






#SAT-Algorithms for Classes of Threshold Circuits Based on Probabilistic Rank

Nutan Limaye   

IT University of Copenhagen, Denmark

Adarsh Srinivasan   

Rutgers University, Piscataway, NJ, USA

Srikanth Srinivasan   

University of Copenhagen, Denmark

Abstract

There is a large body of work that shows how to leverage lower bound techniques for circuit classes to obtain satisfiability algorithms that run in better than brute-force time [24, 38]. For circuits with threshold gates, there are several such algorithms based on either

- Probabilistic Representations by low-degree polynomials, which allow for the use of fast polynomial evaluation algorithms, or
- Low rank, which allows for an efficient reduction to rectangular matrix multiplication.

In this paper, we use a related notion of *probabilistic rank* to obtain satisfiability algorithms for circuit classes contained in $\text{ACC}^0 \circ 3\text{-PTF}$, i.e. constant-depth circuits with modular counting gates and a single layer of degree-3 polynomial threshold functions.

Even for the special case of a single 3-PTF, it is not clear how to use either of the above two strategies to get a non-trivial satisfiability algorithm. The best known algorithm in this case previously was based on memoization and yields worse guarantees than our algorithm.

2012 ACM Subject Classification Theory of computation \rightarrow Computational complexity and cryptography

Keywords and phrases probabilistic polynomials, probabilistic rank, circuit satisfiability, circuit lower bounds, polynomial method, threshold circuits

Digital Object Identifier 10.4230/LIPIcs.MFCS.2025.67

Funding *Nutan Limaye*: Received funding from the Independent Research Fund Denmark (grant agreement No. 10.46540/3103-00116B) and is also supported by the Basic Algorithms Research Copenhagen (BARC), funded by VILLUM Foundation Grants 16582 and 54451, and Digital Research Centre Denmark, project P40.

Adarsh Srinivasan: Supported by the National Science Foundation under Grants CCF-2313372 and CCF-2443697. Part of this work was done during a visit to ITU Copenhagen and BARC funded by Basic Algorithms Research Copenhagen (BARC), supported by VILLUM Foundation Grants 16582 and 54451, and while at INSAIT, Sofia University “St. Kliment Ohridski”, Bulgaria. This work was partially funded from the Ministry of Education and Science of Bulgaria (support for INSAIT, part of the Bulgarian National Roadmap for Research Infrastructure).

Srikanth Srinivasan: Funded by the European Research Council (ERC) under grant agreement no. 101125652 (ALBA).

1 Introduction

Satisfiability algorithms and lower bounds

Given a family of circuits \mathcal{C} , a fundamental algorithmic question one can ask about it is the satisfiability question: does a given $C \in \mathcal{C}$ have a satisfying assignment? This being a canonical NP-complete problem even for very simple \mathcal{C} , it is hard to imagine a polynomial-



© Nutan Limaye, Adarsh Srinivasan, and Srikanth Srinivasan;
licensed under Creative Commons License CC-BY 4.0

50th International Symposium on Mathematical Foundations of Computer Science (MFCS 2025).

Editors: Paweł Gawrychowski, Filip Mazowiecki, and Michał Skrzypczak; Article No. 67; pp. 67:1–67:18

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

time algorithm for this problem. However, it is still reasonable to try to obtain some speed-up over brute-force search. Typically, and also in this paper, we aim for running times of 2^{n-s} where n denotes the number of variables and s is the “savings” over brute-force search.¹

A long line of work has leveraged structural understanding from complexity-theoretic research on circuit lower bounds to devise satisfiability algorithms for various families of circuits. For example, these ideas have led to approaches to satisfiability algorithms for constant-depth circuit classes [24, 17, 37, 25], Boolean formulas [29, 30] and Threshold circuits of various kinds [34, 12, 18, 9, 7].

A profound result of Williams [36, 38] also makes a connection in the opposite direction, by showing that satisfiability algorithms with savings $\omega(\log n)$ for many “reasonable” circuit classes implies lower bounds for these classes.² This gives even more impetus to studying the satisfiability problem in this regime.

Lower bound techniques in satisfiability algorithms

There are a handful of techniques from complexity-theoretic investigations that are useful in devising satisfiability algorithms. These include

- **Memoization:** In the setting of lower bounds, this idea goes back to the work of Nechiporuk [22] who used it to show quadratic Boolean formula lower bounds. This idea can also be used to obtain satisfiability algorithms in some settings (see e.g. [28, 7]). To apply this technique to a circuit class \mathcal{C} , one must be able to prove a strong upper bound on the number of functions in \mathcal{C} of a given size. Unfortunately, in most algorithmic settings (e.g. CNF formulas of unbounded width), one cannot get a strong enough upper bound of this form to get much out of this technique.
- **Restriction-based techniques:** A popular family of techniques used to devise satisfiability algorithms is the idea of restricting a subset of the variables (i.e. setting them to Boolean assignments) in a way that simplifies the underlying circuit and allows us to circumvent having to try all settings of the other variables. This leads to DPLL-style algorithms for SAT and their extensions to AC^0 circuits [17] and Boolean formulas [29] based on results such as the Håstad Switching Lemma [15, 16]. Unfortunately, these arguments do not extend to circuit classes where the gate set contains gates other than the DeMorgan basis (e.g. XOR gates or Majority gates that count the number of 1s) unless severe size restrictions are placed on the circuits [12, 18].

This brings us to the topic of this paper, where we study circuits with threshold gates. For an integer d , a *Polynomial Threshold Function (PTF)* of degree d is defined to be a Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ such that there exists a multilinear polynomial P of degree $\leq d$ with integer coefficients such that for each $x \in \{0, 1\}^n$, $f(x) = 1$ if and only if $P(x) > 0$. We denote by d -PTF the class of polynomial threshold functions of degree d . The special case $d = 1$ is denoted LTF.

Even the problem of checking if a given $f \in 2$ -PTF is satisfiable is an NP-complete problem.³ Further, the problem for d -PTF is a considerable generalization of the problem of optimizing a MAX- d -CSP over Boolean variables. While we know algorithms for MAX-2-CSP with savings $\Omega(n)$ [35], obtaining such a result even in the case of 3-CSPs remains an elusive problem. For the best known result for $d \leq 4$ see the work of Alman, Chan, and Williams [3].

¹ The trivial algorithm evaluates the circuit on every input, which takes time at least 2^n .

² i.e. a superpolynomial improvement over brute-force

³ Here, the input is given as a polynomial P with integer coefficients that represents f in the sense described above.

We will consider algorithms for classes of polynomial-sized circuits using PTF gates (and also other counting gates). In such situations, neither memoization nor restriction-based techniques seem to be useful in devising satisfiability algorithms, except in very special cases (as we will explain below). However, there are some known algorithms that work for such circuit classes. Mostly, they exploit the following ideas.

- **Polynomial representations:** This is a powerful circuit-analysis technique going back to a foundational circuit lower bound due to Razborov [26, 32] and follow-up results of Yao [39] and Beigel and Tarui [8] which were aimed towards proving bounds for constant-depth circuits with AND, OR, NOT and modular counting gates. Here, it is shown that small circuits from this class can be represented by (sometimes randomized) polynomials. By making this technique algorithmic and using fast polynomial evaluation algorithms, Williams [38] devised the first satisfiability algorithms for these circuits with non-trivial savings, leading also to his breakthrough circuit lower bound for the class ACC^0 .
- **Rank techniques:** Another powerful idea is to reduce the problem of checking satisfiability to computing the entries of a low-rank matrix. Typically, this is done by splitting the n variables into two subsets of size $n/2$ and showing that the $2^{n/2} \times 2^{n/2}$ -sized matrix obtained by evaluating the given input circuit C on all pairs of inputs $(x, y) \in \{0, 1\}^{n/2} \times \{0, 1\}^{n/2}$ is (a simple function of) a low-rank matrix. By using fast algorithms for rectangular matrix multiplication [14, 34], this leads to satisfiability algorithms for classes such as $\text{ACC}^0 \circ \text{LTF}$.

In this work, we show how to extend these ideas to polynomial threshold functions of degree up to 3 and generalizations thereof. It is unclear if either of the above techniques can be used directly to obtain non-trivial savings in this setting.⁴ More precisely, we obtain the following results. See Section 2 for definitions.

► **Theorem 1 (Informal).** *We have the following satisfiability algorithms.*

- *For any fixed prime p , a deterministic algorithm for counting the number of satisfying assignments of a given polynomial-sized $\text{AC}^0[\text{MOD}_p] \circ 3\text{-PTF}$ circuit with savings $\tilde{\Omega}(\sqrt{n})$ over brute-force search.*
- *A deterministic algorithm for counting the number of satisfying assignments of a given polynomial-sized $\text{ACC}^0 \circ 3\text{-PTF}$ circuit with savings $\Omega(n^\varepsilon)$ where ε depends on the depth of the circuit.*

Even in the case of a single 3-PTF, the running time of our algorithm beats the best-known previous algorithm of [7], which was based on memoization and yielded savings of $\tilde{\Omega}(n^{1/3})$. Moreover, as mentioned above, the technique of memoization does not seem to be useful even for simple extensions of this class to, say, conjunctions of polynomially many 3-PTFs.

