Transparent Time- and Space-Efficient Arguments From Groups of Unknown Order

Justin Holmgren
NTT Research

Alex Block
Purdue

Alon Rosen
IDC Herzliya

Ron Rothblum
Technion

Pratik Soni
CMU

Pre-Quantum Cryptography with Lattices

Justin Holmgren
NTT Research

Alex Block
Purdue

Alon Rosen
IDC Herzliya

Ron Rothblum
Technion

Pratik Soni
CMU

Please make me "Post-".

Pre-Quantum Cryptography with Lattices

Justin Holmgren
NTT Research

Alex Block
Purdue

Alon Rosen
IDC Herzliya

Ron Rothblum
Technion

Pratik Soni
CMU

Interactive Arguments

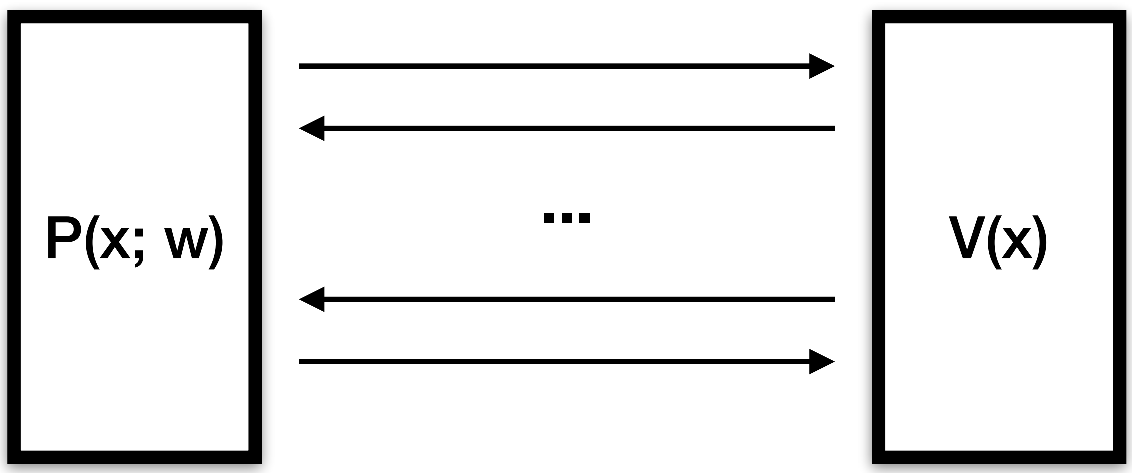
for an NP relation R with corresponding language L

Interactive Arguments

for an NP relation R with corresponding language L

Completeness:

For any $(x, w) \in R$,

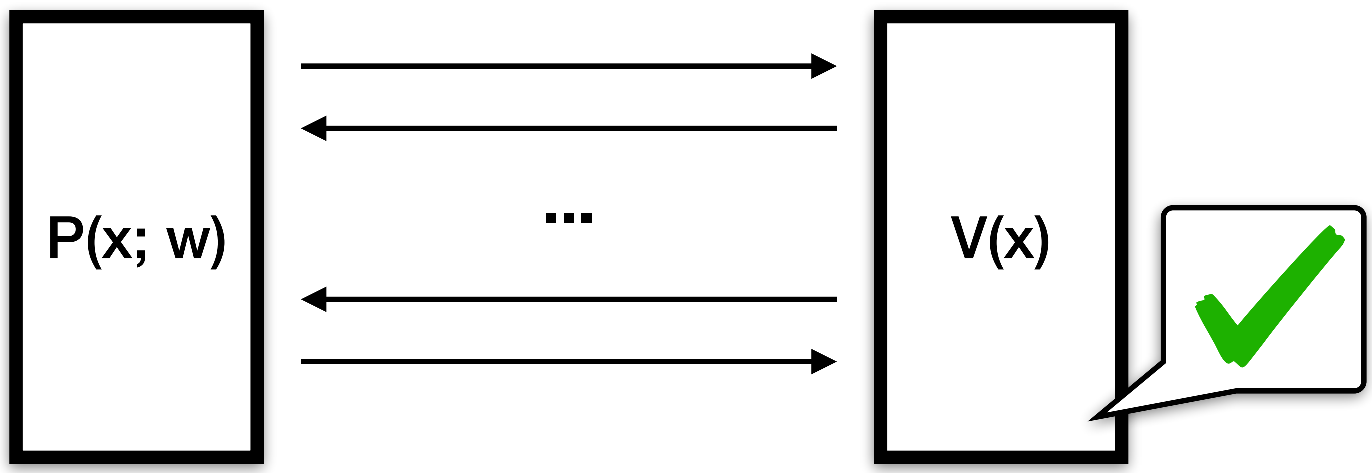


Interactive Arguments

for an NP relation R with corresponding language L

Completeness:

For any $(x, w) \in R$,

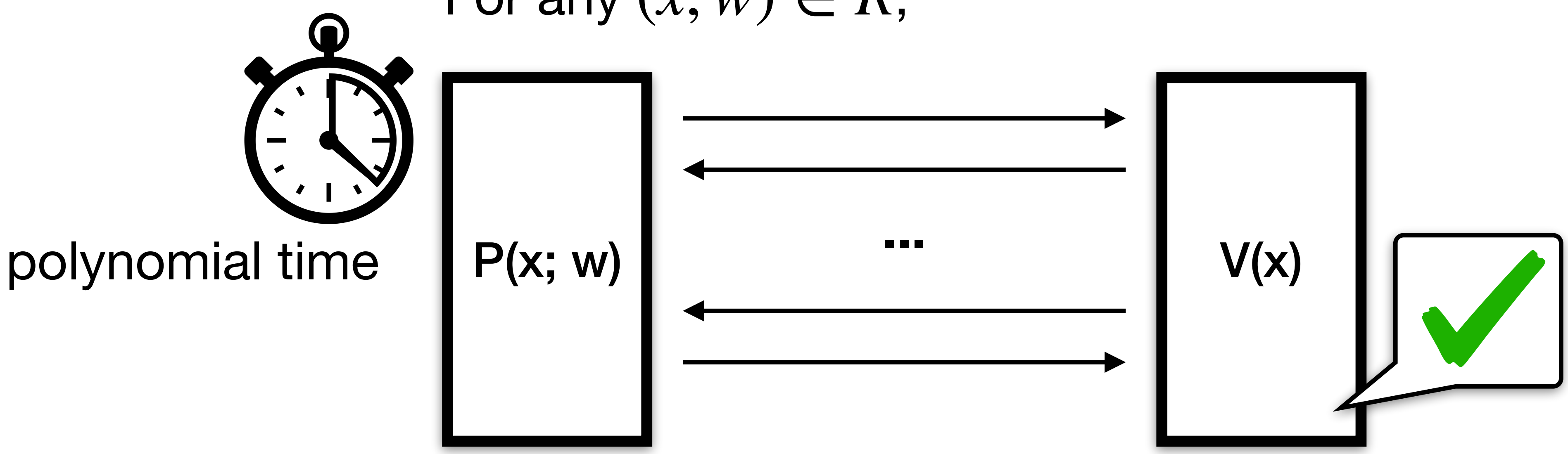


Interactive Arguments

for an NP relation R with corresponding language L

Completeness:

For any $(x, w) \in R$,

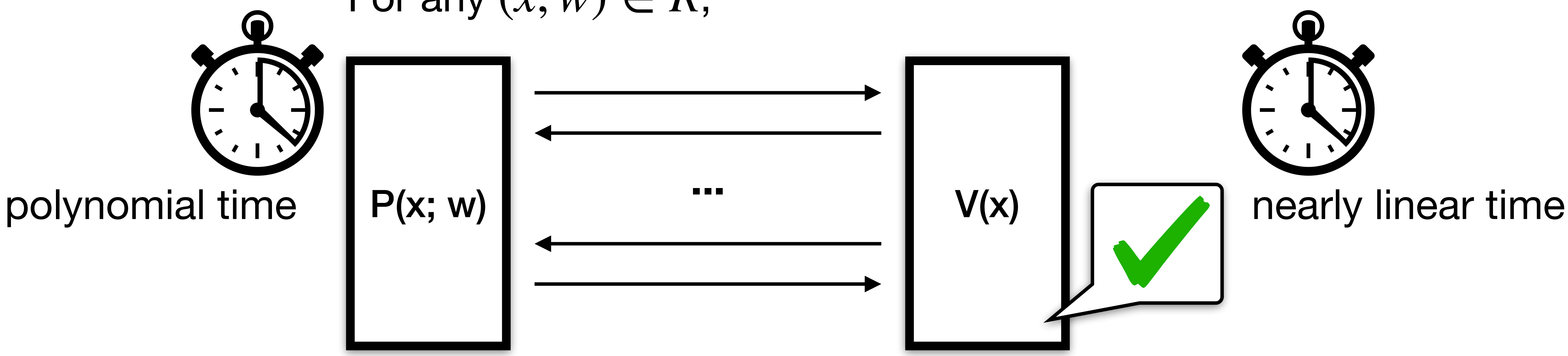


Interactive Arguments

for an NP relation R with corresponding language L

Completeness:

For any $(x, w) \in R$,

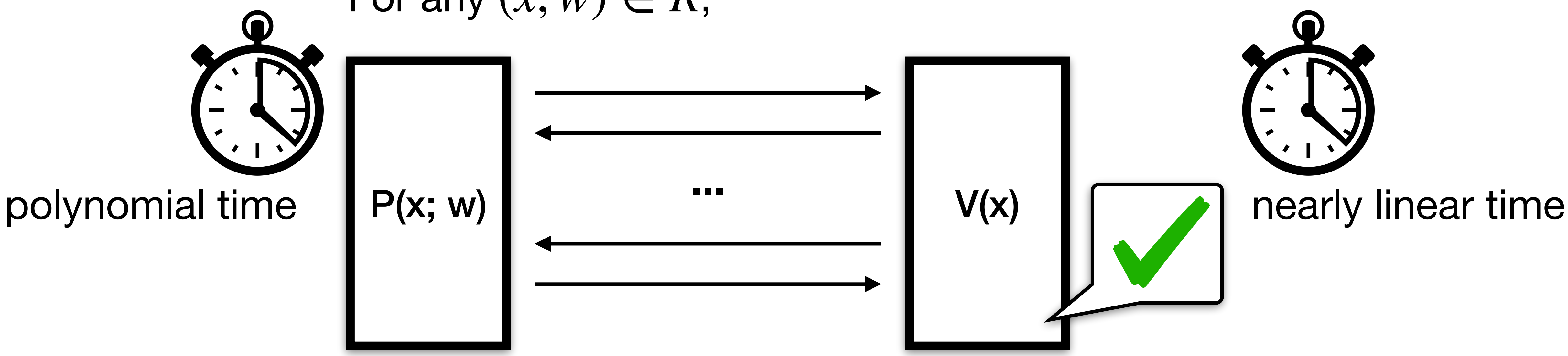


Interactive Arguments

for an NP relation R with corresponding language L

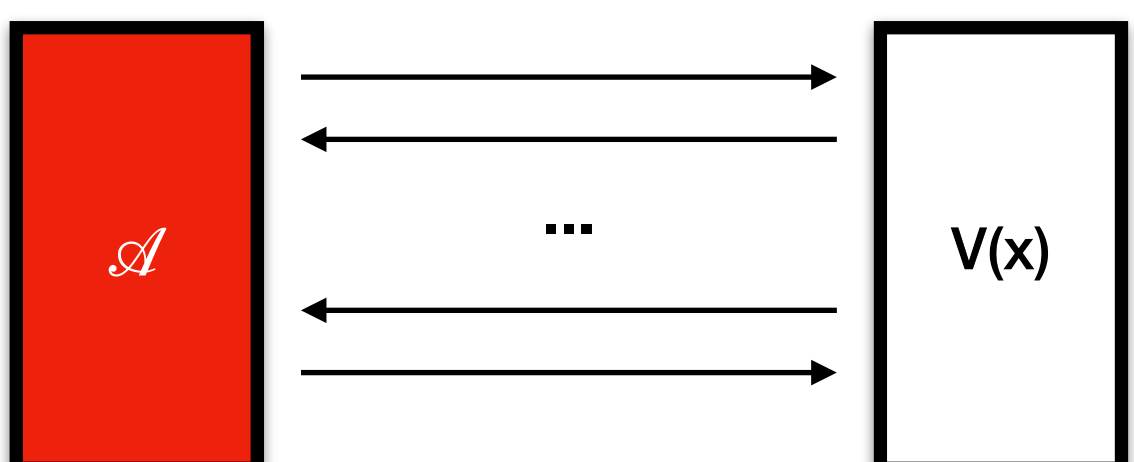
Completeness:

For any $(x, w) \in R$,



Soundness:

For any $x \notin L$, poly-size adversary $\mathscr{A}$,

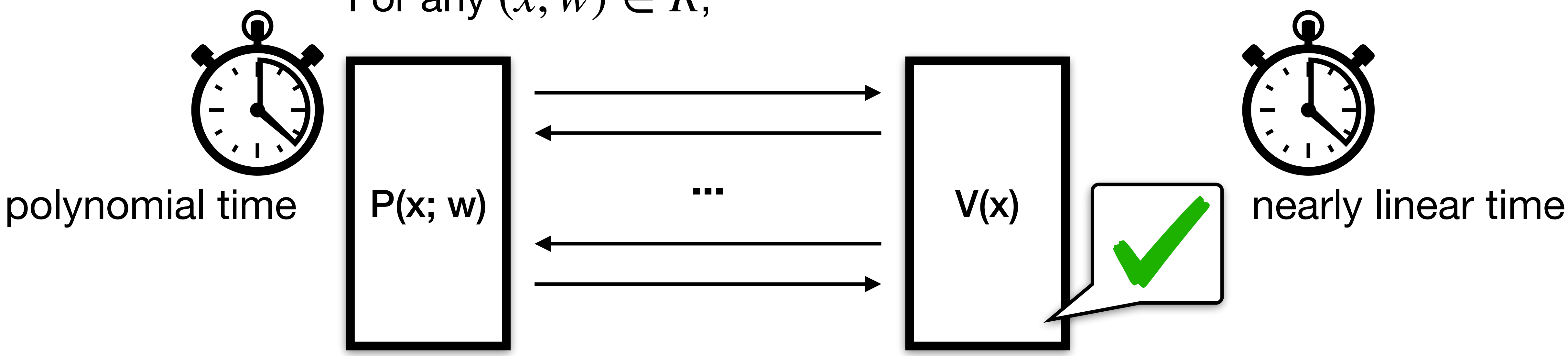


Interactive Arguments

for an NP relation R with corresponding language L

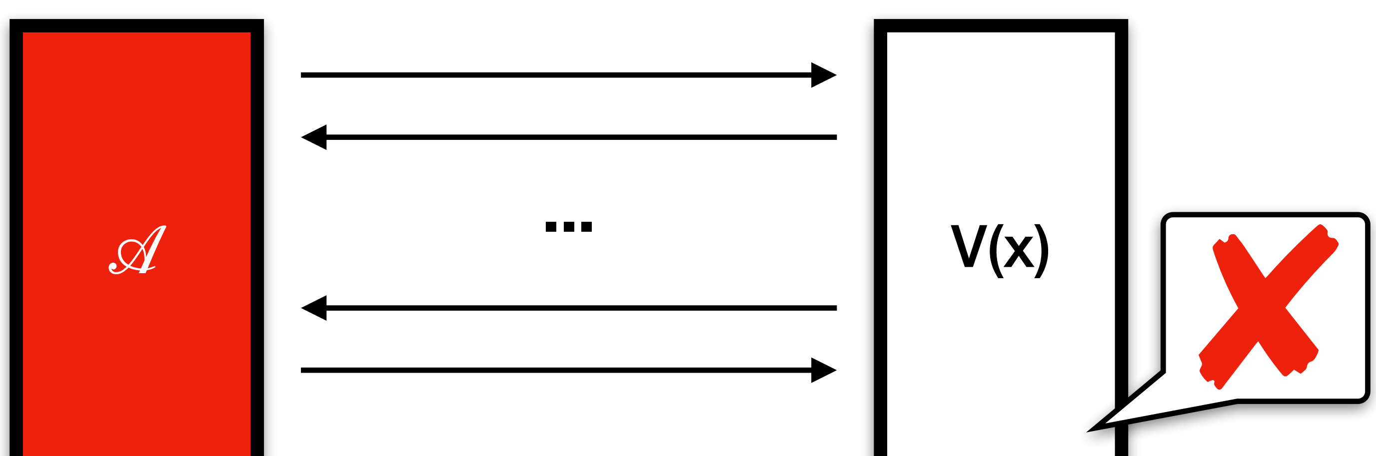
Completeness:

