Hugo Delavenne

Barracuda presentation: Flowering graphs

Thursday 9th January, 2025

Contents

First part

Verifiable computing

1.1 SNARKs

SNARK stands for Succinct Non-interactive ARgument of Knowledge.

Idea. We want to turn

$$A : x, h \longmapsto y$$

that runs in time $\tau(n)$, into

$$A' : x, h \longmapsto y, \boldsymbol{\alpha}$$

such that

- $|\alpha| \ll \tau(n)$ Succinct
- nothing on h is learnt from α zero-knowledge
- A' runs in time $O(\tau \log \tau)$ (or even $O(\tau)$)
- $\exists \mathcal{V}$ running in time $\mathrm{poly}(|\alpha|)$ Non-interactive ARgument of Knowledge

$$\text{if } \exists h, A(x,h) = y \text{ then } \mathbb{P}(\mathcal{V}(\alpha) \text{ accepts}) = 1 \qquad \text{(completeness)}$$

$$\text{if } \forall h, A(x,h) \neq y \text{ then } \mathbb{P}(\mathcal{V}(\alpha) \text{ accepts}) \leqslant s. \qquad \text{(soundness)}$$

Applications

- ▷ Speedrun: Proof of Emulation → Doom
- ▷ Signature holding: Proof of Authenticity → photo authenticity
- ▷ AI regulation: Proof of Training → theoretical
- ▷ Blockchain: Proof of Transaction → practical

1.2 | Arithmetization

Consider a computer as a r-register machine i.e. you can use r variables in your program.

Execution trace. Let $H = \langle g \rangle \subseteq \mathbb{F}_q^*$. The Prover will "send" (in a sense defined later) polynomials $R_1, ..., R_r$ such that

$$\forall 0 \leqslant t \leqslant \tau, \qquad R_i(g^t) = \text{register } i \text{ at } t$$

Program. A program $A = (I_1, ..., I_\tau)$ is a list of instructions.

Definition 1.1 **Instruction**

An instruction is an operation that assigns to a register a polynomial of the value of the registers.

More generally, we consider any instruction I that can be represented as a polynomial $Q_I \in \mathbb{F}[X_1, ..., X_r, Y_1, ..., Y_r]$ such that

$$(R_1, ..., R_r) \overset{I}{\mapsto} (R'_1, ..., R'_r) \iff Q_I(R_1, ..., R_r, R'_1, ..., R'_r) = 0$$

We write an instruction as a **constraint polynomial** whose roots are the matching values.

Main instructions

- ▷ **Addition.** $(R_1, R_2) \overset{+}{\mapsto} (R_1 + R_2, R_2)$ becomes $Q_+(X_1, X_2, Y_1, Y_2) = Y_1 - X_1 - X_2$
- ▷ **Multiplication.** $(R_1, R_2) \overset{\times}{\mapsto} (R_1 \times R_2, R_2)$ becomes $Q(X_1, X_2, Y_1, Y_2) = Y_1 - X_1X_2$
- ▷ **Division.** $(R_1, R_2) \overset{\div}{\mapsto} (R_1/R_2, R_2)$ becomes $Q_\div(X_1, X_2, Y_1, Y_2) = X_1 - X_2Y_1$
- ▷ **Boolean equality testing.** $(R_1, R_2, R_3) \overset{\overset{?}{=}}{\mapsto} (R_2 \overset{?}{=} 0, R_2, R_3)$ becomes 3 constraints

$$\begin{aligned} Q_{\overset{?}{=},\text{bin}} &= Y_1(1 - Y_1) \\ Q_{\overset{?}{=},\text{zero}} &= Y_1X_2 \\ Q_{\overset{?}{=},\text{inv}} &= X_2X_3 - (1 - Y_1) \end{aligned}$$

  where X_3 is **asked** by non-determinism to be the inverse of X_2 when there exists.
- ▷ **Conditions and loops.** As in assembly, we write the program in the memory and use gotos.

Composition with the registers.

$$Q_I \circ \mathbf{R}(X) := Q_I(R_1(X), ..., R_r(X), R_1(gX), ..., R_r(gX))$$

Time specification. Instruction I_{t_0} must only apply at time t_0

$$P_{t_0}(X) := Q_{I_{t_0}} \circ \mathbf{R}(X) \times \prod_{\substack{t=1 \\ t \neq t_0}}^{\tau} (X - g^t) = Q_{I_{t_0}} \circ \mathbf{R}(X) \times \underbrace{\frac{X^\tau - 1}{X - g^{t_0}}}_{=:T_{t_0}(X)}$$

And if an instruction is used several times, we can combine them.

Limitations. We actually also need to make sure that the other registers are not modified, with constraints. It becomes very heavy if there are a lot of registers, so then we can use a RAM model.

