Derandomizing the Isolation Lemma and Lower Bounds for Circuit Size

V. Arvind and Partha Mukhopadhyay
The Institute of Mathematical Sciences
India

27th August 2008

Isolation Lemma (Mulmuley-Vazirani-Vazirani 1987)

- U be a set (universe) of size n and $\mathcal{F} \subseteq 2^U$ be any family of subsets of U.
- Let $w : U \to \mathbb{Z}^+$ be a weight function.
- For $T \subseteq U$, define its weight $w(T)$ as $w(T) = \sum_{u \in T} w(u)$.

Isolation Lemma (Mulmuley-Vazirani-Vazirani 1987)

- U be a set (universe) of size n and $\mathcal{F} \subseteq 2^U$ be any family of subsets of U.
- Let $w : U \rightarrow \mathbb{Z}^+$ be a weight function.
- For $T \subseteq U$, define its weight $w(T)$ as $w(T) = \sum_{u \in T} w(u)$.

Isolation Lemma (Mulmuley-Vazirani-Vazirani 1987)

- U be a set (universe) of size n and $\mathcal{F} \subseteq 2^U$ be any family of subsets of U.
- Let $w : U \to \mathbb{Z}^+$ be a weight function.
- For $T \subseteq U$, define its weight $w(T)$ as $w(T) = \sum_{u \in T} w(u)$.

Isolation Lemma

- Let w be any random weight assignment $w : U \to [2n]$.
- Isolation Lemma guarantees that with high probability (at least $1/2$) there will be a unique minimum weight set in $\mathcal{F}$.

Isolation Lemma

- Let w be any random weight assignment $w : U \rightarrow [2n]$.
- Isolation Lemma guarantees that with high probability (at least $1/2$) there will be a unique minimum weight set in $\mathcal{F}$.

Important applications of Isolation Lemma

- Randomized NC algorithm for computing maximum cardinality matchings for general graphs. (Mulmuley-Vazirani-Vazirani 1987)
- $\mathrm{NL} \subset \mathrm{UL/poly}$ (Klaus Reinhardt and Eric Allender 2000).
- SAT is many-one reducible via randomized reductions to USAT.

Important applications of Isolation Lemma

- Randomized NC algorithm for computing maximum cardinality matchings for general graphs. (Mulmuley-Vazirani-Vazirani 1987)
- $\mathrm{NL} \subset \mathrm{UL/poly}$ (Klaus Reinhardt and Eric Allender 2000).
- SAT is many-one reducible via randomized reductions to USAT.

Important applications of Isolation Lemma

- Randomized NC algorithm for computing maximum cardinality matchings for general graphs. (Mulmuley-Vazirani-Vazirani 1987)
- $\mathrm{NL} \subset \mathrm{UL/poly}$ (Klaus Reinhardt and Eric Allender 2000).
- SAT is many-one reducible via randomized reductions to USAT.

Two outstanding open problems in complexity theory

- Is the matching problem in in deterministic NC ?
- Is $\mathrm{NL} \subseteq \mathrm{UL}$?

Both the problems will be solved if Isolation Lemma can be derandomized.

Two outstanding open problems in complexity theory

- Is the matching problem in in deterministic NC ?
- Is $\mathrm{NL} \subseteq \mathrm{UL}$?

Both the problems will be solved if Isolation Lemma can be derandomized.

Derandomizing Isolation Lemma

- In all well known applications of Isolation Lemma number of set system is $2^{n^{O(1)}}$.
- So derandomization is plausible (Agrawal 2007, Barbados workshop on CC).
- Main Question Can we derandomize some *special cases* of the Isolation Lemma.

Derandomizing Isolation Lemma

- In all well known applications of Isolation Lemma number of set system is $2^{n^{O(1)}}$.
- So derandomization is plausible (Agrawal 2007, Barbados workshop on CC).
- Main Question Can we derandomize some *special cases* of the Isolation Lemma.

Derandomizing Isolation Lemma

- In all well known applications of Isolation Lemma number of set system is $2^{n^{O(1)}}$.
- So derandomization is plausible (Agrawal 2007, Barbados workshop on CC).
- Main Question Can we derandomize some *special cases* of the Isolation Lemma.

Isolation Lemma - Our setting

- The universe $U = [n]$.
- An n-input boolean circuit C and $\mathrm{size}(C) = m$.
- Each subset $S \subseteq U$ corresponds to its characteristic binary string $\chi_S \in \{0,1\}^n$.
- n-input boolean circuit C implicitly defines the set system

$$\mathcal{F}_C = \{S \subseteq [n] \mid C(\chi_S) = 1\}.$$

- Also, there is only exponential number of set systems.

Isolation Lemma - Our setting

- The universe $U = [n]$.
- An n-input boolean circuit C and $\mathrm{size}(C) = m$.
- Each subset $S \subseteq U$ corresponds to its characteristic binary string $\chi_S \in \{0,1\}^n$.
- n-input boolean circuit C implicitly defines the set system

$$\mathcal{F}_C = \{S \subseteq [n] \mid C(\chi_S) = 1\}.$$

- Also, there is only exponential number of set systems.

