Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 69 (2018)

Constant-round interactive proof systems for AC0[2] and NC1

Oded Goldreich[*] Guy N. Rothblum[†]

April 16, 2018

Abstract

We present constant-round interactive proof systems for sufficiently uniform versions of $\mathcal{AC}^0[2]$ and $\mathcal{NC}^1$. Both proof systems are doubly-efficient, and offer a better trade-off between the round complexity and the total communication than the work of Reingold, Rothblum, and Rothblum (*STOC*, 2016). Our proof system for $\mathcal{AC}^0[2]$ supports a more relaxed notion of uniformity and offers a better trade-off between the number of rounds and the round complexity that our proof system for $\mathcal{NC}^1$. We observe that all three aforementioned systems yield constant-round doubly-efficient proof systems for the All-Pairs Shortest Paths problem.

Contents

[*] Department of Computer Science, Weizmann Institute of Science, Rehovot, Israel. oded.goldreich@weizmann.ac.il

[†] Department of Computer Science, Weizmann Institute of Science, Rehovot, Israel. rothblum@alum.mit.edu

ISSN 1433-8092

1 Introduction

The notion of interactive proof systems, put forward by Goldwasser, Micali, and Rackoff [9], and the demonstration of their power by Lund, Fortnow, Karloff, and Nisan [13] and Shamir [17] are among the most celebrated achievements of complexity theory. Recall that an interactive proof system for a set S is associated with an interactive verification procedure, V, that can be made to accept any input in S but no input outside of S. That is, there exists an interactive strategy for the prover that makes V accepts any input in S, but no strategy can make V accept an input outside of S, except with negligible probability. (See [3, Chap. 9] for a formal definition as well as a wider perspective.)

The original definition does not restrict the complexity of the strategy of the prescribed prover and the constructions of [13, 17] use prover strategies of high complexity. Seeking to make interactive proof systems available for a wider range of applications, Goldwasser, Kalai and Rothblum [8] put forward a notion of *doubly-efficient* interactive proof systems. In these proof systems the prescribed prover strategy can be implemented in polynomial-time and the verifier's strategy can be implemented in almost-linear-time. (We stress that unlike in *argument systems*, the soundness condition holds for all possible cheating strategies, and not only for feasible ones.) Restricting the prescribed prover to run in polynomial-time implies that such systems may exist only for sets in $\mathcal{BPP}$, and thus a polynomial-time verifier can check membership in such sets by itself. However, restricting the verifier to run in almost-linear-time implies that something can be gained by interacting with a more powerful prover, even though the latter is restricted to polynomial-time.

The potential applicability of doubly-efficient interactive proof systems was demonstrated by Goldwasser, Kalai and Rothblum [8], who constructed such proof systems for any set that has log-space uniform circuits of bounded depth (e.g., log-space uniform $\mathcal{NC}$). A more recent work of Reingold, Rothblum, and Rothblum [16] provided such (constant-round) proof systems for any set that can be decided in polynomial-time and a bounded amount of space (e.g., for all sets in $\mathcal{SC}$). In our prior works [5, 7], we presented simpler and more efficient constructions of doubly-efficient interactive proof systems for some special cases: In particular, in [5] we considered a class of "locally-characterizable sets", and in [7] we considered the problem of counting t-cliques in graphs.

In this work we consider the construction of constant-round doubly-efficient interactive proof systems for (sufficiently uniform) versions of $\mathcal{AC}^0[2]$ and $\mathcal{NC}^1$. We mention that the proof systems for $\mathcal{NC}$ constucted by Goldwasser, Kalai and Rothblum [8] use $O(d(n)\log n)$ rounds, where $d(n)$ is the depth of the n^{th} circuit. Building on their techniques, Kalai and Rothblum have observed the existence of a constant-round proof system for a highly-uniform version of $\mathcal{NC}^1$, but their notion of uniformity was quite imposing and they never published their work [12]. In Section 3, we use similar ideas towards presenting a constant-round proof system for a sufficiently uniform version of $\mathcal{NC}^1$, which we believe to be less imposing (see also the overview in Section 1.4), but our main contribution is in presenting such a proof system for a sufficiently uniform version of $\mathcal{AC}^0[2]$: The latter proof system is more efficient and refers to a more relaxed notion of uniformity.

1.1 Our main result: A proof system for $\mathcal{AC}^0[2]$

We present constant-round doubly-efficient interactive proof systems for sets acceptable by (sufficiently uniform) constant-depth polynomial-size Boolean circuits of unbounded fan-in and parity gates (i.e., the class $\mathcal{AC}^0[2]$). Note that this class contains "seemingly hard problems in $\mathcal{P}$" (e.g., the t-`CLIQUE` problem for n-vertex graphs can be expressed as a highly uniform DNF with n^t terms (each depending on $\binom{t}{2}$ variables)). Postponing, for a moment, a clarification of what is meant by "sufficiently uniform", our result reads

Theorem 1 (constant-round doubly-efficient interactive proofs for $\mathcal{AC}^0[2]$, loosely stated): *For constants $c, d \in \mathbb{N}$, suppose that $\{C_n : \{0,1\}^n \to \{0,1\}\}$ is a sufficiently uniform family of Boolean circuits with unbounded fan-in parity and conjunction gates such that C_n has size at most n^c and depth d. Then, for every $\delta \in (0,1]$, the set $\{x : C_{|x|}(x)=1\}$ has a $O(cd/\delta)$-round interactive proof system in which the verifier runs in time $O(n^{1+o(1)})$, the prescribed prover can be implemented in time $O(n^{c+o(1)})$, and the total communication is n^δ.*

We mention that the work of Reingold, Rothblum, and Rothblum [16] implies that log-space uniform $\mathcal{AC}^0[2]$ (actually, even log-space uniform $\mathcal{NC}^1$)[1] has constant-round doubly-efficient interactive proof systems. One advantage of our construction over [16] is that, being tailored to $\mathcal{AC}^0[2]$, it is much simpler and more transparent. In addition, the round complexity of our proof systems is considerably better than the round-complexity in [16]; specifically, we present a $O(1/\delta)$-round system with total communication n^δ, whereas in [16] obtaining total communication n^δ requires $\exp(\widetilde{O}(1/\delta))$ many rounds.

Corollaries. Using Theorem 1, we obtain a constant-round doubly-efficient interactive proof system for the *All Pairs Shortest Path* (APSP) problem (see background in [19]). Such a proof system follows also from the work of [16], but this fact was not observed before. The key observation is that verifying the value of APSP can be reduced to matrix multiplication in the $(\min,+)$-algebra *via a doubly-efficient* NP-proof system.

Recall that matrix multiplication in the $(\min,+)$-algebra refers to the case that multiplication is replace by addition and the sum is replace by the minimum; that is, the product of the matrices $A = (a_{i,j})_{i,j\in[n]}$ and $B = (b_{i,j})_{i,j\in[n]}$, denoted $A * B$, equals $C = (c_{i,j})_{i,j\in[n]}$ such that $c_{i,j} = \min_{k\in[n]}\{a_{i,k} + b_{k,j}\}$ for every $i, j \in [n]$. Given a possibly weighted n-vertex digraph G, we consider the matrix $W = (w_{i,j})_{i,j\in[n]}$ such that $w_{i,j}$ denotes the weight (or length) of the edge from i to j, whereas $w_{i,i} = 0$ and $w_{i,j} = \infty$ if there is no edge from i to j. Then, the shortest paths in G can be read from A^n, and the foregoing NP-proof consists of the prover sending the matrices $A_1, A_2, ..., A_{\lceil \log_2 n\rceil}$ such that $A_0 = A$ and $A_i = A_{i-1} * A_{i-1}$ for all i. Hence, the verification of APSP is reduced to the verification of $\log n$ claims regarding matrix multiplication in the $(\min,+)$-algebra, which can be verified in parallel. Focusing on the latter problem, or rather on the set $\{(A, B, A * B) : A, B \in \bigcup_{n\in\mathbb{N}} \mathbb{R}^{n\times n}\}$, we observe that membership can be recognized in $\mathcal{SC}$ (hence the result of [16] applies) as well as by highly uniform $\mathcal{AC}^0$ circuits.

Corollary 2 (a constant-round doubly-efficient interactive proof for APSP): *Let APSP consists of pairs (G, L) such that L is a matrix recoding the lengths of the shortest paths between each pair of vertices in the weighted graph G. For every constant $\delta > 0$, the APSP has a $O(1/\delta)$-round interactive proof system in which the verifier runs in time $O(n^{2+o(1)})$, the prescribed prover can be implemented in time $O(n^{4+o(1)})$, where n denotes the number of vertices in the graph and weights are restricted to $[-\exp(n^{o(1)}, \exp(n^{o(1)}]$. Furthermore, except for the first prover message, in each subsequent round, the prover sends n^δ bits.*

[1] Actually, the result of [16] can be applied to $\mathcal{NC}^1$ circuits that can be constructed in polynomial time and $n^{o(1)}$-space.

As with Theorem 1, the application of [16] to APSP would have yielded $\exp(\widetilde{O}(1/\delta))$ rounds.

Another problem to which Theorem 1 is applicable is set of graphs having no t-cliques, denoted t-no-CLIQUE. For any contant t, constant-round doubly-efficient interactive proof systems for t-no-CLIQUE are implicit or explicit in several prior works. In particular, such proof systems are implied by the aforementioned result of [16] as well as by [5, Sec. 4.3], and were explicitly presented in [7, Sec. 2]. Noting that the said set can be recognized by highly uniform CNFs of size $O(n^t)$ and using Theorem 1, we obtain yet another alternative proof system for t-no-CLIQUE.

