

理解 PLONK (三) : 置换证明

Plonkish 电路编码用两个矩阵 (Q, σ) 描述电路的空白结构, 其中 Q 为运算开关, σ 为置换关系, 用来约束 W 矩阵中的某些位置必须被填入相等的值。本文重点讲解置换证明 (Permutation Argument) 的原理。

回顾拷贝关系

回顾一下 Plonkish 的 W 表格, 总共有三列, 行数按照 2^2 对齐。

i	$w_{a,i}$	$w_{b,i}$	$w_{c,i}$
1	x_6	x_5	out
2	x_1	x_2	x_6
3	x_3	x_4	x_5
4	0	0	out

我们想约束 Prover 在填写 W 表时, 满足下面的拷贝关系: $w_{a,1} = w_{c,2}$ $w_{b,1} = w_{c,3}$ 与 $w_{c,1} = w_{c,4}$, 换句话说, $w_{a,1}$ 位置上的值需要被拷贝到 $w_{c,2}$ 处, 而 $w_{b,1}$ 位置上的值需要被拷贝到 $w_{c,3}$ 处, $w_{c,1}$ 位置上的值被拷贝到 $w_{c,4}$ 处。

问题的挑战性在于, Verifier 要仅通过一次随机挑战就能完成 W 表格中多个拷贝关系的证明, 并且在看不到 W 表格的情况下。

Plonk 的「拷贝约束」是通过「置换证明」 (Permutation Argument) 来实现, 即把表格中需要约束相等的那些值进行循环换位, 然后证明换位后的表格和原来的表格完全相等。

简化一下问题: 如何证明两个等长向量 \vec{a} 和 \vec{a}' 满足一个已知的置换 σ , 并且 $\vec{a} = \vec{a}'$

$$a_i = a'_{\sigma(i)}$$

举一个例子, 假设 $\vec{a} = (a_0, a_1, a_2, a_3)$, $\vec{a}' = (a_1, a_2, a_3, a_0)$, 即他们满足一个「左移循环换位」的置换关系, 那么 $\sigma = \{0 \rightarrow 1; 1 \rightarrow 2; 2 \rightarrow 3; 3 \rightarrow 0\}$ 。如何能证明 $\vec{a} = \vec{a}'$, 那么两个向量对应位置的值都应该相等,

\vec{a}	a_0	a_1	a_2	a_3
\vec{a}'	a_1	a_2	a_3	a_0

那么 $a_0 = a_1$, $a_1 = a_2$, $a_2 = a_3$, $a_3 = a_0$, 于是可以得出结论: $a_0 = a_1 = a_2 = a_3$, 即 \vec{a} 中的全部元素都相等。

对于 W , 我们只需要针对那些需要相等的位置进行循环换位, 然后让 Prover 证明 W 和经过循环换位后的 W' 表格相等, 那么可实现拷贝约束。证明两个表格相等, 这个可以通过多项式编码, 然后进行概率检验的方式完成。剩下的工作就是如何让 Prover 证明 W' 确实是 (诚实地) 按照事先约定的方式进行循环移位。

那么接下来就是理解如何让 Prover 证明两个向量之间满足某一个「置换关系」。置换证明 (Permutation Argument) 是 Plonk 协议中的核心部分, 为了解释它的工作原理, 我们先从一个基础协议开始——连乘证明 (Grand Product Argument)。

冷启动: Grand Product

假设我们要证明下面的「连乘关系」:

$$p = q_0 \cdot q_1 \cdot q_2 \cdots q_{n-2}$$

我们在上一篇文章介绍了如何证明一组「单乘法」, 通过多项式编码, 把多个单乘法压缩成单次乘法的验证。

这里对付连乘的基本思路是: 让 Prover 利用一组单乘的证明来实现多个数的连乘证明, 然后再通过多项式的编码, 交给 Verifier 进行概率检查。

强调下: 思路中的关键点是如何把一个连乘计算转换成多次的单乘计算。

我们需要通过引入一个「辅助向量」, 把「连乘」的计算看成是一步步的单乘计算, 然后辅助向量表示每次单乘之后的「中间值」:

q_i	r_i	$q_i \cdot r_i$
q_0	$r_0 = 1$	$r_1 = q_0$
q_1	r_1	$r_2 = q_0 \cdot q_1$
q_2	r_2	$r_3 = q_0 \cdot q_1 \cdot q_2$
\vdots	\vdots	\vdots
q_{n-2}	r_{n-2}	$r_{n-1} = p$