For any $(x, w) \in R$,



Soundness:

For any $x \notin L$, poly-size adversary $\mathscr{A}$,



Two Desirable Properties for Interactive Arguments

Two Desirable Properties for Interactive Arguments

Public-Coin Verification:

Two Desirable Properties for Interactive Arguments

Public-Coin Verification:

- Uniformly random verifier messages

Two Desirable Properties for Interactive Arguments

Public-Coin Verification:

- Uniformly random verifier messages
- Acceptance depends deterministically on transcript

Two Desirable Properties for Interactive Arguments

Public-Coin Verification:

Necessary for decentralized verification (e.g. in blockchains)

- Uniformly random verifier messages
- Acceptance depends deterministically on transcript

Two Desirable Properties for Interactive Arguments

Public-Coin Verification:

Necessary for decentralized verification (e.g. in blockchains)

- Uniformly random verifier messages
- Acceptance depends deterministically on transcript

Time- and Space-Efficient Prover:

Two Desirable Properties for Interactive Arguments

Public-Coin Verification:

Necessary for decentralized verification (e.g. in blockchains)

- Uniformly random verifier messages
- Acceptance depends deterministically on transcript

Time- and Space-Efficient Prover:

- If $(x; w) \stackrel{?}{\in} R$ is decidable in time T and space S, then prover runs in time $\approx T$ and space $\approx S$

Two Desirable Properties for Interactive Arguments

Public-Coin Verification:

Necessary for decentralized verification (e.g. in blockchains)

- Uniformly random verifier messages
- Acceptance depends deterministically on transcript

Time- and Space-Efficient Prover:

- If $(x; w) \overset{?}{\in} R$ is decidable in time T and space S, then prover runs in time $\approx T$ and space $\approx S$
- Space can be as much of a bottleneck as time, but is often overlooked

Prior Approaches for
Time- and Space-Efficient Proving

Prior Approaches for Time- and Space-Efficient Proving

Approach 1: Recursive Composition [Valiant '08, BCCT '12]

Prior Approaches for Time- and Space-Efficient Proving

Approach 1: Recursive Composition [Valiant '08, BCCT '12]

- Large concrete overheads due to non-black-box crypto

Prior Approaches for Time- and Space-Efficient Proving

Approach 1: Recursive Composition [Valiant '08, BCCT '12]

- Large concrete overheads due to non-black-box crypto (or if brave: algebraic hash functions [BGH19, BCMS20, COS20])

Prior Approaches for Time- and Space-Efficient Proving

Approach 1: Recursive Composition [Valiant '08, BCCT '12]

- Large concrete overheads due to non-black-box crypto (or if brave: algebraic hash functions [BGH19, BCMS20, COS20])
- Soundness relies on **exotic computational assumptions**

Prior Approaches for Time- and Space-Efficient Proving

Approach 1: Recursive Composition [Valiant '08, BCCT '12]

- Large concrete overheads due to non-black-box crypto (or if brave: algebraic hash functions [BGH19, BCMS20, COS20])
- Soundness relies on **exotic computational assumptions**

Approach 2: Compiling IOPs with space-efficient provers

Prior Approaches for Time- and Space-Efficient Proving

Approach 1: Recursive Composition [Valiant '08, BCCT '12]

- Large concrete overheads due to non-black-box crypto (or if brave: algebraic hash functions [BGH19, BCMS20, COS20])
- Soundness relies on **exotic computational assumptions**

Approach 2: Compiling IOPs with space-efficient provers

- Until now: space-preserving compilers produced **private-coin** arguments [Bitansky-Chiesa '12, B**H**RRS '20]

Prior Approaches for Time- and Space-Efficient Proving

Approach 1: Recursive Composition [Valiant '08, BCCT '12]

- Large concrete overheads due to non-black-box crypto (or if brave: algebraic hash functions [BGH19, BCMS20, COS20])
- Soundness relies on **exotic computational assumptions**

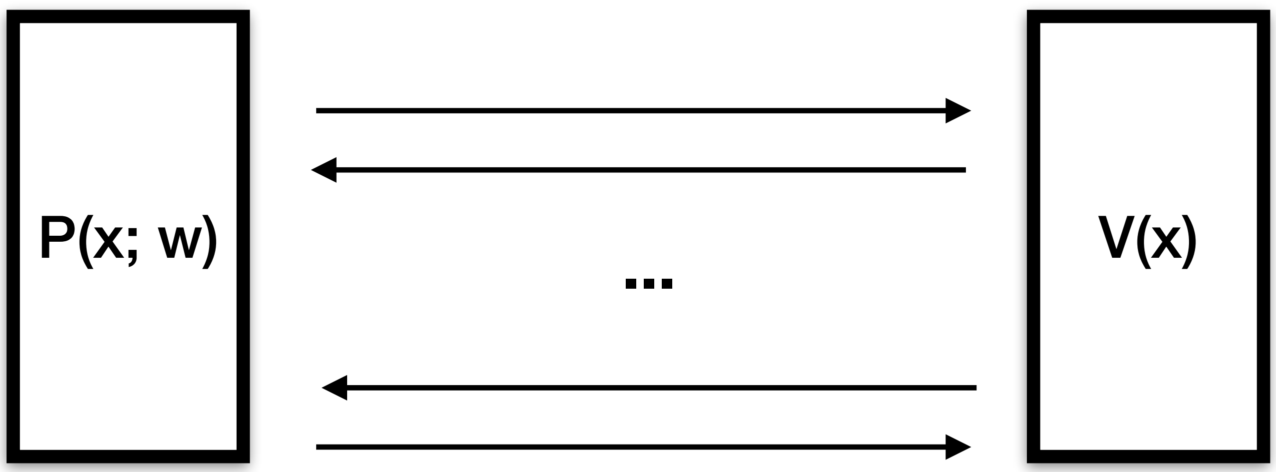
Approach 2: Compiling IOPs with space-efficient provers

- Until now: space-preserving compilers produced **private-coin** arguments [Bitansky-Chiesa '12, B**H**RRS '20]
- **This work**: **public-coin** arguments, based on a **simple & falsifiable** "hidden order" assumption

Compiling IOPs to Arguments

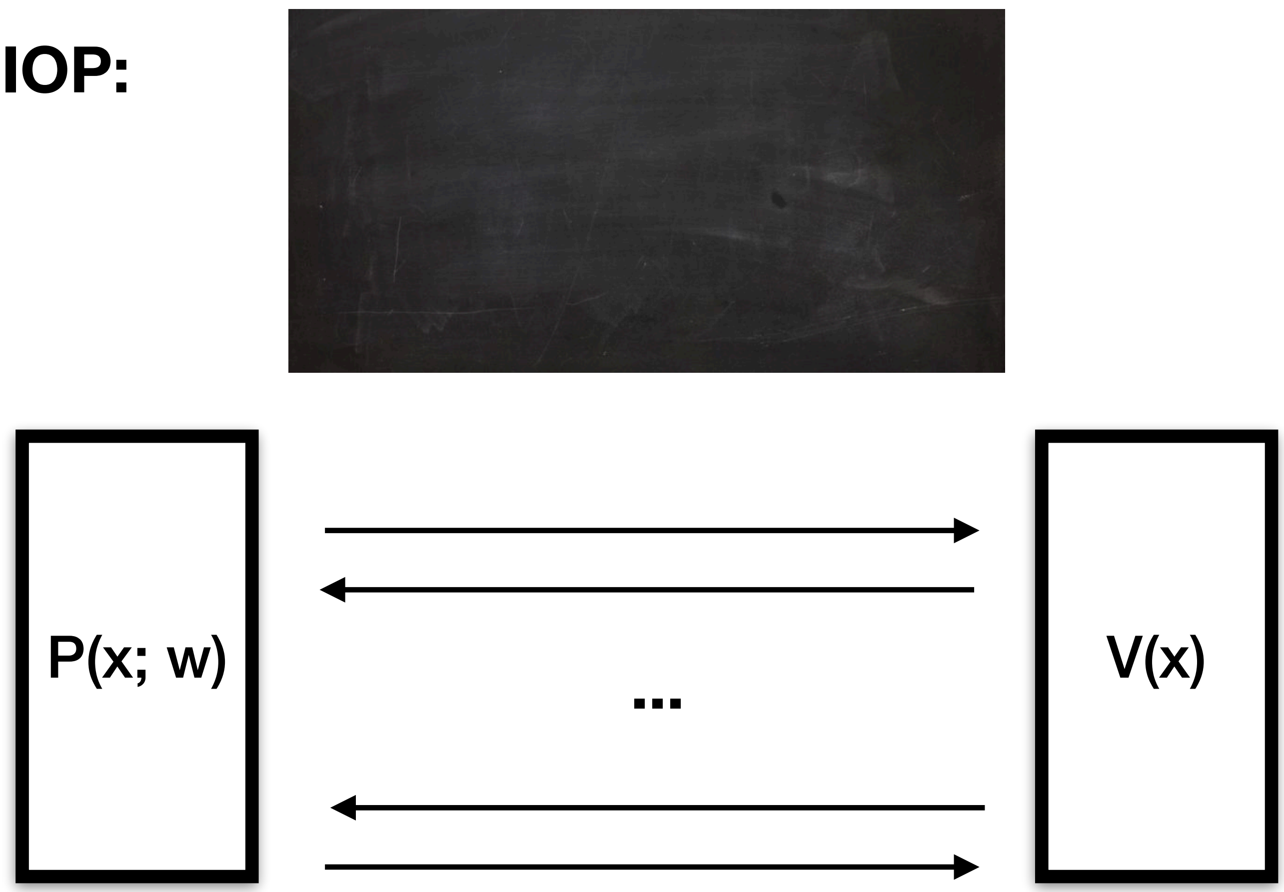
Compiling IOPs to Arguments

IOP:



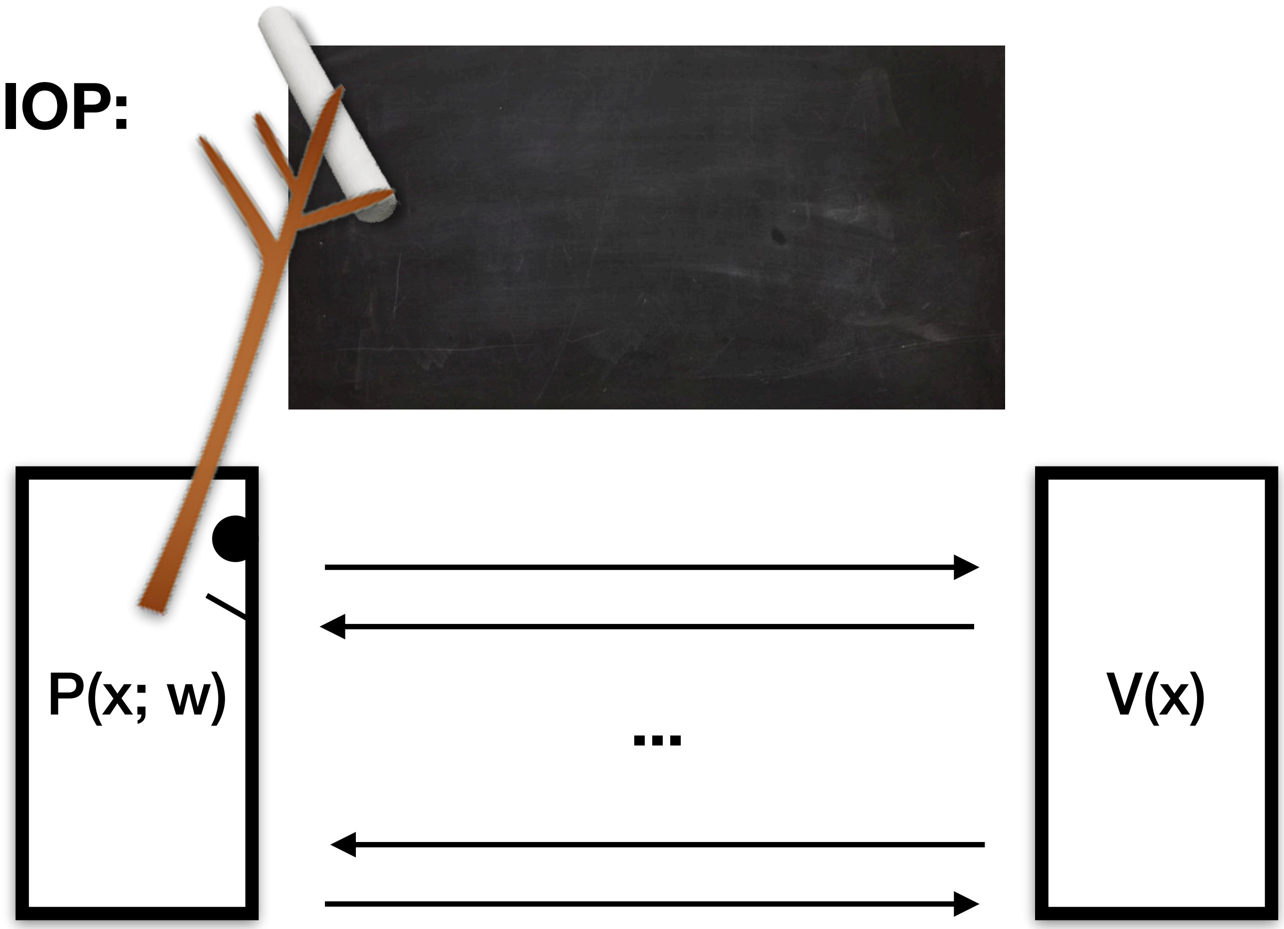
Compiling IOPs to Arguments

IOP:



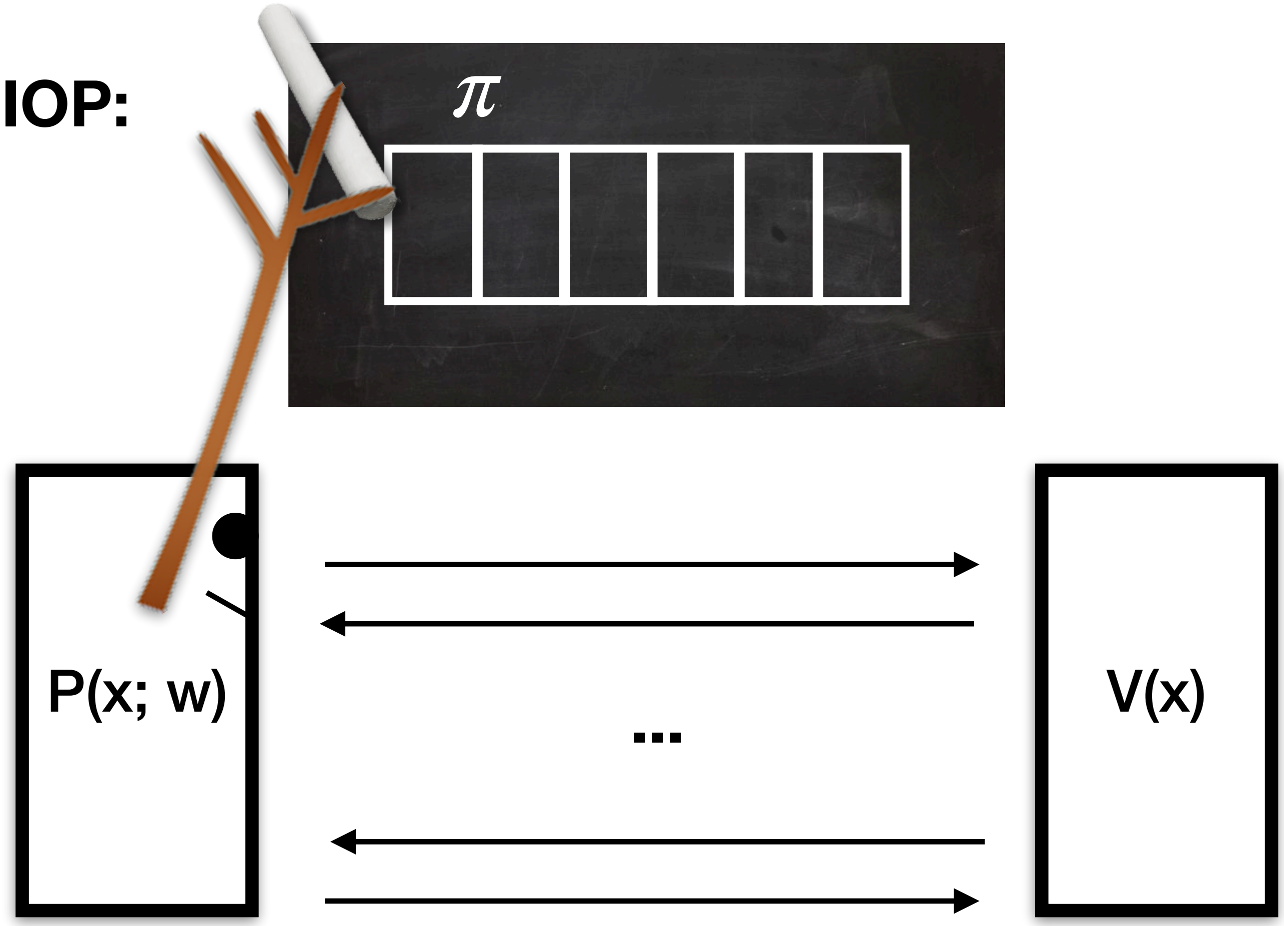
Compiling IOPs to Arguments

IOP:



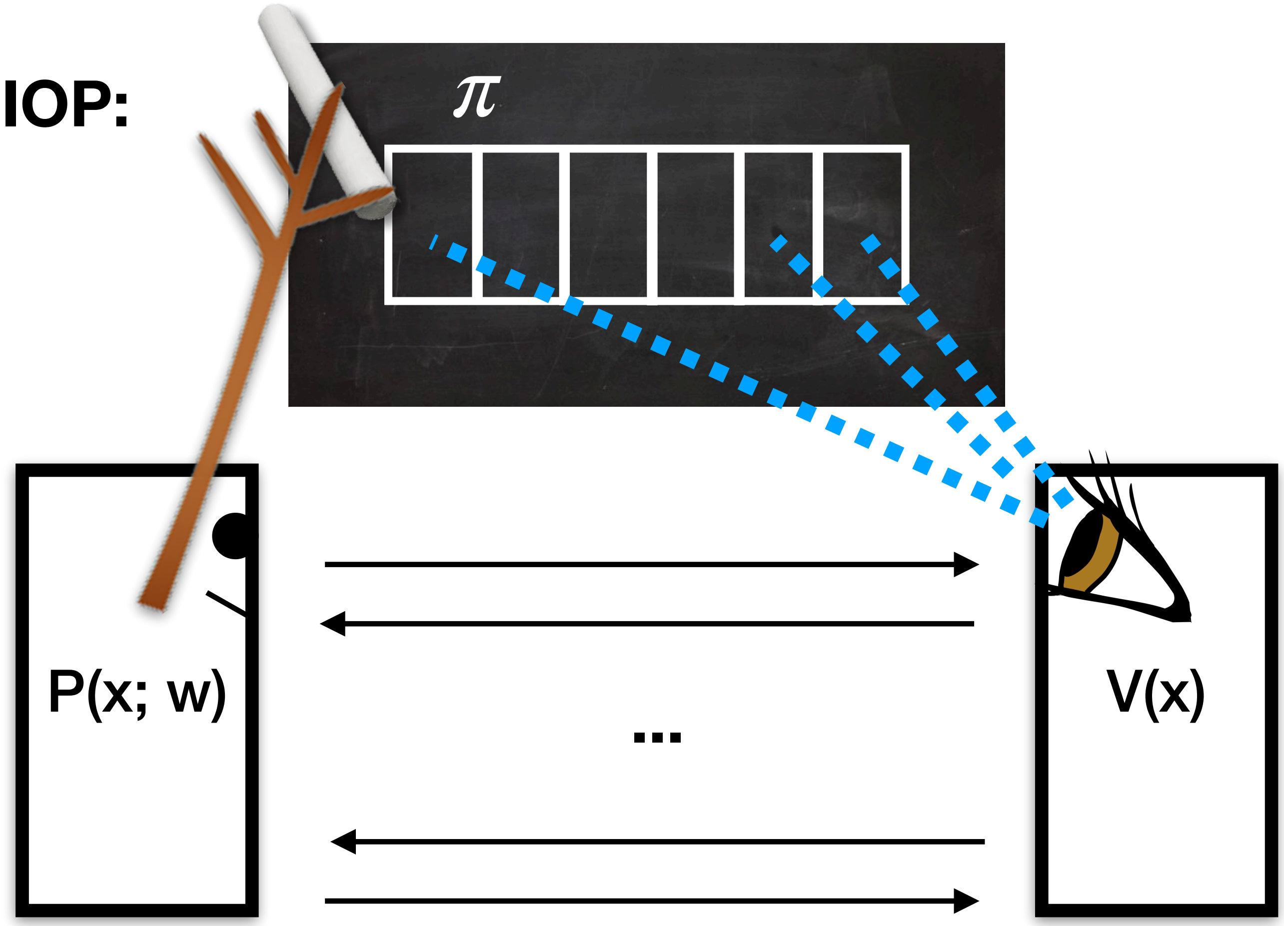
Compiling IOPs to Arguments

IOP:



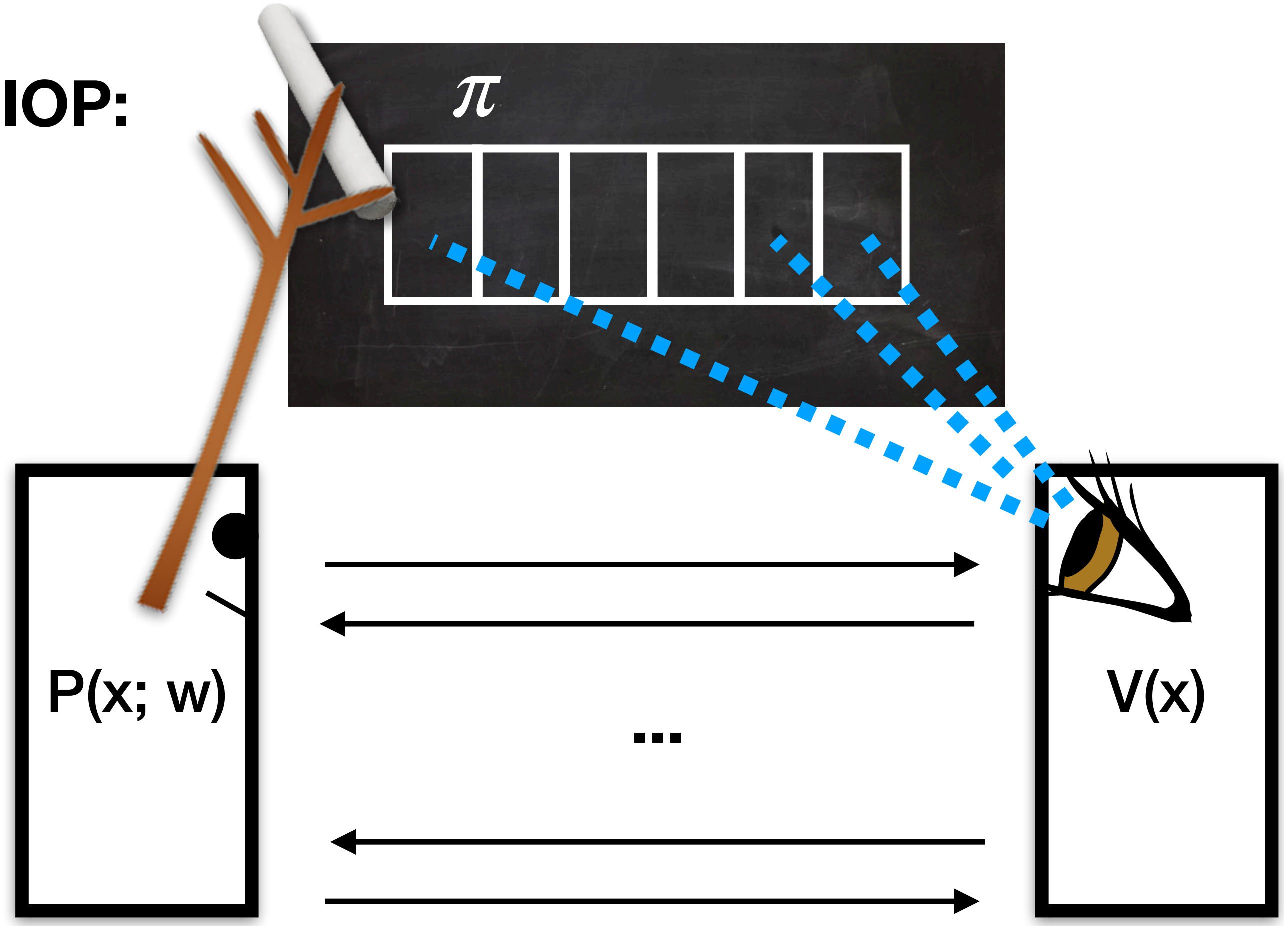
Compiling IOPs to Arguments

IOP:



Compiling IOPs to Arguments

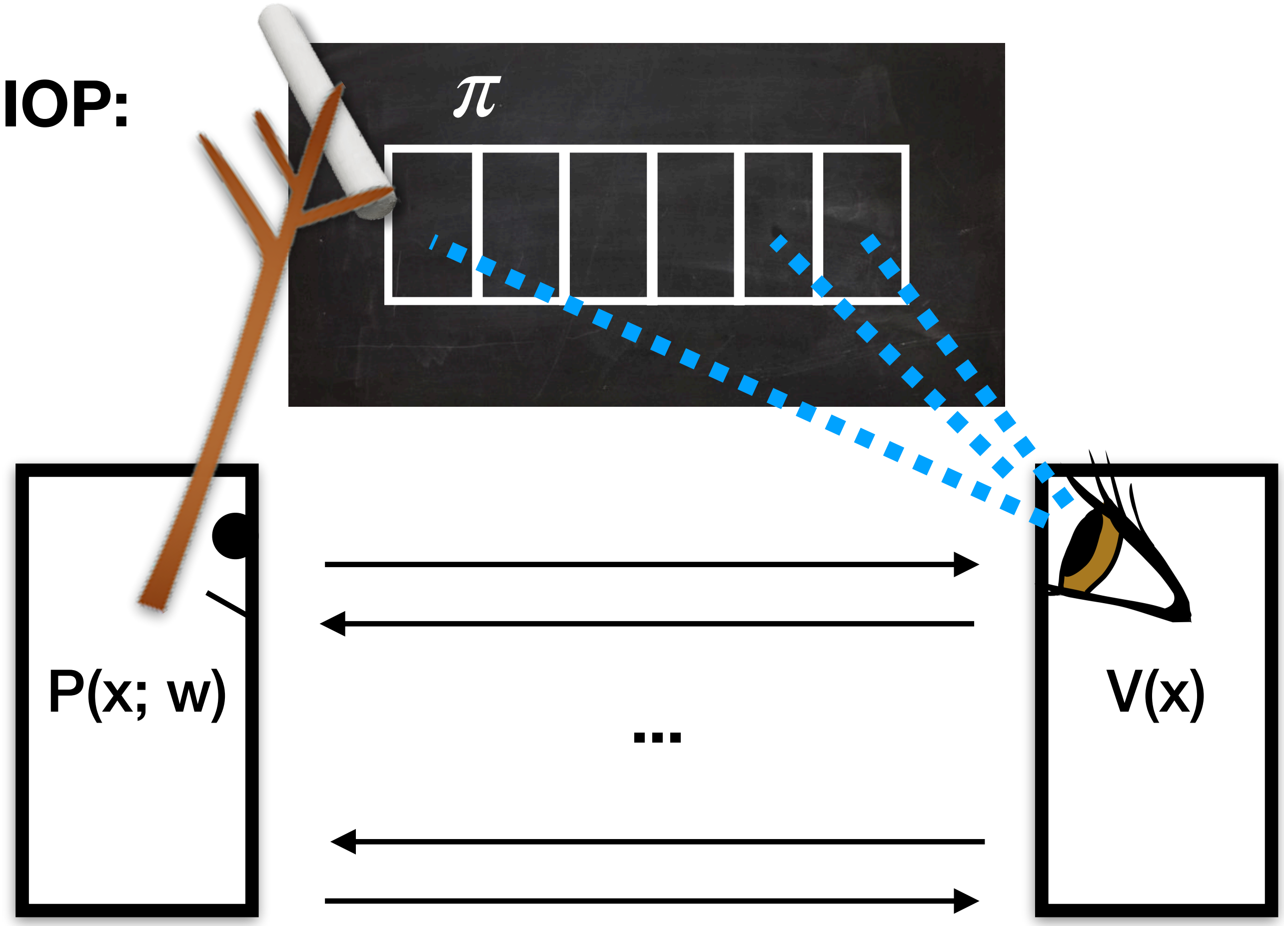
IOP:



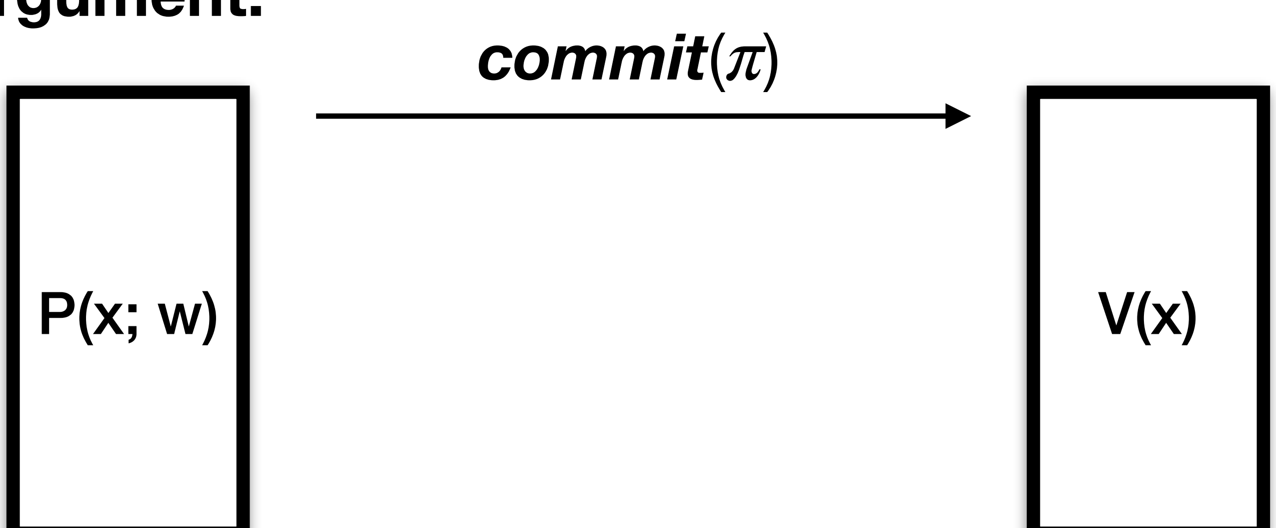
Argument:

Compiling IOPs to Arguments

IOP:

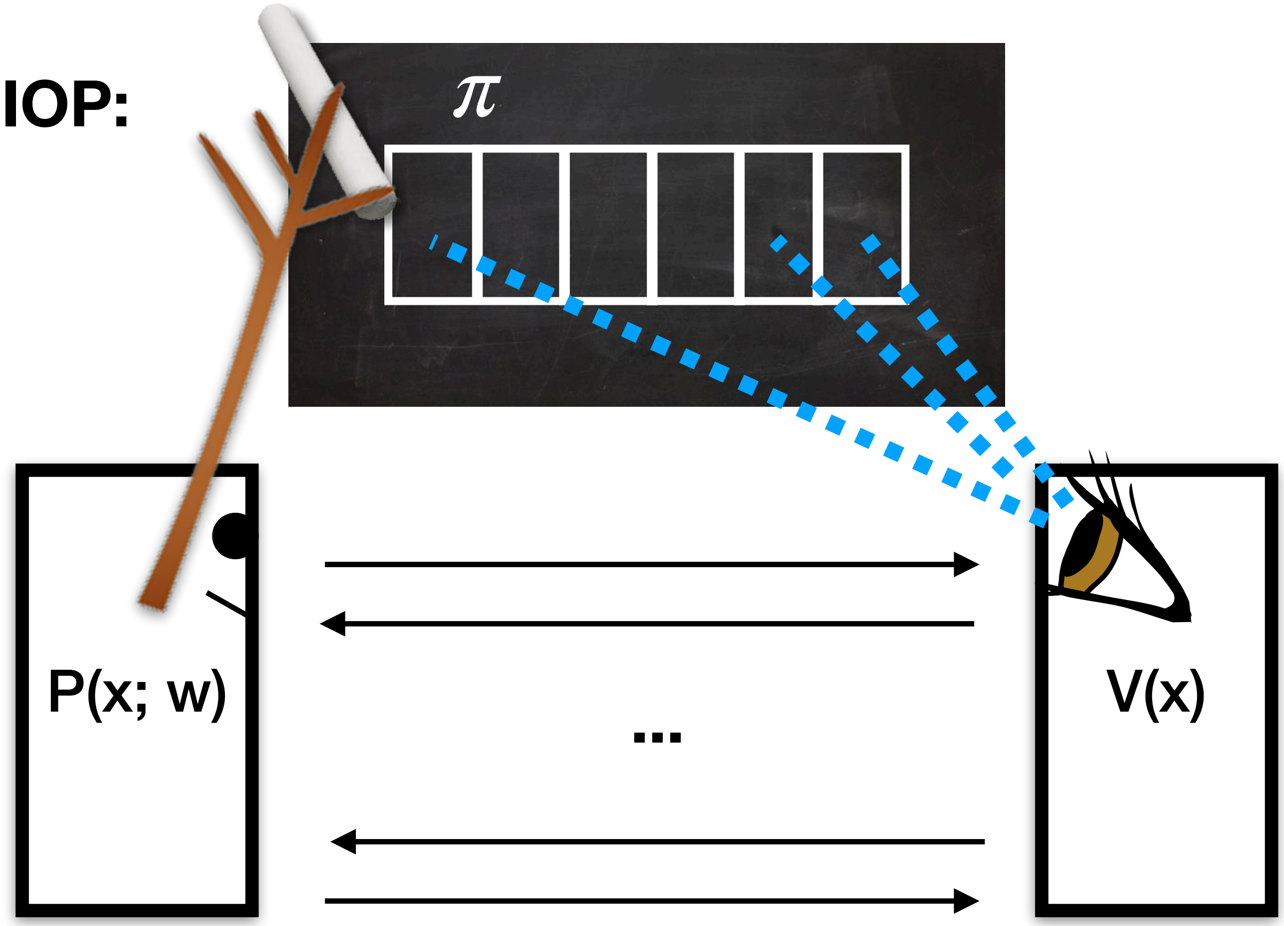


Argument:

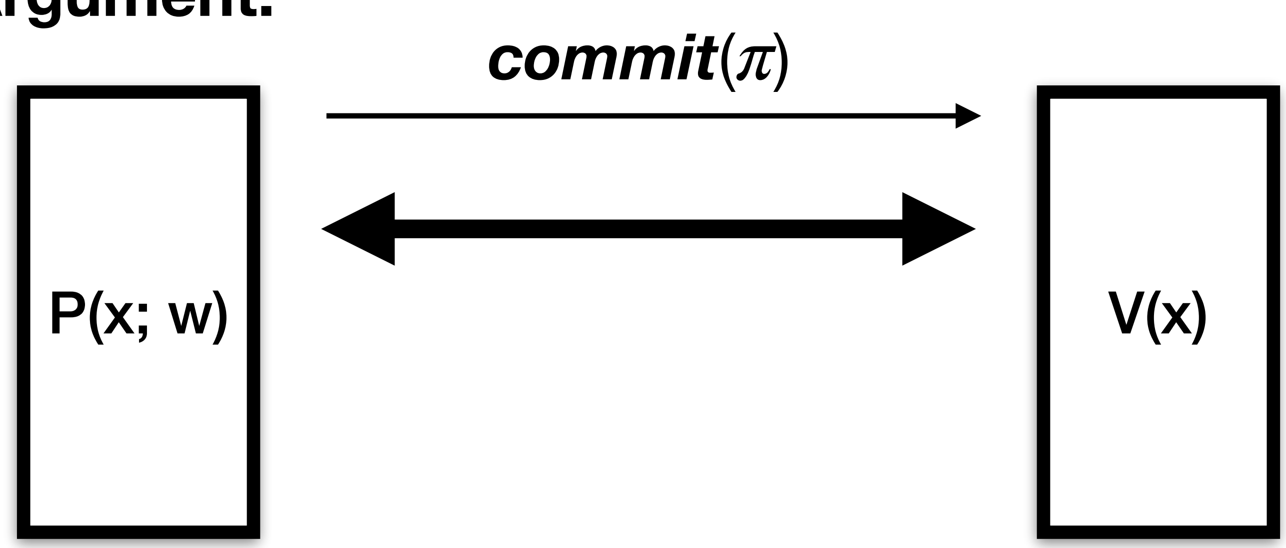


Compiling IOPs to Arguments

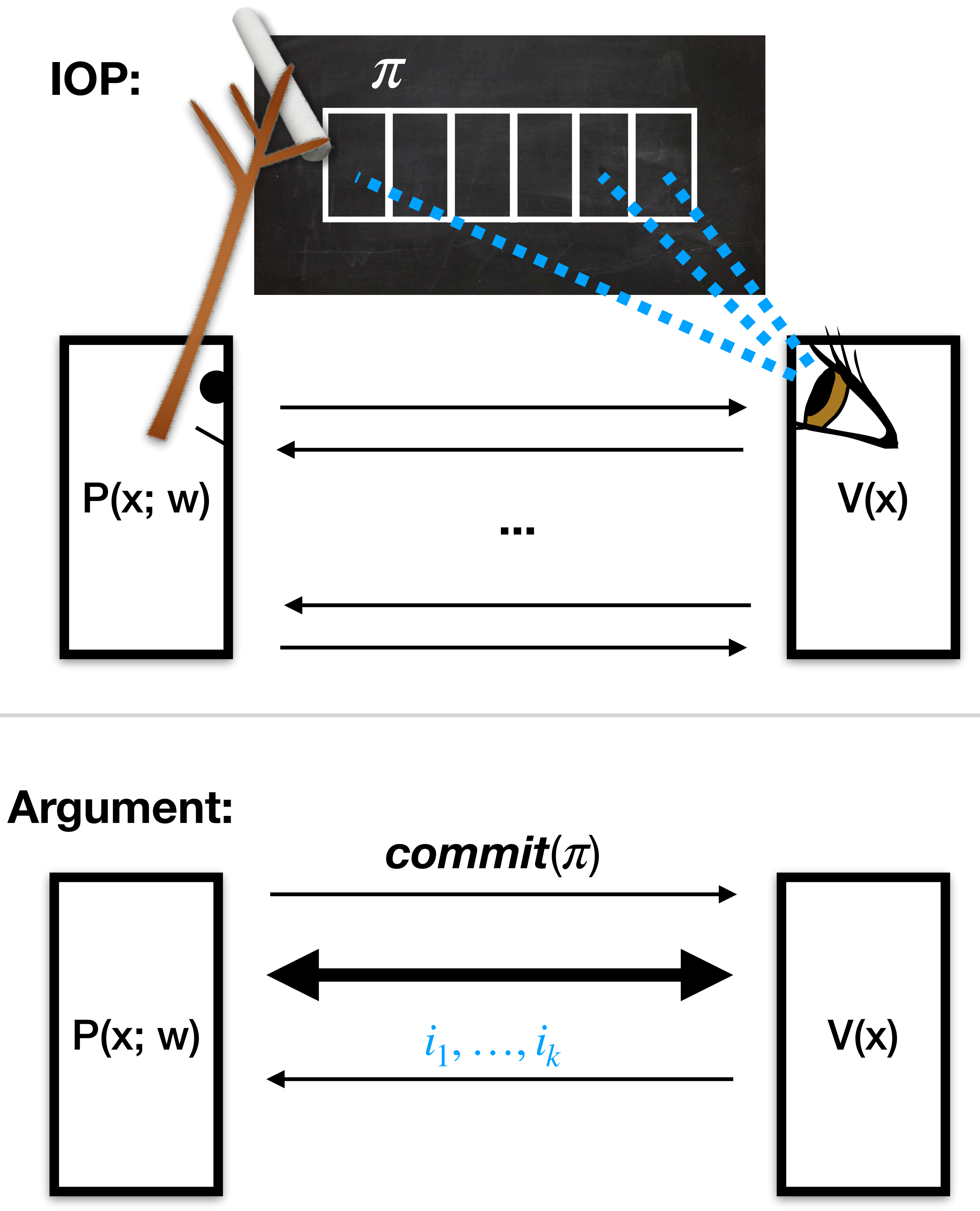
IOP:



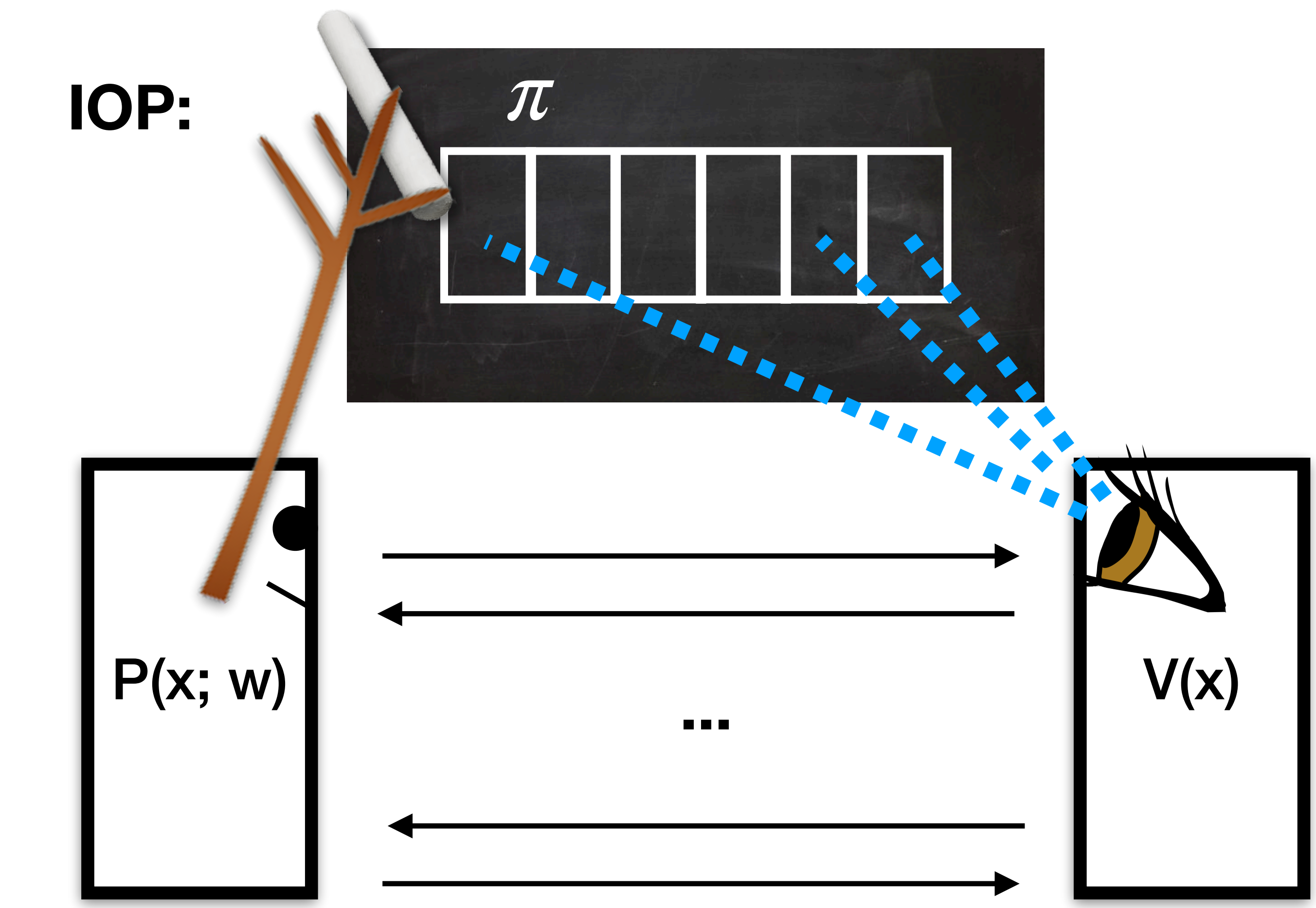
Argument:

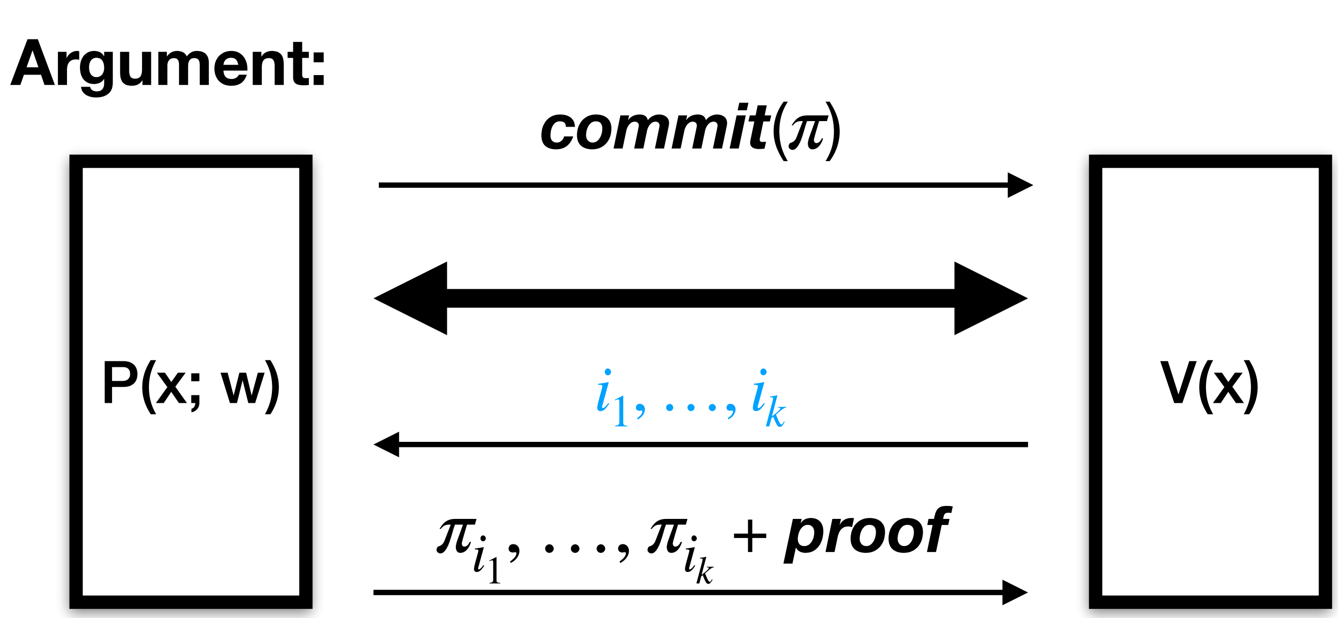


Compiling IOPs to Arguments



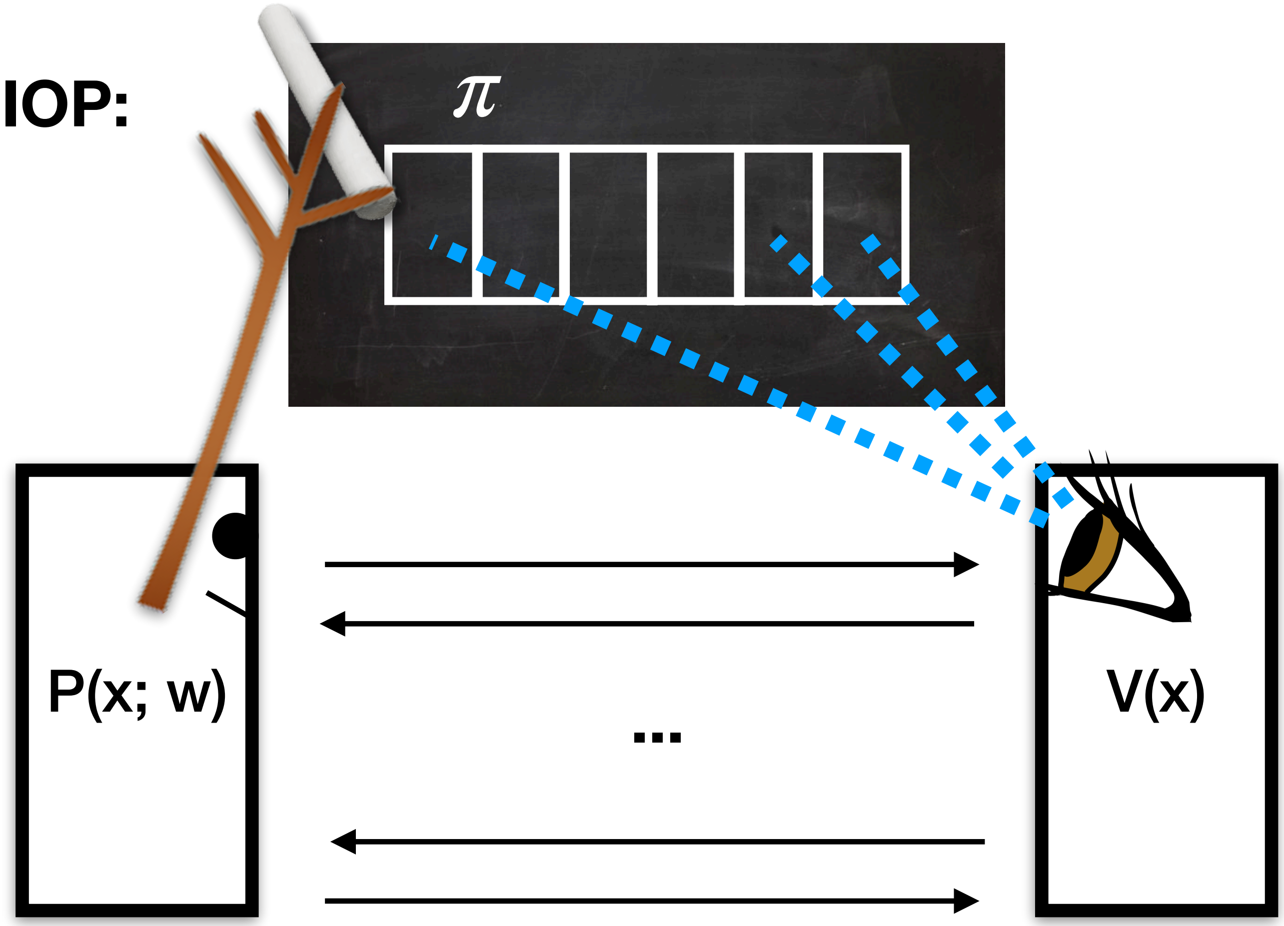
Compiling IOPs to Arguments



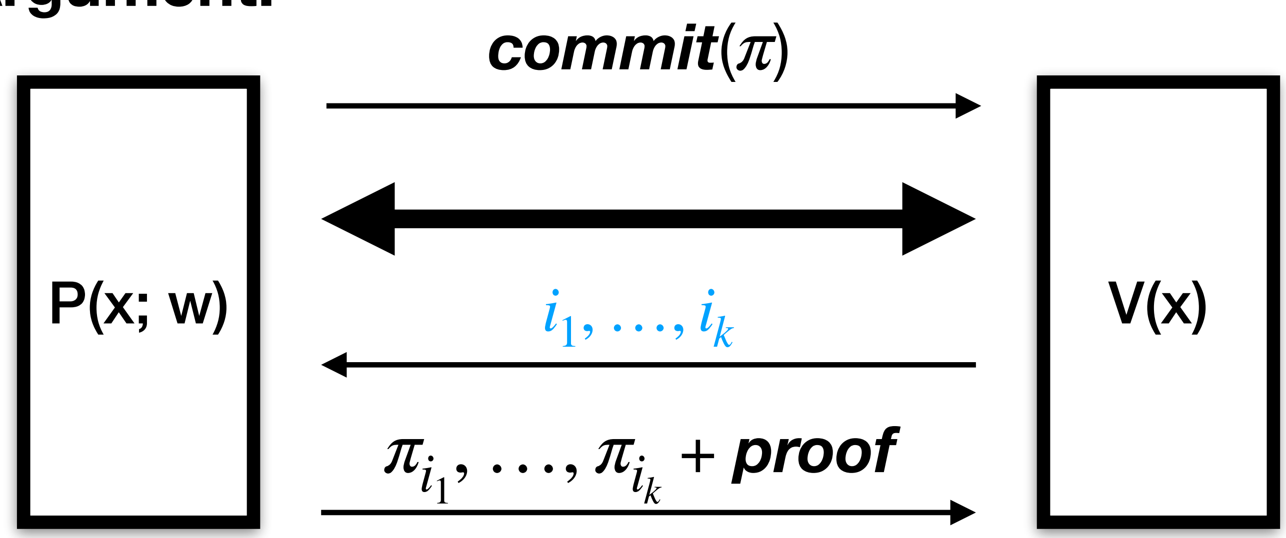


Compiling IOPs to Arguments

IOP:



Argument:



Important Question:

Which IOP prover cost is most relevant to argument prover?

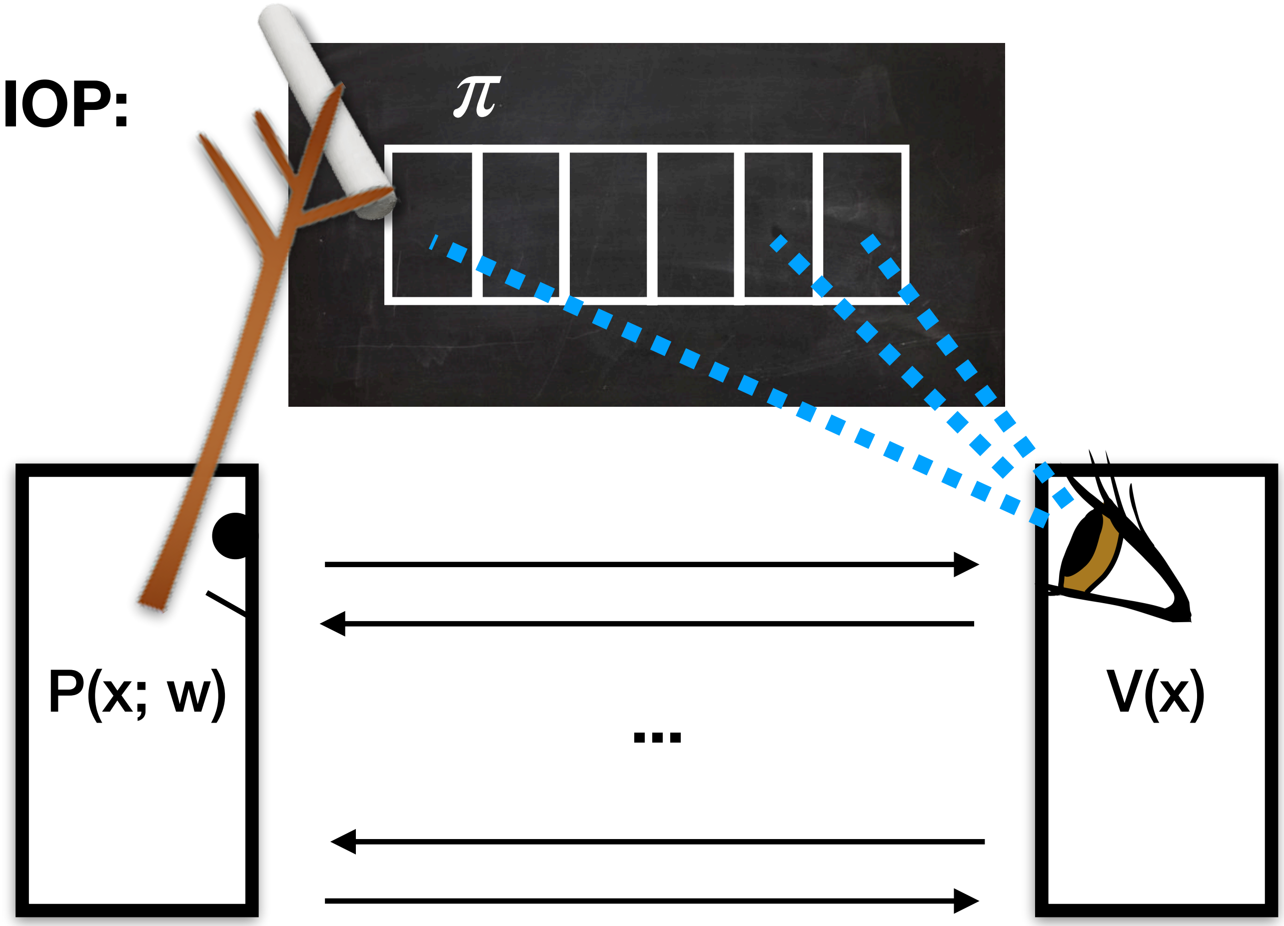
A. **enumerate all of π**

B. **compute π_i given i**

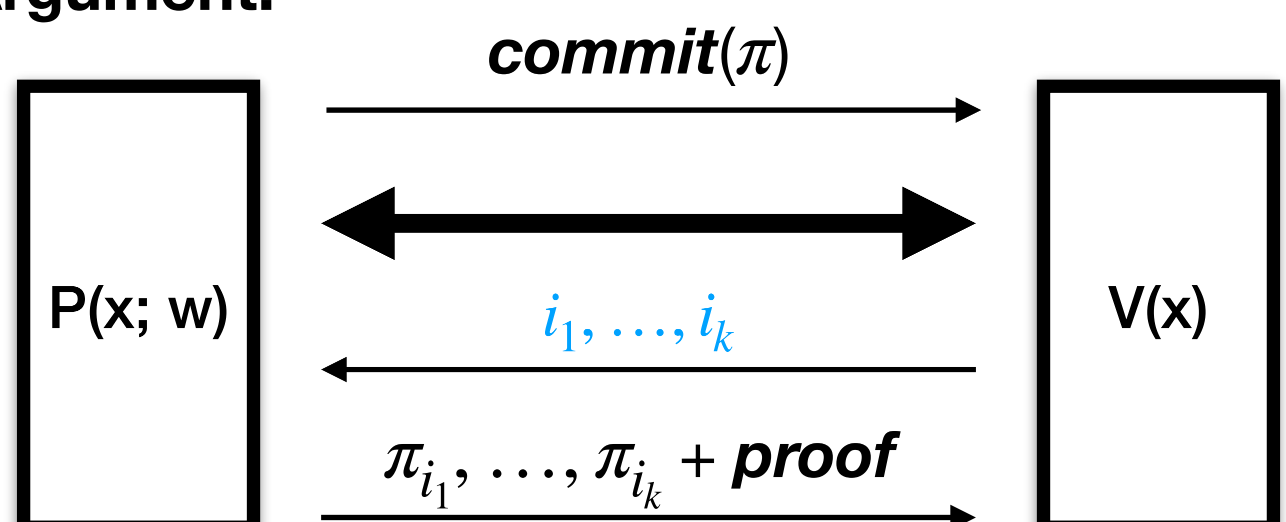
C. **other?**

Compiling IOPs to Arguments

IOP:



Argument:



Important Question:

Which IOP prover cost is most relevant to argument prover?

A. **enumerate all of π**

B. **compute π_i given i**

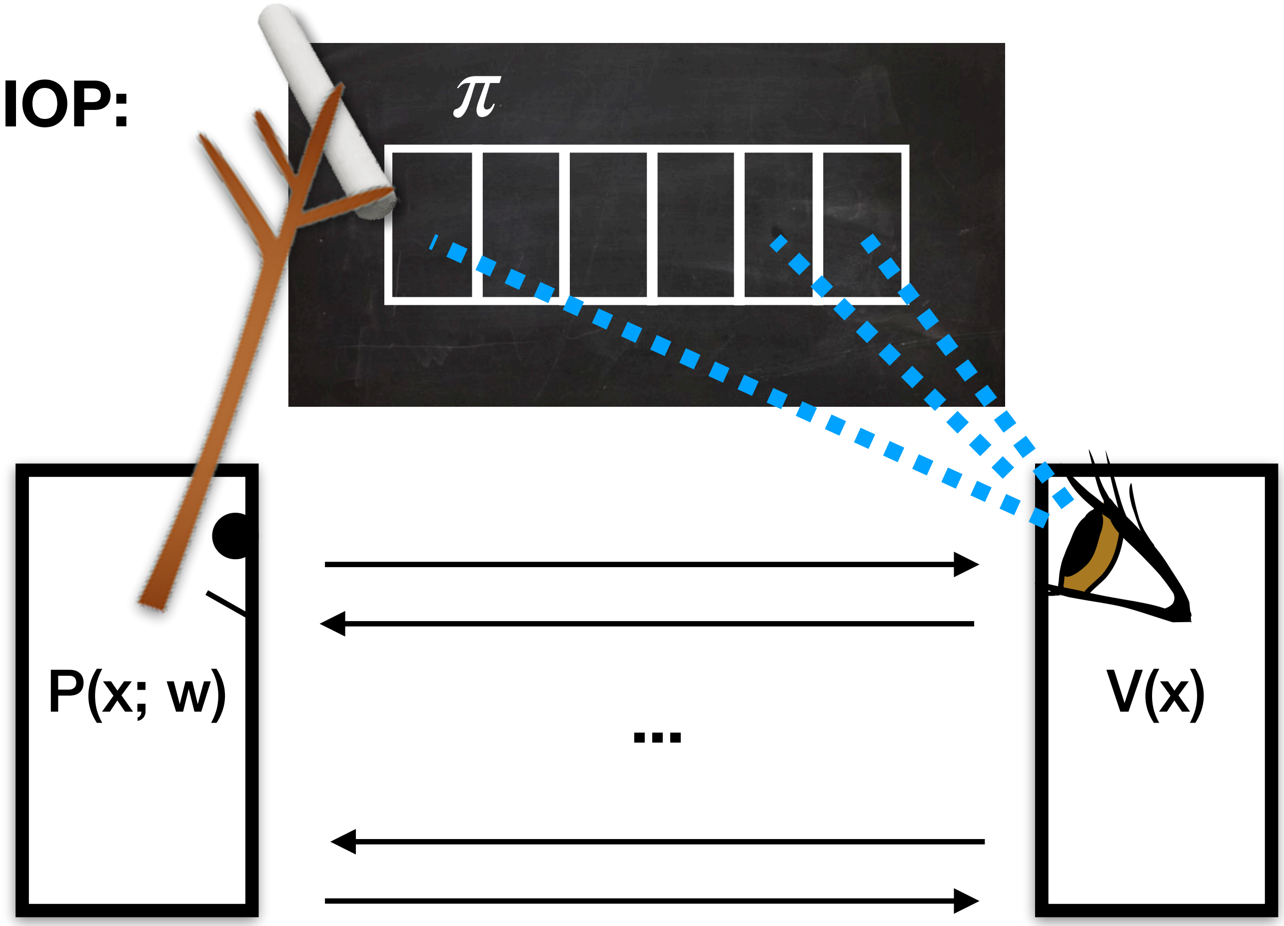
C. **other?**

Non-answer:

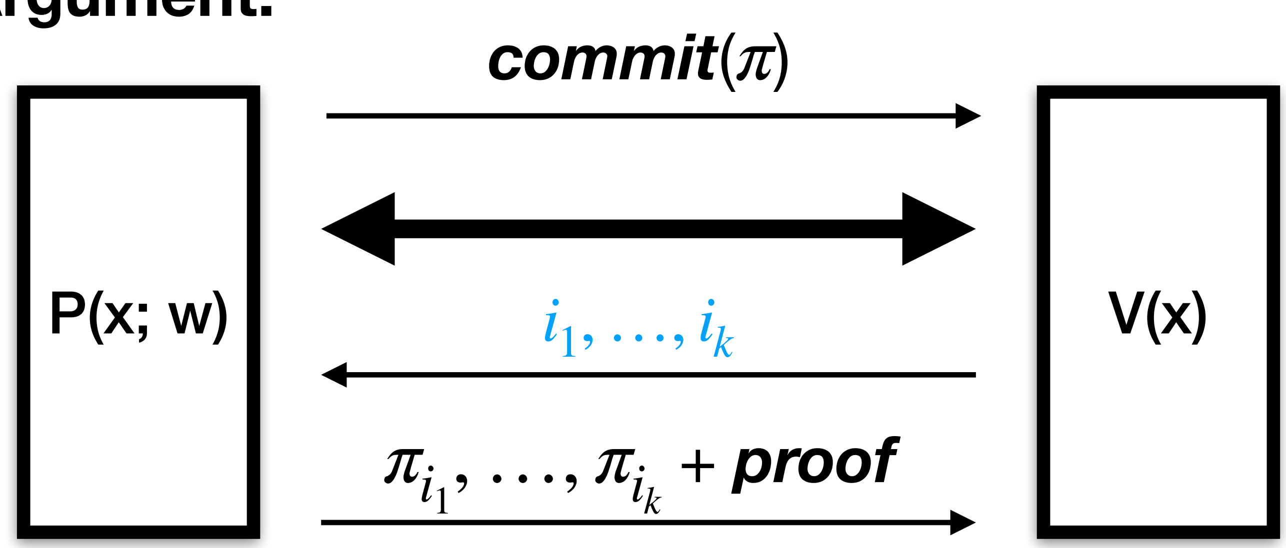
Depends on how "***commit***" and "***proof***" are instantiated...

Compiling IOPs to Arguments

IOP:



Argument:



Important Question:

Which IOP prover cost is most relevant to argument prover?

- **A. enumerate all of π**
- **B. compute π_i given i**
- **C. other?**

Non-answer:

Depends on how "***commit***" and "***proof***" are instantiated...

Why does this matter?

We know IOPs with time- & space-efficient provers in the sense of **(B)** but not **(A)**.

Instantiations of Commit-and-Prove

Instantiations of Commit-and-Prove

A. Merkle commitments

Instantiations of Commit-and-Prove

A. Merkle commitments

- Prover's work: $\approx$ **enumerating all of π**

Instantiations of Commit-and-Prove

A. Merkle commitments
   - Prover's work: $\approx$ **enumerating all of π**

B. Function commitments [BC '12]

Instantiations of Commit-and-Prove

A. Merkle commitments
   - Prover's work: ≈ **enumerating all of π**

B. Function commitments [BC '12]
   - Prover's work: ≈ **computing π_i for a given i.**

Instantiations of Commit-and-Prove

A. Merkle commitments

   - Prover's work: $\approx$ **enumerating all of π**

B. Function commitments [BC '12]

   - Prover's work: $\approx$ **computing π_i for a given i.**
   - **Private coin** proofs

Instantiations of Commit-and-Prove

A. Merkle commitments
   - Prover's work: $\approx$ **enumerating all of π**

B. Function commitments [BC '12]
   - Prover's work: $\approx$ **computing π_i for a given i.**
   - **Private coin** proofs

C. For a "polynomial IOP" ($\pi : \mathbb{F}_q^n \to \mathbb{F}_q$ is truth table of a multilinear polynomial), can use a polynomial commitment [BFS19]

Instantiations of Commit-and-Prove

A. Merkle commitments

   - Prover's work: $\approx$ **enumerating all of π**

B. Function commitments [BC '12]

   - Prover's work: $\approx$ **computing π_i for a given i.**
   - **Private coin** proofs

C. For a "polynomial IOP" ($\pi : \mathbb{F}_q^n \to \mathbb{F}_q$ is truth table of a multilinear polynomial), can use a polynomial commitment [BFS19]

   - Polynomial commitments can be **public-coin**

Instantiations of Commit-and-Prove

A. Merkle commitments

- Prover's work: $\approx$ **enumerating all of π**

B. Function commitments [BC '12]

- Prover's work: $\approx$ **computing π_i for a given i.**
- **Private coin** proofs

C. For a "polynomial IOP" ($\pi : \mathbb{F}_q^n \to \mathbb{F}_q$ is truth table of a multilinear polynomial), can use a polynomial commitment [BFS19]

- Polynomial commitments can be **public-coin**
- **This work:** Prover's work $\approx$ enumerating description of π (not the whole truth table);

Instantiations of Commit-and-Prove

A. Merkle commitments
   - Prover's work: $\approx$ **enumerating all of π**

B. Function commitments [BC '12]
   - Prover's work: $\approx$ **computing π_i for a given i.**
   - **Private coin** proofs

C. For a "polynomial IOP" ($\pi : \mathbb{F}_q^n \to \mathbb{F}_q$ is truth table of a multilinear polynomial), can use a polynomial commitment [BFS19]
   - Polynomial commitments can be **public-coin**
   - **This work:** Prover's work $\approx$ enumerating description of π (not the whole truth table);
     (time- and space-) efficient for known IOPs (e.g. Clover [BTVW14])

Our Polynomial Commitment Efficiency Results

Our Polynomial Commitment Efficiency Results

Informal Theorem 1: Assume a group of "unknown order". Then there is a polynomial commitment scheme with public-coin commit and prove protocols.

Our Polynomial Commitment Efficiency Results

Informal Theorem 1: Assume a group of "unknown order". Then there is a polynomial commitment scheme with public-coin commit and prove protocols.

Moreover, the committer/prover on (multi-linear) input p is efficient given *streaming access* to $\big(p(x)\big)_{x\in\{0,1\}^n}$.

Our Polynomial Commitment Efficiency Results

Informal Theorem 1: Assume a group of "unknown order". Then there is a polynomial commitment scheme with public-coin commit and prove protocols.

Moreover, the committer/prover on (multi-linear) input p is efficient given *streaming access* to $\left(p(x)\right)_{x \in \{0,1\}^n}$.

Informal Theorem 2: There are polynomial IOPs where the prover can compute relevant streams above (as well as all other IOP messages) with time- and space-efficiency.

No More Talking About (Fine-Grained) Efficiency

Our Polynomial Commitment Efficiency Results

Informal Theorem 1: Assume a group of "unknown order". Then there is a polynomial commitment scheme with public-coin commit and prove protocols.

~~Moreover, the committer/prover on input p is efficient (in both time and space) given multi-pass *streaming access* to values of p on $\{0,1\}^n$.~~

~~**Informal Theorem 2:** There are polynomial IOPs where the prover can compute relevant streams above (as well as all other IOP messages) with time- and space-efficiency.~~

Polynomial Commitment Blueprint / Sketch

Polynomial Comm
Blueprint / Ske

[BFS19]: Basic framework, buggy instantiation.

They independently discovered bug

Polynomial Comm
Blueprint / Ske

[BFS19]: Basic framework, buggy instantiation.

They independently discovered bug

Commit($p : \mathbb{F}_q^n \to \mathbb{F}_q$):

Output $h(p)$, where h is a "homomorphic CRHF" (more later)

Polynomial Comm… Blueprint / Ske…

[BFS19]: Basic framework, buggy instantiation.

They independently discovered bug

Commit($p : \mathbb{F}_q^n \to \mathbb{F}_q$):

Output $h(p)$, where h is a "homomorphic CRHF" (more later)

Prove("I know a degree-d poly p s.t. Commit$(p) = c$
and $p(x) = z$")

Polynomial Comm
Blueprint / Ske

[BFS19]: Basic framework, buggy instantiation.

They independently discovered bug

Commit($p : \mathbb{F}_q^n \to \mathbb{F}_q$):

Output $h(p)$, where h is a "homomorphic CRHF" (more later)

Prove("I know a degree-d poly p s.t.

abstractly: $f(p) = (c, z)$, where f is a homomorphism

Polynomial Comm
Blueprint / Ske

[BFS19]: Basic framework, buggy instantiation.

