GKR Protocol

Notes by 6.5610 Staff

MIT - 6.5610
Lecture 18 (April 13, 2026)

> ***Warning:*** This document is a rough draft, so it may contain bugs. Please feel free to email me with corrections.

Outline

- Doubly efficient interactive proofs
- Warmup for the GKR protocol
- Low-degree Extension
- The GKR protocol

Doubly Efficient Interactive Proofs

The definition of interactive proofs places no restrictions on the prover's runtime, and restricts only the verifier's runtime. Indeed, when interactive proofs were original defined they referred to the prover as Merlin (an all powerful wizard). For example, in the $\mathsf{IP} = \mathsf{PSPACE}$ of Shamir [2] the prover runs in time $\geq 2^{S \cdot \log T}$ to prove the correctness of a time-T and space-S computation.

In reality, however, we do care about the computational power of the prover. Of course, we still need to allow the prover more computational power than the verifier, as otherwise the prover is not helpful.

Definition 1 (Doubly-Efficient Interactive Proof (DE-IP))**.** A doubly-efficient interactive proof for a language $L \in \mathsf{DTIME}(T(n))$ is an interactive proof such that:

1. The honest prover's runtime is $\mathrm{poly}(T)$.
2. The verifier's runtime is much less, ideally $\mathrm{polylog}(T) + \tilde{O}(n)$, where $\tilde{O}$ omits $\mathrm{polylog}(n)$ factors.

In practice it is desirable that the prover's runtime is $O(T)$.

We will show how to use the Sumcheck protocol to construct a doubly efficient interactive proof for every bounded depth computation.

Theorem 2. *For any circuit C of depth D and size S (that is log-space uniform) there exists a doubly efficient interactive proof such that*

- *The number of rounds is* $D \cdot \text{polylog}(S)$.
- *The communication complexity is* $D \cdot \text{polylog}(S)$.
- *The verifier's runtime is* $\tilde{O}(n) + D \cdot \text{polylog}(S)$ *where n is the input length (assuming the circuit is log-space uniform)*
- *The prover's runtime is* $\text{poly}(S)$.

We will explain what the log-space uniformity condition is when we describe the GKR protocol.

The doubly efficient interactive proof that achieves this theorem is called the GKR protocol [1]. The **only** ingredient used in the GKR protocol is the Sumcheck protocol!

Intuition for the GKR protocol

Given a circuit C of depth D and size S, an input $x \in \{0,1\}^n$ and an output y, the prover needs to convince the verifier that $C(x) = y$. In other words, the verifier wants to catch the prover if y is incorrect. Here is a simple idea: The verifier will ask the prover for the value of the two children corresponding to the output gate, and will check consistency with y. Note that if the values are consistent and y is false, then the value of at least one of its children must be false. The verifier will guess which one is false randomly, and will continue this process until a leaf x_i is reached. Note that if y is false, and every time the verifier guesses correctly who the false child is, then at the end of this protocol the verifier will recieve a false value of x_i and thus the prover will be rejected!

We assume throughout that the fanin of every gate is ≤ 2.

This interactive proof is extremely simple, but its soundness guarantee is pathetic! The soundness is $1 - 2^{-D}$, since it will catch the prover cheating only if in each and every layer it guesses correctly who the false child is. This happens with probability 2^{-D}. The GKR protocol follows this blue-print but achieves better soundness since it does this over an error correcting code.

Analogy to the Sumcheck protocol

Recall that in the Sumcheck protocol, given a polynomial $f : \mathbb{F}^m \to \mathbb{F}$ of degree $\leq d$ in each variable and a fixed set $H \subseteq \mathbb{F}$, the prover convinces the verifier that

$$\sum_{h_1,\ldots,h_m \in H} f(h_1, \ldots, h_m) = \beta.$$

One way to do this is to have the prover send the $|H|$ values $\beta_{1,h}$ where

$$\beta_{1,h} = \sum_{h_2,\ldots,h_m \in H} f(h, h_2, \ldots, h_m).$$

The verifier will check that $\beta = \sum_{h \in H} \beta_{1,h}$ and then will choose at random $t_1 \xleftarrow{R} H$ and will reduce it to a Sumcheck on $m-1$ variables of the statement

$$\beta_{1,t_1} = \sum_{h_2,\ldots,h_m \in H} f(t_1, h_2 \ldots, h_m) = \beta_1$$

