HyperNova introduction

2023-07-25
0xPARC, London

IVC

For a function F, with initial input z_0, an IVC scheme allows a prover to produce a proof π_i for the statement $z_i = F^{(i)}(z_0)$, given a proof π_{i-1} for the statement $z_{i-1} = F^{(i-1)}(z_0)$
TODO add draw TODO add reference to Valiant paper (2008)

Recursion before folding schemes

We used to use recursive SNARKs to achieve IVC.

- Prove verification in circuit: inside a circuit, verify another proof
  - eg. verifying a Groth16 proof inside a Groth16 circuit.
- Amortized accumulation
  - eg. Halo

R1CS refresher

R1CS instance: $(\{A, B, C\} \in \mathbb{F}^{m \times n},\ io,\ m,\ n,\ l)$, such that for $z = (io \in \mathbb{F}^l, 1, w \in \mathbb{F}^{m-l-1}) \in \mathbb{F}^m$,

$$Az \circ Bz = Cz$$

Typically we use some scheme to prove that the previous equation is fullfilled by some private w (eg. Groth16, Marlin, Spartan, etc).

Random linear combination

Combine 2 instances together through a random linear comibnation, and the outputted instance will still satisfy the relation.

- Have 2 values x_1, x_2.
- Set $r \in^R \mathbb{F}$
- Compute $x_3 = x_1 + r \cdot x_2$.

Random linear combination

Combine 2 instances together through a random linear comibnation, and the outputted instance will still satisfy the relation.

- Have 2 values x_1, x_2.
- Set $r \in^R \mathbb{F}$
- Compute $x_3 = x_1 + r \cdot x_2$.

Combined with homomorphic commitments

- We can do random linear combinations with the commitments and their witnesses, and the output can still be opened

Folding schemes

We're not verifying the entire proof

- Take n instances and 'batch' them together
  - Folds k (eg. 2) instances (eg. R1CS instances) and their respective witnesses into a signle one
- At the end of the chain of folds, we just prove that the last fold is correct through a SNARK
  - Which implies that all the previous folds were correct

Folding schemes

We're not verifying the entire proof

- Take n instances and 'batch' them together
  - Folds k (eg. 2) instances (eg. R1CS instances) and their respective witnesses into a signle one
- At the end of the chain of folds, we just prove that the last fold is correct through a SNARK
  - Which implies that all the previous folds were correct

In Nova: folding without a SNARK, we just reduce the satisfiability of the 2 inputted instances to the satisfiability of the single outputted one.
[TODO image of multiple folding iterations]

Relaxed R1CS

We work with *relaxed R1CS*

$$Az \circ Bz = u \cdot Cz + E$$

($=$ R1CS when $u = 1,\ E = 0$)

- main idea: allows us to fold, but accumulates *cross terms*

Relaxed R1CS

We work with *relaxed R1CS*

$$Az \circ Bz = u \cdot Cz + E$$

($=$ R1CS when $u = 1,\ E = 0$)

- main idea: allows us to fold, but accumulates *cross terms*
- when we do the *relaxed* of higher degree equations (eg. plonkish), the cross terms grow (eg. Sangria with higher degree gates)

NIFS - setup

V and P: *committed relaxed R1CS* instances

$$\varphi_1 = (\overline{E}_1, u_1, \overline{w}_1, x_1)$$
$$\varphi_2 = (\overline{E}_2, u_2, \overline{w}_2, x_2)$$

P: witnesses

$$(E_1, r_{E_1}, w_1, r_{w_1})$$
$$(E_2, r_{E_2}, w_2, r_{w_2})$$

Let $z_1 = (w_1, x_1, u_1)$ and $z_2 = (w_2, x_2, u_2)$.

NIFS

- V, P: folded instance $\varphi = (\overline{E}, u, \overline{w}, x)$

$$\begin{aligned} \overline{E} &= \overline{E}_1 + r\overline{T} + r^2\overline{E}_2 \\ u &= u_1 + ru_2 \\ \overline{w} &= \overline{w}_1 + r\overline{w}_2 \\ x &= x_1 + rx_2 \end{aligned}$$

- P: folded witness (E, r_E, w, r_W)

$$\begin{aligned} E &= E_1 + rT + r^2E_2 \\ r_E &= r_{E_1} + r \cdot r_T + r^2 r_{E_2} \\ w &= w_1 + rw_2 \\ r_W &= r_{w_1} + r \cdot r_{w_2} \end{aligned}$$

