Foundations of Computing

Lecture 25

Arkady Yerukhimovich

April 23, 2024

Outline

Lecture 24 Review

- Interactive Proofs
- Proof for Graph Non-Isomorphism
- Polynomial Identity Testing

Outline

Reviewing the Definition of $\mathcal{IP}$

Definition of $\mathcal{IP}$

$L \in \mathcal{IP}$ if there exist a pair of interactive algorithms (P, V) with V being poly-time (in $|x|$) s.t.

1. (Completeness) If $x \in L$, then $\Pr[\langle P, V\rangle(x) = 1] = 1$

Reviewing the Definition of $\mathcal{IP}$

Definition of $\mathcal{IP}$

$L \in \mathcal{IP}$ if there exist a pair of interactive algorithms (P, V) with V being poly-time (in $|x|$) s.t.

1. (Completeness) If $x \in L$, then $\Pr[\langle P, V\rangle(x) = 1] = 1$
2. (Soundness) If $x \notin L$, then for any (possibly unbounded) P^*, we have $\Pr[\langle P^*, V\rangle(x) = 1] \leq 1/2$

Reviewing the Definition of $\mathcal{IP}$

Definition of $\mathcal{IP}$

$L \in \mathcal{IP}$ if there exist a pair of interactive algorithms (P, V) with V being poly-time (in $|x|$) s.t.

1. (Completeness) If $x \in L$, then $\Pr[\langle P, V\rangle(x) = 1] = 1$
2. (Soundness) If $x \notin L$, then for any (possibly unbounded) P^*, we have $\Pr[\langle P^*, V\rangle(x) = 1] \leq 1/2$

A New Property

We say that a proof is *zero-knowledge* if the verifier learns nothing (other than the truth of the statement) from seeing the proof.

An Example – Where's Waldo

An Example

A Second Example – Puppy and Panda

Outline

Defining Knowledge

Question

What does it mean for a machine to know/learn something?

Defining Knowledge

Question

What does it mean for a machine to know/learn something?

Answer: A poly-time TM M "knows" x, if it can output x after an efficient computation.

Defining Knowledge

Question

What does it mean for a machine to know/learn something?

Answer: A poly-time TM M “knows” x, if it can output x after an efficient computation.

Question

What does it mean for a machine to learn nothing from a proof?

Defining Knowledge

Question

What does it mean for a machine to know/learn something?

Answer: A poly-time TM M "knows" x, if it can output x after an efficient computation.

Question

What does it mean for a machine to learn nothing from a proof?

Answer: Whatever it can (efficiently) compute after seeing the proof, it could have efficiently computed before seeing the proof.

Formalizing the Definition

Consider an interactive proof between Prover (P) and Verifier (V):

$$\langle P, V \rangle(x)$$

Formalizing the Definition

Consider an interactive proof between Prover (P) and Verifier (V):

$$\langle P, V\rangle(x)$$

Define V's view of this interaction by:

$$VIEW_V(\langle P, V\rangle(x))$$

Formalizing the Definition

Consider an interactive proof between Prover (P) and Verifier (V):

$$\langle P, V \rangle(x)$$

Define V's view of this interaction by:

$$VIEW_V(\langle P, V \rangle(x))$$

This includes:

- V's randomness
- Any messages that V receives

Formalizing the Definition

Consider an interactive proof between Prover (P) and Verifier (V):

$$\langle P, V \rangle(x)$$

Define V's view of this interaction by:

$$VIEW_V(\langle P, V \rangle(x))$$

This includes:

- V's randomness
- Any messages that V receives

Zero-Knowledge Proof

A proof $\langle P, V \rangle(x)$ for a language L is *zero-knowledge* if

Formalizing the Definition

Consider an interactive proof between Prover (P) and Verifier (V):

$$\langle P, V\rangle(x)$$

Define V's view of this interaction by:

$$VIEW_V(\langle P, V\rangle(x))$$

This includes:

- V's randomness
- Any messages that V receives

Zero-Knowledge Proof

A proof $\langle P, V\rangle(x)$ for a language L is *zero-knowledge* if

- For any (possibly malicious) poly-time verifier V^*

Formalizing the Definition

Consider an interactive proof between Prover (P) and Verifier (V):

$$\langle P, V \rangle(x)$$

Define V's view of this interaction by:

$$VIEW_V(\langle P, V \rangle(x))$$

This includes:

- V's randomness
- Any messages that V receives

Zero-Knowledge Proof

A proof $\langle P, V \rangle(x)$ for a language L is *zero-knowledge* if

- For any (possibly malicious) poly-time verifier V^*
- There exists a poly-time *Simulator* S s.t.

