Notes on Boolean Read-k and Multilinear Circuits

Stasys Jukna

[a] *Faculty of Mathematics and Computer Science, Vilnius University, Vilnius, 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 at least one 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 already read-1 $(\vee, \wedge)$ circuits are not weaker than monotone arithmetic constant-free $(+, \times)$ circuits computing multilinear polynomials, are not weaker than non-monotone multilinear $(\vee, \wedge, \neg)$ circuits computing monotone Boolean functions, and have the same power as tropical $(\min, +)$ circuits solving $0/1$ minimization problems. Finally, we show that read-2 $(\vee, \wedge)$ circuits can be exponentially smaller than read-1 $(\vee, \wedge)$ circuits.

Keywords: Arithmetic circuits, multilinear circuits, tropical circuits, lower bounds
2000 MSC: 68Q17, 94C11

1. Introduction

Proving lower bounds on the size of arithmetic $(+, \times, -)$ circuits as well as on the size 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 known lower bounds for arithmetic circuits computing explicit multilinear n-variate polynomials with 0-1 coefficients remain barely super-linear bounds $\Omega(n \log n)$ proved by Baur and Strassen [3] already in 1983. For Boolean circuits, known lower bounds are even not super-linear. The first lower bound $2n$ was proved by Schnorr [37] in 1974, and improved to $3n$ by Blum [4] in 1984; the best known lower bound $4.5n - o(n)$ was proved by Lachish and Raz [23] in 2001, and improved to $5n - o(n)$ by Iwama and Morizumi [13]. In both circuit models (arithmetic and Boolean), super-polynomial lower bounds are known only for restricted circuits such as bounded-depth and monotone circuits. The books [5, 15, 48] and recent surveys [6, 42] provide wide coverage of Boolean and arithmetic circuits.

This lack of proofs of strong lower bounds for unrestricted arithmetic and Boolean circuits happens mainly because such circuits can use cancellations $x - x = 0$ in the arithmetic, and can use cancellations $x \wedge \overline{x} = 0$ in the Boolean case. Understanding the role of cancellations in arithmetic and Boolean circuits remains the ultimate goal of circuit complexity.

Monotone arithmetic $(+, \times)$ circuits cannot use cancellations $x - x = 0$, while monotone Boolean $(\vee, \wedge)$ circuits cannot use cancellations $x \wedge \overline{x} = 0$. Still, the task of proving lower bounds even for monotone *Boolean* circuits turned out to be much more difficult than that for monotone *arithmetic* circuits. Although super-polynomial lower bounds on the size of monotone arithmetic $(+, \times)$ circuits

Email address: `stjukna@gmail.com` (Stasys Jukna)
URL: `https://web.vu.lt/mif/s.jukna/` (Stasys Jukna)

were known starting with the notable paper by Schnorr [38] of 1976, it took a decade until such lower bounds for monotone Boolean $(\vee, \wedge)$ circuits were proved by Razborov [34, 35] in 1985; until then, the best known lower bound on the size of monotone Boolean circuits was only $4n$ proved by Tiekenheinrich [45] in 1984. This happens because Boolean circuits can use idempotence laws $x \vee x = x$ and $x \wedge x = x$ as well as the absorption law $x \vee xy = x$, while arithmetic circuits cannot use any of these laws. The current paper attempts to identify a possible source for this discrepancy: the presence of multiplicative idempotence and absorption in Boolean circuits.

It turned out that the absence of *additive* idempotence $x \vee x = x$ in arithmetic circuits (where $x + x \neq x$) is not a crucial issue: most lower bounds (albeit not all, [49] being a nice exception) on the monotone arithmetic $(+, \times)$ circuit complexity, including [8, 9, 14, 17, 33, 38, 41, 46, 47] were proved by only using the *structure* of monomials and fully ignoring actual values of their (nonzero) coefficients.

But the absence of *multiplicative* idempotence $x \wedge x = x$ and absorption $x \vee xy = x$ in the arithmetic world turned out to be crucial even in the case of monotone circuits. The goal of this article is to show that already a *very restricted* use of multiplicative idempotence, in combination with absorption, makes a big difference between Boolean and arithmetic circuits.

To fine grain the "degree" of multiplicative idempotence, and by analogy with read-k branching programs, we introduce (in Section 4) the so-called "read-k" $(\vee, \wedge)$ circuits. Our goal is to show that already read-1 $(\vee, \wedge)$ circuits capture the power of three different types of circuits: monotone *arithmetic* $(+, \times)$ circuits, Boolean *multilinear* DeMorgan $(\vee, \wedge, \neg)$ circuits, as well as *tropical*[1] $(\min, +)$ circuits.

The latter model of (tropical) circuits is motivated by dynamic programming (DP). Namely, 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, the Roy–Floyd–Warshall all-pairs shortest paths algorithm, the Bellman–Held–Karp traveling salesman algorithm, the Dreyfus–Levin–Wagner Steiner tree algorithm, and many others. On the other hand, pure DP algorithms are just *special* (recursively constructed) tropical $(\min, +)$ circuits. Thus, any lower bound on the size of $(\min, +)$ circuits is also a lower bound on the minimum number of operations that any pure DP algorithm solving a given minimization problem must perform, be the designer of an algorithm even omnipotent.

First lower bounds on the size of tropical circuits were proved already decades ago, including Kerr [19], Jerrum and Snir [14], as well as recently, including Grigoriev and Koshevoy [10], Grigoriev and Podolskii [11], Mahajan, Nimbhorkar and Tawari [25, 26], Jukna and Seiwert [17, 18]. In fact, as shown by Jerrum and Snir [14], if an arithmetic polynomial P is multilinear and homogeneous, then the $(\min, +)$ circuit complexity of the corresponding minimization problem is at least the monotone arithmetic $(+, \times)$ circuit complexity of the polynomial P. Thus, many other lower bounds for tropical $(\min, +)$ circuits follow from the aforementioned earlier lower bounds on the monotone arithmetic $(+, \times)$ circuit complexity of the corresponding polynomials, including bounds shown many years ago by Schnorr [38], Shamir and Snir [41], Valiant [47], Tiwari and Tompa [46] and other authors.

2. Results

The model of read-k circuits is quite natural and is by analogy with the intensively investigated model of *read-k branching programs*. Intuitively, "read-k" means that one cannot "benefit" from

[1] The adjective "tropical" was coined by French mathematicians in honor of Imre Simon who lived in Sao Paulo (south tropic). Tropical algebra and tropical geometry are now intensively studied topics in mathematics.

(multiplicative) idempotence for more than k times.

Let F be a monotone Boolean $(\vee, \wedge)$ circuit. Throughout the article, we assume that constants 0 and 1 are not are not used as inputs in Boolean circuits: such inputs are not necessary when computing non-trivial (non-constant) Boolean functions. An *arithmetic version* of F is a monotone arithmetic $(+, \times)$ circuit obtained from F by replacing every OR gate with an addition gate, and every AND gate with a multiplication gate. The obtained arithmetic circuit produces (purely syntactically) some polynomial P_F, which we call the *formal polynomial* of the Boolean circuit F. A *shadow monomial* of a Boolean term $t = \bigwedge_{i\in S} x_i$ is a monomial $p = \prod_{i\in S} x_i^{d_i}$ with the same set of variables as t and all degrees $d_i \geqslant 1$. For example, x^2y^3 is a shadow monomial of xy. It is easy to show (see Lemma 1) that the circuit F computes a monotone Boolean function $f : \{0,1\}^n \to \{0,1\}$ if and only if the formal polynomial P_F of the circuit F has the following two properties:

(i) every monomial of P_F contains all variables of at least one prime implicant of f;
(ii) every prime implicant of f has at least one shadow monomial in P_F.

Intuitively, (i) reflects the absorption property $x \vee xy = x$ (longer monomials "do not matter"), while (ii) reflects the multiplicative idempotence $x \wedge x = x$ (large degree "does not matter"). In *read-k* circuits we strengthen the property (ii) and require that every prime implicant of f has at least one shadow monomial in P_F in which each variable appears with degree $\leqslant k$. There are no restrictions on the degrees of other monomials of P_F.

Example 1. The Boolean $(\vee, \wedge)$ circuit $F = (x \vee y)(x \vee z)(y \vee z)$ computes the Boolean function $f(x, y, z) = 1$ iff $x + y + z \geqslant 2$ (the majority function of three variables). The arithmetic $(+, \times)$ version of F is the circuit $F' = (x + y)(x + z)(y + z)$, and the polynomial produced by it is $P_F = x^2y + xy^2 + x^2z + y^2z + xz^2 + yz^2 + 2xyz$. The (Boolean) circuit F is a read-2 but not a read-1 circuit, because, for example, both shadow monomials x^2y and xy^2 of the prime implicant xy of f in the polynomial P_F have a variable of degree > 1. Note, however, that the Boolean circuit $H = xy \vee xz \vee yz$ (the dual of the circuit F) also computes f but already is a read-1 circuit: the corresponding to this circuit polynomial is $P_H = xy + xz + yz$. □

Our main results are the following. A *DeMorgan circuits* is a Boolean $(\vee, \wedge, \neg)$ circuit with negations only applied to input variables. That is, such a circuit is a monotone $(\vee, \wedge)$ circuit whose inputs are variables $x_1, \ldots, x_n$ and their negations $\overline{x}_1, \ldots, \overline{x}_n$. A DeMorgan $(\vee, \wedge, \neg)$ circuit is *multilinear* if the two Boolean functions g and h computed at the inputs to any AND gate depend on disjoint sets of variables. For example, the functions $g = x \vee xy$ and $h = \overline{y} \vee x\overline{y}$ depend on disjoint sets of variables: g depends only on x, while h depends only on y. Every family $\mathcal{F} \subseteq 2^{[n]}$ of subsets of $[n] := \{1, 2, \ldots, n\}$ defines a multilinear n-variate polynomial $P_{\mathcal{F}}(x) := \sum_{S\in\mathcal{F}} \prod_{i\in S} x_i$. A polynomial Q is *similar* to the polynomial $P_{\mathcal{F}}$ if it is of the form $Q(x) = \sum_{S\in\mathcal{F}} c_S \prod_{i\in S} x_i$ for some integer coefficients $c_S \geqslant 1$; in particular, the polynomial $P_{\mathcal{F}}$ is similar to itself (then all $c_S = 1$). An arithmetic $(+, \times)$ circuit is *constant-free* if it has no constants among the inputs. We prove the following, where $\mathcal{F} \subseteq 2^{[n]}$ is an arbitrary antichain (no two sets of $\mathcal{F}$ are comparable under the inclusion).

1. Read-1 $(\vee, \wedge)$ circuits are *not weaker* than monotone *arithmetic* $(+, \times)$ circuits computing multilinear polynomials in the following sense: if a monotone arithmetic constant-free $(+, \times)$ circuit computes a polynomial similar to $P_{\mathcal{F}}$, then a read-1 $(\vee, \wedge)$ circuit of the same size computes the Boolean version $f(x) = \bigvee_{S\in\mathcal{F}} \bigwedge_{i\in S} x_i$ of $P_{\mathcal{F}}$. If the polynomial $P_{\mathcal{F}}$ is homogeneous (all sets of $\mathcal{F}$ have the same number of elements), then the minimum size of a monotone arithmetic constant-free $(+, \times)$ circuit computing a polynomial similar to $P_{\mathcal{F}}$ even *coincides* with the minimum size of a read-1 $(\vee, \wedge)$ circuit computing f (Theorem 1).
2. Read-1 $(\vee, \wedge)$ circuits have the *same* power as *tropical* $(\min, +)$ circuits in the following

sense: the minimum size of a $(\min, +)$ circuit computing the tropical polynomial $P(x) = \min_{S\in\mathcal{F}} \sum_{i\in S} x_i$ *coincides* with the minimum size of a read-1 $(\vee, \wedge)$ circuit computing the Boolean version $f(x) = \bigvee_{S\in\mathcal{F}} \bigwedge_{i\in S} x_i$ of P (Theorem 2).
3. Read-1 $(\vee, \wedge)$ circuits are *not weaker* than multilinear DeMorgan $(\vee, \wedge, \neg)$ circuits in the following sense: if a multilinear $(\vee, \wedge, \neg)$ circuit computes a Boolean function $f(x)$, then a read-1 $(\vee, \wedge)$ circuit of the same size computes the monotone function $f^{\triangledown}(x) := \bigvee_{z\leqslant x} f(z)$ (Theorem 3); note that $f^{\triangledown} = f$ if f is monotone. If the function f is monotone and homogeneous (all prime implicants of f have the same number of variables), then the minimum size of a multilinear $(\vee, \wedge, \neg)$ circuit computing f even *coincides* with the minimum size of a read-1 $(\vee, \wedge)$ circuit computing f (Theorem 4).
4. Already 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 8).

Organization. In the preliminary Section 3, 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 4. The aforementioned relation of read-1 circuits to monotone arithmetic circuits is established in Section 5. In Section 5.1 we recall one relatively simple argument to show large lower bounds for monotone arithmetic $(+, \times)$ circuits; this is only aimed to demonstrate that the absence of idempotence and absorption in such circuits is indeed a severe restriction. The relation of read-1 circuits to tropical circuits is established in Section 6, and the relation of read-1 circuits to multilinear DeMorgan $(\vee, \wedge, \neg)$ circuits is established in Section 7. An exponential gap between read-1 and read-2 circuits is shown in Section 8. In the concluding Section 9, several open problems are formulated. Appendix A contains a construction of so-called "blocking lines" functions as possible candidates to attack these problems. All proofs are fairly simple.

3. Preliminaries

In this section, we recall the classical model of (combinational) circuits over arbitrary semirings, and introduce one simple but useful concept: the set of (exponent) vectors "produced" (purely syntactically) by a circuit.

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," 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 $c \in R$. A circuit is *constant-free* if only variables $x_1, \ldots, x_n$ are used as inputs. 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[2]:

- $x \oplus y := x + y$ 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, also over the tropical semiring, "multiplication" ($\odot$) distributes over "addition" ($\oplus$) because $x + \min\{y, z\} = \min\{x + y, x + z\}$. Also, 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 holding a constant $c \in R$, the constant polynomial $P = c$ is produced, and at a source node holding a variable x_i, the polynomial $P = x_i$ is produced. At an "addition" ($\oplus$) gate, the "sum" $P \oplus Q$ of the polynomials P and Q produced at its inputs is produced. Finally, the polynomial produced at a "multiplication" ($\odot$) gate is obtained from $P \odot Q$ by the distributivity of $\odot$ over $\oplus$; that is, we "multiply" ($\odot$) every monomial of P with every monomial of Q, 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 $P(x) = \sum_{b \in B} c_b \prod_{i=1}^n x_i^{b_i}$ produced at the output gate of F; here, $B \subseteq \mathbb{N}^n$ is the set of *exponent vectors* of the polynomial P, and $x_i^{b_i}$ stands for the b_i-times "multiplication" $x_i \odot x_i \odot \cdots \odot x_i$, and $x_i^0 = \mathbb{1}$ (the multiplicative neutral element). Since by our assumption, the underlying semiring contains the multiplicative neutral element $\mathbb{1}$, coefficients c_b are semiring elements[3].

Remark 1. Let us stress the difference between what circuits *compute* (as functions) and what they actually *produce* (as formal expressions). The point is that, unlike when computing polynomial *functions*, no terms are canceled when *producing* polynomials. Thus, the polynomial function f *computed* by a circuit F may be different from the produced polynomial P due to apparent cancellations of some terms of P: unlike for the polynomial P (which only depends on the circuit F itself), the function f computed by F already depends on the underlying semiring. For example, over the arithmetic semiring $(\mathbb{R}, +, \times)$, the polynomial produced by the circuit $F = (x + y)(x - y)$ is $P = x^2 + c_1 xy + c_2 y^2$ with the set $B = \{(2, 0), (1, 1), (0, 2)\}$ of exponent vectors, and coefficients $c_1 = 1 - 1 = 0$ and $c_2 = -1$, while the polynomial function *computed* by the circuit F is $f = x^2 - y^2$ whose set of exponent vectors is $A = \{(2, 0), (0, 2)\}$. Note, however, that in monotone Boolean $(\vee, \wedge)$ and tropical $(\min, +)$ circuits we have no cancellations like arithmetic $x - x = 0$ because there are no analogs of arithmetic subtraction in the corresponding semirings. In these circuits, we only have cancellations via absorption $x \vee xy = x$ or $\min\{x, x + y\} = x$ when going from the *produced* polynomials P to the functions f actually *computed* by the circuits. In monotone arithmetic $(+, \times)$ circuits we have no cancellations at all: then $f = P$ (as formal expressions, see Fact 1 in Section 5). □

Produced sets of exponent vectors. Of interest for us will be not as much the polynomials $P(x) = \sum_{b \in B} c_b \prod_{i=1}^n x_i^{b_i}$ produced by circuits F themselves but rather the sets $B = B_F \subseteq \mathbb{N}^n$ of exponent vectors of these polynomials. These sets can be 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:

[2] An exception is Section 7, where we also consider non-monotone Boolean $(\vee, \wedge, \neg)$ circuits.

