Probabilistic Proof Systems

CS 6230: Topics in Information Security

Lecture 1: Introduction

Prashant Nalini Vasudevan

Lecture Plan

1. Course Overview
2. Proof Systems for Languages
   - NP
3. Randomness in proofs
   - MA
   - Benefits of randomness
4. Interactive proofs
   - Definition
   - Relation to NP and MA
   - Relation to PSPACE

Proofs

- Fundamental part of mathematics
- Establish the truth of statements

Important properties:

- **Completeness:** All true statements can be proven
- **Soundness:** No false statements can be proven
- **Efficiency:** The validity of a proof can be determined *efficiently*

Classical Proofs

Sequence of claims leading to theorems from axioms

Theorem: $(a+b)^2 = a^2 + 2ab + b^2$

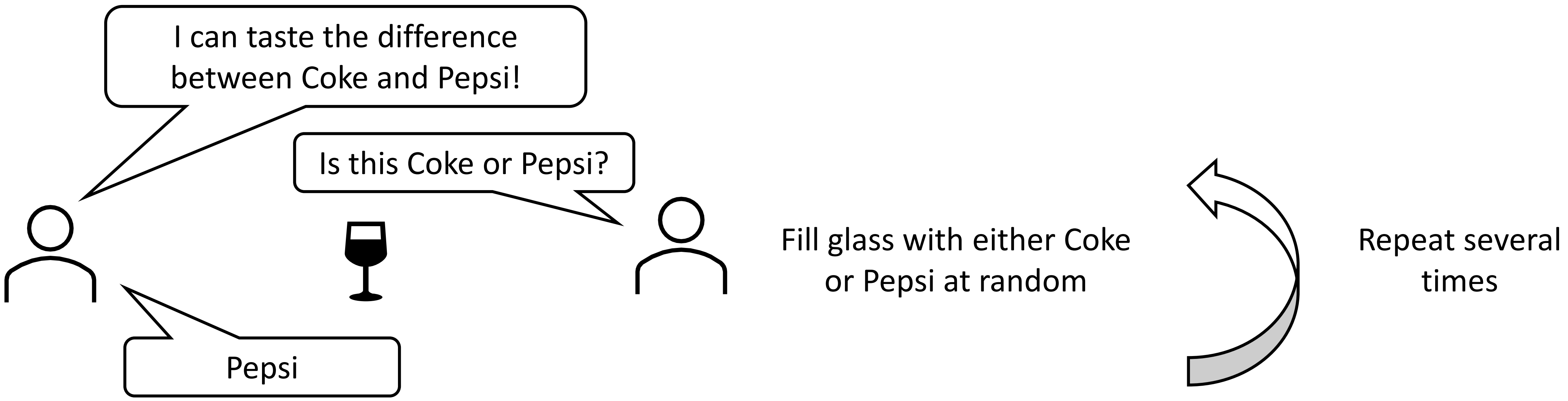
Proof:
$$\begin{aligned}(a+b)^2 &= (a+b)\cdot(a+b) \\ &= a\cdot a + a\cdot b + b\cdot a + b\cdot b \\ &= a^2 + 2ab + b\text{^}2\end{aligned}$$

