NEXP *is not contained in* ACC^0

Nutan Limaye

Indian Institute of Technology, Bombay
nutan@cse.iitb.ac.in

An outline of the proof by Ryan Williams

May 6, 2011

Outline

- Introduction to complexity classes
- The statement of the main theorem
- History and importance
- Proof
- Future directions

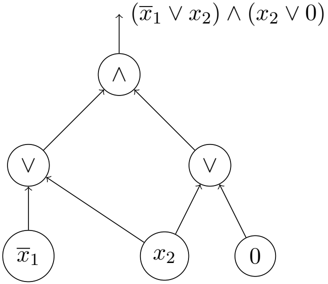
Outline

- Introduction to complexity classes
- The statement of the main theorem
- History and importance
- Proof
- Future directions

Complexity classes and lower bounds

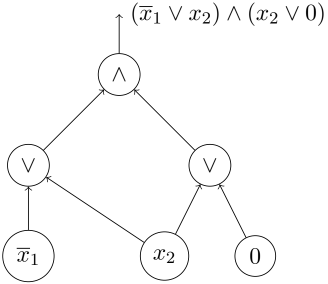
(model of computation, resource bound)	complexity class
(Det TM, poly time)	P
(Non-Det TM, poly time)	NP
(Non-Det TM, exp time)	NEXP
(Circuits, poly size)	P/poly

Circuits as a model of computation



- Set of variables $X = \{x_1, x_2, \ldots, x_n\}$.
- Directed acyclic graph (DAG) with labels from $X \cup \overline{X} \cup \{\wedge, \vee\} \cup \{0, 1\}$.
- Computes a function $f : \{0,1\}^n \to \{0,1\}$.

Circuits as a model of computation



Let $\mathscr{C} = \{C_n\}_{n=0}^{\infty}$.

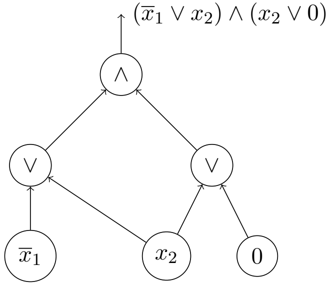
- Set of variables $X = \{x_1, x_2, \ldots, x_n\}$.
- Directed acyclic graph (DAG) with labels from $X \cup \overline{X} \cup \{\wedge, \vee\} \cup \{0, 1\}$.
- Computes a function $f : \{0,1\}^n \to \{0,1\}$.

Definition (Circuits computing a language)

$\mathscr{C}$ is said to compute language L if

$$\forall x : \quad x \in L \cap \{0,1\}^n \Leftrightarrow C_n(x) = 1$$

Circuits as a model of computation



- Set of variables $X = \{x_1, x_2, \ldots, x_n\}$.
- Directed acyclic graph (DAG) with labels from $X \cup \overline{X} \cup \{\wedge, \vee\} \cup \{0, 1\}$.
- Computes a function $f : \{0,1\}^n \to \{0,1\}$.

Let $\mathscr{C} = \{C_n\}_{n=0}^{\infty}$.

Definition (Circuits computing a language)

$\mathscr{C}$ is said to compute language L if

$$\forall x : \quad x \in L \cap \{0,1\}^n \Leftrightarrow C_n(x) = 1$$

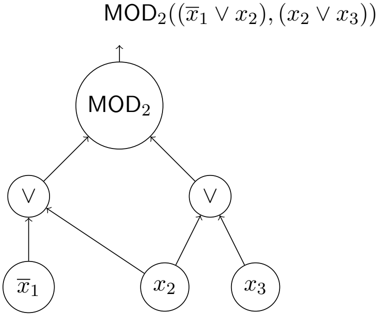
Allow for a different algorithm per input length.
Can compute even undecidable languages.