[3] Because then, by distributivity, we have $x \oplus x = (\mathbb{1} \odot x) \oplus (\mathbb{1} \odot x) = (\mathbb{1} \oplus \mathbb{1})x$, where $\mathbb{1} \oplus \mathbb{1}$ *is* a semiring element. An example, where this is not the case is the semiring $(R, +, \times)$ with $R \subset \mathbb{N}$ being the set of all even integers: then the coefficient "3" of $x + x + x = 3x$ is not a semiring element.

- if $F = c \in R$ (a constant), then $B_F = \{\vec{0}\}$;
- if $F = x_i$ (input variable), then $B_F = \{\vec{e}_i\}$;
- if $F = G \oplus H$, then $B_F = B_G \cup B_H$ (set-theoretic union);
- if $F = G \odot H$, then $B_F = B_G + B_H := \{a + b\colon a \in B_G, b \in B_H\}$ (Minkowski sum).

Since the set of exponent vectors of a "product" ($\odot$) of two polynomials is the Minkowski sum of their sets of exponent vectors, and since no cancellations are performed when producing polynomials, the set $B_F \subseteq \mathbb{N}^n$ of vectors produced by a circuit F is the set of exponent vectors of the polynomial produced by the circuit F.

It is clear that the same circuit 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, and computes the Boolean function $xy \vee z$ over the Boolean $(\vee, \wedge)$ semiring. It is, however, important to note that:

- The set $B_F \subseteq \mathbb{N}^n$ of exponent vectors of the polynomial *produced* by a circuit F over any semiring is always the same—it only depends on the circuit F itself, not on the underlying semiring.

This simple observation turned out to be useful when comparing the powers of circuits over *different* semirings.

Notation. We will use standard terminology and notation regarding Boolean functions (see, for example, the books by Wegener [48] or Crama and Hammer [7]). 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 $a \in \{0,1\}^n$. A Boolean *term* is an AND of a nonempty set of *literals*, each being a variable x_i or its negation $\overline{x}_i$. 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 $t \leqslant t' \leqslant f$ implies $t' = t$. For example, if $f = xy \vee x\overline{y}z$, then xy, $x\overline{y}z$ and xz are implicants of f, but $x\overline{y}z$ is not a prime implicant (since $x\overline{y}z \leqslant xz \leqslant f$). A Boolean function f is *monotone* if $a \leqslant b$ implies $f(a) \leqslant f(b)$. It is well known and easy to show (see, for example, [7, Theorem 1.21]) that prime implicants of monotone Boolean functions do not contain negated variables.

4. 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 $R = \{0, 1\}$. To avoid trivialities, we will only consider monotone Boolean circuits computing *non-constant* Boolean functions f. In every $(\vee, \wedge)$ circuit computing such a function f, constant inputs 0 and 1 can be easily eliminated without increasing the circuit size by iteratively applying $1 \wedge x = x$, $0 \wedge x = 0$, $1 \vee x = 1$ and $0 \vee x = x$. Thus:

- We will always (implicitly) assume that monotone Boolean $(\vee, \wedge)$ circuits are *constant-free*, that is, have no constants 0 or 1 as inputs.

Let us fix some notation. The *support* of a vector $a \in \mathbb{N}^n$ is the set

$$\sup(a) := \{i \in [n]\colon a_i \neq 0\}$$

of its nonzero positions. For a set $A \subseteq \mathbb{N}^n$ of vectors, let

$$\operatorname{Sup}(A) := \{\sup(a)\colon a \in A\} \subseteq 2^{[n]}$$

denote the family of supports of vectors of A. The *upward closure* of a set $A \subseteq \mathbb{N}^n$ of vectors is the set

$$A^{\uparrow} := \{b \in \mathbb{N}^n : b \geqslant a \text{ for some } a \in A\}$$

of vectors containing at least one vector of A; a vector b *contains* a vector a if $b_i \geqslant a_i$ holds for all positions $i \in \{1, \ldots, n\}$.

Now let $f : \{0,1\}^n \to \{0,1\}$ be a monotone Boolean function A *lowest one* of f is a vector $a \in \{0,1\}^n$ such that $f(a) = 1$ but $f(b) = 0$ for every vector $b \leqslant a$, $b \neq a$. We will denote the set of all lowest ones of f by A_f. Note that the set A_f is always an *antichain*: $a, b \in A_f$ and $b \leqslant a$ imply $a = b$. Since the function f is monotone, for every input vector $x \in \{0,1\}^n$, we have

$$f(x) = 1 \text{ iff } x \geqslant a \text{ for some } a \in A_f \text{ iff } x \in (A_f)^{\uparrow}.$$

It is also easy to see that $a \in f^{-1}(1)$ iff the term $t_a = \bigwedge_{i \in \sup(a)} x_i$ is an implicant of f. Thus, $a \in A_f$ iff the term t_a is a *prime* implicant of f. It is, therefore, useful to keep in mind that if we view implicants of f as sets of their variables, then

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

For example, if $f = xz \vee y$, then $A_f = \{(1,0,1), (0,1,0)\}$. Let us stress that the only reason why we use such a "vector-representation" of prime implicants is to *unify* (and simplify) the forthcoming comparisons of the powers of *different* types of circuits: Boolean, arithmetic and tropical.

Our starting point is the following simple structural property of sets of exponent vectors produced by monotone Boolean circuits.

Lemma 1 (Folklore). *Let $f : \{0,1\}^n \to \{0,1\}$ be a monotone Boolean function, F be a monotone Boolean $(\vee, \wedge)$ circuit, and $B_F \subseteq \mathbb{N}^n$ be the set of exponent vectors produced by F. Then the following two assertions are equivalent.*

(i) *The circuit F computes f.*
(ii) *Inclusions* $\mathrm{Sup}(A_f) \subseteq \mathrm{Sup}(B_F)$ *and* $B_F \subseteq (A_f)^{\uparrow}$ *hold.*

Proof. Let $A := A_f$ and $B := B_F$. Our Boolean function f is of the form $f(x) = \bigvee_{a \in A} \bigwedge_{i \in \sup(a)} x_i$, and 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 (ii) $\Rightarrow$ (i) implication directly follows from a simple observation: for every input $x \in \{0,1\}^n$, we have $f(x) = 1$ iff $\sup(x) \supseteq \sup(a)$ holds 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 $f(x) \leqslant F(x)$.

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

A *shadow* of a vector $x \in \mathbb{R}^n$ is a vector $y \in \mathbb{R}^n$ with $\sup(x) = \sup(y)$ (with the same set of nonzero positions as x). The inclusion $\mathrm{Sup}(A_f) \subseteq \mathrm{Sup}(B_F)$ in Lemma 1 means that every lowest one of f has at least one shadow in B_F. In general, these shadows may have large entries, even exponential in the circuit size. In read-k circuits, we restrict the magnitude of entries of shadows. Say that a vector $b \in \mathbb{N}^n$ is *k-bounded* if $b_i \leqslant k$ holds for all $i \in \sup(b)$.

Definition 1 (Read-k circuits). Let F be a monotone $(\vee, \wedge)$ circuit, and $B_F \subseteq \mathbb{N}^n$ be the set of exponent vectors produced by F. The circuit F is a *syntactically read-k circuit* if all vectors of B_F are k-bounded, and is a (semantically) *read-k circuit* if every lowest one of the Boolean function $f : \{0,1\}^n \to \{0,1\}$ computed by F has at least one k-bounded shadow in B_F. In particular, F is a read-1 circuit iff the inclusions $A_f \subseteq B_F \subseteq (A_f)^{\uparrow}$ hold.

Thus, F is a (semantically) read-k circuit if $\mathrm{Sup}(A_f) \subseteq \mathrm{Sup}(B_F \cap \{0,1,\ldots,k\}^n)$ holds, and is a syntactically read-k circuit if also $B_F \subseteq \{0,1,\ldots,k\}^n$ holds. It is clear that every syntactically read-k circuit is also a (semantically) read-k circuit, but not the vice versa. Intuitively, a monotone circuit F computing a monotone Boolean function f is a read-k circuit[4] if it can distinguish each vector $a \in A_f \subseteq f^{-1}(1)$ from all vectors in $f^{-1}(0)$ by "reading/accessing" each 1-entry of the vector a at most k times.

For a monotone Boolean function f, let

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

Remark 2. Note that already read-1 circuits are "universal:" every monotone Boolean function f *can* be computed by a read-1 circuit, and even, by a *syntactically* read-1 circuit, for example, as an OR of all 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 $\mathrm{B}_1(f)/\mathrm{B}_2(f)$ can be exponential. □

4.1. Viewing Boolean Circuits as Arithmetic Circuits

As we already mentioned in Section 2, an equivalent and, apparently, more intuitive definition of read-k $(\vee, \wedge)$ circuits comes by looking at their arithmetic versions. An arithmetic circuit is *constat-free* if it has no constants as inputs. The *arithmetic* $(+, \times)$ *version* of a monotone Boolean $(\vee, \wedge)$ circuit F is the constant-free $(+, \times)$ circuit obtained by replacing $\vee$-gates with $+$-gates, and $\wedge$-gates with $\times$-gates. That is, we replace the "addition" gates by "addition" gates, and "multiplication" gates by "multiplication" gates of the corresponding semirings. A *shadow monomial* of a Boolean term $\bigwedge_{i \in S} x_i$ is a monomial $\prod_{i \in S} x_i^{d_i}$ with $d_i \geqslant 1$ for all $i \in S$.

The *formal polynomial* of a monotone Boolean $(\vee, \wedge)$ circuit F is the polynomial $P_F(x) = \sum_{b \in B_F} c_b \prod_{i=1}^{n} x_i^{b_i}$ produced by the arithmetic $(+, \times)$ version of F. Lemma 1 implies that the circuit F computes a (monotone) Boolean function $f : \{0,1\}^n \to \{0,1\}$ iff the formal polynomial P_F of F has the following two properties.

(i) Absorption $x \vee xy = x$: for every monomial $\prod_{i=1}^{n} x_i^{b_i}$ of P_F, the Boolean term $\bigwedge_{i \in \mathrm{sup}(b)} x_i$ is an implicant of f; this is the property $B_F \subseteq (A_f)^{\uparrow}$ in Lemma 1.

(ii) Idempotence $x \wedge x = x$: every prime implicant of f has at least one shadow monomial in P_F; this is the property $\mathrm{Sup}(A_f) \subseteq \mathrm{Sup}(B_F)$ in Lemma 1.

In read-k circuits, the degree of variables in shadow monomials guaranteed by (ii) is restricted The *individual degree* of a monomial is the maximum degree of its variable.

Definition 2 (Arithmetic equivalent of Definition 1). A monotone $(\vee, \wedge)$ circuit F is a *read-k* circuit if every prime implicant of f has at least one shadow monomial in P_F of individual degree $\leqslant k$. If the individual degree of *every* monomial of P_F is $\leqslant k$, then F is a *syntactically read-k* circuit.

[4]The term "read-k circuit" is by analogy with the well-known term "read-k times branching program;" see the discussion at the end of Section 9 (after Problem 3).

Remark 3. If G and H are monotone $(\vee, \wedge)$ circuits, then the formal polynomial of the circuit $F = G \vee H$ is $P_F = P_G + P_H$, and the formal polynomial of the circuit $F = G \wedge H$ is $P_F = P_G \cdot P_H$. If $g, h : \{0,1\}^n \to \{0,1\}$ are the (monotone) Boolean functions computed by the circuits G and H, then every prime implicant of $g \vee h$ is a prime implicant of g or of h, and every prime implicant of $g \wedge h$ is the AND of some prime implicant of g and some prime implicant of h. Thus, the OR of two read-k circuits is again a read-k circuit, while the AND of such circuits is a read-r circuit for some $k \leqslant r \leqslant 2k$. In particular, the "read parameter" k does not increase at OR gates: it can only increase at AND gates. □

4.2. Reducing the total degree

If F is a *syntactically* read-k $(\vee, \wedge)$ circuit computing a monotone Boolean function $f : \{0,1\}^n \to \{0,1\}$, then all monomials of the formal polynomial P_F of F have degree[5] $\leqslant kn$. However, if F is a (not necessarily syntactically) read-k circuit, then only shadow monomials in the formal polynomial P_F must have degree $\leqslant kn$: in this case, the polynomial P_F may also have other "redundant" monomials of very large degree, up to $2^r n$ where r is the maximum number of AND gates along an input-output path in the circuit F; about such monomials of P_F we only know property (i) above.

Still, using a simple observation, usually attributed to Strassen [43] (see, for example, [42, Theorem 2.2]), one can show that the degree of *all* monomials in P_F can be decreased till kn. For a polynomial P of degree d, let $P[i]$ be the sub-polynomial of P consisting of monomials of degree exactly i; hence, $P = P[0] + P[1] + \cdots + P[d]$.

Homogenization Lemma (Strassen [43]). *If a polynomial P of degree d can be produced by an arithmetic circuit of size s, then for every $r \leqslant d$ all homogeneous parts $P[0], P[1], \ldots, P[r]$ of P can be simultaneously produced by an arithmetic circuit of size $\mathcal{O}(sr^2)$.*

Proof sketch. The idea is very simple. Take an arbitrary $i \in \{0, 1, \ldots, r\}$. If $P = Q + R$, then $P[i] = Q[i] + R[i]$, and if $P = Q \cdot R$, then $P[i] = \sum_{j=0}^{i} Q[j] \cdot R[i-j]$. So, we can take $r+1$ copies $v_0, v_1, \ldots, v_r$ of each gate v and connect them accordingly so that at v_i the homogeneous sub-polynomial of total degree i of the polynomial produced at the gate v is produced. Every addition $(+)$ gate is replaced by $r+1$ addition gates, and each multiplication $(\times)$ gate is replaced by $\sum_{i=0}^{r}(2i+1) = (r+1)^2$ gates. Thus, the obtained circuit has at most $s(r+1)^2$ gates. □

The following easy consequence of this lemma shows that, at the cost of a relatively small increase in circuit size, we can assume that formal polynomials of read-k circuits have degree $\leqslant kn$.

Lemma 2. *Let $f(x_1, \ldots, x_n)$ be a monotone Boolean function, and m be the maximal number of variables in a prime implicant of f. If f can be computed by a read-k circuit F of size s, then f can also be computed by a read-k circuit H of size at most s times $\mathcal{O}(k^2m^2)$ whose formal polynomial P_H has degree* $\deg(P_H) \leqslant km$.

Proof. Let $A := A_f$ be the set of the lowest ones of f, that is, of characteristic 0-1 vectors of prime implicants of f. Since, by our assumption, no prime implicant of f has more than m variables, we have $|\mathrm{sup}(a)| \leqslant m$ for all $a \in A$. Suppose that the function f can be computed by a read-k $(\vee, \wedge)$ circuit F of size s, and let $P(x) = \sum_{b \in B} c_b \prod_{i=1}^{n} x_i^{b_i}$ be the polynomial produced by the arithmetic

[5] As customary, the *degree* of a monomial $\prod_{i=1}^{n} x_i^{b_i}$ is the sum $b_1 + \cdots + b_n$ of the degrees of its variables. Note that, if a monomial has individual degree $\leqslant k$, then its degree is $\leqslant kn$. The *degree* $\deg(P)$ of a polynomial is the maximum degree of its monomial.