As a consequence of our theorem and the *algorithmic method* of Williams, extended by Murray and Williams [38, 21], we also derive the first lower bound for the circuit class $\text{ACC}^0 \circ 3\text{-PTF}$.

⁴ It should be noted that a PTF of degree d has a low-degree polynomial (sign) representation by definition. However, this alone is not enough to devise a satisfiability algorithm. We also need a low-degree polynomial representation for a *disjunction* of superpolynomially many PTFs, and whether this stronger property holds is unclear. In fact, it seems unlikely [31].

Main idea

The main idea behind our algorithm is similar to the idea used in the work of Alman, Chan and Williams [3] for MAX-4-CSPs. Specifically, we combine the two techniques mentioned above into a *probabilistic rank* upper bound for the classes of circuits we consider. For simplicity, consider the case when the input is a polynomial $P(z_1, \dots, z_n)$ of degree 3 representing a 3-PTF f and we want to know if f is satisfiable.

As above, we split the n Boolean variables of P into two disjoint sets $A = \{1, \dots, n/2\}$ and $B = \{n/2 + 1, \dots, n\}$ of size $n/2$ and consider the matrix M_f of size $2^{n/2} \times 2^{n/2}$ where the rows and columns are labelled by settings to variables indexed by A and B respectively and

$$M_f(x, y) = f(x, y)$$

for each $x, y \in \{0, 1\}^{n/2}$. Now, while we cannot argue that M_f is low-rank, we can argue that there is a low-rank (roughly $\exp(\tilde{O}(\sqrt{n})) \ll 2^{n/2}$) *random* matrix \mathbf{M} that agrees with M_f in each entry w.h.p. To do this, we write

$$P(x_1, \dots, x_{n/2}, y_1, \dots, y_{n/2}) = \sum_{i=1}^{n/2} x_i \cdot Q_i(y_1, \dots, y_{n/2}) + \sum_{j=1}^{n/2} y_j \cdot R_j(x_1, \dots, x_{n/2}) + Q(y) + R(x) \quad (1)$$

by noting that each monomial in P must have degree at most 1 either in the variables indexed by A or by B . Partitioning the monomials that have variables in both parts according to choice of this “special” variable gives the decomposition above (Q and R contain the monomials that contain variables indexed by only one of B or A respectively). This type of decomposition is exactly like that obtained by [3]. From this point on our proof outline is also quite similar. The main place where we differ is that we need to handle weights which are not necessarily polynomially bounded.

We continue as follows. Note that $f(x, y)$ can be written as the sign of the above expression. To express M_f as a low-rank matrix (probabilistically), we use constructions of *probabilistic polynomials* (see definition in Section 2) for the Majority function⁵ and variants [6, 3] which allows us to write the above as a random polynomial of degree at most $\tilde{O}(\sqrt{n})$ over the bits of the numbers

$$x_1, \dots, x_{n/2}, R_1(x), \dots, R_{n/2}(x), R(x), y_1, \dots, y_{n/2}, Q_1(y), \dots, Q_{n/2}(y), Q(y).$$

Each monomial in the polynomial defines a rank-1 matrix, as it is a product of the form $F(x) \cdot G(y)$ for some functions F and G . A polynomial of degree $\tilde{O}(\sqrt{n})$ thus defines a matrix of rank $\exp(\tilde{O}(\sqrt{n}))$.

To extend the above to (say) $\text{ACC}^0 \circ 3\text{-PTF}$, we compose the above construction with standard low-degree polynomial representations of the class ACC^0 [2, 8].

This structural result shows that the family of degree-3 PTFs has low probabilistic rank. This may be of independent interest. Alman and Williams [5] proved such a result for the inner product function, resulting in improved bounds on the rigidity of the Hadamard

⁵ This idea works when the coefficients of the underlying polynomial P are all polynomially small. Our algorithm can also handle coefficients that are exponentially large. In this case, we need some additional circuitry to simulate the sum efficiently. This can be converted to a low-degree polynomial using the construction of Razborov [26, 25].

matrix. Follow-up results have used this idea to give better circuits for computing the Walsh-Hadamard transform [4].

For the satisfiability algorithm, we need to combine this idea with the standard “blow-up trick” of Williams. Returning to the case of a single 3-PTF $f(z_1, \dots, z_n)$, we will apply the above idea to

$$g(z_1, \dots, z_{n-\ell}) = \bigvee_{a \in \{0,1\}^\ell} f(z_1, \dots, z_{n-\ell}, a_1, \dots, a_\ell).$$

(Note that g is satisfiable if and only if f is satisfiable.) For suitably small ℓ (roughly \sqrt{n}), we will apply the above idea to g and show that the analogous matrix M_g has low-rank. By using Coppersmith’s rectangular matrix multiplication algorithm [14, 34], it follows that M_g can be computed in time approximately $2^{n-\ell}$, leading to savings ℓ for our algorithm.

Finally, to *count* the number of satisfying assignments, we replace the OR above by a sum and use similar ideas (as in [34, 25]).

► **Remark 2.** The algorithm for MAX-d-CSP (for the dense case) by [3] is closely related to our algorithm. They are able to make it work for $d = 4$, whereas, we only obtain algorithms for 3-PTF. The main bottleneck seems to be that we are dealing with arbitrary weights, whereas they work with polynomially bounded weights. It is unclear how we can make their ideas work even for a single 4-PTF with arbitrary weights.

Another reason our result does not work for d -PTFs for $d \geq 4$ is that the analogous decomposition to (1) yields a decomposition with $m \geq n^2$ terms, and the probabilistic degree of the Majority function on m variables is $\Omega(\sqrt{m})$ [26]. Thus, we only get a polynomial of degree $\Omega(n)$, which does not yield a suitably non-trivial bound on the probabilistic rank of these PTFs. It is an interesting open problem to see if any non-trivial upper bounds can be obtained for the probabilistic rank of PTFs of larger degree.

Related work

Over the past decade, there have been many works studying satisfiability algorithms for circuits using threshold gates and modular counting gates.

In the setting of ACC^0 , the class of constant-depth circuits using AND, OR, NOT and modular counting gates, the first non-trivial satisfiability algorithms were given by Williams [38] followed by improvements in [34], which also yields algorithms for the class $\text{ACC}^0 \circ \text{LTF}$. Further extensions have been obtained by works of [3, 11] but neither of these works allow us to replace the LTF gates by PTFs of higher degree.

For a single d -PTF, the best known algorithms use the memoization technique and have savings $\tilde{\Omega}(n^{1/d})$. [27, 7] This technique is grounded in the fact that the number of degree- d PTFs on n variables is $2^{O(n^{d+1})}$ [13], and this does not extend even to simple combinations of PTFs, such as conjunctions of polynomially many PTFs (or even LTFs).⁶

There are also restriction-based satisfiability algorithms for circuits made up of LTF and, more generally, PTF gates [12, 18]. Unfortunately, though, these algorithms only work when the size of the circuits (defined in terms of the number of wires) is slightly superlinear in the number of variables.

⁶ For example, it is easy to show that the number of distinct functions that can be written as a conjunction of s LTFs is always at least 2^s as long as $n > \log s$.

Paper organization

In Section 3, we upper bound the probabilistic rank of 3-PTFs (Lemma 14) and present a deterministic #SAT algorithm for $AC^0[MOD_p] \circ 3$ -PTF circuits (Theorem 17). In Section 4, we present a #SAT algorithm for $ACC^0 \circ 3$ -PTF circuits (Theorem 20).

2 Preliminaries

Computational model and asymptotic notation

All our algorithms can be implemented in the standard RAM model. The notations $\text{poly}(n)$, $\text{polylog}(n)$ are used to denote arbitrary, but fixed polynomial and polylogarithmic factors in n respectively. We use the $O^*(\cdot)$ and $\tilde{O}(\cdot)$ notation to suppress polynomial and logarithmic factors respectively.

The bit complexity of representing PTF's

A polynomial threshold function can be defined using multiple polynomials. Hence, the number of bits needed to represent a PTF depends on the bit complexity of the polynomial used to define it.

► **Definition 3** (Bit complexity of a polynomial). *Given a multilinear polynomial $P(x_1, x_2, \dots, x_n) = \sum_{S \subseteq [n]} c_S \cdot \prod_{j \in S} x_j$, the bit complexity of P is defined to be $w(P) = \log \left(\sum_{S \subseteq [n]} |c_S| \right)$.*

It is known that any d -PTF can be represented using a polynomial with bit complexity upper bounded by $O(n^d \log n)$ [20]. However, given an arbitrary d -PTF, it is not clear how to efficiently obtain such a low weight representation. Nevertheless, our algorithms in Section 3 can handle PTF's represented by polynomials with bit complexity up to 2^{n^δ} , for some fixed $\delta > 0$. For the purpose of proving lower bounds, we can always assume that the bit complexity of d -PTF functions is upper bounded by $O(n^d \log n)$.

