Notes on HyperNova

arnaucube

May 2023

Abstract

Notes taken while reading about HyperNova [1] and CCS[2].

Usually while reading papers I take handwritten notes, this document contains some of them re-written to $LaTeX$.

The notes are not complete, don't include all the steps neither all the proofs.

Thanks to George Kadianakis for clarifications, and the authors Srinath Setty and Abhiram Kothapalli for answers on chats and twitter.

Contents

1 CCS

1.1 R1CS to CCS overview

R1CS instance $S_{R1CS} = (m, n, N, l, A, B, C)$
where m, n are such that $A \in \mathbb{F}^{m \times n}$, and l such that the public inputs $x \in \mathbb{F}^l$. Also $z = (w, 1, x) \in \mathbb{F}^n$, thus $w \in \mathbb{F}^{n-l-1}$.

CCS instance $S_{CCS} = (m, n, N, l, t, q, d, M, S, c)$
where we have the same parameters than in S_{R1CS}, but additionally:
$t = |M|$, $q = |c| = |S|$, $d=$ max degree in each variable.

R1CS-to-CCS parameters $n = n,\ m = m,\ N = N,\ l = l,\ t = 3,\ q = 2,\ d = 2$, $M = \{A, B, C\}$, $S = \{\{0,\ 1\},\ \{2\}\}$, $c = \{1, -1\}$

The CCS relation check:

$$\sum_{i=0}^{q-1} c_i \cdot \bigcirc_{j \in S_i} M_j \cdot z == 0$$

where $z = (w, 1, x) \in \mathbb{F}^n$.

In our R1CS-to-CCS parameters is equivalent to

$$\begin{aligned} & c_0 \cdot ((M_0 z) \circ (M_1 z)) + c_1 \cdot (M_2 z) == 0 \\ \Longrightarrow & 1 \cdot ((Az) \circ (Bz)) + (-1) \cdot (Cz) == 0 \\ \Longrightarrow & ((Az) \circ (Bz)) - (Cz) == 0 \end{aligned}$$

which is equivalent to the R1CS relation: $Az \circ Bz == Cz$

An example of the conversion from R1CS to CCS implemented in SageMath can be found at
https://github.com/arnaucube/math/blob/master/r1cs-ccs.sage.

Similar relations between Plonkish and AIR arithmetizations to CCS are shown in the CCS paper [2], but for now with the R1CS we have enough to see the CCS generalization idea and to use it for the HyperNova scheme.

1.2 Committed CCS

R_{CCCS} instance: (C, x), where C is a commitment to a multilinear polynomial in $s' - 1$ variables.

Sat if:

i. $\text{Commit}(pp, \widetilde{w}) = C$

ii. $\sum_{i=1}^{q} c_i \cdot \left(\prod_{j \in S_i} \left(\sum_{y \in \{0,1\}^{\log m}} \widetilde{M}_j(x, y) \cdot \widetilde{z}(y) \right) \right)$
where $\widetilde{z}(y) = \widetilde{(w, 1, \mathsf{x})}(x) \; \forall x \in \{0,1\}^{s'}$

1.3 Linearized Committed CCS

R_{LCCCS} instance: $(C, u, \mathsf{x}, r, v_1, \ldots, v_t)$, where C is a commitment to a multilinear polynomial in $s' - 1$ variables, and $u \in \mathbb{F}, \; \mathsf{x} \in \mathbb{F}^l, \; r \in \mathbb{F}^s, \; v_i \in \mathbb{F} \; \forall i \in [t]$.

Sat if:

i. $\text{Commit}(pp, \widetilde{w}) = C$

ii. $\forall i \in [t], \; v_i = \sum_{y \in \{0,1\}^{s'}} \widetilde{M}_i(r, y) \cdot \widetilde{z}(y)$
where $\widetilde{z}(y) = \widetilde{(w, u, \mathsf{x})}(x) \; \forall x \in \{0,1\}^{s'}$

2 Multifolding Scheme for CCS

Recall sum-check protocol notation: $\underline{C \leftarrow \langle P, V(r)\rangle(g, l, d, T)}$ means

$$T = \sum_{x_1 \in \{0,1\}} \sum_{x_2 \in \{0,1\}} \cdots \sum_{x_l \in \{0,1\}} g(x_1, x_2, \ldots, x_l)$$

where g is a l-variate polynomial, with degree at most d in each variable, and T is the claimed value.

Let $s = \log m,\ s' = \log n$.