Corollary 3 (a constant-round doubly-efficient interactive proof for t-no-CLIQUE): *For every constants $t \in \mathbb{N}$ and $\delta > 0$, the set t-no-CLIQUE has a $O(t/\delta)$-round interactive proof system in which the verifier runs in time $O(n^{2+o(1)})$, the prescribed prover can be implemented in time $O(n^{t+o(1)})$, and the total communication is n^δ, where n denotes the number of vertices in the graph.*

In Table 1.1, we compare Corollary 3 to the prior proof systems known for the t-no-CLIQUE problem.

obtained (in)	# rounds	total comm.	verifier time	prover time
via $\mathcal{SC}$ [16]	$\exp(\widetilde{O}(1/\delta))$	n^δ	$\widetilde{O}(m)$	poly(n^t)
via "local characterization" [5]	t/δ	n^δ	$\widetilde{O}(m)$	n^{t+1}
directly [7]	t	$\widetilde{O}(n)$	$\widetilde{O}(m)$	$n^{0.791t}$
via $\mathcal{AC}^0[2]$ (this work)	$O(t/\delta)$	$n^{\delta+o(1)}$	$n^{2+o(1)}$	$n^{t+o(1)}$

Table 1: Comparison of different constant-round interactive proof systems for the t-no-CLIQUE problem, for the constants t and $\delta > 0$, where n (resp., $m > n$) denotes the number of vertices (resp., edges).

Our proof system for t-no-CLIQUE is very similar to the one in [5]. The difference is that we apply the sum-check protocol to an arithmetic circuit defined over an extension field (of size $n^{2\delta}$) of GF(2), whereas in [5] it is implicitely applied to an arithmetic circuit defined over a field of prime characteristic that is larger than $\binom{n}{t}$. Furthermore, here the arithmetic circuit is a pseudorandom linear combination of the $\binom{n}{t}$ tiny circuits that identify specific t-cliques, whereas in [5] the arithmetic circuit counts these t-cliques.

1.2 Notions of sufficiently uniform circuits

Some notion of uniformity is essential for a result such as Theorem 1, since the claim regarding the input x refers to satisfying a poly$(|x|)$-sized circuit $C_{|x|}$, whereas the verifier is restricted to almost-linear time. In the context of this work, we seek the most liberal notion that we can support.

Our notion of uniformity is stronger than the notion of log-space uniformity used in [8] (let alone even weaker notions of uniformity that can be supported when applying the result of Reingold, Rothblum, and Rothblum [16] (see Footnote 1)). Specifically, we consider the complexity of a succinct (implicit) representation of the circuit, rather than the complexity of constructing the circuit itself (i.e., its explicit representation). We consider three such succinct representations, where in all cases we assume that the circuits are *layered* (in the sense detailed below):

Adjacency predicate: Such a predicate indicates, for each pair of gates (u, v), whether or not gate u is fed by gate v. Specifically, dealing with circuits of size $s(n) = \text{poly}(n)$, we consider the adjacency predicate $\texttt{adj} : [s(n)] \times [s(n)] \to \{0, 1\}$,

Incidence function: Such a function indicates, for each gate u and index i, the identity of the i^{th} gate that feeds the gate u, where 0 indicates that u is fed by less than i gates. Specifically, for fan-in bound $b(n) \leq s(n) - 1$, we consider the incidence function $\texttt{inc} : [s(n)] \times [s(n)] \to [b(n)] \cup \{0\}$.

Input assignment in canonical formulae: Here we consider a fixed structure of the circuit as a formula, and specify only the input bit assigned to each leaf of the formula, where the same bit is typically assigned to many leaves. The assignment is merely a function from leaf names to bit locations in the input. Specifically, we consider the input assignment function $\texttt{ia} : [s(n)] \to [n] \cup \{0, n+1\}$, where 0 and 1 are viewed as residing in locations 0 and $n + 1$ of the "augmented" n-bit input (which holds $n + 2$ bits).

In all cases, we assume that the (depth d) circuit is layered in the sense that, for each $i \in [d]$, gates at layer $i-1$ are fed by gates at layer i only, where layer i consists of all gates at distance i from the output gate. Indeed, the output gate is the only gate at layer 0, and the gates at layer d are called leaves, since they are not fed by gates but are rather assigned input bits.[2] (Indeed, for simplicity, we do not allow leaves at other layers.)[3] Furthermore, when using the adjacency predicate and the incidence function representations, we shall assume that the i^{th} leaf is assigned the i^{th} input bit; but in the canonical formulae representation the assignment of input bits to leaves is the only aspects of the circuit that varies.

In all three cases, we make two additional simplifying assumptions. The first is that the circuit contains no "negation" gates (i.e., `not`-gates). This can be assumed, without loss of generality, because we can replace `not`-gates by `parity`-gates (fed by the desired gate and the constant 1, which is the reason for allowing to feed leaves with the constant 1). The second assumption is that, *for each $i \in [d]$, all gates at layer $i - 1$ have the same functionality* (gate-type).[4]

Theorem 1 holds under each of the three representations, when requiring that the corresponding function, which implicitly describes the $\text{poly}(n)$-size circuit C_n, can be represented by a formula of size $n^{o(1)}$ than can be constructed in time $n^{1+o(1)}$. (Recall that the input to the latter formula is of length $O(\log n)$.)

1.3 Overview of our main construction

The construction underlying Theorem 1 combines a central ingredient of the interactive proof system of Goldwasser, Kalai, and Rothblum [8] with the approximation method of Razborov [15] and Smolensky [18]. Specifically, we first reduce the verification of the claim that the input satisfies the predetermined Boolean circuit to an analogous claim regarding an Arithmetic circuit (over GF(2)) that is derived from the Boolean circuit using the approximation method. The crucial fact

[2] We stress that the term 'leaf' is used here also in the case that the circuit is not a formula (i.e., does not have a tree structure). One may prefer using the terms 'terminal' or 'source' instead.

[3] This can be assumed, without loss of generality, by replacing such a potential leaf at layer i with an auxiliary path of dummy gates that goes from layer d to layer i so that it indirectly feeds the value of the desired input bit to the corressponding gate at layer i.

[4] This can be assumed, without loss of generality, by replacing each layer by three consecutive layers so that one layer is devoted to `and`-gates, one to `or`-gates, and one to `parity`-gates.

is that that *all multiplication gates in the Arithmetic circuit have small fan-in* (whereas the fan-in of addition gates may be large). With high probability, this approximation does not affect the computation on the given input, but it does introduces a "completeness error" in the verification procedure, which we eliminate later (so to obtained perfect completeness).

Next, following [8], we consider a computation of the Arithmetic circuit (on the given input), and encode the values of the gates at each layer by a low degree polynomial over a large (extension) field (of GF(2)). Here we use the fact that, by virtue of the approximation method, the gates in the Arithmetic circuit compute polynomials of low degree, whereas in [8] obtaining low degree polynomials relied on "refreshing the variables" after each layer of the circuit (see also Eq. (11) in Section 3). That is, unlike in [8], we do not use a generic low-degree extension of the Boolean values (computed by the gates of the Boolean circuit), but rather use the polynomials that are computed by the gates of the Arithmetic circuit (i.e., the formal polynomials that are defined by the circuit). More importantly (and in fact crucially), relying on the foregoing uniformity condition, *we express the relation between the values of the gates at adjacent layers* (of the circuit) *by low degree polynomials*. These polynomials are derived from the small Boolean formulas that compute the adjacency relation.

Lastly, following [8], we reduce the verification of a claim regarding the values at layer $i-1$ in the circuit to a claim regarding the values at layer i, by using the Sum-Check protocol in each reduction step. Specifically, we use the Sum-Check protocol with respect to variables in a relatively large alphabet (of size n^{δ}), so that the number of rounds is a constant (i.e., $O(1/\delta)$). Actually, this refers to the way in which addition gates of unbounded fan-in are handled, where each such poly(n)-way addition is written as a sum over a $O(1/\delta)$-long sequence over an alphabet of size n^{δ}. In contrast, multiplication gates, which are of logarithmically bounded fan-in, are treated in a straightforward manner (i.e., we branch to verify each of the logarithmically many claimed values).[5]

To summarize: Using the approximation method allows us to replace `or`-gates (and/or `and`-gates) of unbounded fan-in by multiplication gates of logarithmic fan-in, while introducing parity gates of unbounded fan-in. Each layer of parity gates can be handled by the Sum-Check protocol such that each iteration of this protocol cuts the fan-in of the parity gates by a factor of n^{δ}. The degree bound on which the Sum-Check protocol relies is due to the uniformity of the original Boolean circuit and to the fact that the multiplication gates have small fan-in. Specifically, a sufficient level of uniformity of the Boolean circuit implies an upper bound on the degree of the polynomials that relate the values of the gates at adjacent layers (of the circuit), wherae the small fan-in of the multiplication gates implies an upper bound on the degree of the polynomial that expresses the values of the various gates.

We mention that the idea of using the approximation method towards emulating $\mathcal{AC}^0[2]$ by low degree arithmetic circuits, *in the context of interactive proof systems*, was used before by Kalai and Raz [11]. Both in [11] and here, this causes a (small) error probability (in the completeness condition).[6] We regain perfect completeness by letting the prover point out a gate in which the approximation error occurs (with respect to the input), and prove its claim.

[5] Indeed, we could reduce the verification of these logarithmically many claims to the verification of a single claim, by using a curve that passes through all the point in these claims, as done in [8]. But since here the number of rounds is a constant, we can afford an overhead that is exponential in the number of rounds.

[6] In contrast, when using the approximation method in the context of worst-case to average-case reduction for the class $\mathcal{AC}^0[2]$ presented in [6, Apdx A.2], the approximation error is absorbed by the (larger) error rate of the average-case solver.

1.4 The proof system for $\mathcal{NC}^1$

Generalizing and somewhat simplifying the proof systems constructed by Goldwasser, Kalai, and Rothblum [8], we obtain constant-round doubly-efficient interactive proof systems for sufficiently uniform $\mathcal{NC}^1$ (specifically, canonical formulas with a sufficiently uniform input assignment function as discussed in Section 1.2). The simplification is due to relying on a stronger notion of uniformity than the one used in [8], whereas the generalization allows us to reduce the round complexity of [8] by a log-squared factor. Recall that, when handling a (bounded fan-in) circuit $C_n : \{0,1\}^n \to \{0,1\}$ of depth $d(n)$, the proof system of [8] has $O(d(n) \cdot \log n)$ rounds. This is due to invoking the Sum-Check Protocol for each layer in the circuit, and using a version that handles summations over the binary alphabet. Instead, for any constant $\delta > 0$, we invoke the Sum-Check protocol for each block of $\delta \log n$ consecutive layers in the circuit, and use a version that handles summations over an alphabet of size n^δ. Hence, we cut the number of rounds by a factor of $(\delta \log n)^2$.