NIFS

- V, P: folded instance $\varphi = (\overline{E}, u, \overline{w}, x)$

$$\overline{E} = \overline{E}_1 + r\overline{T} + r^2\overline{E}_2$$
$$u = u_1 + ru_2$$
$$\overline{w} = \overline{w}_1 + r\overline{w}_2$$
$$x = x_1 + rx_2$$

- P: folded witness (E, r_E, w, r_W)

$$E = E_1 + rT + r^2 E_2$$
$$r_E = r_{E_1} + r \cdot r_T + r^2 r_{E_2}$$
$$w = w_1 + rw_2$$
$$r_W = r_{w_1} + r \cdot r_{w_2}$$

Note: T are the cross-terms comming from combining the two R1CS instances from

$$\begin{aligned} Az \circ Bz &= A(z_1 + r \cdot z_2) \circ B(z_1 + rz_2) \\ &= Az_1 \circ Bz_1 + r(Az_1 \circ Bz_2 + Az_2 \circ Bz_1) + r^2(Az_2 \circ Bz_2) = \ldots \end{aligned}$$

NIFS

$$E = E_1 + r\underbrace{(Az_1 \circ Bz_2 + Az_2 \circ Bz_1 - u_1Cz_2 - u_2Cz_1)}_{\text{cross-terms}} + r^2E_2$$

$Az \circ Bz = uCz + E$ will hold for valid z (which comes from valid $z_1,\ z_2$).
[TODO add image of function F' with F inside with extra checks]

NIFS

Each fold: $2\ EC_{Add} + 1\ EC_{Mul} + 1\ hash$
20k R1CS constraints (using curve cycles)
(so folding makes sense when we have a circuit with more than $2 \cdot 20k$ constraints)

NIFS

Each fold: $2\ EC_{Add} + 1\ EC_{Mul} + 1\ hash$
20k R1CS constraints (using curve cycles)
(so folding makes sense when we have a circuit with more than $2 \cdot 20k$ constraints)
After all the folding iterations, Nova generates a SNARK proving the last folding instance.
In Nova implementation, they use Spartan.

Benchmarks

Benchmarks that Oskar, Carlos, et al did during the Vietnam residency in April https://hackmd.io/u3qM9s_YR1emHZSg3jteQA

Size	Constraints	Time
2KB	883k	320ms
4KB	1.7m	521ms
8KB	3.4m	1s
16KB	6.8m	1.9s
32KB	13.7m	4.1s

eg. for 8kb, x100 Halo2 and Plonky2

(this is for the folding, without the last snark)

SuperNova

- iteration on Nova, combining *different circuits* in a single one with *selectors*
- so we can work with a big circuit with *subcircuits* without paying the whole size cost on each iteration
- in IVC terms: fold multiple F_i in a single F' (in Nova was a single F in F')

This is useful for example for a VM, doing one F_i for each opcode

R1CS to CCS example

- Kind of a generalization of constraint systems
- Can translate R1CS,Plonk,AIR to CCS

R1CS to CCS example

- Kind of a generalization of constraint systems
- Can translate R1CS,Plonk,AIR to CCS

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

R1CS to CCS example

- Kind of a generalization of constraint systems
- Can translate R1CS,Plonk,AIR to CCS

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:

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

In our R1CS-to-CCS parameters is equivalent to

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

Multifolding

- Nova: 2-to-1 folding
- HyperNova: multifolding, k-to-1 folding
- We fold while through a SumCheck proving the correctness of the fold

SumCheck's polynomial work is trivial, most of the cost comes from Poseidon hash in the transcript
[TODO WIP section]

Multifolding - Overview

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

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

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

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

Multifolding - Overview

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

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

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

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

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

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

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

8. P: output folded witness and the folded r'_w:

$$\widetilde{w}' \leftarrow \widetilde{w}_1 + \rho \cdot \widetilde{w}_2$$
$$r'_w \leftarrow r_{w_1} + \rho \cdot r_{w_2}$$

Mysteries & unsolved things

- how HyperNova compares to Protostar
- prover knows the full witness [TODO update/rm this]

[TODO WIP section]

Wrappup

- HyperNova: https://eprint.iacr.org/2023/573
- multifolding PoC on arkworks:
  github.com/privacy-scaling-explorations/multifolding-poc
- PSE hypernova WIP
  github.com/privacy-scaling-explorations/Nova