Recursive snarks - Nova

Karthik Inbasekar,[a]

[a] *Ingonyama,*
remote, Israel

E-mail: karthik@ingonyama.com

Abstract: These notes are based on the paper [1]

1 Introduction

NOVA is a SNARK for iterative computations (recursions) where the Prover, given a public input x applies an operation F repetitively and outputs y. Formally one can represent this as

$$y = F^{(n)}(x) \tag{1.1}$$

where for example $F^{(2)}(x) = F(F(x))$ is a potentially non-deterministic function, i.e in general it can take in a public input x and a witness w to produce an output $y = F(x, w)$. In other words it has the property

$$y_i = F(F(\dots F(F(x, w_0)w_1), \dots, w_{i-1}), w_i) \tag{1.2}$$

The recursive nature, enables a Prover to prove statements about prior proofs as we will see later. The key characteristics of NOVA are that

- It does not require trusted setup (the commitment scheme needs additive homomorphism eg: Pedersen commitments)
- It does not execute any polynomial multiplications/divisions and hence **n**o FFT's (Fast Fourier Transforms)
- The function F is specified by a relaxed Rank 1 Constraints System (R1CS).
- The prover time is dominated by **t**wo multi-exponentials of size $\mathcal{O}(C)$ where $|C|$ is the size of the computation at each iteration.
- An expected speed up of up-to 2 orders of magnitude based on comparison with [2–4]
- The verifier circuit size is fixed, and is claimed to be the lowest recursion threshold in the literature.
- By using proof compression to $\mathcal{O}(\log(C))$ the proof sizes are in few Kb's.

A naive approach to evaluating (1.1) is to unroll all the n iterations into a single operation or circuit. Then, to apply an efficient zk SNARK to produce a valid proof. As one can anticipate

- This is a memory intensive operation and the requirement scales as $\Omega(n|F|)$
- It is not incrementally updateable. If the prover did the unrolling for n operations, and computed y, in order to get the result for the $n+1$'th iteration, the one has to repeat the process all over again. (may be this can be fixed?)
- The verifier's time can depend on n (possibly SNARK dependent)

Instead of naive unrolling, in the NOVA compiler, there is an IVC (Incrementally verifiable computation) where the prover does not attempt to compute $F^n(x)$, but instead proceeds in incremental steps.

- At each step, the prover proves an augumented circuit F' that executes F and a verifier circuit. To be precise, after the ith application of F, the prover outputs y_i and the the proof of work π_i is the statement that $F^{(i)}(x) = y_i$.
- Given y_i and π_i , the prover **(or in general any of the provers)** can apply $y_{i+1} \leftarrow F(y_i)$ and compute the proof π_{i+1}.
- At step i the prover only needs to remember the output of the previous step y_{i-1} and the preceding proof π_{i-1},
- The verifier circuit uses additive homomorphism properties of polynomial commitments [1] to cheaply combine all proofs $\pi_1, \ldots \pi_n$ into one verifiable statement.
- The alternative to folding, is to verify the proofs generated at every iteration, and this results in a signficant "overhead of recursion".
- Since additive homomorphism combines the proofs generated at each step, the verifier needs to check only the final output of the IVC that includes the final proof that is produced.
- Note that, the cost of checking the final proof is the cost of checking one claim, which is why **homomorphic commitments** are used in this scheme.

2 Folding Schemes

2.1 A naive folding scheme

In [1], they introduce a cryptographic primitive called the folding scheme. It is defined with respect to an **NP**(Non-deterministic polynomial time) relation, as an interactive protocol that runs between an untrusted prover and a verifier algorithm. The initial state consists of the prover and verifier holding two N sized Instances: U_1, U_2, in addition the prover holds a Witness vector for each instance (W_1, W_2) At a high level

[1] (this is all the folding scheme is about)

- The Prover sends a additively homomorphic commitment to the Verifier
- The verifier does a computation and outputs a U such that U can be satisfied only if both U_1 and U_2 are satisfied. The outputed instance is also N sized, this is similar to the IPA (Inner Product Argument) of Bulletproof [5] where the prover and the verifier combine two N sized inner products into a single N sized inner product.
- The folding scheme is called non-trivial, provided verifying U itself is concretely cheaper than verifying U_1/U_2 if the prover shares their respective witnesses W_1/W_2.

There is a reason why a naive folding scheme with the usual R1CS fails. For example, consider three $m \times m$ matrices A, B, C and a tuple $Z = (W, x, 1)$. The question is does there exist a W such that

$$(A \cdot Z) \circ (B \cdot Z) = (C \cdot Z) \tag{2.1}$$

where the $\circ$ represents the (Hadamard product) and $\cdot$ represents the scalar product, W is a witness held by $\mathcal{P}$. In this example, the instance U is of the form

$$U = ((A, B, C), (\hat{W}, x)) \tag{2.2}$$

where $\hat{W}$ is a commitment to the Witness. The commitments are assumed to be additively homomorphic. In the initial state, the prover holds the witness W_1, W_2 and the verifier initiates the instances $U_1 = (\hat{W}_1, x_1), U_2 = (\hat{W}_2, x_2)$. The protocol would proceed as follows

1. $\mathcal{V}$ picks a random r and sends it to $\mathcal{P}$.
2. $\mathcal{V}$ computes the new instance $U_3 \leftarrow (\hat{W}, x)$ where

$$\begin{aligned} \hat{W} &\leftarrow \hat{W}_1 + r.\hat{W}_2 \\ x &\leftarrow x_1 + r.x_2 \end{aligned} \tag{2.3}$$

   This is possible because the commitments $\hat{W}$ are linearly homomorphic.

