Nova

Recursive Zero-Knowledge Arguments from Folding Schemes

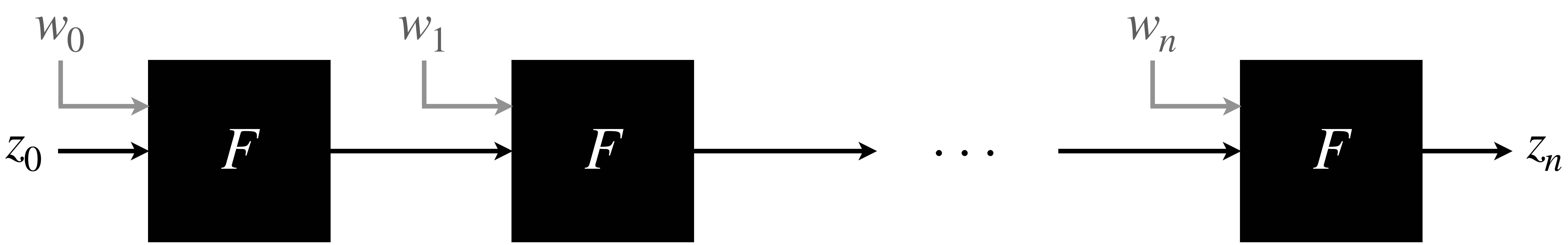
Abhiram Kothapalli (CMU), Srinath Setty (MSR), Ioanna Tzialla (NYU)

eprint.iacr.org/2021/370

github.com/Microsoft/Nova

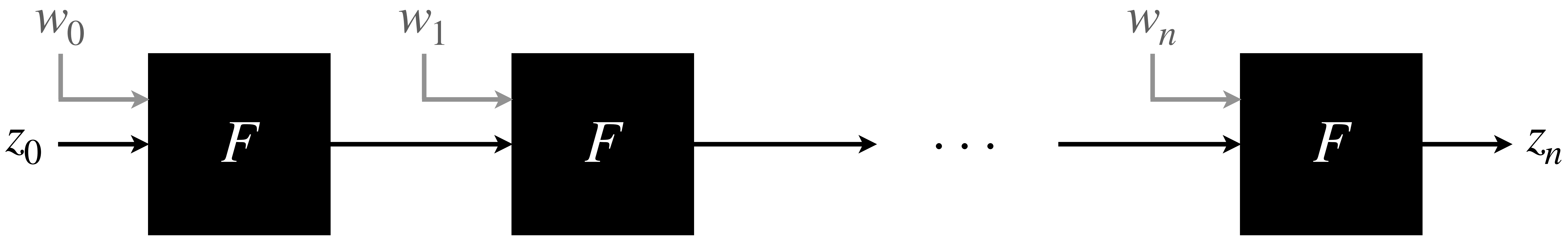
Goal: Practical zkSNARKs for Recursive Computation

Prove that (non-deterministic) function F applied n times to initial input z_0 results in z_n



Goal: Practical zkSNARKs for Recursive Computation

Prove that (non-deterministic) function F applied n times to initial input z_0 results in z_n

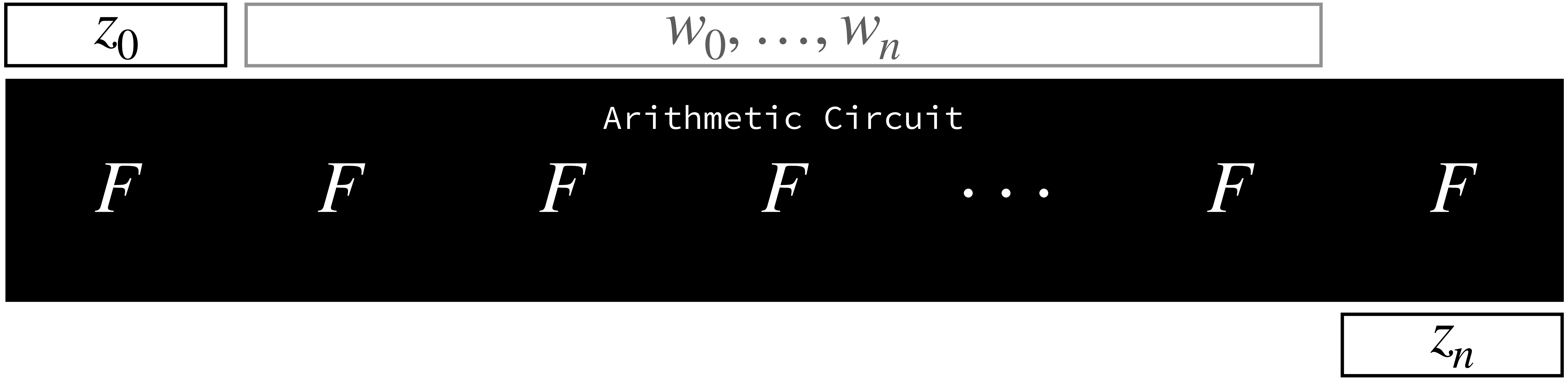


Applications

- Verifiable Delay Function: Let F be a delay function [BBBF19]
- ZK Virtual Machines: Let F be a step of the VM [BCTV14,GPR21]
- ZK-rollups: Let F validate new blockchain transactions

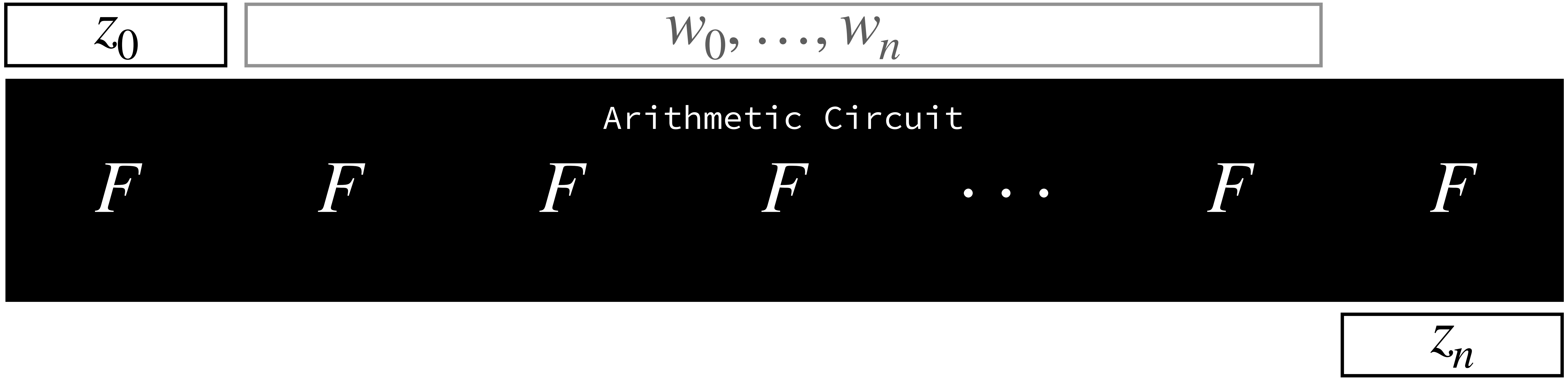
Naive Approach

Use a SNARK to monolithically prove the unrolled statement



Naive Approach

Use a SNARK to monolithically prove the unrolled statement

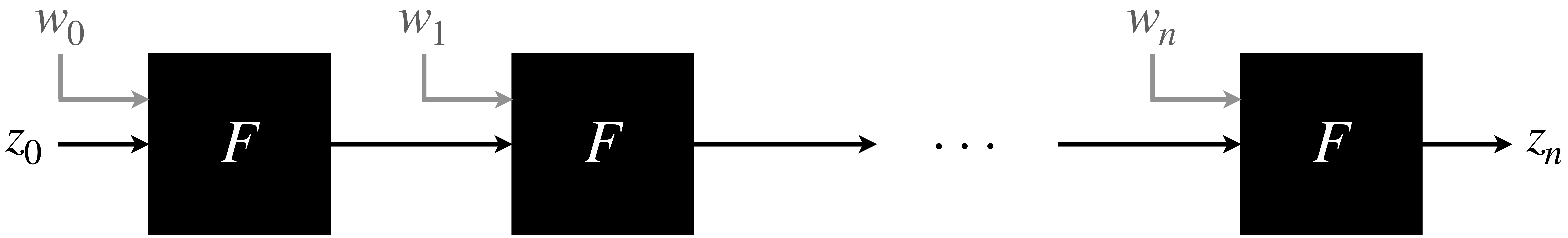


Drawbacks

- Fixes n ahead of time
- $O(n \cdot |F|)$ prover memory and verifier preprocessing time
- Verifier time may depend on n

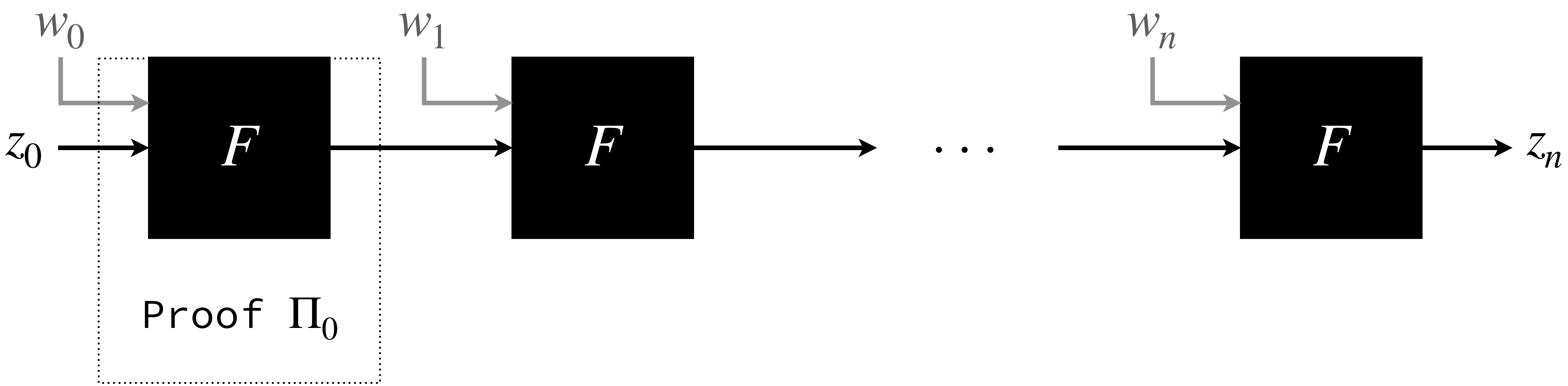
Approach: Incrementally Verifiable Computation (IVC) [Val08]

Incrementally update a proof of i applications to a proof of $i + 1$ applications with the same size



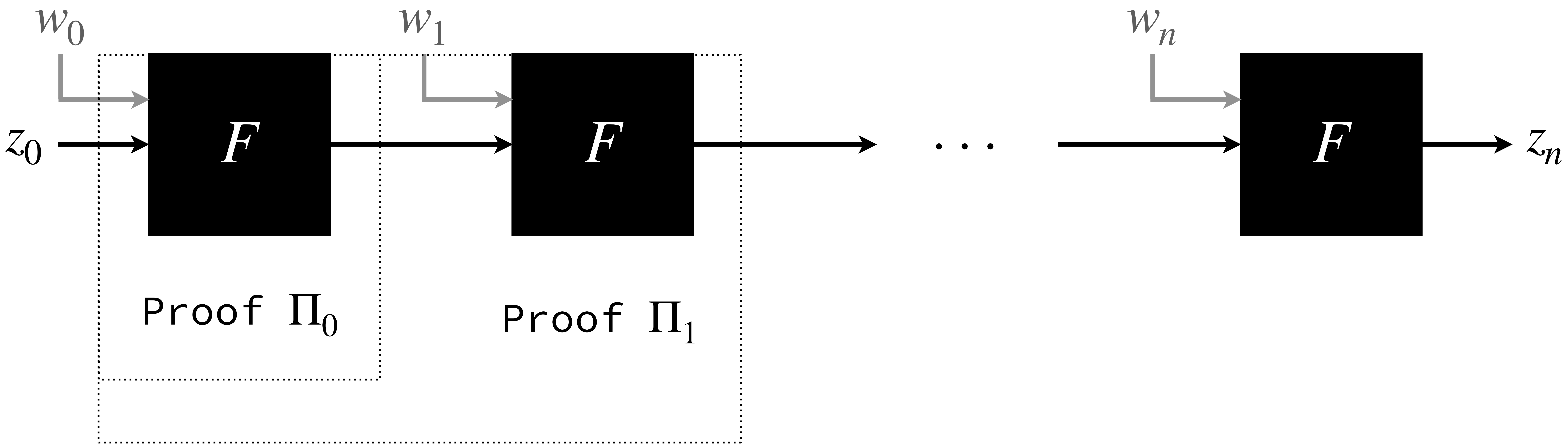
Approach: Incrementally Verifiable Computation (IVC) [Val08]

Incrementally update a proof of i applications to a proof of $i+1$ applications with the same size



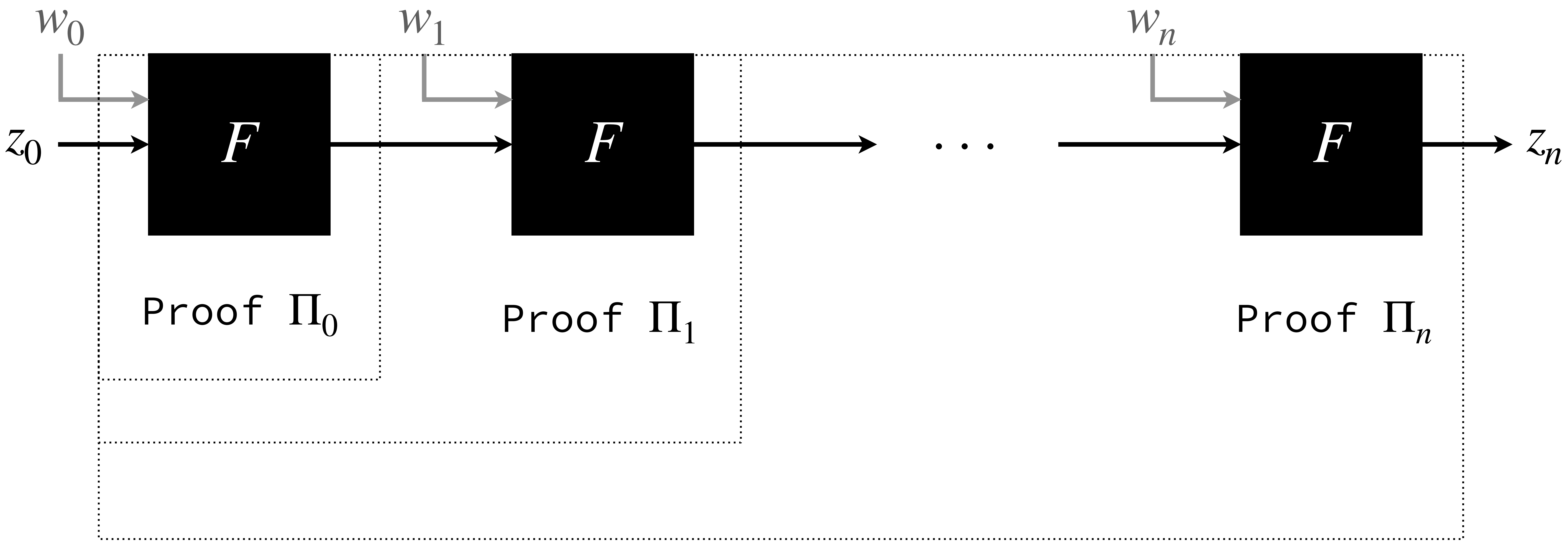
Approach: Incrementally Verifiable Computation (IVC) [Val08]

Incrementally update a proof of i applications to a proof of $i + 1$ applications with the same size

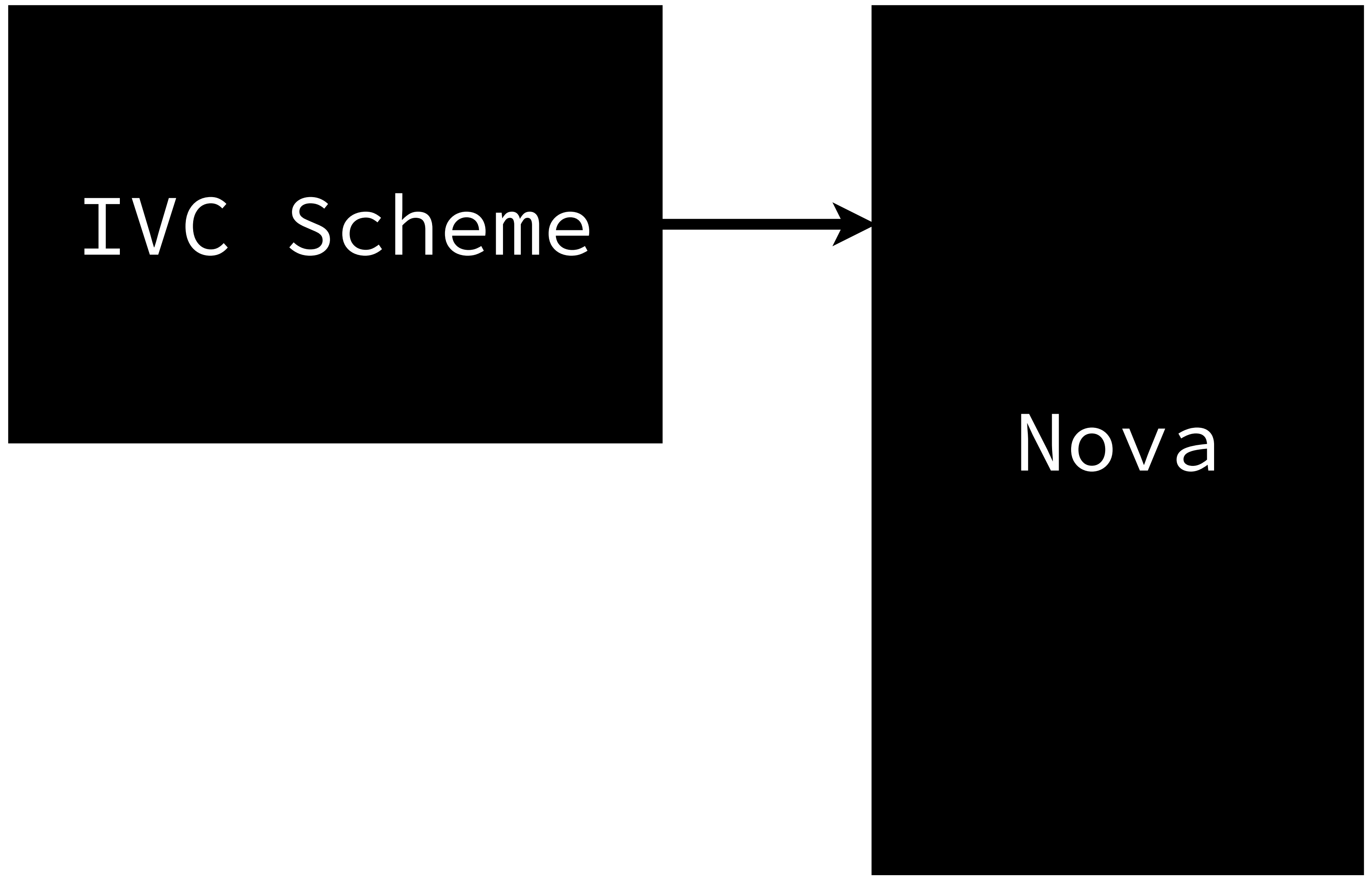


Approach: Incrementally Verifiable Computation (IVC) [Val08]

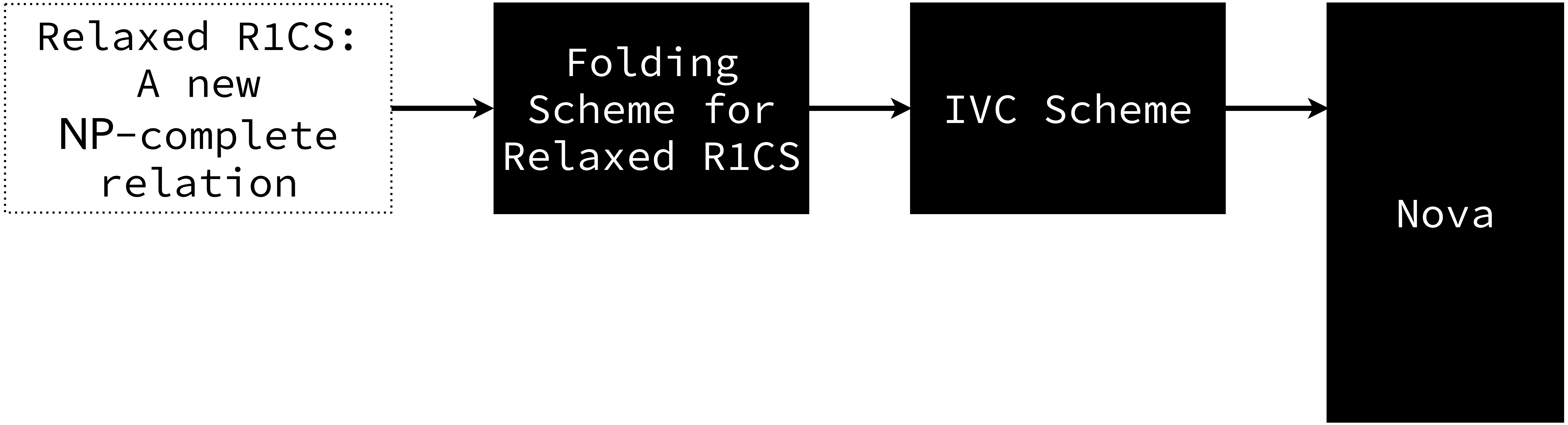
Incrementally update a proof of i applications to a proof of $i+1$ applications with the same size



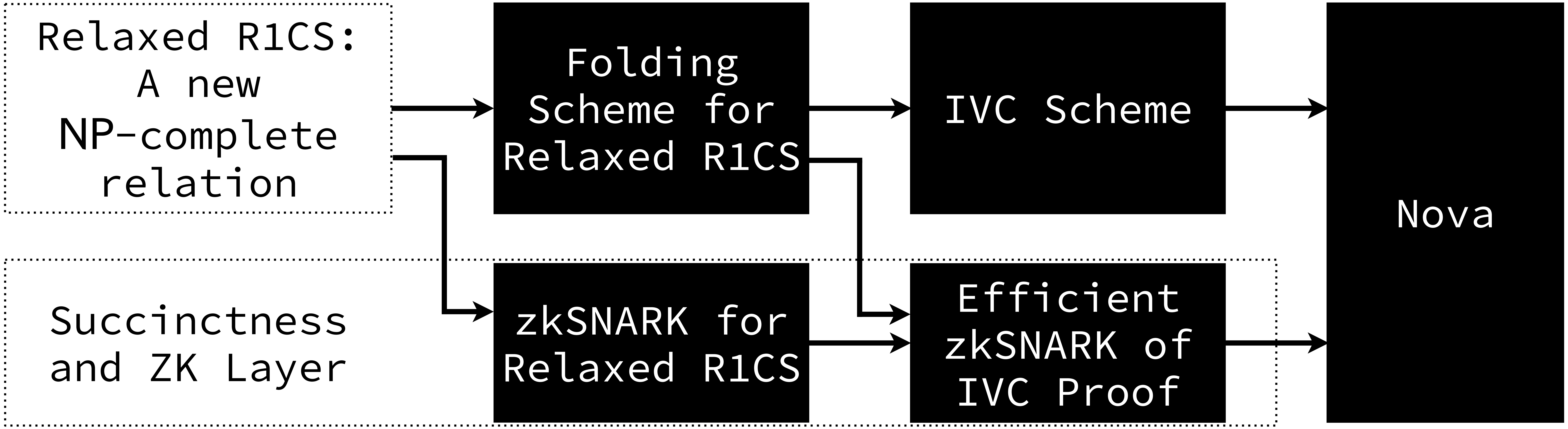
Nova: A Built zkSNARK for Recursive Computation



