Electronic Colloquium on Computational Complexity, Revision 2 of Report No. 94 (2022)

Notes on Boolean Read-k and Multilinear Circuits *

Stasys Jukna†
Faculty of Mathematics and Computer Science, Vilnius University, Lithuania

Abstract

A monotone Boolean $(\vee,\wedge)$ circuit computing a monotone Boolean function f is a read-k circuit if the polynomial produced (purely syntactically) by the arithmetic $(+,\times)$ version of the circuit has the property that for every prime implicant of f, the polynomial contains a monomial with the same set of variables, each appearing with degree $\leqslant k$. Every monotone circuit is a read-k circuit for some k. We show that monotone read-1 circuits have the same power as: tropical $(\min,+)$ circuits solving 0/1 minimization problems, monotone arithmetic $(+,\times)$ circuits computing multilinear homogeneous polynomials, as well as non-monotone $(\vee,\wedge,\neg)$ circuits computing monotone homogeneous Boolean functions. Finally, we show that already monotone read-2 circuits can be exponentially smaller than monotone read-1 circuits.

Keywords: arithmetic circuit, multilinear circuit, tropical circuit, lower bounds
MSC: 68Q17, 94C11

1 Introduction and Results

Proving lower bounds on the size of arithmetic $(+,\times,-)$ circuits as well as of Boolean $(\vee,\wedge,\neg)$ circuits remains a notoriously hard problem. Although the problem has received a great deal of attention for decades, the best lower bounds we know for arithmetic circuits are barely super-linear, while for Boolean circuits the lower bounds are even not super-linear. This happens mainly because general circuits can use cancellations $x-x=0$ in the arithmetic, and can use cancellations $x\wedge\bar{x}=0$ in the Boolean case. Understanding the role of cancellations in arithmetic and Boolean circuits remains the main goal of circuit complexity.

Monotone arithmetic circuits cannot use cancellations $x-x=0$, while monotone Boolean circuits cannot use cancellations $x\wedge\bar{x}=0$. Still, the task of proving lower bounds for monotone *Boolean* circuits turned out to be much more difficult than that for monotone *arithmetic* circuits. This happens because Boolean circuits can use idempotence lows $x\vee x=x$ and $x\wedge x=x$ as well as the absorption low $x\vee xy=x$, while arithmetic circuits cannot use any of these lows. It turned out that the absence of *additive* idempotence $x\vee x=x$ in arithmetic circuits (where $x+x$ is $2x$, not x) is not a crucial issue: most (albeit not all) lower bounds on the monotone arithmetic $(+,\times)$ circuit complexity are proved by only using the *structure* of monomials and fully ignoring actual values of their (nonzero) coefficients. But the absence of *multiplicative* idempotence $x\wedge x=x$ together with absorption $x\vee xy=x$ in the arithmetic world turned out to already be crucial even in the case of monotone circuits.

*Revised version. Added Corollary 1, Section 6.2, Facts 1 and 3 to 5, Remarks 4 and 5.

†Email: stjukna@gmail.com. Homepage: https://web.vu.lt/mif/s.jukna/.

ISSN 1433-8092

The goal of this article is to show that already a *very restricted* use of multiplicative idempotence $x \wedge x = x$ (in combination with absorption) makes a big difference between Boolean and arithmetic circuits. To fine grain the "degree of idempotence," we introduce so-called "read-k" $(\vee,\wedge)$ circuits.

Every monotone Boolean $(\vee,\wedge)$ circuit F defines a unique arithmetic polynomial in a natural way: replace every OR gate with an addition gate, and every AND gate with a multiplication gate. The obtained monotone arithmetic $(+,\times)$ circuit produces (purely syntactically) some polynomial P. If f is the monotone Boolean function computed by F, then the polynomial P has the following two properties: (i) every monomial of P contains all variables of at least one prime implicant of f, and (ii) every prime implicant $\bigwedge_{i\in I} x_i$ of f has at least one its "shadow" in P, that is, a monomial $\prod_{i\in I} x_i^{d_i}$ with the *same* set of variables (this is a "folklore" observation, see Fact 2). Property (i) reflects the absorption property $x \vee xy = x$, while property (ii) reflects the multiplicative idempotence $x \wedge x = x$. In *read-k* circuits we restrict property (ii) and require that every prime implicant of f has at least one "shadow" monomial in P with the degree $d_i \leqslant k$ of each its variable. There are no restriction on the degrees of other monomials of P.

We show that already read-1 circuits are tightly related to dynamic programming (DP). Many classical DP algorithms for minimization problems are "pure" in that they only use $(\min,+)$ operations in their recursion equations. Prominent examples of pure DP algorithms are the Bellman–Ford–Moore shortest *s-t* path algorithm [2, 7, 21], the Roy–Floyd–Warshall all-pairs shortest paths algorithm [6, 30, 36], the Bellman–Held–Karp travelling salesman algorithm [3, 10], the Dreyfus–Levin–Wagner Steiner tree algorithm [5, 19], and many others. Pure DP algorithms are (special, *recursively* constructed) tropical $(\min,+)$ circuits.

We also show that read-1 circuits are related to monotone arithmetic $(+,\times)$ circuits, as well as to so-called multilinear (not necessarily monotone) DeMorgan $(\vee,\wedge,\neg)$ circuits, where the Boolean functions computed at the inputs to any AND gate must *depend* on disjoint sets of variables. For example, $g = x \vee xy$ and $h = \bar{y} \vee x\bar{y}$ depend on disjoint sets of variables: g only depends on x, while h only depends on y.

Our main results are the following.

1. Read-1 $(\vee,\wedge)$ circuits have the *same* power as monotone *arithmetic* $(+,\times)$ circuits: the minimum size of a $(+,\times)$ circuit computing a homogeneous multilinear polynomial $P(x) = \sum_{S\in\mathcal{F}} c_S \prod_{i\in S} x_i$ with positive coefficients $c_S > 0$ *coincides with* the minimum size of a read-1 $(\vee,\wedge)$ circuit computing $f(x) = \bigvee_{S\in\mathcal{F}} \bigwedge_{i\in S} x_i$ (Theorem 1).

2. Read-1 $(\vee,\wedge)$ circuits have the *same* power as *tropical* $(\min,+)$ circuits solving $0/1$ minimization problems: the minimum size of a $(\min,+)$ circuit solving a minimization problem $P(x) = \min_{S\in\mathcal{F}} \sum_{i\in S} x_i$ *coincides with* the minimum size of a read-1 $(\vee,\wedge)$ circuit computing $f(x) = \bigvee_{S\in\mathcal{F}} \bigwedge_{i\in S} x_i$ (Theorem 2).

3. Read-1 $(\vee,\wedge)$ circuits are *not weaker* than multilinear $(\vee,\wedge,\neg)$ circuits: if a multilinear $(\vee,\wedge,\neg)$ circuit computes a Boolean function $f(x)$, then a not larger read-1 $(\vee,\wedge)$ circuit computes $f^{\triangledown}(x) := \bigvee_{z\leqslant x} f(z)$ (Theorem 3). If f is monotone and homogeneous, then the minimum size of a multilinear $(\vee,\wedge,\neg)$ circuit computing f *coincides* with the minimum size of a read-1 circuit computing f (Corollary 1).

4. Read-2 $(\vee,\wedge)$ circuits can be exponentially smaller than read-1 $(\vee,\wedge)$ circuits and, hence, exponentially smaller than tropical $(\min,+)$, monotone arithmetic $(+,\times)$, and multilinear $(\vee,\wedge,\neg)$ circuits (Lemma 9).

Organization: In the preliminary Section 2, we recall one simple but important concept—the set of exponent vectors "produced" (purely syntactically) by a circuit over any semiring. Read-k circuits

are introduced in Section 3. The aforementioned relation of read-1 circuits to monotone arithmetic circuits is established in Section 4, the relation of read-1 circuits to tropical circuits is established in Section 5, and the relation of read-1 circuits to multilinear DeMorgan $(\vee,\wedge,\neg)$ circuits is established in Section 6. In Section 7 we recall one relatively simple argument to show high lower bounds for monotone arithmetic $(+,\times)$ circuits. This is aimed to demonstrate the power of idempotence and absorption in Boolean circuits. An exponential gap between read-1 and read-2 circuits is shown in Section 8. All proofs are fairly simple.

2 Preliminaries

Recall that a (commutative) *semiring* $(R,\oplus,\odot)$ consists of a set R closed under two associative and commutative binary operations "addition" $x\oplus y$ and "multiplication" $x\odot y$, where multiplication distributes over addition: $x\odot(y\oplus z)=(x\odot y)\oplus(x\odot z)$. That is, in a semiring, we can "add" and "multiply" elements, but neither "subtraction" nor "division" are necessarily possible. We will assume that semirings contain a multiplicative neutral element $\mathbb{1}\in R$ such that $x\odot\mathbb{1}=\mathbb{1}\odot x=x$.

A *circuit* F over a semiring $(R,\oplus,\odot)$ is a directed acyclic graph; parallel edges joining the same pair of nodes are allowed. Each indegree-zero node (an *input* node) holds either one of the variables $x_1,\ldots,x_n$ or a semiring element $c\in R$. Every other node, a *gate*, has indegree two and performs one of the two semiring operations $\oplus$ or $\odot$ on the values computed at the two gates entering this gate. The *size* of a circuit is the total number of gates in it. A circuit F *computes* a function $f:R^n\to R$ if $F(x)=f(x)$ holds for all $x\in R^n$.

In this article, we will consider circuits over the following three semirings $(R,\oplus,\odot)$: the arithmetic semiring $(\mathbb{R}_+,+,\times)$, where $\mathbb{R}_+$ is the set of nonnegative real numbers, the tropical semiring $(\mathbb{R}_+,\min,+)$, and the Boolean semiring $(\{0,1\},\vee,\wedge)$. That is, we will consider the following three types of circuits[1]:

- $x\oplus y:=x+y$ and $x\odot y:=xy$ (monotone arithmetic circuits);
- $x\oplus y:=x\vee y$ and $x\odot y:=x\wedge y$ (monotone Boolean circuits);
- $x\oplus y:=\min(x,y)$ and $x\odot y:=x+y$ (tropical circuits).

Note that the multiplicative neutral element $\mathbb{1}$ is constant 1 in arithmetic and Boolean semirings, but is constant 0 in the tropical semiring (because $x+0=x$).

Produced polynomials Every circuit $F(x_1,\ldots,x_n)$ over a semiring $(R,\oplus,\odot)$ not only computes some function $f:R^n\to R$, but also *produces* (purely syntactically) an n-variate polynomial over this semiring in a natural way. Namely, at each source node u holding a constant $c\in R$, the constant polynomial $P_u=c$ is produced, and at a source node u holding a variable x_i, the polynomial $P_u=x_i$ is produced. At an "addition" gate $u=v\oplus w$, the "sum" $P_u=P_v\oplus P_w$ of the polynomials P_v and P_w produced at its inputs is produced. Finally, at a "multiplication" gate $u=v\odot w$, the polynomial P_u obtained from $P_v\odot P_w$ by distributivity of $\odot$ over $\oplus$ is produced; that is, we "multiply" $(\odot)$ every monomial of P_v with every monomial of P_w, and take the "sum" $(\oplus)$ of the obtained monomials. No terms are canceled along the way. The polynomial produced by the entire circuit F is the polynomial

$$P(x)=\sum_{b\in B}c_b\prod_{i=1}^{n}x_i^{b_i} \tag{1}$$

produced at its output gate. Here, $B\subseteq\mathbb{N}^n$ is some set of *exponent vectors*, x_i^k stands for the k-times "multiplication" $x_i\odot x_i\odot\cdots\odot x_i$, and $x_i^0=\mathbb{1}$ (the multiplicative neutral element).

[1] An exception is Section 6, where we also consider non-monotone Boolean $(\vee,\wedge,\neg)$ circuits.

