Electronic Colloquium on Computational Complexity, Report No. 8 (2016)

Algorithms from Natural Lower Bounds*

Marco Carmosino† Russell Impagliazzo‡ Valentine Kabanets§
Antonina Kolokolova¶

January 26, 2016

Abstract

Circuit analysis algorithms such as learning, SAT, minimum circuit size, and compression imply circuit lower bounds. We show a generic implication in the opposite direction: natural properties (in the sense of Razborov and Rudich) imply randomized learning and compression algorithms. This is the first such implication outside of the derandomization setting. As an application, we use known natural lower bounds for $\mathsf{AC}^0[p]$ circuits (due to Razborov and Smolensky) to get the first quasi-polynomial time algorithm for learning $\mathsf{AC}^0[p]$ functions, in the PAC model over the uniform distribution, with membership queries.

*This work was partially supported by the Simons Foundation and NSF grants #CNS-1523467 and CCF-121351 (M. Carmosino, R. Impagliazzo) and by NSERC Discovery grants (V. Kabanets, A. Kolokolova). This work was done in part while all authors were visiting Simons Institute for the Theory of Computing.

†Department of Computer Science, University of California San Diego, La Jolla, CA; mcarmosi@eng.ucsd.edu

‡Department of Computer Science, University of California San Diego, La Jolla, CA; russell@cs.ucsd.edu

§School of Computing Science, Simon Fraser University, Burnaby, BC, Canada; kabanets@cs.sfu.ca

¶Department of Computer Science, Memorial University of Newfoundland, St. John's, NL, Canada; kol@mun.ca

ISSN 1433-8092

Contents

1 Introduction

Circuit analysis problems, problems whose input or output is a Boolean circuit, seem to be a crucial link between designing algorithms and proving lower bounds. For example, Williams [Wil13, Wil14b, Wil14a] shows a way to convert non-trivial Circuit-SAT algorithms into circuit lower bounds. In the other direction, there have been many circuit analysis algorithms inspired by circuit lower bound techniques [LMN93, Bra10, San10, ST12, IMP12, IMZ12, BIS12, CKS14, CKK$^+$15, SW15, CK15, CS15, Tal15], but outside the setting of derandomization [NW94, BFNW93, IW97, IKW02, Uma03, KI04], few formal implications giving generic improvements.

Here we make a step towards such generic connections. While we are not able to show that an *arbitrary* way to prove circuit lower bounds yields circuit analysis algorithms, we show that any circuit lower bound proved through the very general *natural proofs paradigm* of Razborov and Rudich [RR97] does yield such algorithms.

Our main general result is

Theorem 1.1 (Learning Algorithms from Natural Lower Bounds: Informal version)**.** *Natural properties imply randomized learning algorithms and (lossy) compression algorithms.*

We are able to apply our general result with known natural lower bounds [Raz87, Smo87, RR97] to give quasi-polynomial time learning algorithms for the hypothesis class $\mathsf{AC}^0[p]$, for any prime p (polynomial-size constant-depth circuits with AND, OR, NOT, and MOD_p gates).

Theorem 1.2 (Learning for $\mathsf{AC}^0[p]$: Simplified version)**.** *For every prime $p \geq 2$, there is a randomized algorithm that, given membership queries to an arbitrary n-variate Boolean function $f \in \mathsf{AC}^0[p]$, runs in quasipolynomial time $n^{\mathsf{poly}\log n}$ and finds a circuit that computes f on all but $1/\mathsf{poly}(n)$ fraction of inputs.*

No such learning algorithms for $\mathsf{AC}^0[p]$ were previously known. For AC^0, such a learning algorithm was given by Linial, Mansour, and Nisan [LMN93][1], based on Håstad's proof of strong circuit lower bounds for AC^0 [Hås86].

1.1 Compression and learning algorithms from natural lower bounds

Informally, a *natural* lower bound for a circuit class Λ contains an efficient algorithm that distinguishes between the truth tables of "easy" functions (of low Λ-circuit complexity) and those of random Boolean functions. This notion was introduced by Razborov and Rudich [RR97] to capture a common feature of most circuit lower bound proofs: such proofs usually come with efficient algorithms that say something nontrivial about the structure of easy functions in the corresponding circuit class. In [RR97], this observation was used to argue that any circuit class with a natural lower bound is too weak to support cryptography: no strong pseudorandom generator can be computed by a small circuit from the class.

We show that natural circuit lower bounds also imply algorithms for compression and learning of Boolean functions from the same circuit class (provided the circuit class is not too weak).

Recall the compression task for Boolean functions: given the truth table of a Boolean function f, print a circuit that computes f. If f is unrestricted, the best guarantee for the circuit size is

[1] Their algorithm works in a more general learning model without membership queries, but with access to labelled examples $(x, f(x))$ for uniformly random x.

$2^n/n$ [Lup58, Lup59], and such a circuit can be found in time $\mathsf{poly}(2^n)$, polynomial in the truth table size. We might however be able to do much better for restricted classes of functions. Let Λ be the set of functions computed by some circuit class Λ. Recent work has shown that we can "mine" specific lower bounds against Λ to compress functions $g \in \Lambda$ better than the universal construction [CKK$^+$15]. This work suggests that there should be some generic connection between circuit lower bounds and compression algorithms, but such a connection was not known.

We show that any circuit lower bound that is natural in the sense of Razborov and Rudich [RR97] yields a generic compression algorithm for Boolean functions from the same circuit class, provided the circuit class is sufficiently powerful (e.g., containing $\mathsf{AC}^0[p]$ for some prime $p \geq 2$).

A compression algorithm may be viewed as a special case of a natural property: if the compression fails, the function must have high complexity, and compression must fail for most functions. Thus we get an equivalence between these two notions for the case of randomized compression algorithms and BPP-computable natural properties. (As our compression algorithms are randomized, we don't get such an equivalence for the deterministic case.)

The first stage of our algorithm is a lossy compression of the function in the sense that we get a small circuit that computes the function on *most* inputs. Because this first stage only examines the truth table of the function in relatively few locations, we can view this stage as a *learning algorithm*. This algorithm produces a circuit that approximately computes the given function f with respect to the uniform distribution, and uses membership queries to f. So it fits the framework of PAC learning for the uniform distribution, with membership queries.

Our main result also yields a certain "search-to-decision" reduction for the Minimum Circuit Size Problem (MCSP). Recall that in MCSP, one is given the truth table of a Boolean function f, and a parameter s, and needs to decide if the minimum circuit size of f is less than s. Since an efficient algorithm for MCSP would make it a natural property (with excellent parameters), our main result implies the following. If MCSP is in BPP, then, given oracle access to any n-variate Boolean function f of circuit complexity s, one can find (in randomized polynomial time) a circuit of size $\mathsf{poly}(s)$ that computes f on all but $1/\mathsf{poly}(n)$ fraction of inputs.

1.2 Our techniques

The main idea of our lossy compression (learning) algorithm is, given the truth table of a Boolean function f to be compressed,

- use f as the basis for the Nisan-Wigderson (NW) generator [NW94],
- break the generator by applying the natural property algorithm as a distinguisher (between the strings output by the NW generator and truly random strings),
- use an efficient reconstruction algorithm from the analysis of the NW generator to find a small circuit that approximately computes the function f.

For the described approach to work, we need to ensure that (1) each output of the NW generator (when viewed as the truth table of a Boolean function) is computable by a small circuit from the circuit class for which we have a natural lower bound (and so the natural property algorithm can be used as the distinguisher to break the generator), and (2) there is an efficient reconstruction algorithm that takes a circuit breaking the NW generator based on the function f, and constructs a small circuit for (approximately computing) f.

For (1), we utilise the fact that each bit of output of the NW generator (on a fixed seed) is basically the value of the function f applied to some substring of the seed (chosen via a certain combinatorial structure, the NW design), and so the circuit complexity of the truth table output by the NW generator is closely related to the circuit complexity of the original function f.

In particular, we show that if f is in $\mathsf{AC}^0[p]$, and the NW generator has exponential stretch (from $\mathsf{poly}(n)$ bits to 2^{n^γ} bits, for some $\gamma > 0$), then each string output by the NW generator is also a function in $\mathsf{AC}^0[p]$. If, on the other hand, we take the NW generator with certain polynomial stretch, we get that its output strings will be Boolean functions computable by $\mathsf{AC}^0[p]$ circuits of subexponential size. The tradeoff between the chosen stretch of the NW generator and the circuit complexity of the string it outputs will be very important for the efficiency of our learning algorithms.

For (2), we use the known efficient randomized algorithm that takes a distinguisher for the NW generator based on a function f, and constructs a small circuit approximately computing f, provided the algorithm is given oracle access to f. The existence of such a uniform algorithm was first observed by Impagliazzo and Wigderson [IW01] (based on [NW94, BFNW93]) in the context of derandomizing BPP under uniform complexity assumptions. Simulating oracle access to f in the framework of [IW01] was quite nontrivial (and required the downward self-reducibility of f). In contrast, we are explicitly given the truth table of f, and so oracle access to f is not an issue!

Note that if we break the NW generator based on a function f, we only get a circuit that agrees with f on slightly more than half of all inputs. To get a better approximation of f, we employ a standard "hardness amplification" encoding of f, getting a new, amplified function h, and then use h as the basis for the NW generator. The function h has the property that any circuit computing h on better than $1/2$ of the inputs can be efficiently massaged into a new circuit that computes the original f on most inputs.

For this amplification to work in our context, we need to ensure that the amplified function h is in the same circuit class as f, and is of related circuit complexity. We show that standard tools such as the Direct Product and XOR constructions have the required properties for $\mathsf{AC}^0[2]$. For $\mathsf{AC}^0[p]$ where p is prime other than 2, we can't use the XOR construction (as PARITY cannot be computed in $\mathsf{AC}^0[p]$ for any prime $p > 2$ by Smolensky's lower bound [Smo87]). We argue that the MOD_p function can be used for the required amplification within $\mathsf{AC}^0[p]$[2].

Thus, our actual lossy compression algorithm will be:

> Given the truth table of a function f in some circuit class Λ, construct a small circuit approximating f by running a uniform reconstruction algorithm that uses a natural property for the class Λ as the distinguisher to break the NW generator based on the amplified version of f.

To turn this algorithm into the exact compression algorithm, we just patch up the errors by table lookup. Since there are relatively few errors, the size of the patched-up circuit will still be less that the trivial size $2^n/n$.

[2] We stress that for our purposes it is important that the *forward direction* of the conditional PRG construction, from a given function f to a generator based on that f, be computable in some low nonuniform circuit class (such as $\mathsf{AC}^0[p]$). In contrast, in the setting of conditional derandomization, it is usually important that the *reverse direction*, from a distinguisher to a small circuit (approximately) computing the original function f, be computable in some low (nonuniform) circuit class (thereby contradicting the assumed hardness of f for that circuit class). One notable exception is hardness amplification within NP [O'D04, HVV06, Tre05].

More interestingly, we can get from our lossy compression algorithm described above a *learning* algorithm! The idea is that the reconstruction algorithm for the NW generator runs in time polynomial in the size of the output of the generator, and so only needs at most that many oracle queries to the function f. Rather than being given the full truth table of f, such an algorithm can be simulated with just membership queries to f. Thus we get a learning algorithm with membership queries in the PAC model over the uniform distribution.

Since the runtime of this learning algorithm (and hence also the size of the circuit for f it produces) will be polynomial in the output length of the NW generator that we use to learn f, we would like to minimize the stretch of the NW generator[3]. However, as noted above, *shorter stretch* of the generator means *higher circuit complexity* of the truth table it outputs. This in turn means that we need a natural property that works for Boolean functions of higher circuit complexity. In the extreme case, to learn a polysize Boolean function f, we need to use the NW generator with polynomial stretch, and hence need a natural property useful against circuits of exponential size. In general, there will be a tradeoff between the efficiency of our learning algorithm for the circuit class Λ and the quality of a natural circuit lower bound for Λ: the better the circuit lower bound, the more efficient the learning algorithm.

Razborov and Rudich [RR97] showed the $\mathsf{AC}^0[p]$ circuit lower bounds due to Razborov [Raz87] and Smolensky [Smo87] can be made into natural properties that are useful against circuits of weakly exponential size 2^{n^γ}, for some $\gamma > 0$ (dependent on the depth of the circuit). Plugging this natural property into our framework, we get our quasipolynomial-time learning algorithm for $\mathsf{AC}^0[p]$, for any prime p.

We remark that our approach is quite similar to the way Razborov and Rudich [RR97] used natural properties to get new algorithms. They used natural properties to break the cryptographic pseudorandom function generator of [GGM86], which by definition outputs functions of low circuit complexity. Breaking such a generator based on an assumed one-way function F leads to an efficient algorithm for inverting this function F well on average (contradicting the one-wayness of F). We, on the other hand, use the NW generator based on a given function f. The properties of the NW generator construction can be used to show that it outputs (the truth tables of) functions of low circuit complexity, relative to the circuit complexity of f. Thus a natural property for the appropriate circuit complexity class (with an appropriate size parameter) can be used to break this NW generator, yielding an efficient algorithm for producing a small circuit approximating f.