1. $V \to P : \gamma \in^R \mathbb{F},\ \beta \in^R \mathbb{F}^s$
2. $V : r'_x \in^R \mathbb{F}^s$
3. $V \leftrightarrow P$: sum-check protocol:

$$c \leftarrow \langle P, V(r'_x)\rangle(g, s, d+1, \underbrace{\sum_{j \in [t]} \gamma^j \cdot v_j}_{\text{T}})$$

(in fact, $T = (\sum_{j \in [t]} \gamma^j \cdot v_j) \underbrace{+\gamma^{t+1} \cdot Q(x))}_{=0} = \sum_{j \in [t]} \gamma^j \cdot v_j)$

where:

$$g(x) := \underbrace{\left(\sum_{j \in [t]} \gamma^j \cdot L_j(x)\right)}_{\text{LCCCS check}} + \underbrace{\gamma^{t+1} \cdot Q(x)}_{\text{CCCS check}}$$

$$\text{for LCCCS: } L_j(x) := \widetilde{eq}(r_x, x) \cdot \left(\underbrace{\sum_{y \in \{0,1\}^{s'}} \widetilde{M}_j(x, y) \cdot \widetilde{z}_1(y)}_{\text{this is the check from LCCCS}}\right)$$

$$\text{for CCCS: } Q(x) := \widetilde{eq}(\beta, x) \cdot \left(\underbrace{\sum_{i=1}^{q} c_i \cdot \prod_{j \in S_i} \left(\sum_{y \in \{0,1\}^{s'}} \widetilde{M}_j(x, y) \cdot \widetilde{z}_2(y)\right)}_{\text{this is the check from CCCS}}\right)$$

Notice that

$$v_j = \sum_{y \in \{0,1\}^{s'}} \widetilde{M}_j(r, y) \cdot \widetilde{z}(y) = \sum_{x \in \{0,1\}^s} L_j(x)$$

4. $P \to V$: $((\sigma_1, \ldots, \sigma_t), (\theta_1, \ldots, \theta_t))$, where $\forall j \in [t]$,

$$\sigma_j = \sum_{y \in \{0,1\}^{s'}} \widetilde{M}_j(r'_x, y) \cdot \widetilde{z}_1(y)$$

$$\theta_j = \sum_{y \in \{0,1\}^{s'}} \widetilde{M}_j(r'_x, y) \cdot \widetilde{z}_2(y)$$

where σ_j, θ_j are the checks from LCCCS and CCCS respectively with $x = r'_x$.

5. V: $e_1 \leftarrow \widetilde{eq}(r_x, r'_x)$, $e_2 \leftarrow \widetilde{eq}(\beta, r'_x)$
check:

$$c = \left(\sum_{j \in [t]} \gamma^j e_1 \sigma_j + \gamma^{t+1} e_2 \left(\sum_{i=1}^{q} c_i \cdot \prod_{j \in S_i} \sigma \right) \right)$$

which should be equivalent to the $g(x)$ computed by V, P in the sum-check protocol.

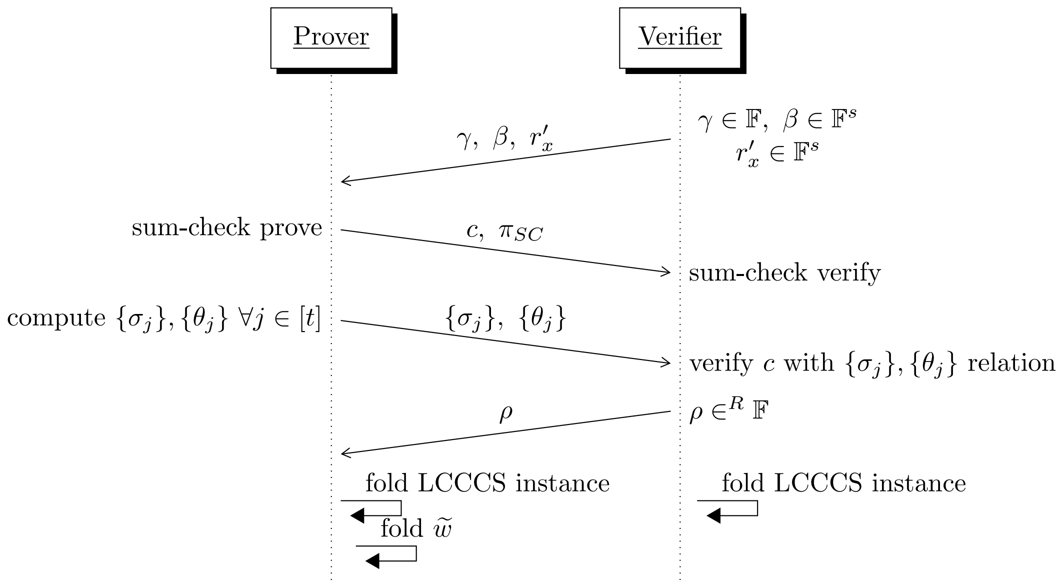
6. $V \to P : \rho \in^R \mathbb{F}$

7. V, P: output the folded LCCCS instance $(C', u', \mathsf{x}', r'_x, v'_1, \ldots, v'_t)$, where $\forall i \in [t]$:

$$\begin{aligned} C' &\leftarrow C_1 + \rho \cdot C_2 \\ u' &\leftarrow u + \rho \cdot 1 \\ \mathsf{x}' &\leftarrow \mathsf{x}_1 + \rho \cdot \mathsf{x}_2 \\ v'_i &\leftarrow \sigma_i + \rho \cdot \theta_i \end{aligned}$$

