Algorithms and Lower Bounds for Threshold Circuits

Atsushi Saito
Division of Electronics and Computing, Gunma University

February 13, 2015

Abstract

A fundamental purpose of theory of computation is to understand differences between uniform computation and nonuniform one. In particular, Boolean circuit has been studied in an area of nonuniform computation models, because Boolean circuits are natural formalization of computer architecture and hardware. Boolean circuit is compared with uniform computation expressed as fixed size programs which run for an arbitrary input length. In the computational complexity theory, cost of non-uniform computation is measured through infinite family of Boolean circuits. Proving computational limitations of Boolean circuits is an extremely important and challenging task in the theoretical computer science.

A remarkable recent result about satisfiability algorithms is a nontrivial algorithm for testing satisfiability of depth two sparse threshold circuits which have linear number of wires by Impagliazzo et. al. In this thesis, we construct a nontrivial algorithm for a larger class of circuits. We give a nontrivial circuit satisfiability algorithm for a class of circuits which may not be sparse in gates with dependency. Two gates in a circuit are dependent, if the output of the one is always greater than or equal to the other one. An independent gate set is a set of gates in which two arbitrary gates are not dependent. In our setting, the number of restrictions to bottom level gates is bounded above because of dependency of bottom gates. We first define some partial order on the set of bottom gates. Next, we define a problem: for given a pair of a circuit and a Hasse diagram relating with the circuit, output YES if and only if the circuit is satisfiable. Because of an upper bound on the expected number of restrictions to bottom level gates, the running time of the randomized algorithm is faster than the complexity of the trivial exhaustive search.

Recently, Williams proved a separation between NEXP and $\mathsf{ACC} \circ \mathsf{THR}$, where an $\mathsf{ACC} \circ \mathsf{THR}$ circuit has a single layer of threshold gates at the bottom and an ACC circuit at the top. Two main ideas of his strategy are a closure property of circuit class and an algorithm for counting satisfying assignments of circuits. In this thesis, we show that this general scheme based on these two ideas can be applied for a certain class of circuits with multilayer of threshold gates. The circuit class we give has the symmetric gate at the top and poly-log layers of threshold gates to which an extra condition on the dependency is imposed. We show that, if the size of a maximum independent gate set of each layer of threshold gates is at most n^γ for sufficiently small $\gamma > 0$, then two key ingredients needed to apply his strategy can be established. We also give a result about lower bounds against NEXP, extending the results by Williams.

Contents

Acknowledgements

First and foremost, I would like to thank my advisor Kazuyuki Amano for helpful discussions, suggestions, patience, and support. Further, I would like to thank Toru Araki, Ken-etsu Fujita, Shinichi Nakano, Koichi Yamazaki for many helpful comments to the preliminary version of this thesis and discussions.

Chapter 1

Introduction

1.1 Satisfiability

Satisfiability problem gives both an integral view on theory of NP-complete problems which is firstly defined in [13] and [31], and one of the most useful methods for constraint satisfaction problems in engineering and other practical fields. In particular, heuristic ways on CNF SAT are applied in the practical area of various combinatorial search problems such as boolean circuit design verification.

There are several well known computational problems related to satisfiability problems. The first one is satisfiability for CNF formulas and its generalization, because CNF is one of fundamental concepts about boolean functions. For example, Santhanam [38] gives an algorithm with a nontrivial exponent for linear size formulas of AND and OR gates with fan-in two. The second one is MAX-k-SAT, the optimization version of k-CNF SAT. Even for MAX-3-SAT, no algorithms with constant savings over brute force search are known while such an algorithm is constructed for MAX-2-SAT in [43]. The third one is Integer Linear Programming (ILP) that is very useful in expressing combinatorial optimization problems both in theory and practice.

Satisfiability for depth two threshold circuits contains these problems as special cases. Satisfiability for CNF formula can be solved by algorithms solving satisfiability for depth two threshold circuits. We should note that we do not obtain a nontrivial algorithm for depth two threshold circuit satisfiability algorithm as a corollary of the result in [38]. The reason is that known transformation from a linear size threshold circuit to formula over AND and OR gates yields a quadratic blow-up of size. MAX-k-SAT, the optimization version of k-CNF SAT, can be computed by algorithms solving satisfiability for depth two threshold circuits, since we can regard the top threshold gate as a counting device of the number of satisfied CNFs and an objective function in an optimization problem. Finally, testing the feasibility for a 0-1 ILP is equivalent to testing the satisfiability of a circuit with two levels: the bottom consisting of threshold gates and the top level being an AND gate. So understanding satisfiability of depth two threshold circuits could give us various view points on both theoretical and practical areas including the above three problems.

