



geometry

A Unifying Framework for Folding and
Accumulation Schemes

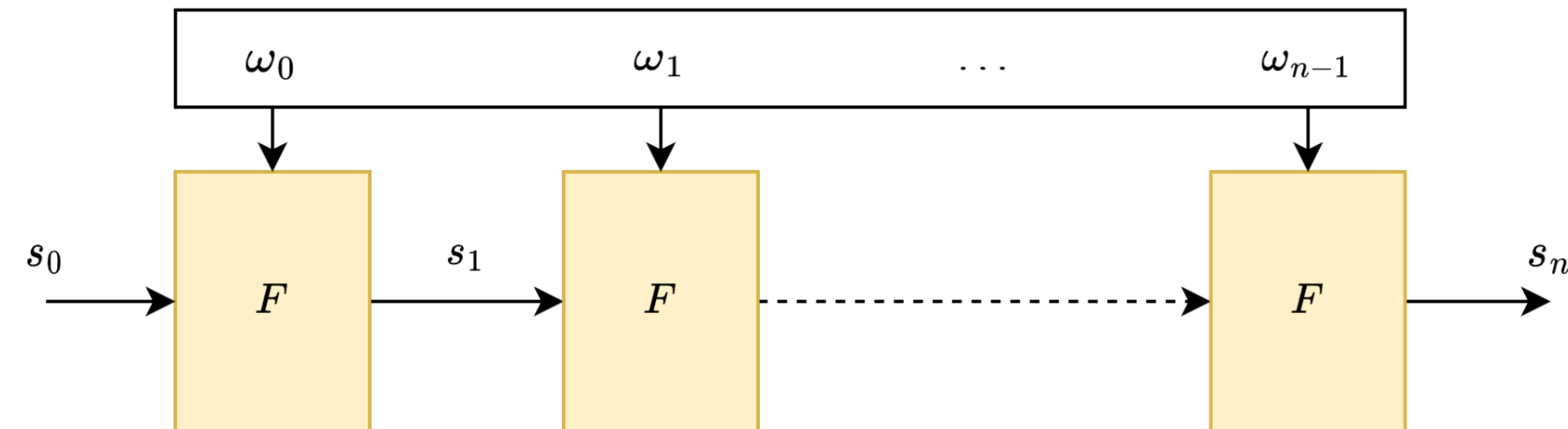
Nicolas Mohnblatt

J U L I 2 3



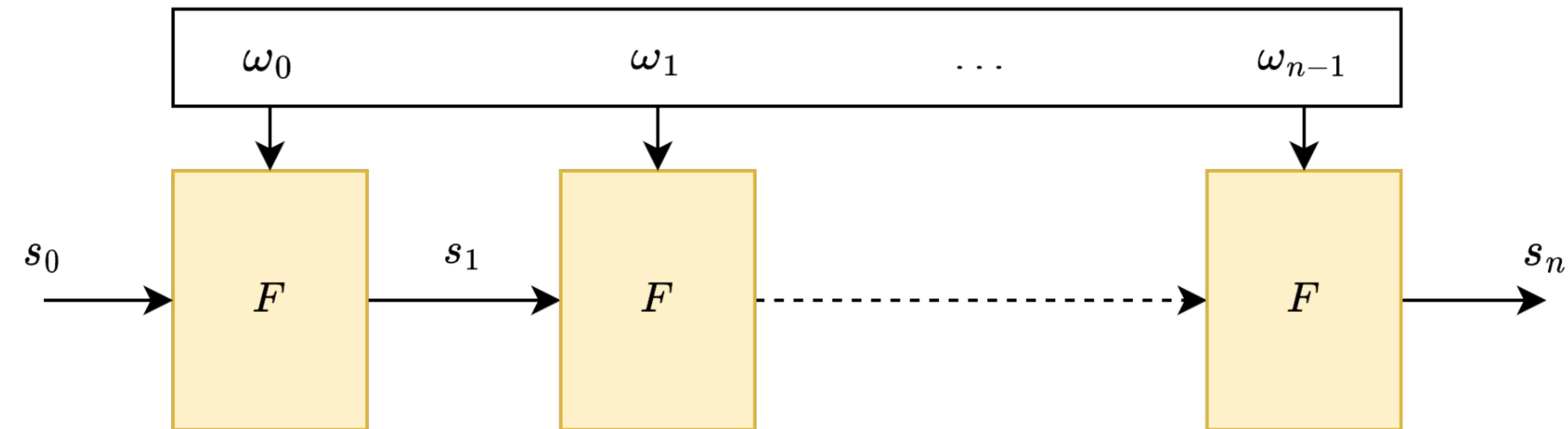
zkVM from Incrementally Verifiable Computation

Imagine a long computation that results from n applications of a function F :



zkVM from Incrementally Verifiable Computation

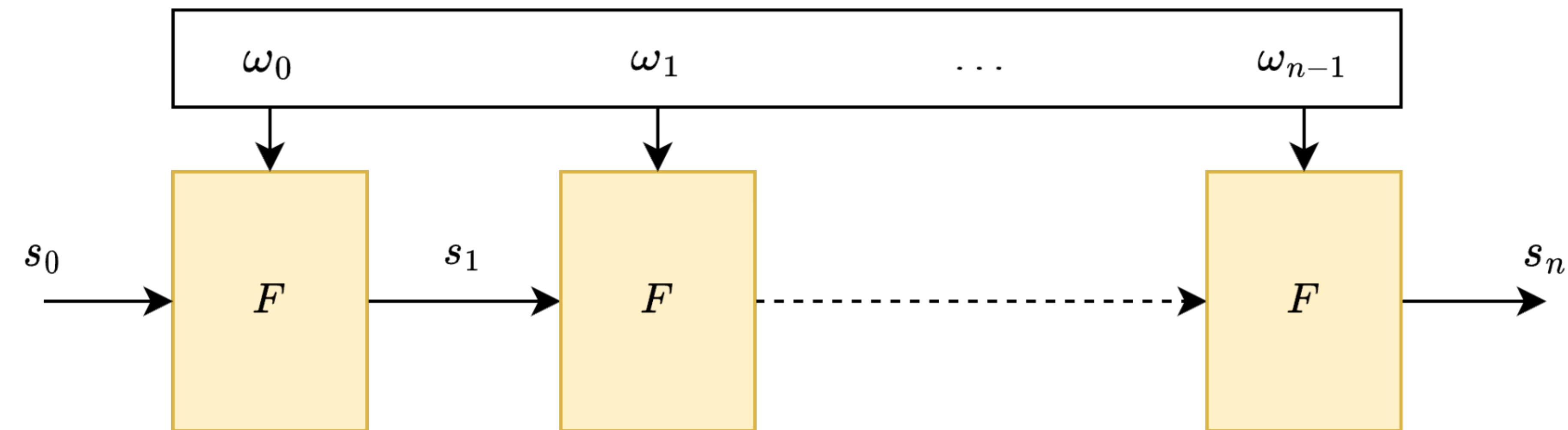
Imagine a long computation that results from n applications of a function F :



We want to produce a succinct proof that the prover knows $\omega_0, \omega_1, \dots, \omega_{n-1}$ such that s_n is the correct final state

zkVM from Incrementally Verifiable Computation

Imagine a long computation that results from n applications of a function F :



We want to produce a succinct proof that the prover knows $\omega_0, \omega_1, \dots, \omega_{n-1}$ such that s_n is the correct final state

Setting F to be the transition function of a VM allows to produce proofs “as the VM advances”

How to build IVC?



REDUCE - AND - COMBINE

How to build IVC?

Nested Amortization!

Recursive Proof Composition without a Trusted Setup

Sean Bowe¹, Jack Grigg¹, and Daira Hopwood¹

¹ Electric Coin Company
 {sean,jack,daira}@electriccoin.co
<https://electriccoin.co/>

Abstract. Non-interactive arguments of knowledge are powerful cryptographic tools that can be used to demonstrate the faithful execution of arbitrary computations with publicly verifiable proofs. Increasingly efficient protocols have been described in recent years, with verification time and/or communication complexity that is sublinear in the size of the computation being described. These efficiencies can be exploited to realize *recursive proof composition*: the concept of proofs that attest to the correctness of other instances of themselves, thereby allowing large computational effort to be incrementally verified. All previously known realizations of recursive proof composition have required a trusted setup and cycles of expensive pairing-friendly elliptic curves. We obtain and implement **Halo**, the first practical example of recursive proof compo-



REDUCE - A N D - COMBINE

How to build IVC?

