(i,j)$ computes a homogeneous polynomial of degree $b-a$ if i appears in layer a and j in layer b.

For layers a, b, $0 \le a < b \le d$ let $t = b - a$ and $P(a,b) = \{A_{s_1,a}A_{s_2,a+1}\dots A_{s_t,b-1} \mid 1 \le s_j \le n, \text{ for } 1 \le j \le t\}$. $P(a,b)$ consists of $n_a \times n_b$ matrices. Thus the dimension of the linear space spanned by $P(a,b)$ is bounded by $n_a n_b$. It follows from Claim 6 that the linear span of $P(a,b)$ is the zero space iff the polynomial computed by ABP $B(i,j)$ is identically zero for every $1 \le i \le n_a$ and $1 \le j \le n_b$.

*Notice that the polynomial computed by the ABP P_A is a constant since P_A has only constants and no variables.

Thus, it suffices to compute a basis for the space spanned by matrices in $P(0,d)$ to check whether the polynomial computed by B is identically zero. We can easily give a deterministic NC^3 algorithm for this problem over any field $\mathbb{F}$: First recursively compute bases M_1 and M_2 for the space spanned by matrices in $P(0,d/2)$ and $P(d/2+1,d)$ respectively. From bases M_1 and M_2 we can compute in deterministic NC^2 a basis M for space spanned by matrices in $P(0,d)$ as follows. We compute the set S of pairwise products of matrices in M_1 and M_2 and then we can compute a maximal linearly independent subset of S in NC^2 (see e.g. [1]). This gives an easy NC^3 algorithm to compute a basis for the linear span of $P(0,d)$. This proves the following.

PROPOSITION 7. *The problem of polynomial identity testing for noncommutative algebraic branching programs over any field (in particular, finite fields $\mathbb{F}$) is in deterministic* NC^3.

Can we give a tight complexity characterization for identity testing of noncommutative ABPs over finite fields? We show that the problem is in nonuniform $\mathrm{Mod}_p\mathrm{L}$ and is hard for $\mathrm{Mod}_p\mathrm{L}$ under logspace reductions. Furthermore, the problem is hard for NL. Hence, it appears difficult to improve the upper bound to uniform $\mathrm{Mod}_p\mathrm{L}$ (as NL is not known to be contained in uniform $\mathrm{Mod}_p\mathrm{L}$).

THEOREM 8. *The problem of polynomial identity testing for noncommutative algebraic branching programs over a finite field $\mathbb{F}$ of characteristic p is in* $\mathrm{Mod}_p\mathrm{L}/\mathrm{poly}$.

Proof. Consider a new ABP B' in which we replace the variables x_i, $1 \le i \le n$ appearing in the linear form associated with an edge from some node in layer l to a node in layer $l+1$ of ABP B by new variable $x_{i,l}$, for layers $l = 0,1,\ldots,d-1$. Let $g \in \mathbb{F}[X]$ denotes the polynomial computed by ABP B' in *commuting* variables $x_{i,l}, 1 \le i \le n, 1 \le l < d$. It is easy to see that the commutative polynomial $g \in \mathbb{F}[X]$ is identically zero iff the noncommutative polynomial $f \in \mathbb{F}\langle X\rangle$ computed by ABP B is identically zero. Now, we can apply the standard Schwartz-Zippel lemma to check if g is identically zero by substituting random values for the variables $x_{i,l}$ from $\mathbb{F}$ (or a suitable finite extension of $\mathbb{F}$). After substitution of field elements, we are left with an iterated matrix product over a field of characteristic p which can be done in $\mathrm{Mod}_p\mathrm{L}$. This gives us a randomized $\mathrm{Mod}_p\mathrm{L}$ algorithm. By standard amplification it follows that the problem is in $\mathrm{Mod}_p\mathrm{L}/\mathrm{poly}$. ∎

Next we show that identity testing noncommutative ABPs over any field is hard for NL by a reduction from directed graph reachability. Let (G,s,t) be a reachability instance. Without loss of generality, we assume that G is a layered directed acyclic graph. The graph G defines an ABP with source s and sink t as follows: label each edge e in G with a *distinct* variable x_e and for each absent edge put the label 0. The polynomial computed by the ABP is nonzero if and only if there is a directed s-t path in G.

