Arithmetic Circuits, Syntactic Multilinearity, and the Limitations of Skew Formulae

Meena Mahajan and B. V. Raghavendra Rao

The Institute of Mathematical Sciences, Chennai 600 113, India.
{meena,bvrr}@imsc.res.in

Abstract. Functions in arithmetic NC^1 are known to have equivalent constant width polynomial degree circuits, but the converse containment is unknown. In a partial answer to this question, we show that syntactic multilinear circuits of constant width and polynomial degree can be depth-reduced, though the resulting circuits need not be syntactic multilinear. We then focus specifically on polynomial-size syntactic multilinear circuits, and study relationships between classes of functions obtained by imposing various resource (width, depth, degree) restrictions on these circuits. Along the way, we obtain a characterisation of NC^1 (and its arithmetic counterparts) in terms of log width restricted planar branching programs. We also study the power of skew formulae, and show that even exponential sums of these are unlikely to suffice to express the determinant function.

1 Introduction

Among the parallel complexity classes, the class NC^1 of boolean functions computed by logarithmic depth polynomial size circuits has several equivalent characterisations, in the form of bounded width branching programs, polynomial size formulae and bounded width circuits of polynomial size. Its subclass AC^0, consisting of polynomial size constant depth unbounded fan-in circuits, has also been characterised via restricted branching programs.

However, when we consider the counting and arithmetic versions of those classes which are equivalent to NC^1, they seem to represent different classes of functions. In [10], it was shown that if inputs take values from $\{0,1\}$, and only the constants $-1, 0, 1$ are allowed, then counting the total weights of paths in a bounded width branching program is equivalent to the functions computable by log depth polynomial size arithmetic circuits, *i.e.* $\mathsf{GapBWBP} = \mathsf{GapNC}^1$. In [12], this study was extended to bounded width circuits of polynomial degree and size, sSC^0, showing that $\mathsf{GapNC}^1 \subseteq \mathsf{GapsSC}^0$, but it left open the question of equality of these classes.

The question of whether GapsSC^0 is in GapNC^1 can be seen as a depth reduction problem for bounded width circuits. We do not have an answer for this general question. So it is natural to ask if there are any restrictions on the circuit so that depth reduction is possible.

Syntactic multilinearity is a restriction which has been studied in the literature. Syntactic multilinear circuits are those in which every multiplication gate operates on disjoint set of variables. The syntactic multilinear restriction is very fruitful in the sense that there are known unconditional separations and lower bounds for these classes (see [13–15]).

We show that depth reduction for small width circuits is possible if the circuit is syntactic multilinear; however, the depth-reduced circuit may not be syntactic multilinear or even multilinear. The setting we consider is more general than that of [10] and [12]; here the input variables are allowed to take arbitrary values from the underlying ring $\mathbb{K}$. The main result (Theorem 1) is that polynomial size, constant width syntactic multilinear circuits can be simulated (non-uniformly) by log depth bounded fan-in circuits of polynomial size, but this construction need not preserve the syntactic multilinearity property.

Once we take up the restriction of syntactic multilinearity for these arithmetic circuits, it is worthwhile to explore the relationships among the syntactic multilinear arithmetic circuit classes close to arithmetic NC^1.

In the model of branching programs, syntactic multilinearity is a well-studied notion, referred to as read-once branching programs (see *e.g.* [6]). There are several known lower bounds for syntactic multilinear branching programs.

For formulae, syntactic multilinearity is defined exactly as for circuits. A careful observation of the depth reduction for poly size arithmetic formula as given in [7] shows that it preserves syntactic multilinearity. Also some of the constructions in [10–12], relating branching programs and formulae, can be shown to preserve syntactic multilinearity.

In [3], the class of bounded depth arithmetic circuits is characterised in terms of a restricted version of grid programs, rGP, of bounded width BWrGP. We observe that this construction can be extended to show a new (non-uniform) characterisation of (arithmetic) NC^1 in terms of restricted planar branching programs of log width LWrGP. In addition, this can be shown to preserve syntactic multilinearity, for arithmetic NC^1 as well as arithmetic AC^0.

We also study the class of polynomial size skew formulas, denoted SkewF. The motivation for this study arises from Valiant's characterisations of the classes VP and VNP (see [18]; also, for more exposure on algebraic complexity theory, the reader is referred to [8, 9]). Valiant proved that every polynomial $p(X) \in \mathsf{VNP}_{\mathbb{K}}$ (where $\mathbb{K}$ is an arbitrary ring), and in particular every polynomial in $\mathsf{VP}_{\mathbb{K}}$, can be written as $p(X) = \sum_{e \in \{0,1\}^m} \phi(X, e)$, where the polynomial ϕ has an arithmetic formula of polynomial size. So we ask if we can prove a similar equivalence in the case of skew circuits. That is, can we write polynomials computed by skew circuits as an exponential sum of polynomials computed by skew formulae? We show that this is highly unlikely, by showing that any polynomial which is expressible as an exponential sum of skew formulae belongs to the class VNC^1.

