Electronic Colloquium on Computational Complexity, Report No. 99 (2015)

Satisfiability Algorithms and Lower Bounds for Boolean Formulas over Finite Bases

Ruiwen Chen

School of Informatics, University of Edinburgh, Edinburgh, UK;
rchen2@inf.ed.ac.uk

Abstract. We give a #SAT algorithm for boolean formulas over arbitrary finite bases. Let B_k be the basis composed of all boolean functions on at most k inputs. For B_k-formulas on n inputs of size cn, our algorithm runs in time $2^{n(1-\delta_{c,k})}$ for $\delta_{c,k} = c^{-O(c^2k2^k)}$. We also show the average-case hardness of computing affine extractors using linear-size B_k-formulas.
We also give improved algorithms and lower bounds for formulas over finite unate bases, i.e., bases of functions which are monotone increasing or decreasing in each of the input variables.

Keywords: boolean formula, satisfiability algorithm, lower bound, random restriction

1 Introduction

The random restriction approach was introduced by Subbotovskaya [15] to prove lower bounds for boolean formulas. For formulas over a basis B, we define the *shrinkage exponent* Γ_B to be the least upper bound on γ such that the formula size shrinks (in expection) by a factor of p^γ under random assignments leaving p fraction of the inputs unfixed. Subbotovskaya [15] showed the shrinkage exponent of de Morgan formulas (formulas over the binary basis $\{\neg, \wedge, \vee\}$) is at least 1.5, and this implies an $\Omega(n^{1.5})$ lower bound on the formula size for computing the parity of n variables. Andreev [1] improved this lower bound by constructing an explicit function which requires size $\Omega(n^{\Gamma+1-o(1)})$ for any formulas with shrinkage exponent Γ. The shrinkage exponent of de Morgan formulas was improved in [8, 12], and finally, Hastad [6] showed the tight bound $\Gamma = 2 - o(1)$, which gives an $\Omega(n^{3-o(1)})$ lower bound for computing Andreev's function.

Recently, the shrinkage property was strengthened to get both satisfiability algorithms [13] and average-case lower bounds [9, 10]. Santhanam [13] gave a simple deterministic #SAT algorithm for de Morgan formulas. The algorithm recursively restricts the most frequent variables, and the number of branches in the recursion tree is bounded via the *concentrated shrinkage* property; that is, along a random branch of the recursion tree, the formula size shrinks by a factor of $p^{1.5}$ with high probability. For cn-size formulas, Santhanam's algorithm runs in time $2^{n(1-\Omega(1/c^2))}$. Combining with memoization, similar algorithms running in time $2^{n-n^{\Omega(1)}}$ were given for formula size $n^{2.49}$ [2] and $n^{2.63}$ [3].

ISSN 1433-8092

Seto and Tamaki [14] extended Santhanam's algorithm to formulas over the full binary basis $\{\neg, \wedge, \vee, \oplus\}$. Note that, the shrinkage exponent here is trivially 1 since $\oplus$ is in the basis, and thus Santhanam's algorithm does not apply directly. Instead, Seto and Tamaki [14] showed that, for a small-size formula over the full binary basis, either satisfiability checking is easy by solving systems of linear equations, or a greedy restriction will shrink the formula non-trivially, which is as required by Santhanam's algorithm. In this work, we will further extend Seto and Tamaki's approach to formulas over arbitrary finite bases.

On the other hand, Komargodski, Raz, and Tal [9, 10] applied concentrated shrinkage to prove average-case lower bounds for de Morgan formulas. In particular, they constructed a generalized Andreev's function which is computable in polynomial time, but de Morgan formulas of size $n^{2.99}$ can compute correctly on at most $1/2 + 2^{-n^{\Omega(1)}}$ fraction of all inputs. The result in [10] also implies a randomized #SAT algorithm in time $2^{n-n^{\Omega(1)}}$ for de Morgan formulas of size $n^{2.99}$.

1.1 Our results and proof techniques

In this work, we focus on formulas over arbitrary finite bases, and generalize previous results on formulas over binary bases [13, 14, 2]. For $k \geqslant 2$, let B_k be the basis consisting of all boolean functions on at most k variables. We consider B_k-formulas, i.e., formulas over the basis B_k.

We first give a satisfiability algorithm for B_k-formulas which is significantly better than brute-force search.

Theorem 1. *For n-input B_k-formulas of size cn, there is a deterministic algorithm counting the number of satisfying assignments in time $2^{n(1-\delta_{c,k})}$, where $\delta_{c,k} = c^{-O(c^2 k 2^k)}$.*

The algorithm is based on a structural property of small formulas, similar to Seto and Tamaki's approach [14] for B_2-formulas. That is, for a small B_k-formula, either satisfiability checking can be done by solving systems of linear equations, or a process of greedy restrictions gives non-trivial shrinkage. The technical difficulty is that, since we have both $\oplus$ and functions of arity larger than 2 in B_k, the formula size shrinks trivially even by restricting the most frequent variables. To get nontrivial shrinkage, we need a formula *weight* function which accounts for not only the formula size but also the basis functions used in the formula, and argue that the weight shrinks nontrivially under certain greedy restrictions (which aims at optimally reducing the weight rather than the size). We also require that the formula weight is proportional to the formula size, and thus, in the end of greedy restrictions, the size also shrinks nontrivially.

The weighting technique was used previously for the shrinkage (in expectation) of de Morgan formulas [8, 12] and formulas over finite unate bases [4]. We use weight functions similar as in [4] but with dedicated parameterizations since the basis contains non-unate functions.

The algorithm implicitly constructs a *parity decision tree* for the given formula; this implies that any B_k-formula of linear size has a parity decision tree of size $2^{n-\Omega(n)}$. By the fact that affine extractors are hard to approximate by parity decision trees, we immediately get the following average-case lower bound.

Theorem 2. *There is a polynomial-time computable function f_n such that, for any family of B_k-formulas F_n of size cn for a constant c, on random inputs $x \in \{0,1\}^n$,*

$$\mathbf{Pr}[F_n(x) = f_n(x)] \leqslant 1/2 + 2^{-\Omega(n)}.$$

Note that, an average-case lower bound for larger B_k-formulas also follow from [9, 10]. That is, B_k-formulas of size $n^{1.99}$ can compute the generalized Andreev's function of [10] correctly on at most $1/2 + 2^{-n^{\Omega(1)}}$ fraction of inputs.

For the more restrictive unate bases, we get nontrivial #SAT algorithms and average-case lower bounds for formulas of super-quadratic size. The results follow easily by extending the expected shrinkage of Chockler and Zwick [4] to concentrated shrinkage, and generalizing the previous algorithms and lower bounds for de Morgan formulas [13, 9, 10, 2].

1.2 Related work

This work, or the general task of finding better-than exhaustive search satisfiability algorithms, is largely motivated by the connection between satisfiability algorithms and circuit lower bounds [18]. Williams [16, 17] showed that nontrivial satisfiability algorithms (running in time $2^n/n^{\omega(1)}$) for various circuit classes would imply circuit lower bounds. This raises the question of which circuit classes have nontrivial satisfiability algorithms. In this work, we show such an algorithm for linear-size formulas over arbitrary finite bases. Our algorithm is also an example of the other direction in the connection (following the works [13, 14, 2, 3]); that is, proof techniques for circuit lower bounds (shrinkage under random restrictions) can be used to design nontrivial satisfiability algorithms.

Shrinkage (in expection) under restrictions was a successful technique for proving formula size lower bounds [15, 8, 12, 6, 4]. Recently, this property was generalized to concentrated shrinkage in several works in order to get nontrivial satisfiability algorithms [13, 14, 3], average-case lower bounds [9, 10], and pseudorandom generators [7].

