GKR
Chap 4 Interative Proofs

Pushpendra Pal

June 2025

Reading notes from Justin Thaler's "Proofs Argument and Zero Knowledge"

Keywords: Interactive Proofs, Wiring predicate, Multiple polynomial evaluation reduction, Multi-Linear Extension, Sum-check protocol

1 Prerequisites

1.1 Reducing multiple polynomial evaluation to one

1.1.1 Need

Take the case of counting triangles in a graph; there we had a function in the MLE version.

$$\tilde{g}(Graph) = \frac{1}{6}\Sigma_i\Sigma_j\Sigma_k\tilde{f}(i,j).\tilde{f}(j,k).\tilde{f}(k,i) \tag{1}$$

Here, we need to evaluate $\tilde{f}$ at three points each time. How to do better?

1.1.2 Idea

- let us begin by supposing that $\tilde{W}$ is a MLE polynomial over F with log n variables, and the verifier wishes to evaluate $\tilde{W}$ at just two points, say b, c ϵ F^{logn}
- A simple one round IP with communication cost O(log n) that reduces the evaluation of $\tilde{W}(b)$ and $\tilde{W}(c)$ to the evaluation of $\tilde{W}(r)$ for a single point r ϵ F^{logn} .
- What this means is that the protocol will force the prover P to send claimed values v_0 and v_1 for $\tilde{W}(b)$ and $\tilde{W}(c)$, as well as claimed values for many other points chosen by the verifier V in a specific manner.
- V will then pick r at random from those points, and it will be safe for V to believe that $v_0 = \tilde{W}(b)$ and $v_1 = \tilde{W}(c)$ as long as P's claim about $\tilde{W}(r)$ is valid.
- In other words, the protocol will ensure that if either $v_0 \neq \tilde{W}(b)$ or $v_1 \neq \tilde{W}(c)$, then with high probability over the V's choice of r, it will also be the case that the prover makes a false claim as to the value of $\tilde{W}(r)$.

1.2 Protocol

- l : F $\rightarrow$ F^{logn} be a line passing through b and c such that l(0) = b and l(1)= c.
- P sends a uni-variate polynomial q (deg $\leq$ logn) claimed to be $\widetilde{W}$o l, restricting $\widetilde{W}$ to l.
- V interprets q(i) = v_i, where $v_0 = \widetilde{W}(b)$ and $v_0 = \widetilde{W}(b)$.

1.3 Example

- Suppose that log n = 2, b = (2, 4), c = (3, 2), and $\widetilde{W}(x_1, x_2) = 3x_1x_2 + 2x_2$.
- Line l is unique with $l(0)$ = b and $l(1)$ = c with paramatrix form $t \rightarrow (t + 2, 4 - 2t)$.
- The restriction of $\widetilde{W}$ to $l = \widetilde{W}$ o $l = 3(t + 2)(4 - 2t) + 2(4 - 2t) = -6t^2 - 4t + 32$.
- **If P sends a degree-2 uni-variate polynomial q claimed to equal $\widetilde{W}$ o l, the verifier will interpret q(0) and q(1) as claims about $\widetilde{W}(b)$ and $\widetilde{W}(c)$ respectively.**

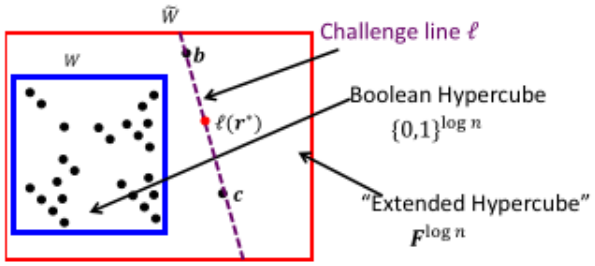

Figure 1:

- The verifier V will then pick a random r* ϵ F, set r = l(r*) and interpret q(r*) as the claimed value of $\widetilde{W}(r)$.
- Observe that $l(r) = (r + 2, 4\text{-}2r)$ is a random point on the line l.

2 GKR protocol

2.1 Objective

The verifier needs to know that a circuit has been evaluated correctly, without him actually performing the circuit evaluation. The verifier does know the circuit and the input values but is lazy enough to not do the evaluation himself.