THEOREM 9. *The problem of polynomial identity testing for noncommutative algebraic branching programs over any field is hard for* NL.

3 Hadamard product of noncommutative circuits

Analogous to Theorem 4 we show that $f \circ g$ has small circuits if f has a small circuit and g has a small ABP. For the proof refer to the full version of the paper [2].

THEOREM 10. *Let $f, h \in \mathbb{F}\langle x_1, x_2, \cdots, x_n\rangle$ be given by a degree d circuit C and a degree d ABP P respectively, where $d = O(n^{O(1)})$. Then we can compute in polynomial time a circuit C' that computes $f \circ h$ where the size of C' is polynomially bounded in the sizes of C and P.*

On the other hand, suppose f and g individually have small circuit complexity. Does $f \circ g$ have small circuit complexity? Can we compute such a circuit for $f \circ g$ from circuits for f and g? We first consider these questions for monotone circuits. It is useful to understand the connection between monotone noncommutative circuits and context-free grammars. We recall the following definition.

DEFINITION 11. *We call a context-free grammar $G = (V, T, P, S)$ an* acyclic CFG *if for any nonterminal $A \in V$ there does not exist any derivation of the form $A \Rightarrow^* uAw$, and for each production $A \Rightarrow \beta$ we have $|\beta| \leq 2$.*

The size *size*(G) of an acyclic CFG $G = (V, T, P, S)$ is defined as $|V| + |T| + \text{size}(P)$, where V, T, and P are the sets of variables, terminals, and production rules. We note the following easy proposition that relates acyclic CFGs to monotone noncommutative circuits over X.

PROPOSITION 12. *Let C be a monotone circuit of size s computing a polynomial $f \in \mathbb{Q}\langle X\rangle$. Then there is an acyclic CFG G for $mon(f)$ with $\text{size}(G) = O(s)$. Conversely, if G is an acyclic CFG of size s computing some finite set $L \subset X^*$ of monomials over X, there exists a monotone circuit of size $O(s)$ that computes a polynomial $\sum_{m \in L} a_m m \in \mathbb{Q}\langle X\rangle$, where the positive integer a_m is the number of derivation trees for m in the grammar G.*

THEOREM 13. *There are monotone circuits C and C' computing polynomials f and g in $\mathbb{Q}\langle X\rangle$ respectively, such that the polynomial $f \circ g$ requires monotone circuits of size exponential in $|X|$, $\text{size}(C)$, and $\text{size}(C')$.*

Proof. Let $X = \{x_1, \cdots, x_n\}$. Define the finite language $L_1 = \{zww^r \mid z, w \in X^*, |z| = |w| = n\}$ and the corresponding polynomial $f = \sum_{m_\alpha \in L_1} m_\alpha$. Similarly let $L_2 = \{ww^rz \mid z, w \in X^*, |z| = |w| = n\}$, and the corresponding polynomial $g = \sum_{m_\alpha \in L_2} m_\alpha$. It is easy to see that there are poly(n) size *unambiguous* acyclic CFGs for L_1 and L_2. Hence, by Proposition 12 there are monotone circuits C_1 and C_2 of size poly(n) such that C_1 computes polynomial f and C_2 computes polynomial g. We first show that the finite language $L_1 \cap L_2$ cannot be generated by any acyclic CFG of size $2^{o(n \lg n)}$. Assume to the contrary that there is an acyclic CFG $G = (V, T, P, S)$ for $L_1 \cap L_2$ of size $2^{o(n \lg n)}$. Notice that $L_1 \cap L_2 = \{t \mid t = ww^rw, w \in X^*, |w| = n\}$.

