THEORY OF COMPUTING, Volume 14 (17), 2018, pp. 1–25
www.theoryofcomputing.org

New Algorithms and Lower Bounds for Circuits With Linear Threshold Gates

R. Ryan Williams*

Received June 28, 2015; Revised June 4, 2018; Published December 10, 2018

Abstract: Let $\mathsf{ACC} \circ \mathsf{THR}$ be the class of constant-depth circuits comprised of AND, OR, and MODm gates (for some constant $m > 1$), with a bottom layer of gates computing arbitrary linear threshold functions. This class of circuits can be seen as a "midpoint" between ACC (where we know nontrivial lower bounds) and depth-two linear threshold circuits (where nontrivial lower bounds remain open). We give an algorithm for evaluating an arbitrary symmetric function of $2^{n^{o(1)}}$ $\mathsf{ACC} \circ \mathsf{THR}$ circuits of size $2^{n^{o(1)}}$, on all possible inputs, in $2^n \cdot \mathrm{poly}(n)$ time. Several consequences are derived:

- The number of satisfying assignments to an $\mathsf{ACC} \circ \mathsf{THR}$ circuit of $2^{n^{o(1)}}$ size is computable in $2^{n-n^{\varepsilon}}$ time (where $\varepsilon > 0$ depends on the depth and modulus of the circuit).
- NEXP does not have quasi-polynomial size $\mathsf{ACC} \circ \mathsf{THR}$ circuits, and NEXP does not have quasi-polynomial size $\mathsf{ACC} \circ \mathsf{SYM}$ circuits. Nontrivial size lower bounds were not known even for $\mathsf{AND} \circ \mathsf{OR} \circ \mathsf{THR}$ circuits.
- Every 0-1 integer linear program with n Boolean variables and s linear constraints is solvable in $2^{n-\Omega(n/\log^4(sM(\log n)))} \cdot \mathrm{poly}(s,n,M)$ time with high probability, where $M \leq 2^{n^{o(1)}}$ is an upper bound on the bit-length of the coefficients. (For example, 0-1 integer programs with weights in $[-2^{n^{o(1)}}, 2^{n^{o(1)}}]$ and $\mathrm{poly}(n)$ constraints can be solved in $2^{n-\Omega(n/\log^4 n)}$ time.)

*Supported by an Alfred P. Sloan Fellowship, a Microsoft Research Faculty Fellowship, a David Morgenthaler II Faculty Fellowship, NSF CCF-1212372 and NSF CCF-1552651 (CAREER). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

ACM Classification: F.1.1, F.2.3, G.1.6

AMS Classification: 68Q15, 68Q17, 68Q15

Key words and phrases: circuit complexity, satisfiability algorithms, linear threshold functions

 DOI: 10.4086/toc.2018.v014a017

We also present an algorithm for evaluating depth-two linear threshold circuits (also known as $\mathsf{THR} \circ \mathsf{THR}$) with exponential weights and $2^{n/24}$ size on all 2^n input assignments, running in $2^n \cdot \mathrm{poly}(n)$ time. This is evidence that non-uniform lower bounds for $\mathsf{THR} \circ \mathsf{THR}$ are within reach.

1 Introduction

In the non-uniform Boolean circuit model, one designs an infinite family of logical circuits $\{C_n\}$, one for each input length n, in order to recognize a given binary language $L \subseteq \{0,1\}^{\star}$. This model is notoriously powerful, even when the size of C_n is bounded from above by a fixed polynomial in n, defining the complexity class P/poly. With polynomial size circuits, one can already "compute" some undecidable languages, such as $L' = \{1^n \mid \text{the } n\text{th Turing machine halts on blank tape}\}$. Nevertheless, it is strongly believed that $\mathsf{NP} \not\subset \mathsf{P}/\mathrm{poly}$, meaning that for even modestly-sized instances of NP-complete problems, the sizes of *computations* on such instances must be inevitably gigantic. However, knowledge of P/poly is rather poor, partly due to the "infinite" nature of the model: it is open if the huge complexity class *nondeterministic exponential time* (NEXP) is contained in P/poly. This containment would imply that problems verifiable with exponentially-long witnesses could be efficiently "solved" with small circuits. The possibility looks obviously absurd, but we do not know at present how to rule it out.

In recent years, it has been demonstrated that the existence of nontrivial circuit-analysis algorithms is closely linked to the NEXP versus P/poly problem. For instance, Impagliazzo, Kabanets, and Wigderson [20] showed that $\mathsf{NEXP} \not\subset \mathsf{P}/\mathrm{poly}$ follows, if there is a $2^{n^{o(1)}}$ time algorithm that can approximate a given circuit's acceptance probability to within $1/10$. They also proved a partial converse, in that $\mathsf{NEXP} \not\subset \mathsf{P}/\mathrm{poly}$ implies a certain kind of derandomization. Subsequent work [53] strengthened the algorithms-to-lower bounds implication, proving that a similar algorithm which (for every k) runs in $2^{n-\omega(\log n)}$ time on all n-input n^k-size circuits still implies $\mathsf{NEXP} \not\subset \mathsf{P}/\mathrm{poly}$. A variant of this implication (for circuit satisfiability algorithms) was combined with an satisfiability algorithm for a restricted circuit class called ACC, implying that NEXP does not have polynomial-size ACC circuits [54]. Recently, it was shown that $\mathsf{NEXP} \not\subset \mathsf{P}/\mathrm{poly}$ is equivalent to establishing a "weak" form of natural proofs [55], building on Impagliazzo, Kabanets, and Wigderson. In particular, $\mathsf{NEXP} \not\subset \mathsf{P}/\mathrm{poly}$ if and only if there is a "constructive" property of Boolean functions that is "useful" against P/poly with $O(\log n)$ bits of advice.[1] Hence, assuming strong cryptography is possible, NEXP lower bounds must somehow confront the framework of natural proofs but sidestep the "large" condition.

To continue progress on circuit lower bounds for NEXP, it is imperative to understand algorithms for analyzing circuits, such as algorithms for circuit satisfiability, evaluating a circuit on all 2^n inputs, and approximating the acceptance probability of a circuit. (Recent surveys on these issues include [52, 46, 10, 37].) In this paper, we make this sort of algorithmic progress for circuits with arbitrary *linear threshold* gates: such a gate outputs 1 if and only if a certain linear inequality $\sum_i w_i x_i \geq t$ is true, where $w_i, t \in \mathbb{Z}$ are *weights* and $x_i \in \{0,1\}$ are inputs to the gate. Linear threshold functions have been studied for decades, coinciding with research on neural networks [32, 33]. Low-depth linear threshold circuits are powerful: many basic functions in arithmetic, algebra, and cryptography are known to be implementable

[1] The natural proofs barrier [41] states that if such a property is also "large" (true of a large fraction of functions) then strong cryptographic pseudorandom generators do not exist.

with only *constant-depth* linear threshold circuits [45, 48, 49, 30, 35]. In terms of lower bounds for such circuits, very weak questions remain major open problems: for example, is all of NEXP solvable with polynomial-size depth-*two* linear threshold circuits with exponential-size weights? (Note that for thresholds with polynomially-bounded weights, depth-two threshold lower bounds are known; however depth-three lower bounds are still open. The survey of Razborov [40] is still relatively current on these points.) Depth-two threshold circuits correspond to *multilayer perceptrons* with only one hidden layer. Despite considerable study in neural networks and deep learning, we still lack understanding of the power of depth-two.

In this paper, we report some new progress on understanding the power of linear threshold gates.

Algorithms and lower bounds for ACC with threshold gates. A MODm gate outputs 1 if and only if the sum of its input bits is divisible by m. Let $\mathsf{ACC} \circ \mathsf{THR}$ denote the class of circuits consisting of AND, OR, MODm gates for some constant m, and linear threshold gates, with unbounded fan-in and constant depth, such that the inputs of all linear threshold gates connect directly to the circuit's input variables. Let $\mathsf{SYM} \circ \mathsf{ACC} \circ \mathsf{THR}$ be the class of circuits where the output gate computes an arbitrary symmetric function, and its inputs connect to the outputs of $\mathsf{ACC} \circ \mathsf{THR}$ circuits. We show that such circuits can be very efficiently evaluated on all 2^n inputs, even if they are of $2^{n^{o(1)}}$ size.

Theorem 1.1. *Given a $\mathsf{SYM} \circ \mathsf{ACC} \circ \mathsf{THR}$ circuit with n inputs and $2^{n^{o(1)}}$ size, we can produce its outputs on all 2^n inputs in $2^n \cdot \mathrm{poly}(n)$ time.*[2] *More generally, such a circuit of size s can be evaluated on all inputs in $2^n \cdot \mathrm{poly}(\log s, n) + 2^{O(\log s)^c}$ time, for some $c \geq 1$ depending on the depth of the circuit and the modulus m of its* MODm *gates.*

The proof of Theorem 1.1 also carries through for $\mathsf{SYM} \circ \mathsf{ACC} \circ \mathsf{SYM}$ circuits, where the bottom-layer gates compute arbitrary symmetric functions (i. e., functions which only depend on the number of true inputs) of $2^{n^{o(1)}}$ wires. The algorithm of Theorem 1.1 can be used to *count* the satisfying assignments to $\mathsf{ACC} \circ \mathsf{THR}$ circuits:

Theorem 1.2. *For every integer $m > 1$ and $d > 0$, there is an $\varepsilon > 0$ such that counting satisfying assignments to $\mathsf{ACC} \circ \mathsf{THR}$ circuits with* MODm *gates of size $2^{n^{\varepsilon}}$ and depth d can be done in $2^{n-n^{\varepsilon}}$ time.*

By modifying prior arguments [54], we can conclude lower bounds for such circuits. The new argument shows that the ability to count SAT assignments entails non-uniform lower bounds for circuit classes with very weak closure properties.

Theorem 1.3. NEXP *does not have non-uniform $\mathsf{ACC} \circ \mathsf{THR}$ circuits of quasi-polynomial size.*

As Theorem 1.1 also holds for $\mathsf{SYM} \circ \mathsf{ACC} \circ \mathsf{SYM}$, it follows that NEXP doesn't have $\mathsf{ACC} \circ \mathsf{SYM}$ circuits of quasi-polynomial size as well.

Twenty years ago, Maciel and Thérien [28] considered lower bounds for $\mathsf{AC}^0 \circ \mathsf{MAJ}$ circuits (which $\mathsf{ACC} \circ \mathsf{THR}$ subsumes), but nontrivial lower bounds have not been reported. Regan [44] studied $\mathsf{MOD}_2 \circ \mathsf{AND} \circ \mathsf{THR}$ circuits and also noted the absence of lower bounds. Lower bounds have been open even for the much weaker class $\mathsf{AND} \circ \mathsf{OR} \circ \mathsf{MAJ}$ [19].

[2] Throughout the paper, we use the notation "poly(n)" to denote arbitrary (but fixed) polynomial factors of n.

Theorem 1.3 moves a little closer to an "unconditional break" of the natural proofs barrier [41]. More precisely, it appears plausible that pseudorandom functions are implementable with $\mathsf{ACC} \circ \mathsf{THR}$ circuits, in which case all lower bounds proved against such circuits must be non-naturalizing. We should note that it is not completely settled whether the proof that $\mathsf{NEXP} \not\subset \mathsf{ACC}$ is "truly" non-naturalizing; it could be that the natural proofs barrier is irrelevant to the problem. (If pseudorandom functions cannot be implemented in ACC, then natural proofs considerations don't apply to ACC anyway; if such functions can be implemented in ACC, then the NEXP lower bound is indeed non-naturalizing.) Plaku [38] has observed that the Naor-Reingold family of pseudorandom functions [35] can be implemented with quasi-polynomial size $\mathsf{OR} \circ \mathsf{THR} \circ \mathsf{AND}$ circuits; it follows that the natural proofs barrier already applies to this circuit class. It is an interesting open problem if $\mathsf{ACC} \circ \mathsf{THR}$ can efficiently simulate such depth-three circuits.