2.2 Idea

- The prover takes a layered circuit in which there is only connection between the adjacent layers, and claims the values at each level one by one in each round.
- The circuit is known to both the verifier and the prover.
- The values at each level are encoded in an MLE of all the values with their position (W_i, where i is the level starting from the top 0). There could be multiple outputs at the top level also. The last level contains the inputs, so the Prover in the last round would send the MLE of all the inputs.
-

$$\widetilde{W}_i(r_i) = f^i_{r_i} = \widetilde{add}_i(r_i, b, c).[\widetilde{W}_{i+1}(b) + \widetilde{W}_{i+1}(c)] + \widetilde{mult}_i(r_i, b, c).[\widetilde{W}_{i+1}(b).\widetilde{W}_{i+1}(c)] \tag{2}$$

2.3 Wiring predicate

The gates are also encoded as functions of three variables, $r_i, b, c.r_i$ is the **output position** and b, c are the input **positions, not the input** to the specific gate. r_i belongs to the higher layer (closer to the root layer), while b,c belong to the lower layer. The positions are typically encoded as binary.
$add(r_i, b, c)$ = 1if wires are connected and 0 otherwise.

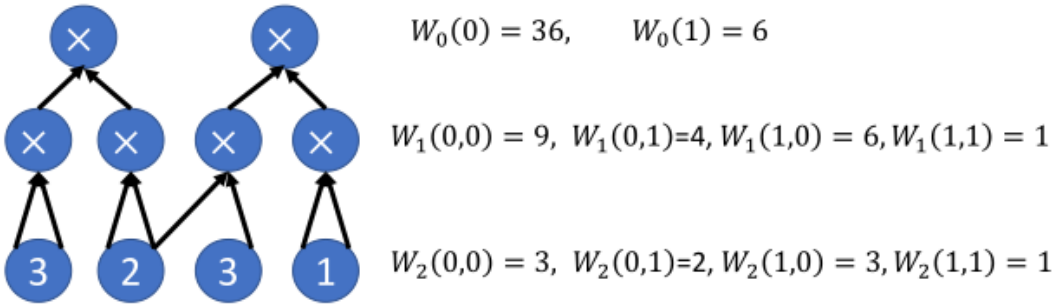

Figure 4.12: Example circuit C and input x, and resulting functions W_i for each layer i of C. Note that C has two output gates.

Similarly, mult_1 is a function on domain $\{0,1\}^2 \times \{0,1\}^2 \times \{0,1\}^2$. It evaluates to 0 on all inputs except for the following four, on which it evaluates to 1:

- $((0,0),(0,0),(0,0))$.
- $((0,1),(0,1),(0,1))$.
- $((1,0),(0,1),(1,0))$.
- $((1,1),(1,1),(1,1))$.

Note that for each layer i, add_i and mult_i depend only on the circuit C and not on the input x to C. In contrast, the function W_i *does* depend on x. This is because W_i maps each gate label at layer i to the value of the gate when C is evaluated on input x.

2.4 Protocol

Description of the GKR protocol, when applied to a layered arithmetic circuit C of depth d and fan-in two on input $x \in \mathbb{F}^n$. Throughout, k_i denotes $\log_2(S_i)$ where S_i is the number of gates at layer i of C.

- At the start of the protocol, $\mathcal{P}$ sends a function $D: \{0,1\}^{k_0} \to \mathbb{F}$ claimed to equal W_0 (the function mapping output gate labels to output values).
- $\mathcal{V}$ picks a random $r_0 \in \mathbb{F}^{k_0}$ and lets $m_0 \leftarrow \tilde{D}(r_0)$. The remainder of the protocol is devoted to confirming that $m_0 = \tilde{W}_0(r_0)$.
- **For** $i = 0, 1, \ldots, d-1$:
  - Define the $(2k_{i+1})$-variate polynomial

$$f_{r_i}^{(i)}(b,c) := \widetilde{\text{add}}_i(r_i,b,c)\left(\tilde{W}_{i+1}(b) + \tilde{W}_{i+1}(c)\right) + \widetilde{\text{mult}}_i(r_i,b,c)\left(\tilde{W}_{i+1}(b) \cdot \tilde{W}_{i+1}(c)\right).$$

