Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 101 (2017)

On the doubly-efficient interactive proof systems of GKR

Oded Goldreich*

January 27, 2018

Abstract

We present a somewhat simpler variant of the doubly-efficient interactive proof systems of Goldwasser, Kalai, and Rothblum (*JACM*, 2015). Recall that these proof systems apply to log-space uniform sets in $\mathcal{NC}$ (or, more generally, to inputs that are acceptable by log-space uniform bounded-depth circuits, where the number of rounds in the proof system is linearly related to the depth of the circuit). Our simplification is in the handling of the log-space uniformity condition. Rather than having the prover provide the verifier with bits of the encoding of the circuit and establish their correctness, we employ the proof system to a highly regular universal circuit that constructs and evaluates the log-space uniform circuit in question.

Contents

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

ISSN 1433-8092

1 Overview

Loosely speaking, doubly-efficient interactive proof systems are interactive proof systems in which the prescribed prover runs in polynomial-time, whereas the prescribed verifier runs in almost-linear-time.[1] We stress that the soundness condition of these systems is information theoretic; that is, it refers to all possible cheating strategies (and not only to feasible ones (as in argument systems)).

The notion of a doubly-efficient interactive proof systems was first defined by Goldwasser, Kalai, and Rothblum [2], who presented such systems for any set in log-space uniform $\mathcal{NC}$. We mention that the recent result of Reingold, Rothblum, and Rothblum [5] provides (constant-round) doubly-efficient interactive proof systems for any set in $\mathcal{SC}$.[2] Our focus, however, is on the prior result of Goldwasser, Kalai, and Rothblum [2]. We provide an (alternative) exposition of their result, which asserts the following –

Theorem 1 (doubly-efficient interactive proof systems for log-space uniform $\mathcal{NC}$): *Let $d : \mathbb{N} \to \mathbb{N}$ and A be an algorithm of logarithmic space complexity that on input 1^n outputs a Boolean circuit $C_n : \{0,1\}^n \to \{0,1\}$ of bounded fan-in and depth $d(n)$. Then, the set $S = \{x : C_{|x|}(x)=1\}$ has an interactive proof system with $\mathrm{poly}(\log n) \cdot d(n)$ rounds in which the prescribed prover strategy runs in polynomial-time while the verifier runs in $\widetilde{O}(n + d(n))$-time.*

(We note that the number of rounds is actually $O(\log n) \cdot d(n)$, but our simplified proof only yield a bound of $O(\log n)^2 \cdot d(n)$.)[3]

The core of the proof system asserted in Theorem 1 is an iterative process in which a claim about the values of the gates that are at distance $i-1$ from the output gate is reduced to a claim about the values of the gates at distance i. (All claims refer to the computation of $C_{|x|}(x)$.) Hence, in $d(n)$ iterations, the claim regarding the value of the output gate is reduced to a claim regarding the values of the bits of the inputs, whereas this claim can be verified in almost linear time.

The aforementioned claims refer to the values of specified locations in corresponding encodings (of the string describing all the gate-values at a certain layer of the circuit). 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, and so (as usual) this structure is represented by a function that describes the adjacency relation (among the gates) of the circuit. We consider a low degree extension of this function, and the problem is allowing the verifier to evaluate this polynomial in almost-linear (in n) time. Here is where the uniformity condition comes into play.

Goldwasser, Kalai, and Rothblum used the uniformity condition in order to construct an interactive proof system by which the evaluation of this low degree polynomial is outsourced to the prover (who proves the validity of the provided answer) [2]. We present an alternative approach: We consider a "universal" circuit that, on input $x \in \{0,1\}^n$, first constructs the log-space uniform circuit C_n, and next emulates the computation of $C_n(x)$. We apply the aforementioned interactive

[1] Such proof systems were called *interactive proofs for muggles* [2] and *interactive proofs for delegating computation* [5]. Here, we interpret the term "almost linear (in n)" as having the form $\widetilde{O}(n)$, whereas a wider interpretation refers to having the form $n^{1+o(1)}$.

[2] The result of [5] actually provides an interactive proof for any set in $\mathtt{TiSp}(\mathrm{poly}, s)$ such that the prescribed prover strategy runs in polynomial-time while the verifier runs in $(\widetilde{O}(n) + \mathrm{poly}(s(n)) \cdot n^{\alpha})$-time, for any $\alpha > 0$.

[3] The first (resp., second) expression omits an additive term of $O(\log n)^2$ (resp., $O(\log n)^2$), which is immaterial in the typical case where $d \geq \log n$.

process to this universal circuit rather than to C_n, with the benefit being that this universal circuit is much more uniform than C_n. Consequently, a low degree extension[4] of the adjacency relation of that (universal) circuit can be evaluated in poly$(\log n)$-time. We stress that the universal circuit preserves the complexities of C_n upto our level of interest; that is, the universal circuit has depth poly$(\log n) \cdot d(n)$ and size poly(n).