ACC^0 *and* MOD$_m$

ACC^0 Circuits:

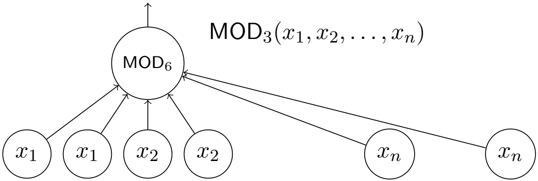
Constant depth circuits over AND, OR, NOT, MOD$_m$ for any $m > 1$.

- Set of variables $X = \{x_1, x_2, \ldots, x_n\}$.
- Directed acyclic graph (DAG) with labels from $X \cup \overline{X} \cup \{\wedge, \vee, \text{MOD}_m\} \cup \{0, 1\}$.
- Computes a function $f : \{0,1\}^n \to \{0,1\}$.



ACC^0 *and* MOD_m

MOD_6 computing MOD_3



Outline

- Introduction to complexity classes
- The statement of the main theorem
- History and importance
- Proof
- Future directions

NEXP *and* ACC^0

Theorem (Williams, 2010)

There exists a language in NEXP *that has no polynomial sized* ACC^0 *circuits.*

Outline

- Introduction to complexity classes
- The statement of the main theorem
- History and importance
- Proof
- Future directions

History: Lower bounds

Circuits allow a different algorithm for every input length.

History: Lower bounds

Circuits allow a different algorithm for every input length.

Are there polynomial sized circuits for NP?

History: Lower bounds

Circuits allow a different algorithm for every input length.

Are there polynomial sized circuits for NP?

Open

History: Lower bounds

Circuits allow a different algorithm for every input length.

Are there polynomial sized circuits for NP? Open

Are there polynomial sized constant depth circuits for NP?

History: Lower bounds

Circuits allow a different algorithm for every input length.

Are there polynomial sized circuits for NP? Open

Are there polynomial sized constant depth circuits for NP?

No not even for P

Parity does not have constant depth polynomial sized circuits.
[Frust-Saxe-Sipser, 1981], [Ajtai, 1983], [Yao, 1985], [Håstad, 1986]

History: Lower bounds

Circuits allow a different algorithm for every input length.

Are there polynomial sized circuits for NP? Open

Are there polynomial sized constant depth circuits for NP? No

Are there poly sized $O(1)$ depth circuits with MOD_2 gates for NP?

History: Lower bounds

Circuits allow a different algorithm for every input length.

Are there polynomial sized circuits for NP? Open

Are there polynomial sized constant depth circuits for NP? No

Are there poly sized $O(1)$ depth circuits with MOD_2 gates for NP?

No not even for P

MOD_3 cannot be computed by constant depth polynomial sized circuits with MOD_2 gates. [Razborov, 1987], [Smolensky, 1987]

History: Lower bounds

Circuits allow a different algorithm for every input length.

Are there polynomial sized circuits for NP? Open

Are there polynomial sized constant depth circuits for NP? No

Are there poly sized $O(1)$ depth circuits with MOD_2 gates for NP? No

Are there poly sized $O(1)$ depth circuits with MOD_6 for NP?

History: Lower bounds

Circuits allow a different algorithm for every input length.

Are there polynomial sized circuits for NP? Open

Are there polynomial sized constant depth circuits for NP? No

Are there poly sized $O(1)$ depth circuits with MOD_2 gates for NP? No

Are there poly sized $O(1)$ depth circuits with MOD_6 for NP?

Open

History: Lower bounds

Circuits allow a different algorithm for every input length.

Are there polynomial sized circuits for NP? Open

Are there polynomial sized constant depth circuits for NP? No

Are there poly sized $O(1)$ depth circuits with MOD_2 gates for NP? No

Are there poly sized $O(1)$ depth circuits with MOD_6 for NP? Open

Are there poly sized $O(1)$ depth circuits with MOD_6 for EXP?

History: Lower bounds

Circuits allow a different algorithm for every input length.