Isolation Lemma - Our setting

- The universe $U = [n]$.
- An n-input boolean circuit C and $\mathrm{size}(C) = m$.
- Each subset $S \subseteq U$ corresponds to its characteristic binary string $\chi_S \in \{0,1\}^n$.
- n-input boolean circuit C implicitly defines the set system

$$\mathcal{F}_C = \{S \subseteq [n] \mid C(\chi_S) = 1\}.$$

- Also, there is only exponential number of set systems.

Isolation Lemma - Our setting

- The universe $U = [n]$.
- An n-input boolean circuit C and $\mathrm{size}(C) = m$.
- Each subset $S \subseteq U$ corresponds to its characteristic binary string $\chi_S \in \{0,1\}^n$.
- n-input boolean circuit C implicitly defines the set system

$$\mathcal{F}_C = \{S \subseteq [n] \mid C(\chi_S) = 1\}.$$

- Also, there is only exponential number of set systems.

Isolation Lemma - Our setting

- The universe $U = [n]$.
- An n-input boolean circuit C and $\mathrm{size}(C) = m$.
- Each subset $S \subseteq U$ corresponds to its characteristic binary string $\chi_S \in \{0,1\}^n$.
- n-input boolean circuit C implicitly defines the set system

$$\mathcal{F}_C = \{S \subseteq [n] \mid C(\chi_S) = 1\}.$$

- Also, there is only exponential number of set systems.

Our Setting

- $w : U \to [2n]$: random weight assignment.
- Isolation Lemma:

$$\mathrm{Prob}_w[\text{ There exists a unique minimum weight set in } \mathcal{F}_C] \geq \frac{1}{2}.$$

- Can we derandomize?

Our Setting

- $w : U \rightarrow [2n]$: random weight assignment.
- Isolation Lemma:

$$\mathrm{Prob}_w[\text{ There exists a unique minimum weight set in } \mathcal{F}_C] \geq \frac{1}{2}.$$

- Can we derandomize?

Our Setting

- $w : U \longrightarrow [2n]$: random weight assignment.
- Isolation Lemma:

$$\mathrm{Prob}_w[\text{ There exists a unique minimum weight set in } \mathcal{F}_C] \geq \frac{1}{2}.$$

- Can we derandomize?

A non black-box derandomization Hypothesis

- C is an n-input boolean circuit.
- A deterministic algorithm $\mathcal{A}_1$ takes as input (C, n).
- $\mathcal{A}$ outputs weight functions $w_1, w_2, \cdots, w_t$ $(w_i : [n] \to [2n])$: $\exists i$, s.t w_i assigns a unique minimum weight set in $\mathcal{F}_C$.
- $\mathcal{A}_1$ runs in time subexponential in $\mathrm{size}(C)$.

A non black-box derandomization Hypothesis

- C is an n-input boolean circuit.
- A deterministic algorithm $\mathcal{A}_1$ takes as input (C, n).
- $\mathcal{A}$ outputs weight functions $w_1, w_2, \cdots, w_t$ $(w_i : [n] \rightarrow [2n])$: $\exists i$, s.t w_i assigns a unique minimum weight set in $\mathcal{F}_C$.
- $\mathcal{A}_1$ runs in time subexponential in $\mathrm{size}(C)$.

A non black-box derandomization Hypothesis

- C is an n-input boolean circuit.
- A deterministic algorithm $\mathcal{A}_1$ takes as input (C, n).
- $\mathcal{A}$ outputs weight functions $w_1, w_2, \cdots, w_t$ $(w_i : [n] \rightarrow [2n])$: $\exists i$, s.t w_i assigns a unique minimum weight set in $\mathcal{F}_C$.
- $\mathcal{A}_1$ runs in time subexponential in $\mathrm{size}(C)$.

A non black-box derandomization Hypothesis

- C is an n-input boolean circuit.
- A deterministic algorithm $\mathcal{A}_1$ takes as input (C, n).
- $\mathcal{A}$ outputs weight functions $w_1, w_2, \cdots, w_t$ $(w_i : [n] \rightarrow [2n])$: $\exists i$, s.t w_i assigns a unique minimum weight set in $\mathcal{F}_C$.
- $\mathcal{A}_1$ runs in time subexponential in $\mathrm{size}(C)$.

Black-box derandomization Hypothesis

- $\mathcal{A}_2$ takes (m, n) in unary.
- $\mathcal{A}$ outputs weight functions $w_1, w_2, \cdots, w_t$ $(w_i : [n] \rightarrow [2n])$.
- For each size m boolean circuit C with n inputs: $\exists i$, s.t w_i assigns a unique minimum weight set in $\mathcal{F}_C$.
- $\mathcal{A}_2$ runs in time polynomial in m.