Verification: Verify each claim

Non-Classical Proof Systems

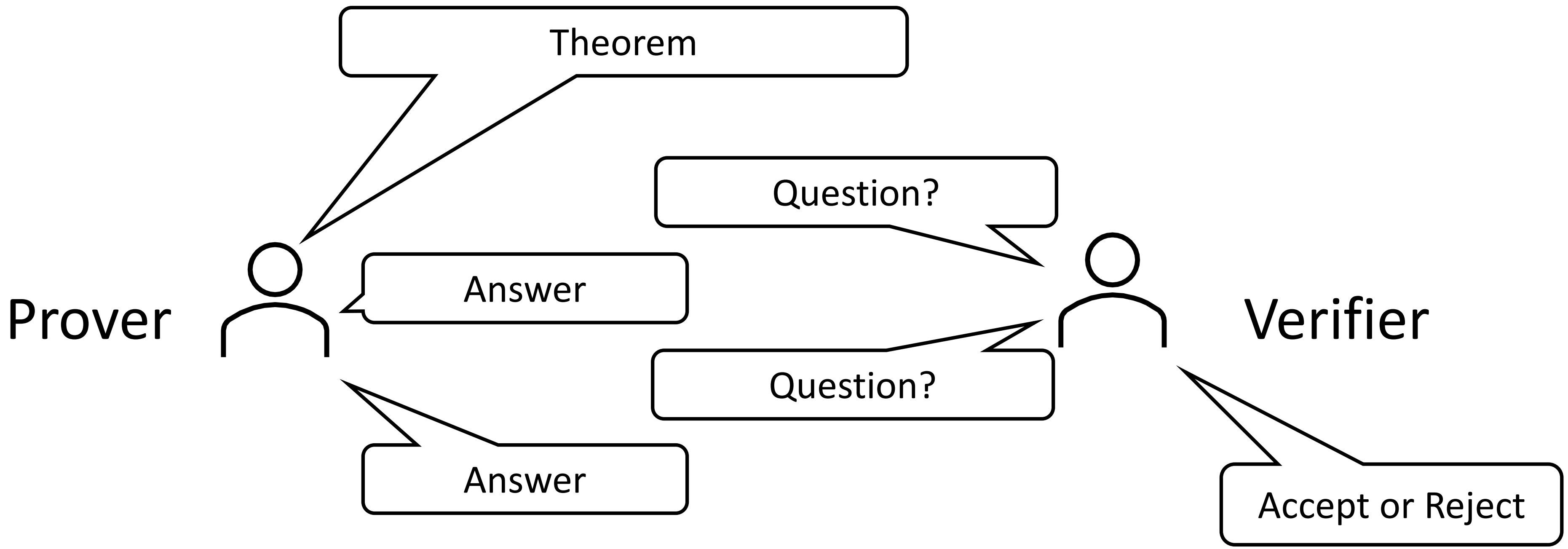
- Studied by computer scientists since the 80’s
- New notions of what it means to “prove” something
- Vastly more “powerful” than classical proofs
- We will study some of these along with:
  - their applications,
  - connections to complexity theory and cryptography, and,
  - relevant tools from cryptography and TCS

Interactive Proofs



- If they can taste the difference, they will answer correctly - *completeness*
- If they cannot they will make a mistake (with high probability) - *soundness*
- You know whether the glass has Coke or Pepsi, so you can check *efficiently*

