A Note on the GKR Protocol

Justin Thaler*

Abstract

This note describes a simplification of the GKR interactive proof for circuit evaluation (Goldwasser, Kalai, and Rothblum, J. ACM 2015), as efficiently instantiated by Cormode, Mitzenmacher, and Thaler (ITCS 2012). The simplification reduces the prover runtime, round complexity, and total communication cost of the protocol by roughly 33%.

1 Introduction

Goldwasser, Kalai, and Rothblum described a powerful general-purpose interactive proof protocol [3]. This protocol is framed in the context of circuit evaluation. Given a layered arithmetic circuit $\mathcal{C}$ of depth d, size $S(n)$, and fan-in 2, the GKR protocol allows a prover to evaluate $\mathcal{C}$ with a guarantee of correctness in time $\mathrm{poly}(S(n))$, while the verifier runs in time $\tilde{O}(n + d \log S(n))$, where n is the length of the input and the $\tilde{O}$ notation hides polylogarithmic factors in n. Thus, for circuits of polynomial size and sublinear depth, the verifier's runtime is quasilinear in the input length.[1] This can be much smaller than the total size of the circuit, thereby saving the verifier substantial time relative to executing the computation locally.

Cormode, Mitzenmacher, and Thaler [2] (henceforth CMT) showed how to bring the runtime of the prover in the GKR protocol down from $\mathrm{poly}(S(n))$ to $O(S(n) \log S(n))$. They also built a full implementation of the protocol and ran it on benchmark problems. Subsequent work by Thaler [5] showed how to further reduce the prover runtime for circuits exhibiting repeated structure, such as that seen in any data parallel computation.

This note describes a refinement of the protocol of CMT. In addition to providing a conceptual simplification, the refinement reduces the prover runtime, round complexity, and total communication cost of the protocol by roughly 33% relative to the implementation of [2].

2 Notation and Background

To keep this note as self-contained as possible, this section defines interactive proofs (Section 2.1), describes the *sum-check protocol* of Lund et al. [4] (Section 2.2), provides a high-level overview of the GKR protocol (Section 2.4), and introduces necessary notation (Sections 2.3 and 2.5). Readers already familiar with the GKR protocol can safely skip to Section 3.

*Yahoo Labs

[1] In fact, the verifier in the GKR protocol can be implemented in $O(n + d \log S(n)))$ time, as shown in subsequent work (cf. [5, 6]).

2.1 Interactive Proofs

We begin by defining a valid interactive proof protocol for a function f.

DEFINITION 1 *Consider a prover $\mathcal{P}$ and verifier $\mathcal{V}$ who both observe an input x and wish to compute a function $f : \{0,1\}^n \to \mathcal{R}$ for some set $\mathcal{R}$. After the input is observed, $\mathcal{P}$ and $\mathcal{V}$ exchange a sequence of messages. Denote the output of $\mathcal{V}$ on input x, given prover $\mathcal{P}$ and $\mathcal{V}$'s random bits R, by out$(\mathcal{V}, x, R, \mathcal{P})$. $\mathcal{V}$ can output $\perp$ if $\mathcal{V}$ is not convinced that $\mathcal{P}$'s claim is valid. We say $\mathcal{P}$ is a* valid prover *with respect to $\mathcal{V}$ if for all inputs x, Pr$_R$[out$(\mathcal{V}, x, R, \mathcal{P}) = f(x)] = 1$. We say $\mathcal{V}$ a* valid verifier *for f if there is at least one valid prover $\mathcal{P}$ with respect to $\mathcal{V}$, and for all provers $\mathcal{P}'$ and all inputs x, Pr[out$(\mathcal{V}, A, R, \mathcal{P}') \notin \{f(x), \perp\}] \leq 1/3$. We say a prover-verifier pair $(\mathcal{P}, \mathcal{V})$ is a* valid interactive proof protocol *for f if $\mathcal{V}$ is a valid verifier for f and $\mathcal{P}$ is a valid prover with respect to $\mathcal{V}$. If $\mathcal{P}$ and $\mathcal{V}$ exchange r messages in total, we say the protocol has $\lceil r/2 \rceil$ rounds.*

The property that there is at least one valid prover $\mathcal{P}$ with respect to $\mathcal{V}$ is often called *completeness*, and the property that for all provers $\mathcal{P}'$ and all inputs x, Pr[out$(\mathcal{V}, A, R, \mathcal{P}') \notin \{f(x), \perp\}] \leq 1/3$ is often called *soundness*.

2.2 Sum-Check Protocol

The main technical primitive used within the GKR protocol is the sum-check protocol of Lund, Fortnow, Karloff, and Nisan [4]. We present a full description of this protocol for completeness. See also [1, Chapter 8] for a complete exposition and proof of soundness.