Organization. The main module of the interactive proof system asserted in Theorem 1 is presented in Section 2, while avoiding the problem of evaluating the low degree extension of the function that represents the adjacency relation of the circuit. Coping with this computational problem is the contents of Section 3. Additional details regarding Sections 2 and 3 are presented in Section 4 (see Sections 4.1 and 4.2, resp.).

2 The main module

For simplicity (and w.l.o.g.), we assume that C_n has fan-in two and uses only NAND-gates. Viewing this gates as operating in GF(2), we have $\texttt{NAND}(a,b) = 1 - (a \cdot b)$.

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 $k(n) = \text{poly}(n)$ 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. Furthermore, we may assume that, for $j \in [n]$, the j^{th} leaf is fed by the j^{th} input-variable, and all other leaves are fed by the constant 0. (The output is produced at the first gate of layer zero.)

The high level protocol. On input $x \in \{0,1\}^n$, the presecribed 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, $\alpha_d = x0^{k-n}$ and $\alpha_0 = C_n(x)0^{k-1}$. For a sufficiently large finite field, denoted $\mathcal{F}$, we let $H \subset \mathcal{F}$ be a fixed set of size approximately $\log n$ and $m = \log_{|H|} k = (\log k)/\log |H|$ such that $k = |H|^m$.[5] Viewing each α_i as a function from H^m to $\{0,1\}$, the prover encodes α_i using its low degree extension $\widehat{\alpha}_i : F^m \to \mathcal{F}$; that is,

$$\widehat{\alpha}_i(z_1,...,z_m) = \sum_{\sigma_1,...,\sigma_m \in H} \texttt{EQ}(z_1 \cdots z_m, \sigma_1 \cdots \sigma_m) \cdot \alpha_i(\sigma_1,...,\sigma_m) \tag{1}$$

where EQ is a low degree extention of the function that tests equality over H^m (e.g., $\texttt{EQ}(z_1 \cdots z_m, \sigma_1 \cdots \sigma_m) = \prod_{i\in[m]} \prod_{\beta \in H'} (z_i - \sigma_i + \beta)/\beta$, where $H' = \{\beta' - \beta'' : \beta', \beta'' \in H\} \setminus \{0\}$). Hence, proving that $x \in S$ 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.

[4] Actually, we use a low degree polynomial that agrees with the function; this low degree polynomial may have higher degree than the standarsd low degree extension.

[5] As noted in [1, Chap. 3], the exposition can be simplified by using $H = \{0,1\}$ and $m = \log k$. This simplifies both the definition of EQ (i.e., we can use $\texttt{EQ}(z_1 \cdots z_m, \sigma_1 \cdots \sigma_m) = \prod_{i\in[m]}(z_i\sigma_i + (1-z_i)(1-\sigma_i))$) and the proof of Theorem 3 (which becomes straightforwards). The advantage of the current choice (of $|H| = \text{poly}(\log k)$) is that, in the case that $|\mathcal{F}| = \text{poly}(|H|)$, which suffices when $d(n) = \text{poly}(\log n)$, we get $|\mathcal{F}|^m = \text{poly}(k) = \text{poly}(n)$, which means that the codewords used are of polynomial length. However, this fact is only of aesthetic value, because these codewords are not constructed explicitly by the prover, who only determines their k-bit long succinct representation. In fact, it may be methodologically better to think that it is infeasible to construct these codewords explicitly.

This proof is conducted in $d(n)$ iterations, where in each iteration a multi-round interactive protocol is employed. Specifically, in i^{th} iteration, the correctness of the claim $\widehat{\alpha}_{i-1}(\overline{r}_{i-1}) = v_{i-1}$, 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 $\widehat{\alpha}_i(\overline{r}_i) = v_i$, 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 $i = d$, the $\widehat{\alpha}_i$'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., they refer to the low degree extension of the values at certain layers of the circuit in a computation of the circuit on input $x \in \{0,1\}^n$).

