Electronic Colloquium on Computational Complexity, Report No. 28 (2014)

The Complexity of Two Register and Skew Arithmetic Computation

V. Arvind and S. Raja
The Institute of Mathematical Sciences
C.I.T. Campus
Chennai 600 113, India
{arvind,rajas}@imsc.res.in

Abstract

We study two register arithmetic computation and skew arithmetic circuits. Our main results are the following:

- For commutative computations, we show that an exponential circuit size lower bound for a model of 2-register straight-line programs (SLPs) which is a universal model of computation (unlike width-2 algebraic branching programs that are not universal [AW11]).
- For noncommutative computations, we show that Coppersmith's 2-register SLP model [BOC88], which can efficiently simulate arithmetic formulas in the commutative setting, is not universal. However, assuming the underlying noncommutative ring has quaternions, Coppersmith's 2-register model can simulate noncommutative formulas efficiently.
- We consider skew noncommutative arithmetic circuits and show:
  - An exponential separation between noncommutative monotone circuits and noncommutative monotone skew circuits.
  - We define k-regular skew circuits and show that $(k+1)$-regular skew circuits are strictly powerful than k-regular skew circuits, where $k \leq \frac{n}{\omega(\log n)}$.
  - We give a deterministic (white box) polynomial-time identity testing algorithm for noncommutative skew circuits.

1 Two register arithmetic computations

An *arithmetic circuit* over a field $\mathbb{F}$ and indeterminates $X = \{x_1, x_2, \cdots, x_n\}$ 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 gate of the arithmetic circuit computes a polynomial, in the commutative ring $\mathbb{F}[X]$, by adding or multiplying its input polynomials. The polynomial computed at the output gate is the polynomial computed by the circuit.

If the indeterminates $X = \{x_1, x_2, \cdots, x_n\}$ are noncommuting with no relations between them, then the circuit is called a *noncommutative circuit* and it computes a polynomial in the free *noncommutative ring* $\mathbb{F}\langle X\rangle$.

ISSN 1433-8092

We can view an arithmetic circuit C as a *straight-line program* (called an SLP, for short), which prescribes an order of gate evaluation for C. More precisely, a straight-line program corresponding to circuit C is a topologically sorted listing of the nodes of the DAG defining C. For each internal gate g_i we have an instruction like $g_i := g_j \circ g_k$, where g_j and g_k are the inputs to g_i and $\circ \in \{+, *\}$. Thus, a straight-line program is a list of input variables and scalars followed by a list of assignment statements of the form $x := y \circ z$, where y and z are already computed.

The above notion of straight-line programs (SLPs) can be refined by introducing *registers*. Each instruction $x := y \circ z$ entails that x be stored in a register. The input variables and scalars (which occur only on the right-hand side) of the assignments are freely accessible by the program. This naturally leads to the notion of *bounded register* SLPs, where the bound is the number of registers used by the program. We give a very general definition of bounded register SLPs that allows for models with different computational power.

Definition 1. *A* straight-line program using w registers $R_1, R_2, \ldots, R_w$ *over the field* $\mathbb{F}$ *and indeterminates from* $X = \{x_1, x_2, \ldots, x_n\}$ *consists of a sequence of instructions. Let* $R_i^{(t)}$ *denote the contents of register* R_i *at stage* t*. Then the* t^{th} *instruction in the sequence transforms the tuple of register contents* $(R_1^{(t-1)}, R_2^{(t-1)}, \ldots, R_w^{(t-1)})$ *to* $(R_1^{(t)}, R_2^{(t)}, \ldots, R_w^{(t)})$*, where each* $R_w^{(t)} = f(R_1^{(t-1)}, R_2^{(t-1)}, \ldots, R_w^{(t-1)}, x_1, \ldots, x_n)$*, where the function* f *comes from a fixed set of polynomials.*

In an important paper in the area of arithmetic complexity, Ben-Or and Cleve [BOC92] showed that *three* registers suffice to efficiently simulate arithmetic formulas. The instructions they use in their SLPs are of the form: $R_i^{(t)} = l_1 R_j^{(t-1)} + l_2 R_k^{(t-1)} + R_i$ where l_1, l_2 are affine linear forms in $x_1, ..., x_n$. In general, when we allow instructions of the form

$$R_i^{(t)} = l_1 R_j^{(t-1)} + l_2 R_k^{(t-1)} + l_3 R_i,$$

where the l_i are affine linear forms, the polynomial in $\mathbb{F}[X]$ computed by such 3-register SLPs can be computed as the $(1,1)^{th}$ entry of the product of a sequence of 3×3 matrices, one for each SLP instruction, where these matrices have affine linear forms in the x_i's as entries. This is precisely the width-3 algebraic branching program (ABP) model of arithmetic computation which, by the Ben-Or and Cleve result is equivalent to arithmetic formulas (upto polynomial size).

A natural question is about the power of 2-register computations. Allender and Wang [AW11] have shown that 2-register ABPs (defined similarly via the iterated product of 2×2 matrices with affine linear form entries) are not even universal. Indeed, they show that the quadratic polynomial $x_1x_2 + x_2x_3 + ... + x_{n-1}x_n$ cannot be computed by width-2 ABPs. However, there are interesting 2-register SLP models that merit further investigation. Coppersmith (as mentioned in [BOC88]) has already observed that 2-register SLPs, with instructions of the form: $R_i = R_i + \alpha R_j^2$, $\alpha \in \mathbb{F}$ and $R_i = R_i + l$ where l is a affine linear form, suffice to simulate formulas efficiently if char $\mathbb{F} \neq 2$.

Notice that Definition 1 allows for SLPs that are more powerful than the Ben-Or Cleve model because the general SLPs allow for multiplication of registers and hence can compute polynomials of degree exponential in their size. If we restrict ourselves to polynomials in $\mathbb{F}[X]$ of polynomially bounded degree, then arbitrary SLPs, indeed arbitrary arithmetic circuits of polynomial degree, can be transformed to $n^{O(\log n)}$ size formulas and hence width-3 ABPs.

This motivates us to study, in the present paper, some *universal* 2-register SLP models of computation. These models are universal because we allow a *reset instruction* $R_i = c$ for any $c \in \mathbb{F}$. However, apart from the reset instruction, even if we restrict ourselves to only the ABP-like "skew" instructions $R_i = \ell_1 R_i + \ell_2 R_j$ for affine linear forms ℓ_1 and ℓ_2, the model is surprisingly quite powerful. It is at least as powerful as $\Sigma\Pi\Sigma$ arithmetic circuits and it would be interesting to prove lower bounds for it or to separate it from $\Sigma\Pi\Sigma$ arithmetic circuits. Although we are unable to show such a lower bound, we consider a somewhat weaker 2-register SLP model, which is still universal, and show exponential size lower bounds for a polynomial that has $\Sigma\Pi\Sigma$ circuits of polynomial size.

Model 1

At each time instant t, we allow instructions of the form

1. $R_i^{(t)} = \alpha R_i^{(t-1)} + \beta R_j^{(t-1)}$ where $i, j \in \{1, 2\}$ and $\alpha, \beta \in \mathbb{F}$.
2. $R_i^{(t)} = x R_i^{(t-1)}$ where x is a variable.
3. $R_i^{(t)} = \alpha R_i^{(t-1)} + l$ where l is any affine linear form and $\alpha \in \mathbb{F}$.

Here $R_i^{(t)}$ denotes the contents of register R_i at time t. At each time instant t, we can apply any instruction to R_1 and R_2 simultaneously. Crucially, the instruction (3) gives the power of resetting contents of a register to a nonzero constant. This is a crucial difference from width-2 ABPs (which cannot do such a reset). However, width-3 ABPs can simulate Model 1 efficiently.