8. P: output folded witness: $\widetilde{w}' \leftarrow \widetilde{w}_1 + \rho \cdot \widetilde{w}_2$.

Multifolding flow:



Now, to see the verifier check from step 5, observe that in LCCCS, since $\widetilde{w}$ satisfies,

$$\begin{aligned}
v_j &= \sum_{y \in \{0,1\}^{s'}} \widetilde{M}_j(r_x, y) \cdot \widetilde{z}_1(y) \\
&= \sum_{x \in \{0,1\}^s} \underbrace{\widetilde{eq}(r_x, x) \cdot \left(\sum_{y \in \{0,1\}^{s'}} \widetilde{M}_j(x, y) \cdot \widetilde{z}_1(y) \right)}_{L_j(x)} \\
&= \sum_{x \in \{0,1\}^s} L_j(x)
\end{aligned}$$

Observe also that in CCCS, since $\widetilde{w}$ satisfies,

$$0 = \underbrace{\sum_{i=1}^{q} c_i \cdot \prod_{j \in S_i} \left(\sum_{y \in \{0,1\}^{s'}} \widetilde{M}_j(x, y) \cdot \widetilde{z}_2(y) \right)}_{q(x)}$$

we have that

$$G(X) = \sum_{x \in \{0,1\}^s} eq(X, x) \cdot q(x)$$

is multilinear, and can be seen as a Lagrange polynomial where coefficients are evaluations of $q(x)$ on the hypercube.

For an honest prover, all these coefficients are zero, thus $G(X)$ must necessarily be the zero polynomial. Thus $G(\beta) = 0$ for $\beta \in^R \mathbb{F}^s$.

$$\begin{aligned}
0 = G(\beta) &= \sum_{x \in \{0,1\}^s} eq(\beta, x) \cdot q(x) \\
&= \sum_{x \in \{0,1\}^s} \underbrace{\widetilde{eq}(\beta, x) \cdot \overbrace{\sum_{i=1}^{q} c_i \cdot \prod_{j \in S_i} \left(\sum_{y \in \{0,1\}^{s'}} \widetilde{M}_j(x, y) \cdot \widetilde{z}_2(y) \right)}^{q(x)}}_{Q(x)} \\
&= \sum_{x \in \{0,1\}^s} Q(x)
\end{aligned}$$

Note: notice that this past equation is related to Spartan paper [3], lemmas 4.2 and 4.3, where instead of

$$q(x) = \sum_{i=1}^{q} c_i \cdot \prod_{j \in S_i} \left(\sum_{y \in \{0,1\}^{s'}} \widetilde{M}_j(x, y) \cdot \widetilde{z}_2(y) \right)$$

for our R1CS example, we can restrict it to just M_0, M_1, M_2, which would be

$$= \left(\sum_{y \in \{0,1\}^s} \widetilde{M_0}(x, y) \cdot \widetilde{z}(y) \right) \cdot \left(\sum_{y \in \{0,1\}^s} \widetilde{M_1}(x, y) \cdot \widetilde{z}(y) \right) - \sum_{y \in \{0,1\}^s} \widetilde{M_2}(x, y) \cdot \widetilde{z}(y)$$

and we can see that $q(x)$ is the same equation $\widetilde{F}_{io}(x)$ that we had in Spartan:

$$\widetilde{F}_{io}(x) = \left(\sum_{y \in \{0,1\}^s} \widetilde{A}(x, y) \cdot \widetilde{z}(y) \right) \cdot \left(\sum_{y \in \{0,1\}^s} \widetilde{B}(x, y) \cdot \widetilde{z}(y) \right) - \sum_{y \in \{0,1\}^s} \widetilde{C}(x, y) \cdot \widetilde{z}(y)$$

where

$$Q_{io}(t) = \sum_{x \in \{0,1\}^s} \widetilde{F}_{io}(x) \cdot \widetilde{eq}(t, x) = 0$$

and V checks $Q_{io}(\tau) = 0$ for $\tau \in^R \mathbb{F}^s$, which in HyperNova is $G(\beta) = 0$ for $\beta \in^R \mathbb{F}^s$. $Q_{io}(\cdot)$ is a zero-polynomial ($G(\cdot)$ in HyperNova), it evaluates to zero for all points in its domain iff $\widetilde{F}_{io}(\cdot)$ evaluates to zero at all points in the s-dimensional boolean hypercube.