They independently discovered bug

Commit($p : \mathbb{F}_q^n \to \mathbb{F}_q$):

Output $h(p)$, where h is a "homomorphic CRHF" (more later)

Prove("I know a degree-d poly p s.t.

abstractly: $f(p) = (c, z)$,
where f is a homomorphism

1. Split claim into similar sub-claims of smaller size

Polynomial Comm
Blueprint / Ske

[BFS19]: Basic framework, buggy instantiation.

They independently discovered bug

Commit($p : \mathbb{F}_q^n \to \mathbb{F}_q$):

Output $h(p)$, where h is a "homomorphic CRHF" (more later)

Prove("I know a degree-d poly p s.t.

abstractly: $f(p) = (c, z)$,
where f is a homomorphism

1. Split claim into similar sub-claims of smaller size
2. Combine sub-claims to reduce number

Polynomial Comm
Blueprint / Ske

[BFS19]: Basic framework, buggy instantiation.

They independently discovered bug

Commit($p : \mathbb{F}_q^n \to \mathbb{F}_q$):

Output $h(p)$, where h is a "homomorphic CRHF" (more later)

Prove("I know a degree-d poly p s.t.

abstractly: $f(p) = (c, z)$, where f is a homomorphism

1. Split claim into similar sub-claims of smaller size
2. Combine sub-claims to reduce number
3. Recurse

Polynomial Comm... Blueprint / Ske...

[BFS19]: Basic framework, buggy instantiation.

They independently discovered bug

Commit($p : \mathbb{F}_q^n \to \mathbb{F}_q$):

Output $h(p)$, where h is a "homomorphic CRHF" (more later)

Prove("I know a degree-d poly p s.t.

abstractly: $f(p) = (c, z)$,
where f is a homomorphism

1. ~~Split claim into similar sub-claims of smaller size~~
2. Combine sub-claims to reduce number
3. ~~Recurse~~

Not today!

Not today!

From Many Claims to Fewer Claims?

From Many Claims to Fewer Claims?

Initial Claims: Knowledge of f-preimages of $y_1, \ldots, y_k$
(think of f as an arbitrary homomorphism)

From Many Claims to Fewer Claims?

Initial Claims: Knowledge of f-preimages of $y_1, \ldots, y_k$ (think of f as an arbitrary homomorphism)

Flawed Protocol:

From Many Claims to Fewer Claims?

Initial Claims: Knowledge of f-preimages of $y_1, \ldots, y_k$ (think of f as an arbitrary homomorphism)

Flawed Protocol:

1. Let $y' = \sum_i r_i y_i$, for $r_i \leftarrow [2^\lambda]$ sampled by verifier

From Many Claims to Fewer Claims?

Initial Claims: Knowledge of f-preimages of $y_1, \ldots, y_k$ (think of f as an arbitrary homomorphism)

Flawed Protocol:

1. Let $y' = \sum_i r_i y_i$, for $r_i \leftarrow [2^\lambda]$ sampled by verifier

2. Prover proves knowledge of $x' \in f^{-1}(y')$

From Many Claims to Fewer Claims?

Initial Claims: Knowledge of f-preimages of $y_1, \ldots, y_k$ (think of f as an arbitrary homomorphism)

Flawed Protocol:

1. Let $y' = \sum_i r_i y_i$, for $r_i \leftarrow [2^\lambda]$ sampled by verifier

2. Prover proves knowledge of $x' \in f^{-1}(y')$

Prover might know $x \in f^{-1}(2y_1)$, but not $x \in f^{-1}(y_1)$; could still win with probability $1/2$.

From Many Claims to Fewer Claims?

Initial Claims: Knowledge of f-preimages of $y_1, \ldots, y_k$ (think of f as an arbitrary homomorphism)

Flawed Protocol:

1. Let $y' = \sum_i r_i y_i$, for $r_i \leftarrow [2^\lambda]$ sampled by verifier

2. Prover proves knowledge of $x' \in f^{-1}(y')$

Prover might know $x \in f^{-1}(2y_1)$, but not $x \in f^{-1}(y_1)$; could still win with probability $1/2$.

Prover might only know $x_{\mathbf{r}} \in f^{-1}(\langle \mathbf{r}, \mathbf{y} \rangle)$ when $\mathbf{r}$ is in a lattice $L \subseteq \mathbb{Z}^k$. Winning probability $\approx$ density of L in $\mathbb{Z}^k$.

From Many Claims to Fewer Claims?

Initial Claims: Knowledge of f-preimages of $y_1, \dots, y_k$
(think of f as an arbitrary homomorphism)

[BFS19] show ***computational*** soundness for a specific f.

Flawed Protocol:

1. Let $y' = \sum_i r_i y_i$, for $r_i \leftarrow [2^\lambda]$ sampled by verifier

2. Prover proves knowledge of $x' \in f^{-1}(y')$

Prover might know $x \in f^{-1}(2y_1)$, but not $x \in f^{-1}(y_1)$;
could still win with probability $1/2$.

Prover might only know $x_{\mathbf{r}} \in f^{-1}(\langle \mathbf{r}, \mathbf{y} \rangle)$ when $\mathbf{r}$ is in a lattice $L \subseteq \mathbb{Z}^k$. Winning probability $\approx$ density of L in $\mathbb{Z}^k$.

From Many Claims to Fewer Claims: Our Protocol

From Many Claims to Fewer Claims: Our Protocol

Let $f : \mathbb{G} \rightarrow \mathbb{H}$ be an arbitrary homomorphism
$\mathbf{y} = (y_1, \ldots, y_k) \in \mathbb{H}^k$ be arbitrary.

From Many Claims to Fewer Claims: Our Protocol

Let $f : \mathbb{G} \to \mathbb{H}$ be an arbitrary homomorphism
$\mathbf{y} = (y_1, \ldots, y_k) \in \mathbb{H}^k$ be arbitrary.

Prover claims to know $\mathbf{x} = (x_1, \ldots, x_k) \in \mathbb{G}^k$ s.t. $f(x_i) = y_i$.

From Many Claims to Fewer Claims: Our Protocol

Let $f : \mathbb{G} \to \mathbb{H}$ be an arbitrary homomorphism
$\mathbf{y} = (y_1, \ldots, y_k) \in \mathbb{H}^k$ be arbitrary.

Prover claims to know $\mathbf{x} = (x_1, \ldots, x_k) \in \mathbb{G}^k$ s.t. $f(x_i) = y_i$.

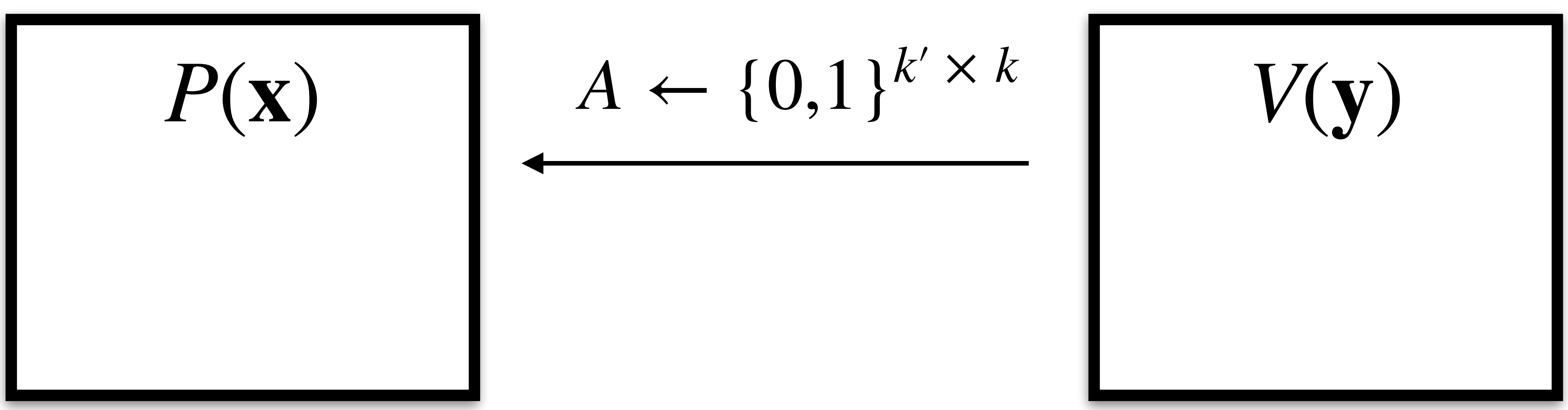
$P(\mathbf{x})$

$V(\mathbf{y})$

From Many Claims to Fewer Claims: Our Protocol

Let $f : \mathbb{G} \to \mathbb{H}$ be an arbitrary homomorphism
$\mathbf{y} = (y_1, \ldots, y_k) \in \mathbb{H}^k$ be arbitrary.

Prover claims to know $\mathbf{x} = (x_1, \ldots, x_k) \in \mathbb{G}^k$ s.t. $f(x_i) = y_i$.



From Many Claims to Fewer Claims: Our Protocol

Let $f : \mathbb{G} \to \mathbb{H}$ be an arbitrary homomorphism
$\mathbf{y} = (y_1, \ldots, y_k) \in \mathbb{H}^k$ be arbitrary.

Prover claims to know $\mathbf{x} = (x_1, \ldots, x_k) \in \mathbb{G}^k$ s.t. $f(x_i) = y_i$.

$P(\mathbf{x})$

$A \leftarrow \{0,1\}^{k' \times k}$

$V(\mathbf{y})$

$\mathbf{y}' := A \cdot \mathbf{y}$

From Many Claims to Fewer Claims: Our Protocol

Let $f : \mathbb{G} \to \mathbb{H}$ be an arbitrary homomorphism
$\mathbf{y} = (y_1, \ldots, y_k) \in \mathbb{H}^k$ be arbitrary.

Prover claims to know $\mathbf{x} = (x_1, \ldots, x_k) \in \mathbb{G}^k$ s.t. $f(x_i) = y_i$.

$P(\mathbf{x})$		$V(\mathbf{y})$
	$\xleftarrow{A \leftarrow \{0,1\}^{k' \times k}}$	
	$\xrightarrow{\mathbf{x}' := A \cdot \mathbf{x}}$	$\mathbf{y}' := A \cdot \mathbf{y}$

From Many Claims to Fewer Claims: Our Protocol

Let $f : \mathbb{G} \to \mathbb{H}$ be an arbitrary homomorphism
$\mathbf{y} = (y_1, \ldots, y_k) \in \mathbb{H}^k$ be arbitrary.

Prover claims to know $\mathbf{x} = (x_1, \ldots, x_k) \in \mathbb{G}^k$ s.t. $f(x_i) = y_i$.

$P(\mathbf{x})$		$V(\mathbf{y})$	
	$\xleftarrow{A \leftarrow \{0,1\}^{k' \times k}}$		
	$\xrightarrow{\mathbf{x}' := A \cdot \mathbf{x}}$	$\mathbf{y}' := A \cdot \mathbf{y}$	**Accept if** $y'_i = f(x'_i)$ **for all** $i \in [k']$

Soundness: Our Batch Extractor

$P(\mathbf{x})$

$A \leftarrow \{0,1\}^{k' \times k}$ ($V \to P$)

$\mathbf{x}' := A \cdot \mathbf{x}$ ($P \to V$)

$V(\mathbf{y})$

$\mathbf{y}' := A \cdot \mathbf{y}$

Accept if
$y_i' = f(x_i')$
for all $i \in [k']$

Soundness: Our Batch Extractor

$P(\mathbf{x})$

$A \leftarrow \{0,1\}^{k' \times k}$

$\mathbf{x}' := A \cdot \mathbf{x}$

$V(\mathbf{y})$

$\mathbf{y}' := A \cdot \mathbf{y}$

Accept if

$y_i' = f(x_i')$

for all $i \in [k']$

Attempt 0:

Soundness: Our Batch Extractor

$P(\mathbf{x})$

$A \leftarrow \{0,1\}^{k' \times k}$

$\mathbf{x}' := A \cdot \mathbf{x}$

$V(\mathbf{y})$

$\mathbf{y}' := A \cdot \mathbf{y}$

Accept if
$y'_i = f(x'_i)$
for all $i \in [k']$

Attempt 0:

Get an accepting transcript $(A, \mathbf{x}')$, **hope** A has an integer left-inverse.

Soundness: Our Batch Extractor

$P(\mathbf{x})$

$A \leftarrow \{0,1\}^{k' \times k}$ $\longleftarrow$

$\mathbf{x}' := A \cdot \mathbf{x}$ $\longrightarrow$

$V(\mathbf{y})$

$\mathbf{y}' := A \cdot \mathbf{y}$

Accept if
$y'_i = f(x'_i)$
for all $i \in [k']$

Attempt 0:

$k' < k$ ...

Get an accepting transcript $(A, \mathbf{x}')$, **hope** A has an integer left-inverse.

Soundness: Our Batch Extractor

$P(\mathbf{x})$

$A \leftarrow \{0,1\}^{k' \times k}$

$\mathbf{x}' := A \cdot \mathbf{x}$

$V(\mathbf{y})$

$\mathbf{y}' := A \cdot \mathbf{y}$

Accept if

$y_i' = f(x_i')$

for all $i \in [k']$

Attempt 0:

$k' < k$ ...

Get an accepting transcript $(A, \mathbf{x}')$, **hope** A has an integer left-inverse.

Compute $\mathbf{x} = A^{-1} \cdot \mathbf{x}'$. (correctness follows from homomorphism)

Soundness: Our Batch Extractor

$P(\mathbf{x})$

$A \leftarrow \{0,1\}^{k' \times k}$

$\mathbf{x}' := A \cdot \mathbf{x}$

$V(\mathbf{y})$

$\mathbf{y}' := A \cdot \mathbf{y}$

Accept if
$y'_i = f(x'_i)$
for all $i \in [k']$

Attempt 1:

Soundness: Our Batch Extractor

$P(\mathbf{x})$

$A \leftarrow \{0,1\}^{k' \times k}$

$\mathbf{x}' := A \cdot \mathbf{x}$

$V(\mathbf{y})$

$\mathbf{y}' := A \cdot \mathbf{y}$

Accept if
$y'_i = f(x'_i)$
for all $i \in [k']$

Attempt 1:

Rewind until B accepting transcripts $\rightarrow A \in \{0,1\}^{Bk' \times k}$, $\mathbf{x}' \in \mathbb{G}^{Bk'}$.

Soundness: Our Batch Extractor

$P(\mathbf{x})$		$V(\mathbf{y})$
	$\xleftarrow{A \leftarrow \{0,1\}^{k' \times k}}$	$\mathbf{y}' := A \cdot \mathbf{y}$
	$\xrightarrow{\mathbf{x}' := A \cdot \mathbf{x}}$	

Accept if
$y'_i = f(x'_i)$
for all $i \in [k']$

Attempt 1:

Rewind until B accepting transcripts $\rightarrow A \in \{0,1\}^{Bk' \times k}$, $\mathbf{x}' \in \mathbb{G}^{Bk'}$.

Hope A has an integer left inverse.