Faster integer linear programming. Building on Theorem 1.1, we also give a new method for solving 0-1 integer linear programs (ILP). Impagliazzo, Paturi, and Schneider [23] showed that for each $c > 1$, there is a $\delta < 1$ such that 0-1 ILP with cn constraints can be solved in $2^{\delta n}$ time. Impagliazzo, Lovett, Paturi, and Schneider [21] have recently shown that such ILPs can be solved in $2^{n(1-\Omega(\mathrm{poly}(1/c)))}$ time. We provide an improvement over exhaustive search for up to subexponentially many constraints:

Theorem 1.4. *Every 0-1 integer linear program with n variables and s constraints can be solved in time $2^{n-\Omega(n/\log^4(sM(\log n)))} \cdot \mathrm{poly}(s,n,M)$ with high probability, where $M \le 2^{n^{o(1)}}$ is an upper bound on bit-length of the coefficients in the program.*

Notice that the theorem allows for enormous coefficients, of size up to $2^M \le 2^{2^{o(n)}}$. The time bound compares favorably with the AC^0 circuit satisfiability algorithm of Impagliazzo, Matthews, and Paturi [22]: there, the authors use random restriction methods to solve satisfiability of AC^0 circuits with depth d and size s in $2^{n-n/(\log s)^{O(d)}}$ randomized time with zero error.

Depth-two linear threshold circuit evaluation. We take an important step towards depth-two linear threshold circuit (also known as $\mathsf{THR} \circ \mathsf{THR}$) lower bounds for the case of exponential weights, by giving an efficient algorithm for evaluating such circuits on all possible assignments.

Theorem 1.5. *Let $k > 1$. Given a depth-two $2^{n/24}$-size linear threshold circuit C with integer weights in $[-2^{n^k}, 2^{n^k}]$, we can evaluate C on all 2^n input assignments in $2^n \cdot \mathrm{poly}(n^k)$ time.*

Theorem 1.5 follows from a more general result showing that any sufficiently large "combinatorial rectangle" of inputs can be evaluated in $\mathrm{poly}(n)$ amortized time per input. Noting that a similar statement for evaluating ACC circuits forms the heart of the proof of $\mathsf{NEXP} \not\subset \mathsf{ACC}$ [54], Theorem 1.5 suggests that large complexity classes (such as NEXP) cannot have small depth-two linear threshold circuits.

However, we do not yet know how to turn Theorem 1.5 into depth-two linear threshold lower bounds; the precise difficulty is the following. The current known theorems connecting circuit evaluation algorithms to circuit lower bounds require that, from the OR of a collection of circuits, we can generate an equivalent circuit in the same class. We do not know how to convert a large OR of $\mathsf{THR} \circ \mathsf{THR}$ circuits into an equivalent $\mathsf{THR} \circ \mathsf{THR}$ circuit, even assuming NEXP has small $\mathsf{THR} \circ \mathsf{THR}$ circuits. (In the case of ACC, we trivially have $\mathsf{OR} \circ \mathsf{ACC} = \mathsf{ACC}$, because an OR of ACC circuits is still an ACC circuit.)

1.1 Prior work

Considerable effort has been expended in proving lower bounds against circuits with linear threshold gates. Here we will provide some major highlights, in addition to the work already mentioned.

It will help to introduce a little (standard) notation. Define MAJ, AND, OR, THR, and SYM to be the class of one-gate circuits corresponding to MAJORITY, AND, OR, linear threshold, and symmetric functions, respectively, with "free" NOT gates that can appear after the output or on the input wires to the gate. (Recall that a symmetric Boolean function's output only depends on the number of true inputs.) For classes of circuits $\mathcal{C}$ and $\mathcal{D}$, define $\mathcal{C} \circ \mathcal{D}$ to be the class of circuits formed by taking a circuit $C \in \mathcal{C}$, and feeding the outputs of circuits from $\mathcal{D}$ as inputs to C. That is, $\mathcal{C} \circ \mathcal{D}$ is simply the composition of circuits from $\mathcal{C}$ and $\mathcal{D}$, with the circuits from $\mathcal{D}$ receiving the input and the circuit from $\mathcal{C}$ giving the output. We will refer to the *size* of a circuit as the number of wires, i. e., the number of directed arcs in the DAG defining the circuit. This is an important measure for circuits with symmetric gates, as the number of wires governs the size of the symmetric function representation.

Much work on depth-two threshold lower bounds has concentrated on lower bounds for inner product modulo 2, i. e., $\mathrm{IP2}(x_1,\ldots,x_n,y_1,\ldots,y_n) = \sum_i x_i \cdot y_i \bmod 2$. Note that IP2 is easy for ACC (being a MOD2 of AND gates). In groundbreaking work, Hajnal et al. [15] proved that every $\mathsf{MAJ} \circ \mathsf{MAJ}$ circuit requires $2^{\Omega(n)}$ gates to compute IP2. They also showed $\mathsf{MAJ} \circ \mathsf{SYM}$ circuits can be efficiently simulated by $\mathsf{MAJ} \circ \mathsf{MAJ}$ circuits, so small $\mathsf{MAJ} \circ \mathsf{SYM}$ circuits also cannot compute IP2. Nisan [36] extended the lower bound to $\mathsf{MAJ} \circ \mathsf{THR}$ circuits, and Forster et al. [12] extended the lower bound to $\mathsf{THR} \circ \mathsf{MAJ}$ circuits. More recently, Sherstov [47] showed that AC^0 requires exponential-size $\mathsf{MAJ} \circ \mathsf{MAJ}$ circuits, Razborov and Sherstov [42] proved that depth-three AC^0 requires exponential-size $\mathsf{MAJ} \circ \mathsf{THR}$ circuits, and Beame and Huynh [4] showed that AC^0 requires $n^{\Omega(\log n)}$-size $\mathsf{MAJ} \circ \mathsf{SYM} \circ \mathsf{AND}$ circuits.

Although superpolynomial-size lower bounds against $\mathsf{MAJ} \circ \mathsf{AC}^0$, $\mathsf{THR} \circ \mathsf{AC}^0$, $\mathsf{MAJ} \circ \mathsf{MAJ} \circ \mathsf{AND}$ and even $\mathsf{MAJ} \circ \mathsf{MAJ} \circ \mathsf{AC}^0$ circuits are known [2, 13, 43, 17], and many lower bounds are known for AC^0 circuits augmented with a small number of threshold gates [5, 3, 9, 51, 16, 14, 27, 39], lower bounds for $\mathsf{AC}^0 \circ \mathsf{MAJ}$ circuits have remained open. Maciel and Thérien [28] conjectured that the majority-of-majority function is not in $\mathsf{AC}^0 \circ \mathsf{MAJ}$.

Recently, Hansen and Podolskii [19] have shown an intriguing reduction: superpolynomial-size $\mathsf{THR} \circ \mathsf{THR}$ lower bounds for a function f would follow from superlogarithmic lower bounds on the 3-party NOF unbounded-error communication complexity of f.

1.2 Comparison and intuition

It is instructive to discuss how this paper's approach relates to prior work on depth-two threshold lower bounds. A certain popular approach [12, 26, 47, 42] applies ingredients from Fourier analysis of Boolean functions, linear algebra, communication complexity, discrepancy theory, *etc.* In particular, these works follow the general scheme:

1. Define some notion of "relaxed rank" of a $2^{n/2} \times 2^{n/2}$ Boolean matrix C. Intuitively, if C has "relaxed rank" r, then there are $2^{n/2} \times r$ and $r \times 2^{n/2}$ matrices A and B such that the entries of $A \cdot B$ correspond to the entries of C in a direct way.
2. Show that every function $f : (\{0,1\}^{n/2} \times \{0,1\}^{n/2}) \to \{0,1\}$ computable with a "small" $\mathcal{C}$ circuit has "small relaxed rank" when construed as an $2^{n/2} \times 2^{n/2}$ Boolean matrix.

3. Show that some explicit family of functions $g_n : (\{0,1\}^{n/2} \times \{0,1\}^{n/2}) \to \{0,1\}$, construed as $2^{n/2} \times 2^{n/2}$ Boolean matrices, requires "high relaxed rank" asymptotically.

Together, these steps prove that the family $g := \{g_n\}$ cannot have "small" $\mathcal{C}$ circuits.

To prove $\mathsf{ACC} \circ \mathsf{THR}$ circuit lower bounds, we define a generalized rank notion we call the *symmetric rank*, informally measuring how efficiently a 0-1 matrix M can be decomposed into a sum of rank-one matrices such that, after applying a fixed symmetric function to each entry of the sum, we obtain the matrix M. Combining several elements from previous work, we show that for a Boolean matrix representing the truth table of a $\mathsf{SYM} \circ \mathsf{ACC} \circ \mathsf{THR}$ circuit of size s, its symmetric rank is $O(2^{\log^c s})$ for some constant $c \geq 1$, depending on the depth d and modulus m of the MODm gates in the circuit. Moreover, given such a circuit we can efficiently compute a low-rank decomposition.

However, we do not know how to use existing methods to prove that an explicit function g has high symmetric rank. Instead, we take a more *computational* approach that still exploits the low symmetric rank property. The idea is that, if we can efficiently compute a low-rank decomposition from a given circuit, then the circuit's truth table can be obtained faster than evaluating the circuit on all its inputs one-by-one. This in turn suggests that these circuits possess considerable structure that make them unsuitable for simulating very complex functions, such as those in NEXP.

Suppose we are given an $\mathsf{SYM} \circ \mathsf{ACC} \circ \mathsf{THR}$ circuit C of size s with n inputs. Let M be a $2^{n/2} \times 2^{n/2}$ matrix defining the function computed by C. First we show how given any such C we can compute $2^{n/2} \times 2^{\log^c s}$ and $2^{\log^c s} \times 2^{n/2}$ matrices A and B (and a symmetric function f) giving a symmetric rank decomposition of M, in $2^{n/2} \cdot 2^{O(\log^c s)}$ time. By multiplying A and B and applying f to each entry of the output matrix, we can obtain M. When s is sufficiently small, a rectangular matrix multiplication of Coppersmith [11] can be applied to compute the product of A and B, and the final matrix M is obtained in $\mathrm{poly}(n)$ time per entry. Hence, given an $\mathsf{SYM} \circ \mathsf{ACC} \circ \mathsf{THR}$ circuit C of size $2^{n^{o(1)}}$, we can evaluate C on all its 2^n inputs in only $2^n \cdot \mathrm{poly}(n)$ time. This fast evaluation algorithm is combined with prior work [53, 54] along with some new tricks to exhibit a $g := \{g_n\} \in \mathsf{NEXP}$ which does not have quasipolynomial-size $\mathsf{ACC} \circ \mathsf{THR}$ circuits.

Our evaluation algorithm for depth-two threshold circuits (Theorem 1.5) also uses Coppersmith's rectangular matrix multiplication as a subroutine, but the rest of the algorithm is rather different from the evaluation algorithm for $\mathsf{SYM} \circ \mathsf{ACC} \circ \mathsf{THR}$. We reduce the problem of efficiently evaluating a depth-two threshold circuit on many inputs to a special type of matrix multiplication. Namely, for two matrices A and B over the integers, we compute a "weighted" matrix product

$$C[i,j] = \sum_k w_k \cdot \mathrm{LEQ}(A[i,k], B[k,j]),$$

where $\mathrm{LEQ}(x,y)$ is a Boolean-valued function equal to 1 if and only if $x \leq y$, and the w_k's are arbitrary integer weights given as parameters to the problem. We show how Coppersmith's algorithm can be combined with a mild brute force search to efficiently compute a rectangular matrix product of the above form.

2 Algorithms and lower bounds for ACC with a layer of threshold gates

The main result of this section is Theorem 1.1.

Reminder of Theorem 1.1. *Given a* $\mathsf{SYM} \circ \mathsf{ACC} \circ \mathsf{THR}$ *circuit with* n *inputs and* $2^{n^{o(1)}}$ *size, we can produce its outputs on all* 2^n *inputs in* $2^n \cdot \mathrm{poly}(n)$ *time. More generally, such a circuit of size* s *can be evaluated on all inputs in* $2^n \cdot \mathrm{poly}(\log s, n) + 2^{O(\log s)^c}$ *time, for some* $c \geq 1$ *depending on the depth of the circuit and the modulus* m *of its* MODm *gates.*

Depth reduction. The first stage of the proof is to convert an arbitrary $\mathsf{SYM} \circ \mathsf{ACC} \circ \mathsf{THR}$ circuit C of size s into a depth-two circuit C'' of symmetric gates, i. e., a $\mathsf{SYM} \circ \mathsf{SYM}$ circuit. The size of the depth-two circuit will be $O(2^{\log^c s})$ for a constant $c \geq 1$, depending on the (constant) depth and (constant) modulus of circuit C. This stage requires several different pieces from prior work.