3. $\mathcal{P}$ computes $W \leftarrow W_1 + r \cdot W_2$
4. The claim is that U_3 does not satisfy a R1CS equation (2.1).

The above scheme does not satisfy the relation (2.1) since it does not satisfy completeness or closure. It is easy to see, for eg, take $Z = Z_1 + r.Z_2$ and evaluate the LHS of (2.1) to get

$$(A \cdot Z_1) \circ (B \cdot Z_1) + r \cdot ((A \cdot Z_1) \circ (B \cdot Z_2) + (A \cdot Z_2) \circ (B \cdot Z_1)) + r^2 \cdot ((A \cdot Z_2) \circ (B \cdot Z_2)) \tag{2.4}$$

where even if $(A \cdot Z_i) \circ (B \cdot Z_i) = (C \cdot Z_i)$ for $i = 1, 2$, the RHS of (2.1) evaluates to

$$C \cdot Z = C \cdot Z_1 + r \cdot (C \cdot Z_2) \tag{2.5}$$

Thus, there exists some r for which (2.1) is not true or in other words there is no folding of the structure of an R1CS instance as described above.

The problem is essentially the cross terms that appear in the LHS. It is fairly straightforward to guess that if we could change the right hand side of (2.1) by including a constant and a shift term, it could probably work. Except that, it wont trivially work unless the prover does send a non-trivial information as commitment, since the verifier has no way of knowing what cross terms to add!. We will describe this below.

2.2 Relaxed folding R1CS scheme

In this section we describe a relaxed folding R1CS scheme with commitments. Given three $m \times m$ matrices A, B, C and a tuple $Z = (W, x, u)$, where u is a scalar. The question is does there exist a W such that

$$(A \cdot Z) \circ (B \cdot Z) = u \cdot (C \cdot Z) + E \tag{2.6}$$

Where the Witness vector is now a concatenation (W, E), where E is a vector referred to as the slack vector. The instance (2.2) is now modified to include the commitments $(\hat{W}, \hat{E})$ as follows

$$U = ((A, B, C)(\hat{W}, \hat{E}, u, x)) \tag{2.7}$$

With these definitions, the initial state is as follows. The prover holds the Witness: W_1, W_2, and the verifier instantiates $U_1 = (\hat{W}_1, \hat{E}_1, u_1, x_1), U_2 = (\hat{W}_2, \hat{E}_2, u_2, x_2)$.

The prover's claim is that the both U_1 and U_2 will satisfy (2.6). The naive way to do this is to open up the commitments $\hat{W}$ and check that the relaxed R1CS scheme is satisfied, which would be expensive. Instead, the verifier takes a random linear combination of the two instances and check that the derived claim is of the same form and indeed satisfies (2.6) iff both U_1 and U_2 are true.

The protocol proceeds as follows

1. In the first step, $\mathcal{P}$ sends a hint to $\mathcal{V}$

   $$T \leftarrow \text{commit}((A \cdot Z_1) \circ (B \cdot Z_2) + (A \cdot Z_2) \circ (B \cdot Z_1) - u_1 \cdot (C \cdot Z_2) - u_2 \cdot (C \cdot Z_1)) \tag{2.8}$$

   In NOVA any additively homomorphic commitment scheme is sufficient for the rest of the protocol.

2. $\mathcal{V}$ responds with a random challenge r and sends it to $\mathcal{P}$.

3. Further $\mathcal{V}$ computes a new instance (2.7) using the additive homomorphism

   $$\begin{aligned} \hat{W} &\leftarrow \hat{W}_1 + r \cdot \hat{W}_2 \\ \hat{E} &\leftarrow \hat{E}_1 + r \cdot T + r^2 \cdot \hat{E}_2 \\ x &\leftarrow x_1 + r \cdot x_2 \\ u &\leftarrow u_1 + r \cdot u_2 \end{aligned} \tag{2.9}$$

4. $\mathcal{P}$ computes

   $$\begin{aligned} W &\leftarrow W_1 + r \cdot W_2 \\ E &\leftarrow E_1 + r \cdot T + r^2 \cdot E_2 \end{aligned} \tag{2.10}$$

Now, the claim is that the computations of the prover and verifier will agree (assuming honest prover) provided that they use the same commitment schemes.