Lemma 2. *Model 1 is universal.*

Proof. For any polynomial $P(X) = \sum_m \alpha_m . m$, P can be computed as follows. Using the instruction $R_i = x R_i$ repeatedly a monomial m can be computed and add it (with appropriate coefficient α_m multiplied) to R_j, reset R_i to 1 and repeat. ■

We can show an exponential lower bound for Model 1. However a slight variation in which we allow $R_i^{(t)} = l R_i^{(t-1)}$ where l is any affine linear form is already at least as powerful as $\Sigma\Pi\Sigma$ circuits. We call this variation is Model 2.

Lemma 3. *Model 2 is at least as powerful as $\Sigma\Pi\Sigma$ circuits. I.e., for any $\Sigma\Pi\Sigma$ circuit of size s there is a size $O(s)$ SLP of Model 2 type.*

Proof. Using $R_i = l R_i$, we can compute a product of affine linear forms and using $R_j = R_j + R_i$, accumulate the sum of such products in R_j. By using $R_i^{(t)} = 0 R_i^{(t-1)} + 1$, we can reset R_i's contents to 1. ■

We do not know how to prove lower bounds for Model 2. It is not clear if the partial derivative method can give superpolynomial lower bounds because the SLP structure is sequential and allows the partial derivative space dimension to grow exponentially.

2 A Lower Bound for 2-Register SLP of Model 1

In this section we show an exponential size lower bound for 2-register SLPs (of Model 1) computing an explicit multivariate polynomial over any field $\mathbb{F}$. The instructions allowed in Model 1 are: (1) $R_i = \alpha R_i + \beta R_j$, where $\alpha, \beta \in \mathbb{F}$, (2) $R_i = xR_i$ where x is a variable, and (3) $R_i = \alpha R_i + l$, where l is an affine linear form and $\alpha \in \mathbb{F}$.

Model 1 is universal by Lemma 2. In the absence of instruction (3) (which allows resets $R_i := c$ for any $c \in \mathbb{F}$), the model can be simulated by width-2 arithmetic branching programs (ABP), which are not universal, as shown by Allender and Wang [AW11]. As mentioned earlier, they give a sparse polynomial that is not width-2 ABP computable. In Model 1 there is an SLPs of size $O(td)$ to compute any t-sparse polynomial of degree d. The idea is to compute the monomials, one at a time, in the first register, add it to the second register, and reset the first register to continue.

Consider the polynomial $Q = P_1 + P_2 + ... + P_k$ where $P_i = \Pi_{j=1}^{n}(x_{ij} + y_{ij})$ for each i. We will show an exponential size lower bound for SLPs in Model 1 computing the polynomial Q. Let $V = \{x_{ij}, y_{ij}\}_{i\in[k],j\in[n]}$ denote the set of indeterminates of polynomial Q. Note that $|V| = 2nk$. Each P_i is a homogeneous multilinear polynomial of degree n with 2^n distinct monomials. Furthermore, each P_i has an $O(n)$ size $\Pi\Sigma$ arithmetic circuit, and Q has an $O(nk)$ size $\Sigma\Pi\Sigma$ arithmetic circuit.

Let R_1, R_2 be the two registers used by an SLP computing polynomial Q, and let R_1 be the output register. A rough outline of the argument is as follows: based on the structure of the SLP, we pick some indeterminates of the polynomial Q and set them to 0. We then analyze the resulting SLP to prove lower bound.

Theorem 4. *The polynomial $Q = P_1 + P_2 + \cdots + P_k$, defined above, for a suitably large constant k, requires $2^{\Omega(n)}$ size SLPs in the 2-register SLP of Model 1.*

Proof. Consider an SLP of Model 1 computing polynomial Q that is of *minimal size* s. Let $V = \{x_{ij} | i \in [k], j \in [n]\} \cup \{y_{ij} | i \in [k], j \in [n]\}$ denote the variable set of Q. The SLP consists of a sequence of instructions (of type (1) – (3)). At time $t, 0 \leq t \leq s$ the SLP computes the register contents $R_1^{(t)}$ and $R_2^{(t)}$ simultaneously from $R_1^{(t-1)}$ and $R_2^{(t-1)}$ by permissible instructions. We can assume that the contents of the registers R_1 and R_2 at time 0 is $R_1^{(0)} = R_2^{(0)} = 0$. The final contents of the two registers is $R_1^{(s)}$ and $R_2^{(s)}$ at time instant s, and let $R_1^{(s)}$ be the output of the SLP. Our aim is to lower bound s.

Clearly, at any time t both $R_1^{(t)}$ and $R_2^{(t)}$ contain polynomials of degree at most t in indeterminates from V over $\mathbb{F}$. We define the set of good monomials as $GOOD = \{m \mid m$ has nonzero coefficient in $Q\}$. Every good monomial has degree n and there are $k.2^n$ good monomials in all. Since the SLP is not monotone, the set of good monomials in $R_1^{(t)}$ and $R_2^{(t)}$ need not increase in cardinality with t. However, the SLP satisfies the following two properties:

- At time s the number of good monomials in $R_1^{(s)}$ is $k2^n$.
- The number of good monomials at time $t' > t$ can be more than the number at time t only by first applying $R_i = xR_i$ for some $x \in V$ to some register R_i and then $R_j = \alpha R_1 + \beta R_2$. Taking only linear combinations of R_1 and R_2 cannot increase the set of good monomials. On the other hand, x occurs in exactly 2^{n-1} good monomials in Q. In

other words, at each time step the total number of good monomials in the two registers can increase by at most 2^{n-1}.

Therefore, there must be a time instant $\hat{t} \leq s$ such that: (i) $l2^n \leq |GOOD_{\hat{t}}| \leq (l+1)2^n$ and (ii) $\forall t \geq \hat{t}$ we have $|GOOD_t| \geq l2^n$. We choose l such that $k/8 < l < k/4$. For $t \geq \hat{t}$, for our convenience, we will rename the registers at each time instant so that $R_1^{(t)}$ has at least as many good monomials as $R_2^{(t)}$ (Note that this can be effected by inserting the swap step: $R_1^{(t+1)} = R_2^{(t)}$, $R_2^{(t+1)} = R_1^{(t)}$ which would at most double the size of the SLP).

Therefore, we can assume, $\forall t \geq \hat{t}$, $R_1^{(t)}$ has at least $\frac{l}{2}.2^n$ many good monomials. This means the SLP cannot use the instruction $R_1^{(t+1)} = xR_1^{(t)}$ for all $\forall t \geq \hat{t}$. Otherwise, $R_1^{(t+1)}$ will have fewer than 2^{n-1} many good monomials, contradicting the above property of $\hat{t}$.

Now, the number of good monomials at time $\hat{t}$ is still bounded by $(l+1)2^n$. There are at least $(k-l-1)2^n$ good monomials left to be included in the registers. In order to increase their number, we must apply the instruction $R_2 = xR_2$ for some $x \in V$. Let $t_1 > \hat{t}$ be the first such time instant. We set $x = 0$. Without loss of generality, assume x occurs in the polynomial P_k. We can set all variables in P_k to zero and the polynomial Q becomes $P_1 + P_2 + ... + P_{k-1}$ which the SLP must still compute. Although the good monomials of P_k have disappeared, we notice that the number of good monomials in $R_1^{(t_1)}$ lies between $(\frac{l}{2} - 1)2^n$ and $l.2^n$ and $R_2^{(t_1)} = 0$.