Boolean circuits

We use the standard definitions for boolean circuits and circuit classes. We refer the reader to [33] for more details. The size of a circuit is defined to be the number of wires and its depth is defined to be the total number of layers. The circuit class AC^0 consists of constant depth circuits with AND, OR, NOT gates of unbounded fan-in. For any $m \in \mathbb{N}$, a MOD_m gate with fan-in t outputs 1 if the sum (over \mathbb{N}) of its inputs is divisible by m and outputs 0 otherwise. For any prime p , the class $AC^0[MOD_p]$ consists of circuits with AND, OR, NOT and MOD_p gates with unbounded fan-in. The class ACC^0 is defined as circuits of constant depth consisting of AND, OR, NOT and MOD_m gates of unbounded fan-in, for any fixed composite number m .

We define the following class of circuits, which can efficiently represent ACC^0 circuits [1, 2, 8].

► **Definition 4** (Symmetric functions and SYM^+ circuits). *A boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$ is called a symmetric function if there exists a function $F : [n] \rightarrow \{0, 1\}$ such that $f(x_1, \dots, x_n) = F(\sum_{i=1}^n x_i)$. A SYM^+ circuit is a depth two boolean circuit with an output gate that computes any symmetric function, and AND gates at the second layer.*

We also use modulus amplifying polynomials in our algorithms for $ACC^0 \circ 3$ -PTF satisfiability.

► **Definition 5** (Beigel-Tarui modulus amplifying polynomials [8]). *Let p be any prime. For any $t \in \mathbb{N}$, the t -th Beigel Tarui modulus amplifying polynomial is defined as*

$$F_t(z) = (-1)^t(z-1)^t \left(\sum_{i=0}^{t-1} \binom{t+i-1}{i} z^i \right) + 1.$$

This polynomial has the property that for all $z \geq 0, p \geq 2$, $F_t(z) \equiv 0 \pmod{p^t}$ if $z \equiv 0 \pmod{p}$ and $F_t(z) \equiv 1 \pmod{p^t}$ if $z \equiv 1 \pmod{p}$.

Probabilistic matrices and polynomials

We review some facts on probabilistic matrices and probabilistic polynomials.

► **Definition 6** (Probabilistic Polynomial). *Let \mathbb{F} be a field. A probabilistic polynomial over n variables is a distribution of polynomials from $\mathbb{F}[x_1, x_2, \dots, x_n]$. An ε -error probabilistic polynomial for a Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ is a probabilistic polynomial \mathbf{P} such that, for every $z \in \{0, 1\}^n$, $\Pr_{P \sim \mathbf{P}}[P(z) = f(z)] \geq 1 - \varepsilon$. A probabilistic polynomial \mathbf{P} is said to have degree D if $\Pr_{P \sim \mathbf{P}}[\deg(P) \leq D] = 1$.*

To derandomize algorithms that use probabilistic polynomials, we will need to design probabilistic polynomials that can be sampled using less randomness. To be precise, we say that a probabilistic polynomial \mathbf{P} can be sampled in time $T(n)$ using $r(n)$ bits of randomness if there exists a deterministic algorithm \mathcal{A} that takes as input a string $\sigma \in \{0, 1\}^{r(n)}$ and in time $T(n)$ outputs polynomials, such that for each polynomial P , $\Pr_{\sigma \sim \{0, 1\}^{r(n)}}[\mathcal{A}(\sigma) = P] = \mathbf{P}(P)$. We will also need to reduce the error of a probabilistic polynomial to any arbitrary $\varepsilon > 0$.

► **Lemma 7** (Randomness efficient error reduction for probabilistic polynomials). *Let f be any boolean function on n variables with an $1/3$ -error probabilistic polynomial of degree D that can be sampled using r bits of randomness in time $T(n)$. Then, for any $\varepsilon > 0$ there exists a probabilistic polynomial \mathbf{Q} for f with error ε , and degree $O(D \log(1/\varepsilon))$. Furthermore, \mathbf{Q} is of the form $\text{MAJ}(\mathbf{P}^1, \dots, \mathbf{P}^\ell)$, where $\ell = O(\log(1/\varepsilon))$ $\mathbf{P}^1, \dots, \mathbf{P}^\ell$ are probabilistic polynomials of degree D , and there exists an algorithm that uses $O(r + \log(1/\varepsilon))$ bits of randomness and outputs $\mathbf{P}^1, \dots, \mathbf{P}^\ell$ as sums of monomials in time $O(\ell \cdot T(n))$.*

Proof. We refer the reader to the proof of [25, Lemma 16]. ◀

The notion of probabilistic matrices and probabilistic rank generalize the notion of probabilistic polynomials and probabilistic degree.

► **Definition 8** (Probabilistic matrix [5]). *Let R be any ring. Consider any matrix $M \in R^{N \times N}$. A probabilistic matrix for M with error ε is a distribution \mathcal{M} over $N \times N$ matrices in $R^{N \times N}$ such that for each $i, j \in [N]^2$, $\Pr_{M' \sim \mathcal{M}}[M'[i, j] = M[i, j]] \geq 1 - \varepsilon$. A probabilistic matrix \mathcal{M} has rank r if every matrix sampled from \mathcal{M} has rank at most r . A matrix M has ε -probabilistic rank r if there exists a probabilistic matrix \mathcal{M} for M with error ε and rank r .*

Fast rectangular matrix multiplication

All our satisfiability algorithms rely on the fact that we can rapidly multiply rectangular matrices using Coppersmith's algorithm.

► **Lemma 9** (Coppersmith's algorithm [14, 34]). *For sufficiently large N and $\alpha \leq 0.172$ and any field \mathbb{F} with $2^{\text{polylog}(N)}$ elements, there exists an algorithm that takes matrices $A \in \mathbb{F}^{N \times N^\alpha}, B \in \mathbb{F}^{N^\alpha \times N}$ as inputs and outputs the matrix $A \cdot B$ in time $N^2 \cdot \text{polylog}(N)$*

We refer the reader to the appendix of [34] for a proof.

3 The probabilistic rank of 3-PTFs

We start with recalling that there exist probabilistic polynomials for AC^0 circuits with degree polylogarithmic in the size of the circuit.

► **Lemma 10** (Randomness efficient probabilistic polynomials for AC^0 [25]). *Let p be any prime. Any $AC^0[MOD_p]$ circuit of depth d and size s on n variables has an ε -error probabilistic polynomial P over the field \mathbb{F}_p of degree $O((\log s)^{d-1} \log(1/\varepsilon))$, and a polynomial can be sampled from the distribution in time $n^{O((\log s)^{d-1} \log(1/\varepsilon))}$ and using $O(\log(s) + \log(1/\varepsilon))$ random bits.*

► **Lemma 11.** *There exists a probabilistic polynomial for any symmetric boolean function of error $1/3$ and degree $d = O(\sqrt{n} \log(n))$. A polynomial can be sampled from the distribution in time $O\left(\binom{n}{d} \text{poly}(n)\right)$ using $O(\log^2(n))$ random bits.*

Proof. Let $S \subseteq [n]$ be the set of all $k \in [n]$ such that $f(x) = 1$ if $\sum_{i=1}^n x_i = k$. Hence, $f(x) = \bigvee_{k \in S} (\sum_{i=1}^n x_i = k)$. The predicate $(\sum_{i=1}^n x_i = k)$ can be computed by a $AND_2 \circ MAJ$ circuit with MAJ gates with $O(n)$ fan-in. Now, MAJ has a $1/3$ -error probabilistic polynomial over any field of degree $O(\sqrt{n})$, and a polynomial can be sampled from the probabilistic polynomial using $O(\log^2 n)$ random bits [3]. The AND_2 gates can be converted into polynomials of degree two trivially. Hence, using Lemma 7, we can sample a probabilistic polynomial of error ε for the predicate $(\sum_{i=1}^n x_i = k)$ of degree $\sqrt{n} \log(1/\varepsilon)$ using $O(\log^2(n) + \log(1/\varepsilon))$ random bits. Finally, we can convert the top OR gate into a probabilistic polynomial of constant degree and error $1/10$ using $O(\log n)$ random bits (Lemma 10). Setting $\varepsilon = \frac{1}{10n}$ for the MAJ gates and composing these polynomials, we get a probabilistic polynomial for the symmetric function of degree $O(\sqrt{n} \log(n))$ using $O(\log^2 n)$ random bits and error $1/3$. ◀

► **Lemma 12.** *There exists an $AC^0 \circ \text{SYM}_n$ circuit with $\text{poly}(n, M)$ gates that takes as input n integers of M bits each and outputs their sum.*