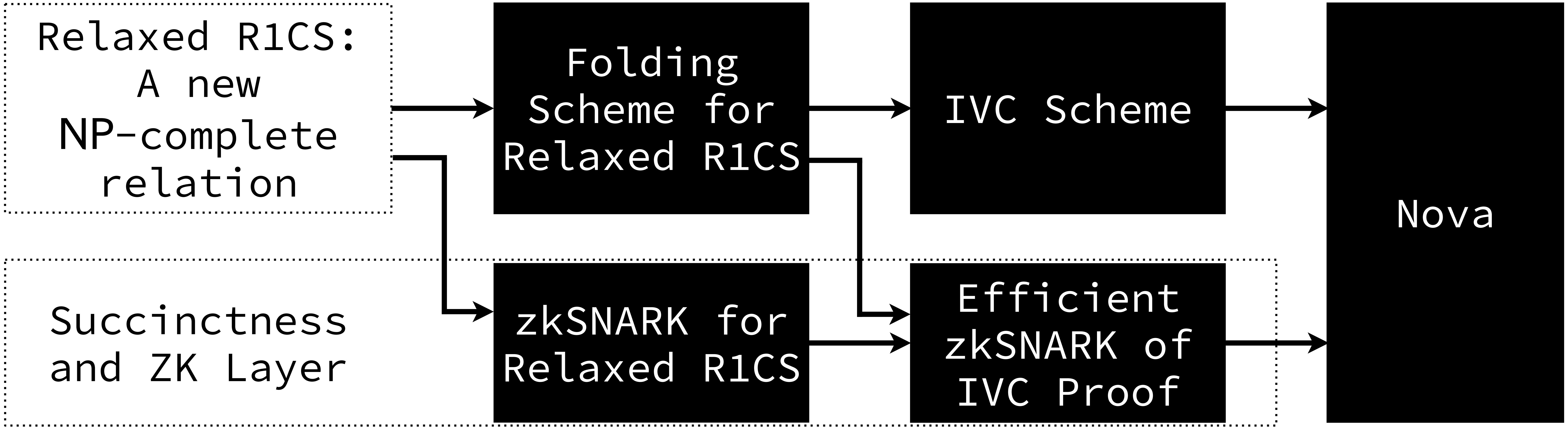
Nova: A Built zkSNARK for Recursive Computation



Nova: A Built zkSNARK for Recursive Computation



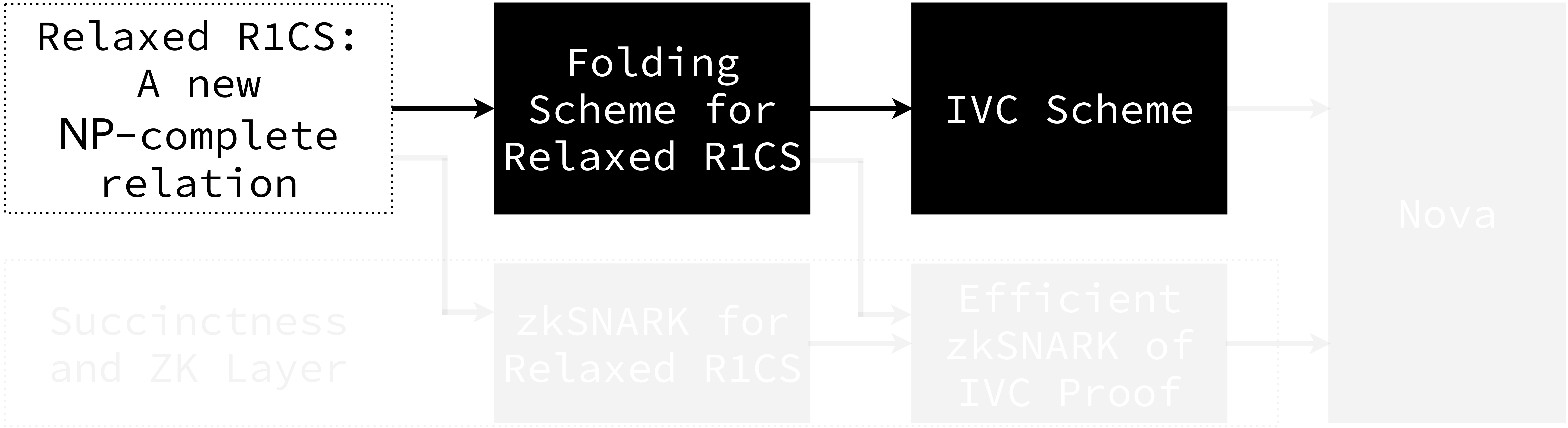
Nova: A Built zkSNARK for Recursive Computation



Implementation and Evaluation

- Implemented in Rust [6000 LOC][github.com/Microsoft/Nova]
- Smallest per step prover time (roughly two $|F| + \mathsf{c}$ size multi-exps)[1M gates: 1.1s]
- Smallest recursion overhead c (roughly two scalar multiplications) [20k gates]
- $O(\log |F|)$ size compressed proofs [1M gates: 8 KB]

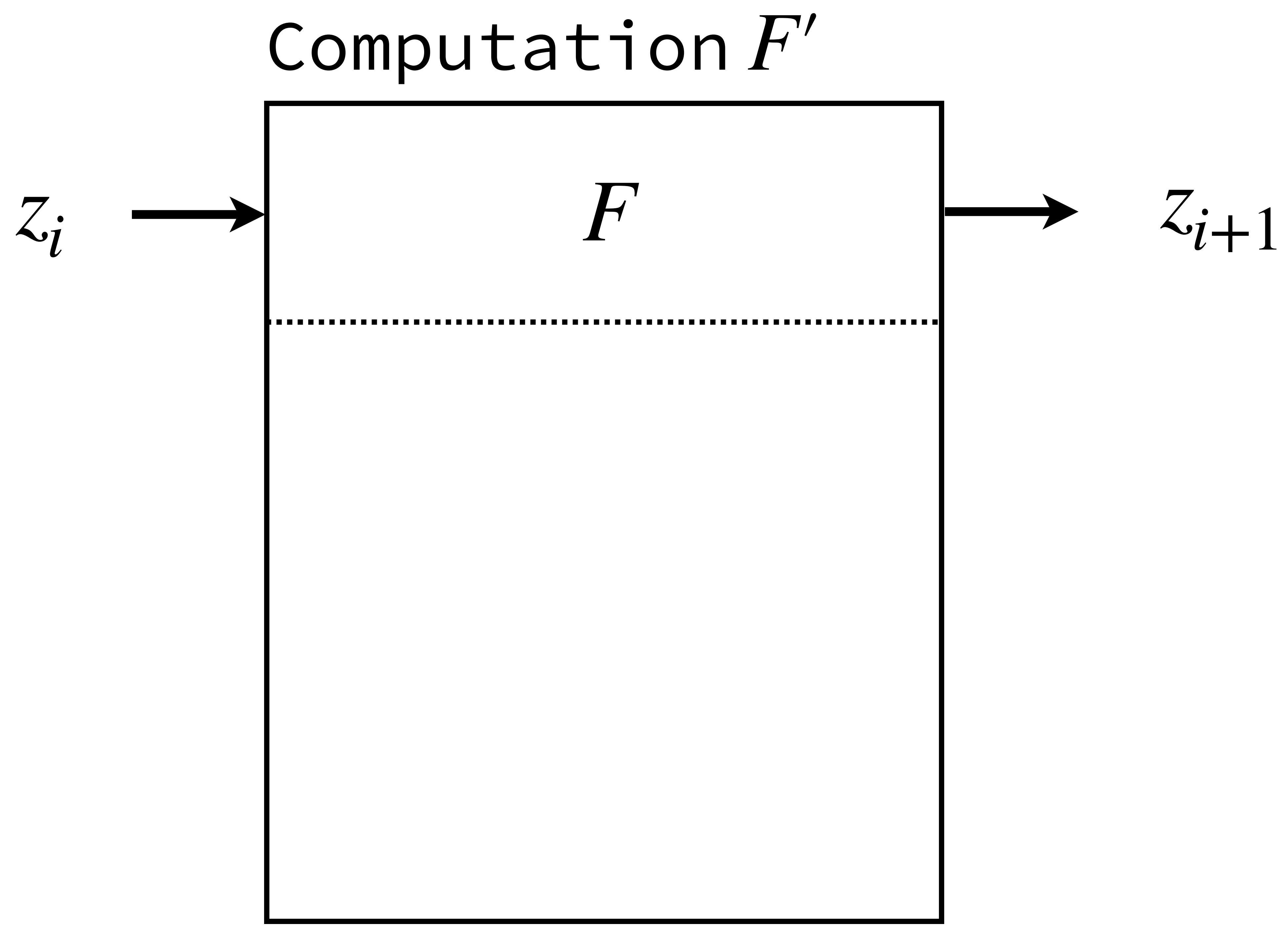
Presented in this Talk



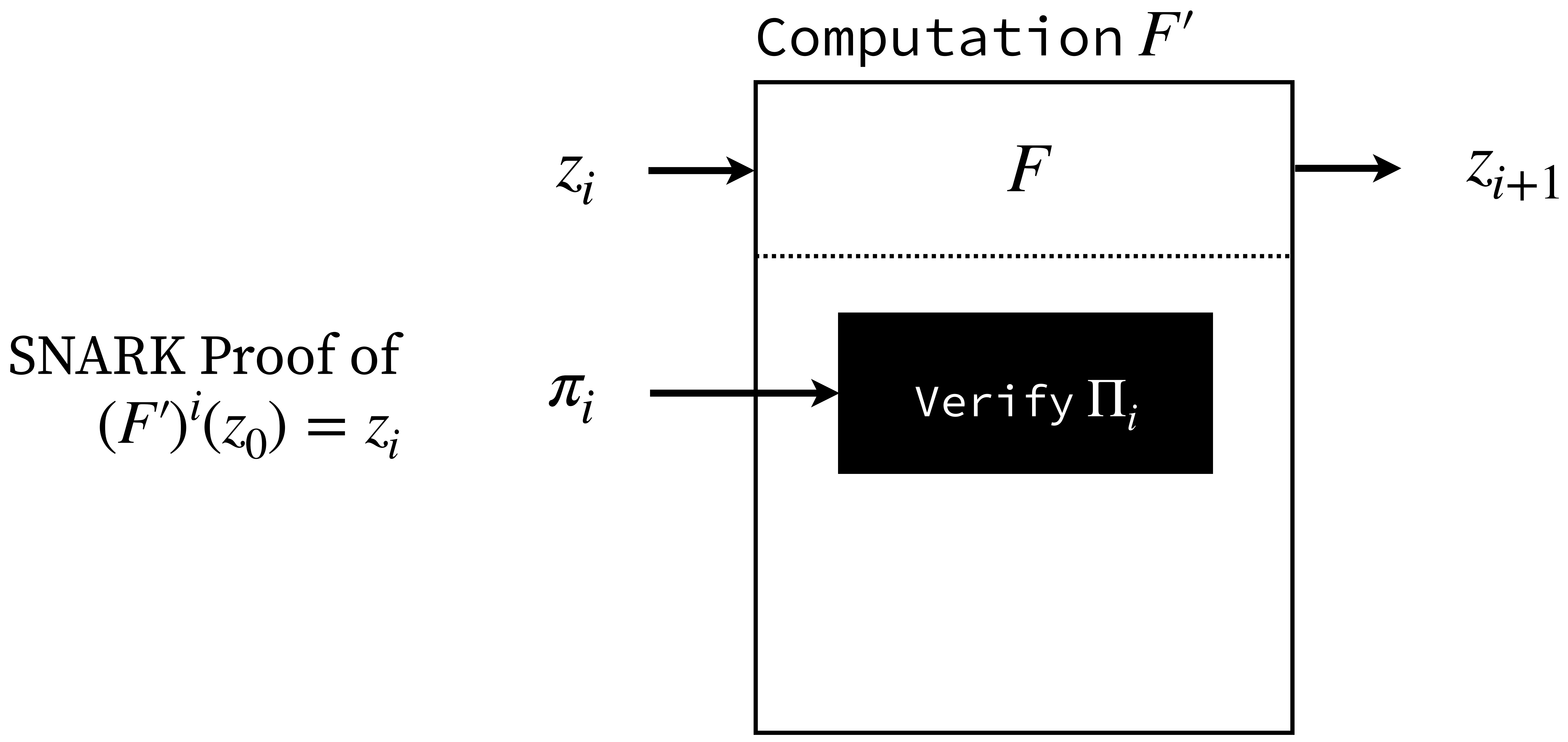
Outline

1. Show that a folding scheme for NP implies IVC
2. Develop a folding scheme for Relaxed R1CS

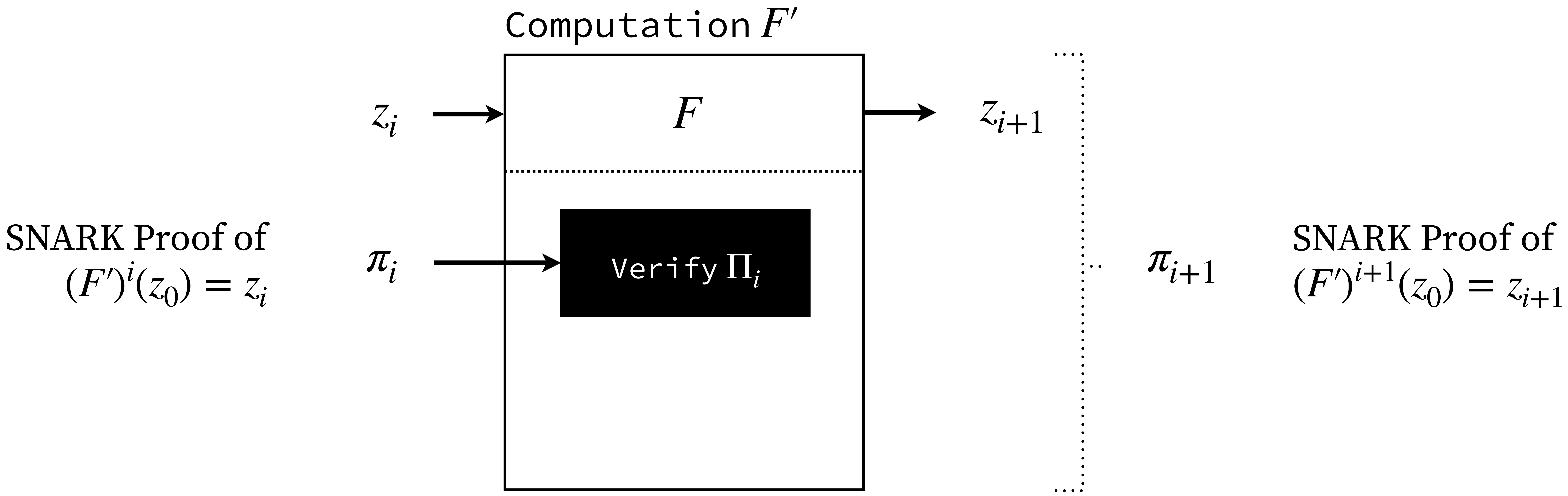
Valiant's Original IVC Approach [Val08,BCTV14]



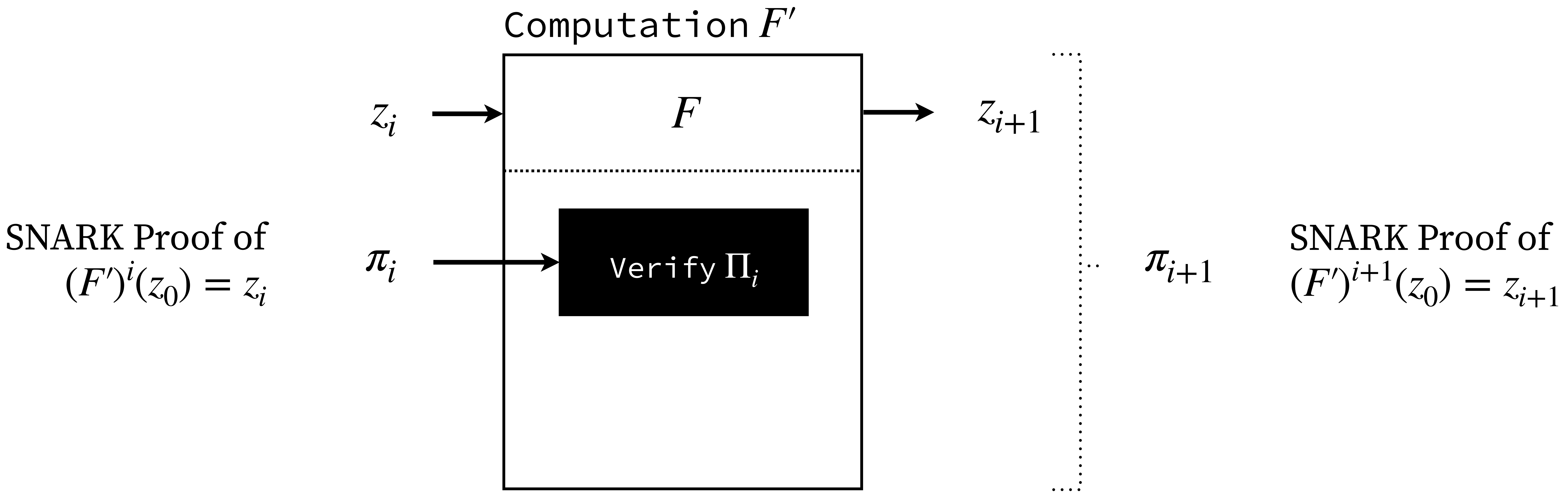
Valiant's Original IVC Approach [Val08,BCTV14]



Valiant's Original IVC Approach [Val08,BCTV14]



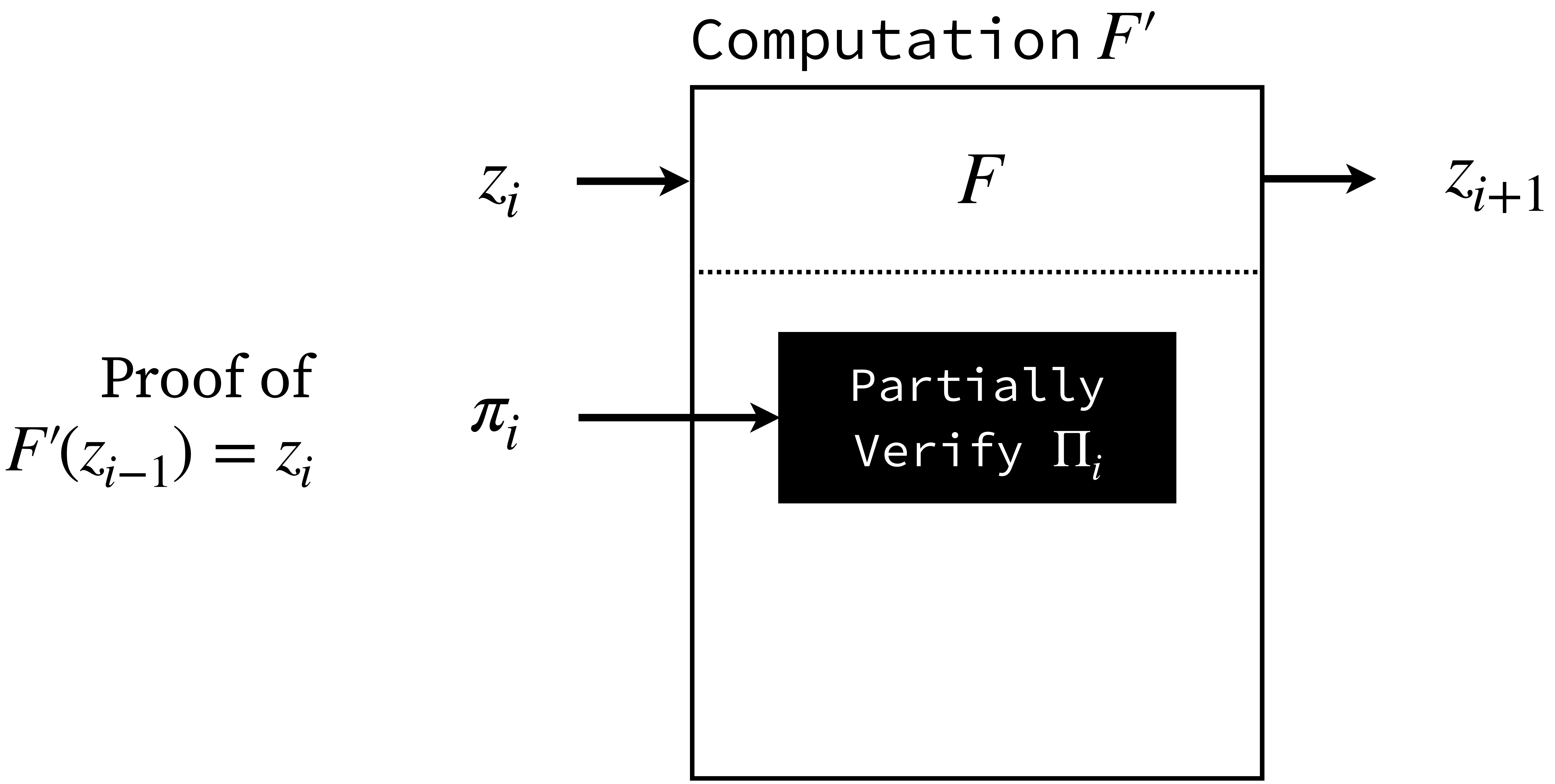
Valiant's Original IVC Approach [Val08,BCTV14]



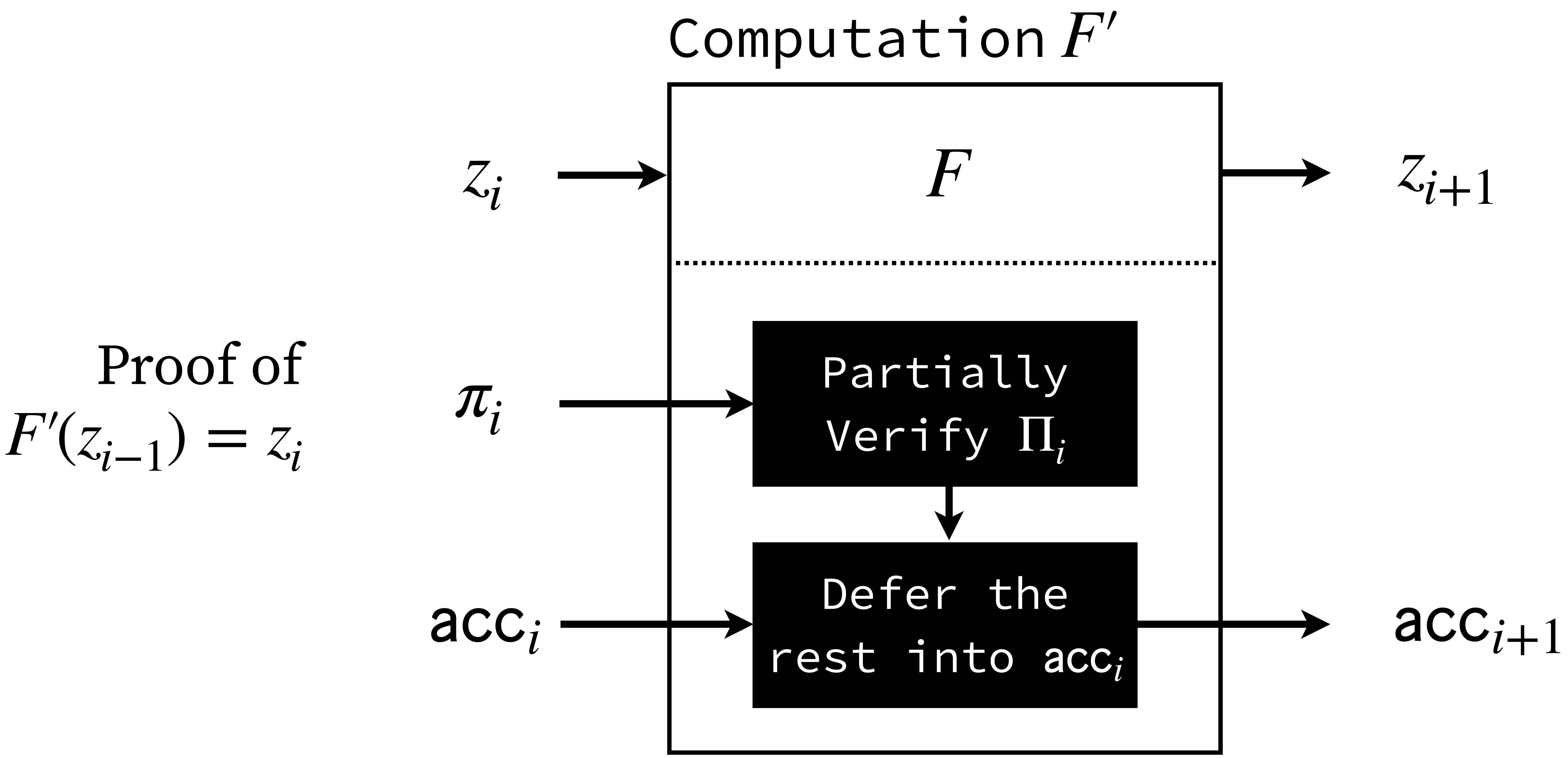
Drawbacks

- SNARKs in [BCTV14] require expensive cycles of pairing-friendly curves and trusted-setup
- Utilizing SNARKs without trusted setup require much larger verifier circuits