Assume that β is false. The prover can cheat on exactly one value of $\beta_{1,h}$. The verifier samples one random $\beta_{1,h}$, and the chance of hitting the bad one is $\frac{1}{|H|}$. Therefore, this creates a large loss in soundness (similarly to the simplified GKR protocol presented above). The main idea behind the Sumcheck protocol is to use the fact that f is of low degree, which intuitively allows the prover to check that all the values in H are correct simultaneously. Specifically, the verifier chooses $t \xleftarrow{R} \mathbb{F}$ and uses the fact that f is a low-degree polynomial to argue that in each round the verifier chooses a "false" t_i with probability $\geq 1 - \frac{d}{|\mathbb{F}|}$. So, by relying on the fact that f is a low-degree polynomial we managed to reduce the loss significantly!

At first it may be unclear how we use this for the GKR protocol since we do not have any low-degree function f there. However, the Sumcheck protocol can be used to prove that

$$\sum_{h_1,\ldots,h_m \in H} f(h_1, \ldots, h_m) = \beta.$$

for *any* $f : H^m \to H$, where f is not necessarily low degree. Namely, we can use the Sumcheck protocol to prove that for *any* function $f : H^m \to H$ it holds that

$$\sum_{h_1,\ldots,h_m \in H} f(h_1, \ldots, h_m) = \beta.$$

This can be done using the concept called *low-degree extension*.

Low-Degree Extension

A low degree extension (LDE) of a function (not necessarily a polynomial) $f : H^m \to \{0,1\}$ is a *polynomial* function $\tilde{f} : \mathbb{F}^m \to \mathbb{F}$ of (minimal) degree $\leq |H| - 1$ in each variable that agrees with f on all inputs in the range H^m; i.e., $\forall h \in H^m, \tilde{f}(h) = f(h)$, or more concisely $\tilde{f}\,|_{H^m} \equiv f$. Notice that the domain of $\tilde{f}$ is $\mathbb{F}^m$, a superset of H^m, hence the name "extension".

One can think of f as an arbitrary string of length $|H|^m$.

Theorem 3. *Let $f : H^m \to \{0,1\}$ be any function (i.e., an arbitrary sequence of $|H|^m$ bits). Let $\mathbb{F}$ be any finite field s.t. $H \subseteq \mathbb{F}$. Then, there exists a unique $\tilde{f} : \mathbb{F}^m \to \mathbb{F}$ s.t. $\tilde{f}\,|_{H^m} \equiv f$ and the degree of every variable in $\tilde{f}$ is at most $|H| - 1$. Moreover, $\tilde{f}$ can be computed in time $|H|^m \cdot \text{poly}(m, |H|)$.*

This theorem is a generalization of Langrange interpolation to the multi-variate setting. When $m = 1$, the LDE $\tilde{f} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial, which follows from *Lagrange Interpolation*.

The univariate case: For any $f : H \to \{0,1\}$, we can write

$$\tilde{f}(x) = \sum_{h \in H} f(h)\chi_h(x)$$

where χ_h satisfies that for every $x \in H$:

The function χ_h is known as the Kronecker delta function.

$$\chi_h(x) = \begin{cases} 1 & x = h \\ 0 & x \neq h \end{cases}$$

From Lagrange interpolation, we have an explicit formula for $\chi_h(\cdot)$ as a degree $|H| - 1$ polynomial function of x:

$$\chi_h(x) = \prod_{h' \in H \setminus \{h\}} \frac{h' - x}{h' - h}. \tag{1}$$

The LDE theorem (Theorem 3) is a multi-variate version of this.

Note that $\chi_h(x)$ is efficiently computable since it is a degree $|H| - 1$ polynomial. Moreover, as we will see, it is also efficiently computable (and low degree) as a function of both h and x. Namely, the function $\widetilde{\beta}(h, x) = \chi_h(x)$ is low-degree (and thus efficiently computable).

Now consider the m-dimensional hypercube H^m. For a point $p = (p_1, \ldots, p_m) \in H^m$, define the *multivariate* basis polynomial

$$\chi_p : \mathbb{F}^m \to \mathbb{F}$$

by

$$\chi_p(x_1, \ldots, x_m) := \prod_{i=1}^{m} \chi_{p_i}(x_i).$$

Each factor $\chi_{p_i}(x_i)$ is univariate, and the product enforces coordinate-wise equality with p.