The weighting and shrinkage technique we use, following from [8, 12, 4, 13], is also related to the measure-and-conquer approach [5], which gives improved exact algorithms for graph problems such as maximum independent set. Both approaches wish to bound the recursion branches via a better measure (weight) on problem instances. A major difference might be that, the usual measure-and-conquer approach reduces the measure additively in each recursion (linear recurrences), whereas the shrinkage approach reduces the measure by a multiplicative factor (non-linear recurrences).

2 Preliminaries

A *basis* is a collection of boolean functions. A formula over a basis B, which we call a *B-formula*, is a tree where each internal node is labeled by a function in B, and each leaf is labeled by a literal (a variable x or its negation $\overline{x}$) or a constant (0 or 1). We call each internal node a *gate*, and require the fan-in of a gate matches with the inputs of the labeling function.

For $k \geqslant 2$, let B_k be the basis composed of all boolean functions on at most k variables.

We say a function $f(x_1, \ldots, x_k)$ is *positive (negative) unate* in x_i if, for all $a_j \in \{0, 1\}$ where $j \neq i$,

$$f(a_1, \ldots, a_{i-1}, 0, a_{i+1}, \ldots, a_k) \leqslant (\geqslant) f(a_1, \ldots, a_{i-1}, 1, a_{i+1}, \ldots, a_k).$$

We say f is *unate* if it is either positive unate or negative unate in each of its input variables. For $k \geqslant 2$, let U_k be the basis consisting of all unate functions on at most k variables.

For convenience, we assume all gates have fan-in at least two; that is, all negations are eliminated by merging to their inputs. We also assume all binary gates are labeled by either $\vee, \wedge$ or $\oplus$; the other binary gates can be represented by adding necessary negations, e.g., replacing $x \equiv y$ by $x \oplus \overline{y}$. When we write $\vee, \wedge$ or $\oplus$, we assume they are binary.

We define the *size* $L(F)$ of a formula F to be the number of non-constant leaves in F. Let $G_\oplus(F)$ be the number of $\oplus$ gates in F, let $G_2(F)$ be the number of $\vee$ and $\wedge$ gates, and let $G_i(F)$ be the number of i-ary gates, for $3 \leqslant i \leqslant k$. Then obviously, for non-constant F, we have $L(F) = 1 + G_\oplus(F) + \sum_{i=2}^{k}(i-1)G_i(F)$. We define the *weight* of F to be

$$W(F) = L(F) + \alpha_\oplus G_\oplus(F) + \sum_{i=2}^{k} \alpha_i G_i(F),$$

where $\alpha_\oplus, \alpha_2, \ldots, \alpha_k$ is a sequence of positive weights associated with gates of the corresponding types. As we will explain later, we require that $\alpha_{i-1} + \alpha_\oplus \leqslant \alpha_i \leqslant (i-1)\alpha_\oplus$ for $i \geqslant 3$.

3 Shrinkage under restrictions

In this section, we will characterize the shrinkage of formula weights under random restrictions. We first present formula simplification rules which will help removing redundancy and transforming the formula into a normalized representation.

3.1 Formula simplification

For $i \geqslant 3$, we say an i-ary gate $g(x_1, \ldots, x_i)$ has a *linear representation* in one of its input variables, say x_1, if $g(x_1, \ldots, x_i) = x_1 \oplus h(x_2, \ldots, x_i)$ for some $(i-1)$-ary gate h. We will replace a gate by its linear representation whenever possible.

We recall some definitions from [14]. A node in the formula tree is called *linear* if (1) it is a leaf, or (2) it is labeled by $\oplus$ and both of its children are linear. We say a linear node is *maximal* if its parent is not linear. Let F_v denote the subformula rooted at a linear node v. Denote by $\mathrm{var}(F_v)$ the set of variables appearing in F_v.

Two maximal linear nodes u and v are *mergeable* if they are connected by a path where every node in between is labeled by $\oplus$. They can be merged into one maximal linear node in the following way. Let s be the parent of u, and u' be the sibling of u; let t be the parent of v, and v' be the sibling of v; that is, $F_s = F_u \oplus F_{u'}$ and $F_t = F_v \oplus F_{v'}$. (Note that the parents of both u and v must be $\oplus$ since otherwise they would not be mergeable.) To merge u and v, we replace F_u by $F_u \oplus F_v$, and replace F_t by $F_{v'}$.

The following are the formula simplification rules, which include the rules in [6, 13, 14] as special cases.

Formula simplification rules:

1. Constant elimination:
   (a) Eliminate constants feeding into binary gates by the following: $1 \vee x = 1,\ 1 \wedge x = x,\ 0 \wedge x = 0,\ 0 \vee x = x,\ 1 \oplus x = \overline{x},\ 0 \oplus x = x$.
   (b) For $3 \leqslant i \leqslant k$, if an i-ary gate is fed by a constant, replace it by an equivalent $(i-1)$-ary gate.
2. Redundant sibling elimination:
   (a) If an $\wedge$ or $\vee$ gate is fed by a literal x and a subformula G, eliminate x and $\overline{x}$ from G (if possible) by the following: $x \vee G = x \vee G|_{x=0},\ x \wedge G = x \wedge G|_{x=1}$.
   (b) For $3 \leqslant i \leqslant k$, if an i-ary gate does not depend on one of its input, replace it an equivalent $(i-1)$-ary gate.
   (c) If an i-ary gate is fed by two literals over the same variable, replace it by an equivalent $(i-1)$-ary gate.
3. Linear node transformation:
   (a) If an i-ary gate has a linear representation in one of its inputs, replace it by $\oplus$ over that input and a new $(i-1)$-ary gate.
   (b) If a variable appears more than once under a linear node, eliminate unnecessary leaves by the commutativity of $\oplus$ and the following: $x \oplus x = 0,\ x \oplus \overline{x} = 1$.
   (c) Merge any mergeable pairs of maximal linear nodes.

We call a formula *simplified* if none of the rules above is applicable. It is easy to check that, given a formula F, one can compute an equivalent simplified formula F' in polynomial time, and it holds that $L(F') \leqslant L(F)$.

3.2 Weight reduction under restrictions

In the following, we analyze weight reductions when a single leaf is randomly fixed. Given a simplified formula F, a variable x, and $b \in \{0, 1\}$, we define the weight reduction from $x = b$ as $\sigma_{x=b}(F) = W(F) - W(F|_{x=b})$, where $F|_{x=b}$ is the simplified formula obtained from F under the restriction $x = b$. We also define $\sigma_x(F) = (\sigma_{x=0}(F) + \sigma_{x=1}(F))/2$.

Lemma 1. *Suppose that* $\alpha_{i-1} + \alpha_\oplus \leqslant \alpha_i \leqslant (i-1)\alpha_\oplus$ *for* $i \geqslant 3$. *Let* $F = g(x, G_1, \ldots, G_{i-1})$ *be a simplified formula where the root gate* g *has arity* $i \geqslant 2$ *and its first input is a literal* x. *Let* $\beta^b = \sigma_{x=b}(F) - \sum_{j=1}^{i-1} \sigma_{x=b}(G_j)$.

- *If* g *is* $\vee$ *or* $\wedge$, *then* $\min\{\beta^0, \beta^1\} \geqslant 1 + \alpha_2$, *and* $\max\{\beta^0, \beta^1\} \geqslant 2 + \alpha_2$.
- *If* g *is* $\oplus$, *then* $\beta^b \geqslant 1 + \alpha_\oplus$ *for* $b \in \{0, 1\}$.
- *If* g *is* i*-ary for* $i \geqslant 3$, *then* $\min\{\beta^0, \beta^1\} \geqslant 1 + \alpha_i - (i-2)\alpha_\oplus$, *and* $\max\{\beta^0, \beta^1\} \geqslant 1 + \alpha_i - \alpha_{i-1}$.