Recursive Proof Composition
Sean Bowe¹, Jack Grigg², Daira Hopwood¹

¹ Electric Coin Company
² Ethereum Foundation
<https://electriccoincompany.com>

Abstract. Non-interactive arguments of knowledge are powerful cryptographic tools that can be used to demonstrate the faithful execution of arbitrary computations with publicly verifiable proofs. Increasingly efficient protocols have been described in recent years, with verification time and/or communication complexity that is sublinear in the size of the computation being described. These efficiencies can be exploited to realize *recursive proof composition*: the concept of proofs that attest to the correctness of other instances of themselves, thereby allowing large computational effort to be incrementally verified. All previously known realizations of recursive proof composition have required a trusted setup and cycles of expensive pairing-friendly elliptic curves. We obtain and implement **Halo**, the first practical example of recursive proof composition.

Proof-Carrying Data from Accumulation Schemes

Benedikt Bünz
benedikt@cs.stanford.edu
Stanford University

Alessandro Chiesa
alexch@berkeley.edu
UC Berkeley

Pratyush Mishra
pratyush@berkeley.edu
UC Berkeley

Halo Infinite: Proof-Carrying Data from Additive Polynomial Commitments

Dan Boneh¹ Justin Drake² Ben Fisch¹ Ariel Gabizon³

¹Stanford University ²Ethereum Foundation ³AZTEC Protocol

Abstract

Polynomial commitment schemes (PCS) have recently been in the spotlight for their key role in building SNARKs. A PCS provides the ability to commit to a polynomial over a finite field and prove its evaluation at points. A *succinct* PCS has commitment and evaluation proof size sublinear in the degree of the polynomial. An *efficient* PCS has sublinear proof verification. Any efficient and succinct PCS can be used to construct a SNARK with similar security and efficiency characteristics (in the random oracle model).

Proof-carrying data (PCD) enables a set of parties to carry out an indefinitely long distributed computation where every step along the way is accompanied by a proof of correctness. It generalizes *incrementally verifiable computation* and can even be used to construct SNARKs. Until recently, however, the only known method for constructing PCD required expensive SNARK recursion. A system called *Halo* first demonstrated a new methodology for building PCD without SNARKs, exploiting an aggregation property of the *Bulletproofs* inner-product argument. The construction was *heuristic* because it makes non-black-box use of a concrete instantiation of the Fiat-Shamir transform. We expand upon this methodology to show that PCD can be (heuristically) built from any homomorphic polynomial commitment scheme (PCS), even if the PCS evaluation proofs are neither succinct nor efficient. In fact, the Halo methodology extends to any PCS that has an even more general property, namely the ability to aggregate linear combinations of commitments into a new succinct commitment that!

Accumulation Schemes!



How to build IVC?

Accumulation Schemes!
Split accumulation schemes!

Proof-Carrying Data from Accumulation Schemes

Proof-Carrying Data without Succinct Arguments

Re

Benedikt Bünz

benedikt@cs.stanford.edu
Stanford University

Alessandro Chiesa

alexch@berkeley.edu
UC Berkeley

William Lin

will.lin@berkeley.edu
UC Berkeley

Pratyush Mishra

pratyush@berkeley.edu
UC Berkeley

Nicholas Spooner

nspooner@bu.edu
Boston University

December 1, 2021

Abstract

Proof-carrying data (PCD) is a powerful cryptographic primitive that enables mutually distrustful parties to perform distributed computations that run indefinitely. Known approaches to construct PCD are based on succinct non-interactive arguments of knowledge (SNARKs) that have a succinct verifier or a succinct accumulation scheme.

In this paper we show how to obtain PCD without relying on SNARKs. We construct a PCD scheme given any non-interactive argument of knowledge (e.g., with linear-size arguments) that has a *split accumulation scheme*, which is a weak form of accumulation that we introduce.

Moreover, we construct a transparent non-interactive argument of knowledge for R1CS whose split accumulation is verifiable via a (small) *constant number of group and field operations*. Our construction is proved secure in the random oracle model based on the hardness of discrete logarithms, and it leads, via the random oracle heuristic and our result above, to concrete efficiency improvements for PCD.

Along the way, we construct a split accumulation scheme for Hadamard products under Pedersen commitments and for a simple polynomial commitment scheme based on Pedersen commitments.

from Additive Polynomial
ents

Fisch¹ Ariel Gabizon³

ndation³ AZTEC Protocol

ntly been in the spotlight for their key role
commit to a polynomial over a finite field
has commitment and evaluation proof size
ient PCS has sublinear proof verification.
struct a SNARK with similar security and
l).

ties to carry out an indefinitely long dis
ay is accompanied by a proof of correct
ation and can even be used to construct
method for constructing PCD required ex
rst demonstrated a new methodology for
gation property of the *Bulletproofs* inner
because it makes non-black-box use of a
. We expand upon this methodology to
y homomorphic polynomial commitment
neither succinct nor efficient. In fact, the
even more general property, namely the
nts into a new succinct commitment that



How to build IVC?

Proof-Carrying Data from Accumulation Schemes

Accumulation Schemes!
Split accumulation schemes!

Folding

Nova: Recursive Zero-Knowledge Arguments from Folding Schemes

Abhiram Kothapalli[†]

[†]Carnegie Mellon University

Srinath Setty^{*}

^{*}Microsoft Research

Ioanna Tzialla[‡]

[‡]New York University

Abstract. We introduce a new approach to realize incrementally verifiable computation (IVC), in which the prover recursively proves the correct execution of incremental computations of the form $y = F^{(\ell)}(x)$, where F is a (potentially non-deterministic) computation, x is the input, y is the output, and $\ell > 0$. Unlike prior approaches to realize IVC, our approach avoids succinct non-interactive arguments of knowledge (SNARKs) entirely and arguments of knowledge in general. Instead, we introduce and employ *folding schemes*, a weaker, simpler, and more efficiently-realizable primitive, which reduces the task of checking two instances in some relation to the task of checking a single instance. We construct a folding scheme for a characterization of NP and show that it implies an IVC scheme with improved efficiency characteristics: (1) the “recursion overhead” (i.e., the number of steps that the prover proves in addition to proving the execution of F) is a constant and it is dominated by two group scalar multiplications expressed as a circuit (this is the smallest recursion overhead in the literature), and (2) the prover’s work at each step is dominated by two multiexponentiations of size $O(|F|)$, providing

Along the way, we construct a split accumulation scheme for Hadamard products under Pedersen commitments and for a simple polynomial commitment scheme based on Pedersen commitments.

uments

a
edu

cholas Spooner
spooner@bu.edu
oston University

from Additive Polynomial
ents

Fisch¹ Ariel Gabizon³
ndation³ AZTEC Protocol

ntly been in the spotlight for their key role
commit to a polynomial over a finite field
has commitment and evaluation proof size
ient PCS has sublinear proof verification.
struct a SNARK with similar security and
l).

ties to carry out an indefinitely long dis
ay is accompanied by a proof of correct
and can even be used to construct
method for constructing PCD required ex
rst demonstrated a new methodology for
gation property of the *Bulletproofs* inner
because it makes non-black-box use of a
. We expand upon this methodology to
y homomorphic polynomial commitment
neither succinct nor efficient. In fact, the
even more general property, namely the
nts into a new succinct commitment that



How to build IVC?

Accumulation Schemes!
Split accumulation schemes!

Folding
Fooooolding !!!

Proof-Carrying Data from Accumulation Schemes

Nova: Recursive Zero-Knowledge Arguments
from Folding Schemes

HyperNova: Recursive arguments for
customizable constraint systems

Abhiram Kothapalli[†]

Srinath Setty^{*}

[†]Carnegie Mellon University

^{*}Microsoft Research

Abstract.

This paper introduces *HyperNova*, a recursive argument for proving computational computations whose steps are expressed with CCS (Setty et al. 2023), a customizable constraint system that simultaneously generalizes R1CS, and AIR without overheads. A distinguishing aspect of HyperNova is that the prover's cost at each step is dominated by a *single* multiplication (MSM) of size equal to the number of variables in the constraint system, which is optimal when using an MSM-based commitment scheme.

To construct HyperNova, we generalize folding schemes (CRYPTO 2022) to allow instances from two (potentially) different NP relations, that share the same structure, to be folded; we refer to this generalization as *multi-folding*.

