CS151 Complexity Theory

Lecture 14
May 18, 2023

1

Interactive Proofs

- **interactive proof system** for L is an interactive protocol (P, V)
  - completeness: $x \in L \Rightarrow$
    $\Pr[V \text{ accepts in } (P, V)(x)] \geq 2/3$
  - soundness: $x \notin L \Rightarrow \forall P^*$
    $\Pr[V \text{ accepts in } (P^*, V)(x)] \leq 1/3$
  - efficiency: V is p.p.t. machine
- **IP** = {L : L has an interactive proof system}

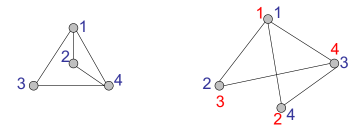
May 18, 2023 CS151 Lecture 14

2

Graph Isomorphism

- graphs $G_0 = (V, E_0)$ and $G_1 = (V, E_1)$ are isomorphic ($G_0 \simeq G_1$) if exists a permutation $\pi: V \rightarrow V$ for which

$$(x, y) \in E_0 \Leftrightarrow (\pi(x), \pi(y)) \in E_1$$



May 18, 2023 CS151 Lecture 14

3

Graph Isomorphism

- GI = $\{(G_0, G_1) : G_0 \simeq G_1\}$
  - in **NP**
  - not known to be in **P**, or **NP**-complete
- GNI = complement of GI
  - not known to be in **NP**

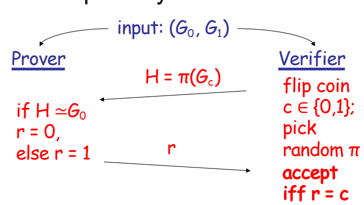
Theorem (GMW): GNI $\in$ **IP**
- indication **IP** may be more powerful than **NP**

May 18, 2023 CS151 Lecture 14

4

GNI in **IP**

- interactive proof system for GNI:



May 18, 2023 CS151 Lecture 14

5

GNI in **IP**

- completeness:
  - if G_0 not isomorphic to G_1 then H is isomorphic to exactly one of (G_0, G_1)
  - prover will choose correct r
- soundness:
  - if $G_0 \simeq G_1$ then prover sees same distribution on H for c = 0, c = 1
  - no information on c $\Rightarrow$ *any* prover P* can succeed with probability at most 1/2

May 18, 2023 CS151 Lecture 14

6

The power of **IP**

- We showed GNI $\in$ **IP**
- GNI $\in$ **IP** suggests **IP** more powerful than **NP**, since we don't know how to show GNI in **NP**
- GNI in **coNP**

Theorem (LFKN): **coNP** $\subseteq$ **IP**

7

The power of **IP**

- Proof idea: input: $\varphi(x_1, x_2, \ldots, x_n)$
  - prover: "I claim φ has k satisfying assignments"
  - true iff
    - $\varphi(0, x_2, \ldots, x_n)$ has k_0 satisfying assignments
    - $\varphi(1, x_2, \ldots, x_n)$ has k_1 satisfying assignments
    - $k = k_0 + k_1$
  - prover sends k_0, k_1
  - verifier sends random $c \in \{0,1\}$
  - prover recursively proves "$\varphi' = \varphi(c, x_2, \ldots, x_n)$ has k_c satisfying assignments"
  - at end, verifier can check for itself.

8

The power of **IP**

- Analysis of proof idea:
  - Completeness: $\varphi(x_1, x_2, \ldots, x_n)$ has k satisfying assignments $\Rightarrow$ accept with prob. 1
  - Soundness: $\varphi(x_1, x_2, \ldots, x_n)$ does not have k satisfying assigns. $\Rightarrow$ accept prob. $\leq 1 - 2^{-n}$
  - Why? It is possible that k is only off by one; verifier only catches prover if coin flips c are successive bits of this assignment

9

The power of **IP**

- Solution to problem (ideas):
  - replace $\{0,1\}^n$ with $(F_q)^n$
  - verifier substitutes random field element at each step
  - *vast majority* of field elements catch cheating prover (rather than just 1)