$(+, \times)$ version of the circuit F; hence, $P = P_F$ is the formal polynomial of the circuit F. Since the circuit F computes the function f, Lemma 1 gives us the inclusions $\mathrm{Sup}(A) \subseteq \mathrm{Sup}(B)$ and $B \subseteq A^{\uparrow}$.

Consider the sub-polynomial $P'(x) := \sum_{b \in B'} c_b \prod_{i=1}^{n} x_i^{b_i}$ of P whose set of exponent vectors is $B' = \{b \in B \colon b_1 + \cdots + b_n \leqslant km\}$. That is, the polynomial $P' = P[0] + P[1] + \cdots + P[km]$ consists of all terms of P of degree $\leqslant km$; hence, $\deg(P') \leqslant km$. Since the circuit F is a read-k circuit, for every lowest one $a \in A_f$ there is a monomial $\prod_{i=1}^{n} x_i^{b_i}$ in P (a shadow of a) with $\sup(b) = \sup(a)$ and $b_i \leqslant k$ for all $i \in \sup(b)$. Since then $b_1 + \cdots + b_n \leqslant k|\sup(a)| \leqslant km$, the polynomial P' contains shadows of *all* prime implicants of f. This is a crucial property that gives us the inclusion $\mathrm{Sup}(A) \subseteq \mathrm{Sup}(B')$.

Since the polynomial P can be produced by a $(+, \times)$ circuit of size s, Strassen's homogenization lemma implies that the sub-polynomial P' of P can be produced by an arithmetic $(+, \times)$ circuit F' of size $\mathcal{O}(sk^2m^2)$. Let H be the Boolean $(\vee, \wedge)$ version of the circuit F' obtained by replacing $+$ gates by $\vee$-gates, and $\times$ gates by $\wedge$-gates. That is, we replace the "addition" gates by "addition" gates, and "multiplication" gates by "multiplication" gates of the corresponding semirings. The polynomial P' is the formal polynomial P_H of the circuit H. In particular, $\deg(P_H) = \deg(P') \leqslant km$. The set of "exponent" vectors produced by the circuit H is the set B' of exponent vectors produced by the circuit F'. Hence, the Boolean function computed by H is the Boolean version $h(x) = \bigvee_{b \in B'} \bigwedge_{i \in \sup(b)} x_i$ of the polynomial $P'(x) = \sum_{b \in B'} c_b \prod_{i=1}^{n} x_i^{b_i}$ produced by F'. Since $\mathrm{Sup}(A) \subseteq \mathrm{Sup}(B')$ and $B' \subseteq B \subseteq A^{\uparrow}$, Lemma 1 implies that the *function* h is the same as our function f. Since the $(\vee, \wedge)$ circuit F was a read-k circuit, the circuit H is also a read-k circuit. ∎

5. From Monotone Arithmetic to Boolean Read-1

A monotone arithmetic $(+, \times)$ circuit is a circuit over the arithmetic semiring $(R, \oplus, \odot)$ with $x \oplus y := x + y$ (arithmetic addition) and $x \odot y := x \times y$ (arithmetic multiplication); the domain is the set $R = \mathbb{R}_+$ of all *nonnegative* real numbers; hence, the adjective "monotone:" since there are no negative constant inputs (like -1), there are no cancellations $x - x = 0$. 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 [2, Lemma 2.1]): If P is a nonzero n-variate polynomial in which every variable has 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 [2] by an easy induction on the number n of variables, an gives the following property of monotone arithmetic circuits.

Fact 1. *If a monotone arithmetic circuit* computes *a given polynomial, then the circuit also* produces *that polynomial.*

Proof. Let F be a monotone arithmetic $(+, \times)$ circuit computing a polynomial P, and let Q be the polynomial produced by F. Since the circuit F is monotone, it has no negative constants as inputs. So, the coefficients in both polynomials P and Q are positive. Suppose for the sake of contradiction that the polynomials P and Q do not coincide (as formal expressions). Then $P - Q$ is a nonzero polynomial of a (possibly large but) *finite* degree d. By taking any set $S \subseteq \mathbb{N}$ of $|S| \geqslant d+1$ numbers, the aforementioned result of Alon and Tarsi implies that $P(x) - Q(x) \neq 0$ for some $x \in S^n$, a contradiction with our assumption that the circuit F computes the polynomial P. ∎

Say that a polynomial Q is *similar* to a polynomial $P_A(x) = \sum_{a \in A} \prod_{i=1}^{n} x_i^{a_i}$ if it is of the form $Q(x) = \sum_{a \in A} c_a \prod_{i=1}^{n} x_i^{a_i}$ for some positive integer coefficients $c_a \geqslant 1$ (note that the set A of exponent vectors of Q is the *same* as that of P_A). In particular, the polynomial P_A is similar to itself. Recall that an arithmetic circuit is *constant-free* if it has no constants as inputs. For a set $A \subseteq \mathbb{N}^n$ of vectors, let

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

In particular, $\mathsf{Arith}(A)$ is a lower bound on the size of any monotone arithmetic $(+,\times)$ circuit computing the polynomial $P_A(x) = \sum_{a\in A}\prod_{i=1}^n x_i^{a_i}$.

The model of monotone arithmetic $(+,\times)$ circuits has been studied in many papers, including [8, 9, 14, 17, 33, 38, 41, 46, 47]. Strong, even exponential lower bounds on $\mathsf{Arith}(A)$ for explicit sets $A \subseteq \{0,1\}^n$ are known. Already in 1976, Schnorr [38] has proved that $\mathsf{Arith}(A) \geqslant |A| - 1$ holds for every set $A \subseteq \mathbb{N}^n$ which is *cover-free* in that $a + b \geqslant c$ with $a, b, c \in A$ implies $c \in \{a, b\}$. For example, any set $A \subseteq \{0,1\}^n$ of vectors with m ones, no two of which share $\lfloor m/2 \rfloor$ ones in common, is cover-free. Also, as shown in [38], the set of characteristic 0-1 vectors of cliques in K_n (viewed as sets of their edges) on the same number of vertices is also cover-free.

Gashkov and Sergeev [9] substantially extended Schnorr's result by showing that $\mathsf{Arith}(A) \geqslant |A|/\max\{t^3, s^2\} - 1$ holds for every set $A \subseteq \mathbb{N}^n$ which is (t,s)-*thin* in the following sense: for every sets $X, Y \subseteq \mathbb{N}^n$ of vectors, the inclusion $X + Y \subseteq A$ implies $|X| \leqslant t$ or $|Y| \leqslant s$. It is easy to see that cover-free sets are $(1,1)$-thin sets[6]. Together with the construction of $(t, t!)$-thin sets $A \subseteq \{0,1\}^n$ by Kollár, Rónyai and Szabó [20] (via so-called *norm-graphs*), this yields the best known lower bound $\mathsf{Arith}(A) \geqslant 2^{n/2-o(n)}$ on the size of monotone arithmetic circuits computing any polynomial similar to the explicit multilinear polynomial $P(x) = \sum_{a\in A}\prod_{i=1}^n x_i^{a_i}$ (see [9, Theorem 3] or [17, Appendix E] for more details).

On the other hand, monotone Boolean read-k $(\vee,\wedge)$ circuits and monotone arithmetic $(+,\times)$ circuits are interrelated. By Fact 1, monotone $(+,\times)$ circuits produce what they compute. Hence, Lemma 1 implies that for every monotone Boolean function $f : \{0,1\}^n \to \{0,1\}$ and for any $k \geqslant 1$, we have

$$\mathsf{B}_k(f) = \min\left\{\mathsf{Arith}(B)\colon\ \mathrm{Sup}(A_f) \subseteq \mathrm{Sup}(B \cap \{0,1,\ldots,k\}^n) \text{ and } B \subseteq (A_f)^{\uparrow}\right\}. \tag{1}$$

In particular, for every monotone Boolean function $f : \{0,1\}^n \to \{0,1\}$ and for every integer $k \geqslant 1$ there is a set $B \subseteq \mathbb{N}^n$ of vectors satisfying the conditions in Eq. (1) such that $\mathsf{B}_k(f) \geqslant \mathsf{Arith}(B)$ holds, that is, $\mathsf{B}_k(f)$ is at least the minimum size of a monotone arithmetic constant-free $(+,\times)$ circuit computing a polynomial similar to $P(x) = \sum_{b\in B}\prod_{i=1}^n x_i^{b_i}$. Thus, at least in principle, lower bounds on the size of Boolean read-k $(\vee,\wedge)$ circuits *can* be obtained by proving lower bounds on the size of monotone arithmetic $(+,\times)$ circuits. The problem, however, is that we know only little about the structure of the sets $B \subseteq \mathbb{N}^n$ of exponent vectors of the polynomial to be considered: we only know that the two inclusions in Eq. (1) hold. Fortunately, for $k = 1$ (read-once circuits), the situation is much better: it is then enough to prove that $\mathsf{Arith}(B)$ is large for the so-called "lower envelope" $B \subseteq A_f$ of the (known to us) set A_f.

The *lower envelope* $\lfloor B \rfloor \subseteq B$ of a set $B \subseteq \mathbb{N}^n$ of vectors consists of all vectors $b \in B$ of the smallest degree, where the *degree* of a vector $b \in \mathbb{N}^n$ is the sum $|b| := b_1 + \cdots + b_n$ of its entries. Note that for a monotone Boolean function f, $\lfloor A_f \rfloor$ is the set of characteristic 0-1 vectors of *shortest* implicants of

[6] Suppose that a set A is *not* a $(1,1)$-thin set. Then the inclusion $\{x, x'\} + \{y, y'\} \subseteq A$ holds for some vectors $x \neq x'$ and $y \neq y'$. But then the sum $(x + y) + (x' + y')$ of two vectors of A contains a third vector $x + y'$ of A, meaning that A is *not* a cover-free set.

f (those with the smallest number of variables). A set $B \subseteq \mathbb{N}^n$ is *homogeneous* of degree m, if all its vectors have the same degree m; note that then $\lfloor B \rfloor = B$ holds.

If d is the minimum degree of a vector in B, then Strassen's homogenization lemma (see Section 4.2) implies that $\mathsf{Arith}(\lfloor B \rfloor)$ is at most $\mathsf{Arith}(B)$ times $\mathcal{O}(d^2)$. However, this additional factor $\mathcal{O}(d^2)$ can be eliminated using the fact that the set B contains no vectors of degree *smaller* than d.

Envelope Lemma (Jerrum and Snir [14]). *For every $B \subseteq \mathbb{N}^n$, $\mathsf{Arith}(\lfloor B \rfloor) \leqslant \mathsf{Arith}(B)$.*

Proof. Take a monotone arithmetic constant-free $(+, \times)$ circuit F of size $s = \mathsf{Arith}(B)$ computing some polynomial $Q(x) = \sum_{b \in B} c_b \prod_{i=1}^n x_i^{b_i}$ similar to the polynomial $P(x) = \sum_{b \in B} \prod_{i=1}^n x_i^{b_i}$. By Fact 1, the circuit F also *produces* the polynomial Q. The polynomial $Q'(x) = \sum_{b \in \lfloor B \rfloor} c_b \prod_{i=1}^n x_i^{b_i}$ is similar to the polynomial $P'(x) = \sum_{b \in \lfloor B \rfloor} \prod_{i=1}^n x_i^{b_i}$. Hence, it is enough to show that the polynomial Q' can also be produced by a monotone arithmetic constant-free $(+, \times)$ circuit F' of size at most s.

We will obtain the desired circuit F' from the circuit F by appropriately discarding some of the edges entering $+$-gates. For a gate v in the circuit F, let $B_v \subseteq \mathbb{N}^n$ be the set of exponent vectors of the polynomial produced at v, and let $\mathrm{dg}(v)$ denote the *minimum* degree $b_1 + \cdots + b_n$ of a vector $b \in B_v$. Note that $B_v = B$ holds for the output gate v of F.

If $v = u \times w$ is a multiplication gate, then $B_v = B_u + B_w$ (Minkowski sum). Since the degree of a sum of two vectors is the sum of their degrees, we have $\mathrm{dg}(v) = \mathrm{dg}(u) + \mathrm{dg}(w)$ and, hence, also $\lfloor B_v \rfloor = \lfloor B_u + B_w \rfloor = \lfloor B_u \rfloor + \lfloor B_w \rfloor$. So, we do nothing in this case. If $v = u + w$ is an addition gate, then $B_v = B_u \cup B_w$. If $\mathrm{dg}(u) = \mathrm{dg}(w)$, then $\lfloor B_v \rfloor = \lfloor B_u \cup B_w \rfloor = \lfloor B_u \rfloor \cup \lfloor B_w \rfloor$, and we also do nothing in this case. However, if $\mathrm{dg}(u) < \mathrm{dg}(w)$, then $\lfloor B_v \rfloor = \lfloor B_u \cup B_w \rfloor = \lfloor B_u \rfloor$. In this case, we discard the edge (w, v): delete the edge (w, v), delete the $+$ operation labeling the gate v, and contract the remaining edge (u, v). If $\mathrm{dg}(u) > \mathrm{dg}(w)$, 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 (meaning that all prime implicants of f have the same number of variables); note that then $\lfloor A_f \rfloor = A_f$ holds. For a monotone Boolean function f, let

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

It is clear that $\mathsf{B}_k(f) \leqslant \mathsf{B}_k^*(f)$ always holds.

Theorem 1. *For every monotone Boolean function f, we have*

$$\mathsf{Arith}(\lfloor A_f \rfloor) \leqslant \mathsf{B}_1(f) \leqslant \mathsf{B}_1^*(f) \leqslant \mathsf{Arith}(A_f)\,.$$

In particular, if f is homogeneous, then $\mathsf{Arith}(A_f) = \mathsf{B}_1(f) = \mathsf{B}_1^*(f)$.

Proof. Let $A := A_f \subseteq f^{-1}(1)$ be the set of the 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 F obtained by replacing $\vee$-gates with $+$ gates, and $\wedge$-gates with $\times$ gates. The arithmetic circuit F' has the same number $s = \mathsf{B}_1(f)$ of gates. Since the Boolean circuit F is constant-free (by our assumption throughout the paper), its arithmetic version F' is also constant-free. So, the circuit F' produces a polynomial $P(x) = \sum_{b \in B} c_b \prod_{i=1}^n x_i^{b_i}$ with the same set B of exponent vectors, and some integer coefficients $c_b \geqslant 1$. Since the polynomial P is similar to the polynomial $\sum_{b \in B} \prod_{i=1}^n x_i^{b_i}$, we have $\mathsf{Arith}(B) \leqslant s$. By Envelope Lemma, we have $\mathsf{Arith}(\lfloor B \rfloor) \leqslant \mathsf{Arith}(B) \leqslant s$. So, it is enough to show that $\lfloor A \rfloor = \lfloor B \rfloor$.

Since the (Boolean) circuit F is a read-1 circuit, we know that the inclusions $A \subseteq B \subseteq A^{\uparrow}$ hold. Let m be the minimum degree $|a| = a_1 + \cdots + a_n$ of a vector $a \in A$; hence, $\lfloor A \rfloor = \{a \in A\colon |a| = m\}$. The inclusion $B \subseteq A^{\uparrow}$ means that for every vector $b \in B$ there is a vector $a \in A$ such that $b \geqslant a$. Together with the inclusion $A \subseteq B$, this implies that $\lfloor B \rfloor = \{b \in B\colon |b| = m\}$ and $\lfloor A \rfloor \subseteq \lfloor B \rfloor$. To show the inclusion $\lfloor B \rfloor \subseteq \lfloor A \rfloor$, take an arbitrary vector $b \in \lfloor B \rfloor$; hence, $|b| = m$. Since $B \subseteq A^{\uparrow}$, there must be a vector $a \in A$ such that $b \geqslant a$. Since the set A has no vectors of degree $< m = |b|$, we have $b = a \in \lfloor A \rfloor$. Thus, $\lfloor A \rfloor = \lfloor B \rfloor$ holds, as desired.

