RESOURCE TRADE-OFFS IN SYNTACTICALLY MULTILINEAR ARITHMETIC CIRCUITS

Maurice Jansen, Meena Mahajan,
and B. V. Raghavendra Rao

Abstract. The class of polynomials computable by polynomial size log depth arithmetic circuits (VNC^1) is known to be computable by constant width polynomial degree circuits (VsSC^0), but whether the converse containment holds is an open problem. As a partial answer to this question, we give a construction which shows that *syntactically multilinear* circuits of constant width and polynomial degree can be depth-reduced, which in our notation shows that $\mathsf{sm\text{-}VsSC}^0 \subseteq \mathsf{sm\text{-}VNC}^1$. We further strengthen this inclusion, by giving a separate construction that provides a width-efficient simulation for constant width syntactically multilinear circuits by constant width syntactically multilinear algebraic branching programs (In our notation: $\mathsf{sm\text{-}VsSC}^0 \subseteq \mathsf{sm\text{-}VBWBP}$).
We then focus on polynomial-size syntactically 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 also observe a characterisation of the class NC^1 in terms of a restricted class of planar branching programs of polynomial size. Finally, in contrast to the general case, we report closure and stability of coefficient functions for the syntactically multilinear classes studied in this paper.

Keywords. Arithmetic Circuits, Valiant's Classes, Syntactic Multilinearity, Circuit Width, Algebraic Branching Programs.

Subject classification. 68Q05, 03D15, 68Q15.

1. Introduction

The class NC^1 is defined to be the class of Boolean functions computed by logarithmic depth polynomial size circuits. It has several equivalent characterisations: it coincides with the classes of functions computed by polyno-

$$\begin{tikzcd}
\mathsf{SAC}^0 \arrow[d] & \mathsf{BWBP}{=}\mathsf{NC}^1 \arrow[r] \arrow[d, leftrightarrow] & \mathsf{LWBP} \arrow[r] \arrow[d] & \mathsf{BP} \arrow[d] \\
\mathsf{AC}^0 = \mathsf{BWrGP} \arrow[ur] & \mathsf{sSC}^0{=}\mathsf{SC}^0 \arrow[r] & \mathsf{sSC}^1 \arrow[r] & \mathsf{SAC}^1
\end{tikzcd}$$

Figure 1.1: Boolean complexity classes around NC^1. Arrow ($\rightarrow$) indicates containment.

mial size bounded width branching programs (BWBP), by polynomial size formulas (F), and by bounded width circuits of polynomial size (SC^0). Thus $\mathsf{NC}^1 = \mathsf{BWBP} = \mathsf{F} = \mathsf{SC}^0$. Its subclass AC^0, consisting of Boolean functions computed by polynomial size constant depth unbounded fan-in circuits, has also been characterised via restricted branching programs (see Allender *et al.* (1999)). See Figure 1.1. However, the counting and arithmetic versions of those classes which are equivalent to NC^1 seem to represent different classes of functions. In Caussinus *et al.* (1998), it was shown that if inputs take the values from $\{0, 1\}$, then the class of functions represented as the total weights of paths in a constant width branching program with edge weights from $\{x_1, \ldots, x_n, -1, 0, 1\}$ coincides with the class of functions computable by polynomial size and log-depth arithmetic circuits over $\{+, \times, -1, 0, 1, x_1, \ldots, x_n\}$, *i.e.* $\mathsf{GapBWBP} = \mathsf{GapNC}^1$. In Limaye *et al.* (2010), this study was extended to bounded width circuits of small (polynomial) degree and size, *i.e.* sSC^0, showing that $\mathsf{GapNC}^1 \subseteq \mathsf{GapsSC}^0$, but it is not known whether this containment is strict or not.

In the Boolean world, bounded and polylog width polynomial size circuits accept the language class SC, corresponding to languages accepted in simultaneous polynomial time and polylog space by a sequential machine (see Cook (1979); Johnson (1990)). Similarly, polylog depth polynomial size circuits (NC) correspond to languages accepted in parallel polylog time using polynomially many processors (see Johnson (1990); Vollmer (1999)). Translations between small width and small depth thus reveal connections between efficient small-space algorithms and efficient parallel algorithms. Similar connections hold in arithmetic settings as well.

The question $\mathsf{GapsSC}^0 \overset{?}{\subseteq} \mathsf{GapNC}^1$ can be seen as a depth reduction problem for bounded width arithmetic circuits. In the algebraic model introduced by Valiant (see Bürgisser (2000); Valiant (1982)), where arbitrary constants from the underlying ring or field are allowed, the analogous question is to ask: is the class of constant width polynomial size arithmetic circuits of polynomial syntactic degree (VsSC^0) contained in the class of polynomial size arithmetic formulas? In other words, is $\mathsf{VsSC}^0 \subseteq \mathsf{VNC}^1$? An ideal result would be a bounded width version of the depth reduction given in Valiant *et al.* (1983), where a circuit with size s degree d is depth-reduced to one of depth $\log d$, with $+$ gates having fan-in s ($\times$ gates have fan-in 2). In a bounded-width version, we would need the resulting circuit to have the $+$ fan-in bounded by a function of the width of the original circuit. But it is not clear how this can be achieved. So, one of the natural ways to proceed is to look for restrictions on the circuits where this can be achieved. The main focus of this paper is the restriction of syntactic multilinearity on the arithmetic circuits and branching programs. We show that the classes VsSC^0, VNC^1 and VBWBP behave very differently in the syntactically multilinear world. The latter class corresponds to polynomial size algebraic branching programs of constant bounded width (See Section 2).

A multilinear arithmetic circuit is the one where every gate computes a multilinear polynomial. Syntactic multilinearity (sm for short) is a further restriction on the syntactic structure of a multilinear arithmetic circuit. In a syntactically multilinear circuit, every multiplication gate operates on disjoint sets of variables. (A formal definition is given in Section 2.) Clearly, this implies multilinearity, although the converse is not necessarily true. However in the case of formulas, it is easy to see that every multilinear formula has an equivalent syntactically multilinear formula of the same size (Proposition 2.1 in Raz (2009)). So multilinear and syntactically multilinear formulas coincide. By this, and exploiting the structure provided by syntactic multilinearity, Raz (2009) proved super-polynomial lower bounds for multilinear arithmetic formula computing the permanent or determinant. Later in Raz (2006), it was shown that the multilinear versions of the classes VNC^1 and VNC^2 are different. In Raz & Yehudayoff (2008), the depth reduction technique of Valiant *et al.* (1983) was shown to preserve the property of syntactic multilinearity. In Raz *et al.* (2008) an explicit polynomial is shown to require a size of $\Omega(n^{4/3}/\log^2 n)$ for any syntactic multilinear arithmetic circuit computing it. Motivated by this recent progress, we explore the question $\mathsf{VsSC}^0 \overset{?}{\subseteq} \mathsf{VNC}^1$ through the lens of syntactic multilinearity.

Firstly, we give a depth reduction for constant width syntactically multilinear arithmetic circuits. We show that a polynomial computed by a syntactically multilinear circuit of constant width and polynomial size can also be computed by a syntactically multilinear formula of logarithmic depth and polynomial size (Theorem 3.1 and Corollary 3.2). Thus, if restricted to be syntactically multilinear, then the class VNC^1 is at least as powerful as VsSC^0. Note that even in the syntactically multilinear world, we can unwind circuits into formulas, and so log depth formulas are equivalent to log depth circuits or VNC^1. Using our abbreviated notation, this show that $\mathsf{sm\text{-}VsSC}^0 \subseteq \mathsf{sm\text{-}VNC}^1$. Ironically however, the *known* converse containment $\mathsf{VNC}^1 \subseteq \mathsf{VsSC}^0$ does not seem to translate into the syntactically multilinear world. This is mainly because the only known translation Ben-Or & Cleve (1992) from a log-depth formula into a constant width branching program (and hence an sSC^0 circuit) does not preserve syntactic multilinearity.

By the above result, we have that $\mathsf{sm\text{-}VBWBP} \subseteq \mathsf{sm\text{-}VsSC}^0 \subseteq \mathsf{sm\text{-}VNC}^1$. Exploring these classes further, we obtain a somewhat surprising result. Namely, we show that syntactically multilinear algebraic branching programs of constant width and polynomial size are as powerful as syntactically multilinear circuits of constant width and polynomial size (Theorem 4.1), i.e. $\mathsf{sm\text{-}VsSC}^0 \subseteq \mathsf{sm\text{-}VBWBP}$. Thus the restriction of syntactic multilinearity pulls VsSC^0 down to VBWBP. In order to establish this, we use the equivalence of skew circuits and branching programs, and the notions of a weakly skew circuit (first studied by Toda (1992)) and a multiplicatively disjoint circuit (introduced by Malod & Portier (2008)).

The two results described above give a reversal in the relationships among the three classes VBWBP, VNC^1 and VsSC^0: In the general world, VsSC^0 is the strongest class and the other two are equal and contained in VsSC^0, *i.e.* $\mathsf{VBWBP} = \mathsf{VNC}^1 \subseteq \mathsf{VsSC}^0$. In the syntactically multilinear world, $\mathsf{sm\text{-}VNC}^1$ turns out to be the strongest class, whereas the other two are equal and contained in it, *i.e.* $\mathsf{sm\text{-}VBWBP} = \mathsf{sm\text{-}VsSC}^0 \subseteq \mathsf{sm\text{-}VNC}^1$. This indicates that standard simulations may fail in the syntactically multilinear world, and need to be examined afresh. We do this next, showing that the classic depth-reduction of Brent (1973) works in this setting (Theorem 5.1), as also the divide-and-conquer technique of Savitch converting branching programs to circuits (Lemma 5.2), and the folklore staggering (see Istrail & Zivkovic (1994)) of a small-depth to a small-width circuit (Lemma 5.3). A more recent characterisation of arithmetic AC^0 via restricted planar branching programs, Allender *et al.* (1999), also carries through (Corollary 5.7). In fact, examining this more closely, we obtain a characterisation of Boolean NC^1 as well as VNC^1 via polynomial size branching programs of log width or unbounded width, with the same restricted planarity condition (Corollary 5.8).

Another context in which we study the effect of syntactic multilinearity is the complexity of coefficient functions, first studied in a systematic way in Malod (2007). In general, these functions can be quite hard to compute. However for most syntactically multilinear classes, we show that the coefficient functions are also computable within the same class. Further, exponential sums are also computable within the respective classes. (Theorems 6.5,6.7)

The rest of the paper is organized as follows: In Section 2 we give formal definitions of syntactically multilinear circuits. Section 3 contains a depth reduction for syntactically multilinear constant-width circuits. In Section 4 we give a width preserving simulation of constant width syntactically multilinear circuits by syntactically multilinear algebraic branching programs. In Section 5 we discuss the relationships among syntactically multilinear classes. In Section 6 we discuss the coefficient functions for syntactically multilinear classes.

2. Preliminaries

2.1. Arithmetic Circuits and Algebraic Branching Programs. Let $\mathbb{K}$ be a fixed ring. Let $X = \{x_1, \ldots, x_n\}$ be variables that take values from $\mathbb{K}$. An arithmetic circuit (or a straight line program) over $\mathbb{K}$ is a directed acyclic multi-graph C, where nodes of zero in-degree are called input gates and are labeled from the set $\mathbb{K} \cup \{x_1, \ldots, x_n\}$. The nodes of zero out-degree are called output gates. The remaining nodes of C are labeled from $\{+, \times\}$. Whenever not stated explicitly, we assume that in-degree of every node is bounded by 2. A gate f computes a polynomial p_f in $\mathbb{K}[X]$ which can be defined inductively in a natural way: an input gate computes a polynomial that is a constant or a single-variable monomial; if $f = g \times h$, then $p_f = p_g \times p_h$, where p_g and p_h are polynomials computed by g and h respectively (available by induction); and if $f = g + h$, then $p_f = p_g + p_h$. The set of polynomials computed at its output gates constitute the polynomials computed by a circuit. If the circuit has a single output gate, we simply refer to the polynomial computed by the circuit. In what follows, we may use the same symbol f for representing both the gate and the polynomial represented by it. A polynomial family $(f_n)_{n\geq 0}$ is said to be computed by a circuit family $(C_n)_{n\geq 0}$ if $\forall n \geq 0$, f_n is computed by C_n.

We say that two circuit families are equivalent if they compute the same set of polynomials.

Without loss of generality, we assume that a circuit is layered, *i.e.* there is a partition $V_1, \ldots, V_\ell$ of the non-input gates in the circuits so that all incoming edges of V_i are either from input gates or from V_{i-1}, $1 < i \leq \ell$. Each part V_i in the partition forms a layer of the circuit. The measures of size, depth, width and syntactic degree of a circuit are defined in the same way as in the case of Boolean circuits: size is the number of gates, depth is the length of the longest path from an input gate to an output gate, width is the maximum number of gates per layer, and the syntactic degree $d(f)$ of a gate f is inductively defined to be 1 at an input gate, $\max\{d(g), d(h)\}$ if $f = g + h$, and $d(g) + d(h)$ if $f = g \times h$. The syntactic degree of a gate is an upper bound on the degree of the polynomial computed by that gate.

A *formula* is a circuit where the out-degree of every gate is bounded by 1. A *skew* arithmetic circuit is a circuit in which for every gate $f = g \times h$, either $g \in \mathbb{K} \cup X$ or $h \in \mathbb{K} \cup X$ or both.

An *algebraic branching program* (ABP) over a ring $\mathbb{K}$ is a layered directed acyclic graph G with two designated nodes s (of zero in-degree) and t (of zero out-degree), in which the edges are labeled from $\mathbb{K} \cup X$, where $X = \{x_1, \ldots, x_n\}$. For any s-t path P in G, $\mathsf{weight}(P)$ is defined to be the product of the labels of edges that appear in P. The polynomial f_G computed by G is defined as $\sum_P \mathsf{weight}(P)$, where P ranges over all s-t paths in G. The size of an ABP is the number of nodes in it. Width is the maximum number of nodes at any layer. Length of an ABP is the total number of layers in it. In this paper, we assume (without loss of generality) that the in and out-degrees of every node in the ABP are bounded by a constant.