Proof. Suppose the M bit numbers are A_1, A_2, \dots, A_n , where $A_i = A_{i,M-1}A_{i,M-2} \dots A_{i,0}$, with $A_{i,j} \in \{0, 1\}$. Let $\ell = \lceil \log n \rceil$. Let $B = B_{\ell+M-1}B_{\ell+M-2} \dots B_0 = \sum_{i=1}^n A_i$. For each $j = 1, 2, \dots, M$, let $y_j = y_{j,\ell-1}y_{j,\ell-2} \dots y_{j,0} = \sum_{i=1}^n A_{i,j}$. There exist functions $f_0, f_1, \dots, f_{\ell-1} : [n] \rightarrow \{0, 1\}$ such that $y_{j,k} = f_k(\sum_{i=1}^n A_{i,j})$ for each $k = 0, 2, \dots, \ell - 1$. Then, we can rewrite the sum B as follows.

$$\begin{aligned} B &= \sum_{j=0}^{M-1} 2^j \cdot \sum_{i=1}^n A_{i,j} = \sum_{j=0}^{M-1} 2^j \cdot y_j = \sum_{j=0}^{M-1} 2^j \cdot \left(\sum_{k=0}^{\ell-1} 2^k f_k \left(\sum_{i=1}^n A_{i,j} \right) \right) \\ &= \sum_{j=0}^{M-1} \sum_{k=0}^{\ell-1} 2^{j+k} \cdot f_k \left(\sum_{i=1}^n A_{i,j} \right). \end{aligned}$$

Switching the order of the summation, this is equal to $\sum_{k=0}^{\ell-1} 2^k \sum_{j=0}^{M-1} 2^j \cdot f_k(\sum_{i=1}^n A_{i,j})$. For each $k = 0, 2, \dots, \ell - 1$ and $j = 0, 1, \dots, M - 1$, the function $f_k(\sum_{i=1}^n A_{i,j})$ can be implemented using a single SYM gate with fan-in n . Hence, the numbers $C_k = 2^k \sum_{j=0}^{M-1} 2^j \cdot f_k(\sum_{i=1}^n A_{i,j})$ can be computed in parallel, for each $k = 0, 1, \dots, \ell - 1$ as bit strings of length $\ell = \lceil \log(n) \rceil$ using a layer of ℓM SYM gates of fan-in n . What remains is to compute $B = \sum_{k=0}^{\ell-1} C_k$. It is known that addition of $\log N$ numbers with N bits each can be implemented using constant depth circuits of size $\text{poly}(N)$ [33]. We can use these circuits with $N = M + n$ to prove the lemma. ◀

► **Remark 13.** A very similar $\text{AC}^0 \circ \text{MAJ}$ -circuit for this problem was first constructed in [19]. However, their construction yields MAJ gates of fan-in n^2 . The results in this section require circuits with MAJ gates with fan-in at most n .

► **Lemma 14.** *Let p be any prime and $F : \{0, 1\}^n \rightarrow \{0, 1\}$ be a 3-PTF defined by a polynomial $P \in \mathbb{Z}[z_1, z_2, \dots, z_n]$ and bit complexity upper bounded by M . Then there exist functions $f, g : \{0, 1\}^{n/2} \rightarrow \{0, 1\}^{n'/2}$, with $n' = O(nM)$ and a probabilistic polynomial $\mathbf{Q} \subseteq \mathbb{F}_p[x_1, \dots, x_{n'}]$ of degree $\sqrt{n} \cdot \text{polylog}(n, M)$, such that for each $x, y \in \{0, 1\}^{n/2}$,*

$$\Pr_{Q \sim \mathbf{Q}} [Q(f(x), g(y)) = F(x, y)] \geq 2/3.$$

Given x, y , $f(x)$ and $g(y)$ can be computed in time $\text{poly}(n, M)$ and a polynomial can be sampled from \mathbf{Q} using $O(\log M + \log^2 n)$ random bits in time $2^{\sqrt{n} \cdot \text{polylog}(n, M)}$.

► **Remark 15.** This implies that there exists an absolute constant $\delta > 0$ such that 3-PTFs defined using polynomials of bit complexity upper bounded by 2^{n^δ} have probabilistic degree $o(n)$. In several situations, it may also be reasonable to assume that $M = \text{poly}(n)$, as noted in Section 2.

Proof. Note that any cubic polynomial $P(x, y)$ can be decomposed in the following fashion.

$$P(x, y) = \sum_{j=1}^{n/2} x_j Q_j(y) + \sum_{j=1}^{n/2} y_j R_j(x) + \tilde{Q}(y) + \tilde{R}(x),$$

for quadratic polynomials $Q_j, R_j, \tilde{Q}, \tilde{R}$. We can now define $f(x)$ and $g(y)$ to be the bits of the integers $R_j(x), \tilde{R}(x)$ and $Q_j(y), \tilde{Q}(y)$ (with an extra bit to store the sign) for each $j \in [n/2]$ respectively. The number of bits needed to express $f(x)$ and $g(y)$ is upper bounded by $O(\log(\tilde{R}(x)) + \sum_{j=1}^{n/2} \log(R_j(x)))$ and $O(\log(\tilde{R}(y)) + \sum_{j=1}^{n/2} \log(Q_j(y)))$ respectively. By the definition of bit-complexity of a polynomial, the values of $R_j(x), \tilde{R}(x), Q_j(y), \tilde{Q}(y)$ are upper bounded by $2^{w(P)}$ and hence by 2^M for each $j \in [n/2], x \in \{0, 1\}^{n/2}$, which means that $f(x)$ and $g(y)$ are bit strings of length $O(nM)$.

Converting F into a $\text{poly}(n, M)$ sized $\text{AC}^0[\oplus] \circ \text{SYM}_n \circ \text{AND}_2$ circuit. We can define a circuit C , which is very similar to the circuit described in the proof of [34, Theorem 1.4] Let $\text{SUM}_{n, M}$ denote a circuit that takes as input up to n natural numbers with bit complexity M and outputs their $t = O(M + \log n)$ bit sum. Let LEQ_t denote a circuit that takes as input two t bit integers a, b and outputs 1 if $a \leq b$ and 0 otherwise. The bottom-most layer of the circuit consists of AND gates with fan-in two. This layer outputs the bits for $x_j Q_j(y), y_j R_j(x)$. Now, given the bits of the integers $x_j Q_j(y), y_j R_j(x), \tilde{Q}(y), \tilde{R}(x)$ for all $j \in [m/2]$, we can compute the output of the 3-PTF using a $\text{LEQ}_t \circ \text{SUM}_{n, M}$ circuit. This can be accomplished by adding up all the negative and positive integers in parallel using two $\text{SUM}_{n, M}$ subcircuits and then comparing the outputs of these two subcircuits using an LEQ_t subcircuit. Now, we note that LEQ_t has $\text{poly}(t)$ sized AC^0 circuit of depth 4 (see [10] and [37, Appendix A]⁷) and $\text{SUM}_{n, M}$ has a $\text{AC}^0 \circ \text{SYM}_n$ circuit with $\text{poly}(n, M)$ gates by Lemma 12. Hence, C has $\text{poly}(n, M)$ gates.

⁷ this follows from the following fact: $\text{LEQ}(x, y) = \left(\bigwedge_{i=1}^t (1 + x_i + y_i) \right) \vee \bigvee_{i=1}^t \left((1 + x_i) \wedge y_i \wedge \bigwedge_{j=1}^{i-1} (1 + x_j + y_j) \right)$, where $\text{LEQ}(x, y) = 1$ if and only if $x \leq y$.

Converting C into a probabilistic polynomial. The next step is to convert C into a probabilistic polynomial \mathbf{Q} . Each of the AND_2 gates at the bottom can be trivially represented by a constant degree polynomial. Next, we convert each symmetric gate into a probabilistic polynomial of error $\varepsilon = \frac{1}{(nM)^c}$ for sufficiently large $c \in \mathbb{R}$ and degree $O(\sqrt{n} \log(1/\varepsilon) \log n) = O(\sqrt{n} \log(M) \text{polylog}(n))$ and $2^{O(\sqrt{n} \log(M) \text{polylog}(n))}$ monomials using Lemma 11. We can use union bound to show that the probability that any of the polynomials for the symmetric gates err is upper bounded by $1/10$. The remaining part of C is an AC^0 circuit with $S = \text{poly}(n, M)$ gates. This can be converted into a probabilistic polynomial of degree $\text{polylog}(n, M)$, $O(M \log n)$ inputs and error $1/10$. Hence, \mathbf{Q} has degree $\sqrt{n} \cdot \text{polylog}(n, M)$. We can then use Lemma 7 obtain a probabilistic polynomial with error ε and degree $\sqrt{n} \cdot \text{polylog}(n, M) \log(1/\varepsilon)$ for any $\varepsilon > 0$.

Randomness and time complexity. The amount of random bits needed for the probabilistic polynomials for the symmetric gates is $O(\log^2(n) + \log(M))$ (Lemma 11). Because we are using the union bound, we can just reuse the same random bits for all the symmetric gates. The probabilistic polynomials for the AC^0 part of C can be constructed using $O(\log(S)) = O(\log M + \log n)$ random bits (Lemma 10). ◀

Lemma 14 implies that all 3-PTF functions have low probabilistic rank.

