1 Notation

F ✓ $\quad$ $char(F)$ ✓ $\quad$ $char(F) > 2$ ✓ $\quad$ $F^* \rightarrow$ all non-zero elements of F under multiplication

Boolean Hypercube ✓✓

$$H_n = \{\pm 1\}^n$$

$$H_n \subseteq (F^*)^n$$

$$H_3 = \{\pm 1\}^3 \Rightarrow (1,1,1),\ (1,1,-1),\ (1,-1,1),\ (-1,1,1),$$
$$(-1,1,-1),\ (-1,-1,1),\ (1,-1,-1),\ (-1,-1,-1)$$

Multilinear polynomials ✓✓

$$p(X_1, \ldots, X_n) = \sum_{S \subseteq (1,\ldots,n)} c_S \prod_{i \in S} X_i$$ ✓✓

for e.g. $\rightarrow$ $p(X_1, X_2, X_3) = c_0 + c_1 X_1 + c_2 X_2 + c_3 X_3 + c_{(1,2)} X_1 X_2$
$+ c_{(1,3)} X_1 X_3 + c_{(2,3)} X_2 X_3 + c_{(1,2,3)} X_1 X_2 X_3$

Lagrange Kernel $(L_n(\vec{X}, \vec{Y}))$

$$L_n(\vec{X}, \vec{Y}) = \frac{1}{2^n} \prod_{i=1}^{n} (1 + x_i y_i)$$

$\vec{X} = (x_1, \ldots, x_n)$

$\vec{Y} = (Y_1, \ldots, Y_n)$

Remark $\rightarrow$ Lagrange Kernel acts as a "delta function" on the Boolean

Hypercube.

if $\vec{Y} = \vec{y} \in H_n$ (each $y_i \in \{\pm 1\}$)

$$\Rightarrow \quad L_n(\vec{x}, \vec{y}) = \begin{cases} 1 & \text{if } \vec{x} = \vec{y} \\ 0 & \text{for } \vec{x} \in H_n \end{cases}$$

e.g. $(n=3)$

$$L_3\left((x_1, x_2, x_3), (y_1, y_2, y_3)\right) = \frac{1}{2^3}\left((1+x_1y_1)(1+x_2y_2)(1+x_3y_3)\right)$$

if $\vec{y} = (1,-1,1) \in H_3$

$$\Rightarrow L_3\left((x_1, x_2, x_3)(1,-1,1)\right) = \frac{1}{2^3}\left((1+x_1)(1-x_2)(1+x_3)\right)$$

$\Rightarrow$ when $\vec{x}$ itself is in $\{\pm 1\}^3$ $\Rightarrow$ if $\vec{x} = (1,-1,1)$ $\Rightarrow$ L_3 becomes

$$= (1+1)\cdot(1+1)\times(1+1)\frac{1}{2^3}$$
$$= \frac{1}{8} \times 8 = 1$$

$\Rightarrow$ if $\vec{x}$ differs from $(1,-1,1)$ in at least one coordinate

$\Rightarrow$ whole product becomes 0.

$\Rightarrow$ $L_n(\vec{x}, \vec{y})$ can be viewed as unique multilinear polynomial that interpolates to 1 at the point $\vec{y}$ on the hypercube and 0 at all other points of H_n.

For our purpose, all multilinear polynomials $p(\vec{x})$ in n variables will be given in Lagrange representation i.e. their values over H_n.

→ we describe our protocol as what is called Lagrange interactive oracle proof.