Lemma 2.1. *There is an algorithm which given an* $\mathsf{SYM} \circ \mathsf{ACC} \circ \mathsf{THR}$ *circuit* C *of size* $s \geq n$*, depth* d*, and* MODm *gates, outputs an equivalent* $\mathsf{SYM} \circ \mathsf{SYM}$ *circuit* C'' *with at most* $2^{(\log s)^c}$ *wires, and runs in time* $O(2^{(\log s)^c})$*, for* $c \geq 1$ *depending only on* d *and* m*.*

In the proof, several constants arise; we will denote all of them by the same constant b which is assumed to be the maximum of these quantities.

Before we begin the proof, let us make a few remarks about the threshold gates of C. It is well known that every THR function is equivalent to a THR function where the weights have absolute value at most $2^{bn\log_2 n}$ [34, 33]. That is, every THR function is equivalent to one with weights of bit-length at most $bn\log_2 n$. In the following, we shall assume the THR gates of C have this property; this assumption is certainly valid for proving the lower bounds of Theorem 1.3, and the weights are dealt with explicitly in our algorithms, such as Theorems 1.4 and 3.1.[3]

The following paragraphs give the proof of Lemma 2.1. Let C be a $\mathsf{SYM} \circ \mathsf{ACC} \circ \mathsf{THR}$ circuit with inputs $x_1, \ldots, x_n$, size s, depth d, and MODm gates, for constants $d > 2$ and $m > 1$. The first step in Lemma 2.1 is to translate the THR layer of C into a SYM layer, by absorbing some of its complexity into the ACC part. Maciel and Thérien [29] provided several fairly tight low-depth circuits for various tasks. We need the following result ([29, Theorem 3.3]).

Theorem 2.2 (Maciel–Therien)**.** *Addition of* n *distinct* n*-bit numbers can be performed with polynomial-size* $\mathsf{AND} \circ \mathsf{OR} \circ \mathsf{SYM}$ *circuits. Furthermore, the circuits can be constructed in polynomial time.*

We can therefore replace every THR gate of C with an $\mathsf{AC}^0 \circ \mathsf{MAJ}$ circuit, as follows. Fix a threshold gate of C, with weights $w_{i_1}, \ldots, w_{i_t}$ for $t \leq n$, computing

$$\sum_{j=1}^{t-1} w_{i_j} x_{i_j} \geq w_{i_t}$$

for some $i_j \in \{1, \ldots, n\}$. Note $|w_{i_j}| \leq 2^{bn\log_2 n}$ for $j = 1, \ldots, t$. Set $W = bn\log_2 n$.

[3] The "small-weight" representation of a THR function can be obtained from a given linear form and threshold value, by evaluating the linear form representation at $n+1$ (linearly independent) points in $\{0,1\}^n$, then solving a linear system in $n+1$ variables to determine the new weights and new threshold value. (See [34], Theorem 16.) However, determining *which* $n+1$ points in $\{0,1\}^n$ to evaluate the THR function on appears to require solving Knapsack problems (finding the "minimal-terms" of the THR function); the best algorithm the author knows for explicitly computing such weights takes $2^{O(n)}$ time. Nevertheless, this determination of small weights is not required for any of our applications.

Let D be a circuit for the addition of $t-1$ W-bit numbers, provided by Theorem 2.2. For $j = 1,\ldots,t-1$, we connect to the jth W-bit input of D a circuit which, given x_{i_j}, feeds w_{i_j} to D if the input bit $x_{i_j} = 1$, and the all-zero W-bit string if $x_{i_j} = 0$. Note this extra circuit actually contains no gates: it simply has a wire from x_{i_j} to all bits of the jth W-bit input where the corresponding bit of w_{i_j} equals 1. Letting this new circuit be D', we have

$$D'(x_1,\ldots,x_n) = \sum_{j=1}^{t-1} w_{i_j} x_{i_j}\,.$$

This can be compared to the value w_{i_t} with an AC^0 circuit, using the fact that the "less-than-or-equal-to" comparison of two integers can be performed in AC^0 [8]. We now have an $\mathsf{AC}^0 \circ \mathsf{SYM}$ circuit D'' of size $\mathrm{poly}(W,t) \leq n^b$ computing the given threshold gate. Applying this construction to each threshold gate in the THR layer of C, we obtain an $\mathsf{SYM} \circ \mathsf{ACC} \circ \mathsf{SYM}$ circuit C' of size at most $s \cdot n^b$.

The next step of Lemma 2.1 is to convert the $\mathsf{SYM} \circ \mathsf{ACC}$ part into a $\mathsf{SYM} \circ \mathsf{AND}$ circuit, using a reduction of Beigel–Tarui [6] (with important details on constructibility filled in by Allender–Gore [1]).

Theorem 2.3 (Beigel–Tarui, Allender–Gore)**.** *Every* $\mathsf{SYM} \circ \mathsf{ACC}$ *circuit of size s can be simulated by a* $\mathsf{SYM} \circ \mathsf{AND}$ *circuit of* $2^{(\log s)^{c'}}$ *size for some constant c' depending only on the depth d and* MOD*m gates of the* ACC *part. Moreover, the* AND *gates of the final circuit have only* $(\log s)^{c'}$ *fan-in, the final circuit can be constructed from the original in* $2^{O((\log s)^{c'})}$ *time, and the final symmetric function at the output can be computed in* $2^{O((\log s)^{c'})}$ *time.*

Applying this reduction to the top $\mathsf{SYM} \circ \mathsf{ACC}$ part of the circuit C' results in an equivalent $\mathsf{SYM} \circ \mathsf{AND}_{(\log(s \cdot n^b))^{c'}} \circ \mathsf{SYM}$ circuit C'' of size $s' = 2^{O((\log(s \cdot n^b))^{c'})}$ (where the subscript on the AND denotes the fan-in of each AND gate). For simplicity of notation, let $t = (\log(s \cdot n^b))^{c'}$ in the following.

Extending a trick of Beigel [5] to symmetric gates, we can convert every $\mathsf{AND}_t \circ \mathsf{SYM}$ subcircuit of C'' with n^b wires into a single SYM gate with $O(n^{b \cdot t})$ wires. Let $S_1(x_1,\ldots,x_n) \wedge \cdots \wedge S_t(x_1,\ldots,x_n)$ be one such subcircuit, where S_i denotes the ith symmetric gate. In particular, for $i = 1,\ldots,t$, let $f_i : \mathbb{Z} \to \{0,1\}$ be such that

$$f_i\left(\sum_{j=1}^{n} c_{i,j} x_j\right) = S_i(x_1,\ldots,x_n)\,,$$

where $c_{i,j}$ denotes the number of copies of x_j that feed into S_i.

Let $B = 1 + \max_i(\sum_{j=1}^{n} c_{i,j})$; note that $B \leq n^b$. Consider the linear form

$$L(x_1,\ldots,x_n) = \sum_{i=1}^{t} B^{i-1} \cdot \left(\sum_{j=1}^{n} c_{i,j} x_j\right).$$

For any Boolean assignment to the x_j's, the number encoded by the linear form $L(x_1,\ldots,x_n)$ is an integer encoded in $O(t \cdot b \log n)$ bits. By construction, the bit representation of this integer contains, for every $i = 1,\ldots,t$, the number of wires input to S_i which are set true, as a string of $(b \log n)$ bits. Therefore, from the linear form $L(x_1,\ldots,x_n)$ we can easily infer whether all $S_i(x_1,\ldots,x_n)$ output 1 or not, and hence output the value of $S_1 \wedge \cdots \wedge S_t$.

To implement this linear form with a single SYM gate, for all $j = 1, \ldots, n$ we put

$$\sum_{i=1}^{t} B^{i-1} c_{i,j}$$

wires from the input variable x_j into the new SYM gate. Hence there are $O(n^{b \cdot t})$ wires from the inputs into this new SYM gate. By choosing the appropriate symmetric function (which outputs 1 if and only if $L(x_1, \ldots, x_n)$ encodes a number such that $S_1 \wedge \cdots \wedge S_t$ is true) we can simulate any $\mathsf{AND}_t \circ \mathsf{SYM}$ circuit of n^b wires with a single SYM gate of $O(n^{b \cdot t})$ wires.

Replacing each $\mathsf{AND} \circ \mathsf{SYM}$ subcircuit in this manner results in a $\mathsf{SYM} \circ \mathsf{SYM}$ circuit of size $O(s' \cdot n^{b \cdot t}) \leq 2^{O(\log s)^c}$ for some constant $c \geq 1$. This concludes the proof of Lemma 2.1.

Symmetric rank. Next, we prove that the truth table of any $\mathsf{SYM} \circ \mathsf{SYM}$ circuit C'' of t wires and n inputs represents a $2^{n/2} \times 2^{n/2}$ matrix of *symmetric rank* at most $\mathrm{poly}(t)$, and this rank decomposition can be efficiently computed. For given matrices A and B over the integers, let $A \cdot B$ denote their matrix product over the integers. Let $M \in \{0,1\}^{m \times n}$. We define the *symmetric rank of M* to be the minimum $r \in \mathbb{N}$ such that there are matrices $A \in \{0,1\}^{m \times r}$, $B \in \{0,1\}^{r \times n}$ and a function $f : \{0,1,\ldots,r\} \to \{0,1\}$ satisfying $M[i,j] = f((A \cdot B)[i,j])$ for all i, j. We call the triple (A, B, f) a *symmetric rank decomposition* of M. The symmetric rank is similar to the typical notion of rank, except for the additional function f providing a "filter" from arbitrary integers back to $\{0,1\}$. This filter function could potentially lead to smaller rank decompositions than the typical notion. However, note that the symmetric rank of M is not necessarily at most (for instance) the rank of M over $\mathbb{R}$, because A and B are required to have Boolean entries.

For simplicity let n be even, and let $z_1, \ldots, z_{2^{n/2}}$ be the list of all $2^{n/2}$ $n/2$-bit strings in lexicographical order. For a circuit C with n inputs, define the *truth table matrix* M_C to be the $2^{n/2} \times 2^{n/2}$ matrix with $M_C[i,j]$ equal to the output of $C(z_i, z_j)$.

Lemma 2.4. *Given a $\mathsf{SYM} \circ \mathsf{SYM}$ circuit C with t wires and n inputs, its truth table matrix M_C has symmetric rank $O(t^3)$, and a symmetric rank decomposition of M_C can be computed from C in $2^{n/2} \cdot \mathrm{poly}(t)$ time.*

Proof. For simplicity we assume n is even; the case of odd n will be apparent. Index the input variables of C by $x_1, \ldots, x_n$. Let $g_1, \ldots, g_s$ be an indexing of the gates of C on the bottom layer (closest to the inputs) and let g' denote the output gate of C. (Note that $s \leq t$.) Let $f : \{0,1,\ldots,s\} \to \{0,1\}$ be the symmetric function of gate g': for all $a \in \{0,1,\ldots,s\}$, $f(a) = b$ if and only if a true inputs make g' output b.

We shall show how to efficiently construct matrices A and B with the appropriate properties. Let $z_1, \ldots, z_{2^{n/2}}$ be the list of all $n/2$-bit strings in lexicographical order, in the following. For every pair $(a,b) \in \{0,1,\ldots,t\}^2$ such that $a + b \leq t$, let $S_{a,b} \subseteq \{g_1, \ldots, g_s\}$ denote the subset of gates g_j such that $a + b$ true inputs makes gate g_j output 1.

The matrices A and B to be constructed show that the symmetric rank of M_C is at most

$$r = \sum_{\substack{a,b \in \{0,1,\ldots,t\} \\ a+b \leq t}} |S_{a,b}| \leq O(t^3)\,.$$

In other words, each pair (a,b) will add $|S_{a,b}|$ additional components to the rows of A and to the columns of B, increasing the lengths of all rows of A (and columns of B) by $|S_{a,b}|$.

