Plonk arithmetisation

Marc Beunardeau – Nomadic Labs – Tezos

April 22, 2023

Table of Contents

Introduction

Plonk

- Algeraic Intermediate Representation
- Selectors
- Permutation argument
- Plonk

Constraint systems particularities

Anemoi Example

Conclusion

Plonk and SNARKs

Plonk is a Succinct Non-interactive Argument of Knowledge

- $\mathcal{R}(x, w)$ is an $\mathcal{NP}$ relation described in a certain language
- a prover can convince a verifier that he knows w such that $\mathcal{R}(x, w)$
- the verifier runs in time independent of $|w|$ and $|\mathcal{R}(\cdot, \cdot)|$

Examples:

- $\mathcal{R}$ encodes the Sudoku rules, x the starting positions and w the solution
- $\mathcal{R}(x|y, w)$ encodes a function f with $f(x) = y$ and w being intermediates variables

SNARKs breakdown

Asymmetric cryptography works on some algebraic structure

1. We want $\mathcal{R}$ in a 'normal' language
2. Reduce the satisfactions of $\mathcal{R}$ to some algebraic equations
3. Do some crypto to get succinctness

This talk : 1 to 2 in Plonk's case

Table of Contents

Introduction

Plonk

- Algeraic Intermediate Representation
- Selectors
- Permutation argument
- Plonk

Constraint systems particularities

Anemoi Example

Conclusion

What does the crypto do ?

- $\mathbb{F}$
- $u, v \in \mathbb{F}^n$
- $P(U, V) \in \mathbb{F}_6[U, V]$
- show succinctly $\forall i,\ P(u[i], v[i]) = 0$

Intuition: $u[i]$ is a register at time i of a program

Let's add a few more

- We can fix a succinct number values (eg. for initialisation) : $u[0] = 1$
- The verifier can choose some values (to choose x)
- We can link i and $i + 1$: $P(u[i], v[i], u[i+1], v[i+1]) = 0$

Cost: The cost will depend on the degree and complexity (nb of multiplication) of the polynomials, the number and length (not for the verifier) of vectors

Let's do Fibonacci

We need two vectors u, v

- $u[0], v[0]$ are chosen by the verifier
- $P_1(U, V, U', V') = U + V - U'$
- $P_2(U, V, U', V') = U' + V - V'$
- $v[n]$ is chosen by the verifier

Prove the identities and send $v[n]$
$\Rightarrow$ I delegated the computation of Fibonacci$(2n+1)$

Multiple operations

What if I want to compute $g(x)$ and $f(y)$ in the same relation?

Pre-processed relations

- The verifier runs in constant time with regard to $|\mathcal{R}(\cdot, \cdot)|$
- However he needs to read it once
- We will create pre-processed vectors q which are agreed upon during setup

We can set a linear number of values in these vectors !

Selectors

- $(I, \bar{I})$ partition of $\{0 \cdots n\}$
- I want to apply $P_I(\vec{X})$ to I and $P_{\bar{I}}(\vec{X})$ to $\bar{I}$
- $Q_I[i] = 1$ if $i \in I$, $Q_I[i] = 0$ otherwise (same for $Q_{\bar{I}}$)
- $P(\vec{X}, Q_I, Q_{\bar{I}}) = Q_I * P_I(\vec{X}) + Q_{\bar{I}} * P_{\bar{I}}(\vec{X})$

<u>Note:</u> Selectors are less expensive thanks to pre-processing

Limitation

I want some long term memory
We will show $u[i] = v[j]$ for some pre-determined i, j

Copy constraint

Assume we can show $u = \sigma(v)$ for $\sigma \in \mathfrak{S}_n$
Show $v[3] = v[7] = v[20]$

- Create σ such that $\sigma(3) = 7$ and $\sigma(7) = 20$ (and $\sigma(20) = 3$)
- $v = \sigma(v) \Rightarrow v[3] = v[7] = v[20]$
- Generalize to show $v[i] = v[j]$ for all i, j in a set
- Apply the technique to $u|v$ to copy from one vector to another

Showing product

$$\prod_i u[i] = \prod_i v[i] \iff \prod_i u[i]/v[i] = 1$$

Ask the prover for a new vector z

- $z[i+1] = z[i] * \frac{u[i+1]}{v[i+1]}$
- $z[0] = u[0]/v[0]$
- $z[n] = 1$

Permutation 1

- $u = \sigma(v) \Rightarrow \prod_i u[i] = \prod_i v[i]$
- Maybe $u[i] = 2 * v[\sigma(i)]$ and $u[j] = v[\sigma(j)]/2$