Example 1.2 Square Fibonacci

- $f_0 = f_1 = 1$ and $f_{i+2} := f_{i+1}^2 + f_i^2 \mod p$
- Three registers: f for "f_{i+1}", g for "f_i" and h for "f_{i-1}"
- The program is:
  - for $i = 1, ..., \tau/3$
    - $h \leftarrow f$
    - $f \leftarrow f^2 + g^2$
    - $g \leftarrow h$
- Three alternating instructions, contraints and time specificators:

$$\begin{array}{ccc} f,g,h \mapsto f,g,f & f,g,h \mapsto f^2+g^2,g,h & f,g,h \mapsto f,h,h \\ Q_h = Y_3 - X_1 & Q_f = Y_1 - X_1^2 - X_2^2 & Q_g = Y_2 - X_1 \\ \begin{aligned} T_h &= \prod_{t=1}^{\tau}(X - g^{3t}) \\ &= X^{\tau/3} - 1 \end{aligned} & \begin{aligned} T_f &= \prod_{t=1}^{\tau}(X - g^{3t+1}) \\ &= X^{\tau/3} - g^{\tau/3} \end{aligned} & \begin{aligned} T_g &= \prod_{t=1}^{\tau}(X - g^{3t+2}) \\ &= X^{\tau/3} - g^{2\tau/3} \end{aligned} \\ P_h = Q_h \circ \mathbf{R} \times T_h & P_f = Q_f \circ \mathbf{R} \times T_f & P_g = Q_g \circ \mathbf{R} \times T_g \end{array}$$

Summary Arithmetization idea

The computation of $A = (I_1, ..., I_\tau)$ is valid $\overset{\text{definition}}{\iff}$ $\exists R_1, ..., R_r$ such that at each step t, $(R_1, ..., R_r) \overset{I_t}{\mapsto} (R'_1, ..., R'_r)$ $\overset{\text{arithmetization}}{\iff}$ $\exists R_1, ..., R_r$ such that for all t, $P_{I_t}(X)$ cancels on $H \subseteq \mathbb{F}$

Second part

FRI protocol: proximity test to Reed-Solomon codes

2.1 Testing proximity suffices

Lemma 2.1 Schwartz-Zippel lemma

Let $P \neq Q \in \mathbb{K}[X_1, ..., X_n]_{\leqslant d}$ and $S \subseteq \mathbb{K}$ finite. Then

$$\mathbb{P}_{(x_1,...,x_n)\in S}(P(x_1, ..., x_n) = Q(x_1, ..., x_n)) \leqslant \frac{d}{|S|}.$$

Definition 2.2 Reed-Solomon code

Let $\mathcal{L} \subseteq \mathbb{F}_q$ and $k < |\mathcal{L}|$. If $f : \mathcal{L} \to \mathbb{F}_q$, denote $\widehat{f} \in \mathbb{F}_q[X]$ the interpolator of f on $\mathcal{L}$. Define RS$[\mathcal{L}, k]$, or RS$[|\mathcal{L}|, k]$ as

$$\left\{ f : \mathcal{L} \to \mathbb{F}_q \mid \deg \widehat{f} < k \right\}.$$

For $f, R_1, ..., R_r : \mathcal{L} \to \mathbb{F}_q$ and $Q(X_1, ..., X_r, Y_1, ..., Y_r)$. Denote $Z(X) := \prod_{t=1}^{\tau}(X - g^t)$. The claim is the algebraic equality

$$\underbrace{Q \circ \widehat{\mathbf{R}}(X) \times T(X)}_{=\widehat{P}(X)} \overset{?}{=} \widehat{f}(X)Z(X).$$

Lemma 2.3 **Proximity suffices**

Let $\delta > 0$. If

- $\forall i, \Delta(R_i, \mathsf{RS}[\mathcal{L}, \tau]) < \delta$ → proximity test
- $\Delta(f, \mathsf{RS}[\mathcal{L}, \tau \deg Q]) < \delta$ → proximity test
- $\mathbb{P}_{x \in \mathcal{L}}(\underbrace{Q \circ \mathbf{R}(x) \times T(x)}_{=P(x)} = f(x)Z(x)) > \frac{\tau(\deg Q + 1)}{|\mathcal{L}|} + \delta$ It's Schwartz-Zippel!

then there exists $\ddot{\breve{R}}_1, \ldots, \ddot{\breve{R}}_r \in \mathbb{F}_q[X]_{<\tau}, \ddot{\breve{f}} \in \mathbb{F}_q[X]_{<\tau \deg Q}$ such that

$$\ddot{\breve{P}}(X) := Q \circ \ddot{\breve{\mathbf{R}}}(X) \times T(X) = \ddot{\breve{f}}(X)Z(X).$$

Proof.
Take c_P and c_f closest codewords to P and f. Suppose by adventure that $\widehat{c_P}(X) \neq \widehat{c_f}(X)Z(X)$. We can prove that $\exists D \subseteq \mathcal{L}$ such that $|D| \geqslant (1-\delta)\mathcal{L}$, and $P_{|D} = c_{P|D}$ and $f_{|D} = c_{f|D}$. Then

$$\begin{aligned}\mathbb{P}_{x \in \mathcal{L}}(P(x) = f(x)Z(x)) &\leqslant \mathbb{P}_{x \in D}(P(x) = f(x)Z(x))\mathbb{P}(x \in D) + \mathbb{P}(x \notin D) \\ &= \mathbb{P}_{x \in D}(\widehat{c_P}(x) = \widehat{c_f}(x)Z(x))\frac{|D|}{|\mathcal{L}|} + 1 - \frac{|D|}{|\mathcal{L}|} \\ &\leqslant \frac{\deg P}{|D|}\frac{|D|}{|\mathcal{L}|} + \delta = \frac{\tau(\deg Q + 1)}{|\mathcal{L}|} + \delta.\end{aligned}$$

Fichtre! Diantre! Vertuchou! The adventure is over.