As in the case of Boolean and counting circuits, a *skew* arithmetic circuit can be transformed into an algebraic branching program and vice versa. See Vinay (1996) for conversions in the Boolean world; the same carry over in the arithmetic world as well. See also Nisan (1991). In fact this transformation increases the width and size by only a constant factor. (The conversion in Nisan (1991) has quadratic blowup in size because there edges are labelled by linear forms.) A series-parallel construction due to Valiant transforms formulas of size s and depth d into ABPs of width $O(d)$ and length $O(s)$.

We now give formal definitions of the Gap classes discussed in the introduction:

Definition 2.1.

$$\mathsf{GapNC}^1 = \left\{ f : \{0,1\}^* \to \mathbb{Z} \;\middle|\; \begin{array}{l} \textit{There exists a family } (C_n)_{n\geq 0} \textit{ of polynomial} \\ \textit{size arithmetic formulas with labels from} \\ \{-1,0,1\} \textit{ such that } \forall x \in \{0,1\}^n,\ f(x) = \\ C_n(x). \end{array} \right\}$$

$$\mathsf{GapsSC}^0 = \left\{ f : \{0,1\}^* \to \mathbb{Z} \;\middle|\; \begin{array}{l} \textit{There exists a family } (C_n)_{n\geq 0} \textit{ of constant} \\ \textit{width arithmetic circuits of polynomial size} \\ \textit{and polynomial syntactic degree with la-} \\ \textit{bels from } \{-1,0,1\} \textit{ such that } \forall x \in \{0,1\}^n, \\ f(x) = C_n(x). \end{array} \right\}$$

$$\mathsf{GapBWBP} = \left\{ f : \{0,1\}^* \to \mathbb{Z} \mid \begin{array}{l} \textit{There exists a family } (B_n)_{n\geq 0} \textit{ of polyno-} \\ \textit{mial size algebraic branching programs} \\ \textit{with labels from } \{-1,0,1\} \textit{ such that} \\ \forall x \in \{0,1\}^n,\ f(x) = B_n(x). \end{array} \right\}$$

The following proposition collects known relationships among the Gap complexity classes:

Proposition 2.2. *Caussinus* et al. *(1998); Limaye* et al. *(2010)* $\mathsf{GapNC}^1 = \mathsf{GapBWBP} \subseteq \mathsf{GapsSC}^0$.

G-graphs are graphs that have planar embeddings where vertices are embedded on a rectangular grid, and all edges are between adjacent columns from left to right. In these graphs, the node s is fixed as the leftmost bottom node and t is the rightmost top node. In Allender *et al.* (1999), 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 represented by one of the graphs $G_{k,i}$ (see figure 2.1) for $0 \leq i \leq 2k+2$. An rGP (short for restricted grid branching program) is an algebraic branching program where the underlying graph is a restricted G-graph of the aforementioned form.

A family of circuits $(C_n)_{n\geq 0}$ is said to be $\mathcal{C}$-uniform if there is an algorithm that uses resources within the complexity class $\mathcal{C}$ and on input 1^n, computes a suitably encoded description of the nth circuit C_n. For the classes considered in this paper, an appropriate notion of uniformity is obtained by letting $\mathcal{C}$ be the class AC^0. However, in the algebraic settings, non-uniform constructions are the norm, so we present our results in a non-uniform framework. Uniform versions are relevant when talking about the counting classes (Gap classes). The interested reader is referred to Barrington *et al.* (1990); Vollmer (1999) for more details concerning uniformity.

2.2. Valiant's Classes. In Valiant's model, a complexity class is a set of families of polynomials $f = (f_n)_{n\geq 0}$, where $f_n \in \mathbb{K}[X_1, \dots X_n]$. In this paper, the prefix V denotes an algebraic class in this model. We now define the different complexity classes that are studied in this model. The classes $\mathsf{VP}, \mathsf{VNP}, \mathsf{VP}_e$, were defined by Valiant (see, for instance, Bürgisser (2000); Malod & Portier (2008)). We define the other classes in a fashion analogous to VP, by placing resource bounds on the circuits computing the polynomials. In general, for a class $\mathcal{C}$ of arithmetic circuits or branching programs, we use the nomenclature $\mathsf{V}\mathcal{C}$ to denote the class of families of polynomials $f = (f_n)_{n\geq 0}$ where f_n has degree polynomial in n, and f can be computed by a circuit family in $\mathcal{C}$.

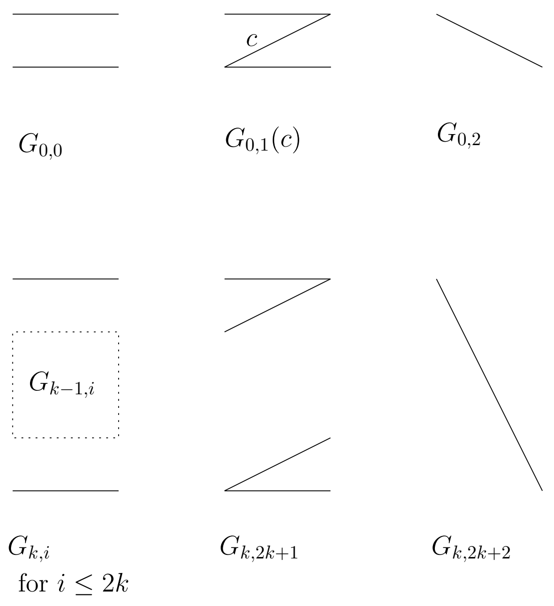


Figure 2.1: The possible patterns between two layers of rGPs. The label c can take a value in $X \cup \mathbb{K}$. Edges without any label are of weight 1.

Definition 2.3.

$$\mathsf{VP} = \left\{ f = (f_n)_{n\geq 0} \;\middle|\; \begin{array}{l} \textit{f_n can be computed by a polynomial size arith-} \\ \textit{metic circuit, and } \deg(f_n) \leq \mathsf{poly}(n). \end{array} \right\}$$

$$\mathsf{VNP} = \left\{ f = (f_n)_{n\geq 0} \;\middle|\; \begin{array}{l} \exists \textit{ a polynomial family } g = (g_m)_{m\geq 0} \in \mathsf{VP} \textit{ such} \\ \textit{that } f_n(X) = \sum_{e\in\{0,1\}^{m'}} g_{m'+n}(X,e), \textit{ where} \\ m', \deg(f_n) \leq \mathsf{poly}(n). \end{array} \right\}$$

$$\mathsf{VF} = \mathsf{VP}_e = \left\{ f = (f_n)_{n\geq 0} \;\middle|\; \begin{array}{l} f \in \mathsf{VP}\textit{; and f_n can be computed by a} \\ \textit{polynomial size arithmetic formula.} \end{array} \right\}$$

$$\mathsf{VP}_{skew} = \left\{ f = (f_n)_{n\geq 0} \;\middle|\; \begin{array}{l} f \in \mathsf{VP}\textit{; and f_n can be computed by a poly-} \\ \textit{nomial size skew arithmetic circuit.} \end{array} \right\}$$

$$\mathsf{VBP} = \left\{ f = (f_n)_{n\geq 0} \;\middle|\; \begin{array}{l} f \in \mathsf{VP}\textit{; and f_n can be computed by a poly-} \\ \textit{nomial size algebraic branching program} \end{array} \right\}$$

$$\mathsf{VBP}[w] = \left\{ f = (f_n)_{n\geq 0} \;\middle|\; \begin{array}{l} \textit{f_n can be computed by a polynomial size} \\ \textit{algebraic branching program of width } O(w) \end{array} \right\}$$

$$\begin{aligned}\mathsf{VLWBP} &= \mathsf{VBP}[\log n] && \log\textit{ width branching programs}\\ \mathsf{VBWBP} &= \mathsf{VBP}[1] && \textit{bounded width branching programs}\end{aligned}$$

$$\mathsf{VNC}^i = \left\{ f=(f_n)_{n\geq 0} \;\middle|\; \begin{array}{l} f \in \mathsf{VP}\textit{; and } f_n \textit{ can be computed by polyno-}\\ \textit{mial size } O(\log^i n) \textit{ depth arithmetic circuits}\\ \textit{of constant fan-in} \end{array} \right\}$$

$$\mathsf{VAC}^0 = \left\{ f=(f_n)_{n\geq 0} \;\middle|\; \begin{array}{l} f \in \mathsf{VP}\textit{; and } f_n \textit{ can be computed by polynomial}\\ \textit{size and constant depth arithmetic circuits with}\\ \textit{unbounded fan-in gates} \end{array} \right\}$$

$$\mathsf{VSAC}^i = \left\{ f=(f_n)_{n\geq 0} \;\middle|\; \begin{array}{l} f \in \mathsf{VP}\textit{; and } f_n \textit{ can be computed by polynomial}\\ \textit{size } O(\log^i n) \textit{ depth arithmetic circuits with con-}\\ \textit{stant fan-in for } \times \textit{ gates and unbounded fan-in for}\\ + \textit{ gates} \end{array} \right\}$$

$$\mathsf{VLWF} = \left\{ f=(f_n)_{n\geq 0} \;\middle|\; \begin{array}{l} f \in \mathsf{VP}\textit{; and } f_n \textit{ can be computed by a polyno-}\\ \textit{mial size arithmetic formula of } O(\log n) \textit{ width} \end{array} \right\}$$

$$\mathsf{VrGP} = \left\{ f=(f_n)_{n\geq 0} \;\middle|\; \begin{array}{l} f \in \mathsf{VP}\textit{; and } f_n \textit{ can be computed by polynomial}\\ \textit{size restricted grid algebraic branching program} \end{array} \right\}$$

$$\mathsf{VBWrGP} = \left\{ f=(f_n)_{n\geq 0} \;\middle|\; \begin{array}{l} f \in \mathsf{VP}\textit{; and } f_n \textit{ can be computed by polyno-}\\ \textit{mial size restricted grid algebraic branching}\\ \textit{program of constant width} \end{array} \right\}$$

$$\mathsf{VsSC}^i = \left\{ f=(f_n)_{n\geq 0} \;\middle|\; \begin{array}{l} f_n \textit{ can be computed by an arithmetic circuit of}\\ \textit{polynomial size and polynomial syntactic degree,}\\ \textit{and width } O(\log^i n). \end{array} \right\}$$

$$\mathsf{VSC}^i = \left\{ f=(f_n)_{n\geq 0} \;\middle|\; \begin{array}{l} f \in \mathsf{VP}\textit{; and } f_n \textit{ can be computed by a polynomial}\\ \textit{size circuit of width } O(\log^i n) \end{array} \right\}$$

Figure 2.2 shows the known relationships among some of the classes defined above. The equivalence between VBWBP and VNC^1 follows from Ben-Or & Cleve (1992); that between VNC^1 and VP_e from Brent (1973). The other containments follow from the definitions.

$$\begin{tikzcd}
\boxed{\begin{array}{l}\mathsf{VBWBP}= \\ \mathsf{VNC}^1=\mathsf{VP}_e\end{array}} \arrow[r] \arrow[dr] & \mathsf{VsSC}^0 \arrow[r] & \mathsf{VSC}^0 \arrow[r] & \mathsf{VSC}^1 \arrow[dr] & \\
 & \mathsf{VLWBP} \arrow[r] \arrow[urr] & \mathsf{VBP} \arrow[rr] & & \mathsf{VP}
\end{tikzcd}$$

Figure 2.2: Relationships among complexity classes in Valiant's model (Arrows denote containment)

2.3. Syntactically Multilinear Circuits. The notion of multilinear circuits was formally introduced by Nisan & Wigderson (1997). We follow the notations from Raz (2009). We call a polynomial p *multilinear*, if for any monomial of p the individual degree of every variable is bounded by one. Let C be an arithmetic circuit over the ring $\mathbb{K}$, and let $X = \{x_1, \ldots, x_n\}$ be its input variables. For a gate g in C, let $p_g \in \mathbb{K}[X]$ be the polynomial computed at g. Let $X_g \subseteq X$ denote the set of variables that occur in the sub-circuit rooted at g. C is called *multilinear* if for every gate $g \in C$, p_g is a multilinear polynomial. C is said to be *syntactically multilinear* if for every multiplication gate $g = h \times f$ in C, $X_h \cap X_f = \emptyset$. Clearly, a syntactically multilinear circuit is also multilinear, though the converse is not necessarily true.

In the case of formulas, the notion of multilinearity and syntactic multilinearity are (non-uniformly) equivalent (Proposition 2.1 in Raz (2009)).

In the case of algebraic branching programs, the notion of syntactic multilinearity coincides with the read-once property, where no variable appears more than once on any path. (See Borodin *et al.* (1993) for more about Boolean read once branching programs). Namely, we say that an algebraic branching program P is multilinear if for every node v in P, the polynomial p_v (sum of weights of all s-to-v paths) is multilinear. Furthermore, P is defined to be syntactically multilinear if in every path of the program (not just s-to-t paths), no variable appears more than once; *i.e.* the algebraic branching program is syntactic read-once.

For any algebraic complexity class $\mathsf{V}\mathcal{C}$, we denote by m-$\mathsf{V}\mathcal{C}$ and sm-$\mathsf{V}\mathcal{C}$ respectively the functions computed by multilinear and syntactically multilinear versions of $\mathsf{V}\mathcal{C}$.

Raz & Yehudayoff (2008) show that the depth reduction give by Valiant *et al.* (1983) preserves syntactic multilinearity. From the fact that a syntactically multilinear circuit has degree $O(n)$, and observing that the construction actually yields a semi-unbounded circuit, we conclude

PROPOSITION 2.4 (Theorem 3.1 in Raz & Yehudayoff 2008). *Any polynomial computed by a syntactically multilinear polynomial size arithmetic circuit is in* $\mathsf{sm\text{-}VSAC}^1$.

3. Depth Reduction in Small Width sm-Circuits

This entire section is devoted to a proof of Theorem 3.1 below, which says that a circuit width bound can be translated to a circuit depth bound, provided the given small-width circuit is syntactically multilinear.