② LogUp in a Nutshell

- the setting
  1. M multilinear polynomials $w_1(X), \dots, w_M(X)$ in n variables over a finite field F with $\mathrm{char}(F) > 2$
  2. A table polynomial $t(X)$ whose values on $H_n = \{\pm 1\}^n$ are all distinct.
  3. The prover knows all values of $w_i(X)$ and $t(X)$ on the hypercube H_n
  4. The goal for the prover is to convince the verifier that each of the M "witness columns" (i.e. the values of w_i on H_n) comes from the set of table values $\{t(x) : x \in H_n\}$.
     Equivalently, every $w_i(X)$ is indeed a value that appears among $\{t(z) \mid z \in H_n\}$

Interpreting "Membership in a Table"

If $t(X)$ has distinct values on the 2^n points $x \in H_n$, it effectively encodes a "table of size 2^n". Then saying "$w_i(X)$ appears in the table" is saying "$w_i(X) = t(z)$ for some $z \in H_n$"

- Defining $m(x)$: The Multiplicities

$$\boxed{m(X) = \sum_{i=1}^{M} \left| \{ y \in H_n : w_i(y) = t(x) \} \right|}$$

→ "how many times, across all columns $w_1, \dots, w_M$ do we see the value $t(x)$"

Note that all values of t on H_n are distinct, $t(x)$ is a well-defined unique table for each $x \in H_n$.

- **The "Virtual Identity"**

$$\underbrace{\prod_{i=1}^{M} \prod_{x \in H_n} \left(X - w_i(x)\right)}_{\text{All witness values as roots}} = \underbrace{\prod_{x \in H_n} \left(X - t(x)\right)^{m(x)}}_{\text{All table values raised to appropriate multiplicities}} \quad (1)$$

If every witness value $w_i(x)$ truly lies in the set of table values $\{t(z) : z \in H_n\}$, then in fact the collection of all witness values is precisely the collection of table values (counted with the same multiplicities)

- **Using "Logarithmic Derivatives" Instead of Directly Checking (1)**

→ why not directly check (1)

↳ expanding $\prod^{M} \prod (X - w_i(x))$ would be huge of size $M \cdot 2^n$

↳ extremely expensive.

→ **Logarithmic derivative idea**

If two polynomials $P_L(X)$ and $P_R(X)$ differ by more than a nonzero constant factor, their logarithmic derivatives will differ

∴ Since, log-derivative of $\prod(X - a_j) = \sum_j 1/(X - a_j)$

⇒ we check

$$\boxed{\sum_{i=1}^{M} \sum_{x \in H_n} \frac{1}{X - w_i(x)} = \sum_{x \in H_n} \frac{m(x)}{X - t(x)}} \qquad (2)$$

- **Substituting $X = \alpha$ in (2)**

we get

$$\boxed{\sum_{\vec{x} \in H_n} \left(\frac{m(\vec{x})}{\alpha - t(\vec{x})} - \sum_{i=1}^{M} \frac{1}{\alpha - w_i(\vec{x})} \right) = 0} \qquad (3)$$

The Sum here is zero

This is now just a single numeric check in the field F.

$$\Rightarrow \sum_{x \in H_n} \left(m(x) \cdot \underbrace{\frac{1}{\alpha - t(\vec{x})}}_{\text{a rational expression}} - \sum_{i=1}^{M} \underbrace{\frac{1}{\alpha - w_i(x)}}_{\text{another rational expression}} \right) = 0$$

(4) Why a "Sumcheck" is Needed

We have rational terms like $\frac{1}{\alpha - a}$ in (3)

Interactive proofs typically prefer polynomials rather than arbitrary rational expressions

Transforming (3) into polynomial Sum

prover supplies helper polynomials $h(X)$, $h_1(X)$, ... $h_M(X)$ such that on each point $x \in H_n$

$$h(x) = \frac{1}{\alpha - t(x)} \; ; \quad h_i(x) = \frac{1}{\alpha - w_i(x)}$$

$\alpha - u(x)$ $\alpha - w_i(x)$

$\Rightarrow$ (3) transforms to $\sum_{x \in H_n} \left(m(x).h(x) - \sum_{i=1}^{M} h_i(x) \right) = 0$

Now we can use multivariate sumcheck to verify

$$\sum_{x \in H_n} P(x) = 0$$

where $P(x) = m(X).h(X) - \sum_{i=1}^{M} h_i(X)$ is a multilinear polynomial.