Consider any derivation tree T' for a word $ww^rw = w_1w_2\dots w_nw_nw_{n-1}\dots w_2w_1w_1\dots w_n$. Starting from the root of the binary tree T', we traverse down the tree always picking the child with larger yield. Clearly, there must be a nonterminal $A \in V$ in this path of the derivation tree such that $A \Rightarrow^* u$, $u \in X^*$ and $n \leq |u| < 2n$. Crucially, note that any word that A generates must have same length since every word generated by the grammar G is in $L_1 \cap L_2$ and hence of length $3n$. Let $ww^rw = s_1us_2$ where $|s_1| = k$. As $|u| < 2n$, the string s_1s_2 completely determines the string ww^rw. Hence, the nonterminal A can derive at most one string u. Furthermore, this string u can occur in at most $2n$ positions in a string of length $3n$. Notice that for each position in which u can occur it completely determines a string of the form ww^rw. Therefore, A can participate in the derivation of at most $2n$ strings from $L_1 \cap L_2$. Since there are n^n distinct words in $L_1 \cap L_2$, it follows that there must be at least $\frac{n^n}{2n}$ *distinct* nonterminals in V. This contradicts the size assumption of G.

Since $L_1 \cap L_2$ cannot be generated by any acyclic CFG of size $2^{o(n\log n)}$, it follows from Lemma 12 that the polynomial $f \circ g$ can not be computed by any monotone circuit of $2^{o(n\log n)}$ size. ∎

Theorem 13 shows that the Hadamard product of monotone circuits is more expressive than monotone circuits. It raises the question whether the permanent polynomial can be expressed as the Hadamard product of polynomial-size (or even subexponential size) monotone circuits. We note here that the permanent *can* be easily expressed as the Hadamard product of $O(n^3)$ many monotone circuits (in fact, monotone ABPs).

THEOREM 14. *Suppose there is a deterministic subexponential-time algorithm that takes two circuits as input, computing polynomials f and g in $\mathbb{Q}\langle x_1, \cdots, x_n\rangle$, and outputs a circuit for $f \circ g$. Then either* NEXP *is not in* P/poly *or the Permanent does not have polynomial size noncommutative circuits.*

Proof. Let C_1 be a circuit computing some polynomial $h \in \mathbb{Q}\langle x_1, \dots, x_n\rangle$. By assumption, we can compute a circuit C_2 for $h \circ h$ in subexponential time. Therefore, h is identically zero iff $h \circ h$ is identically zero iff C_2 evaluates to 0 on the all 1's input. We can easily check if C_2 evaluates to 0 on all 1's input by substitution and evaluation. This gives a deterministic subexponential time algorithm for testing if h is identically zero. By the noncommutative analogue of [11], shown in [5], it follows that either NEXP $\not\subset$ P/poly or the Permanent does not have polynomial size noncommutative circuits. ∎

Next, We show that the identity testing problem: given $f, g \in \mathbb{F}\langle X\rangle$ by circuits test if $f \circ g$ is identically zero is coNP hard via a reduction from *bounded* Post Correspondence Problem. For the proof refer to the full version of the paper [2].

THEOREM 15. *Given two* monotone polynomial-degree *circuits C and C' computing polynomial $f, g \in \mathbb{Q}\langle X\rangle$ it is* coNP*-complete to check if $f \circ g$ is identically zero.*

4 Hadamard product of monotone multilinear circuits

In this section we study the Hadamard product of *commutative* polynomials (defined as in the noncommutative case). First we introduce some notation useful for this section. Given a polynomial $f \in \mathbb{F}[X]$, and a monomial m over the variables X, we define $f(m)$ to be the coefficient of the monomial m in the polynomial f.[†] Recall the Definition 1 of the Hadamard product of two polynomials in $\mathbb{F}\langle X\rangle$. We define the Hadamard product in the commutative case analogously. Thus, for polynomials $f, g \in \mathbb{F}[X]$ we have $F(m) = f(m)g(m)$ for any monomial m, where $F = f \circ g$.