Suppose we are given a v-variate polynomial g defined over a finite field $\mathbb{F}$. The purpose of the sum-check protocol is to compute the sum:

$$H := \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_v \in \{0,1\}} g(b_1, \ldots, b_v).$$

In order to execute the protocol, the verifier needs to be able to evaluate $g(r_1, \ldots, r_v)$ for a randomly chosen vector $(r_1, \ldots, r_v) \in \mathbb{F}^v$.

The protocol proceeds in v rounds as follows. In the first round, the prover sends a polynomial $g_1(X_1)$, and claims that $g_1(X_1) = \sum_{x_2, \ldots, x_v \in \{0,1\}^{v-1}} g(X_1, x_2, \ldots, x_v)$. Observe that if g_1 is as claimed, then $H = g_1(0) + g_1(1)$. Also observe that the polynomial $g_1(X_1)$ has degree $\deg_1(g)$, the degree of variable x_1 in g. Hence g_1 can be specified with $\deg_1(g) + 1$ field elements. In implementations, $\mathcal{P}$ can specify g by sending the evaluation of g at each point in the set $\{0, 1, \ldots, \deg_1(g)\}$.

Then, in round $j > 1$, $\mathcal{V}$ chooses a value r_{j-1} uniformly at random from $\mathbb{F}$ and sends r_{j-1} to $\mathcal{P}$. In return, the prover sends a polynomial $g_j(x_j)$, and claims that

$$g_j(X_j) = \sum_{(x_{j+1}, \ldots, x_v) \in \{0,1\}^{v-j}} g(r_1, \ldots, r_{j-1}, X_j, x_{j+1}, \ldots, x_v). \tag{1}$$

The verifier compares the two most recent polynomials by checking $g_{j-1}(r_{j-1}) = g_j(0) + g_j(1)$, and rejecting otherwise. The verifier also rejects if the degree of g_j is too high: each g_j should have degree $\deg_j(g)$, the degree of variable x_j in g.

In the final round, the prover has sent $g_v(X_v)$ which is claimed to be $g(r_1, \ldots, r_{v-1}, X_v)$. $\mathcal{V}$ now checks that $g_v(r_v) = g(r_1, \ldots, r_v)$ (recall that we assumed $\mathcal{V}$ can evaluate g at this point). If this test succeeds (and so do all previous tests), then the verifier accepts, and is convinced that $H = g_1(0) + g_1(1)$.

PROPOSITION 1 (LUND ET AL. [4]) *Let g be a v-variate polynomial defined over a finite field $\mathbb{F}$, and let $(\mathcal{P}, \mathcal{V})$ be the prover-verifier pair in the above description of the sum-check protocol. $(\mathcal{P}, \mathcal{V})$ is a valid interactive proof protocol for the function $H = \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_v \in \{0,1\}} g(b_1, \ldots, b_v)$.*

2.3 Multilinear Extensions

For any field $\mathbb{F}$, a d-variate polynomial $p : \mathbb{F}^d \to \mathbb{F}$ is said to be *multilinear* if it has degree at most 1 in each of the d input variables. Given a function $W : \{0,1\}^d \to \{0,1\}$ whose domain is the d-dimensional Boolean hypercube, the *multilinear extension* (MLE) of W over $\mathbb{F}$, denoted $\tilde{W}$, is the unique multilinear polynomial $\mathbb{F}^d \to \mathbb{F}$ that agrees with W on all Boolean-valued inputs. That is, $\tilde{W}$ is the unique multilinear polynomial over $\mathbb{F}$ satisfying $\tilde{W}(x) = W(x)$ for all $x \in \{0,1\}^d$.

2.4 Overview of the GKR protocol

In the GKR protocol, $\mathcal{P}$ and $\mathcal{V}$ first agree on an arithmetic circuit C of fan-in 2 over a finite field $\mathbb{F}$ computing the function of interest ($\mathcal{C}$ may have multiple outputs). $\mathcal{C}$ is assumed to be in layered form, meaning that the circuit can be decomposed into layers, and wires only connect gates in adjacent layers. Suppose the circuit has depth d; we will number the layers from 1 to d with layer d referring to the input layer, and layer 1 referring to the output layer.

In the first message, $\mathcal{P}$ tells $\mathcal{V}$ the (claimed) output of the circuit. The protocol then works its way in iterations towards the input layer, with one iteration devoted to each layer. The purpose of iteration i is to reduce a claim about the values of the gates at layer i to a claim about the values of the gates at layer $i + 1$, in the sense that it is safe for $\mathcal{V}$ to assume that the first claim is true as long as the second claim is true. This is accomplished by applying the standard sum-check protocol to a certain polynomial (see Equation (2) in Section 3 for details).