Soundness: Our Batch Extractor

$P(\mathbf{x})$

$A \leftarrow \{0,1\}^{k' \times k}$

$\mathbf{x}' := A \cdot \mathbf{x}$

$V(\mathbf{y})$

$\mathbf{y}' := A \cdot \mathbf{y}$

Accept if
$y'_i = f(x'_i)$
for all $i \in [k']$

Attempt 1:

Rewind until B accepting transcripts $\rightarrow A \in \{0,1\}^{Bk' \times k}$, $\mathbf{x}' \in \mathbb{G}^{Bk'}$.

Hope A has an integer left inverse.

Compute $\mathbf{x} = A^{-1} \cdot \mathbf{x}'$.

Soundness: Our Batch Extractor

$P(\mathbf{x})$

$A \leftarrow \{0,1\}^{k' \times k}$

$\mathbf{x}' := A \cdot \mathbf{x}$

$V(\mathbf{y})$

$\mathbf{y}' := A \cdot \mathbf{y}$

Accept if
$y_i' = f(x_i')$
for all $i \in [k']$

Attempt 1:

Rewind until B accepting transcripts $\rightarrow A \in \{0,1\}^{Bk' \times k}$, $\mathbf{x}' \in \mathbb{G}^{Bk'}$.

Hope A has an integer left inverse.

Compute $\mathbf{x} = A^{-1} \cdot \mathbf{x}'$.

:-) with all but negl(k) **probability if** $5k/k' \leq B \leq O(1)$

Soundness: Our Batch Extractor

$P(\mathbf{x})$

$A \leftarrow \{0,1\}^{k' \times k}$

$\mathbf{x}' := A \cdot \mathbf{x}$

$V(\mathbf{y})$

$\mathbf{y}' := A \cdot \mathbf{y}$

Accept if
$y_i' = f(x_i')$
for all $i \in [k']$

Non-uniform distribution;
accepting transcripts skewed

Attempt 1:

Rewind until B accepting transcripts $\rightarrow A \in \{0,1\}^{Bk' \times k}$, $\mathbf{x}' \in \mathbb{G}^{Bk'}$.

Hope A has an integer left inverse.

Compute $\mathbf{x} = A^{-1} \cdot \mathbf{x}'$.

:-) with all but negl(k) **probability**
if $5k/k' \leq B \leq O(1)$

Soundness: Our Batch Extractor

$P(\mathbf{x})$

$A \leftarrow \{0,1\}^{k' \times k}$

$\mathbf{x}' := A \cdot \mathbf{x}$

$V(\mathbf{y})$

$\mathbf{y}' := A \cdot \mathbf{y}$

Accept if
$y'_i = f(x'_i)$
for all $i \in [k']$

Non-uniform distribution;
accepting transcripts skewed

so A is not
too skewed

Attempt 1:

Rewind until B accepting transcripts $\rightarrow A \in \{0,1\}^{Bk' \times k}$, $\mathbf{x}' \in \mathbb{G}^{Bk'}$.

Hope A has an integer left inverse.

Compute $\mathbf{x} = A^{-1} \cdot \mathbf{x}'$.

:-) with all but negl(k) **probability**
if $5k/k' \leq B \leq O(1)$

Soundness: Our Batch Extractor

$P(\mathbf{x})$

$A \leftarrow \{0,1\}^{k' \times k}$

$\mathbf{x}' := A \cdot \mathbf{x}$

$V(\mathbf{y})$

$\mathbf{y}' := A \cdot \mathbf{y}$

Accept if

$y'_i = f(x'_i)$

for all $i \in [k']$

Non-uniform distribution; *accepting* transcripts skewed

so $B\lambda > k$

so A is not too skewed

Attempt 1:

Rewind until B accepting transcripts $\rightarrow A \in \{0,1\}^{Bk' \times k}$, $\mathbf{x}' \in \mathbb{G}^{Bk'}$.

Hope A has an integer left inverse.

Compute $\mathbf{x} = A^{-1} \cdot \mathbf{x}'$.

:-) with all but negl(k) probability if $5k/k' \leq B \leq O(1)$

Soundness: Our Batch Extractor

$P(\mathbf{x})$

$A \leftarrow \{0,1\}^{k' \times k}$

$\mathbf{x}' := A \cdot \mathbf{x}$

$V(\mathbf{y})$

$\mathbf{y}' := A \cdot \mathbf{y}$

Accept if
$y'_i = f(x'_i)$
for all $i \in [k']$

Non-uniform distribution;
accepting transcripts skewed

so $B\lambda > k$

so A is not too skewed

Attempt 1:

Rewind until B accepting transcripts $\rightarrow A \in \{0,1\}^{Bk' \times k}$, $\mathbf{x}' \in \mathbb{G}^{Bk'}$.

Hope A has an integer left inverse.

Compute $\mathbf{x} = A^{-1} \cdot \mathbf{x}'$.

:-) with all but negl(k) probability if $5k/k' \leq B \leq O(1)$

Bonus: can prove knowledge of "small" $\mathbf{x}$ by bounds-checking $\mathbf{x}'$; extractor works because computed A^{-1} has "small" entries ($2^{\mathrm{poly}(k)}$)

Soundness: Our Batch Extractor

$P(\mathbf{x})$

$A \leftarrow \{0,1\}^{k' \times k}$

$\mathbf{x}' := A \cdot \mathbf{x}$

$V(\mathbf{y})$

$\mathbf{y}' := A \cdot \mathbf{y}$

Accept if
$y_i' = f(x_i')$
for all $i \in [k']$

Non-uniform distribution;
accepting transcripts skewed

so $B\lambda > k$

so A is not too skewed

Attempt 1:

Rewind until B accepting transcripts $\rightarrow A \in \{0,1\}^{Bk' \times k}$, $\mathbf{x}' \in \mathbb{G}^{Bk'}$.

Hope A has an integer left inverse.

Compute $\mathbf{x} = A^{-1} \cdot \mathbf{x}'$.

**:-) with all but negl(k) probability
if $5k/k' \leq B \leq O(1)$**

Bonus: can prove knowledge of "small" $\mathbf{x}$ by bounds-checking $\mathbf{x}'$;
extractor works because computed A^{-1} has "small" entries ($2^{\text{poly}(k)}$)

actually essential &
improperly addressed in [BFS19]

Soundness: Our Batch Extractor

$P(\mathbf{x})$	$A \leftarrow \{0,1\}^{k' \times k}$ ← $\mathbf{x}' := A \cdot \mathbf{x}$ →	$V(\mathbf{y})$ $\mathbf{y}' := A \cdot \mathbf{y}$

Accept if
$y_i' = f(x_i')$
for all $i \in [k']$

Non-uniform distribution;
accepting transcripts skewed

so $B\lambda > k$

so A is not too skewed

Attempt 1:

Rewind until B accepting transcripts $\rightarrow A \in \{0,1\}^{Bk' \times k}, \mathbf{x}' \in \mathbb{G}^{Bk'}$.

Hope A has an integer left inverse.

:-) with all but negl(k) probability if $5k/k' \leq B \leq O(1)$

Compute $\mathbf{x} = A^{-1} \cdot \mathbf{x}'$.

Bonus: can prove knowledge of "small" $\mathbf{x}$ by bounds-checking $\mathbf{x}'$; extractor works because computed A^{-1} has "small" entries ($2^{\mathrm{poly}(k)}$)

actually essential &
improperly addressed in [BFS19]

We want to extract CRHF pre-images, but...

Homomorphic CRHFs

Homomorphic CRHFs

- Let $\mathbb{G} = \langle g \rangle$ be a group where it is hard to find $x \neq 0$ s.t. $g^x = 1$ (any *multiple* of the order of g).

Homomorphic CRHFs

- Let $\mathbb{G} = \langle g \rangle$ be a group where it is hard to find $x \neq 0$ s.t. $g^x = 1$ (any *multiple* of the order of g).
  - Hardness holds in generic group of unknown order

Homomorphic CRHFs

- Let $\mathbb{G} = \langle g \rangle$ be a group where it is hard to find $x \neq 0$ s.t. $g^x = 1$ (any *multiple* of the order of g).
  - Hardness holds in generic group of unknown order
  - Concrete candidates:
    - RSA group (private-coin setup)
    - Class groups of imaginary quadratic order (public-coin setup)

Homomorphic CRHFs

- Let $\mathbb{G} = \langle g \rangle$ be a group where it is hard to find $x \neq 0$ s.t. $g^x = 1$ (any *multiple* of the order of g).
  - Hardness holds in generic group of unknown order
  - Concrete candidates:
    - RSA group (private-coin setup)
    - Class groups of imaginary quadratic order (public-coin setup)
- Then $h(x) = g^x$ is a homomorphic CRHF from $\mathbb{Z}$ to $\mathbb{G}$

Homomorphic CRHFs: Domains beyond $\mathbb{Z}$

Homomorphic CRHFs: Domains beyond $\mathbb{Z}$

- We wanted an (additively) homomorphic CRHF mapping Commit : $\mathbb{Z}[x_1, \ldots, x_n] \to \mathbb{G}$ (& extra property I am ignoring)

Homomorphic CRHFs: Domains beyond $\mathbb{Z}$

- We wanted an (additively) homomorphic CRHF mapping Commit : $\mathbb{Z}[x_1, \ldots, x_n] \to \mathbb{G}$ (& extra property I am ignoring)
- *Almost* follows from a $\mathbb{Z} \to \mathbb{G}$ homomorphic CRHF:

Homomorphic CRHFs: Domains beyond $\mathbb{Z}$

- We wanted an (additively) homomorphic CRHF mapping Commit : $\mathbb{Z}[x_1, \ldots, x_n] \to \mathbb{G}$ (& extra property I am ignoring)
- *Almost* follows from a $\mathbb{Z} \to \mathbb{G}$ homomorphic CRHF:
  - Homomorphically "embed" $\mathbb{Z}[x_1, \ldots, x_n]$ into $\mathbb{Z}$ by setting $x_i = q^i$.

Homomorphic CRHFs: Domains beyond $\mathbb{Z}$

- We wanted an (additively) homomorphic CRHF mapping Commit : $\mathbb{Z}[x_1, \ldots, x_n] \to \mathbb{G}$ (& extra property I am ignoring)
- *Almost* follows from a $\mathbb{Z} \to \mathbb{G}$ homomorphic CRHF:
  - Homomorphically "embed" $\mathbb{Z}[x_1, \ldots, x_n]$ into $\mathbb{Z}$ by setting $x_i = q^i$.
    - Injective only on $D := \{$*small-coefficient* multilinear polynomials$\}$ (each coefficient is a digit base-q).

Homomorphic CRHFs: Domains beyond $\mathbb{Z}$

- We wanted an (additively) homomorphic CRHF mapping Commit : $\mathbb{Z}[x_1, \ldots, x_n] \to \mathbb{G}$ (& extra property I am ignoring)

- *Almost* follows from a $\mathbb{Z} \to \mathbb{G}$ homomorphic CRHF:

  - Homomorphically "embed" $\mathbb{Z}[x_1, \ldots, x_n]$ into $\mathbb{Z}$ by setting $x_i = q^i$.

    - Injective only on $D :=$ {*small-coefficient* multilinear polynomials} (each coefficient is a digit base-q).

    - Thus $\mathbb{Z}[x_1, \ldots, x_n] \to \mathbb{G}$ composition is a CRHF only on D.

Main Extraction Lemma

Main Extraction Lemma

Lemma: Let $A \leftarrow \{0,1\}^{5n \times n}$. With all but $2^{-\Omega(n)}$ probability, A has an integer left-inverse.

Main Extraction Lemma

5 is not tight,
but unimportant today

Lemma: Let $A \leftarrow \{0,1\}^{5n \times n}$. With all but $2^{-\Omega(n)}$ probability, A has an integer left-inverse.

Main Extraction Lemma

5 is not tight,
but unimportant today

Lemma: Let $A \leftarrow \{0,1\}^{5n \times n}$. With all but $2^{-\Omega(n)}$ probability, A has an integer left-inverse.

A taste of our proof:

Main Extraction Lemma

5 is not tight,
but unimportant today

Lemma: Let $A \leftarrow \{0,1\}^{5n \times n}$. With all but $2^{-\Omega(n)}$ probability, A has an integer left-inverse.

A taste of our proof:

- Consider sequence of lattices $\{L_i\}$, where L_i is generated by first i rows of A

Main Extraction Lemma

5 is not tight,
but unimportant today

Lemma: Let $A \leftarrow \{0,1\}^{5n \times n}$. With all but $2^{-\Omega(n)}$ probability, A has an integer left-inverse.

A taste of our proof:

- Consider sequence of lattices $\{L_i\}$, where L_i is generated by first i rows of A
- Show that L_i rapidly approaches (and becomes) $\mathbb{Z}^n$

Main Extraction Lemma

5 is not tight,
but unimportant today

Lemma: Let $A \leftarrow \{0,1\}^{5n \times n}$. With all but $2^{-\Omega(n)}$ probability, A has an integer left-inverse.

A taste of our proof:

- Consider sequence of lattices $\{L_i\}$, where L_i is generated by first i rows of A
- Show that L_i rapidly approaches (and becomes) $\mathbb{Z}^n$
  - Equivalently, $|\det(L_i)| \rightarrow 1$

Main Extraction Lemma

5 is not tight,
but unimportant today

Lemma: Let $A \leftarrow \{0,1\}^{5n \times n}$. With all but $2^{-\Omega(n)}$ probability, A has an integer left-inverse.

A taste of our proof:

- Consider sequence of lattices $\{L_i\}$, where L_i is generated by first i rows of A
- Show that L_i rapidly approaches (and becomes) $\mathbb{Z}^n$
  - Equivalently, $|\det(L_i)| \to 1$
  - We analyze prime factorization of $\det(L_i)$, show that each step kills enough prime powers with enough probability to deduce the lemma.

Conclusion

Conclusion

- First publicly verifiable arguments for NP that are time- and space-efficient and based on a simple complexity assumptions

Conclusion

- First publicly verifiable arguments for NP that are time- and space-efficient and based on a simple complexity assumptions
  - Based on groups of unknown order, but very lattice-related techniques

Conclusion

- First publicly verifiable arguments for NP that are time- and space-efficient and based on a simple complexity assumptions
  - Based on groups of unknown order, but very lattice-related techniques
  - **Open:** from lattice assumptions, or in random oracle model

Conclusion

- First publicly verifiable arguments for NP that are time- and space-efficient and based on a simple complexity assumptions
  - Based on groups of unknown order, but very lattice-related techniques
  - **Open:** from lattice assumptions, or in random oracle model
- Found and fixed bug in DARK polynomial commitment

Conclusion

- First publicly verifiable arguments for NP that are time- and space-efficient and based on a simple complexity assumptions
  - Based on groups of unknown order, but very lattice-related techniques
  - **Open:** from lattice assumptions, or in random oracle model
- Found and fixed bug in DARK polynomial commitment
- Techniques likely more broadly applicable: we also improve Pietrzak's proof of exponentiation protocol to achieve statistical soundness in arbitrary groups

Questions?