Given any function $V : H^m \to \mathbb{F}$, its *low-degree extension* $\widetilde{V} : \mathbb{F}^m \to \mathbb{F}$ is defined by

$$\widetilde{V}(x_1, \ldots, x_m) := \sum_{p \in H^m} V(p)\, \chi_p(x_1, \ldots, x_m).$$

This is a polynomial of degree $< |H|$ in each variable, and it agrees with V on all points of H^m.

The GKR protocol

Fix boolean circuit $C : \{0,1\}^n \to \{0,1\}$ of size (number of gates) S and depth D. The GKR protocol is an interactive proof for the fact that $C(x) = 1$. We assume that the verifier has a succinct description of C. Formally, we assume that C is log-space uniform i.e., it can be generated by some log-space Turing Machine M, and we assume that the verifier has a description of M.

A log-space Turing machine is a Turing machine that is allowed to use only $O(\log n)$ space on its work tape, where n is the size of the input. The verifier in GKR must run in polylogarithmic time. It cannot store the whole circuit. It must be able to compute local wiring information on the fly. Hence, the restriction.

Assume without loss of generality that C is layered which means that each gate belongs to a layer, and each gate in layer i is connected by neighbors only in layer $i+1$. Let layer 0 denotes the output layer and D denotes the input layer.

Recall the intuitive protocol above, where we reduce a claim about the value of a gate in layer i to a claim about the value of a gate in layer $i+1$. The GKR protocol follows this blueprint. The protocol consists of D-subprotocols, where a claim about the (joint) values of gates in layer i is converted to a claim about the (joint) values of gates in layer $i+1$. Eventually, it will be reduced to a claim about the input values (layer d), which are known to the verifier.

One can always layer a circuit by adding dummy intermediate gates. This can be done while increasing the depth to depth at most D^2.

Detailed description of the protocol

Step 1: Arithmetize C. Convert C to a (layered) arithmetic circuit (over $\mathsf{GF}[2]$) with fan-in 2. Arithmetic circuit (over $\mathsf{GF}[2]$) means that it consists only of gates of the form ADD and MULT (where addition and multiplication are done modulo 2). We can convert any Boolean circuit, with gates $\wedge$ and $\neg$, into an arithmetic circuit, by converting a gate $\wedge$ into a gate MULT, and converting a gate $\neg$ into a gate ADD where we add a constant 1 as an input to the gate.

Step 2: Pick a subset $H \subseteq \mathbb{F}$ and an integer m such that $S = |H|^m$. This way, we can give each of the S gates in a given layer a unique label encoded in H^m. A natural choice is

$$H = \{0,1\} \quad \text{and} \quad m = \log S.$$

A less natural choice but a common one is

$$H = \{0,1,\ldots,\log S - 1\} \quad \text{and} \quad m = \frac{\log S}{\log\log S}.$$

Jumping ahead the reason for the latter choice is that one can take a field $\mathbb{F}$ of size $\text{poly}(|H|)$ that contains H so that

$$|\mathbb{F}| >> m \cdot |H| = \frac{\log S}{\log\log S} \cdot \log S \implies |\mathbb{F}| = \text{poly}(\log S). \qquad (2)$$

Meaning

$$|\mathbb{F}^m| = \text{poly}(\log S)^{\frac{\log S}{\log \log S}} = S^{O(1).} \tag{3}$$

This cannot be done with the natural choice of $H = \{0,1\}$ since then we need to take $|\mathbb{F}| \gg \log S$, which results in $|\mathbb{F}^m| = (\log S)^{\log S} = S^{\log \log S}$ which is super-polynomial.

Step 3: The prover computes the values of all gates in every layer of the circuit. For layer i, define the function $V_i : H^m \to \{0,1\}$ as the mapping from an encoding of a gate label to the value of the gate, and $\tilde{V}_i : \mathbb{F}^m \to \mathbb{F}$ be its low-degree extension (LDE), which is the unique function of degree $\leq |H| - 1$ in each variable that agrees with V_i on inputs in H^m.