To show the inequality $\mathrm{B}_1^*(f) \leqslant \mathsf{Arith}(A)$, let F be a constant-free arithmetic $(+,\times)$ circuit of size $s = \mathsf{Arith}(A)$ computing some polynomial Q similar to $P(x) = \sum_{a \in A} \prod_{i=1}^{n} x_i^{a_i}$. Thus, the set of exponent vectors of the polynomial Q is the same set $A = A_f$ as that of the polynomial P. Let F' be the Boolean $(\vee,\wedge)$ version of F obtained by replacing every $+$-gate with a $\vee$-gate, and every $\times$-gate with a $\wedge$-gate. The resulting Boolean circuit F' produces the same set A of exponent vectors as the arithmetic circuit F. Hence, F' computes the Boolean version $f(x) = \bigvee_{a \in A} \bigwedge_{i \in \mathrm{sup}(a)} x_i$ of the polynomial P. Since the set $A = A_f$ of exponent vectors of P consists of only 0-1 vectors, the circuit F' is a *syntactically* read-1 circuit. Hence, $\mathrm{B}_1^*(f) \leqslant s$, as desired. ∎

5.1. *An easy lower bound for monotone arithmetic circuits*

The goal of this section is to demonstrate that strong lower bounds on the size of monotone arithmetic $(+,\times)$ circuits and, hence (by Theorem 1), also on the size of monotone Boolean read-1 $(\vee,\wedge)$ circuits can be proved fairly easily.

The weakness of monotone arithmetic circuits lies in Fact 1: unlike for monotone *Boolean* circuits (where both idempotence $x \wedge x = x$ and absorption $x \vee xy = x$ are allowed), monotone *arithmetic* circuits "produce what they compute." This weakness results in the following "balanced decomposition property" for arithmetic circuits computing homogeneous[7] polynomials observed already by Hyafil [12, Theorem 1] and Valiant [47, Lemma 3].

Decomposition Lemma. *If a homogeneous polynomial f of degree $m \geqslant 3$ can be computed by a monotone arithmetic $(+,\times)$ circuit of size s, then f can be written as a sum $f = g_1h_1 + \cdots + g_th_t$ of $t \leqslant s$ products of homogeneous polynomials such that $m/3 \leqslant \deg(g_i) \leqslant 2m/3$ for all $i = 1,\dots,t$.*

In particular, the inclusions $\mathrm{mon}(g_ih_i) \subseteq \mathrm{mon}(f)$ hold for all $i = 1,\dots,t$, where $\mathrm{mon}(f)$ denotes the set of all monomials of f. That is every monomial of g_ih_i must also be a monomial of the computed polynomial (no "redundant" monomials can be produced); this is in stark contrast with Boolean or tropical circuits, where nothing similar holds. The proof of Decomposition Lemma is simple. If a circuit computes the polynomial f, then (by Fact 1) it also produces that polynomial. Since f is homogeneous, polynomials g_v produced at intermediate gates v are also homogeneous. By walking backward from the output gate, and by always choosing that of the two input gates v with larger $\deg(g_v)$, we will find a gate v with $m/3 \leqslant \deg(g_v) \leqslant 2m/3$. Hence, the polynomial f is of the form $f = g_vh_v + \cdots$ for some polynomial h_v. Replace the gate v by constant 0, and argue by induction on circuit size.

Example 2 (Perfect matchings). The *perfect matching* function is a monotone Boolean function Match_n of n^2 variables, one for each edge of $K_{n,n}$, such that $\mathrm{Match}_n(x) = 1$ iff the subgraph of $K_{n,n}$ specified by the input vector $x \in \{0,1\}^{n\times n}$ contains a perfect matching. We will use the decomposition lemma

[7]A polynomial $f(x) = \sum_{a \in A} c_a \prod_{i=1}^{n} x_i^{a_i}$ is monotone if $c_a > 0$ for all $a \in A$, and is *homogeneous* of degree $\deg(f) = m$ if $a_1 + \cdots + a_n = m$ holds for all $a \in A$.

to prove the following lower bound:

$$\text{For } f = \text{Match}_n, \text{ we have } \mathsf{B}_1(f) = \mathsf{Arith}(A_f) = 2^{\Omega(n)}.$$

Proof. The set A_f of the lowest ones of this function consists of $|A_f| = n!$ characteristic 0-1 vectors $a \in \{0,1\}^{n\times n}$ of all perfect matchings (viewed as sets of their edges). Since the set A_f is homogeneous (of degree n), Theorem 1 yields $\mathsf{B}_1(f) = \mathsf{Arith}(A_f)$, and it remains to prove the lower bound $\mathsf{Arith}(A_f) = 2^{\Omega(n)}$. The polynomial $\sum_{a\in A_f}\prod_{i,j=1}^n x_{i,j}$ is the well known permanent polynomial $\text{Per}_n(x) = \sum_\pi \prod_{i=1}^n x_{i,\pi(i)}$, where the sum is over all $n!$ permutations $\pi : [n] \to [n]$. Hence, $\mathsf{Arith}(A_f)$ is the minimum size s of a monotone arithmetic circuit F computing a polynomial similar to the polynomial Per_n.

To apply the decomposition lemma, take an arbitrary polynomial of the form gh with $\text{mon}(gh) \subseteq \text{mon}(\text{Per}_n)$ and $\deg(g) = r$ for some $n/3 \leqslant r \leqslant 2n/3$. Since the polynomial Per_n is homogeneous of degree n, the polynomials g and h must be homogeneous of degrees $n/3 \leqslant \deg(g) = r \leqslant 2n/3$ and $n/3 \leqslant \deg(h) = n-r \leqslant 2n/3$. Fix any two monomials $p \in \text{mon}(g)$ and $q \in \text{mon}(h)$; hence, p corresponds to a matching in $K_{n,n}$ with r edges, and q corresponds to a matching in $K_{n,n}$ with $n-r$ edges; since the polynomial Per_n is multilinear, these two matchings must be vertex-disjoint. A matching in $K_{n,n}$ with r edges is contained in only $(n-r)!$ perfect matchings. So, $|\text{mon}(h)| = |\text{mon}(p\cdot h)| \leqslant (n-r)!$ and $|\text{mon}(g)| = |\text{mon}(g\cdot q)| \leqslant r!$. This gives an upper bound $|\text{mon}(gh)| = |\text{mon}(g)|\cdot|\text{mon}(h)| \leqslant r!(n-r)!$ on the number of monomials in the polynomial gh. By Decomposition Lemma, the circuit F must have $s \geqslant |\text{mon}(f)|/|\text{mon}(gh)| \geqslant n!/r!(n-r)! = \binom{n}{r}$ gates. Since $\binom{n}{r} \geqslant \binom{n}{n/3}$ for every $n/3 \leqslant r \leqslant 2n/3$, we have $\mathsf{Arith}(A_f) \geqslant \binom{n}{n/3}$. ∎

Remark 4. By using the permanent equivalent of Laplace's expansion rule for determinants, Jerrum and Snir [14] have shown that the permanent polynomial Per_n (of n^2 variables) can be computed by a monotone arithmetic $(+,\times)$ circuit using at most $t := n(2^{n-1}-1)$ multiplication $(\times)$ gates (using a more subtle argument as in the proof above, it is proved in [14] that this number of multiplication gates is also necessarily.) On the other hand, an argument similar to that used by Alon and Boppana [1, Lemma 3.15] for monotone Boolean circuits implies that if a multilinear n-variate polynomial P can be computed by a monotone arithmetic $(+,\times)$ circuit with t multiplication $(\times)$ gates, then a polynomial similar to P can be computed by a monotone arithmetic $(+,\times)$ circuit with t multiplication $(\times)$ gates and $\mathcal{O}(tn+t^2)$ addition $(+)$ gates. Thus, the minimum size of a monotone arithmetic $(+,\times)$ circuit computing Per_n is $2^{\Theta(n)}$. ∎

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

A tropical $(\min,+)$ circuit is a circuit over the tropical semiring $(R,\oplus,\odot)$ with $x\oplus y := \min\{x,y\}$ and $x\odot y := x+y$ (arithmetic addition); the domain is the set $R = \mathbb{R}_+$ of all nonnegative real numbers.

Note that 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 $a_ix_i = x_i+x_i+\cdots+x_i$. 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" $c_a\in\mathbb{R}_+$; vectors $a\in A$ are usually called *feasible solutions* of the corresponding minimization problem. For example, an arithmetic polynomial $P(x,y) = 2x^3y+4xy^2$ turns into the tropical polynomial $f(x,y) = \min\{3x+y+2, x+2y+4\}$

A $(\min,+)$ circuit F *approximates* a given minimization problem $f:\mathbb{R}_+^n\to\mathbb{R}_+$ within a factor $k\geqslant 1$ if for every input weighting $x\in\mathbb{R}_+^n$, the inequalities $f(x)\leqslant F(x)\leqslant k\cdot f(x)$ hold. That is, the circuit is not allowed to output any better (smaller) than optimal values but is allowed to output up to k times worse than the optimal values. In particular, the circuit F *solves* the problem f exactly (approximates f within factor $k = 1$) if $F(x) = f(x)$ holds for all $x \in \mathbb{R}_+^n$.

The minimization problem $f : \mathbb{R}_+^n \to \mathbb{R}_+$ represented by a tropical polynomial $f(x) = \min_{a \in A}\langle a, x\rangle + c_a$ is *constant-free* if $c_a = 0$ for all $a \in A$. Combinatorial optimization problems are usually constant-free. For example, in the famous *MST problem* (minimum weight spanning tree problem) on a given graph G, the goal is to compute the constant-free $(\min, +)$ polynomial $f(x) = \min_{a \in A}\langle a, x\rangle$, where A is the set of characteristic 0-1 vectors of spanning trees of G (viewed as sets of their edges). In the not less prominent *assignment problem*, A is the set of characteristic 0-1 vectors of perfect matchings, etc.

Since constant inputs in $(\min, +)$ circuits can only affect the "coefficients" c_a, such inputs "should" be of little use when solving constant-free minimization problems. This intuition was confirmed in [18, Lemma 3.2] using a simple argument (which we include for completeness): when dealing with tropical $(\min, +)$ circuits approximating constant-free minimization problems, we can safely restrict ourselves to constant-free circuits. The *constant-free version* of a $(\min, +)$ circuit F is obtained by replacing all constant inputs with constant 0.

Lemma 3. *If a* $(\min, +)$ *circuit F approximates a constant-free minimization problem within a factor* $k \geqslant 1$*, then the constant-free version of F also approximates this problem within the same factor.*

Proof. Let $f(x) = \min_{a \in A}\langle a, x\rangle$ be a (constant-free) minimization problem approximated by the circuit F within factor k, 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 approximates f within the factor $k \geqslant 1$, the inequalities $f(x) \leqslant g(x) \leqslant k \cdot f(x)$ hold for all weightings $x \in \mathbb{R}_+^n$. We have to show that $f(x) \leqslant g^o(x) \leqslant k \cdot f(x)$ also holds for all $x \in \mathbb{R}_+^n$. Since the constants c_b are nonnegative, we have $g^o(x) \leqslant g(x) \leqslant k \cdot f(x)$ for all $x \in \mathbb{R}_+^n$. To show that $f(x) \leqslant g^o(x)$ holds as well, 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. So the constant $\lambda := 2c/d$ is positive. Since $g^o(x_0) = f(x_0) - d$, we obtain $g(\lambda x_0) \leqslant g^o(\lambda x_0) + c = \lambda \cdot g^o(x_0) + c = \lambda[f(x_0) - d] + c = \lambda \cdot f(x_0) - c = f(\lambda x_0) - c$, which is strictly smaller than $f(\lambda x_0)$, a contradiction with $f(x) \leqslant g(x)$ for all $x \in \mathbb{R}_+^n$. ◻

If a tropical $(\min, +)$ circuit F solves the minimization problem $f_A(x) = \min_{a \in A} \langle a, x\rangle$ on a given set A of feasible solutions, then the set $B_F \subseteq \mathbb{N}^n$ of "exponent" vectors does not need to coincide with A. For example, the circuit $F = \min\{x, y\} + \min\{x, y\}$ solves the minimization problem $f_A = \min\{2x, 2y\}$ on $A = \{(2, 0), (0, 2)\}$ by producing the set $B_F = \{(2, 0), (1, 1), (0, 2)\}$ of "exponent" vectors. Note that the vector $(1, 1)$ is the convex combination $\frac{1}{2}(2, 0) + \frac{1}{2}(0, 1)$ of the vectors of A.

And indeed, using a version of Farkas' lemma, Jerrum and Snir [14] have shown that the structures of the sets A and B_F are related via convexity. Namely, a $(\min, +)$ circuit F solves the minimization problem f_A of a set $A \subseteq \mathbb{N}^n$ of feasible solutions if and only if every vector of A contains[8] some convex combination of vectors in B_F, and every vector of B_F contains some convex combination of vectors in A. The "if" direction here is simple: the scalar product of $x \in \mathbb{R}^n$ with a convex combination of some collection of vectors is at least the minimum scalar product of $x \in \mathbb{R}^n$ with some of these vectors. The Farkas lemma is used in [14] to show the "only if" direction.

In the case of 0-1 optimization, feasible solutions $a \in A$ are 0-1 vectors. In this case, the following properties of sets B_F can be easily proved without any use of Farkas' lemma.

[8] A vector $x \in \mathbb{R}^n$ *contains* a vector $y \in \mathbb{R}^n$ if $x \geqslant y$ holds, that is, if $x_i \geqslant y_i$ for all $i = 1, \ldots, n$.

Lemma 4 (Structure). *Let F be a constant-free* $(\min, +)$ *circuit, and $B_F \subseteq \mathbb{N}^n$ be the set of "exponent" vectors produced by F. If the circuit F approximates the minimization problem $f_A(x) = \min_{a \in A}\langle a, x\rangle$ on an antichain $A \subseteq \{0,1\}^n$ within a factor $k \geqslant 1$, then $B_F \subseteq A^{\uparrow}$ and for every vector $a \in A$ there is a vector $b \in B_F$ such that* $\sup(b) = \sup(a)$ *and* $\langle a, b\rangle \leqslant k \cdot \langle a, a\rangle$. *In particular, if $k = 1$, then $A \subseteq B_F \subseteq A^{\uparrow}$.*

Proof. Let $B := B_F$. Since the circuit F is constant-free, the problem solved by F is of the form $f_B(x) = \min_{b \in B}\langle b, x\rangle$. Since F approximates the problem f_A within the factor k, inequalities $f_A(x) \leqslant f_B(x) \leqslant k \cdot f_A(x)$ hold for all $x \in \mathbb{R}_+^n$. 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 $\langle a, x\rangle = f_A(x) \leqslant f_B(x) \leqslant \langle b, x\rangle = 0$. Thus, $\sup(a) \subseteq \sup(b)$. Since $b \in \mathbb{N}^n$ and a is a 0-1 vector, this yields $a \leqslant b$, as desired.

Now take an arbitrary vector $a \in$A, and consider the weighting $x \in \{1, kn+1\}^n$ with $x_i := 1$ for all $i \in \sup(a)$ and $x_i := kn + 1$ for all $i \notin \sup(a)$. Let $b \in B$ be a vector on which the minimum $f_B(x) = \langle b, x\rangle$ is achieved. Then $\langle b, x\rangle = f_B(x) \leqslant k \cdot f_A(x) \leqslant k \cdot \langle a, x\rangle = k \cdot \langle a, a\rangle \leqslant kn$. If $b_i \neq 0$ held for some $i \notin \sup(a)$, then we would have $\langle b, x\rangle \geqslant b_i x_i = b_i(kn+1) > kn$, a contradiction. Thus, the inclusion $\sup(b) \subseteq \sup(a)$ holds. 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 $a' \leqslant a$ and, since the set A is an antichain, we have $a' = a$ and, hence, also $\sup(b) = \sup(a)$. By the definition of the weighting x, this yields $\langle a, b\rangle = \langle b, x\rangle = f_B(x) \leqslant k \cdot \langle a, a\rangle$, as desired. ◻

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

$$\mathsf{Min}_k(A) := \text{smallest size of a } (\min,+) \text{ circuit approximating the minimization problem } g(x) = \min_{a \in A}\langle a, x\rangle \text{ on } A \text{ within the factor } k.$$

Theorem 2. *Let f be a monotone Boolean function, m be the largest number of variables in a prime implicant of f, $k \geqslant 1$ be an integer, and $r = (k-1)m + 1$. Then*

$$\mathsf{B}_r(f) \leqslant \mathsf{Min}_k(A_f) \leqslant \mathsf{B}_k(f)\,.$$

In particular, $\mathsf{Min}_1(A_f) = \mathsf{B}_1(f)$.