Thanks to this we have a **gap**!

$$\text{The computation is valid} \iff \begin{array}{l} \exists R_1, \ldots : \mathcal{L} \to \mathbb{F}_q, \exists f_1, \ldots : \mathcal{L} \to \mathbb{F}_q, \\ \begin{cases} \forall i, \Delta(R_i, \mathsf{RS}[\mathcal{L}, \tau]) < \delta \\ \forall t, \Delta(f_t, \mathsf{RS}[\mathcal{L}, \tau \deg Q_{I_t}]) < \delta \\ \forall t, \mathbb{P}(P(x) = f_t(x)Z(x)) > \frac{\tau(\deg Q_{I_t}+1)}{|\mathcal{L}|} + \delta \end{cases}\end{array}$$

Remark. The domain $\mathcal{L}$ and the roots H must be disjoint.

2.2 Locally-testable codes and IOPPs

The Verifier wants to test words so long he can't read them! Thus we need **locality**.

Definition 2.4 **Oracle**

Blackbox function that doesn't count in the complexity (but counts in query complexity).

Once the oracle is commited, it is trusted that it gives the commited value.

Example 2.5

Usually in complexity theory, oracles solve problems: a SAT oracle solves SAT:

$$\mathsf{SAT} : \varphi \mapsto \begin{cases} 1 & \text{if } \varphi \text{ is satisfiable} \\ 0 & \text{otherwise.} \end{cases}$$

Here we use any function $f : \mathcal{L} \to \mathbb{F}_q$ as oracle.

In practice. The Prover commits a Merkle tree and reveals the values on query. Our only security assumption is that the hash function used is secure and that the Prover can't create collisions. The rest is only combinatorics.

Definition 2.6 **Locally-testable code**

A code $\mathcal{C} \subseteq \mathbb{F}_q^n$ is (ℓ, δ, s)-locally-testable if there exists an algorithm $\mathcal{V}$ with only ℓ oracle queries such that for any $w \in \mathbb{F}_q^n$,

$$\text{if } w \in \mathcal{C} \text{ then } \mathbb{P}(\mathcal{V}^w \text{ accepts}) = 1 \qquad \text{(completeness)}$$

$$\text{if } d(w, \mathcal{C}) > \delta \text{ then } \mathbb{P}(\mathcal{V}^w \text{ accepts}) \leqslant s. \qquad \text{(soundness)}$$

Note that the soundness is not "if $w \in \mathcal{C}$". We need the gap!

Theorem 2.7 **Codes with constant rate, minimal distance and locality [DELLM21]**

There exists a family of codes with constant rate, minimal distance and locality: $\exists \ell \in \mathbb{N}, C$ is $(\ell, \delta, 1 - \kappa\delta)$-locally-testable.

I thought it was then possible to build more efficient $\mathbf{PCP}$ protocols so we studied those "3C codes" with Élina Roussel, but they are too rigid and complex to study efficient arithmetizations on them.

Example 2.8 **Reed-Solomon codes don't have a good locality**

For any $\ell \leqslant k$, $\delta > 0$ and $s < 1$, RS$[n, k]$ is not (ℓ, δ, s)-locally-testable.

Here, $k = \Omega(\tau)$ is the length of the computation → impossible for the Verifier

Definition 2.9 **$\mathbf{IOPP}$ [BCS16]**

Locality test with interaction instead of testing alone.

To remove the interaction, we apply a Fiat-Shamir heuristic that requires the Verifier to only send randomness and apply the checks at the end. The Prover then uses a hash function to create "pseudo-randomness".

2.3 Fast Reed-Solomon IOPP

FRI stands for Fast RS IOPP.

Idea. Instead of testing a polynomial, test its even and odd parts, which have degree $\deg f/2$! With $Y = X^2$, let

$$f(X) =: f_{\text{even}}(Y) + X f_{\text{odd}}(Y)$$

$$\text{Fold}[f, \alpha](Y) := f_{\text{even}}(Y) + \alpha f_{\text{odd}}(Y) = \frac{f(X) + f(-X)}{2} + \alpha \frac{f(X) - f(-X)}{2X}.$$

So the Verifier can compute Fold$[f, \alpha](y)$ with only 2 queries to f!