上面表格表述了连乘过程的计算轨迹 (Trace), 每一行代表一次单乘, 顺序从上往下计算, 最后一行计算出最终的结果。

表格的最左列为要进行连乘的向量 $\{q_i\}$ ，中间列 $\{r_i\}$ 为引入的辅助变量，记录每次「单乘之前」的中间值，最右列表示每次「单乘之后」的中间值。

不难发现，「中间列」向量 \vec{r} 向上挪一行与「最右列」几乎一致，除了最后一个元素。该向量的第一个元素用了常数 1 作为计算初始值，「最右列」最后一个向量元素为计算结果。

向量 \vec{r} 是一个 Accumulator，即记录连乘计算过程中的每一个中间结果：

$$r_k = \prod_{i=0}^{k-1} q_i$$

那么显然我们可以得到下面的递归式：

$$r_0 = 1, \quad r_{k+1} = q_k \cdot r_k$$

于是，表格的三列编码后的多项式也将满足下面三个约束。第一个是初始值为 1：

$$L_0(X) \cdot (r(X) - 1) = 0, \quad \forall X \in H$$

第二个约束为递归的乘法关系：

$$q(X) \cdot r(X) = r(\omega \cdot X), \quad \forall X \in H \setminus \{\omega^{-1}\}$$

第三个约束最后结果 $r_{n-1} = p$ ：

$$L_{n-1}(X) \cdot (r(X) - p) = 0, \quad \forall X \in H$$

我们可以用一个小技巧来简化上面的三个约束。我们把计算连乘的表格添加一行，令 $q_{n-1} = 1/p$ （注意： p 为 \vec{q} 向量的连乘积）

q_i	r_i	$q_i \cdot r_i$
q_0	1	r_0
q_1	r_0	r_1
q_2	r_1	r_2
\vdots	\vdots	\vdots
q_{n-2}	r_{n-2}	r_{n-1}
$q_{n-1} = \frac{1}{p}$	r_{n-1}	1

这样一来， $r_n = r_0 = 1$ 。最右列恰好是 \vec{r} 的循环移位。并且上面表格的每一行都满足「乘法关系」！于是，我们可以用下面的多项式约束来表示递归的连乘：

$$q(X) \cdot r(X) = r(\omega \cdot X), \quad \forall X \in H$$

接下来，Verifier 可以挑战下面的多项式等式：

$$L_0(X) \cdot (r(X) - 1) + \alpha \cdot (q(X) \cdot r(X) - r(\omega \cdot X)) = h(X) \cdot z_H(X)$$

其中 α 是用来聚合多个多项式约束的随机挑战数。其中 $h(X)$ 为商多项式， $z_H(X) = (X - 1)(X - \omega) \cdots (X - \omega^{n-1})$ 。

接下来，通过 Schwartz-Zippel 定理，Verifier 可以给出挑战数 ζ 来验证上述多项式等式是否成立。

到此为止，如果我们已经理解了如何证明一个向量元素的连乘，那么接下来的问题是如何利用「连乘证明」来实现「Multiset 等价证明」（Multiset Equality Argument）。

从 Grand Product 到 Multiset 等价

假设有两个向量，其中一个向量是另一个向量的乱序重排，那么如何证明它们在集合意义（注意：集合无序）上的等价呢？最直接的做法是依次枚举其中一个向量中的每个元素，并证明该元素属于另一个向量。但这个方法有个限制，就是无法处理向量中会出现两个相同元素的情况，也即不支持「多重集合」（Multiset）的判等。例如 $\{1, 1, 2\}$ 就属于一个多重集合（Multiset），那么它显然不等于 $\{1, 2, 2\}$ ，也不等于 $\{2, 1\}$ 。