$$\forall x \in L, \qquad VIEW_{V^*}(\langle P, V^* \rangle(x)) = S(x)$$

Parsing the Definition

Zero-Knowledge Proof

A proof $\langle P, V\rangle(x)$ for a language L is *zero-knowledge* if

- For any (possibly malicious) poly-time verifier V^*
- There exists a poly-time *Simulator* S s.t.

$$\forall x \in L, \qquad VIEW_{V^*}(\langle P, V^*\rangle(x)) = S(x)$$

Parsing the Definition

Zero-Knowledge Proof

A proof $\langle P, V \rangle(x)$ for a language L is *zero-knowledge* if

- For any (possibly malicious) poly-time verifier V^*
- There exists a poly-time *Simulator* S s.t.

$$\forall x \in L, \qquad VIEW_{V^*}(\langle P, V^* \rangle(x)) = S(x)$$

- $S(x)$ captures what V knows about x

Parsing the Definition

Zero-Knowledge Proof

A proof $\langle P, V\rangle(x)$ for a language L is *zero-knowledge* if

- For any (possibly malicious) poly-time verifier V^*
- There exists a poly-time *Simulator* S s.t.

$$\forall x \in L, \qquad VIEW_{V^*}(\langle P, V^*\rangle(x)) = S(x)$$

- $S(x)$ captures what V knows about x
- If S can produce V's view in the proof, then this everything in this view is "known" to V before the proof.

Parsing the Definition

Zero-Knowledge Proof

A proof $\langle P, V\rangle(x)$ for a language L is *zero-knowledge* if

- For any (possibly malicious) poly-time verifier V^*
- There exists a poly-time *Simulator* S s.t.

$$\forall x \in L, \qquad VIEW_{V^*}(\langle P, V^*\rangle(x)) = S(x)$$

- $S(x)$ captures what V knows about x
- If S can produce V's view in the proof, then this everything in this view is "known" to V before the proof.
- Thus, the proof is zero-knowledge: V learns nothing more than that $x \in L$

Parsing the Definition

Zero-Knowledge Proof

A proof $\langle P, V \rangle(x)$ for a language L is *zero-knowledge* if

- For any (possibly malicious) poly-time verifier V^*
- There exists a poly-time *Simulator* S s.t.

$$\forall x \in L, \qquad VIEW_{V^*}(\langle P, V^* \rangle(x)) = S(x)$$

- $S(x)$ captures what V knows about x
- If S can produce V's view in the proof, then this everything in this view is "known" to V before the proof.
- Thus, the proof is zero-knowledge: V learns nothing more than that $x \in L$
- IMPORTANT: $VIEW_V^*$ and $S(x)$ are both distributions, not values. So, equality is of distributions

Outline

Where's Waldo

Puppy and Panda

flip

← flip

S

1. choose flip / no flip
output the

$VIEW_V = (win, win)$

Graph Isomorphism

Input: $x = (G_0, G_1)$
Prover's goal: Prove that he knows permutation π s.t. $\pi(G_0) = G_1$

Graph Isomorphism

Input: $x = (G_0, G_1)$
Prover's goal: Prove that he knows permutation π s.t. $\pi(G_0) = G_1$

The Proof

Graph Isomorphism

Input: $x = (G_0, G_1)$
Prover's goal: Prove that he knows permutation π s.t. $\pi(G_0) = G_1$

The Proof

1. P chooses $b \leftarrow \{0,1\}$ and a random permutation σ and sends $H = \sigma(G_b)$ to V

Graph Isomorphism

Input: $x = (G_0, G_1)$
Prover's goal: Prove that he knows permutation π s.t. $\pi(G_0) = G_1$

The Proof

1. P chooses $b \leftarrow \{0,1\}$ and a random permutation σ and sends $H = \sigma(G_b)$ to V
2. V chooses $b' \leftarrow \{0,1\}$ and sends it to P

Graph Isomorphism

Input: $x = (G_0, G_1)$
Prover's goal: Prove that he knows permutation π s.t. $\pi(G_0) = G_1$

The Proof

1. P chooses $b \leftarrow \{0,1\}$ and a random permutation σ and sends $H = \sigma(G_b)$ to V
2. V chooses $b' \leftarrow \{0,1\}$ and sends it to P
3. P sends V the permutation π' mapping $G_{b'}$ to H

$$\pi' = \begin{cases} \sigma & \text{if } b = b' \\ \sigma\pi^{-1} & \text{if } b = 0, b' = 1 \\ \sigma\pi & \text{if } b = 1, b' = 0 \end{cases}$$