Sangria: A Folding Scheme for PLONK

Nicolas Mohnblatt
Geometry
nico@geometry.xyz

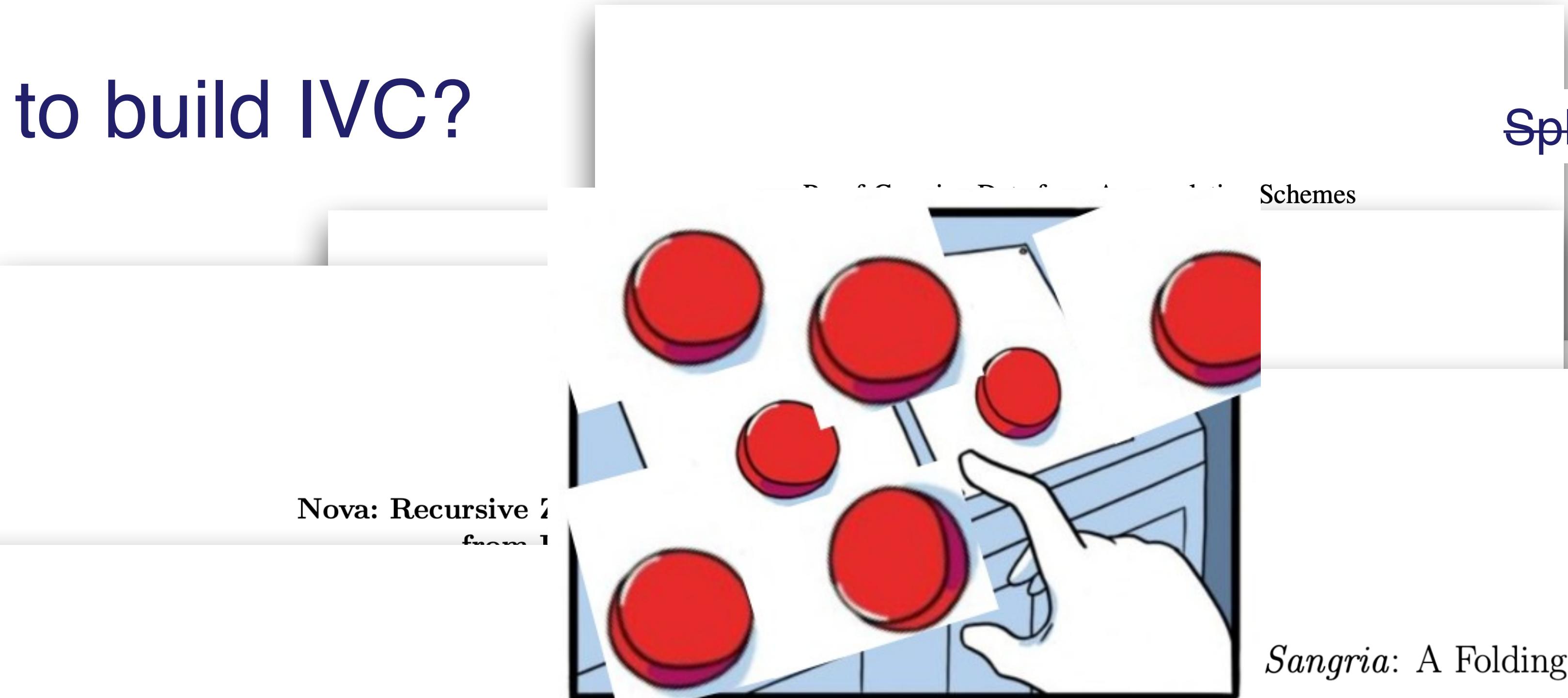
April 17, 2023

PROTOSTAR: Generic Efficient Accumulation/Folding for
Special-sound Protocols

Benedikt Bünz
Stanford University

Binyi Chen
Espresso Systems

How to build IVC?



HyperNova: Recursive argument system for customizable constraint systems

Abhiram Kothapalli[†]

[†]Carnegie Mellon University

Srinath Se

*Microsoft Re

Abstract.

This paper introduces *HyperNova*, a recursive argument system for fundamental computations whose steps are expressed with CCS (Se et al., 2023), a customizable constraint system that simultaneously generalizes, R1CS, and AIR without overheads. A distinguishing aspect of *HyperNova* is that the prover's cost at each step is dominated by a single multiplication (MSM) of size equal to the number of variables in the constraint system, which is optimal when using an MSM-based commitment scheme.

To construct *HyperNova*, we generalize folding schemes (Cohen et al., 2023) to allow instances from two (potentially) different NP relations, that share the same structure, to be folded; we refer to this generalization as *multi-folding*.



JAKE-CLARK.TUMBLR

Accumulation Schemes!
~~Split accumulation schemes!~~

Folding
Fooooolding !!!

Sangria: A Folding Scheme for PLONK

Nicolas Mohnblatt
Geometry
nico@geometry.xyz

April 17, 2023

Efficient Accumulation/Folding for sound Protocols

Benedikt Bünz
Stanford University

Binyi Chen
Espresso Systems

How to build IVC? (continued)

Research is moving fast, how do we choose which scheme to implement?

How should we understand these schemes? Mental model?

How can we compare all these schemes? In theory and in practice



How to build IVC? (continued)

Research is moving fast, how do we choose which scheme to implement?

How should we understand these schemes? Mental model?

How can we compare all these schemes? In theory and in practice

Claim: all these schemes fall under the same framework/pattern, **reduce-and-combine**



Part 1

Finding the Pattern



REDUCE - AND - COMBINE



Halo2 - Accumulation [BGH19, ZcashDoc]



Halo2 - Accumulation [BGH19, ZcashDoc]

I know w such that

$$C_{PLONKish}(x, w) = 1$$



Halo2 - Accumulation [BGH19, ZcashDoc]

I know w such that

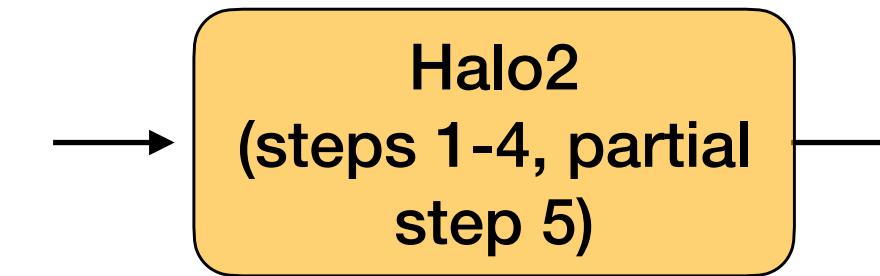
$$C_{PLONKish}(x, w) = 1$$

q _L	q _R	q _O	q _M	q _C	a	b	c
1	0	0	0	0	5		
1	0	0	0	0	10		
-1	0	0	1	0			
-1	0	0	1	0			
1	1	-1	0	0			
0	0	-1	1	0			

rule 1: copy constrains
rule 2: row equation
rule 3: lookup

Halo2 - Accumulation [BGH19, ZcashDoc]

I know w such that
 $C_{PLONKish}(x, w) = 1$



polynomial commitment
opening proof

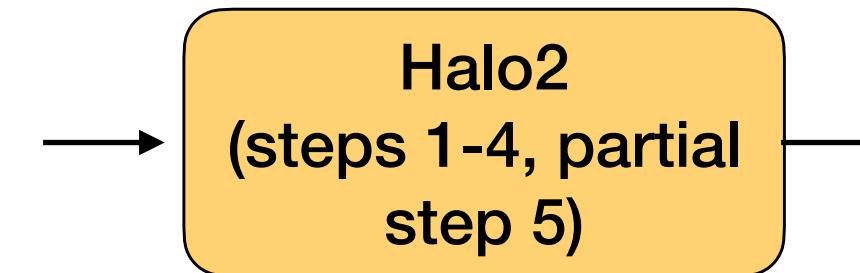
q _L	q _R	q _O	q _M	q _C	a	b	c
1	0	0	0	0	5		
1	0	0	0	0	10		
-1	0	0	1	0			
-1	0	0	1	0			
1	1	-1	0	0			
0	0	-1	1	0			

rule 1: copy constrains
rule 2: row equation
rule 3: lookup



Halo2 - Accumulation [BGH19, ZcashDoc]

I know w such that
 $C_{PLONKish}(x, w) = 1$



polynomial commitment
opening proof

q _L	q _R	q _O	q _M	q _C	a	b	c
1	0	0	0	0	5		
1	0	0	0	0	10		
-1	0	0	1	0			
-1	0	0	1	0			
1	1	-1	0	0			
0	0	-1	1	0			