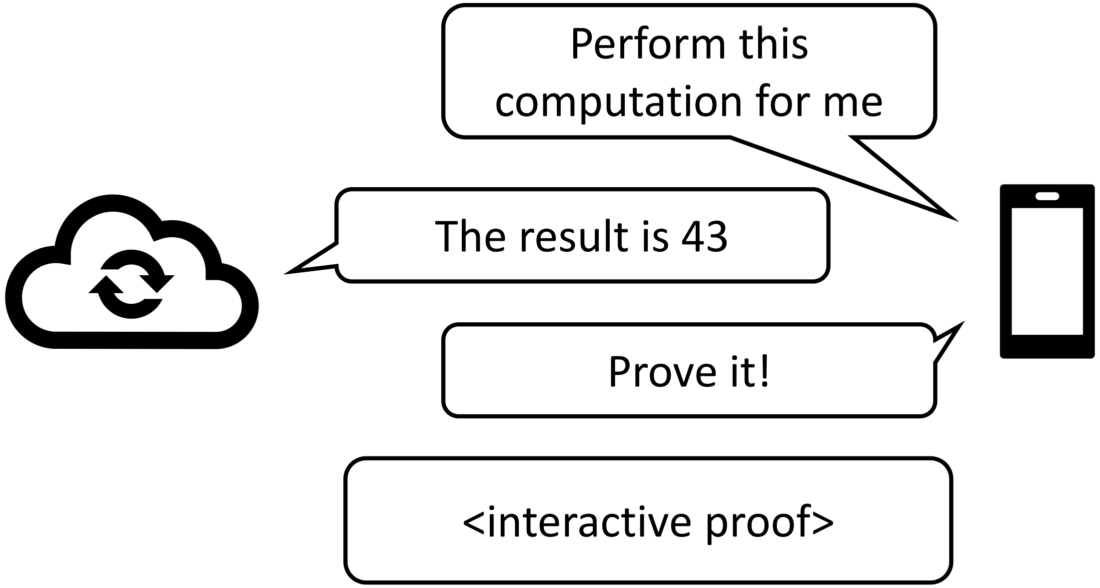
Interactive Proofs



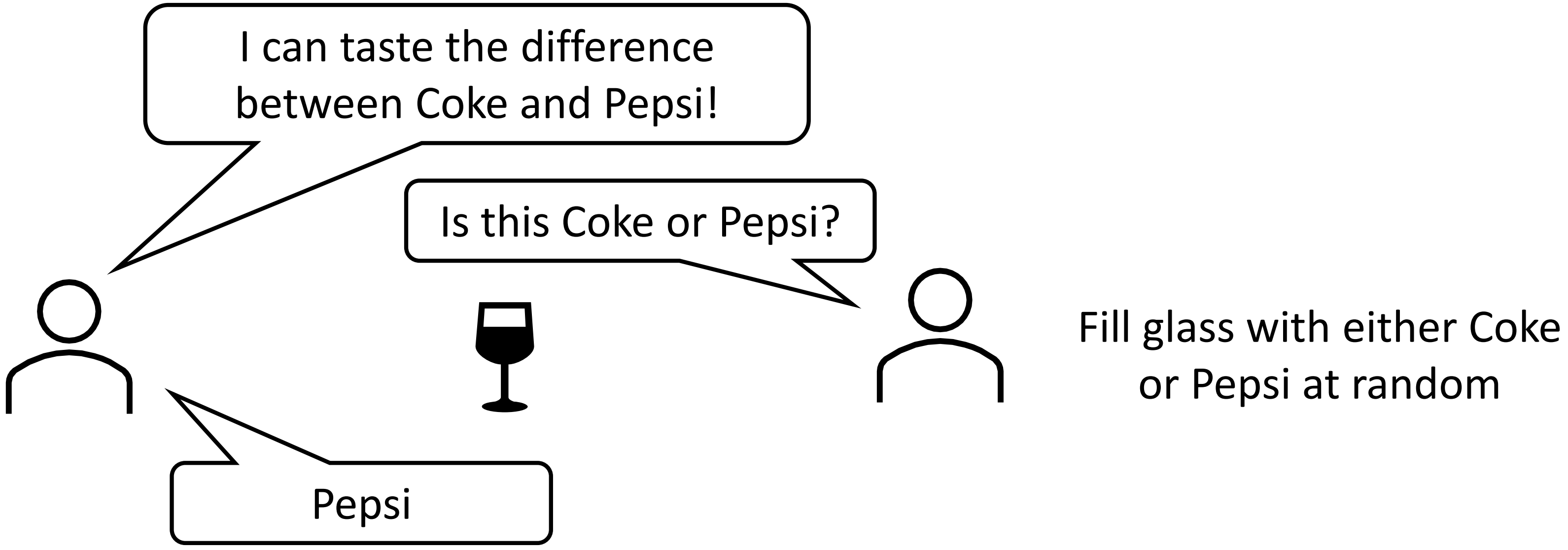
- **Completeness:** If theorem is true, Verifier should Accept with high probability
- **Soundness:** If theorem is false, Verifier should Reject with high probability, even if Prover cheats
- **Efficiency:** Verifier should be computationally efficient

Interactive Proofs

- Fundamentally new notion of what it means to prove something
- Connected to various other concepts in complexity theory
- Potential real-world applications, e.g. delegation of computation