Black-box derandomization Hypothesis

- $\mathcal{A}_2$ takes (m, n) in unary.
- $\mathcal{A}$ outputs weight functions $w_1, w_2, \cdots, w_t$ $(w_i : [n] \rightarrow [2n])$.
- For each size m boolean circuit C with n inputs: $\exists i$, s.t w_i assigns a unique minimum weight set in $\mathcal{F}_C$.
- $\mathcal{A}_2$ runs in time polynomial in m.

Black-box derandomization Hypothesis

- $\mathcal{A}_2$ takes (m, n) in unary.
- $\mathcal{A}$ outputs weight functions $w_1, w_2, \cdots, w_t$ $(w_i : [n] \rightarrow [2n])$.
- For each size m boolean circuit C with n inputs: $\exists i$, s.t w_i assigns a unique minimum weight set in $\mathcal{F}_C$.
- $\mathcal{A}_2$ runs in time polynomial in m.

Black-box derandomization Hypothesis

- $\mathcal{A}_2$ takes (m, n) in unary.
- $\mathcal{A}$ outputs weight functions $w_1, w_2, \cdots, w_t$ $(w_i : [n] \rightarrow [2n])$.
- For each size m boolean circuit C with n inputs: $\exists i$, s.t w_i assigns a unique minimum weight set in $\mathcal{F}_C$.
- $\mathcal{A}_2$ runs in time polynomial in m.

Derandomization Consequences (results)

- Non black-box derandomization $\Rightarrow$ either $\mathrm{NEXP} \not\subset \mathrm{P/poly}$ or *Perm* does not have polynomial size *noncommutative arithmetic circuits*.
- Black-box derandomization $\Rightarrow$ an explicit multilinear polynomial $f_n(x_1, x_2, \cdots, x_n) \in \mathbb{F}[x_1, x_2, \cdots, x_n]$ (in *commuting* variables) does not have commutative arithmetic circuits of size $2^{o(n)}$.

Derandomization Consequences (results)

- Non black-box derandomization $\Rightarrow$ either $\mathrm{NEXP} \not\subset \mathrm{P/poly}$ or *Perm* does not have polynomial size *noncommutative arithmetic circuits*.
- Black-box derandomization $\Rightarrow$ an explicit multilinear polynomial $f_n(x_1, x_2, \cdots, x_n) \in \mathbb{F}[x_1, x_2, \cdots, x_n]$ (in *commuting* variables) does not have commutative arithmetic circuits of size $2^{o(n)}$.

Non black-box derandomization : proof idea

- Using Isolation Lemma, design a randomized polynomial-time identity testing algorithm (PIT) for small degree noncommutative circuits.
- Derandomize the algorithm (subexponential time) using Hypothesis 1.

Non black-box derandomization : proof idea

- Using Isolation Lemma, design a randomized polynomial-time identity testing algorithm (PIT) for small degree noncommutative circuits.
- Derandomize the algorithm (subexponential time) using Hypothesis 1.

Idea behind the proof cont'd.

- Noncommutative version of Impagliazzo-Kabanets 2003: Derandomizing the PIT for small degree noncommutative circuit $\Rightarrow$ either $\mathrm{NEXP} \not\subset \mathrm{P}/\mathrm{poly}$ or permanent has no poly-size noncommutative circuit (Arvind, Mukhopadhyay and Srinivasan 2008).

Noncommutative PIT

- A noncommutative arithmetic circuit C computes a polynomial in $\mathbb{F}\{x_1, x_2, \cdots, x_n\}$ $(x_i x_j \neq x_j x_i)$ using $+$ and $\times$ gate.
- (Bogdanov and Wee'05) Randomized poly-time PIT for noncommutative circuits of small degree (based on classic theorem of Amitsur and Levitzki 1950).
- New algorithm is based on Isolation Lemma and Automata Theory.
- Recently, using automata theory a deterministic PIT algorithm for noncommutative circuit computing sparse polynomial is given (Arvind, Mukhopadhyay and Srinivasan 2008).

Noncommutative PIT

- A noncommutative arithmetic circuit C computes a polynomial in $\mathbb{F}\{x_1, x_2, \cdots, x_n\}$ $(x_i x_j \neq x_j x_i)$ using $+$ and $\times$ gate.
- (Bogdanov and Wee'05) Randomized poly-time PIT for noncommutative circuits of small degree (based on classic theorem of Amitsur and Levitzki 1950).
- New algorithm is based on Isolation Lemma and Automata Theory.
- Recently, using automata theory a deterministic PIT algorithm for noncommutative circuit computing sparse polynomial is given (Arvind, Mukhopadhyay and Srinivasan 2008).