Note that, β^b measures the weight reduction from restricting a single leaf.

Proof. If g is $\vee$ or $\wedge$, then x or $\overline{x}$ does not appear in G_1 by the simplification rule 2(a), which means $\sigma_{x=b}(G_1) = 0$. For both assignments of x, we can eliminate x and g, reducing the weight by $1 + \alpha_2$; in one of the assignments, we can also eliminate G_1, which has size at least 1.

If g is $\oplus$, then, for both assignments of x, we can eliminate x and g, reducing the weight by $1 + \alpha_\oplus$. This will not affect the weight reduction in G_1, since G_1 cannot be a single literal x or $\overline{x}$ by the simplification rule 3(b).

If g is an i-ary gate, for $i \geqslant 3$, then none of G_i's is a literal x or $\overline{x}$ by the simplification rule 2(c), and g does not have any linear representation by the simplification rule 3(a). We restrict $x = b$ in two steps. First restrict $x = b$ on all G_i's, and let $F' = g(x, G_1|_{x=b}, \ldots, G_{i-1}|_{x=b})$, which has weight $W(F') \leqslant W(F) - \sum_{j=1}^{i-1} \sigma_{x=b}(G_j)$; note that F' may not be simplified on the top gate g (however, each $G_i|_{x=b}$ is simplified). Then restrict $x = b$ on the first input of g, and simplify the formula. Next we compute the weight reduction $W(F') - W(F'|_{x=b})$.

This weight reduction is upper bounded by the weight reduction from restricting $x = b$ on $g(x, y_1, \ldots, y_{i-1})$ where $x, y_1, \ldots, y_{i-1}$ are distinct variables. The rest of the proof follows from the next Claim.

Claim. Let $G = g(x, y_1, \ldots, y_{i-1})$ where $x, y_1, \ldots, y_{i-1}$ are distinct variables, and g does not have any linear representation. Then $\min_{b\in\{0,1\}} \sigma_{x=b}(G) \geqslant 1 + \alpha_i - (i-2)\alpha_\oplus$, and $\max_{b\in\{0,1\}} \sigma_{x=b}(G) \geqslant 1 + \alpha_i - \alpha_{i-1}$.

Proof. We consider how g simplifies when x is fixed. Let $g_b := g(b, y_1, \ldots, y_{i-1})$ be an $(i-1)$-ary gate on inputs $y_1, \ldots, y_{i-1}$. Note that g_b could have linear representations. In general, each of g_0 and g_1 could be an $(i-1)$-ary gate, or eventually replaced by an $(i-1-j)$-ary gate together with j of $\oplus$ gates, for $1 \leqslant j \leqslant i-2$; when $j = i-2$, this is just $i-2$ of $\oplus$ gates.

However, we argue that, g_0 and g_1 cannot have linear representations in the same input. For the sake of contradiction, suppose both g_0 and g_1 have linear representations in the same input y, then we get $g_0 = y \oplus h_0$ and $g_1 = y \oplus h_1$, for some $(i-2)$-ary gates h_0 and h_1. But this implies $g = y \oplus ((\overline{x} \wedge h_0) \vee (x \wedge h_1))$, which contradicts the assumption that g does not have any linear representation.

Suppose g_0 does not have any linear representation, then the smallest weight reduction for $x = 1$ is obtained when g_1 is replaced by $i-2$ of $\oplus$ gates. The weight reduction is $1 + \alpha_i - \alpha_{i-1}$ for $x = 0$, and $1 + \alpha_i - (i-2)\alpha_\oplus$ for $x = 1$.

If g_0 is replaced by $(i-2)$ of $\oplus$ gates, then g_1 cannot have any linear representation; this is the same as the previous case.

If g_0 is replaced by an $(i-1-j)$-ary gate together with j of $\oplus$ gates, for $1 \leqslant j < i-2$, then g_1 cannot be completely replaced by $i-2$ of $\oplus$ gates. For both $x=0$ and $x=1$, the weight reduction is at least $1+\alpha_i-\alpha_{i-1}$, since $\alpha_{i-1-j}+j\alpha_\oplus \leqslant \alpha_{i-1}$. □

□

3.3 Upper bounds on formula weights

Random restrictions will only affect gates fed by leaves, but the weight function is defined by attaching weights to all gates of the formula. In order to characterize the weight reduction in terms of the total weight, we give an upper bound on the total weight expressed by the numbers of leaves feeding into different types of gates.

Let F be a simplified formula of size L and weight W. Let L_2 be the number of leaves feeding into $\vee$ or $\wedge$ gates, let $L_\oplus$ be the number of leaves feeding into $\oplus$ gates, and let L_i be the number of leaves feeding into i-ary gates, for $3 \leqslant i \leqslant k$. Then we have $L = L_\oplus + \sum_{i=2}^{k} L_i = G_\oplus(F) + \sum_{i=2}^{k}(i-1)G_i(F) + 1$.

The next lemma gives an upper bound of the formula weight. It is similar to the bound by Chockler and Zwick [4] for formulas over unate bases. We need the following parameters:

$$\gamma_\oplus = 1 + \frac{\alpha_\oplus}{2} + \frac{\alpha_k}{2(k-1)},$$
$$\gamma_i = 1 + \frac{\alpha_i}{i} + \frac{\alpha_k}{i(k-1)}, \quad 2 \leqslant i \leqslant k.$$

Lemma 2 ([4]). *Suppose that* $\alpha_k/(k-1) \geqslant \alpha_\oplus$ *and* $\alpha_k/(k-1) \geqslant \alpha_i/(i-1)$ *for* $2 \leqslant i \leqslant k$. *Then the formula weight* $W \leqslant \gamma_\oplus L_\oplus + \sum_{i=2}^{k} \gamma_i L_i$.

The proof is almost the same as the proof in [4] for U_k-formulas. For completeness, we provide the proof in Appendix A.

3.4 Choice of weight parameters

We next choose suitable weight parameters. To get nontrivial shrinkage, we require that if a leaf feeding into $\oplus$ is restricted, then the weight reduces proportionally; for a leaf feeding into other types of gates, the weight should reduce by a factor strictly larger than 1. Also, since we will use greedy restrictions in our algorithm instead of completely random restrictions as in [4], we require an upper bound γ for both $\gamma_\oplus$ and γ_i's; that is γ does not depend on the gate type. This will give an upper bound of the weight $W \leqslant \gamma L$.

We choose the following:

$$\alpha_i = i - 1 - \frac{1}{2^{i-1}}, \quad i = 2, \ldots, k,$$
$$\alpha_\oplus = \frac{\alpha_k}{k-1} = 1 - \frac{1}{(k-1)\cdot 2^{k-1}},$$
$$\gamma = 2 - \frac{1}{(k-1)\cdot 2^{k-1}}.$$

It is easy to check that $\alpha_{i-1} + \alpha_\oplus \leqslant \alpha_i \leqslant (i-1)\alpha_\oplus$ for $i \geqslant 3$. We also have $\gamma_i < \gamma_k = \gamma_\oplus = \gamma$ for $2 \leqslant i < k$, and, by Lemma 2, $W \leqslant \gamma L$.

