CSC 6011 - Theory of Computation

Lecture 22: #SAT in IP, Arithmetization

Time: Mon & Wed, 4:00 - 5:20 PM

Location: Teaching A, 107

Instructor: Tao LIN (林涛)

TA: Yefan GAO (高叶繁)

Fall 2026

Final Exam: Dec 23 (Wed), 16:00 – 17:30

Location: TxB 205

Lecture 22 – #SAT in IP, Arithmetization

Last time:

- IP $\subseteq$ PSPACE
- The Sum-Check Protocol

Today:

- Review the Sum-Check Protocol
- $\#\mathrm{SAT} \in$ IP
- Arithmetization
- Main idea for proving PSPACE $\subseteq$ IP

Review: The Sum-Check Problem

Given an m-variable polynomial $g: \mathbb{F}_q^m \to \mathbb{F}_q$ over a finite field (arithmetic modulo q).

Known bound on the degree of each variable: $\deg_{x_i}(g) \leq d_i$ for $i = 1, \ldots, m$.

Prover provides H and wants to convince the Verifier that:

$$H = \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_m \in \{0,1\}} g(b_1, b_2, \ldots, b_m)$$

Key observation: H is equal to

$$\underbrace{\sum_{b_2 \in \{0,1\}} \cdots \sum_{b_m \in \{0,1\}} g(0, b_2, \ldots, b_m)}_{H_1(0)} + \underbrace{\sum_{b_2 \in \{0,1\}} \cdots \sum_{b_m \in \{0,1\}} g(1, b_2, \ldots, b_m)}_{H_1(1)}$$

where $H_1(X_1) = \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_m \in \{0,1\}} g(X_1, b_2, \ldots, b_m)$ is a univariate polynomial of X_1.

- If P can provide $H_1(X_1)$ to V and prove its correctness, then V can check $H = H_1(0) + H_1(1)$.

Review: The Sum-Check Protocol [Lund, Fortnow, Karloff, Nisan, 1990]

Start: P sends an integer S, claiming to be the sum H.

Round 1: P sends (the coefficients of) a univariate polynomial $S_1(X_1)$, claiming $S_1 = H_1$, where

$$H_1(X_1) = \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_m \in \{0,1\}} g(X_1, b_2, \ldots, b_m)$$

V checks $\deg(S_1) \leq d_1$ and $S_1(0) + S_1(1) = S$.

Then, V samples a random $r_1 \in \mathbb{F}_q$ and sends it to P.

Round $i > 1$ $(i \leq m)$: P sends a univariate polynomial $S_i(X_i)$, claiming $S_i = H_i$, where

$$H_i(X_i) = \sum_{b_{i+1} \in \{0,1\}} \cdots \sum_{b_m \in \{0,1\}} g(r_1, \ldots, r_{i-1}, X_i, b_{i+1}, \ldots, b_m)$$

V checks $\deg(S_i) \leq d_i$ and $S_i(0) + S_i(1) = S_{i-1}(r_{i-1})$.

Then, V samples a random $r_i \in \mathbb{F}_q$ and sends it to P.

Final check: V checks $S_m(r_m) = g(r_1, \ldots, r_m)$.

Reject if any check fails. Otherwise accept.

Completeness

Start: P sends an integer S, claiming to be the sum H.

Round 1: P sends (the coefficients of) a univariate polynomial $S_1(X_1)$, claiming $S_1 = H_1$, where

$$H_1(X_1) = \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_m \in \{0,1\}} g(X_1, b_2, \ldots, b_m)$$

V checks $\deg(S_1) \leq d_1$ and $S_1(0) + S_1(1) = S$.

Then, V samples a random $r_1 \in \mathbb{F}_q$ and sends it to P.

Round $i > 1$ $(i \leq m)$: P sends a univariate polynomial $S_i(X_i)$, claiming $S_i = H_i$, where

$$H_i(X_i) = \sum_{b_{i+1} \in \{0,1\}} \cdots \sum_{b_m \in \{0,1\}} g(r_1, \ldots, r_{i-1}, X_i, b_{i+1}, \ldots, b_m)$$

V checks $\deg(S_i) \leq d_i$ and $S_i(0) + S_i(1) = S_{i-1}(r_{i-1})$.

Then, V samples a random $r_i \in \mathbb{F}_q$ and sends it to P.

