ZeroMorph 笔记

- Yu Guo yu.guo@secbit.io

Zeromorph [KT23] 是一个基于 KZG10 的 MLE 多项式承诺方案。事实上，Zeromorph 方案是一个更一般性的框架，可以基于不同的 Univariate Polynomial Commitment 方案，比如 FRI-based Zeromorph 方案。

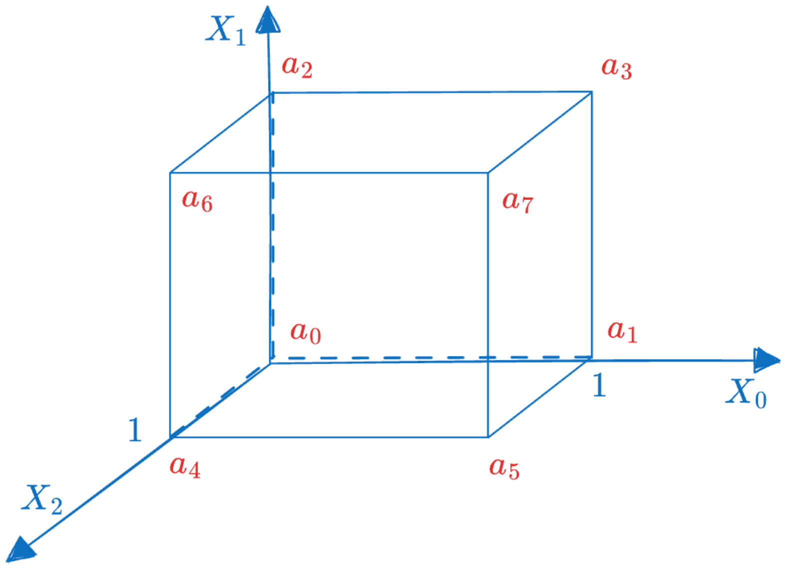
Zeromorph 的核心要点是将 MLE 多项式的 Evaluations 即「点值向量」作为 Univariate 多项式的「系数向量」。这个做法看起来有些奇怪，不过这个框架仍然不失清晰简洁性。

理解 Zeromorph 的关键在于对高维的 Boolean HyperCube 上取值的变换的理解，以及它们如何对应于 Univariate 多项式的运算。

MLE 多项式

所谓的 MLE（Multilinear Extension） 多项式 $\tilde{f}$ 是定义在 Boolean HyperCube 上的一类 Multivariate 多项式。它的每一项中任何一个未知数的次数都不超过 1，例如 $\tilde{f} = 1 + 2X_0 + 3X_1X_0$ 是一个 MLE 多项式，而 $\tilde{f}' = 1 + 2X_0^2 + 3X_1X_0 + X_1$ 则不是，因为 X_0^2 的次数大于 1 。

一个 MLE 多项式可以对应到一个从 Boolean 向量到一个有限域的函数，即 $f:\{0,1\}^n \to \mathbb{F}_q$，我们则称其维度为 n 。下图是一个三维的 MLE 多项式 $\tilde{f}(X_0, X_1, X_2)$ 的示例，这个多项式可以唯一地被 $(a_0, a_1, \ldots, a_7)$ 这个「点值向量」来表示。这对应于 Univariate 多项式中的「点值式」表示，即 Evaluations form。



当然一个 MLE 多项式也可以采用「系数式」来表示，即 Coefficients form ，表示如下：

$$\tilde{f}(X_0, X_1, \ldots, X_{n-1}) = \sum_{i_0=0}^{1}\sum_{i_1=0}^{1}\cdots\sum_{i_{n-1}=0}^{1} f_{i_0 i_1 \cdots i_{n-1}} X_0^{i_0} X_1^{i_1} \cdots X_{n-1}^{i_{n-1}} \tag{1}$$

对于上图三维 MLE 多项式的例子，我们可以将它写为：

$$\tilde{f}(X_0, X_1, X_2) = f_0 + f_1X_0 + f_2X_1 + f_3X_2 + f_4X_0X_1 + f_5X_0X_2 + f_6X_1X_2 + f_7X_0X_1X_2 \tag{2}$$

其中 $(f_0, f_1, \ldots, f_7)$ 为 MLE 多项式的系数向量。注意因为 MLE 多项式属于多元多项式（Multivariate Polynomial），任何表示方式都需要事先确定多项式中的项的排序顺序，本文以及后续讨论我们都基于 Lexicographic Order。

对于 MLE 多项式的「点值式」表示，我们可以定义为：

$$\tilde{f}(X_0, X_1, \ldots, X_{n-1}) = \sum_{i_0=0}^{1}\sum_{i_1=0}^{1}\cdots\sum_{i_{n-1}=0}^{1} a_{i_0 i_1 \cdots i_{n-1}} \cdot eq(i_0, i_1, \ldots, i_{n-1}, X_0, X_1, \ldots, X_{n-1}) \tag{3}$$

其中 eq 为 一组关于 n 维 Boolean HyperCube $\{0,1\}^n$ 的 Lagrange Polynomial：

$$eq(i_0, i_1, \ldots, i_{n-1}, X_0, X_1, \ldots, X_{n-1}) = \prod_{j=0}^{n-1}\Big((1-i_j)\cdot(1-X_j) + i_j \cdot X_j\Big) \tag{4}$$

MLE 多项式在「点值式」和「系数式」之间存在 $N\log(N)$ 的转换算法，这里不再深入讨论。

我们可以使用 ZeroMorph 将一个 MLE 多项式映射到一个 Univariate 多项式，具体一点说，是将 MLE 多项式在 Boolean HyperCube 上的「点值向量」映射到一个 Univariate 多项式 的「系数向量」。