Noncommutative PIT

- A noncommutative arithmetic circuit C computes a polynomial in $\mathbb{F}\{x_1, x_2, \cdots, x_n\}$ $(x_i x_j \neq x_j x_i)$ using $+$ and $\times$ gate.
- (Bogdanov and Wee'05) Randomized poly-time PIT for noncommutative circuits of small degree (based on classic theorem of Amitsur and Levitzki 1950).
- New algorithm is based on Isolation Lemma and Automata Theory.
- Recently, using automata theory a deterministic PIT algorithm for noncommutative circuit computing sparse polynomial is given (Arvind, Mukhopadhyay and Srinivasan 2008).

Noncommutative PIT

- A noncommutative arithmetic circuit C computes a polynomial in $\mathbb{F}\{x_1, x_2, \cdots, x_n\}$ $(x_i x_j \neq x_j x_i)$ using $+$ and $\times$ gate.
- (Bogdanov and Wee'05) Randomized poly-time PIT for noncommutative circuits of small degree (based on classic theorem of Amitsur and Levitzki 1950).
- New algorithm is based on Isolation Lemma and Automata Theory.
- Recently, using automata theory a deterministic PIT algorithm for noncommutative circuit computing sparse polynomial is given (Arvind, Mukhopadhyay and Srinivasan 2008).

Some Automata Theory Background

- A finite automaton $A = (Q, \Sigma = \{x_1, \cdots, x_n\}, \delta, \{q_0\}, \{q_f\})$.
- $(Q, \Sigma, \delta, q_0, q_f) \rightarrow$ (alphabet, states set, transition function, initial state, final state).
- For $b \in \Sigma$, the 0-1 matrix $M_b \in \mathbb{F}^{|Q| \times |Q|}$:

$$M_b(q, q') = \begin{cases} 1 & \text{if } \delta_b(q) = q', \\ 0 & \text{otherwise.} \end{cases}$$

Some Automata Theory Background

- A finite automaton $A = (Q, \Sigma = \{x_1, \cdots, x_n\}, \delta, \{q_0\}, \{q_f\})$.
- $(Q, \Sigma, \delta, q_0, q_f) \rightarrow$ (alphabet, states set, transition function, initial state, final state).
- For $b \in \Sigma$, the 0-1 matrix $M_b \in \mathbb{F}^{|Q| \times |Q|}$:

$$M_b(q, q') = \begin{cases} 1 & \text{if } \delta_b(q) = q', \\ 0 & \text{otherwise.} \end{cases}$$

Some Automata Theory Background

- A finite automaton $A = (Q, \Sigma = \{x_1, \cdots, x_n\}, \delta, \{q_0\}, \{q_f\})$.
- $(Q, \Sigma, \delta, q_0, q_f) \rightarrow$ (alphabet, states set, transition function, initial state, final state).
- For $b \in \Sigma$, the 0-1 matrix $M_b \in \mathbb{F}^{|Q| \times |Q|}$:

$$M_b(q, q') = \begin{cases} 1 & \text{if } \delta_b(q) = q', \\ 0 & \text{otherwise.} \end{cases}$$

Some Automata Theory Background

- For any $w = w_1 w_2 \cdots w_k \in \Sigma^*$, the matrix $M_w = M_{w_1} M_{w_2} \cdots M_{w_k}$.
- Easy fact: $M_w(q_0, q_f) = 1$ if and only if w is accepted by the automaton A.

Some Automata Theory Background

- For any $w = w_1 w_2 \cdots w_k \in \Sigma^*$, the matrix $M_w = M_{w_1} M_{w_2} \cdots M_{w_k}$.
- Easy fact: $M_w(q_0, q_f) = 1$ if and only if w is accepted by the automaton A.

Run of an automaton over a noncommutative circuit

- C be any given *noncommutative* arithmetic circuit computing f.
- Output matrix $M_{out}^{A} = C(M_{x_1}, M_{x_2} \cdots, M_{x_n})$.

Run of an automaton over a noncommutative circuit

- C be any given *noncommutative* arithmetic circuit computing f.
- Output matrix $M_{out}^{A} = C(M_{x_1}, M_{x_2} \cdots, M_{x_n})$.

Crucial Observation

- The output is always 0 when $f \equiv 0$.
- If A accepts precisely one monomial (m) of f then $M^A_{out}(q_0, q_f) = c$ (coefficient of m in f is c).
- That's an identity test !!

Crucial Observation

- The output is always 0 when $f \equiv 0$.
- If A accepts precisely one monomial (m) of f then $M^A_{out}(q_0, q_f) = c$ (coefficient of m in f is c).
- That's an identity test !!

Crucial Observation

- The output is always 0 when $f \equiv 0$.
- If A accepts precisely one monomial (m) of f then $M_{out}^{A}(q_0, q_f) = c$ (coefficient of m in f is c).
- That's an identity test !!

Identity Testing Algorithm based on Isolation Lemma

- Input $f \in \mathbb{F}\{x_1, x_2, \cdots, x_n\}$ given by an arithmetic circuit C of.
- d be an upper bound on the degree of f.
- $[d] = \{1, 2, \cdots, d\}$ and $[n] = \{1, 2, \cdots, n\}$.
- The universe (for Isolation Lemma) $U = [d] \times [n]$.

