A Toolbox for Barriers on Interactive Oracle Proofs

Gal Arnon

Weizmann Institute

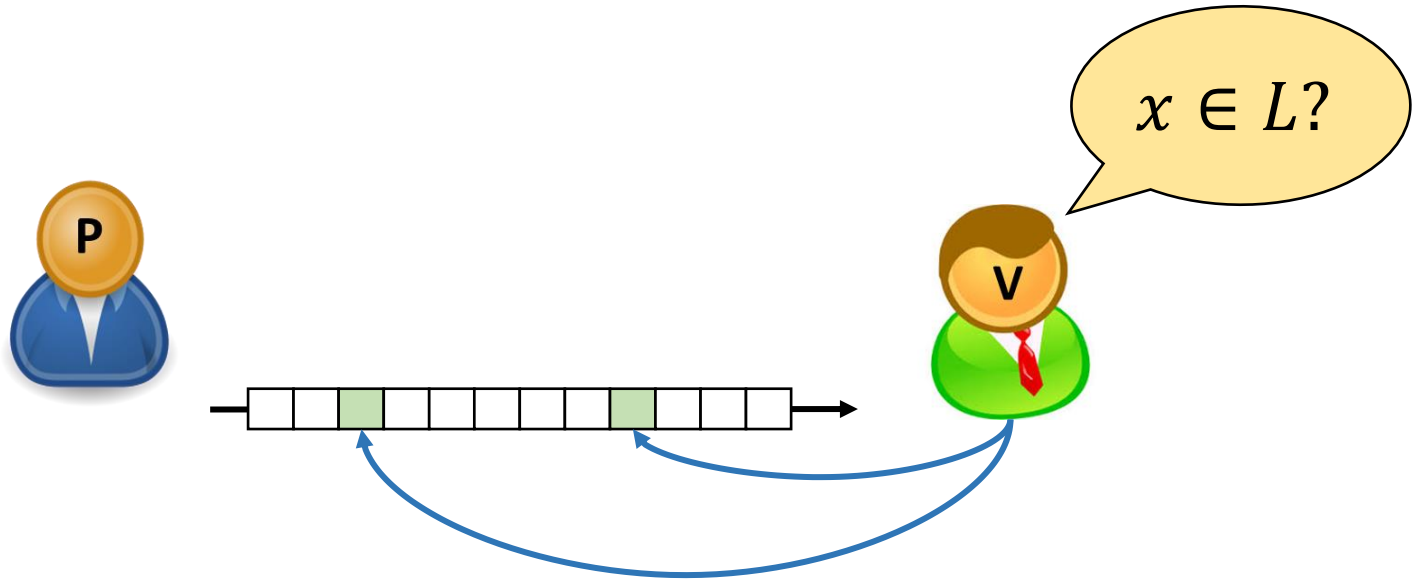
Joint work with:

Amey Bhangale (UC Riverside)

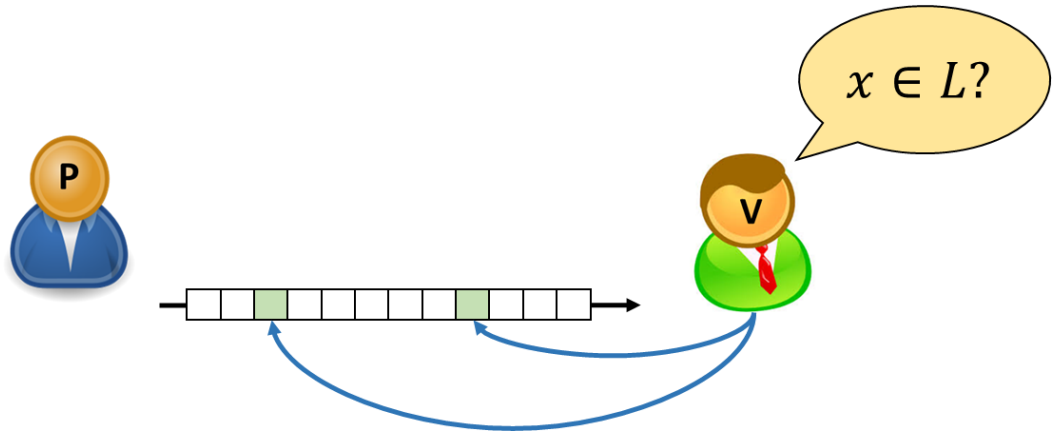
Alessandro Chiesa (EPFL)

Eylon Yogev (Bar-Ilan University)

Probabilistically Checkable Proofs



Probabilistically Checkable Proofs



Main parameters of interest:

- Length: number of symbols in proof
- Alphabet size: size of the proof alphabet
- Queries: number of symbols of the proof that are read by Verifier
- Soundness error: probability that the Verifier accepts a false statement

Additional notable parameters

- Completeness error: probability that the Verifier rejects a true statement
- Randomness: number of random coins used by Verifier

(Some of) The known PCP landscape

	Length	Alphabet size	Queries	Soundness error	Possible?
PCP theorem [AS98,ALMSS98]	$poly(n)$	$O(1)$	$O(1)$	$O(1)$	☑

*Some bounds follow from standard hardness assumptions and/or hold for PCPs with non-adaptive verifiers

(Some of) The known PCP landscape

	Length	Alphabet size	Queries	Soundness error	Possible?
PCP theorem [AS98,ALMSS98]	$poly(n)$	$O(1)$	$O(1)$	$O(1)$	☑
Linear-length PCP	$O(n)$	$O(1)$	$O(1)$	$O(1)$	?

*Some bounds follow from standard hardness assumptions and/or hold for PCPs with non-adaptive verifiers

(Some of) The known PCP landscape

	Length	Alphabet size	Queries	Soundness error	Possible?
PCP theorem [AS98,ALMSS98]	$poly(n)$	$O(1)$	$O(1)$	$O(1)$	☑
Linear-length PCP	$O(n)$	$O(1)$	$O(1)$	$O(1)$	?
Sliding-scale Conjecture [BGLR93]	$poly(n)$	$poly(n)$	$O(1)$	$\frac{1}{n}$	?