MLE 多项式到 Univariate 多项式

我们以一个简单的例子来快速了解下这个映射关系。考虑下面一个维度只有 2 的 MLE 多项式：

$$\tilde{f}(X_0,X_1)=2+X_1+X_0X_1 \tag{5}$$

容易验证，它在 Boolean HyperCube 上的点值表示为：

$$\begin{aligned}\tilde{f}(0,0)=2\\ \tilde{f}(1,0)=2\\ \tilde{f}(0,1)=3\\ \tilde{f}(1,1)=4\end{aligned} \tag{6}$$

如果采用 ZeroMorph 方案，它可以映射到如下的 Univariate 多项式：

$$\hat{f}(X)=2+2\cdot X+3\cdot X^2+4\cdot X^3 \tag{7}$$

假设我们有一个 Univariate 多项式的承诺方案，那么我们就可以计算映射后的 Univariate 多项式的承诺。例如，假设我们有以下的 KZG10 承诺方案的 SRS：

$$SRS=([1]_1,[\tau]_1,[\tau^2]_1,[\tau^3]_1,\ldots,[\tau^D]_1,[1]_2,[\tau]_2,[\tau^2]_2,[\tau^3]_2,\ldots,[\tau^D]_2) \tag{8}$$

根据 KZG10 的承诺算法，我们计算 $\hat{f}(X)$ 的承诺如下：

$$\mathsf{cm}(\hat{f})=2\cdot[1]_1+2\cdot[\tau]_1+3\cdot[\tau^2]_1+4\cdot[\tau^3]_1 \tag{9}$$

后续我们用符号 $[[\tilde{f}]]$ 来表示 MLE 多项式 $\tilde{f}$ 映射所对应的 Univariate 多项式。

多项式映射

这一节，我们讨论下更多的映射情况。为了简化起见，我们先考虑三维 MLE 的情况，即 $\tilde{f}\in\mathbb{F}_q[X_0,X_1,X_2]^{\leq 1}$。

假设 $\tilde{f}$ 只是一个常数多项式，即它的系数向量只有第一项非零，其余元素都为零。多项式可以表示如下：

$$\tilde{c}(X_0,X_1,X_2)=c_0 \tag{10}$$

我们考虑下这样一个常数多项式会映射到一个什么样的 Univariate 多项式。首先我们要把它转换成点值式，考虑在一个三维的 Boolean HyperCube 上，无论 $X_0,X_1,X_2\in\{0,1\}$ 如何取值，这个多项式在 Boolean HyperCube 上的取值都为 c_0，那么这也意味着它的点值式为 $(c_0,c_0,c_0,\ldots,c_0)$，于是它所对应的 Univariate 多项式为：

$$\begin{aligned}[[\tilde{c}]]&=c_0+c_0X+c_0X^2+c_0X^3+\ldots+c_0X^7\\ &=c_0\cdot(1+X+X^2+X^3+\ldots+X^7)\end{aligned} \tag{11}$$

那我们再考虑一个二维的 MLE 多项式 $\tilde{c}'(X_0,X_1)$，它同样是一个常数多项式，即 $\tilde{c}'(X_0,X_1)=c_0$，那么它对应的 Univariate 多项式为：

$$\begin{aligned}[[\tilde{c}']]&=c_0+c_0X+c_0X^2+c_0X^3\\ &=c_0\cdot(1+X+X^2+X^3)\end{aligned} \tag{12}$$

我们可以看到，虽然两个 MLE 多项式 $\tilde{c}$ 和 $\tilde{c}'$ 的系数式表示完全一样，但它们映射到的 Univariate 多项式并不一样。这是因为不管是 Univariate 还是 Multivariate 的多项式，它们的点值式表示都隐含了 Evaluation Domain 的选取。$\tilde{c}$ 的 Evaluation Domain 是 3 维的 Boolean HyperCube，而 $\tilde{c}'$ 的 Evaluation Domain 是 2 维的 Boolean HyperCube。因此，当我们计算多项式的点值式时，需要明确下 Evaluation Domain 的选择，对于 MLE 多项式来说，如果它的 Evaluation Domain 是 n 维的 Boolean HyperCube，那么我们修改下映射记号表示，在映射括加上下标 n，即 $[[\tilde{f}]]_n$ 。下面是 $\tilde{c}$ 在两个不同的 Evaluation Domain 上的映射所产生的两个不同的 Univariate 多项式：

$$\begin{aligned}[[\tilde{c}]]_2&=c_0+c_0X+c_0X^2+c_0X^3\\ [[\tilde{c}]]_3&=c_0+c_0X+c_0X^2+c_0X^3+c_0X^4+c_0X^5+c_0X^6+c_0X^7\end{aligned} \tag{13}$$

映射的加法同态

对于任意的两个 MLE 多项式，如果它们具有相同的维度，比如 $\tilde{f}_1(X_0,X_1)$ 和 $\tilde{f}_2(X_0,X_1)$ ，假如前者的点值式表示为

$$\tilde{f}_1(X_0,X_1) = v_0 \cdot eq(0,0,X_0,X_1) + v_1 \cdot eq(0,1,X_0,X_1) + v_2 \cdot eq(1,0,X_0,X_1) + v_3 \cdot eq(1,1,X_0,X_1) \tag{14}$$

$$\tilde{f}_2(X_0,X_1) = v'_0 \cdot eq(0,0,X_0,X_1) + v'_1 \cdot eq(0,1,X_0,X_1) + v'_2 \cdot eq(1,0,X_0,X_1) + v'_3 \cdot eq(1,1,X_0,X_1) \tag{15}$$

那么它们的和为：$\tilde{f}_1(X_0,X_1) + \tilde{f}_2(X_0,X_1)$，其点值式为：

$$\begin{aligned}\tilde{f}_1(X_0,X_1) + \tilde{f}_2(X_0,X_1) = (v_0+v'_0) \cdot eq(0,0,X_0,X_1) + (v_1+v'_1) \cdot eq(0,1,X_0,X_1) \\ + (v_2+v'_2) \cdot eq(1,0,X_0,X_1) + (v_3+v'_3) \cdot eq(1,1,X_0,X_1)\end{aligned} \tag{16}$$