The Protocol: The protocol consists with D "reduction" protocols, where each reduction protocol reduces a claim of the form $\tilde{V}_i(z_i) = v_i$ about layer i to a claim of the form $\tilde{V}_{i+1}(z_{i+1}) = v_{i+1}$ about layer $i+1$. We start with the output layer where the prover claims that $\tilde{V}_0(z_0) = v_0 = 1$ where $z_0 \in H^m$ is the label of the only non-dummy gate in layer 0 that holds the output of the circuit. At the end of these D reduction protocols we will be left with a claim of the form $\tilde{V}_d(z_d) = v_d$. The verifier can check this on its own since it knows V_d from its input values and thus can compute its LDE on its own.

Actually, the reduction protocol will reduce checking two such claims about layer i to two such claims about layer $i+1$.

The reduction protocol

For every $i \in [D]$ we define two indicator functions $\mathsf{ADD}_i, \mathsf{MULT}_i : (H^m)^3 \to \{0,1\}$ as follows:

$$\mathsf{ADD}_i(p, w_1, w_2) = \begin{cases} 1 & \text{gate } p \text{ in layer } i \text{ is an ADD gate connecting } w_1 \text{ and } w_2 \text{ in layer } i+1 \\ 0 & \text{Otherwise} \end{cases}$$

MULT_i is defined similarly with ADD replaced with MULT in the definition. Let

$$\widetilde{\mathsf{ADD}_i}, \widetilde{\mathsf{MULT}_i} : \mathbb{F}^{3m} \to \mathbb{F}$$

be the LDEs of ADD_i and MULT_i, respectively.

We can expand out the LDE definition and rewrite the claim $\tilde{V}_i(z_i) = v_i$ as

$$v_i = \tilde{V}_i(z_i) = \sum_{p \in H^m} V_i(p) \chi_p(z_i)$$

Then, we can further express $V_i(p)$ as combination of indicators $\widetilde{\mathsf{ADD}}_i, \widetilde{\mathsf{MULT}}_i$:

$$v_i = \sum_{p \in H^m} \sum_{w_1, w_2 \in H^m} \Big[\widetilde{\mathsf{ADD}}_i(p, w_1, w_2)(\tilde{V}_{i+1}(w_1) + \tilde{V}_{i+1}(w_2)) + \widetilde{\mathsf{MULT}}_i(p, w_1, w_2)(\tilde{V}_{i+1}(w_1) \cdot \tilde{V}_{i+1}(w_2))\Big] \chi_p(z_i)$$

This almost looks like a claim that one can run Sumcheck on. However, recall that the Sumcheck protocol only works if the expression inside the sum is a low-degree (multi-variate) polynomial, so we need to argue that $\chi_p(z_i)$ is a low-degree polynomial in p. Note that the $\tilde{V}$'s are by definition LDEs.

Indeed, it turns that that we can compute the Kronecker delta polynomials $\chi_p(z_i)$ via a low-degree polynomial, as follows. Define $\beta : H \times H \to \{0,1\}$ as $\beta(a,b) = \chi_a(b)$, and let $\widetilde{\beta} : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$ be its LDE. For every fixed $a \in H$, we have that the univariate polynomial $\widetilde{\beta}(a, \cdot)$ is a LDE/Lagrange Interpolation of $\beta(a, \cdot)$. In addition, $\chi_a(\cdot)$ is also a LDE of $\beta(a, \cdot)$. Then, by the uniqueness of LDE, we have that $\widetilde{\beta}(a,b) = \chi_a(b)$ for all $a \in H, b \in \mathbb{F}$.

We can thus rewrite v_i as

$$v_i = \sum_{p, w_1, w_2 \in H^m} \Big[\widetilde{\mathsf{ADD}}_i(p, w_1, w_2)(\tilde{V}_{i+1}(w_1) + \tilde{V}_{i+1}(w_2)) + \widetilde{\mathsf{MULT}}_i(p, w_1, w_2)(\tilde{V}_{i+1}(w_1) \cdot \tilde{V}_{i+1}(w_2))\Big] \widetilde{\beta}(p, z_i)$$

Denote the polynomial inside the sum by

$$f_{i,z_i}(p, w_1, w_2) = \Big[\widetilde{\mathsf{ADD}}_i(p, w_1, w_2)(\tilde{V}_{i+1}(w_1) + \tilde{V}_{i+1}(w_2)) + \widetilde{\mathsf{MULT}}_i(p, w_1, w_2)(\tilde{V}_{i+1}(w_1) \cdot \tilde{V}_{i+1}(w_2))\Big] \widetilde{\beta}(p, z_i)$$