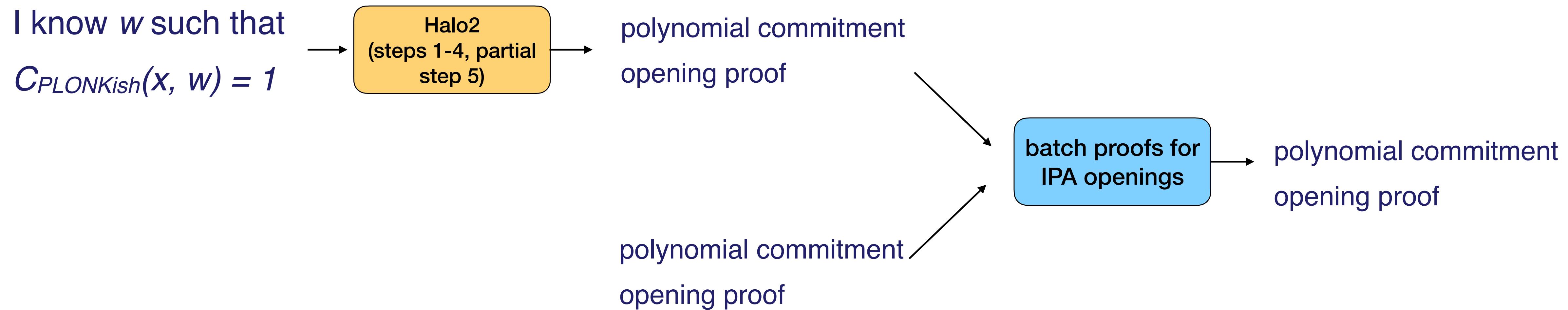
rule 1: copy constrains
rule 2: row equation
rule 3: lookup

The prover will compute a grand product argument, the “quotient” polynomial, etc. Requires some FFTs and polynomial commitments.

The verifier will produce random challenges, linear combination of commitments, check some evaluations and part of the IPA opening



Halo2 - Accumulation [BGH19, ZcashDoc]



[BCLMS21] - Split Accumulation for R1CS



[BCLMS21] - Split Accumulation for R1CS

I know w such that

$$C_{R1CS}(x, w) = 1$$



[BCLMS21] - Split Accumulation for R1CS

I know w such that

$$C_{R1CS}(x, w) = 1$$

Recall R1CS:

$$z = x \parallel w$$

$$Az \circ Bz = Cz$$



[BCLMS21] - Split Accumulation for R1CS

I know w such that

$$C_{R1CS}(x, w) = 1$$

Recall R1CS:
 $z = x \parallel w$
 $Az \circ Bz = Cz$

The almost-trivial NARK*

$$\mathcal{P}(\text{ck}, (A, B, C), x, w)$$

$$z := (x, w) \in \mathbb{F}^N$$

$$z_A := Az \in \mathbb{F}^M$$

$$C_A := \text{Commit}(\text{ck}, z_A) \in \mathbb{G}$$

$$z_B := Bz \in \mathbb{F}^M$$

$$C_B := \text{Commit}(\text{ck}, z_B) \in \mathbb{G}$$

$$z_C := Cz \in \mathbb{F}^M$$

$$C_C := \text{Commit}(\text{ck}, z_C) \in \mathbb{G}$$

$$— C_A, C_B, C_C, w \rightarrow$$

$$\mathcal{V}(\text{ck}, (A, B, C), x)$$

$$z := (x, w)$$

$$z_A := Az \quad C_A \stackrel{?}{=} \text{Commit}(\text{ck}, z_A)$$

$$z_B := Bz \quad C_B \stackrel{?}{=} \text{Commit}(\text{ck}, z_B)$$

$$z_C := Cz \quad C_C \stackrel{?}{=} \text{Commit}(\text{ck}, z_C)$$

$$C_C \stackrel{?}{=} \text{Commit}(\text{ck}, z_A \circ z_B)$$

*the paper extends this basic idea to also provide zero-knowledge

[BCLMS21] - Split Accumulation for R1CS

I know w such that
 $C_{R1CS}(x, w) = 1$



I know openings to the commitments C_A C_B C_c
and
 C_c is a commitment to the Hadamard product of the vectors committed in C_A and C_B



[BCLMS21] - Split Accumulation for R1CS

I know w such that
 $C_{R1CS}(x, w) = 1$

→ NARK
(some of it)

Hadamard product
vector commitment
claims



[BCLMS21] - Split Accumulation for R1CS



Nova [KST22]



R E D U C E - A N D - C O M B I N E



Nova [KST22]

I know v_1 such that

$$C_{RelaxedR1CS}(u_1, v_1) = 1$$

I know v_2 such that

$$C_{RelaxedR1CS}(u_2, v_2) = 1$$

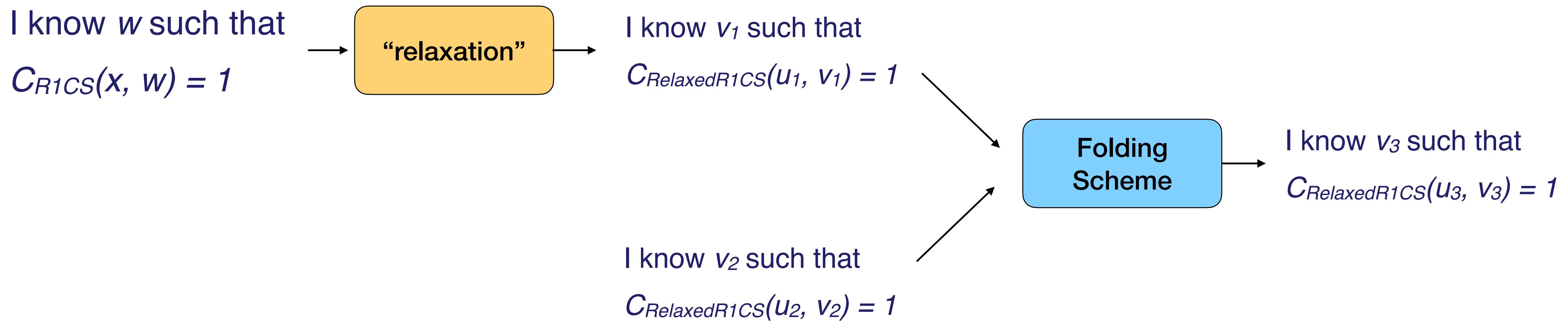
Folding
Scheme

I know v_3 such that

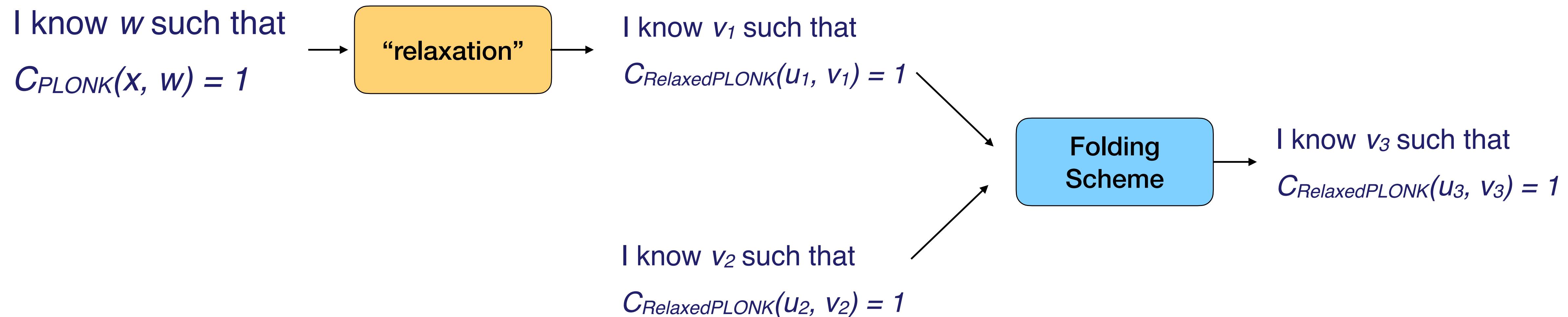
$$C_{RelaxedR1CS}(u_3, v_3) = 1$$



Nova [KST22]



Sangria [Moh23]