Let $\beta_\oplus$ (β_2, β_i for $i \geqslant 3$) be the average weight reduction from restricting a leaf which feeds into an $\oplus$ gate ($\vee$ or $\wedge$ gate, i-ary gate). Then, by Lemma 1,

$$\begin{aligned}
\beta_\oplus &= 1 + \alpha_\oplus = \gamma,\\
\beta_2 &= 1.5 + \alpha_2 = 2,\\
\beta_i &= 1 + \alpha_i - \frac{1}{2}\left[\alpha_{i-1} + (i-2)\alpha_\oplus\right]\\
&= 1 + \left(i - 1 - \frac{1}{2^{i-1}}\right) - \frac{1}{2}\left[\left(i - 2 - \frac{1}{2^{i-2}}\right) + (i-2)\cdot\left(1 - \frac{1}{(k-1)\cdot 2^{k-1}}\right)\right]\\
&= 2 + \frac{i-2}{(k-1)\cdot 2^k} > 2, \qquad \text{for } 3 \leqslant i \leqslant k,\\
\beta_{\min} &= \min\{1 + \alpha_2,\ 1 + \alpha_i - (i-2)\alpha_\oplus\} = 1.5,\\
\beta_{\max} &= \max\{2 + \alpha_2,\ 1 + \alpha_i - \alpha_{i-1}\} = 2.5.
\end{aligned}$$

Note that, we have $\beta_\oplus/\gamma = 1$, and $\beta_i/\gamma \geqslant 2/\gamma = 1 + \frac{1}{(k-1)\cdot 2^k - 1} > 1$, for $2 \leqslant i \leqslant k$. This means when a leaf feeding into $\oplus$ is restricted, the weight reduces by γ, and when a leaf feeding into other types of gates is restricted, the weight reduces nontrivially (by $\gamma(1 + \frac{1}{(k-1)\cdot 2^k - 1})$ on average).

4 A satisfiability algorithm

Seto and Tamaki [14] observed that, for a linear-size formula over B_2, either satisfiability checking is easy by solving systems of linear equations, or one can restrict a constant number of variables such that the formula shrinks non-trivially. This non-trivial shrinkage property leads to a satisfiability algorithm similar to Santhanam's algorithm [13]. The next lemma generalizes this property to B_k-formulas. The proof is essentially the same as the proof for B_2-formulas in [14]. We provide the proof in Appendix B.

Lemma 3 (Seto and Tamaki [14]). *Let F be a simplified n-input B_k-formula of size cn for $c > 0$. Then one of the following cases must be true:*

1. *The total number of maximal linear nodes is at most $3n/4$.*
2. *There exists a variable appearing at least $c + \frac{1}{8c}$ times.*

3. *There exists a maximal linear node v such that (1) the parent of v is not $\oplus$, (2) there are at most $8c$ variables under v, and (3) each variable under v appears at least c times in F.*

Theorem 3 (Theorem 1 restated). *For n-input B_k-formulas of size cn, there is a deterministic algorithm counting the number of satisfying assignments in time $2^{n(1-\delta_{c,k})}$, where $\delta_{c,k} = c^{-O(c^2k2^k)}$.*

Proof. The algorithm first simplifies the given formula, and then runs recursively. At each recursive step, suppose we have a simplified formula F with n free variables and size cn. Consider the following cases as in Lemma 3:

1. If there are at most $3n/4$ maximal linear nodes, we enumerate all possible assignments to the maximal linear nodes, which will generate at most $2^{3n/4}$ branches; for each branch, solve a system of linear equations using Gaussian elimination.
2. If a variable x appears at least $c + 1/8c$ times, build two branches for $x = 0$ and $x = 1$; for each branch, simplify the restricted formula and recurse.
3. Otherwise, find the *smallest* maximal linear node, say v, such that it has a non-$\oplus$ parent and at most $8c$ variables in $\mathrm{var}(F_v)$, and each variable in $\mathrm{var}(F_v)$ appears at least c times in F. We also find all maximal linear nodes $\{v_j\}$ which are over exactly the same set of variables as v, that is, $\mathrm{var}(F_{v_j}) = \mathrm{var}(F_v)$.
   (a) If all v_j's are feeding into different gates, we choose all variables in $\mathrm{var}(F_v)$.
   (b) If there are two v_j's feeding into the same gate, we choose all but one (arbitrary) variable in $\mathrm{var}(F_v)$.

   Enumerate all assignments to the chosen variables; for each assignment, restrict the formula, and recurse.

Each branch ends when the restricted formula becomes a constant. For each branch, we count the number of assignments consisting with the restrictions along the branch. The final answer is the summation of the counts for all branches where the restricted formula becomes 1. In the rest of the proof, we bound the number of branches, and thus the running time of the algorithm.

In cases 2, 3(a) and 3(b), we wish to restrict a constant number of variables such that the formula weight/size reduces non-trivially. (A trivial weight reduction from restricting one variable is $c\gamma$, where c is the average number of appearances and γ is the smallest average weight reduction for each leaf.)

In case 2, we get non-trivial reduction since the selected variable appears more than the average.

In case 3, suppose there are d variables in $\mathrm{var}(F_v)$, and we restrict these variables one by one. For case 3(a), when we restrict each of the first $d-1$ variables, we always eliminate leaves feeding to $\oplus$ gates; this is guaranteed by the minimality of v. For the last variable, we eliminate at least one leaf feeding into a non-$\oplus$ gate. The average weight reduction together is at least $(d-1)c\gamma + (c-1)\gamma + 2 = dc\gamma + 2 - \gamma$.

For case 3(b), whenever two v_j's feed into the same gate, the gate cannot be $\oplus$ since v_j's are maximal linear nodes. Similar to case 3(a), for each variable restricted, we always eliminate leaves feeding to $\oplus$ gates. At the end, the unrestricted variable in $\mathrm{var}(F_v)$ will appear twice as literals feeding into a non-$\oplus$ gate; then at least one leaf can be eliminated. The average weight reduction will be at least $(d-1)c\gamma + 1$.

Consider a partial branch in the recursion tree up to depth $n - l$, for l to be specified later. If case 1 occurs along the branch, then all extensions of this branch will have depth at most $n - l/4$. We next assume case 1 does not occur. We will argue that most branches at depth $n-l$ have formulas of size at most $l/2$, although there are still l variables unrestricted; then extensions of such branches will have depth at most $n - l/2$.

In the following, we claim that the formula weight shrinks with high probability. Intuitively this is because we restrict a small number of variables in each of the cases 2, 3(a) and 3(b) and this gives non-trivial weight reductions. We provide a proof of this claim in Appendix C. A similar result for B_2-formulas was shown in [14].

Claim. Let F be a simplified U_k-formula of size cn and weight W. Let W' be the weight of the formula obtained after restricting $n - l$ variables (according to cases 2, 3(a) and 3(b)). Then for large enough l,

$$\mathbf{Pr}\left[W' \geqslant 2W\left(\frac{l}{n}\right)^{\Gamma}\right] < 2^{-l/bc^2},$$

for $\Gamma = 1 + \Omega(1/c^2k2^k)$ and a constant $b > 0$.

We choose $l = pn$ for $p = (8c)^{-1/(\Gamma-1)}$. Then after restricting $n - l$ variables, there are 2^{n-l} branches, but $1-2^{-l/bc^2}$ fraction of the branches end with formulas of size

$$L' \leqslant W' \leqslant 2W\left(\frac{l}{n}\right)^{\Gamma} \leqslant 2 \cdot \gamma cn \cdot p^{\Gamma} = 2\gamma cp^{\Gamma-1} \cdot l < \frac{l}{2}.$$

These "small" formulas depend on at most $l/2$ variables, although there are still l variables unrestricted. Completing such branches will get to depth at most $n - l/2$. The total number of complete branches will be at most $2^{n-l} \cdot 2^{l/2} = 2^{n-l/2}$.