Produced sets of exponent vectors Of interest for us will be not as much the polynomials P_v produced at gates v of a circuit F themselves but rather the sets $B_v \subseteq \mathbb{N}^n$ of exponent vectors of these polynomials. These sets are inductively obtained as follows, where $\vec{0}$ is the all-0 vector, and $\vec{e}_i \in \{0,1\}^n$ has exactly one 1 in the ith position:

- if v is an input node holding a constant $c \in R$, then $B_v = \{\vec{0}\}$;
- if v is an input node holding a variable x_i, then $B_v = \{\vec{e}_i\}$;
- if $v = u \oplus w$, then $B_v = B_u \cup B_w$ (set-theoretic union of sets B_u and B_w);
- if $v = u \odot w$, then $B_v = B_u + B_w := \{a+b\colon a \in B_u, b \in B_w\}$ (Minkowski sum of sets B_u and B_w).

The set of exponent vectors produced by the entire circuit F is the set $B = B_v$ of vectors produced at the output gate v of F. Since the exponent vector of a "*product*" $(\odot)$ of two monomials is the *sum* of their exponent vectors, this set B is exactly the set of exponent vectors of the polynomial Eq. (1) produced by the circuit F.

It is clear that the same circuit F with "addition" $(\oplus)$ and "multiplication" $(\odot)$ gates can compute *different* functions over different semirings. Say, the circuit $F = (x \odot y) \oplus z$ computes $xy+z$ over the arithmetic $(+,\times)$ semiring, but computes $\min\{x+y,z\}$ over the tropical $(\min,+)$ semiring. It is, however, important to note that:

- The set of exponent vectors of the polynomial *produced* by a circuit over any semiring is always the same—it only depends on the circuit itself, not on the underlying semiring.

Notation. We will use standard terminology and notation regarding Boolean functions (see, for example, the books by Wegener [37] or Crama and Hammer [4]). In particular, for two Boolean functions $f,g : \{0,1\}^n \to \{0,1\}$, we write $g \leqslant f$ iff $g(a) \leqslant f(a)$ holds for all $a \in \{0,1\}^n$; hence, $g = f$ iff $g(a) = f(a)$ holds for all $a \in \{0,1\}^n$. A *term* is an AND of *literals*, each being a variable x_i or its negation $\bar{x}_i$. A term is a *zero term* if it contains a variable and its negation. An *implicant* of a Boolean function $f(x_1,\ldots,x_n)$ is a nonzero term t such that $t \leqslant f$ holds, that is, $t(a) = 1$ implies $f(a) = 1$. An implicant t of f is a *prime implicant* of f if no proper subterm t' of t has this property, that is, if $t \leqslant t' \leqslant f$, then $t' = t$. For example, if $f = xy \vee x\bar{y}z$, then xy, $x\bar{y}z$ and xz are implicants of f, but $x\bar{y}z$ is not a prime implicant (since $x\bar{y}z \leqslant xz$). A Boolean function f is *monotone* if $a \leqslant b$ implies $f(a) \leqslant f(b)$.

3 Read-k Circuits

A monotone Boolean circuit is a circuit over the Boolean semiring $(R,\oplus,\odot)$ with $x \oplus y := x \vee y$ and $x \odot y := x \wedge y$; the domain is $R = \{0,1\}$. We will always assume that such circuits are constant-free: constant inputs 0 and 1 can be easily eliminated without increasing the circuit size (to avoid trivialities, we will only consider circuits computing non-constant functions).

A *lowest one* of a (not necessarily monotone) Boolean function $f : \{0,1\}^n \to \{0,1\}$ is a vector $a \in \{0,1\}^n$ such that $f(a) = 1$ but $f(b) = 0$ for all vectors $b \leqslant a$, $b \neq a$. We will denote the set of all lowest ones of f by A_f, that is, $A_f := \{a \in f^{-1}(1)\colon\ f(b) = 0 \text{ for all } b \leqslant a,\ b \neq a\}$. Note that the set A_f is always an *antichain*: $a \in A$ and $b \leqslant a$ imply $a = b$.

In this section, we will only consider monotone Boolean functions. Note that if the function f is monotone, then A_f uniquely determines the entire function f:

$$f(x_1,\ldots,x_n) = \bigvee_{a \in A_f} \bigwedge_{i \in \sup(a)} x_i\,,$$

where here and throughout, $\sup(x) := \{i \in [n]\colon x_i \neq 0\}$ stands for the *support* of a vector $x \in \mathbb{R}^n$, that is, for the set of its nonzero positions. The following fact is well known (see, for example, [4, Theorem 1.21]).

Fact 1 (Folklore)**.** *Prime implicants of monotone Boolean functions do not contain negated variables.*

Proof. Let $f(x_1,\ldots,x_n)$ be a monotone Boolean function. Suppose for a contradiction that $t=\bar{x}_i t'$ is a prime implicant of f. Since t is prime, t' is not an implicant of f. That is, there is a vector $a\in\{0,1\}^n$ such that $t'(a)=1$ but $f(a)=0$ and, hence, also $t(a)=0$. Thus, $a_i=1$. Since the function f is monotone, we also have $f(b)=0$ for the vector b obtained by flipping to 0 the ith bit of a. But $t(b)=1$ implies $f(b)=1$, a contradiction. □

Thus, for every monotone Boolean function f, we have:

$$A_f = \text{set of characteristic 0-1 vectors of prime implicants of } f.$$

The *upward closure* of a set $A\subseteq\mathbb{N}^n$ of vectors is the set $A^{\uparrow}:=\{b\in\mathbb{N}^n\colon b\geqslant a \text{ for some } a\in A\}$ of integer vectors containing at least one vector of A. A *shadow* of a vector $a\in\mathbb{N}^n$ is any vector $b\in\mathbb{N}^n$ with $\sup(b)=\sup(a)$. For a set $A\subseteq\mathbb{N}^n$ of vectors, let $\mathrm{Sup}(A):=\{\sup(a)\colon a\in A\}\subseteq 2^{[n]}$ denote the family of supports of its vectors.

We start with the following simple (also "folklore") property of sets of exponent vectors produced by monotone Boolean circuits. Let f be the monotone Boolean function, and $A_f\subseteq f^{-1}(1)$ be the set of lowest ones of f. Let also F be a monotone Boolean $(\vee,\wedge)$ circuit, and $B\subseteq\mathbb{N}^n$ be the set of exponent vectors produced by F, that is, the set of exponent vectors of the polynomial produced by the monotone arithmetic $(+,\times)$ version of F.

Fact 2. *The circuit F computes f if and only if $B\subseteq(A_f)^{\uparrow}$ and $\mathrm{Sup}(A_f)\subseteq\mathrm{Sup}(B)$.*

Proof. Let $A:=A_f$. Hence, our Boolean function f is of the form $f(x)=\bigvee_{a\in A}\bigwedge_{i\in\sup(a)}x_i$, while the Boolean function computed by the circuit F is of the form $F(x)=\bigvee_{b\in B}\ \bigwedge_{i\in\sup(b)}x_i$. The "if" direction follows from the simple observation: for every input $x\in\{0,1\}^n$, we have $f(x)=1$ iff $\sup(x)\supseteq\sup(a)$ for some $a\in A$. Hence, $B\subseteq A^{\uparrow}$ yields $F(x)\leqslant f(x)$, while $\mathrm{Sup}(A)\subseteq\mathrm{Sup}(B)$ yields $f(x)\leqslant F(x)$.

Now assume that the circuit F computes f. If $b\notin A^{\uparrow}$ held for some vector $b\in B$, then on the input $x\in\{0,1\}^n$ with $x_i=1$ iff $i\in\sup(b)$, we would have $\sup(a)\setminus\sup(b)\neq\emptyset$ for all $a\in A$ and, hence, $f(x)=0$. But $F(x)=1$, a contradiction. To show the inclusion $\mathrm{Sup}(A)\subseteq\mathrm{Sup}(B)$, suppose for the contradiction that there is a vector $a\in A$ such that $\sup(b)\neq\sup(a)$ holds for all vectors $b\in B$. Since $B\subseteq A^{\uparrow}$ and since A is an antichain, $\sup(b)\subset\sup(a)$ (proper inclusion) cannot hold, for otherwise, the vector $a\in A$ would contain another vector of A. So, we have $\sup(b)\setminus\sup(a)\neq\emptyset$ for all vectors $b\in B$. But then $F(a)=0$ while $f(a)=1$, a contradiction. □

In general, shadows $b\in B$ of vectors $a\in A_f$ guaranteed by Fact 2 may have large entries (as large as 2^s, where s is the size of the Boolean circuit F computing f): we only know that $\sup(b)=\sup(a)$ holds. In read-k circuits, we restrict the magnitude of entries in shadows $b\in B$. A vector $b\in\mathbb{N}^n$ is *k-bounded* if no its entry is larger than k.

Definition 1 (Read-k circuits[2])**.** Let F be a monotone $(\vee,\wedge)$ circuit computing a Boolean function f, and let $B\subseteq\mathbb{N}^n$ be the set of exponent vectors produced by F. The circuit F is a *read-k circuit* if every lowest one $a\in A_f$ of f has at least one k-bounded its shadow in B. In particular, F is a *read*-1 circuit iff the inclusions $A\subseteq B\subseteq(A_f)^{\uparrow}$ hold.

That is, the (arithmetic) polynomial P produced by the monotone arithmetic $(+,\times)$ version of the $(\vee,\wedge)$ circuit F must now contain, for every prime implicant of f, at least one monomial with the

[2]The term "read-k circuit" is by analogy with the established term "read-k branching programs."

same set of variables *and* with each variable appearing in that monomial with a power not exceeding k. There are no restrictions on the degrees of other monomials of P. By Fact 2, every monotone $(\vee,\wedge)$ circuit of size s is a read-k circuit for some $k \leqslant 2^s$.

Remark 1. The arithmetic $(+,\times)$ version F' of a monotone Boolean $(\vee,\wedge)$ circuit F obtained by replacing OR gates by $+$ gates, AND gates by $\times$ gates. The polynomial produced by F' is of the form $P(x) = \sum_{S\in\mathcal{F}} c_S \prod_{i\in S} x_i^{d_i}$ for some family $\mathcal{F} \subseteq 2^{[n]}$, some integer constants $c_S \geqslant 1$ (telling how often the corresponding monomial $\prod_{i\in S} x_i^{d_i}$ appears in the polynomial), and all positive exponents $d_i \geqslant 1$. Fact 2 implies that F computes a given Boolean function f iff (i) for every monomial $\prod_{i\in S} x_i^{d_i}$ of P, the term $\bigwedge_{i\in S} x_i$ is an implicant of f (property $B \subseteq (A_f)^{\uparrow}$), and (ii) for every prime implicant $\bigwedge_{i\in I} x_i$ of f, the polynomial contains a monomial $\prod_{i\in S} x_i^{d_i}$ with $S = I$ (property $\mathrm{Sup}(A_f) \subseteq \mathrm{Sup}(B)$); in the case of read-$k$ circuits, we have an additional restriction $d_i \leqslant k$ for all $i \in I$ in the property (ii). □

Example 1. The Boolean circuit $F = (x\vee y)(x\vee z)\vee xy$ computes the Boolean function $f(x,y,z) = x\vee yz$, whose prime implicants are x and yz; hence, $A_f = \{(1,0,0),(0,1,1)\}$. The arithmetic version $F' = (x+y)(x+z)+xy$ of F produces the polynomial $P = x^2+2xy+xz+yz$. Hence, the set of exponent vectors produced by the Boolean circuit F is $B = \{(2,0,0),(1,1,0),(1,0,1),(0,1,1)\} \subseteq (A_f)^{\uparrow}$. The circuit is a read-2 but not read-1 circuit, because the only shadow x^2 of the prime implicant x of f in the polynomial P has degree 2.