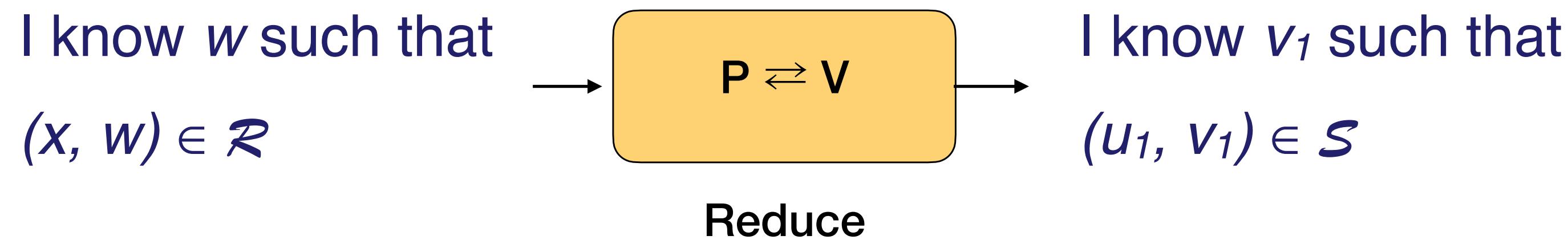
The Reduce-and-Combine Pattern

I know w such that

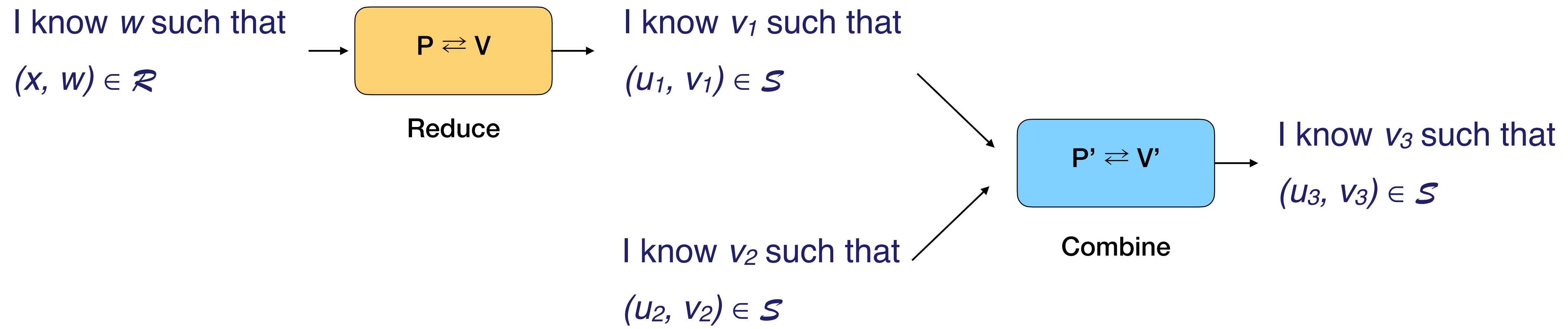
$$(x, w) \in \mathcal{R}$$



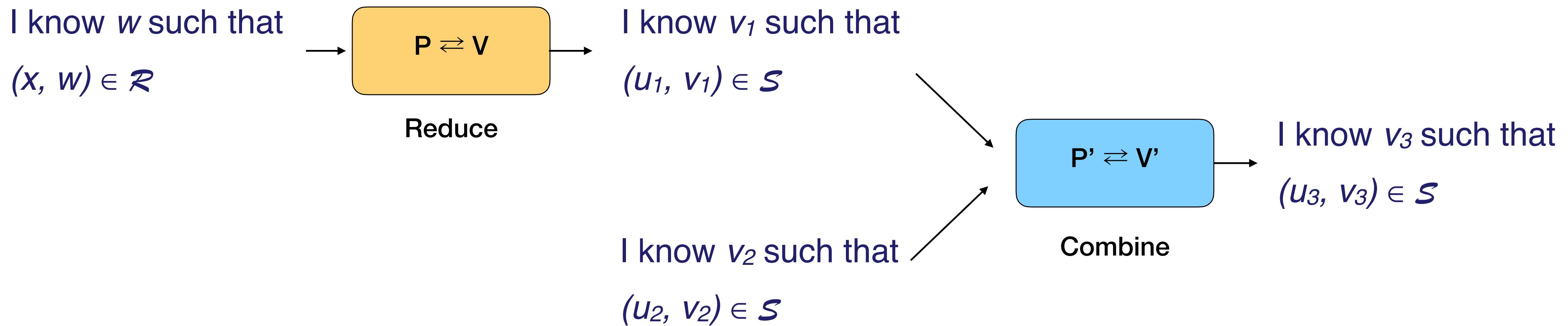
The Reduce-and-Combine Pattern



The Reduce-and-Combine Pattern



The Reduce-and-Combine Pattern

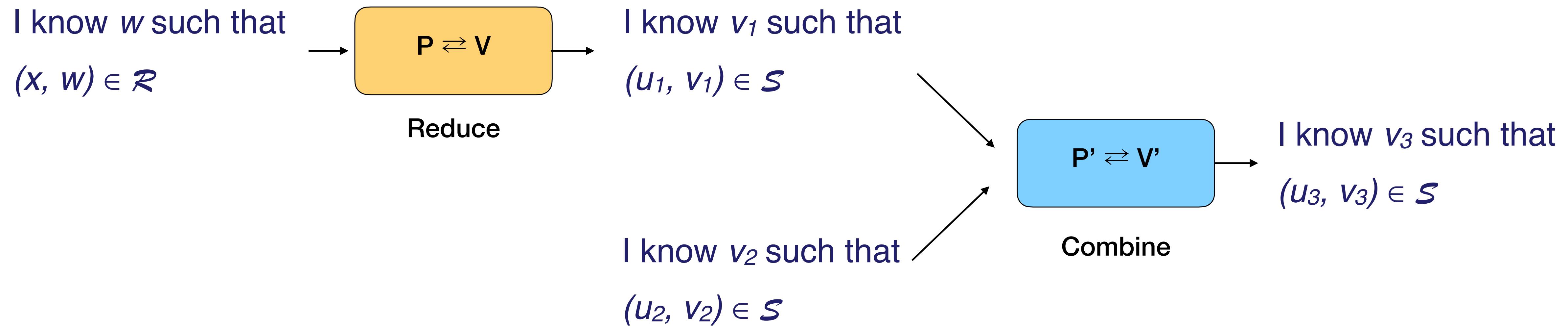


- Choosing \mathcal{R} and \mathcal{S} determines how much work each party does in the “Reduce” and the “Combine” stage
- At each IVC step, the IVC prover will perform the work of (P, P') and evaluate a circuit performing (V, V')

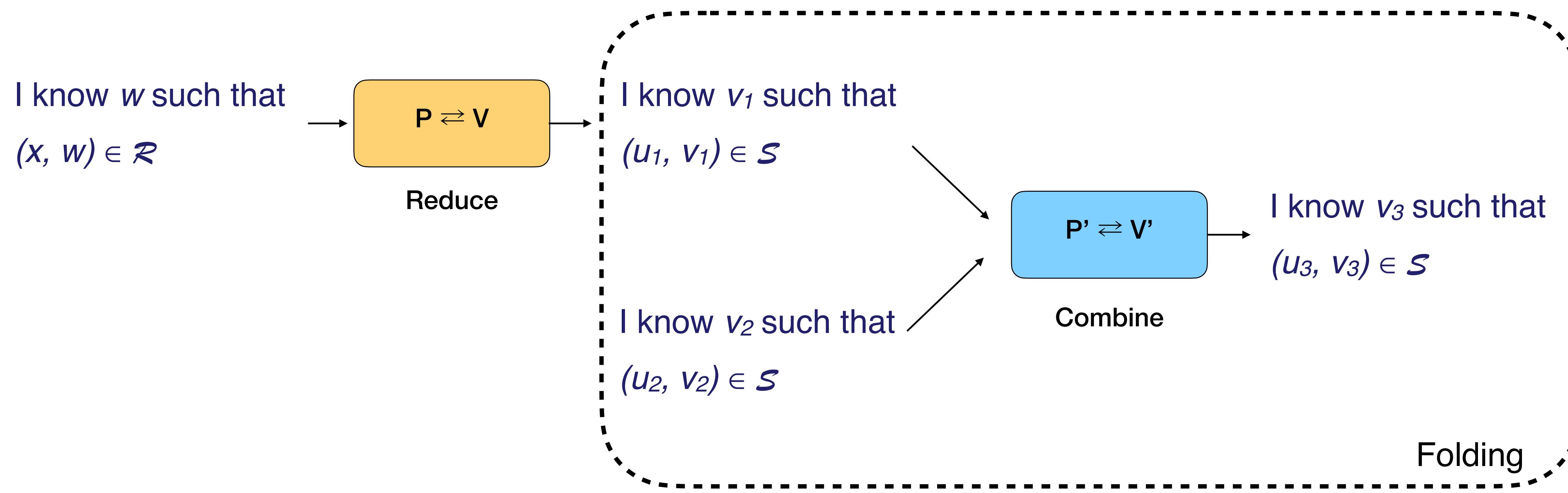
Comparing the examples

Protocol	\mathcal{R}	\mathcal{S}	“Reduce” work	“Combine” work
Halo2-IPA	PLONKish / UltraPLONK	IPA polynomial opening proofs	P: grand product, quotient poly (FFTs and commitments) V: produce challenges, linear comb. of comm., check evaluations and partial IPA opening	P: random linear combination and opening proof V: random linear combination and partial opening proof
BCLSM21	R1CS	Hadamard product vector commitment claims	P: commit to the matrix-vector products V: none	P: commit to the error term V: add commitments with 1 error term
Nova	R1CS	committed relaxed R1CS	P: commit to the witness V: none	P: commit to the error term V: add commitments with 1 error term
Sangria	PLONK	committed relaxed PLONK	P: commit to the witness V: none	P: commit to the error term V: add commitments with 1 error term