Every monomial of degree $> n$ is called a *bad monomial*. For a nonconstant monomial m let $BAD_{m,t}$ be the set of nonzero bad monomials in $R_1^{(t)}$ for t of the form $m^{i_g}g, g \in GOOD$, where i_g is the highest exponent such that $m^{i_g}g$ occurs in $R_1^{(t)}$.

Since $\frac{k}{8} < l < \frac{k}{4}$, there are at least $k - 2l - 1$ polynomials P_i in $\{P_1, P_2, ..., P_{k-1}\}$ such that $R_i^{(t_1)}$ contains fewer than 2^{n-1} good monomials of P_i. Without loss of generality, let these polynomials be $\{P_1, P_2, ..., P_q\}$ where $q \geq k - 1 - 2l$. Let $Var(P_j)$ denote the variable set of the polynomial P_j.

For the SLP to include new good monomials in R_1 of, say, polynomial $P_j, j \in [q]$, after time t_1, the instruction $R_2 = xR_2$ has to be used for some $x \in Var(P_j)$ to generate new good monomials in R_2 which can then be added to R_1. We can distinguish two possible ways in which good monomials from P_j get added to R_1:

(1) The SLP uses R_2 alone to produce an s-sparse polynomial and adds to R_1. But this can only add s good monomials of P_j to R_1 overall. As the SLP needs to add at least 2^{n-1} good monomials of P_j, using only such instructions would lower bound s by 2^{n-1}.

(2) The content of $R_1^{(t_1)}$ is added to $R_2^{(t_1)}$ and R_2 is multiplied by variables and then added back to R_1. This results in a first time instant $t_2 > t_1$ where the contents of $R_1^{(t_2)}$ and $R_2^{(t_2)}$ are expressible as:

$$R_1^{(t_2)} = \alpha R_1^{(t_1)} + \gamma m(R_1^{(t_1)} + m'P) + Sparse, \quad (1)$$

$$R_2^{(t_2)} = \alpha' R_1^{(t_1)} + \gamma' m(R_1^{(t_1)} + m'P) + Sparse, \quad (2)$$

where m and m' are *nontrivial monomials*, P is some (possibly zero) polynomial, $\gamma \neq 0$, $\gamma' \neq 0$, and $Sparse$ denotes any s-sparse polynomial, and $\alpha \neq 0$ but α' may be 0.

Claim 5. *Since $R_1^{(t_2-1)}$ has at least $(\frac{l}{2}-2)2^n$ many good monomials, the number of bad monomials in $R_1^{(t_2)}$ is at least $(\frac{l}{2}-2)2^n$. More precisely, the set BAD_{m,t_2} is of at least this cardinality.*

Proof. If $P=0$ consider the set $\{mg \mid g \text{ a good monomial of } R_1^{(t_1)}\}$. If mg does not occur in $R_1^{(t_1)}$ then mg is a nonzero bad monomial in $R_1^{(t_2)}$. Otherwise, there is a least index i such that $m^i g$ occurs in $mR_1^{(t_1)}$ and does not occur in $R_1^{(t_1)}$. Thus, there is an $m^{i_g} g$ in BAD_{m,t_2} for each good monomial in $R_1^{(t_1)}$.

Otherwise, if $P \neq 0$, in the polynomial $R_1^{(t_1)} + m'P$ we have at least $(\frac{l}{2}-2)2^n$ good monomials of $R_1^{(t_1)}$ since m' is nontrivial and at most 2^{n-1} of its multiples are good monomials. We can apply the same argument as above with mg for these good monomials g. Thus, we have the claimed number of bad monomials in BAD_{m,t_2}. ■

Now, we claim that the number of bad monomials in R_1 must remain $2^{\Omega(n)} - s$ for the remainder of the SLP which will force s to be exponential in n.

Claim 6. *At any time $t > t_2$ the number of bad monomials in $R_1^{(t)}$ is at least $2^{\Omega(n)} - t$.*

Proof. We will prove this claim by induction on $t \geq t_2$. At time t_2 we have already seen that BAD_{m,t_2} is of size $(\frac{l}{2}-2)2^n$ and the status of the two registers is in Equation 1.

We will show, inductively, that at each time $t > t_2$ there is a monomial m_t such that $BAD_{m_t,t}$ is present in $R_1^{(t)}$ and is of size at least $(\frac{l}{2}-2)2^n - t$.

As induction hypothesis assume for all time instants $t_2 \leq t' < t$ that in $R_1^{(t')}$ the set $BAD_{m_{t'},t'}$ is nonzero and has cardinality at least $(\frac{l}{2}-2)2^n - t'$. We now consider $R_1^{(t)}$ and analyze the possible cases. If $R_1^{(t)}$ is obtained from $R_1^{(t-1)}$ by adding a linear form then the claim follows from the induction hypothesis for $R_1^{(t-1)}$. Now consider the general case:

$$R_1^{(t)} = \mu R_1^{(t-1)} + \nu R_2^{(t-1)}.$$

If $R_2^{(t-1)}$ is $Sparse$ then again the induction assumption for $R_1^{(t-1)}$ implies the claim.

Suppose R_2 is never multiplied by a variable in the time interval (t_2, t). In that case $R_1^{(t)}$ is just a linear combination of $R_1^{(t_2)}$, $R_2^{(t_2)}$ and $Sparse$. This case is already argued in Claim 5. A crucial point here is that this linear combination might undo the computation done at time t_2 and $R_1^{(t)}$ reverts to $R_1^{(t_2-1)} + Sparse$. However, in this case we can easily modify the SLP to obtain a new equivalent SLP whose size is strictly less than s, contradicting the minimality of s.

Otherwise, there is a last time instant t_3 such that $t > t_3 \geq t_2$ when R_2 is multiplied by an indeterminate x. Subsequently, there are only linear combinations with R_1 stored in R_2. Therefore, we can write

$$R_1^{(t)} \quad = \quad \mu' R_1^{(t_3)} + \nu' x R_2^{(t_3)} + Sparse, \tag{3}$$

where $\nu' \neq 0$. Notice that μ' also is not zero because $xR_2^{(t_3)}$ has at most 2^{n-1} good monomials and $R_1^{(t)}$ must have over $(\frac{l}{2}-2)2^n$ good monomials.

Now, let us analyze the contents of $R_2^{(t_3)}$. Suppose R_1 is never added to R_2 at any time instant t_4 in the interval (t_2, t_3). In that case from Equation 1 we can see that $R_1^{(t)}$ is of the form

$$R_1^{(t)} = \beta R_1^{(t_1)} + m(\alpha R_1^{(t_1)} + m'P + \hat{m}\hat{P}) + Sparse,$$

where $\beta \neq 0$, m, m' and P are from Equation 1, $\hat{m}$ is a nontrivial monomial, and $\hat{P}$ is some polynomial. Then, as argued before, $\alpha R_1^{(t_1)} + m'P + \hat{m}\hat{P}$ has at least $(\frac{l}{2} - 2)2^n$ many nonzero good monomials of $R_1^{(t_1)}$ because m' and $\hat{m}$ are nontrivial monomials. It follows, as in proof of Claim 5, that for each such good monomial g there is a $m^i g$ in $BAD_{m,t}$ that remains nonzero in $R_1^{(t)}$.

Otherwise, suppose the last time instant less than t_3 when R_1 is added to R_2 is $t_4 < t_3$. The following two cases arise.

Case 1. Suppose $\alpha' = 0$ in Equation 1. Let $t_5 \in [t_2, t_4)$ be largest time instant such that $R_2^{(t_5)} = y\hat{P} + Sparse$ for some arbitrary variable y and polynomial $\hat{P}$. As $R_2^{(t_2)}$ can be written like this, t_5 does exist.