The existing and new relationships amongst the arithmetic classes (prefix a-) can be seen in Figure 1; Figure 2 shows the corresponding picture for the syntactic multilinear classes (prefix sma-). Our main depth-reduction result straddles the two figures, and along with [12] gives $\mathsf{sma\text{-}sSC}^0 \subseteq \mathsf{a\text{-}NC}^1 \subseteq \mathsf{a\text{-}sSC}^0$.

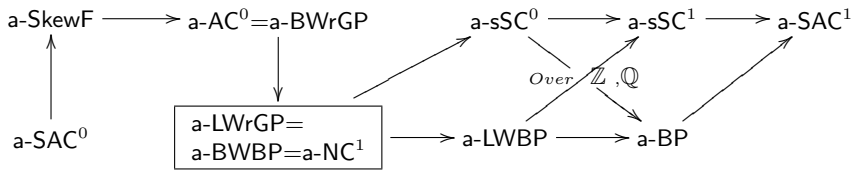


Fig. 1. Arithmetic classes around NC^1

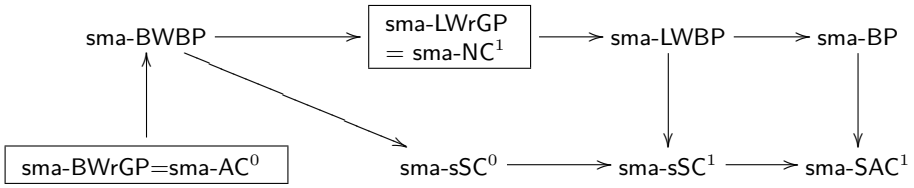


Fig. 2. Relationship among syntactic multilinear classes

The rest of the paper is organised as follows. Section 2 introduces basic definitions. In Section 3 we prove that small-width syntactic multilinear circuits can be depth-reduced. In Section 4, we establish the containments among the syntactic multilinear classes and obtain a new characterisation for NC^1 in terms of a restricted class of grid branching programs. In Section 5 we describe our results concerning skew formulae.

2 Preliminaries

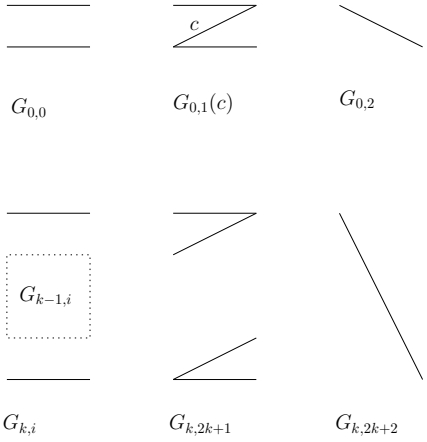
We use standard notation for Boolean circuits and their size, width, depth and degree; see *e.g.* [12],[21]. Unless otherwise stated, fan-in is assumed to be bounded. NC^1 denotes the class of boolean functions which can be computed by boolean circuits of depth $O(\log n)$ and size $\mathsf{poly}(n)$. SC^i denotes the class of boolean functions computed by $\mathsf{poly}(n)$ size circuits of width $O(\log^i n)$. sSC^i is the class of boolean functions computed by $\mathsf{poly}(n)$ degree, $\mathsf{poly}(n)$ size circuits of width $O(\log^i n)$. SAC^i denotes the class of boolean functions computed by polynomial circuits of size $\mathsf{poly}(n)$ and depth $O(\log^i n)$, where $\vee$ gates can have unbounded fan-in. AC^0 denotes the class of boolean functions which can be computed by unbounded fan-in constant depth boolean circuits of size $\mathsf{poly}(n)$.

A *formula* is a circuit where every non-input gate has fan-out bounded by one. F and LWF denote the set of boolean functions which can be computed by polynomial size formulae of unbounded and log width respectively. Without loss of generality, NC^1, AC^0 and SAC^0 circuits can be assumed to be formulae.

A branching program (BP) is a directed acyclic layered graph with edges labelled from $\{x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n, 0, 1\}$, and with two designated nodes s and t. A BP is said to accept its input if and only if there exists an s-t path, in which every edge label evaluates to 1. A BP can also be viewed as a skew-circuit, *i.e.* a circuit where every $\wedge$ gate has at most one non-circuit input.

Let BWBP and LWBP denote the functions computed by constant width and log width branching programs of polynomial size respectively.

G-graphs are the graphs that have planar embeddings where vertices are embedded on a rectangular grid, and all edges are between adjacent columns from left to right. Let BWGP denote the class of boolean functions accepted by constant width polynomial size branching programs which are G-graphs. In these graphs, the node s is fixed as the left-most bottom node and t is the right-most top node. In [3], a restriction of G-graphs is considered where the width of the grid is a constant, and only certain kinds of connections are allowed between any two layers. Namely, for width $2k+2$, the connecting pattern at any layer is one of the graphs $G_{k,i}$ shown alongside for $0 \leq i \leq 2k+2$. Let BWrGP denote the class of boolean functions accepted by constant width polynomial size branching programs that are restricted G-graphs, and LWrGP the class corresponding to log width polynomial size programs that are restricted G-graphs. (see [3]).