We will need something more

Randomisation

I can add a randomised term in my polynomials : for all i, $P(u[i], v[i], \alpha) = 0$ for a random α chosen after u and v

$\iff$ for a non negligible number of different α, $\forall i$ $P(u[i], v[i], \alpha) = 0$

Permutation 2

$$u = \sigma(v) \Rightarrow \prod_i (u[i] + \alpha) = \prod_i (v[i] + \alpha)$$

$$\exists \sigma \text{ s.t. } u = \sigma(v) \iff \prod_i (u[i] + \alpha) = \prod_i (v[i] + \alpha)$$

Maybe $u = \sigma'(v)$

$\Rightarrow$ let's add some dependency to σ

Permutation 3

Create the vector s_σ defined by $s_\sigma[i] = \sigma(i)$ and s_{id} for the identity permutation

$$\prod_i (u[i] + \beta s_{id}[i] + \alpha) = \prod_i (v[i] + \beta s_{sigma}[i] + \alpha)$$

$$\prod_i (u[i] + \beta * i + \alpha) = \prod_i (u[\sigma(i)] + \beta * \sigma(i) + \alpha)$$

Example: $\sigma = (1,3,2)$, $u = (2,5,7)$ $v = (2,7,5)$

$$\prod_u = (2 + \beta * 1 + \alpha) * (5 + \beta * 2 + \alpha) * (7 + \beta * 3 + \alpha)$$

$$\prod_v = (2 + \beta * 1 + \alpha) * (7 + \beta * 3 + \alpha) * (5 + \beta * 2 + \alpha)$$

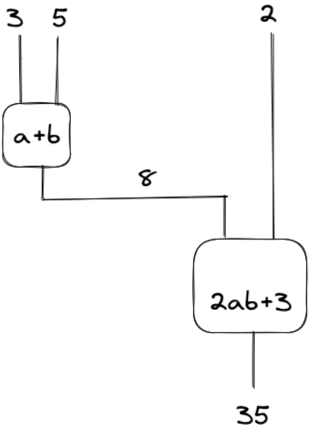
Vanilla Plonk

ql	qr	qo	qm	qc	a	b	c
1	0	0	0	0	3		
1	0	0	0	0	14		
2	1	-1	0	0			
0	0	-1	1	0			
-1	0	0	1	0			
⋮					⋮		

$$q_l * a + q_r * b + q_m * a * b + q_o * c + q_{cst} = 0$$

$$a|b|c = \sigma(a|b|c)$$

Vanilla Plonk example



ql	qr	qo	qm	qc	a	b	c
1	1	-1	0	0	3	5	8
0	0	-1	2	3	8	2	35

$$q_l * a + q_r * b + q_m * a * b + q_o * c + q_{cst} = 0$$

Why Plonk ?

Table of Contents

Introduction

Plonk

- Algeraic Intermediate Representation
- Selectors
- Permutation argument
- Plonk

Constraint systems particularities

Anemoi Example

Conclusion

Field

256-bits prime field
work with it when possible !

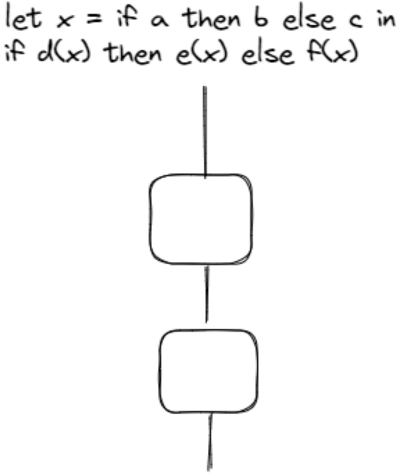
If then else

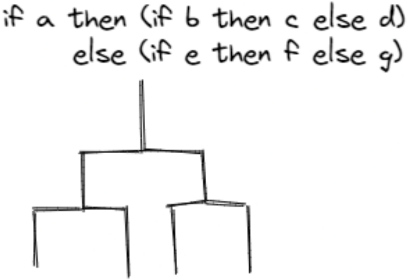
If a then b else $c \iff a * b + (1 - a) * c \wedge a * (1 - a) = 0$

Notes:

- both branch are paid
- booleans are wasteful
- don't forget the boolean constraint !

