Electronic Colloquium on Computational Complexity, Report No. 73 (2009)

On Lower Bounds for Constant Width Arithmetic Circuits

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. The motivation for this paper is to study the complexity of constant-width arithmetic circuits. Our main results are the following.

1. For every $k > 1$, we provide an explicit polynomial that can be computed by a linear-sized monotone circuit of width $2k$ but has no subexponential-sized monotone circuit of width k. It follows, from the definition of the polynomial, that the constant-width and the constant-depth hierarchies of monotone arithmetic circuits are infinite, both in the commutative and the noncommutative settings.
2. We prove hardness-randomness tradeoffs for identity testing constant-width commutative circuits analogous to [KI03,DSY08].

1 Introduction

Using a rank argument, Nisan, in a seminal paper [N91], showed exponential size lower bounds for noncommutative formulas (and noncommutative algebraic branching programs) that compute the noncommutative permanent or determinant polynomials in the ring $\mathbb{F}\langle X\rangle$, where $X = \{x_1, \cdots, x_n\}$ are noncommuting variables.

By Ben-Or and Cleve's result [BC92], we know that bounded-width arithmetic circuits (both commutative and noncommutative) are at least as powerful as formulas (indeed width three is sufficient). Can we extend Nisan's lower bound arguments to prove size lower bounds for *noncommutative* bounded-width circuits? Motivated by this question we make some simple motivating observations in this section. We first recall some basic definitions.

Definition 1. *[N91,RS05] An* Algebraic Branching Program *(ABP) 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 $0, 1, \cdots, d$. Edges only go from layer i to $i+1$ for each i. The source and sink are the unique layer 0 and layer d vertices, respectively. Each edge in the ABP is labeled with a linear form over $\mathbb{F}$ in the input variables. The size of the ABP is the number of vertices. Each source to sink path in the ABP computes the product of the linear forms labeling 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 scalars in an ABP can come from any field $\mathbb{F}$. If the input variables $X = \{x_1, x_2, \cdots, x_n\}$ are noncommuting then the ABP (or circuit) computes a polynomial in the free noncommutative ring $\mathbb{F}\langle X\rangle$. If the variables are commuting then the polynomial computed is in the ring $\mathbb{F}[X]$.

Definition 2. *An arithmetic circuit over $\mathbb{F}$ and variables $x_1, x_2, \cdots, x_n$ is a directed acyclic graph with each node of indegree zero labeled by a variable or a scalar constant. Each internal node g of the DAG is labeled by $+$ or $\times$ (i.e. it is a plus or multiply gate) and is of indegree two. A node of the DAG is designated as the output gate. Each internal gate of the arithmetic*

ISSN 1433-8092

circuit computes a polynomial (by adding or multiplying its input polynomials). The polynomial computed at the output gate is the polynomial computed by the circuit. The circuit is said to be layered *if its vertices are partitioned into vertex sets $V_1 \cup V_2 \cup \ldots \cup V_t$ such that V_1 consists only of leaves, and given any internal node g in V_i for $i > 1$, the children of g are either nodes from V_1 (consisting of constants or variables) or nodes from the set V_{i-1}. The size of a circuit is the number of nodes in it, and the width of a layered circuit is* $\max_{i>1} |V_i|$. *An arithmetic circuit over the field $\mathbb{R}$ is* monotone *if all the scalars used are nonnegative. Finally, a layered arithmetic circuit is* staggered *if, in each layer i with $i > 1$, every node except possibly one is a product gate of the form $g = u \times 1$, for some gate u from the previous layer.*

Note that the notion of bounded (i.e, constant) width staggered circuits of width w is identical to the notion of a straight-line program with w registers. The following lemma shows that staggered circuits of width w are comparable in power to width $w-1$ (not necessarily staggered) arithmetic circuits. It holds in the commutative and the noncommutative settings. We postpone the proof of the lemma to the Appendix.

Lemma 1. *Given any layered arithmetic circuit C of width w and size s computing a polynomial p, there is a staggered arithmetic circuit C' of width at most $w+1$ and size $O(ws)$ computing the same polynomial.*

A seminal result in the area of bounded width circuits is due to Ben-Or and Cleve [BC92] where they show that size s arithmetic formulas computing a polynomial in $\mathbb{F}[X]$ (or in $\mathbb{F}\langle X \rangle$ in the noncommutative case) can be evaluated by staggered arithmetic circuits of width three and size $O(s^2 n)$. Bounded width circuits have also been studied under various restrictions in [LMR07,MR08,JR09]. However, they have not considered the question of proving explicit lower bounds.

What is the power of arithmetic circuits of width 2? It is easy to see that the width-two circuit model is universal. We state this (folklore) observation.

Proposition 1. *Any polynomial of degree d with s monomials in $\mathbb{F}[x_1, x_2, \cdots, x_n]$ (or in $\mathbb{F}\langle x_1, \cdots, x_n \rangle$) can be computed by a width two arithmetic circuit of size $O(d \cdot s)$. Furthermore, any monotone polynomial (i.e, with non-negative real coefficients) can be computed by a width two monotone circuit over $\mathbb{R}$ of size $O(d \cdot s)$.*

Some Observations

To motivate the study of constant-width circuits, we point out that, for the problem of proving lower bounds for noncommutative bounded width circuits, Nisan's rank argument is not useful. For the noncommutative "palindromes" polynomial $P(x_0, x_1) = \sum_{w \in \{x_0, x_1\}^n} ww^R$, the communication matrix $M_n(P)$ is of rank 2^n and hence any noncommutative ABP for it is exponentially large [N91]. However, we can give an easy width-2 noncommutative arithmetic circuit for $P(x_0, x_1)$ of size $O(n)$. Indeed, we can even ensure that each gate in this circuit is *homogeneous.*

Proposition 2. *The palindromes polynomial $P(x_0, x_1)$ has a width-2 noncommutative arithmetic circuit of size $O(n)$.*

What then is a good candidate explicit polynomial that is not computable by width-2 circuits of polynomial size? We believe that the polynomial P_k^ℓ (of Section 2) for suitable k is the right candidate. A lower bound argument still eludes us. However, if we consider *monotone* constant-width circuits then even in the commutative case we can show exponential size lower bounds for monotone width-k circuits computing P_k^ℓ. Since P_k^ℓ is computable by depth $2k$ arithmetic circuits (of unbounded fanin), it follows that the constant-width and the constant-depth hierarchies of monotone arithmetic circuits are infinite. We present these results in Section 2.

Remark 1. Regarding the separation of the constant-depth hierarchy of monotone circuits, we note that a separation has also been proved by Raz and Yehudayoff in [RY09]; their lower bounds show a superpolynomial separation between the power of depth k *multilinear* circuits and depth $k+1$ monotone circuits for any k (see [RY09] for the definition and results regarding multilinear circuits). In contrast, our separation works only for monotone circuits, and only for infinitely many k. Nonetheless, we think that our separation is interesting because the separation we achieve is stronger. More precisely, the results of [RY09] show a separation of the order of $2^{(\log s)^{1+\Omega(1/k)}}$ (that is, there is a polynomial that can be computed by circuits of depth $k+1$ and size s but not by depth k circuits of size $2^{(\log s)^{1+\Omega(1/k)}}$). On the other hand, our separation is at least as large as $2^{(\log s)^c}$ for any $c > 0$ (see Section 2 for the precise separation).

A related question is the comparative power of noncommutative ABPs and noncommutative formulas. Noncommutative formulas have polynomial size noncommutative ABPs. However, $s^{O(\log s)}$ is the best known formula size upper bound for noncommutative ABPs of size s. An interesting question is whether we can prove a separation result between noncommutative ABPs and formulas. We note that such a separation in the *monotone* case follows from an old result of Snir [S80].