For $2^{-l/bc^2}$ fraction of branches having "large" formulas, although they may extend to depth n, the total number of such complete branches will be at most $2^{n-l/bc^2}$.

Therefore, the algorithm generates at most $2^{n-\Omega(l/c^2)}$ branches; the running time is bounded by $2^{n(1-\delta)}$, for $\delta = c^{-O(c^2k2^k)}$. □

5 Average-case lower bounds

The algorithm in Theorem 1 essentially constructs a *parity decision tree*, where at each node of the tree, we can either restrict a variable or a parity function (maximal linear node). The next corollary follows directly.

Corollary 1. *An n-input B_k-formula of size cn has a parity decision tree of size $2^{(1-\delta_{c,k})n}$, where $\delta_{c,k} = c^{-O(c^2 k 2^k)}$.*

An average-case lower bound (also called correlation bound) is a lower bound on the minimal formula size for approximating an explicit function. Santhanam [13] showed that linear-size de Morgan formulas compute the parity function correctly on at most $1/2 + 2^{-\Omega(n)}$ fraction of the inputs. Seto and Tamaki [14] extended this to B_2-formulas by showing that linear-size B_2-formulas can compute affine extractors correctly on at most $1/2 + 2^{-\Omega(n)}$ fraction of the inputs. We next generalize this result for B_k-formulas.

Let F_2 be the finite field with elements $\{0,1\}$. A function $\mathsf{E}\colon F_2^n \to F_2$ is a (k,ϵ)-*affine extractor* if for any uniform distribution X over some k-dimensional affine subspace of F_2^n, it holds that $|\mathbf{Pr}[\mathsf{E}(X) = 1] - 1/2| \leqslant \epsilon$. We will need the following known construction of affine extractors.

Theorem 4 ([19, 11]). *For any $\delta > 0$ there exists a polynomial-time computable (k,ϵ)-affine extractor $\mathsf{E}_\delta\colon \{0,1\}^n \to \{0,1\}$ with $k = \delta n$ and $\epsilon = 2^{-\Omega(n)}$.*

By Corollary 1, any B_k-formulas size cn has a parity decision tree of size $2^{n-\Omega(n)}$. Since most branches of the tree have depth $n - \Omega(n)$ (defining affine subspaces of dimension $\Omega(n)$), by the definition of E_δ, the tree has exponentially small correlation with E_δ. Thus Theorem 2 follows immediately. The proof is similar to the proof for B_2-formulas in [14]; we provide a proof in Appendix D.

6 Formulas over unate bases

Chockler and Zwick [4] showed that U_k-formulas have shrinkage exponent $\Gamma_k > 1$ under random restrictions. This implies that Parity requires U_k-formula size n^{Γ_k} and Andreev's function [1] requires size $n^{1+\Gamma_k - o(1)}$. In this section, we strengthen this result to concentrated shrinkage, which directly gives #SAT algorithms and average-case lower bounds for U_k-formulas of super-quadratic size, similar to the results for de Morgan formulas [13, 9, 10, 2].

6.1 Shrinkage under restrictions

We modify the simplification rules in Section 3.1 as follows. Rules 3(a)-(c) are not necessary since unate functions do not have linear representations. To ensure unateness, we change rule 2(c) to the following:

> 2.(c') If a ternary gate is fed by two literals of the same variable, replace it by a binary gate. If an i-ary gate, for $i \geqslant 4$, is fed by more than two literals of the same variable, replace it by a smaller gate.

If $f(x_1, x_2, x_3)$ is unate, then $g(x, y) := f(x, \overline{x}, y)$ must be unate, since g is positive or negative unate in y, whereas the only non-unate binary functions ($x \oplus y$ and $x \oplus y \oplus 1$) do not have this property. Similarly, for $i \geqslant 4$, unnecessary inputs can be eliminated.

Let F be a U_k-formula with size L and weight W. Let α_i be the weight attached to an i-ary gate, for $2 \leqslant i \leqslant k$. The average weight reductions from restricting a single leaf change to the following:

$$\begin{aligned}
&\beta_2 = 1.5 + \alpha_2,\\
&\beta_3 = 1 + \alpha_3 - \alpha_2,\\
&\beta_i = 1 + \min\{\alpha_i - \alpha_{i-1},\ \frac{1}{2}(\alpha_i - \alpha_{i-2})\},\quad 4 \leqslant i \leqslant k.
\end{aligned}$$

Let L_i be the number of leaves feeding into i-ary gates. As shown in [4] (following Lemma 2), we get the upper bound $W \leqslant \sum_{i=2}^{k} \gamma_i L_i$, where $\gamma_i = 1 + \frac{\alpha_i}{i} + \frac{\alpha_k}{i(k-1)}$, for $2 \leqslant i \leqslant k$.

We choose $\alpha_i = (i-2)/2$, and $\gamma = \gamma_k = 1 + \frac{k-2}{2(k-1)}$. This gives $W \leqslant \gamma L$, and $\beta_i \geqslant 1.5$ for $2 \leqslant i \leqslant k$. Let $\Gamma_k = 1.5/\gamma = 1 + \frac{1}{3k-4}$. Let F' be the formula obtained by randomly fixing the most frequent variable in F. Then $\mathbf{E}[W(F')] \leqslant W - 1.5L/n \leqslant W(1 - \Gamma_k/n) \leqslant W(1-1/n)^{\Gamma_k}$. One can repeat this process and show that the weight shrinks with high probability, following the approach of [13, 2].

6.2 #SAT algorithms and average-case lower bounds

Based on concentrated shrinkage, the algorithms for de Morgan formulas in [13, 2] can be easily generalized for U_k-formulas.

Theorem 5. *There are deterministic algorithms counting satisfying assignments for n-input U_k-formulas of*

- *size cn in time $2^{n(1-\Omega(1/c^{3k-4}))}$;*
- *size $n^{2+\frac{1}{3k-4}-\epsilon}$ in time $2^{n-n^{\Omega(1)}}$, for any constant $\epsilon > 0$.*

We provide a proof in Appendex E. Similar to [13, 2], the algorithms also give non-trivial decision tree size for small U_k-formulas, and an average-case lower bound for computing Parity.

Corollary 2. *An n-input U_k-formula of size cn has a decision tree of size $2^{n(1-\Omega(1/c^{3k-4}))}$.*

Corollary 3. *Any family of U_k-formulas of size cn cannot compute Parity correctly on more than $1/2 + 2^{-\Omega(n/c^{3k-4})}$ fraction of inputs.*

The best known average-case lower bound for de Morgan formulas is $n^{2.99}$ by Komargodski, Raz and Tal [10], which was shown for a generalized Andreev's function [1]. A similar hard function was also given in [2], which works for restrictions where each variable is selected deterministically. It is easy to see that this function is also hard for small U_k-formulas.

Let $\mathsf{H}\colon \{0,1\}^n \to \{0,1\}$ be the generalized Andreev's function in [2].

Theorem 6. *For any family of U_k-formulas F_n of size $n^{2+\frac{1}{3k-4}-\epsilon}$ where $\epsilon > 0$,*

$$\mathbf{Pr}[F_n(x) = \mathsf{H}(x)] \leqslant 1/2 + 2^{-n^{\Omega(1)}}.$$

7 Open questions

For B_k-formulas of size cn, we give a #SAT algorithm improving over exhaustive search by a factor of $2^{\delta n}$, for $\delta = c^{-O(c^2k2^k)}$. An open question is whether we can improve δ to be polynomially small in c, as for U_k-formulas. Another question is whether we have nontrivial satisfiability algorithms for U_k-formulas of size larger than n^{Γ_k+1}, where Γ_k is the shrinkage exponent. The satisfiability algorithms we have for U_k-formulas of super-linear size use exponential space; it would be interesting to have polynomial-space algorithms.