For a monotone Boolean function f, let

$$\mathsf{B}_k(f) := \text{min size of a monotone read-}k\ (\vee,\wedge)\text{ circuit computing } f.$$

Remark 2. Note that already read-1 circuits are "universal:" every monotone Boolean function f *can* be computed by such a circuit (for example, as an OR of prime implicants of f). But read-k circuits for small k can be very inefficient: we will show in Section 8 that already the gap $\mathsf{B}_1(f)/\mathsf{B}_2(f)$ can be exponential. □

4 From Monotone Arithmetic to Boolean Read-1

The main difference of (monotone) arithmetic $(+,\times)$ circuits from Boolean and tropical circuits is that they produce what they compute. This can be easily shown using the following extension to multivariate polynomials of a basic fact that no univariate polynomial of degree d can have more than d roots (see, for example, Alon and Tarsi [1, Lemma 2.1]):

$(*)$ If P is a nonzero n-variate polynomial with every variable occurring with degree $\leqslant d$, and if $S \subseteq \mathbb{R}$ is a set of $|S| \geqslant d+1$ numbers, then $P(x) \neq 0$ holds for at least one point $x \in S^n$.

This is proved in [1] by an easy induction on the number n of variables.

Fact 3. *If a monotone arithmetic circuit* computes *a given polynomial, then the circuit also* produces *that polynomial.*

Proof. Let F be a monotone arithmetic circuit computing a polynomial P_1, and let P_2 be the polynomial produced by F. Since the circuit F is monotone, it has no negative constant inputs. So, the coefficients in both polynomials P_1 and P_2 are positive. Suppose for a contradiction that the polynomials P_1 and P_2 do not coincide (as formal expressions). Then $P = P_1 - P_2$ is a nonzero polynomial of a (possibly large but) *finite* degree, and $(*)$ implies that $P_1(x) - P_2(x) \neq 0$ for some $x \in \mathbb{R}_+^n$, a contradiction with our assumption that the circuit F computes the polynomial P_1. □

Say that two polynomials with positive coefficients are *similar* if they have the same monomials (with possibly different coefficients). An arithmetic circuit is *constat-free* if it has no constants as inputs. For a set $A \subseteq \mathbb{N}^n$ of vectors, let

$$\begin{aligned}\mathsf{Arith}(A) := {} & \text{smallest size of a monotone constant-free arithmetic circuit computing} \\ & \text{a polynomial similar to } P(x) = \textstyle\sum_{a\in A} \prod_{i=1}^n x_i^{a_i}.\end{aligned}$$

The *lower envelope* $\lfloor A \rfloor \subseteq A$ of a set $A \subseteq \mathbb{N}^n$ of vectors consists of all vectors $a \in A$ of smallest degree, where the *degree* of a vector $a \in \mathbb{N}^n$ be the sum $a_1 + \cdots + a_n$ of its entries. A set is $A \subseteq \mathbb{N}^n$ *homogeneous* of degree m, if all its vectors have the same degree m; note that then, $\lfloor A \rfloor = A$ holds. The *lower envelope* of a polynomial $P(x) = \sum_{a\in A} c_a \prod_{i=1}^n x_i^{a_i}$ is the polynomial $Q(x) = \sum_{a\in \lfloor A \rfloor} c_a \prod_{i=1}^n x_i^{a_i}$ consisting of terms of P of smallest degree.

Jerrum and Snir [12, Theorem 2.4] observed that, by appropriately discarding some addition gates, every monotone arithmetic circuit computing a polynomial can be transformed into a monotone arithmetic circuit computing its lower envelope.

Lemma 1 (Envelope lemma [12])**.** *For every $A \subseteq \mathbb{N}^n$, $\mathsf{Arith}(\lfloor A \rfloor) \leqslant \mathsf{Arith}(A)$.*

Proof. Take a monotone arithmetic circuit F computing a polynomial $P(x) = \sum_{a\in A} c_a \prod_{i=1}^n x_i^{a_i}$ with positive coefficients $c_a > 0$. By Fact 3, the circuit also produces the polynomial P. We can obtain a $(+,\times)$ circuit producing the lower envelope $Q(x) = \sum_{a\in \lfloor A \rfloor} c_a \prod_{i=1}^n x_i^{a_i}$ of P by appropriately discarding some of the edges entering addition $(+)$ gates. For a gate v in the circuit F, let $A_v \subseteq \mathbb{N}^n$ be the set of exponent vectors of the polynomial produced at v. If $v = u \times w$ is a multiplication gate, then $A_v = A_u + A_w$ (Minkowski sum). Since the degree of a sum of two vectors is the sum of their degrees, we have $\lfloor A_v \rfloor = \lfloor A_u + A_w \rfloor = \lfloor A_u \rfloor + \lfloor A_w \rfloor$. So, we do nothing in this case.

Now let $v = u + w$ be an addition gate. In this case, we have $A_v = A_u \cup A_w$. If the minimum degree of a vector in A_u is the *same* as the minimum degree of a vector in A_w, then $\lfloor A_v \rfloor = \lfloor A_u \cup A_w \rfloor = \lfloor A_u \rfloor \cup \lfloor A_w \rfloor$, and we do nothing in this case. If the minimum degree of a vector in A_u is *smaller* than the minimum degree of a vector in A_w, then $\lfloor A_v \rfloor = \lfloor A_u \cup A_w \rfloor = \lfloor A_u \rfloor$. In this case, we discard the edge (w,v): delete that edge, delete the $+$ operation labeling the gate v, and contract the edge (u,v). If the minimum degree of a vector in A_w is smaller than the minimum degree of a vector in A_u, then we discard the edge (u,v). ∎

A monotone Boolean function f is *homogeneous* if the set $A_f \subseteq f^{-1}(1)$ of its lowest ones is homogeneous (all prime implicants of f have the same number of variables); note that then $\lfloor A_f \rfloor = A_f$ holds.

Theorem 1. *For every monotone Boolean function f, $\mathsf{Arith}(\lfloor A_f \rfloor) \leqslant \mathsf{B}_1(f) \leqslant \mathsf{Arith}(A_f)$ hold. In particular, if f is homogeneous, then $\mathsf{Arith}(A_f) = \mathsf{B}_1(f)$.*

Proof. Let $A := A_f \subseteq f^{-1}(1)$ be the set of lowest ones of f; hence, $f(x) = \bigvee_{a\in A} \bigwedge_{i\in \sup(a)} x_i$. To show the first inequality $\mathsf{Arith}(\lfloor A \rfloor) \leqslant \mathsf{B}_1(f)$, let F be a monotone read-1 Boolean $(\vee,\wedge)$ circuit of size $s = \mathsf{B}_1(f)$ computing f, and let $B \subseteq \mathbb{N}^n$ be the set of exponent vectors produced by F. Consider the arithmetic $(+,\times)$ version F' of the circuit F obtained by replacing OR gates by $+$ gates, and AND gates by $\times$ gates. The arithmetic circuit F' produces the same set B of exponent vectors. Thus, $\mathsf{Arith}(B) \leqslant s$. Since the (Boolean) circuit F was a read-1 circuit, we know that the inclusions $A \subseteq B \subseteq A^{\uparrow}$ hold. This yields $\lfloor B \rfloor = \lfloor A \rfloor$. Thus, the polynomial $Q(x) = \sum_{a\in \lfloor A \rfloor} c_a \prod_{i=1}^n x_i^{a_i}$ is the lower envelope of the polynomial $P(x) = \sum_{b\in B} c_b \prod_{i=1}^n x_i^{b_i}$ produced by the circuit F', and Lemma 1 yields $\mathsf{Arith}(\lfloor A \rfloor) = \mathsf{Arith}(\lfloor B \rfloor) \leqslant \mathsf{Arith}(B) \leqslant s$.

To show the inequality $\mathsf{B}_1(f) \leqslant \mathsf{Arith}(A)$, let F be a monotone constant-free arithmetic $(+,\times)$ circuit of size $\mathsf{Arith}(A)$ computing some polynomial $P(x) = \sum_{a\in A} c_a \prod_{i=1}^n x_i^{a_i}$ whose set of exponent vectors is A. By Fact 3, the circuit F also (syntactically) produces this polynomial. Hence, A is the set of exponent vectors produced by F. Convert the arithmetic circuit F into a monotone Boolean $(\vee,\wedge)$ circuit: replace every addition gate with an OR gate, and every multiplication gate with an AND gate. The resulting Boolean circuit F' produces the same set A of exponent vectors. Hence, F' computes the Boolean version $f(x) = \bigvee_{a\in A} \bigwedge_{i\in \mathrm{sup}(a)} x_i$ of the polynomial P. □

5 From Tropical (min,+) to Boolean Read-1

In the tropical $(\min,+)$ semiring, powering $x_i^{a_i} = x_i \odot x_i \odot \cdots \odot x_i$ ($a_i \in \mathbb{N}$ times) turns into multiplication by scalars $a_i x_i = x_i + x_i + \cdots + x_i$. So, a generic monomial $\prod_{i=1}^n x_i^{a_i}$ turns into the tropical "monomial" $\langle a,x\rangle = a_1x_1 + \cdots + a_nx_n$, the scalar product of vectors a and x, and a polynomial $\sum_{a\in A} c_a \prod_{i=1}^n x_i^{a_i}$ turns into the *tropical* $(\min,+)$ polynomial $f(x) = \min_{a\in A}\langle a,x\rangle + c_a$ with "exponent" vectors $a \in A$ and "coefficients" c_a. The corresponding minimization problem (represented by such a polynomial) is *constant-free* if $c_a = 0$ for all $a \in A$. A $(\min,+)$ circuit F *solves* a given minimization problem f if $F(x) = f(x)$ holds for all input weightings $x \in \mathbb{R}^n_+$. A $(\min,+)$ circuit F is *constant-free* if it has no constants other than 0 as inputs. The *constant-free version* of a $(\min,+)$ circuit F is obtained by replacing all constant inputs with constant 0.

Lemma 2 ([16])**.** *If a $(\min,+)$ circuit F solves a constant-free minimization problem f, then the constant-free version of F also solves f.*

Proof. Let $f(x) = \min_{a\in A}\langle a,x\rangle$ be a (constant-free) polynomial computed by the circuit F, and let $g(x) = \min_{b\in B}\langle x,b\rangle + c_b$ be the tropical $(\min,+)$ polynomial produced by F. Since constant inputs can only affect the "coefficients" c_b, the polynomial produced by the constant-free version F^o of F is the constant-free version $g^o(x) = \min_{b\in B}\langle x,b\rangle$ of the polynomial $g(x)$. Since the circuit F computes f, we have $g(x) = f(x)$ for all input weightings $x \in \mathbb{R}^n_+$. We have to show that $g^o(x) = f(x)$ holds for all $x \in \mathbb{R}^n_+$ as well. Since the constants c_b are nonnegative, we clearly have $g^o(x) \leqslant g(x) = f(x)$ for all $x \in \mathbb{R}^n_+$. So, it remains to show that also $f(x) \leqslant g^o(x)$ holds for all $x \in \mathbb{R}^n_+$.

Suppose for the sake of contradiction that $f(x_0) > g^o(x_0)$ holds for some input weighting $x_0 \in \mathbb{R}^n_+$. Then the difference $d = f(x_0) - g^o(x_0)$ is positive. We can assume that the constant $c := \max_{b\in B} c_b$ is also positive, for otherwise, there would be nothing to prove. Take the scalar $\lambda := 2c/d > 0$. Since $g^o(x_0) = f(x_0) - d$, we obtain $g(\lambda x_0) \leqslant g^o(\lambda x_0) + c = \lambda \cdot g^o(x_0) + c = \lambda[f(x_0) - d] + c = f(\lambda x_0) - c$, which is strictly smaller than $f(\lambda x_0)$, a contradiction with $f(x) = g(x)$ for all $x \in \mathbb{R}^n_+$. □

Sets of "exponent" vectors produced by constant-free tropical $(\min,+)$ circuits have the following properties, which are even stronger than those for monotone Boolean circuits, as given by Fact 2.