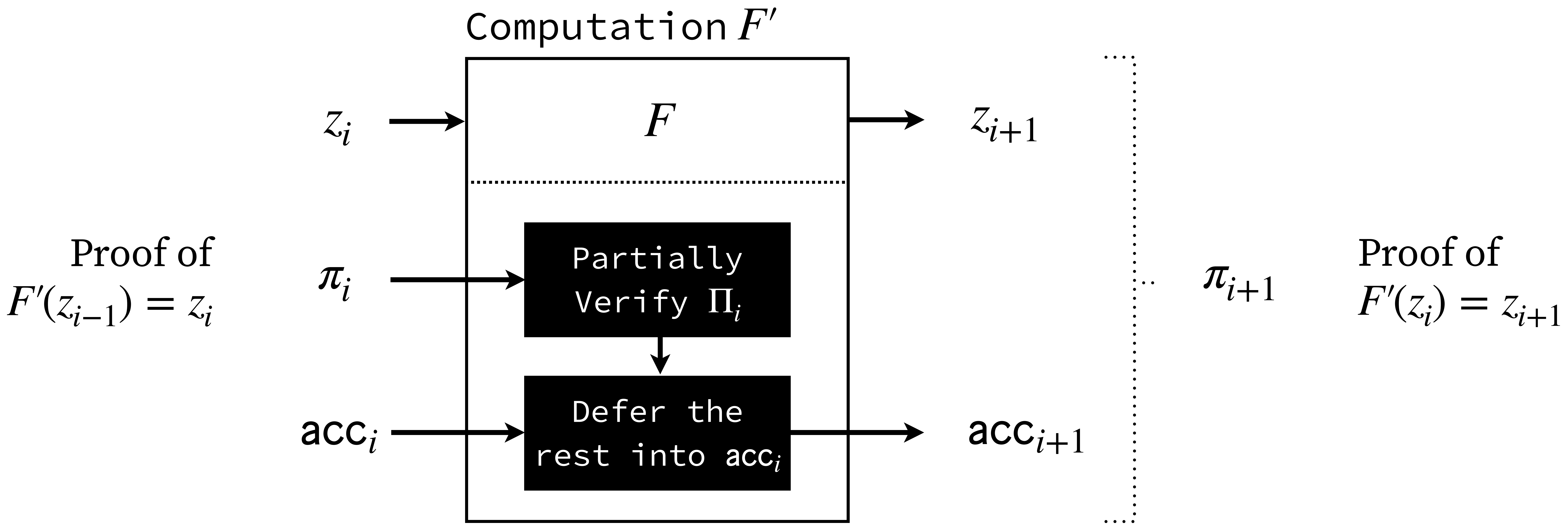
Halo's Approach: Partially Verify Proofs [BGH19, BCLMS20, BDFG20]



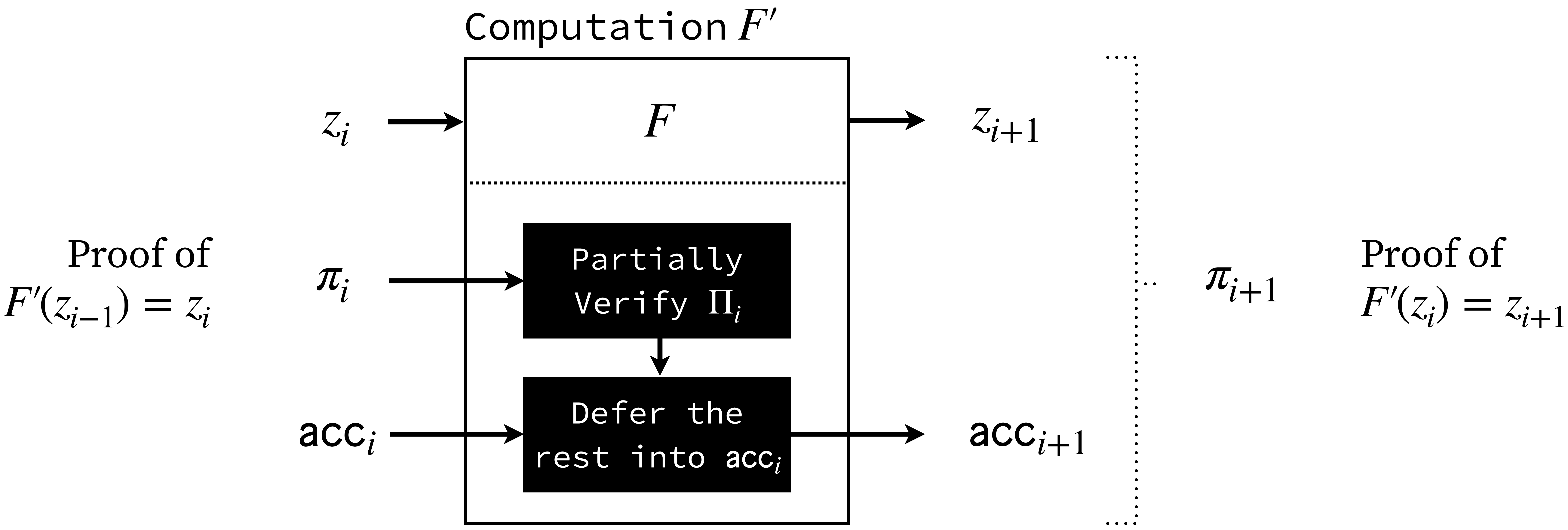
Halo's Approach: Partially Verify Proofs [BGH19,BCLMS20,BDFG20]



Halo's Approach: Partially Verify Proofs [BGH19, BCLMS20, BDFG20]



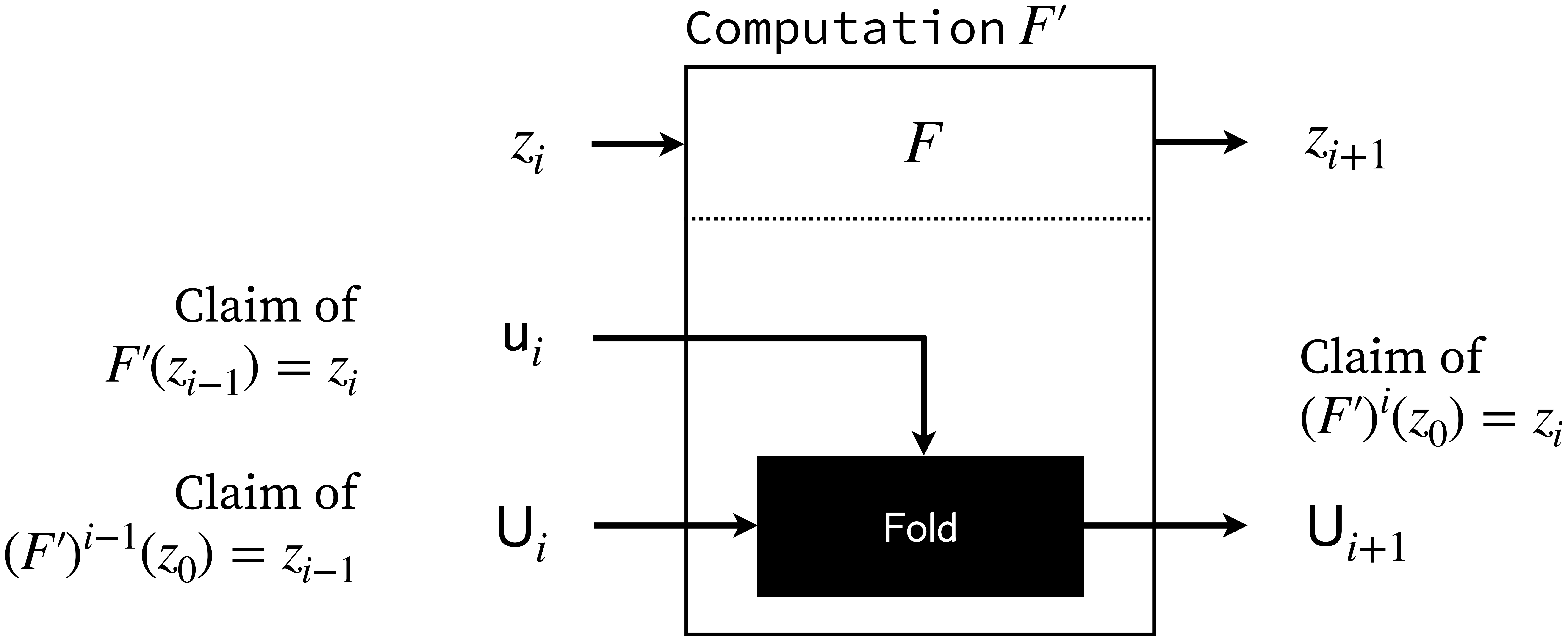
Halo's Approach: Partially Verify Proofs [BGH19,BCLMS20,BDFG20]



Drawbacks

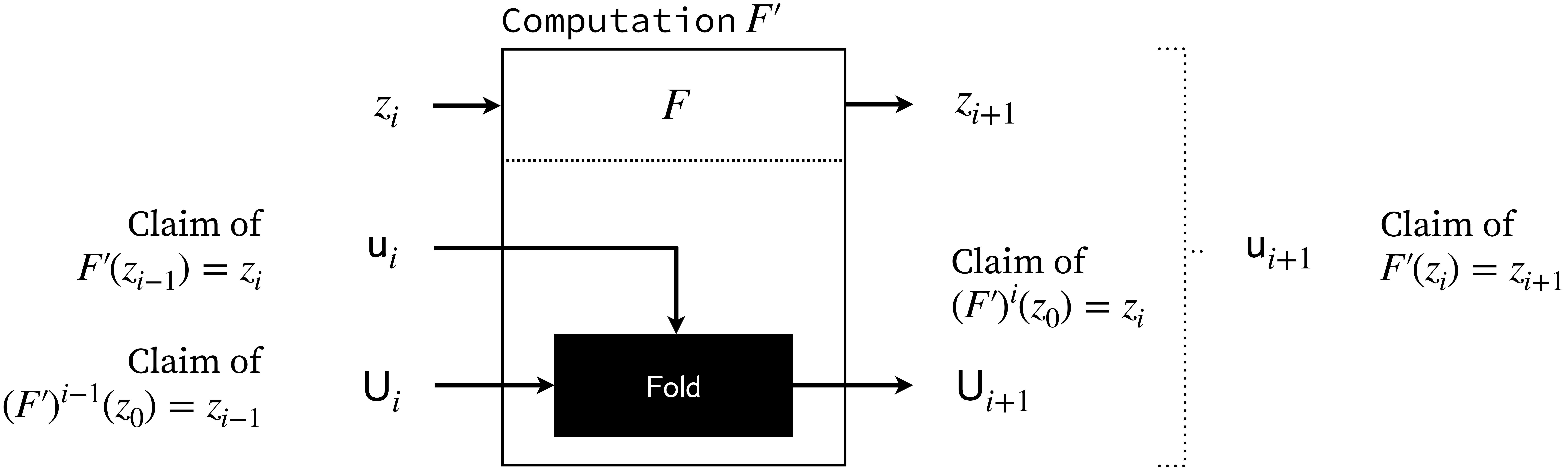
- Partially checking π_i in-circuit is still expensive
- Generating π_i is concretely and asymptotically expensive

Nova: Reduce Claims rather than Verify Proofs



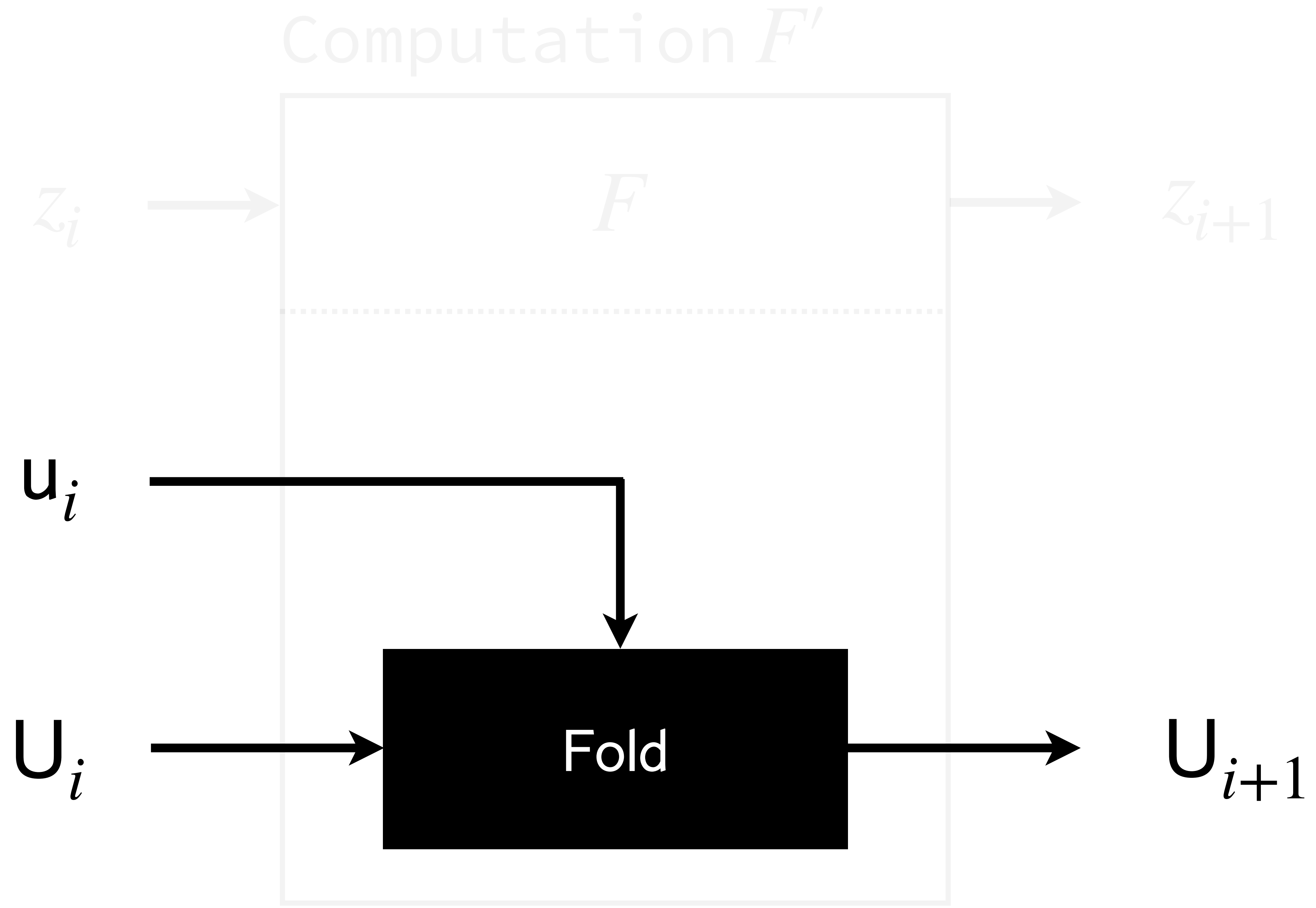
Fold reduces the task of checking two instances to the task of checking a single instance

Nova: Reduce Claims rather than Verify Proofs



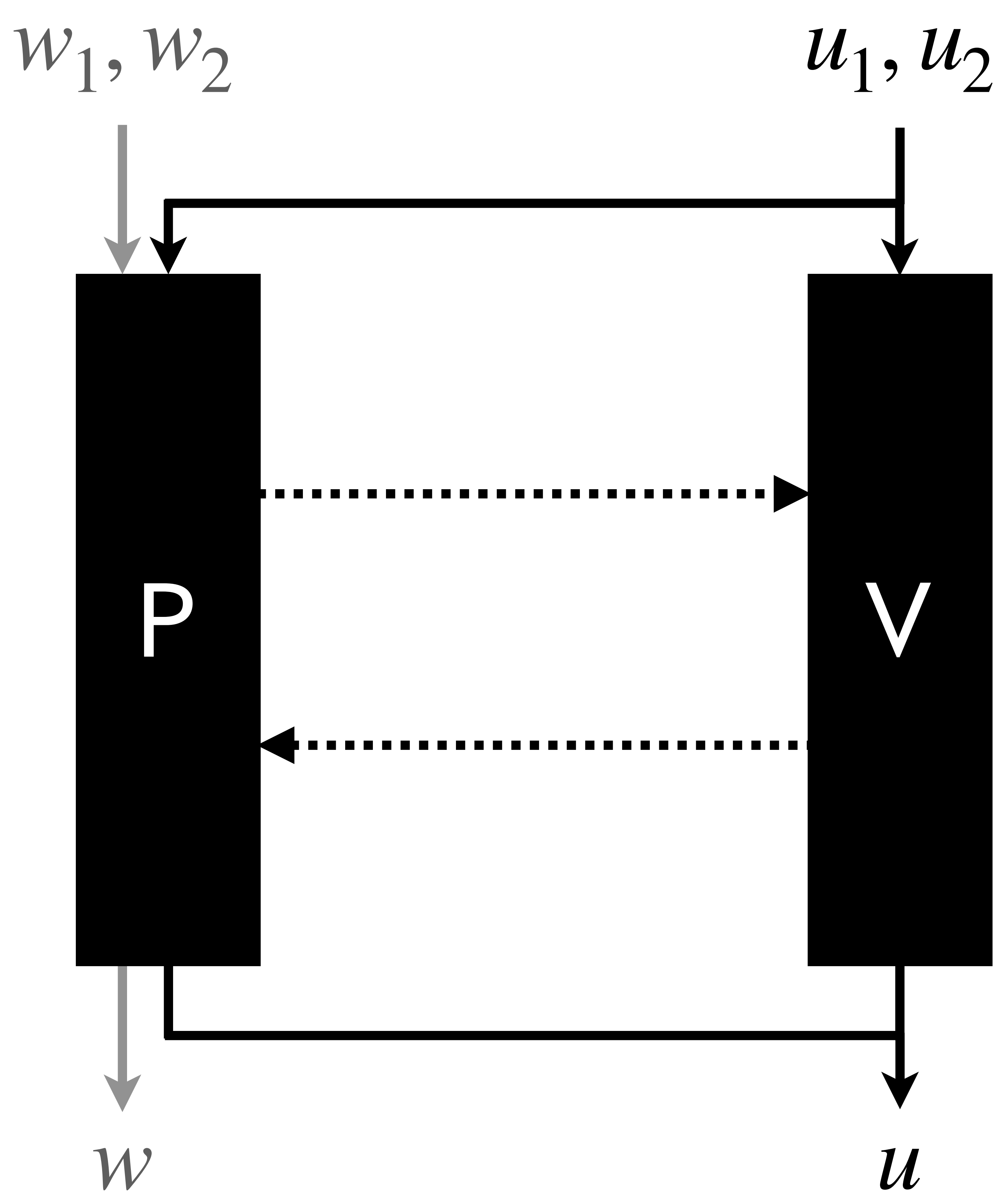
Fold reduces the task of checking two instances to the task of checking a single instance

How do we implement Fold?



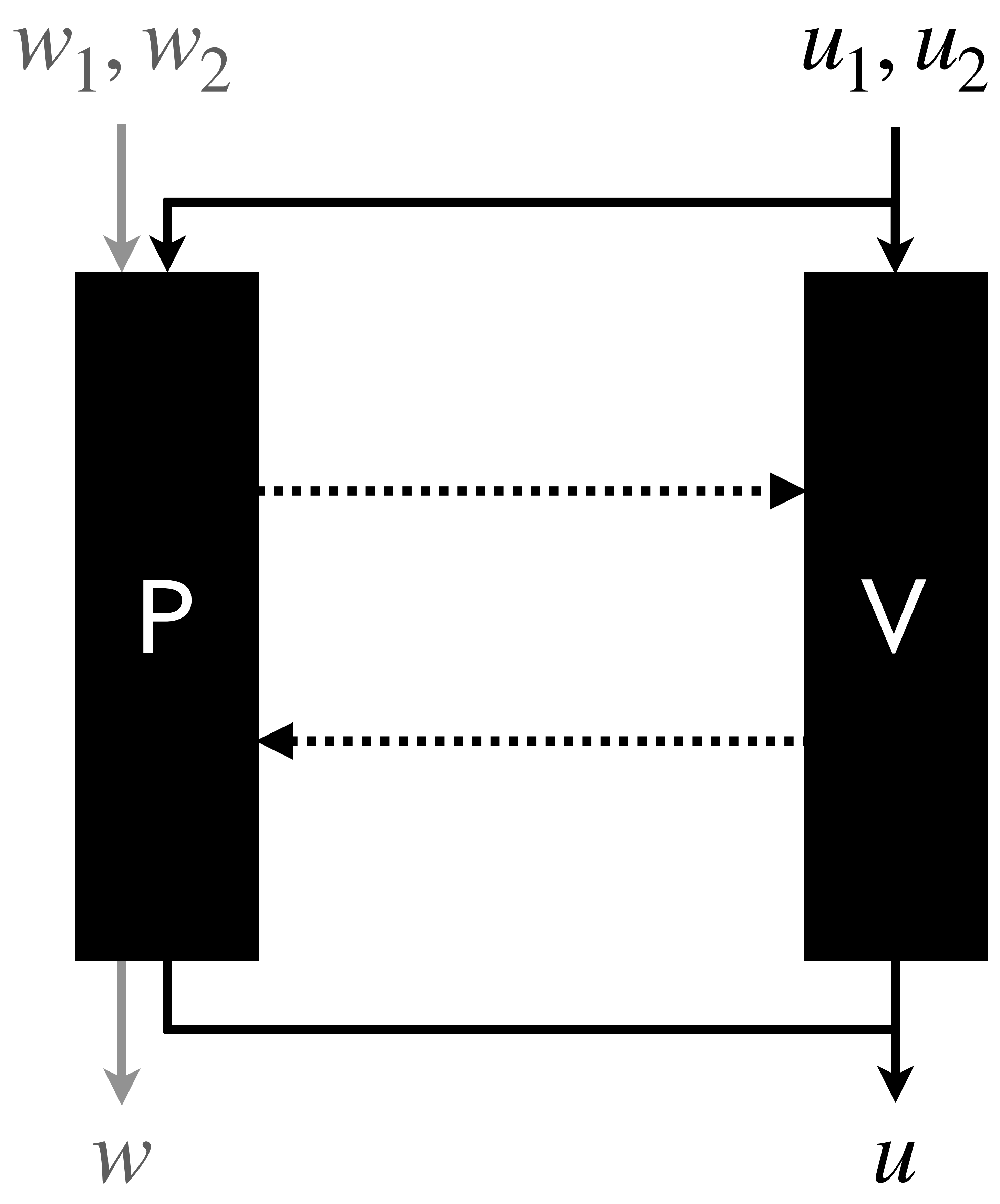
Solution: Folding Schemes

A folding scheme interactively reduces the claims $(u_1, w_1), (u_2, w_2) \in R$ to a claim $(u, w) \in R$



Solution: Folding Schemes

A folding scheme interactively reduces the claims $(u_1, w_1), (u_2, w_2) \in R$ to a claim $(u, w) \in R$