Zero-Knowledge Proofs

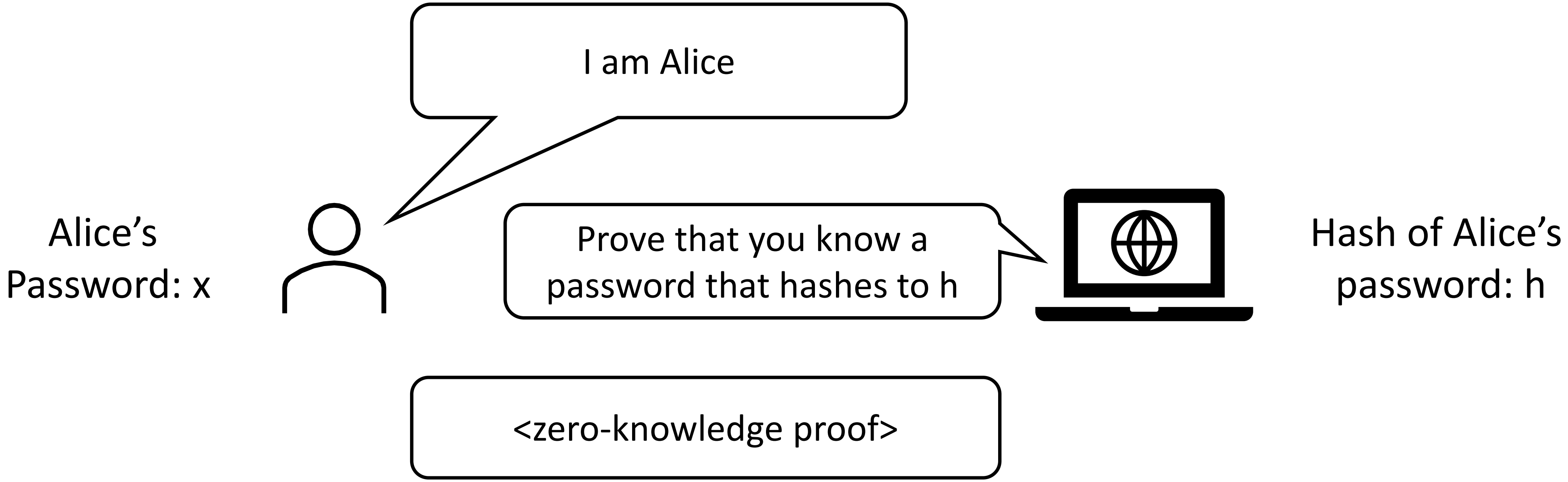


If you cannot distinguish between Coke and Pepsi before the proof, you still cannot after the proof

Zero Knowledge: The Verifier learns nothing during the proof except the truth of the statement being proven

Zero-Knowledge Proofs

- Connected to various concepts in cryptography
- Useful when data needs to be protected while allowing certain functionalities, e.g. authentication

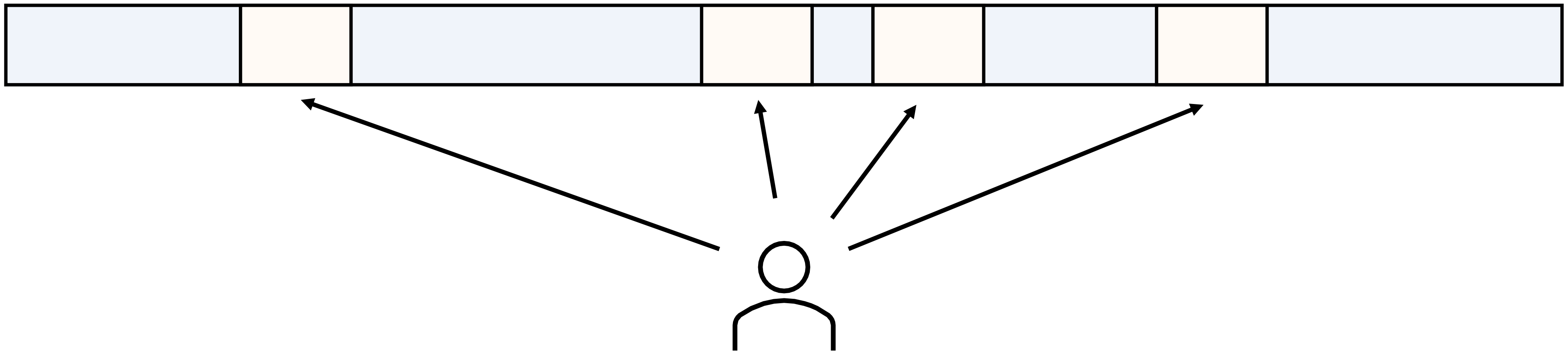


Interactive Arguments

- Like Interactive Proofs, but:
  - The honest Prover should also be computationally efficient
  - Computational Soundness: False statements should not be provable by *efficient* cheating Provers
- Usually built on the security of cryptographic primitives
  - If Prover is able to prove a false statement, it can be used to break the cryptographic primitive
- Can be made very efficient, so most useful in practical applications

Probabilistically Checkable Proofs

Proofs that can be checked without reading them entirely



- Deep connections to complexity theory and approximation algorithms
- Useful in constructing arguments and other cryptographic protocols

What We Will Cover

- Definitions and properties of each of the above proof systems
- What kinds of theorems they can prove
- Their connections to complexity theory and cryptography
- Tools useful in their construction
  - Low-degree extensions
- Related cryptographic concepts
  - One-way functions
  - Collision resistance
- Possibly other models like MIPs, IOPs, etc.