After the last iteration, the verifier is left with a claim of the form $\widehat{\alpha}_d(\overline{r}_d) = v_d$, which it can verify using Eq. (1) and its knowledge of $\alpha_d = x0^{k-n}$. Actually, some care is required here too, since the verifier needs to operate in time $\widetilde{O}(n)$ (rather than in time $\widetilde{O}(k)$, where $k = |H|^m = \text{poly}(n)$). For example, associating $[n]$ with $H^{m'} \equiv H^{m'}1^{m-m'}$, we use the fact that for every $\sigma_1, ..., \sigma_m \in H$ it holds that $\alpha_d(\sigma_1, ..., \sigma_m) = x_{\sigma_1 \cdots \sigma_{m'}}$ if $\sigma_{m-m'+1} \cdots \sigma_m = 1^{m-m'}$ and $\alpha_d(\sigma_1, ..., \sigma_m) = 0$ otherwise. Hence,

$$\begin{aligned}\widehat{\alpha}_d(z_1, ..., z_m) &= \sum_{\sigma_1,...,\sigma_m \in H} \texttt{EQ}(z_1 \cdots z_m, \sigma_1 \cdots \sigma_m) \cdot \alpha_d(\sigma_1, ..., \sigma_m) \\ &= \sum_{\sigma_1,...,\sigma_{m'} \in H} \texttt{EQ}(z_1 \cdots z_m, \sigma_1 \cdots \sigma_{m'}1^{m-m'}) \cdot \alpha_d(\sigma_1, ..., \sigma_{m'}1^{m-m'}).\end{aligned}$$

where $\alpha_d(\sigma_1, ..., \sigma_{m'}1^{m-m'})$ equals the bit of x in location $\sigma_1 \cdots \sigma_{m'} \in H^{m'} \equiv [n]$.

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 structure of the circuit. Specifically, recall that the i^{th} iteration reduces a claim regarding $\widehat{\alpha}_{i-1}$ to a claim regarding $\widehat{\alpha}_i$, where these polynomials encode the values of neighboring layers in the circuit. To describe the relation between such neighboring layers, we consider the adjacency relation of the circuit. Actually, we let $\phi_i : H^m \times H^m \times H^m \to \{0,1\}$ represent the feeding relation between the i^{th} and $(i-1)^{\text{st}}$ layer of the circuit such that $\phi_i(\overline{u}, \overline{v}, \overline{w}) = 1$ if and only if the gate in location $\overline{u} \in H^m$ in layer $i-1$ is fed by the gates in locations $\overline{v}$ and $\overline{w}$ in layer i. Hence, for every $\overline{u} \in H^m$, we have

$$\alpha_{i-1}(\overline{u}) = \sum_{\overline{v},\overline{w} \in H^m} \phi_i(\overline{u}, \overline{v}, \overline{w}) \cdot (1 - \alpha_i(\overline{v}) \cdot \alpha_i(\overline{w})) \tag{2}$$

Recall that, by Eq. (1), we have $\widehat{\alpha}_{i-1}(\overline{z}) = \sum_{\overline{u} \in H^m} \texttt{EQ}(\overline{z}, \overline{u}) \cdot \alpha_{i-1}(\overline{u})$. Combining this with Eq. (2), while letting $\widehat{\phi}_i : \mathcal{F}^{3m} \to \mathcal{F}$ denote the low degree extension of ϕ_i, we have

$$\widehat{\alpha}_{i-1}(\overline{z}) = \sum_{\overline{u},\overline{v},\overline{w} \in H^m} \texttt{EQ}(\overline{z}, \overline{u}) \cdot \widehat{\phi}_i(\overline{u}, \overline{v}, \overline{w}) \cdot (1 - \widehat{\alpha}_i(\overline{v}) \cdot \widehat{\alpha}_i(\overline{w})). \tag{3}$$

For any fixed point $\overline{r}_{i-1} \in F^m$, the expression at the r.h.s of Eq. (3) can be written as $\sum_{\overline{u},\overline{v},\overline{w} \in H^m} p_{\overline{r}_{i-1}}(\overline{u}, \overline{v}, \overline{w})$, where $p_{\overline{r}_{i-1}}(\overline{z}', \overline{z}'', \overline{z}''') \stackrel{\text{def}}{=} \texttt{EQ}(\overline{r}_{i-1}, \overline{z}') \cdot \widehat{\phi}_i(\overline{z}', \overline{z}'', \overline{z}''') \cdot (1 - \widehat{\alpha}_i(\overline{z}'') \cdot \widehat{\alpha}_i(\overline{z}'''))$ is a low degree polynomial. Applying the sum-check protocol to Eq. (3) allows to reduce a claim regarding the value of $\widehat{\alpha}_{i-1}$ at a specific point $\overline{r}_{i-1} \in F^m$ to a claim regarding the value of the polynomial $p_{\overline{r}_{i-1}}$ at a random point $(\overline{r}', \overline{r}'', \overline{r}''')$ in F^{3m} (for details see Section 4.1). This is not quite what we aimed for but it is close enough, since the verifier can easily evaluate EQ at any point. However, there are two problems:

1. The verifier needs to be able to evaluate $\widehat{\phi}_i$ in $\widetilde{O}(n)$-time. The definition of $\widehat{\phi}_i$ does imply that evaluating $\widehat{\phi}_i$ reduces to $|H|^{3m} = k^3$ evaluations of ϕ_i, but $n \leq k = \text{poly}(n)$ which may be much larger than n. Hence, evaluating $\widehat{\phi}_i$ in $\widetilde{O}(n)$-time is not obvious.