The following proposition summaries the known relationships among the boolean complexity classes defined above; see for instance [21].

Proposition 1 ([20, 4, 16, 11, 3]). $\mathsf{SAC}^1 = \mathsf{Circuit\ Size}, \mathsf{Deg}(\mathsf{poly}(n), \mathsf{poly}(n))$*;*

$$\mathsf{NC}^1 = \mathsf{BWBP} = \mathsf{SC}^0 = \mathsf{sSC}^0 = \mathsf{F} = \mathsf{LWF}; \qquad \mathsf{AC}^0 = \mathsf{BWrGP}$$

An arithmetic circuit over a ring $\langle \mathbb{K}, +, -, \times, 0, 1 \rangle$ is a circuit with internal nodes labelled from $\{\times, +\}$, and leaves labelled by input variables $x_1, \ldots, x_n$ that take values in $\mathbb{K}$ or by one of the constants from $\{-1, 0, 1\}$.

The arithmetic circuit classes corresponding to the above defined boolean classes consist of functions $f : \mathbb{K}^* \to \mathbb{K}$, and are defined as follows.

$$\mathsf{BWBP}[\mathbb{K}] = \{f : \mathbb{K}^* \to \mathbb{K} \mid f = \text{sum of weights of all } s \rightsquigarrow t \text{ paths in a BWBP}\}$$

Here the weight of a path is the product of the labels of edges along the path.

$$\mathsf{NC}^1[\mathbb{K}] = \{f \mid f \text{ has a poly size, } O(\log n) \text{ depth, fan-in 2 circuit.}\}$$

$$\mathsf{sSC}^i[\mathbb{K}] = \left\{f \mid f \text{ has a poly size, } O(\log^i n) \text{ width, } \mathsf{poly}(n) \text{ degree circuit. }\right\}$$

For notational convenience we drop the $\mathbb{K}$ where understood from context to be any (or a specific) ring. To distinguish from the boolean case, we write $\mathcal{C}[\mathbb{K}]$ as a-$\mathcal{C}$. The following proposition summarises the known relationships among the arithmetic classes.

Proposition 2 ([3, 10, 12]). $\mathsf{a\text{-}BWrGP} = \mathsf{a\text{-}AC}^0 \subseteq \mathsf{a\text{-}BWBP} = \mathsf{a\text{-}NC}^1 \subseteq \mathsf{a\text{-}sSC}^0$

Multilinear and syntactic multilinear circuits are as defined in [14]. Let C be an arithmetic circuit over the ring $\mathbb{K}$. For a gate g in C, let $P_g \in \mathbb{K}[X]$ be the

polynomial represented at g. Let $X_g \subseteq X$ denote the set of variables that occur in the sub-circuit rooted at g. We call C *multilinear* if for every gate $g \in C$, P_g is a multilinear polynomial, and *syntactic multilinear* if for every multiplication gate $g = h \times f$ in C, $X_h \cap X_f = \emptyset$.

In the case of formulae, the notion of multilinearity and syntactic multilinearity are (non-uniformly) equivalent. Viewing branching programs as skew-circuits, a multilinear branching program P is one which computes multilinear polynomials at every node, and P is syntactic multilinear if in every path of the program (not just s-to-t paths), no variable appears more than once; *i.e.* the branching program is syntactic read-once.

For any arithmetic complexity class $\mathsf{a}\text{-}\mathcal{C}$, we denote by $\mathsf{ma}\text{-}\mathcal{C}$ and $\mathsf{sma}\text{-}\mathcal{C}$ respectively the functions computed by multilinear and syntactic multilinear versions of the corresponding circuits.

In [15] it is shown that the depth reduction of [19] preserves syntactic multilinearity; thus

Proposition 3 ([15]). *Any function computed by a syntactic multilinear polynomial size polynomial degree arithmetic circuit is in* $\mathsf{sma}\text{-}\mathsf{SAC}^1$.

3 Depth reduction in small width sm-circuits

This entire section is devoted to a proof of Theorem 1 below, which says that a circuit width bound can be translated to a circuit depth bound, provided the given small-width circuit is syntactic multilinear.

Theorem 1. *Let C be a* syntactic multilinear *circuit of length l and width w and circuit degree d, with $X = \{x_1, \ldots, x_n\}$ as the input variables, and constants $\{-1, 0, 1\}$ from the ring $\mathbb{K}$. Then there is an equivalent circuit E of depth $O(w^2 \log l + \log d)$ and size $O(2^{w^2+3w} l^{25w} + 4lwd)$.*

Corollary 1. $\mathsf{sma}\text{-}\mathsf{sSC}^0 \subseteq \mathsf{a}\text{-}\mathsf{NC}^1$.