Theorem: L = { (φ, k): CNF φ has exactly k satisfying assignments} is in **IP**

10

The power of **IP**

- First step: **arithmetization**
  - transform $\varphi(x_1, \ldots x_n)$ into polynomial $p_\varphi(x_1, x_2, \ldots x_n)$ of degree d over a field F_q; q prime > 2^n
  - recursively:
    - $x_i \rightarrow x_i$ $\quad\neg\varphi \rightarrow (1 - p_\varphi)$
    - $\varphi \wedge \varphi' \rightarrow (p_\varphi)(p_{\varphi'})$
    - $\varphi \vee \varphi' \rightarrow 1 - (1 - p_\varphi)(1 - p_{\varphi'})$
  - for all $x \in \{0,1\}^n$ we have $p_\varphi(x) = \varphi(x)$
  - degree $d \leq |\varphi|$
  - can compute $p_\varphi(x)$ in poly time from φ and x

11

The power of **IP**

- Prover wishes to prove:

  $$k = \Sigma_{x_1 = 0, 1}\Sigma_{x_2 = 0,1} \cdots \Sigma_{x_n = 0, 1}p_\varphi(x_1, x_2, \ldots, x_n)$$
- Define: $k_z = \Sigma_{x_2 = 0,1} \cdots \Sigma_{x_n = 0, 1}p_\varphi(z, x_2, \ldots, x_n)$
- prover sends: k_z for all $z \in F_q$
- verifier:
  - checks that $k_0 + k_1 = k$
  - sends random $z \in F_q$
- continue with proof that

  $$k_z = \Sigma_{x_2 = 0,1} \cdots \Sigma_{x_n = 0, 1}p_\varphi(z, x_2, \ldots, x_n)$$
- at end: verifier checks for itself

12

The power of **IP**

- Prover wishes to prove:
  $k = \Sigma_{x_1 = 0,1} \Sigma_{x_2 = 0,1} \cdots \Sigma_{x_n = 0,1} p_\varphi(x_1, x_2, \ldots, x_n)$
- Define: $k_z = \Sigma_{x_2 = 0,1} \cdots \Sigma_{x_n = 0,1} p_\varphi(z, x_2, \ldots, x_n)$
- a problem: can't send k_z for all $z \in F_q$
- solution: send the polynomial !
  – recall degree $d \le |\varphi|$

May 18, 2023 CS151 Lecture 14

13

The actual protocol

input: (φ, k)

Prover → Verifier

Prover: $p_1(x) = \Sigma_{x_2,\ldots,x_n \in \{0,1\}} p_\varphi(\mathbf{x}, x_2, \ldots, x_n)$

$p_1(x)$ →

Verifier: $p_1(0)+p_1(1)=k$? pick random z_1 in F_q

← z_1

Prover: $p_2(x) = \Sigma_{x_3,\ldots,x_n \in \{0,1\}} p_\varphi(z_1, \mathbf{x}, x_3, \ldots, x_n)$

$p_2(x)$ →

Verifier: $p_2(0)+p_2(1)=p_1(z_1)$? pick random z_2 in F_q

← z_2

Prover: $p_3(x) = \Sigma_{x_4,\ldots,x_n \in \{0,1\}} p_\varphi(z_1, z_2, \mathbf{x}, x_4 \ldots, x_n)$

$p_3(x)$ →

Verifier: $p_3(0)+p_3(1)=p_2(z_2)$? pick random z_3 in F_q

.
.

$p_n(x)$ →

Verifier: $p_n(0)+p_n(1)=p_{n-1}(z_{n-1})$? pick random z_n in F_q

$p_n(z_n) = p_\varphi(z_1, z_2, \ldots, z_n)$?

May 18, 2023 CS151 Lecture 14

14

Analysis of protocol

- Completeness:
  – if $(\varphi, k) \in L$ then honest prover on previous slide will always cause verifier to accept

May 18, 2023 CS151 Lecture 14

15

Analysis of protocol

- Soundness:
  – let $p_i(x)$ be the correct polynomials
  – let $p_i^*(x)$ be the polynomials sent by (cheating) prover
  – $(\varphi, k) \notin L \Rightarrow p_1(0) + p_1(1) \ne k$
  – either $p_1^*(0) + p_1^*(1) \ne k$ (and V rejects)
  – or $p_1^* \ne p_1 \Rightarrow \Pr_{z_1}[p_1^*(z_1) = p_1(z_1)] \le d/q \le |\varphi|/2^n$
  – assume $(p_{i+1}(0)+p_{i+1}(1)=)\ p_i(z_i) \ne p_i^*(z_i)$
  – either $p_{i+1}^*(0) + p_{i+1}^*(1) \ne p_i^*(z_i)$ (and V rejects)
  – or $p_{i+1}^* \ne p_{i+1} \Rightarrow \Pr_{z_{i+1}}[p_{i+1}^*(z_{i+1}) = p_{i+1}(z_{i+1})] \le |\varphi|/2^n$