THEOREM 3.1. *Let C be a syntactically multilinear arithmetic circuit of depth ℓ and width w and syntactic degree d, with $X = \{x_1, \ldots, x_n\}$ as the input variables, and constants from the ring $\mathbb{K}$. Then, there is a syntactically multilinear circuit E of depth $O(w^2 \log \ell + \log d)$ and size $O(2^{w^2}\ell^{25w^2} + 4\ell wd)$ computing the same polynomial as C.*

An immediate corollary is,

COROLLARY 3.2. $\mathsf{sm\text{-}VsSC}^0 \subseteq \mathsf{sm\text{-}VNC}^1$.

It can also be seen that if we apply Theorem 3.1 to a syntactically multilinear arithmetic circuit of poly-logarithmic width and quasi-polynomial size and degree, then we get a poly-logarithmic depth circuit of quasi-polynomial size. Thus

COROLLARY 3.3.

$$\mathsf{sm\text{-}Size, Width, Deg}(2^{\mathsf{poly}(\log)}, \mathsf{poly}(\log), 2^{\mathsf{poly}(\log)})$$

$$\subseteq \mathsf{sm\text{-}Size, Width, Depth}(2^{\mathsf{poly}(\log)}, \mathsf{poly}, \mathsf{poly}(\log))$$

where $\mathsf{sm\text{-}Size, Width, Deg}(s, w, d)$ *and* $\mathsf{sm\text{-}Size, Width, Depth}(s, w, d)$ *denote the classes of polynomials computable by syntactically multilinear arithmetic circuits of size s, width w and degree d or depth d respectively.*

We first give a brief outline of the technique used. We actually show something stronger: not only can we compute the polynomials corresponding to gates at the output level, but also, if we express these polynomials as polynomials exclusively in the variables at the lowest level, then we can compute all coefficients of the new polynomials in small depth. (Note that the coefficients are not necessarily ring elements but are themselves polynomials in the

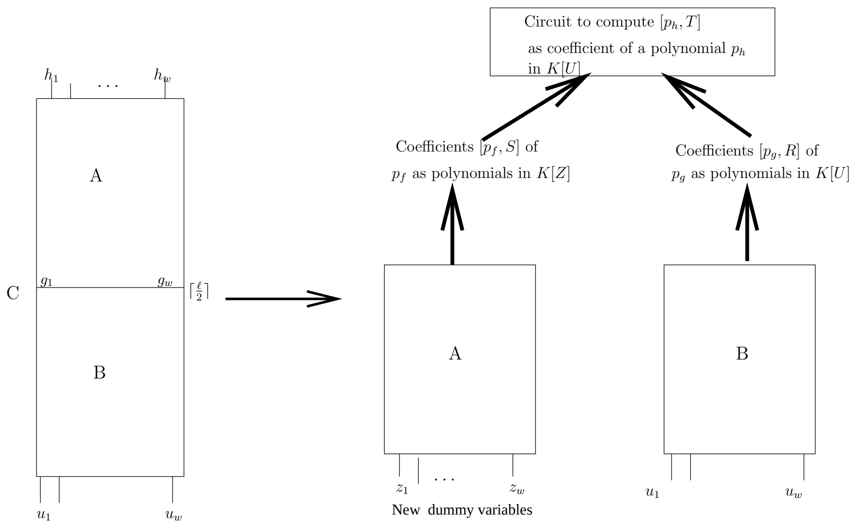

h_1 ... h_w
A
C
g_1 g_w $\lceil\frac{\ell}{2}\rceil$
B
u_1 u_w
A
z_1 ... z_w
New dummy variables
B
u_1 u_w
Coefficients $[p_f, S]$ of p_f as polynomials in $K[Z]$
Coefficients $[p_g, R]$ of p_g as polynomials in $K[U]$
Circuit to compute $[p_h, T]$ as coefficient of a polynomial p_h in $K[U]$

Figure 3.1: Breaking up circuit C into A and B

remaining variables. So when we say we compute the coefficients, we are still computing polynomials and not ring elements.)

To show this stronger claim, we cut the circuit C at depth $\lceil\frac{\ell}{2}\rceil$, to obtain circuits A (the upper part) and B (the lower part). Let $M = \{g_1, \ldots, g_w\}$ be the gates of C at level $\lceil\frac{\ell}{2}\rceil$. 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 depth bounded by $\lceil\frac{\ell}{2}\rceil$, we use induction on each of them. Now we need additional circuitry to patch together the coefficients so computed and obtain the coefficients corresponding to C. See Figure 3.1.

For this approach to work, we need to eliminate constants, since they may violate syntactic multilinearity if replaced by variables at the slice layer. We say that a gate g syntactically computes a constant if each input gate that is a descendant of g is labelled by a constant. Without loss of generality, we can assume that in the circuit C, no gate syntactically computes a constant. If there is such a gate, simply replace it by an input gate with the computed constant. This is a non-uniform step. Alternatively, to preserve uniformity, identify the gates that syntactically compute constants, label all outgoing wires of these gates by new variables from a set Y, and proceed with this new circuit. It is easy to see that the new circuit is syntactically multilinear in $X \cup Y$.

To simplify the following construction, we explicitly relabel each input gate labeled by a constant with a new variable from a set Y.

We now show, in Lemma 3.4, how to achieve depth reduction for syntactically multilinear bounded width circuits which have no constants. This completes the proof of Theorem 3.1 as well; all we need to do is to explicitly plug in the constants corresponding to the variables in Y.

LEMMA 3.4. *Let C' be a width w, depth ℓ syntactically multilinear arithmetic circuit with leaves labeled from $X \cup Y$ (no constants). Then there is an equivalent syntactically multilinear arithmetic formula C'' of size $O(2^{w^2}\ell^{25w^2})$ and depth $O(w^2 \log \ell)$ which computes the same polynomial as C'.*

To establish lemma 3.4, we use the intuitive idea sketched earlier; 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, and variables along the slice Z. The variables Z appear only at the lowest level of this circuit for the top part, which is syntactically multilinear in Z as well.

To complete an inductive proof for Lemma 3.4, we need to show depth-reduction for such circuits. We use Lemma 3.5 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 compute the polynomial from the the original circuit.

More formally, let D be a width w, depth ℓ, syntactically multilinear circuit, with all leaves labeled from $X \cup Y \cup Z$ (no constants), where variables from $Z = \{z_1, \dots z_w\}$ appear only at the lowest level of the circuit. Let $h_1, \dots, h_w$ be the set of output gates of D *i.e.* gates at level ℓ. 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 $T \subseteq \{1, \dots, w\}$, 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 obtain these coefficients $[p_{h_i}, T]$.

LEMMA 3.5. *With circuit D as above, $\forall h \in \{h_1, \dots, h_w\}$ and $T \subseteq \{1, \dots, w\}$, there is a bounded fan-in syntactically multilinear arithmetic formula $D^{h,T}$ of size bounded by $2^{w^2}\ell^{25w^2}$ and depth $O(w^2 \log \ell)$, with leaves labeled from $X \cup Y \cup \{0, 1\}$, such that the polynomial computed at its output gate is exactly the coefficient $[p_h, T]$.*

PROOF. We proceed by induction on the depth ℓ of the circuit.

Basis : $\ell = 1$. The different possibilities are as follows. Here, a can take any value in $X \cup Y \cup \mathbb{K}$.

$h = z_i z_j$:	$[p_h, T] = 1$ for $T = \{i, j\}$ and 0 otherwise.
$h = az_i$:	$[p_h, T] = a$ for $T = \{i\}$ and 0 otherwise.
$h = a$:	$[p_h, T] = a$ for $T = \emptyset$ and 0 otherwise.
$h = z_i + z_j$:	$[p_h, T] = 1$ for $T = \{i\}$ or $T = \{j\}$ and 0 otherwise.
$h = a + z_i$:	$[p_h, \emptyset] = a$, $[p_h, \{i\}] = 1$, and $[p_h, T] = 0$ otherwise.

Hypothesis: Assume that the lemma holds for all circuits D' of depth $\ell' < \ell$ and width w.

Induction Step: Let D be the given circuit of depth ℓ, syntactically multilinear in $X \cup Y \cup Z$, where variables from Z appear only at the lowest level of D. 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 $\ell' = \lceil \frac{\ell}{2} \rceil$. Denote by A the circuit resulting from replacing gates g_i with new variables z'_i for $1 \le i \le w$, 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 $\{f_1, \ldots, f_w\}$. Let $Z' = \{z'_1, \ldots, z'_w\}$. Both A and B are syntactically multilinear circuits of depth 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 $K = \mathbb{K}[X, Y]$.

To simplify the exposition, we first describe how to obtain a depth-reduced circuit, and then describe how to make the new circuit syntactically multilinear.

Applying induction on A and B, for all $S, Q \subseteq \{1, \ldots, w\}$, $[p_{f_i}, S]$ and $[p_{g_i}, Q]$ have syntactically multilinear arithmetic circuits $A^{f_i,S}$ and $B^{g_i,Q}$ respectively of size $2^{w^2}(\lceil \ell/2 \rceil)^{25w^2}$ and depth $w^2 \log(\lceil \ell/2 \rceil)$. Note that $p_{h_i}(Z) = p_{f_i}(p_{g_1}(Z), \ldots, p_{g_w}(Z))$. But due to multilinearity,

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

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

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

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

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

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 p_g in the above expression. So the coefficient of m_T is the product of the corresponding coefficients. Hence

$$[p_{h_i}, T] = \sum_{S=\{s_1,\dots,s_t\}\subseteq[w]} \left([p_{f_i}, S] \sum_{\substack{Q_1,\dots,Q_t : \\ \text{partition of } T}} \prod_{k=1}^{t} [p_{g_{s_k}}, Q_k] \right) \tag{3.6}$$

We use this expression to compute $[p_{h_i}, T]$. We first compute $[p_{f_i}, S]$ and $[p_{g_s}, Q]$ for all $i, s \in [w]$ 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 $w^2 2^w w^w \leq 2^{w^2}$ (for $w \geq 2$) and depth $w \log w + w + \log w = O(w^2)$.

Preserving Syntactic Multilinearity: Clearly, the circuit obtained above need not be syntactically multilinear. To achieve this, we do the following modifications:

- Unwind the expression for each $[p_{h_i}, T]$ into a formula, by creating necessary copies of $[p_{f_i}, S]$ and $[p_{g_s}, Q]$ for all $S, T, Q \subseteq \{1, \dots, w\}$.
- Consider a term $[p_{f_i}, S][p_{g_1}, Q_1] \cdots [p_{g_t}, Q_t]$ which violates syntactic multilinearity. There are two cases:
  - For some $a \neq b$, $[p_{g_a}, Q_a]$ and $[p_{g_b}, Q_b]$ share a variable. If the corresponding term $[p_{f_i}, S][p_{g_1}, Q_1] \cdots [p_{g_t}, Q_t]$ has a non-zero contribution to $[p_{h_i}, T]$, then it means that the original syntactically multilinear circuit C has a $\times$-gate v such that the gates g_a and g_b are reachable via two different input gates of v. This violates the syntactic multilinearity property. So it must be the case that this partition of Q has no contribution to $[p_{h_i}, T]$. Hence this particular term $[p_{f_i}, S][p_{g_1}, Q_1] \cdots [p_{g_t}, Q_t]$ can be replaced by 0 without changing the output.
  - For some $a \in S$, sub-formulas $[p_{g_a}, Q_a]$ and $[p_{f_i}, S]$ share a variable, say x. At least one of the polynomials $[p_{g_a}, Q_a]$ and $[p_{f_i}, S]$ does not depend on x, since otherwise there is a $\times$ gate v in C reachable from

g_a with x in both its sub-formulas, violating the assumption of syntactic multilinearity of C. Now replace x with 0 in the corresponding sub-formula that does not depend on x.

Note that in the above process, we need to unwind the resulting circuit into a formula. By equation 3.6 we need to make at most 2^{w^2} copies of each $[p_{g_k}, Q_k]$ for $k \in [w]$. Hence, the size of the resulting formula will blow up by a factor of $2w2^w2^{w^2} \leq 2^{2w^2}$ at every induction step.

Let $s(\ell, w)$ and $d(\ell, w)$ denote the size and depth of the new circuit $D^{p_h,T}$. Then from the construction above, we have the recurrences

$$s(\ell, w) \leq 2^{2w^2} s(\ell', w) + 2^{w^2} \leq 2^{3w^2} s(\lceil \ell/2 \rceil, w)$$

$$d(\ell, w) \leq d(\lceil \ell/2 \rceil, w) + O(w^2)$$

Note that $\ell' = \lceil \ell/2 \rceil$ satisfies $\ell' \leq 3\ell/4$. Suppose that by induction, $s(\ell', w) \leq 2^{w^2}(\ell')^{cw^2}$ for some constant c to be chosen later. So

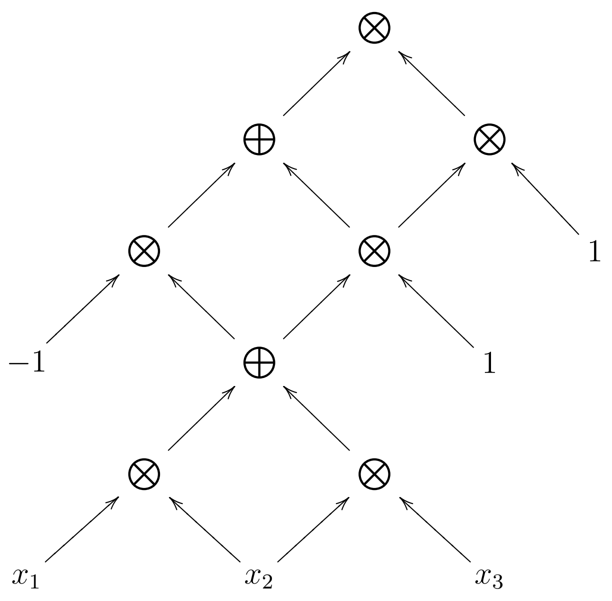
$$\begin{aligned} s(\ell, w) \leq\; & 2^{3w^2} 2^{w^2} (\ell')^{cw^2} & \leq\; & 2^{4w^2} (3\ell/4)^{cw^2} \\ =\; & 2^{w^2} \ell^{cw^2} \left[2^{3w^2} (3/4)^{cw^2} \right] & \leq\; & 2^{w^2} \ell^{cw^2} \end{aligned}$$

where the last inequality holds whenever $8(3/4)^c \leq 1$, say $c \geq 25$.