*Some bounds follow from standard hardness assumptions and/or hold for PCPs with non-adaptive verifiers

(Some of) The known PCP landscape

	Length	Alphabet size	Queries	Soundness error	Possible?
PCP theorem [AS98,ALMSS98]	$poly(n)$	$O(1)$	$O(1)$	$O(1)$	☑
Linear-length PCP	$O(n)$	$O(1)$	$O(1)$	$O(1)$	?
Sliding-scale Conjecture [BGLR93]	$poly(n)$	$poly(n)$	$O(1)$	$\frac{1}{n}$	?
[DHK15]	$poly(n)$	$n^{o(1)}$	$polyloglog(n)$	$\frac{1}{n}$	☑

*Some bounds follow from standard hardness assumptions and/or hold for PCPs with non-adaptive verifiers

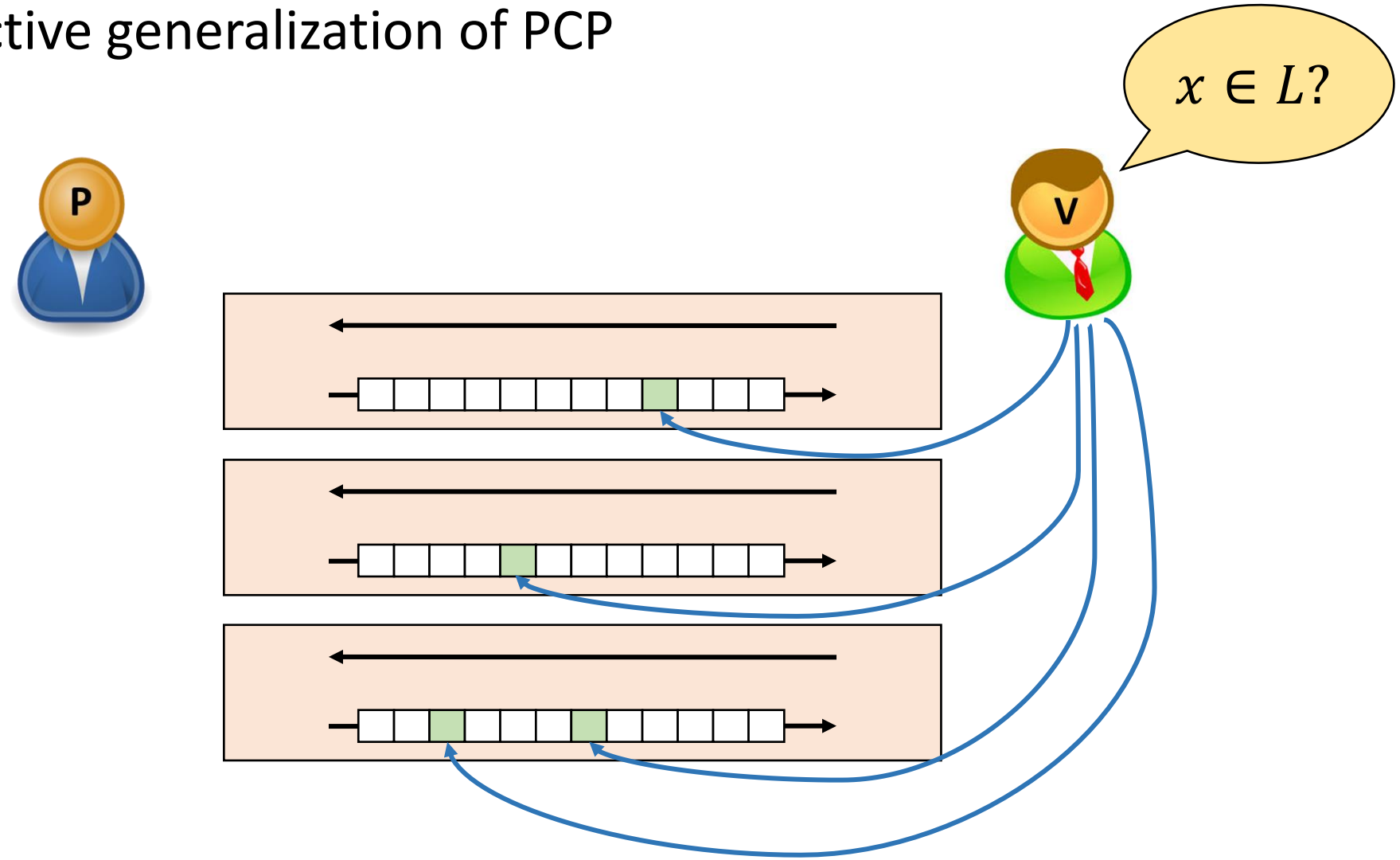
(Some of) The known PCP landscape

	Length	Alphabet size	Queries	Soundness error	Possible?
PCP theorem [AS98,ALMSS98]	$poly(n)$	$O(1)$	$O(1)$	$O(1)$	☑
Linear-length PCP	$O(n)$	$O(1)$	$O(1)$	$O(1)$	?
Sliding-scale Conjecture [BGLR93]	$poly(n)$	$poly(n)$	$O(1)$	$\frac{1}{n}$	?
[DHK15]	$poly(n)$	$n^{o(1)}$	$polyloglog(n)$	$\frac{1}{n}$	☑
Binary-alphabet constant-query PCPs (Folklore)	$2^{o(n)}$	2	2	< 1	☒
	$2^{o(n)}$	2	3	$< \frac{5}{8}$	☒