Lemma 3. *Let F be a constant-free $(\min,+)$ circuit, $B \subseteq \mathbb{N}^n$ be the set of "exponent" vectors produced by F, and $A \subseteq \{0,1\}^n$ be an antichain. Then the circuit F solves the minimization problem $f_A(x) = \min_{a\in A}\langle a,x\rangle$ iff the inclusions $A \subseteq B \subseteq A^\uparrow$ hold.*

Proof. Since constant inputs can only affect the "coefficients," tropical polynomials produced by constat-free circuits are also constant-free. Thus, the minimization problem solved by the circuit F is of the form $f_B(x) = \min_{b\in B}\langle b,x\rangle$. For the "if" direction, suppose that the inclusions $A \subseteq B \subseteq A^\uparrow$ hold, and take an arbitrary input weighting $x \in \mathbb{R}^n_+$. Then $A \subseteq B$ implies $f_A(x) \geqslant f_B(x)$, while $B \subseteq A^\uparrow$ implies $f_A(x) \leqslant f_B(x)$ (the weights are nonnegative). Thus, the circuit F solves our minimization problem f_A

For the "only if" direction, assume that the circuit F solves the minimization problem f_A. Then, in particular, $f_A(x) = f_B(x)$ holds for all input weightings $x \in \{0,1,n+1\}^n$. To show the inclusion $B \subseteq A^{\uparrow}$, take an arbitrary vector $b \in B$, and consider the weighting $x \in \{0,1\}^n$ such that $x_i := 0$ for $i \in \sup(b)$, and $x_i := 1$ for $i \notin \sup(b)$. Take a vector $a \in A$ on which the minimum $f_A(x) = \langle a,x\rangle$ is achieved. Then $\langle a,x\rangle = f_A(x) = f_B(x) \leqslant \langle b,x\rangle = 0$. Thus, $\sup(a) \subseteq \sup(b)$. Since a is a 0-1 vector and $b \in \mathbb{N}^n$, this yields $b \geqslant a$, as desired.

To show the inclusion $A \subseteq B$, take an arbitrary vector $a \in A$, and consider the weighting $x \in \{1,n+1\}^n$ with $x_i := 1$ for all $i \in \sup(a)$ and $x_i := n+1$ for all $i \notin \sup(a)$. Take a vector $b \in B$ for which $\langle b,x\rangle = f_B(x)$ holds. Hence, $\langle b,x\rangle = f_B(x) = f_A(x) \leqslant \langle a,x\rangle = \langle a,a\rangle \leqslant n$. If $b_i \geqslant 1$ held for some $i \notin \sup(a)$, then we would have $\langle b,x\rangle \geqslant b_i x_i = b_i(n+1) > n$, a contradiction. Thus, $\sup(b) \subseteq \sup(a)$. Since $B \subseteq A^{\uparrow}$, there is a vector $a' \in A$ such that $a' \leqslant b$. Hence, $\sup(a') \subseteq \sup(b) \subseteq \sup(a)$. Since both a and a' are 0-1 vectors, this yields $a' \leqslant a$. Since the set A is an antichain, we have $a' = a$, and the equality $\sup(b) = \sup(a)$ follows. Thus, $\langle b,a\rangle = \langle b,x\rangle \leqslant \langle a,a\rangle$. Together with $\sup(b) = \sup(a)$ and $b \in \mathbb{N}^n$, we have $b = a$, meaning that our vector $a \in A$ belongs to the set B, as desired. □

For a finite set $A \subseteq \mathbb{N}^n$ of vectors, let

$$\mathsf{Min}(A) := \text{smallest size of a } (\min,+) \text{ circuit computing } f_A(x) = \min_{a\in A}\langle a,x\rangle.$$

Theorem 2. *For every monotone Boolean function f, we have $\mathsf{Min}(A_f) = \mathsf{B}_1(f)$.*

Proof. Let $A := A_f \subseteq f^{-1}(1)$ be the set of lowest ones of f; hence, A is an antichain and $f(x) = \bigvee_{a\in A}\bigwedge_{i\in\sup(a)} x_i$. To show the inequality $\mathsf{Min}(A) \leqslant \mathsf{B}_1(f)$, take a monotone read-1 $(\vee,\wedge)$ circuit F of size $\mathsf{B}_1(f)$ computing the Boolean function f, and let $B \subseteq \mathbb{N}^n$ be the set of exponent vectors produced by F. By Fact 2, we have $B \subseteq A^{\uparrow}$. Since the circuit F is a read-1 circuit, we also have the inclusion $A \subseteq B$. The tropical $(\min,+)$ version F' of F (obtained by replacing OR gates with min gates, and AND gates with addition gates) is constant-free and produces the same set B of "exponent" vectors. Since the inclusions $A \subseteq B \subseteq A^{\uparrow}$ hold, Lemma 3 implies that the circuit F' solves the minimization problem $f_A(x) = \min_{a\in A}\langle a,x\rangle$. Hence, $\mathsf{Min}(A) \leqslant \mathsf{B}_1(f)$ holds.

To show the inequality $\mathsf{B}_1(f) \leqslant \mathsf{Min}(A)$, take a tropical $(\min,+)$ circuit F of size $\mathsf{Min}(A)$ solving the minimization problem $f_A(x) = \min_{a\in A}\langle a,x\rangle$, and let $B \subseteq \mathbb{N}^n$ be the set of exponent vectors produced by the circuit F. By Lemma 2, we can assume that the circuit F is constant-free. So, the polynomial $f_B(x) = \min_{b\in B}\langle b,x\rangle$ produced by F is also constant-free. Since the circuit F solves the problem f_A, and since $A = A_f$ is an antichain, Lemma 3 gives us the inclusions $A \subseteq B \subseteq A^{\uparrow}$. The Boolean $(\vee,\wedge)$ version F' of F (obtained by replacing min gates by OR gates, and addition gates by AND gates) produces the same set B of "exponent" vectors. Together with Fact 2 and the definition of read-1 circuits, inclusions $A \subseteq B \subseteq A^{\uparrow}$ imply that F' is a read-1 circuit and computes the Boolean function f. □

6 From Non-Monotone Multilinear to Monotone Read-1

Due to the lack of strong lower bounds for general (non-monotone) arithmetic $(+,\times,-)$ circuits, and because they seem to be the most intuitive circuits for computing *multilinear* polynomial functions, a successful approach has been to consider a restriction called "multilinearity" of arithmetic circuits, first introduced by Nisan and Wigderson [22].

Recall that a polynomial in the ring $\mathbb{R}[x_1,\ldots,x_n]$ is multilinear if in each of its monomials, the power of every input variable is at most one. An arithmetic $(+,\times,-)$ circuit F is *syntactically multilinear* if the formal polynomial produced at each gate of the circuit is multilinear. A natural

relaxation was to call the circuit F (semantically) *multilinear* if the polynomial *functions* computed by the produced polynomials are multilinear. For example, the polynomial function $f = y$ computed at a gate producing a formal polynomial $P = x^2 + y - x^2$ is multilinear. In other words, the circuit F is semantically multilinear if the polynomial *functions* g and h computed at the inputs to any multiplication $(\times)$ gate depend on disjoint sets of variables. (Multilinearity can only be destroyed at multiplication gates.) Monotone $(+,\times)$ circuits do not have cancellations $x - x = 0$; hence, monotone semantically multilinear $(+,\times)$ circuits are also syntactically multilinear. Raz [25, Proposition 2.1] observed that also minimal semantically multilinear $(+,\times,-)$ *formulas* (circuits with fanout $\leqslant 1$ gates) are syntactically multilinear. However, it is not clear if every semantically multilinear *circuit* can be efficiently simulated by a syntactically multilinear circuit.

There are several impressing results concerning multilinear (as well as syntactically multilinear) arithmetic $(+,\times,-)$ circuits and formulas; see, for example, the survey [34]. In particular, Raz [25] proved that any multilinear arithmetic *formula* computing the permanent or the determinant of an $n \times n$ matrix is of size $n^{\Omega(\log n)}$. Furthermore, Raz [24] proved that a gap between multilinear arithmetic *formulas* and *circuits* can be super-polynomial. Proving super-polynomial lower bounds for the size of multilinear arithmetic *circuits* remains an open problem.

Due to the lack of even super-linear lower bounds for (unrestricted) DeMorgan $(\vee,\wedge,\neg)$ circuits, and by analogy with arithmetic circuits, the multilinearity restriction was also imposed on DeMorgan circuits. Recall that a DeMorgan $(\vee,\wedge,\neg)$ circuit F fanin-2 AND and OR gates; inputs are the variables $x_1,\ldots,x_n$ and their negations $\bar{x}_1,\ldots,\bar{x}_n$. As before, the *size* of a circuit is the total number of gates in it. A *monotone* Boolean circuit is a DeMorgan circuit without negated input literals as inputs. Being DeMorgan is not a real restriction: every Boolean $(\vee,\wedge,\neg)$ circuit (with negations applied not necessary to only inputs) can be easily transformed into an equivalent DeMorgan $(\vee,\wedge,\neg)$ circuit by only doubling the circuit size (see, e.g., [37, p. 195]): we double all AND and OR gates, one output of a pair is negated, the other one not; after that we can apply the DeMorgan rules without increasing the number of gates.

A DeMorgan $(\vee,\wedge,\neg)$ circuit F is *syntactically multilinear* if the two subcircuits rooted at inputs of any AND gate have no input literals of the same variable in common. For example, the circuit $F = (x \vee x\bar{y})y$ is *not* syntactically multilinear. Sengupta and Venkateswaran [32] considered the *connectivity* function which accepts an input $x \in \{0,1\}^{\binom{n}{2}}$ iff the graph G_x specified by the 0-1 input vector x is connected. By adopting the proof of Jerrum and Snir [12] of a lower bound $(4/3)^{n-1}/n$ on the minimum size of monotone arithmetic circuits computing the directed spanning tree polynomial, it was shown in [32] that every monotone syntactically multilinear Boolean circuits computing the connectivity function must also have at least $(4/3)^{n-1}/n$ gates. Krieger [17] has shown that minimal syntactically multilinear $(\vee,\wedge,\neg)$ circuits computing *monotone* functions f are *monotone*, and that if the set A_f of lowest ones of a monotone Boolean function f is cover-free (that is, if $a,b,c \in A_f$ and $a+b \geqslant c$ imply $c \in \{a,b\}$), then every syntactically multilinear $(\vee,\wedge,\neg)$ circuit computing f must have at least $|A_f| - 1$ gates.

Remark 3. Already in 1976, Schnorr [31] has proved a lower bound $\mathsf{Arith}(A) \geqslant |A| - 1$ on monotone *arithmetic* circuit complexity of polynomials, whose sets A of exponent vector are cover-free. This surprising similarity of the bound in [17] with Schnorr's bound, as well as a possibility to adopt the argument of Jerrum and Snir in [32], served as an indication that there "should" be some *general* relation between multilinear Boolean $(\vee,\wedge,\neg)$ and monotone arithmetic circuits. Our Theorem 3 below gives such a relation, even for semantically (not only syntactically) multilinear $(\vee,\wedge,\neg)$ circuits: such circuits are not stronger than monotone arithmetic circuits.

Following the analogy with arithmetic circuits, Ponnuswami and Venkateswaran [23] relaxed the

syntactic multilinearity restriction of Boolean circuits to their *semantic* multilinearity. Recall that a Boolean function $f(x_1,\dots,x_n)$ *depends* on the ith variable x_i if $f(a)\neq f(b)$ holds for some two vectors $a,b\in\{0,1\}^n$ that differ only in the ith position. The following fact is well known; see, for example, [4, Theorem 1.17].

Fact 4 (Folklore)**.** *A Boolean function $f(x_1,\dots,x_n)$ depends on the ith variable iff x_i or $\bar{x}_i$ appears in at least one prime implicant of f.*

Proof. The "only if" direction follows from the obvious fact that every Boolean function f is an OR of its prime implicants. So, if neither x_i nor $\bar{x}_i$ appears in any prime implicant of f, then f does not depend on the ith variable. To show the "if" direction, let $t=zt'$ be a prime implicant of f, where $z\in\{x_i,\bar{x}_i\}$. Since the implicant t is prime, the term t' is not an implicant of f. That is, there is a vector $a\in\{0,1\}^n$ such that $t'(a)=1$ but $f(a)=0$ and, hence, also $t(a)=0$. Let b be the vector a with its ith bit a_i replaced by $1-a_i$. Then $t(b)=1$ and, hence, also $f(b)=1$, meaning that the function f depends on the ith variable. □

We say that two Boolean functions are *independent* if they depend on disjoint sets of variables. Thus, by Fact 4, two Boolean functions are independent iff their prime implicants share no common variables (negated or not). For example, the functions $g=x\vee x\bar{y}$ and $h=\bar{y}\vee\bar{y}z$ are independent.