Corollary 2. $\mathsf{sma}\text{-}\mathsf{Size}, \mathsf{Width}, \mathsf{Deg}(2^{\mathsf{poly}(\log)}, \mathsf{poly}(\log), 2^{\mathsf{poly}(\log)}) \subseteq \mathsf{a}\text{-}\mathsf{Size}, \mathsf{Depth}(2^{\mathsf{poly}(\log)}, \mathsf{poly}(\log))$

We first give a brief outline of the technique used. The main idea is to first cut the circuit C at length $\lceil \frac{l}{2} \rceil$, to obtain circuits A (the upper part) and B (the lower part). Let $M = \{h_1, \ldots, h_w\}$ be the output gates of C at level $\lceil \frac{l}{2} \rceil$. (Note that output gates are at the topmost layer of the circuit.) A is obtained from C by replacing the gates in M by a set $Z = \{z_1, \ldots, z_w\}$ of new variables. Each gate g of A (or B) represents a polynomial $p_g \in \mathbb{K}[X, Z]$, and can also be viewed as a polynomial in $K[Z]$, where $K = \mathbb{K}[X]$. Since A and B are circuits of length bounded by $\lceil \frac{l}{2} \rceil$, if we can prove inductively that the coefficients of the polynomials at the output gates of A and B can be computed by small depth circuits (say $O(w \log(l/2))$, then, since p_g has at most 2^w monomials in variables from Z, we can substitute for the z_i's by the value at the output gate g_i of B (*i.e.* polynomials in $\mathbb{K}[X]$). This requires an additional depth of $O(w)$.

The first difficulty in the above argument can be seen even when $w = O(1)$. Though C is syntactic multilinear, the circuit A need not be multilinear in the new dummy variables from Z. This is because there can be gates which compute large constants from $\mathbb{K}$ (*i.e.* without involving any of the variables), and hence have large degree (bounded by the degree of the circuit). This means that the polynomials in the new variables Z at the output gates of A can have non-constant degree, and the number of monomials can be large. Thus the additional depth needed to compute the monomials will be non-constant; hence the argument fails.

To overcome this difficulty, we first transform the circuit C into a new circuit C', where no gates compute "large" constants in $\mathbb{K}$. Assume without loss of generality that every gate in C has a maximum fan-out of 2. Let $G = \{g \in C \mid \mathsf{leaf}(g) \cap X = \emptyset\}$, where for a gate $g \in C$, we define

$$\mathsf{leaf}(g) = \{h \in C \mid h \text{ is a leaf node in } C, \text{ and } g \text{ is reachable from } h \text{ in } C\}$$

Thus G is exactly the nodes that syntactically compute constants. Now define C' as a new circuit which is the same as C except that for all $g \in G$, we replace the i^{th} $(i = 1, 2)$ outgoing wire of g by a new variable y_{g_i}. Note that the number of such new variables introduced is at most $4lw$. Let $Y = \{y_{g_i} \mid g \in G, 1 \leq i \leq 2\}$. We show that C' is syntactic multilinear in the variables $X \cup Y$.

Lemma 1. *The circuit C' constructed above is syntactic multilinear in the variables $X \cup Y$. Further, C' does not have any constants.*

Next we show, in Lemma 2, how to achieve depth reduction for syntactic multilinear bounded width circuits which have no constants. Then we complete the proof of Theorem 1 by explicitly computing the constants (*i.e.* the actual values represented by variables in Y) computed by the circuit C.

Lemma 2. *Let C' be a width w, length l syntactic multilinear arithmetic circuit with leaves labelled from $X \cup Y$ (no constants). Then there is an equivalent arithmetic circuit C'' of size $O(2^{w^2+3w}l^{25w})$ and depth $O(w^2 \log l)$ which computes the same function as C'.*

To establish lemma 2, we use the intuitive idea sketched in the beginning of the section; namely, slice the circuit horizontally, introduce dummy variables along the slice, and proceed inductively on each part.

Now the top part has three types of variables: circuit inputs X, variables representing constants Y as introduced in Lemma 1, and variables along the slice Z. The variables Z appear only at the lowest level of this circuit. Note that this circuit for the top part is syntactic multilinear in Z as well (because there are no constants at the leaves).

To complete an inductive proof for Lemma 2, we need to show depth reduction for such circuits. We use Lemma 3 below, which tells us that viewing each gate as computing a polynomial in Z, with coefficients from $K = \mathbb{K}[X, Y]$, there are small-depth circuits representing each of the coefficients. We then combine these circuits to evaluate the original circuit.

Formally, let D be a width w, length l, syntactic multilinear circuit, with leaves labelled from $X \cup Y \cup Z$ (no constants), where variables from $Z = \{z_1, \ldots z_w\}$ appear only at the lowest level of the circuit. Let $h_1, \ldots, h_w$ be the set of output gates of D. Let $p_{h_i} \in \mathbb{K}[X, Y, Z]$ denote the multilinear polynomial computed at h_i. Note that p_{h_i} can also be viewed as a polynomial in $K[Z]$, *i.e.* a multilinear polynomial with variables from Z and polynomials from $\mathbb{K}[X, Y]$ as its coefficients; we use this viewpoint below. For $T \subseteq \{1, \ldots, w\}$, let $[p_{h_i}, T] \in \mathbb{K}[X, Y]$ denote the coefficient of the monomial $m_T = \prod_{j \in T} z_j$ in p_{h_i}. The following lemma tells us how to evaluate these coefficients $[p_{h_i}, T]$.