2. After evaluating $\texttt{EQ}$ and $\widehat{\phi}_i$, the verifier is left with a residual claim that refers to two evaluations of $\widehat{\alpha}_i$ rather than to a single one.

The second problem is handled by augmenting the protocol such that the prover provides the value of $\widehat{\alpha}_i$ at both points (i.e., at $\overline{r}''$ and $\overline{r}'''$) as well as on the line that connects these points, and is asked to prove the correctness of the value at a random point on this line. That is, if the prover claims that $\widehat{\alpha}_i(\overline{r}'') = v''$ and $\widehat{\alpha}_i(\overline{r}''') = v'''$, then it also provides a low degree univariate polynomial p' such that $p'(z) = \widehat{\alpha}_i((1-z)\overline{r}'' + z\overline{r}''')$, which satisfies that $p'(0) = v''$ and $p'(1) = v'''$, and the verifier selects a random $r \in \mathcal{F}$ and sends it to the prover. The claim to be proved in the next iteration is that the value of $\widehat{\alpha}_i$ at $\overline{r}_i = (1-r)\overline{r}'' + r\overline{r}'''$ equals $v_i = p'(r)$. Hence, the full protocol that is run in iteration i is as follows.

Construction 2 (reducing a claim about $\widehat{\alpha}_{i-1}$ to a claim about $\widehat{\alpha}_i$): *For known $\overline{r}_{i-1} \in \mathcal{F}^m$ and $v_{i-1} \in \mathcal{F}$, the* entry claim *is $\widehat{\alpha}_{i-1}(\overline{r}_{i-1}) = v_{i-1}$. The parties proceed as follows.*

1. *Applying the sum-check protocol to the entry claim, when expanded according to Eq. (3), determines $\overline{r}', \overline{r}'', \overline{r}''' \in \mathcal{F}^m$ and a value $v \in \mathcal{F}$ such that the residual claim for verification is*

$$\texttt{EQ}(\overline{r}_{i-1}, \overline{r}') \cdot \widehat{\phi}_i(\overline{r}', \overline{r}'', \overline{r}''') \cdot (1 - \widehat{\alpha}_i(\overline{r}'') \cdot \widehat{\alpha}_i(\overline{r}''')) = v. \tag{4}$$

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

3. *The verifier checks whether v equals*

$$\texttt{EQ}(\overline{r}_{i-1}, \overline{r}') \cdot \widehat{\phi}_i(\overline{r}', \overline{r}'', \overline{r}''') \cdot (1 - p'(0) \cdot p'(1)) \tag{5}$$

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

   Note that this requires evaluating $\texttt{EQ}$ and $\widehat{\phi}_i$.

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

The exit claim *is $\widehat{\alpha}_i(\overline{r}_i) = v_i$.*

The prescribes prover strategy in Construction 2 can be implemented in $\text{poly}(n)$-time. Assuming that $\widehat{\phi}_i$ can be evaluated in $\text{poly}(\log n)$-time, the verifier's strategy in Construction 2 can be implemented in $\text{poly}(\log n)$-time, provided that $\log|\mathcal{F}| \leq \text{poly}(\log n)$ (whereas we will use $|\mathcal{F}| = \text{poly}(\log n) \cdot d(n)$). 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 $1 - O(|H| \cdot m/|\mathcal{F}|)$ the exit claim is false.

We are still left with the problem of evaluating $\widehat{\phi}_i$ in $\text{poly}(\log n)$-time. Actually, in Section 3 we present a solution to a relaxed version of this problem, in which $\widehat{\phi}_i$ is replaced by an arbitrary polynomial of degree at most D that agrees with ϕ_i on H^m. Before turning to Section 3, we note that the foregoing analysis remains valid, except that the error probability grows from $O(|H| \cdot m)/|\mathcal{F}|$ to $O(D + |H| \cdot m)/|\mathcal{F}|$.

3 Evaluating $\widehat{\phi_i}$

The general problem that we face is evaluating the low degree extension of a function $\phi : H^m \to \{0,1\}$ that is computable in $O(\log n)$ space, where $|H|^m = \text{poly}(n)$. (In our setting we are interested in ϕ_i's, which range over H^{3m}, which can be treated as slices of a single $\phi : H^{3m+1} \to \{0,1\}$ such that $\phi_i(\overline{z}) = \phi(i, \overline{z})$, where $i \in [m]$ is viewed as an element of H.)[6] This problem is solved in [2] by using an auxiliary multi-round interactive protocol, but here we bypass it by making several observations.