More concretely, the GKR protocol starts with a claim about the values of the output gates of the circuit, but $\mathcal{V}$ cannot check this claim without evaluating the circuit herself, which is precisely what she wants to avoid. So the first iteration uses a sum-check protocol to reduce this claim about the outputs of the circuit to a claim about the gate values at layer 2 (more specifically, to a claim about an evaluation of the multilinear extension of the gate values at layer 2). Once again, $\mathcal{V}$ cannot check this claim herself, so the second iteration uses another sum-check protocol to reduce the latter claim to a claim about the gate values at layer 3, and so on. Eventually, $\mathcal{V}$ is left with a claim about the inputs to the circuit, and $\mathcal{V}$ can check this claim on her own.

2.5 Additional Notation

Suppose we are given a layered arithmetic circuit $\mathcal{C}$ of size $S(n)$, depth $d(n)$, and fan-in two. Let S_i denote the number of gates at layer i of the circuit $\mathcal{C}$. Assume S_i is a power of 2 and let $S_i = 2^{s_i}$. In order to explain how each iteration of the GKR protocol proceeds, we need to introduce several functions, each of which encodes certain information about the circuit.

To this end, number the gates at layer i from 0 to $S_i - 1$, and let $W_i : \{0,1\}^{s_i} \to \mathbb{F}$ denote the function that takes as input a binary gate label, and outputs the corresponding gate's value at layer i. The GKR protocol makes use of the multilinear extension $\tilde{W}_i$ of the function W_i.

The GKR protocol also makes use of the notion of a "wiring predicate" that encodes which pairs of wires from layer $i + 1$ are connected to a given gate at layer i in C. We define two functions, add_i and mult_i mapping $\{0,1\}^{s_i + 2s_{i+1}}$ to $\{0,1\}$ which together constitute the wiring predicate of layer i of C. Specifically, these functions take as input three gate labels (j_1, j_2, j_3), and return 1 if gate j_1 at layer i is the addition (respectively, multiplication) of gates j_2 and j_3 at layer $i+1$, and return 0 otherwise. Let $\tilde{\text{add}}_i$ and $\tilde{\text{mult}}_i$ denote the multilinear extensions of add_i and mult_i respectively.

Finally, let $\beta_{s_i}(z,p)$ denote the function

$$\beta_{s_i}(z,p) = \prod_{j=1}^{s_i} \left((1-z_j)(1-p_j) + z_j p_j\right).$$

It is straightforward to check that β_{s_i} is the multilinear extension of the function $B(x,y) : \{0,1\}^{s_i} \times \{0,1\}^{s_i} \to \{0,1\}$ that evaluates to 1 if $x = y$, and evaluates to 0 otherwise.

3 Details of CMT's Implementation of the GKR Protocol

CMT's implementation of the GKR protocol exploits the following identity. It can be shown that for any $z \in \mathbb{F}^{s_i}$,

$$\tilde{W}_i(z) = \sum_{(p,\omega_1,\omega_2)\in\{0,1\}^{s_i+2s_{i+1}}} f^{(i)}(p,\omega_1,\omega_2), \tag{2}$$

where

$$f^{(i)}(p,\omega_1,\omega_2) = \beta_{s_i}(z,p)\cdot\Big(\tilde{\text{add}}_i(p,\omega_1,\omega_2)(\tilde{W}_{i+1}(\omega_1)+\tilde{W}_{i+1}(\omega_2)) + \tilde{\text{mult}}_i(p,\omega_1,\omega_2)\tilde{W}_{i+1}(\omega_1)\cdot\tilde{W}_{i+1}(\omega_2)\Big).$$

Iteration i of CMT therefore applies the sum-check protocol to the polynomial $f^{(i)}$. Since $f^{(i)}$ is defined over $s_i + 2s_{i+1}$ variables, the sum-check protocol requires $s_i + 2s_{i+1}$ rounds. CMT show how to implement the prover in time $O(S)$ per round, assuming each addition and multiplication operation in $\mathbb{F}$ can be performed in $O(1)$ time.

4 The Refinement

Our refinement applies the sum-check protocol to a different, simpler polynomial. This cuts the number of rounds down to $2s_{i+1}$, and decreases the total communication and the prover's runtime by a similar constant factor. Moreover, it completely removes the need to use the β_{s_i} polynomial.