For $i = 1, \ldots, 2^{n/2}$, the ith row of A and ith column of B are defined as follows. For every pair (a,b), allocate $|S_{a,b}|$ additional components for the rows of A and columns of B.

For $j = 1, \ldots, |S_{a,b}|$, put a 1 in the jth additional component of the ith row of A if and only if there are a true wires going into the jth gate of $S_{a,b}$ when the input variables $x_1, \ldots, x_{n/2}$ are given assignment z_i. That is, the jth component is 1 if and only if the contribution (from the first half of variables) to the overall sum for the jth gate is a.

Similarly, for $j = 1, \ldots, |S_{a,b}|$, put a 1 in the jth additional component of the ith column of B if and only if there are b true wires going into the jth gate of $S_{a,b}$ when the input variables $x_{n/2+1}, \ldots, x_n$ are given assignment z_i.

Note that each entry of A and B can be determined in $\text{poly}(t)$ time.

For every fixed (a,b), the product of two jth components for the ith row of A and the kth column of B is either 0 or 1, and the product is 1 if and only if:

- the sum of true inputs into the jth gate of $S_{a,b}$ from the inputs $(x_1, \ldots, x_{n/2})$ equals a when the inputs $(x_1, \ldots, x_{n/2})$ are assigned z_i,
- the sum of true inputs into the same gate from $(x_{n/2+1}, \ldots, x_n)$ equals b when the inputs $(x_{n/2+1}, \ldots, x_n)$ are assigned z_k, and
- the jth gate outputs 1 when its sum of true inputs equals $a + b$.

It follows that the *inner product* of the ith row of A and the kth column of B equals the total number $N_{i,k}$ of true wires going into the output gate of C on the variable assignment $(x_1, \ldots, x_n) \mapsto (z_i, z_k)$. By definition, $f(N_{i,k})$ equals the output of C on that variable assignment. $\square$

We need one more lemma to complete the proof of Theorem 1.1.

Lemma 2.5. *For all sufficiently large N, and $\alpha \le .172$, multiplication of an $N \times N^{\alpha}$ matrix with an $N^{\alpha} \times N$ matrix can be done in $N^2 \cdot \text{poly}(\log N)$ arithmetic operations, over any field with $O(2^{\text{poly}(\log N)})$ elements.*

A short exposition of Lemma 2.5 can be found in Appendix C of [54].

Proof of Theorem 1.1. Given a $\mathsf{SYM} \circ \mathsf{ACC} \circ \mathsf{THR}$ circuit C and size s, convert C into a $\mathsf{SYM} \circ \mathsf{SYM}$ circuit C'' of $2^{(\log s)^c}$ size using Lemma 2.1. Compute a symmetric rank decomposition of C into $2^{n/2} \times 2^{3(\log s)^c}$ and $2^{3(\log s)^c} \times 2^{n/2}$ 0-1 matrices A and B respectively, along with a function $f : [2^{3(\log s)^c}] \to \{0,1\}$. Compute the product of A and B in $2^n \cdot \text{poly}(\log s, n)$ time, using Lemma 2.5. Finally, evaluate function f on all entries of the matrix product. This can be done by numerically sorting the entries, replacing each entry v by $f(v)$, then inverting the sorted order, in time $2^n \cdot \text{poly}(\log s, n) + 2^{O(\log s)^c}$. For $s \le 2^{n^{o(1)}}$, the runtime is $2^n \cdot \text{poly}(n)$. $\square$

2.1 Counting satisfying assignments to ACC of linear thresholds

The evaluation algorithm of Theorem 1.1 is quite powerful, substantially extending the class of circuits for which we can perform non-trivial circuit analysis.

Reminder of Theorem 1.2. *For every $m > 1$ and $d > 0$, there is an $\varepsilon > 0$ such that counting satisfying assignments to $\mathsf{ACC} \circ \mathsf{THR}$ circuits of size $2^{n^{\varepsilon}}$, depth d, and MODm gates can be done in $2^{n - n^{\varepsilon}}$ time.*

Proof. For all $k \in \mathbb{N}$ and for $i = 1, \dots, 2k$, define a symmetric function Bit_i^k with 2^{2k} inputs as follows: for all $i = 1, \dots, 2k$, Bit_i^k outputs the ith bit of the sum of its input bits.

Suppose we are given an $\mathsf{ACC} \circ \mathsf{THR}$ circuit C of size s and n inputs, and we wish to count its satisfying assignments. Let $\ell < n/2$ be a parameter to set later. For every assignment $A_j \in \{0,1\}^{2\ell}$ to the last 2ℓ inputs of C, make a copy of C with the assignment A_j plugged into those 2ℓ inputs, calling this copy C_{A_j}. Note that each C_{A_j} has (the same) $n - 2\ell$ inputs $x_1, \dots, x_{n-2\ell}$.

For every $i = 1, \dots, 2\ell$, define

$$B_i(x_1, \dots, x_{n-2\ell}) := \mathrm{Bit}_i^{\ell}(C_{A_1}(x_1, \dots, x_{n-2\ell}), \dots, C_{A_{2^{2\ell}}}(x_1, \dots, x_{n-2\ell}))\,.$$

Each function B_i can be implemented in $s' = 2^{2\ell} \cdot s$ size, as a $\mathsf{SYM} \circ \mathsf{ACC} \circ \mathsf{THR}$ circuit. Applying Theorem 1.1, B_i can be evaluated on all of its $2^{n-2\ell}$ possible assignments in time

$$2^{n-2\ell} \cdot \mathrm{poly}(n) + 2^{\mathrm{poly}(\log s')} \le 2^{n-2\ell} \cdot \mathrm{poly}(n) + 2^{\mathrm{poly}(\ell + \log s)}\,.$$

The above for-loop over all i produces $2\ell \cdot 2^{n-2\ell}$ bits: for each of the $2^{n-2\ell}$ partial assignments to $n - 2\ell$ variables, we learn the number (in 2ℓ bits) of partial assignments on the other 2ℓ variables which result in satisfaction. The number of all satisfying assignments is obtained by simply summing all 2ℓ-bit numbers obtained from the $2^{n-2\ell}$ assignments, in $2^{n-2\ell} \cdot \mathrm{poly}(\ell)$ time.

Letting $\ell = n^{\varepsilon}/2$ for sufficiently small $\varepsilon > 0$, we have a $2^{n-n^{\varepsilon}}$ time algorithm. □

2.2 Faster 0-1 linear programming

$\mathsf{ACC} \circ \mathsf{THR}$ circuits are definitely powerful enough to simulate 0-1 integer linear programming; a straightforward application of Theorem 1.2 would yield a faster algorithm for the problem. However, the improvement over exhaustive search would be rather minor, and tedious to calculate. By modifying the proof of Theorem 1.1 in appropriate places, we can derive a better algorithm in this case.

Reminder of Theorem 1.4. *Every 0-1 integer linear program with n variables and s constraints can be solved in time $2^{n-\Omega(n/\log^4(sM(\log n)))} \cdot \mathrm{poly}(s,n,M)$ with high probability, where $M \le 2^{o(n)}$ is an upper bound on the bit-length of the coefficients in the program.*

Proof. Consider a 0-1 linear program of the form $Ax \le b$, along with a cost function $\langle c, x \rangle$ we wish to maximize, where $A \in \mathbb{Z}^{s \times n}$, $b \in \mathbb{Z}^s$, and $c \in ([-2^M, 2^M] \cap \mathbb{Z})^n$ by assumption on M. First, reduce the optimization problem to one of feasibility, in a standard way: include $\langle c, x \rangle \ge v$ as an additional constraint for various $v \in \mathbb{Z}$, and by binary searching on v, we maximize the value of v such that the $s+1$ constraint system remains feasible. Since the x_i are Boolean valued, the binary search uses at most $t := O(M + \log n)$ calls to feasibility questions.

Next, observe the feasibility questions can be viewed as a satisfiability question for a depth-two circuit D with an AND of fan-in s at the top gate, and linear threshold gates on the bottom layer, by directly translating each constraint in the program into a linear threshold gate. By Theorem 2.2 and the argument in Lemma 2.1, each threshold gate in the circuit D can be replaced with a polynomial-sized $\mathsf{LEQ} \circ \mathsf{AND} \circ \mathsf{OR} \circ \mathsf{SYM}$ circuit, where LEQ computes on t-bit integers a and b whether $a \le b$ (recall $t := O(M + \log n)$). As LEQ has an $\mathsf{XOR} \circ \mathsf{AND} \circ \mathsf{XOR}$ circuit of $O(t^2)$ size for t-bit inputs (see [8] for a reference), the satisfiability question for the circuit D reduces to the SAT question for an $\mathsf{AC}^0[2] \circ \mathsf{SYM}$ circuit C of size $\mathrm{poly}(M, \log n, s)$. In particular, the circuit has the form

$$\mathsf{AND} \circ \mathsf{XOR} \circ \mathsf{AND} \circ \mathsf{XOR} \circ \mathsf{AND} \circ \mathsf{OR} \circ \mathsf{SYM}\,.$$

Following the strategy of Theorem 1.2 (and the author's ACC SAT algorithm [54]), the satisfiability question for C with n inputs and size $\mathrm{poly}(M, \log n, s)$ can be converted into the problem of evaluating a larger $\mathsf{AC}^0[2] \circ \mathsf{SYM}$ circuit C', where C' has $n' = n-k$ inputs, $2^k \cdot \mathrm{poly}(M, \log n, s)$ size, $k < n/2$ is a parameter, and the entire $\mathsf{AC}^0[2]$ part of C' has the form

$$\mathsf{OR} \circ \mathsf{AND} \circ \mathsf{XOR} \circ \mathsf{AND} \circ \mathsf{XOR} \circ \mathsf{AND} \circ \mathsf{OR}\,.$$

More precisely, C' is the OR of 2^k copies of the circuit C, where each copy has its first k inputs assigned to a distinct string from $\{0,1\}^k$. Clearly, this circuit C' is satisfiable if and only if C is satisfiable.

Now we wish to evaluate C' on all 2^{n-k} inputs, efficiently. Rather than applying Beigel-Tarui at this point, as in Lemma 2.1, we instead apply the probabilistic polynomials of Smolensky [50] to convert C' into a $\mathsf{SYM} \circ \mathsf{SYM}$ circuit C''. In particular, we use a slight modification of Smolensky's construction, as described by Kopparty and Srinivasan [25].

Theorem 2.6 (Smolensky, Kopparty–Srinivasan)**.** *For every $\mathsf{AC}^0[2]$ circuit C of depth d, size z, and m inputs, and $\varepsilon > 0$, there is a distribution of n-variate polynomials $\mathcal{D}_C$ over $\mathbb{F}_2$ with the following properties. Each p with nonzero support in $\mathcal{D}_C$ has degree at most $(4\log z)^{d-1} \cdot (\log 1/\varepsilon)$, a polynomial p can be sampled from $\mathcal{D}_C$ in $m^{O(\log z)^{d-1}(\log 1/\varepsilon)}$ time, and for every $x \in \{0,1\}^m$,*

$$\Pr_{p \sim \mathcal{D}_C}[p(x) = C(x)] \geq 1 - \varepsilon\,.$$

In our case, the $\mathsf{AC}^0[2]$ parts of our subcircuits C_i have six different *layers*, where each layer has the same gate type. The degree in the above construction is in fact only $(4\log z)^{d^\star - 1} \cdot (\log 1/\varepsilon)$ where $d^\star$ is the number of layers which are AND or OR (the XOR layers do not contribute to the degree, because XOR can be directly simulated by summation over $\mathbb{F}_2$). Note that for each C_i we have $d^\star = 4$ (two layers are XORs).

We can therefore apply Theorem 2.6 as follows. Recall that C' is an OR of some $\mathsf{AC}^0[2] \circ \mathsf{SYM}$ circuits $C_1, \ldots, C_{2^k}$, each with (the same) $n-k$ inputs. Moreover, the top $\mathsf{AC}^0[2]$ part of each C_i has $z = \mathrm{poly}(M, \log n, s)$ size, and takes $\mathrm{poly}(M, \log n, s)$ inputs (coming from the outputs of SYM gates). For every i, we take the top $\mathsf{AC}^0[2]$ part of C_i, and invoke Theorem 2.6 with $\varepsilon = 1/(10 \cdot 2^k)$ to sample a polynomial $p_i \sim \mathcal{D}_{C_i}$ of degree at most $O(k(\log z)^3)$ having at most $\mathrm{poly}(s, M)^{O(k \cdot (\log z)^3)}$ monomials.