Are there polynomial sized circuits for NP? Open

Are there polynomial sized constant depth circuits for NP? No

Are there poly sized $O(1)$ depth circuits with MOD_2 gates for NP? No

Are there poly sized $O(1)$ depth circuits with MOD_6 for NP? Open

Are there poly sized $O(1)$ depth circuits with MOD_6 for EXP?

Open .. ?

History: Lower bounds

Circuits allow a different algorithm for every input length.

Are there polynomial sized circuits for NP? Open

Are there polynomial sized constant depth circuits for NP? No

Are there poly sized $O(1)$ depth circuits with MOD_2 gates for NP? No

Are there poly sized $O(1)$ depth circuits with MOD_6 for NP? Open

Are there poly sized $O(1)$ depth circuits with MOD_6 for EXP? Open

Are there poly sized $O(1)$ depth circuits with MOD_6 for NEXP?

History: Lower bounds

Circuits allow a different algorithm for every input length.

Are there polynomial sized circuits for NP? Open

Are there polynomial sized constant depth circuits for NP? No

Are there poly sized $O(1)$ depth circuits with MOD_2 gates for NP? No

Are there poly sized $O(1)$ depth circuits with MOD_6 for NP? Open

Are there poly sized $O(1)$ depth circuits with MOD_6 for EXP? Open

Are there poly sized $O(1)$ depth circuits with MOD_6 for NEXP?

Hmm ...

History: Lower bounds

Circuits allow a different algorithm for every input length.

Are there polynomial sized circuits for NP? Open

Are there polynomial sized constant depth circuits for NP? No

Are there poly sized $O(1)$ depth circuits with MOD_2 gates for NP? No

Are there poly sized $O(1)$ depth circuits with MOD_6 for NP? Open

Are there poly sized $O(1)$ depth circuits with MOD_6 for EXP? Open

Are there poly sized $O(1)$ depth circuits with MOD_6 for NEXP?

This is what we will study today.

Outline

- Introduction to complexity classes
- The statement of the main theorem
- History and importance
- Proof
- Future directions

Proof

Preliminaries for the proof:

CIRCUIT-SAT:

Given:	a circuit C on n variables
Check:	does there exist an assignment for the variables that makes the circuit evaluate to 1.

Proof

Preliminaries for the proof:

Circuit-Sat:

Given:	a circuit C on n variables
Check:	does there exist an assignment for the variables that makes the circuit evaluate to 1.

Succinct-3SAT:

Given:	a formula ϕ on 2^n variables and 2^n clauses encoded by a circuit of size $poly(n)$
Check:	is ϕ satisfiable?

Proof

Preliminaries for the proof:

CIRCUIT-SAT:

Given:	a circuit C on n variables
Check:	does there exist an assignment for the variables that makes the circuit evaluate to 1.

SUCCINCT-3SAT:

Given:	a formula ϕ on 2^n variables and 2^n clauses encoded by a circuit of size $poly(n)$
Check:	is ϕ satisfiable?

Theorem (Nondeterministic time heirarchy theorem)

$\forall k > 0$ *and* $\exists f : \{0,1\}^* \rightarrow \{0,1\}$, *such that* $f \in \mathit{NTIME}[2^n]$ *but* $f \notin \mathit{NTIME}[2^n/n^k]$.

Proof

Proof Outline:

Let $L \in \mathsf{NTIME}[2^n]$.

1. Given $x \in \{0,1\}^n$. Reduce to an instance of $\textsc{Succinct-3SAT}$ C_x
2. From C_x obtain a circuit D such that D is unsatisfiable if and only if $x \in L$.
3. Prove D is ACC^0, poly sized.
4. Give a fast algorithm for ACC^0-$\textsc{Circuit-Sat}$, running in time $O(2^n/n^k)$.

Proof

Proof Outline:

Let $L \in \mathsf{NTIME}[2^n]$.

1. Given $x \in \{0,1\}^n$. Reduce to an instance of SUCCINCT-3SAT C_x
2. From C_x obtain a circuit D such that D is unsatisfiable if and only if $x \in L$.
3. Prove D is ACC^0, poly sized.
4. Give a fast algorithm for ACC^0-CIRCUIT-SAT, running in time $O(2^n/n^k)$.