  - $\mathcal{P}$ claims that $\sum_{b,c \in \{0,1\}^{k_{i+1}}} f_{r_i}^{(i)}(b,c) = m_i$.
  - So that $\mathcal{V}$ may check this claim, $\mathcal{P}$ and $\mathcal{V}$ apply the sum-check protocol to $f_{r_i}^{(i)}$, up until $\mathcal{V}$'s final check in that protocol, when $\mathcal{V}$ must evaluate $f_{r_i}^{(i)}$ at a randomly chosen point $(b^*, c^*) \in \mathbb{F}^{k_{i+1}} \times \mathbb{F}^{k_{i+1}}$. See Remark (a) at the end of this codebox.
  - Let ℓ be the unique line satisfying $\ell(0) = b^*$ and $\ell(1) = c^*$. $\mathcal{P}$ sends a univariate polynomial q of degree at most k_{i+1} to $\mathcal{V}$, claimed to equal $\tilde{W}_{i+1}$ restricted to ℓ.
  - $\mathcal{V}$ now performs the final check in the sum-check protocol, using $q(0)$ and $q(1)$ in place of $\tilde{W}_{i+1}(b^*)$ and $\tilde{W}_{i+1}(c^*)$. See Remark (b) at the end of this codebox.
  - $\mathcal{V}$ chooses $r^* \in \mathbb{F}$ at random and sets $r_{i+1} = \ell(r^*)$ and $m_{i+1} \leftarrow q(r_{i+1})$.
- $\mathcal{V}$ checks directly that $m_d = \tilde{W}_d(r_d)$ using Lemma 3.8.

  Note that $\tilde{W}_d$ is simply $\tilde{x}$, the multilinear extension of the input x when x is interpreted as the evaluation table of a function mapping $\{0,1\}^{\log n} \to \mathbb{F}$.

Remark a. Note that $\mathcal{V}$ does not actually know the polynomial $f_{r_i}^{(i)}$, because $\mathcal{V}$ does not know the polynomial $\tilde{W}_{i+1}$ that appears in the definition of $f_{r_i}^{(i)}$. However, the sum-check protocol does not require $\mathcal{V}$ to know anything about the polynomial to which it is being applied, until the very final check in the protocol (see Remark 4.2).

Remark b. We assume here that for each layer i of C, $\mathcal{V}$ can evaluate the multilinear extensions $\widetilde{\text{add}}_i$ and $\widetilde{\text{mult}}_i$ at the point (r_i, b^*, c^*) in polylogarithmic time. Hence, given $\tilde{W}_{i+1}(b^*)$ and $\tilde{W}_{i+1}(c^*)$, $\mathcal{V}$ can quickly evaluate $f_{r_i}^{(i)}(b^*, c^*)$ and thereby perform its final check in the sum-check protocol applied to $f_{r_i}^{(i)}$.

Figure 4.13: Self-contained description of the GKR protocol for arithmetic circuit evaluation.

3 Questions

1. Complexity of prover and verifier?
   ANS.

Communication	Rounds	$\mathcal{V}$ time	$\mathcal{P}$ time	Soundness error
$d \cdot \text{polylog}(S)$ field elements	$d \cdot \text{polylog}(S)$	$O(n + d \cdot \text{polylog}(S))$	$\text{poly}(S)$	$O(d \log(S)/\lvert\mathbb{F}\rvert)$

Table 4.4: Costs of the original GKR protocol [GKR08] when applied to any log-space uniform layered arithmetic circuit $\mathcal{C}$ of size S and depth d over n variables defined over field $\mathbb{F}$. Section 4.6.5 describes methods from [CMT12] for reducing $\mathcal{P}$'s runtime to $O(S \log S)$, and reducing the polylog(S) terms in the remaining costs to $O(\log S)$. It is now known how to achieve prover runtime of $O(S)$ for arbitrary layered arithmetic circuits $\mathcal{C}$ (see Remark 4.5).

$\mathcal{V}$'s runtime. Observe that the polynomial $f_{r_i}^{(i)}$ defined in Equation (4.18) is a $(2k_{i+1})$-variate polynomial of degree at most 2 in each variable, and so the invocation of the sum-check protocol at iteration i requires $2k_{i+1}$ rounds, with three field elements transmitted per round. Thus, the total communication cost is $O(S_0 + d \log S)$ field elements where S_0 is the number of outputs of the circuit. The time cost to $\mathcal{V}$ is $O(n + d \log S + t + S_0)$, where t is the amount of time required for $\mathcal{V}$ to evaluate $\widetilde{\text{add}}_i$ and $\widetilde{\text{mult}}_i$ at a random input, for each layer i of $\mathcal{C}$. Here the n term is due to the time required to evaluate $\widetilde{W}_d(r_d)$, the S_0 term is the time required to read the vector of claimed outputs and evaluate the corresponding multilinear extension, the $d \log S$ term is the time required for $\mathcal{V}$ to send messages to $\mathcal{P}$ and process and check the messages from $\mathcal{P}$. For now, let us assume that t is a low-order cost and that $S_0 = 1$, so that $\mathcal{V}$ runs in total time $O(n + d \log S)$; we discuss this issue further in Section 4.6.6.