What You Need to Know Already

- Basic complexity theory
  - Reductions between problems
  - NP-hardness and its significance
  - Classes like PSPACE, AC^0
- Basic probability theory
  - Random variables
- Basic algebra
  - Linear algebra
  - Finite fields
  - Groups
- Basic graph theory

Course Information

- Instructor: Me
- Email: prashant@comp.nus.edu.sg
- Website: https://www.comp.nus.edu.sg/~prashant/teaching/CS6230/
- Time: Tue 10am – noon SGT
- Location: Online for now
- Office Hours: Wed 10am – noon SGT (book slot on LumiNUS)
- References: See lecture notes and website

Grading

Problem Sets (60%, distributed over 3-4 sets)

- Will be posted on course website and LumiNUS
- Collaboration encouraged, but your submission must be written on your own
- Submit on LumiNUS, in the relevant folder in the Files section
- No late submissions accepted without my explicit permission
- First problem set will be uploaded tomorrow, is ***due next Monday*** (Aug 16)

Final Project (40%)

- Read and write a survey/report on a few related papers
- Work in groups of 1 or 2
- Details will be announced in a few weeks

Proof Systems for Languages

Definition: A *language* is a set of strings from $\{0,1\}^*$

aka inputs or instances

often implicit

Example: SAT is the set of all Boolean formulas that are satisfiable

- $(x_1 \wedge x_2) \in$ SAT
- $(x_1 \wedge \overline{x}_1) \notin$ SAT

We care about proofs of statements of the form $x \in L$ for some language L and string x

Proof Systems for Languages

Classical proofs correspond to languages in the class NP

Definition: The class NP consists of languages L for which there exists a *deterministic polynomial-time* verification algorithm V and a polynomial p such that:

- **Completeness:** For any instance $x \in L$, there exists a "witness" $y \in \{0,1\}^{p(|x|)}$ such that $V(x, y)$ accepts

  theorem proof

- **Soundness:** For any $x \notin L$, for all $y \in \{0,1\}^{p(|x|)}$, $V(x, y)$ rejects

Randomness in Proofs

Most verification procedures will be randomised

Definition: The class MA consists of languages L for which there exists a *probabilistic polynomial-time* verification algorithm V and a polynomial p such that:

- **Completeness:** For any instance $x \in L$, there exists a “witness” $y \in \{0,1\}^{p(|x|)}$ such that:

$$\Pr[V(x, y) \text{ accepts}] \geq 2/3$$

- **Soundness:** For any $x \notin L$, for all $y \in \{0,1\}^{p(|x|)}$,

$$\Pr[V(x, y) \text{ accepts}] \leq 1/3$$

How randomness can help: verifying matrix multiplication

Task: Given matrices $A, B, C \in \mathbb{F}^{n \times n}$, where $\mathbb{F}$ is a finite field, verify that $C = A \cdot B$

Naïve method: Compute $A \cdot B$, and check it is equal to C

Perfectly complete and sound, but takes $\Omega(n^\omega)$ field operations (best known: $\omega \leq 2.378$)

Can we do faster?

How randomness can help: verifying matrix multiplication

Freivald’s protocol:
1. Sample random vector $v \leftarrow \mathbb{F}^n$
2. Compute $u \leftarrow Bv$ and $w \leftarrow Au$
3. Check whether $w = Cv$