于是下面的等式成立：

$$[[\tilde{f}_1(X_0,X_1) + \tilde{f}_2(X_0,X_1)]]_2 = [[\tilde{f}_1(X_0,X_1)]]_2 + [[\tilde{f}_2(X_0,X_1)]]_2 \tag{17}$$

同时不难证明，上面的等式对任意的同样维度的 MLE 多项式都成立。另外也不难证明：

$$[[\alpha \cdot \tilde{f}]]_n = \alpha \cdot [[\tilde{f}]]_n, \quad \forall \alpha \in \mathbb{F}_q \tag{18}$$

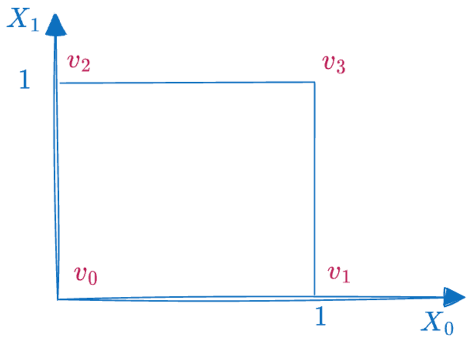
因此，我们说 $[[\tilde{f}]]_n$ 这个映射具有多项式加法同态性，并且是一个一一映射（Injective and Surjective）。

低维到高维的映射

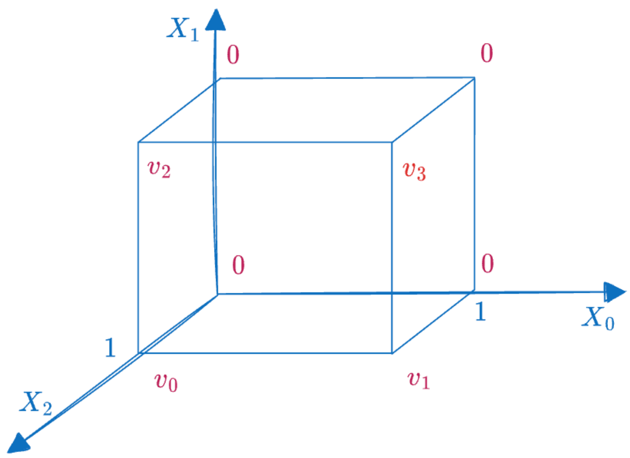
我们考虑更一般多项式的情况，假设一个二维的 MLE 多项式 $\tilde{c}(X_0,X_1)$，它在二维 Boolean HyperCube 上的取值为 (v_0,v_1,v_2,v_3)，那么它对应的 Univariate 多项式为：

$$[[\tilde{c}]]_2 = v_0 + v_1X + v_2X^2 + v_3X^3 \tag{19}$$

而 $X_2 \cdot \tilde{c}(X_0,X_1)$ 也同样是一个 MLE 多项式，维度为 3 。它在 Boolean HyperCube 上的取值为：$(0,0,0,0,v_0,v_1,v_2,v_3)$ ，即前四项为零，后四项等于 $\tilde{c}(X_0,X_1)$ 在二维 MLE 多项式在 的取值，如下图所示：



$\tilde{c}(X_0,X_1)$



$X_2 \cdot \tilde{c}(X_0,X_1)$

这个容易解释，因为当 $X_2=0$ 时，整体多项式取值为零，于是 X_0,X_1 构成的二维正方形顶点上的值都为零，而当 $X_2=1$ 时，多项式 $X_2 \cdot \tilde{c}(X_0,X_1)$ 等于 $\tilde{c}(X_0,X_1)$ 。因此 $X_2=1$ 的平面正方形的顶点上的值等于 $\tilde{c}(X_0,X_1)$ ，进一步我们可以有这样的结论：

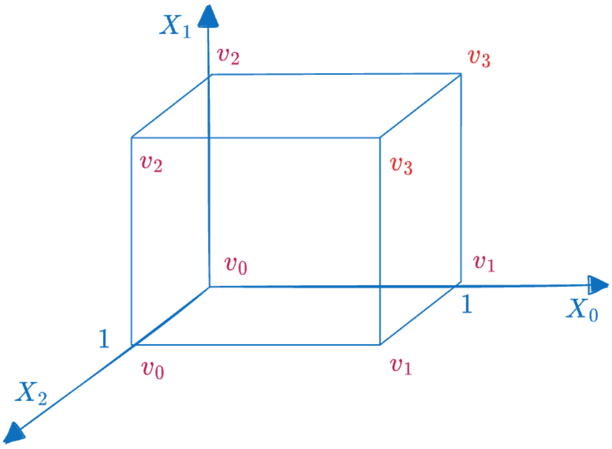
$$[[X_2 \cdot \tilde{c}]]_3 = X^4 \cdot [[\tilde{c}]]_2 \tag{20}$$

快速推导如下：

$$[[X_2 \cdot \tilde{c}]]_3 = v_0X^4 + v_1X^5 + v_2X^6 + v_3X^7 = X^4 \cdot (v_0 + v_1X + v_2X^2 + v_3X^3) = X^4 \cdot [[\tilde{c}]]_2 \tag{21}$$

这里的 X^4 推高了 $[[\tilde{c}]]_2$ 的次数，使得它能够刚好放在 3 维的 HyperCube 的高位区域（即 $X_2=1$ 的区域）。

接下来考虑下 $\tilde{c}$ 在三维 HyperCube 上的取值，我们会发现新增加的未知数 X_2，不管取值为 0 还是 1，多项式的取值只和 X_0,X_1 有关，因此，它的点值式等于二维点值向量复制一份，从而填满 3 维的 HyperCube，如下图所示：