References

1. A.E. Andreev. On a method of obtaining more than quadratic effective lower bounds for the complexity of π-schemes. *Vestnik Moskovskogo Universiteta. Matematika*, 42(1):70–73, 1987. English translation in *Moscow University Mathematics Bulletin*.
2. R. Chen, V. Kabanets, A. Kolokolova, R. Shaltiel, and D. Zuckerman. Mining circuit lower bound proofs for meta-algorithms. In *Proceedings of the 29th Annual IEEE Conference on Computational Complexity*, CCC '14, 2014.
3. R. Chen, V. Kabanets, and N. Saurabh. An improved deterministic #sat algorithm for small de morgan formulas. In *Proceedings of Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Part II*, pages 165–176, 2014.
4. H. Chockler and U. Zwick. Which bases admit non-trivial shrinkage of formulae? *Computational Complexity*, 10(1):28–40, 2001.
5. F. Fomin, F. Grandoni, and D. Kratsch. A measure & conquer approach for the analysis of exact algorithms. *J. ACM*, 56(5):25:1–25:32, August 2009.
6. J. Håstad. The shrinkage exponent of de Morgan formulae is 2. *SIAM Journal on Computing*, 27:48–64, 1998.
7. R. Impagliazzo, R. Meka, and D. Zuckerman. Pseudorandomness from shrinkage. In *Proceedings of the Fifty-Third Annual IEEE Symposium on Foundations of Computer Science*, pages 111–119, 2012.
8. R. Impagliazzo and N. Nisan. The effect of random restrictions on formula size. *Random Structures and Algorithms*, 4(2):121–134, 1993.
9. I. Komargodski and R. Raz. Average-case lower bounds for formula size. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, pages 171–180, 2013.
10. I. Komargodski, R. Raz, and A. Tal. Improved average-case lower bounds for demorgan formula size. In *Proceedings of the Fifty-Fourth Annual IEEE Symposium on Foundations of Computer Science*, pages 588–597, 2013.
11. X. Li. A new approach to affine extractors and dispersers. In *IEEE Conference on Computational Complexity*, pages 137–147, 2011.
12. M. Paterson and U. Zwick. Shrinkage of de Morgan formulae under restriction. *Random Structures and Algorithms*, 4(2):135–150, 1993.
13. R. Santhanam. Fighting perebor: New and improved algorithms for formula and qbf satisfiability. In *Proceedings of the Fifty-First Annual IEEE Symposium on Foundations of Computer Science*, pages 183–192, 2010.

14. K. Seto and S. Tamaki. A satisfiability algorithm and average-case hardness for formulas over the full binary basis. In *Proceedings of the Twenty-Seventh Annual IEEE Conference on Computational Complexity*, pages 107–116, 2012.
15. B.A. Subbotovskaya. Realizations of linear functions by formulas using and, or, not. *Soviet Math. Doklady*, 2:110–112, 1961.
16. R. Williams. Improving exhaustive search implies superpolynomial lower bounds. In *Proceedings of the Forty-Second Annual ACM Symposium on Theory of Computing*, 2010.
17. R. Williams. Non-uniform ACC circuit lower bounds. In *Proceedings of the Twenty-Sixth Annual IEEE Conference on Computational Complexity*, pages 115–125, 2011.
18. R. Williams. Algorithms for circuits and circuits for algorithms. In *IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, June 11-13, 2014*, pages 248–261, 2014.
19. A. Yehudayoff. Affine extractors over prime fields. *Combinatorica*, 31(2):245–256, 2011.

A Proof of Lemma 2

Lemma 4 (Lemma 2 restated [4]). *Suppose that* $\alpha_k/(k-1) \geqslant \alpha_\oplus$ *and* $\alpha_k/(k-1) \geqslant \alpha_i/(i-1)$ *for* $2 \leqslant i \leqslant k$. *Then the formula weight* $W \leqslant \gamma_\oplus L_\oplus + \sum_{i=2}^{k} \gamma_i L_i$.

Proof. The upper bound of W is given by the following linear program:

$$\begin{aligned} \max\ & L + \alpha_\oplus G_\oplus + \sum_{i=2}^{k} \alpha_i G_i \\ \text{where}\ & G_\oplus + \sum_{i=2}^{k} (i-1) G_i = L - 1 \\ & G_\oplus \geqslant L_\oplus/2, \\ & G_i \geqslant L_i/i, \quad 2 \leqslant i \leqslant k. \end{aligned}$$

Note that, we view $L, L_\oplus, L_i$'s as fixed constants, and $G_\oplus, G_i$'s as variables of the linear program. The condition $\alpha_k/(k-1) \geqslant \{\alpha_\oplus, \alpha_i/(i-1)\}$ guarantees that the optimal value is obtained when there are only the minimum required number of $\oplus$ and i-ary gates for $i < k$, and all the rest are k-ary gates. That is, when $G_\oplus = L_\oplus/2$, $G_i = L_i/i$ for $2 \leqslant i < k$, and

$$G_k = \frac{1}{k-1}\left(L - 1 - G_\oplus + \sum_{i=2}^{k-1} (i-1) G_i\right).$$

Since

$$G_k < \frac{1}{k-1}\left(L - \frac{L_\oplus}{2} - \sum_{i=2}^{k-1} \frac{i-1}{i} L_i\right) = \frac{1}{k-1}\left(L_k + \frac{L_\oplus}{2} + \sum_{i=2}^{k-1} \frac{L_i}{i}\right),$$

we have

$$W = L + \alpha_{\oplus} G_{\oplus} + \sum_{i=2}^{k} \alpha_i G_i \leqslant \gamma_{\oplus} L_{\oplus} + \sum_{i=2}^{k} \gamma_i L_i$$

□

B Proof of Lemma 3

Lemma 5 (Lemma 3 restated). *Let F be a simplified n-input B_k-formula of size cn for $c > 0$. Then one of the following cases must be true:*

1. *The total number of maximal linear nodes is at most $3n/4$.*
2. *There exists a variable appearing at least $c + \frac{1}{8c}$ times.*
3. *There exists a maximal linear node v such that (1) the parent of v is not $\oplus$, (2) there are at most $8c$ variables under v, and (3) each variable under v appears at least c times in F.*

Proof. Suppose that neither case 1 nor case 2 is true, and we wish to prove case 3. We need the following claims.

Claim. The number of maximal linear nodes with non-$\oplus$ parents is more than those with $\oplus$ parents.

Proof. We first replace each maximal linear node by a single leaf; this will not change the number of maximal linear nodes or their parents. Obviously, the result holds when there are at most 2 internal nodes. We next prove by induction on the tree size.

There must be one internal node labeled by non-$\oplus$ which is completely fed by leaves. We denote this node by u, and let v be its parent. If v is labeled by non-$\oplus$, then we can replace u by one of its children; this removes at least one maximal linear node with non-$\oplus$ parent (the other children of u), but does not change those with $\oplus$-parents. If v is labeled by $\oplus$, then we can replace v by the sibling of u; this removes at least two maximal linear nodes with non-$\oplus$ parents (the children of u), but removes at most one maximal linear node with $\oplus$-parent. Then the result holds by induction on smaller tree size. □

Claim. There are at most $n/8$ maximal linear nodes which have a variable appearing less than c times in F.