Completeness: If $C = A \cdot B$, then $Cv = (AB)v = A(Bv) = Au = w$

Efficiency: Three matrix-vector multiplications, each $O(n^2)$ field operations

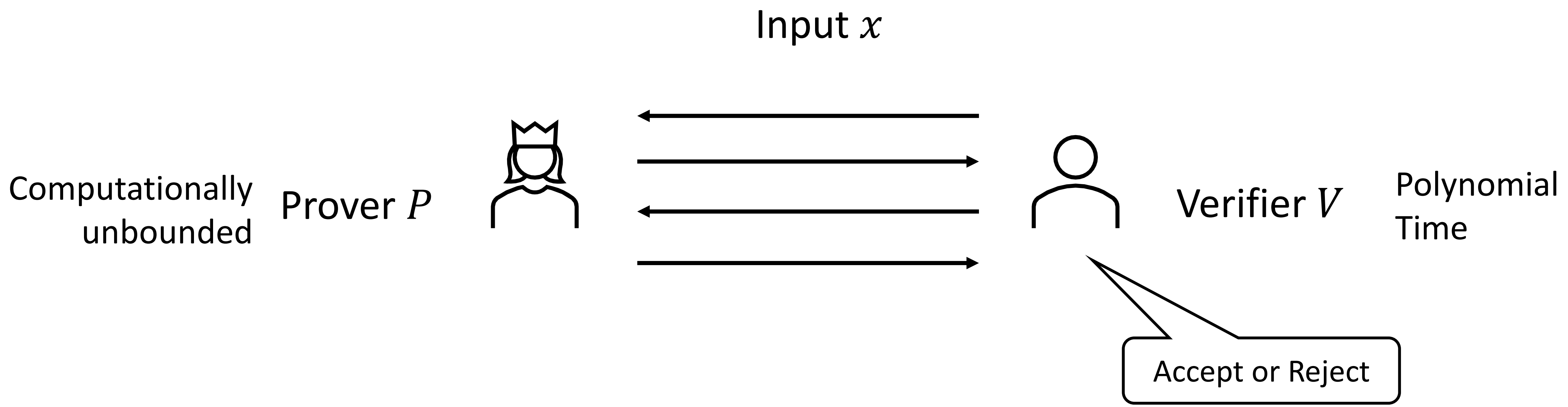
How randomness can help: verifying matrix multiplication

Soundness:

- Let $A \cdot B = D$, and suppose $C \neq D$
- Then, $\exists i \in [n]$ such that the i^{th} rows of C and D – written as C_i and D_i – are different.
- The i^{th} coordinate of $(C - D)v$ is $< (C_i - D_i), v >$
- As $(C_i - D_i)$ is non-zero, this inner product is uniformly distributed over $\mathbb{F}$
- $Cv = Dv$ only if $(Cv)_i = (Dv)_i$, that is, $< (C_i - D_i), v > = 0$
- Thus, $Cv = Dv$ with probability at most $1/|\mathbb{F}|$

Thus, if $C \neq A \cdot B$, the verification passes with probability at most $1/|\mathbb{F}|$.

Interactive Proofs



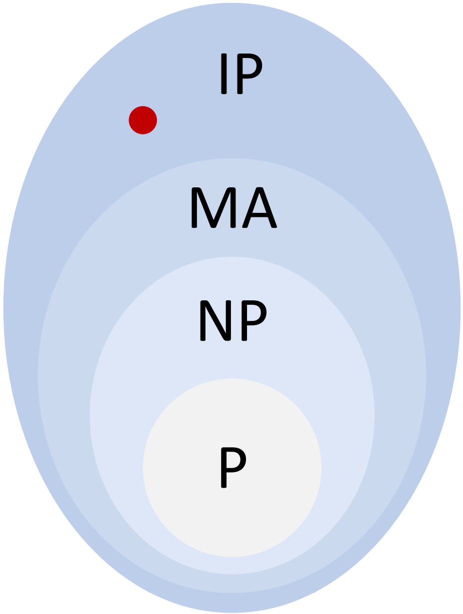
Definition: (P, V) is an Interactive Proof for language L if:

- **Completeness:** For any $x \in L$**,** $\Pr[V \text{ accepts}] \geq 2/3$
- **Soundness:** For any $x \notin L$**,** $\Pr[V \text{ accepts}] \leq 1/3$, irrespective of what P does

Which languages have IPs?

- All languages in NP do – the prover just sends the NP witness
- Similarly, all languages in MA do
- How about others?

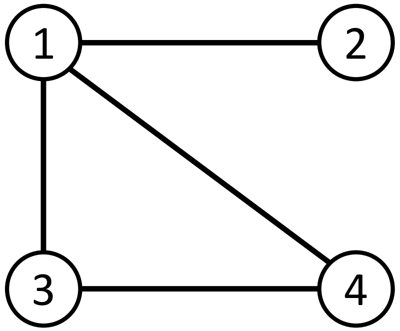
IP – set of all languages that
have an interactive proof



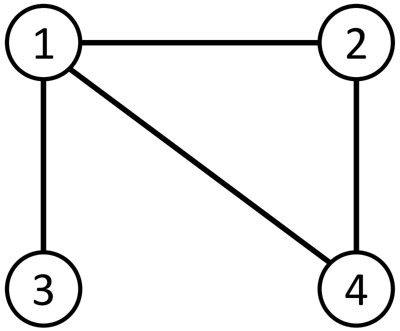
Graph Non-Isomorphism

Definition: Two graphs G_0 and G_1 are isomorphic if there is a relabelling of the vertices of G_0 that makes it the same as G_1