$\tilde{c}(X_0, X_1)$

换句话说，$\tilde{c}$ 在三维 HyperCube 上的点值式为 $(v_0, v_1, v_2, v_3, v_0, v_1, v_2, v_3)$，那么它所映射到的 Univariate 多项式为：

$$\begin{aligned}[[\tilde{c}]]_3 &= v_0 + v_1X + v_2X^2 + v_3X^3 + v_0X^4 + v_1X^5 + v_2X^6 + v_3X^7 \\ &= (1 + X^4) \cdot (v_0 + v_1X + v_2X^2 + v_3X^3) \\ &= (1 + X^4) \cdot [[\tilde{c}]]_2\end{aligned} \tag{22}$$

上面的等式可以这么解释：三维 HyperCube 上的取值由两部分拼接而成，$[[\tilde{c}]]_2$ 与由 X^4 推高次数的 $[[\tilde{c}]]_2$。

同理可推，$\tilde{c}$ 在四维 HyperCube 上的取值为 $(v_0, v_1, v_2, v_3, \quad v_0, v_1, v_2, v_3, \quad v_0, v_1, v_2, v_3, \quad v_0, v_1, v_2, v_3)$，那么它所映射到的 Univariate 多项式为：

$$\begin{aligned}[[\tilde{c}]]_4 &= v_0 + v_1X + v_2X^2 + v_3X^3 + v_0X^4 + v_1X^5 + v_2X^6 + v_3X^7 \\ &= (1 + X^4 + X^8 + X^{12}) \cdot (v_0 + v_1X + v_2X^2 + v_3X^3) \\ &= (1 + X^4 + X^8 + X^{12}) \cdot [[\tilde{c}]]_2\end{aligned} \tag{23}$$

把低维的 MLE 拉升到一个高维的 HyperCube 上，就会出现低维 HyperCube 不断复制自己的现象。我们可以定义一个新的多项式函数，$\Phi_k(X)$，来表示这种重复的操作：

$$\Phi_k(X^h) = 1 + X^h + X^{2h} + \ldots + X^{(2^k-1)h} \tag{24}$$

显然，$[[\tilde{c}]]_4 = \Phi_2(X^4) \cdot [[\tilde{c}]]_2$。

MLE 多项式的余数定理

> TODO: 这个余数定理的正确称呼?

接下来的问题是如何这个 MLE 到 Unvariate 多项式映射来实现 MLE 的 Evaluation Argument 协议。具体点说，问题是如何利用 cm$(\tilde{f})$ 来验证 $\tilde{f}$ 在某个点的取值的正确性，比如 $\tilde{f}(u_0, u_1)$? 我们虽然已经有一个基于 KZG10 的 Evaluation Argument 协议，可惜是基于 Univariate 多项式，而非 MLE 多项式。KZG10 利用了多项式余数定理，如下公式

$$\hat{f}(X) - \hat{f}(z) = q(X) \cdot (X - z) \tag{25}$$

将商多项式 $q(X)$ 的承诺 cm(q) 作为 Evaluation Argument 的证明。那么我们如何将 MLE 在一个多维的点，比如 $(u_0, u_1, \ldots, u_{n-1})$ 上的求值证明问题，转化为 Univariate 多项式在一个点 或者多个点上的求值证明呢?

论文 [PST13] 给出了一个上述定理的多元多项式版本：

$$f(X_0, X_1, \ldots, X_{n-1}) - f(u_0, u_1, \ldots, u_{n-1}) = \sum_{k=0}^{n-1} q_k(X_0, X_1, \ldots, X_{n-1}) \cdot (X_k - u_k) \tag{26}$$

如果 $f(X_0, X_1, \ldots, X_{n-1})$ 是一个 MLE 多项式，那么它可以被简化为下面的公式：

$$\tilde{f}(X_0, X_1, \ldots, X_{n-1}) - \tilde{f}(u_0, u_1, \ldots, u_{n-1}) = \sum_{k=0}^{n-1} \tilde{q}_k(X_0, X_1, \ldots, X_{k-1}) \cdot (X_k - u_k) \tag{27}$$

这是因为 MLE 多项式 $f(X_0, X_1, \ldots, X_{n-1})$ 中每一个未知数的最高次数为 1，对于 $f(X_0, X_1, \ldots, X_k)$，它除以 $(X_k - u_k)$ 因式之后，余数多项式中将不再含有未知数 X_k ，所以当 $f(X_0, X_1, \ldots, X_{n-1})$ 依次除以 $(X_{n-1} - u_{n-1})$ 到 $(X_0 - u_0)$ 这些因式，我们得到的商多项式和余数多项式中的未知数个数一直在逐个减少，直到最后得到一个常数的商多项式 $\tilde{q}_0$，当然还有一个常数的余数多项式，而后者正好是 MLE 多项式在 $(u_0, u_1, \ldots, u_{n-1})$ 处的求值。

我们假设这个最后的求值为 v ，即

$$\tilde{f}(u_0, u_1, \ldots, u_{n-1}) = v \tag{28}$$

那么我们对余数定理等式的左右两边（都看成是一个 MLE 多项式）分别进行 Zeromorph 映射，得到对应的 Univariate 多项式。

$$[[\tilde{f}(X_0, X_1, \ldots, X_{n-1}) - v]]_n = [[\sum_{k=0}^{n-1} \tilde{q}_k(X_0, X_1, \ldots, X_{k-1}) \cdot (X_k - u_k)]]_n \tag{29}$$

由于映射具有加法的同态性，因此我们可以继续化简上面的等式：

$$\begin{aligned}
[[\tilde{f}(X_0, X_1, \ldots, X_{n-1})]]_n - [[v]]_n &= \sum_{k=0}^{n-1} [[\tilde{q}_k(X_0, X_1, \ldots, X_{k-1}) \cdot (X_k - u_k)]]_n \\
&= \sum_{k=0}^{n-1} \Big([[X_k \cdot \tilde{q}_k(X_0, X_1, \ldots, X_{k-1})]]_n - u_k [[\tilde{q}_k(X_0, X_1, \ldots, X_{k-1})]]_n \Big)
\end{aligned} \tag{30}$$