Recall that the function ϕ that we deal with here describes the structure of arbitrary log-space uniform circuits. The first observation is that we may use instead a function that describes the structure of a highly-uniform *universal circuit*. For starters, consider a universal circuit U_n that, given a description of a poly(n)-sized circuit $C_n : \{0,1\}^n \to \{0,1\}$ of depth $d = d(n)$ and an input $x \in \{0,1\}^n$, computes $C_n(x)$; that is, $U_n(\langle C_n \rangle, x) = C_n(x)$. By using a suitable description, we may get a highly uniform circuit U_n of poly(n)-size and depth $O(\log n) \cdot d(n)$. The idea is applying the main module to U_n rather than to C_n.

The problem with the foregoing idea is that after the last iteration (i.e., the last application of Construction 2), the verifier needs to evaluate the low degree extension of the input to the circuit, whereas in our application the input to U_n consists of both $\langle C_n \rangle$ and x, which means that the verifier needs to compute $\langle C_n \rangle$. By the log-space uniformity condition, this can be done in poly(n)-time, but we wish the verifier to perform this step in $\widetilde{O}(n)$-time. So, instead, we consider a modified "universal" circuit U'_n, which is tailored to the log-space algorithm A that constructs the circuit family $\{C_n\}$. On input $x \in \{0,1\}^n$, the circuit U'_n first constructs C_n (see details below), and then computes $U_n(\langle C_n \rangle, x)$. Now, applying the main module to U'_n rather than to U_n, the verifier only needs to evaluate a low degree extension of x (as in the case that we use C_n itself).

Turning to the construction of C_n by U'_n, following [2], we use the observation that log-space uniform circuits (here $\{C_n\}_{n \in \mathbb{N}}$) can be constructed by an *highly uniform* $\mathcal{NC}$ circuit. This observation relies on the standard construction of $\mathcal{NC}$ circuits for log-space computation, a construction that is based on constructing the digraph of instantaneous configurations of the computation of a log-space machine (on a fixed input) and raising it to a power larger than its dimension (by repeated squaring). For details see Section 4.2. Recalling that U_n is also highly uniform (see Section 4.2 for details) and combining these two constructions, we obtain a highly uniform construction of U'_n.

So far, we avoided specifying what we meant by a highly uniform circuit. A notion that suffices for our purposes is presented now. Considering circuits of size $s(n) = \text{poly}(n)$ and letting $\ell = 3 \log s(n)$, we consider the predicate $\phi : \{0,1\}^\ell \to \{0,1\}$ that describes the adjacency relation of the circuit. We say that the circuit is highly uniform if ϕ can be computed by poly(ℓ)-sized formula that can be constructed in poly(ℓ)-time. The reader may verify that the $\mathcal{NC}$ circuits that we used above (for constructing the log-space uniform circuits C_n) are highly uniform; in fact, they are even "more uniform" than that. Likewise for the circuit U_n, and consequently for U'_n. Hence, the predicate $\phi : \{0,1\}^\ell \to \{0,1\}$ that describes the adjacency relation of U'_n can be computed by a poly$(\log n)$-size formula that can be constructed in poly$(\log n)$-time. Viewing ϕ as ranging over H^m, where $m = \ell / \log |H|$, the following result implies that there exists a low degree polynomial over $\mathcal{F}^m$ that agrees with ϕ on H^m and can be evaluated in poly$(\log n)$-time.[7]

[6] Recall that $m = O(\log n)/\log\log n < |H| \approx \log n$. For simplicity of notation, we write $\phi : H^m \to \{0,1\}$ rather than $\phi : H^{3m+1} \to \{0,1\}$.

[7] Recall that $|H| \approx \log n$.

Theorem 3 (evaluating a low degree polynomial that agrees with a highly uniform Boolean circuit): *For $\phi : H^m \to \{0,1\}$, where $H \equiv \{0,1\}^{\ell'}$, let $\phi' : \{0,1\}^{m\ell'} \to \{0,1\}$ be the corresponding Boolean function that views elements of H as ℓ'-bit long strings. Suppose that ϕ' is highly uniform; that is, ϕ' can be computed by formula of size $s' = \mathrm{poly}(m\ell')$ that can be constructed in $\mathrm{poly}(s')$-time. Then, there exists a polynomial $\widehat{\Phi} : \mathcal{F}^m \to \mathcal{F}$ of degree $\mathrm{poly}(s') \cdot |H|$ that agrees with ϕ on H^m and can be evaluated in $\mathrm{poly}(\ell)$-time, where $\ell = \log |H|^m$, provided that $|H| = \mathrm{poly}(m)$.*

We stress that the polynomial $\widehat{\Phi} : \mathcal{F}^m \to \mathcal{F}$ is *not* the (canonical) low degree extension of ϕ (or ϕ'); such a low degree extention has degree $m \cdot |H|$ (or $m\ell' \cdot |H|$) and exists regardless of the size of the formula ϕ. In contrast, the polynomial $\widehat{\Phi}$ is derived from the formula computing ϕ', and its degree bound is derived from the depth of ϕ'. (As stated above, we apply Theorem 3 to the function $\phi : H^{3m+1} \to \{0,1\}$ that describes the relation between gates in a circuit as in Section 2, where $\phi_i(\overline{z}) = \phi(i, \overline{z})$ for every $i \in [m] \subset H$.)