In the paper [25], Impagliazzo et al. constructed the first nontrivial algorithm with constant savings in the exponent over brute force search for the satisfiability of *sparse* depth two threshold circuits which has cn-wires for every constant c. As a consequence, they also got a similar result for linear-size ILP. Here we say an algorithm is nontrivial, if its running time is bounded above by $2^n/w(n)$ where n is the number of input variables and $w(n)$ is a super-polynomial function in n. Note that 2^n is just the number of assignments to n input variables. Their main subroutine is an algorithm for the Vector Domination Problem: given n vectors in $\mathbb{R}^d$, decide whether there is a pair of vectors such that the first vector is larger than the second vector in each coordinate. Relationship between this problem and satisfiability problem is studied in [43].

The Strong Exponential Time Hypothesis (SETH) is a well known conjecture about limitations of efficiency of satisfiability algorithms. The statement of SETH is that for every $\delta < 1$ there is a k such that k-SAT cannot be solved in time $O(2^{\delta n})$. In particular, an algorithm with constant savings for depth two threshold circuits of super linear size would violate SETH [22], since k-CNF for all k can be reduced through Sparsification Lemma [23] to superlinear size depth two threshold circuits [9]. Some algorithms solving CNF-SAT and MAX-SAT with constant savings for linear size formula are given in [38] and [15]. Assuming the SETH, we can not to solve satisfiability problem for super linear size depth two threshold circuits with constant savings as a direct extension of the result in [25].

Thus one of natural directions relating with this result is extending classes of input circuits and constructing an algorithm with constant savings under the SETH for such classes. Considering algorithms for a class of circuits of polynomial size is also crucial to circuit complexity theory. For all above reasons, it is significant to give algorithms for an explicit class which is a subclass of depth two threshold circuits of super linear size.

In Chapter 3, we give all results and proofs on the research in [3].

1.2 Boolean Circuit Lower Bounds

Boolean circuit is one of the most popular and natural computation models. For example, proving the existence of some NP problem having super polynomial size circuits led us to $\mathsf{P} \neq \mathsf{NP}$. The best general boolean circuit lower bounds for NP problems are, however, $5n - o(n)$ by Iwama and Morizumi [27].

Various restricted circuit classes are studied. Bounded depth circuit class is one of the most successful restricted classes with a lot of remarkable results [16, 19, 36, 41]. Williams established a landmark in the circuit complexity theory with the separation between NEXP and ACC^0 [45]. He incorporated many known results [6, 14, 20] into a perspective between algorithms and lower bounds [44]. The class TC^0, which is a class of constant depth polynomial size threshold circuits, is a well known natural circuit class larger than ACC^0. Current understanding of bounded depth threshold circuits is extremely inadequate [18, 24].

Recently, Williams [46] proved a separation between NEXP and $\mathsf{ACC} \circ \mathsf{THR}$, where an $\mathsf{ACC} \circ \mathsf{THR}$ circuit has single layer of threshold gates at the bottom and an ACC circuit at the top. Two main ideas of his strategy are a closure property of circuit class and an algorithm for counting satisfying assignments of circuits. Thus it is a plausible direction to consider the usefulness of the framework based on these ideas.

In this thesis, we show that this general framework based on these two ideas can be applied for some restricted class of circuits with *multi layer* of threshold gates. The circuit class we give has the symmetric gate at the top and at most poly-log layers of threshold gates to which an extra condition on the *dependency* is imposed. Two gates in a circuit are dependent, if the output of the one is always greater than or equal to the output of the other one. An independent gate set is a set of gates in which two arbitrary gates are *not* dependent. Each layer of threshold gates in our class has independent gate sets of size at most n^γ for sufficiently small $\gamma > 0$. We show that two main ideas in [46] are workable for our circuit class. It is notable that our circuit class is universal even if there is no two independent gates and that the general framework can be applied for poly-log depth circuits. First, we show that we can efficiently find a circuit in our class being equivalent to the AND of two input circuits in our class. Thus our class has a closure property (Lemma 4.13). Second, we design an algorithm for counting satisfying assignments for our circuit class (Lemma 4.14). We connect dependency to a structure of a partial order on the gate set. This connection make counting assignments easier than general settings. By pluging them into William's schema (Theorem 4.6), we obtain super quasi-polynomial size lower bounds for our circuit class against NEXP (Theorem 4.12).