Lemma 3. *With circuit D as above, $\forall h \in \{h_1, \ldots, h_w\}$ and $T \subseteq \{1, \ldots, w\}$, there is a bounded fan-in arithmetic circuit $D^{h,T}$ of size bounded by $2^{w^2+2w} l^{25w}$ and depth $O(w^2 \log l)$, with leaves labelled from $X \cup Y \cup \{0, 1\}$, such that the value computed at its output gate is exactly the value of $[p_h, T]$ evaluated at the input setting to $X \cup Y$.*

Proof Sketch. We proceed by induction on the length l of the circuit D. The base case, when $l = 1$, can be handled appropriately. Assume that the lemma holds for all circuits D' of length $l' < l$ and width w.

Now let D be the given circuit of length l, syntactic multilinear in $X \cup Y \cup Z$, where variables from Z appear only at the lowest level. Let $\{h_1, \ldots, h_w\}$ be the output gates of D. Let $\{g_1, \ldots, g_w\}$ be the gates of D at level $l' = \lceil \frac{l}{2} \rceil$. Denote by A the circuit resulting from replacing gates g_i with new variables z'_i for $1 \leq i \leq w$, and removing all the gates below level l', and denote by B the circuit with $\{g_1, \ldots, g_w\}$ as output gates, *i.e.* gates above the g_i's are removed. We rename the output gates of A as $\{f_1, \ldots, f_w\}$. Both A and B are syntactic multilinear circuits of length bounded by l' and width w, and of a form where the inductive hypothesis is applicable. For $i \in \{1, \ldots, w\}$, p_{f_i} is a polynomial in $K[Z']$ and p_{g_i} is a polynomial in $K[Z]$, where $K = \mathbb{K}[X, Y]$.

Applying induction on A and B, for all $S, Q \subseteq \{1, \ldots, w\}$, $[p_{f_i}, S]$ and $[p_{g_i}, Q]$ have circuits $A^{f_i,S}$ and $B^{g_i,Q}$. Note that $p_{h_i}(Z) = p_{f_i}(p_{g_1}(Z), \ldots, p_{g_w}(Z))$. But due to multilinearity,

$$p_{f_i}(Z') = \sum_{S \subseteq [w]} \left([p_{f_i}, S] \prod_{j \in S} z'_j \right) \qquad p_{g_j}(Z) = \sum_{Q \subseteq [w]} \left([p_{g_j}, Q] \prod_{s \in Q} z_s \right)$$

Using this expression for p_{f_i} in the formulation for p_{h_i}, we have

$$p_{h_i}(Z) = \sum_{S \subseteq [w]} \left([p_{f_i}, S] \prod_{j \in S} p_{g_j}(Z) \right)$$

Hence, we can extract coefficients of p_{h_i} as follows. The coefficient of the monomial m_T, for any $T \subseteq [w]$ in p_{h_i} is given by

$$[p_{h_i}, T] = \sum_{S \subseteq [w]} [p_{f_i}, S] \left(\text{ coefficient of } m_T \text{ in } \textstyle\prod_{j \in S} p_{g_j}(Z) \right)$$

If S has t elements, then the monomial m_T is built up in t disjoint parts (not necessarily non-empty), where the kth part is contributed by the kth polynomial of the form p_g in the above expression. So the coefficient of m_T is the product of the corresponding coefficients. Hence

$$[p_{h_i}, T] = \sum_{S=\{j_1,\ldots,j_t\}\subseteq[w]} \left([p_{f_i}, S] \sum_{\substack{Q_1,\ldots,Q_t: \\ \text{partition of } T}} \prod_{k=1}^{t} [p_{g_{j_k}}, Q_k] \right)$$

We use this expression to compute $[p_{h_i}, T]$. We first compute $[p_{f_i}, S]$ and $[p_{g_j}, Q]$ for all $i, j \in [w]$ and all $S, Q \subseteq [w]$ using the inductively constructed sub-circuits. Then a circuit on top of these does the required combination. Since the number of partitions of T is bounded by w^w, while the number of sets S is 2^w, this additional circuitry has size at most 2^{w^2} (for $w \geq 2$) and depth $O(w^2)$.

We can show that this construction satisfies the required bounds. □

Using Lemma 3, we can establish Lemma 2 and hence Theorem 1.

4 Relationships among syntactic multilinear classes

This section explores the relationships among the syntactic multilinear versions of the arithmetic classes which are related to NC^1.

A classical result from [7] shows that for every arithmetic formula F of size s, there is an equivalent arithmetic formula F' which has depth $O(\log s)$ and size $\mathsf{poly}(s)$. A careful observation of this proof shows that if we start with a syntactic multilinear formula F, then the depth-reduced formula F' is also syntactic multilinear.

Theorem 2. *Every syntactic multilinear formula with n leaves has an equivalent syntactic multilinear circuit of depth $O(\log n)$ and size $O(n)$.*
In particular, $\mathsf{sma\text{-}F} \subseteq \mathsf{sma\text{-}NC^1}$.