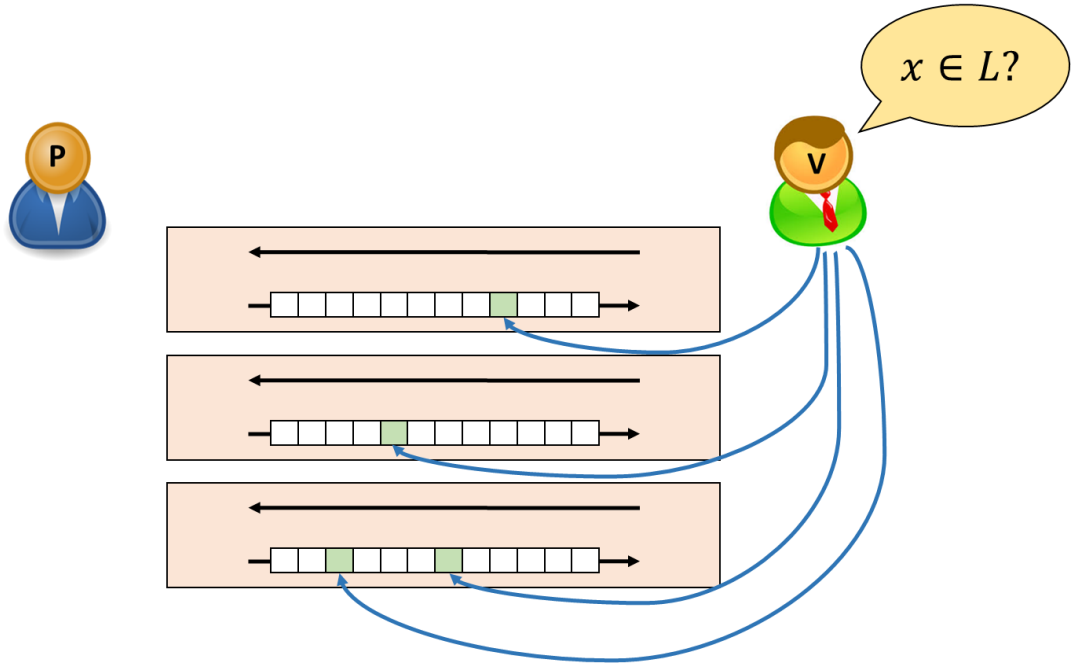
*Some bounds follow from standard hardness assumptions and/or hold for PCPs with non-adaptive verifiers

Interactive Oracle Proofs [BCS16,RRR16]

Interactive generalization of PCP



Interactive Oracle Proofs [BCS16,RRR16]



Main parameters of interest:

- Rounds: number of rounds of interaction
- Length: number of symbols in proof
- Alphabet size: size of the proof alphabet
- Queries: number of symbols of the proof that are read by Verifier
- Soundness error: probability that the Verifier accepts a false statement

Additional notable parameters

- Completeness error: probability that the Verifier rejects a true statement
- Randomness: number of random coins used by Verifier

Why IOPs?

IOPs have proved useful to achieve many results:

- Succinct arguments [BCS16,…]
- Succinct zero-knowledge proofs [KR09,NR22]
- Delegation of computation [RRR16]
- Hardness of approximation [**A**CY21,**A**CY22]

Are IOPs more powerful than PCPs?

	Length	Alphabet size	Queries	Soundness error	Possible for PCPs?	Possible for IOPs?
Linear-length	$O(n)$	$O(1)$	$O(1)$	$O(1)$	?	?
Sliding-scale Conjecture	$poly(n)$	$poly(n)$	$O(1)$	$\frac{1}{n}$	?	?
Binary-alphabet constant-query	$2^{o(n)}$	2	2	< 1	☒	?
	$2^{o(n)}$	2	3	$< \frac{5}{8}$	☒	?

**Some bounds follow from standard hardness assumptions

Are IOPs more powerful than PCPs?

	Length	Alphabet size	Queries	Soundness error	Possible for PCPs?	Possible for IOPs?
Linear-length	$O(n)$	$O(1)$	$O(1)$	$O(1)$	?	☑ [BCGRS17]
Sliding-scale Conjecture	$poly(n)$	$poly(n)$	$O(1)$	$\frac{1}{n}$	?	?
Binary-alphabet constant-query	$2^{o(n)}$	2	2	< 1	☒	?
	$2^{o(n)}$	2	3	$< \frac{5}{8}$	☒	?

**Some bounds follow from standard hardness assumptions

Our results

- We develop a toolbox for deriving barriers for IOPs (and PCPs)

Main results

1. Sliding-scale IOPs are equivalent to sliding-scale PCPs
2. Limitations of binary-alphabet constant-query IOPs
3. Limitations of short IOPs

Our results

- We develop a toolbox for deriving barriers for IOPs (and PCPs)

Main results

1. Sliding-scale IOPs are equivalent to sliding-scale PCPs
2. Limitations of binary-alphabet constant-query IOPs
3. Limitations of short IOPs

Result 1: Sliding-scale IOPs are equivalent to sliding-scale PCPs

We show: a transformation from a sliding-scale IOP with $polylog(n)$ rounds to a sliding-scale PCP.

*Given non-uniform advice or under a hardness assumption

Result 1: Sliding-scale IOPs are equivalent to sliding-scale PCPs

We show: a transformation from a sliding-scale IOP with $polylog(n)$ rounds to a sliding-scale PCP.

*Given non-uniform advice or under a hardness assumption

Negative view: a barrier to constructing sliding-scale IOPs. Constructing sliding-scale IOPs is as hard as constructing sliding-scale PCPs.

Result 1: Sliding-scale IOPs are equivalent to sliding-scale PCPs

We show: a transformation from a sliding-scale IOP with $polylog(n)$ rounds to a sliding-scale PCP.

*Given non-uniform advice or under a hardness assumption

Negative view: a barrier to constructing sliding-scale IOPs. Constructing sliding-scale IOPs is as hard as constructing sliding-scale PCPs.

Positive view: efforts towards constructing sliding-scale PCPs can rely on interaction.

Can IOPs help with PCP limitations?

	Length	Alphabet size	Queries	Soundness error	Possible for PCPs?	Possible for IOPs*?
Linear-length	$O(n)$	$O(1)$	$O(1)$	$O(1)$	?	☑ [BCGRS17]
Sliding-scale Conjecture	$poly(n)$	$poly(n)$	$O(1)$	$\frac{1}{n}$	?	?
Binary-alphabet constant-query	$2^{o(n)}$	2	2	< 1	☒	?
	$2^{o(n)}$	2	3	$< \frac{5}{8}$	☒	?

*IOP results hold for $polylog(n)$ rounds

**Some bounds follow from standard hardness assumptions

Can IOPs help with PCP limitations?

	Length	Alphabet size	Queries	Soundness error	Possible for PCPs?	Possible for IOPs*?
Linear-length	$O(n)$	$O(1)$	$O(1)$	$O(1)$	?	☑ [BCGRS17]
Sliding-scale Conjecture	$poly(n)$	$poly(n)$	$O(1)$	$\frac{1}{n}$	? ⟺ ? equivalent	
Binary-alphabet constant-query	$2^{o(n)}$	2	2	< 1	☒	?
	$2^{o(n)}$	2	3	$< \frac{5}{8}$	☒	?