Similarly, solving the recurrence for $d(\ell, w)$ gives $d(\ell, w) = O(w^2 \log \ell)$. □

Proof (of lemma 3.4). We first relabel all the nodes at the lowest level by new variables $z_1, \ldots, z_w$. Then, applying Lemma 3.5, we obtain circuits for $[p_g, T]$, where g is an output gate of C' and $T \subseteq \{1, \ldots, w\}$. Now, to compute p_g, we sum over all T the values $[p_g, T] \times \prod_{j \in T} val(z_j)$, where $val(z_j)$ denotes the original variable for which z_j was substituted. This adds $O(w)$ to the overall depth of the circuit, thus resulting an overall depth of $O((w + w^2 \log \ell)) = O(w^2 \log \ell)$. The resulting circuit size is bounded by $O(s2^w)$, where s is an upper bound on the size of the circuits constructed in Lemma 3.5, and hence is bounded by $O(2^{w^2} \ell^{25w^2})$ □

Remark 3.7. *If the constant-width circuit C we start with is multilinear but not syntactically multilinear, then the circuit A as in Lemma 3.5 need not be multilinear in the slice variables Z. This is the place where the above construction crucially uses syntactic multilinearity, and does not generalize to multilinear circuits. See Figure 3.2 for an example.*



Circuit A below is obtained by replacing gates at level 2 by variables $\{z_1, z_2, z_3\}$.

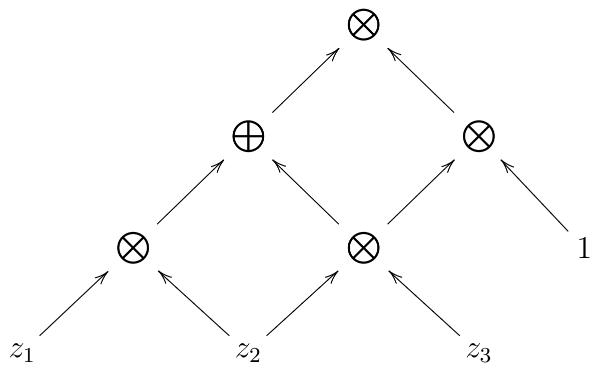


Figure 3.2: A is not multilinear in the slice variable z_2.

4. Making a Circuit Skew

The purpose of this section is to give a direct simulation of width bounded syntactically multilinear circuits by syntactically multilinear ABPs, yielding the following theorem.

THEOREM 4.1. $\mathsf{sm\text{-}VsSC}^0 \subseteq \mathsf{sm\text{-}VBWBP}$

As $\mathsf{sm\text{-}VBWBP} \subseteq \mathsf{sm\text{-}VsSC}^0$ is trivially true, this, along with Corollary 3.2 from the previous section, gives the following relations:

THEOREM 4.2. $\mathsf{sm\text{-}VBWBP} = \mathsf{sm\text{-}VsSC}^0 \subseteq \mathsf{sm\text{-}VNC}^1$.

To establish Theorem 4.1, we proceed as follows. If we use the standard series-parallel construction on a circuit which is not a formula, the size of the resulting ABP can blow up exponentially in the depth of the original circuit (irrespective of its width). But in the case of a syntactically multilinear circuit, one can assume by Proposition 4.3 that the circuit is multiplicatively disjoint (we will define this soon). Along with this, if we have a width bound of w, then for every multiplication gate, one of its sub-circuits is of width at most $w - 1$. We exploit this fact to give, in Theorem 4.8, a simulation of constant width syntactically multilinear circuits by syntactically multilinear ABPs of constant width, with only a polynomial blow up in the size. Thereom 4.1 thus follows from Proposition 4.3 and Theorem 4.8. As a warm-up to establishing Theorem 4.8, we first show in Theorem 4.4 such a depth-reducing simulation for weakly-skew syntactically multilinear circuits of small width.

The rest of this section is organized as follows: Section 4.1 introduces the notion of multiplicatively disjoint circuits and weakly skew circuits. In section 4.2, we give a width efficient simulation of weakly skew circuits by ABPs. Section 4.3 gives width efficient simulation of multiplicatively disjoint circuits by ABPs which preserves syntactic multilinearity.

4.1. Multiplicatively Disjointness and Weakly Skewness.

Multiplicatively Disjoint Circuits: Let C be an arithmetic circuit. C is said to be *multiplicatively disjoint* (MD for short) if every multiplication gate in C operates on disjoint sub-circuits, *i.e.* if $f = g \times h$ is a gate in C, then the sub-circuits rooted at g and h do not share any node (except the input nodes) between them (see Malod & Portier (2008)). We denote the multiplicatively disjoint restriction of a class by the prefix md-, *e.g.* $\mathsf{md\text{-}VSC}^i$ denotes the class of family of polynomials computed by polynomial size multiplicatively disjoint arithmetic circuits of width $O(\log^i n)$. It is not hard to see that syntactic degree of a multiplicatively disjoint circuit is bounded by its size, hence we have $\mathsf{md\text{-}VsSC}^i = \mathsf{md\text{-}VSC}^i$.

An arithmetic circuit that computes polynomials of polynomial degree can be converted into an equivalent MD-circuit without significant blow up in size Malod & Portier (2008). Thus the three restrictions of MD, small syntactic degree and small degree of the output polynomial all coincide at polynomial size and hence are equal to the class VP. However, when the width of the circuit is bounded by $\mathsf{poly}(\log)$, all these restrictions are seemingly different, with MD circuits being the weakest among them, *i.e.* $\mathsf{md\text{-}VsSC}^i = \mathsf{md\text{-}VSC}^i \subseteq \mathsf{VsSC}^i \subseteq \mathsf{VSC}^i$.

A multiplicatively disjoint circuit need not be syntactic multilinear. On the other hand, a syntactically multilinear circuit is already almost multiplicatively disjoint. At any $\times$ gate $f = g \times h$, no variable can appear under both g and h, and so the sub-circuits under g and h can only share constants. As long as the constants are inputs, this does not violate multiplicative disjointness. If a shared constant appears internally, then some gate must be syntactically computing the constant. However, this is redundant in a non-uniform setting. Consider any syntactically multilinear circuit C. Replace all the gates in C that syntactically compute constants (that is, they are reachable only from leaves labeled by values from $\mathbb{K}$) by the values they represent, to obtain a circuit C'. Now, it is easy to see that C' is multiplicatively disjoint and syntactically multilinear, and computes the same polynomial. Thus we can assume without loss of generality that a syntactically multilinear circuit is also multiplicatively disjoint. In particular, we have

Proposition 4.3. $\mathsf{sm\text{-}VsSC}^0 \subseteq \mathsf{md\text{-}sm\text{-}VsSC}^0$.

Also, note that if the circuit C is multilinear but not syntactically multilinear, then C' need not be multiplicatively disjoint.

Weakly Skew Circuits: An arithmetic circuit C is said to be *weakly skew* if for every multiplication gate $f = g \times h$ in C, either the edge (g, f) or the edge (h, f) is a bridge [1] in the underlying graph. By definition, weakly skew arithmetic circuits are also multiplicatively disjoint. We denote this restriction on a class by the prefix weaklyskew-.

[1] A bridge in an undirected graph is an edge (u, v) whose removal disconnects u from v.

Toda (1992) has shown[2] that weakly skew circuits have equivalent skew circuits, *i.e.* $\mathsf{weaklyskew\text{-}VP} = \mathsf{VBP}$. Jansen (2008) extended this result and showed that weakly skew circuits are equivalent to skew circuits in the syntactically multilinear world too, *i.e.* $\mathsf{sm\text{-}weaklyskew\text{-}VP} = \mathsf{sm\text{-}VBP}$. However, the simulation in Jansen (2008) is not width efficient. In the next section, we present a width efficient version of the simulation in Jansen (2008).

4.2. Weakly Skew to Skew. In this section we give a simulation of weakly skew syntactically multilinear constant width arithmetic circuits by syntactically multilinear ABPs of constant width. (Recall, from Section 2, that ABPs are equivalent to skew circuits.) This construction serves as a simpler case of the simulation given in the next section. We include it here since we achieve a slightly better size bound, which allows us to translate the result to higher width (see Corollary 4.7).

We briefly outline the overall idea: Essentially, we do the series-parallel construction. Let C be the given weakly skew circuit of width w. All the $+$ gates in C are left untouched. For a multiplication gate $f = g \times h$, let C_h, the sub-circuit rooted at h, be not connected to rest of the circuit. If $\mathsf{width}(C_h) \leq w - 1$, then we are in good shape, since by placing the ABP $[h]$ (available by induction on the structure of C) in series with (and after) $[g]$ (again available by induction) we can obtain a width bound of $O(w^2)$. If $\mathsf{width}(C_h) = w$, then we have $\mathsf{width}(C_g) \leq w - 1$. In this case, we make a copy of $[g]$ and place it in series with (and after) $[h]$ and again can obtain a width bound of $O(w^2)$, but the size can blow up. Using a careful analysis we argue that size of the new ABP can be bounded by $O(2^w s)$, where s is the size of C. Now we state the main theorem:

Theorem 4.4. *Bounded-width weakly skew circuits can be made skew.*

$$\begin{aligned} \mathsf{weaklyskew\text{-}VsSC}^0 &= \mathsf{VBWBP}. \\ \mathsf{weaklyskew\text{-}sm\text{-}VsSC}^0 &= \mathsf{sm\text{-}VBWBP}. \end{aligned}$$

Proof. We use the following normal form for circuits:

Lemma 4.5. *Let C be an arithmetic circuit of width w and size s. Then there is an equivalent arithmetic circuit C' of width $O(w)$ and size poly(s) such that fan-in and fan-out of every gate is bounded by two, and every layer has at*

[2]The *proof* of Theorem 4.3 in Toda (1992) shows this can be done with constant factor overhead. Proofs of this also were given by Kaltofen and Koiran Kaltofen & Koiran (2008) and independently Jansen (2008), based on a construction by Malod & Portier (2008) .

most one $\times$ gate. Moreover, C' preserves any of the properties of syntactic multilinearity, weakly skewness and multiplicatively disjointness.

Proof. Let k be a bound on maximum fan-in and fan-out of C. First we can reduce the fan-in to two by staggering the circuit and keeping copies of the gates as and when needed. This blows up the width to $2w$ and size to wks. Now in a similar manner we can ensure that the fan-out of a gate is bounded by two and the size blow up will now be w^2k^2s and width will be $4w$. To ensure the second condition we need to push the gates (using staggering and dummy $+$ gates) up by at most $4w$ levels, thus making the total width $8w$ and size $2w^2k^2s$. Since $k \leq w+n$ and $w \leq s$ we have size bounded by $poly(s,n)$. $\square$

We need some more definitions and notations. For an ABP B of depth d with a single source s, we say B *is endowed with a mainline*, if there exist nodes $v_1, v_2, \ldots, v_{d-1}$ reachable only along the path $s, v_1, v_2, \ldots, v_{d-1}$, and if the labels on this path are all set to the ring's multiplicative identity 1. See Figure 4.1(a). For ABPs B_1 and B_2, *piping* the mainline of B_1 into the mainline of B_2 is the operation of removing the edge from the source of B_2 to the first node v of the mainline of B_2, and adding an edge from the last node w of the mainline of B_1 to v. See Figure 4.1(b).

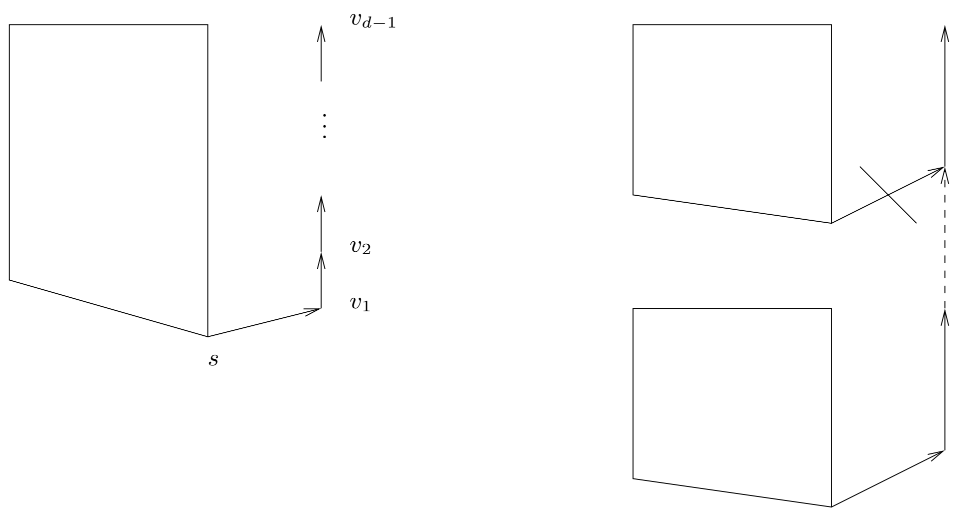
The following lemma now gives Theorem 4.4:

Lemma 4.6. *Let C be a weakly skew arithmetic circuit of width $w > 1$ and size $s > 1$ in the normal form as given by Lemma 4.5. Let $f_1, \ldots, f_w$ be the output gates of C. Then there is an equivalent ABP $[C]$ of width w^2+1, depth 2^ws and size $(w^2+1)2^ws$. $[C]$ has a single start node b and terminal nodes $[f_1], \ldots, [f_w], v$ and will be endowed with a mainline ending in v. Moreover, if C is syntactically multilinear then so is $[C]$.*

Proof. We proceed by induction on $s+w$. If $s=2$, the lemma holds trivially. If $w=2$, then C is a skew-circuit and can be seen as an ABP of width 3 (We also need to add a mainline; hence, width is 3).