先看等式左边的 $[[\tilde{f}(X_0, X_1, \ldots, X_{n-1})]]_n$ 这一项直接映射到 $\hat{f}(X)$，再看 $[[v]]_n$ 这一项，它映射到 $\hat{v}(X)$，

$$[[v]]_n = \hat{v}(X) = v + vX + vX^2 + \ldots + vX^{n-1} \tag{31}$$

或者我们改用 $\Phi_n(X)$ 函数来表示：

$$[[v]]_n = v \cdot \Phi_n(X) \tag{32}$$

看下等式右边的 $[[\tilde{q}_k(X_0, X_1, \ldots, X_{k-1})]]_n$，这一项是将 k 维的 HyperCube填充到 n 维的 HyperCube 上，然后再进行映射。根据前面的讨论，我们需要将 k 维的 HyperCube 连续复制 2^{n-k} 次，从而填满 n 维 HyperCube：

$$[[f(X_0, X_1, \ldots, X_{k-1})]]_n = \Phi_{n-k}(X^{2^k}) \cdot [[f(X_0, X_1, \ldots, X_{k-1})]]_k \tag{33}$$

再解释下，因为 $\Phi_{n-k}(X^{2^k})$ 表示了一个间隔为 2^k 的系数向量，其定义展开如下：

$$\Phi_{n-k}(X^{2^k}) = 1 + X^{2^k} + X^{2 \cdot 2^k} + \ldots + X^{(2^{n-k}-1) \cdot 2^k} \tag{34}$$

它的系数向量为：

$$(1, 0, 0, \ldots, 0, \quad 1, 0, \ldots, 0, \quad 1, 0, \ldots, 0, \quad 1) \tag{35}$$

假设有一个次数受限的多项式 $g(X) \in \mathbb{F}_q[X]$，满足 $\deg(g) < 2^k$ ，那么多项式 $\Phi_{n-k}(X^{2^k}) \cdot g$ 就表示了一个 $2^k - 1$ 次多项式 $g(X)$ 被 2^k 间隔的系数向量重复了 2^{n-k} 次，最终得到了一个 $2^n - 1$ 次的多项式。

最后还剩下 $[[X_k \cdot \tilde{f}(X_0, X_1, \ldots, X_{k-1})]]_n$ 这项，如何继续化简它呢？

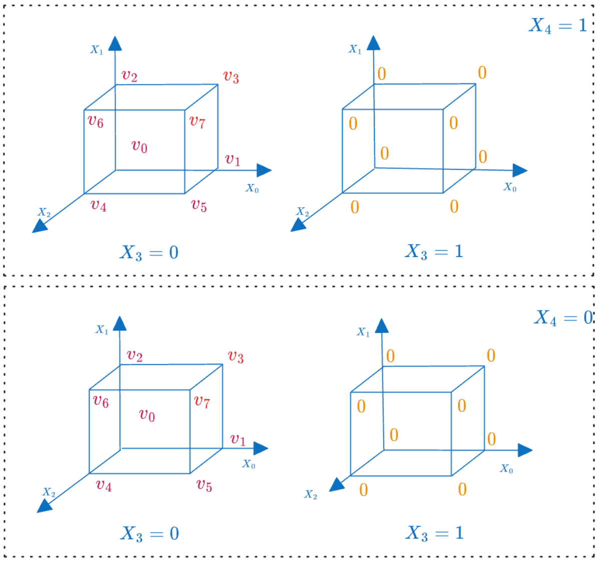
我们可以分两步来构造它的映射，首先看 $\tilde{f}(X_0, X_1, \ldots, X_{k-1})$ 可以由一个 k 维 Hypercube 表示，然后当乘以一个新的未知数 X_k，它就变成了一个 $k+1$ 维的 HyperCube。而这个新 Hypercube 可以分为两部分，一部分都是零（当 $X_k = 0$ 时），另一部分正是 $\tilde{f}(X_0, X_1, \ldots, X_{k-1})$ 。所以我们先利用 $\Phi_n(X)$ 函数，构造一个 HyperCube 的重复模式，其中间隔为 2^{k+1}，然后把 k 维 HyperCube 进行 2^{n-k-1} 次重复，于是我们得到了下面的多项式。

$$\Phi_{n-k-1}(X^{2^{k+1}}) \cdot [[\tilde{f}(X_0, X_1, \ldots, X_{k-1})]]_k \tag{36}$$

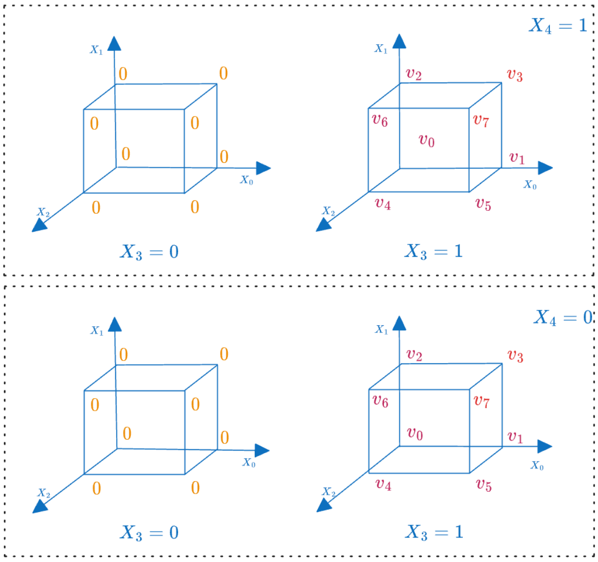
不过这只是第一步。上面这个 Univariate 多项式和 $[[X_k \cdot \tilde{f}(X_0, X_1, \ldots, X_{k-1})]]_n$ 还不相等，因为前者在每一个重复的 $k+1$ 维 HyperCube 中，$X_k = 1$ 部分为零，而 $X_k = 0$ 部分放的是 k 维 HyberCube $\tilde{f}(X_0, X_1, \ldots, X_{k-1})$，这与我们想要的 HyperCube 相反。我们需要再为它补上 X^{2^k} 这样的移位因子，这样就可以调换 X_k 所对应的 k 维的 HyperCube 的位置（从低位区域转移到高位区域）：

$$[[X_k \cdot \tilde{f}(X_0, X_1, \ldots, X_{k-1})]]_n = X^{2^k} \cdot \Phi_{n-k-1}(X^{2^{k+1}}) \cdot [[\tilde{f}(X_0, X_1, \ldots, X_{k-1})]]_k \tag{37}$$