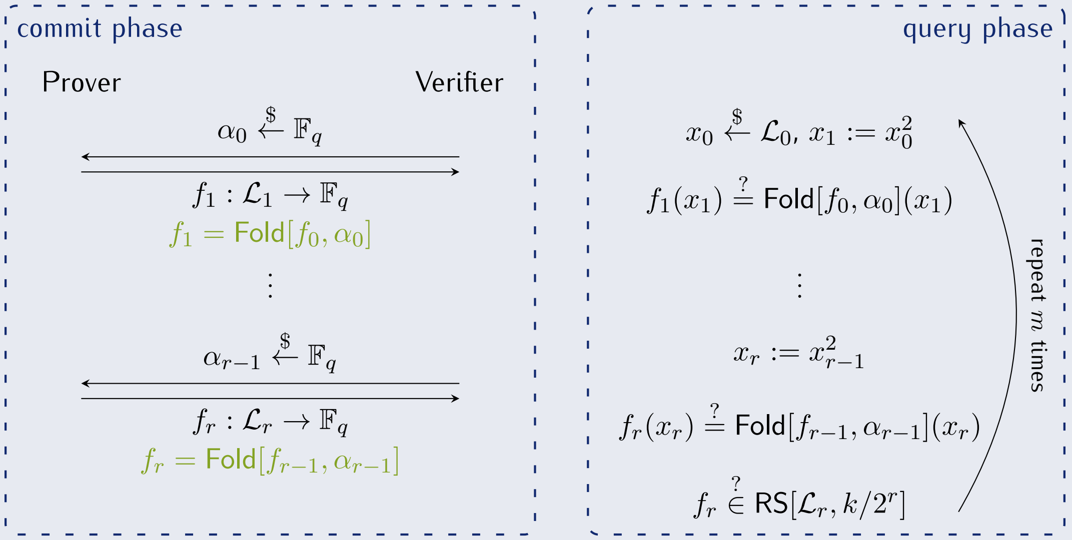
Domains. We require the Verifier to be able to reconstruct the Fold. So

$$\mathcal{L}_{\text{Fold}} := \{x^2 \mid x, -x \in \mathcal{L}\}.$$

Thus to apply successively the Fold, $\mathbb{F}_q$ must have 2^n primitive roots of unity.

Protocol 2.10 **FRI protocol [BBHR18a]**

Prover input: $f_0 : \mathcal{L}_0 \to \mathbb{F}_q$
Verifier input: oracle to f
Claim: $\Delta(f_0, \text{RS}[|\mathcal{L}_0|, k]) < \delta$



commit phase

Prover Verifier

$\alpha_0 \overset{\$}{\leftarrow} \mathbb{F}_q$

$f_1 : \mathcal{L}_1 \to \mathbb{F}_q$

$f_1 = \mathsf{Fold}[f_0, \alpha_0]$

$\vdots$

$\alpha_{r-1} \overset{\$}{\leftarrow} \mathbb{F}_q$

$f_r : \mathcal{L}_r \to \mathbb{F}_q$

$f_r = \mathsf{Fold}[f_{r-1}, \alpha_{r-1}]$

query phase

$x_0 \overset{\$}{\leftarrow} \mathcal{L}_0,\ x_1 := x_0^2$

$f_1(x_1) \overset{?}{=} \mathsf{Fold}[f_0, \alpha_0](x_1)$

$\vdots$

$x_r := x_{r-1}^2$

$f_r(x_r) \overset{?}{=} \mathsf{Fold}[f_{r-1}, \alpha_{r-1}](x_r)$

$f_r \overset{?}{\in} \mathsf{RS}[\mathcal{L}_r, k/2^r]$

repeat m times

The Verifier accepts iff all the tests pass.

2.4 Idea of the proof of soundness

The idea is, for $f : \mathcal{L} \to \mathbb{F}$

$$\deg \widehat{f} < 2k \iff \deg \widehat{f}_{\text{even}}, \deg \widehat{f}_{\text{odd}} < k \iff \underset{\alpha\in\mathbb{F}_q}{\mathbb{P}}\left(\deg\left(\mathsf{Fold}[\widehat{f}, \alpha]\right) < k\right) > \frac{1}{|\mathbb{F}_q|}$$

$$f \in \mathsf{RS}[2n, 2k] \iff f_{\text{even}}, f_{\text{odd}} \in \mathsf{RS}[n, k] \iff \underset{\alpha\in\mathbb{F}_q}{\mathbb{P}}(\mathsf{Fold}[f, \alpha] \in \mathsf{RS}[n, k]) > \frac{1}{|\mathbb{F}_q|}$$

Theorem 2.11 **Commit soundness [BCIKS20]**

Let $\mathsf{RS}_i = \mathsf{RS}[\mathcal{L}_i, k_i]$. Then $\forall \varepsilon > 0$, with $\delta_i := \min(\Delta(f_i, \mathsf{RS}_i), 1 - \sqrt{\rho} - \varepsilon)$.

$$\underbrace{\underset{\alpha\in\mathbb{F}_q}{\mathbb{P}}(\Delta(\mathsf{Fold}[f_i, \alpha], \mathsf{RS}_{i+1}) < \delta_i - \varepsilon)} \leqslant \frac{k^2}{(2\varepsilon)^7 |\mathbb{F}_q|}.$$

By the total probability,

$$\mathbb{P}(\mathcal{V}\text{ accepts}) \leqslant \mathbb{P}(\overbrace{\text{commit error}}) + \mathbb{P}\left(\mathcal{V}\text{ accepts} \mid \overline{\text{commit error}}\right).$$

Theorem 2.12 **FRI soundness [BKS18,BCIKS20]**

$$\mathbb{P}(\mathcal{V}\text{ accepts}) \leqslant \min_{\varepsilon > 0}\left(\frac{rk^2}{(2\varepsilon)^7 |\mathbb{F}_q|} + (1 - \delta_0 + r\varepsilon)^m\right)$$