Theorem 4 (constant-round doubly-efficient interactive proofs for $\mathcal{NC}^1$, loosely stated): *Let $\{C_n : \{0,1\}^n \to \{0,1\}\}$ be a sufficiently uniform family of canonical Boolean circuits of fan-in two and logarithmic depth. Then, for every $\delta \in (0,1]$, the set $\{x : C_{|x|}(x) = 1\}$ has a $O(1/\delta^2)$-round interactive proof system in which the verifier runs in time $O(n^{1+o(1)})$, the prescribed prover can be implemented in polynomial-time, and the total communication is $n^{\delta+o(1)}$.*

We stress that Theorem 4 does not subsume Theorem 1. First, the proof system in Theorem 4 uses a larger number of rounds as a function of total communication complexity (i.e., $O(1/\delta^2)$ rather than $O(1/\delta)$ rounds). Second, the uniformity condition in Theorem 4 is stronger (cf., Theorem 10 and Theorem 7).

2 The interactive proof system for $\mathcal{AC}^0[2]$

Recall that we consider a sufficiently uniform family of layered circuits $\{C_n\}$ of constant depth $d \in \mathbb{N}$ and unbounded fan-in. For simplicity of our presentation, we work with the adjacency predicate representation, while noting that the handling of other representations can be reduced to it (as detailed in Remarks 8 and 9). We also assume, for simplicity, that the circuit has only gates of the `or` and `parity` type, since `and`-gates can be emulated by these. Letting $s = s(n) = \text{poly}(n)$ be a bound on the number of gates in C_n, for each $i \in [d]$, we consider the $n^{o(1)}$-sized formula $\psi_i : [s] \times [s] \to \{0,1\}$ such that $\psi_i(j,k) = 1$ if and only if gate j resides in layer $i-1$ and is fed by the gate k (which resides in layer i). In doing so, we associate $[s]$ with $\{0,1\}^\ell$, where $\ell = \log s$, and view ψ_i as a function over $\{0,1\}^{2\ell}$.

On input $x \in \{0,1\}^n$, we proceed in three steps: First, we reduce the Boolean problem (of verifying that $C_n(x) = 1$) to an Arithmetic problem (of verifying that a related Arithmetic circuit A_n evaluates to 1 on a related input (x,s)). Next, we express the latter problem as a sequence of $O(d)$ equations that relate the value of gates at adjacent layers of the circuit A_n. Here we shall use low degree polynomials that extend the ψ_i's, while deriving succinct representations of these polynomials from the corresponding Boolean formulas that compute the ψ_i's. Last, we present a constant-round doubly-efficient interactive proof system for the verification of each equation. Hence, we obtain a constant-round doubly-efficient interactive proof system for the set $\{x : C_{|x|}(x) = 1\}$.

Step 1: Approximation by Arithmetic circuits. The first step is a randomized reduction of solving the Boolean problem to solving a corresponding Arithmetic problem. This reduction follows the ideas underlying the approximation method of Razborov [15] and Smolensky [18], while working with the field GF(2) (as [15], rather than with GF(p) for some prime $p > 2$ as [18]). When doing so, we replace the random choices made at each layer by pseudorandom choices that are generated by a small bias generator [14]; specifically, we use a highly uniform generator that uses a seed of logarithmic (i.e., $O(\log n)$) length and produces each output bit by a small (i.e., $n^{o(1)}$-size) circuit of constant depth [1, 10].[7] Hence, for a fixed Boolean circuit C_n, on input $x \in \{0,1\}^n$, we randomly reduce the question of whether $C_n(x) = 1$ to the question of whether $A_n(x, y) = 1$, where A_n is the resulting Arithmetic circuit and $y \in \{0,1\}^{n^\delta}$ represents a uniformly distributed sequence of seeds for the said generator (i.e., logarithmically different seeds are used for each layer, and the same seeds are used for all gates of that layer). Specifically, the choice of y will be made by the verifier, and we observe that $\Pr_y[A_n(x,y) = C_n(x)] = 1 - s(n) \cdot \exp(-O(\log n)) = 1 - o(1)$. In the rest of the analysis, we assume that the verifier was not extremely unlucky (in its choice of y), and so that $A_n(x, y) = C_n(x)$ holds.

Let us stop for a moment and take a closer look at A_n. Recall that each `or`-gate in C_n is essentially replaced by a $O(\log n)$-way multiplication gate that is fed by the inner product of the values of the original feeding gates and a pseudorandom sequence, where each element in the latter sequence is a simple function of y (computed by a small constant-depth circuit). Specifically, if the `or`-gate indexed j (at layer $i - 1$ of C_n) was fed by gates indexed $k_1, ..., k_{n'}$ (of layer i), then it is replaced in A_n by the formula

$$1 + \prod_{j' \in [\ell']} \left(1 + \sum_{k \in [s]} \psi_i(j,k) \cdot g_k \cdot p_k^{(i,j')}\right) \quad \text{which equals} \quad 1 + \prod_{j' \in [\ell']} \left(1 + \sum_{t \in [n']} g_{k_t} \cdot p_{k_k}^{(i,j')}\right)$$

where $\ell' = O(\log n)$ is the arity of the multiplication gate, g_k is the output of the gate indexed k (at layer i of C_n), and $p_k^{(i,j')}$ is the $k^{\rm th}$ bit in the pseudorandom sequence generated based on the $j'^{\rm th}$ seed of layer i (which is a part of y).[8] Note that if $\bigvee_{t \in [n']} g_{k_t} = 0$, then the forgoing expression simplifies to $1 + (1 + 0)^{\ell'} = 0$. On the other hand, if $\bigvee_{t \in [n']} g_{k_t} = 0$, then, with probability at least $1 - 0.6^{\ell'}$, the forgoing expression simplifies to $1 + 0 = 1$. Hence, the result of the ℓ'-way multiplication gate is 1 if $\bigvee_{t \in [n']} g_{k_t} = 0$ and is likely to be 0 otherwise, and so we will not use the output of the foregoing multiplication gate but rather its "negation" (i.e., its value plus 1). To summarize, the multiplication gate indexed j (in the corresponding layer of A_n) is fed by gates indexed $\langle j, 1\rangle, ..., \langle j, \ell'\rangle$, and the gate indexed $\langle j, j'\rangle$ computes $1 + \sum_{k \in [s]} \psi_i(j,k) \cdot g_k \cdot p_k^{(i,j')}$.

We highlight the following features of the Arithmetic circuit A_n: Its depth is $O(d)$, its size is $O(\log n)^d \cdot s(n)$, it computes a polynomial of degree $O(\log n)^d$, and it has a succinct representation of size $n^{o(1)}$ that can be constructed in time $n^{1+o(1)}$. Furthermore, each of its gates computes a polynomial of degree $O(\log n)^d$.

Indeed, $A_n : \mathrm{GF}(2)^{n+n^\delta} \to \mathrm{GF}(2)$ as defined above is an arithmetic circuit, over GF(2), consisting solely of addition and multiplication gates. But we can view A_n as an arithmetic circuit over

[7] Using the third constriction in [1], we need to perform exponentiation in a field of size 2^ℓ, where $\ell = O(\log n)$ is half the length of the seed. Using [10], this operation can be performed by highly uniform constant-depth circuits (with parity gates) and size $\exp(\widetilde{O}(\sqrt{\ell})) = n^{o(1)}$. The same pseudorandom sequences can be used for all gates in the same layer of the circuit, since logarithmically many independent choices guarantee sufficiently small error probability (allowing the application of a union bound).

[8] Indeed, we use ℓ' different seeds per each layer, and use each of these seeds for all gates of the layer. Furthermore, we use the same pseudorandom bits for all gates in the layer (although we could afford using disjoint parts of the same pseudorandom sequence for different gates).

$\mathcal{F} = \mathrm{GF}(2^{2\delta\log n})$, and consider its value at $(x,y) \in \{0,1\}^{n+n^\delta}$, which is viewed as an $(n+n^\delta)$-long sequence over $\mathcal{F}$. (It suffices to have $|\mathcal{F}| \geq n^{\delta+\Omega(1)}$; on the other hand, we also use $\log|\mathcal{F}| \leq n^{o(1)}$.)

Step 2: Relating the values of layers in the computation. Letting $\alpha_{i-1}(j)$ denote the value of gate j at layer $i-1$, in a computation of $A_n(x,y)$, we can relate these values as already hinted above. Specifically, in case $j \in [s]$ is an ℓ'-way multiplication gate, we have $\alpha_{i-1}(j) = \prod_{j' \in [\ell']} \alpha_i(\langle j,j'\rangle)$ and

$$\alpha_i(\langle j,j'\rangle) = 1 + \sum_{k\in[s]} \psi_i(j,k)\cdot\alpha_{i+1}(k)\cdot\alpha_{i+1}(j's+k), \tag{1}$$

where $\alpha_{i+1}(j's+k)$ holds the value of the k^{th} bit of the j'^{th} pseudorandom sequence. (For sake of simplicity, our presentation merges the layer that computes the s-way sum with the layer that computes the quadratic forms $\alpha_{i+1}(k)\cdot\alpha_{i+1}(j's+k)$; that is, Eq. (1) refers to a fictitious bilinear gate. In addition, our description ignores the small circuits that compute the pseudorandom bits $\alpha_{i+1}(j's+k)$.)[9] In the case of an addition gate (i.e., a layer of addition gates), we have

$$\alpha_{i-1}(j) = \sum_{k\in[s]} \psi_i(j,k)\cdot\alpha_i(k). \tag{2}$$

Recall that we actually consider $j,k \in \{0,1\}^\ell$, where $\ell = \log s$. (Actually, we should let $\ell' = \log((\ell'+1)\cdot s)$, in order to account also for the gates computing the bits of the pseudorandom sequences.) Furthermore, we consider the arithmetic formula $\widehat{\psi}_i : \mathcal{F}^{\ell+\ell} \to \mathcal{F}$ that is derived from $\psi_i : \{0,1\}^{\ell+\ell} \to \{0,1\}$ in the obvious manner (i.e., replacing and-gates by multiplication gates and negation gates by gates that add the constant 1). Recalling that ψ_i is a formula of size $n^{o(1)}$, it follows that $\widehat{\psi}_i$ computes a polynomial of degree $n^{o(1)}$. Now, Eq. (2) is replaced by

$$\alpha_{i-1}(j) = \sum_{k\in\{0,1\}^\ell} \widehat{\psi}_i(j,k)\cdot\alpha_i(k), \tag{3}$$

where $j \in \{0,1\}^\ell$, and Eq. (1) is replaced analogously.