Definition 2 (Multilinear circuits)**.** A DeMorgan $(\vee,\wedge,\neg)$ circuit F is *multilinear* (or *semantically multilinear*) if the two Boolean functions g and h computed at the inputs to any AND gate are independent.

It is clear that every syntactically multilinear DeMorgan circuit is also (semantically) multilinear. But the converse does not need to hold: for example, the circuit $F=(x\vee xy)(y\vee yz)$ is not syntactically multilinear, but is (semantically) multilinear.

The *upward closure* of a Boolean function $f:\{0,1\}^n\to\{0,1\}$ is the monotone Boolean function

$$f^{\triangledown}(x):=\bigvee_{z\leqslant x} f(z)\,.$$

For example, the upward closure of the parity function $f=x_1\oplus x_2\oplus\cdots\oplus x_n$ is $f^{\triangledown}=x_1\vee x_2\vee\cdots\vee x_n$. Note that $f^{\triangledown}=f$ holds for *monotone* functions f. It is also easy to verify that the lowest ones of f and of $f^{\triangledown}$ are the same, that is, $A_{f^{\triangledown}}=A_f$ holds.

Our goal in this section is to prove the following general lower bounds. For a Boolean function f, let $\mathsf{B}_{\mathrm{lin}}(f)$ denote the minimum size of a multilinear DeMorgan $(\vee,\wedge,\neg)$ circuit computing f. For a *monotone* Boolean function f, let $\mathsf{B}^{+}_{\mathrm{lin}}(f)$ denote the minimum size of a monotone multilinear $(\vee,\wedge)$ circuit computing f. Recall that $A_f\subseteq f^{-1}(1)$ is the set of lowest ones of a (not necessarily monotone) Boolean function f, that is, the set of vectors a such that $f(a)=1$ but $f(b)=0$ for all $b\leqslant a$, $b\neq a$.

Theorem 3. *For every Boolean function f, we have*

$$\mathsf{B}_{\mathrm{lin}}(f)\geqslant\mathsf{B}^{+}_{\mathrm{lin}}(f^{\triangledown})\geqslant\mathsf{B}_1(f^{\triangledown})\geqslant\mathsf{Arith}(\lfloor A_f\rfloor)\,. \tag{2}$$

We will first prove the following consequence of Theorem 3 for *monotone* Boolean functions f, and then will prove the theorem itself (for arbitrary functions f). A monotone Boolean function is *homogeneous* if all its prime implicants have the same number of variables, and is *k-homogeneous* if this number is k.

Recently, Lingas [20] has shown that if a monotone Boolean function f is k-homogeneous, then $\mathsf{B}^{+}_{\mathrm{lin}}(f)\geqslant\mathsf{Arith}(A_f)/\mathcal{O}(k^2)$. Theorem 3 implies that, for such functions f, we actually have the

equality $\mathsf{B}^+_{\mathrm{lin}}(f) = \mathsf{Arith}(A_f)$, and even the equality $\mathsf{B}_{\mathrm{lin}}(f) = \mathsf{Arith}(A_f)$ (for non-monotone multilinear circuits). That is, the minimum size of a multilinear $(\vee,\wedge,\neg)$ circuit computing a Boolean function $f(x) = \bigvee_{S\in\mathcal{F}}\bigwedge_{i\in S} x_i$, where all sets $S \in \mathcal{F}$ are of the same size, *coincides* with the minimum size of a monotone arithmetic $(+,\times)$ circuit computing a polynomial similar to $P(x) = \sum_{S\in\mathcal{F}}\prod_{i\in S} x_i$.

Corollary 1. *For every monotone Boolean function f, we have $\mathsf{B}_{\mathrm{lin}}(f) = \mathsf{B}^+_{\mathrm{lin}}(f)$. If f is monotone and homogeneous, then $\mathsf{B}_{\mathrm{lin}}(f) = \mathsf{B}^+_{\mathrm{lin}}(f) = \mathsf{B}_1(f) = \mathsf{Arith}(A_f)$.*

Proof. Since f is monotone, we have $f^\triangledown = f$. So, the equality $\mathsf{B}_{\mathrm{lin}}(f) = \mathsf{B}^+_{\mathrm{lin}}(f)$ for monotone Boolean functions f follows from the lower bound $\mathsf{B}_{\mathrm{lin}}(f) \geqslant \mathsf{B}^+_{\mathrm{lin}}(f^\triangledown) = \mathsf{B}^+_{\mathrm{lin}}(f)$ given by Eq. (2), and from a trivial upper bound $\mathsf{B}_{\mathrm{lin}}(f) \leqslant \mathsf{B}^+_{\mathrm{lin}}(f)$.

Suppose now that the function f is monotone and homogeneous. By Eq. (2), it is enough to show that then $\mathsf{B}^+_{\mathrm{lin}}(f) = \mathsf{Arith}(A)$ holds for the set $A := A_f$ of the lowest ones of the function f. Since f is homogeneous, we have $\lfloor A \rfloor = A$. Hence, Eq. (2) yields $\mathsf{B}^+_{\mathrm{lin}}(f) \geqslant \mathsf{Arith}(\lfloor A \rfloor) = \mathsf{Arith}(A)$, and ir remains to show the inequality $\mathsf{B}^+_{\mathrm{lin}}(f) \leqslant \mathsf{Arith}(A)$.

For this, take a monotone constant-free arithmetic $(+,\times)$ circuit F of size $s = \mathsf{Arith}(A)$ computing some polynomial similar to $P(x) = \sum_{a\in A}\prod_{i=1}^n x_i^{a_i}$. By Fact 3, the circuit F also *produces* the set A of exponent vectors of P. Let F' be the Boolean version of the circuit F obtained by replacing each $+$-gate by an OR gate, and each $\times$-gate by an AND gate. The circuit F' produces the same set A of exponent vectors and, hence, computes our Boolean function $f(x) = \bigvee_{a\in A}\bigwedge_{i\in\mathrm{sup}(a)} x_i$. So, it remains to show that the circuit F' is multilinear.

To show this, take an arbitrary AND gate $u = v \wedge w$ in the Boolean circuit F'. We have to show that the Boolean functions f_v and f_w computed at the inputs v and w of this gate are independent. In the arithmetic circuit F, this was a multiplication gate $u = v \times w$. Let $B_v \subseteq \mathbb{N}^n$ and $B_w \subseteq \mathbb{N}^n$ be the sets of exponent vectors of the polynomials P_v and P_w produced at the gates v and w of F. Since the polynomial P produced by the entire circuit F is multilinear, the polynomials produced at its gates must be multilinear. In particular, the polynomial P_u produced at the gate $u = v \times w$ must be also multilinear. Thus, the sets B_v and B_w must consist of vectors (actually, of 0-1 vectors due to multilinearity) with *disjoint* supports, that is, $\mathrm{sup}(x) \cap \mathrm{sup}(y) = \emptyset$ must hold for all vectors $x \in B_v$ and $y \in B_w$.

In the Boolean version F' of the circuit F, the same sets B_v and B_w of exponent vectors are produced at the inputs v and w of the AND gate $u = v \wedge w$. By Fact 2, the inclusions[3] $\mathrm{Sup}(A_{f_v}) \subseteq \mathrm{Sup}(B_v)$ and $\mathrm{Sup}(A_{f_w}) \subseteq \mathrm{Sup}(B_w)$ hold. That is, for every lowest one $b \in A_{f_v}$ of the function f_v computed at the gate v there is a vector $x \in B_v$ with $\mathrm{sup}(x) = \mathrm{sup}(b)$, and for every lowest one $c \in A_{f_w}$ of f_w there is a vector $y \in B_w$ with $\mathrm{sup}(y) = \mathrm{sup}(c)$. Thus, $\mathrm{sup}(b) \cap \mathrm{sup}(c) = \emptyset$ holds for all lowest ones $b \in A_{f_v}$ and $c \in A_{f_w}$ as well. Since prime implicants of a *monotone* Boolean function have no negated literals (Fact 1), and since lowest ones of such a function are characteristic 0-1 vectors of its prime implicants, this means that prime implicants of f_v and f_w share no variables in common. Thus, by Fact 4, the functions f_v and f_w are independent, as desired. □

6.1 Proof of Theorem 3

Since lowest ones of a Boolean function f and of its upward closure $g := f^\triangledown$ are the same, we have $\lfloor A_g \rfloor = \lfloor A_f \rfloor$ and, hence, also $\mathsf{Arith}(\lfloor A_g \rfloor) = \mathsf{Arith}(\lfloor A_f \rfloor)$. By Theorem 1, $\mathsf{B}_1(g) \geqslant \mathsf{Arith}(\lfloor A_g \rfloor) = \mathsf{Arith}(\lfloor A_f \rfloor)$. This shows the last inequality in Eq. (2). To prove the first two inequalities in Eq. (2), we first establish (in Lemmas 4 and 5) the behavior of sets A_g of lowest ones as well as of upward closures $g^\triangledown$ of functions g computed at the gates of DeMorgan $(\vee,\wedge,\neg)$ circuits. In both cases, we will use the following simple fact.

[3] Recall that $\mathrm{Sup}(A) = \{\mathrm{sup}(a) : a \in A\} \subseteq 2^{[n]}$ is the family of supports $\mathrm{sup}(a) = \{i \in [n] : a_i \neq 0\}$ of vectors in A.

Fact 5. *Let $g,h:\{0,1\}^n\to\{0,1\}$ be Boolean functions, and let $b\in A_g$ and $c\in A_h$ be their lowest ones. If g and h are independent, then* $\sup(b)\cap\sup(c)=\emptyset$ *and* $g(b\vee c)=h(b\vee c)=1$.

Proof. Note that $t(b)=1$ holds for some prime implicant $t=\bigwedge_{i\in S}x_i\wedge\bigwedge_{j\in T}\bar{x}_j$ of g with $S=\sup(b)$: we have $S\subseteq\sup(b)$ since $t(b)=1$, and $\sup(b)\subseteq S$ since $g(b)=1$ and b is a *lowest* one of g. So, since g and h are independent, the disjointness $\sup(b)\cap\sup(c)=\emptyset$ follows from Fact 4. In particular, $b+c=b\vee c$ is a 0-1 vector. Since the function g does not depend on any variable x_i with $i\in\sup(c)$, we have $g(b\vee c)=g(b+c)=g(b+\vec{0})=g(b)=1$. Similarly, since function h does not depend on any variable x_i with $i\in\sup(b)$, we also have $h(b\vee c)=h(b+c)=h(\vec{0}+c)=h(c)=1$. □

Lemma 4 (Lowest ones)**.** *For any Boolean functions $g,h:\{0,1\}^n\to\{0,1\}$ we have $A_{g\vee h}\subseteq A_g\cup A_h$ and $A_{g\wedge h}\subseteq(A_g\vee A_h)^{\uparrow}$, with $A_{g\wedge h}\subseteq A_g\vee A_h$ if g and h are monotone. If g and h are independent, then $A_{g\wedge h}\subseteq A_g+A_h$.*

Proof. Let first $f=g\vee h$, and take an arbitrary lowest one $a\in A_f$ of f. Then $g(a)=1$ or $h(a)=1$, and both $g(b)=0$ and $h(b)=0$ hold for every vector $b\leqslant a$, $b\neq a$. Thus, either $a\in A_g$ or $a\in A_h$, as desired.