Idea.
Since

$$\underset{x\in\mathcal{L}}{\mathbb{P}}(f_{i+1}(x) \neq \mathsf{Fold}[f_i, \alpha_i](x)) \overset{\text{def}}{=} \Delta(f_{i+1}, \mathsf{Fold}[f_i, \alpha_i]) \quad \text{and}$$

$$\Delta(\mathsf{Fold}[f_i, \alpha_i], \mathsf{RS}_{i+1}) \overset{\text{triangle}}{\leqslant} \Delta(\mathsf{Fold}[f_i, \alpha_i], f_{i+1}) + \Delta(f_{i+1}, \mathsf{RS}_{i+1}),$$

we have

$$\underset{x\in\mathcal{L}}{\mathbb{P}}(f_{i+1}(x) \neq \mathsf{Fold}[f_i, \alpha_i](x)) \geqslant \Delta(f_i, \mathsf{RS}_i) - \Delta(f_{i+1}, \mathsf{RS}_{i+1}) - \varepsilon.$$

By summing over all $i \in [r]$, we obtain the result.

Summary **FRI's principle**

Reduction from $f \in \mathsf{RS}[2n, 2k]$ to $f', f'' \in \mathsf{RS}[n, k]^2$ such that f', f'' can be computed with a constant number of queries to f.

Third part

Flowering protocol: proximity test to codes on graphs

3.1 Codes on graphs

Multigraph. Usually, a multigraph is $\Gamma = (V, E)$ where E is a multiset over V^2. Here we consider n-regular indexed multigraphs, so $E := V \times [n]/\sim$ where

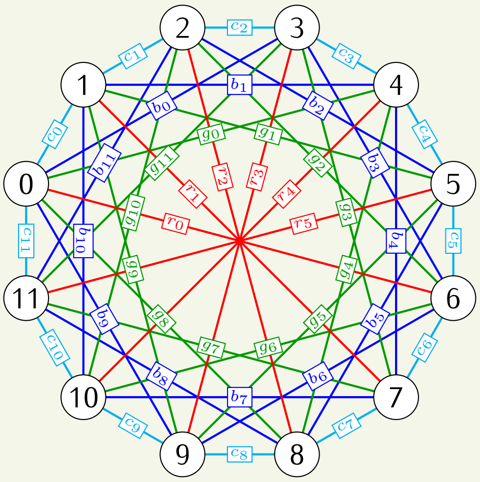
$$(v, \ell) \sim (v', \ell') \iff \ell = \ell' \text{ and } v, v' \text{ are neighbors with edge } \ell.$$

Definition 3.1 **Word on graph, code on graph**

A word on Γ is a labeling of the edges: $f : E \to \mathbb{F}_q$. Denote $f(v, \cdot)$ the vector $(f(v, \ell))_{\ell=1}^{n}$.
If Γ is n-regular and $C_0 \subseteq \mathbb{F}_q^n$, the code on Γ

$$\mathcal{C}[\Gamma, C_0] := \{f : E \to \mathbb{F}_q \mid \forall v, f(v, \cdot) \in C_0\}.$$

Example 3.2



$f = (c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9, c_{10}, c_{11}, b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9, b_{10}, b_{11}, g_0, g_1, g_2, g_3, g_4, g_5, g_6, g_7, g_8, g_9, g_{10}, g_{11}, r_0, r_1, r_2, r_3, r_4, r_5)$

We will only build our graphs on Reed-Solomon codes:

$$\mathcal{C}[\Gamma, k] := \mathcal{C}[\Gamma, \mathsf{RS}[n, k]].$$

Proposition 3.3 **Lower bound on dimension**

If Γ is n-regular, $\mathrm{rate}(\mathcal{C}[\Gamma, k]) \geqslant \dfrac{2k}{n} - 1$.

Idea.
By taking the $|V|$ parity check matrices of the local RS, we obtain a matrix with $|E|$ columns and $(n-k)|V|$ rows.

$$H = \begin{array}{c} \xleftrightarrow{\quad\quad\quad |E| \quad\quad\quad} \\ \begin{pmatrix} H_0^{(1)} \\ H_0^{(2)} \\ \vdots \\ H_0^{|V|} \end{pmatrix} \end{array} \Big\updownarrow (n-k)|V|$$

Thus $\dim \mathcal{C}[\Gamma, k] \geqslant |E| - (n-k)|V| = (k - n/2)|V|$.

Definition 3.4 **Vertex distance**

$\Delta_V(f, f') := \frac{1}{|V|}|\{v \in V \mid f(v, \cdot) \neq f'(v, \cdot)\}|$

Proposition 3.5

For $f, f' : E \to \mathbb{F}_q$, $\Delta_V(f, f') \geqslant \Delta(f, f')$.