In Chapter 4, we give all results and proofs on the research in [4]. We also give a result lower bounds against NEXP, extending results in [46].

Chapter 2

Boolean Circuits and Relationship between Satisfiability and Lower Bounds

In this chapter, we give a survey on boolean circuits and relationship between satisfiability and lower bounds. In particular, we give a self contained proof of a remarkable general result on relationship between improvements of circuit satisfiability algorithms and circuit lower bounds in [44].

Williams established a landmark in the circuit complexity theory with the separation between NEXP and ACC^0 [45]. He integrated a perspective between algorithms and lower bounds [44] and many known results [6, 14, 20]. These known results are: conditional results about the existence of small size boolean circuits regarded as compression of NEXP witness which is proved by Impagliazzo, Kabanets, and Wigderson [20], a procedure transforming ACC^0 circuits to $\mathsf{SYM} \circ \mathsf{AND}$ circuits with quasi polynomial overheads designed by Beigel and Tarui [6], and a fast matrix multiplication algorithm constructed by Coppersmith [14], and the nondeterministic time hierarchy theorem. His perspective is as follows: constructing a faster meta algorithm for a restricted circuit class is useful to prove the limitation of power of the restricted circuits. Note that we regard meta algorithms as algorithms running on algorithms or circuits. This perspective relies on the following ideas: **(1)** The meta algorithm in this perspective essentially runs on an *arbitrary algorithm* having computational power such as NEXP or E^{NP}, **(2)** The computational process of such an extremely powerful algorithm is expressed as a family of the restricted small size circuits, and **(3)** The meta algorithm simulates an arbitrary powerful algorithm so fast that we can derive a contradiction to the time hierarchy theorem. We assume that the reader has knowledge about several basic definitions of Turing machine and complexity classes like P, NP, and NEXP. (see eg: [5])

A Boolean circuit with n-inputs is a directed acyclic graph with n *sources* and one *sink*. All non-source vertices are called gates and have labels which is one of $\{\vee, \wedge, \neg\}$, that is, disjunction (OR), conjunction (AND), and negation gates. The in-degree of all negation gates is one.

The *depth* of the circuit C is the number of edges in the longest path between the sink and a source. The *fan-in* is the maximum in-degree of the graph. The *fan-out* is the maximum out-degree of the gates in the graph. The size of a circuit C is able to be defined in two ways: the number of gates in the graph, and the number of wires in it. For an evaluation of a circuit on an input $x = (x_1, ..., x_n) \in \{0, 1\}^n$, for any vertex of the circuit, we compute the output value of the gate as follows. If the vertex is the i-th source , then its value is the i-th bit of the input (i.e. x_i). Otherwise the value is defined recursively by evaluating the logical operation of the vertex on the values of the vertices connected to the gate. The output of the circuit is the value of the sink.

A Turing machine deals with inputs of every length. By contrast, Boolean circuits can only get inputs of a fixed length, that is, a circuit computing inputs of certain length cannot be used for computing inputs of different lengths. This conception is natural in practical sense, because circuits are natural formalization of computer hardware and algorithms are the one of computer programs. Thus, the computational model of circuits is defined as a family of circuits $\mathcal{C} = \{C_n\}_{n\in\mathbb{N}}$, where the circuit C_n has n inputs. This kind of computational model is called *nonuniform*, since it allows a different treatment for inputs of varying length, or infinite number of algorithms, if we wish. The nonuniform computation models can have a strong power. Indeed, it can even decide undecidable languages like Halting problem. Note that we can construct a circuit for each input length. In the case of unary languages, which has only one input of each length, we can consider a circuit which outputs the right answer for each input.