The key to the refinement is to derive a simpler expression for $\tilde{W}_i(z)$ than that given in Equation (2). This derivation crucially exploits the fact that the protocol is using only *multilinear* extensions $\tilde{W}_i$ of W_i, $\tilde{\text{add}}_i$ of add_i, and $\tilde{\text{mult}}_i$ of mult_i.[2]

THEOREM 2 *For any $z \in \mathbb{F}^{s_i}$,*

$$\tilde{W}_i(z) = \sum_{(\omega_1,\omega_2)\in\{0,1\}^{2s_{i+1}}} g^{(i)}(\omega_1,\omega_2), \tag{3}$$

where

$$g^{(i)}(\omega_1,\omega_2) = \Big(\tilde{\text{add}}_i(z,\omega_1,\omega_2)(\tilde{W}_{i+1}(\omega_1)+\tilde{W}_{i+1}(\omega_2)) + \tilde{\text{mult}}_i(z,\omega_1,\omega_2)\tilde{W}_{i+1}(\omega_1)\cdot\tilde{W}_{i+1}(\omega_2)\Big). \tag{4}$$

[2]More precisely, Theorem 2 holds for *any* extensions of $\text{add}_i(j_1,j_2,j_3)$ and $\text{mult}_i(j_1,j_2,j_3)$, so long as the extensions are multilinear in the variables corresponding to the *first* gate label, j_1.

PROOF. Notice that $g^{(i)}$, and hence also the right hand side of Equation (3), is a multilinear polynomial in the variables of z. We now turn to showing that the right hand and left hand sides of Equation (3) agree for all $z \in \{0,1\}^{s_i}$. This implies that the claim, because if two multilinear polynomials agree at all Boolean inputs, they must be equal as formal polynomials.

Fix any $z \in \{0,1\}^{s_i}$, and assume that gate z at layer i is an addition gate (the derivation when z is a multiplication gate is entirely analogous). Let $a, b \in \{0,1\}^{s_{i+1}}$ respectively denote the binary labels of the first and second in-neighbors of the gate with label z at layer i.

Then by definition of of $g^{(i)}$ (cf. Equation (4)),

$$\sum_{(\omega_1,\omega_2)\in\{0,1\}^{2s_{i+1}}} g^{(i)}(\omega_1,\omega_2) =$$

$$\sum_{(\omega_1,\omega_2)\in\{0,1\}^{2s_{i+1}}} \left(\tilde{\text{add}}_i(z,\omega_1,\omega_2)(\tilde{W}_{i+1}(\omega_1)+\tilde{W}_{i+1}(\omega_2)) + \tilde{\text{mult}}_i(z,\omega_1,\omega_2)\tilde{W}_{i+1}(\omega_1)\cdot\tilde{W}_{i+1}(\omega_2)\right). \tag{5}$$

Observe that $\tilde{\text{mult}}_i(z,\omega_1,\omega_2) = 0$ for all $(\omega_1,\omega_2) \in \{0,1\}^{2s_{i+1}}$. Moreover, $\tilde{\text{add}}_i(z,a,b) = 1$, and $\tilde{\text{add}}_i(z,\omega_1,\omega_2) = 0$ for all other $(\omega_1,\omega_2) \in \{0,1\}^{2s_{i+1}}$. Together, these facts imply that Expression (5) is equal to

$$\tilde{W}_{i+1}(a) + \tilde{W}_{i+1}(b). \tag{6}$$

Since $\tilde{W}_{i+1}$ extends W_{i+1}, Expression (6) is equal to

$$W_{i+1}(a) + W_{i+1}(b). \tag{7}$$

Since z is an addition gate with in-neighbors a and b, Expression (7) is equal to $W_i(z)$, which is in turn equal to $\tilde{W}_i(z)$, since $\tilde{W}_i$ extends W_i. This completes the proof.
□

By Theorem 2, rather than applying the sum-check protocol to the polynomial $f^{(i)}$ in iteration i, it suffices to apply the sum-check protocol to the polynomial $g^{(i)}$. This requires $2s_{i+1}$ rounds, since $g^{(i)}$ is defined over this many variables. The prover's computation in each round of the protocol is essentially identical to the prover's computation in the last $2s_{i+1}$ rounds of the sum-check protocol applied to $f^{(i)}$ as in CMT. The claimed efficiency improvements over CMT (in rounds, total communication, and prover time) follow.

References

[1] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[2] G. Cormode, M. Mitzenmacher, and J. Thaler. Practical verified computation with streaming interactive proofs. In *ITCS*, pages 90-112, 2012.

[3] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. Delegating computation: interactive proofs for muggles. In *STOC*, pages 113–122, 2008.

[4] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39:859–868, 1992.

[5] J. Thaler. Time-Optimal Interactive Proofs for Circuit Evaluation. In *CRYPTO*, pages 71–89, 2013.

[6] V. Vu, S. Setty, A. J. Blumberg, and M. Walfish. A hybrid architecture for interactive verifiable computation. Pre-print, November 2012. In *IEEE Symposium on Security and Privacy (Oakland)*, May 2013.