Thus, after the last round in Sumcheck for

$$v_i = \sum_{p, w_1, w_2 \in H^m} f_{i,z_i}(p, w_1, w_2),$$

the verifier must be able to compute $f_{i,z_i}(z_{i+1,0}, z_{i+1,1}, z_{i+1,2})$ for some random $z_{i+1,0}, z_{i+1,1}, z_{i+1,2} \in \mathbb{F}^m$ chosen by the verifier. We assume for now that the verifier can compute on its own $\widetilde{\mathsf{ADD}}_i$ and $\widetilde{\mathsf{MULT}}_i$. This is precisely where we use the log-space uniformity condition of the underlying circuit C.

So we reduced checking the value v_i in layer i to checking the value of two elements in round $i+1$. It seems like if we continue in this way the number of elements we will need to check will grow exponentially! However, surprisingly this is not the case! We can reduce checking two elements in round $i+1$ to checking two elements in round $i+2$.

The idea is to run two Sumcheck protocols (one for each element) but where the verifier uses the same randomness in both these Sumcheck protocols! This will convert checking two elements in round i to checking two elements in round $i+1$.

A Numeric GKR Example over $\mathbb{F}_{17}$

Step 1: Circuit definition and evaluation.

We work over the prime field

$$\mathbb{F} = \mathbb{F}_{17} = \mathbb{Z}/17\mathbb{Z}.$$

Consider a depth-3 layered arithmetic circuit

$$C : \{0,1\}^3 \to \mathbb{F}_{17}$$

with layers numbered output-first:

- Layer 0 (output): one gate g_0.
- Layer 1: two gates g_1, g_2.
- Layer 2 (inputs): three input gates x_1, x_2, x_3 (plus one dummy).

The gate definitions (over $\mathbb{F}_{17}$) are:

$$g_1 = x_1 + x_2, \qquad g_2 = x_2 \cdot x_3, \qquad g_0 = g_1 + g_2.$$

Fix the concrete input

$$(x_1, x_2, x_3) = (1, 0, 1) \in \{0,1\}^3.$$

Then, the gate values (computed in $\mathbb{F}_{17}$) are

$$g_1 = 1 + 0 = 1, \qquad g_2 = 0 \cdot 1 = 0, \qquad g_0 = 1 + 0 = 1.$$

The prover claims that $C(x) = 1$, i.e.,

$$\widetilde{V_0}(z_0) = 1$$

for a suitable label z_0 of the output gate.

Step 2: Labeling gates with H^m.

Let

$$H = \{0,1\}, \qquad m = 2,$$

so that $|H|^m = 4$ labels are available per layer. We assign labels in each layer as follows.

Layer 0.

$$g_0 \leftrightarrow z_0 = (0,0) \in H^2.$$

Layer 1.

$$g_1 \leftrightarrow (0,0), \qquad g_2 \leftrightarrow (0,1),$$

and the remaining labels $(1,0)$ and $(1,1)$ are dummy gates with value 0.

Layer 2.

$$x_1 \leftrightarrow (0,0), \quad x_2 \leftrightarrow (0,1), \quad x_3 \leftrightarrow (1,0), \quad \text{dummy} \leftrightarrow (1,1).$$

Step 3: The value tables V_i and their LDEs over $\mathbb{F}_{17}$.

For each layer $i \in \{0,1,2\}$, define

$$V_i : H^2 \to \{0,1\} \subset \mathbb{F}_{17}$$

to be the gate-value function at layer i.

Layer 2.

$$V_2(p) = \begin{cases} 1 & p = (0,0) \text{ (for } x_1\text{)}, \\ 0 & p = (0,1) \text{ (for } x_2\text{)}, \\ 1 & p = (1,0) \text{ (for } x_3\text{)}, \\ 0 & p = (1,1) \text{ (dummy)}. \end{cases}$$

Layer 1.

$$V_1(p) = \begin{cases} 1 & p = (0,0) \text{ (for } g_1\text{)}, \\ 0 & p = (0,1) \text{ (for } g_2\text{)}, \\ 0 & p = (1,0), \\ 0 & p = (1,1). \end{cases}$$

Layer 0.