(3) GKR for fractional Sumchecks

- Issue with fractional Sumchecks

here the sum to be checked is

$$\sum_{x \in H_n} \frac{p(x)}{q(x)} = 0$$

we no longer have a single polynomial in X but a quotient of two polynomials

$\downarrow$

a way to solve this would be introduce "helper" poly. as discussed above where

$$h(X) = \frac{1}{q(X)} \Rightarrow \sum_{X \in H_n} \frac{p(X)}{q(X)} = \sum_{X \in H_n} p(X).h(X)$$

but the issue is that now we need to additionally prove "$h(X) = 1$"

which can take extra effort - additional "helper columns" in the protocol - which might complicate the proof.

- **Enter GKR: Using "Projective Coordinates"**

A fraction $\frac{a}{b}$ (with $b \neq 0$) can be represented by the pair $(a,b) \in F^2$

↳ e.g. $7/3$ can be written as $(7,3)$

$\because b \neq 0 \Rightarrow (a,b)$ can be interpreted as ab^{-1} in the field.

→ addition of fractions $\left(\frac{a_0}{b_0} + \frac{a_1}{b_1}\right) = \frac{a_0 b_1 + b_0 a_1}{b_0 b_1}$

$$(a_0, b_0) +_F (a_1, b_1) = (a_0 b_1 + b_0 a_1,\ b_0 b_1)$$

why called projective → cause $\frac{a}{b}$ can also be scaled and λ $(\neq 0)$ without changing fraction's value i.e. $(\lambda a, \lambda b)$

- The pair (a,b) is an equivalence class up to scaling.

- **The layered Circuit Structure**

→ **Binary-Tree "Topology" of the Hypercube**

- $H_n = \{\pm 1\}^n$ has 2^n points
- A convenient way to sum over H_n is to organize 2^n points in a binary tree of height n.
  - ↳ Layer n (the bottom) has leaves corresponding to each $x \in H_n$
  - ↳ layer k has "nodes" corresponding to the k-dimensional slices $H_k \subset H_n$
  - ↳ layer 0 is the root of the tree - a single node that aggregates everything from below.

Therefore, each parent node at layer k represents a partial sum over a block of $2^{\cdots}$ points in H_m.

→ <u>Storing fractional values at Each Node</u>

- In projective form, each leaf is holding $(p(x), q(x)) \in F^2$ for a unique $x \in H_n$

- The parent node a layer up stores sum of it's two children fractions

$$\Rightarrow (a_{parent}, b_{parent}) = (a_0 b_1 + b_1 a_0, b_0 b_1)$$

↪ this is actually

$$\left(\frac{a_0}{b_0} + \frac{a_1}{b_1}\right) = \left(\frac{a_0 b_1 + a_1 b_0}{b_1 b_0}\right)$$

- Top layer (layer 0) stores (A_{root}, B_{root})

$$\boxed{\frac{A_{root}}{B_{root}} = \sum_{x \in H_m} \frac{p(x)}{q(x)}} \longrightarrow$$

"cumulative sum" of all leaf fraction

$\boxed{3.1}$

- <u>Bottom layer $(k = n)$</u>

$$\Rightarrow (p_n(x), q_n(x)) = (p(x), q(x))$$

[Both $p(x)$ and $q(x)$ are multilinear polynomials evaluated at x]

- <u>Internal layers $(0 \le k < n)$</u>

$$\frac{p_k(x)}{q_k(x)} = \frac{p_{k+1}(x, +1)}{q_{k+1}(x, +1)} + \frac{p_{k+1}(x, -1)}{q_{k+1}(x, -1)}$$

[Each node at layer k corres. to a partial vector in $\{\pm 1\}^k$. This node has two children at layer $k+1$ labeled $(x, +1)$ and $(x, -1)$.
→ "going down" one layer appends a next coordinate ± 1]

- <u>Top layer $(k = 0)$</u>

$$\boxed{\frac{p_0}{q_0} = \sum_{y \in H_m} \frac{p(y)}{q(y)}}$$

3.2

- At each layer k, we want to confirm the correctness of some function ϕ_k that feed into layer $k-1$.
- Non-standard twist : Projective wires
  - ↳ each 'wire' in layer k is a pair (p_k, q_k) encoding a fraction.
  - ↳ So at layer k, we must prove correctness of two multilinear polynomials $p_k(X)$ and $q_k(X)$

↓

How ?