Graph Isomorphism

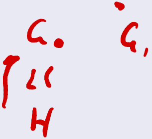
Input: $x = (G_0, G_1)$
Prover's goal: Prove that he knows permutation π s.t. $\pi(G_0) = G_1$

The Proof

1. P chooses $b \leftarrow \{0,1\}$ and a random permutation σ and sends $H = \sigma(G_b)$ to V
2. V chooses $b' \leftarrow \{0,1\}$ and sends it to P
3. P sends V the permutation π' mapping $G_{b'}$ to H

$$\pi' = \begin{cases} \sigma & \text{if } b = b' \\ \sigma\pi^{-1} & \text{if } b = 0, b' = 1 \\ \sigma\pi & \text{if } b = 1, b' = 0 \end{cases}$$

4. V accepts iff $H = \pi'(G_{b'})$



Graph Isomorphism

The Proof

1. P chooses $b \leftarrow \{0,1\}$ and a random permutation σ and sends $H = \sigma(G_b)$ to V
2. V chooses $b' \leftarrow \{0,1\}$ and sends it to P
3. P sends V the permutation π' mapping $G_{b'}$ to H

$$\pi' = \begin{cases} \sigma & \text{if } b = b' \\ \sigma\pi^{-1} & \text{if } b = 0, b' = 1 \\ \sigma\pi & \text{if } b = 1, b' = 0 \end{cases}$$

4. V accepts iff $H = \pi'(G_{b'})$

Graph Isomorphism

The Proof

1. P chooses $b \leftarrow \{0,1\}$ and a random permutation σ and sends $H = \sigma(G_b)$ to V
2. V chooses $b' \leftarrow \{0,1\}$ and sends it to P
3. P sends V the permutation π' mapping $G_{b'}$ to H

$$\pi' = \begin{cases} \sigma & \text{if } b = b' \\ \sigma\pi^{-1} & \text{if } b = 0, b' = 1 \\ \sigma\pi & \text{if } b = 1, b' = 0 \end{cases}$$

4. V accepts iff $H = \pi'(G_{b'})$

1. Completeness: If $\pi(G_0) = G_1$, then π' correctly maps $G_{b'}$ to H

Graph Isomorphism

The Proof

1. P chooses $b \leftarrow \{0,1\}$ and a random permutation σ and sends $H = \sigma(G_b)$ to V
2. V chooses $b' \leftarrow \{0,1\}$ and sends it to P
3. P sends V the permutation π' mapping $G_{b'}$ to H

$$\pi' = \begin{cases} \sigma & \text{if } b = b' \\ \sigma\pi^{-1} & \text{if } b = 0, b' = 1 \\ \sigma\pi & \text{if } b = 1, b' = 0 \end{cases}$$

4. V accepts iff $H = \pi'(G_{b'})$

1. Completeness: If $\pi(G_0) = G_1$, then π' correctly maps $G_{b'}$ to H
2. Soundness: Suppose G_0 is not isomorphic to G_1, so there is no such π. Then, if $b \neq b'$, there is no permutation that P can give that V will accept

Graph Isomorphism

The Proof

1. P chooses $b \leftarrow \{0,1\}$ and a random permutation σ and sends $H = \sigma(G_b)$ to V
2. V chooses $b' \leftarrow \{0,1\}$ and sends it to P
3. P sends V the permutation π' mapping $G_{b'}$ to H
$$\pi' = \begin{cases} \sigma & \text{if } b = b' \\ \sigma\pi^{-1} & \text{if } b = 0, b' = 1 \\ \sigma\pi & \text{if } b = 1, b' = 0 \end{cases}$$
4. V accepts iff $H = \pi'(G_{b'})$

Graph Isomorphism

The Proof

1. P chooses $b \leftarrow \{0,1\}$ and a random permutation σ and sends $H = \sigma(G_b)$ to V
2. V chooses $b' \leftarrow \{0,1\}$ and sends it to P
3. P sends V the permutation π' mapping $G_{b'}$ to H
$$\pi' = \begin{cases} \sigma & \text{if } b = b' \\ \sigma\pi^{-1} & \text{if } b = 0, b' = 1 \\ \sigma\pi & \text{if } b = 1, b' = 0 \end{cases}$$
4. V accepts iff $H = \pi'(G_{b'})$

Zero-Knowledge

The Proof

1. P chooses $b \leftarrow \{0, 1\}$ and a random permutation σ and sends $H = \sigma(G_b)$ to V
2. V chooses $b' \leftarrow \{0, 1\}$ and sends it to P
3. P sends V the permutation π' mapping $G_{b'}$ to H
$$\pi' = \begin{cases} \sigma & \text{if } b = b' \\ \sigma\pi^{-1} & \text{if } b = 0, b' = 1 \\ \sigma\pi & \text{if } b = 1, b' = 0 \end{cases}$$
4. V accepts iff $H = \pi'(G_{b'})$