Proposition 3. *Consider two noncommuting variables $\{x_0, x_1\}$. Let L denote the set of all monomials of degree $2n$ with an equal number of x_0 and x_1, and consider the polynomial $E \in \mathbb{Q}\langle x_0, x_1\rangle$, where $E = \sum_{w\in L} w$.*

1. *There is a monotone homogeneous ABP for E of size $O(n^2)$.*
2. *Any monotone formula computing E is of size $n^{\Omega(\lg n)}$.*

Proof. The first part is directly from a standard $O(n^2)$ size DFA that accepts precisely the set $L = \{w \in \{x_0, x_1\}^{2n} \mid w$ has an equal number of x_0's and x_1's$\}$. The second part follows from the fact that such a monotone formula would yield a commutative monotone formula for the symmetric polynomial of degree n over the variables $y_1, y_2, \cdots, y_{2n}$: this is obtained by first observing that the formula must compute homogeneous polynomials at each gate. Furthermore, we can label each gate (and each leaf) by a triple (i, j, d) where $j - i + 1 = d$ is the degree of the homogeneous polynomial computed at this gate such that each monomial generated at this gate will occupy the positions from i to j in the output monomials containing it. Hence we have x_0's at the leaf nodes labeled by triples $(i, i, 1)$ for all $2n$ values of i. We replace the x_0's labeled $(i, i, 1)$ by y_i and each x_1 by 1. The resulting formula computes the symmetric polynomial as claimed. Snir in [S80] has shown a tight $n^{\Omega(\log n)}$ lower bound for monotone formulas computing the symmetric polynomial of degree n over the variables $y_1, y_2, \cdots, y_{2n}$. ■

To illustrate again the power of constant width circuits, we note that there is, surprisingly, a width-2 circuit for computing the polynomial E.

Proposition 4. *There is a width-2 circuit of size $n^{O(1)}$ for computing E if the field $\mathbb{F}$ has at least cn^2 distinct elements for some constant c.*

Proof Sketch. This is based on the well-known Ben-Or trick [B80] for computing the symmetric polynomials in depth 3. We consider the polynomial $g(x_0, x_1, z) = (x_0 z^{2^{k+1}+1} + x_1 z + 1)^{2^{k+1}}$, where $2^{k-1} < n \leq 2^k$. (z will eventually be a scalar from $\mathbb{F}$.) The coefficient of $z^{(2^{k+1}+1)n+n}$ in g is precisely the polynomial E. Following Ben-Or's argument, the problem of recovering the polynomial E can be reduced to solving a system of linear equations with an invertible coefficient matrix. Hence E can be expressed as a sum $E = \sum_{i=1}^{2n} \beta_i g(x_0, x_1, z_i)$, where the z_is are all distinct field elements. The terms $\beta_i g(x_0, x_1, z_i)$ can be evaluated with *one* register using repeated squaring of $x_0 z_i^{2^{k+1}+1} + x_1 z_i + 1$. The second register is used as an accumulator to compute the sum of these terms. ■

These observations are additional motivation for the study of constant-width arithmetic circuits. In Section 2 we prove lower bound results for monotone constant-width circuits. In Section 3 we explore the connection between lower bounds and polynomial identity testing for constant-width commutative circuits analogous to the work of Dvir et al [DSY08].

2 Monotone constant width circuits

In this section we study *monotone* constant-width arithmetic circuits. We prove that they form an infinite hierarchy. As a by-product, the separating polynomials that we construct yield the consequence that constant-depth monotone arithmetic circuits too form an infinite hierarchy. All our polynomials will be commutative, unless we explicitly state otherwise.

For positive integers k and ℓ we define a polynomial P_k^ℓ on ℓ^{2k} variables as follows:

$$P_1^\ell(x_1, x_2, \ldots, x_{\ell^2}) = \sum_{i=1}^{\ell} \prod_{j=1}^{\ell} x_{(i-1)\ell+j}$$

$$P_{k+1}^\ell(x_1, x_2, \ldots, x_{\ell^{2k+2}}) = \sum_{i=1}^{\ell} \prod_{j=1}^{\ell} P_k^\ell(x_{(i-1)\ell^{2k+1}+(j-1)\ell^{2k}+1}, \ldots, x_{(i-1)\ell^{2k+1}+j\ell^{2k}})$$

An easy inductive argument from the definition gives the following.

Lemma 2. *The polynomial P_k^ℓ is homogeneous of degree ℓ^k on ℓ^{2k} variables and has $\ell^{\frac{\ell^k-1}{\ell-1}}$ distinct monomials.*

By definition, P_k^ℓ can be computed by a depth $2k$ monotone formula of size $O(\ell^k)$. Furthermore, we can argue that the polynomials P_k^ℓ are the "hardest" polynomials for constant-depth circuits. We make this more precise in the following observation.

Proposition 5. *Given a depth k arithmetic circuit C of size s, there is a projection reduction from C to the polynomial P_k^ℓ where $\ell = O(s^{2k})$.*

Proof Sketch. We sketch the easy argument. We can transform C into a formula. Furthermore, we can make it a layered formula with $2k$ alternating $+$ and $\times$ layers such that the output gate is a plus gate. This formula is of size at most s^{2k}. Clearly, a projection reduction (mapping variables to variables or constants) will transform P_k^ℓ to this formula, for $\ell = O(s^{2k})$. ■

It is easy to see the following from the fact that a monotone depth $2k$ arithmetic circuit of size s can be simulated by a monotone width $2k$ circuit of size $O(s)$.

Proposition 6. *For any positive integers ℓ and k there is a monotone circuit of width $2k$ and size $O(\ell^{2k})$ that computes P^{ℓ}_{2k}.*

We now state the main lower bound result. For each $k > 0$ there is $\ell_0 \in \mathbb{Z}^+$ such that for all $\ell > \ell_0$ any width k monotone circuit for P^{ℓ}_{k} is of size $\Omega(2^{\ell})$. We will prove this result by induction on k. For the induction argument it is convenient to make a stronger induction hypothesis.

For a polynomial $f \in \mathbb{F}[X]$, where $X = \{x_1, x_2, \cdots, x_n\}$ let $\mathrm{mon}(f) = \{m \mid m$ is a nonzero monomial in $f\}$. I.e. $\mathrm{mon}(f)$ denotes the set of nonzero monomials in the polynomial f. Also, let $\mathrm{var}(f)$ denote the set of variables occurring in the monomials in $\mathrm{mon}(f)$. Similarly, for an arithmetic circuit C we denote by $\mathrm{mon}(C)$ and $\mathrm{var}(C)$ respectively the set of nonzero monomials and variables occurring in the polynomial computed by C.

We call a layered circuit C *minimal* if there is no smaller circuit C' of the same width s.t $\mathrm{mon}(C) = \mathrm{mon}(C')$. It can be seen that for any monotone circuit C, there is a minimal circuit C' of the same width s.t $\mathrm{mon}(C') = \mathrm{mon}(C)$ and has the following properties.

- The only constants used in C' are 0 and 1. Furthermore, no gate is ever multiplied by a constant.
- By the minimality of C' every node g in C' has a path to the output node of C'. Hence, given any node g in C' computing a polynomial p, there is a monomial m such that $\mathrm{mon}(m \cdot p) \subseteq \mathrm{mon}(C')$. In particular, this implies that if C' computes a homogeneous multilinear polynomial, then p must be a homogeneous multilinear polynomial.
- If C' computes a homogeneous multilinear polynomial of degree d, and if a node g in layer i also computes a polynomial p of degree d, then in layer $i+1$, there is a sum gate g' such that g is one of its children. Thus, the gate g' computes a homogeneous multilinear polynomial p' of degree d such that $\mathrm{mon}(p) \subseteq \mathrm{mon}(p')$. In particular, $\mathrm{mon}(p) \subseteq \mathrm{mon}(C')$.

We call a minimal circuit satisfying the above a *good* minimal circuit. We now show a useful property of minimal circuits C, which applies to circuits satisfying $\mathrm{mon}(C) \subseteq P^{\ell}_{k}$, for all $\ell, k \geq 1$.

Lemma 3. *Let $f = \sum_{i=1}^{\ell} P_i$ be a homogeneous monotone polynomial of degree $d \geq 1$ with $var(P_i) \cap var(P_j) = \emptyset$ for all $i \neq j$. Given any good minimal circuit such that $mon(C) \subseteq mon(f)$, we have the following: if a gate g in C computes a polynomial p of degree less than d, or a product of two such polynomials, then $var(p) \subseteq var(P_i)$ for a* unique *i.*

Proof. For any polynomial $q \in \mathbb{F}[x_1, x_2, \cdots, x_n]$ we can define a bipartite graph $G(q)$ as follows: one partition of the vertex set is $\mathrm{mon}(q)$ and the other partition $\mathrm{var}(q)$. A pair $\{x, m\}$ is an undirected edge if the variable x occurs in monomial m. It is clear that the graph $G(f)$ is just the disjoint union of all the $G(P_i)$.