May 18, 2023 CS151 Lecture 14

16

Analysis of protocol

- Soundness (continued):
  – if verifier does not reject, there must be some i for which:

    $p_i^* \ne p_i$ and yet $p_i^*(z_i) = p_i(z_i)$
  – for each i, probability is $\le |\varphi|/2^n$
  – union bound: probability that there exists an i for which the bad event occurs is

    $\le n|\varphi|/2^n \le \mathrm{poly}(n)/2^n << 1/3$

May 18, 2023 CS151 Lecture 14

17

Analysis of protocol

- Conclude: L = { (φ, k): CNF φ has exactly k satisfying assignments} is in **IP**
- L is **coNP**-hard, so **coNP** ⊆ **IP**
- Question remains:
  – **NP, coNP** ⊆ **IP.** Potentially larger. How much larger?

May 18, 2023 CS151 Lecture 14

18

IP = PSPACE

Theorem: (Shamir) **IP = PSPACE**
- Note: **IP** $\subseteq$ **PSPACE**
  - enumerate all possible interactions, explicitly calculate acceptance probability
- interaction extremely powerful !
- An implication: you can interact with master player of Generalized Geography and determine if she can win from the current configuration even if you do not have the power to compute optimal moves!

May 18, 2023 CS151 Lecture 14

19

IP = PSPACE

- need to prove **PSPACE** $\subseteq$ **IP**
  - use same type of protocol as for **coNP**
  - some modifications needed

May 18, 2023 CS151 Lecture 14

20

IP = PSPACE

- protocol for QSAT
  - arithmetization step produces **arithmetic expression** p_φ:
    - $(\exists x_i)\ \varphi \rightarrow \Sigma_{x_i = 0, 1}\ p_\varphi$
    - $(\forall x_i)\ \varphi \rightarrow \prod_{x_i = 0, 1}\ p_\varphi$
  - start with QSAT formula in special form ("simple")
    - no occurrence of x_i separated by more than one "∃" from point of quantification

May 18, 2023 CS151 Lecture 14

21

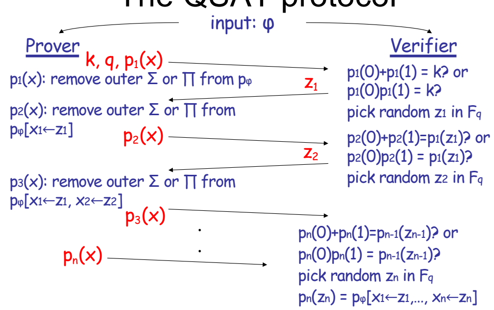
IP = PSPACE

- quantified Boolean expression φ is true if and only if $p_\varphi > 0$
- Problem: ∏'s may cause $p_\varphi > 2^{2^{|\varphi|}}$
- Solution: evaluate mod $2^n \leq q \leq 2^{3n}$
- prover sends "good" q in first round
  - "good" q is one for which p_φ mod q > 0
- Claim: good q exists
  - # primes in range is at least 2^n

May 18, 2023 CS151 Lecture 14

22

The QSAT protocol



May 18, 2023 CS151 Lecture 14

23

Analysis of the QSAT protocol

- Completeness:
  - if φ ∈ QSAT then honest prover on previous slide will always cause verifier to accept

May 18, 2023 CS151 Lecture 14

24

Analysis of the QSAT protocol

- Soundness:
  - let $p_i(x)$ be the correct polynomials
  - let $p_i^*(x)$ be the polynomials sent by (cheating) prover
  - $\varphi \notin$ QSAT $\Rightarrow 0 = p_1(0) +/\times p_1(1) \neq k$
  - either $p_1^*(0) +/\times p_1^*(1) \neq k$ (and V rejects)
  - or $p_1^* \neq p_1 \Rightarrow \Pr_{z_1}[p_1^*(z_1) = p_1(z_1)] \leq 2|\varphi|/2^n$
  - assume $(p_{i+1}(0) +/\times p_{i+1}(1) =)\ p_i(z_i) \neq p_i^*(z_i)$
  - either $p_{i+1}^*(0) +/\times p_{i+1}^*(1) \neq p_i^*(z_i)$ (and V rejects)
  - or $p_{i+1}^* \neq p_{i+1} \Rightarrow \Pr_{z_{i+1}}[p_{i+1}^*(z_{i+1}) = p_{i+1}(z_{i+1})] \leq 2|\varphi|/2^n$