Let $\rho' R_1^{(t_4)} + \rho'' R_2^{(t_5)}$ be the contents of $R_2^{(t_4+1)}$. Now, the contents of $R_2^{(t_3)}$ and $R_1^{(t_3)}$ can be expressed as follows:

$$R_1^{(t_3)} = \theta R_1^{(t_4)} + (\sum_{i=1}^{k} \rho_i m_i) R_2^{(t_4+1)} + Sparse, \quad (4)$$

$$R_2^{(t_3)} = \rho m_k R_2^{(t_4+1)} + Sparse, \quad (5)$$

where $\rho' \neq 0$ and $\rho \neq 0$. Furthermore, $\theta \neq 0$, $\rho m_k \neq 0$. The monomials m_i divide m_{i+1} for each $i < k$.

Plugging these into Equation 3 for $R_1^{(t)}$ we obtain

$$R_1^{(t)} = \mu'\theta R_1^{(t_4)} + \mu'(\sum_{i=1}^{k} \rho_i m_i + \nu'\rho x m_k) R_2^{(t_4+1)} + Sparse.$$

Notice that $R_2^{(t_4+1)} = \rho' R_1^{(t_4)} + \rho'' R_2^{(t_5)}$ contains at least $(\frac{l}{2} - 2)2^n$ many good monomials, since $R_2^{(t_5)}$ is $y\hat{P} + Sparse$. Now, the monomial xm_k is of higher degree than m_i for all i and $\nu'\rho \neq 0$. Thus, $\mu'(\sum_{i=1}^{k} \rho_i m_i + \nu'\rho x m_k) R_2^{(t_4+1)}$ is nonzero. For any good monomial g in $R_2^{(t_4+1)}$ the monomial $xm_k g$ cannot be cancelled in $\mu'(\sum_{i=0}^{k} \rho_i m_i + \nu'\rho x m_k) R_2^{(t_4+1)}$. If it is cancelled in $R_1^{(t_4)}$, then by the argument in proof of Claim 5 we get BAD_{xm_k,t_4} of size $(\frac{l}{2} - 2)2^n$ in $R_1^{(t)}$. This proves the claim in this case.

Case 2. Suppose $\alpha' \neq 0$ in Equation 1 and that there is no multiplication by variable to R_2 in the interval (t_2, t_4). For, if there was a multiplication, we can choose the last one as t_5 and the argument in the above case will again apply without any modification.

Consequently, both $R_1^{(t_4)}$ and $R_2^{(t_4)}$ are linear combinations of $R_1^{(t_1)}$ and $m(R_1^{(t_1)} + m'P)$. Again, let $R_2^{(t_4+1)}$ be $aR_1^{(t_1)} + bm(R_1^{(t_1)} + m'P)$. As before, Equation 4 holds. Hence, an equation of the form $R_1^{(t)} = \mu'\theta R_1^{(t_4)} + \mu'(\sum_{i=1}^{k} \rho_i m_i + \nu'\rho x m_k) R_2^{(t_4+1)} + Sparse$ holds for $R_1^{(t)}$, where now $R_1^{(t_4)}$ is a linear combination of $R_1^{(t_1)}$ and $m(R_1^{(t_1)} + m'P)$, with the coefficient of $R_1^{(t_1)}$ nonzero in it. Since xm_k is not cancelled in $\sum_{i=1}^{k} \rho_i m_i + \nu'\rho x m_k$ we can show

following the argument of Claim 5 again that $R_1^{(t)}$ will contain many bad monomials of the form $(xm_k)^i g$, where g is a good monomial of $R_1^{(t_1)}$. ■

Putting it together, we conclude that the polynomial Q requires $2^{\Omega(n)}$ size SLPs in the 2-register SLP of Model 1. ■

We note that Model 1 cannot efficiently simulate width-2 ABPs. This can be shown by modifying the proof of Theorem 4.

2.1 Noncommutative 2-register arithmetic computations

We now briefly consider 2-register noncommutative arithmetic computations. Here we are working in the free noncommutative ring $\mathbb{F}\langle X\rangle$ where $X = \{x_1, ..., x_n\}$ is a set of noncommuting free variables and $\mathbb{F}$ is any field. Thus monomials are words in the noncommutative free monoid X^*, and polynomials are $\mathbb{F}$-linear combinations of monomials. Nisan [Nis91] has shown that noncommutative ABPs can simulate noncommutative formulas efficiently. An examination of Ben-or and Cleve [BOC92] result shows that width-3 *noncommutative* ABPs can efficiently simulate noncommutative arithmetic formulas and are, in fact, equivalent to them. This has been observed before (e.g. see [AJS09]). Therefore, it is interesting to examine the power of 2-register noncommutative arithmetic computations. Width-2 noncommutative ABPs are also not universal [AW11]. However, the noncommutative version of Model 1 is universal. We can consider a noncommutative generalization of Model 1 in which we allow both left and right multiplication by an indeterminate: $R_i^t = xR_i^{t-1}$ and $R_i^t = R_i^{t-1}x$.

Lemma 7. *Noncommutative 2-register SLPs of Model 1 type have $O(n)$ size SLPs for the Palindrome polynomial $P_n(x_0, x_1) = \sum_{w\in\{x_0,x_1\}^n} w.w^R$.*

We omit the easy proof of the above lemma. In the noncommutative setting, $Q = P_1 + P_2 + ... + P_k$, where $P_i = \Pi_{j=1}^n(x_{ij} + y_{ij})$, has linear size $\Sigma\Pi\Sigma$ circuits. But by Theorem 4 any 2-register SLP of Model 1 for Q requires exponential size. The palindrome polynomial cannot be computed by polynomial size noncommutative $\Sigma\Pi\Sigma$ circuits [Nis91] but we have linear sized 2-register SLP of Model 1 by Lemma 7.

Corollary 8. *In the noncommutative setting, 2-register SLPs of Model 1 are incomparable to $\Sigma\Pi\Sigma$ circuits (or even ABPs).*

Next, we show that Coppersmith's model [BOC88] is not universal for noncommutative polynomials. Recall that the model is $R_i = R_i + \alpha R_j^2$, $R_i = R_i + l$ where $\alpha \in \mathbb{F}$ and l is a affine linear form.

Proposition 9. *The Coppersmith model is not universal in the noncommutative ring of polynomials $\mathbb{F}x, y$ for any field $\mathbb{F}$.*

Proof. We show that the polynomial xy cannot be computed by this model. We will show by induction on t that the register contents $R_1^{(t)}$ and $R_2^{(t)}$ has the property that both xy and yx have the *same coefficient*.
Base Case: t=1 At time instant 0, let $R_1^{(0)} = c_1$ and $R_2^{(0)} = c_2$ for constants c_1, c_2. Clearly the above property holds at time instant $t = 1$.
Induction Hypothesis: Suppose for all $t' < t$, the monomials xy and yx have the same coefficient in $R_1^{(t')}$ and $R_2^{(t)}$.

Induction Step: Consider $R_1^{(t)}$ and $R_2^{(t)}$. If $R_1^{(t)} = R_1^{(t-1)} + l$ for an affine linear form l, the coefficients of xy and yx in $R_1^{(t)}$ are the same as in $R_1^{(t-1)}$ and hence the property holds. Otherwise, the instruction $R_1^{(t)} = R_1^{(t-1)} + \alpha(R_2^{(t-1)})^2$ is used. We can write

$$R_2^{t-1} = f_{\geq 3} + f_2 + f_1 + f_0,$$

where f_j denotes the homogeneous degree-j component of $R_2^{(t-1)}$. Now, expanding the square $(R_2^{(t-1)})^2$ we have

$$R_2^{(t)} = R_1^{(t-1)} + f'_{>2} + (2f_0f_2 + f_1^2) + 2f_0f_1 + f_0^2,$$

where $f'_{>2}$ denotes the part of $(R_2^{(t-1)})^2$ consisting of terms of degree greater than 2. Notice that the coefficients of xy and yx are identical in the following polynomials:

- In $R_1^{(t-1)}$ by induction hypothesis.
- In f_2 by induction hypothesis for $R_2^{(t-1)}$.
- In f_1^2 (by verification) as f_1 is a linear form.