Definition 2.1. Let $s : \mathbb{N} \to \mathbb{N}$ be a function. The complexity class $\mathsf{SIZE}[s(n)]$ is the class of languages such that there is some family of boolean circuits $\mathcal{C} = \{C_n\}_{n\in\mathbb{N}}$ deciding L and the size of C_n is at most $s(n)$ for all n.

Definition 2.2. The class $\mathsf{P/poly}$ is defined as the class of languages decided by families of circuits of polynomial size, namely,

$$\mathsf{P/poly} = \bigcup_{c\geq 1} \mathsf{SIZE}[n^c]$$

In this chapter, we give the proof of the following result.

Theorem 2.3 ([44])**.** Suppose there is a super polynomial function $s(n)$ such that CIRCUIT SAT on circuits with n variables and n^k gates can be solve solve in $2^n \cdot \mathsf{poly}(n^k)/s(n)$ time by a (co-non)deterministic algorithm , for all k. Then $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$.

We introduce the following hierarchy theorem in the area of nonuniform circuits.

Theorem 2.4. For arbitrary function $s(n)$ such that $n \leq s(n) \leq 2^n/4n$, the following holds.

$$\mathsf{SIZE}[s(n)] \subsetneq \mathsf{SIZE}[4s(n)]$$

Another important theorem on polynomial size Boolean circuits is the following one.

Theorem 2.5 (eg: [8])**.**

$$\mathsf{DTIME}[T(n)] \subseteq \mathsf{SIZE}[T(n)\log T(n)]$$

2.1 Polynomial Hierarchy and Meyer's Theorem

The polynomial Hierarchy, denoted by PH is introduced by Meyer and Stockmeyer. This is a kind of generalization of the classes P, NP, and coNP.

Definition 2.6. Let $\Sigma_0^p = \mathsf{P}$, and let $\Sigma_1^p = \mathsf{NP}$. Then, for any integer $i \leq 0$, let $\Sigma_{i+1}^p = \mathsf{NP}^{\Sigma_i^p}$. The polynomial hierarchy is defined by $\mathsf{PH} = \bigcup_{i \geq 1} \Sigma_i^p$

Let Π_1^p be coNP, and for any integer $i \leq 0$, let $\Pi_{i+1}^p = \mathsf{coNP}^{\Pi_i^p}$. It is not hard to prove that $\mathsf{PH} = \bigcup_{i \geq 1} \Pi_i^p$. Thus PH is generalized from NP as well as coNP.

The following theorem is one of the most basic results about collapsing the polynomial hierarchy.

Theorem 2.7 ([29]).

$$\mathsf{EXP} \subseteq \mathsf{P/poly} \Rightarrow \mathsf{EXP} = \Sigma_2^p$$

Proof. Suppose $\mathsf{EXP} \subseteq \mathsf{P/poly}$. Let M be a single tape Turing machine deciding L in time 2^{n^k} for some constant k. We consider the computational configuration for M and at the (i, t) position of the tableau, the string $z_{i,t}$ is written, which encodes the content of the i-th cell at time t and the internal state of the TM's head in the i-th cell or string corresponding to the absence of head at the i-th cell.

We define the following language associating with the computational tableau for M.

$$L_M = \{\langle x, i, t, z \rangle : \text{running on input } x, \text{ we have } z_{i,t} \text{ for } M\}$$

By a simulation for M, $L_M \in \mathsf{EXP} \subseteq \mathsf{P/poly}$. Thus, using polynomial size circuits for L_M, we can obtain an polynomial size multi output circuit C such that $C(\langle x, i, t \rangle) = z$. Finally, we have the following *local* characterization of the language L_M, and this shows that $L_M \in \Sigma_2^p$: $x \in L_M \iff \exists C \forall i, t$ s.t. $C(\langle x, i-1, t-1 \rangle), C(\langle x, i, t-1 \rangle), C(\langle x, i+1, t-1 \rangle)$, and $C(\langle x, 1, 2^{n^k} \rangle)$ are accepting. We note that the length of C, i, and t are polynomial length in $|x|$. $\square$

2.2 Randomized Complexity Classes and PRG

We define a randomized Turing machine and complexity classes related to the model.

Definition 2.8. A randomized Turing machine is a Turing machine with an additional state q_{random}. If the machine is in state q_{random} the next state will be either q_0 or q_1 with probability $1/2$ for each state.