At this point, the standard approach taken in [8] (and followed also in Section 3) is to extend Eq. (3) to any $j \in \mathcal{F}^\ell$ by using a low-degree extension (see also Eq. (5)-(6)). This is redundent in the case of Eq. (3), since the r.h.s of this equation is well defined also when $j \in \mathcal{F}^\ell$. Hence, for $z \in \mathcal{F}^\ell$, we have

$$\widehat{\alpha}_{i-1}(z) = \sum_{k\in\{0,1\}^\ell} \widehat{\psi}_i(z,k)\cdot\alpha_i(k). \tag{4}$$

Assuming that α_i is also extended to a low-degree polynomial, denoted $\widehat{\alpha}_i$, we can replace $\alpha_i(k)$ by $\widehat{\alpha}_i(k)$. In the case of Eq. (1), the foregoing consideration applies to $j \in \{0,1\}^\ell$, but not to $j' \in [\ell']$. Applying a low-degree extension to the dependence of the expression on j', we have, for $z \in \mathcal{F}^\ell$ and $z' \in \mathcal{F}$,

$$\widehat{\alpha}_i(\langle z,z'\rangle) = 1 + \sum_{k\in[s]} \widehat{\psi}_i(z,k)\cdot\alpha_{i+1}(k)\cdot\sum_{j'\in[\ell']}\left(\prod_{j''\in[\ell']\setminus\{j'\}}\frac{j''-z'}{j''-j'}\right)\cdot\alpha_{i+1}(j's+k), \tag{5}$$

[9] The issue is not describing these small circuits, since this was addressed in Footnote 7, but rather presenting a single equation that interpolates both their operation and the bulk of the computation. For example, Eq. (2)-(4) refer to the value of $\alpha_{i-1}(j)$ for $j \in [s] \equiv \{0,1\}^\ell$, whereas for $j \in [(\ell'+1)\cdot s]\setminus[s]$ the value of $\alpha_{i-1}(j)$ refers to this auxiliary computations. That is, $\alpha_{i-1}(j)$ is defined in different ways depending on whether or not $j \in [s]$, and these different formulas are combined by a selection circuit (actually, a single gate that check a relevant bit in the binary expansion of j will do).

where the product equals 1 if $z' = j'$ and equals 0 if $z' \in [\ell'] \setminus \{j'\}$. (As with Eq. (4), assuming that α_{i+1} is also extended to a low-degree polynomial, denoted $\widehat{\alpha}_{i+1}$, we can replace $\alpha_{i+1}(j's+k)$ by $\widehat{\alpha}_{i+1}(j's+k)$.) Lastly, in the case of $\alpha_{i-1}(j) = \prod_{j' \in [\ell']} \alpha_i(\langle j, j' \rangle)$, we can just replace both α_{i-1} and α_i by the corresponding polynomials $\widehat{\alpha}_{i-1}$ and $\widehat{\alpha}_i$, obtaining $\widehat{\alpha}_{i-1}(z) = \prod_{j' \in [\ell']} \widehat{\alpha}_i(\langle z, j' \rangle)$, and infer that $\widehat{\alpha}_{i-1}$ has low degree since so does $\widehat{\alpha}_i$.

Note, however, that the foregoing can not be applied to α_d, which is well-defined only over $[s] \equiv \{0,1\}^\ell$; specifically, recall that $\alpha_d(j) = x_j$ for $j \in [n]$, whereas $\alpha_d(n+1) = 1$ and $\alpha_d(j) = 0$ for $j \in [s] \setminus [n+1]$.[10] Hence, we augment the foregoing definitions by postulating that $\widehat{\alpha}_d$ is a low-degree extension of the values of α_d at $\{0,1\}^\ell$; that is, for $z \in \mathcal{F}^\ell$, we have

$$\widehat{\alpha}_d(z) = \sum_{k \in \{0,1\}^\ell} \mathtt{EQ}(z,k) \cdot \alpha_d(k), \tag{6}$$

where EQ is the bilinear polynomial that extends the function that tests equality over $\{0,1\}^\ell$ (e.g., $\mathtt{EQ}(\sigma_1 \cdots \sigma_\ell, \tau_1 \cdots \tau_\ell) = \prod_{i \in [\ell]} (\sigma_i \tau_i + (1-\sigma_i)(1-\tau_i)))$. Note that r.h.s of Eq. (6) depends only on $n+1$ terms, since $\widehat{\alpha}_d(j) = 0$ for $j \notin [n+1]$.

Lastly, we wish to replace summation over $\{0,1\}^\ell$ by summation over H^m, where $|H| = n^\delta$ (and $m = \frac{\ell}{\delta \log n} = O(1/\delta)$). Towards this end, we introduce a mapping $\mu : H \to \{0,1\}^{\ell''}$, where $\ell'' = \delta \log n = \ell/m$. This mapping can be extended to a sequence of polynomials of degree $|H|-1$ over $\mathcal{F}$, and to an m-long sequence of such sequences; that is, we use $\widehat{\mu} : \mathcal{F}^m \to (\mathcal{F}^{\ell''})^m$ such that for every $v = (v_1, ..., v_m) \in H^m$ it holds that $\widehat{\mu}(v) = (\mu(v_1), ..., \mu(v_m)) \in \{0,1\}^{m \cdot \ell''}$, whereas $m\ell'' = \ell$. Hence, for example, Eq. (4) is replaced by

$$\widehat{\alpha}_{i-1}(z) = \sum_{k \in H^m} \widehat{\psi}_i(z, \widehat{\mu}(k)) \cdot \widehat{\alpha}_i(\widehat{\mu}(k)) \tag{7}$$

where again $z \in \mathcal{F}^\ell$. (Recall that $\widehat{\alpha}_i$ is defined over $\mathcal{F}^\ell$ by virtue of subsequent expressions of similar type.) With these preliminaries in place, we can describe the basic interactive proof system.

Step 3: Interactive proof system (with imperfect completeness). On input $x \in \{0,1\}^n$, the verifier selects uniformly $y \in \{0,1\}^{n^\delta}$, and sends y to the prover. The prover now attempts to prove that $A_n(x,y) = 1$, where a succinct representation of A_n (of size $n^{o(1)}$) can be constructed in time $n^{1+o(1)}$. The initial claim is re-interpreted as $\widehat{\alpha}_0(1^\ell) = 1$, where $\widehat{\alpha}_i : \mathcal{F}^\ell \to \mathcal{F}$. The parties proceed in $O(d)$ steps such that the i^{th} step starts with a claim regarding the value of $\widehat{\alpha}_{i-1}$ at few points, and ends with a claim regarding the value of $\widehat{\alpha}_i$ at few points, where the said number of points may increase by at most a logarithmic factor. We distinguish between the case that the current layer is of addition gates and the case that it is of multiplication gates.

Handling a layer of addition gates: (Recall that these gates are supposed to satisfy Eq. (7).) For each claim of the form $\widehat{\alpha}_{i-1}(j) = v$, where $i \in [O(d)]$, $j \in \mathcal{F}^\ell$ and $v \in \mathcal{F}$ are known, we invoke the Sum-Check protocol on the r.h.s of Eq. (7). The execution of the (m-round) protocol results in a claim regarding the value $\widehat{\psi}_i(j, \widehat{\mu}(r)) \cdot \widehat{\alpha}_i(\widehat{\mu}(r))$ for a random $r \in \mathcal{F}^m$ selected via

[10] Recall that we need to provide the circuit with the constant 1, hence we set $\alpha_d(n+1) = 1$. The setting of $\alpha_d(j) = 0$ for $j \in [s] \setminus [n+1]$ is used in order to facilitate the evaluation of r.h.s of Eq. (6), as discussed below. Alternatively, we could have used the setting $\alpha_d(j) = 1$ for $j \in [s] \setminus [n]$, and rely on the fact that for every $j = (j_1, ..., j_\ell) \in \mathcal{F}^\ell$ the sum $\sum_{k \in \{0,1\}^\ell} \mathtt{EQ}(j,k)$ equals 1.

the execution. Since the verifier can evaluate $\widehat{\mu}$ and $\widehat{\psi}_i$, it is left with a claim regarding the value of $\widehat{\alpha}_i$ at one point.

Recall that the Sum-Check protocol proceeds in $m = O(1/\delta)$ rounds, where in each round the prover sends the value of the relevant univariate polynomial. This is a polynomial of degree $|H| \cdot n^{o(1)} = n^{\delta+o(1)}$, where the degree bound is due to the composition of the polynomials $\widehat{\mu}$ and $\widehat{\psi}_i$ (and likewise of $\widehat{\mu}$ and $\widehat{\alpha}_i$).[11]

Handling a layer of multiplication gates: For each claim of the form $\widehat{\alpha}_{i-1}(j) = v$, where i, j and v are known, we let the prover send the values $\widehat{\alpha}_i(\langle j, 1\rangle), ..., \widehat{\alpha}_i(\langle j, \ell'\rangle)$. The verifier checks that their product equals v, and obtains $\ell' = O(\log n)$ claims to be verified next (i.e., claims regarding the values $\widehat{\alpha}_i(\langle j, j'\rangle)$ for $j' \in [\ell']$).