下图用一个特定的例子演示，其中 $k = 3, n = 5$， 左边为移位前的 5 维 HyperCube，其中上下两半场表示第五个维度，每个半场有两个三维的立方体，表示第四个维度。我们可以看到，仅当 $X_3 = 0$ 的三维立方体恰好对应 $\tilde{f}(X_0, X_1, X_2)$，而当 $X_3 = 1$ 时，三维立方体上全为零。而下图右边为移位后的 5 维 HyperCube，其中的 $\tilde{f}(X_0, X_1, X_2)$ 立方体被移位到了右边，也就是 $X_3 = 1$ 所对应的区域。



$\Phi_2(X^{2^3}) \cdot [[\tilde{f}(X_0, X_1, X_2)]]_3$



$X^{2^3} \cdot \Phi_2(X^{2^3}) \cdot [[\tilde{f}(X_0, X_1, X_2)]]_3$

到此，我们可以得到 Zeromorph 协议的关键等式：

$$[[\tilde{f}(X_0, X_1, \ldots, X_{n-1})]]_n - v \cdot \Phi_n(X) = \sum_{k=0}^{n-1} \Big(X^{2^k} \cdot \Phi_{n-k-1}(X^{2^{k+1}}) - u_k \cdot \Phi_{n-k}(X^{2^k}) \Big) \cdot [[\tilde{q}_k(X_0, X_1, \ldots, X_{k-1})]]_k \quad (38)$$

基于 KZG10 的 Evaluation Argument

注意到上节我们推导出的 Zeromorph 等式是一个关于 Univariate 多项式的等式，我们简写为：

$$\hat{f}(X) - v \cdot \Phi_n(X) = \sum_k \Big(X^{2^k} \cdot \Phi_{n-k-1}(X^{2^{k+1}}) - u_k \cdot \Phi_{n-k}(X^{2^k}) \Big) \cdot \hat{q}_k(X) \quad (39)$$

这里 $\hat{f}(X)$ 与 $\hat{q}_k(X)$ 定义如下：

$$\begin{aligned} \hat{f}(X) &= [[\tilde{f}(X_0, X_1, \ldots, X_{n-1})]]_n \\ \hat{q}_k(X) &= [[\tilde{q}_k(X_0, X_1, \ldots, X_{k-1})]]_k \end{aligned} \quad (40)$$

我们要证明 $\tilde{f}(X_0, X_1, \ldots, X_{n-1})$ 在点 $(u_0, u_1, \ldots, u_{n-1})$ 处的取值为 v，那么我们只需要检查上面的多项式是否相等即可。这里利用 Schwartz-Zippel 引理的思想，让 Verifier 随机挑选一个点 $X = \zeta$，然后让 Prover 提供 $\hat{f}(\zeta)$ 和 $\hat{q}_k(\zeta)$ 的值，以便于 Verifier 验证下面的等式是否成立：

$$\hat{f}(\zeta) - v \cdot \Phi_n(\zeta) = \sum_k \Big(\zeta^{2^k} \cdot \Phi_{n-k-1}(\zeta^{2^{k+1}}) - u_k \cdot \Phi_{n-k}(\zeta^{2^k}) \Big) \cdot \hat{q}_k(\zeta) \quad (41)$$

不过这还不够，因为 Prover 实际上承诺的是 $\hat{q}_k(X)$，为了保证 MLE 余数多项式关系成立，我们要强制要求所有的商多项式 $\hat{q}_k(X)$ 的次数都小于 2^k，即 $\deg(\hat{q}_k) < 2^k$，以确保 Prover 没有作弊的空间。

不管是 FRI 还是 KZG10，都提供了证明 $\deg(\hat{q}_k) < 2^k$ 的方法。本文我们仅考虑基于 KZG10 的 Zeromorph 协议。一个简单的基于 KZG10 的 Degree Bound 证明协议如下：

- Prover 提供 $\mathsf{cm}(\hat{q}_k)$ 并附加上 $\mathsf{cm}(X^{D-2^k+1} \cdot \hat{q}_k(X))$ 发送给 Verifier，
- Verifier 验证下面的等式：

$$e\big(\mathsf{cm}(\hat{q}_k),\ [\tau^{D-2^k+1}]_2\big) = e\big(\mathsf{cm}(X^{D-2^k+1} \cdot \hat{q}_k(X)),\ [1]_2\big) \quad (42)$$

这里 $X^{D-2^k+1} \cdot \hat{q}_k(X)$ 的作用是把 $\hat{q}_k(X)$ 的 Degree 对齐到 D。因为 KZG10 的 SRS 中，能承诺的多项式的 Degree 最多为 D，所以如果 $\hat{q}_k(X)$ 的 Degree 超过了 2^k，那么 $\deg(X^{D-2^k+1} \cdot \hat{q}) > D$，这样就无法用 KZG10 的 SRS 中进行承诺。反之如果 Prover 可以正确承诺 $X^{D-2^k+1} \cdot \hat{q}_k(X)$，那就证明了 $\deg(\hat{q}_k) < 2^k$。