We can give an equivalent description of this model as a Turing machine having an additional tape for random bits. This tape is read only and the machine can only go right on the tape.

Definition 2.9. Let $T : \mathbb{N} \rightarrow \mathbb{N}$. $L \in \mathsf{BPTIME}[T(n)]$, if there is a randomized Turing machine M running in time $O(T(n))$ for any input string of length n such that the following two conditions (1) and (2) hold.

$$(1)\forall x \in L, Pr[M(x) = 1] \geq \frac{2}{3}$$

$$(2)\forall x \notin L, Pr[M(x) = 1] \leq \frac{1}{3}$$

We define one of the most basic randomized complexity classes as follows.

Definition 2.10.

$$\mathsf{BPP} = \bigcup_{c \geq 1} \mathsf{BPTIME}[n^c]$$

The following theorem is about a strict separation of uniform randomized efficient computation and nonuniform efficient computation, because some undecidable language can be computed by circuits of small size.

Theorem 2.11 (**[1]**).

$$\mathsf{BPP} \subseteq \mathsf{P/poly}.$$

Lemma 2.12. For any $L \in \mathsf{BPP}$ and arbitrary constant $c > 0$, there exists a randomized Turing Machine M such that it runs on input x in polynomial time in $|x|$ with error probability $2^{-c \cdot |x|}$.

Proof. Let $L \in \mathsf{BPP}$. There exists a randomized Turing machine running in polynomial time such that the following conditions hold.

$$\forall x \in L, Pr[M(x) = 1] \geq \frac{2}{3}$$

and

$$\forall x \notin L, Pr[M(x) = 1] \leq \frac{1}{3}$$

Given M, x we can think of $M(x)$ as a random variable which can be sampled effectively. Because the running time on x is polynomial in $|x|$. If M accepts x, this variable has high expectation, whereas if M rejects x this random variable has low expectation. We prove that the number of samples of $M(x)$ is enough to approximate the expectation of $M(x)$ within relatively small constant with success probability $2^{-c \cdot |x|}$.

We design a randomized Turing machine M' such that on input x it computes $E[M(x)]$ = Pr[M(x)=1] and simulate m times independently M on x. Thus, we consider to calculate the following ratio A.

$$A = \frac{\text{the number of accepting computational paths of } M}{m}$$

.

The machine M' will accept if $A \geq \frac{2}{3} - \frac{1}{10}$. To calculate error probability of M', let A_i be the random variable $M(x)$ on the i-th run. Using this notation, $A = \frac{1}{m}\sum_{i=}^{m} A_i$. By linearity of expectation, it holds that $E[A|x \in L] \geq 2/3$ and $E[A|x \notin L] \leq 1/3$. These two expectations are far away, and we can distinguish between the two cases with high probability. Because of independent m simulations, we can bound the error probability by Chernoff's inequality. We also note that $m = O(n)$ □

Prof of Theorem 2.11

Let $L \in \mathsf{BPP}$. By Lemma 2.12, there is a polynomial time randomized Turing machine M with error probability less than $2^{-(n+1)}$. Let t_n be the maximum length of all random strings M uses for inputs of length n. Note that t_n is polynomial in n. Let $M_r(x)$ denote the output of M on any input x, using r as the random strings. Since the error probability is less than $2^{-(n+1)}$, for arbitrary x of length n, the following holds.

$$|\{r \in \{0,1\}^{t_n}\}| \leq 2^{-(n+1)} \cdot 2^{t_n}.$$

Taking the union bound of these sets forall $x \in \{0,1\}^n$.

$$|\{r \in \{0,1\}^n \text{ such that } M_r(x) \text{ is wrong }\}| \leq 2^{t_n - 1}.$$

For any input length n, there are at least one random string r_n, such that M can execute correctly when it uses the random string and runs for arbitrary input string of length n. There is a family of *deterministic* Turing machine $\{M_{r_n}\}_{n\in\mathbb{N}}$, which uses r_n whenever M runs for any input string of length n. By the simulation to prove $\mathsf{P} \subseteq \mathsf{P/poly}$, we obtain a corresponding polynomial size circuit family to simulate such family of Turing machine. □

Now we introduce the notion of pseudo random generator (PRG, in short).