1.3 Related work

This work was prompted by results that circuit analysis algorithms imply circuit lower bounds. A natural question is: given that these algorithms are *sufficient* for circuit lower bounds, to what degree are they *necessary*? Apart from derandomization, no other equivalences between circuit analysis algorithms and circuit lower bounds are known. Below we list a number of known circuit-analytic algorithmic tasks that would imply some kind of circuit lower bounds:

- Derandomization [IKW02, KI04, AvM12, CIKK15]
- Deterministic (lossy) compression or MCSP [CKK$^+$15, IKW02]
- Deterministic learning [FK09, KKO13]

[3]This is in sharp contrast to the setting of derandmomization where one wants to *maximize* the stretch of the generator, as it leads to a more efficient derandomization algorithm.

- Deterministic (QBF) SAT algorithms [Wil13, SW15]

Bracketing the hardness vs. randomness setting, special cases of using circuit lower bounds to construct circuit analysis algorithms abound. Often, lower bounds are the *only* way that we know to construct these algorithms. Each of the following results uses the proof of a lower bound to construct an algorithm. The character and number of these results gives empirical evidence that there should be generic algorithms for circuit analysis based on generic lower bounds.

- Parity $\notin \mathsf{AC}^0 \rightsquigarrow \mathsf{AC}^0$-Learning [LMN93], AC^0-SAT [IMP12], and AC^0-Compression [CKK$^+$15]
- $\mathrm{MOD}_q \notin \mathsf{AC}^0[p]$, p, q distinct primes, $\rightsquigarrow \mathsf{AC}^0[p]$-Compression [Sri15]
- Andreev's function $\notin$ deMorgan$[n^{3-\epsilon}] \rightsquigarrow$ subcubic formula Compression [CKK$^+$15]

All the lower bounds listed above belong to the natural proofs framework. Given these results, the obvious conjecture was that natural proofs imply some kind of generic circuit analysis algorithm. For instance, [CKK$^+$15] suggested that every natural circuit lower bound should imply a compression algorithm. We take a step towards proving such an implication by showing that any natural circuit lower bound for a sufficiently powerful circuit class ($\mathsf{AC}^0[p]$ or bigger) does indeed lead to a randomized compression algorithm for the same circuit class.

The remainder of the paper. We give the necessary background in Section 2. We define and show the constructions of a black-box generator and black-box amplification, the tools we need to prove our main results, in Sections 3 and 4, respectively. In Section 5, we use these tools to prove our main result that natural properties yield learning algorithms for circuit classes $\mathsf{AC}^0[p]$ and above. On the other hand, in Section 6, we argue that AC^0 is not sufficiently powerful to carry out the proof of our main result. Section 7 contains concluding remarks and open questions. Some auxiliary results and additional technical details are given in the appendix.

2 Definitions and tools

2.1 Circuits and circuit construction tasks

For a circuit class Λ and a set of size functions $\mathcal{S}$, we denote by $\Lambda[\mathcal{S}]$ the set of $\mathcal{S}$-size n-input circuits of type Λ. When no $\mathcal{S}$ is explicitly given, it is assumed to be $\mathsf{poly}(n)$.

Definition 2.1 (Circuits (Approximately) Computing f)**.** Let $f : \{0,1\}^n \to \{0,1\}$ be some Boolean function, and let $\epsilon : \mathbb{N} \to [0,1]$ be an approximation bound. Then $\mathsf{CKT}_n(f)$ denotes the set of circuits that compute the function f on all n-bit inputs, and $\widetilde{\mathsf{CKT}}_n(f,\epsilon)$ the set of all circuits that compute f on all but an ϵ fraction of inputs.

Definition 2.2 (Circuit Builder Declarations (adapted from [IW01]))**.** Let A and B be indexed sets of circuits. A $T(n)$-*construction of* B *from* A is a probabilistic machine $\mathcal{M}(n, \alpha, A_n)$ which outputs a member of B_n with probability at least $1 - \alpha$ in time $T(n)$, where the size of B_n is $\mathsf{poly}(|A_n|)$. We declare that such a machine exists by writing:

$$\textsc{Cons}(A \to B;\ T(n)).$$

Read this notation as "from A we can construct B in time $T(n)$." To assert the existence of a $T(n)$-*construction of* B *from* A*, with oracle* $\mathcal{O}$, where the machine $\mathcal{M}$ is equipped with an oracle for the language $\mathcal{O}$ but otherwise is as above, write: $\textsc{Cons}^{\mathcal{O}}(A \to B;\ T(n))$.

2.2 Learning tasks

Let $f \in \Lambda$ be some Boolean function. The learner is allowed membership queries to f. That is, the learner may query an input $x \in \{0,1\}^n$ to the oracle, getting back the value $f(x)$.

Definition 2.3 (PAC learning over the uniform distribution with membership queries)**.** Let Λ be any class of Boolean functions. An algorithm A PAC-learns Λ if for any n-variate $f \in \Lambda$ and for any $\epsilon, \delta > 0$, given membership query access to f algorithm A prints with probability at least $1 - \delta$ over its internal randomness a circuit $C \in \widetilde{\mathsf{CKT}}_n(f, \epsilon)$. The runtime of A is measured as a function $T = T(n, 1/\epsilon, 1/\delta, \text{size}(f))$.

Note that we do not require "proper learning": the output is an unrestricted circuit.

2.3 Compression tasks

Definition 2.4 (Λ-Compression)**.** Given the truth table of n-variate Boolean function $f \in \Lambda$, print some Boolean circuit $C \in \mathsf{CKT}_n(f)$ computing f such that $|C| < 2^n/n$, the trivial bound.

Definition 2.5 (ϵ-Lossy Λ-Compression)**.** Given the truth table of n-variate Boolean function $f \in \Lambda$, print some Boolean circuit $C \in \widetilde{\mathsf{CKT}}_n(f, \epsilon)$ such that $|C| < 2^n/n$, the trivial bound.

The relevant parameters for compression are runtime and printed circuit size. We say that a compression algorithm is efficient if it runs in time $\mathsf{poly}(2^n)$, which is polynomial in the size of the truth-table supplied to the algorithm. Though we count any output circuit of size less than $2^n/n$ as a successful compression, we will of course want to optimize this. In previous work, the size of the resulting circuits approximately matches the size of circuits for which we have lower bounds. Note that we do not require "proper compression" in either case: the output is an unrestricted circuit.

2.4 Natural properties

Let F_n be the collection of all Boolean functions on n variables. Λ and Γ denote complexity classes. A combinatorial property is a sequence of subsets of F_n for each n.

Definition 2.6 (Natural Property [RR97])**.** A combinatorial property R_n is Γ-natural against Λ with density δ_n if it satisfies the following three conditions:

Constructivity: The predicate $f_n \stackrel{?}{\in} R_n$ is computable in Γ

Largeness: $|R_n| \geq \delta_n \cdot |F_n|$

Usefulness: For any sequence of functions f_n, if $f_n \in \Lambda$ then $f_n \notin R_n$, almost everywhere.

For each n, δ_n is a lower bound on the probability that $g \in F_n$ has R_n. The original definition in [RR97] sets $\delta_n \geq 2^{-O(n)}$. However, we show (see Lemma 2.7 below) that one may usually assume that $\delta_n \geq 1/2$. Note that in the wild, nearly all natural properties have δ_n close to one and $\Gamma \subseteq \mathsf{NC}^2$.

Lemma 2.7 (Largeness for natural properties)**.** *Suppose P is a P-natural property of n-variate Boolean functions that is useful against class Λ of size $s(n)$, and has largeness $\delta_n \geq 2^{-cn}$, for some constant $c \geq 0$. Then there is another P-natural property P' that is useful against the class Λ of size $s'(n) := s(n/(c+1))$, and has largeness $\delta'_n \geq 1/2$.*

Proof. Define P' as follows:

> The truth table of a given $f : \{0,1\}^n \to \{0,1\}$ is in P' iff for at least one string $a \in \{0,1\}^k$, for $k = cn/(c+1)$, the restriction $f_a(y_1, \ldots, y_{n-k}) := f(a_1, \ldots, a_k, y_1, \ldots, y_{n-k})$ is in P (as a function on $n - k = n/(c+1)$ variables).

Observe that testing P' on a given n-variate Boolean function f can be done in time $O(2^k) \cdot \mathsf{poly}(2^{n-k}) \leq \mathsf{poly}(2^n)$; so we have constructivity for P'. Next, if $f : \{0,1\}^n \to \{0,1\}$ has a Λ circuit of size less $s'(n)$, then each restricted subfunction $f_a : \{0,1\}^{n-k} \to \{0,1\}$ has a Λ circuit of size less than $s(n-k) \leq s'(n)$. Finally, a random function $f : \{0,1\}^n \to \{0,1\}$ yields 2^k independent random subfunctions, on $n-k$ variables each, and the probability that at least one of these $(n-k)$-variate functions satisfies P is at least

$$\begin{aligned} 1 - (1 - 2^{-c(n-k)})^{2^k} &= 1 - (1 - 2^{-k})^{2^k} \\ &\geq 1 - \frac{1}{e} \\ &\geq \frac{1}{2}. \end{aligned}$$

Thus we have largeness at least $1/2$ for P'. □

2.5 NW generator

Definition 2.8 (NW Design)**.** For parameters $n, m, L \in \mathbb{N}$, a sequence of sets $S_1, \ldots, S_L \subseteq [m]$ is called an *NW design* if

- $|S_i| = n$, for all $1 \leq i \leq L$, and
- $|S_i \cap S_j| \leq \log L$, for all $1 \leq i \neq j \leq L$.

It is well-known that NW designs exist and can be efficiently constructed for any n, $m = O(n^2)$, and $L < 2^n$ [NW94]. In Section 2.5.1 below, we review the construction of NW designs from [NW94], and show that it can be implemented in $\mathsf{AC}^0[p]$ (Theorem 2.13). The efficiency of this construction of designs is necessary for our transfer theorem.

Definition 2.9 (NW Generator)**.** Let $f\colon \{0,1\}^n \to \{0,1\}$. For $m = n^2$ and a stretch function $L(m)\colon \mathbb{N} \to \mathbb{N}$, where $L(m) < 2^n$, let $S_1, \ldots, S_L \subseteq [m]$ be an NW design. Define the NW generator $G_f\colon \{0,1\}^m \to \{0,1\}^{L(m)}$ as:

$$G_f(z) = f(z|_{S_1}) f(z|_{S_2}) \ldots f(z|_{S_{L(m)}}), \tag{1}$$

where $z|_S$ denotes the $|S|$-length bit-string obtained by restricting z to the bit positions indexed by the set S.

Recall the notion of a distinguisher, a circuit that breaks a given generator.

Definition 2.10 (Distinguishers)**.** Let $L\colon \mathbb{N} \to \mathbb{N}$ be a stretch function, let $\mathcal{G} = \{g_m\colon \{0,1\}^m \to \{0,1\}^{L(m)}\}$ be a sequence of functions, and let $0 < \epsilon < 1$ be an error bound. Define the set $\mathsf{DIS}(\mathcal{G}, \epsilon)$ as all circuits D with $L(m)$ inputs satisfying:

$$\Pr_{z \in \{0,1\}^m}[D(g_m(z))] - \Pr_{y \in \{0,1\}^{L(m)}}[D(y)] > \epsilon.$$

Theorem 2.11 (NW Reconstruction [NW94, IW01])**.** *We have*

$$\textsc{Cons}^f(\mathsf{DIS}(G_f, 1/5) \to \widetilde{\mathsf{CKT}}(f, 1/2 - 1/L(m));\ \mathsf{poly}(L(m))).$$

2.5.1 NW designs in $\mathsf{AC}^0[p]$

Here we show that the particular NW designs we need in our compression and learning algorithms can be constructed by small $\mathsf{AC}^0[p]$ circuits, for any fixed prime p. Consider an NW design $S_1, \ldots, S_L \subseteq [m]$, for $m = O(n^2)$, where

- each set S_i is of size n,
- the number of sets is $L = 2^\ell$ for $\ell \leq n$, and
- for any two distinct sets S_i and S_j, $i \neq j$, we have $|S_i \cap S_j| \leq \ell$.

We show a particular construction of such a design that has the following property: the index set $[m]$ is partitioned into n disjoint subsets $U_1, \ldots, U_n$ of equal size $(m/n) \in O(n)$. For each $1 \leq i \leq L$, the set S_i contains exactly one element from each subset U_j, over all $1 \leq j \leq n$. For $1 \leq j \leq n$ and $1 \leq k \leq O(n)$, we denote by $(U_j)_k$ the kth element in the subset U_j.

To describe such a design, we use the following characteristic function g: for $1 \leq i \leq L$, $1 \leq j \leq n$, and $1 \leq k \leq O(n)$,

$$g(i,j,k) = \begin{cases} 1 & \text{if } (U_j)_k \in S_i \\ 0 & \text{otherwise.} \end{cases}$$

We will prove the following.