Final check: V checks $S_m(r_m) = g(r_1, \ldots, r_m)$.

Reject if any check fails. Otherwise accept.

Completeness:

When the prover P is honest, V accepts with probability 1.

Soundness

Claim: If P doesn't follow the protocol, then V accepts with probability at most $\frac{\sum_{i=1}^{m} d_i}{q} \leq \frac{md}{q}$.

Proof: Induction on the number of variables m:

- **Base case:** $m = 1$.
  - P sends a single message $S_1(X_1)$ claimed to equal $g(X_1)$.
  - V picks a random $r_1 \in \mathbb{F}_q$, checks that $S_1(r_1) = g(r_1)$.
  - If $S_1 \neq g$, then by Schwartz-Zippel, $\Pr_{r_1 \in \mathbb{F}_q}[\, S_1(r_1) = g(r_1) \,] \leq \frac{d_1}{q}$.

Soundness

Claim: If P doesn't follow the protocol, then V accepts with probability at most $\frac{\sum_{i=1}^{m} d_i}{q} \leq \frac{md}{q}$.

Proof: Induction on the number of variables m:

- **Inductive case:** $m > 1$.
  - P's first message $S_1(X_1)$ is claimed to equal $H_1(X_1) = \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_m \in \{0,1\}} g(X_1, b_2, \ldots, b_m)$
  - Then V picks a random $r_1 \in \mathbb{F}_q$ and sends r_1 to P. They recursively check $S_1(r_1) = H_1(r_1)$.
  - If $S_1 \neq H_1$, then by Schwartz-Zippel, $\Pr_{r_1 \in \mathbb{F}_q}[\, S_1(r_1) = H_1(r_1)\,] \leq \frac{d_1}{q}$.
  - If $S_1(r_1) \neq H_1(r_1)$, then P needs to prove a false claim in the recursion.
    - The recursion applies the Sum-Check protocol to $g(r_1, X_2, \ldots, X_m)$, which has $m - 1$ variables.
    - By induction, P can convince V in the recursion with probability at most $\frac{\sum_{i=2}^{m} d_i}{q}$.
  - **Summary:** If $S_1 \neq H_1$, then the probability that V accepts is at most:

$$\Pr_{r_1 \in \mathbb{F}_q}[S_1(r_1) = H_1(r_1)] + \Pr_{r_2,\ldots,r_m \in \mathbb{F}_q}[\,\text{V accepts} \mid S_1(r_1) \neq H_1(r_1)\,]$$

$$\leq \frac{d_1}{q} + \frac{\sum_{i=2}^{m} d_i}{q} \leq \frac{\sum_{i=1}^{m} d_i}{q}$$

Costs of the protocol

Total communication: $O(md)$ field elements:

- P sends m messages, each a univariate polynomial of degree at most d.
- V sends $m-1$ messages, each being a field element.

Each field element (in $\mathbb{F}_q$) takes $O(\log q)$ bits.

V**'s runtime:** $O(md)$ field operations + time to evaluate g.

P**'s runtime:** $O(md \cdot 2^m \cdot$ time to evaluate g at one point $)$.

Mini-Quiz 22.1

In the Sum-Check protocol, how many evaluations of g does V need to do?

a) 2^m, one at every Boolean input.
b) m, one in each round.
c) Just one, $g(r_1, \ldots, r_m)$.

The #SAT problem ("number SAT")

Input: a Boolean formula $\varphi(x_1, \dots, x_m)$.

Output: the number of satisfying assignments:

$$N(\varphi) = |\{b \in \{0,1\}^m : \varphi(b) = 1\}|$$

This is a **counting problem**.

Define a **decision problem (language)**:

$$\#\text{SAT} = \{\langle \varphi, K\rangle \mid K = N(\varphi)\}$$

"φ has exactly K satisfying assignments."

Observation: #SAT is coNP-hard.

Proof: Show $\text{UNSAT} \leq_m^P \#\text{SAT}$.

$$f(\langle\varphi\rangle) = \langle\varphi, 0\rangle$$

$$\varphi \in \text{UNSAT} \Leftrightarrow N(\varphi) = 0$$

If we can prove #SAT ∈ IP, then coNP ⊆ IP

The #SAT problem ("number SAT")

Input: a Boolean formula $\varphi(x_1, \dots, x_m)$.