3.2 Flowering protocol

From $f : E \to \mathbb{F}_q$, we want to create $f' : E' \to \mathbb{F}_q$ and $f'' : E'' \to \mathbb{F}_q$ such that

$$f \in \mathcal{C} \quad \Longleftrightarrow \quad f', f'' \in \mathcal{C}' \quad \Longleftrightarrow \quad \underset{\alpha \in \mathbb{F}_q}{\mathbb{P}}(\mathsf{Fold}[f, \alpha] \in \mathcal{C}') > s$$

Definition 3.6 **Cut-graph, cut-word**

If $\Gamma = (V, E)$ and $V' \subseteq V$, $\mathsf{Cut}[\Gamma, V'] := (V', E')$ where

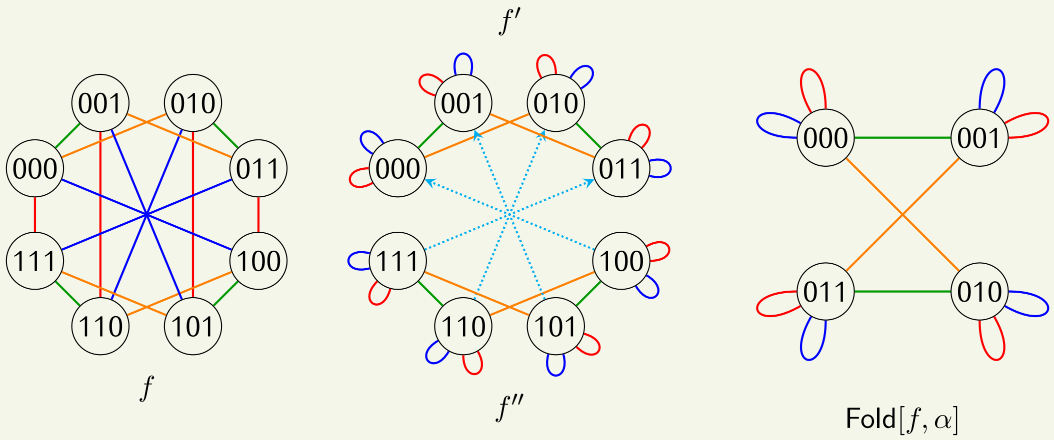
$$E'(v, \ell) := \begin{cases} E(v, \ell) & \text{if } E(v, \ell) \in V' \\ v & \text{otherwise.} \end{cases}$$

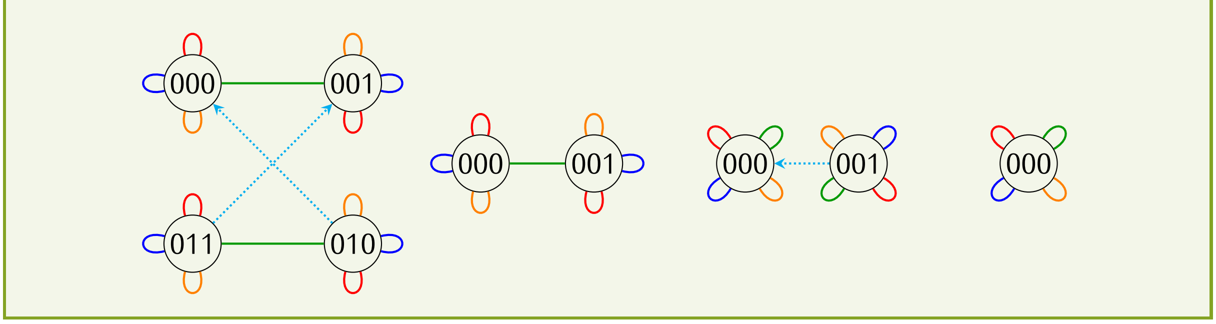
If $f : E \to \mathbb{F}_q$, $\mathsf{Cut}[f, V'] := f_{|V' \times [n]}$.

Definition 3.7 **Flowering cut**

If $V' \sqcup V'' = V$ and there exists a graph isomorphism $\varphi : \mathsf{Cut}[\Gamma, V'] \to \mathsf{Cut}[\Gamma, V'']$, $F = (V', \varphi)$ is a **flowering cut**.
Denote $\pi_\varphi : V \to V'$ the projection on V'.

Example 3.8





Let $f' := \mathsf{Cut}[f, V']$ and $f'' := \mathsf{Cut}[f, V'']$. Then

$$\mathsf{Fold}[f, \alpha](v, \ell) := f'(v, \ell) + \alpha f''(\varphi(v), \ell).$$

Protocol 3.9 **Flowering protocol [DMR25]**

Prover input: $f_0 : E_0 \to \mathbb{F}_q$
Verifier input: oracle to f
Claim: $\Delta_V(f_0, \mathcal{C}[\Gamma_0, k]) < \delta$

commit phase

Prover Verifier

$$\alpha_0 \overset{\$}{\leftarrow} \mathbb{F}_q \quad \xleftarrow{\hspace{3cm}}$$

$$\xrightarrow{\hspace{3cm}} \quad f_1 : E_1 \to \mathbb{F}_q$$

$f_1 = \mathsf{Fold}[f_0, \alpha_0]$

$\vdots$

$$\alpha_{r-1} \overset{\$}{\leftarrow} \mathbb{F}_q \quad \xleftarrow{\hspace{3cm}}$$

$$\xrightarrow{\hspace{3cm}} \quad f_r : E_r \to \mathbb{F}_q$$

$f_r = \mathsf{Fold}[f_{r-1}, \alpha_{r-1}]$

query phase

$v_0 \overset{\$}{\leftarrow} V_0,\ v_1 := \pi_1(v_0)$

$f_1(v_1, \cdot) \overset{?}{=} \mathsf{Fold}[f_0, \alpha_0](v_1, \cdot)$

$\vdots$

$v_r := \pi_r(v_{r-1})$

$f_r(v_r, \cdot) \overset{?}{=} \mathsf{Fold}[f_{r-1}, \alpha_{r-1}](v_r, \cdot)$

$f_r \overset{?}{\in} \mathsf{RS}[n, k]$

repeat m times

The Verifier accepts iff all the tests pass.