Let $s > 2$ and $w > 2$ be given, and assume that C has at least 2 layers. By the induction hypothesis, the lemma holds for all circuits of size s' and w', where either $s' < s$ and $w' \leq w$ or $s' \leq s$ and $w' < w$.

Without loss of generality, assume that f_1 is a $\times$ gate and $f_2, \ldots, f_w$ are $+$ gates. Let C' be the circuit obtained by removing the output gates of C. Let $g_1, \ldots, g_w$ be the output gates of C'. Assume that (without loss of generality) $f_1 = g_1 \times g_2$, and also that the edge (g_1, f_1) is a bridge in the circuit. We define the sub-circuits D and E of C' as follows: D is obtained from C' by deleting the



(a) A BP with a mainline

(b) Piping two mainlines

Figure 4.1: BPs with mainlines

sub-circuit rooted at g_1, E is the sub-circuit rooted at g_1. See Figure 4.2(a). Let $s' = \mathsf{size}(C')$, $w' = \mathsf{width}(C')$, $s_J = \mathsf{size}(J)$ and $w_J = \mathsf{width}(J)$ for $J \in \{D, E\}$. Note that $s = s' + w$, and $s_J < s$ for $J \in \{D, E\}$.

By the induction hypothesis, we have branching programs $[D]$ and $[E]$, both endowed with a mainline. Let $[g_1], v'$ denote the output of $[E]$ and $[g_2], \ldots, [g_w], v''$ denote the output nodes of $[D]$, where v' and v'' are the last nodes on the mainlines. Let $[F]$ be the subprogram of $[D]$, which consists of all paths from the source of $[D]$ to $[g_2]$ and v''. Construct the program $[C]$ with output nodes $[f_1], \ldots, [f_w], v$ as follows:

case 1: $w_E \leq w - 1$.

We compose the ABPs $[D]$ followed by $[E]$ as described below. (See Figure 4.2(b).)

1. For $i, j \geq 2$, $[g_j]$ has an edge to $[f_i]$ if and only if g_j is an input to f_i.

2. For input gates f_i, draw an edge from v'' to $[f_i]$ with the appropriate label.

3. Identify $[g_2]$ with the start node of $[E]$ and relabel the output node of $[E]$ as $[f_1]$. Pipe the mainline of $[D]$ into the mainline of $[E]$.

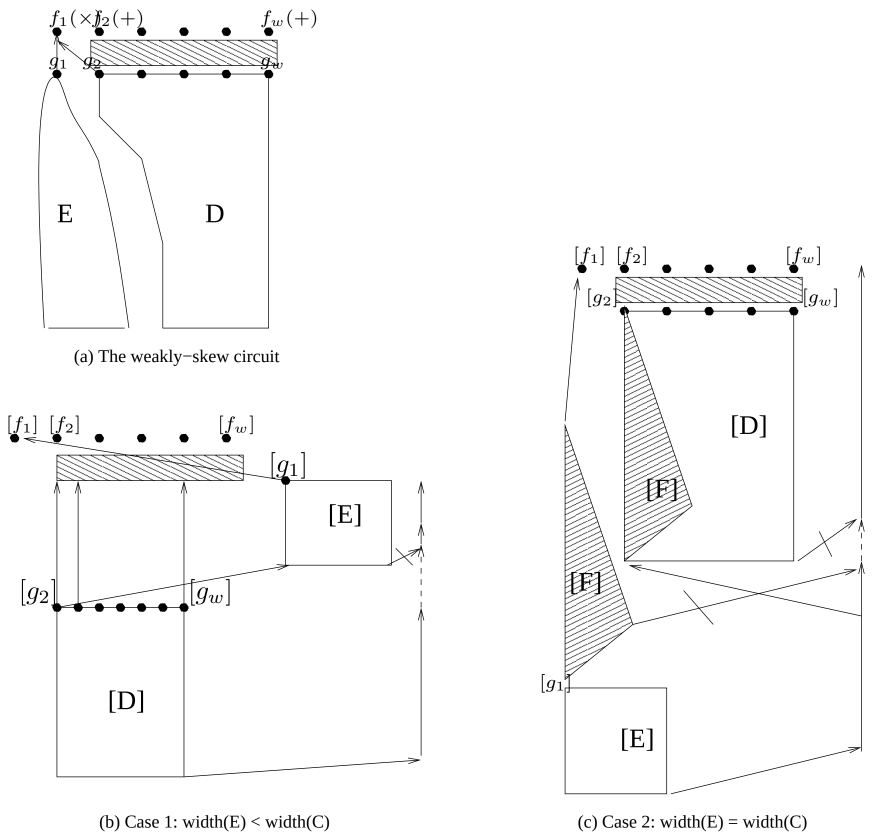


Figure 4.2: Weakly skew circuits to skew circuits

4. Stagger the nodes $[f_2], \ldots, [f_w]$ until the last level of the new program.

Size and width analysis: By the induction hypothesis, we have

$$\begin{aligned}
\mathsf{width}([E]) &\leq (w_E)^2 + 1 \leq (w-1)^2 + 1 \\
\mathsf{width}([D]) &\leq w^2 + 1 \\
\mathsf{length}([E]) &\leq 2^{w-1} size(E) \\
\mathsf{length}([D]) &\leq 2^{w} size(D)
\end{aligned}$$

$$\begin{aligned}
\text{Hence } \mathsf{width}([C]) &= \max\{\mathsf{width}([D]), \mathsf{width}([E]) + w - 1\} \leq w^2 + 1 \quad \text{and} \\
\mathsf{length}([C]) &= \mathsf{length}([D]) + \mathsf{length}([E])] \\
&\leq 2^{w_D} s_D + 2^{w_E} s_E \\
&\leq 2^{w} s_D + 2^{w-1} s_E \leq 2^{w} s \quad (\text{as } s = s_D + s_E + w).
\end{aligned}$$

case 2: $w_E = w$, and hence $w_F \leq w - 1$ and $w_D \leq w - 1$.

We compose ABPs $[E]$, $[F]$ and $[D]$ as follows. (See Figure 4.2(c). Note that a copy of $[F]$ is also inside $[D]$, but when we refer to $[F]$ below, we mean the independent copy.)

1. Identify $[g_1]$ with the source of $[F]$, and pipe the mainline of $[E]$ into the mainline of $[F]$.
2. Add an edge from v' (last node of mainline of $[F]$) to the source of $[D]$,
3. Pipe the mainline of $[F]$ into the mainline of $[D]$.
4. Alongside $[D]$ stagger the output of $[F]$ (which now equals $[f_1]$).
5. For $i, j \geq 2$, $[g_j]$ has an edge to $[f_i]$ if and only if g_j is an input to f_i.
6. Finally, for input gates f_i, draw an edge $(v'', [f_i])$ with the appropriate label.

Size and width analysis: By induction hypothesis,

$$\begin{aligned}
\mathsf{width}([E]) &\leq w^2 + 1 \\
\mathsf{width}([D]) &\leq (w-1)^2 + 1 \\
\mathsf{width}([F]) &\leq (w-1)^2 + 1
\end{aligned}$$

Observe that

$$\begin{aligned}
\mathsf{width}([C]) &\leq \max(\mathsf{width}([E]), \mathsf{width}([F]), \mathsf{width}([D]) + 1) \\
&\leq w^2 + 1
\end{aligned}$$

$$\begin{aligned}
\text{Further,}\quad \mathsf{length}([C]) &= \mathsf{length}([E]) + \mathsf{length}([F]) + \mathsf{length}([D]) + 1 \\
&\leq 2^w s_E + 2^{w-1} s_F + 2^{w-1} s_D + 1 \\
&\leq 2^w (s_D + s_E) + 1 \leq 2^w s.
\end{aligned}$$

Since the size of a layered ABP is $\mathsf{length} \times \mathsf{width}$, we have the required size bound. If C was syntactically multilinear to start with, then it is easy to see that so is $[C]$. $\square$

This completes the proof of Theorem 4.4. $\square$

By the parameters in the Lemma 4.6, it is not hard to see that if we start with a syntactically multilinear weakly skew circuit of width $O(\log n)$, we get a syntactically multilinear ABP of width $O(\log^2 n)$, *i.e.*

Corollary 4.7. $\mathsf{weaklyskew\text{-}sm\text{-}VsSC}^1 \subseteq \mathsf{sm\text{-}VBP}[\mathsf{width} = \log^2 n]$.

4.3. Multiplicatively Disjoint to Skew. We extend the simulation in Lemma 4.6, and hence Theorem 4.4, to multiplicatively disjoint circuits.

Theorem 4.8. *Bounded-width multiplicatively disjoint circuits can be made skew.*

$$\begin{aligned} \mathsf{md\text{-}VsSC}^0 &= \mathsf{VBWBP}. \\ \mathsf{md\text{-}sm\text{-}VsSC}^0 &= \mathsf{sm\text{-}VBWBP}. \end{aligned}$$

The theorem follows directly from the lemma stated below. The idea is the same as that used in Lemma 4.6, but with a weaker bound on the size of the resulting ABP: $O(s^w)$ instead of $O(2^w s)$.

Lemma 4.9. *C be a multiplicatively disjoint arithmetic circuit of width w and size s in the normal form as given by Lemma 4.5. Let $f_1, \ldots, f_w$ be the output gates of C. Then there exists an equivalent arithmetic branching program $[C]$ of width $O(w^2)$, length $O(s^w)$, and size $O(w^2 s^w)$. $[C]$ has a single start node b and terminal nodes $[f_1], \ldots, [f_w], v$, and is endowed with a mainline ending in v. Moreover, if C is syntactically multilinear, then so is $[C]$.*

Proof. We proceed by induction on $s + w$. If $s = 2$, the lemma holds trivially. If $w = 2$, C is a weakly skew circuit, and hence can be seen as a BP of width 5.

Let $s > 2$ and $w > 2$ be given, and assume that C has at least 2 layers. Suppose, by induction hypothesis that the lemma holds for all circuits of size s' and w', where either $s' < s$ and $w' \leq w$ or $s' \leq s$ and $w' < w$.

Let C' be the sub-circuit obtained by deleting $f_1, \ldots, f_w$. Let $g_1, \ldots, g_w$ be the output gates of C'. Without loss of generality, let $f_1 = g_1 \times g_2$ be the only multiplication gate at the output layer of C. Let D denote the sub-circuit rooted at g_1 and E be the sub-circuit rooted at g_2. Since C is multiplicatively disjoint, we have either $\mathsf{width}(D) \leq w - 1$ or $\mathsf{width}(E) \leq w - 1$. Without loss of generality, assume that $\mathsf{width}(D) \leq w - 1$.

Let $s' = \mathsf{size}(C')$, $s_D = \mathsf{size}(D)$, $w' = \mathsf{width}(C')$, and $w_D = \mathsf{width}(D)$. By induction hypothesis, we obtain ABPs $[C']$ and $[D]$. $[C']$ has $w + 1$ output nodes, namely $[g_1], \ldots, [g_w], v$. $[D]$ has two output nodes $[g_1']$ and v'.

Now construct the ABP $[C]$ with output nodes $[f_1], \ldots, [f_w], v$ by composing $[C']$ followed by $[D]$ as follows: For all $i \geq 2$, connect $[g_j]$s to $[f_i]$s according to the edges in the circuit C, i.e edge $([g_j], [f_i])$ is in $[C]$ if and only if g_j is an

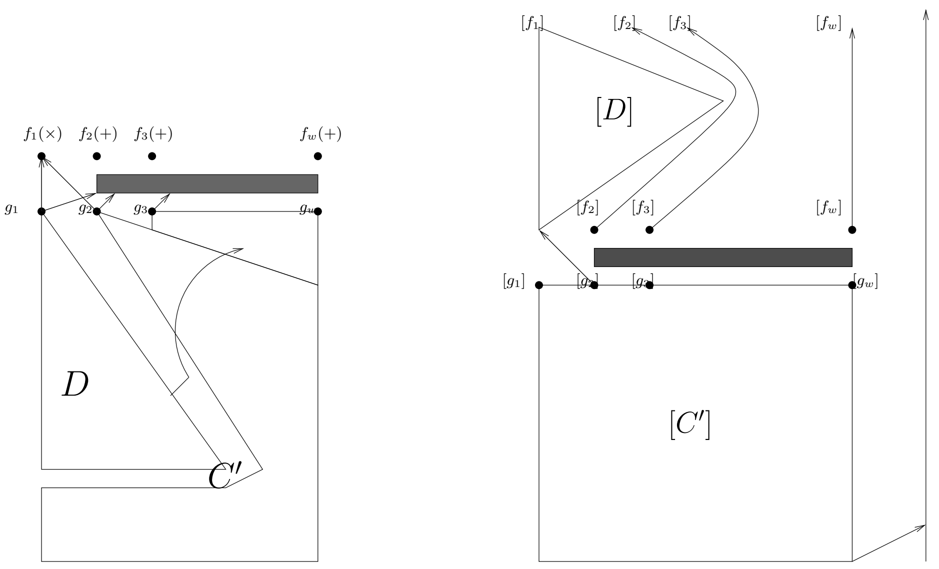


Figure 4.3: Multiplicatively disjoint to skew

input for f_i. In case f_i is an input gate, draw an appropriately labeled edge from v. Put an edge from $[g_2]$ to $[f_1]$. Now identify the start node of $[D]$ with $[f_1]$ and re-label the terminal node of $[D]$ as $[f_1]$. Do the necessary staggerings to carry on the values $f_2, \ldots, f_w$ to the last layer. We also pipe the mainline of $[C']$ into the mainline of $[D]$.