Output: the number of satisfying assignments:

$$N(\varphi) = |\{b \in \{0,1\}^m : \varphi(b) = 1\}| = \sum_{b \in \{0,1\}^m} \varphi(b)$$

This is a **counting problem**.

This is a Sum-Check problem, but φ is not a polynomial.

Define a **decision problem (language)**:

$$\#\text{SAT} = \{ \langle \varphi, K \rangle \mid K = N(\varphi) \}$$

"φ has exactly K satisfying assignments."

Observation: $\varphi(b) = 1$ if b is a satisfying assignment; 0 otherwise.

Main idea: turn φ into a polynomial and apply the Sum-Check protocol.

"arithmetization"

Arithmetization

Goal: Turn a Boolean formula φ into a polynomial g_φ, preserving the values on $\{0,1\}^m$ input.

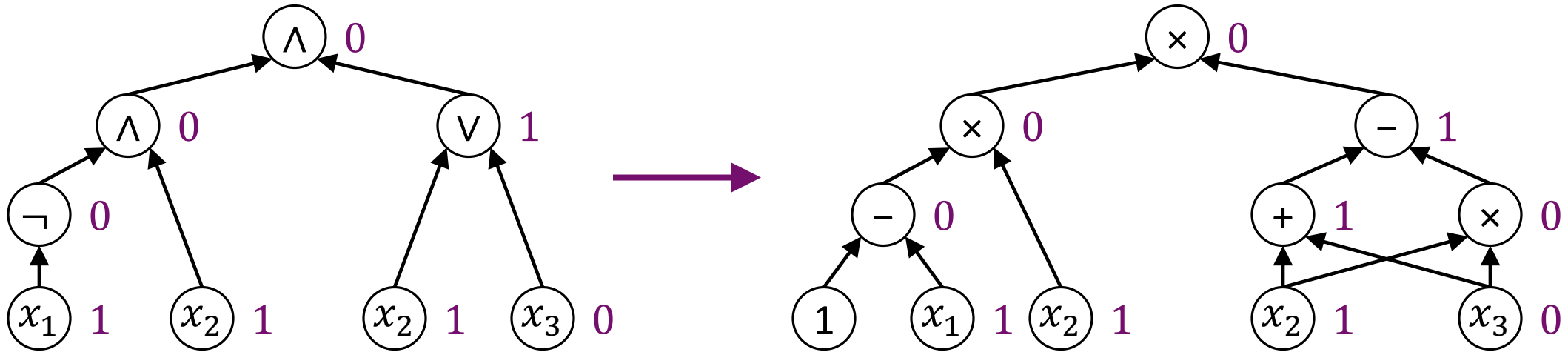
How: Replace each Boolean operation by an arithmetic operation:

$$\begin{aligned} \neg x &\mapsto 1-x \\ x \wedge y &\mapsto xy \\ x \vee y &\mapsto x+y-xy = 1-(1-x)(1-y) \end{aligned}$$

For $x, y \in \{0,1\}$, the value is unchanged.

Example:

$x_1 = 1$
$x_2 = 1$
$x_3 = 0$



$$\varphi = (\neg x_1 \wedge x_2) \wedge (x_2 \vee x_3)$$

$$g_\varphi = (1-x_1)x_2(x_2 + x_3 - x_2x_3)$$

Arithmetization

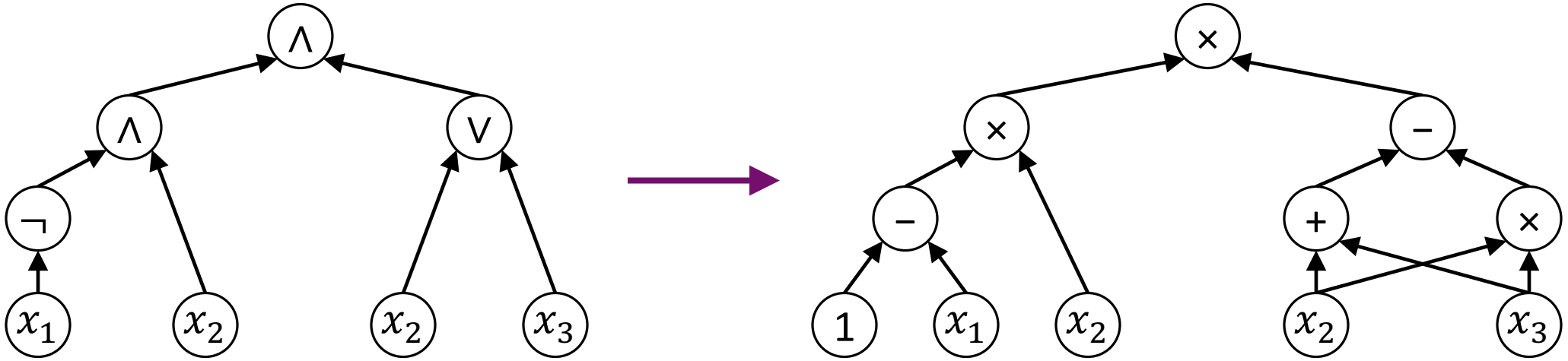
Lemma: For a Boolean formula φ of size n, the total degree of g_φ is at most n.