► **Corollary 16.** 3-PTF functions over n variables have probabilistic rank $2^{\tilde{O}(\sqrt{n})}$.

Proof. Any 3-PTF function $F : \{0, 1\}^n \rightarrow \{0, 1\}$ can be defined using a cubic polynomial P with $\text{poly}(n)$ bit complexity [20]. Hence, we can use Lemma 14 to define probabilistic matrices $\mathbf{A} \subseteq \mathbb{F}_p^{2^{n/2} \times r}$, $\mathbf{B} \subseteq \mathbb{F}_p^{r \times 2^{n/2}}$ for $r = 2^{\tilde{O}(\sqrt{n})}$ such that for each $x, y \in \{0, 1\}^{n/2}$, $\Pr_{\mathbf{A} \sim \mathbf{A}, \mathbf{B} \sim \mathbf{B}}[(\mathbf{A} \cdot \mathbf{B})_{x,y} = F(x, y)] \geq 2/3$. We can sample matrices from the distributions \mathbf{A}, \mathbf{B} as follows. We first sample a polynomial Q from the distribution \mathbf{Q} defined in Lemma 14. The columns of A (and the rows of B) are indexed by monomials in Q . Define the entry of A defined by a monomial $c_S \prod_{i \in S} x_i$ for a set $S \subseteq [n]$ and an assignment $x \in \{0, 1\}^{n/2}$ to be $c_S \prod_{i \in S \cap \{1, \dots, n/2\}} (f(x))_i$, and the entry of B defined by the same monomial and an assignment $y \in \{0, 1\}^{n/2}$ to be $\prod_{i \in S \cap \{n/2+1, \dots, n\}} (g(y))_i$, where $(f(x))_i$ and $(g(y))_i$ denote the i -th entry of the strings $f(x)$ and $g(y)$ respectively. Hence, $(\mathbf{A} \cdot \mathbf{B})_{x,y} = P(x, y)$, which is equal to $F(x, y)$ with probability at least $2/3$. ◀

We use Lemma 14 to obtain a satisfiability algorithm for 3-PTF-SAT, as well as for more expressive complexity classes involving 3-PTF gates. The following theorem follows directly from combining Lemma 14 with Lemma 10 and derandomization techniques developed in [25] to design a deterministic #SAT algorithm for $\text{AC}^0[\text{MOD}_p] \circ 3\text{-PTF}$ circuits.

► **Theorem 17.** Let p be any prime number. There exists a deterministic algorithm that counts the number of satisfying assignments to $\text{AC}^0[\text{MOD}_p] \circ 3\text{-PTF}$ circuits of depth d in time $\text{poly}(n, M) \cdot 2^{n - \frac{\sqrt{n}}{\text{polylog}(n, M) \log^{d-1}(s)}}$.

► **Remark 18.** This algorithm can handle circuits of size up to $2^{n^{\delta'}}$, for some δ' that depends on d as well as circuits with 3-PTFs defined by polynomials with weights upper bounded by 2^{n^δ} , for the previously defined δ .

Proof. We start with designing a probabilistic polynomial for C . Let F_1, F_2, \dots, F_{s_1} be the 3-PTF gates in C , for $s_1 \leq s$. Using Lemma 14, design probabilistic polynomials $\mathbf{Q}^1, \dots, \mathbf{Q}^{s_1}$ with $2^{\sqrt{n} \cdot \log(s) \text{polylog}(n, M)}$ monomials and functions $f^1, g^1, \dots, f^{s_1}, g^{s_1}$, such that $\Pr_{Q^i \sim \mathbf{Q}^i}[Q^i(f^i(x), g^i(y)) = F_i(x, y)] \geq 1 - \frac{1}{2^{10s}}$. The polynomials can be sampled using a

common set of $O(\log M + \log^2 n + \log(s))$ random bits, using Lemma 14 and Lemma 7. We can sample a probabilistic polynomial of error $1/3$ for the AC^0 part of C of degree $O(\log^{d-2} s)$ and $O(\log s)$ random bits using Lemma 10. Composing these polynomials and using Lemma 7 again, we can obtain a probabilistic polynomial \mathbf{P} with $2^{\sqrt{n} \cdot \text{polylog}(n, M) \cdot \log^{d-1} s \log(1/\varepsilon)}$ monomials using $O(\log M + \log^2 n + \log s + \log(1/\varepsilon))$ bits of randomness and functions $f, g : \{0, 1\}^{n/2} \rightarrow \{0, 1\}^{O(nMs)}$ such that $\Pr_{P \sim \mathbf{P}}[P(f(x), g(y)) = C(x, y)] \geq 1 - \varepsilon$. Furthermore, $\mathbf{P} = \text{MAJ}(\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_\ell)$, for probabilistic polynomials $\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_\ell$ with $2^{\sqrt{n} \cdot \text{polylog}(n, M) \cdot \log^{d-1} s}$ monomials and $\ell = O(\log(1/\varepsilon))$.

The satisfiability algorithm now follows from a standard approach to designing circuit satisfiability algorithms, first used by Williams [38] for ACC^0 satisfiability algorithms. Later on, in [25], this algorithm was derandomized and extended to count the number of satisfying assignments. Pick some $n' \leq n/2$, and let $m := n - n'$. For each $x, y \in \{0, 1\}^{m/2}$, let $G(x, y) := |\{z \in \{0, 1\}^{n'} : C(z, x, y) = 1\}|$. For each $z \in \{0, 1\}^{n'}$, let C^z denote the circuit C , with the first n' variables restricted according to the assignment z . By definition,

$$G(x, y) = \sum_{z \in \{0, 1\}^{n'}} C(z, x, y) = \sum_{z \in \{0, 1\}^{n'}} C^z(x, y),$$

and $\#\text{SAT}(C) = \sum_{x, y \in \{0, 1\}^{m/2}} G(x, y)$. Now, for each $z \in \{0, 1\}^{n'}$, define functions f^z, g^z and probabilistic polynomials \mathbf{P}^z on $O(mMs)$ variables such that $\Pr_{P^z \sim \mathbf{P}^z}[P^z(f^z(z), g^z(y)) = C^z(x, y)] \geq 1 - 1/2^{10n'}$ using Lemma 14. These can be sampled using $r = O(\log M + \log^2 n + \log s + n')$ common random bits. For each $\sigma \in \{0, 1\}^r$, let $P^{z, \sigma} = \text{MAJ}(P_1^{z, \sigma}, \dots, P_\ell^{z, \sigma})$ for $\ell = O(n')$ denote the polynomial sampled from \mathbf{P}^z using the string σ as the random bits. Hence,

$$\sum_{\sigma \in \{0, 1\}^r} P^{z, \sigma}(f(x), g(y)) = \begin{cases} \geq 2^r \left(1 - \frac{1}{2^{10n'}}\right) & \text{if } C^z(x, y) = 1 \\ \leq \frac{2^r}{2^{10n'}} & \text{if } C^z(x, y) = 0 \end{cases}$$

Summing over all $z \in \{0, 1\}^{n'}$,

$$\begin{aligned} 2^r \left(1 - \frac{1}{2^{9n'}}\right) G(x, y) &\leq \sum_{\sigma \in \{0, 1\}^r} \sum_{z \in \{0, 1\}^{n'}} (P^{z, \sigma}(f^z(x), g^z(y)) \mod p) \\ &\leq 2^r \left(1 + \frac{1}{2^{9n'}}\right) G(x, y) \end{aligned} \tag{2}$$

where the sum is over \mathbb{Z} , not \mathbb{F}_p .

Fourier coefficients

Letting MAJ be a function from $\mathbb{Z}_p^\ell \rightarrow \mathbb{Z}_p$ ⁸, we can define coefficients $k_a \in \mathbb{C}$ for each $a \in \mathbb{F}_p^\ell$, such that $\text{MAJ}(z_1, \dots, z_\ell) = \sum_{a \in \mathbb{F}_p^\ell} k_a \cdot \left(\sum_{i=1}^\ell a_i z_i \mod p\right)$. The quantity $\left(\sum_{i=1}^\ell a_i z_i \mod p\right)$ is interpreted as a real number. The numbers k_a can be computed for each $a \in \mathbb{F}_p^\ell$ in time $2^{O(\ell)}$ using the FFT algorithm. We refer the reader to [23] for more details. Define the polynomials $P_a^{z, \sigma} := \sum_{i=1}^\ell a_i P_i^{z, \sigma}$, and let F_t denote the t -th Beigel-Tarui modulus amplifying polynomial (Definition 5).

⁸ We just let MAJ be the standard majority function on $z \in \{0, 1\}^\ell$, and arbitrarily set it to (say) 0 on all other $z \in \mathbb{F}_p^\ell$

The parameters

Set $n' = \frac{\sqrt{n}}{p(\log M, \log n) \log^{d-1} s}$ for a suitable polynomial p . Let $m := n - n'$. We remind the reader that $\ell = An'$ and that $r = B(\log M + \log^2 n + \log s + \log(1/\varepsilon)) = O(\log M + \log^2 n + \log s + \log n')$ for constants A, B . We choose $t = n'/\log_2(p) + \ell + r/\log_2(p) + 10$. Now, we can present the deterministic algorithm.