$$V_0(p) = \begin{cases} 1 & p = (0,0) \text{ (for } g_0\text{)}, \\ 0 & \text{otherwise.} \end{cases}$$

Lagrange basis over $H = \{0,1\}$. Over $\mathbb{F}_{17}$, the univariate Lagrange basis polynomials for $H = \{0,1\}$ are

$$\chi_0(u) = \frac{u-1}{0-1} = 1-u, \qquad \chi_1(u) = \frac{u-0}{1-0} = u.$$

For $p = (p_1, p_2) \in H^2$ and $u = (u_1, u_2) \in \mathbb{F}_{17}^2$, define

$$\chi_p(u) := \chi_{p_1}(u_1)\, \chi_{p_2}(u_2).$$

Low-degree extensions. The LDE of V_i is

$$\widetilde{V}_i(u_1, u_2) = \sum_{(h_1,h_2)\in H^2} V_i(h_1, h_2)\, \chi_{h_1}(u_1)\, \chi_{h_2}(u_2).$$

Explicitly:

- For layer 2:

  $$\widetilde{V_2}(u_1, u_2) = V_2(0,0)\chi_0(u_1)\chi_0(u_2) + V_2(0,1)\chi_0(u_1)\chi_1(u_2) + V_2(1,0)\chi_1(u_1)\chi_0(u_2) + V_2(1,1)\chi_1(u_1)\chi_1(u_2).$$

  Using $V_2(0,0) = 1$, $V_2(0,1) = 0$, $V_2(1,0) = 1$, $V_2(1,1) = 0$:

  $$\widetilde{V_2}(u_1, u_2) = \chi_0(x_1)\chi_0(u_2) + \chi_1(u_1)\chi_0(u_2) = (1-u_1)(1-u_2) + u_1(1-u_2).$$

  Simplify in $\mathbb{F}_{17}$:

  $$(1-u_1)(1-u_2) + u_1(1-u_2) = (1-u_2)\big((1-u_1) + u_1\big) = (1-u_2)\cdot 1 = 1-u_2.$$

  So

  $$\widetilde{V_2}(u_1, u_2) = 1 - u_2 \pmod{17}.$$

- For layer 1:

  $$\widetilde{V_1}(u_1, u_2) = V_1(0,0)\chi_0(u_1)\chi_0(u_2) + V_1(0,1)\chi_0(u_1)\chi_1(u_2) + V_1(1,0)\chi_1(u_1)\chi_0(u_2) + V_1(1,1)\chi_1(u_1)\chi_1(u_2).$$

  Using $V_1(0,0) = 1$ and all others 0:

  $$\widetilde{V_1}(u_1, u_2) = \chi_0(u_1)\chi_0(u_2) = (1-u_1)(1-u_2).$$

- For layer 0:

  $$\widetilde{V_0}(u_1, u_2) = V_0(0,0)\chi_0(u_1)\chi_0(u_2) = (1-u_1)(1-u_2).$$

  In particular,

  $$\widetilde{V_0}(z_0) = \widetilde{V_0}(0,0) = (1-0)(1-0) = 1,$$

  matching the claimed output.

Step 4: The claim at layer o and its Sumcheck polynomial.

The prover's top-level claim is

$$\widetilde{V_0}(z_0) = 1, \qquad z_0 = (0,0).$$

By LDE expansion,

$$\widetilde{V_0}(z_0) = \sum_{p\in H^2} V_0(p)\, \chi_p(z_0).$$

We now express $V_0(p)$ in terms of layer 1 using wiring predicates.
Define

$$ADD_0, MULT_0 : (H^2)^3 \to \{0,1\}$$

by

$$ADD_0(p, w_1, w_2) = \begin{cases} 1 & \text{if } p \text{ is an ADD gate with children } w_1, w_2 \text{ in layer 1,} \\ 0 & \text{otherwise,} \end{cases}$$

and similarly for $MULT_0$ with ADD replaced by MULT.

In our tiny circuit, layer 0 has a single ADD gate g_0 with children g_1, g_2, so

$$ADD_0(p, w_1, w_2) = 1 \quad \text{iff} \quad p = z_0,\ w_1 = (0,0),\ w_2 = (0,1),$$

and $MULT_0(p, w_1, w_2) = 0$ for all (p, w_1, w_2).