*IOP results hold for $polylog(n)$ rounds

**Some bounds follow from standard hardness assumptions

Our results

- We develop a toolbox for deriving barriers for IOPs (and PCPs)

Main results

1. Sliding-scale IOPs are equivalent to sliding-scale PCPs
2. Limitations of binary-alphabet constant-query IOPs
3. Limitations of short IOPs

Result 2: Limitations of binary-alphabet constant-query IOPs

Folkore: No binary PCPs for NP with

- 2 queries.
- 3 queries and soundness error $< \frac{5}{8}$.

*Under standard hardness assumptions

Result 2: Limitations of binary-alphabet constant-query IOPs

Folkore: No binary PCPs for NP with

- 2 queries.
- 3 queries and soundness error $< \frac{5}{8}$.

*Under standard hardness assumptions

We show: No $polylog(n)$-round binary IOPs for NP with

- 2 queries and soundness error $< 1 - 2^{-o(n)}$.
- 3 queries and soundness error $< \frac{5}{8} - 2^{-o(n)}$.

**Under standard hardness assumptions

Result 2: Limitations of binary-alphabet constant-query IOPs

Folkore: No binary PCPs for NP with

- 2 queries.
- 3 queries and soundness error $< \frac{5}{8}$.

*Under standard hardness assumptions

We show: No $polylog(n)$-round binary IOPs for NP with

- 2 queries and soundness error $< 1 - 2^{-o(n)}$.
- 3 queries and soundness error $< \frac{5}{8} - 2^{-o(n)}$.

**Under standard hardness assumptions

Takeaway: in this regime, interaction doesn't buy us anything.

Can IOPs help with PCP limitations?

	Length	Alphabet size	Queries	Soundness error	Possible for PCPs?	Possible for IOPs*?
Linear-length	$O(n)$	$O(1)$	$O(1)$	$O(1)$	?	☑ [BCGRS17]
Sliding-scale Conjecture	$poly(n)$	$poly(n)$	$O(1)$	$\frac{1}{n}$	? ⟺ ? equivalent	
Binary-alphabet constant-query	$2^{o(n)}$	2	2	< 1	☒	?
	$2^{o(n)}$	2	3	$< \frac{5}{8}$	☒	?

*IOP results hold for $polylog(n)$ rounds

**Some bounds follow from standard hardness assumptions

Can IOPs help with PCP limitations?

	Length	Alphabet size	Queries	Soundness error	Possible for PCPs?	Possible for IOPs*?
Linear-length	$O(n)$	$O(1)$	$O(1)$	$O(1)$	?	☑ [BCGRS17]
Sliding-scale Conjecture	$poly(n)$	$poly(n)$	$O(1)$	$\frac{1}{n}$	? ⟺ ? equivalent	
Binary-alphabet constant-query	$2^{o(n)}$	2	2	$< 1 - 2^{-o(n)}$	☒	☒
	$2^{o(n)}$	2	3	$< \frac{5}{8} - 2^{-o(n)}$	☒	☒

*IOP results hold for $polylog(n)$ rounds

**Some bounds follow from standard hardness assumptions

Our results

- We develop a toolbox for deriving barriers for IOPs (and PCPs)

Main results

1. Sliding-scale IOPs are equivalent to sliding-scale PCPs
2. Limitations of binary-alphabet constant-query IOPs
3. Limitations of short IOPs

Result 3: Limitations of short IOPs

We show: no short IOPs in the [DHK15] parameter regime

	Proof System	Rounds	Length	Alphabet size	Queries	Soundness error	Possible?
[DHK15]	PCP	0	$poly(n)$	$n^{o(1)}$	$polyloglog(n)$	$\frac{1}{n}$	☑
[This Work]	IOP	$polylog(n)$	$\tilde{O}(n)$	$n^{polylog(n)}$	$polyloglog(n)$	$\frac{1}{n}$	☒

*full theorem considers more general trade-offs between parameters

Our results

- We develop a toolbox for deriving barriers for IOPs (and PCPs)

Main results

1. Sliding-scale IOPs are equivalent to sliding-scale PCPs
2. Limitations of binary-alphabet constant-query IOPs
3. Limitations of short IOPs

Our results

- We develop a toolbox for deriving barriers for IOPs (and PCPs)

Main results

1. Sliding-scale IOPs are equivalent to sliding-scale PCPs
2. Limitations of binary-alphabet constant-query IOPs
3. Limitations of short IOPs

Additional barriers: related to interactive proofs

Our toolbox

Round and length reduction:

- Round reduction
- Length reduction
- Unrolling to PCP

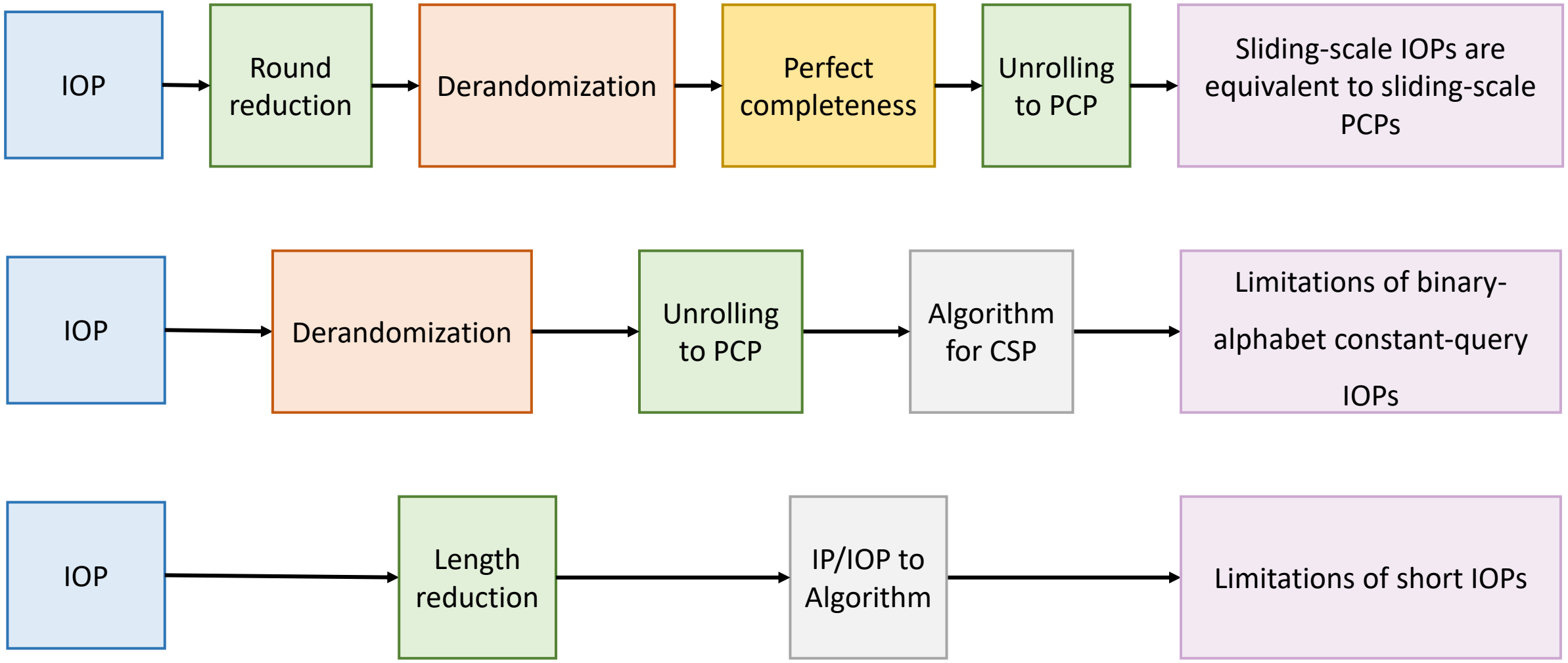
Improving completeness:

- Completeness amplification
- Perfect completeness

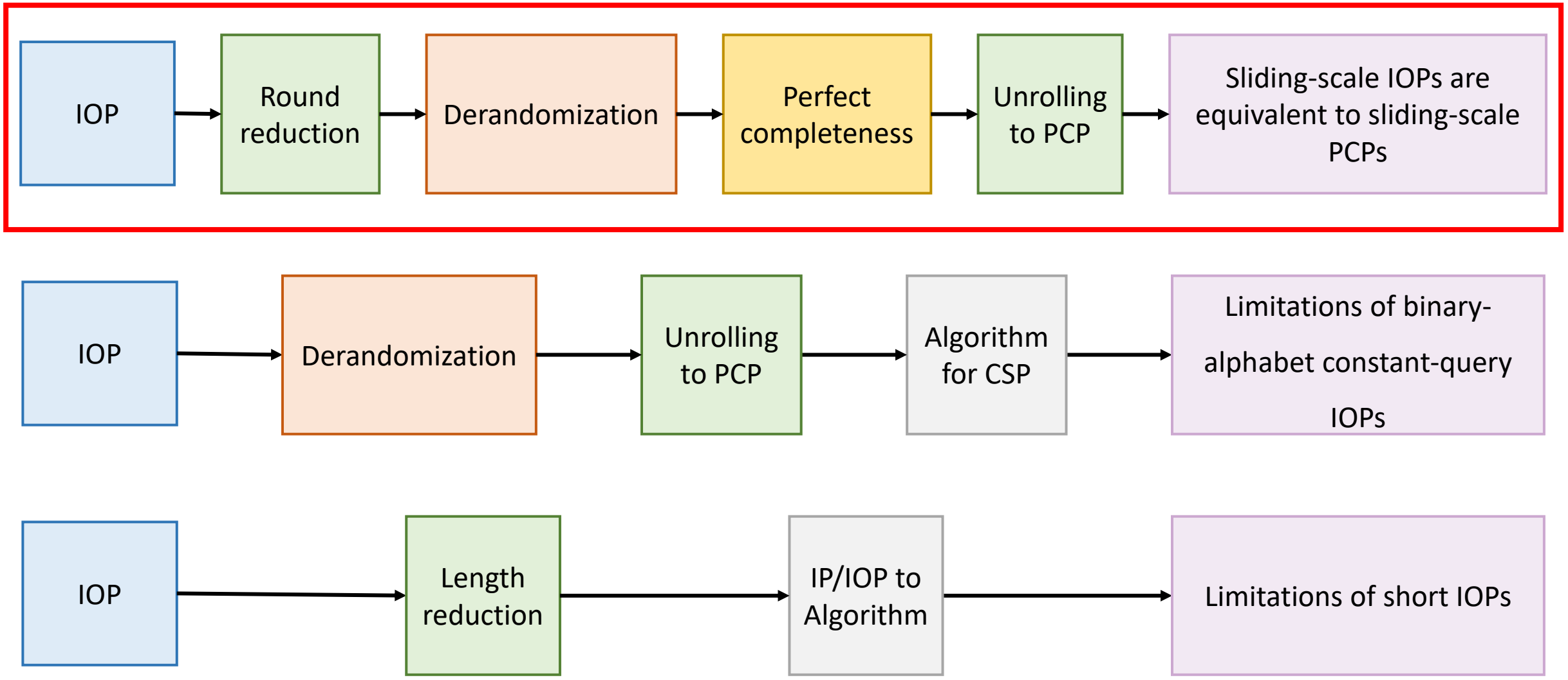
Derandomizing IOPs:

- Using non-uniform advice
- Using PRGs

Using the toolbox to get our results



Using the toolbox to get our results



Sliding-scale IOP to sliding-scale PCP

Sliding-scale IOP to sliding-scale PCP

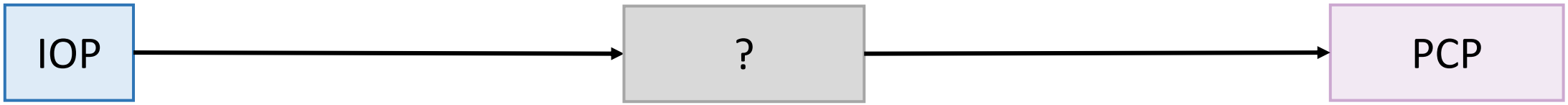
Theorem:*

$$\mathrm{IOP}\left[\begin{array}{c}\text{Rounds} = \text{polylog}(n)\\ \text{Soundness error} = 1/\text{poly(n)}\\ \text{Queries} = O(1)\\ \text{Alphabet size} = poly(n)\\ \text{Length} = poly(n)\end{array}\right] \subseteq \mathrm{PCP}\left[\begin{array}{c}\text{Soundness error} = 1/\text{poly(n)}\\ \text{Queries} = O(1)\\ \text{Alphabet size} = poly(n)\\ \text{Length} = poly(n)\end{array}\right]$$