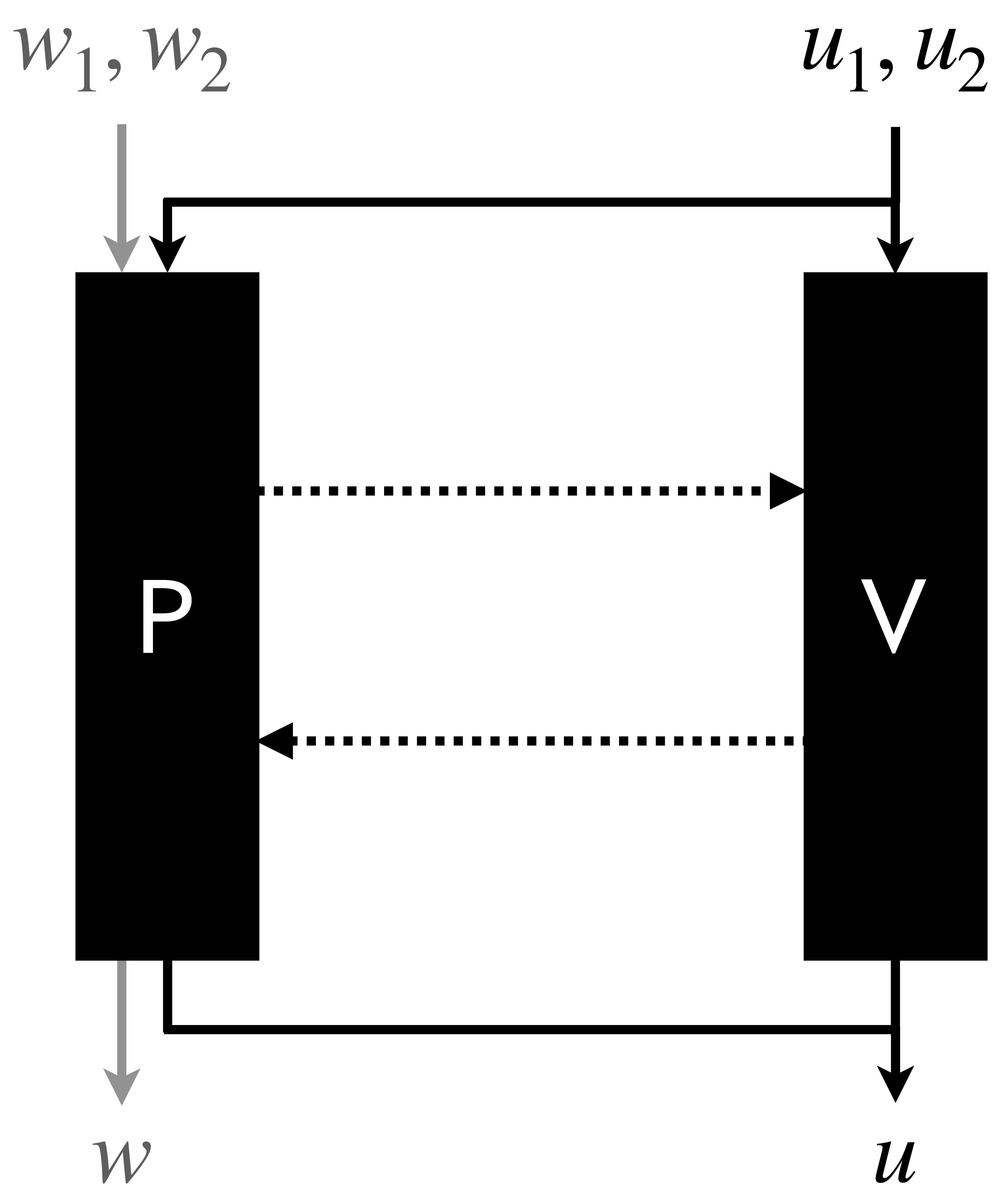


Completeness

If u_1, u_2 are satisfiable then u is satisfiable

Solution: Folding Schemes

A folding scheme interactively reduces the claims $(u_1, w_1), (u_2, w_2) \in R$ to a claim $(u, w) \in R$



Completeness

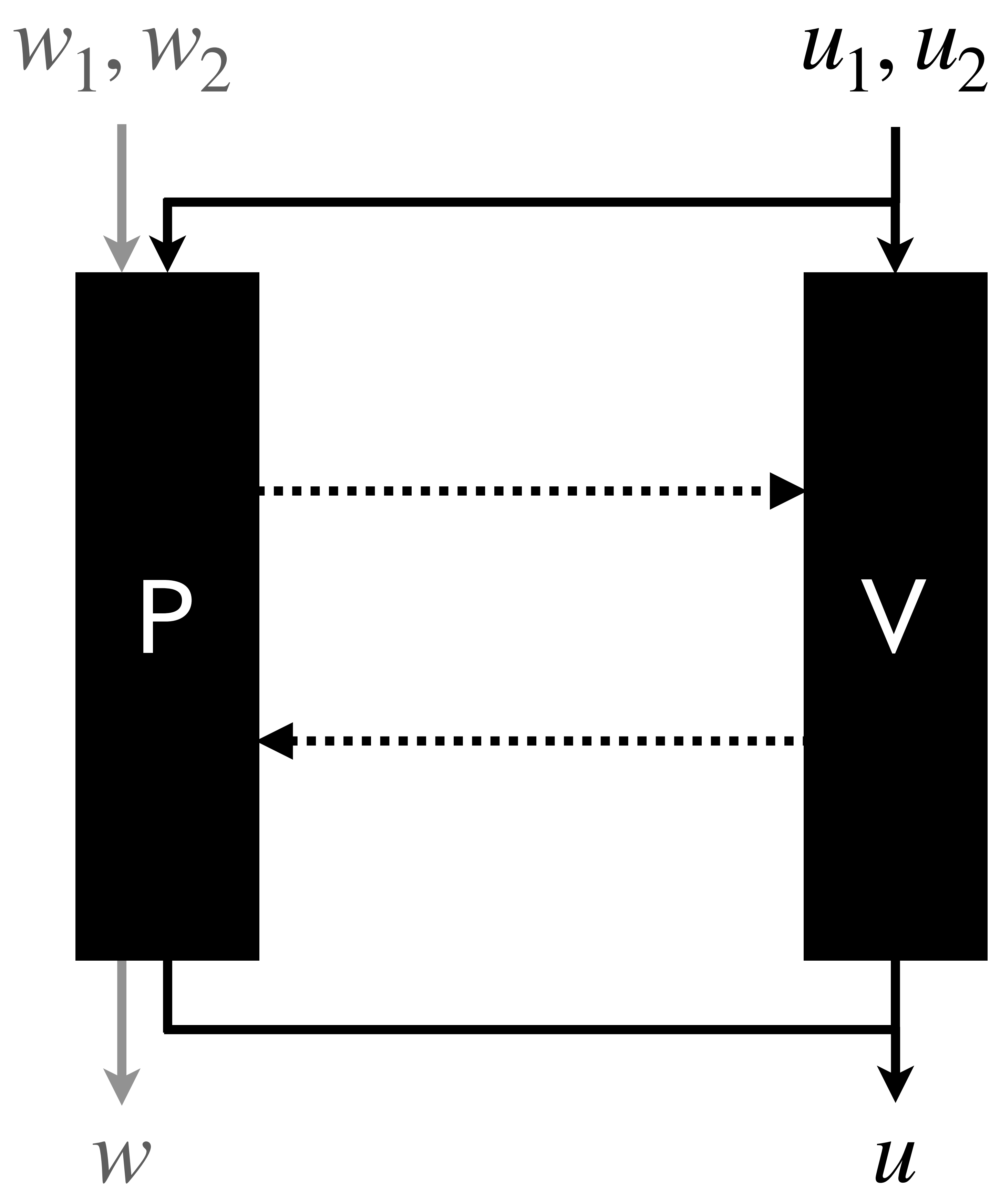
If u_1, u_2 are satisfiable then u is satisfiable

Knowledge Soundness

If prover outputs satisfying w then it must know satisfying w_1, w_2

Solution: Folding Schemes

A folding scheme interactively reduces the claims $(u_1, w_1), (u_2, w_2) \in R$ to a claim $(u, w) \in R$



Completeness

If u_1, u_2 are satisfiable then u is satisfiable

Efficiency

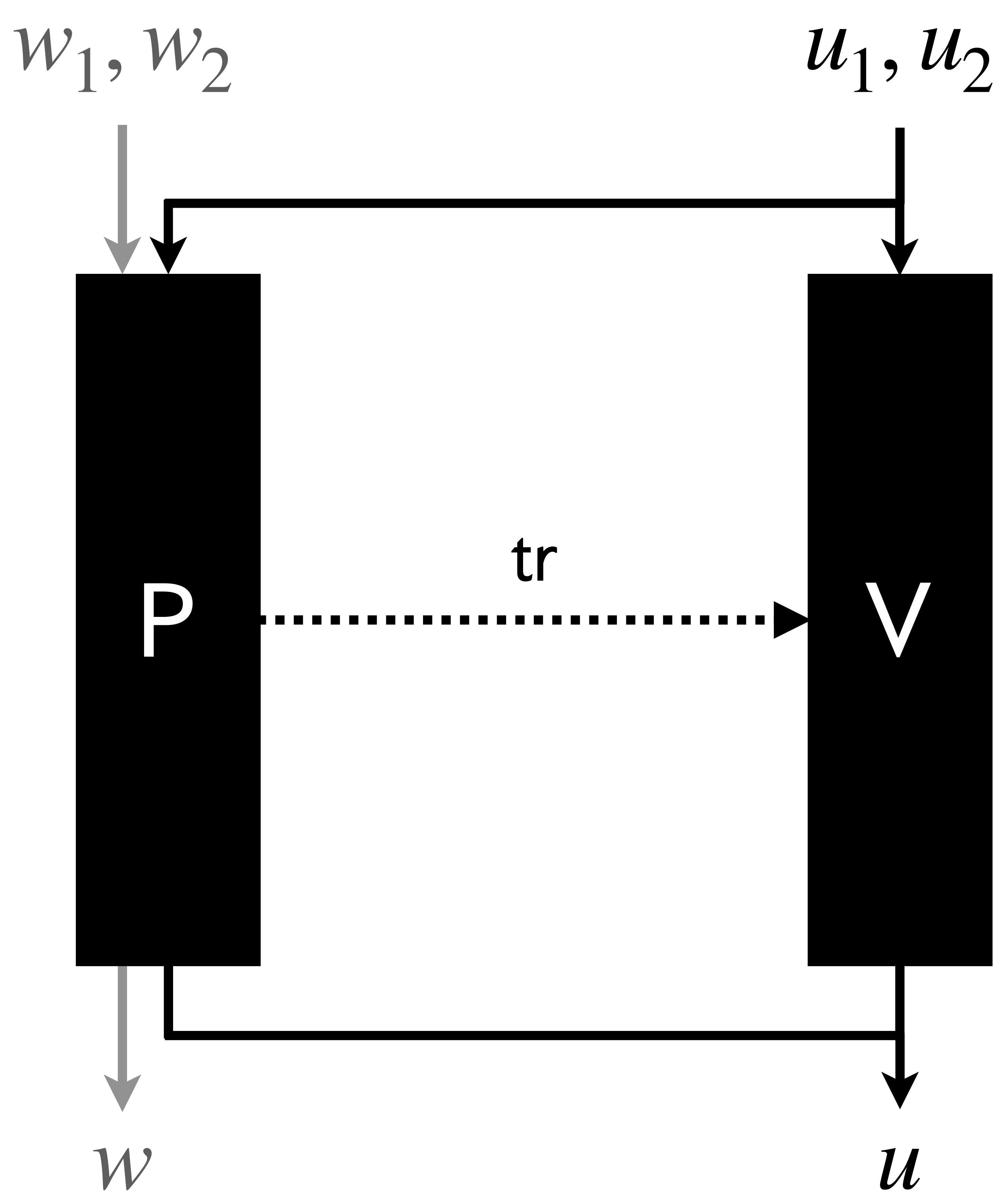
Folding should be much cheaper for the verifier than checking an instance

Knowledge Soundness

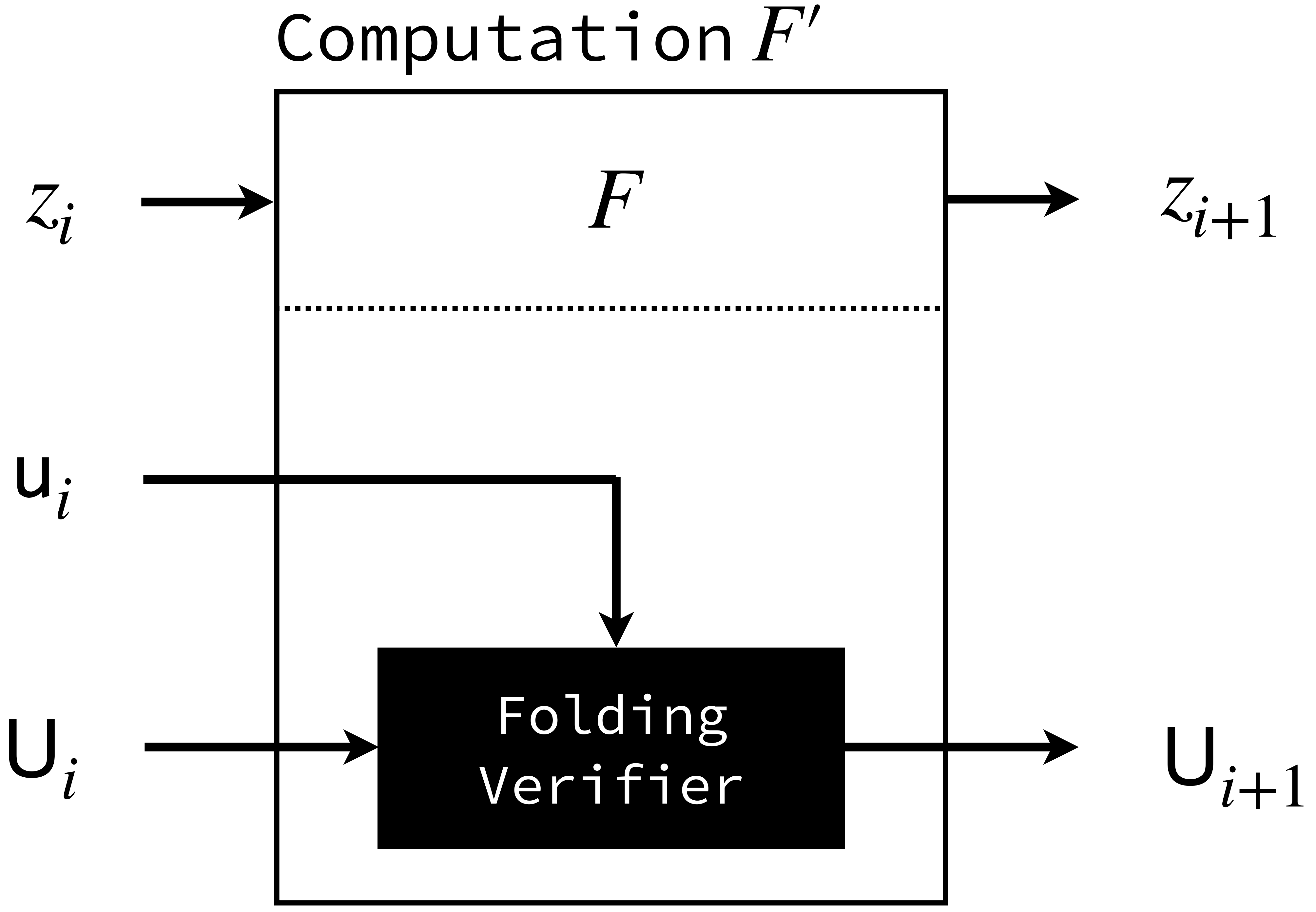
If prover outputs satisfying w then it must know satisfying w_1, w_2

Non-Interactive Folding Schemes

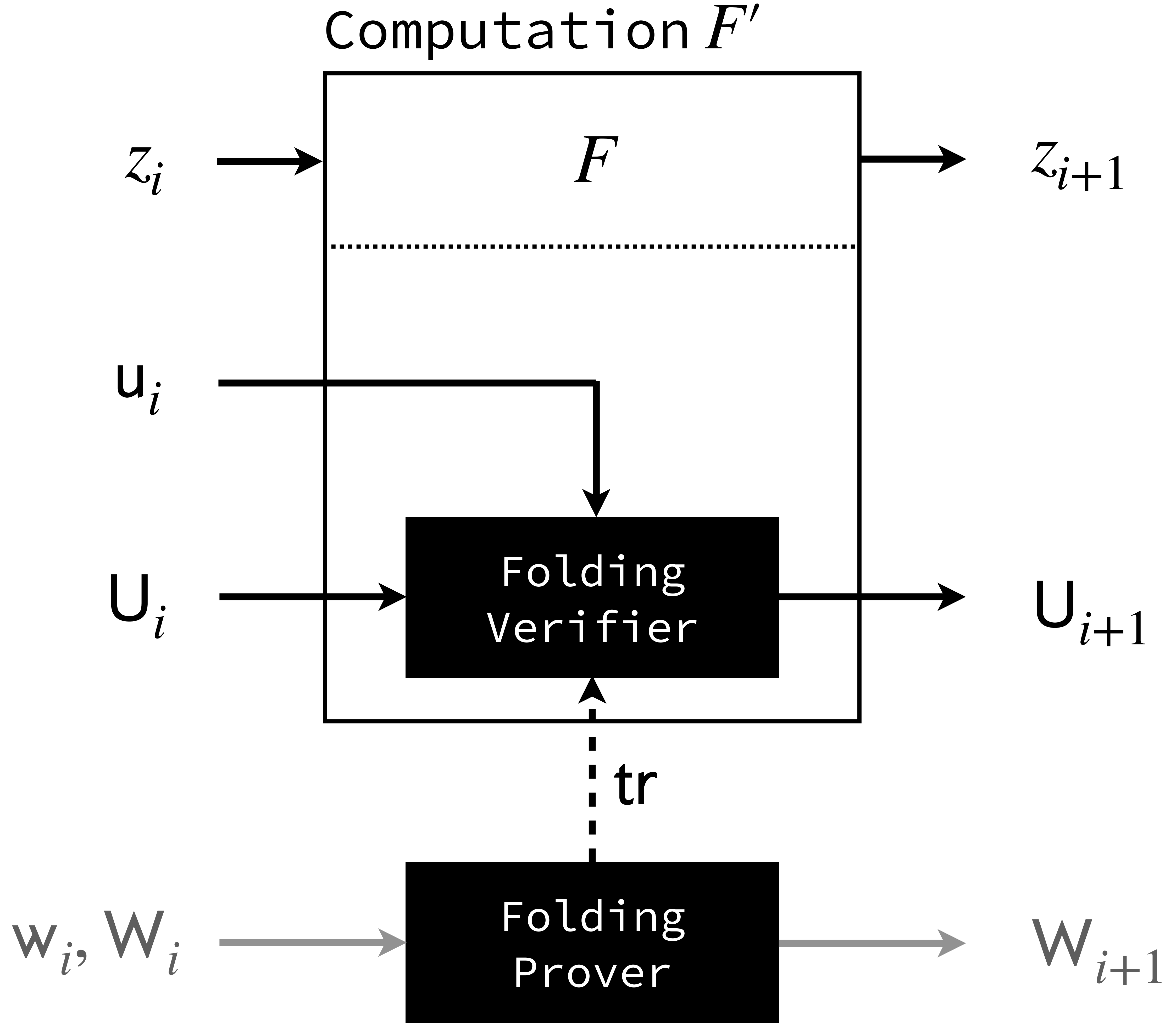
A public-coin folding scheme can be made non-interactive via the Fiat-Shamir Transform.



Given a non-interactive folding scheme for NP, F' can verifiably fold u_i and U_i by running the folding verifier.



Given a non-interactive folding scheme for NP, F' can verifiably fold u_i and U_i by running the folding verifier.



Witness w_i consists of a trace of the execution of $F(z_{i-1}) = z_i$

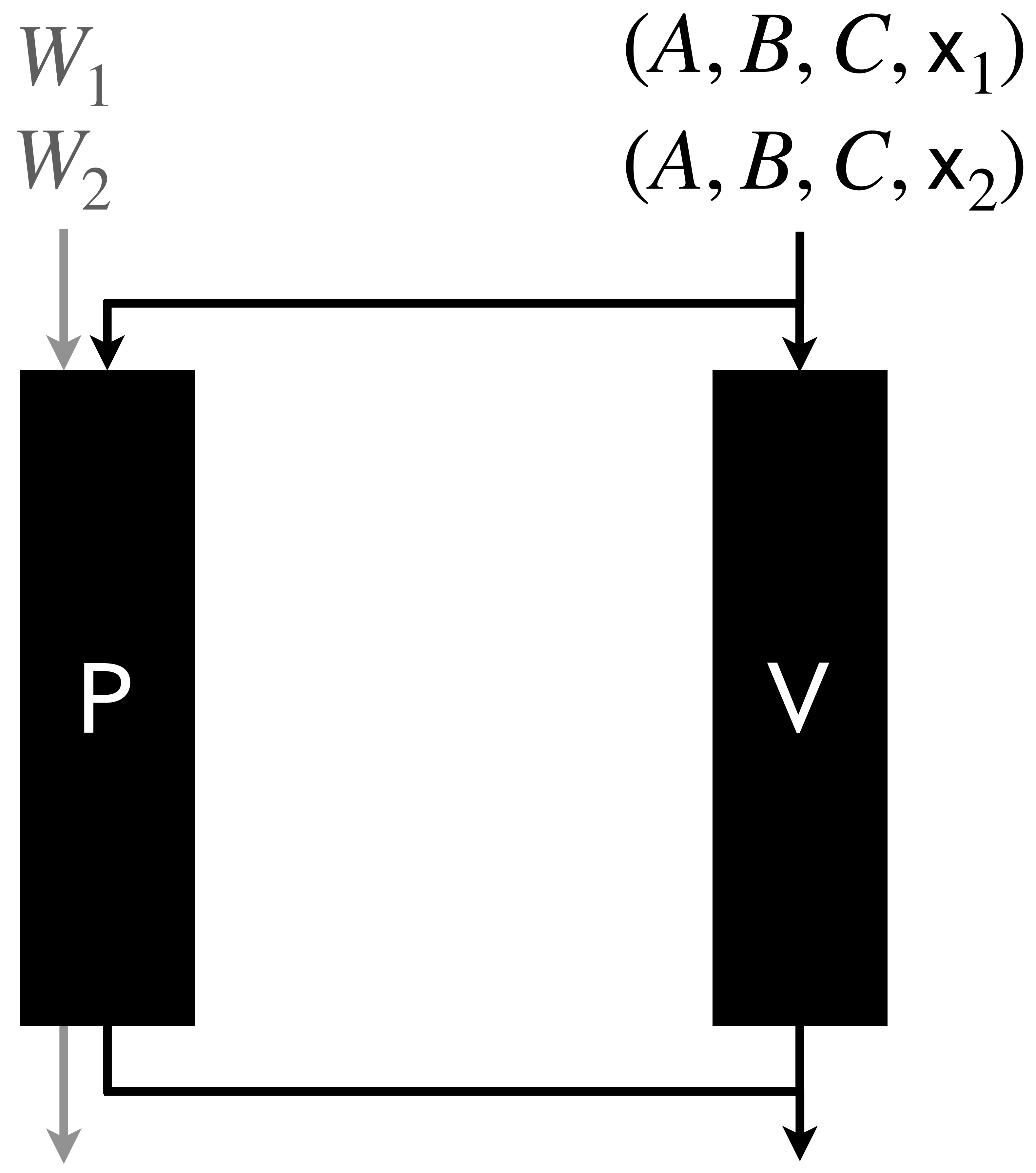
Folding an NP-Complete Relation

We start with R1CS a popular algebraic constraint system for NP

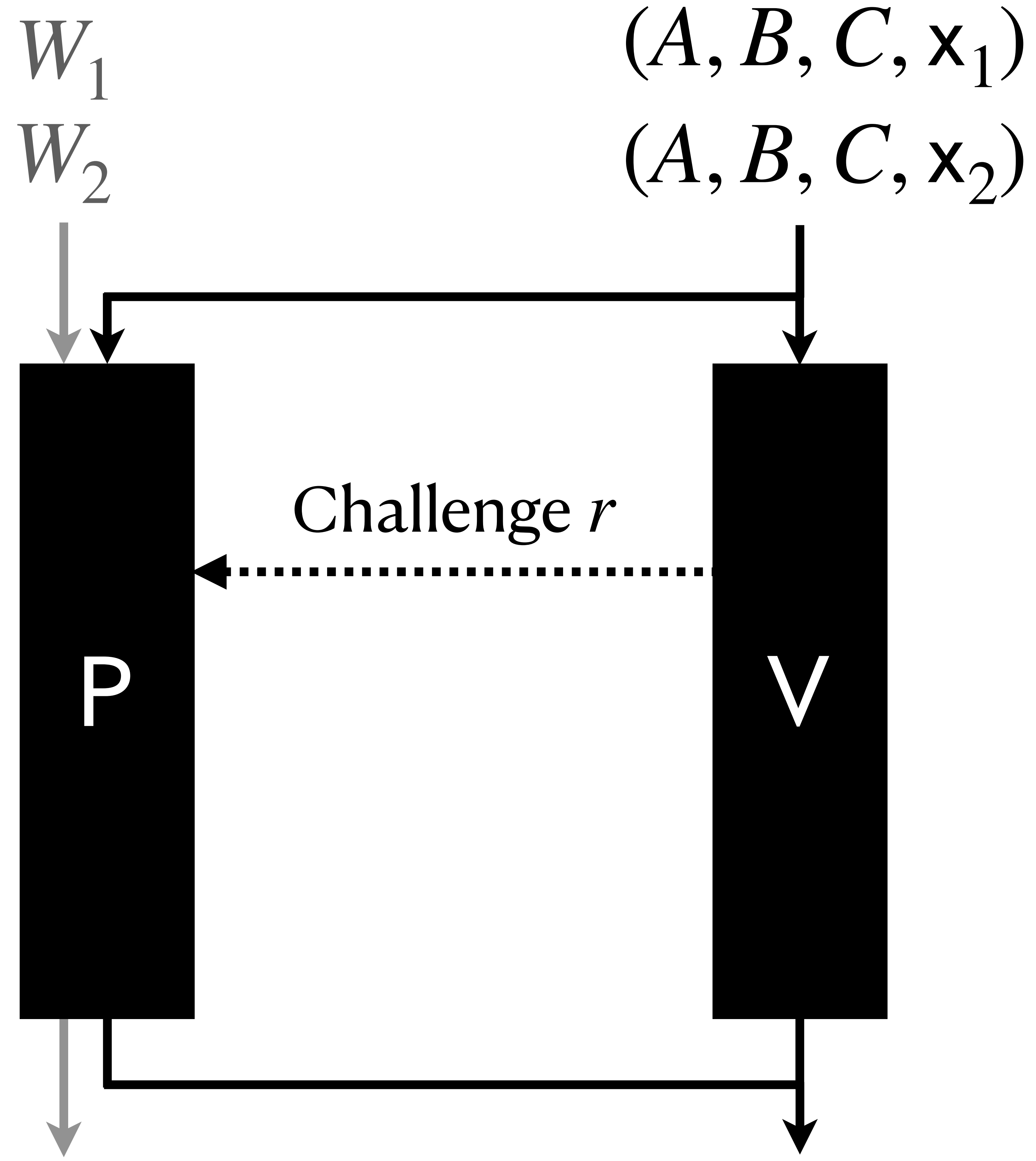
An R1CS statement consists of constraint matrices A, B, C, and vector x. A witness vector W is satisfying if for $Z = (W, \mathsf{x}, 1)$