In this section our interest is the expressive power of the Hadamard product. Can we express a hard explicit polynomial (like the Permanent) as the Hadamard product $f \circ g$ where f and g have small arithmetic circuits? It turns out that we easily can.

PROPOSITION 16. *There are multilinear polynomials $f, g \in \mathbb{F}[x_{11}, x_{12}, \cdots, x_{nn}]$ such that both f and g have arithmetic formulas of size $O(n^2)$ and $f \circ g$ is the Permanent polynomial. Furthermore, for $\mathbb{F} = \mathbb{Q}$ these formulas for f and g are* monotone.

Proof. Define the polynomials f and g on the variables $\{x_{ij} \mid 1 \le i,j \le n\}$ as follows $f = \prod_{i=1}^{n}(\sum_{j=1}^{n} x_{ij})$ and $g = \prod_{j=1}^{n}(\sum_{i=1}^{n} x_{ij})$. Clearly, their Hadamard product is $Perm(x_{11}, \cdots, x_{nn})$. The formulas for f and g over rationals are monotone. ∎

Nevertheless, we will define an explicit monotone multilinear polynomial that cannot be written as the Hadamard product of multilinear polynomials computed by subexponential sized monotone arithmetic circuits. Our construction adapts a result of Raz and Yehudayoff [15] describing an explicit monotone polynomial that has no monotone arithmetic circuits of size $2^{\epsilon n}$, for some constant $\epsilon > 0$. Our proof closely follows the arguments in [15]. Due to lack of space, we provide only proof sketches for several technical statements.

DEFINITION 17. *For $\epsilon > 0$, a multilinear polynomial $f \in \mathbb{C}[x_1, \ldots, x_n]$ is an ϵ-*product polynomial *if there are disjoint sets $A, B \subseteq X = \{x_1, \ldots, x_n\}$ such that $|A| \ge \epsilon n$ and $|B| \ge \epsilon n$ and $f = gh$ where $g \in \mathbb{C}[A]$ and $h \in \mathbb{C}[B]$.*

In the sequel, we often identify a multilinear polynomial f in $\mathbb{C}[X]$ with its coefficients vector (indexed by monomials in the natural lexicographic order). The complex inner product of vectors $w, w' \in \mathbb{C}^k$ is $\langle w, w'\rangle = \sum_i w_i \overline{w'_i}$. Let $\mathcal{M}(X)$ denote the set of multilinear monomials over the variables in X.

DEFINITION 18. *The* correlation *of multilinear polynomials f and g in $\mathbb{C}[X]$ is defined as* Corr $(f, g) = |\sum_{m \in \mathcal{M}(X)} f(m)\overline{g(m)}|$. *Notice that* Corr (f, f) *is the ℓ_2-norm $\|f\|$ of f.*

[†]There should be no confusion with evaluating the multivariate polynomial f at a point $(a_1, \cdots, a_n)$ as we denote that by $f(a_1, a_2, \cdots, a_n)$.

The explicit polynomial from [15]

The explicit polynomial F we define is essentially the same as the one in [15] (the difference is in the constants). Let $s \in \mathbb{N}$ be a constant, to be chosen later and $t = 40s$. Let $n = tp = 40sp$, for a prime p, and $X = \{x_1, \ldots, x_n\}$. Partition X into t many sets of variables $X(1), \ldots, X(t)$ with p variables each, where $X(i) = \{x_{(i-1)p+j} \mid j \in [p]\}$.