If the polynomial p computed by gate g is of degree $d' < d$, then, since C is good, there is a monomial m of degree $d' - d$ such that $\mathrm{mon}(m \cdot p) \subseteq \mathrm{mon}(C) \subseteq \mathrm{mon}(f)$. This implies that $G(m \cdot p)$ is a subgraph of $G(f)$. On the other hand, $G(m \cdot p)$ is clearly seen to be a connected graph. This implies that, in fact, $G(m \cdot p)$ is a subgraph of $G(P_i)$ for some i and hence, $\mathrm{var}(p) \subseteq \mathrm{var}(P_i)$ for a *unique* i. This proves the lemma in this case.

Similarly, if p is a product of two polynomials of degree less than d, then $G(p)$ is a connected graph, and by the above reasoning, it must be the subgraph of some $G(P_i)$. Hence, the lemma follows. ∎

We now state and prove a stronger lower bound statement. It shows that P_k^ℓ is even hard to "approximate" by polynomial size width-k monotone circuits.

Theorem 1. *For each $k > 0$ there is $\ell_0 \in \mathbb{Z}^+$ such that for all $\ell > \ell_0$ and any width-k monotone circuit C such that*

$$mon(C) \subseteq mon(P_k^\ell) \text{ and } |mon(C)| \geq \frac{|mon(P_k^\ell)|}{2},$$

the circuit C is of size at least $\frac{2^\ell}{10}$.

Proof. Let us fix some notation: given $i \in \mathbb{Z}^+$ and $j \in [w]$, we denote by $g_{i,j}$ the jth node in layer i of C and by $f_{i,j}$ the polynomial computed by $g_{i,j}$. Also, given a set of monomials M, we say that a circuit C_1 *computes* M if $\text{mon}(C_1) \supseteq M$.

Without loss of generality, we assume throughout that C is a good minimal circuit. The proof is by induction on k. The case $k = 1$ is distinct and easy to handle. Thus, we consider as the induction base case the case $k = 2$. Consider a width two monotone circuit C such that $\text{mon}(C) \subseteq \text{mon}(P_2^\ell)$ and $|\text{mon}(C)| \geq |\text{mon}(P_2^\ell)|/2 = \ell^{\ell+1}/2$. Let f denote the polynomial computed by C. By Lemma 2 both f and P_2^ℓ are homogeneous polynomials of degree $d = \ell^2$.

We write the polynomial P_2^ℓ as $\sum_{i=1}^{\ell} P_i$, where $\text{var}(P_i) = \{x_{(i-1)\ell^3+1}, \ldots, x_{i\ell^3}\}$. Note that $\text{var}(P_i) \cap \text{var}(P_j) = \emptyset$ for $i \neq j$. Let $f = \sum_{i=1}^{\ell} P'_i$ where $\text{mon}(P'_i) \subseteq \text{mon}(P_i)$ for each i.

Since C is good and f is homogeneous, each gate of C computes only homogeneous polynomials. Moreover, since $\text{mon}(C) \subseteq \text{mon}(P_2^\ell)$ and $\text{var}(P_i) \cap \text{var}(P_j) = \emptyset$ for $i \neq j$, Lemma 3 implies that given any node g in C that computes a polynomial p of degree less than d or a product of such polynomials satisfies $\text{var}(p) \subseteq \text{var}(P_i)$ for *one* i. Consider the lowest layer (i_0 say) when the circuit C computes a degree d monotone polynomial. W.l.o.g assume that $f_{i_0,1}$ is such a polynomial. We list some crucial properties satisfied by $g_{i_0,1}$ and C.

1. By the minimality of i_0, the node $g_{i_0,1}$ is a product gate computing the product of polynomials of degree less than d. Hence, $\text{var}(f_{i_0,1}) \subseteq \text{var}(P_i)$ for exactly one i. W.l.o.g , we assume $i = 1$. Since $\deg(f_{i_0}, 1) = d$ and C is good, we in fact have $\text{mon}(f_{i_0,1}) \subseteq \text{mon}(P_1)$.
2. Since $\deg(f_{i_0,1}) = d$ and C is good, we know that there is a node $g_{i_0+1,j_{i_0+1}}$ that is a sum gate with $g_{i_0,1}$ as child; $g_{i_0+1,j_{i_0+1}}$ computes a homogeneous polynomial of degree d and $\text{mon}(f_{i_0+1,j_{i_0+1}}) \supseteq \text{mon}(f_{i_0,1})$. Iterating this argument, we see that there must be a sequence of nodes g_{i,j_i}, for $i > i_0$ such that for each i, g_{i,j_i} is a sum gate with $g_{i-1,j_{i-1}}$ as child, such that $\text{mon}(f_{i_0,1}) \subseteq \text{mon}(f_{i_0+1,j_{i_0+1}}) \subseteq \text{mon}(f_{i_0+2,j_{i_0+2}}) \ldots$, and each f_{i,j_i} is a homogeneous polynomial of degree d. We assume, w.l.o.g, that $j_i = 1$ for each $i > i_0$.

By the choice of i_0, note that the node $g_{i_0,2}$ either computes a polynomial of degree less than d or computes a product of polynomials of degree less than d. Hence, $\text{var}(f_{i_0,2}) \subseteq \text{var}(P_i)$ for some i. If $i > 1$, we assume w.l.o.g. that $\text{var}(p) \subseteq \text{var}(P_2)$. Let us consider the circuit C with the variables in $\text{var}(P_1) \cup \text{var}(P_2)$ set to 0. The polynomial computed by the new circuit C' is now $f' = f - P'_1 - P'_2 = \sum_{i=3}^{\ell} P'_i$. Let $q_{i,j}$ denote the new polynomial computed by the node $g_{i,j}$. Note that each $q_{i_0,j}$ is now a constant polynomial.

Consider the monotone circuit C'' obtained from C' as follows: we remove all the gates below layer i_0; the gate $g_{i_0,2}$ in layer i_0 is replaced by a product gate $c \times 1$, where c is the constant it computes in C'; from layer i_0 onwards, all nodes of the form $g_{i,1}$ are removed; in any edge connecting nodes $g_{i,1}$ and $g_{i+1,2}$, the node $g_{i,1}$ is replaced by the constant 0. Clearly, C'' is a width 1 circuit. For ease of notation, we will refer to the nodes of C'' with the same names as the corresponding nodes in C'. For any node $g_{i,2}$ in C'' ($i \geq i_0$), let $q'_{i,2}$ be the polynomial it now computes. Crucially, we observe the following from the above construction.

Claim 2. *For each $i \geq i_0$, $mon(q'_{i,2}) \supseteq mon(q_{i,2}) \setminus mon(q_{i,1})$.*

We now finish the proof of the base case. Define a sequence $i_1 < i_2 < \ldots < i_t$ of layers as follows: for each $j \in [t]$, i_j is the least $i > i_{j-1}$ such that $\mathrm{mon}(q_{i,1}) \supsetneq \mathrm{mon}(q_{i_{j-1},1})$, and $\mathrm{mon}(q_{i_t,1}) = \mathrm{mon}(f')$. Clearly, t is at most the size of C. Note that it must be the case that $q_{i_j,1} = q_{i_j-1,1} + q_{i_j-1,2}$. Hence, we have $\mathrm{mon}(q_{i_j,1}) = \mathrm{mon}(q_{i_j-1,1}) \cup \mathrm{mon}(q_{i_j-1,2}) = \mathrm{mon}(q_{i_j-1,1}) \cup (\mathrm{mon}(q_{i_j-1,2}) \setminus \mathrm{mon}(q_{i_j-1,1}))$. By the above claim, the set $\mathrm{mon}(q_{i_j-1,2}) \setminus \mathrm{mon}(q_{i_j-1,1})$, which we will denote by S_j, can be computed by a width-1 circuit. Thus, $\mathrm{mon}(f') = \mathrm{mon}(q_{i_t,1}) = \mathrm{mon}(q_{i_0,1}) \cup \bigcup_{j=1}^{t} S_j$, where each S_j can be computed by a width-1 circuit. Since $q_{i_0,1}$ is the zero polynomial, we have $\mathrm{mon}(f') = \bigcup_{j=1}^{t} S_j$.

Now, consider any width-1 monotone circuit computing a set $S \subseteq P_2^{\ell}$. It is easy to see that the set S computed must have a very restricted form.