Zero-Knowledge

We define simulator $S(G_0, G_1)$ as follows:

Graph Isomorphism

The Proof

1. P chooses $b \leftarrow \{0,1\}$ and a random permutation σ and sends $H = \sigma(G_b)$ to V
2. V chooses $b' \leftarrow \{0,1\}$ and sends it to P
3. P sends V the permutation π' mapping $G_{b'}$ to H
$$\pi' = \begin{cases} \sigma & \text{if } b = b' \\ \sigma\pi^{-1} & \text{if } b = 0, b' = 1 \\ \sigma\pi & \text{if } b = 1, b' = 0 \end{cases}$$
4. V accepts iff $H = \pi'(G_{b'})$

Zero-Knowledge

We define simulator $S(G_0, G_1)$ as follows:

1. S chooses $b \leftarrow \{0,1\}$ and a random permutation σ

Graph Isomorphism

The Proof

1. P chooses $b \leftarrow \{0,1\}$ and a random permutation σ and sends $H = \sigma(G_b)$ to V
2. V chooses $b' \leftarrow \{0,1\}$ and sends it to P
3. P sends V the permutation π' mapping $G_{b'}$ to H
$$\pi' = \begin{cases} \sigma & \text{if } b = b' \\ \sigma\pi^{-1} & \text{if } b = 0, b' = 1 \\ \sigma\pi & \text{if } b = 1, b' = 0 \end{cases}$$
4. V accepts iff $H = \pi'(G_{b'})$

Zero-Knowledge

We define simulator $S(G_0, G_1)$ as follows:

1. S chooses $b \leftarrow \{0,1\}$ and a random permutation σ
2. Set $H = \sigma(G_b)$ and let $b' = V^*(G_0, G_1, H)$

Graph Isomorphism

The Proof

1. P chooses $b \leftarrow \{0,1\}$ and a random permutation σ and sends $H = \sigma(G_b)$ to V
2. V chooses $b' \leftarrow \{0,1\}$ and sends it to P
3. P sends V the permutation π' mapping $G_{b'}$ to H
$$\pi' = \begin{cases} \sigma & \text{if } b = b' \\ \sigma\pi^{-1} & \text{if } b = 0, b' = 1 \\ \sigma\pi & \text{if } b = 1, b' = 0 \end{cases}$$
4. V accepts iff $H = \pi'(G_{b'})$

Zero-Knowledge

We define simulator $S(G_0, G_1)$ as follows:

1. S chooses $b \leftarrow \{0,1\}$ and a random permutation σ
2. Set $H = \sigma(G_b)$ and let $b' = V^*(G_0, G_1, H)$
3. If $b' = b$, output (b', H, σ). Otherwise, restart with new σ, b.

Zero-Knowledge

Zero-Knowledge

We define simulator $S(G_0, G_1)$ as follows:

1. S chooses $b \leftarrow \{0,1\}$ and a random permutation σ
2. Set $H = \sigma(G_b)$ and let $b' = V^*(G_0, G_1, H)$
3. If $b' = b$, output (b', H, σ). Otherwise, restart with new σ, b.

Zero-Knowledge

Zero-Knowledge

We define simulator $S(G_0, G_1)$ as follows:

1. S chooses $b \leftarrow \{0, 1\}$ and a random permutation σ
2. Set $H = \sigma(G_b)$ and let $b' = V^*(G_0, G_1, H)$
3. If $b' = b$, output (b', H, σ). Otherwise, restart with new σ, b.

Observations:

Zero-Knowledge

Zero-Knowledge

We define simulator $S(G_0, G_1)$ as follows:

1. S chooses $b \leftarrow \{0, 1\}$ and a random permutation σ
2. Set $H = \sigma(G_b)$ and let $b' = V^*(G_0, G_1, H)$
3. If $b' = b$, output (b', H, σ). Otherwise, restart with new σ, b.

Observations:

- If $b' = b$, then S's simulation is perfect.

Zero-Knowledge

Zero-Knowledge

We define simulator $S(G_0, G_1)$ as follows:

1. S chooses $b \leftarrow \{0, 1\}$ and a random permutation σ
2. Set $H = \sigma(G_b)$ and let $b' = V^*(G_0, G_1, H)$
3. If $b' = b$, output (b', H, σ). Otherwise, restart with new σ, b.