Theorem 2.12. *There exists a constant $d_{NW} \geq 1$ such that, for any prime p, there exists a family of functions $g : \{0,1\}^{\ell+2\log n} \to \{0,1\}$ that are the characteristic functions for some NW design with the parameters as above, so that $g \in \mathsf{AC}^0[p]$ of size $O(n^2 \log n)$ and depth d_{NW}.*

Proof. Recall the standard construction of NW designs from [NW94]. Let F be a field of size $O(n)$. Consider an enumeration of L polynomials of degree at most ℓ over F, with all coefficients in $\{0,1\}$; there are at least $2^\ell = L$ such polynomials. We associate each such polynomial with a binary string $i = i_1 \ldots i_\ell \in \{0,1\}^\ell$, so that i corresponds to the polynomial

$$A_i(x) = \sum_{j=1}^{\ell} i_j \cdot x^{j-1}$$

over the field F. Let $r_1, \ldots, r_{|F|}$ be some canonical enumeration of the elements of F. For each binary string $i \in \{0,1\}^\ell$, we define a set

$$S_i = \{(r_j, A_i(r_j)) \mid 1 \leq j \leq n\}.$$

Note that $|S_i| = n$, and S_i defines a set of n pairs in the universe $F \times F$ of $O(n^2)$ pairs (hence the universe size for this construction is $O(n^2)$). Finally, any two distinct degree $(\ell - 1)$ polynomials $A_i(x)$ and $A_j(x)$ may agree on at most ℓ points $r \in F$, and so we have $|S_i \cap S_j| \leq \ell$ for the sets S_i and S_j, corresponding to the polynomials $A_i(x)$ and $A_j(x)$.

Arrange the elements of the universe $[m]$ on an $n \times (m/n)$ grid. The n rows of the grid are indexed by the first n field elements $r_1, \ldots, r_n$, and the columns by all fields elements $r_1, \ldots, r_{|}F|$. For each j, $1 \leq j \leq n$, define U_j to be the elements of $[m]$ that belong to the row j of the grid. We get that every set $S_i = \{(r_j, A_i(r_j)) \mid 1 \leq j \leq n\}$ picks exactly one element from each of the n sets $U_1, \ldots, U_n$.

We will argue that this particular design construction is computable in $\mathsf{AC}^0[p]$ of size polynomial in ℓ, for each prime p. Let p be any fixed prime (which we think of as a constant). Let F be an extension field over $\mathbf{GF}(p)$ of the least size so that $|F| \geq n$; such a field is described by some polynomial over $\mathbf{GF}(p)$ of degree $O(\log_p n)$, and is of size at most $pn = O(n)$. As before, let $r_1, \ldots, r_{|F|}$ be some canonical enumeration of the field elements in F.

Define the following $n \times \ell$ matrix M: for $1 \leq j \leq n$ and $1 \leq k \leq \ell$, we have

$$M_{j,k} = (r_j)^k,$$

where the power $(r_j)^k$ is computed within the field F. Then the values $A_i(r_1), \ldots, A_i(r_n)$ may be read off from the column vector obtained by multiplying the matrix M by the column vector $i \in \{0,1\}^\ell$, in the field F. For a particular $1 \leq j \leq n$, we have

$$A_i(r_j) = \sum_{k=1}^{\ell} M_{j,k} \cdot i_k.$$

Since each $i_k \in \{0,1\}$, the latter reduces to the task of adding a subset of ℓ field elements. Each field element of F is a polynomial over $\mathbf{GF}(p)$ of degree $k \leq O(\log n)$, and so adding a collection of elements from F reduces to the coordinate-wise summation modulo p of k-element vectors in $(\mathbf{GF}(p))^k$. The latter task is easy to do in $\mathsf{AC}^0[p]$[4].

For any $1 \leq i \leq L$, $1 \leq j \leq n$, and $1 \leq k \leq |F|$, $g(i,j,k) = 1$ iff $A_i(r_j) = r_k$. To compute $g(i,j,k)$, we need to evaluate the polynomial $A_i(x)$ at r_j, and then check if the result is equal to r_k. To this end, we "hard-code" the matrix M into the circuit (which incurs the cost at most $O(n\ell \log n)$ bits of advice). We compute $A_i(r_j)$ by computing the matrix-vector product $M \cdot i$, and restricting to the jth coordinate of the resulting column vector. This computation involves $O(\log n)$ summations of ℓ field elements of $\mathbf{GF}(p)$ modulo p, over n rows of the matrix M. The resulting field element is described an $O(\log n)$-element vector of elements from the underlying field $\mathbf{GF}(p)$. Using $O(\log n)$ operations over $\mathbf{GF}(p)$, we can check if this vector equals the vector corresponding to r_k.

It is easy to see that this computation can be done in some fixed constant depth d_{NW} by an $\mathsf{AC}^0[p]$ circuit of size $O(\ell \cdot n \log n)$, which can be bounded by $O(n^2 \log n)$, as required. □

As a corollary, we get

Theorem 2.13. *Let p be any prime. There exists a constant $d_{MX} \geq 1$ such that, for any n and $L < 2^n$, there exists an NW design $S_1, \ldots, S_L \subseteq [m]$ with $m = O(n^2)$, each $|S_i| = n$, and $|S_i \cap S_j| \leq \ell = \log L$ for all $1 \leq i \neq j \leq L$, such that the function $MX_{NW} : \{0,1\}^\ell \times \{0,1\}^m \to \{0,1\}^n$, defined by*

$$MX_{NW}(i,z) = z|_{S_i},$$

is computable by an $\mathsf{AC}^0[p]$ circuit of size $O(\ell \cdot n^3 \log n)$ and depth d_{MX}.

[4]We code elements of $\mathbf{GF}(p)$ by p-wire bundles, where wire i is on iff the bundle codes the ith element of $\mathbf{GF}(p)$. An addition, multiplication, or inverse in the field $\mathbf{GF}(p)$ can be implemented in AC^0. To add up a tuple of field elements, we first convert each field element from the representation above to the unary representation (using constant-depth selection circuits). Then we lead these unary encodings into a layer of p gates, $\oplus_p^j$, for $0 \leq j \leq p-1$, where $\oplus_p^j$ is the gate $\oplus_p$ with $p-j$ extra inputs 1. Thus the gate $\oplus_p^j$ on inputs $x_1, \ldots, x_n \in \mathbf{GF}(p)$ outputs 1 iff $x_1 + \cdots + x_n = j \mod p$. Note that exactly one of the gates $\oplus_p^j$ will output 1, giving us the desired field element in our encoding.

Proof. Let $g(i,j,k)$ be the characteristic function for the NW design from Theorem 2.12, where $|i| = \ell$, $|j| = \log n$, and $|k| = \log n + \log c$, for some constant $c \geq 1$. We have $g \in \mathsf{AC}^0[p]$ of size $O(\ell \cdot n \log n)$ and depth d_{NW}. Let $U_1, \ldots, U_n \subseteq [m]$ be the sets of size cn each that partition $[m]$ so that every S_i contains exactly one element from every U_j, $1 \leq j \leq n$.

Let $i_1, \ldots, i_\ell$ and $z_1, \ldots, z_m$ denote the input gates of MX_{NW}, and let $y_1, \ldots, y_n$ denote its output gates. Associate each gate y_j with the set U_j of indices in $[m]$, for $1 \leq j \leq n$. For each $1 \leq i \leq L$ and each $1 \leq j \leq n$, define

$$y_j = \vee_{k=1}^{cn} \; g(i,j,k) \wedge (z|_{U_j})_k.$$

Clearly, the defined circuit computes MX_{NW}. It has size $O(\ell \cdot n^3 \log n)$ and depth $d_{MX} \leq d_{NW} + 2$, as required. □

Let G_f be the NW generator based on a function f, using the NW design $S_1, \ldots, S_L$ from Theorem 2.13. For each fixed seed z, define the function $g_z : \{0,1\}^\ell \to \{0,1\}$, for $\ell = \log L$, as

$$\begin{aligned} g_z(i) &= (G_f(z))_i \\ &= f(z|_{S_i}), \end{aligned}$$

where $1 \leq i \leq L$. By Theorem 2.13, we get $g_z \in (\mathsf{AC}^0[p])^f$. See Figure 1 for the description of a small circuit for g_z that combines the $\mathsf{AC}^0[p]$ circuit for MX_{NW} with a circuit for f.

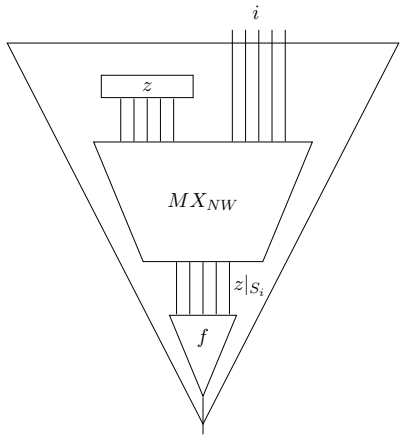

Figure 1: A circuit for $g_z(i) = f(z|_{S_i})$.

3 Black-box generators

The main tool we need for our learning algorithms is a transformation, which we call *black-box generator*, taking a given function $f\colon \{0,1\}^n \to \{0,1\}$ to a family $G = \{g_z\}_{z \in I}$ of new Boolean functions $g_z\colon \{0,1\}^{n'} \to \{0,1\}$ satisfying the following properties:

- [NONUNIFORM EFFICIENCY] each function g_z has "small" circuit complexity relative to the circuit complexity of f, and

- [RECONSTRUCTION] any circuit distinguishing a random function g_z (for a uniformly random $z \in I$) from a random n'-variate Boolean function can be used (by an efficient randomized algorithm with oracle access to f) to construct a good approximating circuit for f.

Once we have such a black-box generator, we get our learning algorithm as follows: To learn a function $f : \{0,1\}^n \to \{0,1\}$, use the natural property as a distinguisher that rejects (the truth tables of) all functions g_z, $z \in I$, but accepts a constant fraction of truly random functions; apply the efficient reconstruction procedure to learn a circuit approximating f. Intuitively, we use [NONUNIFORM EFFICIENCY] to argue that if f is an easy function in some circuit class Λ, then so is each function g_z, $z \in I$.

Next we give a more formal definition of a black-box generator.

Definition 3.1 (Black-Box (ϵ, L)-Generator Within Λ)**.** For a given error parameter $\epsilon\colon \mathbb{N} \to [0,1]$ and a stretch function $L\colon \mathbb{N} \to \mathbb{N}$, a *black-box* (ϵ, L)*-generator within* Λ is a mapping GEN that associates with a given function $f\colon \{0,1\}^n \to \{0,1\}$ a family $\textsc{Gen}(f) = \{g_z\}_{z \in \{0,1\}^m}$ of Boolean functions $g_z\colon \{0,1\}^\ell \to \{0,1\}$, where $\ell = \log L(n)$, satisfying the following conditions for every $f : \{0,1\}^n \to \{0,1\}$:

Small Family Size: $m \leq \mathsf{poly}(n, 1/\epsilon)$,

Nonuniform Λ-Efficiency: for all $z \in \{0,1\}^m$, $g_z \in \Lambda^f[\mathsf{poly}(m)]$, and

Reconstruction: $\textsc{Cons}^f(\mathsf{DIS}(\textsc{Gen}(f), 1/5) \to \widetilde{\mathsf{CKT}}(f, \epsilon); \mathsf{poly}(n, 1/\epsilon, L(n)))$, where we think of $\textsc{Gen}(f)$ as the distribution over the truth tables of functions $g_z \in \textsc{Gen}(f)$, for uniformly random $z \in \{0,1\}^m$.

We will prove the following.

Theorem 3.2. *Let p be any prime. For every $\epsilon\colon \mathbb{N} \to [0,1]$ and $L\colon \mathbb{N} \to \mathbb{N}$ such that $L(n) \leq 2^n$, there exists a black-box (ϵ, L)-generator within $\mathsf{AC}^0[p]$.*

For the proof, we shall need the following notion of black-box amplification. Let Λ be any circuit class.

Definition 3.3 (Black-Box (ϵ, δ)-Amplification within Λ)**.** For given $\epsilon, \delta > 0$, (ϵ, δ)*-amplification within* Λ is a mapping that associates with a given function $f\colon \{0,1\}^n \to \{0,1\}$ its *amplified* version, $\textsc{Amp}(f)\colon \{0,1\}^{n'} \to \{0,1\}$, satisfying the following conditions for every $f\colon \{0,1\}^n \to \{0,1\}$:

Short Input: $n' \leq \mathsf{poly}(n, 1/\epsilon, \log 1/\delta)$,

Nonuniform Λ-Efficiency: $\textsc{Amp}(f) \in \Lambda^f[\mathsf{poly}(n')]$,

Uniform P-Efficiency: $\textsc{Amp}(f) \in \mathsf{P}^f$, and

Reconstruction: $\textsc{Cons}^f(\widetilde{\mathsf{CKT}}(\textsc{Amp}(f), 1/2 - \delta) \to \widetilde{\mathsf{CKT}}(f, \epsilon);\ \mathsf{poly}(n, 1/\epsilon, 1/\delta))$.