3.3 Idea of the proof of soundness

We use, for $f : E \to \mathbb{F}_q$ that

$$f \in \mathcal{C}[\Gamma, k] \iff f', f'' \in \mathcal{C}[\mathsf{Cut}(\Gamma, V'), k] \iff \mathop{\mathbb{P}}_{\alpha \in \mathbb{F}_q}(\mathsf{Fold}[f, \alpha] \in \mathcal{C}[\mathsf{Cut}(\Gamma, V'), k]) > s$$

Theorem 3.10 **Commit soundness [DMR25]**

Let $C_i = \mathcal{C}[\Gamma_i, k]$. Let $\delta_i := \Delta_V(f_i, C_i)$. Then $\forall \varepsilon > 0$,

$$\mathop{\mathbb{P}}_{\alpha \in \mathbb{F}_q}(\Delta(\mathsf{Fold}[f_i, \alpha], C_{i+1}) < \delta_i - \varepsilon) \leqslant \frac{1}{\varepsilon |\mathbb{F}_q|}.$$

Theorem 3.11 **Flowering soundness [DMR25]**

$$\mathbb{P}(\mathcal{V} \text{ accepts}) \leqslant \min_{\varepsilon > 0} \left(\frac{r}{\varepsilon |\mathbb{F}_q|} + (1 - \delta_0 + r\varepsilon)^m \right)$$

Idea.
Exacty like for the FRI soundness, since

$$\mathop{\mathbb{P}}_{v \in V}(f_{i+1}(v, \cdot) \neq \mathsf{Fold}[f_i, \alpha_i](v, \cdot)) \quad = \quad \Delta_V(f_{i+1}, \mathsf{Fold}[f_i, \alpha_i]),$$

we have

$$\underset{v\in V}{\mathbb{P}}(f_{i+1}(v,\cdot)\neq \mathsf{Fold}[f_i,\alpha_i](v,\cdot)) \quad \geqslant \quad \Delta_V(f_i,C_i) \;-\; \Delta_V(f_{i+1},C_{i+1}) \;-\; \varepsilon.$$

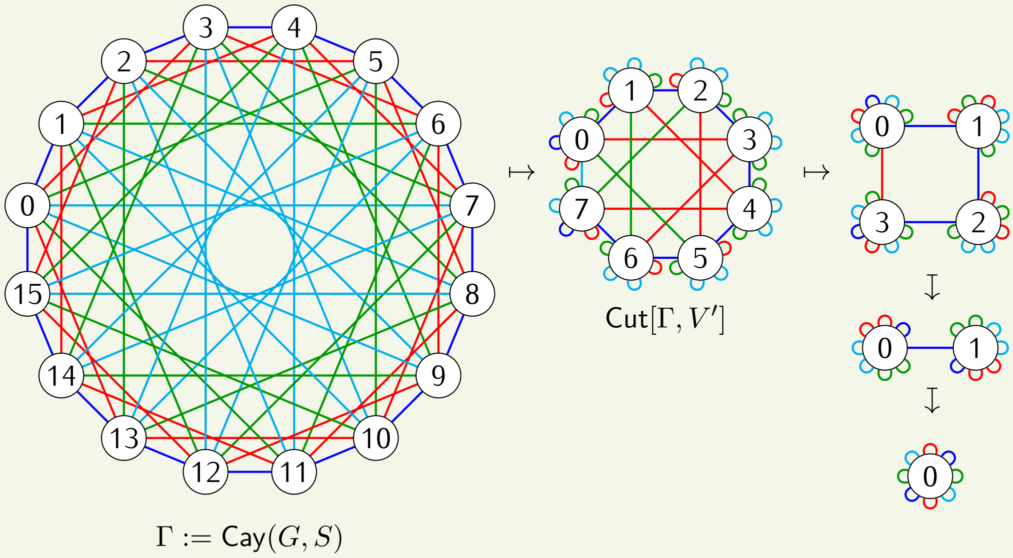
3.4 | Our codes

Definition 3.12 **Cayley graphs**

Let G be a group and $S=\{s_1,..,s_n\}\subseteq G$. Then $\mathsf{Cay}(G,S):=(V,E)$ where $V=G$ and $E:(v,i)\mapsto v+s_i$.