We replace the $\mathsf{AC}^0[2]$ part of C_i with the polynomial p_i construed as an XOR of ANDs circuit. Now the circuit C' is an OR of 2^k XOR of AND of SYM circuits; call them $C_1'', \ldots, C_{2^k}''$. For every input $x \in \{0,1\}^{n-k}$, the SYM gates of C' produce a single $\mathrm{poly}(s, M)$-bit length input y. Taking the union bound over all 2^k subcircuits, every $C_1'', \ldots, C_{2^k}''$ outputs the same values as $C_1, \ldots, C_{2^k}$ on x, with probability at least $1 - 1/10$, by our choice of ε.

Now we randomly convert the topmost OR in C' to an XOR, via the usual Razborov-Smolensky subsum trick: we pick $r_{1,1}, r_{2,1}, r_{1,2}, r_{2,2}, \ldots, r_{1,2^k}, r_{2,2^k} \in \{0,1\}$ uniformly at random, and replace $C =$

$\mathsf{OR}(C''_1, \ldots, C''_{2^k})$ with

$$\begin{aligned} C''(x_1,\ldots,x_{n-k}) &:= \left(\sum_{i=1}^{2^k} r_{1,i}\cdot C''_i(x_1,\ldots,x_{n-k}) \bmod 2\right) \vee \left(\sum_{i=1}^{2^k} r_{2,i}\cdot C''_i(x_1,\ldots,x_{n-k}) \bmod 2\right) \\ &= \sum_{i=1}^{2^k} r_{1,i}\cdot C''_i(x_1,\ldots,x_{n-k}) + \sum_{i=1}^{2^k} r_{2,i}\cdot C''_i(x_1,\ldots,x_{n-k}) \\ &\quad + \left(\sum_{i=1}^{2^k} r_{1,i}\cdot C''_i(x_1,\ldots,x_{n-k})\right)\cdot\left(\sum_{i=1}^{2^k} r_{2,i}\cdot C''_i(x_1,\ldots,x_{n-k})\right) \bmod 2\,, \end{aligned}$$

which means that C'' equals

$$\sum_{i=1}^{2^k} r_{1,i}\cdot C''_i(x_1,\ldots,x_{n-k}) + \sum_{i=1}^{2^k} r_{2,i}\cdot C''_i(x_1,\ldots,x_{n-k}) \\ + \sum_{i,j=1}^{2^k} r_{1,i}\cdot r_{2,j}\cdot C''_i(x_1,\ldots,x_{n-k})\cdot C''_j(x_1,\ldots,x_{n-k}) \bmod 2\,.$$

Now for every $x \in \{0,1\}^{n-k}$,

$$\begin{aligned} &\Pr_{p_i\sim\mathcal{D}, r_{i,j}\in\{0,1\}}[C''(x) \neq C'(x)] \\ &\quad \leq \Pr_{p_1,\ldots,p_{2^k}\sim\mathcal{D}_{C_i}}[\exists\, i, C''_i(x)\neq C_i(x)] + \Pr_{r_{i,j}\in\{0,1\}}[\mathsf{OR}(C''_1(x),\ldots,C''_{2^k}(x)) = C'(x) \mid \forall\, i, C''_i(x) = C_i(x)] \\ &\quad \leq 1/10 + 1/4 \leq 1/3\,. \end{aligned}$$

That is, for every input $x \in \{0,1\}^{n-k}$, the probability that $C'(x) = C''(x)$ will be greater than $2/3$.

Since each polynomial p_i has degree at most $O(k\cdot(\log z)^3)$, the AND gates representing the monomials of p_i have $f \leq O(k\cdot(\log z)^3)$ fan-in. Applying another part of Lemma 2.1, the $\mathsf{AND}_f \circ \mathsf{SYM}$ subcircuits of C'' with $\mathrm{poly}(s,M)$ wires can be replaced by a single SYM gate with $\mathrm{poly}(s,M)^{O(f)}$ input wires. This results in an $\mathsf{XOR}\circ\mathsf{SYM}$ circuit C'' of $\mathrm{poly}(s,M)^{O(k\cdot(\log z)^3)}$ total wires; note this is also a $\mathsf{SYM}\circ\mathsf{SYM}$ circuit.

Let $\varepsilon > 0$ be a parameter, and set

$$k := \max\left\{1, \frac{\varepsilon n}{\log^4(Ms(\log n))}\right\}.$$

(Note that if $k=1$, the statement of Theorem 1.4 is trivially true.) Following the proof of Theorem 1.1, we can apply fast rectangular matrix multiplication to evaluate C'' on all 2^{n-k} inputs. Recalling that $z = \mathrm{poly}(M, \log n, s)$, note that $(\log z) \leq O(\log(Ms(\log n)))$. So for sufficiently small $\varepsilon > 0$, the evaluation of C'' runs in time

$$2^{n-k}\cdot\mathrm{poly}(O(k\cdot(\log z)^3), \log M, n-k) + \mathrm{poly}(s,M)^{O(k\cdot(\log z)^3)} \leq 2^{n-\Omega(n/\log^4(sM(\log n)))}\cdot\mathrm{poly}(s,n,M)\,.$$

The output of this procedure is a 2^{n-k}-bit string which, for every $x \in \{0,1\}^{n-k}$, contains the correct output $C'(x)$ with probability at least $2/3$.

Suppose we repeat the above randomized procedure for n^2 times: that is, for n^2 times, we independently sample 2^k polynomials p_i for each C_i and sample $r_{i,j} \in \{0,1\}$, constructing n^2 different circuits $C''_1, \ldots, C''_{n^2}$ from C'. Then, standard tail bound arguments show that the majority value output by $C''_1(x), \ldots, C''_{n^2}(x)$ equals $C'(x)$ for every $x \in \{0,1\}^{n-k}$, with high probability. If some assignment $x^\star$ has majority value 1, we conclude that the integer program is *feasible*; otherwise, we output *infeasible*. $\square$

2.3 Non-uniform $\mathsf{ACC} \circ \mathsf{THR}$ lower bounds

We now turn to the main application of the evaluation algorithm:

Reminder of Theorem 1.3. NEXP *does not have non-uniform* $\mathsf{ACC} \circ \mathsf{THR}$ *circuits of quasi-polynomial size.*

To set the context, let us discuss the prior connection between known circuit satisfiability algorithms and circuit lower bounds.

Definition 2.7. Let $\mathcal{C}$ be a circuit class. $\mathcal{C}$ is said to be *typical* if, given any circuit D from one of the classes $\mathcal{C} \circ \mathcal{C}$, $\mathsf{AND} \circ \mathcal{C}$, $\mathsf{OR} \circ \mathcal{C}$, $\mathsf{NOT} \circ \mathcal{C}$, an equivalent $D' \in \mathcal{C}$ can be produced in $\text{poly}(\text{size}(D))$ time.

That is, $\mathcal{C}$ is typical if it is *efficiently closed under composition, unbounded fan-in* AND*,* OR*, and negations*. Most well-studied circuit classes have this property.

From prior work, we know there are connections between the existence of good SAT algorithms for typical circuit classes, and lower bounds against those classes:

Theorem 2.8 ([54])**.** *Let $\mathcal{C}$ be typical. Suppose for every $c \geq 1$, there is an $\varepsilon > 0$ and an algorithm for satisfiability of $\mathcal{C}$ circuits running in time $O(2^{n-n^\varepsilon})$ on circuits with n inputs and $n^{\log^c n}$ size. Then* NEXP *does not have quasi-polynomial size $\mathcal{C}$ circuits.*

For example, the proof that $\mathsf{NEXP} \not\subset \mathsf{ACC}$ follows from giving a faster-than-exhaustive-search ACC satisfiability algorithm, noting that ACC is typical, and applying Theorem 2.8.

This theorem cannot be directly applied to a class such as $\mathsf{ACC} \circ \mathsf{THR}$, because it is not known whether $\mathsf{ACC} \circ \mathsf{THR} \circ \mathsf{ACC} \circ \mathsf{THR}$ can be efficiently simulated with $\mathsf{ACC} \circ \mathsf{THR}$. However, by modifying the argument of Theorem 2.8 and using an algorithm for *counting* SAT assignments, we can extend the theorem to circuits with a very weak closure property.

Definition 2.9. Let $\mathcal{C}$ be a circuit class. We say $\mathcal{C}$ is *efficiently closed under* AND if, given the AND of two circuits of $\mathcal{C}$, an equivalent circuit in $\mathcal{C}$ can be produced in polynomial time.

Efficient closure under AND is satisfied by strictly more circuit classes than the property of being typical. To give an example, any class of the form $\mathsf{SYM} \circ \cdots$ is efficiently closed under AND, because an AND of t SYM gates with s wires can be collapsed into a single symmetric gate with $O(s^t)$ wires (as seen in the proof of Lemma 2.1). However, classes like $\mathsf{SYM} \circ \mathsf{SYM}$ and $\mathsf{THR} \circ \mathsf{THR}$ are *not* known to be efficiently closed under composition or unbounded-fan in AND/OR, hence Theorem 2.8 does not apply to such classes. We prove:

Theorem 2.10. *Let $\mathcal{C}$ be efficiently closed under AND. Suppose for every $c \geq 1$, there is an $\varepsilon > 0$ and an algorithm for counting the satisfying assignments of $\mathcal{C}$ circuits in time $O(2^{n-n^{\varepsilon}})$ on circuits with n inputs and $n^{\log^c n}$ size. Then* NEXP *does not have quasi-polynomial size $\mathcal{C}$ circuits.*

Note that Theorem 1.3 (the $\mathsf{ACC} \circ \mathsf{THR}$ lower bound) follows immediately from Theorem 2.10 and the counting algorithm of Theorem 1.2.[4] It is our hope that Theorem 2.10 may be applicable in the future to depth-two classes such as $\mathsf{SYM} \circ \mathsf{SYM}$ and depth-two *exact* threshold circuits [18], both of which are efficiently closed under AND. A nontrivial counting SAT algorithm for one of these classes would entail new lower bounds.

Proof of Theorem 2.10. Let us start with $\mathcal{C}$ as typical. We survey what is needed to conclude $\mathcal{C}$ lower bounds in the proof of Theorem 2.8 (from [54]), and show that the new hypothesis supplies these needs.

The proof of Theorem 2.8 starts by assuming that $\mathsf{NEXP} \subset \mathcal{C}$ and the theorem hypothesis. From these two assumptions, we derive that every $L \in \mathsf{NTIME}[2^n]$ can be simulated in nondeterministic $2^n/n$ time, contradicting the nondeterminstic time hierarchy [56]. In particular, it is shown that the assumptions imply that the NEXP-complete problem SUCCINCT 3SAT on AND/OR/NOT circuits of fan-in at most two, n inputs, and $\mathrm{poly}(n)$ size can be nondeterministically solved in $O(2^{n-n^{\varepsilon}})$ time, which is also provably false (see for example [52] for a detailed exposition). Recall that SUCCINCT 3SAT is the problem: *given an* AND/OR/NOT *circuit C of fan-in two, does the truth table of C encode a satisfiable 3-CNF formula?* That is, SUCCINCT 3SAT is a "compressed" version of the 3SAT problem.

Suppose we are given an (arbitrary) circuit C of s size and n inputs, and wish to determine if it is a yes-instance of SUCCINCT 3SAT. Assuming NEXP has quasipolynomial-size circuits, it is proved (again in [54]) that for every C encoding a satisfiable 3-CNF F, there is a quasipolynomial-size circuit D which *succinctly encodes* a satisfying assignment for F: for all i, $D(i)$ outputs the value of variable x_i in the satisfying assignment. Our "faster" nondeterministic algorithm for SUCCINCT 3SAT guesses this circuit D, and uses it to construct a circuit E with n inputs and $n^{\log^c n}$ size for some c, which is unsatisfiable if and only if D encodes a satisfying assignment to the formula F encoded by C.