The handling of "bilinear gates" (or rather the computation of $\alpha_{i+1}(k) \cdot \alpha_{i+1}(j's + k)$ that takes place in Eq. (1)) is similar, except that here we have only two factors.[12]

After $O(d)$ steps, the verifier is left with polylogarithmically (i.e., $O(\log n)^d$) many claims, where each claim refers to the value of $\widehat{\alpha}_d$ at a single point $j \in \mathcal{F}^\ell$. Such a claim can be checked by the verifier itself by using the revised form of Eq. (6), which reads

$$\widehat{\alpha}_d(z) = \sum_{k \in H^m} \mathtt{EQ}(z, \widehat{\mu}(k)) \cdot \alpha_d(\widehat{\mu}(k)). \tag{8}$$

Recall that $\widehat{\alpha}_d(k) = x_k$ for every $k \in [n]$, $\widehat{\alpha}_d(n+1) = 1$, and $\widehat{\alpha}_d(k) = 0$ for every $k \in [s] \setminus [n+1]$. Hence, per each point $j \in \mathcal{F}^\ell$, computing $\widehat{\alpha}_d(j)$ reduces to evaluating $\mathtt{EQ}(j, \widehat{\mu}(k)) \cdot \widehat{\alpha}_d(\widehat{\mu}(k))$ at $n+1$ points (only); that is, letting I denote the subset of $H^m \equiv [2^\ell]$ that correspond to $[n+1]$, the verifier just computes $\sum_{k \in I} \mathtt{EQ}(j, \widehat{\mu}(k)) \cdot \widehat{\alpha}_d(\widehat{\mu}(k))$.

Analysis of the forgoing interactive proof system. We first observe that the complexities of the foregoing protocol are as stated in Theorem 1. Specifically, the protocol proceeds in $O(d)$ steps and in each step a Sum-Check protocol is invoked on a sum that ranges over H^m, where $m = O(1/\delta)$ and $|H| = n^\delta$. Since the relevant polynomial is of degree $n^{\delta+o(1)}$, the total (m-round) communication is of this order.[13] The verification time is dominated by the final check (i.e., evaluating $\widehat{\alpha}_d$ on few points), which runs in time $O(n^{1+o(1)})$. The complexity of the prescribed prover is dominated by its operation in the Sum-Check protocol, which can be implemented in time $|H|^m \cdot n^{o(1)} = s(n)^{1+o(1)}$. Next, we show that this protocol constitutes an interactive proof system (with imprefect completeness) for $\{x : C_{|x|}(x) = 1\}$.

Claim 5 (imperfect completeness): *If $C_n(x) = 1$ and the prover follows the presecribed strategy, then the verifier accepts with probability $1 - o(1)$.*

Proof: The probability that the value of an $\mathtt{or}$-gate, under a fixed setting of its input wires, is correctly emulated by the multiplication of t random affine combinations of these wires is at least $1-$

[11] Note that the degree of $\widehat{\psi}_i$ is upper bounded by $n^{o(1)}$, since it is obtained by arithmetizing the formula ψ_i which has size $n^{o(1)}$, whereas $\widehat{\mu}$ has degree $|H| - 1$. Likewise, the composition of $\widehat{\mu}$ and $\widehat{\alpha}_i$ is a polynomial of degree $\mathrm{poly}(\log n) \cdot |H| = \widetilde{O}(n^\delta)$, since the polynomial $\widehat{\alpha}_i$ has degree $O(\log n)^2$.

[12] Indeed, Eq. (5), which extends Eq. (1), is not used. Recall that Eq. (5) actually merges two layers: A layer of addition gates that performs the s-way sums, which is handled as in the previous item, and a layer of two-way multiplication gates that is handled as a layer of ℓ'-way multiplication gates.

[13] This dominates the length of the initial verifier-message $y \in \{0,1\}^{n^\delta}$.

2^{-t}, where in case all wires feed 0 the emulation is always correct.[14] The same holds (approximately) when the random linear combinations are replaced by inner products with a small biased sequence; specifically, if the sequence is 0.1-biased, then the emulation is correct with probability at least $1-(0.5+0.1)^t$. Using $t = 2\log s(n) = O(\log n)$ and employing a union bound, it follows that with probability $1-o(1)$ over the choice of $y \in \{0,1\}^{n^\delta}$, it holds that $A_n(x,y) = C_n(x)$. Observing that the verifier always accepts when $A_n(x,y) = 1$ (and the prover follows the prescribed strategy), the claim follows. ■

Claim 6 (soundness): *If $C_n(x) = 0$, then, no matter what strategy the prover employs, the verifier accepts with probability at most $o(1)$.*

Proof: As shown in the proof of Claim 5, with probability $1-o(1)$ it holds that $A_n(x,y) = C_n(x)$. Recalling that the soundness error of the Sum-Check protocol is proportional to the ratio of the degree of the polynomial over the size of the field, it follows that the prover can fool the verifier into accepting a wrong value of $A_n(x,y)$ with proability $O(dm) \cdot |H| \cdot n^{o(1)}/|\mathcal{F}| = n^{\delta+o(1)-2\delta} = o(1)$, where we used the setting $|\mathcal{F}| = n^{2\delta}$. The claim follows. ■

Getting rid of the completeness error. Claims 5 and 6 assert that the foregoing protocol constitutes a proof system for $\{x : C_{|x|}(x)=1\}$, but this proof system carries a completeness error (see Claim 5). Recalling that this error is only due to the (unlikely) case that $A_n(x,y) \neq C_n(x)$, the begging fix is to have the prover prove to the verifier that this case has occurred (with respect to the random string y chosen by the verifier). Specifically, the (unlikely) case that $A_n(x,y) \neq C_n(x)$ may occur only when at least one `or`-gate of C_n is badly emulated by A_n; that is, the value of this gate is 1 in the computation of $C_n(x)$ whereas the corresponding (negated multiplication) gate in $A_n(x,y)$ evaluates to 0. This means that at least one of the gates (in A_n) that correspond to the children of the `or`-gate in C_n evaluates to 1, whereas the gate (in A_n) that corresponds to the `or`-gate (of C_n) evaluates to 0. So all that the prover needs to do is point out these two gates in A_n, and prove that their values are as stated. Hence, we regain perfect completeness, whereas the soundness claim remains valid (since in order to cheat the prover has to prove a false claim regarding the value of a gate in A_n). Thus, we obtain:

Theorem 7 (Theorem 1, restated): *For constants $c, d \in \mathbb{N}$, let $\{C_n : \{0,1\}^n \to \{0,1\}\}$ be a family of layered Boolean circuits with unbounded fan-in conjunction and parity gates such that C_n has size at most n^c and depth d. Suppose that C_n can be described by an adjacency predicate that is computable by a $n^{o(1)}$-size formula that can be constructed in $n^{1+o(1)}$-time. Then, for every $\delta \in (0,1]$, the set $\{x : C_{|x|}(x)=1\}$ has a $O(cd/\delta)$-round interactive proof system of perfect completeness*

[14] Recall that we actually refer to the "negation" of the output of the multiplication gate (i.e., the value of the output plus 1). That is, for fixed $b \in \{0,1\}^w$, we consider the probability

$$p \stackrel{\rm def}{=} \Pr_{r^{(1)},...,r^{(t)} \in \{0,1\}^w}\left[1 + \prod_{i\in[t]}(1 + \langle b, r^{(i)}\rangle_2) = 1\right]$$

where $\langle b, r\rangle_2$ denotes the inner product mod 2 of b and r. In other words, p equals the probability that $\prod_{i\in[t]}(1 + \langle b, r^{(i)}\rangle_2) = 0$. Note that $\prod_{i\in[t]}(1 + \langle 0^w, r^{(i)}\rangle_2) = 1$ for every $r^{(1)}, ..., r^{(t)} \in \{0,1\}^w$, which implies that $p = 0$ when $b = 0^w$. On the other hand, for any $b \in \{0,1\}^w \setminus \{0^w\}$, it holds that $\Pr_{r\in\{0,1\}^w}[1 + \langle b, r\rangle_2 = 0] = 1/2$, which implies that $p = 1 - 2^{-t}$.

in which the verifier runs in time $O(n^{1+o(1)})$, the prescribed prover can be implemented in time $O(n^{c+o(1)})$, and the total communication is $n^{\delta+o(1)}$.

Note that the foregoing adjacency predicate refers to gates of C_n, which are identified by ℓ-bit long strings, where $\ell = O(\log n)$. Thus, the uniformity condition postulates that this predicate can be computed by a formula of size $\exp(o(\ell))$ (equiv., by a bounded fan-in circuit of depth $o(\ell)$) that can be constructed in time $\exp(\ell/O(1))$.

Remark 8 (from incidence functions to adjacency predicate): *The foregoing presentation refers to the adjacency predicates $\psi_i : \{0,1\}^{2\ell} \to \{0,1\}$, which were extended to $\widehat{\psi}_i : \mathcal{F}^{2\ell} \to \mathcal{F}$. Suppose that we are given, instead, incidence functions of the form $\phi_i : [s] \times [s] \to [s] \cup \{0\}$, which we view as $\phi_i : \{0,1\}^{2\ell} \to \{0,1\}^{\ell+1}$, where $[s] \equiv \{0,1\}^\ell \equiv \{1\sigma : \sigma \in \{0,1\}^\ell\}$ and $0 \equiv 0^{\ell+1}$. In such a case, we may simply replace $\widehat{\psi}_i(j,k)$ by $\sum_{p\in\{0,1\}^\ell} \mathtt{EQ}(\widehat{\phi}_i(j,p), 1k)$* (or actually by $\sum_{p\in H^m} \mathtt{EQ}(\widehat{\phi}_i(j, \widehat{\mu}(p)), 1k)$).[15] *This means that, in invocations of the Sum-Check protocol, the relevant summations are over H^{2m} rather than over H^m.*