Claim 3. *The set S is of the form $mon(p)$ where $p = (\sum_{i \in X_1} x_i) \prod_{j \in X_2} x_j$, and $X_1 \cap X_2 = \emptyset$.*

Clearly, as each set S_j satisfies $S_j \subseteq \mathrm{var}(P'_i)$ for some i, it can have at most ℓ^3 monomials. Therefore, if the monotone circuit C is of overall size less than 2^{ℓ} then it can compute a polynomial of the form $P'_1 + P'_2 + f'$, where f' has at most $2^{\ell}\ell^3$ monomials. Since $|\mathrm{mon}(P'_i)| \leq |\mathrm{mon}(P_i)| = \ell^{\ell}$ for each i, we have for suitably large ℓ

$$|\mathrm{mon}(C)| \leq 2\ell^{\ell} + 2^{\ell}\ell^3 < 3\ell^{\ell} < \frac{\ell^{\ell+1}}{2} = \frac{|\mathrm{mon}(P_2^{\ell})|}{2}$$

and the base case follows.

The induction step.

Consider any monotone circuit $\hat{C}$ of width $k-1$ such that $\mathrm{mon}(\hat{C}) \subseteq \mathrm{mon}(P_{k-1}^{\ell})$ and $|\mathrm{mon}(\hat{C})| \geq |\mathrm{mon}(P_{k-1}^{\ell})|/2$. As induction hypothesis we assume that $\hat{C}$ must be of size at least $2^{\ell}/10$.

Let $P_k^{\ell} = \sum_{i=1}^{\ell} P_i$, with $\mathrm{var}(P_i) = \{x_{(i-1)\ell^{2k+1}+1}, \ldots, x_{i\ell^{2k+1}}\}$ as in the base case. By definition, the ℓ variable sets $\mathrm{var}(P_i)$ are mutually disjoint and each P_i has degree $d = \ell^k$. It is convenient to also write $P_i = \prod_{j=1}^{\ell} Q_{ij}$, where each Q_{ij} is of type P_{k-1}^{ℓ}. We have $\mathrm{var}(Q_{ij}) = \{x_{(i-1)\ell^{2k+1}+(j-1)\ell^{2k}+1}, \ldots, x_{(i-1)\ell^{2k+1}+j\ell^{2k}}\}$.

We start by considering any width $k-1$ circuit $\hat{C}$ of size less than $2^{\ell}/10$ such that $\mathrm{mon}(\hat{C}) \subseteq \mathrm{mon}(P_k^{\ell})$. For any $i \in [\ell]$, by fixing all the variables outside $\mathrm{var}(P_i)$ to 0, we obtain a width $k-1$ circuit $\hat{C}_i$ of the same size s.t $\mathrm{mon}(\hat{C}_i) \subseteq \mathrm{mon}(P_i)$. Further, by setting all the variables outside $\mathrm{var}(Q_{ij})$ to 1 for some $j \in [\ell]$, we obtain a circuit $\hat{C}_{ij}$ s.t $\mathrm{mon}(\hat{C}_{ij}) \subseteq \mathrm{mon}(Q_{ij})$. By the induction hypothesis, we see that $|\mathrm{mon}(\hat{C}_{ij})| \leq |\mathrm{mon}(Q_{ij})|/2$. Clearly $\mathrm{mon}(\hat{C}_i) \subseteq \mathrm{mon}(\hat{C}_{i1}) \times \mathrm{mon}(\hat{C}_{i2}) \times \ldots \times \mathrm{mon}(\hat{C}_{i\ell})$. Therefore, $|\mathrm{mon}(\hat{C}_i)| \leq \prod_j |\mathrm{mon}(\hat{C}_{ij})| \leq |\mathrm{mon}(P_i)|/2^{\ell}$. Finally, as $\mathrm{mon}(\hat{C}) = \bigcup_i \mathrm{mon}(\hat{C}_i)$, $|\mathrm{mon}(\hat{C})| \leq \sum_i |\mathrm{mon}(\hat{C}_i)| \leq |\mathrm{mon}(P_k^{\ell})|/2^{\ell}$. We have established the following claim.

Claim 4. *For any width $k-1$ circuit $\hat{C}$ of size less than $2^{\ell}/10$ such that $mon(\hat{C}) \subseteq mon(P_k^{\ell})$, we have $|mon(\hat{C})| \leq \frac{|mon(P_k^{\ell})|}{2^{\ell}}$.*

For the induction step, consider any monotone width-k circuit C such that $\mathrm{mon}(C) \subseteq \mathrm{mon}(P_k^\ell)$ and of size at most $2^\ell/10$. We will show that $|\mathrm{mon}(C)| < |\mathrm{mon}(P_k^\ell)|/2$. W.l.o.g, we can assume that C is a good minimal circuit. Let f denote the polynomial computed by C; we write $f = \sum_{i=1}^{\ell} P'_i$, where $\mathrm{mon}(P'_i) \subseteq \mathrm{mon}(P_i)$ for each i.

As in the base case, let i_0 be the first layer where a polynomial of degree d is computed. W.l.o.g. we can assume that $f_{i_0,1}$ is such a polynomial. By the minimality of i_0, the node $g_{i_0,1}$ must be a product node with children computing polynomials of degree less than d. This implies, as in the base case, that $\mathrm{var}(f_{i_0,1}) \subseteq \mathrm{var}(P_i)$ for a unique i. W.l.o.g. we assume that $i = 1$. As before, we can fix a sequence of nodes g_{i,j_i} for each $i > i_0$ such that g_{i,j_i} is a sum gate with $g_{i-1,j_{i-1}}$ as a child. It is easily seen that $\mathrm{mon}(f_{i_0,1}) \subseteq \mathrm{mon}(f_{i_0+1,j_{i_0+1}}) \subseteq \mathrm{mon}(f_{i_0+2,j_{i_0+2}}) \ldots$, and each f_{i,j_i} computes a homogeneous polynomial of degree d. Renaming nodes if necessary, we assume $j_i = 1$ for all i.

Now consider $f_{i_0,j}$ for $j > 1$. By the minimality of i_0, we see that each $f_{i_0,j}$ is either a polynomial of degree less than d or a product of two such polynomials. Hence, $\mathrm{var}(f_{i_0,j}) \subseteq \mathrm{var}(P_s)$ for some $s \in [\ell]$. Thus, there is a set $S \subseteq [\ell]$ s.t $|S| = k' < k$ such that $\bigcup_{j>1} \mathrm{var}(f_{i_0,j}) \subseteq \bigcup_{s\in S} \mathrm{var}(P_s)$. Without loss of generality, we assume that those $s \in S$ that are greater than 1 are among $\{2, 3, \ldots, k\}$.

Consider the circuit C' obtained when each of the variables in $\bigcup_{s\in[k]} \mathrm{var}(P_s)$ is set to 0. Let $q_{i,j}$ be the polynomial computed by $g_{i,j}$ in C'. The polynomial computed by C' is just $f' = f - \sum_{s\in[k]} P'_s$. Note that $q_{i_0,j}$ is now simply a constant for each j, and that the size of C' is at most the size of C which by assumption is bounded by $2^\ell/10$. Using this size bound we will argue that C' cannot compute too many monomials.

We now modify C' as follows: we remove all the gates below layer i_0; each gate $g_{i_0,j}$ with $j > 1$ is replaced by a product gate of the form $c \times 1$ where c is the constant $g_{i_0,1}$ computes in C'; from layer i_0 onwards, all nodes of the form $g_{i,1}$ are removed; in any edge connecting nodes $g_{i,1}$ and $g_{i+1,j}$ for $j > 1$, the node $g_{i,1}$ is replaced by the constant 0. Call this new circuit C''. Clearly, C'' has size at most the size of C and width at most $k-1$. For ease of notation, we will refer to the nodes of C'' with the same names as the corresponding nodes in C'. For any node $g_{i,j}$ in C'' ($i \geq i_0$ and $j > 1$), let $q'_{i,j}$ be the polynomial it now computes. As in the base case, we observe the following from the above construction.

Claim 5. *For each $i \geq i_0$ and each $j > 1$, $mon(q'_{i,j}) \supseteq mon(q_{i,j}) \setminus mon(q_{i,1})$.*

Using this, we show that the circuit C' was essentially just using the gates $g_{i,1}$ to store the sum of polynomials computed using width $k-1$ circuits.