$$A \; Z \circ B \; Z = C \; Z$$

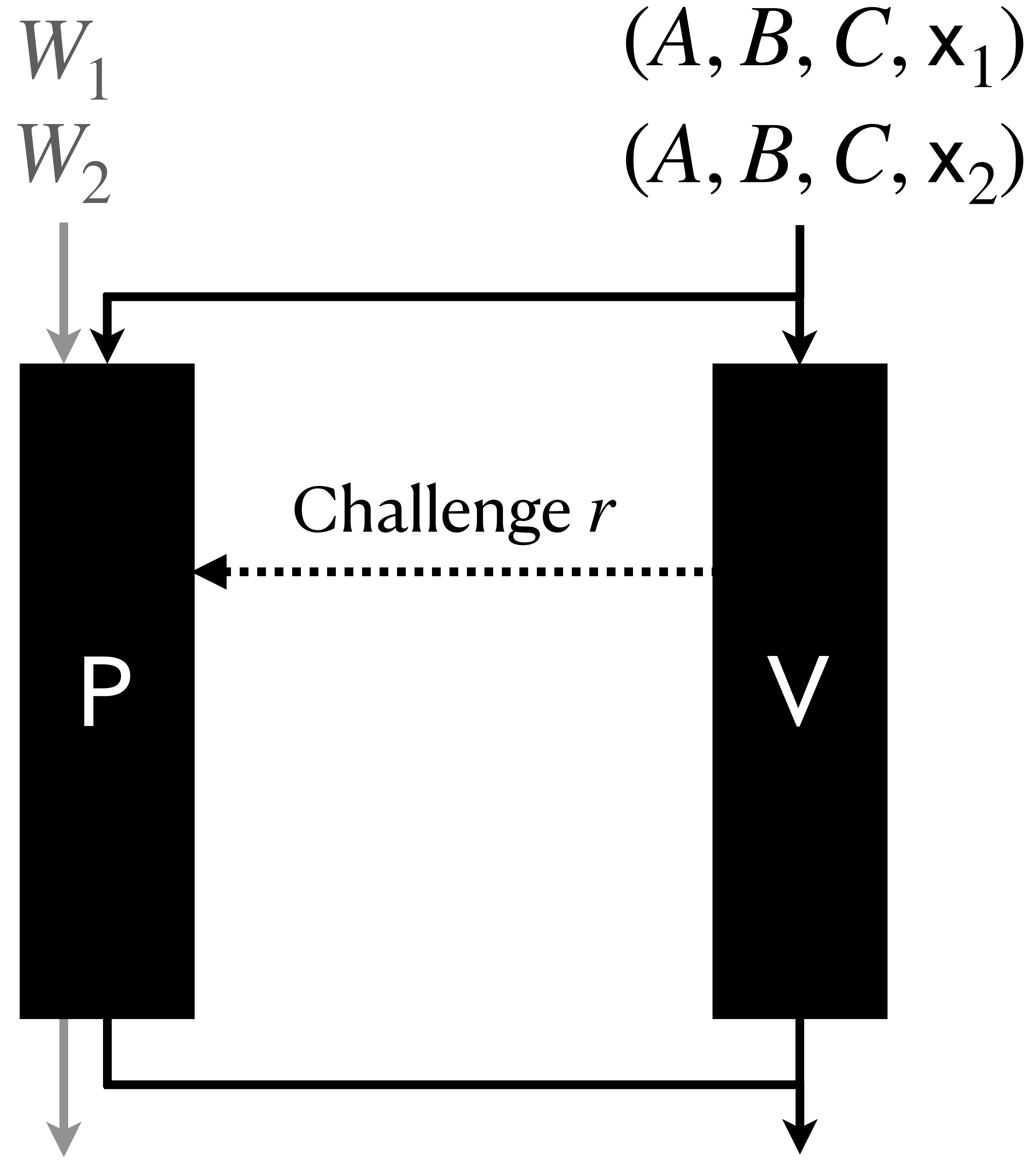
Attempt to Fold R1CS Instances



Attempt to Fold R1CS Instances



Attempt to Fold R1CS Instances



Take a random linear combination

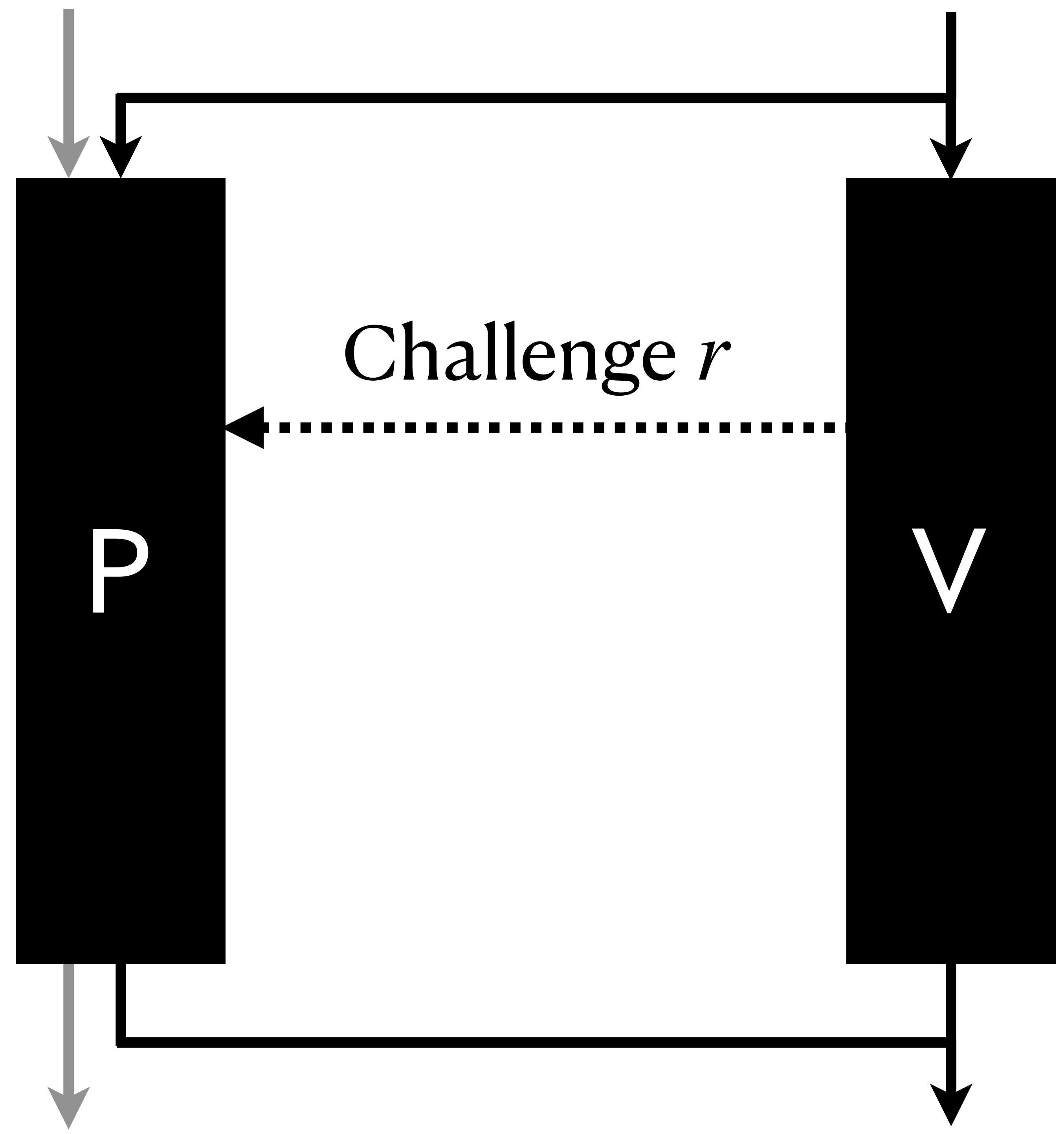
$\mathsf{x} \leftarrow \mathsf{x}_1 + r \cdot \mathsf{x}_2$ (Prover side)

$\mathsf{x} \leftarrow \mathsf{x}_1 + r \cdot \mathsf{x}_2$ (Verifier side)

Attempt to Fold R1CS Instances

W_1
W_2

(A, B, C, x_1)
(A, B, C, x_2)



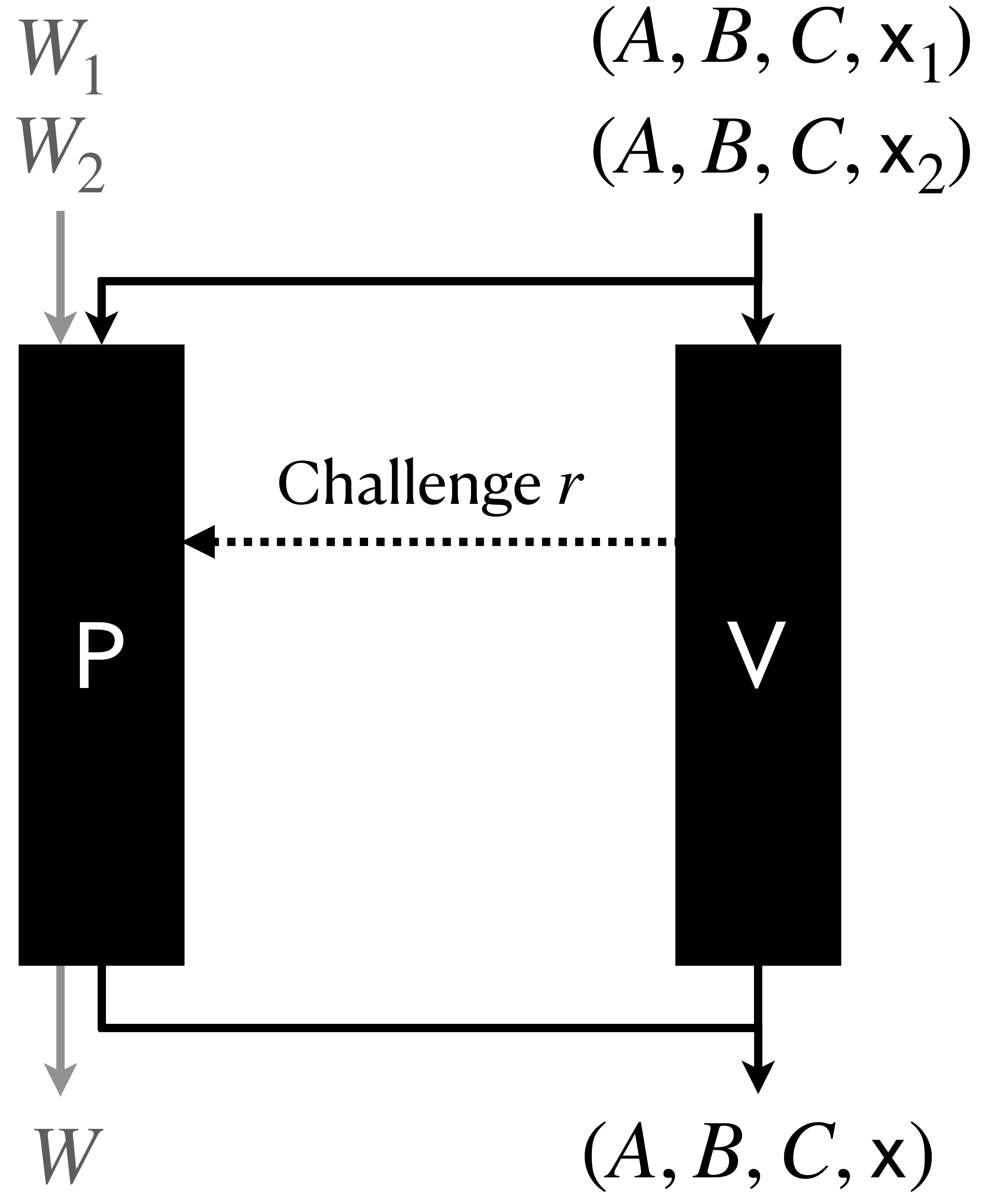
Take a random linear combination

$\mathsf{x} \leftarrow \mathsf{x}_1 + r \cdot \mathsf{x}_2$

$W \leftarrow W_1 + r \cdot W_2$

$\mathsf{x} \leftarrow \mathsf{x}_1 + r \cdot \mathsf{x}_2$

Attempt to Fold R1CS Instances



Take a random linear combination

$\mathsf{x} \leftarrow \mathsf{x}_1 + r \cdot \mathsf{x}_2$

$\mathsf{x} \leftarrow \mathsf{x}_1 + r \cdot \mathsf{x}_2$

$W \leftarrow W_1 + r \cdot W_2$

Attempt to Fold R1CS Instances

Unfortunately, letting $Z_i = (W_i, \mathsf{x}_i, 1)$ and $Z = Z_1 + r \cdot Z_2$, we have that $AZ \circ BZ \neq CZ$

Attempt to Fold R1CS Instances

Unfortunately, letting $Z_i = (W_i, \mathsf{x}_i, 1)$ and $Z = Z_1 + r \cdot Z_2$, we have that $AZ \circ BZ \neq CZ$

$$CZ = AZ_1 \circ BZ_1 + r \cdot AZ_2 \circ BZ_2$$

Attempt to Fold R1CS Instances

Unfortunately, letting $Z_i = (W_i, \mathsf{x}_i, 1)$ and $Z = Z_1 + r \cdot Z_2$, we have that $AZ \circ BZ \neq CZ$

$$CZ = AZ_1 \circ BZ_1 + r \cdot AZ_2 \circ BZ_2$$

$$AZ \circ BZ = AZ_1 \circ BZ_1 + r \cdot (AZ_1 \circ BZ_2 + AZ_2 \circ BZ_1) + r^2 \cdot (AZ_2 \circ BZ_2)$$

Attempt to Fold R1CS Instances

Unfortunately, letting $Z_i = (W_i, \mathsf{x}_i, 1)$ and $Z = Z_1 + r \cdot Z_2$, we have that $AZ \circ BZ \neq CZ$

$$CZ = AZ_1 \circ BZ_1 + r \cdot AZ_2 \circ BZ_2$$

$$AZ \circ BZ = AZ_1 \circ BZ_1 + r \cdot (AZ_1 \circ BZ_2 + AZ_2 \circ BZ_1) + r^2 \cdot (AZ_2 \circ BZ_2)$$

To absorb the error terms
we introduce an error
vector E in the statement

Attempt to Fold R1CS Instances

Unfortunately, letting $Z_i = (W_i, \mathsf{x}_i, 1)$ and $Z = Z_1 + r \cdot Z_2$, we have that $AZ \circ BZ \neq CZ$

$$CZ = AZ_1 \circ BZ_1 + r \cdot AZ_2 \circ BZ_2$$

To absorb the extra r factor we introduce scalar u in the statement

$$AZ \circ BZ = AZ_1 \circ BZ_1 + r \cdot (AZ_1 \circ BZ_2 + AZ_2 \circ BZ_1) + r^2 \cdot (AZ_2 \circ BZ_2)$$

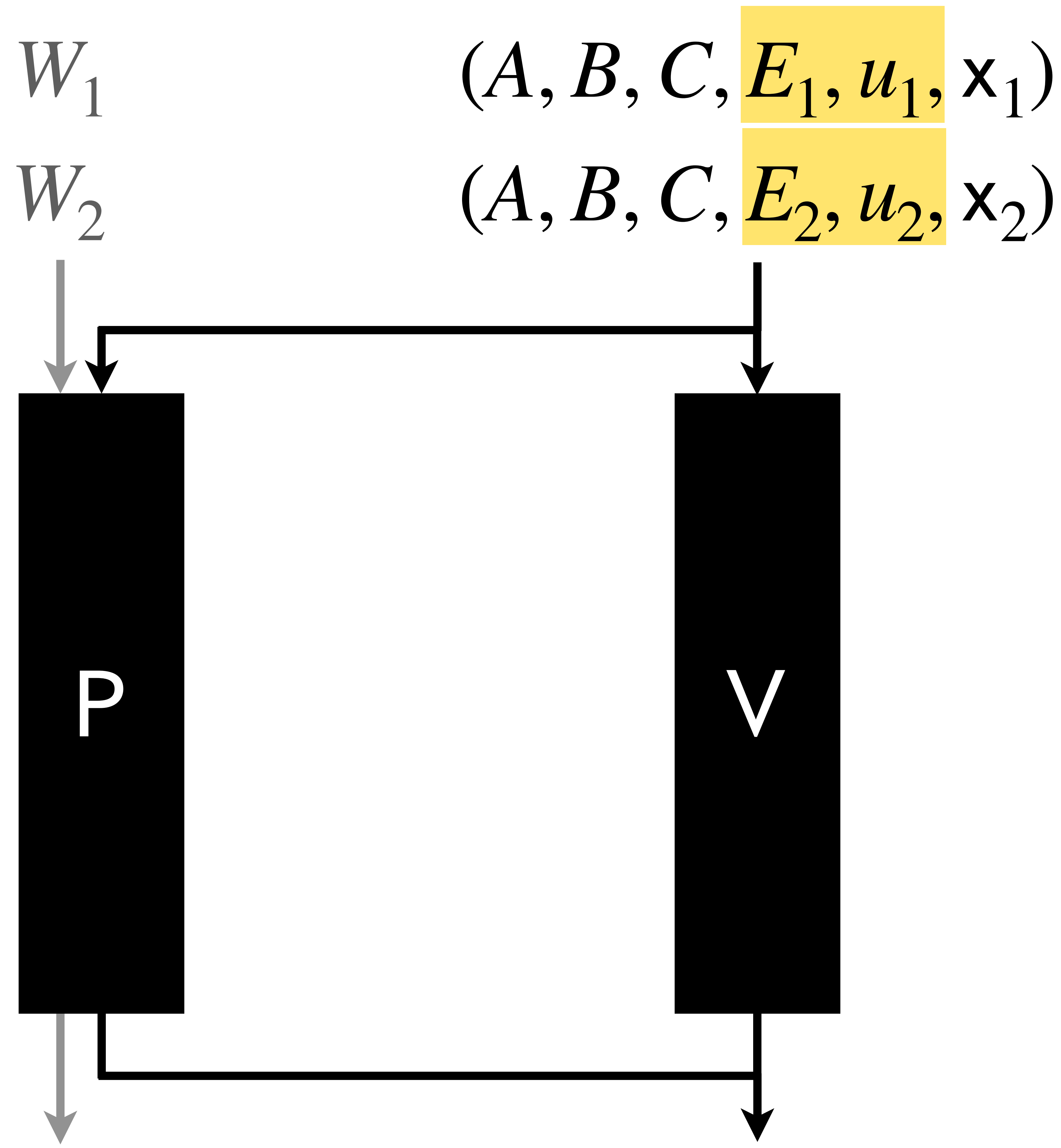
To absorb the error terms we introduce an error vector E in the statement

Relaxed R1CS: A Variant of R1CS Amenable to Folding

A relaxed R1CS statement additionally contains an error vector E and scalar u. A witness vector W is satisfying if for $Z = (W, \mathsf{x}, u)$

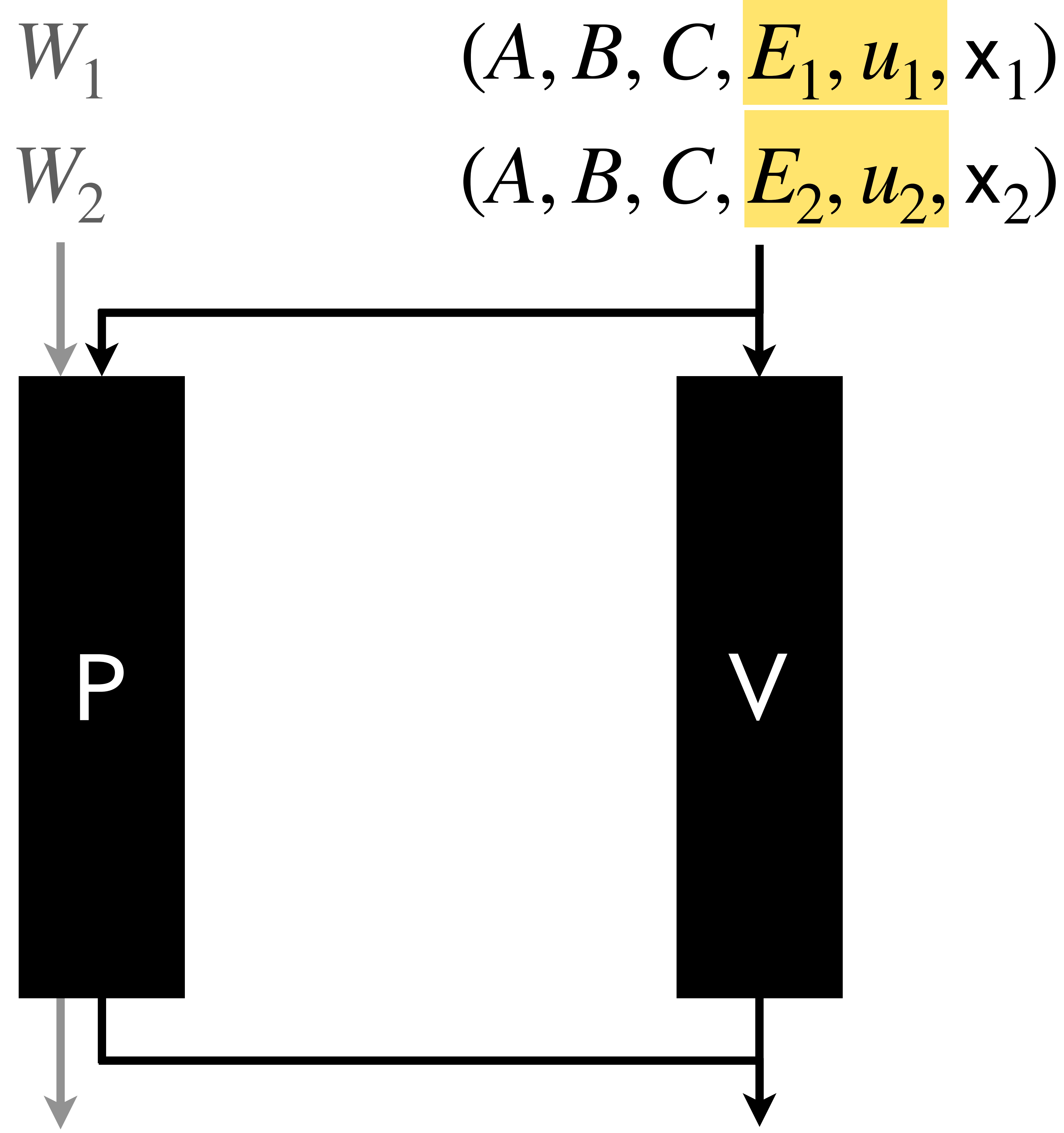
$$A \; Z \circ B \; Z = u \cdot C \; Z + E$$

Folding Scheme for Relaxed R1CS (Almost)