Assuming that NEXP has quasipolynomial-size $\mathcal{C}$ circuits and that there is an $O(2^{n-n^{\varepsilon}})$ time algorithm for $\mathcal{C}$ satisfiability, it is then proved that there is a nondeterministic algorithm A running in $2^{n-\Omega(n^{\varepsilon})}$ time which, given an AND/OR/NOT circuit E of fan-in two with s size and n inputs, outputs an equivalent E' of $s^{\log^c s}$ size and $n + O(\log(\mathrm{size}(E))) \leq n + O(\log^{c+1} n)$ inputs from the class $\mathcal{C}$ on at least one nondeterministic branch (and prints *no* on other branches). Running this algorithm A, obtaining E', then running the $\mathcal{C}$ satisfiability algorithm on E', we nondeterministically determine that C is a yes-instance of SUCCINCT-3SAT in $2^{n-\Omega(n^{\varepsilon})}$ time.

Now assume $\mathcal{C}$ is only efficiently closed under AND. In the proof sketch above, the closure properties of $\mathcal{C}$ become relevant, precisely in the argument that the nondeterministic algorithm A exists. In fact, if the present theorem's hypothesis and the assumption that NEXP has quasipolynomial-size $\mathcal{C}$ circuits implies that such an algorithm A exists, it can be observed that the rest of the proof carries over without any modifications. We now construct such an algorithm A.

[4] The reader is also recommended to read [24, 37, 7], which consider other (stronger) closure properties. Their work also implies that faster SAT algorithms for $\mathsf{ACC} \circ \mathsf{THR}$ circuits yield lower bounds such as Theorem 1.3. In that light, the significant contribution of this section is to show how circuit lower bounds still follow from a notably weaker closure property, provided we can *count* SAT assignments faster.

Our new algorithm A is supposed to take a circuit E of size s and n inputs, and print an equivalent E' of $s^{\log^c s}$ size from the class $\mathcal{C}$. Given E, our A starts by guessing a $\mathcal{C}$ circuit E'' of $n^{\log^c n}$ size, which takes as input a pair $(x,g) \in \{0,1\}^n \times \{0,1\}^{\log(\text{size}(E))}$, and outputs 1 if and only if the gate g in E outputs 1 when E is given the input x. Observe that such an E'' exists, assuming that P has quasi-polynomial size $\mathcal{C}$ circuits (if P has quasi-polynomial size $\mathcal{C}$ circuits, then the *Circuit Evaluation problem* has quasi-polynomial size $\mathcal{C}$ circuits; see [54] for details).

Now we verify that for every gate g indexed by $1,2,\ldots,\text{size}(E)$, $E''(x,g)$ outputs what gate g of $E(x)$ outputs, on all x. Each gate g is either an input, an AND of two previous gates g_1 and g_2, an OR of two previous gates g_1 and g_2, or a NOT of a previous gate g_1.

To aid this verification, we show how to efficiently check for arbitrary $\mathcal{C}$ circuits G and H whether $G(x) = H(x)$ for all inputs x, using an algorithm for counting SAT assignments. Let $\#SAT(C)$ be the number of satisfying assignments to a circuit C. Observe that $G(x) = H(x)$ for all x if and only if $\#SAT(G) = \#SAT(H) = \#SAT(G \wedge H)$. (Note the third quantity can be efficiently computed, assuming $\mathcal{C}$ is efficiently closed under AND.) Moreover, $G(x) \neq H(x)$ for all x if and only if $\#SAT(G) + \#SAT(H) = 2^n$ and $\#SAT(G \wedge H) = 0$. Therefore, by counting SAT assignments, we have algorithms checking whether G is equivalent to H, and whether G is equivalent to the negation of H, both running in time $O(2^{n-n^{\varepsilon}})$.

We claim that the verification problem for E'' can be reduced to a small number of calls to the above kinds of checks. First, nondeterministically guess a circuit E''_{not}, intended to satisfy $E''_{\text{not}}(x,g) = \neg E''(x,g)$ for all x and g. Verifying this condition can be done by counting SAT assignments, as described above.

Checking E'' is correct on the input gates of E means that for all $i = 1,\ldots,n$, $E''(x_1,\ldots,x_n,i) = x_i$. Both $E''(x_1,\ldots,x_n,i)$ and $I(x_1,\ldots,x_n) = x_i$ are $\mathcal{C}$ circuits, hence their equivalence can be verified by $\#SAT$ calls. Checking a NOT gate g of E with input gate g_1 is equivalent to checking that $E''_{\text{not}}(x,g_1) = E''(x,g)$ on all x.

Checking an AND gate g of two previous gates g_1 and g_2 amounts to checking that $E''(x,g) = E''(x,g_1) \wedge E''(x,g_2)$ on all x. To do this, compute $G_{\text{and}}(x) := E''(x,g_1) \wedge E''(x,g_2)$ (assuming $\mathcal{C}$ is efficiently closed under AND), then check $G_{\text{and}}(x) = E''(x,g)$ for all x.

Finally, for an OR gate g with inputs g_1 and g_2, we want to check that $E''(x,g) = E''(x,g_1) \vee E''(x,g_2)$ on all x. This is equivalent to $\neg E''(x,g) = ((\neg E''(x,g_1)) \wedge (\neg E''(x,g_2)))$ for all x. This can be checked by forming $G_{\text{or}}(x) := E''_{\text{not}}(x,g_1) \wedge E''_{\text{not}}(x,g_2)$, then checking that $G_{\text{or}}(x) = E''_{\text{not}}(x,g)$ for all x.

On a circuit E with $s \leq n^{\log^c n}$ gates, the above procedure runs in $O(2^{n-n^{\varepsilon}} \cdot s) \leq 2^{n-\Omega(n^{\varepsilon})}$ time. When it concludes, we know that for all gates g and all x that $E''(x,g)$ outputs the correct value. The circuit $E'(x)$ output by A simply evaluates $E''(x,g^{\star})$, where $g^{\star}$ is the output gate of E. □

3 Fast evaluation of depth-two threshold circuits

Finally we show, that, in a strong sense, depth-two threshold circuits are *weak*, by giving a fast algorithm for evaluating such circuit on many assignments in batch. The general result is the following.

Theorem 3.1. *Given a depth-two linear threshold circuit C with $2k$ inputs and at most $n^{1/12}$ gates with weights on the bottom layer of absolute value at most W_b, weights on the output gate of absolute value at most W_o, and given two sets $A,B \subseteq \{0,1\}^k$ where $|A| = |B| = n$, we can evaluate C on all n^2 points in $A \times B$ using $n^2 \cdot \text{poly}(\log W_o, \log n) + n^{1+1/12} \cdot \text{poly}(\log n, \log W_o, \log W_b)$ time.*

The following is immediate from Theorem 3.1:

Reminder of Theorem 1.5. *Let $k > 1$. Given a depth-two $2^{n/24}$-size linear threshold circuit C with integer weights in $[-2^{n^k}, 2^{n^k}]$, we can evaluate C on all 2^n input assignments in $2^n \cdot \text{poly}(n^k)$ time.*

While the proof of Theorem 3.1 also ultimately depends on Coppersmith's rectangular matrix multiplication, the rest of the algorithm is rather different from the evaluation algorithm of Theorem 1.1.

Proof of Theorem 3.1. We reduce the evaluation task to a special kind of matrix multiplication, then combine Coppersmith's matrix multiplication with a mild brute force to expedite the matrix multiplication.

Define the function $\text{LEQ} : \mathbb{Z} \times \mathbb{Z} \to \{0,1\}$ to output 1 on (a,b) if and only if $a \leq b$. Given a vector $w = (w_1, \ldots, w_d) \in \mathbb{Z}^d$, and given two matrices M and N which are $n \times d$ and $d \times n$, define their *w-weighted threshold product* to be

$$(M \circledast_w N)[i,j] := \sum_{k=1}^{d} w_k \cdot \text{LEQ}(M[i,k], N[k,j]) .$$

We shall show that the w-weighted threshold product of an $n \times n^{1/12}$ matrix and an $n^{1/12} \times n$ matrix can be computed in essentially $n^2 \cdot \text{poly}(\log n)$ time (with some additional but negligible overhead in terms of the weights). Let us postpone this algorithm for the moment, and first show how to embed the evaluation problem into the weighted threshold product.

Let C be a depth-two circuit of size s, with the $2k$ input variables $x_1, \ldots, x_k, y_1, \ldots, y_k$. Let $w_1, \ldots, w_s$ be the weights of the top threshold gate of C, and let $\ell_1, t_1, \ldots, \ell_s, t_s$ be the corresponding linear forms and threshold values from the bottom layer of threshold gates: that is, the output of $\text{LEQ}(t_i, \ell_i)$ is multipled by w_i in the output gate. Without loss of generality, we may assume that all weights w_i are multiplied by the output of some threshold gate at the bottom layer (there are at most $2k$ wires from the input directly to the output gate, and they can be replaced by $O(k)$ dummy gates at the bottom layer with wires to the output gate). Let $A = \{A_1, \ldots, A_n\} \subseteq \{0,1\}^k$ and $B = \{B_1, \ldots, B_n\} \subseteq \{0,1\}^k$.

We partition each linear form ℓ_j on the bottom layer into two sums $\ell_j^{(x)}$ and $\ell_j^{(y)}$, such that $\ell_j^{(x)}$ involves only input variables $x_1, \ldots, x_k$, $\ell_j^{(y)}$ involves only $y_1, \ldots, y_k$, and $\ell_j^{(x)} + \ell_j^{(y)} = \ell_j$. Let $A_i(\ell_j^{(x)})$ and $B_j(\ell_j^{(y)})$ denote the value of the linear form $\ell_j^{(x)}$ (respectively, $\ell_j^{(y)}$) evaluated on assignment A_i (respectively, B_j).

We now define two special matrices M and N with respect to the A_i's and B_j's. Define a matrix M with rows indexed by elements of A, and columns indexed by the bottom layer gates $1, \ldots, s$, where

$$M[i,k] := t_k - A_i(\ell_k^{(x)}) .$$

Further define a matrix N with rows indexed by the bottom layer gates $1, \ldots, s$, and columns indexed by elements of B, where

$$N[k,j] := B_j(\ell_k^{(y)}) .$$

Now consider the w-weighted threshold product $M \circledast_w N$, where w is the same as above. The i,j entry of this product equals

$$\sum_{k=1}^{s} w_k \cdot \text{LEQ}\left(t_k - A(\ell_k^{(x)}), B_j(\ell_k^{(y)})\right) = \sum_{k=1}^{s} w_k \cdot \text{LEQ}\left(t_k, A_i(\ell_k^{(x)}) + B_j(\ell_k^{(y)})\right) .$$

This is precisely the value of the linear form in the output gate of C, when $x_1,\dots,x_k$ are given the assignment A_i and $y_1,\dots,y_k$ are assigned B_j. The truth table of C on $A \times B$ can be recovered by simply checking which entries in $(M \circledast_w N)$ exceed the output gate's threshold.

Next, we shall show how to compute a weighted threshold matrix product efficiently. Let δ be a parameter, and let M and N be $n \times n^{\delta}$ and $n^{\delta} \times n$ matrices, respectively. The first step is to reduce the weights significantly. For all $k = 1,\dots,n^{\delta}$, let S_k be a list of all entries in the kth column of M, plus the kth row of N. Sort S_k, obtaining a ranking of $2n$ items, and replace each entry in the kth column of M and the kth row of N by their rank in the sorted list S_k. This step reduces the domains of M and N to $\{1,\dots,2n\}$, and the w-weighted threshold matrix product remains the same: all inequalities $M[i,k] \le N[k,j]$ are preserved. Note this step takes $n^{1+\delta} \cdot \mathrm{poly}(\log n, \log W_b)$ time.

In order to reduce this matrix computation to a standard matrix multiplication, we perform two strategies with different advantages. (The reduction is inspired by work of Matousek [31] on computing dominances in high dimensions.) Let $s \in \{1,\dots,n\}$ be a parameter. Partition each sorted list S_k into $t = \lceil n/s \rceil$ contiguous buckets $T_1,\dots,T_t$, where each bucket T_i contains at most s entries. (For all $i < j$, the largest entry in T_i is at most the smallest entry in T_j.)