Proof (by Induction): **Base case:** Leaf nodes (of size 1) have degree 1.

Inductive case: Let t be a node with subtrees u, v of sizes a, b. Induction gives $\deg(u) \le a$ and $\deg(v) \le b$.

- $\neg$: $\deg(1-u) \le \max\{0, \deg(u)\} \le a \le \text{size}(t)$.
- $\wedge$: $\deg(uv) \le a + b \le \text{size}(t)$.
- $\vee$: $\deg(u + v - uv) \le \max\{a, b, a+b\} = a + b \le \text{size}(t)$.

Example:



$$\varphi = (\neg x_1 \wedge x_2) \wedge (x_2 \vee x_3)$$

$$g_\varphi = (1 - x_1)x_2(x_2 + x_3 - x_2x_3)$$

Arithmetization

Lemma: For a Boolean formula φ of size n, the total degree of g_φ is at most n.

Proof (by Induction): **Base case:** Leaf nodes (of size 1) have degree 1.

Inductive case: Let t be a node with subtrees u, v of sizes a, b. Induction gives $\deg(u) \leq a$ and $\deg(v) \leq b$.

- $\neg$: $\deg(1-u) \leq \max\{0, \deg(u)\} \leq a \leq \text{size}(t)$.
- $\wedge$: $\deg(uv) \leq a + b \leq \text{size}(t)$.
- $\vee$: $\deg(u + v - uv) \leq \max\{a, b, a+b\} = a + b \leq \text{size}(t)$.

Mini-Quiz 22.2

For $\varphi = (x_1 \vee x_2) \wedge \neg x_1$, which of the following polynomial is its arithmetization?

a) $(x_1 + x_2 - x_1x_2) + (1 - x_1)$

b) $(x_1 + x_2 - x_1x_2)(1 - x_1)$

c) $(x_1 + x_2)(1 - x_1)$

#SAT ∈ IP

Recall: $\#\mathrm{SAT} = \{\, \langle \varphi, K \rangle \mid K = N(\varphi) \,\}$

Theorem: $\#\mathrm{SAT} \in \mathrm{IP}$.

Proof: Given input $\langle \varphi, K \rangle$, with $|\varphi| = n$.

- Arithmetize the Boolean formula φ into a polynomial g_φ.
- Apply the Sum-Check Protocol to $\langle g_\varphi, K \rangle$ to verify:

$$N(\varphi) = \sum_{b \in \{0,1\}^m} \varphi(b) = \sum_{b \in \{0,1\}^m} g_\varphi(b) \stackrel{?}{=} K$$