It is easy to see that the path-preserving simulation of a constant width branching program by a log depth circuit preserves syntactic multilinearity:

Lemma 4. *For any syntactic multilinear branching program P of width w and size s over ring $\mathbb{K}$, there is an equivalent syntactic multilinear circuit C of depth $O(\log s)$ and size $O(s)$ with fan-in of $+$ gate bounded by w (or alternatively, depth $O(\log w \log s)$ and bounded fan-in).*
In particular, $\mathsf{sma\text{-}BWBP} \subseteq \mathsf{sma\text{-}NC^1}$ *and* $\mathsf{sma\text{-}BP} \subseteq \mathsf{sma\text{-}SAC^1}$.

It is also easy to see that the construction of [11], staggering a small-depth formula into a small-width one, preserves syntactic multilinearity. Thus

Lemma 5. *Let Φ be any sm-formula with depth d and size s. Then there is an equivalent syntactic multilinear formula Φ' of length $2s$ and width d.*
In particular, $\mathsf{sma\text{-}NC^1} \subseteq \mathsf{sma\text{-}LWF}$.

From Lemma 5 and Theorem 2, we have the following equivalence.

Corollary 3. *Over any ring* $\mathbb{K}$,
sma-F= sma-LWF= sma-NC^1= sma-Formula-Depth,Size(log, poly).

A straightforward inductive construction of a branching program from a log depth formula results in a log width BP and preserves syntactic multilinearity. But the reverse containment may not hold. However, by restricting the branching program as in [3], we can obtain a characterisation for a-NC^1 which also preserves syntactic multilinearity. In [3] a characterisation for bounded depth arithmetic circuits in terms of counting number of paths in a restricted version of bounded width grid graphs is presented. We note that the characterisation given in [3] works for bounded depth arithmetic circuits over arbitrary rings, showing that a-BWrGP = a-AC^0. By closely examining the parameters in [3], we obtain a characterisation for a-NC^1 in terms of the restricted version of log width grid branching programs. We also note that these constructions preserve syntactic multilinearity. In the statements and proofs below, we use the notion of alternation-depth: a circuit C has alternation depth a if on every root-to-leaf path, the number of maximal segments of gates of the same type is at most a. Also, for an rGP (and in fact any branching program) P, we denote by $\mathsf{Var}(P)$ the set of variables that appear on some s-to-t path in P. For a formula F, $\mathsf{Var}(F)$ denotes the variables appearing anywhere in the formula F; if h is the root of F, then without loss of generality $\mathsf{Var}(F) = X_h$.

Lemma 6. *Let Φ be an arithmetic formula of size s (i.e. number of wires) and alternation-depth $2d$ over $\mathbb{K}$ and with input variables $X \in \mathbb{K}^n$. Then there is a restricted grid program P of length $s^2 + 2s$ (i.e. the number of edge layers) and width $\max\{2, 2d\}$, where the edges are labelled from $\mathsf{Var}(\Phi) \cup \mathbb{K}$, such that the weighted sum of s-to-t paths in P is equal to the function computed by Φ.*
Further, if Φ is syntactic multilinear, then so is P.

Lemma 7. *Let P be an arithmetic rGP of length l (number of edge layers) and of width $2w + 2$ with variables from $X \in \mathbb{K}$. Then there exists an equivalent arithmetic formula Φ over $\mathbb{K}$, with alternation depth at most $2w+2$, size (number of wires) at most $2l$, and $\mathsf{Var}(\Phi) = \mathsf{Var}(P)$.*
Further, if P is syntactic multilinear, then so is Φ.

Corollary 4. sma-AC^0 = sma-BWrGP.
sma-NC^1 = sma-LWrGP; a-NC^1 = a-LWrGP.

The above construction also holds in the case of boolean circuits, giving

Corollary 5. NC^1 = LWrGP.

Thus we get a characterisation for NC^1 and a-NC^1 in terms of a restricted class of log width polynomial size planar branching programs.

In [5] it is shown that any $O(\log n)$ depth polynomial size formula has an equivalent 3-register straight line program. This proves that a-$\mathsf{NC}^1 \subseteq$ a-BWBP. Can the same be stated for sma-NC^1 and sma-BWBP? It turns out that applying the construction of [5] does not preserve syntactic multilinearity; in fact the resulting program need not even be multilinear.

5 Skew formulae

In this section, we consider a question motivated by the setting of Valiant's algebraic complexity classes defined in [18]. VP is the class of polynomials of polynomial degree, computable by polynomial-sized circuits. Similarly one can define VF, VNC^1, and so on. VNP is the class of polynomials expressible as $p(x_1, \ldots, x_n) = \sum_{e \in \{0,1\}^m} g(X, e)$ where $m \in O(\mathsf{poly}(n))$ and the polynomial g is in VP. Thus, loosely speaking, VNP equals $\sum \cdot \mathsf{VP}$. See [8, 9] for more details.

It is well known that the complexity class NP is equivalent to $\exists \cdot \mathsf{P}$ and in fact even to $\exists \cdot \mathsf{F}$. A similar result holds in the case of Valiant's algebraic complexity classes too. Valiant has shown that $\mathsf{VNP} = \sum \cdot \mathsf{VF}$, and thus the polynomial g in the expression above can be assumed to be computable by a formula of polynomial size and polynomial degree.