Theorem (Nondeterministic time heirarchy theorem)

$\forall k > 0$ and $\exists f : \{0,1\}^* \to \{0,1\}$, such that $f \in \mathsf{NTIME}[2^n]$ but $f \notin \mathsf{NTIME}[2^n/n^k]$.

Proof

Proof Outline:

Let $L \in \mathsf{NTIME}[2^n]$.

1. Given $x \in \{0,1\}^n$. Reduce to an instance of SUCCINCT-3SAT C_x
2. From C_x obtain a circuit D such that D is unsatisfiable if and only if $x \in L$. *
3. Prove D is ACC^0, poly sized. *
4. Give a fast algorithm for ACC^0-CIRCUIT-SAT, running in time $O(2^n/n^k)$.

* Assuming NEXP in ACC^0.

Step 1: Reduction to Succinct-3SAT

Step 1: Reduction to SUCCINCT-3SAT

SUCCINCT-3SAT:

Given:	a formula ϕ on 2^n variables and 2^n clauses using a circuit of size $poly(n)$
Check:	is ϕ satisfiable?

Step 1: Reduction to SUCCINCT-3SAT

SUCCINCT-3SAT:

Given:	a formula ϕ on 2^n variables and 2^n clauses using a circuit of size $poly(n)$
Check:	is ϕ satisfiable?

Deterministic polynomial time reduction from any language L in NTIME$[2^n]$ to SUCCINCT-3SAT:

[Tourlakis, 2001], [Fortnow-Lipton-van Melkebeek-Viglas, 2005]

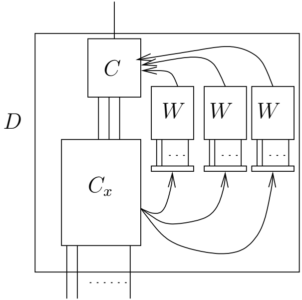
x	$\longrightarrow$	C_x
$\lvert x\rvert = n$	$\longrightarrow$	size of C_x $O(n^5)$ $n + 5\log n$ inputs
$x \in L$	$\Leftrightarrow$	C_x is satisfiable

Proof Outline

Let $L \in \mathsf{NTIME}[2^n]$.

1. Given $x \in \{0,1\}^n$. Reduce to an instance of $\textsc{Succinct-3SAT}$ C_x
2. From C_x obtain a circuit D such that D is unsatisfiable if and only if $x \in L$. (Assuming NEXP in ACC^0.)
3. Prove D is ACC^0, poly sized. (Assuming NEXP in ACC^0.)
4. Give a fast algorithm for ACC^0-$\textsc{Circuit-Sat}$, running in $\mathsf{NTIME}[2^n/n^k]$.

SUCCINCT-3SAT *to* CIRCUIT-SAT



C_x

Input:	index i of a clause
Output:	variables x_{i1}, x_{i2}, x_{i3} in clause i with signs

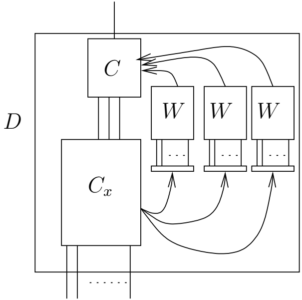
W

Input:	a variable index i
Output:	value of x_i in the lex first satisfying assignment

D

Input:	index i of a clause
Output:	1 iff clause i **not** satisfied by assignment given by W

Succinct-3SAT *to* Circuit-Sat



C_x

Input: index i of a clause

Output: variables x_{i1}, x_{i2}, x_{i3} in clause i with signs

W

Input: a variable index i

Output: value of x_i in the lex first satisfying assignment

D

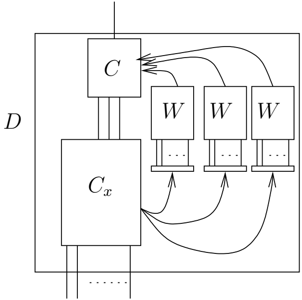
Input: index i of a clause

Output: 1 iff clause i **not** satisfied by assignment given by W

If formula encoded by C_x is satisfiable then D is not satisfiable.