We prove the following in the next section (see Theorem 4.11).

Lemma 3.4. *Let p be any fixed prime. For all $0 < \epsilon, \delta < 1$, there is black-box (ϵ, δ)-amplification within $\mathsf{AC}^0[p]$.*

Now we are ready to prove Theorem 3.2.

Proof of Theorem 3.2. Let f be an n-variate Boolean function. Set $f^\star = (\epsilon(n), 1/L(n))$-$\textsc{Amp}(f)$, for the black-box amplification within $\mathsf{AC}^0[p]$ that exists by Lemma 3.4. We have that $f^\star$ is a function on $n' = \mathsf{poly}(n, 1/\epsilon, \log L) = \mathsf{poly}(n, 1/\epsilon)$ variables (using the assumption that $L(n) \leq 2^n$).

Let $G_{f^\star}\colon \{0,1\}^m \to \{0,1\}^{L(n)}$ be the NW generator based on the function $f^\star$, with the seed size $m = (n')^2$. Define

$$\textsc{Gen}(f) = \{g_z\}_{z \in \{0,1\}^m},$$

where $g_z = G_{f^\star}(z)$. We claim that this $\textsc{Gen}(f)$ is an (ϵ, L)-black-box generator within $\mathsf{AC}^0[p]$. We verify each necessary property:

Small Family Size: $m = (n')^2 \leq \mathsf{poly}(n, 1/\epsilon)$.

Nonuniform $\mathsf{AC}^0[p]$-Efficiency: We know that $f^\star = \textsc{Amp}(f) \in (\mathsf{AC}^0[p])^f[\mathsf{poly}(m)]$. For each fixed $z \in \{0,1\}^m$, we have $g_z(i) = (G_{f^\star}(z))_i$, for $i \in \{0,1\}^\ell$, where $\ell = \log L(n)$. By the definition of the NW generator, $g_z(i) = f^\star(z|_{S_i})$. By Theorem 2.13, the restriction $z|_{S_i}$, as a function of z and i, is computable in $\mathsf{AC}^0[p]$ of size $\mathsf{poly}(n')$ and some fixed depth d_{MX}. It follows that each g_z is computable in $(\mathsf{AC}^0[p])^f[\mathsf{poly}(m)]$.

Reconstruction: The input to reconstruction is $D \in \mathsf{DIS}(G_{\textsc{Amp}(f)}, 1/5)$. Let $\mathcal{M}_{NW}$ be the reconstruction machine from the NW construction, and let $\mathcal{M}_{\textsc{Amp}}$ be the reconstruction machine from $(\epsilon, 1/L)$-amplification. We first run $\mathcal{M}_{NW}^{\textsc{Amp}(f)}(D)$ to get, in time $\mathsf{poly}(L)$, a circuit $C \in \widetilde{\mathsf{CKT}}(\textsc{Amp}(f), 1/2 - 1/L(n))$; note that we can provide this reconstruction algorithm oracle access to $\textsc{Amp}(f)$, since $\textsc{Amp}(f) \in \mathsf{P}^f$ by the uniform P-efficiency property of black-box amplification. Next we run $\mathcal{M}_{\textsc{Amp}}^f$ on C to get $C' \in \widetilde{\mathsf{CKT}}(f, \epsilon)$, in time $\mathsf{poly}(n, 1/\epsilon, L(n))$. □

4 Black-box amplification

We will show that black-box amplification is possible within $\Lambda = \mathsf{AC}^0[p]$, to prove the results of the previous section. This section can be skipped at a first reading.

Let Λ be any circuit class (e.g., $\mathsf{AC}^0[p]$ for some prime $p \geq 2$). For a function f, we denote by Λ^f the class of oracle circuits in Λ that have f-oracle gates. Also recall that $\Lambda[s]$ denotes the class of Λ-circuits of size at most s.

Definition 4.1 (Black-Box (ϵ, δ)-Amplification within Λ)**.** For given $\epsilon, \delta > 0$, (ϵ, δ)-*amplification within* Λ is a mapping that associates with a given function $f : \{0,1\}^n \to \{0,1\}$ its *amplified* version, $\textsc{Amp}(f) : \{0,1\}^m \to \{0,1\}$, satisfying the following conditions:

Short Input: $m \leq \mathsf{poly}(n, 1/\epsilon, \log 1/\delta)$,

Nonuniform Λ-Efficiency: $\textsc{Amp}(f) \in \Lambda^f[\mathsf{poly}(m)]$,

Uniform P-Efficiency: $\textsc{Amp}(f) \in \mathsf{P}^f$, and

Reconstruction: $\textsc{Cons}^f(\widetilde{\mathsf{CKT}}(\textsc{Amp}(f), 1/2 - \delta) \to \widetilde{\mathsf{CKT}}(f, \epsilon);\ \mathsf{poly}(n, 1/\epsilon, 1/\delta))$.

We will show that black-box amplification is possible within $\Lambda = \mathsf{AC}^0[p]$, for any prime $p \geq 2$.

Remark 4.2. *In our construction, we actually get a better bound on the parameter m: we have $m \leq O(n \cdot 1/\epsilon \cdot \log^2(1/\delta))$, and $\textsc{Amp}(f) \in \Lambda^f[O(m)]$. Moreover, we get that there is a fixed constant $d_{Amp} \geq 1$ such that (for any prime $p \geq 2$) the depth of the $\mathsf{AC}^0[p]$ circuit for $\mathsf{Amp}(f)$ is at most d_{Amp} plus the depth of the $\mathsf{AC}^0[p]$ circuit for f.*

For $\mathsf{AC}^0[2]$, we shall use standard hardness amplification tools from pseudorandomness: Direct Product and XOR construction. For $\mathsf{AC}^0[p]$, $p \neq 2$, we will need to use something else in place of XOR, as small $\mathsf{AC}^0[p]$ circuits can't compute Parity [Smo87]. We will replace XOR with a MOD_p function, also using an efficient conversion from $\{0, 1, \ldots, p-1\}$-valued functions to Boolean functions, which preserves the required amplification parameters.

First we discuss the Direct Product amplification. For a Boolean function $f : \{0,1\}^n \to \{0,1\}$ and a parameter $k \in \mathbb{N}$, the k-wise direct product of f is $f^k : \{0,1\}^{nk} \to \{0,1\}^k$ where

$$f^k(x_1, \ldots, x_k) = (f(x_1), \ldots, f(x_k))$$

for $x_i \in \{0,1\}^n$, $1 \leq i \leq k$.

It is well-known that the Direct Product construction amplifies hardness of a given function f in the sense that a circuit somewhat nontrivially approximating the function f^k may be used to get a new circuit that approximates the original function f quite well [GNW11], and, moreover, this new circuit for f can be constructed efficiently uniformly [IW01]. We shall use the following algorithm due to [IJKW10] that has optimal parameters (up to constant factors).

Theorem 4.3 (Amplification via Direct Product [IJKW10])**.** *There is a constant c and a probabilistic algorithm $\mathcal{A}$ with the following property. Let $k \in \mathbb{N}$, and let $0 < \epsilon, \delta < 1$ be such that $\delta > e^{-\epsilon k/c}$. For a Boolean function $f : \{0,1\}^n \to \{0,1\}$, let C' be any circuit in $\widetilde{\mathsf{CKT}}(f^k, 1-\delta)$. Given such a circuit C', algorithm $\mathcal{A}$ outputs with probability $\Omega(\delta)$ a circuit $C \in \widetilde{\mathsf{CKT}}(f, \epsilon)$.*

The algorithm $\mathcal{A}$ is a uniform randomized NC^0 algorithm (with one C'-oracle gate), and the produced circuit C is an AC^0 circuit of size $\mathsf{poly}(n, k, \log 1/\epsilon, 1/\delta)$ (with $O((\log 1/\epsilon)/\delta)$ of C'-oracle gates).

Next, we need to convert a non-Boolean function $f^k : \{0,1\}^{kn} \to \{0,1\}^k$ into a Boolean function h such that a circuit approximately computing h would uniformly efficiently yield a circuit approximately computing f^k, where the quality of approximation is essentially preserved.

4.1 XOR construction

For the case of $\mathsf{AC}^0[2]$ circuits, we will use the XOR construction due to Goldreich and Levin [GL89]: Given a $g : \{0,1\}^m \to \{0,1\}^k$, define $g^{GL} : \{0,1\}^{m+k} \to \{0,1\}$ by

$$g^{GL}(x_1, \ldots, x_m, r_1, \ldots, r_k) = \sum_{i=1}^{k} r_i \cdot g(x_1, \ldots, x_m)_i \mod 2.$$

For strings $x, y \in \{0,1\}^k$, let $\langle x, y \rangle$ denote the inner product $\sum_{i=1}^k x_i \cdot y_i \mod 2$. We need the following algorithm of Goldreich and Levin.

Theorem 4.4 ([GL89])**.** *There is a probabilistic algorithm A with the following property. Let $h \in \{0,1\}^k$ be any string, and let $B : \{0,1\}^k \to \{0,1\}$ be any predicate such that*

$$|\mathbf{Pr}_{r \in \{0,1\}^n}[B(r) = \langle h, r \rangle] - 1/2| \geq \gamma$$

for some $\gamma > 0$. Then, given oracle access to B and given γ, the algorithm A runs in time poly$(k, 1/\gamma)$ and outputs a list of size $O(1/\gamma^2)$ such that, with probability at least $1/2$, the string h is on this list.

Theorems 4.3 and 4.4 imply the following.

Theorem 4.5 (Black-Box Amplification within $\mathsf{AC}^0[2]$)**.** *For any $0 < \epsilon, \gamma < 1$, there is black-box (ϵ, γ)-amplification within $\mathsf{AC}^0[2]$.*

Proof. Given $f : \{0,1\}^n \to \{0,1\}$ in $\mathsf{AC}^0[2]$ of size s, and given $0 < \epsilon, \delta < 1$, define $\textsc{Amp}(f)$ as follows:

1. Set $k = \lceil (3c) \cdot 1/\epsilon \cdot \ln 1/\gamma \rceil + 1$, where c is the constant in Theorem 4.3.
2. Define g to be the direct product $f^k : \{0,1\}^{nk} \to \{0,1\}^k$.
3. Define $\textsc{Amp}(f)$ to be $g^{GL} : \{0,1\}^{nk+k} \to \{0,1\}$.

Claim 4.6. *For any $\gamma > 0$, we have $\textsc{Cons}^f(\widetilde{\mathsf{CKT}}(g^{GL}, 1/2-\gamma) \to \widetilde{\mathsf{CKT}}(g, 1-\Omega(\gamma^3)); \mathsf{poly}(n, k, 1/\gamma))$.*

Proof. Suppose we are given a circuit $C' \in \widetilde{\mathsf{CKT}}(g^{GL}, 1/2 - \gamma)$. Let $\mathcal{A}_{GL}$ be the Goldreich-Levin algorithm of Theorem 4.4. Consider the following algorithm $\mathcal{A}_1$ that attempts to compute g:

> For a given input $x \in \{0,1\}^{nk}$, define a circuit $B_x(r) := C'(x, r)$, for $r \in \{0,1\}^k$. Run $\mathcal{A}_{GL}$ on B_x, with parameter $\gamma/2$, getting a list L of k-bit strings. Output a uniformly random k-bit string from the list L.

Correctness Analysis of $\mathcal{A}_1$: By averaging, we get that for each of at least $\gamma/2$ fraction of strings $x \in \{0,1\}^{nk}$, the circuit $B_x(r) := C'(x, r)$ agrees with $g^{GL}(x, r) = \langle g(x), r \rangle$ on at least $1/2 + \gamma/2$ fraction of strings $r \in \{0,1\}^k$. For each such x, the circuit B_x satisfies the condition of Theorem 4.4, and so the Goldreich-Levin algorithm will find, with probability at least $1/2$, a list L of $O(1/\gamma^2)$ strings in $\{0,1\}^k$ that contains the string $g(x)$. Conditioned on the list containing the string $g(x)$, if we output a random string on that list, we get the string $g(x)$ with probability at least $1/|L| \geq \Omega(\gamma^2)$. Over, the fraction of inputs x where $\mathcal{A}_1$ correctly computes $g(x)$ is at least

$$\frac{\gamma}{2} \cdot \frac{1}{2} \cdot \Omega(\gamma^2) \geq \Omega(\gamma^3).$$

The runtime of $\mathcal{A}_1$ is $\mathsf{poly}(|C'|, k, n, 1/\gamma)$. □

By Theorem 4.3, we have

$$\textsc{Cons}^f(\widetilde{\mathsf{CKT}}(f^k, 1-\mu) \to \widetilde{\mathsf{CKT}}(f, \epsilon); \mathsf{poly}(n, k, \log 1/\epsilon, 1/\mu)),$$

as long as $\mu > e^{-\epsilon k/c}$, for some fixed constant $c > 0$. Combining this with Claim 4.6 yields

$$\textsc{Cons}^f(\widetilde{\mathsf{CKT}}(\textsc{Amp}(f), 1/2 - \gamma) \to \widetilde{\mathsf{CKT}}(f, \epsilon); \mathsf{poly}(n, 1/\epsilon, 1/\gamma)),$$

as long as $\gamma^3 > e^{-\epsilon k/c}$, which is equivalent to $\gamma > e^{-\epsilon k/c'}$, for $c' = 3c$. Our choice of k satisfies this condition.