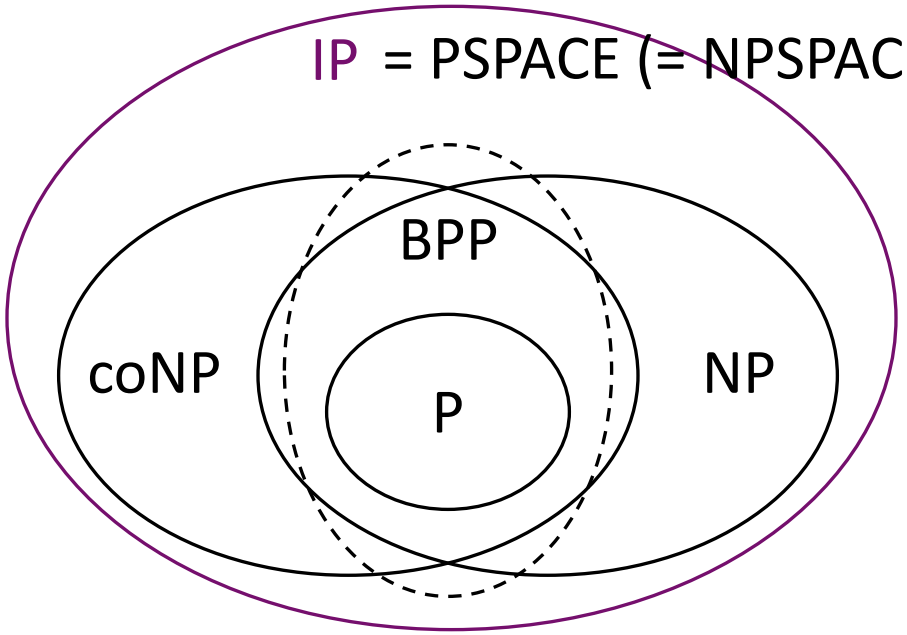
The number of variables of g_φ is $\leq n$.

The degree of each variable is $\leq n$.

Pick a prime $q > \max\{2^n,\ 3n^2\}$. $q > 2^n$ ensures $N(\varphi) = K \pmod q$ means $N(\varphi) = K$.

The error probability is at most $\frac{md}{q} \leq \frac{n^2}{q} \leq \frac{1}{3}$.

IP = PSPACE: by proving TQBF ∈ IP



Recall: TQBF (the set of true quantified Boolean formulas) is PSPACE-complete.

$$\forall x_1 \exists x_2 \cdots Q_m x_m \, \varphi(x_1, \ldots, x_m), \qquad Q_i \in \{\forall, \exists\}.$$

Theorem: TQBF ∈ IP.

Corollary: PSPACE ⊆ IP, hence PSPACE = IP.

Arithmetizing a QBF

Proof Idea: Arithmetize a quantified Boolean formula:

$$\forall x_1 \exists x_2 \cdots Q_m x_m \, \varphi(x_1, \ldots, x_m), \qquad Q_i \in \{\forall, \exists\}.$$

(1) Arithmetize φ into g_φ (as we did in #SAT).

(2) Arithmetize the quantifiers: From right to left. Let g be an intermediate result.

$$\forall x_m \, g(x_1, \ldots, x_m) \;\Rightarrow\; \prod_{x_m \in \{0,1\}} g(x_1, \ldots, x_m) \overset{\text{def}}{=} g(x_1, \ldots, x_{m-1}, 0) \cdot g(x_1, \ldots, x_{m-1}, 1)$$

$$\exists x_m \, g(x_1, \ldots, x_m) \;\Rightarrow\; \coprod_{x_m \in \{0,1\}} g(x_1, \ldots, x_m) \overset{\text{def}}{=} 1 - \big(1 - g(x_1, \ldots, x_{m-1}, 0)\big)\big(1 - g(x_1, \ldots, x_{m-1}, 1)\big)$$

Result:

$$\forall x_1 \exists x_2 \cdots \forall x_m \, \varphi(x_1, \ldots, x_m) \;\Rightarrow\; \prod_{x_1 \in \{0,1\}} \coprod_{x_2 \in \{0,1\}} \cdots \prod_{x_m \in \{0,1\}} g_\varphi(x_1, \ldots, x_m)$$

The QBF is TRUE $\Leftrightarrow$ The right-hand-side constant is 1

Degree Explosion

Result after arithmetization:

$$\forall x_1 \exists x_2 \cdots \forall x_m \, \varphi(x_1, \ldots, x_m) \;\Rightarrow\; \prod_{x_1 \in \{0,1\}} \coprod_{x_2 \in \{0,1\}} \cdots \prod_{x_m \in \{0,1\}} g_\varphi(x_1, \ldots, x_m)$$

Problem: The degrees of intermediate polynomials are too large (2^m):

- Because each product operation doubles the degree:

$$\deg_{x_i}\left(\prod_{x_m \in \{0,1\}} g_\varphi(x_1, \ldots, x_m) \right) \leq d + d = 2d.$$

The Check-Sum Protocol only works for polynomials with a small degree.

How to fix?