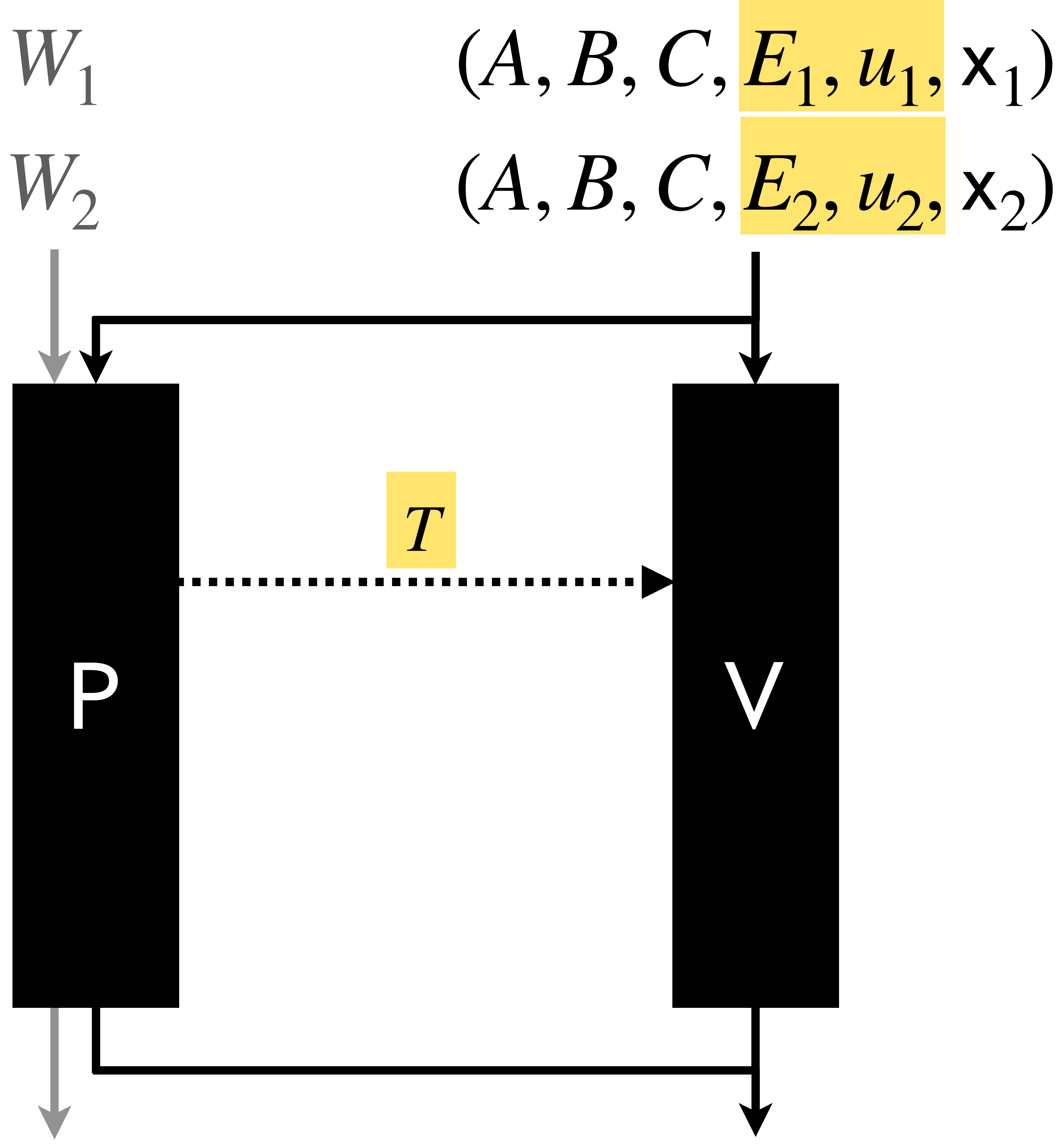
Folding Scheme for Relaxed R1CS (Almost)

$$T \leftarrow AZ_1 \circ BZ_2 + AZ_2 \circ BZ_1 \\ -u_1 \cdot CZ_2 - u_2 \cdot CZ_1$$



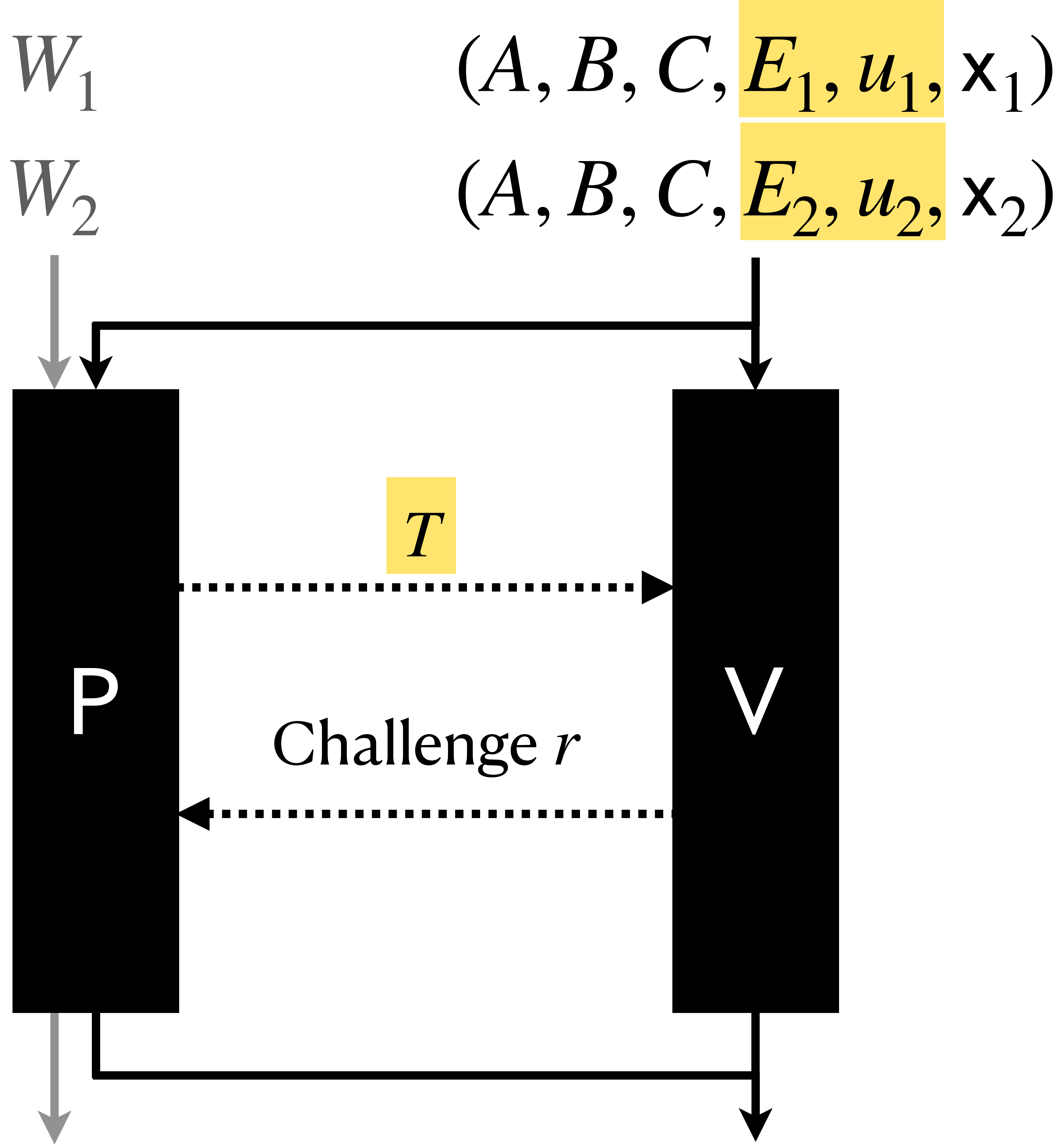
Folding Scheme for Relaxed R1CS (Almost)

$$T \leftarrow AZ_1 \circ BZ_2 + AZ_2 \circ BZ_1 - u_1 \cdot CZ_2 - u_2 \cdot CZ_1$$



Folding Scheme for Relaxed R1CS (Almost)

$$T \leftarrow AZ_1 \circ BZ_2 + AZ_2 \circ BZ_1 - u_1 \cdot CZ_2 - u_2 \cdot CZ_1$$

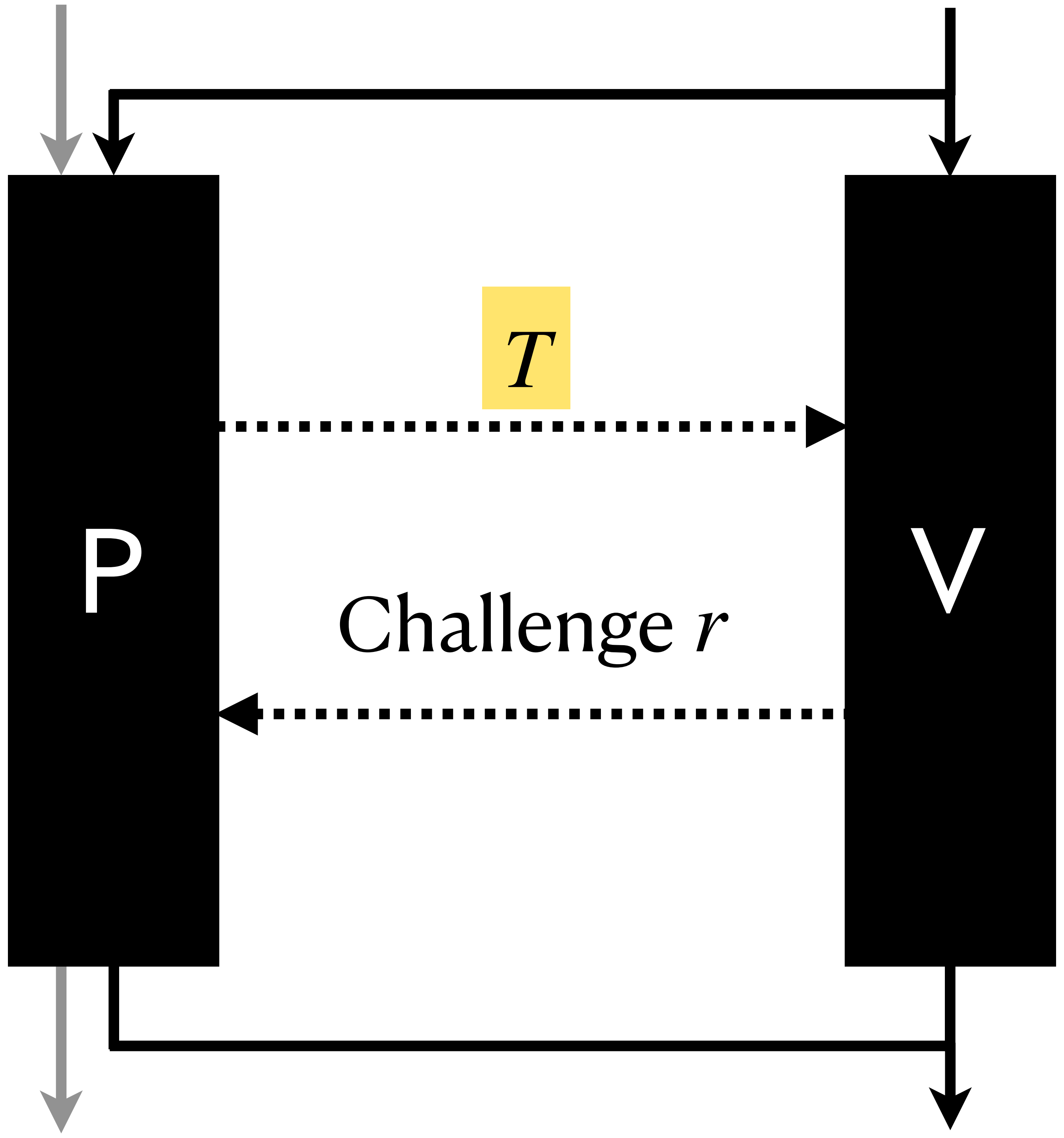


Folding Scheme for Relaxed R1CS (Almost)

$W_1 \qquad (A, B, C, E_1, u_1, \mathsf{x}_1)$

$W_2 \qquad (A, B, C, E_2, u_2, \mathsf{x}_2)$

$$T \leftarrow AZ_1 \circ BZ_2 + AZ_2 \circ BZ_1 - u_1 \cdot CZ_2 - u_2 \cdot CZ_1$$



$$u \leftarrow u_1 + r \cdot u_2$$

$$\mathsf{x} \leftarrow \mathsf{x}_1 + r \cdot \mathsf{x}_2$$

Folding Scheme for Relaxed R1CS (Almost)

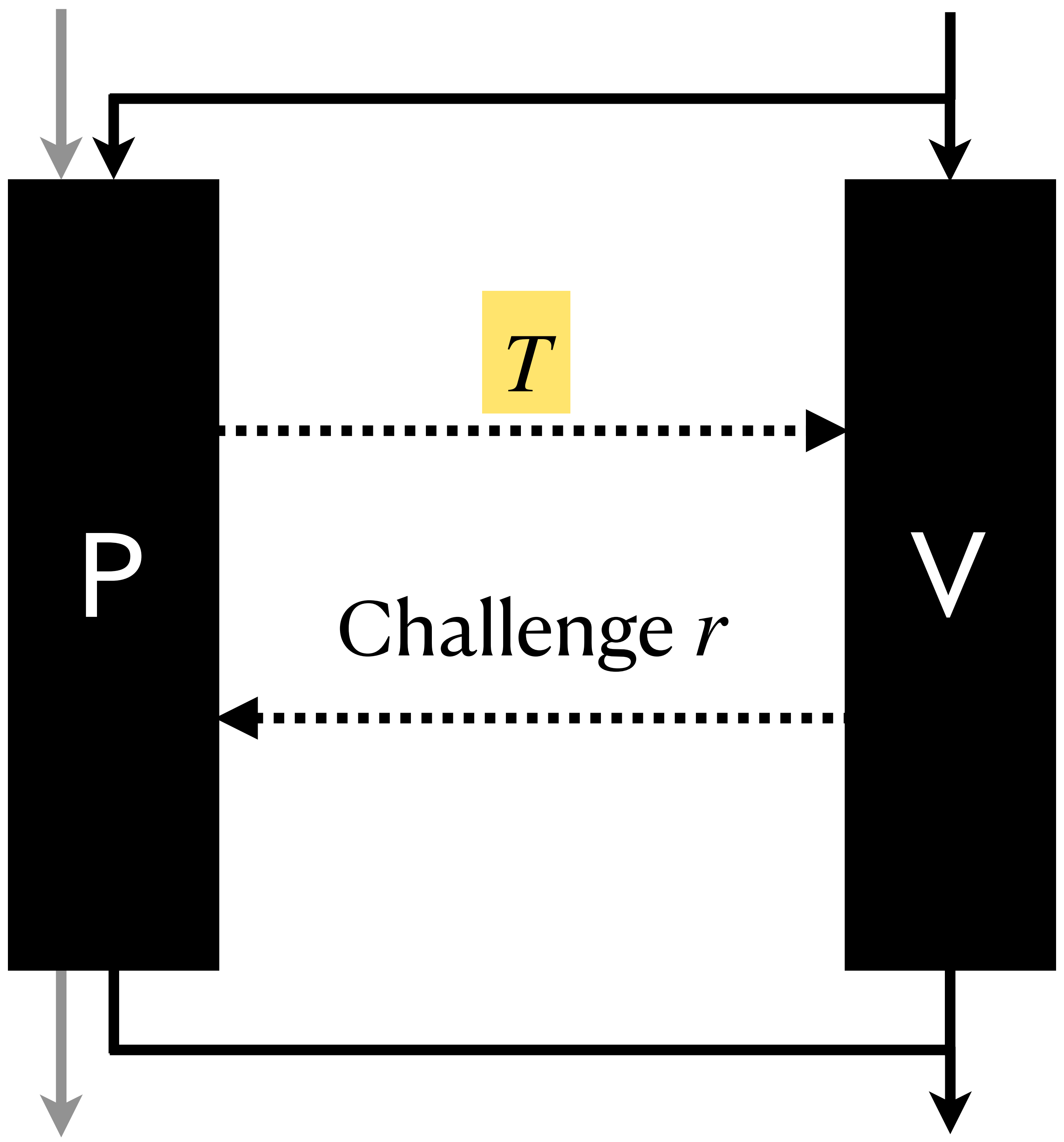
W_1 $\quad (A, B, C, E_1, u_1, \mathsf{x}_1)$

W_2 $\quad (A, B, C, E_2, u_2, \mathsf{x}_2)$

$$T \leftarrow AZ_1 \circ BZ_2 + AZ_2 \circ BZ_1 - u_1 \cdot CZ_2 - u_2 \cdot CZ_1$$

$$W \leftarrow W_1 + r \cdot W_2$$

$$E \leftarrow E_1 + r \cdot T + r^2 \cdot E_2$$

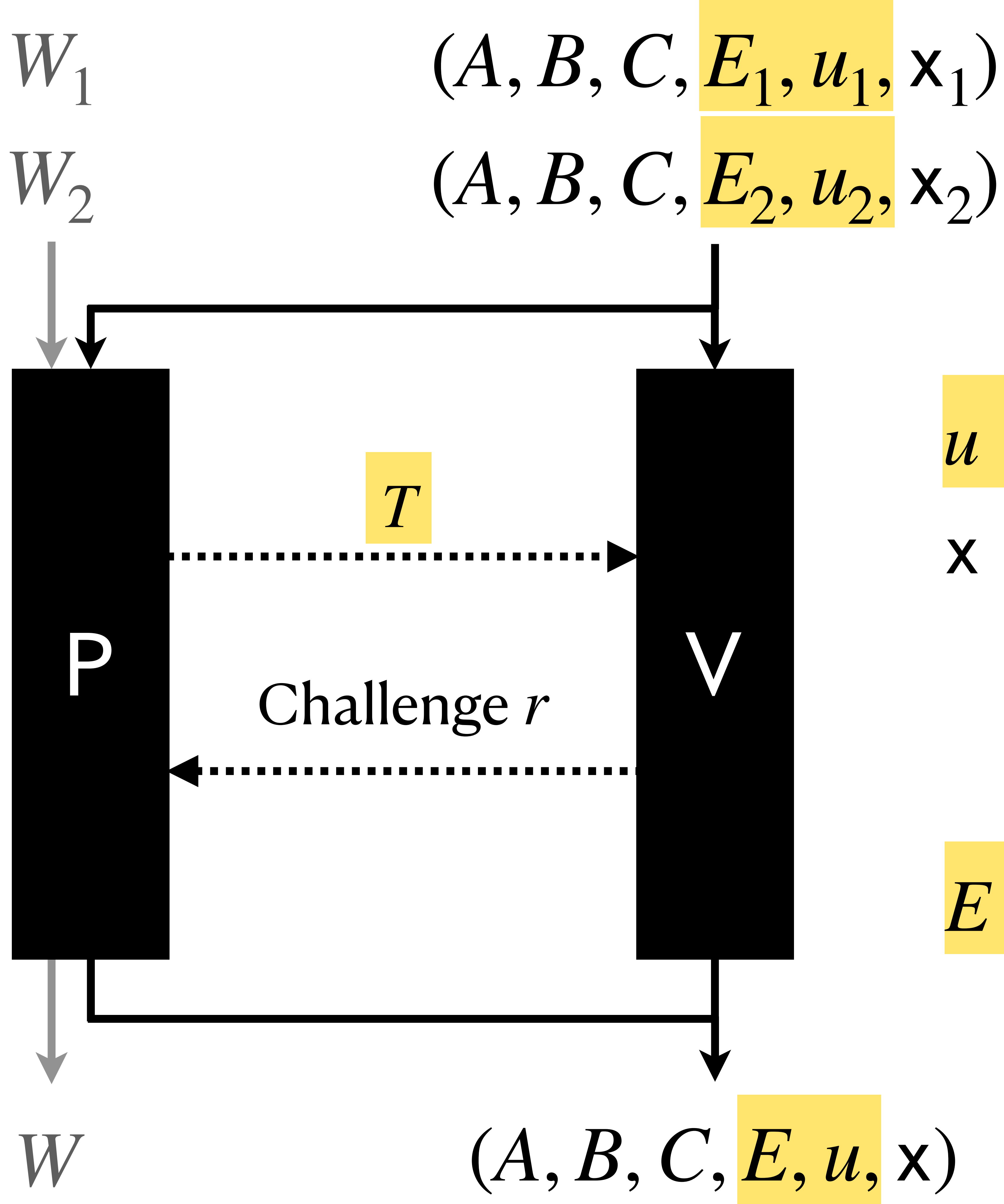


$$u \leftarrow u_1 + r \cdot u_2$$

$$\mathsf{x} \leftarrow \mathsf{x}_1 + r \cdot \mathsf{x}_2$$

$$E \leftarrow E_1 + r \cdot T + r^2 \cdot E_2$$

Folding Scheme for Relaxed R1CS (Almost)



$$T \leftarrow AZ_1 \circ BZ_2 + AZ_2 \circ BZ_1 - u_1 \cdot CZ_2 - u_2 \cdot CZ_1$$

$$W \leftarrow W_1 + r \cdot W_2$$

$$E \leftarrow E_1 + r \cdot T + r^2 \cdot E_2$$

$$u \leftarrow u_1 + r \cdot u_2$$

$$\mathrm{x} \leftarrow \mathrm{x}_1 + r \cdot \mathrm{x}_2$$

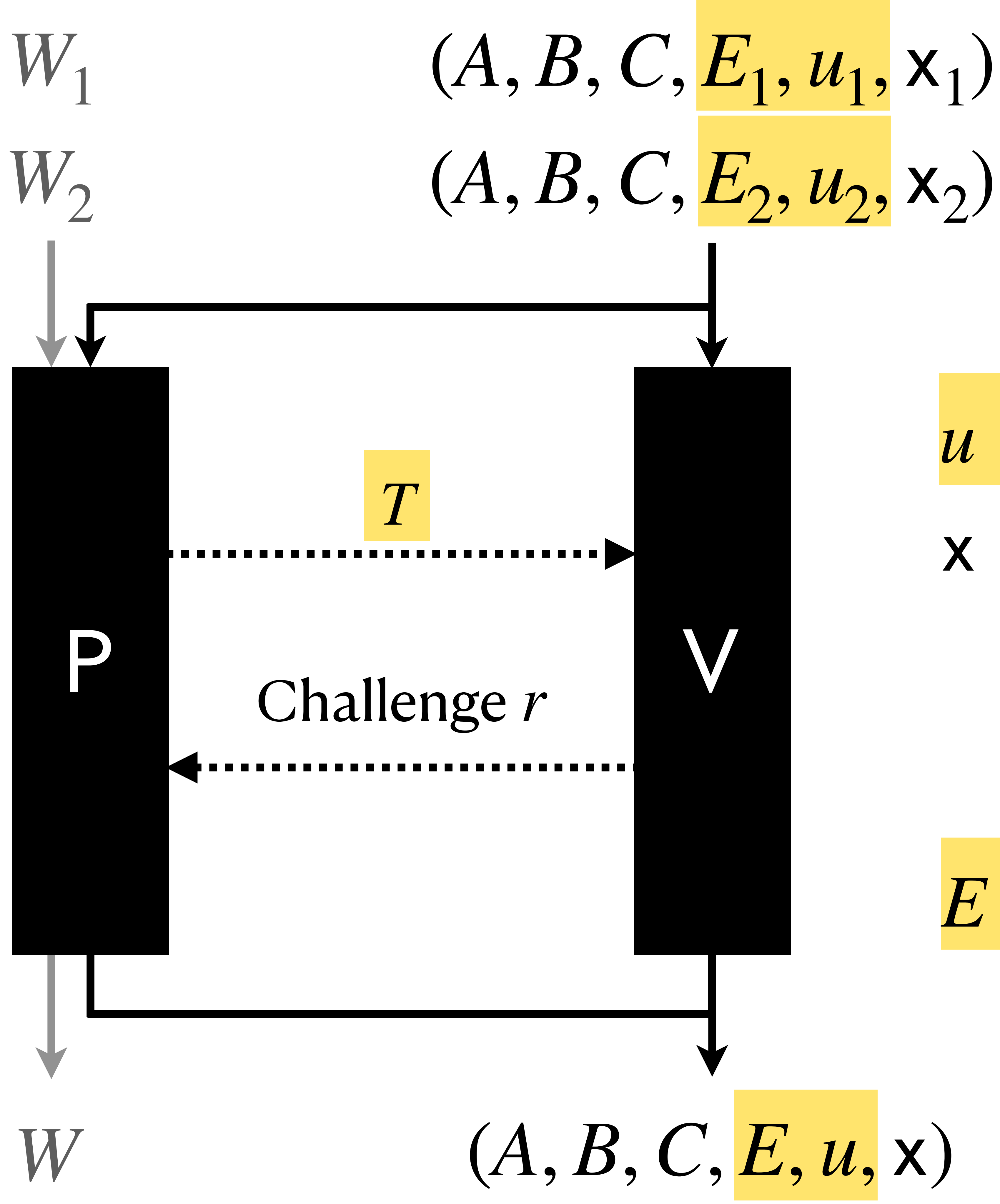
$$E \leftarrow E_1 + r \cdot T + r^2 \cdot E_2$$