Proof: Associating H with $\{0,1\}^{\ell'}$, we first consider the computation of the j^{th} bit in the binary representation of $z \in H$. Denote by H_j the set of elements in H that are represented by ℓ'-bit strings in which the j^{th} position holds the value 1. Observe that, for every $z \in H$, it holds that the expression $\sum_{a \in H_j} \prod_{b \in H \setminus \{a\}} (z-b)/(a-b)$ (with arithmetic of the field $\mathcal{F}$) equals 1 if $z \in H_j$ (i.e., the j^{th} bit of z is 1) and equals 0 otherwise (i.e., if the j^{th} bit of z is zero).

Next, consider the $\mathrm{poly}(\ell)$-size formula that computes $\phi' : \{0,1\}^{\ell} \to \{0,1\}$, where $\ell = m\ell'$. Then, for $z_1, ..., z_m \in H$, it holds that

$$\phi(z_1, ..., z_m) = \phi'(z_{i,1}, ..., z_{i,\ell'}, ..., z_{m,1}, ..., z_{m,\ell'}), \tag{6}$$

where $z_{i,j} \leftarrow \sum_{a \in H_j} \mathtt{EQ}(z_i, a) \in \{0,1\}$ (and $\mathtt{EQ} : \mathcal{F} \times \mathcal{F} \to \mathcal{F}$ is a low degree extension of the function that tests equality over H; i.e., $\mathtt{EQ}(z,a) = \prod_{b \in H \setminus \{a\}} (z-b)/(a-b)$). Now, construct an arithmetic circuit $\widehat{\Phi}$ (which is actually a formula) that computes a polynomial that agrees with ϕ on H^m by mimicing the Boolean formula ϕ'. Specifically, the input gate $z_{i,j}$ is replaced by the formula $\sum_{a \in H_j} \mathtt{EQ}(z_i, a)$, and the $\mathtt{NAND}$ of the Boolean subformulae ϕ'_1 and ϕ'_2 is replaced by a gate that computes $1 - \widehat{\Phi}_1 \cdot \widehat{\Phi}_2$, where $\widehat{\Phi}_1$ and $\widehat{\Phi}_2$ are the corresponding arithmetic subformulae. (Indeed, we may assume, w.l.o.g., that ϕ' uses only $\mathtt{NAND}$-gates, and that its depth is $O(\log s')$.)

Note that the degree of the resulting polynomial $\widehat{\Phi}$ is upper-bounded by $\mathrm{poly}(s') \cdot |H|$, where the $|H|$ factor is due to the computation of the $z_{i,j}$'s and the $\mathrm{poly}(s')$ is exponential in the depth of the formula ϕ' (which, w.l.o.g., is logarithmic in its size). To see that $\widehat{\Phi}$ equals ϕ on H^m, note that when evaluated on input in H^m each of the $z_{i,j}$'s is assigned a Boolean value, whereas the residual subformula preserves Boolean values. Hence, the value of ϕ on a point in H^m is given by the value of ϕ' on the corresponding bit string (per Eq. (6)), which in turn equals the value of $\widehat{\Phi}$ on this bit string. ■

Wrapping-up. As stated above, we modify Construction 2 by replacing the polynomials $\widehat{\phi}_i$ by the polynomial $\widehat{\Phi}$ (i.e., replacing $\widehat{\phi}_i(\cdot)$ by $\widehat{\Phi}(i, \cdot)$ where $\widehat{\phi}_i : \mathcal{F}^{3m} \to \mathcal{F}$ and where $\widehat{\Phi}_i : \mathcal{F}^{1+3m} \to \mathcal{F}$). Doing so provides an efficient implementation of Step 3 of Construction 2, whereas the soundness error grows from $d(n) \cdot O(|H| \cdot m/|\mathcal{F}|)$ to $d(n) \cdot O((m + \mathrm{poly}(s')) \cdot |H|) \cdot |H|/|\mathcal{F}|)$, where s' is as in Theorem 3. Hence, we need to guarantee that $O((m + \mathrm{poly}(s')) \cdot |H|^2/|\mathcal{F}|) = o(1/d(n))$, where $s' = \mathrm{poly}(m \log |H|)$ is an upper bound on the size of the formula ϕ' that we use. (Note that

$s' \geq 3m \log |H|$ definitely holds, whereas $s' = O(m \log |H|)$ may actually suffice.) Recalling that $|H| \approx \log n$ and $m < \log n$, it suffices to have $|\mathcal{F}| = \omega(d(n) \cdot \text{poly}(\log n))$, which implies that $|\mathcal{F}| = \text{poly}(|H|) \cdot d(n)$ suffices. This completes the proof of Theorem 1, modulo some details provided in Section 4.