Proof. By assumption, all variables appear less than $c+1/8c$ times. By averaging, the largest number of times a variable appears must be an integer $c' \in [c, c + 1/8c)$. This implies there are at most $n/8c$ variables appearing less than c (at most $c' - 1$) times. The number of leaves labeled by these variables is at most $n/8$. □

(Proof of the lemma continued.) By Claim B, the number of maximal linear node with non-$\oplus$ parents is at least $(3n/4)/2 \geqslant 3n/8$. By the averaging argument, the number of maximal linear nodes with $8c$ leaves is at most $n/8$. Then following from Claim B, there are at least $n/8$ maximal linear nodes as required in case 3. □

C Proof of Claim 4

A sequence of random variables $X_0, X_1, \ldots, X_n$ is called a *supermartingale* with respect to a sequence of random variables $R_1, \ldots, R_n$ if $\mathbf{E}[X_i \mid R_{i-1}, \ldots, R_1] \leqslant X_{i-1}$, for $1 \leqslant i \leqslant n$.

Lemma 6 (Azuma-Hoeffding Inequality). *If $\{X_i\}_{i=0}^n$ is a supermartingale such that, for each i, $|X_i - X_{i-1}| \leqslant c_i$, then, for any $\lambda \geqslant 0$,*

$$\mathbf{Pr}[X_n - X_0 \geqslant \lambda] \leqslant \exp\left(-\frac{\lambda^2}{2\sum_{i=1}^n c_i^2}\right).$$

Claim (Claim 4 restated). Let F be a simplified U_k-formula of size cn and weight W. Let W' be the weight of the formula obtained after restricting $n-l$ variables (according to cases 2, 3(a) and 3(b)). Then for $l = \Omega(n)$,

$$\mathbf{Pr}\left[W' \geqslant 2W\left(\frac{l}{n}\right)^{\Gamma}\right] < 2^{-l/bc^2},$$

for $\Gamma = 1 + \Omega(1/c^2k2^k)$ and a constant $b > 0$.

Proof. The recursive procedure can be viewed as building up a decision tree, where at each step, we pick a variable from the formula, build two branches by restricting the variable. For cases 3(a) and 3(b) where multiple variables are chosen, we can restrict each variable one by one. We label each node of this decision tree by the restricted formula, with the root labeled by F. We will consider this decision tree built up to depth $n - l$.

Let $F_0 := F$, and let F_i be the formula we obtained after restricting i variables, where $i = 1, \ldots, n - l$. Let $L_i = L(F_i)$ and $W_i = W(F_i)$.

We wish to use the Azuma-Hoeffding inequality to get concentrated weight reduction. However, the "bounded differences" condition required in the Azuma-Hoeffding inequality does not hold for $\{W_i\}$, since the formula may be simplified dramatically under restrictions. Therefore, we will introduce a new sequence $\{W_i'\}$ that captures only the *necessary* weight reductions. We wish to have $W_i' \geqslant W_i$, and the *extra* weight reductions $W_i' - W_i$ is a non-negative sequence which will be disregarded.

We call restrictions according to cases 2, 3(a), 3(b) as a *super-step*. In each super-step, we restrict $1 \leqslant d \leqslant 8c$ variables altogether.

We first define ΔW below to be the *necessary weight reduction* from a super-step; it will account only the weight reductions from eliminating the restricted leaf, the gate fed by the leaf, and, for $\vee, \wedge$ gates, one sibling leaf if it is eliminated. In other words, ΔW is the smallest possible weight reduction from restricting leaves (with respect to all possible formula structure). Suppose at the beginning of a super-step, we have a formula on n variables with size $L = cn$ and weight W. Note that $1.5L \leqslant W \leqslant \gamma L$. Let $W_0' = W_0 = W$, and we will define W_i' below.

– Case 2: If a variable appears at least $c + 1/8c$ times, let ΔW be the weight reduction from restricting exactly $c+1/8c$ leaves. This gives $1.5(c+1/8c) = \beta_{\min}(c + 1/8c) \leqslant \Delta W \leqslant \beta_{\max}(c + 1/8c) = 2.5(c + 1/8c)$, and $\mathbf{E}[\Delta W] \geqslant \gamma(c + 1/8c)$.
Let $W'_1 = W - \Delta W$. Then

$$\mathbf{E}[W'_1] \leqslant W\left(1 - \frac{1 + \frac{1}{8c^2}}{n}\right) \leqslant W(1 - 1/n)^{1+\frac{1}{8c^2}} \tag{1}$$

$$W\left(1 - \frac{\frac{2.5}{1.5}(1 + \frac{1}{8c^2})}{n}\right) \leqslant W'_1 \leqslant W\left(1 - \frac{\frac{1.5}{\gamma}(1 + \frac{1}{8c^2})}{n}\right) \tag{2}$$

– Case 3(a): If we are restricting all d variables in a maximal linear node, then let ΔW be the weight reduction from restricting exactly c leaves of each variable, and, for the last variable, we include its appearance as a leaf feeding into a non-$\oplus$ gate. For each of the first $d-1$ variables, the weight reduction is $c\gamma$, since they all feed into $\oplus$. For the last variable, the weight reduction is between $1.5c$ and $2.5c$, and $(c-1)\gamma+2$ on average. Then $(d-1)\gamma c + 1.5c \leqslant \Delta W \leqslant (d-1)\gamma c + 2.5c$ and $\mathbf{E}[\Delta W] \geqslant d\gamma c + 2 - \gamma$.
Let $W'_d = W - \Delta W$. Then

$$\mathbf{E}[W'_d] \leqslant W\left(1 - \frac{d + \frac{2-\gamma}{\gamma c}}{n}\right) \leqslant W(1 - d/n)^{1+\frac{2-\gamma}{8c^2\gamma}} \tag{3}$$

$$W\left(1 - \frac{(d-1)\gamma/1.5 + 2.5/1.5}{n}\right) \leqslant W'_d \leqslant W\left(1 - \frac{(d-1) + 1.5/\gamma}{n}\right) \tag{4}$$

For $1 \leqslant i \leqslant d-1$, simply let $W'_i = W(1 - \frac{i}{n})^{1+\frac{2-\gamma}{8c^2\gamma}}$.

– Case 3(b): If we are restricting all but one variable in a maximal linear node, then, let ΔW be the weight reduction from restricting exactly c leaves for each variable, and we also add the reduction from eliminating one leaf labeled by the unrestricted variable. Suppose there are $d+1$ variables in the maximal linear node. We have $\Delta W = cd\gamma + 1$.
Let $W'_d = W - \Delta W$. Then

$$\mathbf{E}[W'_d] \leqslant W\left(1 - \frac{d + \frac{1}{\gamma c}}{n}\right) \leqslant W(1 - d/n)^{1+\frac{1}{8c^2\gamma}} \tag{5}$$

$$W\left(1 - \frac{(d\gamma + 1/c)/1.5}{n}\right) \leqslant W'_d \leqslant W\left(1 - \frac{d + 1/\gamma c}{n}\right) \tag{6}$$

For $1 \leqslant i \leqslant d-1$, simply let $W'_i = W(1 - \frac{i}{n})^{1+\frac{1}{8c^2\gamma}}$.

Let $\gamma' = 1 + \min\{1/8c^2,\ (2-\gamma)/8c^2\gamma,\ 1/8c^2\gamma\} = 1 + (2-\gamma)/8c^2\gamma$. Then we have, for $1 \leqslant i \leqslant d$,

$$\mathbf{E}[W'_i] \leqslant W(1 - i/n)^{\gamma'}. \tag{7}$$

Let $R_1, R_2, \ldots$ be the random bits assigned to variables. Note that, given $R_1, \ldots, R_i$, the formula F_i and then W_i are fixed. Suppose for F_i, a super-step starts and it will restrict d variables; we can define $W'_{i+1}, \ldots, W'_{i+d}$ as above, based on F_i and the value of W_i. Note that, F_i was produced from the previous super-step, and W'_i was defined by the previous super-step; we also have $W'_i \geqslant W_i$.