Now let $f=g\wedge h$, and take an arbitrary lowest one $a\in A_f$ of f. Since then $g(a)=1$ and $h(a)=1$, there are lowest ones $b\in A_g$ and $c\in A_h$ such that $b\leqslant a$ and $c\leqslant a$. Hence, $a\geqslant b\vee c$, meaning that vector a belongs to $(A_g\vee A_h)^{\uparrow}$, as desired. If g and h are monotone, then $g(b)=1$ implies $g(b\vee c)=1$, and $h(c)=1$ implies $h(b\vee c)=1$. Hence, $f(b\vee c)=1$. Since $b\vee c\leqslant a$ and a is a *lowest* one of f, we actually have an equality $a=b\vee c$, meaning that $a\in A_g\vee A_h$, as desired.

Now suppose that the (not necessarily monotone) functions g and h are independent, and take an arbitrary lowest one $a\in A_f$. As just shown, $a\geqslant b\vee c$ holds for some lowest ones $b\in A_g$ and $c\in A_h$. By Fact 5, we have $b\vee c=b+c$ and $g(b+c)=h(b+c)=1$; hence, also $f(b+c)=1$. Since $a\geqslant b+c$ and since vector a is a *lowest* one of f, this yields the equality $a=b+c$. Thus, $a\in A_g+A_h$, as desired. □

Remark 4. In general, the inclusion $A_{g\wedge h}\subseteq A_g\vee A_h$ does not need to hold. Take, for example $g=x_1\bar{x}_2\bar{x}_3\vee x_1x_2$ and $h=\bar{x}_1\bar{x}_2x_3\vee x_2x_3$; hence, $g\wedge h=x_1x_2x_3$. The only lowest one of $g\wedge h$ is $a=(1,1,1)$, the only lowest one of g is $b=(1,0,0)$, and the only lowest one of h is $b=(0,0,1)$. But $a\neq b\vee c$. □

Lemma 5 (Upward closures)**.** *For any Boolean functions g and h, we have $(g\vee h)^{\triangledown}=g^{\triangledown}\vee h^{\triangledown}$ and $(g\wedge h)^{\triangledown}\leqslant g^{\triangledown}\wedge h^{\triangledown}$. If g and h are independent, then $(g\wedge h)^{\triangledown}=g^{\triangledown}\wedge h^{\triangledown}$.*

Proof. Let first $f=g\vee h$. To show the inequality $f^{\triangledown}\leqslant g^{\triangledown}\vee h^{\triangledown}$, take any vector $x\in\{0,1\}^n$ for which $f^{\triangledown}(x)=1$ holds. Then $x\geqslant a$ for some lowest one $a\in A_f$. By Lemma 4, we have $a\in A_g$ or $a\in A_h$. Hence, either $g^{\triangledown}(x)=1$ or $h^{\triangledown}(x)=1$, as desired. To show the opposite inequality $f^{\triangledown}\geqslant g^{\triangledown}\vee h^{\triangledown}$, take any vector $x\in\{0,1\}^n$ for which $g^{\triangledown}(x)=1$ holds. Then $g(z)=1$ and, hence, also $f(z)=1$ holds for some $z\leqslant x$, meaning that $f^{\triangledown}(x)=1$, as desired. The same happens if $h^{\triangledown}(x)=1$ holds.

Now let $f=g\wedge h$. Then the inequality $f^{\triangledown}\leqslant g^{\triangledown}\wedge h^{\triangledown}$ is trivial: if $f^{\triangledown}(x)=1$, then $f(z)=1$ holds for some vector $z\leqslant x$ and, hence, both $g(z)=1$ and $h(z)=1$ hold. So, assume that the functions g and h are independent. We have to show that then also $g^{\triangledown}\wedge h^{\triangledown}\leqslant f^{\triangledown}$ holds. For this, take any vector $x\in\{0,1\}^n$ for which both $g^{\triangledown}(x)=1$ and $h^{\triangledown}(x)=1$ hold. Then $g(b)=1$ and $h(c)=1$ hold for some lowest one $b\leqslant x$ of g and for some lowest one $c\leqslant x$ of h. By Fact 5, we have $g(b\vee c)=h(b\vee c)=1$ and, hence, also $f(b\vee c)=1$. Since $b\vee c\leqslant x$, this yields $f^{\triangledown}(x)=1$, as desired. □

Remark 5. In general, the inequality $g^{\triangledown}\wedge h^{\triangledown}\leqslant(g\wedge h)^{\triangledown}$ does not need to hold. Take, for example, $f=g\wedge h$ with $g=x_1\bar{x}_2\vee x_3$ and $h=\bar{x}_1x_2\vee x_3$; hence, $f=x_3$. On the vector $a=(1,1,0)$, we have $f^{\triangledown}(a)=f(a)=0$, but $g^{\triangledown}(a)\geqslant g(1,0,0)=1$ and $h^{\triangledown}(a)\geqslant h(0,1,0)=1$. □

Now we turn to the proof of the remaining two inequalities $\mathsf{B}_{\text{lin}}(f) \geqslant \mathsf{B}^{+}_{\text{lin}}(f^{\triangledown})$ and $\mathsf{B}^{+}_{\text{lin}}(f^{\triangledown}) \geqslant \mathsf{B}_1(f^{\triangledown})$ claimed in Theorem 3. This is done in Lemmas 6 and 7.

We can view every DeMorgan $(\vee,\wedge,\neg)$ circuit $F(x)$ computing a Boolean function $f(x)$ of n variables as a *monotone* $(\vee,\wedge)$ circuit $H(x,y)$ of $2n$ variables with the property that $f(x) = H(x,\bar{x})$ holds for all $x \in \{0,1\}^n$, where $\bar{x} = (\bar{x}_1,\ldots,\bar{x}_n)$ is the complement of $x = (x_1,\ldots,x_n)$. The *monotone version* of the circuit $F(x)$ is the monotone circuit $F^{+}(x) = H(x,\vec{1})$ obtained by replacing every negated input literal $\bar{x}_i$ with constant 1. For example, the monotone version of the circuit $F = y\bar{z} \vee x(\bar{y} \vee \bar{x}y)$ is $F^{+} = y \cdot 1 \vee x(1 \vee 1 \cdot y) = x \vee y$.

Lemma 6 (Multilinear to monotone multilinear)**.** *Let F be a DeMorgan $(\vee,\wedge,\neg)$ circuit computing a Boolean function f. If F is multilinear, then the circuit F^{+} is also multilinear and computes $f^{\triangledown}$. In particular, $\mathsf{B}_{\text{lin}}(f) \geqslant \mathsf{B}^{+}_{\text{lin}}(f^{\triangledown})$ holds.*

Proof. Suppose that the circuit $F(x) = H(x,\bar{x})$ is multilinear. To verify that the circuit $F^{+}(x) = H(x,\vec{1})$ is also multilinear, take an arbitrary AND gate $u = v \wedge w$ in F. By Fact 4, the prime implicants of the Boolean functions f_v and f_w computed at gates v and w do not share common variables. The prime implicants of the functions ${f_v}^{+}$ and ${f_w}^{+}$ computed at gates v and w in the monotone version F^{+} of F are obtained from the prime implicants of f_v and f_w by replacing negated literals $\bar{x}_i$ with constant 1. So again, by Fact 4, the functions ${f_v}^{+}$ and ${f_w}^{+}$ are independent, as desired. It remains to show that the circuit F^{+} computes the upward closure $f^{\triangledown}$ of the function f computed by F.

Upward closures of input variables x_i are the variables $x_i^{\triangledown} = x_i$ themselves, while upward closures of negated input variables $\bar{x}_i$ are constant-1 functions $\bar{x}_i^{\triangledown} = 1$. Let g and h be the Boolean functions computed at the two inputs of an arbitrary gate of F. If this is an OR gate, then Lemma 5 yields the equality $(g \vee h)^{\triangledown} = g^{\triangledown} \vee h^{\triangledown}$. If this is an AND gate then, since the circuit F is multilinear, the functions g and h are independent, and Lemma 5 also yields the equality $(g \wedge h)^{\triangledown} = g^{\triangledown} \wedge h^{\triangledown}$. Thus, in the circuit $F^{+} = H(x,\vec{1})$, the upward closures $g^{\triangledown}$ of the functions g computed at the gates of F are computed. Since this also holds for the output gate of F, at which the function f was computed, the upward closure $f^{\triangledown}$ of f is computed at this gate of the circuit F^{+}, as desired. □

Lemma 7 (Monotone multilinear to read-1)**.** *Monotone multilinear Boolean circuits are read-1 circuits. In particular, $\mathsf{B}^{+}_{\text{lin}}(f) \geqslant \mathsf{B}_1(f)$ holds for every monotone Boolean function f.*

Proof. Let F be a monotone multilinear Boolean circuit computing a monotone Boolean function f. Let $B \subseteq \mathbb{N}^n$ be the set of exponent vectors produced by F. By Fact 2, the inclusion $B \subseteq (A_f)^{\uparrow}$ holds. So, to show that F is a read-1 circuit, we have only to show that also the inclusion $A_f \subseteq B$ holds, i.e., that every lowest one $a \in A_f$ of f is produced by the circuit F.

Let F_1 and F_2 be the subcircuits of F whose output gates enter the output gate of F, and let f_1 and f_2 be the monotone Boolean functions computed by these subcircuits. Let also $B_1 \subseteq \mathbb{N}^n$ be the set of exponent vectors produced by the subcircuit F_1, and $B_2 \subseteq \mathbb{N}^n$ be the set of exponent vectors produced by the subcircuit F_2. We argue by induction on the number s of gates in F. In the basis case $s = 1$, we have $F_1 = x_i$ and $F_2 = x_j$ for some $i,j \in [n]$. Hence, $B_1 = \{\vec{e}_i\} = A_{f_1}$ and $B_2 = \{\vec{e}_j\} = A_{f_2}$. So, if $F = F_1 \vee F_2 = x_i \vee x_j$ then $B = B_1 \cup B_2 = \{\vec{e}_i,\vec{e}_j\} = A_f$. If $F = F_1 \wedge F_2 = x_i x_j$, then $i \neq j$ due to the multilinearity of the circuit F. Hence, $B = B_1 + B_2 = \{\vec{e}_i + \vec{e}_j\} = A_f$.

Now suppose that the lemma holds for all monotone Boolean circuits of size at most $s-1$, and let F be a monotone Boolean circuit of size s. Since the circuit F is multilinear, both subcircuit F_1 and F_2 are also multilinear. Since each of F_1 and F_2 has at most $s-1$ gates, the lemma holds for both these subcircuits. Thus, both inclusions $A_{f_1} \subseteq B_1$ and $A_{f_2} \subseteq B_2$ hold.

If $F = F_1 \vee F_2$, then $B = B_1 \cup B_2$ and Lemma 4 gives the inclusion $A_f \subseteq A_{f_1} \cup A_{f_2}$. So, the desired inclusion $A_f \subseteq B$ follows from the induction hypothesis. If $F = F_1 \wedge F_2$, then $B = B_1 + B_2$ (Minkowski

sum). Since the circuit F is multilinear, the functions f_1 and f_2 are independent, and Lemma 4 yields $A_f \subseteq A_{f_1} + A_{f_2}$. So, the desired inclusion $A_f \subseteq B$ follows again from the induction hypothesis. □

6.2 Multilinear Circuits Impede Zero Terms

Lemma 6 rises a natural question: if $F = F(x,\bar{x})$ is a DeMorgan $(\vee,\wedge,\neg)$ circuit computing a Boolean function f, when does its monotone version $F^+ = F(x,\vec{1})$ computes f^∇? It can be easily shown that such are exactly DeMorgan circuits F that "impede zero terms" in the following sense.

Every DeMorgan $(\vee,\wedge,\neg)$ circuit F *produces* (purely syntactically) a unique set $T(F)$ of terms in a natural way: $T(F) = \{z\}$ if $F = z$ is an input literal $z \in \{x_i,\bar{x}_i\}$, $T(F_1 \vee F_2) = T(F_1) \cup T(F_2)$, and $T(F_1 \wedge F_2) = \{t_1 \wedge t_2 \colon t_1 \in T(F_1), t_2 \in T(F_2)\}$. During the production of terms, we can use the "shortening" rule $z \wedge z = z$, but do *not* use the "annihilation" rule $z \wedge \bar{z} = 0$. So, $T(F)$ can contain zero terms, that is, terms with a variable x_i and its negation $\bar{x}_i$. The *positive factor* t^+ of a term t is obtained by replacing every its negated literal $\bar{x}_i$ with constant 1.