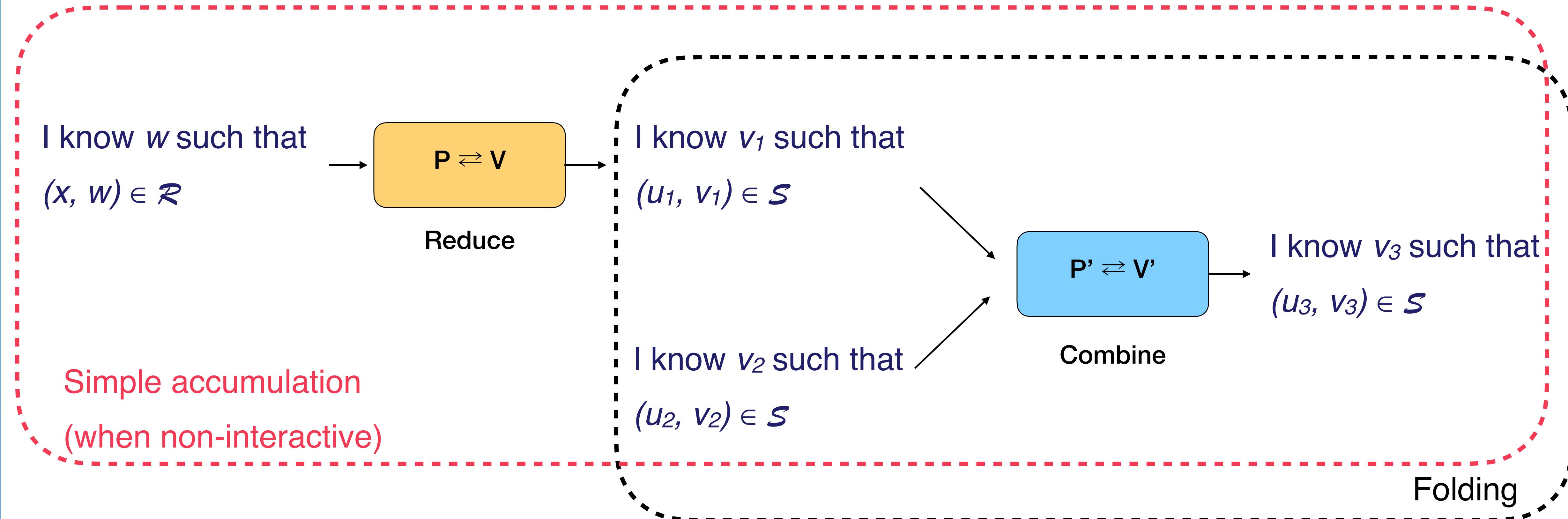
Mapping existing definitions



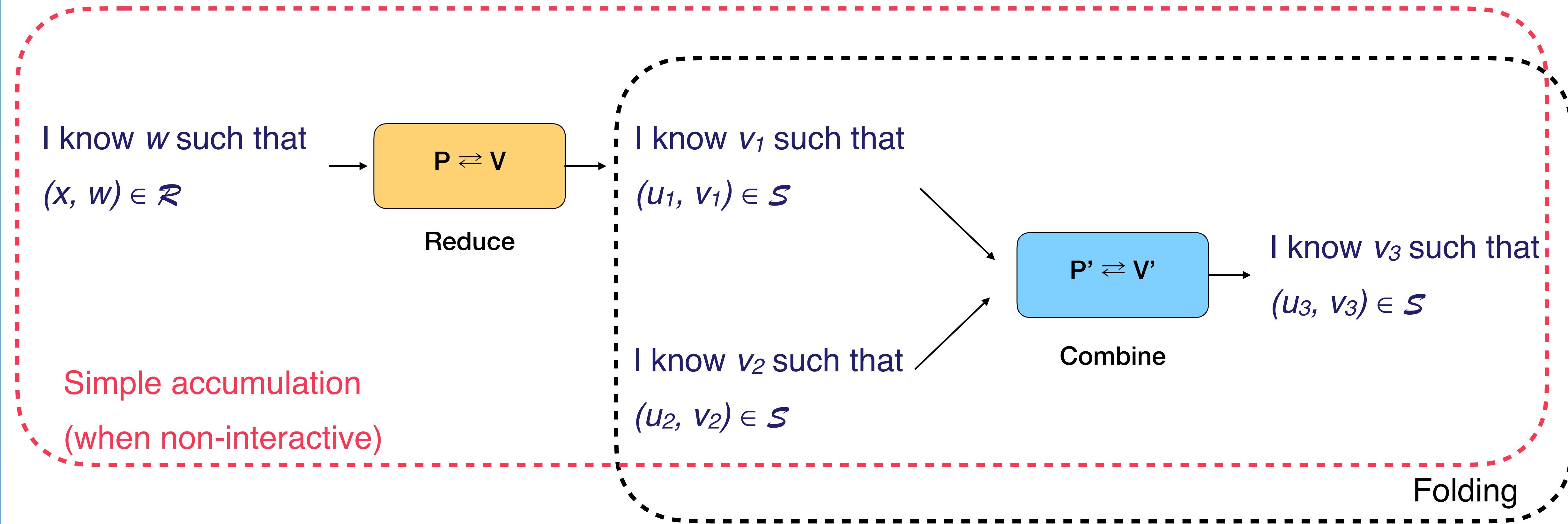
Mapping existing definitions



Mapping existing definitions



Mapping existing definitions



Split accumulation and multi-folding consider the case of a many-to-1 combination

Part 2

Understanding the Latest Constructions



REDUCE - AND - COMBINE

HyperNova [KS23]



REDUCE - AND - COMBINE

HyperNova [KS23] - a small detour via CCS and SuperSpartan [STW23]

$$Az \circ Bz - Cz = 0$$

The diagram shows a mathematical equation involving three matrices: Az , Bz , and Cz . The equation is $Az \circ Bz - Cz = 0$. The matrices are represented by grids of colored squares. Az is a 4x4 grid of light blue squares. Bz is a 2x2 grid of white squares. Cz is a 4x4 grid of light purple squares. The operation \circ between Az and Bz indicates element-wise multiplication. The result of $Az \circ Bz$ is then subtracted from Cz . The final result is a 2x2 grid of zero vectors, represented by a horizontal line with arrows at both ends.

$$\begin{matrix} \text{light blue} & \text{white} & \circ & \text{light blue} & \text{white} & - & \text{light purple} & \text{white} \\ \begin{matrix} \text{light blue} \\ \text{light blue} \\ \text{light blue} \\ \text{light blue} \end{matrix} & \begin{matrix} \text{white} \\ \text{white} \end{matrix} & & \begin{matrix} \text{light blue} \\ \text{light blue} \\ \text{light blue} \\ \text{light blue} \end{matrix} & \begin{matrix} \text{white} \\ \text{white} \end{matrix} & & \begin{matrix} \text{light purple} \\ \text{light purple} \\ \text{light purple} \\ \text{light purple} \end{matrix} & \begin{matrix} \text{white} \\ \text{white} \end{matrix} \\ = & & & & & & & \vec{0} \end{matrix}$$



HyperNova [KS23] - a small detour via CCS and SuperSpartan [STW23]

$$Az \circ Bz - Cz = 0$$

The diagram shows a mathematical expression involving matrices and vectors. On the left, there is a 4x4 matrix with light blue cells. To its right is a vertical vector with four light blue cells. An operation symbol (\circ) is placed between the matrix and the vector. To the right of the vector is another operation symbol ($-$). Following the minus sign is a 4x4 matrix with light purple cells. To its right is a vertical vector with four light purple cells. The entire expression is followed by an equals sign and a horizontal arrow pointing right ($\vec{0}$), indicating that the sum of the two terms is the zero vector.