In poly(n) time we can construct the field $\mathbb{F} = \mathbb{F}_{2^p}$ which is in bijective correspondence with $\{0,1\}^p$. We can assume that $0 \in \mathbb{F}$ is associated with the all 0s vector 0^p. Fix a nontrivial additive character ψ of $\mathbb{F}$. Since char$(\mathbb{F}) = 2$ we have $\psi(x) = \pm 1$ for all $x \in \mathbb{F}$. Each monomial $m \in \mathcal{M}(X)$ defines a subset A_m of X and is thus represented by its characteristic vector $w \in \{0,1\}^n$. Split w into t blocks $w_1, \ldots, w_t$ of size p each (w_i is the characteristic vector of $A_m \cap X(i)$), and consider the p field elements $y_1(m), y_2(m), \ldots, y_t(m) \in \mathbb{F}$ associated with these strings. The bijection between $\mathbb{F}$ and $\{0,1\}^p$ implies for any $m \in \mathcal{M}(X)$ that $y_i(m) = 0$ iff no variable $x \in X(i)$ appears in m.

Let us now define the polynomial F. Given a monomial $m \in \mathcal{M}(X)$, we define $F(m)$ to be $\psi(\prod_{i=1}^{t} y_i(m))$. We define a polynomial $f \in \mathbb{F}[X]$ to be *explicit* if the coefficient $f(m)$ of any monomial m can be computed in time polynomial in n. Note that the polynomial F is explicit.

We now state our main correlation result using which we will obtain the lower bound against the Hadamard product of monotone multilinear polynomials in $\mathbb{C}[x_1, \ldots, x_n]$. A proof sketch is given in the full version of the paper [2].

LEMMA 19. *Let $F \in \mathbb{C}[x_1, \ldots, x_n]$ be the explicit multilinear polynomial defined above and $f_1, f_2 \in \mathbb{C}[x_1, \ldots, x_n]$ be any $1/3$-product polynomials. Then*

1. $\sum_{m \in \mathcal{M}(\{x_1,\ldots,x_n\})} F(m) \geq 0$.
2. $\text{Corr}\,(F, f_1 \circ f_2) \leq 2^{-\alpha n} \|F\| \|f_1 \circ f_2\|$, *for a constant $\alpha > 0$ that is independent of f_1 and f_2.*

Using the above lemma bounding the correlation between F and the Hadamard product of 1/3-product polynomials, we will prove the main lower bound. We first recall a crucial lemma of Raz and Yehudayoff [15].

LEMMA 20. *For $n \geq 3$, let $F \in \mathbb{C}[x_1, \ldots, x_n]$ be a monotone multilinear polynomial computed by a monotone circuit of size s (i.e. the circuit has at most s edges). Then, there are $s+1$ monotone $1/3$-product polynomials $f_1, f_2, \ldots, f_{s+1}$ such that $F = \sum_{i=1}^{s+1} f_i$.*

THEOREM 21. *For large enough $n \in \mathbb{N}$, there is an explicit monotone multilinear polynomial $F' \in \mathbb{Q}[x_1, \ldots, x_n]$ that cannot be written as a Hadamard product of two monotone multilinear polynomials $f_1, f_2 \in \mathbb{R}[x_1, \ldots, x_n]$ such that each f_i is computed by monotone circuits of size less than $2^{\alpha n}$, where $\alpha > 0$ is an absolute constant.*

Proof. By the density of primes it suffices to consider n of the form tp, for prime p, where t is the constant in the definition of F. Let X denote the set of variables $\{x_1, \ldots, x_n\}$, and let F be the explicit polynomial mentioned in Lemma 19 above. For any monomial $m \in \mathcal{M}(X)$, let $F'(m) = (F(m)+1)/2$. Clearly, the coefficients of F' all lie in $\{0,1\}$. Consider the correlation between F and F', $\text{Corr}\,(F, F') = \left|\sum_{m:F(m)=1} 1\right| \geq 2^{n-1}$, where the inequality follows from the point 1 of Lemma 19.