φ is "simple"

25

Analysis of protocol

- Soundness (continued):
  - if verifier does not reject, there must be some i for which:

    $p_i^* \neq p_i$ and yet $p_i^*(z_i) = p_i(z_i)$
  - for each i, probability is $\leq 2|\varphi|/2^n$
  - union bound: probability that there exists an i for which the bad event occurs is

    $\leq 2n|\varphi|/2^n \leq \text{poly}(n)/2^n << 1/3$
- Conclude: QSAT is in **IP**

26

Example

- Papadimitriou – pp. 475-480

$\varphi = \forall x \exists y (x \vee y) \wedge \forall z ((x \wedge z) \vee (y \wedge \neg z)) \vee \exists w (z \vee (y \wedge \neg w))$

$p_\varphi = \prod_{x=0,1} \Sigma_{y=0,1}[(x + y) * \prod_{z=0,1}[(xz + y(1-z)) + \Sigma_{w=0,1}(z + y(1-w))]]$

(p_φ = 96 but V doesn't know that yet !)

27

Example

$p_\varphi = \prod_{x=0,1} \Sigma_{y=0,1}[(x + y) * \prod_{z=0,1}[(xz + y(1-z)) + \Sigma_{w=0,1}(z + y(1-w))]]$

Round 1: (prover claims $p_\varphi > 0$)

- prover sends q = 13; claims p_φ = 96 mod 13 = 5; sends k = 5
- prover removes outermost "∏"; sends

  $p_1(x) = 2x^2 + 8x + 6$
- verifier checks:

  $p_1(0)p_1(1) = (6)(16) = 96 \equiv 5 \pmod{13}$
- verifier picks randomly: $z_1 = 9$

28

Example

$\varphi = \forall x \exists y (x \vee y) \wedge \forall z ((x \wedge z) \vee (y \wedge \neg z)) \vee \exists w (z \vee (y \wedge \neg w))$

$p_\varphi = \prod_{x=0,1} \Sigma_{y=0,1}[(x + y) * \prod_{z=0,1}[(xz + y(1-z)) + \Sigma_{w=0,1}(z + y(1-w))]]$

$p_\varphi[x \leftarrow 9] = \Sigma_{y=0,1}[(9 + y) * \prod_{z=0,1}[(9z + y(1-z)) + \Sigma_{w=0,1}(z + y(1-w))]]$

29

Example

$p_1(9) = \Sigma_{y=0,1}[(9 + y) * \prod_{z=0,1}[(9z + y(1-z)) + \Sigma_{w=0,1}(z + y(1-w))]]$

Round 2: (prover claims this = 6)

- prover removes outermost "Σ"; sends

  $p_2(y) = 2y^3 + y^2 + 3y$
- verifier checks:

  $p_2(0) + p_2(1) = 0 + 6 = 6 \equiv 6 \pmod{13}$
- verifier picks randomly: $z_2 = 3$

30

Example

$\varphi = \forall x \exists y (x \lor y) \land \forall z ((x \land z) \lor (y \land \neg z)) \lor \exists w (z \lor (y \land \neg w))$

$p_\varphi = \prod_{x=0,1} \Sigma_{y=0,1}[(x + y) * \prod_{z=0,1}[(xz + y(1-z)) + \Sigma_{w=0,1}(z + y(1-w))]]$

$p_\varphi[x\leftarrow 9, y\leftarrow 3] = [(9 + 3) * \prod_{z=0,1}[(9z + 3(1-z)) + \Sigma_{w=0,1}(z + 3(1-w))]]$

May 18, 2023 CS151 Lecture 14

31

Example

$p_2(3) = [(9 + 3) * \prod_{z=0,1}[(9z + 3(1-z)) + \Sigma_{w=0,1}(z + 3(1-w))]]$

Round 3: (prover claims this = 7)
- everyone agrees expression = 12*(…)
- prover removes outermost "∏"; sends
  $p_3(z) = 8z + 6$
- verifier checks:
  $p_3(0) * p_3(1) = (6)(14) = 84; 12*84 \equiv 7 \pmod{13}$
- verifier picks randomly: $z_3 = 7$