For any $p \in H^2$,

$$V_0(p) = \sum_{w_1, w_2 \in H^2} \Big(ADD_0(p, w_1, w_2)\big(V_1(w_1) + V_1(w_2)\big) + MULT_0(p, w_1, w_2)\big(V_1(w_1)V_1(w_2)\big) \Big).$$

Plugging into the expression for $\widetilde{V_0}(z_0)$:

$$v_0 := \widetilde{V_0}(z_0) = \sum_{p, w_1, w_2 \in H^2} \Big(ADD_0(p, w_1, w_2)\big(V_1(w_1) + V_1(w_2)\big) + MULT_0(p, w_1, w_2)\big(V_1(w_1)V_1(w_2)\big) \Big)\, B(p, z_0),$$

where $B(p, z_0)$ is the polynomial version of $\chi_p(z_0)$ (viewed as a low-degree polynomial in p).

Define the Sumcheck polynomial

$$f_{0,z_0}(p, w_1, w_2) = \Big(ADD_0(p, w_1, w_2)\big(\widetilde{V_1}(w_1) + \widetilde{V_1}(w_2)\big) + MULT_0(p, w_1, w_2)\big(\widetilde{V_1}(w_1)\widetilde{V_1}(w_2)\big) \Big)\, B(p, z_0),$$

and the Sumcheck instance is

$$v_0 \stackrel{?}{=} \sum_{p, w_1, w_2 \in H^2} f_{0,z_0}(p, w_1, w_2).$$

Step 5: A concrete Sumcheck transcript for layer 0.

Let us index the six scalar variables as

$$u_1, u_2 \text{ (for } p), \quad u_3, u_4 \text{ (for } w_1), \quad u_5, u_6 \text{ (for } w_2),$$

each ranging over $H = \{0,1\}$ in the sum, but over $\mathbb{F}_{17}$ as polynomial variables.

The Sumcheck protocol proceeds in 6 rounds. We describe the structure and fix concrete random challenges in $\mathbb{F}_{17}$ for illustration.

Round 1. The prover sends a univariate polynomial $g_1(X)$ claiming

$$g_1(X) = \sum_{u_2,u_3,u_4,u_5,u_6 \in H} f_{0,z_0}(X, u_2, u_3, u_4, u_5, u_6).$$

The verifier checks

$$g_1(0) + g_1(1) \stackrel{?}{=} v_0 = 1.$$

If the check passes, the verifier samples a random $r_1 \in \mathbb{F}_{17}$, say

$$r_1 = 3,$$

and fixes $u_1 := 3$.

Round 2. Now the prover sends $g_2(X)$ claiming

$$g_2(X) = \sum_{u_3,u_4,u_5,u_6 \in H} f_{0,z_0}(r_1, X, u_3, u_4, u_5, u_6).$$

The verifier checks

$$g_2(0) + g_2(1) \stackrel{?}{=} g_1(r_1),$$

and if it passes, samples $r_2 \in \mathbb{F}_{17}$, say $r_2 = 5$, and fixes $u_2 := 5$.

Rounds 3–6. Proceed similarly:

- Round 3: prover sends $g_3(X)$, verifier checks sum over H, then fixes $u_3 := r_3$ with, say, $r_3 = 7$.
- Round 4: prover sends $g_4(X)$, verifier checks, then fixes $u_4 := r_4$ with, say, $r_4 = 11$.
- Round 5: prover sends $g_5(X)$, verifier checks, then fixes $u_5 := r_5$ with, say, $r_5 = 2$.
- Round 6: prover sends $g_6(X)$, verifier checks, then fixes $u_6 := r_6$ with, say, $r_6 = 13$.

At the end, all coordinates are fixed:

$$(p^*, w_1^*, w_2^*) = \big((u_1, u_2), (u_3, u_4), (u_5, u_6)\big) = \big((3,5), (7,11), (2,13)\big) \in \mathbb{F}_{17}^2 \times \mathbb{F}_{17}^2 \times \mathbb{F}_{17}^2.$$

The verifier must now check

$$f_{0,z_0}(p^*, w_1^*, w_2^*) \stackrel{?}{=} g_6(r_6).$$

Since $ADD_0, MULT_0, B$ are known, this reduces to checking the values of $\widetilde{V_1}$ at w_1^* and w_2^*. Recall

$$\widetilde{V_1}(u_1, u_2) = (1 - u_1)(1 - u_2) \pmod{17}.$$

Thus,

$$\widetilde{V_1}(w_1^*) = \widetilde{V_1}(7,11) = (1-7)(1-11) = (-6)(-10) = 60 \equiv 9 \pmod{17},$$

since $60 = 3 \cdot 17 + 9$.