Finally, we check the remaining conditions that black-box amplification $\textsc{Amp}(f)$ must satisfy:

- $(f^k)^{GL}$ is defined on inputs of size $kn + k \leq O(n \cdot 1/\epsilon \cdot \log 1/\gamma)$.
- If $f \in \mathsf{AC}^0[2]$ of size s, then f^k is in $\mathsf{AC}^0[2]$ of size

$$O(s \cdot k) = O(s \cdot 1/\epsilon \cdot \log 1/\gamma),$$

  and $(f^k)^{GL}$ is of size at most the additive term $O(k)$ larger.
- $(f^k)^{GL}$ is in P^f.

Thus, $\textsc{Amp}(f)$ defined above is black-box (ϵ, γ)-amplification of f, as required. □

4.2 $\mathbf{MOD}_p$ construction for prime $p > 2$

For $\mathsf{AC}^0[p]$ circuits, with $p > 2$, we can't use the XOR construction above, as PARITY is not computable by small $\mathsf{AC}^0[p]$ circuits [Smo87]. However, we can use MOD_p instead of XOR in the Goldreich-Levin construction.

The Goldreich-Levin algorithm was generalized to the case of all prime finite fields $\mathbf{GF}(p)$ by Goldreich, Rubinfeld, and Sudan [GRS00]. Let $F = \mathbf{GF}(p)$ be a prime field. For tuples $x, y \in F^k$, let $\langle x, y \rangle$ denote the inner product modulo p, i.e.,

$$\langle x, y \rangle = \sum_{i=1}^{k} x_i \cdot y_i \mod p.$$

Theorem 4.7 ([GRS00])**.** *There is a probabilistic algorithm $\mathcal{A}$ with the following property. Let $h \in F^k$ be any tuple, and let $B : F^k \to F$ be any function such that*

$$\mathbf{Pr}_{r \in F^k}[B(r) = \langle h, r \rangle] \geq 1/p + \gamma$$

for some $\gamma > 0$. Then, given oracle access to B and parameters γ, the algorithm $\mathcal{A}$ runs in time $\mathsf{poly}(k, 1/\gamma)$ and outputs a list of size $O(1/\gamma^2)$ such that, with probability at least $1/2$, the tuple h is on the list.

Now the idea is to amplify a given function f by first defining its direct product f^k, for appropriate k, and then apply the analogue of the Goldreich-Levin construction over $F = \mathbf{GF}(p)$: For $g : \{0,1\}^m \to \{0,1\}^k$, define $g^{GL} : \{0,1\}^m \times F^k \to F$ to be

$$g^{GL}(x_1, \ldots, x_m, r_1, \ldots, r_k) = \sum_{i=1}^{k} r_i \cdot g(x_1, \ldots, x_m)_i,$$

where all arithmetic is over the field F.

Theorem 4.7 guarantees that if we have a circuit that computes g^{GL} on more than $1/p + \gamma$ fraction of inputs, then we can efficiently construct a circuit that computes g on $\Omega(\gamma^3)$ fraction of inputs; the proof is identical to that of Claim 4.6 inside the proof of Theorem 4.5 for the case of $\mathsf{AC}^0[2]$ above.

The only problem is that the function g^{GL} defined here is *not* Boolean-valued, but we need a Boolean function to plug into the NW generator in order to complete our construction of a black-box generator within $\mathsf{AC}^0[p]$. We need to convert g^{GL} into a Boolean function h in such a way

that if h can be computed by some circuit on at least $1/2 + \mu$ fraction of inputs, then g^{GL} can be computed by a related circuit on at least $1/p + \mu'$ fraction of inputs, where μ and μ' are close to each other.

We use von Neumann's idea for converting a coin of unknown bias into a perfectly unbiased coin [vN51]. Given a coin that is "heads" with some (unknown) probability $0 < p < 1$, flip the coin twice in a row, independently, and output

$$\begin{cases} 0 & \text{if the trials were ("heads", "tails"),} \\ 1 & \text{if the trials were ("tails", "heads").} \end{cases}$$

In case both trials came up the same (i.e., both "heads", or both "tails"), flip the coins again.

Observe that, conditioned on producing an answer $b \in \{0,1\}$, the value b is uniform over $\{0,1\}$ (as both conditional probabilities are equal to $p(1-p)/(1-p^2-(1-p)^2)$. The probability of not producing an answer in one attempt is $p^2 + (1-p)^2$, the collision probability of the distribution $(p, 1-p)$. If p is far away from 0 and 1, the probability that we need to repeat the flipping for more than t trials diminishes exponentially fast in t.

In our case, we can think of the value of g^{GL} on a uniformly random input as a distribution over F. Assuming that this distribution is close to uniform over F, we will define a new Boolean function h based on g^{GL} so that the output of h on a uniformly random input is close to uniform over $\{0,1\}$. Our analysis of h will be constructive in the following sense. If we are given a circuit that distinguishes the distribution of the outputs of h from uniform, then we can efficiently construct a circuit that distinguishes the distribution of the outputs of g^{GL} from uniform over F. Finally, using the standard tools from pseudorandomness (converting distinguishers into predictors), we will efficiently construct from this distinguisher circuit a new circuit that computes g^{GL} on noticeably more than $1/p$ fraction of inputs.

The construction of this function h follows the von Neumann trick above. First, for a parameter t, define the following function:

Definition 4.8 (von Neumann trick function)**.**

$$E^{vN} : (F^2)^t \to \{0,1\}$$

as follows: For pairs $(a_1, b_1), \ldots, (a_t, b_t) \in F \times F$, set

$$E^{vN}((a_1,b_1),\ldots,(a_t,b_t)) = \begin{cases} 1 & \text{if, for each } 1 \le i \le t,\ a_i = b_i \\ 1 & \text{if } (a_i,b_i) \text{ is the first unequal pair and } a_i > b_i \\ 0 & \text{if } (a_i,b_i) \text{ is the first unequal pair and } a_i < b_i \end{cases}$$

It is not hard to see that E^{vN} is computable in AC^0 (see Lemma C.1 in Section C). Moreover, for independent uniformly distributed inputs, the output of E^{vN} is a random coin flip, with bias at most $(1/p)^t$.

Claim 4.9. *Let $\mathcal{F}$ be the uniform distribution over the field $F = \mathbf{GF}(p)$, and let $\mathcal{G} = (\mathcal{F}^2)^t$ be the uniform distribution over sequences of t pairs of elements from F. Then*

$$\left|\mathbf{Pr}_{r \in (\mathcal{G}}[E^{vN}(r) = 1] - \mathbf{Pr}_{r \in \mathcal{G}}[E^{vN}(r) = 0]\right| \le p^{-t}.$$

Proof. Conditioned on having some unequal pair in the sample from $\mathcal{G}$, the bias of the random variable $E^{vN}(\mathcal{G})$ is 0. Conditioned on having no such unequal pair, the bias is at most 1.

Note that the collision probability of the uniform distribution over $\mathbf{GF}(p)$ is $\sum_{i=1}^{p} p^{-2} = p^{-1}$. So the probability of having collisions in all t independent samples from $\mathcal{F}^2$ is p^{-t}. Thus, the overall bias is at most p^{-t}. □

Next, given $g^{GL} : D \to F$, for the domain $D = \{0,1\}^m \times F^k$, define $h^{vN} : (D^2)^t \to \{0,1\}$ as follows:

$$h^{vN}((a_1,b_1),\ldots,(a_t,b_t)) = E^{vN}((g^{GL}(a_1),g^{GL}(b_1)),\ldots,(g^{GL}(a_t),g^{GL}(b_t))).$$

Theorem 4.10. *For any $0 < \mu < 1$ and $1 > \gamma > \Omega(\mu/(\log 1/\mu))$, we have*

$$\textsc{Cons}^f(\widetilde{\mathsf{CKT}}(h^{vN}, 1/2-\mu) \to \widetilde{\mathsf{CKT}}(g^{GL}, 1-1/p-\gamma);\ \mathsf{poly}(k,m,\mathsf{poly}(1/\mu))).$$

Proof. Recall some basic definition from pseudorandomness theory. We say that distributions X and Y are computationally (η,s)-indistinguishable, denoted by $X \overset{\eta,s}{\approx} Y$ if, for any circuit T of size s, the probability that T accepts a sample from X is the same as the probability T accepts a sample from Y, to within $\pm\eta$.

We want to show that if h^{vN} is predictable with probability better than $1/2$, then g^{GL} is predictable with probability better than $1/p$. We will argue the contrapositive: suppose g^{GL} is unpredictable, and show that h^{vN} is unpredictable. This will take a sequence of steps.

Let $\mathcal{D}$ denote the uniform distribution over D, $\mathcal{F}$ the uniform distribution over F, and $\mathcal{U}$ the uniform distribution over $\{0,1\}$. Assume g^{GL} is unpredictable by circuits of size s with probability better than $1/p+\gamma$, for some $\gamma > 0$. This implies the following sequence of statements:

1. $(\mathcal{D}, g^{GL}(\mathcal{D})) \overset{2\gamma,\Omega(s)}{\approx} (\mathcal{D},\mathcal{F})$ (unpredictable ⇒ indistinguishable (Lemma B.1))
2. $(\mathcal{D}, g^{GL}(\mathcal{D}))^{2t} \overset{4t\gamma,\Omega(s/t)}{\approx} (\mathcal{D},\mathcal{F})^{2t}$ (hybrid argument)
3. $(\mathcal{D}^{2t}, g^{GL}(\mathcal{D})^{2t}) \overset{4t\gamma,\Omega(s/t)}{\approx} (\mathcal{D}^{2t}, F^{2t})$ (re-arranging)
4. $(\mathcal{D}^{2t}, E^{vN}(g^{GL}(\mathcal{D})^{2t})) \overset{4t\gamma,\Omega((s/t)-\mathsf{poly}(t))}{\approx} (\mathcal{D}^{2t}, E^{vN}(F^{2t}))$ (applying h^{vN})
5. $(\mathcal{D}^{2t}, h^{vN}(\mathcal{D}^{2t})) \overset{4t\gamma+p^{-t},\Omega((s/t)-\mathsf{poly}(t))}{\approx} (\mathcal{D}^{2t},\mathcal{U})$ (by Claim 4.9)

Finally, the last statement implies (via the "indistinguishable to unpredictable" direction) that h^{vN} cannot be computed on more than $1/2+\mu$ fraction of inputs by any circuit of size $\Omega((s/t) - \mathsf{poly}(t))$, where $\mu = \Omega(t\gamma + p^{-t})$. For $t = O(\log 1/\mu)$, we get $\gamma \geq \Omega(\mu/(\log 1/\mu))$.

In the standard way, the sequence of implications above yields an efficient randomized algorithm, with the runtime $\mathsf{poly}(k,m,\log 1/\mu)$, for going in the reverse direction: from a predictor circuit for h^{vN} to a predictor circuit for g^{GL}. To be able to carry out the hybrid argument with uniform algorithms, we need efficient sampleability of the distribution $(\mathcal{D}, g^{GL}(\mathcal{D}))$. Such sampling is possible when we have membership queries to f (as $g^{GL} \in \mathsf{P}^f$); in fact, here it would suffice to have access to uniformly random labelled examples $(x, f(x))$. Another issue is that we need to sample uniformly from $\mathbb{Z}_p$, while we only have access to uniformly random bits. However, it

is easy to devise an efficient sampling algorithm for $\mathbb{Z}_p$, with the distribution statistically almost indistinguishable from uniform over $\mathbb{Z}_p$.[5] □

As for the case of $\mathsf{AC}^0[2]$, we get black-box amplification within $\mathsf{AC}^0[p]$, by following the same proof template as before, with an extra step given by Theorem 4.10.

Theorem 4.11 (Black-Box Amplification within $\mathsf{AC}^0[p]$)**.** *For any $0 < \epsilon, \gamma < 1$, there is black-box (ϵ, γ)-amplification within $\mathsf{AC}^0[p]$.*

Proof. Use the proof of Theorem 4.5 as a template, replacing the XOR construction step by the combination of the MOD_p construction of Theorem 4.7 and the von Neumann trick of Theorem 4.10. The only change in the parameters is the slightly worse dependence on the parameter $1/\gamma$: from $1/\gamma$ to $(1/\gamma) \cdot \log 1/\gamma$, which is at most $1/\gamma^2$, and so can be easily tolerated. □

5 Natural properties imply randomized learning

In this section, we prove the general implication from natural properties to learning algorithms. The "approximate learning" section contains the core idea for all our algorithms. Every other generic algorithm is easily derived from this one.