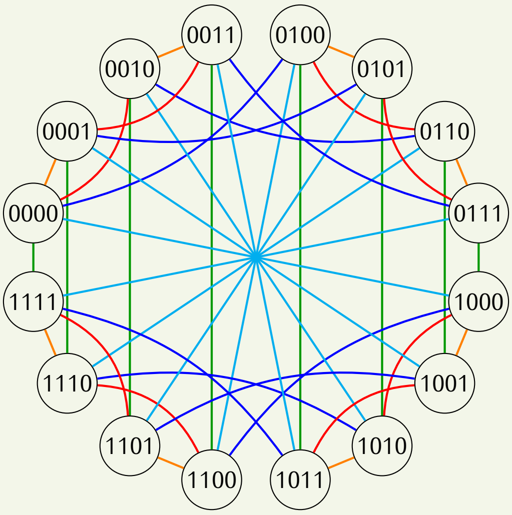
Example 3.13

Take $G=\mathbb{Z}/2^r\mathbb{Z}$ and $S=\{\pm1,\pm3,\pm5,\pm7\}$, $V'=\{0,...,2^{r-1}-1\}$ and $\varphi(v)=2^r-1-v$



Example 3.14 **Binary construction**

$G=\mathbb{F}_2^4$, $S=\{0001,0010,0100,1000,1111\}$



Proposition 3.15 **Parameters of the code (binary construction) [DMR25]**

With $\Gamma:=\mathsf{Cay}(\mathbb{F}_2^r,S)$ and $n:=|S|$,

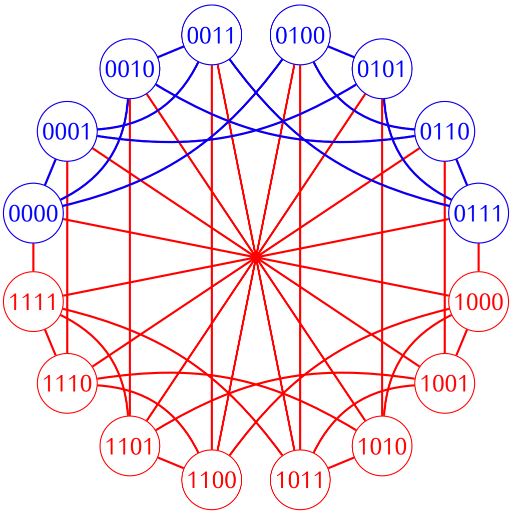
- **Length.** $N=n2^{r-1}$
- **Minimal distance.** If there exists a code $[n,n-r,d]_2$ then there exists S^a such that $\frac{1}{2}\delta\leqslant$

$$\Delta_H(\mathcal{C}[\Gamma, k]) \leqslant \delta, \text{ where } \delta := \frac{1}{2^{r-d+1}}\left(1 - \frac{k-1}{n}\right) = \frac{n2^{d-2}}{N}\left(1 - \frac{k-1}{n}\right).$$

---

[a] take the columns of a parity check matrix

Idea of the upper bound.
If there is $S' \subseteq S$ of size $n-k+1$ such that $|\langle S' \rangle| = 2^{d-1}$, then we can have a word with only nonzero coordinates in the subgraph Cay$[\langle S' \rangle, S']$.



An idea would be to take a MDS code with a bigger alphabet.

Proposition 3.16 **Parameters of the code (enlarge your alphabet) [DMR25]**

With $n = 2^m$, S the columns of a $[n, n-r, r+1]_{2^m}$ (MDS) code, and $\Gamma :=$ Cay$(\mathbb{F}_{2^m}^r, S)$,

- **Length.** $N = n2^{mr-1} = \dfrac{n^{r+1}}{2}$
- **Minimal distance.** $\Delta_H(\mathcal{C}[\Gamma, k]) \geqslant \dfrac{1}{2}\left(\dfrac{2}{n}\right)^r\left(1 - \dfrac{k-1}{n}\right) = \dfrac{2^{r-1}}{N}\left(1 - \dfrac{k-1}{n}\right)$

Summary **Flowering's idea**

We reduce testing proximity to $\mathcal{C}[\Gamma, k]$ to testing proximity to $\mathcal{C}[\Gamma', k]$ where Γ' is "twice" smaller, like the FRI for RS.

3.5 Is there an arithmetization?

The question now is: do we have an arithmetization like this?

$$\begin{array}{c}\text{The computation}\\ \text{of } A \text{ is valid}\end{array} \overset{?}{\iff} \begin{array}{c}\exists R_1, \ldots \text{ such that for}\\ \text{each } t,\ Q_t \circ \mathbf{R} \in \mathcal{C}[\Gamma, k]\end{array}$$

On the one hand we can write a computation as a graph.

Computation as a regular graph

- **Circuits.** We can represent a computation as a circuit rather than a program.
- **De Bruijn graph.** There are regular graphs that allow to represent circuits.

And on the other hand, we can prove statements on the local node views. Given $\Gamma(V, E)$, $P : E \to \mathbb{F}_q$ and $Z \in \mathbb{F}_q[X]$, we can use a variant of that Flowering protocol to prove,

$$\exists f : E \to \mathbb{F}_q, \forall v \in V, \widehat{P}(v, X) = Z(X) \times \widehat{f}(v, X),$$

i.e. if $Z(X) = \prod_{h\in H}(X - h)$,

$$\forall v \in V, \widehat{P}(v, X) \text{ cancels on } H.$$

However, if the R_i are $E \to \mathbb{F}_q$, then composing with a polynomial constraint does not give a word on a graph.

That's what I will work on next.