If formula encoded by C_x is not satisfiable then D is satisfiable.

SUCCINCT-3SAT *to* CIRCUIT-SAT



C_x

Input: index i of a clause

Output: variables x_{i1}, x_{i2}, x_{i3} in clause i with signs

W

Input: a variable index i

Output: value of x_i in the lex first satisfying assignment

D

Input: index i of a clause

Output: 1 iff clause i **not** satisfied by assignment given by W

Assume NEXP has polysized circuits.

The satisfiable instance of SUCCINCT-3SAT have polynomial size circuits encoding the satisfying assignments.
[Impagliazzo-Kabanets-Wigderson, 2002]

Proof Outline

Let $L \in \mathsf{NTIME}[2^n]$.

1. Given $x \in \{0,1\}^n$. Reduce to an instance of $\textsc{Succinct-3SAT}$ C_x
2. From C_x obtain a circuit D such that D is unsatisfiable if and only if $x \in L$. (Assuming NEXP in ACC^0.)
3. Prove D is ACC^0, poly sized. (Assuming NEXP in ACC^0.)
4. Give a fast algorithm for ACC^0-$\textsc{Circuit-Sat}$, running in $\mathsf{TIME}[2^n/n^k]$.

Making D ACC^0*?*

Lemma

For an instance C_x of SUCCINCT-3SAT *there is an equivalent* ACC^0 *circuit C such that for all $y \in \{0,1\}^{n+5\log n}$ $C_x(y) = C(y)$.*

Proof: As NEXP is contained in ACC^0, P is contained in ACC^0.

Let A be an ACC^0 circuit for the Circuit Value Problem.

Feed C_x as an input to A.

$A(C_x, y)$ evaluates C_x on y and is an ACC^0 circuit.

Proof Outline

Let $L \in \mathsf{NTIME}[2^n]$.

1. Given $x \in \{0,1\}^n$. Reduce to an instance of $\textsc{Succinct-3SAT}$ C_x
2. From C_x obtain a circuit D such that D is unsatisfiable if and only if $x \in L$. (Assuming NEXP in ACC^0.)
3. Prove D is ACC^0, poly sized. (Assuming NEXP in ACC^0.)
4. Give a fast algorithm for ACC^0-$\textsc{Circuit-Sat}$, running in $\mathsf{NTIME}[2^n/n^k]$.

Design fast algorithm for ACC^0-Circuit-Sat

Outline:

- Any ACC^0 circuit can be converted into a SYM^+ circuit.
- There is a fast dynamic programming algorithm for circuit satisfiability of SYM^+ circuits.

Design fast algorithm for ACC^0-Circuit-Sat

SYM^+:

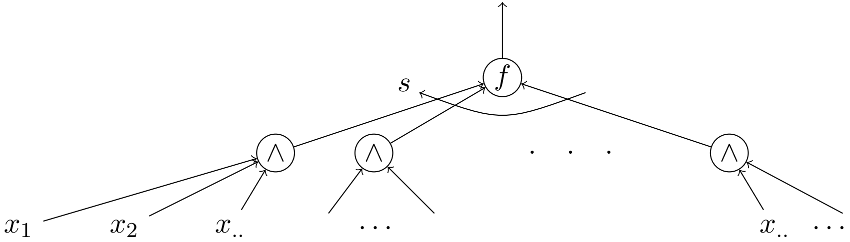
Depth 2 circuit which is a symmtric function * of ANDs of the input variables.

* Symmetric function: output depends only on the number of 1s in the input.

Design fast algorithm for ACC^0-Circuit-Sat

SYM$^+$:

Depth 2 circuit which is a symmtric function of ANDs of the input variables.