5.1 Approximate learning

Theorem 5.1 (Learning from a natural property)**.** *Let Λ be any circuit class containing $\mathsf{AC}^0[p]$ for some prime p. Let R be a P-natural property, with largeness at least $1/5$, that is useful against $\Lambda[u]$, for some size function $u\colon \mathbb{N} \to \mathbb{N}$. Then there is a randomized algorithm that, given oracle access to any function $f\colon \{0,1\}^n \to \{0,1\}$ from $\Lambda[s_f]$, produces a circuit $C \in \widetilde{\mathsf{CKT}}(f, \epsilon)$ in time $\mathsf{poly}(n, 1/\epsilon, 2^{u^{-1}(\mathsf{poly}(n, 1/\epsilon, s_f))})$.*

Proof. Let $\textsc{Gen}(f) = \{g_z\}$ be an (ϵ, L)-black-box generator based on f, for $L(n)$ such that

$$\log L(n) > u^{-1}(\mathsf{poly}(n, 1/\epsilon(n), s_f)).$$

Using the nonuniform Λ-efficiency of black-box generators, we have that $g_z \in \Lambda^f[\mathsf{poly}(n, 1/\epsilon)]$, for every z. Hence, we get, by replacing the f-oracle with the Λ-circuit for f, that $g_z \in \Lambda[s_g]$, for some $s_g \leq \mathsf{poly}(n, 1/\epsilon, s_f)$. We want $s_g < u(\log L(n))$. This is equivalent to $u^{-1}(s_g) < \log L(n)$.

Let D be the circuit obtained from the natural property R restricted to truth tables of size $L(n)$. By usefulness, we have

$$\Pr_z[\neg D(g_z) = 1] = 1,$$

and by largeness,

$$\Pr_{y \in \{0,1\}^{L(n)}}[\neg D(y) = 1] \leq 1 - 1/5.$$

So $\neg D$ is a 1/5-distinguisher for $\textsc{Gen}(f)$. By the reconstruction property of black-box generators, we have a randomized algorithm that constructs a circuit $C \in \widetilde{\mathsf{CKT}}(f, \epsilon)$ in time

$$\mathsf{poly}(n, 1/\epsilon(n), L(n)) = \mathsf{poly}(n, 1/\epsilon, 2^{u^{-1}(\mathsf{poly}(n, 1/\epsilon, s_f))}),$$

[5] We divide an interval $[0, 2^{k-1}]$ into p almost equal pieces (all but the last piece are equal to $\lfloor 2^k/p \rfloor$), and check in AC^0 which piece we fall into. The statistical difference between the uniform distribution over $\mathbb{Z}_p$ and this distribution is at most $p/2^k$. So we can make it negligible by choosing k to be a large enough polynomial in the relevant parameters.

as required. □

For different usefulness bounds u, we get different runtimes for our learning algorithm:

- polynomial $\mathsf{poly}(ns_f/\epsilon)$, for $u(n) = 2^{\Omega(n)}$,
- quasipolynomial $\mathsf{quasi\text{-}poly}(ns_f/\epsilon)$, for $u(n) = 2^{n^\alpha}$ where $\alpha < 1$, and
- subsexponential $\mathsf{poly}(n, 1/\epsilon, 2^{(ns_f/\epsilon)^{o(1)}})$, for $u(n) = n^{\omega(1)}$.

Corollary 5.2. *Under the same assumptions as in Theorem 5.1, we also get randomized compression for $\Lambda[\mathsf{poly}]$ to the circuit size at most $O(\epsilon(n) \cdot 2^n \cdot n)$, for any $0 < \epsilon(n) < 1$ such that $\log(\epsilon(n) \cdot 2^n \cdot n) \geq u^{-1}(\mathsf{poly}(n, 1/\epsilon))$.*

Proof. We use Theorem 5.1 to learn a small circuit that computes f on all but at most $\epsilon \cdot 2^n$ inputs, and then patch up this circuit by hardwiring all the error inputs, using extra circuitry of size at most $O(\epsilon \cdot 2^n \cdot n)$. This size will dominate the overall size of the patched-up circuit for the ϵ satisfying the stated condition. □

5.2 Application: Learning and compression algorithms for $\mathsf{AC}^0[p]$

We have the following; see Section A in the appendix for proofs.

Theorem 5.3 ([RR97])**.** *For every prime p, there is an NC^2-natural property of n-variate Boolean functions, with largeness at least $1/2$, that is useful against $\mathsf{AC}^0[p]$ circuits of depth d of size up to $2^{\Omega(n^{1/(2d)})}$.*

This theorem, in conjunction with Theorem 5.1, immediately yields our main application.

Corollary 5.4 (Learning $\mathsf{AC}^0[p]$ in quasipolytime)**.** *For every prime p, there is a randomized algorithm that, using membership queries, learns a given n-variate Boolean function $f \in \mathsf{AC}^0[p]$ of size s_f to within error ϵ over the uniform distribution, in time $\mathsf{quasi\text{-}poly}(ns_f/\epsilon)$.*

Using Corollary 5.2, we also immediately get the following compression result, first proved (with somewhat stronger parameters) by Srinivasan [Sri15].

Corollary 5.5. *There is a randomized compression algorithm for depth-d $\mathsf{AC}^0[p]$ functions that compresses an n-variate function to the circuit size at most 2^{n-n^μ}, for $\mu \geq \Omega(1/d)$.*

6 NW designs cannot be computed in AC^0

In Section 2.5.1 we showed that NW designs (with parameters of interest to us) are computable by small $\mathsf{AC}^0[p]$ circuits, for any prime p. It is natural to ask if one can compute such NW designs by small AC^0 circuits, without modular gates. Here we show that this is *not* possible.

Consider an NW design $S_1, \ldots, S_L \subseteq [n^2]$, where

- each set S_i is of size n,
- the number of sets is $L = 2^\ell$ for $\ell = n^\epsilon$ (for some $\epsilon > 0$), and
- for any two distinct sets S_i and S_j, $i \neq j$, we have $|S_i \cap S_j| \leq \ell$.

To describe such a design, we use the following characteristic function g: for $1 \leq i \leq L$, and for $1 \leq k \leq n^2$,

$$g(i,k) = \begin{cases} 1 & \text{if } k \in S_i \\ 0 & \text{otherwise.} \end{cases}$$

We will prove the following.

Theorem 6.1. *Let $g : \{0,1\}^{\ell+2\log n} \to \{0,1\}$ be the characteristic function for any NW design with the parameters as above. Then g requires depth d AC^0 circuits of size at least $\exp(\ell^{1/d})$.*

To prove this result, we shall define a family of functions f_T, parameterized by sets $T \subseteq [n^2]$: for $1 \leq i \leq L$,

$$f_T(i) = \begin{cases} 1 & \text{if } T \cap S_i \neq \emptyset \\ 0 & \text{otherwise.} \end{cases}$$

Observe that if $g(i,k)$ is computable by AC^0 circuits of depth d and size s, then, for every set T, the function

$$f_T(i) = \vee_{k \in T} \ g(i,k)$$

is computable by AC^0 circuits of depth at most $d+1$ and size $O(s \cdot |T|)$. Therefore, to prove Theorem 6.1, it will suffice to prove the following.

Lemma 6.2. *There exists a set $T \subseteq [n^2]$ such that $f_T : \{0,1\}^\ell \to \{0,1\}$ requires depth $d+1$ AC^0 circuits of size at least $\exp(\ell^{1/d})$.*

The idea of the proof of Lemma 6.2 is to show that for a random set T (of expected size $O(n)$), the function f_T has high average sensitivity (i.e., is likely to flip its value for many Hamming neighbors of a randomly chosen input). By averaging, we get the existence of a particular function f_T of high average sensitivity. On the other hand, it is well-known that AC^0 functions have low average sensitivity. This will imply that f_T must require large AC^0 circuits. We provide the details in the next subsection.

6.1 Proof of Lemma 6.2

Recall that the *sensitivity* of a Boolean function $f : \{0,1\}^n \to \{0,1\}$ at input $x \in \{0,1\}^n$ is defined as the number of Hamming neighbors $y \in \{0,1\}^n$ of x (where y and x differ in exactly one coordinate i, $1 \leq i \leq n$) such that $f(x) \neq f(y)$. The *average sensitivity* of a function f, denoted $\mathbf{AS}(f)$, is the expected sensitivity of f at x, over uniformly random inputs $x \in \{0,1\}^n$.

We use the following result by Boppana [Bop97].

Theorem 6.3 ([Bop97])**.** *The average sensitivity of a size s AC^0 circuit of depth d is at most $O((\log s)^{d-1})$.*

We shall prove the following simple claims that will imply that f_T has high average sensitivity, for a random $T \subseteq [n^2]$ of expected size $t = O(n)$. Below we shall choose a set $T \subseteq [n^2]$ by placing each index k, $1 \leq k \leq n^2$, into T with probability t/n^2, independently, for $t = n/2$. Clearly, the expected size of T is t.

Claim 6.4. *For every $1 \leq i \leq L$, $\mathbf{Pr}_T \ [f_T(i) = 1] \approx 1/2$.*

Proof. The probability a random set T misses all n positions of S_i is

$$\begin{aligned}\left(1-\frac{t}{n^2}\right)^n &\approx 1-\frac{tn}{n^2}\\ &= 1-\frac{t}{n},\end{aligned}$$

which is approximately $1/2$ by our choice of t. □

Claim 6.5. *For every $1 \le i \ne j \le L$, $\mathbf{Pr}_T\ [f_T(i)=1\ \wedge\ f_T(j)=1] \le 1/4 + o(1)$.*

Proof. We have $\mathbf{Pr}_T\ [f_T(i)=1\ \wedge\ f_T(j)=1]$ is equal to

$$\mathbf{Pr}_T\ [T\cap(S_i\cap S_j)\ne\emptyset] + \mathbf{Pr}_T\ [f_T(i)=f_T(j)=1 \mid T\cap(S_i\cap S_j)=\emptyset]\cdot\mathbf{Pr}_T\ [T\cap(S_i\cap S_j)=\emptyset]\,.$$

Using the fact that $|S_i\cap S_j|\le \ell = n^\epsilon$ and arguing as in the proof of Claim 6.4, we get

$$\begin{aligned}\mathbf{Pr}_T\ [T\cap(S_i\cap S_j)\ne\emptyset] &\le (t\ell)/n^2\\ &= \ell/(2n)\\ &= o(1).\end{aligned}$$

Next we have $\mathbf{Pr}_T\ [f_T(i)=f_T(j)=1 \mid T\cap(S_i\cap S_j)=\emptyset]$ equals

$$\mathbf{Pr}_T\ [f_T(i)=1 \mid T\cap(S_i\cap S_j)=\emptyset]\cdot\mathbf{Pr}_T\ [f_T(j)=1 \mid f_T(i)=1\ \wedge\ T\cap(S_i\cap S_j)=\emptyset]\,.$$

Conditioned on T missing the intersection $S_i\cap S_j$, the conditional probability that T intersects S_i is

$$\begin{aligned}1-\left(1-\frac{t}{n^2}\right)^{n-|S_i\cap S_j|} &\approx \frac{n-|S_i\cap S_j|}{2n}\\ &\le \frac{1}{2}.\end{aligned}$$

Similarly, conditioned on T missing the intersection $S_i\cap S_j$ but intersecting S_i, the conditional probability of T intersecting S_j is also approximately at most $1/2$ (following the same calculations as for the case of S_i above).

Putting everything together, we get that $\mathbf{Pr}_T\ [f_T(i)=1\ \wedge\ f_T(j)=1] \le 1/4+o(1)$. □

Claim 6.6. *For every $1\le i\ne j\le L$, $\mathbf{Pr}_T\ [f_T(i)\ne f_T(j)] \ge \frac{1}{5}$.*

Proof. For every fixed $i\ne j$, we have

$$\begin{aligned}\mathbf{Pr}_T\ [f_T(i)\ne f_T(j)] &\ge \mathbf{Pr}_T\ [f_T(i)=1\ \wedge\ f_T(j)=0]\\ &= \mathbf{Pr}_T\ [f_T(i)=1] - \mathbf{Pr}_T\ [f_T(i)=1\ \wedge\ f_T(j)=1],\end{aligned}$$

which, by Claims 6.4 and 6.5, is at least $1/2-1/4-o(1) = 1/4-o(1) > 1/5$. □

Claim 6.7. *There exists a set T such that $\mathbf{AS}(f_T)\ge \ell/5$.*

Proof. For a string $x \in \{0,1\}^n$, we denote by $N(x)$ the set of all strings $y \in \{0,1\}^n$ that differ from x in exactly one coordinate; that is, $N(x)$ is the set of all Hamming neighbors of x in the Boolean cube $\{0,1\}^n$. Also, for a condition A, we denote by $\{A\}$ the indicator function of A, i.e., $\{A\} = 1$ if the condition A is true, and $\{A\} = 0$ otherwise.