1. Calculate the coefficients k_a for each $a \in \mathbb{F}_p^\ell$ for MAJ.
2. For each $z \in \{0, 1\}^{n'}$, $\sigma \in \{0, 1\}^r$, $a \in \mathbb{F}_p^\ell$, construct the polynomials $P_a^{z, \sigma}$ as sums of monomials and functions f_i, g_i , using Lemma 14.
3. For each $i \in \{1, 2, \dots, 2^{n'}\}$, construct as a sum of monomials, the polynomial

$$R := \sum_{z \in \{0, 1\}^{n'}} \sum_{\sigma \in \{0, 1\}^r} \sum_{a \in \mathbb{F}_p^\ell} k_a \cdot F_t \circ P_a^{z, \sigma}.$$

4. For each $x, y \in \{0, 1\}^{m/2}$, evaluate $R(f(x), g(y))$ using Coppersmith's algorithm.
5. Output $\sum_{(x, y) \in \{0, 1\}^{m/2} \times \{0, 1\}^{m/2}} [(R(f(x), g(y)) \bmod p^t) / 2^r]$ (where $[\cdot]$ denotes the nearest integer function).

Running time

Step 1 takes time $2^{O(\ell)} = 2^{O(n')}$, which can be upper bounded (very loosely) by $2^{n/10} \cdot \text{poly}(n, M, s)$. Step 2 takes time $2^{n' + r + \ell \log_2 p} \cdot 2^{\sqrt{n} \log^{d-1} s \cdot \text{polylog}(n, M)}$, which can be upper bounded by $2^{n/10} \cdot \text{poly}(n, M, s)$ as well. To upper bound the complexity of Step 3, note that R has $2^{n' + r + \ell \log_2 p} \cdot 2^{t \sqrt{n} \log^{d-1} s \cdot \text{poly}(\log n, \log M)}$ monomials. Choosing the polynomial p appropriately, this can be upper bounded by $2^{0.01n}$. Hence, we can construct the polynomial for each a, z, σ in time $\text{poly}(n, M, s) \cdot 2^{n/10}$. Because the number of monomials in R is upper bounded by $2^{0.05m}$, Step 4 takes $2^m \text{poly}(m)$ time (Lemma 9). Step 5 runs in $2^m \text{poly}(m)$ time as well.

Correctness

From the properties of the modulus amplifying polynomials (Definition 5), $F_t(P_a^{z, \sigma}(f(x), g(y))) \equiv 0 \bmod p^t$ if $P_a^{z, \sigma}(f(x), g(y)) \equiv 0 \bmod p$, and $F_t(P_a^{z, \sigma}(f(x), g(y))) \equiv 1 \bmod p^t$ if $P_a^{z, \sigma}(f(x), g(y)) \equiv 1 \bmod p$. Because we have chosen t such that $p^t > p^\ell 2^r 2^{n'}$,

$$\begin{aligned} & \sum_{z \in \{0, 1\}^{n'}} \sum_{\sigma \in \{0, 1\}^r} (P^{z, \sigma}(f(x), g(y)) \bmod p) \\ &= \sum_{z \in \{0, 1\}^{n'}} \sum_{\sigma \in \{0, 1\}^r} (\text{MAJ}(P_1^{z, \sigma}(f(x), g(y)), \dots, P_\ell^{z, \sigma}(f(x), g(y)) \bmod p)) \\ &= \sum_{z \in \{0, 1\}^{n'}} \sum_{\sigma \in \{0, 1\}^r} \sum_{a \in \mathbb{F}_p^\ell} k_a \left(\sum_{i=1}^{\ell} a_i P_i^{z, \sigma}(f(x), g(y)) \bmod p \right) \\ &= \sum_{z \in \{0, 1\}^{n'}} \sum_{\sigma \in \{0, 1\}^r} \sum_{a \in \mathbb{F}_p^\ell} k_a (P_a^{z, \sigma}(f(x), g(y)) \bmod p) \\ &= \sum_{z \in \{0, 1\}^{n'}} \sum_{\sigma \in \{0, 1\}^r} \sum_{a \in \mathbb{F}_p^\ell} k_a (F_t(P_a^{z, \sigma}(f(x), g(y))) \bmod p^t) \\ &= \sum_{z \in \{0, 1\}^{n'}} \sum_{\sigma \in \{0, 1\}^r} \sum_{a \in \mathbb{F}_p^\ell} k_a F_t(P_a^{z, \sigma}(f(x), g(y))) \bmod p^t \\ &= R(f(x), g(y)) \bmod p^t. \end{aligned}$$

Note that all summations above are over \mathbb{Z} unless they are enclosed by brackets. Now, consider the quantity $\sum_{z \in \{0,1\}^{n'}} \sum_{\sigma \in \{0,1\}^r} (P^{z,\sigma}(f(x), g(y)) \bmod p)$. Using Equation (2), $R(f(x), g(y)) \bmod p^t = \sum_{z \in \{0,1\}^{n'}} \sum_{\sigma \in \{0,1\}^r} (P_a^{z,\sigma}(f(x), g(y)) \bmod p) \in [2^r(1 - 1/2^{9n'}), 2^r(1 + 1/2^{9n'})G(x, y)]$. Hence, $[\frac{1}{2^r} (R(f(x), g(y)) \bmod p^t)] = G(x, y)$. Because $\#SAT(C) = \sum_{x,y} G(x, y)$, the algorithm indeed outputs $\#SAT(C)$. ◀

4 #SAT algorithms for $ACC^0 \circ 3\text{-PTF}$ circuits

In this section, we show that Lemma 14 can be combined with well known techniques [1, 2, 38] for simplifying $ACC^0 \circ 3\text{-PTF}$ circuits to design #SAT algorithms.

Notation

We say that a $SYM \circ ACC^0 \circ 3\text{-PTF}$ circuit has parameters (n, s, M) if it has n inputs, s wires, and contains 3-PTF gates defined by polynomials with bit-complexity upper bounded by M .

We present a better than brute force algorithm to evaluate $SYM \circ ACC^0 \circ 3\text{-PTF}$ circuits on all inputs.

► **Theorem 19.** *There exists an absolute constant $\delta \in (0, 1)$ and a function $\varepsilon : \mathbb{N} \rightarrow (0, 1)$ such that the following holds. There exists a deterministic algorithm that takes as input a $SYM \circ ACC^0 \circ 3\text{-PTF}$ circuit C with parameters (n, s, M) such that $s \leq 2^{n^{\varepsilon(d)}}$, $M \leq 2^{n^\delta}$ that can enumerate all satisfying assignments to C in time $O^*(2^n)$.*

Using the approach developed in [34] for reducing #SAT to rapid evaluation of large circuits of all inputs, we can design a #SAT algorithm for $ACC^0 \circ 3\text{-PTF}$ circuits. We refer the reader to the proof of [34, Theorem 1.2].

► **Theorem 20.** *There exists an absolute constant $\delta \in (0, 1)$ and a function $\varepsilon : \mathbb{N} \rightarrow (0, 1)$ such that the following holds. There exists a deterministic algorithm that takes as input an $ACC^0 \circ 3\text{-PTF}$ circuit C with parameters (n, s, M) such that $s \leq 2^{n^{\varepsilon(d)}}$, $M \leq 2^{n^\delta}$ that can count the number of satisfying assignments to C in time $O^*(2^{n - n^{\varepsilon(d)}})$.*

Using [21, Theorem 1.2], we can infer the following circuit lower bound, which is an extension of [21, Theorem 1.3].

► **Corollary 21.** *For every constant k, d, m , there exists $e \geq 1$ and a problem in $NTIME(2^{\log^e n})$ which does not have $ACC^0 \circ 3\text{-PTF}$ circuits with MOD_m gates of depth d and size $2^{\log^k n}$.*

The rest of this section is devoted to the proof of Theorem 19. The first step is to convert $SYM \circ ACC^0 \circ 3\text{-PTF}$ circuits into equivalent SYM^+ circuits.

► **Lemma 22.** *There exists a function $c : \mathbb{N} \rightarrow \mathbb{N}$ and an absolute constant $e \in \mathbb{N}$ such that the following holds. Let C be any $SYM \circ ACC^0 \circ 3\text{-PTF}$ circuit with parameters (n, s, M) . Then, there exists a SYM^+ circuit D and functions f, g such that for each $x, y \in \{0, 1\}^{n/2}$, $C(x, y) = D(f(x), g(y))$. The circuit D has size $2^{O(\log^{c(d)} s)} + 2^{O(\log^e s \sqrt{n}(\log n \log M)^e)}$, and there exists a deterministic algorithm that takes C as input and outputs D in time $2^{O(\log^{c(d)} s)} + 2^{O(\log^e s \sqrt{n}(\log n \log M)^e)}$. The functions f and g output strings of length $O(nMs)$ and are computable in time $O(s \cdot \text{poly}(n, M))$. The output of the symmetric gate can be computed in time $2^{O(\log^{c(d)} s)}$, given the number of input wires that evaluate to 1.*

Proof. Let g denote the symmetric gate at the top, with fan-in h , and let $H : [h] \rightarrow \{0, 1\}$ be the symmetric function it computes. Let C_i , for $i \in [h]$ denote the subcircuits of C that feed into the top SYM gate.