Let $w_i = \log(W_i)$. and $w'_i = \log(W'_i)$. Conditioning on $R_1, \ldots, R_{i-1}$, if the i-th variable restricted is the first variable in a super-step, let

$$Z_i = w'_i - w_{i-1} - \gamma' \log\left(1 - \frac{1}{n-(i-1)}\right),$$

otherwise,

$$Z_i = w'_i - w'_{i-1} - \gamma' \log\left(1 - \frac{1}{n-(i-1)}\right).$$

Given $R_1, \ldots, R_{i-1}$, if the i-th variable restricted is not the last variable in a super-step, then $Z_i \leqslant 0$. Otherwise, by (7) and $\mathbf{E}[w'_i] \leqslant \log \mathbf{E}[W'_i]$, we have $\mathbf{E}[Z_i \mid R_1, \ldots, R_{i-1}] \leqslant 0$.

By (2), (4) and (6), following from the Taylor series for the ln function, and that $1.5 \leqslant \gamma < 2$, $c > 3/4$, and $l = \Omega(n)$, there is some large enough constant $b = \Omega(c) > 0$ such that $|Z_i| < c_i := -b \log\left(1 - \frac{1}{n-(i-1)}\right) = b \log\left(1 + \frac{1}{n-i}\right) \leqslant \frac{b}{n-i}$.

Let $X_0 = 0$ and $X_i = \sum_{j=1}^{i} Z_i$; then the sequence $\{X_i\}$ is a supermartingale with respect to $\{R_i\}$.

Let $i = n - l$; we have

$$\sum_{j=1}^{i} c_j^2 \leqslant b^2 \sum_{j=1}^{i} \left(\frac{1}{n-j}\right)^2 \leqslant b^2 \sum_{j=1}^{i} \left(\frac{1}{n-j} - \frac{1}{n-(j-1)}\right) \leqslant b^2 \frac{1}{n-i} = \frac{b^2}{l}.$$

By the Azuma-Hoeffding inequality (Lemma 6), for any $\lambda > 0$,

$$\mathbf{Pr}\left[X_i - X_0 \geqslant \lambda\right] \leqslant \exp\left(-\frac{\lambda^2}{2\sum_{j=1}^{i} c_j^2}\right) \leqslant e^{-\lambda^2 l / b^2}.$$

We have

$$\begin{aligned} X_i - X_0 &= \sum_{j=1}^{i} Z_j \\ &\geqslant w'_i - w_0 - \gamma' \log(l/n). \end{aligned}$$

Therefore, for $\lambda = \ln 2$,

$$\mathbf{Pr}\left[W'_{n-l} \geqslant 2W\left(\frac{l}{n}\right)^{\gamma'}\right] < 2^{-l/3b^2}.$$

We can find $l' < l + 8c$ such that $\mathbf{Pr}\left[W_{n-l'} \geqslant 2W\left(\frac{l'}{n}\right)^{\gamma'}\right] < 2^{-l'/c^2 b'}$, for some constant $b' > 0$.

□

D Proof of Theorem 2

Theorem 7 (Theorem 2 restated). *For any family of B_k-formulas F_n of size cn for a constant $c > 0$, there is $\delta > 0$ such that*

$$\mathbf{Pr}[F_n(x) = \mathsf{E}_\delta(x)] \leqslant 1/2 + 2^{-\Omega(n)}.$$

Proof. As proved in Theorem 1, we can construct a parity decision tree for F_n such that, at depth $n - l'$ for $l' = l/4 = \Omega(n)$, a fraction $1 - 2^{-l'}$ of the branches have restricted formulas becoming constants. Restrictions along any such branch define an affine subspace of dimension at least l'; by Theorem 4, F_n can compute E_δ correctly on at most $1/2 + 2^{-\Omega(n)}$ fraction of inputs from the subspace.

On the other hand, there are at most $2^{-l'}$ fraction of the branches with larger depth; even when F_n computes E_δ correctly for all these branches, the fraction of inputs is at most $2^{-l'} = 2^{-\Omega(n)}$.

Therefore, the fraction of inputs where F_n computes E_δ correctly is at most $1/2 + 2^{-\Omega(n)}$.

□

E Proof of Theorem 5

Theorem 8 (Theorem 5 restated). *There are deterministic algorithms counting satisfying assignments for n-input U_k-formulas of*

- *size cn in time $2^{n(1-\Omega(1/c^{3k-4}))}$;*
- *size $n^{2+\frac{1}{3k-4}-\epsilon}$ in time $2^{n-n^{\Omega(1)}}$, for any constant $\epsilon > 0$.*

Proof. Let F be a U_k-formula on n inputs of size L and weight W. Let $\Gamma = 1 + \frac{1}{3k-4}$. The algorithms starts by simplifying F, and then runs recursively. At each recursive step, we find the most frequent variable, say x, build two branches by restricting $x = 0$ and $x = 1$, and, for each branch, recurse on the simplified formula.

Let F_1 be the formula obtained after restricting one variable. Since the most frequent variable appears L/n times, we have $\mathbf{E}[W(F_1)] \leqslant W - 1.5L/n = W(1 - 1.5L/nW) \leqslant W(1 - (1.5/\gamma)/n) \leqslant W(1 - 1/n)^\Gamma$. We repeat this for $n - l$ steps, for l to fixed later; let F' be the formula we get after restricting $n - l$ variables. Using a similar proof as in [13, 2], or a simplified version of Claim 4, one can show that the weight shrinks with high probability:

$$\mathbf{Pr}[W(F') \geqslant 2W \cdot (l/n)^\Gamma] \leqslant 2^{-l/4}.$$

If $L(F) = cn$, we choose $l = pn$ for $p = (8c)^{-1/(\Gamma-1)}$. Then, for all but $2^{-l/4}$ fraction of the 2^{n-l} branches, the restricted formulas have small size $L(F') \leqslant W(F') \leqslant l/2$. We use brute-force search; all small formulas (size at most $l/2$) take time $2^{n-l} \cdot 2^{l/2}$, and all large formulas take time at most $2^{n-l} \cdot 2^{-l/4} \cdot 2^{l}$. The total running time is bounded by $2^{n-\Omega(l)} = 2^{n(1-\Omega(c^{-1/(\Gamma-1)}))}$.

If $L(F) = n^{1+\Gamma-\epsilon}$, we choose $l = n^{\alpha}$ for $\alpha = \epsilon/2$. Then, all but $2^{-l/4}$ fraction of the 2^{n-l} branches have restricted formulas of size $L(F') < 4n^{1-\epsilon+\Gamma\alpha} < n^{\delta}$ for $0 < \delta < 1$.

For formulas of size at most n^{δ}, we use memoization. We first count satisfying assignments for all possible formulas of size at most n^{δ}, and store the results into a lookup table. Then whenever we encounter a formula of size at most n^{δ} (on at most n^{α} variables) in the recursion, we can simply look up the stored answer. The running time for building the lookup table is at most $2^{n^{\delta} \log n} \cdot 2^{n^{\alpha}} \cdot \mathsf{poly}(n) = 2^{n^{\Omega(1)}}$. The running time for solving all small formulas is at most $2^{n-l} \cdot \mathsf{poly}(n)$.

For formulas of size larger than n^{δ}, there are only few of them; we use brute-force search. The total running time is bounded by $2^{n-n^{\Omega(1)}}$. □

ECCC ISSN 1433-8092

http://eccc.hpi-web.de