Let us assume that F' can be written as $f_1 \circ f_2$, where f_1 and f_2 are multilinear polynomials computed by monotone arithmetic circuits of size at most s. We assume $n \geq 3$, so that Lemma 20 is applicable. By Lemma 20, there exist monotone 1/3-product polynomials $f_{1,1}, \ldots, f_{1,s+1}, f_{2,1}, \ldots, f_{2,s+1}$ such that $f_i = \sum_{j=1}^{s+1} f_{i,j}$, for each $i \in \{1,2\}$. Thus, we have,

$$F' = \left(\sum_{j=1}^{s+1} f_{1,j}\right) \circ \left(\sum_{k=1}^{s+1} f_{2,k}\right) = \sum_{1\leq j,k \leq s+1} f_{1,j} \circ f_{2,k}$$

Taking correlation with F on both sides, we see that,

$$2^{n-1} \leq \sum_{1\leq j,k\leq s+1} \text{Corr}\left(F, f_{1,j} \circ f_{2,k}\right) \leq \sum_{1\leq j,k\leq s+1} 2^{-\beta n}\|F\|\|f_{1,j} \circ f_{2,k}\|,$$

by applying triangle inequality and then part 2 of Lemma 19, where $\beta > 0$ is some constant.

Since, $f_{1,j} \circ f_{2,k}$'s are monotone polynomials adding up to F', it follows that for any monomial $m \in \mathcal{M}(X)$ its coefficient in $f_{1,j} \circ f_{2,k}$ is at most 1. Hence, $\|f_{1,j} \circ f_{2,k}\| \leq \|F\|$ and we have

$$2^{n-1} \quad \leq \textstyle\sum_{1\leq j,k\leq s+1} 2^{-\beta n}\|F\|^2 = (s+1)^2 2^{n-\beta n}$$

Consequently, we have $s \geq 2^{\beta n/4}$, for large enough n. ■

References

[1] E. Allender, R. Beals, and M. Ogihara, The complexity of matrix rank and feasible systems of linear equations, *Computational Complexity* , 8(2):99-126, 1999.

[2] V. Arvind, P. S. Joglekar, S. Srinivasan Arithmetic Circuits and the Hadamard Product of Polynomials *CoRR* abs/0907.4006: (2009)

[3] E. Allender, M. Ogihara, Relationships among PL, #L and the determinant. *RAIRO - Theoretical Informatics and Applications*, 30:1–21, 1996.

[4] E. Allender, K. Reinhardt, S. Zhou, Isolation, matching and counting uniform and nonuniform upper bounds. *Journal of Computer and System Sciences*, 59(2):164–181, 1999.

[5] V. Arvind, P. Mukhopadhyay, S. Srinivasan New results on Noncommutative Polynomial Identity Testing*In Proc. of Annual IEEE Conference on Computational Complexity,*268-279,2008.

[6] R. Bhatia, Matrix Analysis, Springer-Verlag Publishing Company, 1997.

[7] J. Bourgain: "On the Construction of Affine Extractors", Geometric & Functional Analysis GAFA, Vol. 17, No. 1. (April 2007), pp. 33-57.

[8] J. Bourgain, A. Glibichuk, S. Konyagin: "Estimate for the number of sums and products and for exponential sums in fields of prime order", J. London Math. Soc. 73 (2006), pp. 380-398.

[9] M. R. Garey, D. S. Johnson Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman.*p. 228. ISBN 0-7167-1045-5*, 1979.

[10] J. E. Hopcroft, R. Motawani, J. D. Ullman, Introduction to Automata Theory Languages and Computation,*Second Edition*, Pearson Education Publishing Company.

[11] V. Kabanets, R. Impagliazzo, Derandomization of polynomial identity test means proving circuit lower bounds, *In Proc. of 35th ACM Sym. on Theory of Computing*,355-364,2003.

[12] A. Klivans, D. van Melkebeek, Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM Journal on Computing*, 31(5):1501–1526, 2002.

[13] N. Nisan, Lower bounds for noncommutative computation *In Proc. of 23rd ACM Sym. on Theory of Computing*, 410-418, 1991.

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

[15] Ran Raz, Amir Yehudayoff, "Multilinear Formulas, Maximal-Partition Discrepancy and Mixed-Sources Extractors." FOCS 2008: 273-282.

[16] S. Toda, Counting Problems Computationally Equivalant to the Determinant, manuscript.