May 18, 2023 CS151 Lecture 14

32

Example

$\varphi = \forall x \exists y (x \lor y) \land \forall z ((x \land z) \lor (y \land \neg z)) \lor \exists w (z \lor (y \land \neg w))$

$p_\varphi = \prod_{x=0,1} \Sigma_{y=0,1}[(x + y) * \prod_{z=0,1}[(xz + y(1-z)) + \Sigma_{w=0,1}(z + y(1-w))]]$

$p_\varphi[x\leftarrow 9, y\leftarrow 3, z\leftarrow 7] =$
$12 * [(9*7 + 3(1-7)) + \Sigma_{w=0,1}(7 + 3(1-w))]$

May 18, 2023 CS151 Lecture 14

33

Example

$12*p_3(7) = 12 * [(9*7 + 3(1-7)) + \Sigma_{w=0,1}(7 + 3(1-w))]$

Round 4: (prover claims = 12*10)
- everyone agrees expression = 12*[6+(…)]
- prover removes outermost "Σ"; sends
  $p_4(w) = 10w + 10$
- verifier checks:
  $p_4(0)+p_4(1) = 10 + 20 = 30; 12*[6+30] \equiv 12*10 \pmod{13}$
- verifier picks randomly: $z_4 = 2$
- Final check:
  $12*[(9*7+3(1-7))+(7+3(1-2))] = 12*[6+p_4(2)] = 12*[6+30]$

May 18, 2023 CS151 Lecture 14

34

Arthur-Merlin Games

- **IP** permits verifier to keep coin-flips private
  - necessary feature?
  - GNI protocol breaks without it
- Arthur-Merlin game: interactive protocol in which coin-flips are public
  - Arthur (verifier) may as well just send results of coin-flips and ask Merlin (prover) to perform any computation Arthur would have done

May 18, 2023 CS151 Lecture 14

35

Arthur-Merlin Games

- Clearly **Arthur-Merlin ⊆ IP**
  - "private coins are at least as powerful as public coins"
- Proof that **IP** = **PSPACE** actually shows
  PSPACE ⊆ Arthur-Merlin ⊆ IP = PSPACE
  - "public coins are at least as powerful as private coins" !

May 18, 2023 CS151 Lecture 14

36

Arthur-Merlin Games

- Delimiting # of rounds:
  - **AM[k]** = Arthur-Merlin game with k rounds, Arthur (verifier) goes first
  - **MA[k]** = Arthur-Merlin game with k rounds, Merlin (prover) goes first

Theorem: **AM[k]** (**MA[k]**) equals **AM[k]** (**MA[k]**) with perfect completeness.

- i.e., $x \in L$ implies accept with probability 1
- proof on problem set

Arthur-Merlin Games

Theorem: for all constant $k \geq 2$

AM[k] = AM[2].

- Proof:
  - we show **MA[2]** ⊆ **AM[2]**
  - implies can move all of Arthur's messages to beginning of interaction:

AMAMAM…AM = AAMMAM…AM

… = AAA…AMMM…M

Arthur-Merlin Games

- Proof (continued):
  - given $L \in$ **MA[2]**

$x \in L \Rightarrow \exists m \; \Pr_r[(x, m, r) \in R] = 1$

$\Rightarrow \Pr_r[\exists m \; (x, m, r) \in R] = 1$

order reversed

$x \notin L \Rightarrow \forall m \; \Pr_r[(x, m, r) \in R] \leq \varepsilon$

$\Rightarrow \Pr_r[\; \forall m \; (x, m, r) \in R] \leq 2^{|m|}\varepsilon$

  - by repeating t times with independent random strings r, can make error $\varepsilon < 2^{-t}$
  - set $t = m+1$ to get $2^{|m|}\varepsilon < ½$.

MA and **AM**

- Two important classes:
  - **MA = MA[2]**
  - **AM = AM[2]**
- definitions without reference to interaction:
  - $L \in$ **MA** iff ∃ poly-time language R

$x \in L \Rightarrow \exists \; m \; \Pr_r[(x, m, r) \in R] = 1$

$x \notin L \Rightarrow \forall \; m \; \Pr_r[(x, m, r) \in R] \leq ½$

  - $L \in$ **AM** iff ∃ poly-time language R

$x \in L \Rightarrow \Pr_r[\; \exists \; m \; (x, m, r) \in R] = 1$

$x \notin L \Rightarrow \Pr_r[\; \exists \; m \; (x, m, r) \in R] \leq ½$