Analysis: As $s' = s - w$ and $w' \leq w$, using the induction hypothesis we have

$$\mathsf{length}([C']) \leq s'^{w'} \leq (s-w)^w.$$

Furthermore, as $s_D \leq s - w$ and $w_D \leq w - 1$, we have

$$\begin{aligned}
\mathsf{width}([C']) &\leq w'^2 + 1 \leq w^2 + 1 \\
\mathsf{length}([D]) &\leq s_D^{w_D} \leq (s-w)^{w-1} \\
\mathsf{width}([D]) &\leq (w-1)^2 + 1
\end{aligned}$$

Now, by the construction,

$$\begin{aligned}
\mathsf{width}([C]) &= \max\{\mathsf{width}([C']), \mathsf{width}([D]) + w - 1\} \\
&\leq \max\{w^2 + 1, (w-1)^2 + w - 1\} \leq w^2 + 1
\end{aligned}$$

Also,

$$\begin{aligned}\mathsf{length}([C]) &= \mathsf{length}([C']) + \mathsf{length}([D]) \\ &\leq (s-w)^w + (s-w)^{w-1} \leq s^w\end{aligned}$$

for $w > 2$ and $w < s$. Thus, $\mathsf{size}([C]) = (w^2+1)s^w$. It is easy to see that this construction preserves the syntactic multilinearity property. □

Note that Lemma 4.9 works for all multiplicatively disjoint circuits. Consequently, the class $\mathsf{md\text{-}VSC}^0$ becomes the "largest" fragment of VsSC^0 known to us, that is still equivalent to $\mathsf{VNC}^1 = \mathsf{VBWBP}$. Recall that $\mathsf{md\text{-}VsSC}^0 = \mathsf{md\text{-}VSC}^0$. We summarize the situation as follows:

Corollary 4.10. $\mathsf{weaklyskew\text{-}VSC}^0 = \mathsf{md\text{-}VSC}^0 = \mathsf{VNC}^1 = \mathsf{VBWBP}$

Remark 4.11. *The simulation of weakly skew circuits from Lemma 4.6 carries over to multilinear circuits too. However, as a multilinear circuit need not be multiplicatively disjoint (see Section 4.1), Lemma 4.9 does not work for multilinear circuits which are not syntactically multilinear.*

5. An Overview of Syntactically Multilinear Classes

Now we turn our attention to the overall picture of the algebraic classes around VNC^1 in the syntactically multilinear world. In other words, we attempt to redraw the Figure 2.2 when all the classes are restricted to be syntactically multilinear. We consider and compare the classes $\mathsf{sm\text{-}VP}_e$, $\mathsf{sm\text{-}VNC}^1$, $\mathsf{sm\text{-}VsSC}^0$, $\mathsf{sm\text{-}VBWBP}$, and $\mathsf{sm\text{-}VrGP}$.

A classical result from Brent (1973) 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 syntactically multilinear formula F, then the depth-reduced formula F' is also syntactically multilinear.

Theorem 5.1. *Every syntactically multilinear formula with n leaves has an equivalent syntactically multilinear circuit of depth $O(\log n)$ and size $O(n)$. In particular,* $\mathsf{sm\text{-}VP}_e \subseteq \mathsf{sm\text{-}VNC}^1$.

Proof. By simultaneous induction on the number of leaves in the formula, we can prove the following statements. This is exactly the construction of Brent (1973), analyzed carefully for syntactic multilinearity. For a formula F, let $|F|$ denote the number of leaves in F. We inductively establish the following statements:

(i) If F is a syntactically multilinear formula with n leaves, then there is an equivalent syntactically multilinear circuit F' of depth $\lceil 4 \log n \rceil$ and size $2n$.

(ii) If x is an input gate in F, then we can express F as $F' = Ax + B$, where A, B are syntactically multilinear circuits that do not depend on x and are of depth $\lceil 4 \log |A| \rceil$ and $\lceil 4 \log |B| \rceil$ respectively.

Then we can unwind the circuit so obtained into a syntactically multilinear formula; due to the depth bound, the resulting formula will still be of polynomial size.

In the base case, there is either a single variable or a constant, and the claim holds trivially.

To proceed by induction, we use the folklore tree separator theorem which says that in any rooted binary tree T, we can find a node u such that the sub-tree T_u rooted at u and the sub-tree $T \setminus T_u$ are both of size at most $3/4|T|$. Here the size of a tree is the number of leaf nodes (nodes with degree one) in it.

Let X be a tree separator for F, with children L, R, so that $X = L \ \mathsf{op}\ R$. Replace the whole subtree under X by a new variable x. By inductive statement (ii), we have $F' = Ax + B$ where A, B are as above (*i.e.* they are both syntactically multilinear and do not depend on X). Also by inductive statement (i), we have syntactically multilinear formulas L', R' equivalent to L, R of small depth. Thus we have $F' = A \times (L' \ \mathsf{op}\ R') + B$. Since A does not depend on any variable below X, F' is syntactically multilinear. Also, $\mathsf{depth}(F') = \max\{\mathsf{depth}(A) + 2, \mathsf{depth}(L') + 3, \mathsf{depth}(R') + 3, \mathsf{depth}(B) + 1\} \leq \lceil 4 \log n \rceil$.

To prove the second half of the statement above, let x be any input gate in F. Now find a tree separator $X = L \ \mathsf{op}\ R$ such that the subtree at one of its children, say L, contains x as a leaf node and is of size $< n/2$. Then, by inductive statement (ii) applied to L, $L' = Ax+B$, where A, B are independent of x, syntactically multilinear and of small depth. Now replace the subtree at X by a new variable y. Applying inductive statement (ii), we have $F' = Cy + D$, where C, D are syntactically multilinear small depth formulas which do not depend on y (*i.e.* $L \ \mathsf{op}\ R$). Applying inductive statement (i) to R, we have an equivalent small-depth R'.

Case 1: $\mathsf{op} = +$. Then $F' = C((Ax+B)+R')+D = CAx+(CB+CR'+D)$. This is again syntactically multilinear since C does not depend on y, *i.e.* $Ax + B + R$.

Case 2: $\mathsf{op} = \times$. Then $F' = C(Ax + B)R' + D = CAR'x + (CBR' + D)$. Here again F' is syntactically multilinear since C does not depend on A, B, R', and also because A and B do not share any variables with R'.

Since we are constructing a circuit and not a formula, we don't need to replicate the circuits for C and R'. For details about the size/depth, see the analysis in Brent (1973). $\square$

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 5.2. *For any syntactically multilinear branching program P of width w and size s over ring $\mathbb{K}$, there is an equivalent syntactically 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, sm-VBWBP $\subseteq$ sm-VNC^1 *and* sm-VBP $\subseteq$ sm-VSAC^1.

Proof. Let ℓ be the length of P ($s = \ell w$), and let $p_{s,t}$ denote the weighted sum of the directed paths between nodes s and t. Let $v_1, \ldots v_w$ denote the nodes at the level $\ell' = \lceil \ell/2 \rceil$ of P. Then $p_{s,t} = \sum_{i=1}^{w} p_{s,v_i} \times p_{v_i,t}$. Thus the depth and size of the inductively constructed circuit satisfy the recurrences $d(\ell) = 2 + d(\ell')$ and $s(\ell) = (3w)s(\ell')$, giving the desired bounds. It is clear that the circuit so constructed is syntactically multilinear; if it were not, the offending $\times$ gate would pinpoint a path in P that reads some variable twice. $\square$

Note that Lemma 5.2 and Theorem 4.1 together give another proof of Corollary 3.2.

It is also straightforward to see that the construction of Istrail & Zivkovic (1994), staggering a small-depth formula into a small-width one, preserves syntactic multilinearity. Thus

Lemma 5.3. *Let Φ be any sm-formula with depth d and size s. Then there is an equivalent syntactically multilinear formula Φ' of depth $2s$ and width d.*
In particular, sm-VNC^1 $\subseteq$ sm-VLWF.

Proof. For completeness we give a detailed proof here. The construction is by induction on the structure of the formula Φ. The base case is when Φ is a single variable or a constant, in which case the lemma holds trivially.

Suppose the lemma holds for any formula of depth at most $d-1$. Consider the root gate f of a formula Φ of depth d. Suppose $f = \sum_{i=1}^{k} g_i$ (respectively $f = \prod_{i=1}^{k} g_i$). As the depth of each formula g_i is bounded by $d-1$, by induction we have formulas g_i' of width $d-1$ and depth bounded by s_i (the size of g_i), computing the same function as g_is. Place the node corresponding to f with two children. At one child, place the formula g_1'; at the other, place a series of no-op (*i.e.* $\times 1$ or $+0$) gates till the last level of g_1'. Then give the last no-op gate two children, place g_2' at one child, and so on. The width of the new formula Φ' thus obtained is bounded by $\max_i \mathsf{width}(g_i') + 1$, and its depth is bounded by $\sum_i \mathsf{depth}(g_i') + 1 \le \sum_i s_i + 1 \le s$. Note that in this process, for any gate g in Φ the variables it operates on are not changed in the new formula Φ', that is, the only new gates which are introduced in Φ' are the no-op gates which are used for staggering, which only multiply by the constant 1. Thus if Φ is syntactically multilinear then so is Φ. □

From Lemma 5.3 and Theorem 5.1, we have the following equivalence.

Corollary 5.4. *Over any ring* $\mathbb{K}$,
$\mathsf{sm\text{-}VP}_e = \mathsf{sm\text{-}VLWF} = \mathsf{sm\text{-}VNC}^1 = \mathsf{sm\text{-}Formula\text{-}Depth,Size}(\log, \mathsf{poly})$.

In Allender *et al.* (1999) a characterisation for unbounded fanin 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 Allender *et al.* (1999) works for unbounded fanin bounded depth arithmetic circuits over arbitrary rings, showing that $\mathsf{VBWrGP} = \mathsf{VAC}^0$. By closely examining the parameters in Allender *et al.* (1999), we obtain a characterisation for VNC^1 in terms of the restricted version of log width grid branching programs. We also note that these constructions preserve syntactic multilinearity. In the statement and proof below, we use the notion of alternation-depth: a circuit C has alternation depth a if on every input-to-root 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 5.5. *Let Φ be an arithmetic formula of size s 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 syntactically multilinear, then so is P.

Proof. We first use associativity of $+$ and $\times$ and convert the input formula to a depth $2d$ unbounded fan-in formula Φ'. We can thus assume without loss of generality that in Φ', all nodes in a particular layer represent the same type of gate and two successive layers have different kind of gates. Also, we can assume that Φ' is height balanced, *i.e.* any root-to-input path in Φ' is of length exactly $2d$. We further assume that the root is a $\times$ gate. If these conditions do not hold, then ensuring them will blow up the size of Φ' to at most s^2, and increase the depth by at most 2. (Since Φ' has unbounded fan-in gates, its size is measured by the number of wires rather than the number of gates.)

So now we assume that s and $a = 2d$ are the number of wires and the alternation depth of a formula Φ already in this normal form. The construction here is exactly the same as in Allender *et al.* (1999); it is included here for completeness in arguing, over more general parameters, that syntactic multilinearity is preserved.

We proceed by induction on the depth of the formula Φ. The base case is when $d \leq 1$. If the depth is 0, then Φ is either a variable or a constant in the underlying ring. In this case the graph is $G_{0,1}(c)$ where $\Phi = c$. If $d = 1$, then Φ is a product of linear factors, and a suitable composition of $G_{0,1}(c)$ graphs and $G_{0,2}$ represents it.

Suppose that for any (syntactically multilinear) formula F with alternation depth $2d' < 2d$ and number of wires s' (in the normal form described above), there is a (syntactically multilinear) restricted grid program P of width $2d'$ and length $s'^2 + 2s'$, where P uses variables from $\mathsf{Var}(F)$.

Now let Φ be a normal form formula with alternation depth $2d$. Consider the root gate g of Φ. Let $g_1, \ldots, g_k$ be the children of g, where $g_i = \sum_{j=1}^{t_i} g_{i_j}$. Let s_{i_j} and $2d_{i_j} = 2d - 2$ respectively denote the number of wires and alternation depth of the sub-formula rooted at g_{i_j}. Note that $s = k + \sum_i (t_i + \sum_j s_{i_j})$. Applying induction on the sub-formula rooted at each g_{i_j}, let Q_{i_j} denote the resulting restricted grid program for the formula at g_{i_j}. Now place the Q'_{i_j}s $(1 \leq j \leq t_i)$ as in Figure 5.2 to get the program P_i, and connect the P_i's as shown in Figure 5.1 to get the desired program P. By the inductive hypothesis, $\mathsf{length}(Q_{i_j}) \leq s_{i_j}^2 + 2s_{i_j}$ and $\mathsf{width}(Q_{i_j}) \leq 2d_{i_j}$. From the construction as above, we have $\mathsf{length}(P_i) = t_i + 1 + \sum_j \mathsf{length}(Q_{i_j}) \leq t_i + 1 + \sum_j (s_{i_j}^2 + 2s_{i_j})$ and hence $\mathsf{length}(P) = k - 1 + \sum_i \mathsf{length}(P_i) \leq k - 1 + \sum_i ((t_i + 1) + \sum_j (s_{i_j}^2 + 2s_{i_j})) \leq s^2 + 2s$. Note that the construction in Figure 5.2 adds 2 to the width and the construction in Figure 5.1 does not change the width. Hence the width of P is bounded by $2 \max_{i,j} d_{i_j} + 2 = 2d$.