↓

The process

- **First - Round** $(k=0)$

→ At the topmost layer $(k=0)$

↳ we have pair $(p_1(+1), q_1(+1))$ and $(p_1(-1), q_1(-1))$ as children of root node

→ verifier picks a random $u_0 \in F$ and forms

$r_0 = 1 - u_0$

The idea : combine $p_1(+1)$ and $p_1(-1)$ linearly via u_0, Similarly for $q_1(+1)$ and $q_1(-1)$ and get a "single-point claim" on $(p_1(r_0), q_1(r_0))$

→ Essentially, we use property of the multilinear poly. $f(X)$

$$\boxed{f(u.(+1) + (1-u).(-1)) = u\, f(+1) + (1-u)\, f(-1)}$$

in each variable

- **Further Rounds** $(1 \le k \le n-1)$: Recursively checking (p_k, q_k) via (p_{k+1}, q_{k+1})

Next we have,

$$\frac{p_k(X)}{q_k(X)} = \frac{p_{k+1}(X, +1)}{q_{k+1}(X, -1)} + \frac{p_{k+1}(X, -1)}{q_{k+1}(X, -1)}$$

Also, the lagrange Kernel $L_k(X, Y)$ ensures that this expression is correct pointwise on the hypercube H_k.

→ **The GKR check at layer k**

- verifier choses a random point "r_k"
- prover must show that $(p_k(r_k), q_k(r_k))$ matches poly. def in terms of (p_{k+1}, q_{k+1}).

↓

Normally, to prove $p_k(r_k) = \alpha$ and $q_k(r_k) = \beta$, we might do two different sumchecks.

↳ Instead, we pick random $\lambda_k \in F$ each time, and merges

$$p_k(r_k) + \lambda_k \cdot q_k(r_k)$$

→ The single expression is again a 'multilinear poly' in X (fixing r_k and λ_k), so we can use a single Sum Check

Descending to the Next layer $k+1$

from $p_k(r_k) + \lambda_k \cdot q_k(r_k)$

↓

we move to

$$\left.\begin{array}{l} p_{k+1}(\rho_k, +1) \\ q_{k+1}(\rho_k, +1) \\ p_{k+1}(\rho_k, -1) \\ q_{k+1}(\rho_k, -1) \end{array}\right\} \rho_k \in F^k \text{ being randomness sampled in the course of the protocol}$$

we use $\mu_k \leftarrow F$ to combine two-point claims to

$p_{k+1}(r_{k+1})$, $q_k(r_{k+1})$

↳ another combined sumcheck

↓

until we reach (p_n, q_n)

- The Final Condition $\sum_{x \in H_n} \frac{p(x)}{q(x)} = 0$

$$p_1(+1) \cdot q_1(+1) + p_1(-1) \cdot q_1(-1) = 0$$

and

$$q_1(+1) \cdot q_1(-1) = 0$$

Soundness Error

$$\boxed{\varepsilon_{GKR} \leq \frac{2 \cdot (n-1) + 1}{|F|} + \sum_{K=1}^{n-1} \varepsilon_{\text{Sumcheck}}(|H_K|) \leq \frac{1}{2} \cdot \frac{n(3 \cdot n + 1)}{|F|}}$$

- Computational Cost

The total cost after all the operations using the formula:

$$\boxed{|H_K| \cdot (d+1) \cdot ((v + |Q|_M) \cdot M + (v + |Q|_A) \cdot A)}$$ from [Hab22]

equals to about

$$|H_n| \cdot (43 \cdot M + 29 \cdot A)$$

$|H_n| = 2^n$ terms

$M \rightarrow$ cost of one field multiplication

$N \rightarrow$ cost of one field addition