Observation: For a Boolean variable $x_i \in \{0,1\}$, $x_i^k = x_i$ for every $k \geq 1$.

Solution: Reduce the degree to 1 by replacing x_i^k with x_i. Denote this operation by R_{x_i}

Example: $R_{x_i}\left(x_i^3 x_j^2 + 2x_i^2 + 5 \right) = x_i x_j^2 + 2x_i + 5$

Equivalently: $R_{x_i}\, g = (1 - x_i)\, g|_{x_i=0} + x_i\, g|_{x_i=1}$. "Linearization"

Arithmetization + Linearization:

Result after Arithmetization + Linearization:

$$\forall x_1 \exists x_2 \cdots \forall x_m \, \varphi(x_1, \ldots, x_m)$$

$$\Rightarrow \prod_{x_1 \in \{0,1\}} R_{x_1} \coprod_{x_2 \in \{0,1\}} R_{x_1} R_{x_2} \prod_{x_3 \in \{0,1\}} \cdots R_{x_1} \cdots R_{x_{m-1}} \prod_{x_m \in \{0,1\}} R_{x_1} \cdots R_{x_m} \, g_\varphi(x_1, \ldots, x_m)$$

$$= \mathcal{O}_1 \mathcal{O}_2 \cdots \mathcal{O}_\ell \, g_\varphi(x_1, \ldots, x_m)$$

The QBF is TRUE $\Leftrightarrow \mathcal{O}_1 \mathcal{O}_2 \cdots \mathcal{O}_\ell \, g_\varphi(x_1, \ldots, x_m) = 1$

Verify by an interactive protocol (similar to Sum-Check):

- Remove one operator at each round:
  - Prover sends a univariate polynomial $P_k(X_i) = \mathcal{O}_{k+1} \cdots \mathcal{O}_\ell \, g_\varphi(r_1, , \ldots, r_{i-1}, X_i, x_{i+1}, \ldots, x_m)$
  - Verifier sends a random r_i to verify the correctness of $P_k(r_i)$.
- Finally, check $g_\varphi(r_1, \ldots, r_m) = P_\ell(r_m)$

Details: "IP = PSPACE", Notes on Complexity Theory (Lecture 19), by Jonathan Katz:
https://www.cs.umd.edu/~jkatz/complexity/f11/lecture19.pdf

Arithmetization + Linearization:

Result after Arithmetization + Linearization:

$$\forall x_1 \exists x_2 \cdots \forall x_m \, \varphi(x_1, \ldots, x_m)$$

$$\Rightarrow \prod_{x_1 \in \{0,1\}} R_{x_1} \coprod_{x_2 \in \{0,1\}} R_{x_1} R_{x_2} \prod_{x_3 \in \{0,1\}} \cdots R_{x_1} \cdots R_{x_{m-1}} \prod_{x_m \in \{0,1\}} R_{x_1} \cdots R_{x_m} \, g_\varphi(x_1, \ldots, x_m)$$

$$= \mathcal{O}_1 \mathcal{O}_2 \cdots \mathcal{O}_\ell \, g_\varphi(x_1, \ldots, x_m)$$

The QBF is TRUE $\Leftrightarrow \mathcal{O}_1 \mathcal{O}_2 \cdots \mathcal{O}_\ell \, g_\varphi(x_1, \ldots, x_m) = 1$

Mini-Quiz 22.3

For $\varphi = (x_1 \vee x_2) \wedge \neg x_1$ (from Mini-Quiz 22.2),

what is the polynomial after arithmetization and linearization?

Review of today

- $\#\mathrm{SAT} \in$ IP, hence coNP $\subseteq$ IP.

- Arithmetization reduces $\#\mathrm{SAT}$ to a Sum-Check problem with a low-degree polynomial.

- "Arithmetization + Linearization" gives $\mathrm{TQBF} \in$ IP, hence IP $=$ PSPACE.

Additional References

Jonathan Katz: Notes on Complexity Theory, Lecture 19: IP = PSPACE (2011)
https://www.cs.umd.edu/~jkatz/complexity/f11/lecture19.pdf

Justin Thaler: Interactive Proofs (Part II)
Simons Institute tutorial, 26 August 2019.
https://www.youtube.com/watch?v=w-6R3TxJ5dw

Michael Sipser: Introduction to the Theory of Computation
Chapter 10.4: Interactive Proof Systems.