Similarly,

$$\widetilde{V_1}(w_2^*) = \widetilde{V_1}(2,13) = (1-2)(1-13) = (-1)(-12) = 12 \pmod{17}.$$

The prover must provide values $a_1, a_2 \in \mathbb{F}_{17}$ such that

$$a_1 = \widetilde{V_1}(w_1^*), \qquad a_2 = \widetilde{V_1}(w_2^*),$$

i.e.,

$$a_1 = 9, \qquad a_2 = 12.$$

If the prover cheats on these, the final equality

$$f_{0,z_0}(p^*, w_1^*, w_2^*) \stackrel{?}{=} g_6(r_6)$$

will fail with high probability.

Thus, after Sumcheck for layer 0, the verifier is left with *two claims about layer* 1:

$$\widetilde{V_1}(w_1^*) = 9, \qquad \widetilde{V_1}(w_2^*) = 12.$$

Step 6: Propagation to deeper layers and final input checks.

Abstracting to a general layer i, the same reasoning yields:

- At layer i, after running Sumcheck on the relation expressing v_i in terms of $\widetilde{V_{i+1}}$, the verifier ends with two claims

  $$\widetilde{V_{i+1}}(r_{i,1}) = v_{i+1,1}, \qquad \widetilde{V_{i+1}}(r_{i,2}) = v_{i+1,2},$$

  where $r_{i,1}, r_{i,2} \in \mathbb{F}_{17}^m$ are random points determined by the Sumcheck randomness.

Naively, one would run two independent Sumchecks at layer $i+1$, one for each claim, which would double the number of claims at each layer. The GKR protocol avoids this blowup by using the *same randomness* in both Sumchecks at layer $i+1$:

- For the first claim $\widetilde{V_{i+1}}(r_{i,1})$, the verifier sends random challenges $s_1, \ldots, s_k$.
- For the second claim $\widetilde{V_{i+1}}(r_{i,2})$, the verifier reuses the same $s_1, \ldots, s_k$.

This couples the two Sumchecks so that, after layer $i+1$, the verifier again has exactly two claims about layer $i+2$ (not four). Repeating this down to the input layer D, the verifier finally obtains two claims

$$\widetilde{V_D}(r_{D-1,1}) = v_{D,1}, \qquad \widetilde{V_D}(r_{D-1,2}) = v_{D,2}.$$

In our depth-3 example, $D = 2$ (input layer is layer 2). We already computed

$$\widetilde{V_2}(u_1, u_2) = 1 - u_2 \pmod{17}.$$

Thus, for any random points $r_{1,1} = (a, b)$ and $r_{1,2} = (c, d)$ in $\mathbb{F}_{17}^2$, the verifier can directly evaluate

$$\widetilde{V_2}(r_{1,1}) = 1 - b, \qquad \widetilde{V_2}(r_{1,2}) = 1 - d,$$

using only the known input layer. The prover's claimed values $v_{2,1}, v_{2,2}$ must match these evaluations. If all checks at all layers pass, the verifier accepts the original claim $C(x) = 1$; otherwise, it rejects.

GKR Proof Size

- For a circuit with layer widths $S_0, S_1, \ldots, S_{D-1}$, the total GKR proof size is
$$O\left(\sum_{i=0}^{D-1} \log S_i\right).$$
- If every layer has width at most S, this simplifies to $O(D \log S)$.
- Each layer contributes $O(\log S_i)$ communication because the Sumcheck protocol runs for $\log S_i$ rounds, and each round sends a degree-2 univariate polynomial (three field elements).
- Thus the total communication is polylogarithmic in the circuit size.
- For a circuit with $S = 2^{20}$ gates and depth $D = 20$, the proof contains only a few hundred field elements (a few kilobytes).

Remark. The GKR protocol can be extended to handle circuits of arbitrary depth, but we will not cover this and other extensions in this class.

References

[1] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 113–122. ACM, 2008.

[2] Adi Shamir. Ip=pspace. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 11–15. IEEE Computer Society, 1990.