Spartan [Set20] is a two-layered sumcheck based proof for R1CS



HyperNova [KS23] - a small detour via CCS and SuperSpartan [STW23]

$$Az \circ Bz - Cz = 0$$

The diagram illustrates the equation $Az \circ Bz - Cz = 0$. It shows three components: a 4x4 grid of light blue squares representing matrix A , a vertical column of four light blue squares representing vector z , and a 4x4 grid of light blue squares representing matrix B . An operation symbol \circ is placed between the first two components, indicating element-wise multiplication. A minus sign $-$ is placed before the third component, which is a 4x4 grid of light purple squares representing matrix C . The result of the entire expression is a vertical column of four white squares, representing the zero vector $\vec{0}$.

Spartan [Set20] is a two-layered sumcheck based proof for R1CS

Sumcheck 1 (zero check):

$$a \circ b - c = 0$$

The diagram illustrates the sumcheck 1 (zero check) for the equation $a \circ b - c = 0$. It shows three components: a vertical column of four light blue squares representing vector a , a vertical column of four light blue squares representing vector b , and a vertical column of four light purple squares representing vector c . An operation symbol \circ is placed between the first two components, indicating element-wise multiplication. A minus sign $-$ is placed before the third component. The result of the entire expression is a vertical column of four white squares, representing the zero vector $\vec{0}$.



HyperNova [KS23] - a small detour via CCS and SuperSpartan [STW23]

$$Az \circ Bz - Cz = 0$$

A diagram illustrating a linear combination of matrix-vector products. It shows three terms: a light blue square matrix multiplied by a white vector, followed by a dot product symbol (\circ), another light blue square matrix multiplied by a white vector, followed by a minus sign ($-$), and finally a purple square matrix multiplied by a white vector. The result is set equal to a zero vector ($\vec{0}$).

Spartan [Set20] is a two-layered sumcheck based proof for R1CS

Sumcheck 1 (zero check):

$$a \circ b - c = 0$$

A diagram illustrating a linear combination of vector products. It shows two terms: a light blue vector multiplied by a light blue vector, followed by a dot product symbol (\circ), and a purple vector multiplied by a purple vector, followed by a minus sign ($-$). The result is set equal to a zero vector ($\vec{0}$).

Sumcheck(s) 2 (“linchecks”, i.e. matrix-vector products):

$$Az = a, Bz = b, Cz = c$$

A diagram illustrating matrix-vector multiplication. It shows three terms: a light blue square matrix multiplied by a white vector, followed by an equals sign ($=$); another light blue square matrix multiplied by a white vector, followed by an equals sign ($=$); and finally a purple square matrix multiplied by a white vector, followed by an equals sign ($=$).



HyperNova [KS23] - a small detour via CCS and SuperSpartan [STW23]

$$M_1Z \circ M_2Z \circ M_2Z + M_3Z + M_4Z \circ M_2Z - 3 M_3Z \circ M_5Z = 0$$

$$\begin{matrix} \text{cyan grid} & \text{white bar} & \text{cyan grid} & \text{white bar} \\ \text{white bar} & \circ & \text{cyan grid} & \circ & \text{cyan grid} & \text{white bar} \\ & & & & & \\ & & & & & \\ & & & & & \end{matrix} + \begin{matrix} \text{orange grid} & \text{white bar} \\ \text{orange grid} & \circ \\ \text{white bar} & \end{matrix} + \begin{matrix} \text{purple grid} & \text{white bar} \\ \text{purple grid} & \circ \\ \text{white bar} & \end{matrix} - 3 \begin{matrix} \text{orange grid} & \text{white bar} \\ \text{orange grid} & \circ \\ \text{white bar} & \end{matrix} + \begin{matrix} \text{red grid} & \text{white bar} \\ \text{red grid} & \circ \\ \text{white bar} & \end{matrix} = \vec{0}$$

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



HyperNova [KS23] - a small detour via CCS and SuperSpartan [STW23]

$$M_1Z \circ M_2Z \circ M_2Z + M_3Z + M_4Z \circ M_2Z - 3 M_3Z \circ M_5Z = 0$$

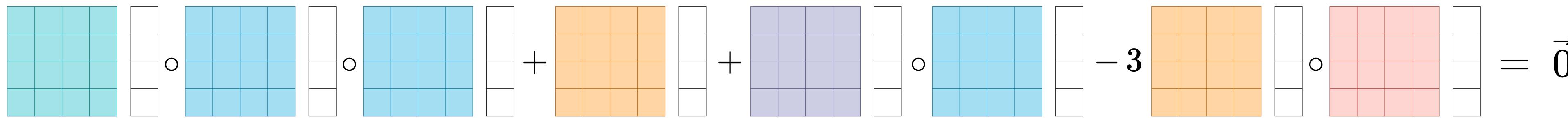
The diagram shows a mathematical expression involving matrices and vectors. It consists of several terms separated by plus signs (+) and minus signs (-). Each term is composed of a matrix (represented by a grid of colored squares) followed by a vector (represented by a vertical stack of colored squares). The first term is $M_1Z \circ M_2Z \circ M_2Z$, which is equivalent to $M_1Z \circ (M_2Z \circ M_2Z)$. The second term is M_3Z . The third term is $M_4Z \circ M_2Z$. The fourth term is $-3 M_3Z \circ M_5Z$. The final result of the entire expression is $\vec{0}$.

SuperSpartan is a two-layered sumcheck based proof for CCS



HyperNova [KS23] - a small detour via CCS and SuperSpartan [STW23]

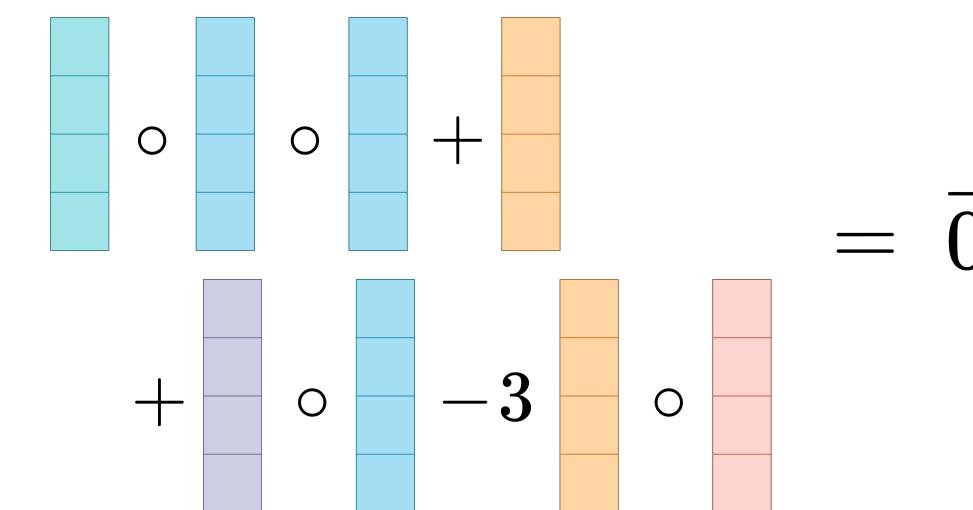
$$M_1Z \circ M_2Z \circ M_2Z + M_3Z + M_4Z \circ M_2Z - 3 M_3Z \circ M_5Z = 0$$


$$\begin{matrix} \text{cyan grid} & \circ & \text{cyan grid} & \circ & \text{cyan grid} \\ \text{cyan grid} & \circ & \text{cyan grid} & \circ & \text{cyan grid} \end{matrix} + \begin{matrix} \text{orange grid} \\ \text{orange grid} \end{matrix} + \begin{matrix} \text{purple grid} \\ \text{purple grid} \end{matrix} \circ \begin{matrix} \text{cyan grid} \\ \text{cyan grid} \end{matrix} - 3 \begin{matrix} \text{orange grid} \\ \text{orange grid} \end{matrix} \circ \begin{matrix} \text{red grid} \\ \text{red grid} \end{matrix} = \vec{0}$$