We have

$$\begin{aligned}
\mathbf{Exp}_T\left[\mathbf{AS}(f_T)\right] &= \mathbf{Exp}_{T\subseteq[n^2],i\in\{0,1\}^\ell}\left[\sum_{j\in N(i)}\{f_T(i)\neq f_T(j)\}\right] \\
&= \mathbf{Exp}_i\left[\sum_{j\in N(i)}\mathbf{Exp}_T[\{f_T(i)\neq f_T(j)\}]\right] \\
&= \mathbf{Exp}_i\left[\sum_{j\in N(i)}\mathbf{Pr}_T[f_T(i)\neq f_T(j)]\right] \\
&\geq \mathbf{Exp}_i\left[\sum_{j\in N(i)}\frac{1}{5}\right] \qquad \text{(by Claim 6.6)} \\
&= \frac{\ell}{5}.
\end{aligned}$$

By averaging, there exists a set T, such that $\mathbf{AS}(f_T) \geq \ell/5$. □

Now we finish the proof of Lemma 6.2. Suppose the function f_T given by Claim 6.7 is computed by an AC^0 circuit of depth $d+1$ and size s. By Theorem 6.3, we get that $\mathbf{AS}(f_T) \leq O((\log s)^d)$. It follows that

$$\frac{\ell}{5} \leq O((\log s)^d),$$

which implies that $s \geq \exp(\ell^{1/d})$, as required.

7 Conclusions

For our applications, we need Λ strong enough to carry out a (version of) the construction, yet weak enough to have a natural property useful against it. Here we show that $\Lambda = \mathsf{AC}^0[p]$ for any prime p satisfies both conditions. A logical next step would be ACC^0: if one can get a natural property useful against ACC^0, for example by naturalizing Williams's [Wil14b] proof, then a learning algorithm for ACC^0 would follow. (As MOD_p can be simulated with MOD_m, $m = p \cdot a$ gates by duplicating each input to the Mod_m gate a times (without any penalty in the number of gates), our construction for MOD_p can be applied directly by taking p to be any prime factor of m.)

Can we get an *exact* compression algorithm for $\mathsf{AC}^0[p]$ (or even AC^0) functions that would produce circuits of *subexponential* size? Can our learning algorithm be derandomized? Finally, is there a way to get nontrivial SAT algorithms from natural properties?

References

[AvM12] Baris Aydinlioglu and Dieter van Melkebeek. Nondeterministic circuit lower bounds from mildly de-randomizing arthur-merlin games. In *Proceedings of the 27th Conference on Computational Complexity, CCC 2012, Porto, Portugal, June 26-29, 2012*, pages 269–279, 2012.

[BFNW93] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993.

[BIS90] David A. Mix Barrington, Neil Immerman, and Howard Straubing. On uniformity within nc^1. *J. Comput. Syst. Sci.*, 41(3):274–306, 1990.

[BIS12] Paul Beame, Russell Impagliazzo, and Srikanth Srinivasan. Approximating AC^0 by Small Height Decision Trees and a Deterministic Algorithm for $\#AC^0$SAT. In *Proceedings of the 27th Conference on Computational Complexity, CCC 2012, Porto, Portugal, June 26-29, 2012*, pages 117–125, 2012.

[Bop97] Ravi B. Boppana. The average sensitivity of bounded-depth circuits. *Information Processing Letters*, 63(5):257–261, September 1997.

[Bra10] Mark Braverman. Polylogarithmic independence fools AC^0 circuits. *Journal of the ACM*, 57:28:1–28:10, 2010.

[CIKK15] Marco Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Tighter connections between derandomization and circuit lower bounds. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2015, August 24-26, 2015, Princeton, NJ, USA*, pages 645–658, 2015.

[CK15] Ruiwen Chen and Valentine Kabanets. Correlation bounds and #sat algorithms for small linear-size circuits. In Dachuan Xu, Donglei Du, and Dingzhu Du, editors, *Computing and Combinatorics - 21st International Conference, COCOON 2015, Beijing, China, August 4-6, 2015, Proceedings*, volume 9198 of *Lecture Notes in Computer Science*, pages 211–222. Springer, 2015.

[CKK+15] Ruiwen Chen, Valentine Kabanets, Antonina Kolokolova, Ronen Shaltiel, and David Zuckerman. Mining circuit lower bound proofs for meta-algorithms. *Computational Complexity*, 24(2):333–392, 2015.

[CKS14] Ruiwen Chen, Valentine Kabanets, and Nitin Saurabh. An Improved Deterministic #SAT Algorithm for Small De Morgan Formulas. In *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, pages 165–176, 2014.

[CS15] Ruiwen Chen and Rahul Santhanam. Improved algorithms for sparse MAX-SAT and MAX-k-CSP. In *Theory and Applications of Satisfiability Testing - SAT 2015 - 18th International Conference, Austin, TX, USA, September 24-27, 2015, Proceedings*, pages 33–45, 2015.

[FK09] Lance Fortnow and Adam R. Klivans. Efficient learning algorithms yield circuit lower bounds. *J. Comput. Syst. Sci.*, 75(1):27–36, 2009.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In David S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 25–32. ACM, 1989.

[GNW11] Oded Goldreich, Noam Nisan, and Avi Wigderson. On Yao's XOR-lemma. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 273–301. Springer, 2011.

[GRS00] Oded Goldreich, Ronitt Rubinfeld, and Madhu Sudan. Learning polynomials with queries: The highly noisy case. *SIAM J. Discrete Math.*, 13(4):535–570, 2000.

[Hås86] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pages 6–20, 1986.

[HVV06] Alexander Healy, Salil Vadhan, and Emanuele Viola. Using nondeterminism to amplify hardness. *SIAM Journal on Computing*, 35(4):903–931, 2006.

[IJKW10] Russell Impagliazzo, Ragesh Jaiswal, Valentine Kabanets, and Avi Wigderson. Uniform direct product theorems: Simplified, optimized, and derandomized. *SIAM J. Comput.*, 39(4):1637–1665, 2010.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *J. Comput. Syst. Sci.*, 65(4):672–694, 2002.

[IMP12] 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, 2012.

[IMZ12] Russell Impagliazzo, Raghu Meka, and David Zuckerman. Pseudorandomness from shrinkage. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 111–119, 2012.

[IW97] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 220–229. ACM, 1997.

[IW01] Russell Impagliazzo and Avi Wigderson. Randomness vs time: Derandomization under a uniform assumption. *J. Comput. Syst. Sci.*, 63(4):672–688, 2001.

[KI04] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

[KKO13] Adam Klivans, Pravesh Kothari, and Igor Carboni Oliveira. Constructing hard functions using learning algorithms. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, Palo Alto, California, USA, 5-7 June, 2013*, pages 86–97. IEEE, 2013.

[LMN93] Nathan Linial, Yishay Mansour, and Noam Nisan. Constant Depth Circuits, Fourier Transform, and Learnability. *J. ACM*, 40(3):607–620, 1993.

[Lup58] Oleg B. Lupanov. On the synthesis of switching circuits. 119(1):23–26, 1958. English translation in *Soviet Mathematics Doklady*.

[Lup59] Oleg B. Lupanov. A method of circuit synthesis. *Izvestiya VUZ, Radiofizika*, 1(1):120–140, 1959. (in Russian).

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[O'D04] Ryan O'Donnell. Hardness amplification within NP. *J. Comput. Syst. Sci.*, 69(1):68–94, 2004.

[Raz87] Alexander A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical Notes*, 41(4):333–338, 1987.

[RR97] Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997.

[San10] 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 Computer Society, 2010.

[Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 77–82. ACM, 1987.

[Sri15] Srikanth Srinivasan. A Compression Algorithm for $AC^0[\oplus]$ circuits using Certifying Polynomials. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:142, 2015.

[ST12] K. Seto and S. Tamaki. A satisfiability algorithm and average-case hardness for formulas over the full binary basis. In *Proceedings of the Twenty-Seventh Annual IEEE Conference on Computational Complexity*, pages 107–116, 2012.

[SW15] Rahul Santhanam and Richard Ryan Williams. Beating exhaustive search for quantified boolean formulas and connections to circuit complexity. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 231–241, 2015.

[Tal15] Avishay Tal. #SAT Algorithms from Shrinkage. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:114, 2015.

[Tre05] Luca Trevisan. On uniform amplification of hardness in NP. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 31–38. ACM, 2005.

[Uma03] Christopher Umans. Pseudo-random generators for all hardnesses. *J. Comput. Syst. Sci.*, 67(2):419–440, 2003.

[vN51] John von Neumann. Various techniques used in connection with random digits. *J. Research Nat. Bur. Stand., Appl. Math. Series*, 12:36–38, 1951.

[Wil13] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM J. Comput.*, 42(3):1218–1244, 2013.

[Wil14a] Ryan Williams. New algorithms and lower bounds for circuits with linear threshold gates. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 194–202, 2014.

[Wil14b] Ryan Williams. Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2:1–2:32, 2014.

A Natural properties useful against $\mathsf{AC}^0[p]$

Here we present the natural properties useful against the class of AC^0 circuits with mod p gates, for any fixed prime p, as given in [RR97]. We follow the lower bound of Razborov [Raz87] (showing that MAJORITY is not in $\mathsf{AC}^0[2]$) to get a natural property useful against $\mathsf{AC}^0[2]$, and the lower bound of Smolensky [Smo87] (showing that PARITY is not in $\mathsf{AC}^0[p]$, for any prime $p \neq 2$) for the case of $\mathsf{AC}^0[p]$ for any prime $p > 2$. In both cases, the natural property is NC^2-computable, and is useful for circuit size up to $2^{\Omega(n^{1/(2d)})}$, where d is the circuit depth, and n is the input size.

A.1 The case of $\mathsf{AC}^0[2]$

Theorem A.1 ([RR97])**.** *There is an NC^2-natural property of n-variate Boolean functions, with largeness at least $1/2$, that is useful against $\mathsf{AC}^0[2]$ circuits of depth d of size up to $2^{\Omega(n^{1/(2d)})}$.*

Proof. For $0 \leq a, b \leq n$, define a linear transformation $A_{a,b}$ that maps a Boolean function $f : \{0,1\}^n \to \{0,1\}$ to a matrix $M = A_{a,b}(f)$ of dimension $\binom{n}{a} \times \binom{n}{b}$, whose rows are indexed by size a subsets of $[n]$, and rows by size b subsets of $[n]$. For every $K \subseteq [n]$, define the set

$$Z(K) = \{(x_1, \ldots, x_n) \in \{0,1\}^n \mid \forall i \in K,\ x_i = 0\}.$$

For a size a subset $I \subseteq [n]$ and size b subset $J \subseteq [n]$, define

$$M_{I,J} = \oplus_{x \in Z(I \cup J)} f(x).$$

Razborov [Raz87] showed that if $\mathrm{rank}(A_{a,b}(f)) \geq \Omega(2^n/n^2)$, for $a = n/2 - \sqrt{n}$ and some $b \leq a$, then f requires $\mathsf{AC}^0[2]$ circuits of depth d of size at least $2^{\Omega(n^{1/(2d)})}$. He also showed the existence of an n-variate Boolean (symmetric) function h such that, for some $0 \leq b \leq a$, $\mathrm{rank}(A_{a,b}(h)) \geq \frac{2^n}{70n^2}$, and hence, h requires large $\mathsf{AC}^0[2]$ circuits.

This yields the following natural property useful against large $\mathsf{AC}^0[2]$ circuits:

Given an n-variate Boolean function f, construct matrices $M_b = A_{a,b}(f)$ for $a = n/2 - \sqrt{n}$ and for every $0 \leq b \leq a$. Accept f if, for at least one b, $\text{rank}(M_b) \geq \frac{2^n}{140n^2}$.

First, observe that such a property of n-variate Boolean function f is computable in NC^2: we first construct $O(n)$ matrices of size at most $2^n \times 2^n$, and then compute the rank (over $\mathbf{GF}(2)$) of each of them. Thus, this property is NC^2-natural.

Secondly, by Razborov's result mentioned above, any f accepted by the property must require $\mathsf{AC}^0[2]$ depth d circuits of size at least $2^{\Omega(n^{1/(2d)})}$. Thus we have usefulness against exponential-size circuits.

Finally, to argue largeness, we use the function h mentioned above with $\text{rank}(A_{a,b}(h)) \geq \frac{2^n}{70n^2}$, for some $0 \leq b \leq a$. For each Boolean function f, we will show that either $A_{a,b}(f)$ or $A_{a,b}(f \oplus h)$ has rank at least $\frac{2^n}{140n^2}$, which implies that at least $1/2$ of all Boolean functions are accepted by our property.