Definition 2.13. For $S : \mathbb{N} \to \mathbb{N}$, a function $G : \{0,1\}^* \to \{0,1\}^*$ is called *S-pseudo random generator*, if the following conditions (1), (2), and (3) hold, for any circuit C of size $O(S(l)^3)$, where U_l is an uniform random strings of length l.

$$(1)|G(z)| = S(|z|), \text{for any } z \in \{0,1\}^*$$
$$(2) \text{ Running time for inputs of length } l \text{ is } 2^{O(l)}$$
$$(3)|Pr[C(U_{S(l)}) = 1] - Pr[C(G(U_l))]| \leq \frac{1}{10}$$

We note that the machine G prints a pseudo random string whose length is increased for an input random string.

Theorem 2.14. If there is some S-pseudo random generator, then the following holds.

$$\mathsf{BPTIME}[S(l)] \subseteq \mathsf{DTIME}[2^{O(S(l))}].$$

Proof. Let L be a language that is determined by the randomized Turing machine M with running time $S(l)$, on input of length l. Let $r \in \{0,1\}^{S(l)}$ be the random bits that M uses. We consider the following two cases. First, if a PRG G can derandomize the PRG then we obtain the desired result. Second, if not, this PRG can be used to obtain circuits which will be a contradiction to the fact that G is a PRG.

If G is in the first case, then the following holds.

$$|\underset{r\in\{0,1\}^{S(l)}}{Pr}[M_r(x)=1] - \underset{z\in\{0,1\}^{l}}{Pr}[M_{G(z)}(x)=1]| \le \frac{1}{10}$$

Note that $|z| = S(l)$ by Definition 2.13. This G is able to be used to derandomize M: For any $z \in \{0,1\}^l$, M' simulates $M_{G(z)}(x)$ and decides by the majority. M' wil be correct on all input x, for each $x \in L$:

$$\underset{r\in\{0,1\}^{S(l)}}{Pr}[M_r(x)=1] \ge \frac{2}{3}$$

$$\underset{z\in\{0,1\}^{l}}{Pr}[M_{G(z)}(x)=1] \ge \frac{2}{3} - \frac{1}{10} = \frac{17}{30}$$

And for each $x \notin L$:

$$\underset{z\in\{0,1\}^{l}}{Pr}[M_{G(z)}(x)=1] \le \frac{1}{3} + \frac{1}{10} = \frac{13}{30}$$

The runtime will be $S(l) \cdot 2^l$, which is $2^{O(l)}$ (because $S(l) = 2^{O(l)}$).

We consider the second case. Suppose the following inequality.

$$|\,Pr_{r\in\{0,1\}^{S(l)}}[M_r(x)=1] - Pr_{z\in\{0,1\}^{l}}[M_{G(z)}(x)=1]| > \tfrac{1}{10} \tag{2.1}$$

for an infinite number of x's, then it can be used to contradict the definition of G as a PRG. If this holds only for a finite number x's, then we can construct a machine M'' and can use G to derandomize M''.

Let $\{x_i\}_{i\in I}$ be an infinite sequence of x's, which satisfy the above condition (2.1). The series of circuits $\{C_i\}$ such that C_i on input r, has x_i hard-coded, and simulates $M_r(x_i)$. If there is no x_i of its length, C_i outputs 0. $\{C_i\}$ distinguishes between r and $G(z)$ with probability larger than 1/10. Because there is a circuit of size t^2 for any deterministic Turing machine with runtime t, $\{C_i\}$ contradicts that G is a PRG. □

Nisan and Wigderson proved that given a strong enough circuit lower bound, in particular super polynomial size, it is possible to construct a PRG and thus obtain a derandomization of BPP[33].

2.3 Interactive Proof Systems

We first give the notion of an interactive proof system.

Definition 2.15. An interactive proof system is a multi-round protocol between two parties, a prover and a verifier, such that on each round, messages are exchanged between the verifier and the prover to establish if a string belongs to the language or not. We suppose that the prover is all powerful, but cannot be trusted, while the verifier has bounded resources. An interactive proof system must satisfy the following properties:

(1) (Completeness) There is a proof strategy for the prover such that if a string is in the language then the verifier is convinced of this.

(2) (Soundness) If a string is not in the language, then no prover can convince the verifier that the string is in the language.