4 Details

In this section we provide some tedious details, which some readers may find quite straightforward.

4.1 Applying the sum-check protocol to Eq. (3)

Here we detail how the sum-check protocol is used to reduce the claim regarding the value of $\widehat{\alpha}_i$ at a given point $\overline{r}_{i-1}$ to a claim regarding $\widehat{\alpha}_{i-1}$. Towards this goal, we define a polynomial $p_{\overline{r}_{i-1}} : \mathcal{F}^{3m} \to \mathcal{F}$ such that

$$p_{\overline{r}_{i-1}}(\overline{z}', \overline{z}'', \overline{z}''') = \texttt{EQ}(\overline{r}_{i-1}, \overline{z}') \cdot \widehat{\phi}_i(\overline{z}', \overline{z}'', \overline{z}'') \cdot (1 - \widehat{\alpha}_i(\overline{z}'') \cdot \widehat{\alpha}_i(\overline{z}''')). \tag{7}$$

Recall that the claim that we wish to verify is $\widehat{\alpha}_{i-1}(\overline{r}_{i-1}) = v_{i-1}$. Expanding $\widehat{\alpha}_{i-1}$ according to Eq. (3) and using Eq. (7), we have

$$\widehat{\alpha}_{i-1}(\overline{r}_{i-1}) = \sum_{\overline{u},\overline{v},\overline{w} \in H^m} p_{\overline{r}_{i-1}}(\overline{u}, \overline{v}, \overline{w}). \tag{8}$$

Using $3m$ rounds of interaction, the sum-check protocol reduces the claim that the r.h.s of Eq. (8) equals v_{i-1} to a claim regarding the value of $p_{\overline{r}_{i-1}}$ at a random point $(\overline{r}', \overline{r}'', \overline{r}''') \in \mathcal{F}^{3m}$. This is captured by Step 1 of Construction 2.

4.2 The highly uniform circuits in use

Here we detail the construction of the universal circuit U'_n, which combines a construction of $\mathcal{NC}$ circuits for constructing log-space uniform circuits and a construction of the universal circuit U_n. The point that we wish to establish here is that both these constructions are highly uniform (in the sense that the adjacency relation function describing each of these two $\text{poly}(n)$-sized circuit can be computed by a $\text{poly}(\log n)$-size formula that can be computed in $\text{poly}(\log n)$-time).

We start with the highly uniform construction of $\mathcal{NC}$ circuits for constructing log-space uniform circuits, denoted $\{C_n\}_{n \in \mathbb{N}}$. Recall that this construction relies on the standard construction of $\mathcal{NC}$ circuits for log-space computations, a construction that is based on constructing the digraph of instantaneous configurations of the computation of a log-space machine (on a fixed input) and raising it to a power larger than its dimension (by repeated squaring). In our case, the input (i.e., 1^n) is actually used only to determine the space bound, and so we can ignore it. However, we should note that the said log-space computation produces many output bits (which describe the circuit C_n, rather than a single decision bit). Without loss of generality, we can incorporate the index of the next bit to be produced in the instantaneous configuration (since the log-space machine can maintain a corresponding counter).

Constructing the digraph of instantaneous configurations. Observe that the digraph that represents the consecutive configurations in the foregoing log-space computation can be constructed very easily (i.e., by a highly uniform $\mathcal{NC}$ circuit); that is, the pair $(\gamma,\gamma') \in \{0,1\}^{2\cdot O(\log n)}$ is an edge in that digraph if and only if the log-space machine moves from the instantaneous configuration represented by γ to the configuration represented by γ' in a single step. Note that we actually consider an $\mathcal{NC}$ circuit that has no input and always produces this digraph as its output. The output gate that corresponds to the entry (γ,γ') in the adjacency matrix representing this digraph is 1 if and only if the pair (γ,γ') satisfies a very simple relation that is only slightly more complex than testing equality.[8] Hence, the value of the entry (γ,γ') in the matrix can be determined in $\mathrm{poly}(\log n)$-time as a function of γ and γ'. It follows that the circuit that produces this matrix, which consists merely of computing the constants 0 and 1 and feeding them to the $\mathrm{poly}(n)$-many output bits, is highly uniform; that is, its adjacency relation can be expressed a formula of size $O(\log n)$ that can be constructed in time $\mathrm{poly}(\log n)$.