*Under complexity assumptions or using non-uniform advice

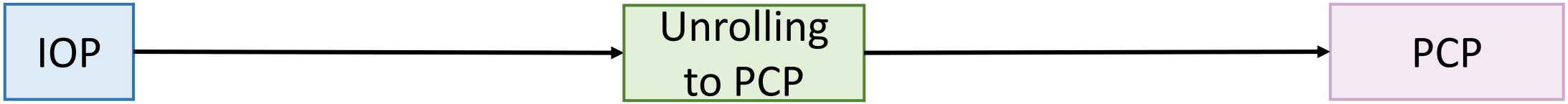
Sliding-scale IOP to sliding-scale PCP

Proof sketch:



Sliding-scale IOP to sliding-scale PCP

Proof sketch:

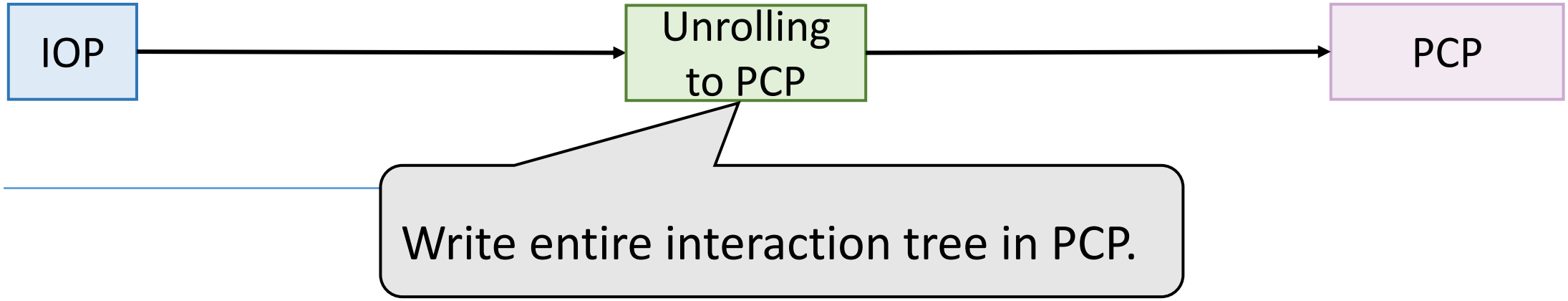


	Length	Rounds	Randomness
IOP	$poly(n)$	$polylog(n)$	$poly(n)$
Unroll to PCP	$2^{poly(n)}$	0	$poly(n)$

For all steps: Query complexity = $O(1)$ Soundness error = $1/poly(n)$

Sliding-scale IOP to sliding-scale PCP

Proof sketch:

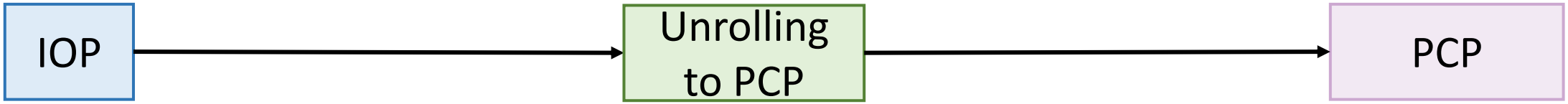


IOP	$poly(n)$	$polylog(n)$	$poly(n)$
Unroll to PCP	$2^{poly(n)}$	0	$poly(n)$

For all steps: Query complexity = $O(1)$ Soundness error = $1/poly(n)$

Sliding-scale IOP to sliding-scale PCP

Proof sketch:

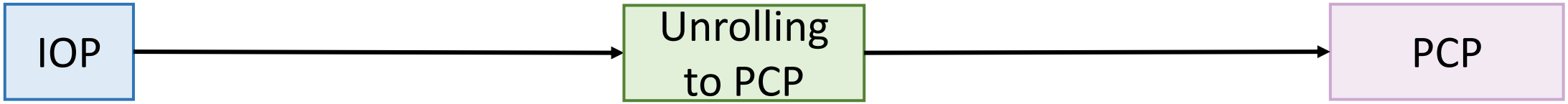


	Length	Rounds	Randomness
IOP	$poly(n)$	$polylog(n)$	$poly(n)$
Unroll to PCP	$2^{poly(n)}$	0	$poly(n)$

For all steps: Query complexity = $O(1)$ Soundness error = $1/poly(n)$

Sliding-scale IOP to sliding-scale PCP

Proof sketch:



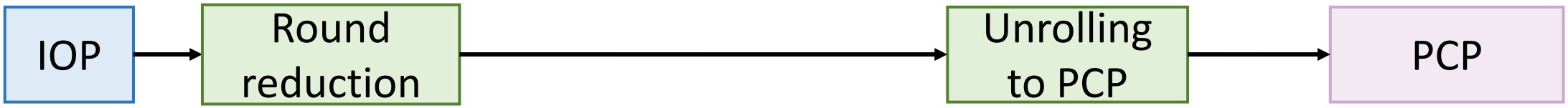
	Length	Rounds	Randomness
IOP	$poly(n)$	$polylog(n)$	$poly(n)$
Unroll to PCP	$2^{poly(n)}$	0	$poly(n)$

Length is exponential in #rounds and randomness complexity

For all steps: Query complexity = $O(1)$ Soundness error = $1/poly(n)$

Sliding-scale IOP to sliding-scale PCP

Proof sketch:

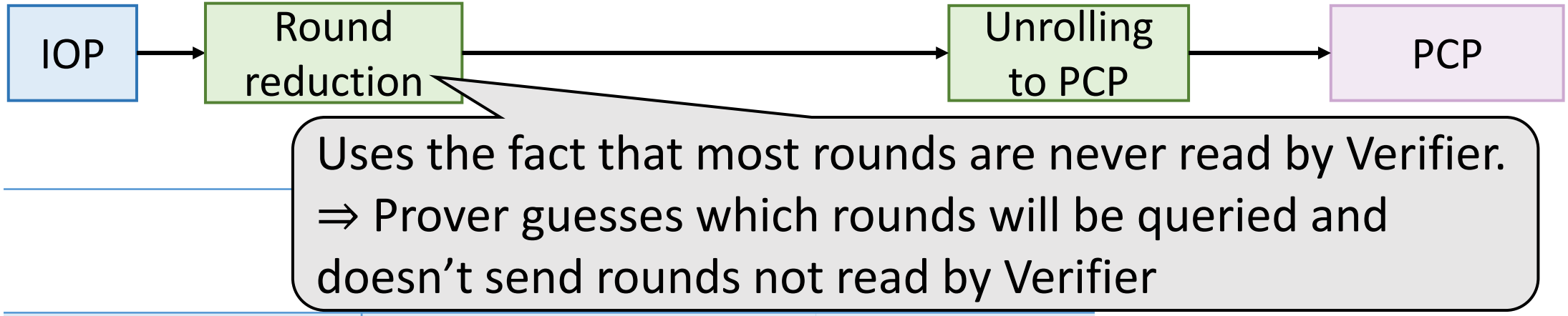


	Length	Rounds	Randomness
IOP	$poly(n)$	$polylog(n)$	$poly(n)$
Round reduction	$poly(n)$	$O(1)$	$poly(n)$
Unroll to PCP	$2^{poly(n)}$	0	$poly(n)$

For all steps: Query complexity = $O(1)$ Soundness error = $1/poly(n)$

Sliding-scale IOP to sliding-scale PCP

Proof sketch:



IOP	$poly(n)$	$polylog(n)$	$poly(n)$
Round reduction	$poly(n)$	$O(1)$	$poly(n)$
Unroll to PCP	$2^{poly(n)}$	0	$poly(n)$

For all steps: Query complexity = $O(1)$ Soundness error = $1/poly(n)$

Sliding-scale IOP to sliding-scale PCP

Proof sketch:

	Length	Rounds	Randomness
IOP	$poly(n)$	$polylog(n)$	$poly(n)$
Round reduction	$poly(n)$	$O(1)$	$poly(n)$
Derandomization	$poly(n)$	$O(1)$	$O(logn)$
Unroll to PCP	$poly(n)$	0	$O(logn)$

For all steps: Query complexity = $O(1)$ Soundness error = $1/poly(n)$

Sliding-scale IOP to sliding-scale PCP

Proof sketch:

Similar to derandomization of IPs.
Verifier samples from a subset of the random coins

	Length		
IOP	$poly(n)$	$polylog(n)$	$poly(n)$
Round reduction	$poly(n)$	$O(1)$	$poly(n)$
Derandomization	$poly(n)$	$O(1)$	$O(logn)$
Unroll to PCP	$poly(n)$	0	$O(logn)$

For all steps: Query complexity = $O(1)$ Soundness error = $1/poly(n)$

Sliding-scale IOP to sliding-scale PCP

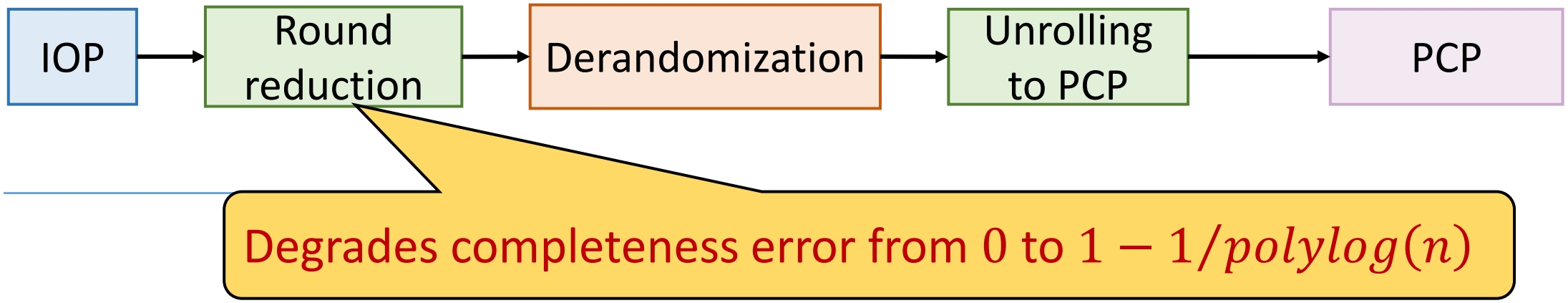
Proof sketch:

	Length	**Rounds**	**Randomness**
IOP	$poly(n)$	$polylog(n)$	$poly(n)$
Round reduction	$poly(n)$	$O(1)$	$poly(n)$
Derandomization	$poly(n)$	$O(1)$	$O(logn)$
Unroll to PCP	$poly(n)$	0	$O(logn)$

For all steps: Query complexity = $O(1)$ Soundness error = $1/poly(n)$

Sliding-scale IOP to sliding-scale PCP

Proof sketch:



IOP	$poly(n)$	$polylog(n)$	$poly(n)$
Round reduction	$poly(n)$	$O(1)$	$poly(n)$
Derandomization	$poly(n)$	$O(1)$	$O(logn)$
Unroll to PCP	$poly(n)$	0	$O(logn)$

For all steps: Query complexity = $O(1)$ Soundness error $= 1/poly(n)$

Sliding-scale IOP to sliding-scale PCP

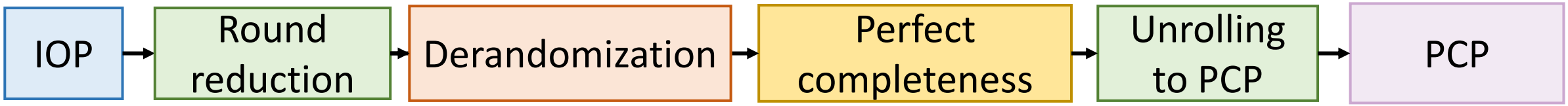
Proof sketch:

	Length	Rounds	Randomness	Completeness error
IOP	$poly(n)$	$polylog(n)$	$poly(n)$	0
Round reduction	$poly(n)$	$O(1)$	$poly(n)$	$1 - 1/polylog(n)$
Derandomization	$poly(n)$	$O(1)$	$O(logn)$	$1 - 1/polylog(n)$
Unroll to PCP	$poly(n)$	0	$O(logn)$	$1 - 1/polylog(n)$

For all steps: Query complexity = $O(1)$ Soundness error = $1/poly(n)$

Sliding-scale IOP to sliding-scale PCP

Proof sketch:

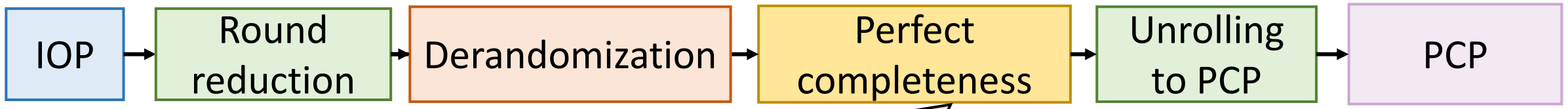


	Length	Rounds	Randomness	Completeness error
IOP	$poly(n)$	$polylog(n)$	$poly(n)$	0
Round reduction	$poly(n)$	$O(1)$	$poly(n)$	$1 - 1/polylog(n)$
Derandomization	$poly(n)$	$O(1)$	$O(logn)$	$1 - 1/polylog(n)$
Perfect completeness	$poly(n)$	$O(1)$	$O(logn)$	0
Unroll to PCP	$poly(n)$	0	$O(logn)$	0

For all steps: Query complexity = $O(1)$ Soundness error = $1/poly(n)$

Sliding-scale IOP to sliding-scale PCP

Proof sketch:



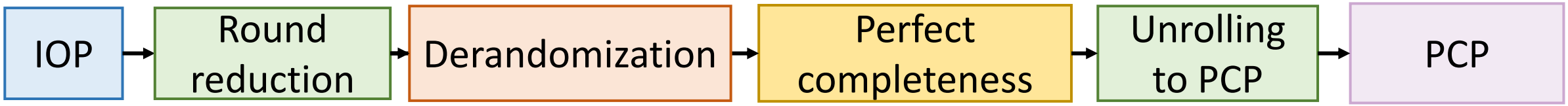
Based on [FGMSZ89] for perfect completeness of IPs, taking care not to harm query complexity

	Leng			
IOP	$poly(n)$	$polylog(n)$	$poly(n)$	0
Round reduction	$poly(n)$	$O(1)$	$poly(n)$	$1 - 1/polylog(n)$
Derandomization	$poly(n)$	$O(1)$	$O(logn)$	$1 - 1/polylog(n)$
Perfect completeness	$poly(n)$	$O(1)$	$O(logn)$	0
Unroll to PCP	$poly(n)$	0	$O(logn)$	0

For all steps: Query complexity = $O(1)$ Soundness error = $1/poly(n)$

Sliding-scale IOP to sliding-scale PCP

Proof sketch:

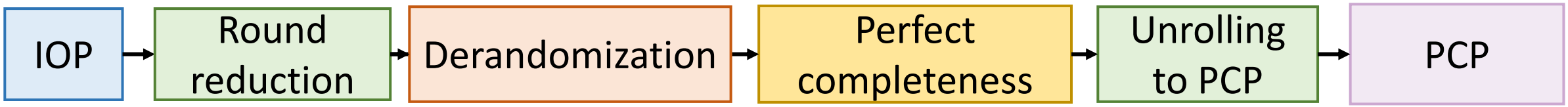


	Length	Rounds	Randomness	Completeness error
IOP	$poly(n)$	$polylog(n)$	$poly(n)$	0
Round reduction	$poly(n)$	$O(1)$	$poly(n)$	$1 - 1/polylog(n)$
Derandomization	$poly(n)$	$O(1)$	$O(logn)$	$1 - 1/polylog(n)$
Perfect completeness	$poly(n)$	$O(1)$	$O(logn)$	0
Unroll to PCP	$poly(n)$	0	$O(logn)$	0

For all steps: Query complexity = $O(1)$ Soundness error = $1/poly(n)$

Sliding-scale IOP to sliding-scale PCP

Proof sketch:



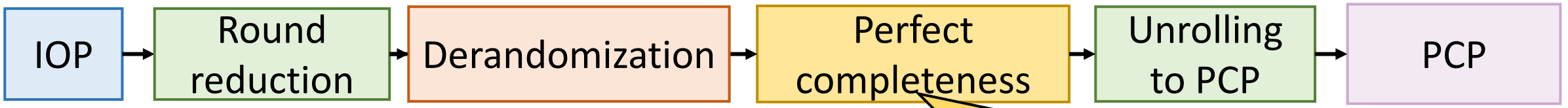
Order of steps matters!

		Rounds	Randomness	Completeness error
IOP	$poly(n)$	$polylog(n)$	$poly(n)$	0
Round reduction	$poly(n)$	$O(1)$	$poly(n)$	$1 - 1/polylog(n)$
Derandomization	$poly(n)$	$O(1)$	$O(logn)$	$1 - 1/polylog(n)$
Perfect completeness	$poly(n)$	$O(1)$	$O(logn)$	0
Unroll to PCP	$poly(n)$	0	$O(logn)$	0

For all steps: Query complexity = $O(1)$ Soundness error = $1/poly(n)$

Sliding-scale IOP to sliding-scale PCP

Proof sketch:



Order of steps matters!

Increases alphabet size as a factor of randomness complexity

		Rounds		error
IOP	$poly(n)$	$polylog(n)$	$poly(n)$	0
Round reduction	$poly(n)$	$O(1)$	$poly(n)$	$1 - 1/polylog(n)$
Derandomization	$poly(n)$	$O(1)$	$O(logn)$	$1 - 1/polylog(n)$
Perfect completeness	$poly(n)$	$O(1)$	$O(logn)$	0
Unroll to PCP	$poly(n)$	0	$O(logn)$	0

For all steps: Query complexity = $O(1)$ Soundness error = $1/poly(n)$

Summary of results

- We develop a toolbox for deriving barriers for IOPs (and PCPs)

Main results

1. Sliding-scale IOPs are equivalent to sliding-scale PCPs
2. Limitations of binary-alphabet constant-query IOPs
3. Limitations of short IOPs

Summary of results

- We develop a toolbox for deriving barriers for IOPs (and PCPs)

Main results

1. Sliding-scale IOPs are equivalent to sliding-scale PCPs
2. Limitations of binary-alphabet constant-query IOPs
3. Limitations of short IOPs

General question: when are IOPs more powerful than PCPs?

Summary of results

- We develop a toolbox for deriving barriers for IOPs (and PCPs)

Main results

1. Sliding-scale IOPs are equivalent to sliding-scale PCPs
2. Limitations of binary-alphabet constant-query IOPs
3. Limitations of short IOPs

General question: when are IOPs more powerful than PCPs?

THANKS!