G_0



G_1

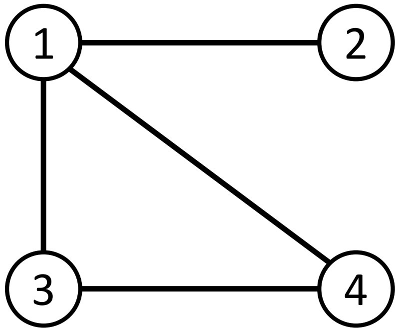


G_0 and G_1 isomorphic by relabelling: $1 \to 1, 2 \to 3, 3 \to 2, 4 \to 4$

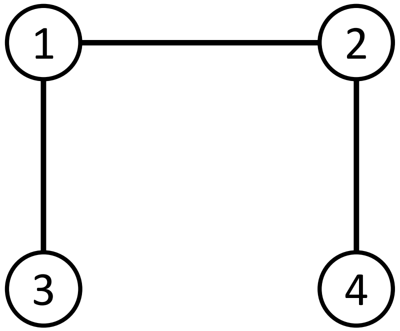
Graph Non-Isomorphism

Definition: Two graphs G_0 and G_1 are isomorphic if there is a relabelling of the vertices of G_0 that makes it the same as G_1

G_0



G_1



G_0 and G_1 not isomorphic

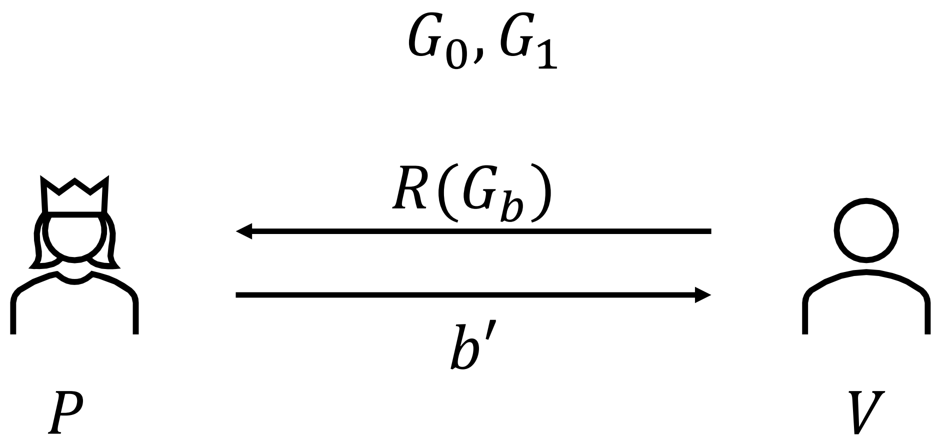
Graph Non-Isomorphism

Definition: Two graphs G_0 and G_1 are isomorphic if there is a relabelling of the vertices of G_0 that makes it the same as G_1

Definition: The language GNI consists of pairs of graphs (G_0, G_1) such that G_0 and G_1 have the same number of vertices, and are *not* isomorphic.

GNI ∈ coNP, not known to be in NP or MA

IP for Graph Non-Isomorphism



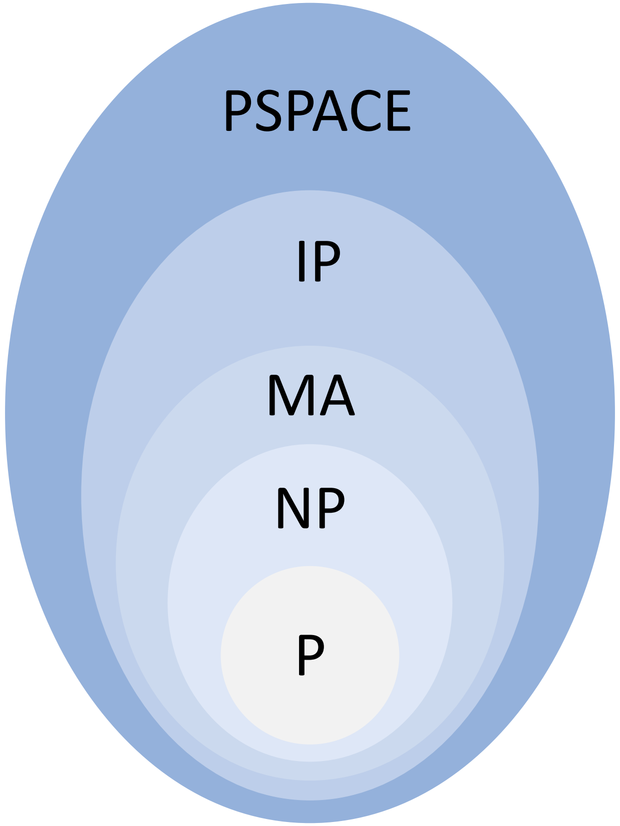
Pick uniformly random relabelling R, random bit b