Thus, the coefficient of xy and yx remain identical in $R_1^{(t-1)} + (2f_0f_2 + f_1^2) + 2f_0f_1 + f_0^2$. This proves the induction step for $R_1^{(t)}$. The same argument applies for $R_2^{(t)}$. ■

However, if we assume the presence of quaternions in the ring $\mathbb{F}\langle x_1, ..., x_n\rangle$ we can show that Coppersmith's 2-register model is universal and even efficiently simulates noncommutative arithmetic formulas.

Lemma 10. *Let $R = \mathbb{F}\langle x_1, ..., x_n, i, j, k\rangle$ be a noncommutative ring where char$\mathbb{F} \neq 2$, and $x_1, ..., x_n$ are free noncommuting variables and i, j, k satisfy the relations: $i^2 = j^2 = k^2 = -1$, $ij = k$,$ji = -k$,$jk = i$,$kj = -i$,$ki = j$,$ik = -j$, $\forall y \in \{i, j, k\}$ $\forall x \in \{x_1, ..., x_n\}$ $yx = xy$. For any noncommuting arithmetic formula of size s we can give an equivalent 2-register SLP of size $s^{O(1)}$.*

Proof. The construction is by induction on the formula size. If the output gate is addition, it is handled in the same way as explained in [BOC88]. We explain the product gate case. Recall that the model is $R_i = R_i + \alpha R_j^2$, $R_i = R_i + l$ where $\alpha \in \mathbb{F}$ and l is a affine linear form. In the original Coppersmith proof for the commutative case [BOC88] the general inductive step is to obtain an offset of the polynomial product $f.g$ assuming we can obtain offsets of f and g. I.e., assuming we have 2-register SLPs that can transform (R_1, R_2) contents from (a, b) to $(f + a, b)$ and (a, b) to $(g + a, b)$, we need to give an SLP that can transform (a, b) to $(fg + a, b)$. As in the Proposition 9, by the "Coppersmith identity":

$$f.g = -\frac{1}{2}.b^2 + \frac{1}{2}(f - b)^2 - \frac{1}{2}(g - f + b)^2 + \frac{1}{2}(g + b^2),$$

we get $(a, b) \to (\frac{gf+fg}{2} + a, b)$, where the noncommutativity of the ring prevents obtaining fg. Instead, we obtain $(a, b) \to (\frac{if+fi}{2} + a, b) = (if + a, b)$, and similarly obtain $(a, b) \to (jg + a, b)$. Now, by applying (1) we can obtain an SLP that transforms

$$(a, b) \to (\frac{ijfg - ijgf}{2} + a, b) = (\frac{kfg - kgf}{2} + a, b).$$

By applying (2) and multiplying by $-k$ we can get: $(a, b) \to (\frac{fg-gf}{2} + a, b)$. We now obtain an offset of $\frac{fg+gf}{2}$ as in (1) to get

$$(a, b) \to (\frac{fg + gf}{2} + \frac{fg - gf}{2} + a, b) = (fg + a, b).$$

■

3 Skew Noncommutative Computation

An arithmetic circuit is *skew* if for every multiplication gate one of its inputs is a scalar or an indeterminate $x_i \in X$. This model of arithmetic circuits has been studied in [AJMV98], especially in connection with depth reduction for noncommutative circuits. Although commutative skew circuits are equivalent to ABPs, in the noncommutative setting, skew circuits are known to be strictly more powerful than noncommutative ABPs [Nis91, AJMV98]. This is because multiplications could be either to the left or right. For instance, consider the palindrome polynomial $P(x_0, x_1) = \sum_{w \in \{x_0, x_1\}^n} ww^R$, where w^R denotes the reverse of w can be computed by an $O(n)$ size noncommutative skew circuit (using both left and right skew multiplications), but requires $2^{\Omega(n)}$ size ABPs as show by Nisan [Nis91].

Remark 11. *It is an interesting question whether we can prove superpolynomial size lower bounds for noncommutative skew circuits computing the noncommutative Permanent or Determinant. A lower bound argument given in [AJMV98, Theorem 7.12] is unfortunately not correct. The idea there was to convert a given skew circuit into a left skew circuit (which is just a noncommutative ABP) by moving all right skew multiplications to the left, and then to apply Nisan's rank argument, to the resulting ABP. However, the modified circuit, in general, does not compute a polynomial weakly equivalent (in the sense of [Nis91]) to the one computed by the original circuit. Recently Limaye, Malod, and Srinivasan [LMS14] have shown a $2^{\Omega(n)}$ lower bound for general noncommutative skew circuits.*

Although our results fall short of proving a lower bound for general noncommutative skew circuits, we show some exponential lower bounds separations. Specifically, we define a natural subclass of skew circuits, which we call k-regular skew circuits, and show exponential separations between k-regular and $k+1$-regular skew circuits for each k, resulting in an infinite hierarchy of separations above noncommutative algebraic branching programs. Indeed, noncommutative ABPs form a proper subclass of 1-regular skew circuits.

However, for general skew circuits we give a deterministic (white-box) polynomial-time identity testing algorithm.

We also compare the power of monotone noncommutative skew circuits with unrestricted noncommutative monotone circuits. Exponential size lower bounds for arbitrary monotone circuits computing the Permanent in both commutative and noncommutative settings are already well known (e.g. see [Nis91] for one proof). Here, we show an exponential separation between noncommutative monotone circuits and noncommutative monotone skew circuits.

Lower bounds for k-regular skew circuits

Let C be a noncommutative skew circuit of size s computing a homogeneous polynomial in $\mathbb{F}\langle X\rangle$, where $X = \{x_1, x_2, \ldots, x_n\}$ are noncommuting free variables. We can first convert C

into a *layered* circuit of size poly(n, s) such that (i) all gates at layer i compute polynomials of degree i, (ii) all edges are between gates at layer i and $i+1$ for each i, (iii) each layer consists of either all $+$ gates or all $\times$ gates and the $+$ and $\times$ gate layers alternate. Furthermore, we let the $+$ gates be of arbitrary fanin. The (skew) multiplication gates have fanin 2 and for scalar multiplications we label the edges by elements from the field. If the polynomial f computed by C is not homogeneous we can compute each homogeneous part of f by a layered skew circuit as described above. Suppose C is a layered skew circuit that computes a homogeneous polynomial $f \in \mathbb{F}\langle X\rangle$ of degree d.

A path ρ from a gate g in layer i to the output gate is said to be an (a, b)-*type* if there are exactly a left-skew multiplications and b right-skew multiplications in ρ, where $a + b = d - i$.

Definition 12. *A layered skew circuit C is said to be* k-regular *if for each multiplication layer i, we can associate a set of types $S_i = \{(c, d) \mid c, d \geq 0 \text{ and } c + d = d - i\}$ such that $|S_i| \leq k$ and for each gate g in layer i, if there is a path of (a, b)-type from gate g, where $a + b = d - i$, then $(a, b) \in S_i$.*

Note that in a k-regular skew circuit, the number of gates in each layer can be unbounded. Furthermore, the type sets S_i are not fixed and can depend on the computed polynomial. Of course the set S_i and the circuit structure between the i^{th} and $i + 1^{st}$ multiplication layers will determine S_{i+1}. Given a layered skew circuit C we can check if it is k-regular for a given k, and also efficiently compute the minimum k for which C is k-regular.