Identity Testing Algorithm based on Isolation Lemma

- Input $f \in \mathbb{F}\{x_1, x_2, \cdots, x_n\}$ given by an arithmetic circuit C of.
- d be an upper bound on the degree of f.
- $[d] = \{1, 2, \cdots, d\}$ and $[n] = \{1, 2, \cdots, n\}$.
- The universe (for Isolation Lemma) $U = [d] \times [n]$.

Identity Testing Algorithm based on Isolation Lemma

- Input $f \in \mathbb{F}\{x_1, x_2, \cdots, x_n\}$ given by an arithmetic circuit C of.
- d be an upper bound on the degree of f.
- $[d] = \{1, 2, \cdots, d\}$ and $[n] = \{1, 2, \cdots, n\}$.
- The universe (for Isolation Lemma) $U = [d] \times [n]$.

Identity Testing Algorithm based on Isolation Lemma

- Input $f \in \mathbb{F}\{x_1, x_2, \cdots, x_n\}$ given by an arithmetic circuit C of.
- d be an upper bound on the degree of f.
- $[d] = \{1, 2, \cdots, d\}$ and $[n] = \{1, 2, \cdots, n\}$.
- The universe (for Isolation Lemma) $U = [d] \times [n]$.

Identity Testing Algorithm

- Let $v = x_{i_1} x_{i_2} \cdots x_{i_t}$ be a nonzero monomial of f.
- Identify v with $S_v \subset U$:

$$S_v = \{(1, i_1), (2, i_2), \cdots, (t, i_t)\}$$

- Set system:

$$\mathcal{F} = \{S_v \mid v \text{ is a nonzero monomial in } f\}$$

Identity Testing Algorithm

- Let $v = x_{i_1} x_{i_2} \cdots x_{i_t}$ be a nonzero monomial of f.
- Identify v with $S_v \subset U$:

$$S_v = \{(1, i_1), (2, i_2), \cdots, (t, i_t)\}$$

- Set system:

$$\mathcal{F} = \{S_v \mid v \text{ is a nonzero monomial in } f\}$$

Identity Testing Algorithm

- Let $v = x_{i_1} x_{i_2} \cdots x_{i_t}$ be a nonzero monomial of f.
- Identify v with $S_v \subset U$:

$$S_v = \{(1, i_1), (2, i_2), \cdots, (t, i_t)\}$$

- Set system:

$$\mathcal{F} = \{S_v \mid v \text{ is a nonzero monomial in } f\}$$

Intuition behind the Identity Testing Algorithm

- Assign random weights from $[2dn]$ to the elements of U,
- (Isolation Lemma) With probability at least 1/2, there is a unique minimum weight set in $\mathcal{F}$.
- Goal Construct a family of small size automatons $\{A_{w,t}\}_{w \in [2nd^2], t \in [d]}$.
- $A_{w,t}$ precisely accepts all the strings (corresponding to the monomials) v of length t, such that the weight of S_v is w.

Intuition behind the Identity Testing Algorithm

- Assign random weights from $[2dn]$ to the elements of U,
- (Isolation Lemma) With probability at least $1/2$, there is a unique minimum weight set in $\mathcal{F}$.
- Goal Construct a family of small size automatons $\{A_{w,t}\}_{w\in[2nd^2],t\in[d]}$.
- $A_{w,t}$ precisely accepts all the strings (corresponding to the monomials) v of length t, such that the weight of S_v is w.

Intuition behind the Identity Testing Algorithm

- Assign random weights from $[2dn]$ to the elements of U,
- (Isolation Lemma) With probability at least $1/2$, there is a unique minimum weight set in $\mathcal{F}$.
- Goal Construct a family of small size automatons $\{A_{w,t}\}_{w\in[2nd^2],t\in[d]}$:
- $A_{w,t}$ precisely accepts all the strings (corresponding to the monomials) v of length t, such that the weight of S_v is w.

Intuition behind the Identity Testing Algorithm

- Assign random weights from $[2dn]$ to the elements of U,
- (Isolation Lemma) With probability at least $1/2$, there is a unique minimum weight set in $\mathcal{F}$.
- Goal Construct a family of small size automatons $\{A_{w,t}\}_{w\in[2nd^2],t\in[d]}$.
- $A_{w,t}$ precisely accepts all the strings (corresponding to the monomials) v of length t, such that the weight of S_v is w.

Intuition of the Identity Testing algorithm

- For each $A \in \{A_{w,t}\}$ compute the run of A on C.
- (Using the isolation lemma) The automata corresponding to the minimum weight will precisely accept (isolate) only one string (monomial).
- The automata family is easy to construct.