Remind the definition of NP. A language Ł $\in \mathsf{NP}$ if there exists a Turing machine M for which the following holds:

$$x \in L \iff \exists y \in \{0,1\}^{|x|^c}, M(x,y) = 1,$$

for some constant c. Therefore NP is an Interactive Proof System where the verifier is a P machine. The prover produces a polynomial size certificate and the verifier verifies it in polynomial time. We note that there is no assumption on the power to compute the string $y = y(x)$ for given x. The fact that the prover is computationally unlimited is formalized by the existential quantifier. In the following definition, we look at another proof system, where the verifier can use random bits to decide if to accept a certificate sent by the prover.

Definition 2.16. We define MA as the class of languages L for which there exists a probabilistic turing machine M such that

$$x \in L \Rightarrow \exists y \in \{0,1\}^{|x|^c} Pr[M(x,y) = 1] \geq 2/3.$$

$$x \notin L \Rightarrow \forall y \in \{0,1\}^{|x|^c} Pr[M(x,y) = 1] \leq 2/3.$$

One can informally think of MA as a randomized version NP, which means that MA contains NP and BPP. The inclusion $\mathsf{MA} \subseteq \Sigma_2^p$ is also proved. We call the prover and the verifier in MA protocols Merlin and Arthur, respectively.

Definition 2.17 ([17])**.** IP is the class of languages defined by an Interactive Proof System, where a prover P and a verifier V communicate using random bits r and messages of polynomial length sent over polynomially many rounds and let $C(V, P, x, r)$ denote the decision of the communication protocol for given input x. That is, there is a verifier V such that

$$P[C(V, P, x, r) = 1] \geq 2/3,$$

for some prover P, and

$$P[C(V, Q, x, r) = 0] \leq 2/3,$$

for any prover Q, where $C(V, P, x, r) = 1$, if V accepts, and otherwise rejects.

While there exists an oracle O such that $\mathsf{coNP}^O \not\subseteq \mathsf{IP}^O$, it was later proved that IP has very powerful computational ability.

Theorem 2.18 ([**39**])**.**

$$\mathsf{IP} = \mathsf{PSPACE}$$

Theorem 2.19.

$$\mathsf{PSPACE} \subseteq \mathsf{P/poly} \Rightarrow \mathsf{PSPACE} = \mathsf{MA}$$

Proof. We use the fact that $\mathsf{PSPACE} = \mathsf{IP}$. The interaction between Merlin and Arthur is an instance of *True Quantified Boolean Formula* (TQBF, in short), and Merlin is a PSPACE machine. Because of the equivalence between PSPACE and IP, Merlin can be replaced with a polynomial size circuit family $\{C_n\}$. Note that the prover in the IP-protocol is a function computing a massage sent to the verifier for given input string, random bits, and all massages which is already sent by the two parties.

The interaction between Merlin and Arthur can now be executed in only one round. Given input string x of length n, Merlin sends to Arthur C_n of polynomial in $|x|$. Arthur then simulates the interactive proof getting answers from C_n instead of Merlin. Note that if the input is not in the language, then every circuit sent to Arthur by Merlin fails to act as a prover, and it does not have sufficient probability to fool the verifier. $\square$

Theorem 2.20.

$$\mathsf{EXP} \subseteq \mathsf{P/poly} \Rightarrow \mathsf{EXP} = \mathsf{MA}$$

Proof. Suppose $\mathsf{EXP} \subseteq \mathsf{P/poly}$. We have the following inclusion:

$$\mathsf{EXP} \subseteq \Sigma_2^p \subseteq \mathsf{PSPACE} \subseteq \mathsf{EXP}$$

Thus, $\mathsf{EXP} = \mathsf{PSPACE} = \mathsf{MA}$. $\square$

2.4 Turing Machine with Advice Strings

We defined the class $\mathsf{P/poly}$ as class of languages which is computable by nonuniform boolean circuit families of polynomial size. We will introduce a model which is an extended version of Turing machine and is essentially equivalent to circuit family models.

A Turing machine is said to take advice if the machine has access to a string α_n on top of the string x for any input string x of length n.