Proof. Let $A := A_f \subseteq \{0,1\}^n$ be the set of the lowest ones of f. To show the inequality $\mathsf{Min}_k(A) \leqslant \mathsf{B}_k(f)$, let F be a read-k $(\vee, \wedge)$ circuit of size $\mathsf{B}_k(f)$ computing f, and let $B := B_F \subseteq \mathbb{N}^n$ be the set of "exponent" vectors produced by F. The tropical $(\min, +)$ version F' of F is a constant-free $(\min, +)$ circuit obtained by replacing $\vee$ gates with min gates, and $\wedge$ gates with addition gates. That is, we (again) replace "addition" gates by "addition" gates, and "multiplication" gates by "multiplication" gates of the corresponding semirings. The circuit F' produces the same set B of "exponent" vectors. Hence, the circuit F' solves the minimization problem $g_B(x) = \min_{b \in B}\langle b, x\rangle$ on the set B. The minimization problem on the given set A is $g_A(x) = \min_{a \in A}\langle a, x\rangle$. It thus remains to show that for every input weighting $x \in \mathbb{R}_+^n$ the inequalities $g_A(x) \leqslant g_B(x) \leqslant k \cdot g_A(x)$ hold. So, take an arbitrary input weighting $x \in \mathbb{R}_+^n$.

Since the Boolean circuit F computes f, Lemma 1 gives us the inclusion $B \subseteq A^{\uparrow}$, that is, for every vector $b \in B$ there is a vector $a \in A$ such that $b \geqslant a$. Since the weights are nonnegative, this gives the first inequality $g_A(x) \leqslant g_B(x)$. To show the second inequality $g_B(x) \leqslant k \cdot g_A(x)$, take a vector $a \in A$ on which the minimum $g_A(x) = \langle a, x\rangle$ on the input weighting $x \in \mathbb{R}_+^n$ is achieved, and let $S := \sup(a)$ be the support of a. Since F is a read-k circuit, there is a vector $b \in B$ such that $\sup(b) = S$ and $b_i \leqslant k$ for all $i \in S$. Thus, $g_B(x) \leqslant \langle b, x\rangle = \sum_{i \in S} b_i x_i \leqslant k \cdot \sum_{i \in S} x_i = k \cdot \langle a, x\rangle = k \cdot g_A(x)$, as desired.

To show the inequality, $\mathsf{B}_r(f) \leqslant \mathsf{Min}_k(A)$ take a tropical $(\min,+)$ circuit F of size $\mathsf{Min}_k(A)$ approximating the (constant-free) minimization problem $g_A(x) = \min_{a\in A}\langle a,x\rangle$ within the factor k, and let $B := B_F \subseteq \mathbb{N}^n$ be the set of "exponent" vectors produced by the circuit F. By Lemma 3, we can assume that the circuit F is constant-free, that is, has no nonzero constant inputs. So, the polynomial $g_B(x) = \min_{b\in B}\langle b,x\rangle$ produced by F is also constant-free. The Boolean $(\vee,\wedge)$ version F' of F (obtained by replacing min gates with $\vee$ gates, and addition gates with $\wedge$ gates) produces the same set B of "exponent" vectors. Since the $(\min,+)$ circuit F approximates the problem g_A within the factor k, Lemma 4 implies that the set B has the following two properties: (i) $B \subseteq A^{\uparrow}$, and (ii) for every vector $a \in A$ there is a vector $b \in B$ such that $\mathrm{sup}(b) = \mathrm{sup}(a)$ and $\langle a,b\rangle \leqslant k\cdot\langle a,a\rangle$. This, in particular, yields the inclusion $\mathrm{Sup}(A) \subseteq \mathrm{Sup}(B)$. Together with $B \subseteq A^{\uparrow}$, Lemma 1 implies that the $(\vee,\wedge)$ circuit F' computes our Boolean function f, and it remains to show that F' is a read-r circuit for $r := (k-1)m+1$.

To show this, take an arbitrary lowest one $a \in A$ of f, and let $S := \mathrm{sup}(a)$ be its support; hence, $|S| \leqslant m$. By property (ii), there is a vector $b \in B$ such that $\mathrm{sup}(b) = S$ and $\sum_{i\in S} b_i \leqslant k|S|$. It remains to show that the vector b is r-bounded. Suppose for the sake of contradiction that $b_j \geqslant r+1 = (k-1)m+2$ holds for some position $j \in S$. Since $b_i \geqslant 1$ holds for all $i \in S$, we then have $\sum_{i\in S} b_i \geqslant r+1+(|S|-1) = (k-1)m+2+(|S|-1) \geqslant (k-1)|S|+2+(|S|-1) = k|S|+1$, a contradiction with $\sum_{i\in S} b_i \leqslant k|S|$. ∎

7. From Non-Monotone Multilinear to Monotone Read-1

Due to the lack of strong lower bounds for (non-monotone) arithmetic $(+,\times,-)$ circuits, and because they seem to be the most intuitive circuits for computing *multilinear* polynomials, a successful approach has been to consider a restriction called "multilinearity" of arithmetic circuits, first introduced by Nisan and Wigderson [28].

Recall that a polynomial is multilinear if it does not have any variable with degree larger than 1. An arithmetic $(+,\times,-)$ circuit F is *syntactically multilinear* if the two subcircuits rooted at inputs of any multiplication $(\times)$ gate have no input variables in common. The circuits F is (semantically) *multilinear* if the polynomial *functions* computed at its gates polynomials are multilinear. For example, the polynomial function $f = y$ computed at a gate producing the polynomial $P = x^2 + y - x^2$ is multilinear. Raz [32, Proposition 2.1] observed that minimal semantically multilinear $(+,\times,-)$ *formulas* (circuits whose underlying graphs are trees) are syntactically multilinear. It remains 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 surveys [6, 42]. In particular, Raz [32] 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 [31] proved that a gap between multilinear arithmetic *formulas* and *circuits* can be super-polynomial. Proving super-polynomial lower bounds of the size of multilinear arithmetic *circuits* remains an open problem.

Due to the lack of even super-linear lower bounds on the size of (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[9] F is an $(\vee,\wedge)$ circuit whose inputs

[9] Let us note that every Boolean $(\vee,\wedge,\neg)$ circuit (with negations applied to any gates, not necessarily to only inputs) can be easily transformed into an equivalent DeMorgan $(\vee,\wedge,\neg)$ circuit by only doubling the circuit size (see, e.g., [48, p. 195]): we double all AND and OR gates, one output of a pair is negated, the other one not; after that, we can move negation gates toward the input variables by applying the DeMorgan rules.

are the variables $x_1, \dots, x_n$ and their negations $\overline{x}_1, \dots, \overline{x}_n$. 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.

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 $F = (x \vee x\overline{y})y$ is *not* syntactically multilinear. Sengupta and Venkateswaran [39] considered the *connectivity* function which accepts an input $x \in \{0,1\}^{\binom{n}{2}}$ iff the subgraph G_x of K_n specified by the characteristic 0-1 vector x of its set of edges is connected. By adopting the proof of Jerrum and Snir [14] of a lower bound $(4/3)^{n-1}/n$ on the minimum size of monotone arithmetic $(+, \times)$ circuits computing the directed spanning tree polynomial, it was shown in [39] that every monotone syntactically multilinear $(\vee, \wedge)$ circuit computing the connectivity function must have at least $\sqrt{(4/3)^{n-1}/n}$ gates.

Krieger [21] has shown that if the set A_f of the 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 DeMorgan $(\vee, \wedge, \neg)$ circuit computing f must have at least $|A_f| - 1$ gates.

Remark 5. As mentioned in Section 5.1, already in 1976, Schnorr [38] has proved a general lower bound $\mathrm{Arith}(A) \geqslant |A| - 1$ on the monotone *arithmetic* circuit complexity of polynomials, whose sets $A \subseteq \mathbb{N}^n$ of exponent vector are cover-free. This surprising similarity of Krieger's bound in [21] with Schnorr's bound, as well as a possibility to adopt in [39] the argument of Jerrum and Snir, already served as an indication that there "should" be some *general* relation between multilinear DeMorgan $(\vee, \wedge, \neg)$ circuits and monotone arithmetic circuits. Our Theorems 3 and 4 below give such a relation, even for semantically (not only syntactically) multilinear DeMorgan $(\vee, \wedge, \neg)$ circuits: such circuits are *not* stronger than monotone arithmetic circuits. □

Following the analogy with arithmetic circuits, Ponnuswami and Venkateswaran [30] relaxed the *syntactic* multilinearity restriction of Boolean circuits to their *semantic* multilinearity. 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 simple fact is well known; see, for example, [7, Theorem 1.17].

Fact 2 (Folklore). *A Boolean function $f(x_1, \dots, x_n)$ depends on the ith variable x_i iff x_i or $\overline{x}_i$ 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 $\overline{x}_i$ 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 $z \in \{x_i, \overline{x}_i\}$. 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 $t(a) = 0$ (because $t \leqslant f$). 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. □

Say that two Boolean functions are *independent* if they depend on disjoint sets of variables.

Definition 3 (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.

By Fact 2, the functions g and h are independent iff their prime implicants share no common variables (negated or not). However, the terms actually *produced* at the gates *computing* these functions *can* share common variables. This explains the use of the adjective "semantically." For example, the circuit $F = (x \vee xy)(\overline{y} \vee \overline{y}z)$ is not syntactically multilinear, but is (semantically) multilinear because $g = x \vee xy$ depends only on x, while $h = \overline{y} \vee \overline{y}z$ depends only on y.

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. Also note that $(g \vee h)^{\triangledown} = g^{\triangledown} \vee h^{\triangledown}$ holds for any Boolean functions $g, h : \{0,1\}^n \to \{0,1\}$. Thus, if $\mathrm{PI}(f)$ is the set of all prime implicants of a Boolean function $f : \{0,1\}^n \to \{0,1\}$, then

$$f^{\triangledown} = \Big(\bigvee_{t \in \mathrm{PI}(f)} t \Big)^{\triangledown} = \bigvee_{t \in \mathrm{PI}(f)} t^{\triangledown} = \bigvee_{t \in \mathrm{PI}(f)} t^{+}\,,$$

where t^+ is the *positive factor* of a term t obtained from t by replacing every negated literal $\overline{x}_i$ with constant 1. Thus, the upward closure $f^{\triangledown}$ of any Boolean function f is the OR of positive factors of prime implicants of f.

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 $b < a$; for vectors $a, b \in \mathbb{R}^n$ we write $b < a$ if $b \leqslant a$ and $b_i < a_i$ for at least one position i. Let, as before, $A_f \subseteq f^{-1}(1)$ denote the set of all lowest ones of f. For example, the set $\{\vec{e}_1, \ldots, \vec{e}_n\}$ of n unit vectors is the set of the lowest ones of $x_1 \oplus x_2 \oplus \cdots \oplus x_n$ as well as of $x_1 \vee x_2 \vee \cdots \vee x_n$. Note that, unlike for *monotone* Boolean functions, $a \in A_f$ does not exclude that $f(c) = 0$ holds for some vectors $c \geqslant a$. For example, $a = (1,0)$ is a lowest one of the function $f = x\overline{y} \vee \overline{x}y = x \oplus y$, but $f(c) = 0$ for $c = (1,1)$.

Remark 6. It is easy to verify that the lowest ones of a function $f : \{0,1\}^n \to \{0,1\}$ and of its downward closure $f^{\triangledown}$ are the same, that is, $A_{f^{\triangledown}} = A_f$ holds. Indeed, if $a \in A_f$, then $f(a) = 1$ but $f(b) = 0$ for all $b < a$. Hence, also $A_{f^{\triangledown}}(a) = 1$ but $A_{f^{\triangledown}}(b) = 0$ for all $b < a$. This shows the inclusion $A_f \subseteq A_{f^{\triangledown}}$. If $a \in A_{f^{\triangledown}}$, then $A_{f^{\triangledown}}(a) = 1$ but $A_{f^{\triangledown}}(b) = 0$ and, hence, also $f(b) = 0$ holds for all $b < a$. Since $f^{\triangledown}(a) = 1$ still holds, this can happen only if $f(a) = 1$. Hence, the converse inclusion $A_{f^{\triangledown}} \subseteq A_f$ also holds. ◻

For a Boolean function f, let $\mathsf{B}_{\mathrm{lin}}(f)$ denote the minimum size of a (semantically) 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* (semantically) multilinear $(\vee, \wedge)$ circuit computing f. It is clear that $\mathsf{B}_{\mathrm{lin}}(f) \leqslant \mathsf{B}^{+}_{\mathrm{lin}}(f)$ holds for every *monotone* Boolean function f.

Theorem 3 (Arbitrary functions). *For every Boolean function f, we have*

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

We will actually prove slightly stronger results: (i) instead of just an inequality $\mathsf{B}_1(f^{\triangledown}) \leqslant \mathsf{B}^{+}_{\mathrm{lin}}(f^{\triangledown})$, we shown that every multilinear *monotone* $(\vee, \wedge)$ circuit *is* a read-1 circuit (Lemma 7), and (ii) instead of just an inequality $\mathsf{B}^{+}_{\mathrm{lin}}(f^{\triangledown}) \leqslant \mathsf{B}_{\mathrm{lin}}(f)$, we shown that if a multilinear DeMorgan $(\vee, \wedge, \neg)$ circuit F computes a Boolean function f, then the monotone $(\vee, \wedge)$ circuit F^+, obtained from F by replacing every negated input variable $\overline{x}_i$ with constant 1, is also multilinear and computes $f^{\triangledown}$ (Lemma 6).

Let us first prove the following easy consequence of Theorem 3 for multilinear $(\vee, \wedge, \neg)$ circuits computing *monotone* Boolean functions f, and then prove Theorem 3 itself.

Theorem 4 (Monotone functions). *For every monotone Boolean function f, we have*

$$\mathsf{Arith}(\lfloor A_f \rfloor) \leqslant \mathsf{B}_1(f) \leqslant \mathsf{B}_{\mathrm{lin}}(f) = \mathsf{B}^+_{\mathrm{lin}}(f) \leqslant \mathsf{Arith}(A_f)\,.$$

In particular, of f is also homogeneous, then $\mathsf{Arith}(A_f) = \mathsf{B}_1(f) = \mathsf{B}_{\mathrm{lin}}(f) = \mathsf{B}^+_{\mathrm{lin}}(f)$.

Proof. The inequality $\mathsf{Arith}(\lfloor A_f \rfloor) \leqslant \mathsf{B}_1(f)$ is given by Theorem 1. Since f is monotone, we have $f^\triangledown = f$. So, the equality $\mathsf{B}_{\mathrm{lin}}(f) = \mathsf{B}^+_{\mathrm{lin}}(f)$ follows from a trivial upper bound $\mathsf{B}_{\mathrm{lin}}(f) \leqslant \mathsf{B}^+_{\mathrm{lin}}(f)$ and from the lower bound $\mathsf{B}_{\mathrm{lin}}(f) \geqslant \mathsf{B}^+_{\mathrm{lin}}(f^\triangledown) = \mathsf{B}^+_{\mathrm{lin}}(f)$ given by Theorem 3. It therefore remains to prove the upper bound $\mathsf{B}^+_{\mathrm{lin}}(f) \leqslant \mathsf{Arith}(A_f)$.