另一个直接的想法是将两个向量中的所有元素都连乘起来，然后判断两个向量的连乘值是否相等。但这个方法同样有一个严重的限制，就是向量元素必须都为素数，比如 $3 \cdot 6 = 9 \cdot 2$ ，但 $\{3, 6\} \neq \{9, 2\}$ 。

修改下这个方法，我们假设向量 $\{q_i\}$ 为一个多项式 $q(X)$ 的根集合，即对向量中的任何一个元素 q_i ，都满足 $q(r_i) = 0$ 。这个多项式可以定义为：

$$q(X) = (X - q_0)(X - q_1)(X - q_2) \cdots (X - q_{n-1})$$

如果存在另一个多项式 $p(X)$ 等于 $q(X)$ ，那么它们一定具有相同的根集合 $\{q_i\}$ 。比如

$$\prod_i (X - q_i) = q(X) = p(X) = \prod_i (X - p_i)$$

那么

$$\{q_i\} =_{\text{multiset}} \{p_i\}$$

我们可以利用 Schwartz-Zippel 定理来进一步地检验：向 Verifier 索要一个随机数 γ ，那么 Prover 就可以通过下面的等式证明两个向量 $\{p_i\}$ 与 $\{q_i\}$ 在多重集合意义上等价：

$$\prod_{i \in [n]} (\gamma - p_i) = \prod_{i \in [n]} (\gamma - q_i)$$

还没结束，我们需要用上一节的连乘证明方案来继续完成验证，即通过构造辅助向量（作为一个累积器），把连乘转换成多个单乘来完成证明。需要注意的是，这里的两个连乘可以合并为一个连乘，即上面的连乘相等可以转换为

$$\prod_{i \in [n]} \frac{(\gamma - p_i)}{(\gamma - q_i)} = 1$$

到这里，我们已经明白如何证明「Multiset 等价」，下一步我们将完成构造「置换证明」（Permutation Argument），用来实现协议所需的「Copy Constraints」。

从 Multiset 等价到置换证明

Multiset 等价可以被看作是一类特殊的置换证明。即两个向量

p_i 和

q_i 存在一个「未知」的置换关系。

而我们需要的是一个支持「已知」的特定置换关系的证明和验证。也就是对一个有序的向量进行一个「公开特定的重新排列」。

先简化下问题，假如我们想让 Prover 证明两个向量满足一个奇偶位互换的置换：

$$\begin{aligned} \vec{a} &= (a_0, a_1, a_2, a_3, \dots, a_{n-1}, a_n) \\ \vec{b} &= (a_1, a_0, a_3, a_2, \dots, a_n, a_{n-1}) \end{aligned}$$

我们仍然采用「多项式编码」的方式把上面两个向量编码为两个多项式， $a(X)$ 与 $b(X)$ 。思考一下，我们可以用下面的「位置向量」来表示「奇偶互换」：

$$\vec{i} = (1, 2, 3, 4, \dots, n-1, n), \quad \sigma = (2, 1, 4, 3, \dots, n, n-1)$$

我们进一步把这个位置向量和 \vec{a} 与 \vec{b} 并排放在一起：

a_i	i	b_i	$\sigma(i)$
a_0	0	$b_0 = a_1$	1
a_1	1	$b_1 = a_0$	0
a_2	2	$b_2 = a_3$	3
a_3	3	$b_3 = a_2$	2
\vdots	\vdots	\vdots	\vdots
a_n	n	$b_n = a_{n-1}$	$n-1$
a_{n-1}	$n-1$	$b_{n-1} = a_n$	n

接下来，我们要把上表的左边两列，还有右边两列分别「折叠」在一起。换句话说，我们把 (a_i, i) 视为一个元素，把 $(b_i, \sigma(i))$ 视为一个元素，这样上面表格就变成了：

$a'_i = (a_i, i)$	$b'_i = (b_i, \sigma(i))$
$(a_0, 0)$	$(b_0 = a_1, 1)$
$(a_1, 1)$	$(b_1 = a_0, 0)$
\vdots	\vdots
$(a_{n-1}, n-1)$	$(b_{n-1} = a_n, n)$
(a_n, n)	$(b_n = a_{n-1}, n-1)$