Remark 13. *We note here that even* 1*-regular skew circuits are strictly more powerful than ABPs. For, ABPs are just* 1*-regular skew circuits in which all multiplications are right skew, and there is an $O(n)$ size* 1*-regular skew circuit (which is also monotone) for the palindrome polynomial which requires exponential size ABPs [Nis91].*

It is convenient to first convert a given noncommutative skew circuit into one which is homogeneous in a specific sense that we call *type homogeneous*. Something more general is known for general noncommutative circuits [HWY10]. This conversion can be carried out in deterministic polynomial time.

Type homogenization of skew circuits

Let C be a noncommutative skew circuit of size s computing a polynomial in $\mathbb{F}\langle X\rangle$, where $X = \{x_1, x_2, \ldots, x_n\}$ are noncommuting free variables. We can first convert C into a *layered* circuit of size poly(n, s) such that:

(i) All gates at layer i compute polynomials of degree i,

(ii) All edges are between gates at layer i and $i + 1$ for each i,

(iii) An edge from a gate u in layer i to a gate v layer $i+1$ is labeled by a homogeneous linear form and the symbol l or r (indicating whether the linear form is to be multiplied to the left or right of the polynomial produced at gate u). This product of the linear form and the polynomial at gate u is the contribution of u to v. The polynomial computed at gate v is the sum of contributions over all incoming edges to v from layer i.

Suppose C is a layered skew circuit that computes a homogeneous polynomial $f \in \mathbb{F}\langle X\rangle$ of degree d. A gate g in layer i is said to be an (a, b)-*type gate* if for all paths from g to

the output gate there are exactly a left-skew multiplications and b right-skew multiplications, where $a+b=d-i$. Hence, all monomials at an (a,b)-type gate g are of degree $d-(a+b)$. Furthermore, crucially, each monomial produced at gate g can occupy positions in the interval $[a+1, d-b]$ in a monomial of degree d in the output polynomial.

Notice that the total number of types (a,b) that are possible in layer i is $d-i+1$. We say that g is *type homogeneous* if it is of (a,b) type for some (a,b) and we say that the skew circuit C is type homogeneous if every gate g in it is type homogeneous. Clearly, the output gate is always type homogeneous. Starting from the output gate and proceedings downward in a given layered skew circuit C of size s we can convert it, layer by layer, into a type homogeneous layered skew circuit of size $\text{poly}(d,s)$ size.

Lemma 14. *A layered skew circuit C of size s computing a homogeneous polynomial of degree d in $\mathbb{F}\langle X\rangle$ can be transformed in polynomial time (in s and n) into a layered type homogeneous skew circuit C^T of size poly(d,s) computing f.*

In general the gates in layer i of a type homogeneous skew circuit C can have any of the $d-i+1$ possible types. Notice that, crucially, type homogenization of skew circuit does not alter the set of types of the paths at any layer. Specifically, if C is k-regular then C^T remains k-regular.

Now, we show that for each fixed $k>0$, an exponential size separation between k-regular and $(k+1)$-regular skew circuits. We define the following homogeneous degree $2(k+1)n$ polynomial $P_{i,k}(X,Y,Z)$ where the $2(k+1)n$ variables are partitioned as $X \cup Y \cup Z$ such that $|X|=|Y|=n$ and $|Z|=2kn$.

$$P_{i,k}(X,Y,Z) = \sum_{w \in S} z_1 z_2 ... z_{2(i-1)n}.w.w^R.z_{2(i-1)n+1}.z_{2(i-1)n+2} ... z_{2kn}$$

where $S = \{x_1, y_1\} \times \{x_2, y_2\} \times ... \times \{x_n, y_n\}$.

Theorem 15. *Let $P = \sum_{i \in [k+1]} P_{i,k}(X_i, Y_i, Z_i)$ where variable sets X_i, Y_i, Z_i are disjoint $\forall i \in [k+1]$. Any k-regular skew circuit for P requires size at least 2^n. But there is polynomial sized $(k+1)$-regular skew circuit for P.*

Proof. Let $d=2(k+1)n$ denote the degree of polynomial P and C be a k-regular skew circuit of size s computing P. Applying Lemma 14 we can compute in polynomial time a circuit C^T from C that computes P such that C^T is a type-homogeneous layered circuit.

We show an exponential lower bound for size(C^T) which imply an exponential lower bound for size(C). Consider the first multiplication layer of C^T. Let the k possible gate types at this layer be $(a_1,b_1),(a_2,b_2),\ldots,(a_k,b_k)$, where we know that $a_i+b_i=2(k+1)n-1$. Now, for each $i \in [k+1]$, notice that the monomials of the summand polynomial $P_{i,k}$ has the variables from $X_i \cup Y_i$ located precisely in the interval $[2(i-1)n+1, 2in]$. We say that a gate type (a_j,b_j) touches the summand polynomial $P_{i,k}$ if a_j is in the interval $[2(i-1)n+1, 2in]$. Since the intervals for the $k+1$ summand polynomials are all mutually disjoint, for some $u \in [k+1]$ the summand polynomial $P_{u,k}$ is not touched by any gate type in this layer.

Now, $\forall i \neq u$ if we set the variables in $X_i \cup Y_i \cup Z_i$ to 0 and all the variables in Z_u to 1 in the circuit C^T, we obtain a layered skew circuit, we call this modified circuit also by C^T, computing the polynomial $P'_u = P_{u,k}(X_u, Y_u, \overline{1}) = \sum_{w \in S_u} w.w^R$ where $S_u = \{x_{1,u}, y_{1,u}\} \times \{x_{2,u}, y_{2,u}\} \times ... \times \{x_{n,u}, y_{n,u}\}$.

In the circuit C^T, by the choice of u, the types of all surviving gates in the first multiplication layer is either of the form $(a, 0)$ or $(0, b)$ for some values of a and b. We can collect all gates of type $(a, 0)$ and the paths from them to the output gate into a circuit C_1^T that has only left skew multiplications. Similarly, the gates of type $(0, b)$ and the paths from them to the output gate yield a circuit C_2^T that has only right skew multiplications. The sum of C_1^T and C_2^T computes the polynomial P'_u. Now, clearly both C_1^T and C_2^T can be simulated by noncommutative ABPs of the same size, which means we obtain a noncommutative ABP of size at most s for the palindrome polynomial P'_u, where s is the size of the original k-regular skew circuit for P. By Nisan's rank argument [Nis91], it now follows that size$(C^T) = s = 2^n$.

We can give a polynomial size $(k+1)$-regular skew circuit for P as follows: we can compute each $P_{i,k}$ with an $O(k^2 n)$ size 1-regular skew circuit C_i of $O(kn)$ size, as $P_{i,k}$ is an easy variant of the palindrome polynomial. Thus, overall P can be computed as the sum $\sum_{i=1}^{k+1} C_i$, which is a $(k+1)$-regular skew circuit. ■

Examining the above proof we note that in the definition of polynomial P if we set $k = n$ (hence k is not constant) then the polynomial P can be computed by a polynomial size layered skew circuit but any $\frac{n}{\omega(\log n)}$-regular skew circuit that computes P is of exponential size. Putting it together we have the following.