Intuition of the Identity Testing algorithm

- For each $A \in \{A_{w,t}\}$ compute the run of A on C.
- (Using the isolation lemma) The automata corresponding to the minimum weight will precisely accept (isolate) only one string (monomial).
- The automata family is easy to construct.

Intuition of the Identity Testing algorithm

- For each $A \in \{A_{w,t}\}$ compute the run of A on C.
- (Using the isolation lemma) The automata corresponding to the minimum weight will precisely accept (isolate) only one string (monomial).
- The automata family is easy to construct.

Crucial Observation

- C be a noncommutative arithmetic circuit of small degree and m is a given monomial.
- Easy algorithm to check if m is a nonzero monomial in C.
- Construct an automaton A that accepts only m and compute run on C.
- Thus, a boolean circuit $\hat{C}$ (of size $\mathrm{poly}(\mathrm{size}(C))$), $\mathcal{F}_{\hat{C}}$ defines the monomials of C.

Crucial Observation

- C be a noncommutative arithmetic circuit of small degree and m is a given monomial.
- Easy algorithm to check if m is a nonzero monomial in C.
- Construct an automaton A that accepts only m and compute run on C.
- Thus, a boolean circuit $\hat{C}$ (of size $\mathrm{poly}(\mathrm{size}(C))$), $\mathcal{F}_{\hat{C}}$ defines the monomials of C.

Crucial Observation

- C be a noncommutative arithmetic circuit of small degree and m is a given monomial.
- Easy algorithm to check if m is a nonzero monomial in C.
- Construct an automaton A that accepts only m and compute run on C.
- Thus, a boolean circuit $\hat{C}$ (of size $\mathrm{poly}(\mathrm{size}(C))$), $\mathcal{F}_{\hat{C}}$ defines the monomials of C.

Crucial Observation

- C be a noncommutative arithmetic circuit of small degree and m is a given monomial.
- Easy algorithm to check if m is a nonzero monomial in C.
- Construct an automaton A that accepts only m and compute run on C.
- Thus, a boolean circuit $\hat{C}$ (of size $\mathrm{poly}(\mathrm{size}(C))$), $\mathcal{F}_{\hat{C}}$ defines the monomials of C.

Non black-box derandomization

- Given noncommutative arithmetic circuit C.
- Compute boolean circuit $\hat{C}$.
- $\mathcal{A}_1(\hat{C}, n) = \{w_1, w_2, \cdots, w_n\}$.
- Identity testing using $\{w_i\}$'s.
- Run time: $\mathrm{subexp}(\mathrm{size}(\hat{C}, n))$.

Non black-box derandomization

- Given noncommutative arithmetic circuit C.
- Compute boolean circuit $\hat{C}$.
- $\mathcal{A}_1(\hat{C}, n) = \{w_1, w_2, \cdots, w_n\}$.
- Identity testing using $\{w_i\}$'s.
- Run time: $\mathrm{subexp}(\mathrm{size}(\hat{C}, n))$.

Non black-box derandomization

- Given noncommutative arithmetic circuit C.
- Compute boolean circuit $\hat{C}$.
- $\mathcal{A}_1(\hat{C}, n) = \{w_1, w_2, \cdots, w_n\}$.
- Identity testing using $\{w_i\}$'s.
- Run time: $\mathrm{subexp}(\mathrm{size}(\hat{C}, n))$.

Non black-box derandomization

- Given noncommutative arithmetic circuit C.
- Compute boolean circuit $\hat{C}$.
- $\mathcal{A}_1(\hat{C}, n) = \{w_1, w_2, \cdots, w_n\}$.
- Identity testing using $\{w_i\}$'s.
- Run time: $\mathrm{subexp}(\mathrm{size}(\hat{C}, n))$.

Non black-box derandomization

- Given noncommutative arithmetic circuit C.
- Compute boolean circuit $\hat{C}$.
- $\mathcal{A}_1(\hat{C}, n) = \{w_1, w_2, \cdots, w_n\}$.
- Identity testing using $\{w_i\}$'s.
- Run time: $\mathrm{subexp}(\mathrm{size}(\hat{C}, n))$.

Consequence of Hypothesis 2

- Goal To construct an explicit multilinear polynomial f in $\mathbb{F}[x_1, x_2, \cdots, x_n]$ that does not have $2^{o(n)}$ size arithmetic circuit.
- Define a multilinear polynomial:

$$f(x_1, x_2, \cdots, x_n) = \sum_{S \subseteq [n]} c_S \prod_{i \in S} x_i,$$

- we need to fix c_S's suitably.

Consequence of Hypothesis 2

- Goal To construct an explicit multilinear polynomial f in $\mathbb{F}[x_1, x_2, \cdots, x_n]$ that does not have $2^{o(n)}$ size arithmetic circuit.
- Define a multilinear polynomial:

$$f(x_1, x_2, \cdots, x_n) = \sum_{S \subseteq [n]} c_S \prod_{i \in S} x_i,$$

- we need to fix c_S's suitably.