SuperSpartan is a two-layered sumcheck based proof for CCS

Sumcheck 1 (zero check):

$$m_1 \circ m_2 \circ m_2 + m_3 + m_4 \circ m_2 - 3 m_3 \circ m_5 = 0$$


$$\begin{matrix} \text{cyan} \\ \text{cyan} \end{matrix} \circ \begin{matrix} \text{cyan} \\ \text{cyan} \end{matrix} \circ \begin{matrix} \text{cyan} \\ \text{cyan} \end{matrix} + \begin{matrix} \text{orange} \\ \text{orange} \end{matrix} = \vec{0}$$
$$+ \begin{matrix} \text{purple} \\ \text{purple} \end{matrix} \circ \begin{matrix} \text{cyan} \\ \text{cyan} \end{matrix} - 3 \begin{matrix} \text{orange} \\ \text{orange} \end{matrix} \circ \begin{matrix} \text{red} \\ \text{red} \end{matrix}$$



HyperNova [KS23] - a small detour via CCS and SuperSpartan [STW23]

$$M_1Z \circ M_2Z \circ M_2Z + M_3Z + M_4Z \circ M_2Z - 3 M_3Z \circ M_5Z = 0$$

SuperSpartan is a two-layered sumcheck based proof for CCS

Sumcheck 1 (zero check):

Sumcheck(s) 2 (“linchecks”, i.e. matrix-vector products):

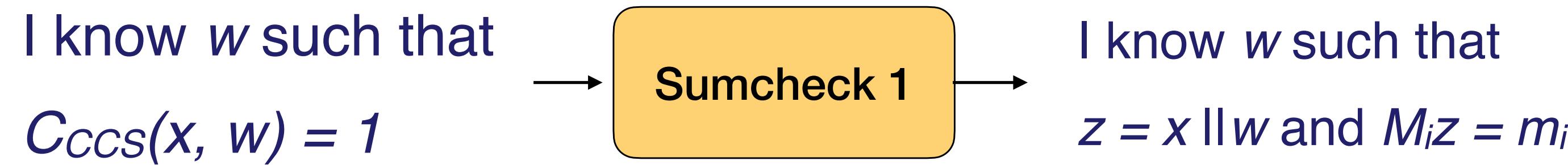
$$m_1 \circ m_2 \circ m_2 + m_3 + m_4 \circ m_2 - 3 m_3 \circ m_5 = 0$$

$$M_i Z = m_i$$

The diagram illustrates the equation $M_i Z = m_i$ through two rows of seven columns each. The first row contains a 4x4 teal grid, a 2x1 white column, a 2x4 teal column, a 4x4 orange grid, a 2x1 white column, a 2x4 orange column, and a 4x4 red grid. The second row contains a 4x4 blue grid, a 2x1 white column, a 2x4 blue column, a 4x4 purple grid, a 2x1 white column, a 2x4 purple column, and a 4x4 red grid. Each column is followed by a double equals sign ($=$) indicating the result of the multiplication.



HyperNova [KS23]



HyperNova [KS23]

I know w such that
 $C_{CCS}(x, w) = 1$



I know w such that
 $z = x || w$ and $M_i z = m_i$

This is linearised CCS! (almost)



HyperNova [KS23]

I know w such that
 $C_{CCS}(x, w) = 1$



I know w such that
 $z = x \parallel w$ and $M_i z = m_i$

This is linearised CCS! (almost)



HyperNova [KS23]

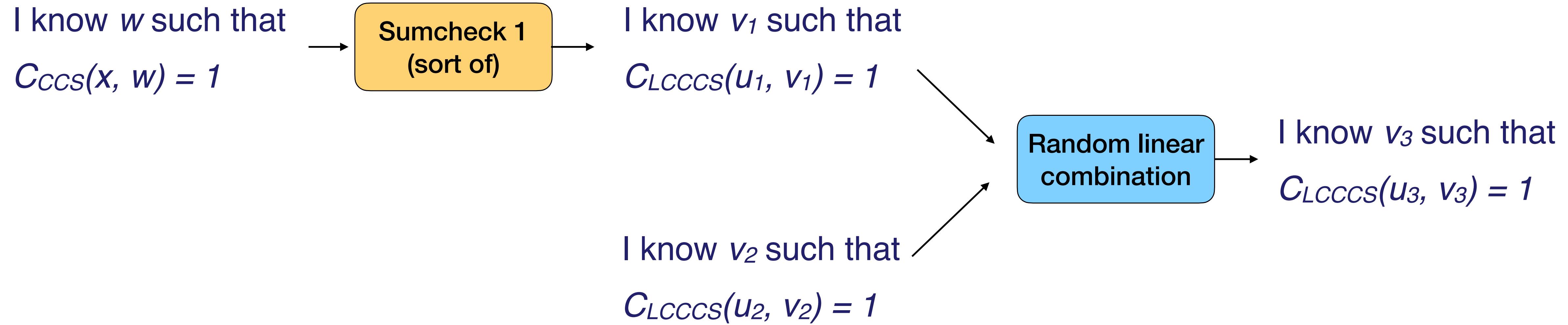
I know w such that
 $C_{CCCS}(x, w) = 1$

→ Sumcheck 1
(sort of)

→ I know v_1 such that
 $C_{LCCCS}(u_1, v_1) = 1$



HyperNova [KS23]



ProtoStar [BC23]

I know w such that

$$f(x, w) = 0$$



ProtoStar [BC23]

I know w such that
 $f(x, w) = 0$



I know w , all the messages m_i and challenges r_i such that:
 $C_i = \text{Comm}(m_i)$
 $\text{relaxed_verif}(x, m_i, r_i, \mu) = e$



ProtoStar [BC23]

I know w such that
 $f(x, w) = 0$



I know w , all the messages m_i and challenges r_i such that:
 $C_i = \text{Comm}(m_i)$
 $\text{relaxed_verif}(x, m_i, r_i, \mu) = e$



I know w , all the messages m_i and challenges r_i such that:
 $C_i = \text{Comm}(m_i)$
 $\text{relaxed_verif}(x, m_i, r_i, \mu) = e$

I know w , all the messages m_i and challenges r_i such that:
 $C_i = \text{Comm}(m_i)$
 $\text{relaxed_verif}(x, m_i, r_i, \mu) = e$

Comparing the latest schemes

Protocol	\mathcal{R}	\mathcal{S}	“Reduce” work	“Combine” work
Nova	R1CS	committed relaxed R1CS	P: commit to the witness V: none	P: commit to the error term V: add commitments with 1 error term
HyperNova	CCS	linearised committed CCS	P: commits to the witness, then P and V run the sumcheck protocol	P: random linear combination V: random linear combination
ProtoStar	any relation with algebraic verifier	commitments to all the messages and compressed verifier check	P: commit to each message V: produce random challenges	P: compute the compressed cross terms V: add commitments and compressed cross terms

Conclusion

- My mental model for folding schemes: **reduce-and-combine**
- Seems there is a trade-off between how much work we do in reduction and how easy it is to fold the resulting relation
- Language suggestion: let's say folding for all protocol in this class, easier on the ear than split accumulation or multi-folding



References

- [BC23] Bünz, Benedikt, and Binyi Chen. "Protostar: Generic efficient accumulation/folding for special sound protocols." *Cryptology ePrint Archive* (2023).
- [BCLMS21] Bünz, B., Chiesa, A., Lin, W., Mishra, P., & Spooner, N. (2021). Proof-carrying data without succinct arguments. In *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part I 41* (pp. 681-710). Springer International Publishing.
- [BCMS20] Bünz, B., Chiesa, A., Mishra, P., & Spooner, N. (2020). Proof-carrying data from accumulation schemes. *Cryptology ePrint Archive*.
- [BGH19] Bowe, S., Grigg, J., & Hopwood, D. (2019). Recursive proof composition without a trusted setup. *Cryptology ePrint Archive*.
- [KS23] Kothapalli, Abhiram, and Srinath Setty. "HyperNova: Recursive arguments for customizable constraint systems." *Cryptology ePrint Archive* (2023).
- [KST22] Kothapalli, Abhiram, Srinath Setty, and Ioanna Tzialla. "Nova: Recursive zero-knowledge arguments from folding schemes." *Annual International Cryptology Conference*. Cham: Springer Nature Switzerland, 2022.
- [Moh23] Mohnblatt, Nicolas. "Sangria: a folding scheme for PLONK". *Github*. https://github.com/geometryresearch/technical_notes/blob/main/sangria_folding_plonk.pdf
- [Set20] Setty, Srinath. "Spartan: Efficient and general-purpose zkSNARKs without trusted setup." *Annual International Cryptology Conference*. Cham: Springer International Publishing, 2020.
- [STW23] Setty, Srinath, Justin Thaler, and Riad Wahby. "Customizable constraint systems for succinct arguments." *Cryptology ePrint Archive* (2023).
- [ZcashDoc] Zcash. "halo2 Documentation". <https://zcash.github.io/halo2/index.html>