Construct a sequence of layers $i_1 < i_2 < \ldots < i_t$ in C' as follows: for each $j \in [t]$, i_j is the least $i > i_{j-1}$ such that $\mathrm{mon}(q_{i,1}) \supsetneq \mathrm{mon}(q_{i_{j-1},1})$, and $\mathrm{mon}(q_{i_t,1}) = \mathrm{mon}(f')$. Surely, t is at most the size of C'. Now, fix any i_j for $j \geq 1$. Clearly, it must be the case that $q_{i_j,1} = q_{i_j-1,1} + q_{i_j-1,s}$ for some $s > 1$; therefore, we have $\mathrm{mon}(q_{i_j,1}) \subseteq \mathrm{mon}(q_{i_j-1,1}) \cup (\mathrm{mon}(q_{i_j-1,s}) \setminus \mathrm{mon}(q_{i_j-1,1}))$. Denote the set $\mathrm{mon}(q_{i_j-1,s}) \setminus \mathrm{mon}(q_{i_j-1,1})$ by S_j. Since the above holds for all j, and $\mathrm{mon}(q_{i_j-1,1}) = \mathrm{mon}(q_{i_{j-1},1})$, we see that $\mathrm{mon}(f') = \mathrm{mon}(q_{i_t,1}) \subseteq \mathrm{mon}(q_{i_0,1}) \cup \bigcup_j S_j = \bigcup_j S_j$, since $q_{i_0,1}$ is the zero polynomial.

We will now analyze $|S_j|$ for each j. By the above claim, there is a width $k-1$ circuit C'' of size at most the size of C such that $S_j \subseteq \mathrm{mon}(C'') \subseteq P_k^\ell$. If the size of C (and hence that of C' and C'') is at most $2^\ell/10$, it follows from Claim 4 that $|S_j| \leq |\mathrm{mon}(P_k^\ell)|/2^\ell$. Hence, we see that $|\mathrm{mon}(f')| \leq t|\mathrm{mon}(P_k^\ell)|/2^\ell$, which is at most $|\mathrm{mon}(P_k^\ell)|/10$. But we know that the polynomial f computed by the circuit C is of the form $f' + \sum_{i\in[k]} P'_i$, where $|\mathrm{mon}(P'_i)| \leq$

$|\mathrm{mon}(P_i)| = |\mathrm{mon}(P_k^\ell)|/\ell$. Therefore,

$$|\mathrm{mon}(f)| \leq \frac{k}{\ell}|\mathrm{mon}(P_k^\ell)| + |\mathrm{mon}(f')| \leq |\mathrm{mon}(P_k^\ell)|\left(\frac{k}{\ell} + \frac{1}{10}\right) < \frac{|\mathrm{mon}(P_k^\ell)|}{2}$$

for large enough ℓ. This proves the induction step. ■

For $k \in \mathbb{Z}^+$ and $c > 0$ let $\mathrm{Depth}_{k,c}$ and $\mathrm{Width}_{k,c}$ denote the set of families $\{f_n\}_{n>0}$ of monotone polynomials $f_n \in \mathbb{R}[x_1, x_2, \ldots, x_n]$ computed by $c \cdot n^c$-sized monotone circuits of depth k and width k respectively. For $k \in \mathbb{Z}^+$, let $\mathrm{Depth}_k = \bigcup_{c>0} \mathrm{Depth}_{k,c}$ and $\mathrm{Width}_k = \bigcup_{c>0} \mathrm{Width}_{k,c}$. Thus, Depth_k and Width_k denote the set of families of monotone polynomials computed by $\mathrm{poly}(n)$-sized monotone circuits of depth k and width k respectively. Note that, for each $k \in \mathbb{Z}^+$ we have $\mathrm{Depth}_k \subseteq \mathrm{Width}_k$. Moreover, from the definition of P_k^ℓ, we see that the family $\{P_k^{\lfloor n^{1/2k} \rfloor}\}_n \in \mathrm{Depth}_{2k}$. Finally, in Theorem 1 we have shown that the family $\{P_k^{\lfloor n^{1/2k} \rfloor}\}_n \notin \mathrm{Width}_k$, for constant k. Hence, we have the following corollary of Theorem 1.

Corollary 1. *For any fixed $k \in \mathbb{Z}^+$, $\mathrm{Width}_k \subsetneq \mathrm{Width}_{2k}$ and $\mathrm{Depth}_k \subsetneq \mathrm{Depth}_{2k}$.*

Theorem 1 can also be used to give a separation between the power of circuits of width (respectively, depth) k and $k+1$ for infinitely many k. We now state this separation. For any $k \in \mathbb{N}$ and any function $f : \mathbb{N} \to \mathbb{N}$, let us denote by f^k the *k-th iterate* of f, i.e the function $\underbrace{f \circ f \circ \ldots \circ f}_{k \text{ times}}$. Given non-decreasing functions $f, g : \mathbb{N} \to \mathbb{N}$, call f a *sub $1/k$-th iterate of g* if $f^k(n) < g(n)$, for large enough n (closely related notions have been defined in [Sz61] and [RR97]). It can be verified that sub $1/k$-th iterates of exponential functions can grow fairly quickly: for example, for any $\varepsilon > 0$ and any $k, c \in \mathbb{N}$, the function $2^{(\log n)^c}$ is a sub $1/k$-th iterate of 2^{n^ε}.

We now state the precise separation that can be inferred from the above theorem. For any $k, n \in \mathbb{N}$ with $k \geq 2$ and any polynomial $p \in \mathbb{R}[x_1, x_2, \ldots, x_n]$, let $w_k(p)$ (resepctively $d_k(p)$) denote the size of the smallest monotone width k (respectively depth k) circuit that computes p.

Corollary 2. *There is an absolute constant $\alpha > 0$ such that the following holds. Fix any $k \in \mathbb{N}$ where $k \geq 2$. Also, fix any non-decreasing function $f : \mathbb{N} \to \mathbb{N}$ that is a sub $1/k$-th iterate of $2^{\alpha n^{1/2k}}$. Then, for large enough n, there is a monotone polynomial $p \in \mathbb{R}[x_1, x_2, \ldots, x_n]$ such that for some $k', k'' \in \{k, k+1, \ldots, 2k-1\}$, $w_{k'}(p) \geq f(w_{k'+1}(p))$ and $d_{k''}(p) \geq f(d_{k''+1}(p))$.*

Proof. Let p denote the monotone polynomial $P_k^{\lfloor n^{1/2k} \rfloor} \in \mathbb{R}[x_1, x_2, \ldots, x_n]$. Theorem 1 tells us that $w_k(p) = \Omega(2^{\lfloor n^{1/2k} \rfloor})$. To obtain a lower bound on $d_k(p)$, note that any polynomial computed by a circuit of size s and depth k can be computed by a width k circuit of size $O(s^k)$; this tells us that $d_k(p) = 2^{\Omega(n^{1/2k})}$. Hence, there is some constant $\beta > 0$ such that $\min\{w_k(p), d_k(p)\} \geq 2^{\beta n^{1/2k}}$, for large enough n.

By definition, $p = P_k^{\lfloor n^{1/2k} \rfloor}$ has a depth $2k$ circuit of size $O(n)$, i.e $d_{2k}(p) = O(n)$. Proposition 6 tells us that $w_{2k}(p) = O(n)$ also. Hence, for some constant $\gamma > 0$ and large enough n, we have $\max\{w_{2k}(p), d_{2k}(p)\} \leq \gamma n$.

The above statements imply that $w_k(p) \geq g(w_{2k}(p))$ and $d_k(p) \geq g(d_{2k}(p))$, where $g(n) = 2^{\alpha n^{1/2k}}$ for some constant $\alpha > 0$ and n is large enough. Now, fix any non-decreasing function

$f : \mathbb{N} \to \mathbb{N}$ that is a sub $1/k$-th iterate of g. We see that $w_k(p) \geq g(w_{2k}(p)) > f^k(w_{2k}(p))$ for large enough n; clearly, this implies that for some $k' \in \{k, k+1, \ldots, 2k-1\}$, we must have $w_{k'}(p) \geq f(w_{k'+1}(p))$. Similarly, there is also a $k'' \in \{k, k+1, \ldots, 2k-1\}$ such that $d_{k''}(p) \geq f(d_{k''+1}(p))$. ■

Similar corollaries hold for noncommutative circuits too. We define the polynomial P_k^ℓ in exactly the same way in the noncommutative setting. Note that any monotone bounded width noncommutative circuit computing P_k^ℓ automatically gives us a monotone commutative circuit of the same size and width computing the commutative version of P_k^ℓ. Hence, the lower bound of Theorem 1 also holds for noncommutative width-k circuits. For $k \in \mathbb{Z}^+$, let $\mathrm{ncDepth}_k$ and $\mathrm{ncWidth}_k$ denote the set of families of monotone polynomials $\{f_n \in \mathbb{R}\langle x_1, x_2, \ldots, x_n\rangle \mid n \in \mathbb{Z}^+\}$ computed by poly(n)-sized monotone (noncommutative) circuits of depth k and width k respectively. Analogous to the commutative case, we obtain the following.