$$\begin{gathered}
\text{Spartan} \longleftrightarrow \text{HyperNova} \\
\tau \longleftrightarrow \beta \\
\widetilde{F}_{io}(x) \longleftrightarrow q(x) \\
Q_{io}(\tau) \longleftrightarrow G(\beta)
\end{gathered}$$

So, in HyperNova

$$0 = \sum_{x \in \{0,1\}^s} Q(x) = \sum_{x \in \{0,1\}^s} \widetilde{eq}(\beta, x) \cdot q(x)$$

Comming back to HyperNova equations, we can now see that

$$\begin{aligned} c &= g(r'_x) \\ &= \left(\sum_{j \in [t]} \gamma^j \cdot L_j(r'_x) \right) + \gamma^{t+1} \cdot Q(r'_x) \\ &= \left(\sum_{j \in [t]} \gamma^j \cdot \overbrace{e_1 \cdot \sigma_j}^{L_j(r'_x)} \right) + \gamma^{t+1} \cdot \overbrace{e_2 \cdot \sum_{i \in [q]} c_i \prod_{j \in S_i} \theta_j}^{Q(x)} \end{aligned}$$

where $e_1 = \widetilde{eq}(r_x, r'_x)$ and $e_2 = \widetilde{eq}(\beta, r'_x)$.
Which is the check that V performs at step 5.

A Appendix: Some details

This appendix contains some notes on things that don't specifically appear in the paper, but that would be needed in a practical implementation of the scheme.

A.1 Matrix and Vector to Sparse Multilinear Extension

Let $M \in \mathbb{F}^{m \times n}$ be a matrix. We want to compute its MLE

$$\widetilde{M}(x_1, \ldots, x_l) = \sum_{e \in \{0,1\}^l} M(e) \cdot \widetilde{eq}(x, e)$$

We can view the matrix $M \in \mathbb{F}^{m \times n}$ as a function with the following signature:

$$M(\cdot) : \{0,1\}^s \times \{0,1\}^{s'} \to \mathbb{F}$$

where $s = \lceil \log m \rceil,\ s' = \lceil \log n \rceil$.
An entry in M can be accessed with a $(s + s')$-bit identifier.
eg.:

$$M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \in \mathbb{F}^{3 \times 2}$$

$m = 3,\ n = 2, \quad s = \lceil \log 3 \rceil = 2,\ s' = \lceil \log 2 \rceil = 1$
So, $M(x, y) = x$, where $x \in \{0,1\}^s,\ y \in \{0,1\}^{s'},\ x \in \mathbb{F}$

$$M = \begin{pmatrix} M(00,0) & M(01,0) & M(10,0) \\ M(00,1) & M(01,1) & M(10,1) \end{pmatrix} \in \mathbb{F}^{3 \times 2}$$

This logic can be defined as follows:

Algorithm 1 Generating a Sparse Multilinear Polynomial from a matrix

```
set empty vector v ∈ (index: ℤ, x : 𝔽^{s×s'})
for i to m do
    for j to n do
        if M_{i,j} ≠ 0 then
            v.append({index : i · n + j, x : M_{i,j}})
        end if
    end for
end for
return v                ▷ v represents the evaluations of the polynomial
```

Once we have the polynomial, its MLE comes from

$$\widetilde{M}(x_1, \ldots, x_{s+s'}) = \sum_{e \in \{0,1\}^{s+s'}} M(e) \cdot \widetilde{eq}(x, e)$$

$$M(X) \in \mathbb{F}[X_1, \ldots, X_s]$$

Multilinear extensions of vectors Given a vector $u \in \mathbb{F}^m$, the polynomial $\widetilde{u}$ is the MLE of u, and is obtained by viewing u as a function mapping ($s = \log m$)

$$u(x) : \{0,1\}^s \to \mathbb{F}$$

$\widetilde{u}(x, e)$ is the multilinear extension of the function $u(x)$

$$\widetilde{u}(x_1, \ldots, x_s) = \sum_{e \in \{0,1\}^s} u(e) \cdot \widetilde{eq}(x, e)$$

References

[1] Abhiram Kothapalli and Srinath Setty. Hypernova: Recursive arguments for customizable constraint systems. Cryptology ePrint Archive, Paper 2023/573, 2023. https://eprint.iacr.org/2023/573.

[2] Srinath Setty, Justin Thaler, and Riad Wahby. Customizable constraint systems for succinct arguments. Cryptology ePrint Archive, Paper 2023/552, 2023. https://eprint.iacr.org/2023/552.

[3] Srinath Setty. Spartan: Efficient and general-purpose zksnarks without trusted setup. Cryptology ePrint Archive, Paper 2019/550, 2019. https://eprint.iacr.org/2019/550.