Noting that VNP is the class of polynomials which are projection equivalent to the "permanent" polynomial, a natural question arises about the polynomials which are equivalent to determinant. Since the determinant exactly characterises the class of polynomials which are computable by skew arithmetic circuits ([17]), the question one could ask is: can the determinant be written as an exponential sum of partial instantiations of a polynomial that can be computed by *skew formula* of poly size, SkewF? Recall that a circuit is said to be skew if every $\times$ (or $\wedge$) gate has at most one child that is not a circuit input. Skew circuits are essentially equivalent to branching programs. Thus one could ask the related question: since $\mathsf{VP} \subseteq \sum \cdot \mathsf{VP} = \sum \cdot \mathsf{VF}$, can we show that $\mathsf{VSkew} \subseteq \sum \cdot \mathsf{VSkewF}$?

We show that this is highly unlikely. We first give a characterisation of polynomials computed by skew formulae (Lemma 8) in terms of their degree and number of monomials. (As a corollary, this places $\mathsf{a\text{-}SkewF}$ inside $\mathsf{a\text{-}AC}^0$.) We then use this to show that $\sum \cdot \mathsf{VSkewF}$ is in fact contained in VNC^1 (Theorem 3). Thus placing VSkew in $\sum \cdot \mathsf{VSkewF}$ is analogous to the statement that GapL equals GapNC^1, which we believe is quite unlikely. [1]

Lemma 8. *Let $f \in \mathbb{Z}[X]$ have degree d, where m monomials have non-zero coefficients. Then f can be computed by a skew formula Φ of size $O(md)$. Further, if all coefficients in f are bounded by c, then f can be computed by a skew formula Φ' that uses as constants only $-1, 0, 1$ and has size $O(md + mc)$. Conversely, let $f \in \mathbb{Z}[X]$ be computed by a skew formula Φ of size s. Then the degree and number of monomials in f are bounded by s. Further, if $-1, 0, 1$ are the only constants in Φ, then the absolute values of coefficients in f are bounded by s.*

Corollary 6. $\mathsf{a\text{-}SAC}^0 \subset \mathsf{a\text{-}SkewF} \subset \mathsf{a\text{-}AC}^0$.

Theorem 3. *Let $f \in \mathbb{Z}[X]$ be expressible as $f(X) = \sum_{e \in \{0,1\}^m} \phi(X, e)$, where ϕ has a poly size* skew *formula. Then $f \in \mathsf{VNC}^1$. (i.e. , $\sum \cdot \mathsf{VSkewF} \subseteq \mathsf{VNC}^1$.)*

[1] For $\mathcal{C} \in \{\mathsf{SkewF}, \mathsf{NC}^1, \mathsf{BP}\}$, $\mathsf{a\text{-}}\mathcal{C}$ is essentially the same as $\mathsf{V}\mathcal{C}$ except that $\mathsf{V}\mathcal{C}$ allows arbitrary constants from $\mathbb{K}$.

Proof Sketch. Since $\phi(X,Y)$ has a poly size skew formula, by Lemma 8 we know that the number of non-zero monomials in ϕ is bounded by some polynomial $q(n,m)$. Hence the number of non-zero monomials in $\phi(X,Y)|_X$, is also bounded by $q(n,m)$.

For any $\alpha \in \mathbb{N}^n$, consider the monomial $X^\alpha = \prod_{\alpha_i} X_i^{\alpha_i}$. Define the set S_α as $S_\alpha = \{\beta \in \{0,1\}^m \mid X^\alpha Y^\beta$ has a non-zero coefficient $a_{\alpha,\beta}$ in $\phi\}$. Clearly, for each α, $|S_\alpha| \leq q(n,m)$. Since $\phi(X,Y)$ is evaluated only at Boolean settings of Y, we can assume, w.l.o.g., that it is multilinear in Y. Hence

$$\phi(X,Y) = \sum_{\alpha \in \mathbb{N}^n} \sum_{\beta \in \{0,1\}^m} a_{\alpha,\beta} X^\alpha Y^\beta$$

Hence we can show that

$$f(X) = \sum_{\alpha \in \mathbb{N}^n} \left(X^\alpha \sum_{\beta \in S_\alpha} a_{\alpha,\beta} 2^{m-l_\beta} \right)$$

where l_β = number of 1's in the bit vector $\beta \in \{0,1\}^m$.

Now it is easy to see that the above expression can be computed in VNC^1. □

Thus, if the Determinant polynomial is expressible as $\sum.\mathsf{VSkewF}$ then it belongs to VNC^1.

We briefly consider (syntactic) multilinear versions of these classes. From Lemma 8, we know that $\mathsf{a\text{-}SkewF}$ is characterised by polynomials with polynomially many coefficients. The construction yields, for any multilinear polynomial computed by a skew formula, an equivalent skew formula which is syntactic multilinear. Hence the notion of multilinearity and syntactic multilinearity are the same for skew formulae. Since any multilinear polynomial that can be computed by an $\mathsf{a\text{-}SAC}^0$ circuit has a small number of monomials, the containments of corollary 6 hold in the syntactic multilinear case too. Also, note that the polynomial $\prod_i (x_i + y_i)$ is multilinear, and can be computed by a $\mathsf{sma\text{-}AC}^0$ circuit.