Corollary 3. *For any fixed $k \in \mathbb{Z}^+$, $ncWidth_k \subsetneq ncWidth_{2k}$ and $ncDepth_k \subsetneq ncDepth_{2k}$.*

And finally, we observe that the separations between width and depth k and $k+1$ that hold in the commutative monotone case also hold in the noncommutative monotone case. Define, for any $k, n \in \mathbb{N}$ with $k \geq 2$ and any polynomial $p \in \mathbb{R}\langle x_1, x_2, \ldots, x_n\rangle$, let $ncw_k(p)$ (resepctively $ncd_k(p)$) denote the size of the smallest monotone width k (respectively depth k) circuit that computes p. We have the following.

Corollary 4. *There is an absolute constant $\alpha > 0$ such that the following holds. Fix any $k \in \mathbb{N}$ where $k \geq 2$. Also, fix any non-decreasing function $f : \mathbb{N} \to \mathbb{N}$ that is a sub $1/k$-th iterate of $2^{\alpha n^{1/2k}}$. Then, for large enough n, there is a monotone polynomial $p \in \mathbb{R}\langle x_1, x_2, \ldots, x_n\rangle$ such that for some $k', k'' \in \{k, k+1, \ldots, 2k-1\}$, $ncw_{k'}(p) \geq f(ncw_{k'+1}(p))$ and $ncd_{k''}(p) \geq f(ncd_{k''+1}(p))$.*

3 Identity testing for constant width circuits

In this section we study polynomial identity testing for constant-width commutative circuits. Impagliazzo and Kabanets [KI03] showed that derandomizing polynomial identity testing is equivalent to proving arithmetic circuit lower bounds. Specifically, assuming that there are explicit polynomials that require superpolynomial size arithmetic circuits, they use these polynomials in a Nisan-Wigderson type "arithmetic" pseudorandom generator that can be used to derandomized polynomial identity testing. This idea was refined by Dvir et al [DSY08] to show that if there are explicit polynomials that require superpolynomial size constant-depth arithmetic circuits then polynomial identity testing for constant-depth arithmetic circuits can be derandomized (the precise statement involves the depth parameter explicitly [DSY08]).

In this section we prove a similar result showing that hardness for constant-width arithmetic circuits yields a derandomization of polynomial identity testing for constant-width circuits. We say that a family of multilinear polynomials $\{P_n\}_{n>0}$ where $P_n(\overline{x}) \in \mathbb{F}[x_1, \cdots, x_n]$ is *explicit* if the coefficient of each monomial m of the polynomial P_n can be computed in time $2^{n^{O(1)}}$.

Recall the notion of a staggered arithmetic circuit (Definition 2).

Lemma 4. *Let $f \in \mathbb{F}[x_1, x_2, \cdots, x_n]$ of degree m be computed by a staggered arithmetic circuit of size s and width w. Then $H_i(f)$ (the i^{th} homogeneous component of f) can be computed by* *a staggered circuit of size* $\text{poly}(s, m)$ *and width* $w + O(1)$*, provided* $\mathbb{F}$ *has at least* $\deg(f) + 1$ *many elements.*

Proof. Define a new polynomial $g(\overline{x}, z) \in \mathbb{F}[x_1, x_2, \cdots, x_n, z]$ as $g(\overline{x}, z) = f(x_1 z, x_2 z, \cdots, x_n z)$.

We can write $f(x_1 z, x_2 z, \cdots, x_n z) = \sum_{i=0}^{m} H_i(f) z^i$ where $m = \deg(f)$ and $H_i(f)$ is the i^{th} homogeneous part of f. Let $\{z_0, z_1, \cdots, z_m\}$ be $m + 1$ distinct field elements. Consider the matrix M defined as

$$M = \begin{pmatrix} 1 \; z_0 \; z_0^2 & \cdots \; z_0^m \\ 1 \; z_1 \; z_1^2 & \cdots \; z_1^m \\ \cdots\cdots & \cdots\cdots \\ 1 \; z_m \; z_m^2 & \cdots \; z_m^m \end{pmatrix}.$$

We have the system of equations

$$M(H_0(f), H_1(f), \cdots, H_m(f))^T = (g(\overline{x}, z_0), g(\overline{x}, z_1), \cdots, g(\overline{x}, z_m))^T.$$

Since M is invertible, it follows that there are scalars $a_{ij} \in \mathbb{F}$ such that $H_i(f) = \sum_{j=0}^{m} a_{ij} g(\overline{x}, z_j)$.

Since $f(\overline{x})$ has a width w circuit of size s, $g(\overline{x}, z)$ clearly has a (staggered) circuit of width $w + O(1)$ of size $O(s)$. It follows easily from the above equation for $H_i(f)$ that each $H_i(f)$ has a circuit of width $w + O(1)$ and size $O(ms)$. ■

Lemma 5. *Let* $P(x_1, x_2, \cdots, x_n, y)$ *be a polynomial, over a sufficiently large field* $\mathbb{F}$*, computed by a width* w *staggered circuit of size* s*. Suppose the maximum degree of* y *in* P *is* r*. Then for each* j *the* j^{th} *partial derivative* $\frac{\partial^j P}{\partial y^j}$ *can be computed by a staggered circuit of width* $w + O(1)$ *and size* $(rs)^{O(1)}$*.*

Proof. Let $P(\overline{x}, y) = \sum_{i=0}^{r} C_i(\overline{x}) y^i$. As in Lemma 4 each $C_i(\overline{x})$ can be computed by a width $w + O(1)$ staggered circuit of size $O(rs)$. Clearly, for each j the polynomial $\frac{\partial^j P}{\partial y^j}$ can be written as

$$\frac{\partial^j P}{\partial y^j} = \sum_{i=j}^{r} a_{ij} C_i(\overline{x}) y^{i-j},$$

for $a_{ij} \in \mathbb{F}$, where a_{ij} are field elements that depend *only* upon j. Therefore, we can easily give a staggered circuit of size $O(r^2 s)$ and width $w + O(1)$ for each polynomial $\frac{\partial^j P}{\partial y^j}$. ■

The following lemma is proved in [DSY08]. For any polynomial $g \in \mathbb{F}[x_1, x_2, \cdots, x_n]$ let $H_{\leq k}(g) = \sum_{i=0}^{k} H_i(g)$.

Lemma 6. [DSY08, Lemma 3.2] *Let* $P \in \mathbb{F}[x_1, x_2, \cdots, x_n, y]$ *and* $\deg_y(P) = r$*. Suppose* $f \in \mathbb{F}[x_1, x_2, \cdots, x_n]$ *such that* $P(\overline{x}, f(\overline{x})) = 0$ *and* $\frac{\partial P}{\partial y}(\overline{0}, f(\overline{0}))$ *is equal to* $\xi \neq 0$*. Let* $P(\overline{x}, y) = \sum_{i=1}^{r} C_i(\overline{x}) y^i$*. Then for each* $k \geq 0$ *there is a polynomial* $Q_k \in \mathbb{F}[y_0, y_1, \cdots, y_r]$ *such that*

$$H_{\leq k}(f) = H_{\leq k}(Q_k(C_0, C_1, \cdots, C_r)).$$

Using the above lemmata we prove our first theorem.