If Φ is syntactically multilinear, then the formulas rooted at g_{i_j} are all syntactically multilinear, and for $i \neq i'$, $\mathsf{Var}(g_i) \cap \mathsf{Var}(g_{i'}) = \emptyset$. Thus, by the

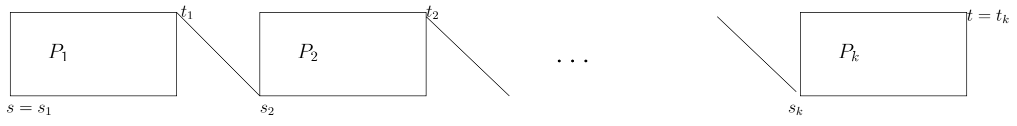


Figure 5.1: Multiplication of rGP's

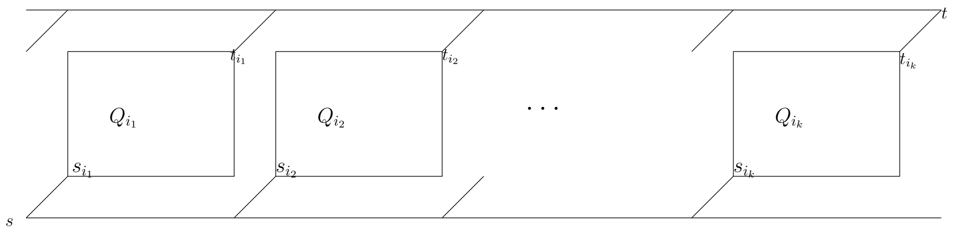


Figure 5.2: Addition of rGP's

inductive hypothesis, the programs Q_{i_j} are syntactically multilinear, and only use variables from $\mathsf{Var}(g_{i_j})$, and hence the programs P_i (for each i) only use variables from $\mathsf{Var}(g_i)$. Thus for every $i \neq i'$, $\mathsf{Var}(P_i) \cap \mathsf{Var}(P_{i'}) = \emptyset$. Since each path in the final program goes through exactly one Q_{i_j} for each i, it follows that P is syntactically read-once. □

We now establish the converse to Lemma 5.5. The proof of the converse as in Allender *et al.* (1999) is uniform and it produces a circuit rather than a formula. If we do not insist on uniformity of the circuit, then we actually get a formula. Thus it can be shown that functions computed by width $2w+2$, length ℓ restricted grid programs can be computed (non uniformly) by unbounded fan-in formulas of depth $2w + 2$ with $O(\ell)$ wires.

LEMMA 5.6. *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 at most 2ℓ, and $\mathsf{Var}(\Phi) = \mathsf{Var}(P)$.*
Further, if P is syntactically multilinear, then so is Φ.

PROOF. Again, this construction is the same as in Allender *et al.* (1999); it is presented here with the induction unfolded to allow arguing, over more general parameters, that syntactic multilinearity is preserved. It yields an unbounded fan-in circuit of depth $d = 2w + 2$ with 2ℓ wires, which can then be converted to a bounded fan-in circuit of the same size with alternation depth d.

For a program B, let $f(B)$ denote the function computed by B. We proceed by induction on w. The base case is when $w = 0$, *i.e.* we have a rGP P of width 2. A contiguous sequence of layers with pattern $G_{0,1}(c_1), G_{0,1}(c_2), \ldots, G_{0,1}(c_r)$ computes $c_1 + c_2 + \ldots c_r$. If two such maximal sequences are connected by a layer with $G_{0,2}$, then the corresponding functions are multiplied. Thus $f(P)$ can be computed by a depth 2 formula with one $\times$ gate as root and a number of $+$ gates as its inputs, where the $+$ gates get input from $X \cup \mathbb{K}$. The total fan-in of the $+$ gates is bounded by the number of layers which contain the graph $G_{0,1}(c)$, for some c. The fan-in of the $\times$ gate is one more than the number of layers which have the graph $G_{0,2}$. (The layers having $G_{0,0}$ do not contribute to the formula.) Thus the total number of wires is bounded by $\ell + 1 \leq 2\ell$, and depth is 2. If P is syntactically multilinear, no path reads the same variable twice, and so the inner blocks separated by $G_{0,2}$ have disjoint sets of variables. Hence the top $\times$ gate operates on disjoint sets of variables.

Suppose that for any $w' < w$ the claim holds, *i.e.* for a (syntactically multilinear) rGP P' of width $2w' + 2$ and length ℓ', there is an equivalent (syntactically multilinear) formula Φ' of depth $2w' + 2$ and size $2\ell'$ and using only variables from $\mathsf{Var}(P')$.

Now P is the given rGP of width $2w + 2$, length ℓ. Let P be composed as $g_1, \ldots, g_\ell$. Let $i_1 < i_2 < \ldots < i_m$ be the (uniquely defined) set of all indices where $g_{i_1}, \ldots, g_{i_m}$ are the graph $G_{w,2w+2}$. Define $i_0 = 0$, $i_{m+1} = \ell + 1$.

For each $0 \leq j \leq m$, let P_j denote the program $g_{i_j+1}, \ldots, g_{i_{j+1}-1}$ sandwiched between the jth and $(j+1)$th incidence of $G_{w,2w+2}$.

The nodes s_j and t_j for each P_j are defined accordingly. Let ℓ_j denote the length of P_j; then $\ell = m + \sum \ell_j$. Note that these P_js do not have $G_{w,2w+2}$ at any layer, and $f(P) = \prod_j f(P_j)$.

Consider P_j for some j. Let $h_{j_1}, \ldots h_{j_{r_j}}$ denote the layers of P_j which are the connecting graph $G_{w,2w+1}$. Let $Q_{j,k}$ denote the part of the program between h_{j_k} and $h_{j_{k+1}}$, and $Q_{j,0}$ denote the part between g_{i_j} and h_{j_1} and Q_{j,r_j} denote the part between h_{j_r} and $g_{i_{j+1}}$. Let $Q'_{j,k}$ denote the graph obtained from $Q_{j,k}$ be removing the top-most and bottom-most lines and the edges connecting them. Then $\mathsf{width}(Q'_{j,k}) = \mathsf{width}(Q_{j,k}) - 2 = 2w$. Let $\ell_{j,k}$ denote the length of $Q'_{j,k}$; so $\ell_j \leq r_j + \sum_{k=1}^{r_j-1} \ell_{j,k}$. The nodes $s'_{j,k}$ and $t'_{j,k}$ for $Q'_{j,k}$ are defined accordingly. Now $f(P_j) = \sum_{k=1}^{r_j-1} f(Q'_{j,k})$. (Note that $Q_{j,0}$ and Q_{j,r_j}, even if non-trivial, play no role in $f(P_j)$ because there is no connection from s_j to these blocks.)

By induction, for each $Q'_{j,k}$ we obtain an equivalent (syntactically multilinear) formula $\Phi_{j,k}$ with variables from $\mathsf{Var}(Q'_{j,k})$, $\mathsf{size}(\Phi_{j,k}) = s_{j,k} = 2\ell_{j,k}$ and $\mathsf{depth}(\Phi_{j,k}) = d_{j,k} = 2w$. Now define $\Phi = \prod_j \sum_{k=1}^{r_j-1} \Phi_{j,k}$. Then $\mathsf{size}(\Phi) = s =$

$m + \sum_j (r_j - 1 + \sum_k 2\ell_{j,k}) \leq 2\ell$ and $\mathsf{depth}(\Phi) = 2w + 2$ as desired. Clearly $\mathsf{Var}(\Phi) = \mathsf{Var}(P)$.

If P is syntactically multilinear, then inductively we have $\Phi_j = \sum_{k=1}^{r_j} \Phi_{j,k}$ operating on $\mathsf{Var}(P_j)$, and each $\Phi_{j,k}$ is syntactically multilinear. Consider the root gate of Φ. If it is not syntactically multilinear, then for some $j < j'$, and for some k, k', $\Phi_{j,k}$ and $\Phi_{j',k'}$ use the same variable x. Thus, by induction, P_j has an s_j-to-t_j path using x, and $P_{j'}$ also has an $s_{j'}$-to-$t_{j'}$ path using x. Combining these paths with (1) the s-to-s_j path along the bottom-line, (2) the t_j-to-$s_{j'}$ path using g_{i_j+1} and then the bottom line, and (3) the $t_{j'}$-to-t path along the top line, gives a path in P that reads x twice, contradicting the read-once property of P. ☐

As an immediate consequence of the above two lemmas, we have:

Corollary 5.7. *(i)* $\mathsf{sm\text{-}VAC}^0 = \mathsf{sm\text{-}VBWrGP}$*;*

(ii) $\mathsf{VNC}^1 = \mathsf{VrGP} = \mathsf{VLWrGP}$.

(iii) $\mathsf{sm\text{-}VNC}^1 = \mathsf{sm\text{-}VrGP} = \mathsf{sm\text{-}VLWrGP}$*;*

Proof. (1) follows directly from Lemmas 5.5 and 5.6.
(2) $\mathsf{VNC}^1 \subseteq \mathsf{VLWrGP} \subseteq \mathsf{VrGP} \subseteq \mathsf{VP}_e = \mathsf{VNC}^1$, where the first containment follows from Lemma 5.5, the second is obvious, the third follows from Lemma 5.6, and the last equality from Brent (1973).
(3) is similar to (2), with the last equality following from Theorem 5.1. ☐

Further, noting that the constructions do not introduce constants other than 0 and 1, we see that they hold in the Boolean setting as well. Therefore we have the following corollary.

Corollary 5.8. $\mathsf{NC}^1 = \mathsf{rGP} = \mathsf{LWrGP}$.

We summarize these relationships in Figure 5.3.

6. Coefficient Functions

Let f be a polynomial over variables $X = \{x_1, x_2, \ldots, x_n\}$; we denote this by $\mathsf{Var}(f) = X$. For a monomial m in variables from X, the partial coefficient function $\mathsf{coef}(f, m)$ is defined to be the unique polynomial g such that f can be written as $f = mg + h$, where h is a polynomial with none of its monomials divisible by m.

$$\begin{tikzcd}
 & \boxed{\begin{array}{l}\mathsf{sm\text{-}VNC}^1 = \ \mathsf{sm\text{-}VP}_e = \\ \mathsf{sm\text{-}VrGP} = \mathsf{sm\text{-}VLWrGP}\end{array}} \arrow[r] \arrow[d] \arrow[rr, bend left, "\subsetneq"] & \mathsf{sm\text{-}VsSC}^1 \arrow[r] & \boxed{\begin{array}{l}\mathsf{sm\text{-}VP} = \\ \mathsf{sm\text{-}SAC}^1\end{array}} \\
\boxed{\begin{array}{l}\mathsf{sm\text{-}VBWBP} = \\ \mathsf{sm\text{-}VsSC}^0\end{array}} \arrow[ur] \arrow[r] \arrow[rr, dotted, bend right, "\overset{?}{\subsetneq}"'] & \mathsf{sm\text{-}VLWBP} \arrow[r] \arrow[ur] & \mathsf{sm\text{-}VBP} \arrow[ur] &
\end{tikzcd}$$

Figure 5.3: Relationship among syntactically multilinear classes

Malod studies the complexity of computing coefficient functions of polynomials computed by classes of arithmetic circuits Malod (2007). From an old observation by Hammon, it can be seen that the permanent polynomial equals $\mathsf{coef}(f, y_1y_2 \dots y_n)$, where f can be computed by the depth-3 formula $f = \prod_{i\in[n]} \sum_{j\in[n]} x_{ij}y_j$. In Malod (2007) it is shown that the Hamiltonian polynomial can be represented as a coefficient of a polynomial g computed by polynomial size arithmetic circuits. A closer inspection shows that this polynomial g is actually in VBP. Thus arithmetic circuit classes which are only as powerful as VAC^0 or VBP can generate VNP-complete polynomials as coefficient functions, and hence the coefficient functions are hard in general.

In the case of polynomials computed by syntactically multilinear circuits we will prove that the situation is markedly different compared to the general case. For a multilinear polynomial f over variables $x_1, x_2, \dots, x_n$, we define the coefficient function $\mathsf{mcoef}(f, \cdot)$ as follows:

$$\text{for each } a = a_1a_2 \dots a_n \in \{0,1\}^n, \quad \mathsf{mcoef}(f, a) = \mathsf{coef}(f, x_1^{a_1}x_2^{a_2} \dots x_n^{a_n}).$$

Given $\mathsf{mcoef}(f, \cdot)$, there is a unique multilinear polynomial $g(x, e)$ in variables from X and E, such that for all $a \in \{0,1\}^n$, $g(x, a) = \mathsf{mcoef}(f, a)$. This polynomial can be obtained from f by interpolation, as follows:

$$g(x, e) = \sum_{b\in\{0,1\}^n} \mathsf{mcoef}(f, b) \prod_{i=1}^{n} \left(e_ib_i + (1 - e_i)(1 - b_i)\right).$$

With some abuse of notation we will denote this polynomial g by $\mathsf{mcoef}(f, e)$.

6.1. Closure Property. A syntactically multilinear complexity class $\mathsf{sm}\text{-}\mathcal{C}$ is said to be *closed under taking coefficients*, if for any polynomial $f \in \mathsf{sm}\text{-}\mathcal{C}$, the polynomial $\mathsf{mcoef}(f, e)$ is also in $\mathsf{sm}\text{-}\mathcal{C}$. We have the following identities:

For any polynomials f and g, and for all $a_1, \ldots, a_n \in \{0, 1\}$, the partial coefficient of $x_1^{a_1} \cdots x_n^{a_n}$ in $f + g$ is the sum of the coefficients of $x_1^{a_1} \cdots x_n^{a_n}$ in f and g. Hence

$$\mathsf{mcoef}(f + g, e) = \mathsf{mcoef}(f, e) + \mathsf{mcoef}(g, e) \tag{6.1}$$

Let f and g be multilinear polynomials in $\mathbb{K}[x_1, \ldots, x_n]$ with $\mathsf{Var}(f) \cap \mathsf{Var}(g) = \emptyset$. For $a_1, \ldots, a_n \in \{0, 1\}$, if there is an i such that $a_i = 1$ and $x_i \notin \mathsf{Var} f \cup \mathsf{Var}(g)$, then the partial coefficient of $x_1^{a_1} \cdots x_n^{a_n}$ is zero. Otherwise, since $\mathsf{Var}(f) \cap \mathsf{Var}(g) = \emptyset$, the monomial $x_1^{a_1} \cdots x_n^{a_n}$ factors across f and g in a unique way, *i.e.*

$$\mathsf{mcoef}(fg, e) = \mathsf{mcoef}(f, e^f) \cdot \mathsf{mcoef}(g, e^g) \cdot \left(\prod_{x_i \notin \mathsf{Var}(f) \cup \mathsf{Var}(g)} (1 - e_i) \right) \tag{6.2}$$

where e^f and e^g are the projection of e to $\mathsf{Var}(f)$ and $\mathsf{Var}(g)$ (*i.e.* $e_i^f = 0$ for $x_i \notin \mathsf{Var}(f)$, $e_i^f = e_i$ for $x_i \in \mathsf{Var}(f)$; similarly for e^g).