Observations:

- If $b' = b$, then S's simulation is perfect.
- If G_0 and G_1 are isomorphic, then b' cannot depend on b

Zero-Knowledge

Zero-Knowledge

We define simulator $S(G_0, G_1)$ as follows:

1. S chooses $b \leftarrow \{0, 1\}$ and a random permutation σ
2. Set $H = \sigma(G_b)$ and let $b' = V^*(G_0, G_1, H)$
3. If $b' = b$, output (b', H, σ). Otherwise, restart with new σ, b.

Observations:

- If $b' = b$, then S's simulation is perfect.
- If G_0 and G_1 are isomorphic, then b' cannot depend on b
- So, $b = b'$ with probability $1/2$. Thus, S expected to stop after two rounds

Zero-Knowledge

Zero-Knowledge

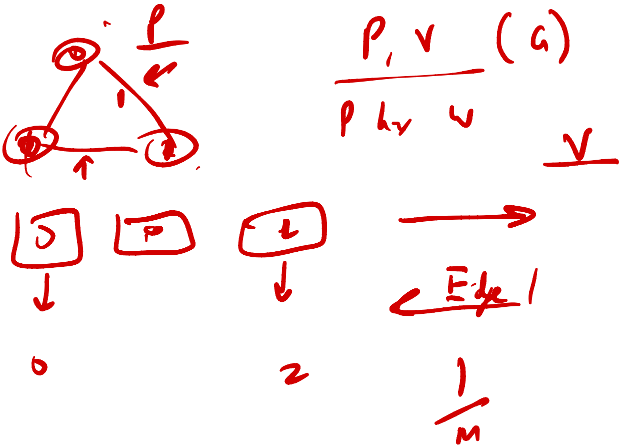
We define simulator $S(G_0, G_1)$ as follows:

1. S chooses $b \leftarrow \{0,1\}$ and a random permutation σ
2. Set $H = \sigma(G_b)$ and let $b' = V^*(G_0, G_1, H)$
3. If $b' = b$, output (b', H, σ). Otherwise, restart with new σ, b.

Observations:

- If $b' = b$, then S's simulation is perfect.
- If G_0 and G_1 are isomorphic, then b' cannot depend on b
- So, $b = b'$ with probability $1/2$. Thus, S expected to stop after two rounds
- When S stops, he produces a perfect simulation

Graph 3-Coloring



Outline

Zero-Knowledge (ZK) Proofs on the Blockchain

ZK proofs have found practical importance in critical Blockchain applications:

Zero-Knowledge (ZK) Proofs on the Blockchain

ZK proofs have found practical importance in critical Blockchain applications:

Blockchain Transactions in a Nutshell

Zero-Knowledge (ZK) Proofs on the Blockchain

ZK proofs have found practical importance in critical Blockchain applications:

Blockchain Transactions in a Nutshell

- Blockchain consists of a sequence of "valid" transactions (e.g., Send 5 Bitcoins from Alice to Bob)

Zero-Knowledge (ZK) Proofs on the Blockchain

ZK proofs have found practical importance in critical Blockchain applications:

Blockchain Transactions in a Nutshell

- Blockchain consists of a sequence of "valid" transactions (e.g., Send 5 Bitcoins from Alice to Bob)
- Miners need to be able to verify that transactions are valid

Zero-Knowledge (ZK) Proofs on the Blockchain

ZK proofs have found practical importance in critical Blockchain applications:

Blockchain Transactions in a Nutshell

- Blockchain consists of a sequence of “valid” transactions (e.g., Send 5 Bitcoins from Alice to Bob)
- Miners need to be able to verify that transactions are valid
- Bitcoin does this by making the transactions public

Zero-Knowledge (ZK) Proofs on the Blockchain

ZK proofs have found practical importance in critical Blockchain applications:

Blockchain Transactions in a Nutshell

- Blockchain consists of a sequence of "valid" transactions (e.g., Send 5 Bitcoins from Alice to Bob)
- Miners need to be able to verify that transactions are valid
- Bitcoin does this by making the transactions public
- But, what if we want to keep the details of our transaction secret?

Zero-Knowledge (ZK) Proofs on the Blockchain

ZK proofs have found practical importance in critical Blockchain applications:

Blockchain Transactions in a Nutshell

- Blockchain consists of a sequence of "valid" transactions (e.g., Send 5 Bitcoins from Alice to Bob)
- Miners need to be able to verify that transactions are valid
- Bitcoin does this by making the transactions public
- But, what if we want to keep the details of our transaction secret?

ZK Proofs enable privacy-preserving transactions on a public Blockchain!