Remark 9 (the case of canonical circuits): *In this case, the s-sized circuit of depth d has the form of a w-ary tree of depth d such that $w = s^{1/d}$, and the input assignment is represented by the function $\phi : \{0,1\}^\ell \to [n] \cup \{0, n+1\}$. Hence, we effectively refer to the adjacency predicate $\psi_i : [w]^d \times [w]^d \to \{0,1\}$ such that $\psi_i(j_1 \cdots j_d, k_1 \cdots k_d) = 1$ if and only if $j_1 \cdots j_{i-1} j_{i+1} \cdots j_d = k_1 \cdots k_{i-1} k_{i+1} \cdots k_d$* (or rather $\psi_i : [w]^{i-1} \times [w]^i \to \{0,1\}$ such that $\psi_i(j_1 \cdots j_{i-1}, k_1 \cdots k_i) = 1$ if and only if $j_1 \cdots j_{i-1} = k_1 \cdots k_{i-1}$).[16] *In addition, instead of Eq. (8), for $z \in \mathcal{F}^\ell$, we have*

$$\widehat{\alpha}_d(z) = \sum_{k\in H^m} \mathtt{EQ}(z, \widehat{\mu}(k)) \cdot \mathtt{I}(\widehat{\mu}(k)). \tag{9}$$

where $\mathtt{I}(z') \stackrel{\rm def}{=} \sum_{v\in[n+1]} \mathtt{EQ}(v, \widehat{\phi}(z')) \cdot x_v$ and $\widehat{\phi}$ is polynomial that is obtained by transforming the Boolean formula that computes ϕ to a corresponding arithmetic formula. (Hence, for $k' \in \{0,1\}^\ell$, it holds that $\mathtt{I}(k') = x_{\phi(k')}$, where $x_0 = 0$ and $x_{n+1} = 1$.)[17] *Recall that once the $O(d)$ iterations are completed, the verifier is left with the verification of polylogarithmically many claims, where each claim refers to the value of $\widehat{\alpha}_d$ at a single point $j \in \mathcal{F}^\ell$. Here we cannot afford having the verifier evaluating $\widehat{\alpha}_d$ at j by itself* (since $|H|^m = s$). *Instead, we instruct the parties to run the Sum-Check protocol on Eq. (9), and the verifier is left with a claim referring to the value of $\mathtt{EQ}(j, \widehat{\mu}(k)) \cdot \mathtt{I}(\widehat{\mu}(j))$ at a random point $k \in \mathcal{F}^m$, which can be done in $\widetilde{O}(n)$-time since $\mathtt{I}$ is the sum of $n+1$ easily computable terms.*

3 The interactive proof system for $\mathcal{NC}^1$

In this section we prove the following result.

Theorem 10 (Theorem 4, restated): *For a logarithmic function $d : \mathbb{N} \to \mathbb{N}$, let $\{C_n : \{0,1\}^n \to \{0,1\}\}$ be a family of canonical Boolean circuits of fan-in two and depth d. Suppose that the input*

[15] Recall that $k \in [s] \equiv \{1\sigma : \sigma \in \{0,1\}^\ell\}$, whereas 0 is encoded as $0^{\ell+1}$.

[16] Indeed, we can replace Eq. (2) by $\alpha_{i-1}(j_1 \cdots j_d) = \sum_{k_i\in[w]} \alpha_i(j_1 \cdots j_{i-1} k_i j_{i+1} \cdots j_d)$ (or rather by $\alpha_{i-1}(j_1 \cdots j_{i-1}) = \sum_{k_i\in[w]} \alpha_i(j_1 \cdots j_{i-1} k_i)$), and ditto for Eq. (5).

[17] Note that if $\phi(k') = 0 \in \{0,1\}^\ell \setminus [n+1]$, then $\mathtt{I}(k') = 0 = x_0$.

assignment of C_n can be computed by a $n^{o(1)}$-size formula that can be constructed in $n^{1+o(1)}$-time. Then, for every $\delta \in (0,1]$, the set $\{x : C_{|x|}(x) = 1\}$ has a $O(d(n)/\delta \log n)^2$-round interactive proof system of perfect completeness in which the verifier runs in time $O(n^{1+o(1)})$, the prescribed prover can be implemented in polynomial-time, and the total communication is $n^{\delta+o(1)}$.

We leave open the question of whether the round complexity can be reduced to $O(\delta^{-1} \cdot (d(n)/\log n)^2)$, meeting the bound in Theorem 7.

3.1 Overview

The construction generalizes and somewhat simplifies the proof systems constructed by Goldwasser, Kalai, and Rothblum [8]. The simplification is due to working with canonical circuits rather than with general (log-space) uniform circuits as in [8], whereas the generalization allows us to reduce the round complexity of [8] by a log-squared factor. Specifically, the canonical form of the circuit allows us to relate the values of layers in the circuit that are at distance $\delta \log n$ apart, whereas [8] relate values at adjacent layers (only). In addition, we use a version of the sum-check protocol that handles summations over an alphabet of size n^δ (rather than over the alphabet $\{0,1\}$).

Fixing a constant $\delta \in (0,1)$, let $\ell' = \delta \log n$. The core of the proof system asserted in Theorem 10 is an iterative process in which a claim about the values of the gates that are at layer $(i-1) \cdot \ell'$ is reduced to a claim about the values of the gates at layer $i \cdot \ell'$. We stress that each of these claims refers to the values of the polynomially many gates at a specific layer of the circuit $C_{|x|}$ during the computation on input x, but these poly$(|x|)$ values are not communicated explicitly but rather only referred to. Nevertheless, in $t = d(|x|)/\ell'$ iterations, the claim regarding the value of the output gate (i.e., the value $C_{|x|}(x)$) is reduced to a claim regarding the values of the bits of the input x, whereas the latter claim (which refers to x itself) can be verified in almost linear time.

Each of the aforementioned claims regarding the values of the gates at layer $i \cdot \ell'$, where $i \in \{0, 1, ..., t\}$, is actually a claim about the value of a specified location in the corresponding encoding of the (string that describing all the) gate-values at layer $i \cdot \ell'$. Specifically, the encoding used is the low degree extension of the said string (viewed as a function), and the claims are claims about the evaluations of these polynomials at specific points.

The different codewords (or polynomials) are related via the structure of the circuit $C_{|x|}$, which is the case of canonical circuit is straightforward to implement (avoiding a main source of technical difficulty in [8] (see also [4])). Indeed, this reduces a claim regarding one value in the encoding of layer $(i-1) \cdot \ell'$ to $2^{\ell'} = n^\delta$ analogous claims regarding layer $i \cdot \ell'$, but (as in [8]) "batch verification" is possible, reducing these $2^{\ell'}$ claims to a single claim.

3.2 The actual construction

For simplicity (and w.l.o.g.), we assume that C_n contains only NAND-gates of (fan-in two), where $\mathtt{NAND}(a,b) = \neg(a \wedge b)$. Viewing this gate as operating in a finite field that contains $\{0,1\}$, we have $\mathtt{NAND}(a,b) = 1 - (a \cdot b)$ for $a, b \in \{0,1\}$. The function computed by tree of depth i of such gates is given by

$$\mathtt{NAND}_i(b_1, ..., b_{2^i}) = 1 - (\mathtt{NAND}_{i-1}(b_1, ..., b_{2^{i-1}}) \cdot \mathtt{NAND}_{i-1}(b_{2^{i-1}+1}, ..., b_{2^i})), \tag{10}$$

where $b_1, ..., b_{2^i} \in \{0,1\}$ are the values at its leaves and $\mathtt{NAND}_0(b) = b$; indeed, $\mathtt{NAND}_1 = \mathtt{NAND}$.

By augmenting the circuit with gates that are fed by no gate (and feed no gate), we can present the circuit as having $d(n) + 1$ layers of gates such that each layer has exactly $k(n) =$

$2^{d(n)} = \mathrm{poly}(n)$ gates, where (by convention) gates that are fed nothing always evaluate to 0. As usual, the gates at layer i are only fed by gates at layer $i+1$, and the leaves (at layer $d(n)$) are input-variables or constants. Recall that the latter assignment is represented by the function $\phi : \{0,1\}^\ell \to [n] \cup \{0, n+1\}$, where $\ell = d(n)$ and $[k(n)] \equiv \{0,1\}^\ell$, such that the j^{th} leaf is fed the variable $x_{\phi(j)}$ if $\phi(j) \in [n]$ (and the constant $\phi(j) \bmod n$ otherwise). The output is produced at the first gate of layer zero.

The high level protocol. On input $x \in \{0,1\}^n$, the prescribed prover computes the values of all layers. Letting $d = d(n)$ and $k = k(n)$, we denote the values at the i^{th} layer by $\alpha_i \in \{0,1\}^k$; in particular, $\alpha_0 = C_n(x)0^{k-1}$ and α_d is the sequence of values given by $x_{\phi(0^\ell)}, ..., x_{\phi(1^\ell)}$, where $x_0 = 0$ and $x_{n+1} = 1$. For a sufficiently large finite field, denoted $\mathcal{F}$, consider an arbitrary fixed set $H \subset \mathcal{F}$ of size $2^{\ell'}$, where $\ell' = \delta \cdot \log n$, and let $m = \log_{|H|} k = (\log k)/\log|H| = d/\ell' = O(1/\delta)$.[18] For each $i \in \{0, 1, ..., d-1\}$, viewing α_i as a function from $H^m \equiv [k]$ to $\{0,1\}$, the prover encodes α_i by a low degree polynomial $\widehat{\alpha}_i : \mathcal{F}^m \to \mathcal{F}$ that extends it (i.e., $\widehat{\alpha}_i(\sigma) = \alpha_i(\sigma)$ for every $\sigma \in H^m$); that is,

$$\widehat{\alpha}_i(z_1, ..., z_m) = \sum_{\sigma_1,...,\sigma_m \in H} \mathtt{EQ}(z_1 \cdots z_m, \sigma_1 \cdots \sigma_m) \cdot \alpha_i(\sigma_1, ..., \sigma_m) \qquad (11)$$

where $\mathtt{EQ}$ is a low degree polynomial in the z_i's that tests equality over H^m (i.e., $\mathtt{EQ}(z_1 \cdots z_m, \sigma_1 \cdots \sigma_m) = \prod_{i\in[m]} \mathtt{EQ}_{\sigma_i}(z_i)$ and $\mathtt{EQ}_\sigma(z) = \prod_{\beta \in H\setminus\{\sigma\}} (z-\beta)/(\sigma-\beta)$). Actually, recalling that all but the first 2^i gates of layer i evaluate to 0, for i's that are multiples of ℓ', we re-write Eq. (11) as

$$\widehat{\alpha}_{i'\cdot\ell'}(z_1, ..., z_m) = \sum_{\sigma_1,...,\sigma_{i'} \in H} \mathtt{EQ}(z_1 \cdots z_m, 1^{m-i'}\sigma_1 \cdots \sigma_{i'}) \cdot \alpha_i(1, ..., 1, \sigma_1, ..., \sigma_{i'}) \qquad (12)$$

Either way, $\widehat{\alpha}_i$ is a polynomial of individual degree $|H|-1$.