Corollary 16. *There is an explicit noncommutative polynomial in n variables that has polynomial-size skew circuits but any $\frac{n}{\omega(\log n)}$-regular skew circuit that computes it is of size $n^{\omega(1)}$.*

Remark 17. *Chien and Sinclair [CS07] considered the question of lower bounds for the Determinant polynomial whose entries are 2×2 matrices over a field $\mathbb{F}$, where char $\mathbb{F} \neq 2$. Based on Nisan's rank argument they showed that any ABP computing the order n Determinant, DET_n, whose entries are 2×2 matrices requires $2^{\Omega(n)}$ size.*

As in the proof of Theorem 15, we can show that any k-regular skew circuit computing DET_n can be transformed into an ABP computing the $\frac{n}{k+1} \times \frac{n}{k+1}$ determinant. It now follows from Chien and Sinclair's result that for any constant k, k-regular skew circuits computing DET_n over 2×2 matrices are of size $2^{\Omega(n)}$.

A rank based approach to lower bounds for skew circuits

Let $P \in \mathbb{F}\langle X \rangle$ be a homogeneous noncommutative polynomial of degree d on the variables $X = \{x_1, x_2, \ldots, x_n\}$. For each $0 \leq k \leq d$, we can associate a matrix $M_k(P)$ over the field $\mathbb{F}$ with n^k rows and $(d-k+1)n^{(d-k)}$ columns. Each row of $M_k(P)$ is labeled by a distinct monomial m of degree k. Each column is labeled by a pair of monomials (m_1, m_2) such that the sum of their degrees is $d-k$. A monomial $\hat{m}$ of degree d can be factored as $\hat{m} = m_1 m m_2$ in $d-k+1$ different ways, where m is of degree k. The property we demand of the matrix $M_k(P)$ is that the coefficient $P(\hat{m})$ of the monomial $\hat{m}$ in P can be written as

$$P(\hat{m}) = \sum_{\hat{m}=m_1 m m_2} M_K(m, (m_1, m_2)), \tag{6}$$

where m is a degree k monomial.

Remark 18. *Note that unlike for ABPs, the matrix $M_k(P)$ is not uniquely defined for the polynomial P. In a skew circuit a monomial $\hat{m}$ can occupy more than one entry in $M_k(P)$,*

but we require that these entries sum to the coefficient of $\hat{m}$. In particular, it is clear that in a skew circuit computing a homogeneous degree d polynomial, each monomial can occupy $O(d)$ nonzero entries in $M_k(P)$.

However, we can still relate the minimum size of a skew circuit computing a homogeneous degree d polynomial P to the rank of $M_k(P)$. Let $S(P)$ denote the minimum size of a layered skew circuit computing the polynomial P. Similar to Nisan's [Nis91] ABP size characterization we have the following.

For $0 \leq k \leq d$, let $rank_k(P)$ denote the *minimum rank* attained by a matrix $M_k(P)$ satisfying Equation 6.

Theorem 19. *For any homogeneous polynomial P of degree d, $S(P) \geq \sum_{k=0}^{d} rank_k(P)$.*

Proof. Let C be a minimum size layered skew circuit computing the polynomial P. We define two matrices L_k and R_k as follows. Let $v_1, v_2, ..., v_{t_k}$ be the gates in the k^{th} multiplicative layer of circuit C. The rows of matrix L_k are labeled by distinct monomial of degree k and columns by $v_1, v_2, ..., v_{t_k}$. The entry $L_k[m, v_i]$ is defined as the coefficient of monomial m in the polynomial computed at the gate v_i in C. The rows of R_k are labelled $v_1, v_2, ..., v_{t_k}$ and columns by a pair (m_1, m_2) of monomials whose degrees sum to $d - k$. The entry $R_k[v_i, (m_1, m_2)]$ is defined as follows: for every path in C from the gate v_i to the output gate that appends m_1 as prefix by left multiplication and m_2 as suffix by right multiplication compute the product of the scalars on these multiplication edges to obtain a scalar, and them sum up these scalars over all such paths.

If follows by construction that we can define $M_k(P)$ to be the matrix $L_k R_k$, as it satisfies Equation 6. The claim now follows, since for each $0 \leq k \leq d$, we have $t_k \geq rank(L_k) \geq min\{rank(L_k), rank(R_k)\} \geq rank(M_k(P))$ and $\sum_{k=0}^{d} t_k$ is the size of the skew circuit C. ■

Theorem 19 seems difficult to use for general skew circuits as we need to lower bound $rank_k(P)$. However, we can use it for the monotone case for the following reason. Suppose $P \in \mathbb{R}\langle X \rangle$ is a monotone noncommutative homogeneous polynomial of degree d and C is a monotone layered skew circuit computing P. As the circuit C is monotone, the matrices L_k and R_k in the proof of Theorem 19 will both be monotone. Let $mrank_k(P)$ denote the minimum rank of a *monotone* matrix $M_k(P)$ corresponding to P and let $mS(P)$ denote the minimum size of a monotone skew circuit for P. It follows from the proof of Theorem 19 that

$$mS(P) \geq \sum_{k=0}^{d} mrank_k(P). \tag{7}$$

Let $Q = P_1(x_1, x_2)P_2(x_3, x_4)$ where P_i are the palindrome polynomials of degree $2n$. Clearly, Q has polynomial size monotone noncommutative circuits: we can compute P_1 and P_2 with $O(n)$ size monotone skew circuits and multiply their outputs. We will show that Q requires exponential size monotone skew circuits. Note that Q is a homogeneous degree $4n$ polynomial with 2^{2n} monomials. Let $M_k(Q)$ be a *nonnegative* matrix corresponding to polynomial Q with rows labeled by degree k monomials. By $M_k^{a,b}(Q)$ we mean the submatrix of $M_k(Q)$ with all columns (m_1, m_2) of $M_k(Q)$ such that $|m_1| = a, |m_2| = b$.

Theorem 20. *Let $Q = P_1(x_1, x_2)P_2(x_3, x_4)$, where P_i are palindrome polynomials of degree $2n$ each. Suppose $M_{3n}(Q)$ is a nonnegative matrix. Then $rank(M_{3n}(Q)) = 2^{\Omega(n)}$ and hence $mS(Q) = 2^{\Omega(n)}$.*

Proof. Since Q has 2^{2n} monomials, there is an (a, b) such that the submatrix $M_{3n}^{a,b}(Q)$ has at least $\frac{2^{2n}}{n+1}$ non-zero entries. Fix such a submatrix $M_{3n}^{a,b}(Q)$. We will lower bound its rank. The number of nonzero columns in $M_{3n}^{a,b}(Q)$ is $\geq \frac{2^{2n}}{(n+1)2^n}$. Since $a + b = n$, we can assume without loss of generality $a \geq \frac{n}{2}$. There are 2^n columns in $M_{3n}^{a,b}(Q)$ and columns are labelled by monomial pairs (m_1, m_2) such that m_1 has degree a and m_2 has degree b. We consider a column for each m_1. In the (m_1, m_2) column, the number of nonzero entries is at most 2^n because of the way Q is defined. For a fixed m_1, the number of columns (m_1, m_2) is $2^b \leq 2^{\frac{n}{2}}$. For each m_1, we call the set $\{(m_1, m_2)|m_2 \in \{x_3, x_4\}^b\}$ of columns a block B_{m_1}. Since $M_{3n}^{a,b}(Q)$ has at least $\frac{2^{2n}}{(n+1)}$ nonzero entries, there are a set of columns $\{(m_{i1}, m_{i2}) \mid 1 \leq i \leq N\}$ where $N \geq 2^{\frac{n}{2}-O(\log n)}$ such that the $m_{i1}, 1 \leq i \leq N$ are distinct and each such column is nonzero. We consider the rows labelled by monomials m such that $m_{i1}mm_{i2}$ is a nonzero monomial of Q for some i. The number of such rows is 2^{2n}. For column (m_{i1}, m_{i2}) let S_i be the set of monomials m, labeling rows such that the monomial $m_{i1}mm_{i2}$ is non-zero in Q. We claim that $S_i \cap S_j = \phi$ for $i \neq j$. Suppose otherwise. For different columns (m_{i1}, m_{i2}) and (m_{j1}, m_{j2}), we have ensured $m_{i1} \neq m_{j1}$. If $m \in S_i \cap S_j$ then $m_{i1}mm_{i2}$ and $m_{j1}mm_{j2}$ are both valid monomials of Q. This forces $m = n_1 m_{i1}^R n_2 = n_1 m_{j1}^R n_2$, for some monomials n_1 and n_2, which is not possible as $m_{i1} \neq m_{j2}$. Thus, the columns labeled $\{(m_{i1}, m_{i2}) \mid 1 \leq i \leq N\}$ are linearly independent which proves that $rank(M_{3n}(Q)) = N \geq 2^{\Omega(n)}$. The lower bound for $mS(Q)$ follows from Equation 7. ■

Notice that in the proof of Theorem 15, the upper bounds are by monotone skew circuits. Thus we also have an exponential separation between the size of monotone k-regular skew circuits and monotone $k + 1$-regular skew circuits.