For individual variables x_i and constants μ we have

$$\mathsf{mcoef}(x_i, e) = (x_i(1 - e_i) + e_i) \left(\prod_{j \in [n], j \neq i} (1 - e_j) \right) \tag{6.3}$$

$$\mathsf{mcoef}(\mu, e) = \mu \left(\prod_{j \in [n]} (1 - e_j) \right) \tag{6.4}$$

THEOREM 6.5. *Each of the following syntactically multilinear classes is closed under taking coefficients:* $\mathsf{sm}\text{-}\mathsf{VP}, \mathsf{sm}\text{-}\mathsf{VBP}, \mathsf{sm}\text{-}\mathsf{VNC}^1, \mathsf{sm}\text{-}\mathsf{VSC}^i, \mathsf{sm}\text{-}\mathsf{VBWBP}$, *and* $\mathsf{sm}\text{-}\mathsf{VAC}^i$, *for all* $i \geq 0$.

PROOF. Let C be a syntactically multilinear circuit computing the polynomial f. We inductively construct a circuit C' on variables $X \cup e = \{x_1, \ldots, x_n\} \cup \{e_1 \ldots, e_n\}$, computing $\mathsf{mcoef}(f, e)$, as follows:

Base Case If C is a single input gate, $C = a$ where $a \in X \cup \mathbb{K}$, then the coefficient function is given by Equations 6.3,6.4.

Induction

- $C = C_1 + C_2$: Let C'_1 and C'_2 be the circuits obtained from induction for the coefficient functions of the polynomials computed by C_1 and C_2. Then from Equation 6.1, $C'(X, e) = C'_1(X, e) + C'_2(X, e)$.

- $C = C_1 \times C_2$: Let C'_1 and C'_2 be the circuits obtained from induction for the coefficient functions of the polynomials computed by C_1 and C_2. From syntactic multilinearity of C we know that $\mathsf{Var}(C_1) \cap \mathsf{Var}(C_2) = \emptyset$. Then from Equation 6.2, we have $C'(X, e) = C'_1(X, e') \times C'_2(X, e'') \times \left(\prod_{x_i \notin \mathsf{Var}(C_1) \cup \mathsf{Var}(C_2)} (1 - e_i)\right)$, where $e' = \{e_i \mid x_i \in \mathsf{Var}(C_1)\}$ and $e'' = \{e_j \mid x_j \in \mathsf{Var}(C_2)\}$.

We first establish that C' is syntactically multilinear. The construction for the base case is trivially syntactically multilinear. When $C = C_1 + C_2$, by induction hypothesis C'_1 and C'_2 are syntactically multilinear, so $C' = C'_1 + C'_2$ is also syntactically multilinear. For $C = C_1 \times C_2$, as $\mathsf{Var}(C_1) \cap \mathsf{Var}(C_2) = \emptyset$, we have $e' \cap e'' = \emptyset$. So, by definition $\mathsf{Var}(C'_1(X, e') \cap \mathsf{Var}(C'_2(X, e'')) = \emptyset$. As the e_i variables that appear in the product $\prod_{x_i \notin \mathsf{Var}(C_1) \cup \mathsf{Var}(C_2)} (1 - e_i)$ do not appear in either $C'_1(X, e')$ or $C'_2(X, e'')$, we conclude that C' is syntactically multilinear.

Now we consider the size/depth of C'. Each $+$ gate of C has a corresponding $+$ gate in C'. Each $\times$ gate in C is replaced by a product of n terms in C'. Hence $\mathsf{size}(C') \le O(n \times \mathsf{size}(C))$. The products add a $O(\log n)$ multiplicative factor to the depth. Since the n-term product can be computed using staggering with just one additional gate per layer, we have $\mathsf{width}(C') \le \mathsf{width}(C) + 1$. Moreover, if C is a formula to begin with, then so is C'. Thus all the claims in the Theorem are established. □

Consequently, it follows from Raz (2009) that we have no analogue of Hammon's observation for the permanent with $f \in \mathsf{sm\text{-}VNC}^1$.

Corollary 6.6. *The Permanent and the Determinant polynomials cannot be expressed as a coefficient of some monomial of a polynomial computed by a syntactically multilinear arithmetic formula of polynomial size.*

6.2. Stability. Following Malod (2007), we say a complexity class $\mathsf{sm\text{-}}\mathcal{C}$ is *stable for coefficient functions* if it satisfies the following two conditions:

1. $\mathsf{sm\text{-}}\mathcal{C}$ is closed under taking coefficients, and

2. Whenever $\mathsf{mcoef}(f, e) \in \mathsf{sm}\text{-}\mathcal{C}$, then $f \in \mathsf{sm}\text{-}\mathcal{C}$.

For a multilinear polynomial $f(x, e)$, let $\Sigma(E)\ f$ denote $\sum_{b\in\{0,1\}^m} f(x, b)$. We say a complexity class $\mathsf{sm}\text{-}\mathcal{C}$ is *closed under taking exponential sums*, if whenever $f(x, e) \in \mathsf{sm}\text{-}\mathcal{C}$, then $\Sigma(E)f \in \mathsf{sm}\text{-}\mathcal{C}$. One can obtain the permanent as $\Sigma(E)\ f$, for $f \in \mathsf{VNC}^1$ Valiant (1982), cf. Bürgisser (2000). But the situation is contrary for the syntactically multilinear case, because of the following theorem.

THEOREM 6.7. *The following syntactically multilinear classes are closed under exponential sums, and hence are stable for coefficient functions:* sm-VP, sm-VBP, sm-VNC^1, sm-VSC^i, sm-VBWBP, *and* sm-VAC^i, *for all* $i \geq 0$.

The theorem is an easy consequence of the following straightforward proposition, by patching a given circuit at gates with constant multiplications of appropriate powers of two.

PROPOSITION 6.8. *Let f and g be multilinear polynomials over X and E. Then*

$$\Sigma(E)\ (f + g) = 2^a\Sigma(\mathsf{Var}(f) \cap E)\ f + 2^b\Sigma(\mathsf{Var}(g) \cap E)\ g,$$

where $a = |E - \mathsf{Var}(f)|$ and $b = |E - \mathsf{Var}(g)|$. Furthermore, if f and g are defined on disjoint variables sets, then

$$\Sigma(E)\ fg = 2^c \left[\Sigma(\mathsf{Var}(f) \cap E)\ f\right] \cdot \left[\Sigma(\mathsf{Var}(g) \cap E)\ g\right]$$

where $c = |E| - |\mathsf{Var}(f) \cup \mathsf{Var}(g)|$.

7. Conclusion and Open Questions

We have studied the relationships among syntactically multilinear arithmetic circuit classes. In the syntactically multilinear world, the relationship VBWBP = VNC^1 $\subseteq$ VsSC^0 gets reversed, *i.e.* sm-VBWBP = sm-VsSC^0 $\subseteq$ sm-VNC^1. Except the simulation from arithmetic formulas to constant width branching programs Ben-Or & Cleve (1992), all known equivalences translate into the multilinear world.

We have sm-VsSC^0 = sm-VBWBP $\subseteq$ sm-VNC^1 $\subseteq$ sm-VLWBP $\subseteq$ sm-VBP. Can any one of these containments be shown to be strict? To separate sm-VNC^1 from sm-VBP, it would be sufficient to show that the full rank polynomial of Raz & Yehudayoff (2008) can be computed by syntactically multilinear ABPs. The separation of sm-VBWBP from sm-VBP would also be interesting, though this will be slightly weaker than separating sm-VNC^1 from sm-VBP. One reason why the separation of sm-VBWBP from sm-VBP would be interesting and possible is that they are defined over the same model, algebraic branching programs. The result of Raz (2006); Raz & Yehudayoff (2008) can be seen as separation of constant + fan-in circuits from polynomial fan-in circuits at logarithmic depth and polynomial size. Analogously, separating sm-VBWBP from sm-VBP can be viewed as separating constant width from polynomial width in ABPs of polynomial size.

Acknowledgements

The results in this paper were announced in MFCS 2008 and CSR 2009, in Jansen & Rao (2009); Mahajan & Rao (2008). We thank the anonymous referees of MFCS 2008, CSR 2009, and of this journal for their insightful comments and suggestions for improving the presentation of the paper. This work was done while the last author was at the Institute of Mathematical Sciences, Chennai.

References

Eric Allender, Andris Ambainis, David A.Mix Barrington, Samir Datta & Huong LêThanh (1999). Bounded Depth Arithmetic circuits: Counting and Closure. In *International Colloquium on Automata, Languages, and Programming ICALP*, ICALP'99, 149–158.

David A. Mix Barrington, Neil Immerman & Howard Straubing (1990). On uniformity within NC^1. *Journal of Computer and System Sciences* **41**(3), 274 – 306. ISSN 0022-0000.

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

A. Borodin, A. Razborov & R. Smolensky (1993). On lower bounds for read-k-times branching programs. *Comput. Complex.* **3**(1), 1–18. ISSN 1016-3328.

Richard P. Brent (1973). 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)*, 83–102. Academic Press, New York.

Peter Bürgisser (2000). *Completeness and Reduction in Algebraic Complexity Theory.* Algorithms and Computation in Mathematics. Springer-Verlag.

Hervé Caussinus, Pierre McKenzie, Denis Thérien & Heribert Vollmer (1998). Nondeterministic NC^1 Computation. *Journal of Computer and System Sciences* **57**, 200–212.

Stephen A. Cook (1979). Deterministic CFL's Are Accepted Simultaneously in Polynomial Time and Log Squared Space. In *Proceedings of the ACM Symposium on Theory of Computing STOC*, 338–345.

Sorin Istrail & Dejan Zivkovic (1994). Bounded width polynomial size Boolean formulas compute exactly those functions in AC^0. *Information Processing Letters* **50**, 211–216.

Maurice Jansen & B.V.Raghavendra Rao (2009). Simulation of Arithmetical Circuits by Branching Programs Preserving Constant Width and Syntactic Multilinearity. In *CSR, LNCS Vol. 5675*, 179–190.

Maurice J. Jansen (2008). Lower Bounds for Syntactically Multilinear Algebraic Branching Programs. In *MFCS LNCS vol. 5162*, 407–418.

David S. Johnson (1990). A Catalog of Complexity Classes. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity (A)*, Jan van Leeuwen, editor, 67–161. Elsevier and MIT Press.

E. Kaltofen & P. Koiran (2008). Expressing a fraction of two determinants as a determinant. In *Proceedings, The 19th International Symposium on Symbolic and Algebraic and Computation (ISSAC)*, 141–146.

Nutan Limaye, Meena Mahajan & B. V. Raghavendra Rao (2010). Arithmetizing Classes Around NC^1 and L. *Theory of Computing Systems* **46**(3), 499–522. Preliminary version in STACS 2007, LNCS vol. 4393 pp. 477–488.

Meena Mahajan & B. V. Raghavendra Rao (2008). Arithmetic Circuits, Syntactic Multilinearity and Skew formulae. In *MFCS, LNCS vol. 5162*, 455–466. Full version in ECCC TR08-048.

Guillaume Malod (2007). The Complexity of Polynomials and Their Coefficient Functions. In *IEEE Conference on Computational Complexity*, 193–204.

Guillaume Malod & Natacha Portier (2008). Characterizing Valiant's algebraic complexity classes. *Journal of Complexity* **24**(1), 16–38.

Noam Nisan (1991). Lower bounds for non-commutative computation. In *STOC '91: Proceedings of the twenty-third annual ACM symposium on Theory of computing*, 410–418. ACM, New York, NY, USA. ISBN 0-89791-397-3.

NOAM NISAN & AVI WIGDERSON (1997). Lower Bounds on Arithmetic Circuits Via Partial Derivatives. *Computational Complexity* **6**(3), 217–234.

RAN RAZ (2006). Separation of Multilinear Circuit and Formula Size. *Theory of Computing* **2**(1), 121–135. Preliminary version in FOCS 2004.

RAN RAZ (2009). Multi-linear formulas for permanent and determinant are of super-polynomial size. *J. ACM* **56**, 8:1–8:17. ISSN 0004-5411. Preliminary version in STOC 2004.

RAN RAZ, AMIR SHPILKA & AMIR YEHUDAYOFF (2008). A Lower Bound for the Size of Syntactically Multilinear Arithmetic Circuits. *SIAM J. Comput.* **38**(4), 1624–1647.

RAN RAZ & AMIR YEHUDAYOFF (2008). Balancing Syntactically Multilinear Arithmetic Circuits. *Computational Complexity* **17**(4), 515–535. ISSN 1016-3328.

SEINOSUKE TODA (1992). Classes of arithmetic circuits capturing the complexity of computing the determinant. *IEICE Transactions on Informations and Systems* **E75-D**, 116–124.

LESLIE G. VALIANT (1982). Reducibility by algebraic projections. *Logic and Algorithmic: an International Symposium held in honour of Ernst Specker* **30**, 365–380.

LESLIE G. VALIANT, SVEN SKYUM, S. BERKOWITZ & CHARLES RACKOFF (1983). Fast Parallel Computation of Polynomials Using Few Processors. *SIAM J. Comput.* **12**(4), 641–644.

V VINAY (1996). Hierarchies of Circuit Classes that are Closed Under Complement. In *Proceedings of the 11th Annual IEEE Conference on Computational Complexity CCC*, 108–117. IEEE Computer Society, Washington, DC, USA.

H. VOLLMER (1999). *Introduction to Circuit Complexity: A Uniform Approach.* Springer-Verlag New York Inc.

Manuscript received 14 July 2010

MAURICE JANSEN
Laboratory for Foundations of Computer Science
School of Informatics
The University of Edinburgh
maurice.julien.jansen@gmail.com
http://homepages.inf.ed.ac.uk/mjansen1/

MEENA MAHAJAN
The Institute of Mathematical Sciences
Chennai 600113, India
meena@imsc.res.in http://www.imsc.res.in/~meena/

B. V. Raghavendra Rao
Universität des Saarlandes, Informatik
66041 Saarbrücken, Germany
bvrr@cs.uni-sb.de http://www.imsc.res.
in/~bvrr/