Start with an $n \times n$ output matrix P that is all zeroes. For every $(i,k) \in [n] \times [n^{\delta}]$, look up the bucket T_ℓ containing $M[i,k]$ in the sorted list S_k. For all $N[k,j]$ contained in T_ℓ such that $M[i,k] \le N[k,j]$, add the weight w_k to the entry $P[i,j]$. This loop adds to P all terms $w_k \cdot \mathrm{LEQ}(M[i,k],N[k,j])$ such that $M[i,k]$ and $N[k,j]$ appear in the same bucket of S_k. Observe that this step takes $n \cdot n^{\delta} \cdot s \cdot \mathrm{poly}(\log(W_o),\log(W_b))$ time.

To handle the $(M[i,k],N[k,j])$ pairs that do not appear in the same bucket, we use matrix multiplication. For each $(i,k) \in [n] \times [n^{\delta}]$, replace the entry $M[i,k]$ with a row vector $v_{i,k} \in \{0,w_k\}^t$, such that $v_{i,k}[\ell] := w_k$ if and only if $M[i,k]$ is in bucket T_ℓ of S_k. That is, $v_{i,k}$ has w_k in exactly one entry, and zeroes elsewhere. This forms a matrix M' of dimensions $n \times (n^{\delta} \cdot t)$. For $(k,j) \in [n^{\delta}] \times [n]$, replace each entry $N[k,j]$ with a column vector $u_{k,j} \in \{0,1\}^t$, such that $u_{k,j}[\ell'] := 1$ if and only if $N[k,j]$ is in bucket T_ℓ of S_k and $\ell > \ell'$. This forms a matrix N' of dimensions $(n^{\delta} \cdot t) \times n$. The matrix product $M' \cdot N'$ over the integers computes a sum of inner products

$$(M' \cdot N')[i,j] = \sum_{n^{\delta}} \langle v_{i,k}, u_{k,j} \rangle .$$

If $M[i,k] > N[k,j]$, or $M[i,k]$ and $N[k,j]$ are in the same bucket of S_k, then $\langle v_{i,k}, u_{k,j} \rangle = 0$. If $M[i,k] \le N[k,j]$ but $N[k,j]$ and $M[i,k]$ are in different buckets of S_k then $\langle v_{i,k}, u_{k,j} \rangle = w_k$.

Letting $P := P + (M' \cdot N')$, this procedure adds to P all terms $w_k \cdot \mathrm{LEQ}(M[i,k],N[k,j])$ such that $M[i,k]$ and $N[k,j]$ appear in different buckets of S_k. Therefore $P[i,j]$ contains the value of the linear form for the output gate of C, under variable assignment (A_i,B_j), for all i,j.

The above algorithm runs in time

$$O(n \cdot n^{\delta} \cdot s \log W_o + MM(n, n^{1+\delta}/s, n) \cdot \mathrm{poly}(\log W_o)),$$

where $MM(a,b,c)$ is the running time for multiplying $a \times b$ and $b \times c$ matrices. If we set $n^{1+\delta}/s = n^{0.172}$, then Coppersmith's algorithm (Lemma 2.5) can be applied to the second term of the running time, implementing it in $n^2 \cdot \mathrm{poly}(\log n)$ time. Under this setting, $s = n^{\delta} \cdot n^{0.828}$ and the first term of the running time is $n^{1+2\delta+0.828}$. Setting $\delta = 0.086 > 1/12$, the first term becomes n^2 (note that $s = n^{.914}$). □

It is easy to see that, since the above algorithm outputs the *values* of the linear form at the output gate of a depth-two threshold circuit over all inputs, we can also efficiently evaluate large $\mathsf{SYM} \circ \mathsf{THR}$ circuits.

4 Discussion

In this paper, we have shown how Boolean functions computable by small $\mathsf{ACC} \circ \mathsf{THR}$ circuits have low "symmetric rank," how to efficiently compute an appropriate rank decomposition (given a circuit), and how to exploit this rank decomposition for new satisfiability algorithms and new circuit lower bounds. Our methods provide ways to algorithmically handle symmetric and threshold gates at the bottom of a circuit, which also lead to faster algorithms for problems where threshold functions are part of the constraint set, such as 0-1 linear programming.

Recall that in Section 3 (Theorem 3.1), we showed how to evaluate a given $\mathsf{THR} \circ \mathsf{THR}$ circuit of $2^{\Omega(n)}$ size on all 2^n inputs in $2^{n+o(n)}$ time. Perhaps the most interesting problem left open by this paper is: *can we apply the fast evaluation methods of Section 3 to prove lower bounds for* $\mathsf{THR} \circ \mathsf{THR}$ *circuits?* Drawing from the results of this paper, we believe that $\mathsf{THR} \circ \mathsf{THR}$ circuits should be provably weaker than arbitrary (unrestricted) circuits of polynomially related size. First let us recall a simple observation regarding the connection between multipoint circuit evaluation and Circuit-SAT (implicit in [54, Theorem 4.5]).

Theorem 4.1. *Suppose there is an* $\alpha > 0$ *such that we can evaluate* arbitrary *(unrestricted) circuits of* $2^{\alpha n+o(n)}$ *size on all* 2^n *inputs, in* $2^{n+o(n)}$ *time. Then there is a* $\delta < 1$ *such that* Circuit-SAT *for* $2^{o(n)}$*-size circuits with* n *inputs is solvable in* $2^{\delta n}$ *time.*

Proof. The proof is similar in nature to the proof of Theorem 1.2: given a circuit C of size $2^{o(n)}$, for every assignment $A_j \in \{0,1\}^{\ell}$, make a circuit $C_{A_j}(x_1,\ldots,x_{n-\ell}) := C(x_1,\ldots,x_{n-\ell},A_j)$, and define D to be the OR over all j of C_{A_j}. D has $n-\ell$ inputs and size $2^{\ell} \cdot 2^{o(n)}$. Setting $\ell = \alpha n$, the hypothesis of the theorem implies that we can evaluate D on all its possible inputs (and thereby solve SAT for C) in $2^{n-\ell+o(n)}$ time, so we can set $\delta = 1-\alpha$. $\square$

The conclusion of Theorem 4.1 is *extremely strong*; the author considers it to be unlikely to be true. For one, it is contrary to what we generally believe about the power of arbitrary circuits: it seems unlikely that we can solve the SAT problem much faster on circuits of exponential size. Moreover, this conclusion has very strong consequences: it would imply *exponential-size, unrestricted* circuit lower bounds for $\mathsf{EXP}^{\mathsf{NP}}$ by the known connections between SAT algorithms and lower bounds [53]. As the hypothesis of Theorem 4.1 is *true* for $\mathsf{THR} \circ \mathsf{THR}$ circuits, it therefore seems unlikely that one can convert unrestricted circuits of size s into $\mathsf{THR} \circ \mathsf{THR}$ circuits of size $\mathrm{poly}(s)$. Indeed, we have the following result.

Theorem 4.2. *Suppose the circuit evaluation problem has* $\mathsf{THR} \circ \mathsf{THR}$ *circuits of* $\mathrm{poly}(n)$ *size, which are constructible in* $\mathrm{poly}(n)$ *time. Then there is a* $\delta < 1$ *such that* Circuit-SAT *for* $2^{o(n)}$*-size circuits with* n *inputs is solvable in* $2^{\delta n}$ *time.*

Proof. We prove that the hypothesis of the theorem implies the hypothesis of Theorem 4.1. Let $k \geq 1$ be such that the circuit evaluation problem (on circuits of size up to n) has $\mathsf{THR} \circ \mathsf{THR}$ circuits of at most n^k size, which are constructible in $O(n^k)$ time. Let $\alpha = 1/(24k)$. Given an unrestricted circuit C with

n inputs of $2^{\alpha n}$ size, we first generate an $O(2^{k\alpha n})$-size $\mathsf{THR} \circ \mathsf{THR}$ circuit D for the circuit evaluation problem, in $O(2^{k\alpha n}) \leq O(2^{n/24})$ time. The circuit D takes a pair (C', x) as input (where C' is a circuit and x is an input to C'), and outputs the value of $C'(x)$. If we hard-code the description of C into the input of D, we obtain a $\mathsf{THR} \circ \mathsf{THR}$ circuit of at most $2^{k\alpha n} \leq O(2^{n/24})$ size which is equivalent to C. By Theorem 3.1, we can evaluate D (and therefore C) on all possible inputs in $2^n \cdot \mathrm{poly}(n)$ time. □

Therefore, under the hypothesis that general Circuit-SAT cannot be solved much faster than 2^n time on subexponential-size circuits, we can conclude a super-polynomial lower bound for $\mathsf{THR} \circ \mathsf{THR}$ circuits computing the circuit evaluation problem (which is in P). While this exact line of reasoning is highly unlikely to lead to an unconditional lower bound itself (proving *any* lower bound against Circuit-SAT is difficult!), it provides further confidence that $\mathsf{THR} \circ \mathsf{THR}$ circuits are much weaker than their unrestricted counterparts.

Together, the above two theorems show the following.

> *If* P *has uniform* $\mathsf{THR} \circ \mathsf{THR}$ *circuits of polynomial size, then there is an* $\alpha > 0$ *such that* $\mathsf{EXP}^{\mathsf{NP}}$ *does not have* $2^{\alpha n}$*-size circuits.*

For reasons such as these, we believe that lower bounds such as "NP does not have uniform $\mathsf{THR} \circ \mathsf{THR}$ circuits of polynomial size" may be within reach, but we have not yet found the right form of argument.

Acknowledgements. Thanks to Igor Carboni Oliveira for sending a preliminary version of his survey, which helped the ideas in the proof of Theorem 2.10 to congeal. I also thank Rahul Santhanam, Emanuele Viola, and the anonymous STOC and ToC reviewers, for helpful comments (and in the case of the ToC reviewers, I also thank them for years of patience).

References

[1] ERIC ALLENDER AND VIVEK GORE: On strong separations from AC^0. In *Proc. 8th Internat. Conf. Fundamentals of Computation Theory (FCT'91)*, pp. 1–15. Springer, 1991. [doi:10.1007/3-540-54458-5_44] 8

[2] JAMES ASPNES, RICHARD BEIGEL, MERRICK FURST, AND STEVEN RUDICH: The expressive power of voting polynomials. *Combinatorica*, 14(2):135–148, 1994. Conference version in STOC'91. [doi:10.1007/BF01215346] 5

[3] DAVID A. MIX BARRINGTON AND HOWARD STRAUBING: Complex polynomials and circuit lower bounds for modular counting. *Comput. Complexity*, 4(4):325–338, 1994. Conference version in LATIN'92. [doi:10.1007/BF01263421] 5

[4] PAUL BEAME AND TRINH HUYNH: Multiparty communication complexity and threshold circuit size of AC^0. *SIAM J. Comput.*, 41(3):484–518, 2012. Conference version in FOCS'09. [doi:10.1137/100792779] 5

[5] RICHARD BEIGEL: When do extra majority gates help? Polylog(N) majority gates are equivalent to one. *Comput. Complexity*, 4(4):314–324, 1994. Conference version in STOC'92. [doi:10.1007/BF01263420] 5, 8

[6] RICHARD BEIGEL AND JUN TARUI: On ACC. *Comput. Complexity*, 4(4):350–366, 1994. Conference version in FOCS'91. [doi:10.1007/BF01263423] 8

[7] ELI BEN-SASSON AND EMANUELE VIOLA: Short PCPs with projection queries. In *Proc. 41st Internat. Colloq. on Automata, Languages and Programming (ICALP'14)*, pp. 163–173. Springer, 2014. [doi:10.1007/978-3-662-43948-7_14] 15

[8] ASHOK K. CHANDRA, LARRY STOCKMEYER, AND UZI VISHKIN: Constant depth reducibility. *SIAM J. Comput.*, 13(2):423–439, 1984. [doi:10.1137/0213028] 8, 11

[9] ARKADEV CHATTOPADHYAY AND KRISTOFFER ARNSFELT HANSEN: Lower bounds for circuits with few modular and symmetric gates. In *Proc. 32nd Internat. Colloq. on Automata, Languages and Programming (ICALP'05)*, pp. 994–1005. Springer, 2005. [doi:10.1007/11523468_80] 5

[10] GIL COHEN: A taste of circuit complexity pivoted at NEXP $\not\subset$ ACC^0 (and more), 2013. Available at ECCC. 2