In light of the foregoing, proving that $C_n(x) = 1$ is equivalent to proving that $\widehat{\alpha}_0(1^m) = 1$, where $1^m \in H^m$ corresponds to the fixed (e.g., first) location of the output gate in the zero layer. This proof is conducted in $t = d/\ell'$ iterations, where in each iteration a multi-round interactive protocol is employed. Specifically, in i^{th} iteration, the correctness of the claim $\widehat{\alpha}_{(i-1)\cdot\ell'}(\overline{r}_{i-1}) = v_{i-1}$, where $\overline{r}_{i-1} \in \mathcal{F}^m$ and $v_{i-1} \in \mathcal{F}$ are known to both parties, is reduced (via the interactive protocol) to the claim $\widehat{\alpha}_{i\cdot\ell'}(\overline{r}_i) = v_i$, where $\overline{r}_i \in \mathcal{F}^m$ and $v_i \in \mathcal{F}$ are determined (by this protocol) such that both parties get these values. We stress that, with the exception of $i = t$, the $\widehat{\alpha}_{i\cdot\ell'}$'s are not known (or given) to the verifier; still, the claims made at the beginning (and at the end) of each iteration are well defined (i.e., each claim refers to a predetermined low degree polynomial that extends the values assigned to the gates (of a certain layer) of the circuit in a computation of the circuit on input $x \in \{0,1\}^n$).

Once the last iteration is completed, the verifier is left with a claim of the form $\widehat{\alpha}_d(\overline{r}_t) = v_t$, where $\widehat{\alpha}_d$ is defined as in Eq. (9). Recall that Eq. (9) reads $\widehat{\alpha}_d(y) = \sum_{k\in\{0,1\}^\ell} \mathtt{EQ}(y,k) \cdot \mathtt{I}(k)$, where $\mathtt{I}(z) \overset{\text{def}}{=} \sum_{v\in[n+1]} \mathtt{EQ}(v, \widehat{\phi}(z)) \cdot x_v$ and $\widehat{\phi}$ is polynomial that is obtained by transforming the Boolean formula that computes ϕ to a corresponding arithmetic formula.) Hence, the verifier cannot evaluate $\widehat{\alpha}_d$ by itself, but it can verify its value via the Sum-Check protocol, since $\mathtt{I}$ is a low degree polynomial that can be evaluated in almost linear (in n) time. So, at this point, the parties run the Sum-Check protocol (as in Remark 9).

[18] The setting of $|H| = 2^{\ell'}$ simplifies the exposition. Actually, we may use arbitrary $t = d/\ell'$ and $m = d/\log|H|$, although letting m be an integer multiple of t (equiv., ℓ' be an integer multiple of $\log|H|$) simplifies the exposition. In the general case, we obtain a $O(tm)$-round protocol of total communication $\widetilde{O}(2^{d/t} \cdot 2^{d/m})$.

A single iteration. The core of the iterative proof is the interactive protocol that is performed in each iteration. This protocol is based on the relation between subsequent α_i's, which is based on the canonical structure of the circuit. Specifically, recall that the i^{th} iteration reduces a claim regarding $\widehat{\alpha}_{(i-1)\cdot\ell'}$ to a claim regarding $\widehat{\alpha}_{i\cdot\ell'}$, where these polynomials encode the values of the corresponding layers in the circuit (i.e., layers $(i-1)\cdot\ell'$ and $i\cdot\ell'$). The relation between these layers is given by

$$\alpha_{(i-1)\cdot\ell'}(1,...,1,u_1,...,u_{i-1}) = \texttt{NAND}_{\ell'}((\alpha_{i\cdot\ell'}(1,...,1,u_1,...,u_{i-1},u))_{u\in H}) \tag{13}$$

where $u_1,...,u_{i-1} \in H \equiv \{0,1\}^{\ell'}$ and $\texttt{NAND}_{\ell'}$ is as defined in Eq. (10). Combining Eq. (12) with Eq. (13), it holds that $\widehat{\alpha}_{(i-1)\cdot\ell'}(z_1,...,z_m)$ equals

$$\sum_{u_1,...,u_{i-1}\in H} \texttt{EQ}(z_1\cdots z_m, 1^{m-(i-1)}u_1\cdots u_{i-1}) \cdot \texttt{NAND}_{\ell'}((\widehat{\alpha}_{i\cdot\ell'}(1,...,1,u_1,...,u_{i-1},u))_{u\in H}). \tag{14}$$

In preparation to applying the Sum-Check protocol to Eq. (14), we observe that the corresponding $(i-1)$-variate is of individual degree $O(2^{\ell'}\cdot|H|) = O(n^{2\delta})$. This is the case because, for any fixed point $(\overline{r}',\overline{r}'') \in \mathcal{F}^{m-i+1}\times\mathcal{F}^{i-1}$, we can write Eq. (14) as

$$\begin{aligned}
&\texttt{EQ}(\overline{r}',1^{m-i+1})\cdot \sum_{u_1,...,u_{i-1}\in H} \texttt{EQ}(\overline{r}'',u_1\cdots u_{i-1})\cdot\texttt{NAND}_{\ell'}((\widehat{\alpha}_{i\cdot\ell'}(1,...,1,u_1,...,u_{i-1},u))_{u\in H}) \\
&= \texttt{EQ}(\overline{r}',1^{m-i+1})\cdot \sum_{u_1,...,u_{i-1}\in H} P_{\overline{r}''}(u_1,...,u_{i-1}),
\end{aligned}$$

where $P_{\overline{r}''}(y_1,...,y_{i-1}) \overset{\text{def}}{=} \texttt{EQ}(\overline{r}'',y_1\cdots y_{i-1})\cdot\texttt{NAND}_{\ell'}((\widehat{\alpha}_{i\cdot\ell'}(1,...,1,y_1,...,y_{i-1},u))_{u\in H})$ is a low degree $(i-1)$-variate polynomial; specifically, its individual degree is dominated by the product of the total degree of $\texttt{NAND}_{\ell'}$ and the individual degree of $\widehat{\alpha}_{i\cdot\ell'}$, which are $2^{\ell'}$ and $|H|-1$, respectively.

Applying the Sum-Check protocol to Eq. (14) allows to reduce a claim regarding the value of $\widehat{\alpha}_{(i-1)\cdot\ell'}$ at a specific point $\overline{r}_{i-1} = (\overline{r}'_{i-1},\overline{r}''_{i-1}) \in \mathcal{F}^{m-i+1}\times\mathcal{F}^{i-1}$ to a claim regarding the value of the polynomial $P_{\overline{r}''_{i-1}}$ at a random point $(r''_1,....,r''_{i-1})$ in $\mathcal{F}^{i-1}$, which in turn depends on the values of $\widehat{\alpha}_{i\cdot\ell'}$ at $2^{\ell'}$ points in $\mathcal{F}^m$ (i.e., the points $((1,...,1,r''_1,...,r''_{i-1},u))_{u\in H})$).

To reduce this claim to a claim regarding the value of $\widehat{\alpha}_{i\cdot\ell'}$ at a single point, we let the prover send these $2^{\ell'}$ values and perform "batch verification" for them. Specifically, the prover provides a low degree polynomial that describes the value of $\widehat{\alpha}_{i\cdot\ell'}$ on the axis-parallel line that goes through these points, and the claim to be proved in the next iteration is that the value of $\widehat{\alpha}_{i\cdot\ell'}$ at a random point on this line equals the value provided by the polynomial sent by the prover.[19] Hence, the full protocol that is run in iteration i is as follows.

Construction 11 (reducing a claim about $\widehat{\alpha}_{(i-1)\cdot\ell'}$ to a claim about $\widehat{\alpha}_{i\cdot\ell'}$): *For known $\overline{r}_{i-1}\in\mathcal{F}^m$ and $v_{i-1}\in\mathcal{F}$, the* entry claim *is $\widehat{\alpha}_{(i-1)\cdot\ell'}(\overline{r}_{i-1}) = v_{i-1}$. The parties proceed as follows.*

[19] We mention that the fact that these $2^{\ell'}$ points reside on a line makes the argument simpler, but not in a fundamental way. In general, the prover could have picked a curve of degree $2^{\ell'}-1$ that goes through any $2^{\ell'}$ points of interest, and provide a low degree polynomial describing the value of $\widehat{\alpha}_{i\cdot\ell'}$ on this curve. In this case, the claim to be proved in the next iteration would have been that the value of $\widehat{\alpha}_{i\cdot\ell'}$ at a random point on this curve equals the value provided by the polynomial sent by the prover.