Corollary 7. $\mathsf{sma\text{-}SAC}^0 \subset \mathsf{sma\text{-}SkewF} = \mathsf{ma\text{-}SkewF} \subset \mathsf{sma\text{-}AC}^0$.

6 Conclusion

This work came out of an attempt to close the gap in $\mathsf{a\text{-}NC}^1 \subseteq \mathsf{a\text{-}sSC}^0$. We have not been able to do this; we can only show that $\mathsf{sma\text{-}sSC}^0 \subseteq \mathsf{a\text{-}NC}^1$. Can the depth-reduction be pushed to all of $\mathsf{a\text{-}sSC}^0$? At least $\mathsf{ma\text{-}sSC}^0$? Alternatively, can the depth-reduced circuit be made multilinear?

Another unsettled question is to better understand the Boolean containments $\mathsf{NC}^1 = \mathsf{LWrGP} \subseteq \mathsf{LWGP} \subseteq \mathsf{LWBP} \subseteq \mathsf{sSC}^1 \subseteq \mathsf{SC}^1 = \mathsf{L}$. Where exactly does the power of the classes actually jump from NC^1 to L?

Making the constructions described here uniform would also be of interest.

Acknowledgements: The referees' comments are gratefully acknowledged.

References

1. M. Agrawal, E. Allender, and S. Datta. On TC^0, AC^0, and arithmetic circuits. *Journal of Computer and System Sciences*, 60(2):395–421, 2000.
2. E. Allender. Arithmetic circuits and counting complexity classes. In J. Krajicek, editor, *Complexity of Computations and Proofs*, Quaderni di Matematica Vol. 13, pages 33–72. Seconda Universita di Napoli, 2004.
3. E. Allender, A. Ambainis, D. A. Barrington, S. Datta, and H. LêThanh. Bounded depth arithmetic circuits: Counting and closure. In *International Colloquium on Automata, Languages, and Programming ICALP*, ICALP'99, pages 149–158, 1999.
4. D. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, 38(1):150–164, 1989.
5. M. Ben-Or and R. Cleve. Computing algebraic formulas using a constant number of registers. *SIAM J. Comput.*, 21(1):54–58, 1992.
6. A. Borodin, A. A. Razborov, and R. Smolensky. On lower bounds for read-k-times branching programs. *Computational Complexity*, 3:1–18, 1993.
7. R. P. Brent. The parallel evaluation of arithmetic expressions in logarithmic time. In *Complexity of sequential and parallel numerical algorithms (Proc. Sympos., Carnegie-Mellon Univ., Pittsburgh, Pa., 1973)*, pages 83–102. Academic Press, New York, 1973.
8. P. Bürgisser. *Completeness and Reduction in Algebraic Complexity Theory*. Algorithms and Computation in Mathematics. Springer-Verlag, 2000.
9. P. Bürgisser, M. Clausen, and M. Shokrollahi. *Algebraic Complexity Theory*. Springer-Verlag, 1997.
10. H. Caussinus, P. McKenzie, D. Thérien, and H. Vollmer. Nondeterministic NC^1 computation. *Journal of Computer and System Sciences*, 57:200–212, 1998.
11. S. Istrail and D. Zivkovic. Bounded width polynomial size Boolean formulas compute exactly those functions in AC^0. *Information Processing Letters*, 50:211–216, 1994.
12. N. Limaye, M. Mahajan, and B. V. R. Rao. Arithmetizing classes around NC^1 and L. ECCC TR07-087,2007. Preliminary version appeared in STACS 2007.
13. R. Raz. Multi-linear formulas for permanent and determinant are of super-polynomial size. In *STOC*, pages 633–641, 2004.
14. R. Raz. Multilinear-$NC^1 \neq$ multilinear-NC^2. In *FOCS*, pages 344–351, 2004.
15. R. Raz and A. Yehudayoff. Balancing syntactically multilinear arithmetic circuits. *Computational Complexity*, to appear.
16. P. M. Spira. On time hardware complexity tradeoffs for boolean functions. In *Fourth Hawaii International Symposium on System Sciences*, pages 525–527, 1971.
17. S.Toda. Counting problems computationally equivalent to the determinant. Technical Report CSIM 91-07, Dept. Comp. Sci. and Inf. Math., Univ. of Electro-Communications, Tokyo, 1991.
18. L. G. Valiant. Completeness classes in algebra. In *Symposium on Theory of Computing STOC*, pages 249–261, 1979.
19. L. G. Valiant, S. Skyum, S. Berkowitz, and C. Rackoff. Fast parallel computation of polynomials using few processors. *SIAM J. Comput.*, 12(4):641–644, 1983.
20. H. Venkateswaran. Circuit definitions of nondeterministic complexity classes. *SIAM Journal on Computing*, 21:655–670, 1992.
21. H. Vollmer. *Introduction to Circuit Complexity: A Uniform Approach*. Springer-Verlag New York Inc., 1999.