[11] DON COPPERSMITH: Rapid multiplication of rectangular matrices. *SIAM J. Comput.*, 11(3):467–471, 1982. [doi:10.1137/0211037] 6

[12] JÜRGEN FORSTER, MATTHIAS KRAUSE, SATYANARAYANA V. LOKAM, RUSTAM MUBARAKZJANOV, NIELS SCHMITT, AND HANS ULRICH SIMON: Relations between communication complexity, linear arrangements, and computational complexity. In *Proc. 21st Internat. Conf. Foundations of Software Technology and Theoretical Computer Science (FSTTCS'01)*, pp. 171–182. Springer, 2001. [doi:10.1007/3-540-45294-X_15] 5

[13] MIKAEL GOLDMANN: On the power of a threshold gate at the top. *Inform. Process. Lett.*, 63(6):287–293, 1997. [doi:10.1016/S0020-0190(97)00141-5] 5

[14] PARIKSHIT GOPALAN AND ROCCO A. SERVEDIO: Learning and lower bounds for AC^0 with threshold gates. In *Proc. 14th Internat. Workshop on Randomization and Computation (RANDOM'10)*, pp. 588–601. Springer, 2010. [doi:10.1007/978-3-642-15369-3_44] 5

[15] ANDRÁS HAJNAL, WOLFGANG MAASS, PAVEL PUDLÁK, MARIO SZEGEDY, AND GYÖRGY TURÁN: Threshold circuits of bounded depth. *J. Comput. System Sci.*, 46(2):129–154, 1993. Conference version in FOCS'87. [doi:10.1016/0022-0000(93)90001-D] 5

[16] KRISTOFFER ARNSFELT HANSEN: Computing symmetric boolean functions by circuits with few exact threshold gates. In *Proc. 13th Ann. Internat. Computing and Combinatorics Conf. (COCOON'07)*, pp. 448–458. Springer, 2007. [doi:10.1007/978-3-540-73545-8_44] 5

[17] KRISTOFFER ARNSFELT HANSEN AND PETER BRO MILTERSEN: Some meet-in-the-middle circuit lower bounds. In *Proc. 33rd Internat. Symp. Mathematical Foundations of Comp. Sci. (MFCS'04)*, pp. 334–345. Springer, 2004. [doi:10.1007/978-3-540-28629-5_24] 5

[18] KRISTOFFER ARNSFELT HANSEN AND VLADIMIR V. PODOLSKII: Exact threshold circuits. In *Proc. 25th IEEE Conf. on Computational Complexity (CCC'10)*, pp. 270–279. IEEE Comp. Soc. Press, 2010. [doi:10.1109/CCC.2010.33] 15

[19] KRISTOFFER ARNSFELT HANSEN AND VLADIMIR V. PODOLSKII: Polynomial threshold functions and boolean threshold circuits. *Inform. and Comput.*, 240:56–73, 2015. Conference version in MFCS'13. [doi:10.1016/j.ic.2014.09.008] 3, 5

[20] RUSSELL IMPAGLIAZZO, VALENTINE KABANETS, AND AVI WIGDERSON: In search of an easy witness: Exponential time vs. probabilistic polynomial time. *J. Comput. System Sci.*, 65(4):672–694, 2002. Conference version in CCC'01. [doi:10.1016/S0022-0000(02)00024-7] 2

[21] RUSSELL IMPAGLIAZZO, SHACHAR LOVETT, RAMAMOHAN PATURI, AND STEFAN SCHNEIDER: 0-1 integer linear programming with a linear number of constraints, 2014. [arXiv:1401.5512] 4

[22] RUSSELL IMPAGLIAZZO, WILLIAM MATTHEWS, AND RAMAMOHAN PATURI: A satisfiability algorithm for AC^0. In *Proc. 23rd Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'12)*, pp. 961–972. ACM Press, 2012. ACM DL. [arXiv:1107.3127] 4

[23] RUSSELL IMPAGLIAZZO, RAMAMOHAN PATURI, AND STEFAN SCHNEIDER: A satisfiability algorithm for sparse depth two threshold circuits. In *Proc. 54th FOCS*, pp. 479–488. IEEE Comp. Soc. Press, 2013. [doi:10.1109/FOCS.2013.58, arXiv:1212.4548] 4

[24] HAMIDREZA JAHANJOU, ERIC MILES, AND EMANUELE VIOLA: Local reductions, 2013. [arXiv:1311.3171] 15

[25] SWASTIK KOPPARTY AND SRIKANTH SRINIVASAN: Certifying polynomials for $AC^0[\oplus]$ circuits, with applications to lower bounds and circuit compression. *Theory of Computing*, 14(12):1–24, 2018. Conference version in FSTTCS'12. [doi:10.4086/toc.2018.v014a012] 12

[26] SATYANARAYANA V. LOKAM: Complexity lower bounds using linear algebra. *Foundations and Trends in Theoretical Computer Science*, 4(1-2):1–155, 2009. [doi:10.1561/0400000011] 5

[27] SHACHAR LOVETT AND SRIKANTH SRINIVASAN: Correlation bounds for poly-size AC^0 circuits with $n^{1-o(1)}$ symmetric gates. In *Proc. 15th Internat. Workshop on Randomization and Computation (RANDOM'11)*, pp. 640–651. Springer, 2011. [doi:10.1007/978-3-642-22935-0_54] 5

[28] ALEXIS MACIEL AND DENIS THÉRIEN: Threshold circuits for iterated multiplication: Using AC^0 for free. In *Proc. 10th Symp. Theoretical Aspects of Comp. Sci. (STACS'93)*, pp. 545–565. Springer, 1993. [doi:10.1007/3-540-56503-5_54] 3, 5

[29] ALEXIS MACIEL AND DENIS THÉRIEN: Threshold circuits of small majority-depth. *Inform. and Comput.*, 146(1):55–83, 1998. [doi:10.1006/inco.1998.2732] 7

[30] ALEXIS MACIEL AND DENIS THÉRIEN: Efficient threshold circuits for power series. *Inform. and Comput.*, 152(1):62–73, 1999. [doi:10.1006/inco.1998.2783] 3

[31] JIŘÍ MATOUŠEK: Computing dominances in E^n. *Inform. Process. Lett.*, 38(5):277–278, 1991. [doi:10.1016/0020-0190(91)90071-O] 18

[32] MARVIN MINSKY AND SEYMOUR A. PAPERT: *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969. 2

[33] SABURO MUROGA: *Threshold Logic and its Applications*. John Wiley & Sons, Inc., 1971. 2, 7

[34] SABURO MUROGA, IWAO TODA, AND SATORU TAKASU: Theory of majority decision elements. *Journal of the Franklin Institute*, 271(5):376–418, 1961. [doi:10.1016/0016-0032(61)90702-5] 7

[35] MONI NAOR AND OMER REINGOLD: Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2):231–262, 2004. Conference version in FOCS'97. [doi:10.1145/972639.972643] 3, 4

[36] NOAM NISAN: The communication complexity of threshold gates. In *Combinatorics, Paul Erdős is Eighty*, pp. 301–315. János Bolyai Mathematical Society, Budapest, 1994. 5

[37] IGOR CARBONI OLIVEIRA: Algorithms versus circuit lower bounds, 2013. [arXiv:1309.0249] 2, 15

[38] ERION PLAKU: Multiplicity automata, polynomials and the complexity of small-depth boolean circuits. Master's thesis, Clarkson University, 2002. Available at robotmotionplanning.org. 4

[39] VLADIMIR V. PODOLSKII: Exponential lower bound for bounded depth circuits with few threshold gates. *Inform. Process. Lett.*, 112(7):267–271, 2012. [doi:10.1016/j.ipl.2011.12.011] 5

[40] ALEXANDER A. RAZBOROV: On small depth threshold circuits. In *Proc. 3rd Scandinavian Workshop on Algorithm Theory (SWAT'92)*, pp. 42–52. Springer, 1992. [doi:10.1007/3-540-55706-7_4] 3

[41] ALEXANDER A. RAZBOROV AND STEVEN RUDICH: Natural proofs. *J. Comput. System Sci.*, 55(1):24–35, 1997. [doi:10.1006/jcss.1997.1494] 2, 4

[42] ALEXANDER A. RAZBOROV AND ALEXANDER A. SHERSTOV: The sign-rank of AC^0. *SIAM J. Comput.*, 39(5):1833–1855, 2010. Conference version in FOCS'08. [doi:10.1137/080744037] 5

[43] ALEXANDER A. RAZBOROV AND AVI WIGDERSON: $n^{\Omega(\log n)}$ lower bounds on the size of depth-3 threshold circuits with AND gates at the bottom. *Inform. Process. Lett.*, 45(6):303–307, 1993. Conference version in STOC'94. [doi:10.1016/0020-0190(93)90041-7] 5

[44] KENNETH W. REGAN: Polynomials and combinatorial definitions of languages. In *Complexity Theory Retrospective II*, pp. 261–293. Springer, 1997. 3

[45] JOHN H. REIF AND STEPHEN R. TATE: On threshold circuits and polynomial computation. *SIAM J. Comput.*, 21(5):896–908, 1992. Conference version in SCTC'87. [doi:10.1137/0221053] 3

[46] RAHUL SANTHANAM: Ironic complicity: Satisfiability algorithms and circuit lower bounds. *Bulletin of the EATCS*, 106:31–52, 2012. 2

[47] ALEXANDER A. SHERSTOV: Separating AC^0 from depth-2 majority circuits. *SIAM J. Comput.*, 38(6):2113–2129, 2009. Conference version in STOC'07. [doi:10.1137/08071421X] 5

[48] KAI-YEUNG SIU, JEHOSHUA BRUCK, THOMAS KAILATH, AND THOMAS HOFMEISTER: Depth efficient neural networks for division and related problems. *IEEE Trans. Inform. Theory*, 39(3):946–956, 1993. [doi:10.1109/18.256501] 3

[49] KAI-YEUNG SIU AND VWANI P. ROYCHOWDHURY: On optimal depth threshold circuits for multiplication and related problems. *SIAM J. Discrete Math.*, 7(2):284–292, 1994. [doi:10.1137/S0895480192228619] 3

[50] ROMAN SMOLENSKY: Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proc. 19th STOC*, pp. 77–82. ACM Press, 1987. [doi:10.1145/28395.28404] 12

[51] EMMANUELE VIOLA: Pseudorandom bits for constant-depth circuits with few arbitrary symmetric gates. *SIAM J. Comput.*, 36(5):1387–1403, 2007. Conference version in CCC'05. [doi:10.1137/050640941] 5

[52] R. RYAN WILLIAMS: Guest column: a casual tour around a circuit complexity bound. *ACM SIGACT News*, 42(3):54–76, 2011. [doi:10.1145/2034575.2034591, arXiv:1111.1261] 2, 15

[53] R. RYAN WILLIAMS: Improving exhaustive search implies superpolynomial lower bounds. *SIAM J. Comput.*, 42(3):1218–1244, 2013. Conference version in STOC'10. [doi:10.1137/10080703X] 2, 6, 19

[54] R. RYAN WILLIAMS: Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2:1–2:32, 2014. Conference version in CCC'11. [doi:10.1145/2559903] 2, 3, 4, 6, 10, 12, 14, 15, 16, 19

[55] R. RYAN WILLIAMS: Natural proofs versus derandomization. *SIAM J. Comput.*, 45(2):497–529, 2016. Conference version in STOC'13. [doi:10.1137/130938219, arXiv:1212.1891] 2

[56] STANISLAV ŽÁK: A Turing machine time hierarchy. *Theoret. Comput. Sci.*, 26(3):327–333, 1983. [doi:10.1016/0304-3975(83)90015-4] 15

AUTHOR

R. Ryan Williams
Associate professor
Electrical Engineering and Computer Science
MIT, Cambridge, MA
rrw@mit.edu
http://people.csail.mit.edu/rrw/

ABOUT THE AUTHOR

R. RYAN WILLIAMS received his Ph. D. from Carnegie Mellon University in 2007, advised by the incomparable Manuel Blum. Up until recently, he lived on the Stanford campus, where marble-sized tomatoes grew on his balcony. Nowadays, he has no balcony and no tomatoes, but at least he can afford a place larger than a closet. He does not understand everyone else's distinction between algorithms and complexity.