Round complexity and communication cost. By direct inspection of the protocol description, there are $O(d \log S)$ rounds in the GKR protocol, and the total communication cost is $O(d \log S)$ field elements.

$\mathcal{P}$'s runtime. Analogously to the MATMULT protocol of Section 4.4, we give two increasingly sophisticated implementations of the prover when the sum-check protocol is applied to the polynomial $f_{r_i}^{(i)}$.
Method 1: $f_{r_i}^{(i)}$ is a v-variate polynomial for $v = 2k_{i+1}$. As in the analysis of Method 1 for implementing the prover in the matrix multiplication protocol from Section 4.4, $\mathcal{P}$ can compute the prescribed method in round j by evaluating $f_{r_i}^{(i)}$ at $3 \cdot 2^{v-j}$ points. It is not hard to see that $\mathcal{P}$ can evaluate $f_{r_i}^{(i)}$ at any point in $O(S_i + S_{i+1})$ time using techniques similar to Lemma 3.8. This yields a runtime for $\mathcal{P}$ of $O(2^v \cdot (S_i + S_{i+1}))$ Over all d layers of the circuit, $\mathcal{P}$'s runtime is bounded by $O(S^3)$.

2. What if the S_i not a power of 2, let's say S_i=3, then what should the k_i be? Would it be $\lceil log_2 S_i \rceil$, or something else?
   ANS. It is only needed to define the variables and find their MLE, and MLE require boolan field, so I think the ceiling function should be reasonable enough.

3. Why is the function f is $2k_{i+1}$ variate polynomial?
   ANS. Because f is defined over 2 variables of the lower layer, each of which requires atleast $log_2 S_{i+1} = k_{i+1}$ bits to be represented.

4. If the Verifier does not know the function $f_{r_i}^i$, how does he do the final check in the protocol,because the Σ-check protocol requires evaluation of f at a random point?
   ANS. Using the reduction of multiple poynomial evaluations to one explained in prerequisites.

5. Why do we need the MLE of add and mult? Is it just so because we have changed the domain of W for $\widetilde{W}$ and add/mult and $\widetilde{W}$ have the same input variables? Is that the only reason or are there others?
   ANS. The [Eq. 2] is valid using any extension of add_i and $mult_i$, htat are multilinear in the first k_i variables, where $k_i = log_2 S_i$. But the MLE of add and mult are critical to achieving a prover runtime that is nearly **linear** in S (circuit size, total number of gates),

6. The verifier does know the circuit and the inputs but does he know the intermediate gate vlaues?
   ANS. No, he gets a MLE of all the values at a particular level

7. Comparing the PLONK permutation check and the wiring predicate here, they certainly are doing the same job of storing info about the circuit wiring? What does PLONK permutation check doesn't do?
   ANS. PLONK permutation check can't store the operation itself, it just says which variable is reused again.

8. In the beginning, the P and V agree on a log space uniform arithmetic circuit C of fan-in 2 over a finite field F. What is a log space uniform algorithm?
   ANS.

evaluation problem. In this problem, $\mathcal{V}$ and $\mathcal{P}$ first agree on a *log-space uniform* arithmetic circuit $\mathcal{C}$ of fan-in 2 over a finite field $\mathbb{F}$, and the goal is to compute the value of the output gate(s) of $\mathcal{C}$. A log-space uniform circuit $\mathcal{C}$ is one that possesses a succinct implicit description, in the sense that there is a logarithmic-space algorithm that takes as input the label of a gate a of $\mathcal{C}$, and is capable of determining all relevant information about that gate. That is, the algorithm can output the labels of all of a's neighbors, and is capable of determining if a is an addition gate or a multiplication gate.

9. How is the function $\mathrm{D}(r_0)$ sent by the prover at the beginning claimed to be equal to the $\mathrm{W}(r_0)$?
ANS. The verifier finds the MLE of D, $\tilde{D}$, and picks a random $r_0 \, \epsilon \, F^{k_0}$, and evaluate $\tilde{D}(r_0)$ in time $\mathrm{O}(S_0)$. Schwartz Zippel lemma if $\tilde{D}(r_0) = \tilde{W}(r_0)$, then it safe to assume that these are the same polynomail. Remember that the MLEs are disatnce ampilfying algorithms, if two function differ even slightly, they would differ almost everywhere in their MLE.

10. who finds the $add(r, b, c)$ and $mult(r, b, c)$, the oriver ir the verifier? **ANS**. Verifier, bocoz the circuit is known to both so why would orover do so.