This yields a highly uniform $\mathrm{poly}(n)$-by-$\mathrm{poly}(n)$ matrix M that when raised to the power $\mathrm{poly}(n)$ yields the desired bits; that is, the value of the i^{th} output bit (i.e., the i^{th} bit in the description of C_n) equals the value of the entry $(\mathtt{s}_i,\mathtt{f}_i)$ in the resulting matrix (i.e., of $M^{\mathrm{poly}(n)}$), where $\mathtt{s}_i$ denotes the initial configuration that is used for producing the i^{th} bit and $\mathtt{f}_i$ denotes the final configuration that actually produces this bit. As we show next, there exists a highly uniform $\mathcal{NC}$ circuit for raising a given $\mathrm{poly}(n)$-by-$\mathrm{poly}(n)$ matrix to the power $\mathrm{poly}(n)$ (by repeated squaring), since the inner product of $\mathrm{poly}(n)$-bit long vectors has a highly uniform $\mathcal{NC}$ circuit.

Constructing circuits for matrix multiplication. We now detail the highly uniform $\mathcal{NC}$ circuit used to compute matrix multiplication. Let A and B be n-by-n matrices, then the $(i,j)^{\text{th}}$ bit in their product is given by $\sum_{k\in[n]} A_{i,k}B_{k,j}$. Hence, we first use n^3 gates such that the gate index by (i,j,k) computes $A_{i,k}B_{k,j}$, which means that this gate is fed by inputs with indices $(1,i,k)$ and $(2,k,j)$, and then we use n^2 trees of $n-1$ addition-gates for computing each n-way sum. Specifically, the sum that corresponds to index (i,j) is computed by gates that are indexed by (i,j,α), where $\alpha \in \{0,1\}^{\leq \log n}$, such that the gate indexed by (i,j,α) is fed by the gates indexed $(i,j,\alpha 0)$ and $(i,j,\alpha 1)$. Hence, the description of the adjacency relation of this circuit is very simple; for example, for gates that correspond to the n-way sums, we can use

$$\phi((i,j,\alpha),(i',j',\alpha'),(i'',j'',\alpha'')) = \mathtt{EQ}(ij,i'j') \wedge \mathtt{EQ}(ij,i''j'') \wedge \mathtt{EQ}(\alpha 0,\alpha') \wedge \mathtt{EQ}(\alpha 1,\alpha'').$$

Constructing the universal circuit U_n. Recall that, for fixed depth $d = d(n)$ and size $s = s(n)$, the circuit U_n is given a description of a circuit C_n (of depth d and size s) and a string $x \in \{0,1\}^n$, and is supposed to output $C_n(x)$. The circuit U_n emulates the computation of C_n in a layer-by-layer manner. Specifically, U_n computes the value of the j^{th} gate in layer i in the emulated computation of C_n (on x), denoted $v_{i,j}$, as a function of the description of the circuit C_n and the two adequate values of gates in layer $i+1$. In particular, $v_{d,j}$ is the j^{th} bit of x if $j \leq n$ and equals 0 otherwise,

[8] Specifically, recall that γ represents the contents of the work-space as well as the finite state of the machine and the location of its head (on the work-tape). The bits of γ' should equal the corresponding bits of γ, except for the bit at the head's location and the bits encoding the finite state that change according to a finite function. (The location of the head on the work-tape is indicated by the corresponding location of a special symbol in γ.) Hence, the value of the entry (γ,γ') can be determined as a conjunction of $O(\log n)$ conditions, where each of these conditions refers to a constant number of bits.

and for every $i \in [d]$ and $j \in [s]$,

$$v_{i-1,j} = \bigvee_{k',k'' \in [s]} (c_{(i-1,j),(i,k'),(i,k'')} \wedge \mathtt{NAND}(v_{i,k'}, v_{i,k''})),$$

where $c_{(i-1,j),(i,k'),(i,k'')}$ is a bit in the description of C_n that indicates whether or not the $(i-1,j)^{\text{th}}$ gate is fed by gates indexed the (i,k') and (i,k''). Again, the description of the adjacency relation of this circuit is very simple.

Acknowledgements

We are grateful to Guy Rothblum for many useful discussions regarding the Goldwasser-Kalai-Rothblum proof system.

References

[1] Oded Goldreich. On Doubly-Efficient Interactive Proof Systems. *Foundations and Trends in Theoretical Computer Science*, to appear

[2] Shafi Goldwasser, Yael Tauman Kalai, 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.

[3] 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. Earlier versions date to 1982.

[4] 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.

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

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

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il