For this, let $A := A_f$ be the set of the lowest ones of f, and take a monotone arithmetic constant-free $(+,\times)$ circuit F of size $s = \mathsf{Arith}(A)$ computing some polynomial $P(x) = \sum_{a\in A} c_a \prod_{i=1}^n x_i^{a_i}$ similar to $\sum_{a\in A}\prod_{i=1}^n x_i^{a_i}$. By Fact 1, the circuit F also *produces* the polynomial P. Let F' be the Boolean $(\vee,\wedge)$ version of the circuit F obtained by replacing each $+$-gate by a $\vee$-gate, and each $\times$-gate by $\wedge$-gate. The circuit F' produces the same set A of exponent vectors and, hence, computes our Boolean function $f(x) = \bigvee_{a\in A}\bigwedge_{i\in \sup(a)} x_i$. Since A consists of only 0-1 vectors, the polynomial P produced by the arithmetic circuit F is multilinear, meaning that the polynomials produced at inputs of any multiplication gate cannot share any variables in common. Thus, the Boolean version F' of F is (even syntactically) multilinear. ∎

Remark 7. Ponnuswami and Venkateswaran [30] proved a lower bound $\mathsf{B}^+_{\mathrm{lin}}(f) = \Omega(2^{.459n})$ for the perfect matching function $f = \mathrm{Match}_n$ (which we considered in Section 5.1). On the other hand, using arguments tighter than we used in Example 2, Jerrum and Snir [14] have proved a lower bound $\mathsf{Arith}(A_f) \geqslant n(2^{n-1}-1)$ for f. The function f is homogeneous (each prime implicant has n variables). So, by Theorem 4, the same lower bound $\mathsf{B}_{\mathrm{lin}}(f) \geqslant n(2^{n-1}-1)$ holds even for non-monotone circuits. ∎

Remark 8. Lingas [24] has proved a lower bound $\mathsf{B}^+_{\mathrm{lin}}(f) \geqslant \mathsf{Arith}(A_f)/\mathcal{O}(m^2)$ for every monotone homogeneous Boolean function, where m is the number of variables in the prime implicants of f. On the other hand, Theorem 4 shows that, for homogeneous monotone functions f, we actually have the equality $\mathsf{B}^+_{\mathrm{lin}}(f) = \mathsf{Arith}(A_f)$, and even the equality $\mathsf{B}_{\mathrm{lin}}(f) = \mathsf{Arith}(A_f)$. That is, multilinear (not necessarily monotone) DeMorgan $(\vee,\wedge,\neg)$ circuits computing monotone homogeneous Boolean functions have the *same* power as monotone arithmetic constant-free $(+,\times)$ circuits. ∎

7.1. Proof of Theorem 3

Since lowest ones of a Boolean function f and of its upward closure $g := f^\triangledown$ are the same (see Remark 6), we have $\lfloor A_f \rfloor = \lfloor A_g \rfloor$. By Theorem 1, $\mathsf{Arith}(\lfloor A_f \rfloor) = \mathsf{Arith}(\lfloor A_g \rfloor) \leqslant \mathsf{B}_1(g)$. This shows the first inequality in Theorem 3. To prove the remaining two inequalities of Theorem 3, we first establish (in Lemma 5) the behavior of sets of lowest ones as well as of upward closures of functions computed at the gates of DeMorgan $(\vee,\wedge,\neg)$ circuits. When doing this, we will use the following simple property of independent Boolean functions following from Fact 2. For 0-1 vectors $a,b \in \{0,1\}^n$, $a \vee b \in \{0,1\}^n$ denotes their componentwise OR. For example, if $a = (1,1,0)$ and $b = (0,1,1)$ then $a + b = (1,2,1)$ but $a \vee b = (1,1,1)$.

Fact 3. *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* $\sup(b) \cap \sup(c) = \emptyset$ *(hence, also* $b \vee c = b + c$*) and* $g(b \vee c) = h(b \vee c) = 1$.

Proof. Note that $t(b) = 1$ holds for some prime implicant $t = \bigwedge_{i\in S} x_i \wedge \bigwedge_{j\in T} \overline{x}_j$ 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 2. 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 $h(b \vee c) = h(b + c) = h(\vec{0} + c) = h(c) = 1$. ∎

Recall that the Minkowski sum of two sets $A, B \subseteq \mathbb{R}^n$ is the set $A + B = \{a + b \colon a \in A, b \in B\}$.

Lemma 5. *Let $g, h : \{0,1\}^n \to \{0,1\}$ be Boolean functions.*

(i) *If $f = g \vee h$, then $A_f \subseteq A_g \cup A_h$ and $f^\triangledown = g^\triangledown \vee h^\triangledown$.*
(ii) *If $f = g \wedge h$ and g, h are independent, then $A_f \subseteq A_g + A_h$ and $f^\triangledown = g^\triangledown \wedge h^\triangledown$.*

Proof. To show (i), let $f = g \vee h$. The inclusion $A_f \subseteq A_g \cup A_h$ is trivial: if $a \in A_f$, then $g(a) = 1$ or $h(a) = 1$, and both $g(b) = 0$ and $h(b) = 0$ hold for every vector $b < a$. Thus, either $a \in A_g$ or $a \in A_h$, as desired. 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; hence, $x \geqslant a$ for some lowest one $a \in A_f$. Then, as we have just shown, either $a \in A_g$ or $a \in A_h$ (or both) hold. 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 $h^\triangledown(x) = 1$.

To show (ii), let $f = g \wedge h$, where the functions g and h are independent. Take an arbitrary lowest one $a \in A_f$. 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, $a \geqslant b \vee c$. Since the functions g and h are independent, Fact 3 yields $b \vee c = b + c$ 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 $a = b + c$; hence, $a \in A_g + A_h$. This shows the inclusion $A_f \subseteq A_g + A_h$.

The inequality $f^\triangledown \leqslant g^\triangledown \wedge h^\triangledown$ is trivial and holds for arbitrary (not necessarily independent) Boolean functions g and h: if $f^\triangledown(x) = 1$, then $f(z) = 1$ holds for some vector $z \leqslant x$ and, hence, both $g(z) = 1$ and $h(z) = 1$ also hold. To show the converse inequality $g^\triangledown \wedge h^\triangledown \leqslant f^\triangledown$ for independent functions g and h, 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. Since the functions g and h are independent, Fact 3 yields $g(b \vee c) = h(b \vee c) = 1$ and, hence, also $f(b \vee c) = 1$. Since $b \vee c \leqslant x$, this yields $f^\triangledown(x) = 1$, as desired. ∎

Remark 9. In general, if $f = g \wedge h$, and if the functions g and h are not independent, then even the inclusion $A_f \subseteq A_g \vee A_h := \{b \vee c \colon b \in A_g, x \in A_h\}$ does not need to hold. Take, for example, $g = x\overline{y}\,\overline{z} \vee xy$ and $h = \overline{x}\,\overline{y}z \vee yz$; hence, $f = xyz$. The functions g and h are dependent (their prime implicants xy and yz share a common variable y). The only lowest one of f is $a = (1,1,1)$, the only lowest one of g is $b = (1,0,0)$, and the only lowest one of h is $c = (0,0,1)$. But $a \neq b \vee c$. Also, in general, the inequality $g^\triangledown \wedge h^\triangledown \leqslant f^\triangledown$ does not need to hold. Take, for example, $g = x\overline{y} \vee z$ and $h = \overline{x}y \vee z$; hence, $f = z$. 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 $h^\triangledown(a) \geqslant h(0,1,0) = 1$. ∎

We now turn to the actual proof of the remaining two inequalities $\mathsf{B}^+_{\mathrm{lin}}(f^\triangledown) \leqslant \mathsf{B}_{\mathrm{lin}}(f)$ and $\mathsf{B}_1(f^\triangledown) \leqslant \mathsf{B}^+_{\mathrm{lin}}(f^\triangledown)$ claimed in Theorem 3. This is done in Lemmas 6 and 7 bellow.

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 $f(x) = H(x, \overline{x})$ holds for all $x \in \{0,1\}^n$, where $\overline{x} = (\overline{x}_1, \ldots, \overline{x}_n)$ is the complement of $x = (x_1, \ldots, x_n)$. For example, if $x = (1,0,1,1)$, then $\overline{x} = (0,1,0,0)$. 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 $\overline{x}_i$ in the circuit F with constant 1.

Take, for example, the circuit $F = (x\overline{y} \vee z)(\overline{x}y \vee z)$ computing the Boolean function $f = z$. Its monotone version $F^+ = (x \cdot 1 \vee z)(1 \cdot y \vee z) = (x \vee z)(y \vee z)$ computes the Boolean function $xy \vee z$

which is *different* from the upward closure $f^\triangledown = z$ of f. The following lemma shows that this cannot happen for *multilinear* circuits F.

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, $\mathsf{B}^+_{\mathrm{lin}}(f^\triangledown) \leqslant \mathsf{B}_{\mathrm{lin}}(f)$ holds.*

Proof. Suppose that the circuit $F(x) = H(x, \overline{x})$ is multilinear. To show that the (monotone) circuit $F^+(x) = H(x, \vec{1})$ is multilinear, let g and h be the Boolean functions computed at some AND gate of the circuit $F(x) = H(x, \overline{x})$. Since the circuit $F(x)$ is multilinear, the functions g and h are independent. By Fact 2, this happens precisely when their prime implicants share no variables in common. Since the upward closure of any Boolean function is the OR of positive factors of its prime implicants, Fact 2 implies that the functions $g^\triangledown$ and $h^\triangledown$ are also independent.

Let us now show that the monotone version $F^+ = H(x, \vec{1})$ of F computes the upward closure $f^\triangledown$ of f. Upward closures of input variables x_i are the variables $x_i^\triangledown = x_i$ themselves, while upward closures of negated input variables $\overline{x}_i$ are constant-1 functions $\overline{x}_i^\triangledown = 1$. 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 is computed, the upward closure $f^\triangledown$ of f is computed at this gate in the circuit F^+, as desired. ∎

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

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

Let G and H be the subcircuits of F whose output gates enter the output gate of F, and let g and h be the monotone Boolean functions computed by these subcircuits. Let also $B_G \subseteq \mathbb{N}^n$ and $B_H \subseteq \mathbb{N}^n$ be the sets of "exponent" vectors produced by the subcircuits G and H. We argue by induction on the number s of gates in F. In the basis case $s = 1$, we have $G = x_i$ and $H = x_j$ for some $i, j \in [n]$. Hence, $B_G = \{\vec{e}_i\} = A_g$ and $B_H = \{\vec{e}_j\} = A_h$. So, if $F = G \vee H = x_i \vee x_j$ then $B_F = B_G \cup B_H = \{\vec{e}_i, \vec{e}_j\} = A_f$. If $F = G \wedge H = x_i \wedge x_j$, then $i \neq j$ due to the multilinearity of the circuit F. Hence, also in this case, we have $B_F = B_G + B_H = \{\vec{e}_i + \vec{e}_j\} = A_f$.

Now suppose that the lemma holds for all monotone multilinear $(\vee, \wedge)$ circuits of size at most s, and let F be a monotone multilinear $(\vee, \wedge)$ circuit of size $s + 1$. Since the circuit F is multilinear, both subcircuits G and H are also multilinear. Since each of G and H has at most s gates, the lemma holds for both these subcircuits. Thus, both inclusions $A_g \subseteq B_G$ and $A_h \subseteq B_H$ hold.

If $F = G \vee H$, then $B_F = B_G \cup B_H$ and Lemma 5 gives the inclusion $A_f \subseteq A_g \cup A_h$. So, the desired inclusion $A_f \subseteq B_F$ follows from the induction hypothesis. If $F = G \wedge H$, then $B_F = B_G + B_H$ (Minkowski sum). Since the circuit F is multilinear, the functions g and h are independent, and Lemma 5 yields $A_f \subseteq A_g + A_h$. So, the desired inclusion $A_f \subseteq B_F$ follows again from the induction hypothesis. ∎

7.2. Multilinear Circuits Impede Zero Terms

Lemma 6 rises a natural question: if $F = F(x, \overline{x})$ is a DeMorgan $(\vee, \wedge, \neg)$ circuit computing a Boolean function f, when does its monotone version $F^+ = F(x, \vec{1})$ computes $f^{\triangledown}$? 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 not only computes some Boolean function $f : \{0,1\}^n \to \{0,1\}$ but also *produces* (purely syntactically) a unique set $T(F)$ of Boolean terms in a natural way:

- if $F = z$ is an input literal $z \in \{x_i, \overline{x}_i\}$, then $T(F) = \{z\}$;
- if $F = F_1 \vee F_2$, then $T(F_1 \vee F_2) = T(F_1) \cup T(F_2)$;
- if $F = F_1 \wedge F_2$, then $T(F) = \{t_1 \wedge t_2 \colon t_1 \in T(F_1), t_2 \in T(F_2)\}$.

During the production of terms, the "annihilation" law $x \wedge \overline{x} = 0$ is not used. So, $T(F)$ can contain *zero terms*, that is, terms containing a variable x_i together with its negation $\overline{x}_i$. For example, the set $T(F) = \{x\overline{y}, x\overline{y}z, xy\overline{y}, xy\overline{y}z\}$ of terms produced by the circuit $F = (x \vee xy)(\overline{y} \vee \overline{y}z)$ contains two zero terms $xy\overline{y}$ and $xy\overline{y}z$.

Recall that the *positive factor* t^+ of a Boolean term t is obtained by replacing every its negated literal $\overline{x}_i$ 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 t^+ of zero terms $t \in T(F)$ produced by F (if there are any) are implicants of $f^{\triangledown}$, that is, if $t^+ \leqslant f^{\triangledown}$ holds for every zero term $t \in T(F)$. Note that such a circuit does *not* forbid a production of zero terms as such, but rather "impedes" produced zero terms to unfold the full power of cancellations $x \wedge \overline{x} = 0$.

Fact 4. *Let F be a DeMorgan $(\vee, \wedge, \neg)$ circuit computing a Boolean function f with $f(\vec{0}) = 0$. The circuit F^+ computes $f^{\triangledown}$ 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 $t^{\triangledown} = 0$ (the constant 0 function) if t is a zero term, and $t^{\triangledown} = t^+$ if t is a nonzero term. So, if $T_0 \subseteq T$ is the set of all zero terms in T, then (where the second equality follows from Lemma 5(i)):

$$f^{\triangledown} = \Big(\bigvee_{t \in T} t\Big)^{\triangledown} = \bigvee_{t \in T} t^{\triangledown} = \bigvee_{t \in T \setminus T_0} t^{\triangledown} = \bigvee_{t \in T \setminus T_0} t^{+} \leqslant \bigvee_{t \in T} t^{+} = F^{+}$$

with the equality iff $t^+ \leqslant f^{\triangledown}$ holds for all terms $t \in T \setminus T'$. ◻

DeMorgan $(\vee, \wedge, \neg)$ circuits that do not produce zero terms at all obviously impede zero terms. Such circuits were considered by several authors, starting with Kuznetsov [22] (already in 1981, under the name "circuits without null-chains"), where he proved a surprisingly large lower bound $2^{n/3}$ on the size of such circuits computing an explicit n-variate Boolean function. Sengupta and Venkateswaran [40] 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 $f^{\triangledown}$. Since non-cancellative circuits produce no zero terms, this also follows from Fact 4.

Multilinear DeMorgan $(\vee, \wedge, \neg)$ circuits already *can* produce zero terms. For example, the DeMorgan $(\vee, \wedge, \neg)$ circuit $F = (x \vee xy)(\overline{y} \vee \overline{y}z)$ computing $f = x\overline{y}$ is multilinear but produces zero terms $xy\overline{y}$ and $xy\overline{y}z$. Still, together with Lemma 6, Fact 4 implies that multilinear DeMorgan circuits impede the produced zero terms as well.

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

Theorems 1 to 4 show that read-1 $(\vee, \wedge)$ circuits are not weaker than monotone arithmetic constant-free $(+, \times)$ circuits, not weaker than tropical $(\min, +)$ circuits, and 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. For this, let $n = m^2$ and consider the following monotone Boolean function of n variables whose inputs are Boolean $m \times m$ matrices $x = (x_{i,j})$:

$$\mathrm{Lines}_{n,2}(x) = 1 \text{ iff every line of } x \text{ has at least one 1,}$$

where *lines* are rows and columns; hence, there are $2m$ lines. Examples of lowest ones $a \in A_f$ of $f = \mathrm{Lines}_{n,2}$ in the case $n = 9$ is a permutation matrix $a = \left(\begin{smallmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{smallmatrix}\right)$ but also matrices like $a = \left(\begin{smallmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{smallmatrix}\right)$. The function $\mathrm{Lines}_{n,2}$ is a special (2-dimensional) version of so-called "blocking lines" functions described in Appendix A.