协议描述

下面我们先给出一个简单朴素的协议实现，方便理解。

公共输入

- MLE 多项式 $\tilde{f}$ 的承诺 $\mathsf{cm}([[\tilde{f}]]_n)$
- 求值点 $\mathbf{u} = (u_0, u_1, \ldots, u_{n-1})$
- 求值结果 $v = \tilde{f}(\mathbf{u})$

Witness

- MLE 多项式 $\tilde{f}$ 在 n 维 HyperCube 上的点值向量 $\mathbf{a} = (a_0, a_1, \ldots, a_{2^n-1})$

交互过程

第一轮：Prover 发送余数多项式的承诺

- 计算 n 个余数 MLE 多项式， $\{\tilde{q}_k\}_{k=0}^{n-1}$
- 构造余数 MLE 多项式所映射到的 Univariate 多项式 $Q_k = [[\tilde{q}_k]]_k, \quad 0 \leq k < n$
- 计算并发送它们的承诺：$\mathsf{cm}(Q_0), \mathsf{cm}(Q_1), \ldots, \mathsf{cm}(Q_{n-1})$

$$\tilde{f}(X_0, X_1, \ldots, X_{n-1}) - v = \sum_{k=0}^{n-1} (X_k - u_k) \cdot \tilde{q}_k(X_0, X_1, \ldots, X_{k-1}) \tag{43}$$

第二轮：Prover 计算，$\pi_k = \mathsf{cm}(X^{D_{max} - 2^k} \cdot Q_k), \quad 0 \leq k < n$，作为 $\deg(Q_k) < 2^k$ 的 Degree Bound 证明，一并发送给 Verifier

第三轮：Verifier 发送随机数 $\zeta \in \mathbb{F}_p^*$

第四轮：Prover 计算辅助多项式 $R(X)$ 与商多项式 $H(X)$，并发送 $\mathsf{cm}(H)$

- 计算 $R(X)$，

$$R(X) = F(X) - v \cdot \Phi_n(\zeta) - \sum_{k=0}^{n-1} \left(\zeta^{2^k} \cdot \Phi_{n-k-1}(\zeta^{2^{k+1}}) - u_k \cdot \Phi_{n-k}(\zeta^{2^k}) \right) \cdot Q_k(X) \tag{44}$$

- 计算 $H(X)$ 及其承诺 $\mathsf{cm}(H)$，作为 $R(X)$ 在 $X = \zeta$ 点取值为零的证明

$$H(X) = \frac{R(X)}{X - \zeta} \tag{45}$$

第五轮：Verifier 验证下面的等式

- 构造 $\mathsf{cm}(R)$ 的承诺：

$$\mathsf{cm}(R) = \mathsf{cm}(F) - \mathsf{cm}(v \cdot \Phi_n(\zeta)) - \sum_{i=0}^{n-1} \left(\zeta^{2^i} \cdot \Phi_{n-i-1}(\zeta^{2^{i+1}}) - u_i \cdot \Phi_{n-i}(\zeta^{2^i}) \right) \cdot \mathsf{cm}(Q_i) \tag{46}$$

- 验证 $R(\zeta) = 0$

$$e(\mathsf{cm}(R),\ [1]_2) = e(\mathsf{cm}(H), [\tau]_2 - \zeta \cdot [1]_2) \tag{47}$$

- 验证 $(\pi_0, \pi_1, \ldots, \pi_{n-1})$ 是否正确，即验证所有的余数多项式的 Degree Bound： $\deg(Q_i) < 2^i$，对于 $0 \leq i < n$

$$e(\mathsf{cm}(Q_i), [\tau^{D_{max} - 2^i}]_2) = e(\pi_i, [1]_2), \quad 0 \leq i < n \tag{48}$$

效率概述

- 证明尺寸： $(2n+1)\mathbb{G}_1$
- Verifier 计算量： $(2n+2)P$，$(n+2)\mathsf{EccMul}^{\mathbb{G}_1}$

优化协议

朴素协议中有 n 个商多项式，它们的 Degree Bound 有 $2n$ 个 $\mathbb{G}_1$，这显然不够高效。不过，我们可以批量地证明这 n 个 degree bound。下面是传统的批量证明的思路：

- Verifer 先发送一个随机数 β
- Prover 把 n 个商多项式聚合在一起，得到 $P(X)$，聚合的时候把这些商多项式的 Degree 补齐到同一个值，即最大的那个商多项式 Degree 2^{n-1}：

$$P(X)=\sum_{k=0}^{n-1}\beta^k\cdot X^{2^n-2^k}\cdot Q_i(X) \tag{49}$$

- Prover 发送 $P(X)$ 的承诺 $\mathsf{cm}(P)$
- Verifier 发送随机数 ζ
- Prover 构造多项式 $S(X)$，它在 $X=\zeta$ 处取值为零，即 $S(\zeta)=0$

$$S(X)=P(X)-\sum_{k=0}^{n-1}\beta^k\cdot\zeta^{2^n-2^k}\cdot Q_i(X) \tag{50}$$

- Prover 构造商多项式 $H_1(X)$ 并将其 Degree 对齐到最大的 Degree Bound D，然后证明 $S(\zeta)=0$ ，并发送承诺 $\mathsf{cm}(H_1)$

$$H_1(X)=\frac{S(X)}{X-\zeta}\cdot X^{D-2^n+1} \tag{51}$$

- Verifier 手里有 $\mathsf{cm}(P)$ 与 $\mathsf{cm}(Q_i)$，他可以根据下面的等式，还原出 $\mathsf{cm}(S)$ 的承诺：

$$\mathsf{cm}(S)=\mathsf{cm}(P)-\sum_{i=0}^{n-1}\beta^i\cdot\zeta^{2^n-2^k}\cdot Q_i(X) \tag{52}$$