The folding property is relatively easy to see. Let us take equation (2.6), and substitute $Z = Z_1 + r \cdot Z_2$ and $u = u_1 + r \cdot u_2$. After expanding and distributing in r, we get

$$\begin{aligned}(A \cdot Z) \circ (B \cdot Z) &= (A \cdot Z_1) \circ (B \cdot Z_1) + r \cdot ((A \cdot Z_1) \circ (B \cdot Z_2) + (A \cdot Z_2) \circ (B \cdot Z_1)) \\ &\quad + r^2 \cdot ((A \cdot Z_2) \circ (B \cdot Z_2)) \\ &= u_1 \cdot (C \cdot Z_1) + r \cdot (u_1 \cdot (C \cdot Z_2) + u_2 \cdot (C \cdot Z_1)) + r^2 \cdot u_2 \cdot (C \cdot Z_2) + E \end{aligned} \tag{2.11}$$

and rewrite it as

$$\begin{aligned}E = &((A \cdot Z_1) \circ (B \cdot Z_1) - u_1 \cdot (C \cdot Z_1)) + r^2 \cdot ((A \cdot Z_2) \circ (B \cdot Z_2) - u_2 \cdot (C \cdot Z_2)) \\ &+ r \cdot ((A \cdot Z_1) \circ (B \cdot Z_2) + (A \cdot Z_2) \circ (B \cdot Z_1) - u_1 \cdot (C \cdot Z_2) - u_2 \cdot (C \cdot Z_1)) \end{aligned} \tag{2.12}$$

It is then clear that with the identification

$$E_i = (A \cdot Z_i) \circ (B \cdot Z_i) - u_i \cdot (C \cdot Z_i);\ \forall\ i = 1, 2 \tag{2.13}$$

$$T = (A \cdot Z_1) \circ (B \cdot Z_2) + (A \cdot Z_2) \circ (B \cdot Z_1) - u_1 \cdot (C \cdot Z_2) - u_2 \cdot (C \cdot Z_1) \tag{2.14}$$

and when the committed vector by the Prover is T (2.13), then (2.6) holds for any random r with probability 1, or the folding scheme has the same structure after any given instance. In case, the prover commits to a different vector other than T, then the elements of the LHS and RHS of (2.6) will be two distinct degree-2 polynomials in $r \in \mathbb{F}$ and hence will disagree with probability $1 - \frac{2}{|\mathbb{F}|}$. Notice that the prover will commit terms to T at every step, it follows that T will absorb all the cross terms generated at each step and thus the computation preserves the folding scheme.

2.3 IVC from folding scheme for Relaxed R1CS

The relaxed R1CS system described in the previous section is the running instance F in each round of the prover computation. The augumented function F' (described in §1 has two roles. Firstly, it applies F and secondly the verifier circuit does the verifier's work in the current step of the folding scheme. Thus there is an overhead at the provers end and it performs the verifier's processing of π across all the rounds in the protocol.

Note that the verifier circuit in F' keeps track of the

- round count (number of applications of F)
- $u \in \mathcal{F}$ the scalar, commitment to the witness $\hat{W}$, commitment to the slack vector $\hat{E}$ and x the public input to F at each instance of U.
- Output of previous round $y_{i-1} = F^{(i-1)}(x)$
- Output of current round $y_i = F^{(i)}(x)$
- The interactive protocol of the relaxed R1CS can be replaced with a non-interactive one by using the Fiat-Shamir heuristic.

In the random message interaction of the prover-verifier system, the random message sent by the verifier is equivalent to blackbox access from a random oracle, and this is equivalent to the output of a hash function. In summary, all public inputs and in this case, (the prover messages too) are replaced with their corresponding hashes and this makes the protocol non interactive.

1. At the beginning of round 1 the prover sends the public vector (x, y_1) where $y_1 = F(x)$ the commitment to the witness $\hat{W}_1$. Begins a running R1CS instance

$$U : \quad (AZ) \circ (BZ) = u(CZ) + E$$

2. Upon recieving the prover's message, the verifier circuit begins the round count, sets the state vector to (x, y_1), sets the witness vector $\hat{W} = \hat{W}_1$, sets the scalar $u = 1$, and $\hat{E} = 0$ (in the beginning there is no folding operation) and begins the committed R1CS instance

$$U_i : \quad (AZ_i) \circ (BZ_i) = (CZ_i)$$

. The goal of the rounds $i > 1$ is to fold U_i into U.

Acknowledgments

We stand on the shoulders of giants.

References

[1] A. Kothapalli, S. Setty, and I. Tzialla, "Nova: Recursive zero-knowledge arguments from folding schemes." Cryptology ePrint Archive, Report 2021/370, 2021. https://ia.cr/2021/370.

[2] J. Groth, "On the size of pairing-based non-interactive arguments." Cryptology ePrint Archive, Report 2016/260, 2016. https://ia.cr/2016/260.

[3] S. Bowe, J. Grigg, and D. Hopwood, "Recursive proof composition without a trusted setup." Cryptology ePrint Archive, Report 2019/1021, 2019. https://ia.cr/2019/1021.

[4] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge." Cryptology ePrint Archive, Report 2019/953, 2019. https://ia.cr/2019/953.

[5] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more." Cryptology ePrint Archive, Report 2017/1066, 2017. https://ia.cr/2017/1066.