1. *Applying the Sum-Check protocol to the entry claim, when expanded according to Eq. (14), determines $\overline{r}' \in \mathcal{F}^{i-1}$ and a value $v \in \mathcal{F}$ such that the residual claim for verification is*

$$\texttt{EQ}(\overline{r}_{i-1}, 1^{m-(i-1)}\overline{r}') \cdot \texttt{NAND}_{\ell'}((\widehat{\alpha}_{i\cdot\ell'}(1,...,1,\overline{r}',u))_{u\in H}) = v. \tag{15}$$

2. *The prover sends a univariate polynomial p' of degree smaller than $m \cdot |H|$ such that $p'(z) = \widehat{\alpha}_i(1,...,1,\overline{r}',z)$.*

3. *Upon receiving the polynomial p', the verifier checks whether v equals*

$$\texttt{EQ}(\overline{r}_{i-1}, 1^{m-(i-1)}\overline{r}') \cdot \texttt{NAND}_{\ell'}((p'(u))_{u\in H}), \tag{16}$$

and continues only if equality holds (otherwise it rejects)*.*

4. *The verifier selects a random $r \in \mathcal{F}$, and sends it to the prover. Both parties set $\overline{r}_i = (1,...,1,\overline{r}',r)$ and $v_i = p'(r)$.*

The exit claim *is $\widehat{\alpha}_{i\cdot\ell'}(\overline{r}_i) = v_i$.*

The complexities of Construction 11 are dominated by the application of the Sum-Check protocol, which refers to a polynomial of degree $O(2^{\ell'} \cdot |H|) = O(n^{2\delta})$. In particular, this implies that the verifier's strategy can be implemented in time $\widetilde{O}(n^{2\delta})$, provided that $|\mathcal{F}| = \text{poly}(n)$. In this case, the prescribed prover strategy (as defined in Construction 11) can be implemented in time $\widetilde{O}(2^{d(n)}) = \text{poly}(n)$,

Recall that after the last iteration of Construction 11, the resulting claim is checked by the Sum-Check protocol (applied to Eq. (9)), which leaves the verifier with the task of evaluating $\texttt{I}$, where $\texttt{I}(z) \stackrel{\text{def}}{=} \sum_{v\in[n]\cup\{0,n+1\}} \texttt{EQ}(v, \widehat{\phi}(z)) \cdot x_v$. Using the hypothesis regarding ϕ, it follows that the verifier runs in $n^{1+o(1)}$-time. The round complexity of the i^{th} iteration of Construction 11 is $i \leq m$, and so the total round complexity is $m \cdot m + m = O(d(n)/\delta \log n)^2$.

One can readily verify that if the entry claim is correct, then the exit claim is correct, whereas if the entry claim is false, then with probability at least $1 - O(m \cdot 2^{\ell'} \cdot |H|/|\mathcal{F}|)$ the exit claim is false. Recall that the soundness error of the entire protocol is upper-bounded by the probability that there exists an iteration in which the entry claim is false but the exist claim is true. Hence, the total soundness error is $O(n^{2\delta}/|\mathcal{F}|) = o(1)$.

Acknowledgements

As noted in the body of the paper, an unpublished work by Yael Kalai and Guy Rothblum [12] proposed a constant-round doubly-efficient proof system for $\mathcal{NC}^1$ under a very strict notion of uniformity. This unpublished work has inspired our own work, and we thank Yael for her contribution to it as well as for many other helpful conversations on these topics.

References

[1] Noga Alon, Oded Goldreich, Johan Hastad, and Rene Peralta. Simple Construction of Almost k-wise Independent Random Variables. *Random Structures and Algorithms*, Vol. 3 (3), pages 289–304, 1992.

[2] David A. Mix Barrington, Neil Immerman, and Howard Straubing. On Uniformity within NC1. *Journal of Computer and System Science*, Vol. 41 (3), pages274–306, 1990.

[3] Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.

[4] Oded Goldreich. On the doubly-efficient interactive proof systems of GKR. *ECCC*, TR17-101, 2017.

[5] Oded Goldreich and Guy N. Rothblum. Simple doubly-efficient interactive proof systems for locally-characterizable sets. *ECCC*, TR17-018, 2017.

[6] Oded Goldreich and Guy N. Rothblum. Worst-case to Average-case reductions for subclasses of P. *ECCC* TR17-130, 2017.

[7] Oded Goldreich and Guy N. Rothblum. Counting t-cliques: Worst-case to average-case reductions and direct interactive proof systems. *ECCC* TR18-046, 2018.

[8] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating Computation: Interactive Proofs for Muggles. *Journal of the ACM*, Vol. 62(4), Art. 27:1-27:64, 2015. Extended abstract in *40th STOC*, pages 113–122, 2008.

[9] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on Computing*, Vol. 18, pages 186–208, 1989. Preliminary version in *17th STOC*, 1985.

[10] Alexander Healy and Emanuele Viola. Constant-Depth Circuits for Arithmetic in Finite Fields of Characteristic Two. *ECCC* TR05-087, 2005.

[11] Yael Tauman Kalai and Ran Raz. Interactive PCP. In *35th International Colloquium on Automata, Languages and Programming* (Part II), pages 536–547, 2008.

[12] Yael Tauman Kalai and Guy N. Rothblum. Constant-round interactive proofs for NC1. Unpublished observation, 2009.

[13] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, Vol. 39, No. 4, pages 859–868, 1992. Extended abstract in *31st FOCS*, 1990.

[14] Joseph Naor and Moni Naor. Small-Bias Probability Spaces: Efficient Constructions and Applications. *SIAM Journal on Computing*, Vol. 22 (4), pages 838–856, 1993. Preliminary version in *22nd STOC*, 1990.

[15] Alexander A. Razborov. Lower bounds on the size of bounded-depth networks over a complete basis with logical addition. In *Matematicheskie Zametki*, Vol. 41, No. 4, pages 598–607, 1987 (in Russian). English translation in *Mathematical Notes of the Academy of Sci. of the USSR*, Vol. 41 (4), pages 333–338, 1987.

[16] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In *48th ACM Symposium on the Theory of Computing*, pages 49–62, 2016.

[17] Adi Shamir. IP = PSPACE. *Journal of the ACM*, Vol. 39, No. 4, pages 869–877, 1992. Preliminary version in *31st FOCS*, 1990.

[18] Roman Smolensky. Algebraic Methods in the Theory of Lower Bounds for Boolean Circuit Complexity. In *19th ACM Symposium on the Theory of Computing* pages 77–82, 1987.

[19] Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *46th ACM Symposium on the Theory of Computing*, pages 664–673, 2014.

Appendix: The Sum-Check protocol

The Sum-Check protocol, designed by Lund, Fortnow, Karloff, and Nisan [13], is a key ingredient in the constructions that we present.

Fixing a finite field $\mathcal{F}$ and a set $H \subset \mathcal{F}$ (e.g., H may be a two-element set), we consider an m-variate polynomial $P : \mathcal{F}^m \to \mathcal{F}$ of individual degree d. Given a value v, the Sum-Check protocol is used to prove that

$$\sum_{\sigma_1,...,\sigma_m \in H} P(\sigma_1, ..., \sigma_m) = v, \tag{17}$$

assuming that the verifier can evaluate P by itself. The Sum-Check protocol proceeds in m iterations, such that in the $i^{\rm th}$ iteration the number of summations (over H) decreases from $m-i+1$ to $m-i$. Specifically, the $i^{\rm th}$ iteration starts with a claim of the form $\sum_{\sigma_i,...,\sigma_m \in H} P(r_1, ..., r_{i-1}, \sigma_i, ..., \sigma_m) = v_{i-1}$, where $r_1, ..., r_{i-1}$ and v_{i-1} are as determined in prior iterations (with $v_0 = v$), and ends with a claim of the form $\sum_{\sigma_{i+1},...,\sigma_m \in H} P(r_1, ..., r_i, \sigma_{i+1}, ..., \sigma_m) = v_i$, where r_i and v_i are determined in the $i^{\rm th}$ iteration. Initializing the process with $v_0 = v$, in the $i^{\rm th}$ iteration the parties act as follows.

Prover's move: The prover computes a univariate polynomial of degree d over $\mathcal{F}$

$$P_i(z) \stackrel{\rm def}{=} \sum_{\sigma_{i+1},...,\sigma_m \in H} P(r_1, ..., r_{i-1}, z, \sigma_{i+1}, ..., \sigma_m) \tag{18}$$

where $r_1, ..., r_{i-1}$ are as determined in prior iterations, and sends P_i to the verifier (claiming that $\sum_{\sigma \in H} P_i(\sigma) = v_{i-1}$).

Verifier's move: Upon receiving a degree d polynomial, denoted $\widetilde{P}$, the verifier checks that $\sum_{\sigma \in H} \widetilde{P}(\sigma) = v_{i-1}$ and rejects if inequality holds. Otherwise, it selects r_i uniformly in $\mathcal{F}$, and sends it to the prover, while setting $v_i \leftarrow \widetilde{P}(r_i)$.

If all m iterations are completed successfully (i.e., without the verifier rejecting in any of them), the verifier conducts a final check. It computes the value of $P(r_1, ..., r_m)$ and accepts if and only if this value equals v_m.

Clearly, if Eq. (17) holds (and the prover acts according to the protocol), then the verifier accepts with probability 1. Otherwise, no matter what the prover does, the verifier accepts with probability at most $m \cdot d/|\mathcal{F}|$, because in each iteration if the prover provides the correct polynomial, then the verifier rejects (since $\sum_{\sigma \in H} P_i(\sigma) = P_{i-1}(r_{i-1}) \neq v_{i-1}$), and otherwise the (degree d) polynomial sent agrees with P_i on at most d points.[20]

The complexity of verification is dominated by the complexity of evaluating P (on a single point). As for the prescribed prover, it may compute the relevant P_i's by interpolation, which is based on computing the value of P at $(d+1) \cdot |H|^{m-i}$ points, for each $i \in [m]$. (That is, the polynomial P_i is computed by obtaining its values at $d+1$ points, where the value of P_i at each point is obtained by summing the values of P at $|H|^{m-i}$ points.)[21]

[20] If P_i does not satisfy the current claim (i.e., $\sum_{\sigma \in H} P_i(\sigma) \neq v_{i-1}$), then the prover can avoid upfront rejection only if it sends $\widetilde{P} \neq P_i$. But in such a case, $\widetilde{P}$ and P_i (both being degree d polynomials) may agree on at most d points. Hence, if the chosen $r_i \in \mathcal{F}$ is not one of these points, it holds that $v_i = \widetilde{P}(r_i) \neq P_i(r_i)$, which means that the next iteration will also start with a false claim. Hence, starting with a false claim (i.e., $\sum_{\sigma \in H} P_1(\sigma) \neq v_0$ since Eq. (17) does not hold), with probability at least $1 - m \cdot d/|\mathcal{F}|$, after m iterations we reach a false claim regarding the value of P at a single point.

[21] Specifically, the value of P_i at p is obtained from the values of P at the points $(r_1, ..., r_{i-1}, p, \sigma)$, where $\sigma \in H^{m-i}$.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il