Step 1: Replace the 3-PTF gates with probabilistic polynomials. Suppose that C has s_1 3-PTF gates F^1, F^2, \dots, F^{s_1} in the bottom layer, where $s_1 \leq s$. Let $\varepsilon = \frac{1}{10hs}$, and construct probabilistic polynomials $\mathbf{P}^1, \dots, \mathbf{P}^{s_1}$ with error ε for each of the 3-PTF gates, with functions $f^1, f^2, \dots, f^{s_1}, g^1, g^2, \dots, g^{s_1}$ such that for each $j \in [s_1]$, $\Pr_{P \sim \mathbf{P}^j}[P(f^j(x), g^j(y)) = F^j(x, y)] \geq 1 - \frac{1}{10hs}$ using Lemma 14. Note that each polynomial can be represented as an XOR \circ AND circuit, of size at most $2^{O(\sqrt{n} \cdot \log s \cdot \text{polylog}(n, M))}$. Hence, we can replace the SYM \circ ACC⁰ \circ 3-PTF circuit C with a probabilistic SYM \circ ACC⁰ \circ AND circuit C' of size $s' = 2^{O(\sqrt{n} \cdot \log s \cdot \text{polylog}(n, M))}$, depth $d + 1$ such that for each $x, y \in \{0, 1\}^{n/2}$, $C(x, y) = C'(f^1(x), \dots, f^{s_1}(x), g^1(y), \dots, g^{s_1}(y))$ with probability at least $9/10$.⁹ The fan-in of the AND gates at the bottom is upper bounded by $O(\sqrt{n} \log s \cdot \text{polylog}(n, M))$ and the number of AND gates is upper bounded by $2^{O(\sqrt{n} \cdot \log s \cdot \text{polylog}(n, M))}$. Note that if the wires from the AND gates at the bottom layer to the rest of the ACC⁰ circuit are not counted, C' has size $O(s)$, and that the polynomials $\mathbf{P}^1, \dots, \mathbf{P}^{s_1}$ and hence the circuit C' can be sampled using $O(\log^2 n + \log s)$ random bits.

Step 2: Replace the SYM \circ ACC⁰ circuit C' with a deterministic SYM⁺ circuit. We use the ideas of Allender and Gore [1, 2]. We refer the reader to Williams [38, Appendix A] for more details. Specifically, we will refer to transformations 1,2,3 and 4, as defined by Williams. Here, we just demonstrate where our proof differs from theirs.

Step 2a. Apply transformation 1 to the circuit C' . This transformation only acts on the ACC⁰ part of C' and leaves the bottom layer of AND gates untouched. The MOD _{m} gates for composite m are eliminated and replaced by subcircuits with prime modulo gates and AND gates. Further, all the AND and OR gates are all replaced by fixed depth circuits with only AND gates with $\text{polylog}(n)$ fan-in and MOD _{p} gates (for some prime number p). All these gates share a common set of $\text{polylog}(s)$ probabilistic inputs. Let the new circuit be C'' , and let $C''_1, C''_2, \dots, C''_h$ be the subcircuits that feed into the symmetric gate g . The circuit C'' has size $\text{poly}(s)$, without counting the bottom layer. Over the probabilistic inputs as well as the $O(\log^2 n + \log s)$ random bits used to sample the probabilistic polynomials, $\Pr[C''_i(f^1(x), \dots, f^{s_1}(x), g^1(y), \dots, g^{s_1}(y)) = C_i(x, y)] \geq \frac{1}{10h}$. This step takes $\text{poly}(s)$ time.

Step 2b. Let $r = O(\log^2 n + \text{polylog}(s))$ denote the number of random bits needed in total to sample C'' . For each $\sigma \in \{0, 1\}^r$, let C''_σ denote the circuit C'' sampled using the string σ as the random bits, and let $C''_{i,\sigma}$ for $i \in [h]$ denote the corresponding subcircuits that feed into the top SYM gate. Now, suppose that $\sum_{i \in [h]} C_i(x, y) = t$. For each $x, y \in \{0, 1\}^{n/2}$, $\frac{1}{2^r} \sum_{\sigma \in \{0, 1\}^r} C''_{i,\sigma}(f(x), g(y)) \in [1 - 1/(10h), 1]$ if $C_i(x, y) = 1$ and $\frac{1}{2^r} \sum_{\sigma \in \{0, 1\}^r} C''_{i,\sigma}(f(x), g(y)) \in [0, 1/(10h)]$ if $C_i(x, y) = 0$. Hence, $\frac{1}{2^r} \sum_{i \in [h]} \sum_{\sigma \in \{0, 1\}^r} C''_{i,\sigma}(f(x), g(y)) \in [t - 1/10, t + 1/10]$. Now, we can make the circuit C'' deterministic. Define the circuit C''' as follows. The top symmetric gate g' has fan-in $2^r \cdot h$, and the symmetric function is $H' : [2^r \cdot h] \rightarrow \{0, 1\}$, defined by $H'(z) = H(\lceil \frac{1}{2^r} z \rceil)$, where $\lceil \cdot \rceil$ denotes the closest integer function. The circuits $C''_{i,\sigma}$ feed into g' for each $i \in [h], \sigma \in \{0, 1\}^r$. The size of the circuit C''' is $2^r \text{poly}(s) = 2^{\text{polylog}(n)} 2^{\text{polylog}(s)}$ without

⁹ Technically, this is not an ACC⁰ circuit as it contains MOD _{m} gates as well as MOD₂ gates, but rest of the transformations will work for this circuit as well.

counting the bottom layer and is $2^r \left(\text{poly}(s) + 2^{\sqrt{n} \log(s) \text{polylog}(n, M)} \right) = 2^{\text{polylog}(s)} 2^{\text{polylog}(n)} + 2^{\text{polylog}(s)} 2^{\sqrt{n} \log(s) \text{polylog}(n, M)}$ in total. Note that this step can also be completed in $2^{\text{polylog}(s)} 2^{\text{polylog}(n)} + 2^{\text{polylog}(s)} 2^{\sqrt{n} \log(s) \text{polylog}(n, M)}$ time.

Step 2c. Now, we perform transformation 3 and transformation 4 on C''' to get a new deterministic circuit C'''' . As earlier, these transformations leave the bottom layer of AND gates untouched. The AND gates of C''' are pushed to the bottom layer, and the MOD_p gates get absorbed into the symmetric function. Let s''' be the size of the ACC^0 part of C''' . There exists a function $f: \mathbb{N} \rightarrow \mathbb{N}$ such that the ACC^0 part of C''' is replaced with a SYM^+ circuit with size $2^{\log^{f(d-1)}(s''')}$. This transformation takes $2^{\log^{f(d-1)}(s''')}$ time and given the number of input wires that evaluate to 1, the symmetric function itself can be computed in time $2^{\log^{f(d-1)}(s''')}$. As before, the bottom layer of AND gates remains untouched. All the previously defined polynomial factors do not depend on the depth of the circuit. Hence, we can define a function $c: \mathbb{N} \rightarrow \mathbb{N}$ such that the final circuit is a SYM^+ circuit with $2^{\log^{c(d)} s} + 2^{\log^e s} 2^{\sqrt{n}(\log n \log M)^e}$ wires, for a fixed absolute constant $e \in \mathbb{N}$ that can be chosen to absorb all the polynomial factors. ◀

Using this lemma, we can efficiently evaluate a $\text{SYM} \circ \text{ACC}^0 \circ 3\text{-PTF}$ circuit C of size s on all inputs:

1. Use Lemma 22 to convert C to a SYM^+ circuit D of size $S = 2^{\log^{c(d)} s} + 2^{\log^e s} 2^{\sqrt{n}(\log n \log M)^e}$ in time $2^{\log^{c(d)} s} + 2^{\log^e s} 2^{\sqrt{n}(\log n \log M)^e}$.
2. Define the matrices $A \in \{0, 1\}^{2^{n/2} \times S}$, $B \in \{0, 1\}^{S \times 2^{n/2}}$ as follows. The rows of A and the columns of B are indexed by partial assignments to the first half and second half of the variables respectively. The columns of A and the rows of B are indexed by the AND gates of D . For each AND gate g in the circuit D and partial assignment x to the first $n/2$ coordinates, define $A(x, g)$ to be 1 if x does not falsify g and 0 otherwise, and for each partial assignment y to the second $n/2$ variables, define $B(g, y)$ to be 1 if y does not falsify g and 0 otherwise. The matrix $M = A \cdot B$ is a $2^{n/2} \times 2^{n/2}$ matrix with rows and columns indexed by partial assignments to the first and second halves of the variables such that $M(x, y)$ is the number of input wires to the top symmetric gate of D set to 1 by the assignment x, y .
3. Evaluate the symmetric function on each entry of the matrix M . This can be done in time $O(2^n + S)$, by pre-evaluating the symmetric function H on all $i \in [h]$, where h is the fan-in of the top symmetric gate in D , and then using this as a lookup table to compute $H(M(x, y))$ for each $x, y \in \{0, 1\}^{n/2}$.