Indeed, since $A_{a,b}$ is an $\mathbf{GF}(2)$-linear map, and using the subadditivity of rank, we get

$$\begin{aligned}\text{rank}(A_{a,b}(h)) &= \text{rank}(A_{a,b}(h \oplus f \oplus f)) \\ &\leq \text{rank}(A_{a,b}(h \oplus f)) + \text{rank}(A_{a,b}(f)).\end{aligned}$$

Thus, at least one of $A_{a,b}(f)$ or $A_{a,b}(f \oplus h)$ must have the rank at least $1/2$ of the rank of $A_{a,b}(h)$, as required. □

A.2 The case of $\mathsf{AC}^0[p]$ for all primes $p > 2$

Theorem A.2 ([RR97])**.** *For every prime $p > 2$, there is an NC^2-natural property of n-variate Boolean functions, with largeness at least $1/2$, that is useful against $\mathsf{AC}^0[p]$ circuits of depth d of size up to $2^{\Omega(n^{1/(2d)})}$.*

Proof. Let f be a given n-variate Boolen function. Without loss of generality, assume n is odd. Denote by L the vector space of all multilinear polynomials of degree less than $n/2$ over $\mathbf{GF}(p)$. Let $\bar{f}$ be the unique multilinear polynomial over $\mathbf{GF}(p)$ that represents f on the Boolean cube $\{-1, 1\}^n$ (after the linear transfofrmation mapping the Boolean 0 to $1 \mod p$, and the Boolean 1 to $-1 \mod p$), i.e., f and $\bar{f}$ agree over all points of $\{-1, 1\}^n$.

The natural property given by [RR97] is the following:

Given an n-variate Boolean function f, construct its unique multilinear polynomial extension $\bar{f}$ over $\mathbf{GF}(p)$. Accept f if $\dim(\bar{f}L + L) \geq \frac{3}{4} \cdot 2^n$ (over $\mathbf{GF}(p)$).

It is easy to see that this property is computable in NC^2. It is also argued in [RR97] that this property has largeness at least $1/2$. Finally, it also follows from [RR97], based on Smolensky's lower bound proof [Smo87], that any n-variate function f accepted by this property must have $\mathsf{AC}^0[p]$ circuits of depth d of size at least $2^{\Omega(n^{1/(2d)})}$.

Indeed, Smolensky [Smo87] shows that, for every Boolean function f computed by an $\mathsf{AC}^0[p]$ circuit of depth d and size s, there exists a multilinear polynomial q over $\mathbf{GF}(p)$ of degree $D = O(\log^d(s/\epsilon))$ that agrees with f on all but at most $w = \epsilon 2^n$ points W of the Boolean cube, where we think of ϵ as a small constant (e.g., $\epsilon = 0.2$). For any such f that also satisfies the condition $\dim(\bar{f}L + L) \geq \frac{3}{4} \cdot 2^n$, we will show that $D \geq \Omega(\sqrt{n})$. This would imply that any such f must have d-depth $\mathsf{AC}^0[p]$ circuits of size $2^{\Omega(n^{1/(2d)})}$, as required.

Suppose some f can be approximated by a multilinear degree D polynomial on all Boolean points except the set W of size $w \leq \epsilon 2^n$, for small constant ϵ (to be determined). Suppose that f also satisfies the condition $\dim(\bar{f}L + L) \geq \frac{3}{4} \cdot 2^n$ over $\mathbf{GF}(p)$. Using Smolensky's arguments from [Smo87], we get that at least $p^{2^n(3/4-\epsilon)}$ functions from $\{-1,1\}^n \setminus W$ to $\mathbf{GF}(p)$ are computable by multilinear polynomials over $\mathbf{GF}(p)$ of degree at most $(n-1)/2 + D$. For $D = \lambda\sqrt{n}$, with some $\lambda > 0$, the number of distinct multilinear monomials of degree at most $(n-1)/2 + D$ is

$$\begin{aligned}
\sum_{i=0}^{(n-1)/2+D} \binom{n}{i} &= \sum_{i=0}^{(n-1)/2} \binom{n}{i} + \sum_{i=(n-1)/2+1}^{(n-1)/2+D} \binom{n}{i} \\
&\leq \frac{1}{2} \cdot 2^n + D \cdot \binom{n}{\frac{n-1}{2}} \\
&\leq \frac{1}{2} \cdot 2^n + D \cdot \frac{2^n}{\sqrt{\pi n/2}} \cdot \left(1 + O\left(\frac{1}{\sqrt{n}}\right)\right) \qquad \text{(by Stirling's approximation)} \\
&\leq 2^n \cdot \left(\frac{1}{2} + \frac{2D}{\sqrt{\pi n}}\right) \\
&= 2^n \cdot \left(\frac{1}{2} + \lambda \cdot \frac{2}{\sqrt{\pi}}\right),
\end{aligned}$$

which can be made at most $(0.51) \cdot 2^n$, by taking λ a sufficiently small constant (e.g., $\lambda = \sqrt{\pi}/20$).

Thus, the number of distinct multilinear polynomials over $\mathbf{GF}(p)$ of degree at most $(n-1)/2+D$ is at most $p^{0.51 \cdot 2^n}$. On the other hand, as mentioned above, there are at least $p^{(3/4-\epsilon)2^n}$ functions from $\{-1,1\}^n \setminus W$ to $\mathbf{GF}(p)$ that are supposed to be computable by such low-degree multilinear polynomials. For ϵ small enough so that $3/4 - \epsilon > 0.51$ (e.g., $\epsilon = 0.2$), we get that there are too many functions to be represented by low-degree polynomials. So it must be the case that the degree $D > \lambda\sqrt{n}$, for some constant $\lambda > 0$. □

B Yao's "Distinguisher to Predictor" reduction

The following result is a simple generalization of Yao's "distinguisher to predictor" reduction for the case of non-binary alphabets. We give the proof as we could not find a reference in the literature for this version of the result.

Lemma B.1 (Yao)**.** *Let $f\colon \{0,1\}^n \to \mathbb{Z}_p$. Suppose there is a function $T\colon \{0,1\}^n \times \mathbb{Z}_p \to \{0,1\}$ such that*

$$\mathbf{Pr}_{x \in \{0,1\}^n}[T(x, f(x)) = 1] - \mathbf{Pr}_{x \in \{0,1\}^n, g \in \mathbb{Z}_p}[T(x,g) = 1] \geq \epsilon, \tag{2}$$

then the following algorithm P computes f with probability at least $1/p + \epsilon/(p-1)$ with respect to the uniform distribution over $\{0,1\}^n$:

> *On input $x \in \{0,1\}^n$, pick a uniformly random $g \in \mathbb{Z}_p$. Compute $b = T(x,g)$. If $b = 1$, then output g; otherwise, output a uniformly random $g' \in \mathbb{Z}_p \setminus \{g\}$.*

Proof. Using Bayes's formula, the probability that the algorithm P above is correct on a uniformly random $x \in \{0,1\}^n$, $\mathbf{Pr}_x[P(x) = f(x)]$, can be written as the sum of the following two expressions:

$$\mathbf{Pr}_{x,g}[T(x,g) = 1 \mid g = f(x)] \cdot \mathbf{Pr}_{x,g}[g = f(x)], \tag{3}$$

and

$$\mathbf{Pr}_{x,g,g'}[T(x,g)=0 \;\&\; g'=f(x) \mid g \neq f(x)] \cdot \mathbf{Pr}_{x,g}[g \neq f(x)] \tag{4}$$

where x is a uniformly random sample from $\{0,1\}^n$, g is a uniformly random sample from $\mathbb{Z}_p$, and g' is uniform over the set $\mathbb{Z}_p \setminus \{g\}$. Since g is independent of x, we have $\mathbf{Pr}_{x,g}[g=f(x)] = 1/p$. Thus we can replace the last factor in Eq. (3) by $1/p$, and the last factor in Eq. (4) by $(p-1)/p$.

Next, applying Bayes's formula to the first factor of Eq. (4), we can re-write this factor as

$$\mathbf{Pr}_{x,g}[T(x,g)=0 \mid g \neq f(x)] \cdot \mathbf{Pr}_{x,g'}[g'=f(x) \mid g \neq f(x),\ T(x,g)=0],$$

which equals

$$\mathbf{Pr}_{x,g}[T(x,g)=0 \mid g \neq f(x)] \cdot \frac{1}{p-1} \tag{5}$$

(since $f(x) \neq g$ and $g' \in \mathbb{Z}_p \setminus \{g\}$ is independent of x and g).

Putting Eqs. (3)–(5) together, we get

$$\begin{aligned}\mathbf{Pr}_x[P(x)=g(x)] &= \frac{1}{p} \cdot (\mathbf{Pr}_x[T(x,f(x))=1] + \mathbf{Pr}_{x,g}[T(x,g)=0 \mid g \neq f(x)]) \\ &= \frac{1}{p} \cdot (\mathbf{Pr}_x[T(x,f(x))=1] + (1 - \mathbf{Pr}_{x,g}[T(x,g)=1 \mid g \neq f(x)])) .\end{aligned}$$

So we have

$$\mathbf{Pr}_x[P(x)=g(x)] = \frac{1}{p} + \frac{1}{p} \cdot (\mathbf{Pr}_x[T(x,f(x))=1] - \mathbf{Pr}_{x,g}[T(x,g)=1 \mid g \neq f(x)]) . \tag{6}$$

On the other hand, we have

$$\mathbf{Pr}_{x,g}[T(x,g)=1] = \frac{1}{p} \cdot \mathbf{Pr}[T(x,f(x))=1] + \left(1 - \frac{1}{p}\right) \cdot \mathbf{Pr}[T(x,g)=1 \mid g \neq f(x)].$$

Therefore, we get

$$\mathbf{Pr}[T(x,f(x))=1] - \mathbf{Pr}[T(x,g)=1] = \frac{p-1}{p} \cdot (\mathbf{Pr}[T(x,f(x))=1] - \mathbf{Pr}[T(x,g)=1 \mid g \neq f(x)]) ,$$

and so, using Eq. (2), we have

$$\mathbf{Pr}[T(x,f(x))=1] - \mathbf{Pr}[T(x,g)=1 \mid g \neq f(x)] \geq \frac{p}{p-1} \cdot \epsilon. \tag{7}$$

Plugging in Eq. (7) into Eq. (6), we conclude

$$\mathbf{Pr}[P(x)=f(x)] \geq \frac{1}{p} + \frac{\epsilon}{p-1},$$

as required. □

C The von Neumann function in AC^0

It is straightforward to implement the von Neumann trick in AC^0 by using the Descriptive Complexity framework to write uniform AC^0 circuits as formulas of first-order logic (FO) over finite models. Specifically, DLOGTIME-uniform AC^0 is captured by FO-formulas over finite models equipped with the following relations: $\{=, <, +, \times, BIT\}$. For details on how first-order logic corresponds to circuit classes, see [BIS90].

To code a length-n string of coinflips $s \in \{H, T\}^n$ as a finite model, we think of the universe set as representing positions or indices into s. Speficially, start with a size-n model equipped with the "default" relations $\{=, <, +, \times, BIT\}$, where the universe elements are interpreted as the n-initial prefix of $\mathbb{N}$ for the purpose of these relations. Then add the following unary relations to represent the character at each position of s:

$$H(i) = \begin{cases} \text{true} & \text{if } s[i] = H \\ \text{false} & \text{otherwise} \end{cases}$$

$$T(i) = \begin{cases} \text{true} & \text{if } s[i] = T \\ \text{false} & \text{otherwise} \end{cases}$$

See definition 4.8 for a complete description of E^{vN}, the von Neumann trick function. Using the above coding of strings into finite models, we prove the following:

Lemma C.1. $E^{vN} \in \mathsf{AC}^0$

Proof. We write the von Neumann trick as a FO forumla by considering, for the t trials, $2t$-size finite models equipped with the heads and tails relations. Our objective is to detect if the first mismatched pair of indicies is HT or TH, returning true if this pair is TH. We consider a trial to have failed if both flips match. So our formula should say "the first trial that didn't fail is TH ". For now, assume that the following predicates TRIAL and FAIL are FO-expressible:

$$TRIAL(i, j) = \begin{cases} \text{true} & \text{if } (i, j) \text{ are a trial pair with } i < j \\ \text{false} & \text{otherwise} \end{cases}$$

$$FAIL(i, j) = \begin{cases} \text{true} & \text{if } (i, j) \text{ are a failed TRIAL pair} \\ \text{false} & \text{otherwise} \end{cases}$$

We can use FO to detect the first useful trial by asserting that every previous trial failed:

$$\exists i, j (TRIAL(i, j) \wedge T(i) \wedge H(j) \wedge \forall k, \ell \; (k < i \wedge \ell < j \implies FAIL(k, \ell)))$$

This formula is true iff the first useful coinflip is TH. Otherwise, if the first useful trial is HT or there are no useful trials, it is false. This is exactly the behavior we want to implement the the von Neumann trick. All that remains is to give FO formulas for TRIAL and FAIL. These are straightforward, because they involve only simple arithmetic on indices of the string and we have built-in numeric predicates for this. □

ECCC ISSN 1433-8092

http://eccc.hpi-web.de