Consequence of Hypothesis 2

- Goal To construct an explicit multilinear polynomial f in $\mathbb{F}[x_1, x_2, \cdots, x_n]$ that does not have $2^{o(n)}$ size arithmetic circuit.
- Define a multilinear polynomial:

$$f(x_1, x_2, \cdots, x_n) = \sum_{S \subseteq [n]} c_S \prod_{i \in S} x_i,$$

- we need to fix c_S's suitably.

Important Observation

- Let f be a multilinear polynomial given by a circuit $\hat{C}$ and $m = \prod_{i \in S} x_i$ is a monomial.
- A small size boolean circuit C can decide whether m is a nonzero monomial in f.
- Just substitute y for each x_i such that $i \in S$ and 0 otherwise.
- C evaluates $\hat{C}$ to check whether the coefficient of the maximum degree of y is nonzero.

Important Observation

- Let f be a multilinear polynomial given by a circuit $\hat{C}$ and $m = \prod_{i \in S} x_i$ is a monomial.
- A small size boolean circuit C can decide whether m is a nonzero monomial in f.
- Just substitute y for each x_i such that $i \in S$ and 0 otherwise.
- C evaluates $\hat{C}$ to check whether the coefficient of the maximum degree of y is nonzero.

Important Observation

- Let f be a multilinear polynomial given by a circuit $\hat{C}$ and $m = \prod_{i \in S} x_i$ is a monomial.
- A small size boolean circuit C can decide whether m is a nonzero monomial in f.
- Just substitute y for each x_i such that $i \in S$ and 0 otherwise.
- C evaluates $\hat{C}$ to check whether the coefficient of the maximum degree of y is nonzero.

Important Observation

- Let f be a multilinear polynomial given by a circuit $\hat{C}$ and $m = \prod_{i \in S} x_i$ is a monomial.
- A small size boolean circuit C can decide whether m is a nonzero monomial in f.
- Just substitute y for each x_i such that $i \in S$ and 0 otherwise.
- C evaluates $\hat{C}$ to check whether the coefficient of the maximum degree of y is nonzero.

Consequence of Hypothesis 2

- Let $w_1, w_2, \cdots, w_t$ are the weight functions output by $\mathcal{A}_2$, $t \leq m^c$ where m is the size of the boolean circuit that defines the monomial of f.
- Let $w_i = (w_{i,1}, w_{i,2}, \cdots, w_{i,n})$.
- Goal is to fool every weight function w_i.
- For all i, write down the equation

$$g_i(y) = f(y^{w_{i,1}}, y^{w_{i,2}}, \cdots, y^{w_{i,n}}) = 0.$$

.

Consequence of Hypothesis 2

- Let $w_1, w_2, \cdots, w_t$ are the weight functions output by $\mathcal{A}_2$, $t \leq m^c$ where m is the size of the boolean circuit that defines the monomial of f.
- Let $w_i = (w_{i,1}, w_{i,2}, \cdots, w_{i,n})$.
- Goal is to fool every weight function w_i.
- For all i, write down the equation

$$g_i(y) = f(y^{w_{i,1}}, y^{w_{i,2}}, \cdots, y^{w_{i,n}}) = 0.$$

.

Consequence of Hypothesis 2

- Let $w_1, w_2, \cdots, w_t$ are the weight functions output by $\mathcal{A}_2$, $t \leq m^c$ where m is the size of the boolean circuit that defines the monomial of f.
- Let $w_i = (w_{i,1}, w_{i,2}, \cdots, w_{i,n})$.
- Goal is to fool every weight function w_i.
- For all i, write down the equation

$$g_i(y) = f(y^{w_{i,1}}, y^{w_{i,2}}, \cdots, y^{w_{i,n}}) = 0.$$

.

Consequence of Hypothesis 2

- Let $w_1, w_2, \cdots, w_t$ are the weight functions output by $\mathcal{A}_2$, $t \leq m^c$ where m is the size of the boolean circuit that defines the monomial of f.
- Let $w_i = (w_{i,1}, w_{i,2}, \cdots, w_{i,n})$.
- Goal is to fool every weight function w_i.
- For all i, write down the equation

$$g_i(y) = f(y^{w_{i,1}}, y^{w_{i,2}}, \cdots, y^{w_{i,n}}) = 0.$$

.

Consequence of Hypothesis 2

- The degree of $g_i(y)$ is $\leq 2n^2$.
- Total number of linear constraints for c_S's is at most $2n^2 m^c < 2^n$ for $m = 2^{o(n)}$.
- There always exists a nontrivial solution for f.

Consequence of Hypothesis 2

- The degree of $g_i(y)$ is $\leq 2n^2$.
- Total number of linear constraints for c_S's is at most $2n^2 m^c < 2^n$ for $m = 2^{o(n)}$.
- There always exists a nontrivial solution for f.