Corollary 21. *There is an exponential size separation between the following monotone circuit classes via explicit monotone noncommutative polynomials: (i) monotone circuits and monotone skew circuits, (ii) monotone skew circuits and k-regular monotone skew circuits, (iii) $k + 1$-regular and k-regular monotone skew circuits.*

4 Identity Testing for Noncommutative Skew Circuits

In this section we give a polynomial-time deterministic identity testing algorithm (white box) for noncommutative skew circuits. This extends a well-known result due to Raz and Shpilka [RS05] who gave a polynomial-time deterministic identity test for noncommutative algebraic branching programs. We use a similar approach.

The Identity Test

Theorem 22. *There is a deterministic algorithm that takes as input a noncommutative skew circuit C of size s computing a polynomial in $\mathbb{F}\langle X\rangle$ and decides if C is identically zero in time polynomial in n and s.*

Proof. Let C be a skew circuit computing a polynomial f of degree at most s. Without loss generality, we can assume C is a type homogeneous skew circuit. For $1 \leq d \leq s$ let f_d denote the homogeneous degree-d part of f. It suffices to describe the identity testing algorithm for C_d computing f_d.

Let L_i be the set of all gates at the layer i. The idea of the algorithm is simple. For each degree i monomial m over X we can associate its coefficient vector v_m at the set of gates $g \in L_i$. The vector v_m is a $|L_i|$-dimensional vector over $\mathbb{F}$ whose coordinates are indexed by the gates $g \in L_i$ from left to right. Since $|L_i| \leq s$, we will keep track of only at most $|L_i|$ pairs (v_m, m) such that the $\mathbb{F}$-linear span of these v_m contains the coefficient vectors of all monomials of degree i at the gate set L_i. Let us denote this set of vector-monomial pairs for L_i by S_i.

Clearly, at layer 1 we can explicitly compute these pairs (v_m, m). We will inductively show that if we are given these sets S_i for layer i, in deterministic polynomial time we can compute the sets S_{i+1} in layer $i+1$. Now, consider any gate v of L_{i+1} (in layer $i+1$).

Let $|L_i| = r'$ and $|L_{i+1}| = r$. Corresponding to the edges from L_i to L_{i+1}, for each variable $x \in X$ we define an $r \times r'$ matrix M_x^L over $\mathbb{F}$ such that, for $v \in L_{i+1}$, the v^{th} row of M_x^L is the vector of coefficients of x on edges from u to v for $u \in L_i$ such that these edges are labelled $(L, \alpha_{uv}x)$ for some $\alpha_{uv} \in \mathbb{F}$. Now, for each $(v_m, m) \in S_i$ we collect the vector-monomial pair $(M_x^L v_m, xm)$. Similarly we define M_x^R and for each $(v_m, m) \in S_i$ we collect the vector-monomial pair $(M_x^R v_m, mx)$. ¿From their union over $x \in X$, we pick only those pairs into the set $S_{i+1,c,d}$ such that the corresponding vectors are (maximally) linearly independent. Let $V_{i+1} = \{v_m \mid (v_m, m) \in S_{i+1}\}$.

Claim 23. *For any degree $i+1$ monomial $\hat{m}$ its coefficient vector at the gates in L_{i+1} is in the linear span of the vectors in the set V_{i+1}.*

Proof. Assume the coefficient vector $v_{\hat{m}}$ of $\hat{m}$ is nonzero (otherwise there is nothing to prove). We can write $\hat{m} = xm' = m''y$ for monomials m' and m'' of degree i. By the structure of the circuit C_d, the coefficient vector $v_{\hat{m}}$ of $\hat{m}$ is generated at L_{i+1} is as

$$v_{\hat{m}} = M_x^L v_{m'} + M_y^R v_{m''} \tag{8}$$

Now, by induction hypothesis, the vector $v_{m'}$ is in the span of vectors from V_i, and the vector $v_{m''}$ is in the span of vectors from V_i. Since the vector $v_{\hat{m}}$ is given by the linear expression in the above equation 8, and since the vectors in V_{i+1} span the vectors in $\{M_x^L v_m \mid x \in X \text{ and } (v_m, m) \in S_i\} \cup \{M_x^R v_m \mid x \in X \text{ and } (v_m, m) \in S_i\}$, it follows that $v_{\hat{m}}$ is also in the span of V_{i+1} which completes the proof of the claim. ■

Returning to the proof of the theorem, we notice that at the output layer l_d, vectors in V_d are 1-dimensional. If the polynomial computed by C_d is nonzero then at the d^{th} layer we will have a pair (v_m, m) where v_m is a nonzero scalar and is the coefficient of the degree d monomial m in f_d. This completes the overall proof. ■

Acknowledgments. We thank Meena Mahajan and Yadu Vasudev for discussions and comments.

References

[AJMV98] E. Allender, J. Jiao, M. Mahajan, and V. Vinay, *Non-commutative arithmetic circuits: Depth reduction and size lower bounds*, Theor. Comput. Sci. **209** (1998), no. 1-2, 47–86.

[AW11] E. Allender and F. Wang, *On the power of algebraic branching programs of width two*, Proc. 38th International Colloquium on Automata, Languages, and Programming (ICALP), Springer, 2011, pp. 736–747.

[AJS09] V. Arvind, P. S. Joglekar, and S. Srinivasan, *Arithmetic circuits and the hadamard product of polynomials*, FSTTCS, 2009, pp. 25–36.

[BOC88] M. Ben-Or and R. Cleve, *Computing algebraic formulas with a constant number of registers*, Proceedings of the twentieth annual ACM Symposium on Theory of Computing (STOC), ACM, 1988, pp. 254–257.

[BOC92] M. Ben-Or and R. Cleve, *Computing algebraic formulas using a constant number of registers*, SIAM Journal on Computing **21** (1992), no. 1, 54–58.

[CS07] S. Chien and A. Sinclair, *Algebras with polynomial identities and computing the determinant*, SIAM Journal on Computing **37** (2007), no. 1, 252–266.

[Nis91] N. Nisan, *Lower bounds for non-commutative computation (extended abstract)*, STOC, 1991, pp. 410–418.

[RS05] R. Raz and A. Shpilka, *Deterministic polynomial identity testing in non-commutative models*, Computational Complexity **14** (2005), no. 1, 1–19.

[HWY10] P. Hrubes and A. Wigderson and A. Yehudayoff. *Noncommutative arithmetic circuits and the sum-of-squares problem,* STOC 2010.

[LMS14] N. Limaye, G. Malod, and S. Srinivasan. *Talk at an arithmetic circuits workshop,* Feb, 2014.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de