The *dual* of a Boolean function $f(x_1, \ldots, x_n)$ is[10] $f^*(x_1, \ldots, x_n) := \neg f(\overline{x}_1, \ldots, \overline{x}_n)$. That is, we negate the input bits as well as the obtained value. For example, by using DeMorgan rules $\neg(x \vee y) = \neg x \wedge \neg y$ and $\neg(x \wedge y) = \neg x \vee \neg y$, we obtain that the dual of $f(x) = \bigwedge_{S \in \mathcal{F}} \big(\bigvee_{i \in S} x_i\big)$ is $f^*(x) = \bigvee_{S \in \mathcal{F}} \big(\bigwedge_{i \in S} x_i\big)$. That is, $f(x) = 1$ iff every set $S \in \mathcal{F}$ contains an $i \in S$ with $x_i = 1$, while $f^*(x) = 1$ iff there is a set $S \in \mathcal{F}$ with $x_i = 1$ for all $i \in S$.

Recall that $\mathsf{B}_k^*(f)$ denotes the minimum size of a monotone *syntactically* read-k $(\vee, \wedge)$ circuit computing a monotone Boolean function f. In particular, $\mathsf{B}_k(f) \leqslant \mathsf{B}_k^*(f)$ always holds.

Lemma 8. *For the function $f = \mathrm{Lines}_{n,2}$, we have $\mathsf{B}_{\mathrm{lin}}(f) \geqslant \mathsf{Min}_1(A_f) = \mathsf{B}_1(f) = 2^{\Omega(\sqrt{n})}$ but $\mathsf{Min}_2(A_f) \leqslant \mathsf{B}_2(f) \leqslant \mathsf{B}_2^*(f) \leqslant 2n$ and $\mathsf{B}_1(f^*) \leqslant 2n$.*

Proof. Let $A := A_f$ be the set of the lowest ones of the function $f = \mathrm{Lines}_{n,2}$. The equality $\mathsf{Min}_1(A) = \mathsf{B}_1(f)$ and the inequality $\mathsf{Min}_2(A) \leqslant \mathsf{B}_2(f)$ are given by Theorem 2, while the inequality $\mathsf{B}_{\mathrm{lin}}(f) \geqslant \mathsf{B}_1(f)$ is given by Theorem 3.

To show the upper bound $\mathsf{B}_1(f^*) \leqslant 2m^2 = 2n$ for the dual f^* of the function $f = \mathrm{Lines}_{n,2}$, note that, for every input matrix x, we have $f^*(x) = 1$ iff x has only 1s on at least one line of x. In particular, prime implicants of f^* are the ANDs of variables corresponding to the $2m$ lines in the matrix x. Hence, to obtain a read-1 circuit of size $\leqslant 2m^2 = 2n$ for f^* it is enough to take the OR of these ANDs.

To show the lower bound $\mathsf{B}_1(f) = 2^{\Omega(\sqrt{n})}$, recall that $m \times m$ matrices $a \in f^{-1}(1)$ accepted by f must have at least one 1 in each line (row or column). None of such matrices can have fewer than m 1s, because otherwise, 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 m, 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 A is the set A_g of the lowest ones of the perfect matching function $g = \mathrm{Match}_m$, and we already know that $\mathsf{Arith}(A_g) = 2^{\Omega(m)}$ holds (Example 2). Together with Theorem 1, this yields $\mathsf{B}_1(f) \geqslant \mathsf{Arith}(\lfloor A \rfloor) = \mathsf{Arith}(A_g) = 2^{\Omega(m)}$.

To show the upper bound $\mathsf{B}_2^*(f) \leqslant 2m^2 = 2n$, observe that f can be computed by a trivial $(\vee, \wedge)$ circuit

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

[10] As before, for variables x_i, we write $\overline{x}_i$ instead of $\neg x_i$.

of size at most $2m^2$. That is, we first compute the $2m$ ORs of variables along each line, and take the AND of these values. The arithmetic $(+, \times)$ version of F produces the polynomial

$$P(x) = \prod_{i=1}^{m} \Big(\sum_{j=1}^{m} x_{i,j} \Big) \cdot \prod_{j=1}^{m} \Big(\sum_{i=1}^{m} x_{i,j} \Big) .$$

Since no variable $x_{i,j}$ appears in this polynomial with a degree larger than 2, the circuit F is a syntactically read-2 circuit, as desired. ∎

9. Concluding Remarks and Open Problems

We have shown that already very restricted monotone Boolean $(\vee, \wedge)$ circuits (read-1 circuits) capture the power of three different types of circuits: monotone *arithmetic* $(+, \times)$ circuits, *tropical* $(\min, +)$ circuits, and non-monotone Boolean *multilinear* $(\vee, \wedge, \neg)$ circuits. The next natural problem is to understand the power of read-k $(\vee, \wedge)$ circuits for $k \geqslant 2$, with $k = 2$ being the first nontrivial case.

It is clear that $\mathsf{B}_1(f) \geqslant \mathsf{B}_2(f) \geqslant \ldots \geqslant \mathsf{B}_k(f) \geqslant \ldots, \geqslant \mathsf{B}(f)$ holds for any monotone Boolean function f, where $\mathsf{B}(f)$ is the minimum size of a monotone $(\vee, \wedge)$ circuit computing f. Super-polynomial lower bounds on $\mathsf{B}(f)$ can be proved using the celebrated "Method of Approximations" invented by Razborov [34, 35, 36]. However, this method, as well as its later "symmetric" versions, can be only applied to Boolean functions with a very special combinatorial property: *both* minterms and maxterms[11] must be highly "dispersed" (not too many of them can share a given number of variables in common). For example, already the application in [35] of the Method of Approximations to prove the lower bound $\mathsf{B}(f) = n^{\Omega(\log n)}$ for the perfect matching function $f = \mathrm{Match}_n$ (which we considered in Section 5.1) is rather nontrivial, going deeply into the structure of maxterms of this particular function: unlike the minterms, the maxterms of Match_n are dispersed not highly enough (see, e.g., [15, Chapter 9] for more information).

In a sharp contrast, lower bounds on the size of monotone read-1 circuits can be obtained *without* using the Method of Approximations: as demonstrated in Section 5.1, an exponential lower bound $\mathsf{B}_1(f) = 2^{\Omega(n)}$ for $f = \mathrm{Match}_n$ can be proved using a relatively simple argument: we had only to consider the minterms of f. But what about read-k circuits for larger values of k? In particular, what about read-2 circuits?

Problem 1. *Can super-polynomial lower bounds on the size of read-2 or at least of syntactically read-2 circuits be proved without using the Method of Approximations?*

Monotone Boolean read-2 circuits constitute the *first* model of computation—after tropical and monotone arithmetic circuits—which can use *both* the idempotence $x \wedge x = x$ *and* the absorption $x \vee xy = x$ laws (albeit the usage of idempotence is restricted). Let us stress that only *together* these two laws can unfold their full power.

Namely, the model where absorption $x \vee xy = x$ is allowed (without any restriction), but (when producing prime implicants) idempotence $x \wedge x = x$ is not allowed, is that of read-1 circuits considered

[11] A *minterm* (resp., *maxterm*) of a monotone Boolean function f is a minimal under inclusion set of variables such that setting all these variables to 1 (resp., to 0) forces f to output 1 (resp., 0) regardless of the values given to other variables. Since the function f is monotone, every minterms intersects every maxterm. Note that prime implicants of f are ANDs of all variables in minterms; ORs of all variables in maxterms are known as prime *implicates* of f; see, for example, [7].

Underlying semiring $(R, \oplus, \odot)$	$\oplus$-idempotence $x \oplus x = x$	$\odot$-idempotence $x \odot x = x$	absorption $x \oplus (x \odot y) = x$
Arithmetic $(+, \times)$	$-$	$-$	$-$
Tropical $(\min, +)$	$+$	$-/+$	$+$
Read-1 $(\vee, \wedge)$	$+$	$-/+$	$+$
Multilinear $(\vee, \wedge)$	$+$	$-/+$	$+$
Tight $(\vee, \wedge)$	$+$	$+$	$-$
Unrestricted $(\vee, \wedge)$	$+$	$+$	$+$

Table 1: Laws allowed $(+)$ or forbidden $(-)$ in various models of circuits. In read-1 and multilinear $(\vee, \wedge)$ circuits, as well as in tropical $(\min, +)$ circuits the usage of "multiplicative" idempotence is only *partially* forbidden $(-/+)$, because the usage of the absorption law $x \vee xy = x$ or, respectively, $\min\{x, x+y\} = x$ in these circuits is unrestricted; hence, the produced "redundant" terms can be eliminated using these laws.

in this paper. We have seen that exponential lower bounds for such circuits *can* be relatively easily proved without using the Method of Approximations (Example 2).

On the other hand, the model where idempotence $x \wedge x = x$ is allowed (without any restriction), but absorption $x \vee xy = x$ is not allowed, is that of so-called "tight" $(\vee, \wedge)$ circuits. A monotone $(\vee, \wedge)$ circuit F computing a Boolean function f is *tight* if the set $B_F \subseteq \mathbb{N}^n$ of exponent vectors of the formal $(+, \times)$ polynomial of F satisfies the equality $\mathrm{Sup}(B_F) = \mathrm{Sup}(A_f)$, not only the inclusions $\mathrm{Sup}(A_f) \subseteq \mathrm{Sup}(B_F)$ and $B_F \subseteq (A_f)^{\uparrow}$ as given by Lemma 1. That is, the circuit F is tight if *every* monomial of the formal $(+, \times)$ polynomial of F is a shadow of some prime implicant of f (see Table 1 for a schematic comparison of various types of circuits). Thus, tight $(\vee, \wedge)$ circuits cannot use the absorption law $x \vee xy = x$, but (unlike in read-k circuits) the usage of idempotence law $x \wedge x = x$ is not restricted (degrees of variables in the formal polynomial can be arbitrarily large). Note that the read-2 $(\vee, \wedge)$ circuit used in the proof of Lemma 8 to compute the function $\mathrm{Lines}_{n,2}$ is tight. This shows that tight circuits of degree already 2 can be exponentially smaller than (not necessarily tight) read-1 circuits.

Still, despite their alleged power, lower bounds for tight $(\vee, \wedge)$ circuits (of arbitrary high degree) *can* be proved without using the Method of Approximations. This was demonstrated in [16, Theorem 2], where a lower bound $2^{\Omega(n)}$ on the size of tight $(\vee, \wedge)$ circuits computing the perfect matching function Match_n is shown using a fairly simple argument similar to that we used in Section 5.1 for read-1 circuits. The point is that, because of the absence of the absorption $x \vee xy = x$, the complexity of the function Match_n is also predetermined by the minterms of that function *alone*. However, this argument fails if the absorption law is allowed.

In the case of read-1 circuits, we were able (in Theorem 1) to eliminate the influence of absorption $x \vee xy = x$ by considering lower envelopes. But already in read-2 circuits, absorption can (at least potentially) show its power. So, a solution of Problem 1 could probably shed some light on where the power of multiplicative idempotence $x \wedge x = x$ *in combination with* absorption $x \vee xy = x$ comes from.

The next natural question is: can larger allowed "degree k of idempotence" always substantially decrease the size of read-k circuits? Lemma 8 shows that, for $k = 1$, the gap $\mathsf{B}_k(f)/\mathsf{B}_{k+1}(f)$ can be exponential. But what about larger values of k?

Problem 2 (Degree hierarchy). *Can the gap $\mathsf{B}_k(f)/\mathsf{B}_{k+1}(f)$ or at least the gap $\mathsf{B}_r(f)/\mathsf{B}_k(f)$ be super-*

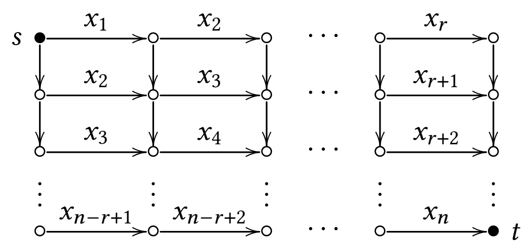

Figure 1: A monotone read-1 branching program computing the threshold-r function $\mathrm{Th}_r^n(x) = 1$ iff $x_1 + x_2 + \cdots + x_n \geqslant r$. Unlabeled edges are rectifiers (are labeled by constant 1). This BP is even a *syntactically* read-1 BP, and has $r(n-r+1)$ switches. On the other hand, Markov [27] has shown that *every* monotone BP for Th_r^n must have at least this number $r(n-r+1)$ of switches. Thus, at least for $k = 1$, monotone read-k BPs *can* be optimal among all monotone BPs.

polynomial for $k \geqslant 2$ and r not "much" smaller than k?

To show such a gap, we need a function f for which $\mathsf{B}_r(f)$ is "large" but $\mathsf{B}_k(f)$ is "small." Hence, Problem 2 *cannot* be solved using the Method of Approximations because any lower bound on $\mathsf{B}_r(f)$ obtained using this method holds for *every* r.

Yet another natural question is whether the gaps between the read-k $(\vee, \wedge)$ circuit complexities of Boolean functions f and their *duals* f^* can be large. Lemma 8 shows that, at least for $k = 1$, the gap $\mathsf{B}_k(f)/\mathsf{B}_k(f^*)$ *can* be large.

Problem 3 (Duals). *Can the gap $\mathsf{B}_k(f)/\mathsf{B}_k(f^*)$ be super-polynomial for all $k \geqslant 2$? In particular, can it be such for $k = 2$?*

Note that also this question *cannot* be answered using the Method of Approximations because $\mathsf{B}(f^*) = \mathsf{B}(f)$ always holds: given a $(\vee, \wedge)$ circuit for f, we can obtain a $(\vee, \wedge)$ circuit of the same size for the dual function f^* by just interchanging AND and OR gates.

As mentioned in Section 4, the model of "read-k circuits" is by analogy with the well-known computation model of "read-k times branching programs." So, let us briefly recall this latter model. A (nondeterministic) *branching program* (BP), also known as a *switching-and-rectifier network*, is a directed acyclic graph, each edge of which is either a *switch* (is labeled by either a variable x_i or by a negated variable $\overline{x}_i$) or is a *rectifier* (is labeled by constant 1). There is one node s of zero indegree and one node t of zero outdegree. The term *defined* by an s-t path is the AND of labels of its edges. The Boolean function *computed* by a branching program is the OR of terms defined by all s-t paths. The *size* of such a program is the total number of switches. A variable x_i is *read* along a path if x_i or $\overline{x}_i$ appears as a label of some edge along that path. A branching program is a *syntactically read-k* program if no variable is read more than k times along *any* s-t path ("syntactically" because the restriction is on *all* s-t paths).

In *monotone* branching programs, none of the edges is labeled by a negated variable $\overline{x}_i$. By Lemma 1, a monotone branching program F computes a monotone Boolean function f iff the term defined by any s-t path is an implicant of f, and for every prime implicant p of f there is an s-t path in F (a shadow path of p) along which only the variables of p are read. The program F is a *semantically read-k* BP if every prime implicant of f has *at least one* shadow path along which no variable is read more than k times (see Figure 1 for an example). Thus, monotone semantically read-k branching programs correspond to read-k $(\vee, \wedge)$ circuits considered in this paper: the restriction is only on shadow s-t paths: there are no restrictions on the remaining s-t paths.

For a monotone Boolean function f, let $\mathsf{BP}_k(f)$ denote the minimum number of switches in a monotone semantically read-k branching program computing f. Due to the sequential nature of computation in branching programs (rather than parallel nature, as in the case of circuits), their structure could be easier to analyze.

Problem 4 (Read-k branching programs). *Problems 1 to 3 but for monotone semantically read-k branching programs instead of circuits, that is, for the measure* $\mathsf{BP}_k(f)$ *instead of* $\mathsf{B}_k(f)$.

As possible candidates for separating functions f in Problems 2 to 4, one could try so-called "blocking lines" functions, including the functions $\mathrm{Lines}_{n,k}$ and $\mathrm{Cov}_{n,k}$ described in Examples 3 and 4 of Appendix A (n stands for the number of variables of these functions). Each of these functions can be computed by a monotone syntactically read-k $(\vee, \wedge)$ circuit with $\leqslant kn$ gates, as well as by a monotone syntactically read-k branching program with $\leqslant kn$ switches.