If then else explosion





Non determinism

$$a \neq 0 \iff \exists b \text{ st. } a * b = 1$$

Note: b is not used anywhere else
$\Rightarrow$ we can exclude it from the permutation argument

f^{-1} vs f

$$y = f(x) \iff f^{-1}(y) = x$$

High degree trick: $a = b^{1/5} \iff a^5 = b$

Table of Contents

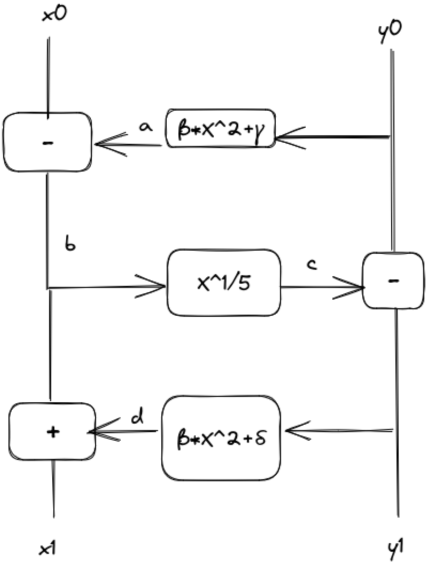
Introduction

Plonk
- Algeraic Intermediate Representation
- Selectors
- Permutation argument
- Plonk

Constraint systems particularities

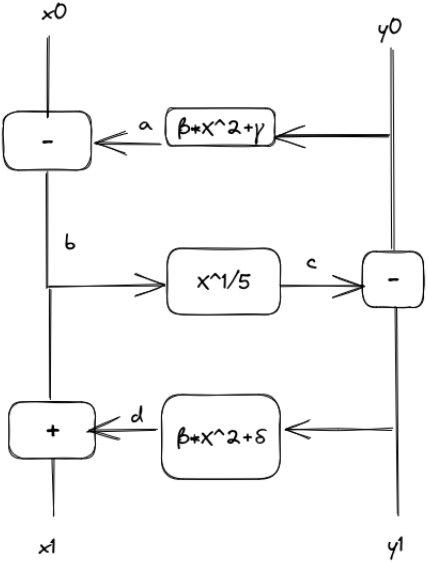
Anemoi Example

Conclusion



$$a = \beta * y_0^2 + \gamma$$
$$b = x_0 - a$$
$$c' = c * c$$
$$c'' = c' * c'$$
$$c * c'' = b$$
$$y_1 = y_0 - c$$
$$d = \beta * y_1^2 + \delta$$
$$x_1 = b + d$$

Let's use custom constraints



$$(y_0 - y_1)^5 = x_0 - \beta * y_0^2 - \gamma$$
$$x_1 = (y_0 - y_1)^5 + \beta * y_1^2 + \delta$$

Two rounds

$$
\begin{aligned}
(y_0 - y_1)^5 &= x_0 - \beta * y_0^2 - \gamma \\
x_1 &= (y_0 - y_1)^5 + \beta * y_1^2 + \delta \\
(y_1 - y_2)^5 &= x1 - \beta * y_1^2 + \gamma \\
x_2 &= (y_1 - y_2)^5 + \beta * y_2^2 + \delta
\end{aligned}
$$

$$x1 \mapsto (y_0 - y_1)^5 + \beta * y_1^2 + \delta$$

- Inline linear terms
- Maybe quadratic or cubic
- Don't inline higher degrees

When to use custom constraints

custom constraints are paid for everywhere $\Rightarrow$ use them depending on the application

Table of Contents

Introduction

Plonk

Algeraic Intermediate Representation

Selectors

Permutation argument

Plonk

Constraint systems particularities

Anemoi Example

Conclusion

Design space

The design space is huge !

- Field
- custom constraints
- number of wires
- number of wires in the permutation
- access to $i+1$, $i+2$ etc...
- maximum degree of identities
- lookup

Parametric (not only in the field) primitives are helpful !
Comparisons are hard !

Poseidon example

- Does partial rounds and full rounds
- Can minimize the number of rounds or the number of full rounds
- Initially for R1CS
- Change the parametrisation for Plonk

open question

Is this parametrisation detrimental to security ?

Sources

- Plonk paper: https://eprint.iacr.org/2019/953
- Plonk blogpost: https://hackmd.io/@aztec-network/plonk-arithmetiization-air
- Anemoi paper: https://eprint.iacr.org/2022/840
- Poseidon paper: https://eprint.iacr.org/2019/458

Thank you !