ACC^0 *to* SYM^+

Theorem ([Yao, 1990],[Beigel-Tarui, 1994],[Allender-Gore, 1994])

Any ACC^0 *circuit of size* n^5 *can be converted into a* SYM^+ *circuit of size* $n^{O(\log^d n)}$ *in time* $n^{O(\log^d n)}$. *And the symmetric function can be computed in time* $n^{O(\log^d n)}$. *(d: a constant depending on the depth)*

Fast evaluation of SYM$^+$

Let $S \subseteq [n]$. For a SYM$^+$ circuit:

Let $h : 2^{[n]} \longrightarrow \mathbb{N}$ be defined as:

$h(S) = j$ if j-many AND gates have S feeding into them.

Fast evaluation of SYM$^+$

Let $S \subseteq [n]$. For a SYM$^+$ circuit:

Let $h : 2^{[n]} \to \mathbb{N}$ be defined as:

$h(S) = j$ if j-many AND gates have S feeding into them.

Let $g : 2^{[n]} \to \mathbb{N}$ be defined as: $g(T) = \sum_{S \subseteq T} h(S)$.

$g(T) = \#$ of true AND gates under
$x_i = 1$ for $i \in T$ and $x_i = 0$ for $i \notin T$

Fast evaluation of SYM$^+$

Let $S \subseteq [n]$. For a SYM$^+$ circuit:

Let $h : 2^{[n]} \rightarrow \mathbb{N}$ be defined as:

$h(S) = j$ if j-many AND gates have S feeding into them.

Let $g : 2^{[n]} \rightarrow \mathbb{N}$ be defined as: $g(T) = \sum_{S \subseteq T} h(S)$.

$g(T) = \#$ of true AND gates under
$x_i = 1$ for $i \in T$ and $x_i = 0$ for $i \notin T$

Computing SYM$^+$ circuit on all its inputs

equivalent to

Computing g for all $T \subseteq [n]$

Computing g *on all* $T \subseteq [n]$

- Computing h

  Takes time $O(2^n + s\ poly(n))$, where s is the size of the circuit.

Computing g on all $T \subseteq [n]$

- Computing h

  Takes time $O(2^n + s\ poly(n))$, where s is the size of the circuit.

- Computing g: Dynamic Programming
  Set $g_0(T) = h(T)$ for all $T \subseteq [n]$

$$g_{i+1}(T) = \begin{cases} g_i(T) + g_i(T \setminus \{i\}) & \text{if } i \in T \\ g_i(T) & \text{otherwise} \end{cases}$$

Computing g on all $T \subseteq [n]$

- Computing h

  Takes time $O(2^n + s\ poly(n))$, where s is the size of the circuit.

- Computing g: Dynamic Programming
  Set $g_0(T) = h(T)$ for all $T \subseteq [n]$

$$g_{i+1}(T) = \begin{cases} g_i(T) + g_i(T \setminus \{i\}) & \text{if } i \in T \\ g_i(T) & \text{otherwise} \end{cases}$$

$g_i(T)$ equals $\sum_{S \subseteq T} h(S)$, for S such that $S \cap [n] \setminus [i] = T \cap [n] \setminus [i]$

Note: $g_n = g$.

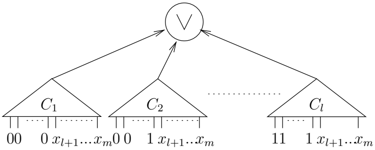
The function g_{i+1} can be computed from g_i in time $O(2^n poly(n))$
Therefore the total time $= O(2^n poly(n) + s\ poly(n))$.

Fast ACC^0-CIRCUIT-SAT *algorithm*

Theorem

CIRCUIT-SAT *for any* ACC^0 *circuit* C *with* $n + 5\log n$ *inputs and* n^5 *size can be determined in time* $O(2^{n-\log^2 n} poly(n))$.