- Verifier 只需要两个 Pairing 运算即可验证 $S(\zeta)=0$，从而得到 n 个 Degree Bound 证明成立

$$e\big(\mathsf{cm}(S),\ [\tau^{D_{max}-2^n+1}]_2\big)=e\big(\mathsf{cm}(H),[\tau]_2-\zeta\cdot[1]_2\big) \tag{53}$$

此外，Verfier 还可以发一个随机数 α，进一步聚合 $R(X)$ 与 $S(X)$ 的取值证明，因为它们两个在 $X=\zeta$ 处的取值都为零。

下面是优化版本的 Zeromorph 协议，参见 Zeromorph 论文 [KT23] Section 6。优化的技术主要是将多个 Degree Bound 证明聚合在一起，同时将 $R(X)$ 的求值证明也聚合在一起。这样可以仅使用两个 Pairing 运算来验证验证（这个版本暂时不考虑 Zero-knowledge 的性质）。

公共输入

- MLE 多项式 $\tilde{f}$ 映射到 Univariate 多项式 $F(X)=[[\tilde{f}]]_n$ 的承诺 $\mathsf{cm}([[\tilde{f}]]_n)$
- 求值点 $\mathbf{u}=(u_0,u_1,\ldots,u_{n-1})$
- 求值结果 $v=\tilde{f}(\mathbf{u})$

Witness

- MLE 多项式 $\tilde{f}$ 的求值向量 $\mathbf{a}=(a_0,a_1,\ldots,a_{2^n-1})$

协议

第一轮：Prover 发送余数多项式的承诺

- 计算 n 个余数 MLE 多项式， $\{q_i\}_{i=0}^{n-1}$
- 构造余数 MLE 多项式所映射到的 Univariate 多项式 $Q_i=[[q_i]]_i,\quad 0\le i<n$
- 计算并发送它们的承诺：$\mathsf{cm}(Q_0),\mathsf{cm}(Q_1),\ldots,\mathsf{cm}(Q_{n-1})$

$$\tilde{f}(X_0,X_1,\ldots,X_{n-1})-v=\sum_{i=0}^{n-1}(X_k-u_k)\cdot q_i(X_0,X_1,\ldots,X_{k-1}) \tag{54}$$

第二轮：Verifier 发送随机数 $\beta\in\mathbb{F}_p^*$ 用来聚合多个 Degree Bound 证明

第三轮：Prover 计算 $P(X)$ 并发送其承诺 $\mathsf{cm}(P)$

- 计算 $P(X)$,

$$P(X)=\sum_{i=0}^{n-1}\beta^i\cdot X^{2^n-2^k}Q_i(X) \tag{55}$$

第四轮：Verifier 发送随机数 $\zeta\in\mathbb{F}_p^*$，用来挑战多项式在 $X=\zeta$ 处的取值

第五轮：Prover 计算 $H_0(X)$ 与 $H_1(X)$

- 计算 $R(X)$,

$$R(X) = F(X) - v\cdot\Phi_n(\zeta) - \sum_{i=0}^{n-1}\left(\zeta^{2^i}\cdot\Phi_{n-i-1}(\zeta^{2^{i+1}}) - u_i\cdot\Phi_{n-i}(\zeta^{2^i})\right)\cdot Q_i(X) \tag{56}$$

- 计算 $S(X)$,

$$S(X) = P(X) - \sum_{k=0}^{n-1}\beta^k\cdot\zeta^{2^n-2^k}\cdot Q_i(X) \tag{57}$$

- 计算商多项式 $H_0(X)$ 与 $H_1(X)$

$$H_0(X) = \frac{R(X)}{X-\zeta}, \qquad H_1(X) = \frac{S(X)}{X-\zeta} \tag{58}$$

第六轮：Verifier 发送随机数 $\alpha\in\mathbb{F}_p^*$ ，用来聚合 $H_0(X)$ 与 $H_1(X)$

第七轮：Prover 计算 $H(X)$ 并发送其承诺 $\mathsf{cm}(H)$

- 计算 $H(X) = (H_0(X) + \alpha\cdot H_1(X))\cdot X^{D_{max}-2^n+1}$

第八轮：Verifier 验证下面的等式

- 还原 $\mathsf{cm}(R)$ 的承诺：

$$\mathsf{cm}(R) = \mathsf{cm}(F) - \mathsf{cm}(v\cdot\Phi_n(\zeta)) - \sum_{i=0}^{n-1}\left(\zeta^{2^i}\cdot\Phi_{n-i-1}(\zeta^{2^{i+1}}) - u_i\cdot\Phi_{n-i}(\zeta^{2^i})\right)\cdot\mathsf{cm}(Q_i) \tag{59}$$

- 还原 $\mathsf{cm}(S)$ 的承诺：

$$\mathsf{cm}(S) = \mathsf{cm}(P) - \sum_{i=0}^{n-1}\beta^i\cdot\zeta^{2^n-2^k}\cdot Q_i(X)) \tag{60}$$

- 验证 $R(\zeta)=0$ 与 $S(\zeta)=0$

$$e(\mathsf{cm}(R) + \alpha\cdot\mathsf{cm}(S),\ [\tau^{D-2^n+1}]_2) = e(\mathsf{cm}(H), [\tau]_2 - \zeta\cdot[1]_2) \tag{61}$$

总结

Zeromorph 总体上来说是一个简洁的协议，它将 MLE 的点值式直接映射到 Univariate 多项式的系数，然后利用 KZG10 协议来完成 Evaluation 的证明。后续文章将讨论如何将 Zeromorph 结合 FRI 协议来实现 MLE PCS。

Reference:

- [KT23] Kohrita, Tohru, and Patrick Towa. "Zeromorph: Zero-knowledge multilinear-evaluation proofs from homomorphic univariate commitments." Cryptology ePrint Archive (2023). https://eprint.iacr.org/2023/917
- [PST13] Papamanthou, Charalampos, Elaine Shi, and Roberto Tamassia. "Signatures of correct computation." Theory of Cryptography Conference. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. https://eprint.iacr.org/2011/587