容易看出，如果两个向量 \vec{a} 与 \vec{b} 满足 σ 置换，那么，合并后的两个向量 \vec{a}' 和 \vec{b}' 将满足 Multiset 等价关系。

也就是说，通过把向量和位置值合并，就能够把一个「置换证明」转换成一个「多重集合等价证明」，即不用再针对某个特定的「置换关系」进行证明。

这里又出现一个问题，表格的左右两列中的元素为二元组 (Pair)，二元组无法作为一个「一元多项式」的根集合。

我们再使用一个技巧：再向 Verifier 索取一个随机数 β ，把一个元组「折叠」成一个值：

$a'_i = (a_i + \beta \cdot i)$	$b'_i = (b_i + \beta \cdot \sigma(i))$
$(a_0 + \beta \cdot 0)$	$(b_0 + \beta \cdot 1)$
$(a_1 + \beta \cdot 1)$	$(b_1 + \beta \cdot 0)$
\vdots	\vdots
$(a_{n-1} + \beta \cdot n - 1)$	$(b_{n-1} + \beta \cdot n)$

$a'_i = (a_i + \beta \cdot i)$	$b'_i = (b + \beta \cdot \sigma(i))$
$(a_n + \beta \cdot n)$	$(b_n + \beta \cdot (n - 1))$

接下来，Prover 可以对 \vec{a}' 与 \vec{b}' 两个向量进行 Multiset 等价证明，从而可以证明它们的置换关系。

完整的置换协议

公共输入：置换关系 σ

秘密输入：两个向量 \vec{a} 与 \vec{b}

预处理：Prover 和 Verifier 构造 $[id(X)]$ 与 $[\sigma(X)]$

第一步：Prover 构造并发送 $[a(X)]$ 与 $[b(X)]$

第二步：Verifier 发送挑战数 β 与 γ

第三步：Prover 构造辅助向量 \vec{z} ，构造多项式 $z(X)$ 并发送 $[z(X)]$

$$z_0 = 1$$

$$z_{i+1} = z_i \cdot \frac{a_i + \beta \cdot i + \gamma}{b_i + \beta \cdot \sigma(i) + \gamma}$$

第四步：Verifier 发送挑战数 α

第五步：Prover 构造 $f(X)$ 与 $q(X)$ ，并发送 $[q(X)]$

$$f(X) = L_0(X)(z(X) - 1) + \alpha \cdot (z(\omega \cdot X)(b(X) + \beta \cdot \sigma(X) + \gamma) - z(X)(a(X) + \beta \cdot X))$$

$$q(X) = \frac{f(X)}{z_H(X)}$$

第六步：Verifier 向 $[a(X)], [b(X)], [q(X)]$ 查询这三个多项式在 $X = \zeta$ 处的取值，得到 $a(\zeta), b(\zeta), q(\zeta)$ ；向 $[z(X)]$ 查询 $X = \zeta, X = \omega \cdot \zeta$ 两个位置处的取值，即 $z(\zeta), z(\omega \cdot \zeta)$ ；向 $[\sigma(X)]$ 与 $[id(X)]$ 这两个多项式发送求值查询 $X = \zeta$ ，得到 $id(\zeta)$ 与 $\sigma(\zeta)$ ；Verifier 自行计算 $z_H(\zeta), L_0(\zeta)$

验证步：Verifier 验证

$$L_0(\zeta)(z(\zeta) - 1) + \alpha \cdot (z(\omega \cdot \zeta)(b(\zeta) + \beta \cdot \sigma(\zeta) + \gamma) - z(\zeta)(a(\zeta) + \beta \cdot id(\zeta))$$

协议完毕。

References:

- [WIP] Copy constraint for arbitrary number of wires.
https://hackmd.io/CfFCbA0TTJ6X08vHg0-9_g
- Alin Tomescu. Feist-Khovratovich technique for computing KZG proofs fast.
<https://alinush.github.io/2021/06/17/Feist-Khovratovich-technique-for-computing-KZG-proofs-fast.html#fn:FK20>
- Ariel Gabizon. Multiset checks in PLONK and Plookup.
<https://hackmd.io/@arielg/ByFgSDA7D>