Proof: Obtain C' from C:



C':

size: $2^l n^5$,

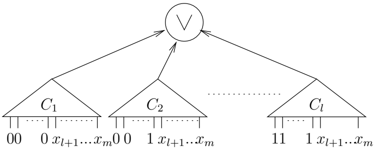
inputs: $m - l$, where $m = n + 5\log n$.

Fast ACC^0-Circuit-Sat *algorithm*

Theorem

Circuit-Sat for any ACC^0 circuit C with $n + 5\log n$ inputs and n^5 size can be determined in time $O(2^{n-\log^2 n} poly(n))$.

Proof: Obtain C' from C:



C':

size: $2^l n^5$,

inputs: $m - l$, where $m = n + 5\log n$.

Recall:

Theorem ([Yao, 1990],[Beigel-Tarui, 1994],[Allender-Gore, 1994])

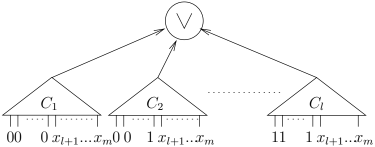
Any ACC^0 circuit of size s can be converted into a SYM$^+$ circuit of size $s^{O(\log^d s)}$ in time $s^{O(\log^d s)}$. Assuming the symmetric function can be computed in time $s^{O(\log^d s)}$. (d: a constant depending on the depth)

Fast ACC^0-Circuit-Sat *algorithm*

Theorem

Circuit-Sat for any ACC^0 circuit C with $n + 5\log n$ inputs and n^5 size can be determined in time $O(2^{n-\log^2 n} poly(n))$.

Proof: Obtain C' from C:



C':

size:	$2^l n^5$,
inputs:	$m - l$, where
	$m = n + 5\log n$.

SYM$^+$ circuit C'':

size:	$(2^l n^5)^{O(l \log^d n)}$
inputs:	$m - l$

Fast ACC^0-Circuit-Sat *algorithm*

Theorem

Circuit-Sat for any ACC^0 circuit C with $n + 5\log n$ inputs and n^5 size can be determined in time $O(2^{n-\log^2 n} poly(n))$.

Proof:

SYM$^+$ circuit C'':

size: $(2^l n^5)^{O(l \log^d n)}$

inputs: $m - l$

Fast satisfiability algorithm:

To evaluate AND gates: $O(2^{m-l} poly(n))$

To evaluate symmetric function: $(2^l n^5)^{O(l \log^d n)}$

Fast ACC^0-CIRCUIT-SAT *algorithm*

Theorem

CIRCUIT-SAT *for any* ACC^0 *circuit* C *with* $n + 5\log n$ *inputs and* n^5 *size can be determined in time* $O(2^{n-\log^2 n} poly(n))$.

Proof:

SYM$^+$ circuit C'':

size: $(2^l n^5)^{O(l \log^d n)}$

inputs: $m - l$

Fast satisfiability algorithm:

For $l = \log^2 n$

To evaluate AND gates: $O(2^{m-l} poly(n)) = O(2^{n-\log^2 n} poly(n))$

To evaluate symmetric function: $(2^l n^5)^{O(l \log^d n)} = n^{O(\log^{d'} n)}$

Fast ACC^0-CIRCUIT-SAT *algorithm*

Theorem

CIRCUIT-SAT for any ACC^0 circuit C with $n + 5\log n$ inputs and n^5 size can be determined in time $O(2^{n-\log^2 n} poly(n))$.

Proof:

SYM$^+$ circuit C'':

size: $(2^l n^5)^{O(l \log^d n)}$

inputs: $m - l$

Fast satisfiability algorithm:

For $l = \log^2 n$

To evaluate AND gates: $O(2^{m-l} poly(n)) = O(2^{n-\log^2 n} poly(n))$

To evaluate symmetric function: $(2^l n^5)^{O(l \log^d n)} = n^{O(\log^{d'} n)}$

□

Outline

- Introduction to complexity classes
- The statement of the main theorem
- History and importance
- Proof
- Future directions

Thank you!