Folding Scheme for Relaxed R1CS (Almost)

$$T \leftarrow AZ_1 \circ BZ_2 + AZ_2 \circ BZ_1 - u_1 \cdot CZ_2 - u_2 \cdot CZ_1$$

$$W \leftarrow W_1 + r \cdot W_2$$

$$E \leftarrow E_1 + r \cdot T + r^2 \cdot E_2$$



$$u \leftarrow u_1 + r \cdot u_2$$

$$\mathsf{x} \leftarrow \mathsf{x}_1 + r \cdot \mathsf{x}_2$$

$$E \leftarrow E_1 + r \cdot T + r^2 \cdot E_2$$

Problem: Verifier cannot enforce that prover folds W correctly

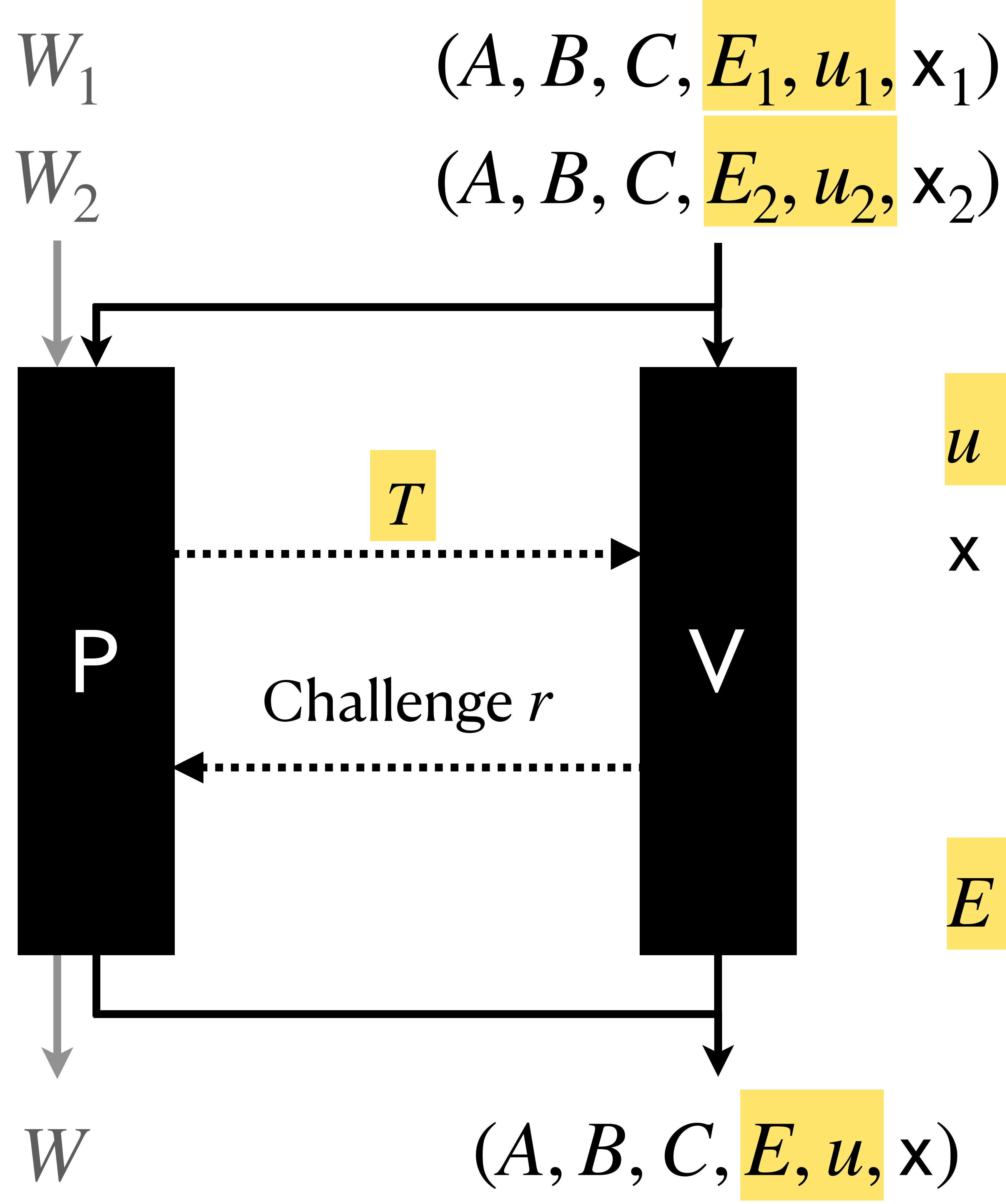
Folding Scheme for Relaxed R1CS (Almost)

Problem: Because $|E| = O(|W|)$ the verifier is not succinct

$$T \leftarrow AZ_1 \circ BZ_2 + AZ_2 \circ BZ_1 - u_1 \cdot CZ_2 - u_2 \cdot CZ_1$$

$$W \leftarrow W_1 + r \cdot W_2$$

$$E \leftarrow E_1 + r \cdot T + r^2 \cdot E_2$$



$$u \leftarrow u_1 + r \cdot u_2$$

$$\mathsf{x} \leftarrow \mathsf{x}_1 + r \cdot \mathsf{x}_2$$

$$E \leftarrow E_1 + r \cdot T + r^2 \cdot E_2$$

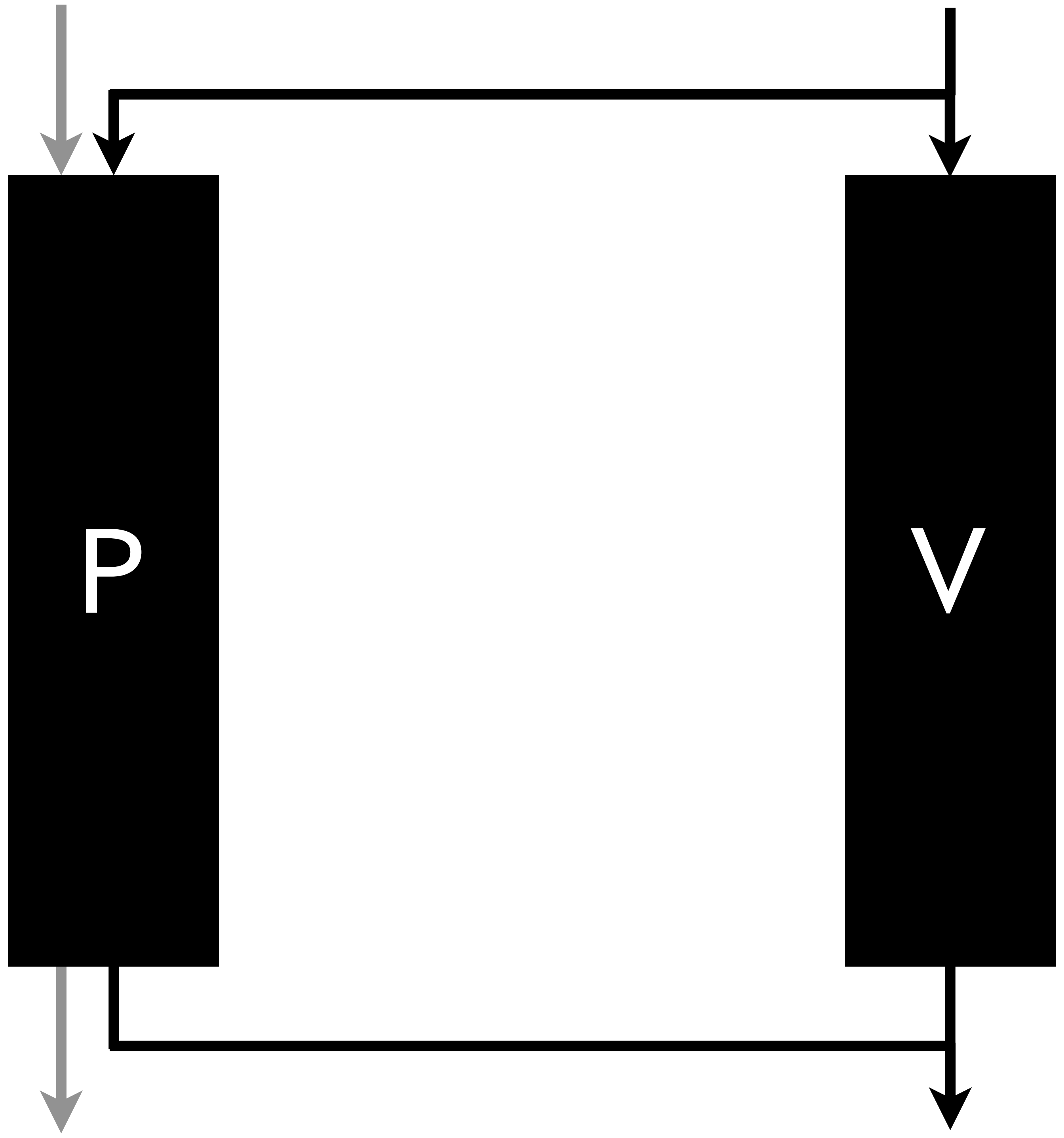
Problem: Verifier cannot enforce that prover folds W correctly

Folding Scheme for Relaxed R1CS from Additive Commitments

Treat (E, W) as part of the witness and store their commitments in the statement

$$(E_1, W_1) \qquad (A, B, C, \overline{E}_1, u_1, \overline{W}_1, \mathsf{x}_1)$$

$$(E_2, W_2) \qquad (A, B, C, \overline{E}_2, u_2, \overline{W}_2, \mathsf{x}_2)$$

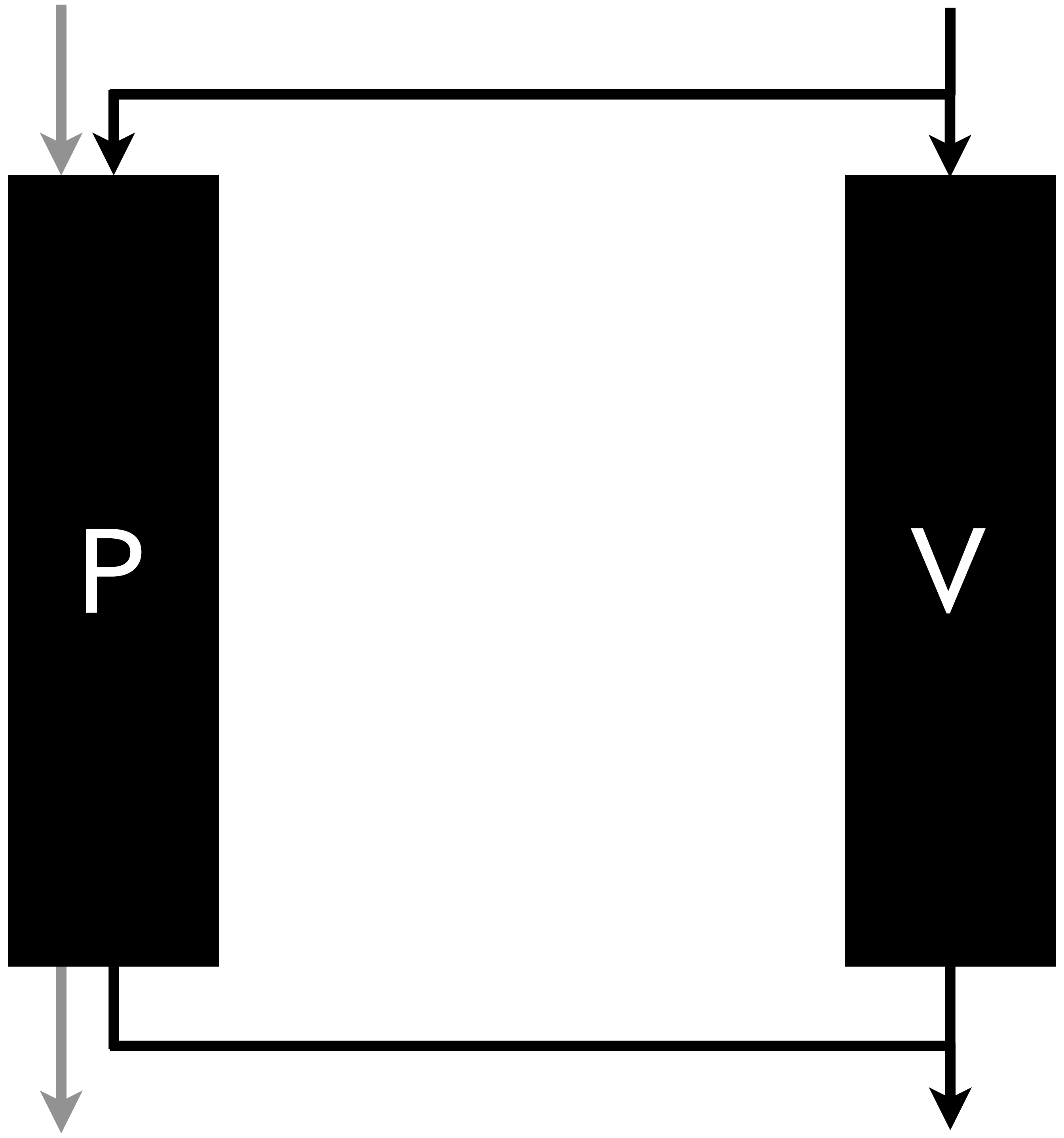


Folding Scheme for Relaxed R1CS from Additive Commitments

Treat (E, W) as part of the witness and store their commitments in the statement

$$(E_1, W_1) \qquad (A, B, C, \overline{E}_1, u_1, \overline{W}_1, \mathsf{x}_1)$$

$$(E_2, W_2) \qquad (A, B, C, \overline{E}_2, u_2, \overline{W}_2, \mathsf{x}_2)$$

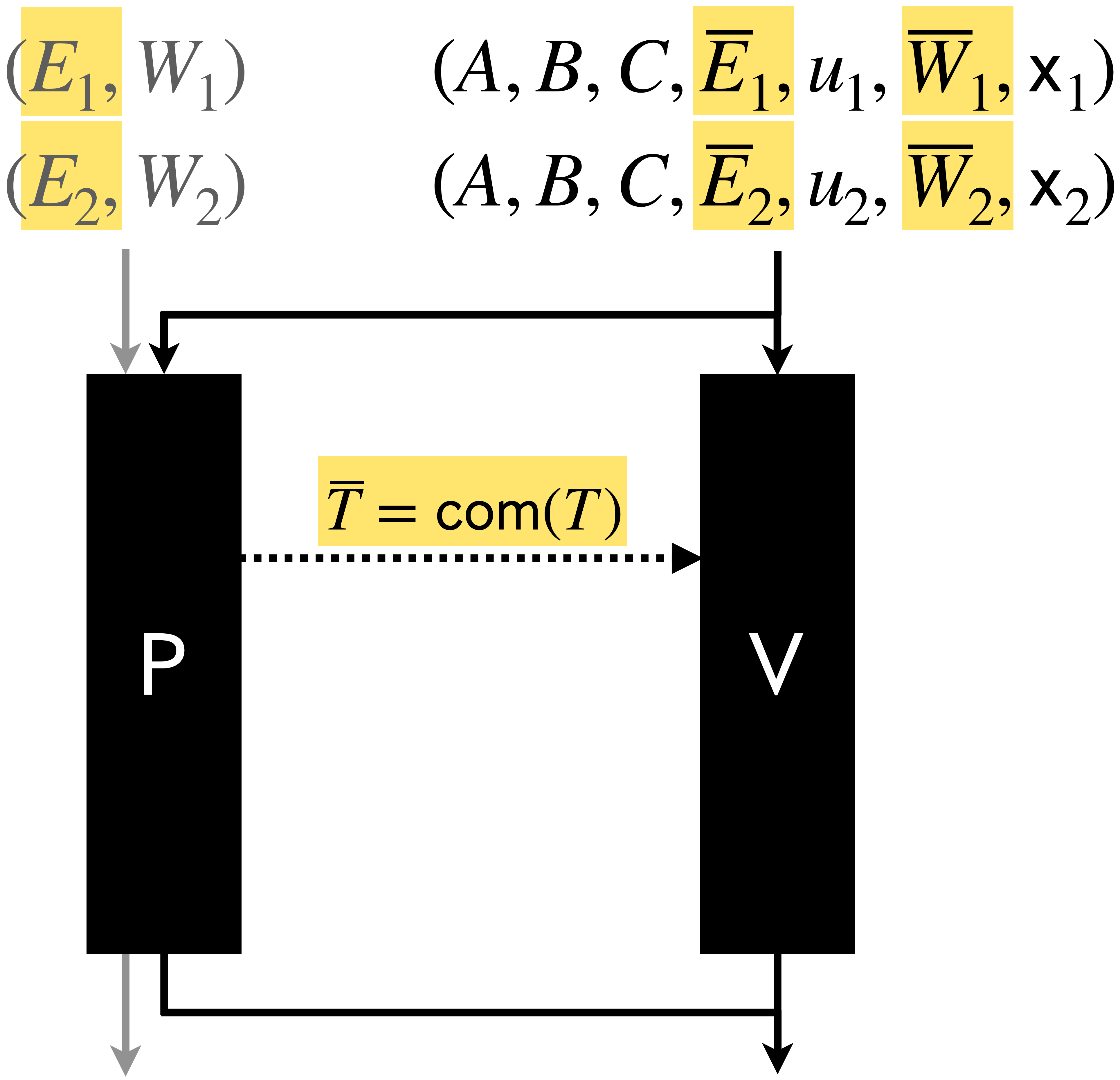


$$T \leftarrow AZ_1 \circ BZ_2 + AZ_2 \circ BZ_1 - u_1 \cdot CZ_2 - u_2 \cdot CZ_1$$

Folding Scheme for Relaxed R1CS from Additive Commitments

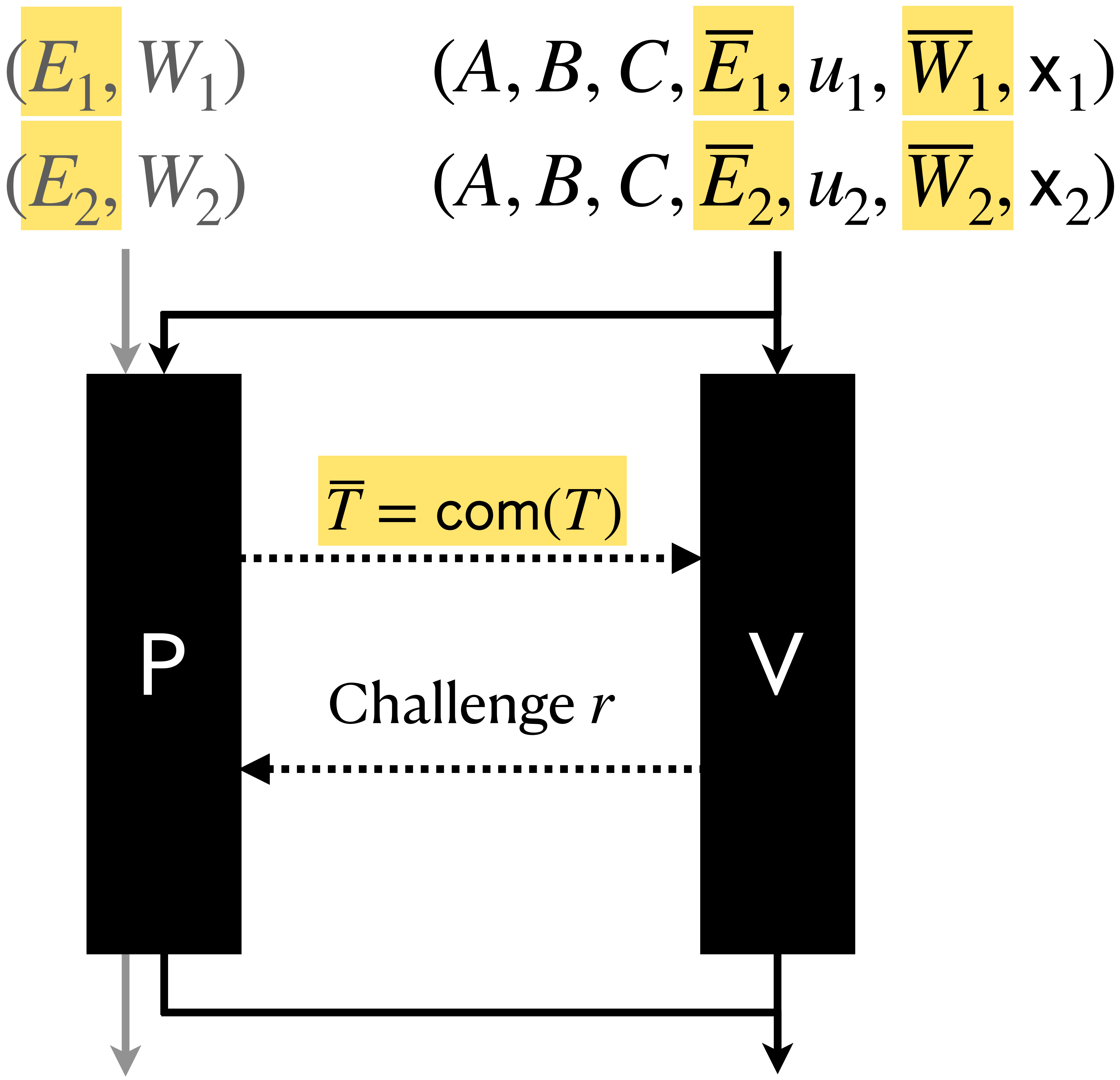
Treat (E, W) as part of the witness and store their commitments in the statement

$$T \leftarrow AZ_1 \circ BZ_2 + AZ_2 \circ BZ_1 - u_1 \cdot CZ_2 - u_2 \cdot CZ_1$$



Folding Scheme for Relaxed R1CS from Additive Commitments

Treat (E, W) as part of the witness and store their commitments in the statement



$$T \leftarrow AZ_1 \circ BZ_2 + AZ_2 \circ BZ_1$$
$$-u_1 \cdot CZ_2 - u_2 \cdot CZ_1$$

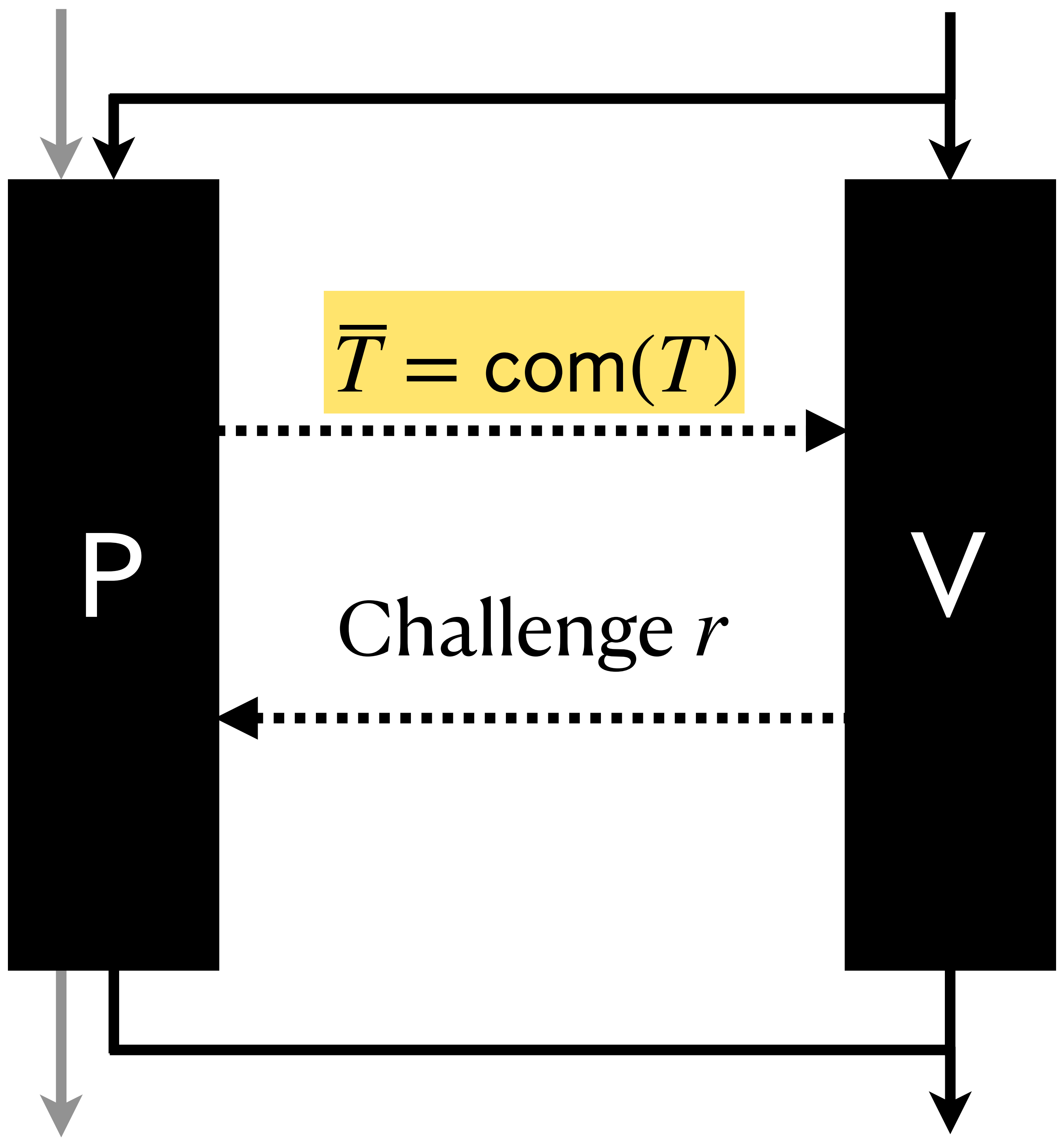
Folding Scheme for Relaxed R1CS from Additive Commitments