Theorem 6. *Let* $P \in \mathbb{F}[x_1, x_2, \cdots, x_n, y]$ *and* $\deg_y(P) = r \geq 1$ *such that* P *has a staggered circuit of size* s *and width* w*. Suppose that* $P(\overline{x}, f(\overline{x})) = 0$ *for some polynomial* $f \in \mathbb{F}[x_1, x_2, \cdots, x_n]$ *with* $\deg(f) = m$*. Then* f *has a staggered circuit of size* $\text{poly}(s, (m+r)^r)$ *and width* $w + O(1)$ *if* $char(\mathbb{F}) > r$ *and* $\mathbb{F}$ *is sufficiently large.*

Proof. First we argue that we can assume w.l.o.g., as in Dvir et al [DSY08], that $\frac{\partial P}{\partial y}(\overline{0}, f(\overline{0})) = \xi \neq 0$. If $\frac{\partial P}{\partial y}(x, f(x)) \equiv 0$ we can replace P by $\frac{\partial P}{\partial y}$. Since $char(\mathbb{F}) > r$ it is easy to see that there exists $j : 1 \leq j \leq r$ such that $\frac{\partial^j P}{\partial y^j}(x, f(x)) \not\equiv 0$. Hence, we can assume $\frac{\partial P}{\partial y}(x, f(x)) \not\equiv 0$. Therefore, there is an $a \in \mathbb{F}^n$ such that $\frac{\partial P}{\partial y}P(a, f(a)) \neq 0$. We can assume that $a = 0$ by appropriately shifting P as in [DSY08]. Let

$$P(\overline{x}, y) = \sum_{i=1}^{r} C_i(\overline{x})y^i.$$

By Lemma 6 there is a polynomial $Q_k \in \mathbb{F}[y_0, \cdots, y_r]$ such that $H_{\leq k}(f) = H_{\leq k}(Q_k(C_0, C_1, \cdots, C_r))$ for each $0 \leq k \leq m$. Putting $k = m$ and letting $Q_m = Q$ we have $f(\overline{x}) = H_{\leq m}(Q(C_0, C_1, \cdots, C_r))$.

Let $y^* = (C_0(0), \cdots, C_r(0))$ and $\deg(Q) = M$. Define $I_M = \{(\alpha_0, \alpha_1, \cdots, \alpha_r) \mid \alpha_i \in \mathrm{N}, \sum \alpha_i \leq M\}$. By expanding the polynomial Q at the point y^* we get $Q(\overline{y}) = \sum_{\overline{\alpha} \in I_M} Q_\alpha \prod_{i=0}^{r}(y_i - y_i^*)^{\alpha_i}$.

Thus, we can write

$$f(\overline{x}) = H_{\leq m}[\sum_{\overline{\alpha} \in I_M} Q_\alpha \prod_{i=0}^{r}(C_i(\overline{x}) - C_i(0))^{\alpha_i}].$$

As the constant term of $C_i(\overline{x}) - C_i(0)$ is zero, if we consider $\prod_{i=1}^{r}(C_i(\overline{x}) - C_i(0))^{\alpha_i}$ for some $\overline{\alpha}$ with $\sum_i \alpha_i > m$ then we will get monomials of degree more than m whose net contribution to $f(\overline{x})$ must be zero. Hence we can write $f(\overline{x})$ as

$$f(\overline{x}) = H_{\leq m}[\sum_{\overline{\alpha} \in I_m} Q_\alpha \prod_{i=0}^{r}(C_i(\overline{x}) - C_i(0))^{\alpha_i}],$$

where $I_m = \{(\alpha_0, \alpha_1, \cdots, \alpha_r) \mid \alpha_i \in \mathrm{N}, \sum \alpha_i \leq m\}$. Clearly, $|I_m| \leq (m + r)^r$. Now, the polynomial $\prod_{i=0}^{r}(y_i - y_i^*)^{\alpha_i}$ has a simple $O(1)$-width circuit C'. We can compute $\prod_{i=0}^{r}(C_i(\overline{x}) - C_i(0))^{\alpha_i}$ by plugging in the staggered width $w + O(1)$ circuit for $C_i(\overline{x})$ (as obtained in Lemma 5) where y_i is input to C'. Thus, we obtain a circuit of width $w + O(1)$ for $\sum_{\overline{\alpha} \in I_m} Q_\alpha \prod_{i=0}^{r}(C_i(\overline{x}) - C_i(0))^{\alpha_i}$ that is of size polynomial in s and $(m + r)^r$. By Lemma 4 we can compute its homogeneous components and their partial sums with constant increase in width. Putting it together, it follows that $f(\overline{x})$ can be computed in width $w + O(1)$ of size polynomial in s and $(m + r)^r$. ■

We apply Theorem 6 to prove the main result of this section.

Theorem 7. *There is a constant $c_1 > 0$ so that the following holds. Suppose there is an explicit sequence of multilinear polynomials $\{P_m\}_{m>0}$ where $P_m(\overline{x}) \in \mathbb{F}[x_1, \cdots, x_m]$ and P_m cannot be computed by arithmetic circuits of width $w + c_1$ and size 2^{m^ϵ}, for constants $w \in \mathbb{Z}^+$ and $\epsilon > 0$. Then, for any constant $c_2 > 0$, there is a deterministic $2^{(\log n)^{O(1)}} \cdot b^{O(1)}$ time algorithm that, when given as input a circuit C of size $n^{O(1)}$ and width w computing a polynomial $f(x_1, x_2, \ldots, x_n)$ of maximum coefficient size b, with each variable of individual degree at most $(\log n)^{c_2}$, checks if the polynomial computed by C is identically zero, assuming that the field $\mathbb{F}$ is sufficiently large and $char(\mathbb{F}) > (\log n)^{c_2}$.*

Proof Sketch. The overall construction is based on the Nisan-Wigderson construction as applied in Impagliazzo-Kabanets [KI03] and Dvir et al [DSY08]. Hence it suffices to sketch the argument.

1. Let $m = (\log n)^{c'(\epsilon, c_2)}$ and $\ell = (\log n)^{c''(\epsilon, c_2)}$ where c'' is suitably larger than c'.
2. Construct the Nisan-Wigderson design $S_1, \cdots, S_n \subset [\ell]$ such that $|S_i| = m$ for each i and $|S_i \cap S_j| \leq \log n$.
3. Consider the polynomial $F(y_1, y_2, \cdots, y_\ell) = C(P_m(\overline{y}|S_1), P_m(\overline{y}|S_2), \cdots, P_m(\overline{y}|S_n))$. For any input $\overline{y} \in \mathbb{F}^\ell$ we can evaluate F by evaluating $P_m(\overline{y}|S_i)$ for each i and then evaluating C on the resulting values. Since the P_m are explicit polynomials and $|S_i|$ has polylog(n) size we can evaluate P_m in time $2^{(\log n)^{O(1)}}$.
4. We test if $F(\overline{y}) \equiv 0$ using a brute-force algorithm based on the Schwartz-Zippel lemma. Consider a finite set $S \subseteq \mathbb{F}$, such that $|S|$ is more than $\deg(F)$. Check if $F(\overline{a}) \equiv 0$ for all $\overline{a} \in S^\ell$ in time $n^{O(\ell)}$. If all the tests returned zero then return $C \equiv 0$ otherwise $C \not\equiv 0$.

The proof of correctness is exactly as in [KI03,DSY08]. Assuming the algorithm fails, after hybridization and fixing variables in C, we get a nonzero polynomial F_2 of the form

$$F_2(\overline{y}|S_{i+1}, x_{i+1}) = F_1(P_m(\overline{y}|S_1 \cap S_{i+1}), P_m(\overline{y}|S_2 \cap S_{i+1}), \cdots, P_m(\overline{y}|S_i \cap S_{i+1}), x_{i+1}).$$

where $F_1(x_1, x_2, \ldots, x_{i+1})$ can be computed by a width w circuit of size poly(n) and $F_2(\overline{y}|S_{i+1}, P_m(\overline{y}|S_{i+1})) \equiv 0$. Note that the multilinear polynomials $P_m(\overline{y}|S_j \cap S_{i+1})$ depend only on $\log n$ variables and hence, they can be computed using brute force width-2 staggered circuits of size $O(n \log n)$. Also, by Lemma 1, we know that F_1 can be computed by a staggered circuit of size poly(n) and width at most $w + 1$. Putting the above circuits together, it is easy to see that F_2 can be computed by a staggered circuit C' of size at most poly(n).$n \log n$ = poly(n) and width $w + O(1)$. Now, by applying Theorem 6 to C', we get a circuit of width $w + O(1)$ to compute P_m contradicting the hardness assumption. ■

Finally, we observe that the following analogue of [KI03, Theorem 4.1] holds for bounded width circuits. The proof is in the appendix.