Consequence of Hypothesis 2

- The degree of $g_i(y)$ is $\leq 2n^2$.
- Total number of linear constraints for c_S's is at most $2n^2 m^c < 2^n$ for $m = 2^{o(n)}$.
- There always exists a nontrivial solution for f.

Finishing the proof

- Let f has a arithmetic circuit of size $2^{o(n)}$,
- Then a boolean circuit C of size $2^{o(n)}$ defines the monomials of f.
- Then for some weight function w_i there is a unique monomial $\prod_{j\in S} x_j$ such that $\sum_{j\in S} w_{i,j}$ takes the minimum value (by the property of $\mathcal{A}_2$).
- So the polynomial $g_i(y) \neq 0$, a contradiction.

Finishing the proof

- Let f has a arithmetic circuit of size $2^{o(n)}$,
- Then a boolean circuit C of size $2^{o(n)}$ defines the monomials of f.
- Then for some weight function w_i there is a unique monomial $\prod_{j \in S} x_j$ such that $\sum_{j \in S} w_{i,j}$ takes the minimum value (by the property of $\mathcal{A}_2$).
- So the polynomial $g_i(y) \neq 0$, a contradiction.

Finishing the proof

- Let f has a arithmetic circuit of size $2^{o(n)}$,
- Then a boolean circuit C of size $2^{o(n)}$ defines the monomials of f.
- Then for some weight function w_i there is a unique monomial $\prod_{j\in S} x_j$ such that $\sum_{j\in S} w_{i,j}$ takes the minimum value (by the property of $\mathcal{A}_2$).
- So the polynomial $g_i(y) \neq 0$, a contradiction.

Finishing the proof

- Let f has a arithmetic circuit of size $2^{o(n)}$,
- Then a boolean circuit C of size $2^{o(n)}$ defines the monomials of f.
- Then for some weight function w_i there is a unique monomial $\prod_{j \in S} x_j$ such that $\sum_{j \in S} w_{i,j}$ takes the minimum value (by the property of $\mathcal{A}_2$).
- So the polynomial $g_i(y) \neq 0$, a contradiction.

Other Result

- (Spielman and Klivans 2001) Randomized PIT for small degree (commutative) polynomial based on a more general formulation of isolation lemma.
- Observation Derandomization of the corresponding isolation lemma imply the result of Impagliazzo and Kabanets 2003.

Other Result

- (Spielman and Klivans 2001) Randomized PIT for small degree (commutative) polynomial based on a more general formulation of isolation lemma.
- Observation Derandomization of the corresponding isolation lemma imply the result of Impagliazzo and Kabanets 2003.

Summary

- We study the connections between derandomization of Isolation Lemma and circuit lower bounds.
- We formulate versions of Isolation Lemma based on set system defined by boolean circuits.
- A (non black-box) derandomization of above implies circuit lower bound in the *noncommutative* model.
- A black-box derandomization yields a circuit lower bound in usual *commutative model*.
- The derandomization of the Isolation Lemma used by Spielman-Klivans (2001) implies the result of Impagliazzo and Kabanets (2003).

Summary

- We study the connections between derandomization of Isolation Lemma and circuit lower bounds.
- We formulate versions of Isolation Lemma based on set system defined by boolean circuits.
- A (non black-box) derandomization of above implies circuit lower bound in the *noncommutative* model.
- A black-box derandomization yields a circuit lower bound in usual *commutative model*.
- The derandomization of the Isolation Lemma used by Spielman-Klivans (2001) implies the result of Impagliazzo and Kabanets (2003).

Summary

- We study the connections between derandomization of Isolation Lemma and circuit lower bounds.
- We formulate versions of Isolation Lemma based on set system defined by boolean circuits.
- A (non black-box) derandomization of above implies circuit lower bound in the *noncommutative* model.
- A black-box derandomization yields a circuit lower bound in usual *commutative model*.
- The derandomization of the Isolation Lemma used by Spielman-Klivans (2001) implies the result of Impagliazzo and Kabanets (2003).

Summary

- We study the connections between derandomization of Isolation Lemma and circuit lower bounds.
- We formulate versions of Isolation Lemma based on set system defined by boolean circuits.
- A (non black-box) derandomization of above implies circuit lower bound in the *noncommutative* model.
- A black-box derandomization yields a circuit lower bound in usual *commutative model*.
- The derandomization of the Isolation Lemma used by Spielman-Klivans (2001) implies the result of Impagliazzo and Kabanets (2003).

Summary

- We study the connections between derandomization of Isolation Lemma and circuit lower bounds.
- We formulate versions of Isolation Lemma based on set system defined by boolean circuits.
- A (non black-box) derandomization of above implies circuit lower bound in the *noncommutative* model.
- A black-box derandomization yields a circuit lower bound in usual *commutative model*.
- The derandomization of the Isolation Lemma used by Spielman-Klivans (2001) implies the result of Impagliazzo and Kabanets (2003).