That the blocking lines function $\mathrm{Lines}_{n,2}$ exhibits a large read-1/read-2 gap is shown by Lemma 8. That blocking lines functions $\mathrm{Cov}_{n,k}$ *can* exhibit large gaps for read-k branching programs even for larger parameters k was shown by Okolnishnikova [29]. Namely, she has proved that if $k \geqslant 4$ is a constant and $1 \leqslant r \leqslant \sqrt{k}$, then every (even not monotone but) *syntactically* read-r branching program computing $\mathrm{Cov}_{n,k}$ must have an exponential (in n) number of switches. Using different (*non*-monotone) functions, Thathachar [44] has proved such a gap even for $r = k - 1$. But, to my best knowledge, no similar gaps are known for monotone but *semantically* read-k branching programs (where the read-k restriction is only on shadow s-t paths).

Appendix A. Blocking lines functions

The Boolean function $\mathrm{Lines}_{n,2}$ we used in the proof of Lemma 8 is just a very special case of the following more general construction of Boolean functions that have small read-k circuits but could (apparently) require large read-r circuits for $r < k$. That these functions sometimes indeed *can* exhibit such gaps is shown by Lemma 8, as well as by the aforementioned result of Okolnishnikova [29].

Let $\mathcal{L} \subseteq 2^P$ be a family of subsets of a finite set P; let us call elements $p \in P$ *points*, and sets $L \in \mathcal{L}$ *lines*. Suppose that the family $\mathcal{L}$ is m-*uniform* (each line has exactly m points), and is k-*regular* (each point belongs to exactly k lines). By double-counting, we have $m|\mathcal{L}| = k|P|$. The *blocking lines function* $f_{\mathcal{L}}$ has $n = |P|$ variables x_p, one for each point $p \in P$ and, for every input $x \in \{0,1\}^P$, $f_{\mathcal{L}}(x) = 1$ iff the set of points $S_x = \{p \in P \colon x_p = 1\}$ blocks (intersects) every line $L \in \mathcal{L}$. The monotone circuit $F_{\mathcal{L}}(x) = \bigwedge_{L\in\mathcal{L}} \left(\bigvee_{p\in L} x_p\right)$ computes $f_{\mathcal{L}}$ and has $(m-1)|\mathcal{L}| + (|\mathcal{L}| - 1) = m|\mathcal{L}| - 1$ fanin-2 gates. Moreover, since no point belongs to more than k lines, $F_{\mathcal{L}}$ is a read-k circuit. In particular, we have an upper bound $\mathsf{B}_k(f_{\mathcal{L}}) \leqslant m|\mathcal{L}| = k|P| = kn$. Note that for the dual $f^*_{\mathcal{L}}(x) = \bigvee_{L\in\mathcal{L}} \bigwedge_{p\in L} x_p$ of $f_{\mathcal{L}}$, we even have $\mathsf{B}_1(f^*_{\mathcal{L}}) \leqslant kn$, that is, the duals of blocking lines functions can be computed by small *read*-1 circuits. Since none of the points belongs to more than k lines, every implicant $t_S = \bigwedge_{p\in S} x_p$ of $f_{\mathcal{L}}$ corresponds to a blocking set $S \subseteq P$ consisting of $|S| \geqslant |\mathcal{L}|/k$ points, while shortest (prime) implicants t_S of $f_{\mathcal{L}}$ (those with the smallest number of variables) correspond to *smallest* blocking sets, that is, blocking sets $S \subseteq P$ consisting of $|S| = |\mathcal{L}|/k$ pairwise noncollinear points; two points are *collinear* if they both belong to some line. Note that maxterms of $f_{\mathcal{L}}$ are sets $\{x_p \colon p \in L\}$ of variables corresponding to lines $L \in \mathcal{L}$, while minterms of $f_{\mathcal{L}}$ are sets $\{x_p \colon p \in S\}$ of variables corresponding to blocking sets $S \subseteq P$, none proper subset of which is a blocking set.

Each point of a smallest blocking set $S \subseteq P$ blocks (intersects) its *own* collection of k lines: for every point $p \in S$ there is a collection $\mathcal{L}_p \subseteq \mathcal{L}$ of $|\mathcal{L}_p| = k$ lines such that $\mathcal{L}_p \cap \mathcal{L}_q = \emptyset$ holds for all points $p \neq q \in S$ (for, otherwise, points p and q would be collinear). That is, each point $p \in S$ is the *only* point of S blocking the k lines $\mathcal{L}_p$. In other words, each variable x_p of a shortest prime implicant $t_S = \bigwedge_{p\in S} x_p$ of $f_{\mathcal{L}}$ blocks its *own* collection of k lines. Intuitively, this means that the variable x_p "should" be accessed by a circuit or a branching program at least k times to produce this implicant.

Example 3 (Many short lines). Instead of 2-dimensional tensors (matrices), as in the case of the function $\mathrm{Lines}_{n,2}$ used in Lemma 8, one can consider k-dimensional tensors for $k \geqslant 3$. Let n be of the form $n = m^k$. As the underlying set P of points, take the set $P = [m]^k$ of k-tuples $p = (p_1, \ldots, p_k) \in [m]^k$, and consider the family $\mathcal{L} \subseteq 2^P$ of all (combinatorial) lines, where the *line* in the ith direction through a point $p \in P$ is the set $L = \big\{(p_1, \ldots, p_{i-1}, *, p_{i+1}, \ldots, p_k) \colon * = 1, \ldots, n\big\}$ of $|L| = m$ points. Thus, we have $|\mathcal{L}| = km^{k-1} = (k/m)|P|$ distinct lines, no two sharing more than one point. Since each point belongs to exactly k lines (there are k possible positions for $*$), the family $\mathcal{L}$ is k-regular. Hence, the corresponding (to this family $\mathcal{L}$) blocking lines function $\mathrm{Lines}_{n,k}(x) := f_{\mathcal{L}}(x)$ can be computed by a monotone read-k $(\vee, \wedge)$ circuit of size $\leqslant m|\mathcal{L}| = k|P| = km^k = kn$; in particular, $\mathsf{B}_k(\mathrm{Lines}_{n,k}) \leqslant kn$ holds. Note that in this family $\mathcal{L}$ of lines, two points (k-tuples) $p \neq q \in [m]^k$ are collinear iff they differ in exactly one position. In particular, any set of *pairwise* collinear points must entirely lie in one line. In the case $k = 2$ (matrices, as in Lemma 8), smallest blocking sets consist of the entries of a permutation matrix. □

Example 4 (Few long lines). Let n be of the form $n = \binom{m}{k}$, were m is divisible by k. As the underlying set P of points, take the collection $P = \binom{[m]}{k}$ of all $|P| = n$ k-element sets $p \subseteq [m] := \{1, \ldots, m\}$. For $i \in [m]$, let the line in the ith direction be the set $L_i = \{p \in P \colon i \in p\}$ of $|L_i| = \binom{m-1}{k-1}$ points containing i. Let $\mathcal{L} = \{L_1, \ldots, L_m\}$ be the family of all m lines (in all m directions). Since every point $p \in P$ consists of k *distinct* elements of $[m]$, each of them belongs to exactly k lines; hence, the family $\mathcal{L}$ is k-regular For the corresponding (to this family $\mathcal{L}$) blocking lines function $\mathrm{Cov}_{n,k}(x) := f_{\mathcal{L}}(x)$, we have an upper bound $\mathsf{B}_k(\mathrm{Cov}_{n,k}) \leqslant m|\mathcal{L}| = m\binom{m-1}{k-1} = k\binom{m}{k} = k|P| = kn$. In this family $\mathcal{L}$ of lines, two points $p \neq q \in \binom{[m]}{k}$ are collinear iff $p \cap q \neq \emptyset$, and a set $S \subseteq \binom{[m]}{k}$ of points (k-element subsets of $[m]$) is a blocking set iff the union of these subsets is the entire set $[m]$. Hence, smallest blocking sets $S \subseteq P$ consist of $|S| = m/k$ points forming a partition of $[m]$ into m/k disjoint blocks of size k. In the case $k = 2$, points $p \in P$ correspond to the edges $p = \{i, j\}$ of the complete graph K_m on $[m]$, and the line L_i in the ith direction is the set of all $m - 1$ edges incident with vertex i. Then $\mathrm{Cov}_{n,2}(x) = 1$ iff the subgraph G_x of K_m specified by x has no isolated vertices. Smallest blocking sets in this case are perfect matchings in K_m. □

Acknowledgments. I am thankful to both referees for very useful comments and suggestions.

References

[1] N. Alon and R. Boppana. The monotone circuit complexity of Boolean functions. *Combinatorica*, 7(1):1–22, 1987.

[2] N. Alon and M. Tarsi. Colorings and orientations of graphs. *Combinatorica*, 12:125–134, 1992.

[3] W. Baur and V. Strassen. The complexity of partial derivatives. *Theoret. Comput. Sci.*, 22:317–330, 1983.

[4] N. Blum. A Boolean function requiring 3n network size. *Theoret. Comput. Sci.*, 28:337–345, 1984.

[5] P. Bügisser, M. Clause, and A. Shokrollahi. *Algebraic Complexity Theory*. Springer, 1997.

[6] Xi Chen, N. Kayal, and A. Wigderson. Partial derivatives in arithmetic complexity and beyond. *Found. Trends Theor. Comput. Sci.*, 6(1-2):1–138, 2011.

[7] Y. Crama and P. L. Hammer, editors. *Boolean Functions: Theory, Algorithms, and Applications*, volume 142 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Pess, 2011.

[8] S. B. Gashkov. On one method of obtaining lower bounds on the monotone complexity of polynomials. *Vestnik MGU, Series 1 Mathematics, Mechanics*, 5:7–13, 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*, 203(10):1411–1147, 2012.

[10] D. Grigoriev and G. A. Koshevoy. Complexity of tropical Schur polynomials. *J. Symb. Comput.*, 74:46–54, 2016.

[11] D. Grigoriev and V. V. Podolskii. Tropical combinatorial Nullstellensatz and sparse polynomials. *Found. Comput. Math.*, 20(4):753–781, 2020.

[12] L. Hyafil. On the parallel evaluation of multivariate polynomials. *SIAM J. Comput.*, 8(2):120–123, 1979.

[13] K. Iwama and H. Morizumi. An explicit lower bound of 5n - o(n) for boolean circuits. In *Proc. of 27th Int. Symp. MFCS*, volume 2420 of *Lect. Notes in Comp. Sci.*, pages 353–364. Springer, 2002.

[14] M. Jerrum and M. Snir. Some exact complexity results for straight-line computations over semirings. *J. ACM*, 29(3):874–897, 1982.

[15] S. Jukna. *Boolean Function Complexity: Advances and Frontiers.* Springer-Verlag, 2012.

[16] S. Jukna. Lower bounds for monotone counting circuits. *Discrete Appl. Math.*, 213(139–152), 2016.

[17] S. Jukna. Tropical complexity, Sidon sets and dynamic programming. *SIAM J. Discrete Math.*, 30(4):2064–2085, 2016.

[18] S. Jukna and H. Seiwert. Approximation limitations of pure dynamic programming. *SIAM J. Comput.*, 49(1):170–207, 2020.

[19] L. R. Kerr. *The effect of algebraic structure on the computation complexity of matrix multiplications.* PhD thesis, Cornell Univ., Ithaca, N.Y., 1970.

[20] J. Kollár, L. Rónyai, and T. Szabó. Norm-graphs and bipartite Turán numbers. *Combinatorica*, 16(3):399–406, 1996.

[21] M. P. Krieger. On the incompressibility of monotone DNFs. *Theory of Comput. Syst.*, 41(2):211–231, 2007.

[22] S. E. Kuznetzov. Circuits composed of functional elements without zero paths in the basis $\{\&, \vee, -\}$. *Izv. Vyssh. Uchebn. Zaved. Mat.*, 228(5):56–63, 1981. In Russian.

[23] O. Lachish and R. Raz. Explicit lower bound of 4.5n-o(n) for boolean circuits. In *Proc. of 33rd Ann. ACM Symp. on Theory of Computing, STOC*, pages 399–408. ACM, 2001.

[24] A. Lingas. A note on lower bounds for monotone multilinear Boolean circuits. Technical report, ECCC TR-22-85, 2022.

[25] M. Mahajan, P. Nimbhorkar, and A. Tawari. Computing the maximum using (min,+) formulas. In *42nd Int. Symp. on Math. Foundations of Comput. Sci. (MFCS 2017)*, volume 83 of *Leibniz Int. Proc. in Informatics*, pages 74:1–74:11, 2017.

[26] M. Mahajan, P. Nimbhorkar, and A. Tawari. Shortest path length with bounded-alternation (min,+) formulas. *Int. J. of Advances in Engineering Sci. and Applied Math.*, 11(1):68–74, 2019.

[27] A. A. Markov. Minimal relay-diode bipoles for monotonic symmetric functions. *Problemy Kibernetiki*, 8:117–121, 1962. English transl. in *Problems of Cybernetics* 8 (1964), 205–212.

[28] N. Nisan and A. Wigderson. Lower bounds on arithmetic circuits via partial derivatives. *Comput. Complexity*, 6(3):217–234, 1997.

[29] E. A. Okolnishnikova. Comparing the sizes of nondeterministic branching read-k-times programs. *Discret. Appl. Math.*, 135(1-3):205–222, 2004. English translation of the paper: "On comparing the complexities of binary k-programs," Discrete Anal. Oper. Res. 2 :4 (1995) 54-73 (in Russian).

[30] 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*, volume 3328 of *Lect. Notes in Comput. Sci.*, pages 460–468. Springer, 2004.

[31] R. Raz. Separation of multilinear circuit and formula size. *Theory of Comput.*, 2(6):121–135, 2006.

[32] R. Raz. Multi-linear formulas for Permanent and Determinant are of super-polynomial size. *J. ACM*, 56(2):1–17, 2009.

[33] R. Raz and A. Yehudayoff. Multilinear formulas, maximal-partition discrepancy and mixed-sources extractors. *J. Comput. Syst. Sci.*, 77(1):167–190, 2011.

[34] A. A. Razborov. Lower bounds for the monotone complexity of some boolean functions. *Soviet Math. Dokl.*, 31:354–357, 1985.

[35] A. A. Razborov. Lower bounds on monotone complexity of the logical permanent. *Math. Notes of the Acad. of Sci. of the USSR*, 37(6):485–493, 1985.

[36] A. A. Razborov. On the method of approximations. In *Proc. of 21st Ann. ACM Symp. on Theory of Computing, STOC*, pages 167–176. ACM, 1989.

[37] C. Schnorr. Zwei lineare untere Schranken für die Komplexität Boolescher Funktionen. *Computing*, 1974.

[38] C. P. Schnorr. A lower bound on the number of additions in monotone computations. *Theor. Comput. Sci.*, 2(3):305–315, 1976.

[39] R. Sengupta and H. Venkateswaran. Multilinearity can be exponentially restrictive (preliminary version). Technical Report GIT-CC-94-40, Georgia Institute of Technology. College of Computing, 1994.

[40] R. Sengupta and H. Venkateswaran. Non-cancellative boolean circuits: a generalization of monotone boolean circuits. *Theor. Comput. Sci.*, 237:197–212, 2000.

[41] E. Shamir and M. Snir. On the depth complexity of formulas. *Math. Syst. Theory*, 13:301–322, 1980.

[42] A. Shpilka and A. Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010.

[43] V. Strassen. Vermeidung von Divisionen. *J. Reine Angew. Math.*, 264:184–202, 1973.

[44] J. S. Thathachar. On separating the read-k-times branching program hierarchy. In *Proc. of 30th Ann. ACM Symp. on the Theory of Computing, STOC*, pages 653–662. ACM, 1998.

[45] J. Tiekenheinrich. A $4n$-lower bound on the mononotonenetwork complexity of a one-output boolean function. *Inf. Process. Letters*, 18:201–202, 1984.

[46] P. Tiwari and M. Tompa. A direct version of Shamir and Snir's lower bounds on monotone circuit depth. *Inf. Process. Lett.*, 49(5):243–248, 1994.

[47] L. G. Valiant. Negation can be exponentially powerful. *Theor. Comput. Sci.*, 12:303–314, 1980.

[48] I. Wegener. *The complexity of Boolean functions.* Wiley-Teubner, 1987.

[49] A. Yehudayoff. Separating monotone VP and VNP. In *Proc. of 51st Ann. ACM SIGACT Symp. on Theory of Computing, STOC*, pages 425–429. ACM, 2019.