Let us say that a DeMorgan $(\vee,\wedge,\neg)$ circuit F computing a Boolean function f *impedes zero terms* if positive factors of zero terms produced by F (if there are any) are implicants of f^∇, that is, if $t^+ \leqslant f^\nabla$ holds for every zero term $t \in T(F)$. Note that such a circuit does *not* forbid production of zero terms as such, but rather "impedes" produced zero terms to unfold the full power of cancellations $x \wedge \bar{x} = 0$.

Fact 6. *Let F be a DeMorgan $(\vee,\wedge,\neg)$ circuit computing a Boolean function f with $f(\vec{0}) = 0$. The circuit F^+ computes f^∇ if and only if F impedes zero terms.*

Proof. Since F computes f, we have $f = \bigvee_{t\in T} t$, where $T = T(F)$ is the set of all terms produced by the circuit F. Since $f(\vec{0}) = 0$, none of the terms $t \in T$ consist of solely negated variables. For every term t, we have $t^\nabla = 0$ (the constant 0 function) if t is a zero term, and $t^\nabla = t^+$ if t is a nonzero term. So, if $T' \subseteq T$ is the set of all nonzero terms in T, then (where the second equality follows from Lemma 5):

$$f^\nabla = \left(\bigvee_{t\in T} t\right)^\nabla = \bigvee_{t\in T} t^\nabla = \bigvee_{t\in T'} t^+ \leqslant \bigvee_{t\in T} t^+ = F^+$$

with the equality iff $t \leqslant f^\nabla$ holds for all terms $t \in T \setminus T'$. □

DeMorgan $(\vee,\wedge,\neg)$ circuits that do not produce zero terms at all were considered by several authors, starting with Kuznetsov [18] (already in 1981, under the name "circuits without null-chains"), where he proved a lower bound $|f^{-1}(1)|^{d/(n+d)}$ on the size of such circuits for any Boolean function f such that the Hamming distance between any two vectors $a \neq b \in f^{-1}(1)$ is at least d (this gives a lower bound $2^{\Omega(\sqrt{n})}$ for the characteristic function of Reed–Muller codes), and a surprisingly high lower bound $2^{n/3}$ on the size of such circuits computing another explicit n-variate Boolean function. Sengupta and Venkateswaran [33] also considered DeMorgan $(\vee,\wedge,\neg)$ circuits that do not produce zero terms (under the name of "non-cancellative circuits"). They showed that for every such circuit F computing a Boolean function f, the monotone version F^+ of F computes f^∇. Since non-cancellative circuits produce no zero terms, this also follows from Fact 6.

Multilinear DeMorgan $(\vee,\wedge,\neg)$ circuits already *can* produce zero terms. For example, the DeMorgan $(\vee,\wedge,\neg)$ circuit $F = (x \vee xy)(\bar{y} \vee \bar{y}z)$ computing $f = x\bar{y}$ is multilinear but produces zero terms $xy\bar{y}$ and $xy\bar{y}z$. Still, together with Lemma 6, Fact 6 implies that multilinear DeMorgan circuits impede the produced zero terms as well.

7 An easy lower bound on $\mathsf{Arith}(A)$

There are relatively many strong lower bounds on the monotone arithmetic $(+,\times)$ circuit complexity $\mathsf{Arith}(A)$ of explicit polynomials $P(x)=\sum_{a\in A}\prod_{i=1}^{n}x_i^{a_i}$, for example [8, 9, 12, 15, 26, 31, 35]. The obtained bounds are quite high, some even of the form $\mathsf{Arith}(A)\geqslant 2^{n/2-o(n)}$ (see [9, Theorem 3] or [15, Appendix E]). By Theorems 1 to 3, these bounds extend to Boolean multilinear, to monotone read-1, and to tropical circuits.

To demonstrate that the *absence* of multiplicative idempotence $x\wedge x=x$ and absorption $x\vee xy=x$ in arithmetic circuits can make the task of proving lower bounds much easier than in the case of monotone Boolean circuits, let us recall a simple "decomposition trick" for arithmetic circuits observed already by Hyafil [11, Theorem 1] and Valiant [35, Lemma 3].

A set $A\subseteq\mathbb{N}^n$ of vectors is *m-homogeneous* if $x_1+\cdots+x_n=m$ holds for all vectors $x\in X$. A *rectangle* is a Minkowski sum $X+Y=\{x+y\colon x\in X, y\in Y\}$ of sets $X,Y\subseteq\mathbb{N}^n$ of vectors. Such a rectangle is *r-homogeneous* if the set X is r-homogeneous. Let $h_r(A)$ denote the maximum possible number $|X+Y|$ of vectors in an r-homogeneous rectangle $X+Y\subseteq A$. Note that we only have to consider rectangles lying entirely within the set A—this is in stark contrast to Boolean and even to tropical circuits (where also "redundant things" can be produced along the way).

Lemma 8 ([11, 35])**.** *Let $m\geqslant 3$, and let $A\subseteq\{0,1\}^n$ be m-homogeneous. Then $\mathsf{Arith}(A)\geqslant |A|/h_r(A)$ holds for some r between $m/3$ and $2m/3$.*

Proof. By the definition, $\mathsf{Arith}(A)$ is the minimum size s of a monotone arithmetic circuit F computing a polynomial $f(x)=\sum_{a\in A}c_a\prod_{i=1}^{n}x_i^{a_i}$ whose set of exponent vectors is the set A. By Fact 3, the polynomial f is also *produced* by F. Since the polynomial f is multilinear, the circuit is also multilinear in that the polynomial produced at each its gate is multilinear. We will prove that the polynomial f can be written as a sum $f=g_1h_1+\cdots+g_th_t$ of $t\leqslant s$ polynomials, where each polynomial g_i is homogeneous of degree lying between $m/3$ and $2m/3$. If X_i and Y_i are the sets of exponent vectors of polynomials g_i and h_i, then the rectangle X_i+Y_i is the set of exponent vectors of the product polynomial g_ih_i. Thus, the set A of exponent vectors of the polynomial f can be written as a union $A=(X_1+Y_1)\cup\cdots\cup(X_t+Y_t)$ of $t\leqslant s$ rectangles, each being r-balanced for some $m/3\leqslant r\leqslant 2m/3$, and the lower bound $s\geqslant|A|/h_r(A)$ follows.

The proof of the decomposition $f=g_1h_1+\cdots+g_th_t$ is based on the observation that, due to $\deg(gh)=\deg(g)+\deg(h)$, there must be a gate u in a given circuit F of size s for f, at which a polynomial g_1 of degree $m/3<\deg(g_1)\leqslant 2m/3$ is computed: start at the output gate and go backwards by always choosing that input gate, the polynomial computed at which has larger degree. If we replace the gate u by a new input variable y then, due to the multilinearity of the circuit F, the variable y appears in each monomial of the produced polynomial $f'(x_1,\ldots,x_n,y)$ with degree 0 or 1. Hence, f' is of the form $f'=y\cdot h_1+f_1$, where the polynomials h_1 and f_1 do not depend on y, and f_1 is the polynomial produced by the circuit F' obtained after substitution $y=0$. Hence, we have a decomposition $f=g_1\cdot h_1+f_1$. If $\deg(f_1)<m$, then $f_1=0$ (since the entire polynomial f is homogeneous of larger degree m), and we are done. Otherwise, we have $\deg(f_1)=m$, and can apply the same argument to the circuit F' of size $s-1$ producing the polynomial f_1. □

Example 2 (Perfect matchings). The *perfect matching* function is a monotone Boolean function $f=\mathsf{Match}_n$ which accepts a subgraph of $K_{n,n}$ iff it contains a perfect matching. The set $A=A_f$ of lowest ones of this function consists of $|A|=n!$ characteristic 0-1 vectors of all perfect matchings (viewed as sets of their edges). Since the set A is homogeneous, Theorems 1 to 3 yield $\mathsf{B}_{\mathrm{lin}}(f)\geqslant\mathsf{Min}(A)=\mathsf{B}_1(f)\geqslant\mathsf{Arith}(A)$. Moreover, using Lemma 8, one can show a lower bound $\mathsf{Arith}(A)\geqslant\binom{n}{n/3}=2^{\Omega(n)}$.

To show this, take any rectangle $X+Y$ such that $X+Y \subseteq A$ and the set X is r-homogeneous. Fix an arbitrary vector $x \in X$; hence, x is the characteristic 0-1 vector of a matching with r edges. Since $x+Y \subseteq A$, all vectors in $x+Y = \{x+y\colon y \in Y\}$ are characteristic 0-1 vectors of $|x+Y| = |Y|$ perfect matchings. Since all these perfect matchings contain all r edges of the (fixed) matching corresponding to x, we have $|Y| = |x+Y| \leqslant (n-r)!$. Similarly, every vector $y \in Y$ corresponds to a matching with $n-r$ edges, and we have $|X| = |X+y| \leqslant r!$. Thus, $|X+Y| = |X| \cdot |Y| \leqslant (n-r)!r! = n!\binom{n}{r}^{-1}$. Since $\binom{n}{r} \geqslant \binom{n}{n/3}$ for every $n/3 \leqslant r \leqslant 2n/3$, this yields $h_r(A) \leqslant n!\binom{n}{n/3}^{-1}$ for every such r, and the claimed lower bound $\mathsf{Arith}(A) \geqslant |A|/h_r(A) = \binom{n}{n/3}$ follows. □

Remark 6. Ponnuswami and Venkateswaran [23] used direct arguments to prove a lower bound $\mathsf{B}^+_{\mathrm{lin}}(f) = \Omega(2^{.459n})$ for $f = \mathrm{Match}_n$. On the other hand, using arguments tighter than in Example 2, Jerrum and Snir [12] have proved a lower bound $\mathsf{Arith}(A_f) \geqslant n(2^{n-1}-1)$ for $f = \mathrm{Match}_n$. Together with Theorem 3, this yields lower bounds $\mathsf{B}^+_{\mathrm{lin}}(f) = \mathsf{B}_{\mathrm{lin}}(f) \geqslant \mathsf{B}_1(f) \geqslant n(2^{n-1}-1)$ for this particular function. □

8 The Read-1/Read-2 Gap Can be Exponential

Theorems 1 to 3 show that read-1 $(\vee,\wedge)$ circuits have the *same* power as tropical $(\min,+)$ and as monotone arithmetic $(+,\times)$ circuits computing homogeneous multilinear polynomials, and are *not weaker* than (non-monotone) multilinear $(\vee,\wedge,\neg)$ circuits. Let us now show that already read-2 $(\vee,\wedge)$ circuits can be much smaller than read-1 $(\vee,\wedge)$ circuits and multilinear $(\vee,\wedge,\neg)$ circuits. For this, consider the following monotone Boolean function whose inputs are Boolean $n \times n$ matrices $x = (x_{i,j})$:

$$\mathrm{Isol}_n(x) = 1 \text{ iff every row and every column of } x \text{ has at least one 1.}$$

That is, $\mathrm{Isol}_n(x) = 0$ iff the matrix x has an "isolated" (all-0) row or column.

Lemma 9. *For* $f = \mathrm{Isol}_n$, *we have* $\mathsf{B}_1(f) = 2^{\Omega(n)}$ *but* $\mathsf{B}_2(f) \leqslant 2n^2$.

Proof. The set $A := f^{-1}(1)$ consists of all Boolean $n \times n$ matrices $a = (a_{i,j})$ with at least one 1 in each line (row or column). None of such matrices can have fewer than n 1s, because then it would have an all-0 row or an all-0 column. So, the smallest number of 1s in a matrix $a \in A$ is n, and the matrices in A with this number of 1s are permutation matrices (with exactly one 1 in each row and in each column). This means that the lower envelope $\lfloor A \rfloor$ of f is the set A_g of lowest ones of the perfect matching function $g = \mathrm{Match}_n$, and we already know that $\mathsf{Arith}(A_g) = 2^{\Omega(n)}$ holds (see Example 2). Together with Theorem 1, this yields $\mathsf{B}_1(f) \geqslant \mathsf{Arith}(\lfloor A \rfloor) = \mathsf{Arith}(A_g) = 2^{\Omega(n)}$.