Accept iff $b' = b$

Completeness: If (G_0, G_1) are non-isomorphic, then $R(G_b)$ is isomorphic to G_0 or G_1, but not both. Prover can thus learn b given $R(G_b)$. So V always accepts.

Soundness: If (G_0, G_1) are isomorphic, then $R(G_b)$ could have been produced either from G_0 or G_1, so prover learns nothing about b. So V accepts with probability at most ½.

How big is IP?



IP and PSPACE

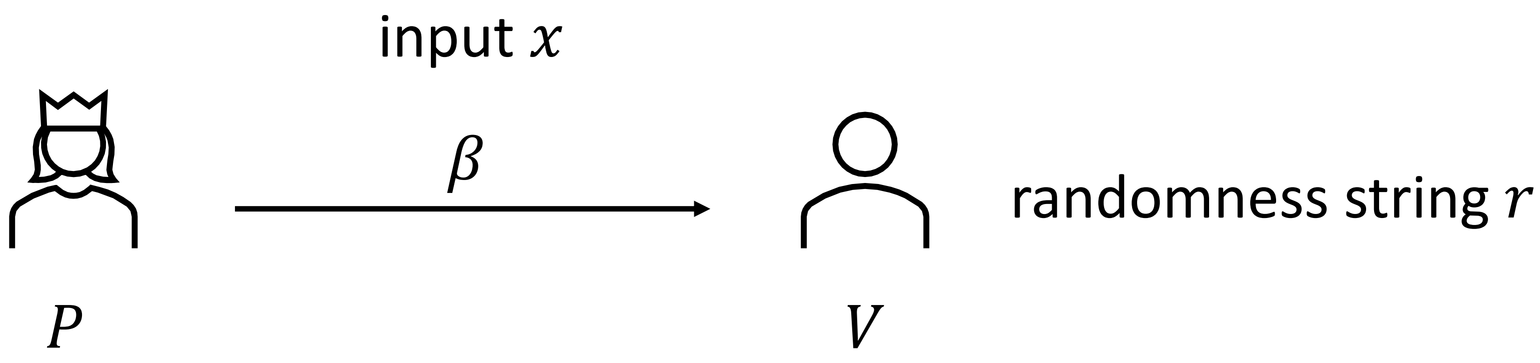
Definition: PSPACE is the set of languages L such that there is a polynomial-*space* algorithm that, given an input x, determines whether $x \in L$.

Theorem: IP $\subseteq$ PSPACE

Need to show: If a language L has an interactive proof, then there is a polynomial-space algorithm that, given input x, determines whether $x \in L$

IP ⊆ PSPACE

Warm-up: If a language L has a *1-message* interactive proof (MA), then there is a polynomial-space algorithm that, given input x, determines whether $x \in L$



The poly-space algorithm proceeds in two steps:

1. Given input x, find the β that maximises V's acceptance probability
2. If this probability is more than 2/3, say $x \in L$, otherwise $x \notin L$

IP $\subseteq$ PSPACE

Warm-up: If a language L has a *1-message* interactive proof (MA), then there is a polynomial-space algorithm that, given input x, determines whether $x \in L$

The poly-space algorithm proceeds in two steps:

1. Given input x, find the β that maximises V's acceptance probability
2. If this probability is more than 2/3, say $x \in L$, otherwise $x \notin L$

Step 1:

1. For all string β of appropriate length:
   - Iterate over all possible random strings r, and count the number on which $V(x, \beta; r)$ accepts
2. Output the β that has the greatest count above

In Conclusion

- Check out the course website
- Watch out for the problem set tomorrow
- See you next week