Choosing $\varepsilon(d), \delta$

If $s \leq 2^{n^{\varepsilon(d)}}$ and $M \leq 2^{n^\delta}$, then $\log S \leq n^{\varepsilon(d)}$ and $\log M \leq n^\delta$, which implies that $S \leq 2^{n^{\varepsilon(d)c(d)}} + 2^{e\varepsilon(d)} 2^{\sqrt{n}(n^\delta \log n)^e}$. Hence, if we pick $\delta = \frac{1}{10e}$ and $\varepsilon(d) = \min\{\frac{1}{2c(d)}, \frac{1}{10e}\}$, then $S < 2^{0.05n}$, which implies that step 2 can be done in time $O^*(2^n)$ using Coppersmith's algorithm, and step 3 takes time $O(2^n)$. This proves Theorem 19.

References

- 1 Eric Allender and Vivek Gore. On strong separations from AC^0 . In *International Symposium on Fundamentals of Computation Theory*, pages 1–15. Springer, 1991. doi:10.1007/3-540-54458-5_44.
- 2 Eric Allender and Vivek Gore. A uniform circuit lower bound for the permanent. *SIAM J. Comput.*, 23(5):1026–1049, 1994. doi:10.1137/S0097539792233907.

- 3 Josh Alman, Timothy M. Chan, and R. Ryan Williams. Polynomial representations of threshold functions and algorithmic applications. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 467–476. IEEE, IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.57.
- 4 Josh Alman and Kevin Rao. Faster walsh-hadamard and discrete fourier transforms from matrix non-rigidity. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 455–462. ACM, 2023. doi:10.1145/3564246.3585188.
- 5 Josh Alman and R. Ryan Williams. Probabilistic rank and matrix rigidity. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 641–652. ACM, 2017. doi:10.1145/3055399.3055484.
- 6 Josh Alman and Ryan Williams. Probabilistic polynomials and hamming nearest neighbors. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 136–150. IEEE, IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.18.
- 7 Swapnam Bajpai, Vaibhav Krishan, Deepanshu Kush, Nutan Limaye, and Srikanth Srinivasan. A #sat algorithm for small constant-depth circuits with PTF gates. *Algorithmica*, 84(4):1132–1162, 2022. doi:10.1007/s00453-021-00915-7.
- 8 Richard Beigel and Jun Tarui. On ACC. *Comput. Complex.*, 4:350–366, 1994. doi:10.1007/BF01263423.
- 9 Timothy M. Chan. Orthogonal range searching in moderate dimensions: k-d trees and range trees strike back. In Boris Aronov and Matthew J. Katz, editors, *33rd International Symposium on Computational Geometry, SoCG 2017, July 4-7, 2017, Brisbane, Australia*, volume 77 of *LIPIcs*, pages 27:1–27:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.SoCG.2017.27.
- 10 Ashok K. Chandra, Larry J. Stockmeyer, and Uzi Vishkin. Constant depth reducibility. *SIAM J. Comput.*, 13(2):423–439, 1984. doi:10.1137/0213028.
- 11 Lijie Chen and R. Ryan Williams. Stronger connections between circuit analysis and circuit lower bounds, via pcps of proximity. In Amir Shpilka, editor, *34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA*, volume 137 of *LIPIcs*, pages 19:1–19:43. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.CCC.2019.19.
- 12 R. Chen, R. Santhanam, and S. Srinivasan. Average-case lower bounds and satisfiability algorithms for small threshold circuits. *Theory of Computing*, 14, 2018. doi:10.4086/toc.2018.v014a009.
- 13 Chao-Kong Chow. On the characterization of threshold functions. In *2nd Annual Symposium on Switching Circuit Theory and Logical Design (SWCT 1961)*, pages 34–38. IEEE, 1961. doi:10.1109/FOCS.1961.24.
- 14 Don Coppersmith. Rapid multiplication of rectangular matrices. *SIAM J. Comput.*, 11(3):467–471, 1982. doi:10.1137/0211037.
- 15 Johan Håstad. Almost optimal lower bounds for small depth circuits. In Juris Hartmanis, editor, *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 6–20. ACM, 1986. doi:10.1145/12130.12132.
- 16 Johan Håstad. *Computational limitations for small depth circuits*. PhD thesis, Massachusetts Institute of Technology, 1986.
- 17 Russell Impagliazzo, William Matthews, and Ramamohan Paturi. A satisfiability algorithm for ac^0 . In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 961–972. SIAM, SIAM, 2012. doi:10.1137/1.9781611973099.77.

- 18 Valentine Kabanets and Zhenjian Lu. Satisfiability and derandomization for small polynomial threshold circuits. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2018)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.APPROX-RANDOM.2018.46.
- 19 Alexis Maciel and Denis Thérien. Threshold circuits of small majority-depth. *Inf. Comput.*, 146(1):55–83, 1998. doi:10.1006/inco.1998.2732.
- 20 Saburo Muroga. *Threshold logic and its applications*. John Wiley & Sons, 1971.
- 21 Cody Murray and R. Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime: an easy witness lemma for NP and NQP. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 890–901. ACM, 2018. doi:10.1145/3188745.3188910.
- 22 Eduard Ivanovich Nechiporuk. On a boolean function. In *Dokl. Akad. Nauk SSSR*, volume 169(4), pages 765–766. Russian Academy of Sciences, 1966.
- 23 Ryan O’Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014. URL: <http://www.cambridge.org/de/academic/subjects/computer-science/algorithms-complexity-computer-algebra-and-computational-g-analysis-boolean-functions>.
- 24 Ramamohan Paturi, Pavel Pudlák, and Francis Zane. Satisfiability coding lemma. In *38th Annual Symposium on Foundations of Computer Science, FOCS ’97, Miami Beach, Florida, USA, October 19-22, 1997*, pages 566–574. IEEE, IEEE Computer Society, 1997. doi:10.1109/SFCS.1997.646146.
- 25 Ninad Rajgopal, Rahul Santhanam, and Srikanth Srinivasan. Deterministically counting satisfying assignments for constant-depth circuits with parity gates, with implications for lower bounds. In Igor Potapov, Paul G. Spirakis, and James Worrell, editors, *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, volume 117 of *LIPIcs*, pages 78:1–78:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.MFCS.2018.78.
- 26 Alexander A Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mat. Zametki*, 41(4):598–607, 1987.
- 27 Takayuki Sakai, Kazuhisa Seto, Suguru Tamaki, and Junichi Teruyama. Improved exact algorithms for mildly sparse instances of max SAT. *Theor. Comput. Sci.*, 697:58–68, 2017. doi:10.1016/j.tcs.2017.07.011.
- 28 Takayuki Sakai, Kazuhisa Seto, Suguru Tamaki, and Junichi Teruyama. Bounded depth circuits with weighted symmetric gates: Satisfiability, lower bounds and compression. *J. Comput. Syst. Sci.*, 105:87–103, 2019. doi:10.1016/j.jcss.2019.04.004.
- 29 Rahul Santhanam. Fighting perebor: New and improved algorithms for formula and QBF satisfiability. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 183–192. IEEE, IEEE Computer Society, 2010. doi:10.1109/FOCS.2010.25.
- 30 Kazuhisa Seto and Suguru Tamaki. A satisfiability algorithm and average-case hardness for formulas over the full binary basis. *Comput. Complex.*, 22(2):245–274, 2013. doi:10.1007/s00037-013-0067-7.
- 31 Alexander A. Sherstov. The intersection of two halfspaces has high threshold degree. *SIAM J. Comput.*, 42(6):2329–2374, 2013. doi:10.1137/100785260.
- 32 Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 77–82. ACM, 1987. doi:10.1145/28395.28404.
- 33 Heribert Vollmer. *Introduction to circuit complexity: a uniform approach*. Springer Science & Business Media, 1999. doi:10.1007/978-3-662-03927-4.

- 34 R. Ryan Williams. New algorithms and lower bounds for circuits with linear threshold gates. *Theory Comput.*, 14(1):1–25, 2018. doi:10.4086/toc.2018.v014a017.
- 35 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005. doi:10.1016/j.tcs.2005.09.023.
- 36 Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM J. Comput.*, 42(3):1218–1244, 2013. doi:10.1137/10080703X.
- 37 Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 664–673. ACM, 2014. doi:10.1145/2591796.2591811.
- 38 Ryan Williams. Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2:1–2:32, 2014. doi:10.1145/2559903.
- 39 Andrew Chi-Chih Yao. On ACC and threshold circuits. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume II*, pages 619–627. IEEE, IEEE Computer Society, 1990. doi:10.1109/FSCS.1990.89583.