To show $\mathsf{B}_2(f) \leqslant 2n^2$, observe that f can be computed by a trivial monotone Boolean circuit

$$F(x) = \bigwedge_{i=1}^{n} \Big(\bigvee_{j=1}^{n} x_{i,j} \Big) \bigwedge_{j=1}^{n} \Big(\bigvee_{i=1}^{n} x_{i,j} \Big)$$

of size at most $2n^2$. So, it remains to verify that F is a read-2 circuit. Let $B \subseteq \mathbb{N}^{n \times n}$ be the set of exponent vectors (or, better to say, of "exponent matrices") of the polynomial

$$P(x) = \prod_{i=1}^{n} \Big(\sum_{j=1}^{n} x_{i,j} \Big) \prod_{j=1}^{n} \Big(\sum_{i=1}^{n} x_{i,j} \Big) = \sum_{g,h:[n]\to[n]} x_{1,g(1)} x_{2,g(2)} \cdots x_{n,g(n)} x_{h(1),1} x_{h(2),2} \cdots x_{h(n),n}$$

produced by the arithmetic $(+,\times)$ version of the circuit F. Note that each matrix $b \in B$ is the sum $b = x+y$ of a matrix x (corresponding to the first product) with exactly one 1 is each row, and a matrix

y (corresponding to the second product) with exactly one 1 in each column, while each matrix $a \in A_f$ is the entry-wise OR $a = x \vee y$ of two such matrices. Since $\sup(x+y) = \sup(x \vee y)$, and since none of the matrices $b \in B$ has any entry larger than 2, the circuit F is a read-2 circuit. □

9 Concluding Remarks

It is clear that $\mathsf{B}_k(f) \geqslant \mathsf{B}(f)$ holds for any monotone Boolean function f and for *any* $k \geqslant 1$, where $\mathsf{B}(f)$ is the minimum size of a monotone $(\vee,\wedge)$ circuit computing f. Super-polynomial lower bounds on $\mathsf{B}(f)$ for explicit functions f are known since the celebrated Method of Approximations was invented by Razborov [27, 28, 29]. Although strong lower bounds on $\mathsf{B}(f)$ were obtained for some explicit functions, this method (and its latter symmetric versions) can be only applied to Boolean functions with very special combinatorial properties: *both* minterms and maxterms must be highly "dispersed" (not too many of them can share a given number of variables in common). For example, already the application in [28] of the Method of Approximations to prove the lower bound $\mathsf{B}(f) = n^{\Omega(\log n)}$ for the perfect matching function $f = \mathrm{Match}_n$ is rather nontrivial (since the maxterms of f are dispersed not highly enough), going deeply into the structure of maxterms of this particular function (see, e.g., [13, Chapter 9] for more information).

On the other hand, the *absence* of multiplicative idempotence $x \wedge x = x$ and absorption $x \vee xy = x$ in arithmetic circuits was crucial in the proofs of all lower bounds for such circuits, including those in [8, 9, 12, 15, 26, 31, 35, 38]. The absence of multiplicative idempotence and absorption was also crucial in the elementary proof of the lower bound $\mathsf{Arith}(A_f) = 2^{\Omega(n)}$ for $f = \mathrm{Match}_n$ in Section 7.

Problem 1. Can a super-polynomial lower bound on $\mathsf{B}_2(f)$ be proved *without* using the Method of Approximations? In particular, can this be done for $f = \mathrm{Match}_n$?

Note that monotone Boolean read-2 circuits constitute the *first* model of computation—after tropical and monotone arithmetic circuits—which *can* use multiplicative idempotence $x \wedge x = x$ and absorption $x \vee xy = x$. It is worth to note that without absorption, idempotence alone *cannot* unfold its full power. Namely, say that a monotone Boolean $(\vee,\wedge)$ circuit F computing a given Boolean function f is *tight* if for every monomial $\prod_{i\in I} x_i^{d_i}$ of the polynomial produced by the arithmetic $(+,\times)$ version of F, $\bigwedge_{i\in I} x_i$ is a prime implicant of f. In other words, a circuit is tight if it produces no "redundant" terms, that is, implicants of f that are not *prime* implicants of f. Note that there are no restrictions on the degrees d_i of variables in terms produced by tight circuits, only on the form of produced terms. As shown in [14, Theorem 2], a lower bound $2^{\Omega(n)}$ for tight circuits computing the perfect matching function Match_n can be proved using a similar (fairly simple) argument similar to that used in Section 7.

But this argument fails if also absorption $x \vee xy = x$ (not only idempotence $x \wedge x = x$) is allowed. In the case of read-1 circuits, we were able (in Theorem 1) to eliminate the influence of absorption by considering lower envelopes. But already in read-2 circuits, absorbtion can (at least potentially) show its power. So, a solution of Problem 1 could probably shed more light on where the power of multiplicative idempotence $x \wedge x = x$ in combination with absorption $x \vee xy = x$ comes from. Of course, the ultimate goal remains to understand the power of cancellations $x - x = 0$ in arithmetic $(+,\times,-)$ circuits and, even more ultimately, to understand the power of cancellations $x \wedge \bar{x} = 0$ in Boolean $(\vee,\wedge,\neg)$ circuits.

References

[1] N. Alon and M. Tarsi, "Colorings and orientations of graphs," *Combinatorica*, vol. 12, pp. 125–134, 1992.

[2] R. Bellman, "On a routing problem," *Quarterly of Appl. Math.*, vol. 16, pp. 87–90, 1958.
[3] ——, "Dynamic programming treatment of the Travelling Salesman problem," *J. ACM*, vol. 9, no. 1, pp. 61–63, 1962.
[4] Y. Crama and P. L. Hammer, Eds., *Boolean Functions: Theory, Algorithms, and Applications*, ser. Encyclopedia of Mathematics and Its Applications. Cambridge University Pess, 2011, vol. 142.
[5] S. Dreyfus and R. Wagner, "The Steiner problem in graphs," *Networks*, vol. 1, no. 3, pp. 195–207, 1971.
[6] R. W. Floyd, "Algorithm 97, shortest path," *Comm. ACM*, vol. 5, p. 345, 1962.
[7] L. R. Ford, "Network flow theory," Rand Corp., Santa Monica, Calif., Tech. Rep. P-923, 1956.
[8] S. B. Gashkov, "On one method of obtaining lower bounds on the monotone complexity of polynomials," *Vestnik MGU, Series 1 Mathematics, Mechanics*, vol. 5, pp. 7–13, 1987.
[9] S. B. Gashkov and I. S. Sergeev, "A method for deriving lower bounds for the complexity of monotone arithmetic circuits computing real polynomials," *Sbornik: Mathematics*, vol. 203, no. 10, pp. 1411–1147, 2012.
[10] M. Held and R. M. Karp, "A dynamic programming approach to sequencing problems," *SIAM J. on Appl. Math.*, vol. 10, pp. 196–210, 1962.
[11] L. Hyafil, "On the parallel evaluation of multivariate polynomials," *SIAM J. Comput.*, vol. 8, no. 2, pp. 120–123, 1979.
[12] M. Jerrum and M. Snir, "Some exact complexity results for straight-line computations over semirings," *J. ACM*, vol. 29, no. 3, pp. 874–897, 1982.
[13] S. Jukna, *Boolean Function Complexity: Advances and Frontiers*. Springer-Verlag, 2012.
[14] ——, "Lower bounds for monotone counting circuits," *Discrete Appl. Math.*, vol. 213, no. 139–152, 2016.
[15] ——, "Tropical complexity, Sidon sets and dynamic programming," *SIAM J. Discrete Math.*, vol. 30, no. 4, pp. 2064–2085, 2016.
[16] S. Jukna and H. Seiwert, "Approximation limitations of pure dynamic programming," *SIAM J. on Comput.*, vol. 49, no. 1, pp. 170–207, 2020.
[17] M. P. Krieger, "On the incompressibility of monotone DNFs," *Theory Comput. Syst.*, vol. 41, no. 2, pp. 211–231, 2007.
[18] S. E. Kuznetzov, "Circuits composed of functional elements without zero paths in the basis $\{\&, \vee, -\}$," *Izv. Vyssh. Uchebn. Zaved. Mat.*, vol. 228, no. 5, pp. 56–63, 1981, in Russian.
[19] A. Levin, "Algorithm for the shortest connection of a group of graph vertices," *Sov. Math. Dokl.*, vol. 12, pp. 1477–1481, 1971.
[20] A. Lingas, "A note on lower bounds for monotone multilinear Boolean circuits," ECCC Report Nr. 85, Tech. Rep., 2022.
[21] E. F. Moore, "The shortest path through a maze," in *Proc. Internat. Sympos. Switching Theory*, vol. II, 1957, pp. 285–292.
[22] N. Nisan and A. Wigderson, "Lower bounds on arithmetic circuits via partial derivatives," *Comput. Complexity*, vol. 6, no. 3, pp. 217–234, 1997.
[23] A. K. Ponnuswami and H. Venkateswaran, "Monotone multilinear boolean circuits for bipartite perfect matching require exponential size," in *Proc. of 24th Int. Conf. on Foundations of Software Technology and Theoret. Comput. Sci. FSTTCS'04*, ser. Lect. Notes in Comput. Sci., vol. 3328. Springer, 2004, pp. 460–468.
[24] R. Raz, "Separation of multilinear circuit and formula size," *Theory Comput.*, vol. 2, no. 6, pp. 121–135, 2006.
[25] ——, "Multi-linear formulas for Permanent and Determinant are of super-polynomial size," *J. of the ACM*, vol. 56, no. 2, pp. 1–17, 2009.
[26] R. Raz and A. Yehudayoff, "Multilinear formulas, maximal-partition discrepancy and mixed-sources extractors," *J. Comput. Syst. Sci.*, vol. 77, no. 1, pp. 167–190, 2011.
[27] A. A. Razborov, "Lower bounds for the monotone complexity of some boolean functions," *Soviet Math. Dokl.*, vol. 31, pp. 354–357, 1985.
[28] ——, "Lower bounds on monotone complexity of the logical permanent," *Math. Notes of the Acad. of Sci. of the USSR*, vol. 37, no. 6, pp. 485–493, 1985.
[29] ——, "On the method of approximations," in *Proc. of 21st Ann. ACM Symp. on Theory of Computing*. ACM, 1989, pp. 167–176.
[30] B. Roy, "Transitivité at connexité," *C. R. Acad. Sci. Paris*, vol. 249, pp. 216–218, 1959, in French.
[31] C. P. Schnorr, "A lower bound on the number of additions in monotone computations," *Theor. Comput. Sci.*, vol. 2, no. 3, pp. 305–315, 1976.
[32] R. Sengupta and H. Venkateswaran, "Multilinearity can be exponentially restrictive (preliminary version)," Georgia Institute of Technology. College of Computing, Tech. Rep. GIT-CC-94-40, 1994.
[33] ——, "Non-cancellative boolean circuits: a generalization of monotone boolean circuits," *Theor. Comput. Sci.*, vol. 237, pp. 197–212, 2000.
[34] A. Shpilka and A. Yehudayoff, "Arithmetic circuits: A survey of recent results and open questions," *Foundations and Trends in Theoretical Computer Science*, vol. 5, no. 3-4, pp. 207–388, 2010.
[35] L. G. Valiant, "Negation can be exponentially powerful," *Theor. Comput. Sci.*, vol. 12, pp. 303–314, 1980.

[36] S. Warshall, "A theorem on boolean matrices," *J. ACM*, vol. 9, pp. 11–12, 1962.
[37] I. Wegener, *The complexity of Boolean functions*. Wiley-Teubner, 1987.
[38] A. Yehudayoff, "Separating monotone VP and VNP," in *Proc. of 51st Ann. ACM SIGACT Symp. on Theory of Computing, STOC*. ACM, 2019, pp. 425–429.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il