Proposition 7. *One of the following three statements is false.*

1. NEXP $\subseteq$ P/poly.
2. *The Permanent polynomial is computable by polynomial size width w arithmetic circuit over $\mathbb{Q}$, where w is a constant.*
3. *The identity testing problem for bounded width arithmetic circuits over $\mathbb{Q}$ is in* NSUBEXP.

Acknowledgements

We would like to thank Amir Yehudayoff for pointing out the separation in [RY09] and for many valuable comments.

References

[B80] Michael Ben-Or. Unpublished notes.

[BC92] Michael Ben-Or, Richard Cleve. Computing Algebraic Formulas Using a Constant Number of Registers. *SIAM J. Comput.* 21(1): 54-58 (1992).

[DSY08] Zeev Dvir, Amir Shpilka, Amir Yehudayoff. Hardness-randomness tradeoffs for bounded depth arithmetic circuits. *Proc. Symp. on Theory of Computing,* 2008: 741-748.

[JR09] Maurice Jansen and B.V.Raghavendra Rao. Simulation of arithmetical circuits by branching programs preserving constant width and syntactic multilinearity. In *CSR, 2009.* To Appear.

[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.

[LMR07] Nutan Limaye, Meena Mahajan, and B. V. Raghavendra Rao. Arithmetizing classes around NC1 and L. Technical Report 087, Electronic Colloquium on Computational Complexity (ECCC), 2007. Preliminary version in *STACS 2007*, LNCS vol. 4393 pp. 477488.

[MR08] Meena Mahajan and B. V. Raghavendra Rao. Arithmetic circuits, syntactic multilinearity, and the limitations of skew formulae. In *MFCS*, pages 455 466, 2008.

[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.

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

[RY09] Ran Raz and Amir Yehudayoff. Lower Bounds and Separations for Constant Depth Multilinear Circuits. *Computational Complexity 18(2)*: 171-207, 2009.

[RR97] Alexander A. Razborov, Steven Rudich. Natural Proofs. J. Comput. Syst. Sci. 55(1): 24-35, 1997.

[S80] Marc Snir. On the Size Complexity of Monotone Formulas. *Proc. 7th Intl. Colloquium on Algorithms Languages and Programming,* 1980: 621-631.

[Sz61] G. Szekeres. Fractional iteration of exponentially growing functions. *J. Austral. Math. Soc. 2 (1961/62)*, 301-320.

Appendix

Proof Sketch of Lemma 1

The circuit C' is constructed by showing how to compute, for $i \geq 1$, the polynomials computed in layer $i+1$ of C from the polynomials computed in the ith layer in C in a staggered fashion, using at most w layers of width at most $w+1$. Equivalently, it amounts to designing a straight-line program with $w+1$ registers such that: initially, w of the registers contain the polynomials computed in the w nodes of the i^{th} layer. In the end, w of the $w+1$ registers will contain the polynomials computed at the $i+1^{st}$ layer of C. Note that this is trivial for $i=2$ since all nodes in layer 2 have only leaves as children. For some $i>1$, let the U denote the nodes of C in layer i and V the nodes of C in layer $i+1$.

We define an undirected multigraph G corresponding to layers i and $i+1$ as follows: its vertex set $V(G)$ is U. For each gate $v \in V$ in circuit C that takes inputs $u_1, u_2 \in U$ we include the edge $\{u_1, u_2\}$ in $E(G)$. Notice that if $u_1 = u_2$ we add a self-loop to $E(G)$. Furthermore, if $v \in V$ takes one input as a $u \in U$ and the other inputs is a constant or a variable, then too we add a self-loop at vertex u. Finally, if both inputs to v are constants and/or variables, there is no edge in G corresponding to v. We note some properties of this graph G.

1. We have $|V(G)| \leq w$ and $|E(G)| + |V'| \leq w$, where V' is the set of those nodes in V that take only constants and/or variables as input.
2. Each vertex $u \in V(G)$ corresponds to a polynomial p_u computed at u in the i^{th} layer. Each edge $e \in E(G)$ is defined by some $v \in V$ and it corresponds to the polynomial q_e computed at v. In order to compute the polynomial corresponding to e we need the polynomials corresponding to its end points.

We have $w+1$ registers, w of which contain the polynomials $p_u, u \in U$. Our goal is to compute the polynomials $q_e, e \in E(G)$ using these registers. Using the graph structure of G, we will give an ordering of the edges $E(G)$. If we compute the polynomials q_e in that order then for every q_e computed we will have a free register to store q_e (when we do not need a polynomial p_u for further computation, we can free the register containing p_u).

Thus, what we want to do is compute an ordering of the edges $E(G)$[1] from the vertex set $V(G)$.

We pick edges from $E(G)$ one by one. When $e \in E(G)$ is picked, we delete e from the graph and store q_e in a free register. Crucially, note that when a vertex $u \in V(G)$ becomes isolated in this process the polynomial p_u is not required for further computation and the register containing p_u is freed. Thus, at any point of time in this edge-deletion procedure, the number of registers required is equal to the sum of the number of edges removed from G and the number of *non-isolated* vertices left in G.

The edge picking procedure works as follows. We break G into its connected components $G_1 \cup G_2 \cup \ldots \cup G_s \cup G_{s+1} \cup \ldots \cup G_{s+t}$, where $G_1, G_2, \ldots, G_s$ are the *acyclic* components and $G_{s+1}, \ldots, G_{s+t}$ have cycles. We first compute the edges of G_1, and then those of G_2, and so on. At the end, we compute the polynomials corresponding to the nodes in V'.

Each connected component G_i is processed as follows: if there is an edge e in G_i that is not a cut edge, we pick the edge e and delete it from the graph; otherwise, since every edge of G_i is a cut edge, G_i must be a tree, and in this case, we remove any edge e that is incident to a degree-1 vertex. Proceeding thus, we maintain the invariant that at all points, all but one

[1] We can blur the distinction between vertices and edges and the polynomials they represent.

of the components of G_i are isolated vertices. We can use this to show that the number of registers required at any point in the computation of q_e for $e \in E(G_i)$ is at most $|E(G_i)| + 1$ (in particular, if G_i is acyclic this is at most $|V(G_i)|$).

Putting it all together, we can also show that the maximum number of nodes used in computing the edges of G is bounded by $\max\{|V(G)|, |E(G)| + 1, |E(G)| + |V'|\} \leq w + 1$. Moreover, since at each step the polynomial of some node $v \in V$ is computed, the total number of steps in the straight-line program is at most w. This proves the lemma.

Proof of Proposition 7

The proof follows the same lines as that of [KI03, Theorem 4.1]. A similar result for *bounded-depth* circuits is noted in [DSY08, Section 5]. We give a brief proof sketch. Assume to the contrary that all three statements hold. Following the proof in [KI03], NEXP will collapse to $\mathrm{NP}^{\mathrm{Perm}}$. Hence, it suffices to show $\mathrm{P}^{\mathrm{Perm}} \subseteq \mathrm{NSUBEXP}$ to derive a contradiction (to the nondeterministic time hierarchy theorem). The language Perm consists of all tuples (M, v), where M is an integer matrix and v is the binary encoding of $\mathrm{Perm}(M)$. The NSUBEXP machine will guess a polynomial size, width-w circuit C for the $n \times n$ Permanent polynomial over variables $\{x_{ij} | 1 \leq i, j \leq n\}$. Next, we want to check whether C indeed computes the Permanent polynomial. We can easily obtain a width-w polynomial-sized circuit C_k that computes the permanent of $k \times k$ matrix over variables $\{x_{ij} | 1 \leq i, j \leq k\}$ from circuit C. Next we check whether $B_1 = C_1(x) - x \equiv 0$. For $n \geq k > 1$ check that $B_k = C_k(X^{(k)}) - \sum_{i=1}^{k} x_{1,i} C_{k-1}(X_i^{(k)}) \equiv 0$, where $X^{(k)} = (x_{i,j})_{i,k \in [k]}$ is the $k \times k$ matrix and $X_i^{(k)}$ is a minor obtained by deleting first row and i^{th} column of $X^{(k)}$. It follows that if all the B_i's are identically zero polynomials then C computes the Permanent polynomial. Since C_k has a width-w polynomial size circuit it follows that B_k can be computed by a polynomial-size width $w + O(1)$ circuit. We can now use the assumed deterministic subexponential time algorithm for identity testing of bounded width circuits to check whether each B_k is identically zero. Putting it together, we have $\mathrm{P}^{\mathrm{Perm}} \subseteq \mathrm{NSUBEXP}$.