Treat (E, W) as part of the witness and store their commitments in the statement

$$(E_1, W_1) \qquad (A, B, C, \overline{E}_1, u_1, \overline{W}_1, \mathsf{x}_1)$$

$$(E_2, W_2) \qquad (A, B, C, \overline{E}_2, u_2, \overline{W}_2, \mathsf{x}_2)$$

$$T \leftarrow AZ_1 \circ BZ_2 + AZ_2 \circ BZ_1 - u_1 \cdot CZ_2 - u_2 \cdot CZ_1$$



$$u \leftarrow u_1 + r \cdot u_2$$

$$\mathsf{x} \leftarrow \mathsf{x}_1 + r \cdot \mathsf{x}_2$$

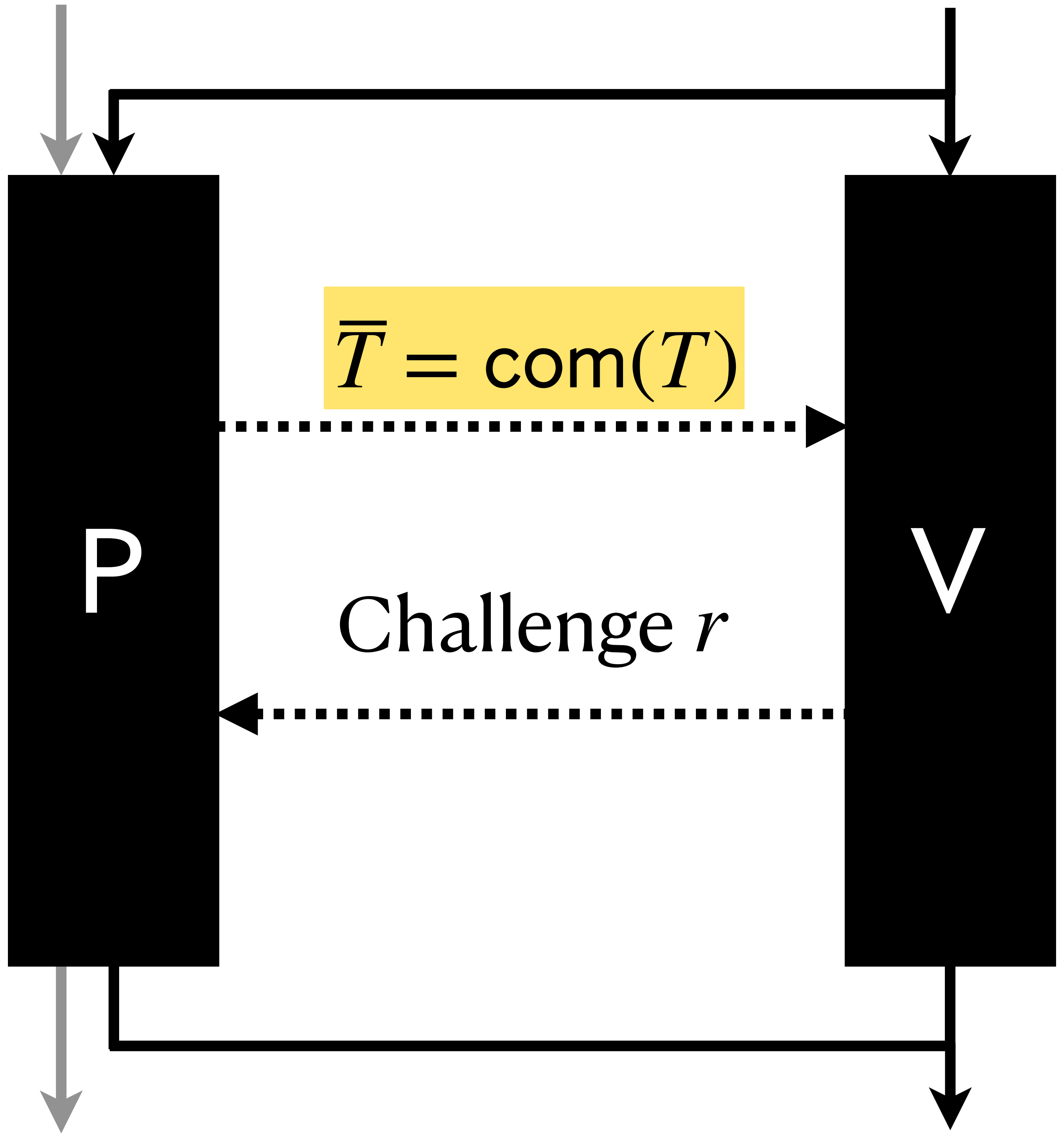
Folding Scheme for Relaxed R1CS from Additive Commitments

Treat (E, W) as part of the witness and store their commitments in the statement

(E_1, W_1) $\quad (A, B, C, \overline{E}_1, u_1, \overline{W}_1, \mathsf{x}_1)$

(E_2, W_2) $\quad (A, B, C, \overline{E}_2, u_2, \overline{W}_2, \mathsf{x}_2)$

$$T \leftarrow AZ_1 \circ BZ_2 + AZ_2 \circ BZ_1 - u_1 \cdot CZ_2 - u_2 \cdot CZ_1$$



$$u \leftarrow u_1 + r \cdot u_2$$

$$\mathsf{x} \leftarrow \mathsf{x}_1 + r \cdot \mathsf{x}_2$$

$$\overline{W} \leftarrow \overline{W}_1 + r \cdot \overline{W}_2$$

$$\overline{E} \leftarrow \overline{E}_1 + r \cdot \overline{T} + r^2 \cdot \overline{E}_2$$

Folding Scheme for Relaxed R1CS from Additive Commitments

Treat (E, W) as part of the witness and store their commitments in the statement

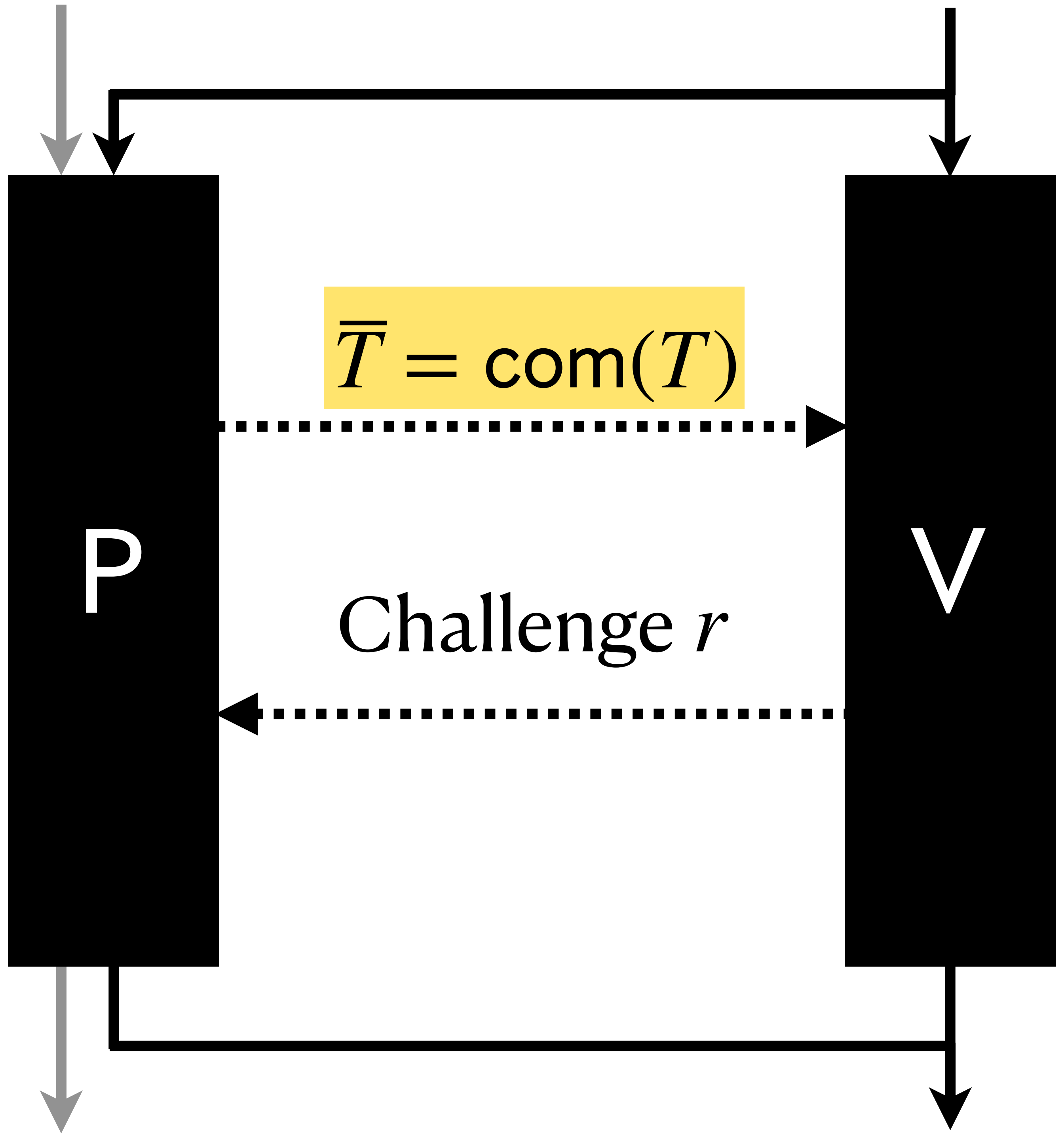
$$(E_1, W_1) \qquad (A, B, C, \overline{E}_1, u_1, \overline{W}_1, \mathsf{x}_1)$$

$$(E_2, W_2) \qquad (A, B, C, \overline{E}_2, u_2, \overline{W}_2, \mathsf{x}_2)$$

$$T \leftarrow AZ_1 \circ BZ_2 + AZ_2 \circ BZ_1 - u_1 \cdot CZ_2 - u_2 \cdot CZ_1$$

$$W \leftarrow W_1 + r \cdot W_2$$

$$E \leftarrow E_1 + r \cdot T + r^2 \cdot E_2$$



$$u \leftarrow u_1 + r \cdot u_2$$

$$\mathsf{x} \leftarrow \mathsf{x}_1 + r \cdot \mathsf{x}_2$$

$$\overline{W} \leftarrow \overline{W}_1 + r \cdot \overline{W}_2$$

$$\overline{E} \leftarrow \overline{E}_1 + r \cdot \overline{T} + r^2 \cdot \overline{E}_2$$

Folding Scheme for Relaxed R1CS from Additive Commitments

Treat (E, W) as part of the witness and store their commitments in the statement

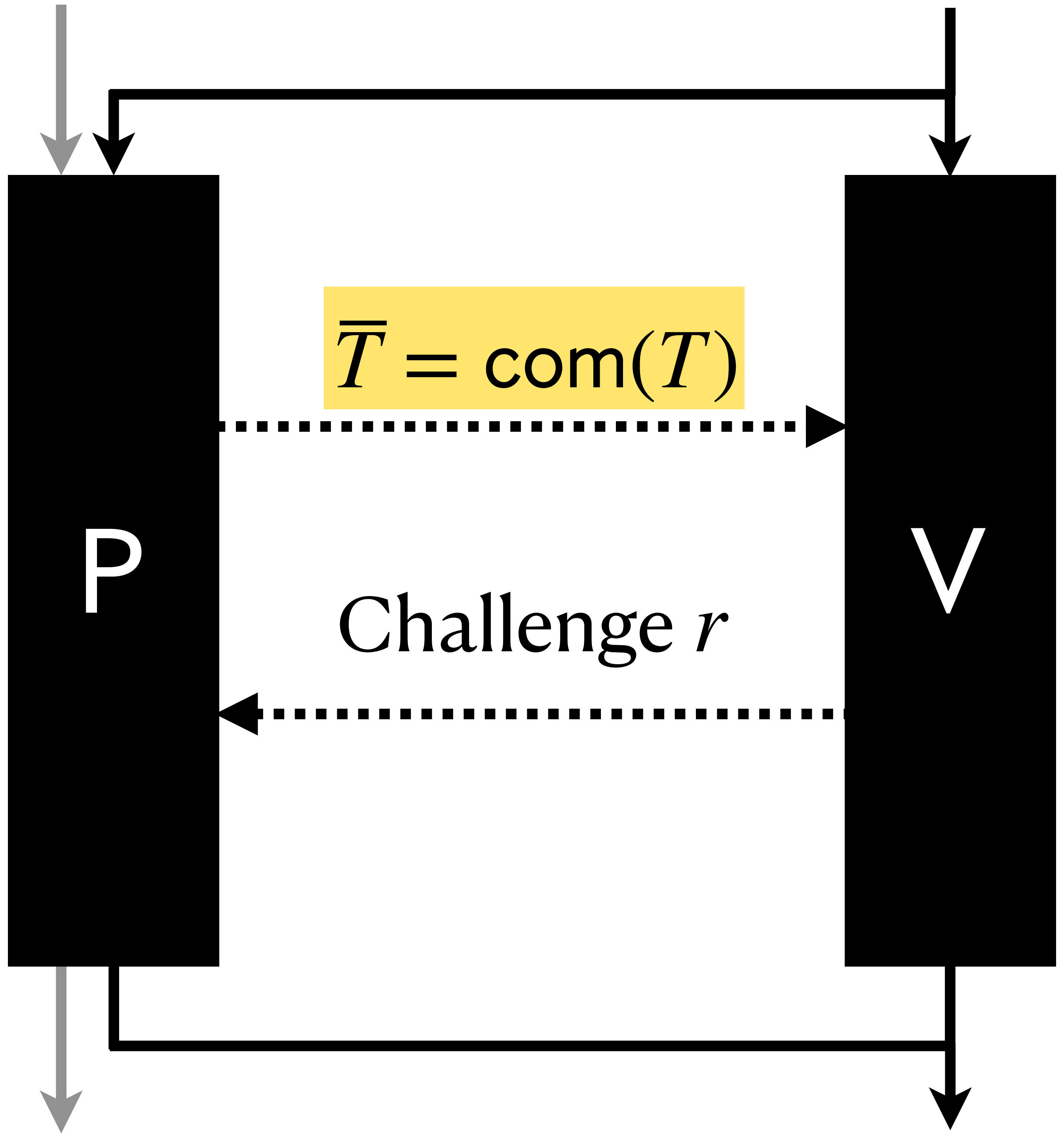
(E_1, W_1) $\quad$ $(A, B, C, \overline{E}_1, u_1, \overline{W}_1, \mathsf{x}_1)$

(E_2, W_2) $\quad$ $(A, B, C, \overline{E}_2, u_2, \overline{W}_2, \mathsf{x}_2)$

$$T \leftarrow AZ_1 \circ BZ_2 + AZ_2 \circ BZ_1 - u_1 \cdot CZ_2 - u_2 \cdot CZ_1$$

$$W \leftarrow W_1 + r \cdot W_2$$

$$E \leftarrow E_1 + r \cdot T + r^2 \cdot E_2$$



$$u \leftarrow u_1 + r \cdot u_2$$

$$\mathsf{x} \leftarrow \mathsf{x}_1 + r \cdot \mathsf{x}_2$$

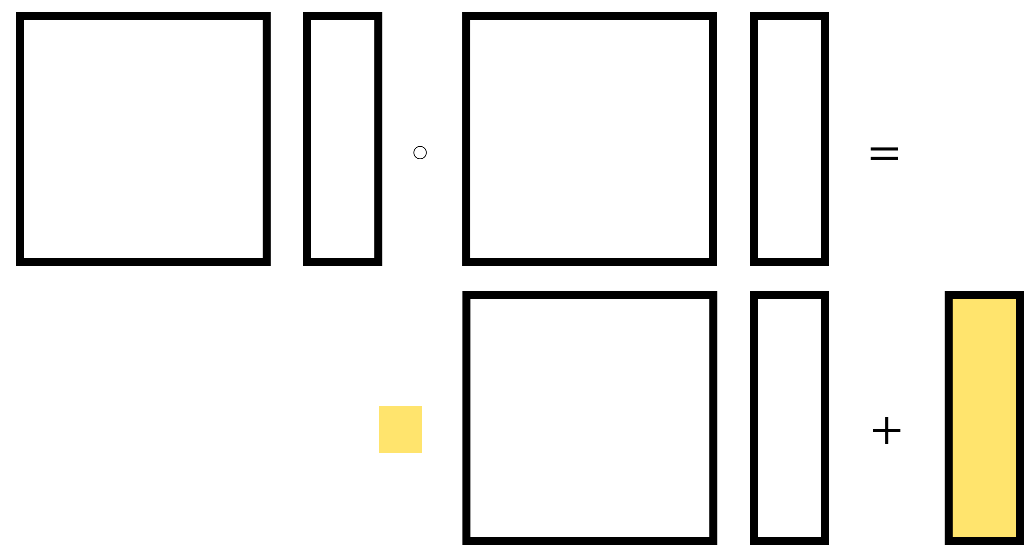
$$\overline{W} \leftarrow \overline{W}_1 + r \cdot \overline{W}_2$$

$$\overline{E} \leftarrow \overline{E}_1 + r \cdot \overline{T} + r^2 \cdot \overline{E}_2$$

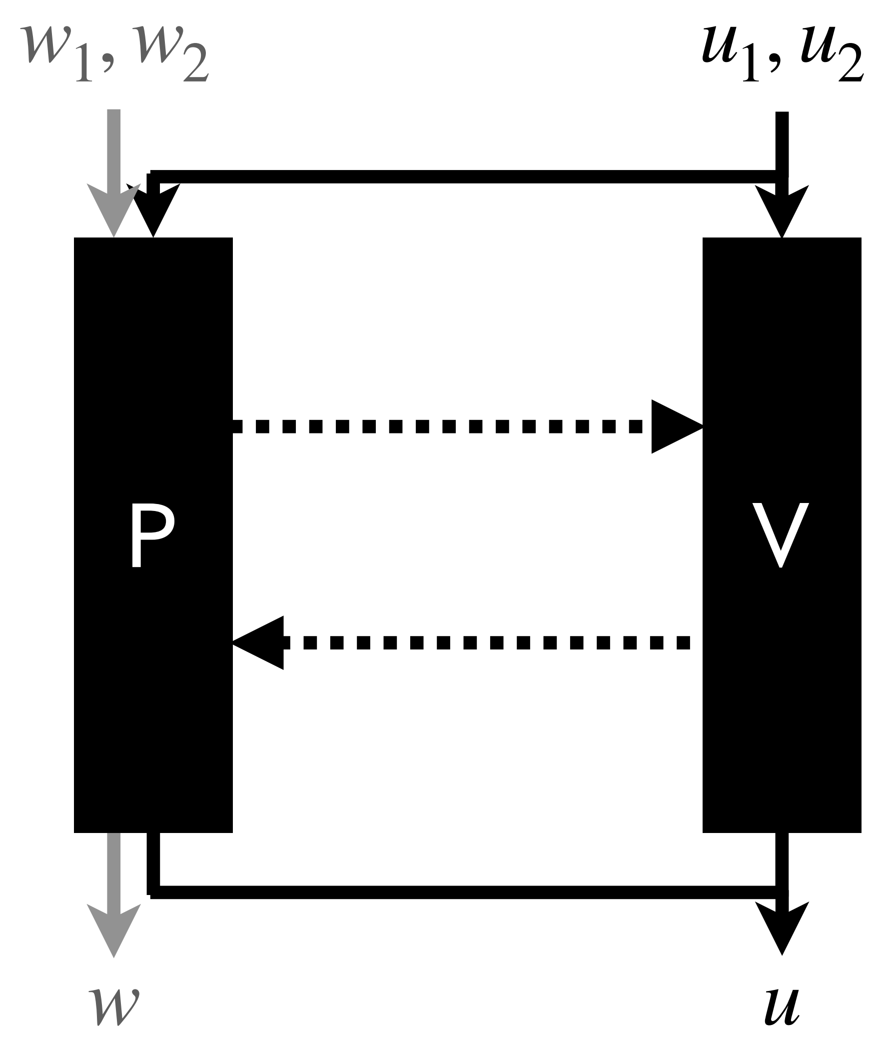
(E, W) $\quad$ $(A, B, C, \overline{E}, u, \overline{W}, \mathsf{x})$

Summary

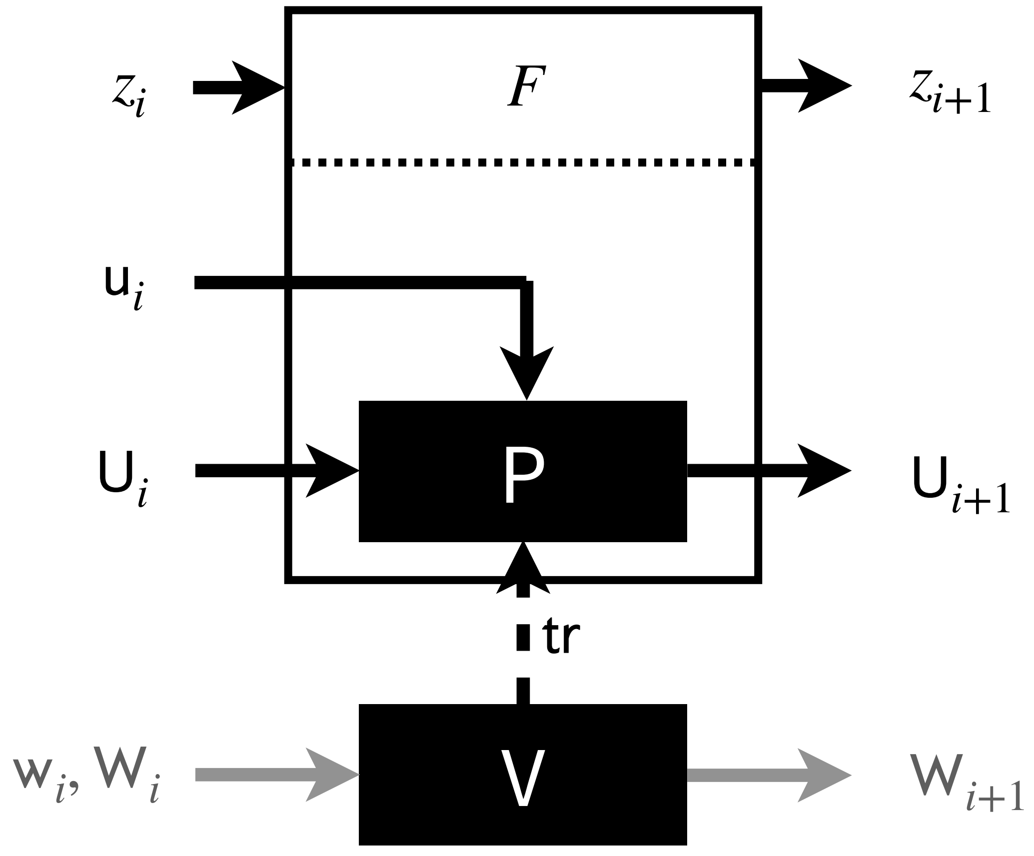
We design a folding-friendly variant of R1CS, Relaxed R1CS



We construct a folding scheme for Relaxed R1CS



We construct IVC using our folding scheme for Relaxed R1CS



Our techniques result in a recursive zkSNARK with state-of-the-art efficiency

eprint.iacr.org/2021/370

github.com/Microsoft/Nova

akothapalli@cmu.edu

References

[Val08] Valiant. Incrementally Verifiable Computation or Proofs of Knowledge imply Time/Space Efficiency

[BCTV14] Ben-Sasson, Chiesa, Tromer, Virza. Scalable Zero-Knowledge via Cycles of Elliptic Curves

[BGH19] Bowe, Green, Hopwood. Halo: Recursive Proof Composition without a Trusted Setup

[BCLMS20] Bünz, Chiesa, Lin, Mishra, Spooner. Proof Carrying Data without Succinct Arguments

[BDFG20] Boneh, Drake, Fisch, Gabizon. Halo Infinite: Recursive zkSNARKs from any Additive Polynomial Commitment Scheme

[CCS22] Chen, Chiesa, Spooner. On Succinct Non-Interactive Arguments in Relativized Worlds.

[BBBF19] Boneh, Bonneau, Bünz, Fisch. Verifiable Delay Functions

[GPR21] Goldberg, Papini, Riabzev. Cairo – a Turing complete STARK-friendly CPU Architecture