Definition 2.21. Let $t, \alpha : \mathbb{N} \to \mathbb{N}$ be two functions, and we later regard t and α as *time* and *advice* functions, respectively. A language L is in the complexity class $\mathsf{DTIME}[t(n)]/\alpha(n)$, if there exist some Turing machine M running in time $t(n)$ on input strings of length n and a family of strings $\{\alpha_n\}_{n\in\mathbb{N}}$ such that **(1)** $|\alpha_n| \le a(n)$ for all n and **(2)** $s \in L$ iff $M(x, \alpha_{|x|}) = 1$.

The name of the class $\mathsf{P/poly}$ is perhaps clearer at this point: to the left of the slash we have the complexity class P and to the right poly which means advice of polynomial length. We formally state this correspondence as follows.

Theorem 2.22.

$$\mathsf{P/poly} = \bigcup_{a,b\in\mathbb{N}} \mathsf{DTIME}[n^a]/n^b$$

Proof. We first prove the $\subseteq$ direction. Evaluation of a circuit on a given input can be executed in polynomial time in the description length of the circuit and the input. Thus, advice string is the description of the circuit, which is stored during all evaluation processes.

Next, we prove the other direction. Let $L \in \mathsf{DTIME}[n^a]/n^b$ for some constants a, b. The idea again is simple. Because $\mathsf{P} \subseteq \mathsf{P/poly}$, the Turing machine for L can be simulated by a circuit family. We then take an advantage of the nonuniformity by hard-writing the advices, one in each circuit. There is some Turing machine M running in time $O(n^a)$ for inputs of length n, and a family of strings $\{\alpha_n\}$ with $|\alpha_n| \le n^b$ such that $x \in L$ iff $M(x, \alpha_{|x|}) = 1$. We remind that there is a family of circuits $\{C_n\}$ of size $O(n^{2a})$ that agrees with M. Note that we can fix all input variables of the circuit C_n corresponding to $\alpha_{|x|}$ and can obtain a circuit $C_{n'}$. Hence, we get a family of circuit $\{C'_n\}$ deciding L. □

In this proof, we note that evaluating a circuit is done in time linear in its size. When we consider Turing machine with advice strings, one can separate the computation from the length of advice strings. Indeed, one can consider $\mathsf{P}/1$, which is the complexity class with efficient computation using just one bit advice. This class is actually strong enough to decide a undecidable problem like Halting problem.

Remind that $\mathsf{NEXP} = \bigcup_a \mathsf{NTIME}[n^a]$ and $\mathsf{P/poly} = \bigcup_b \mathsf{SIZE}[n^b]$. We consider the following question. Is there a $b = b(a)$ for any a it holds that $\mathsf{NTIME}[2^{n^a}] \subseteq \mathsf{SIZE}[n^b]$? For general sets, this doesn't hold. However, we can give an affirmative answer to this question for some complexity classes, because we consider only complexity classes which are sets with specific structures. This fact is useful for us.

Lemma 2.23. If $\mathsf{NEXP} \subseteq \mathsf{P/poly}$, then the following holds.

$$\forall a \in \mathbb{N}, \exists b = b(a) \in \mathbb{N}, \mathsf{NTIME}[2^{n^a}]/n \subseteq \mathsf{SIZE}[n^b]$$

Proof. For a given $a \in \mathbb{N}$, let $U_a(\cdot,\cdot)$ be a universal nonderterministic Turing machine which simulates the i-th nondeterministic Turing machine M_i for input x in $2^{|x|^a}$steps. We note that $L(U_a)$, which is the language of all strings accepted by U_a, is in NEXP. Hence, we have $L(U_a) \in \mathsf{P/poly}$ by the assumption. Therefore, for some constant c, there is a family of circuits $\{C_n\}$ of size $|C_n| \le n^c$ such that $C_{|x,i|}$ computes $L(U_a)$, i.e., $x \in L(U_a)$ iff $C_{|x,i|}(|x, i|) = 1$.

Next, we prove $\mathsf{NTIME}[2^{n^a}]/n \subseteq \mathsf{SIZE}[n^b]$. Take a language $L \in \mathsf{NTIME}[2^{n^a}]/n$. Then, there is a sequence of advices $\{\alpha_n\}_{n\in\mathbb{N}}$ with $n = |\alpha_n|$, and an index $i = i_L$ such that for any $x \in \{0,1\}^*$ we have $x \in L$ if and only if $M_i(x, \alpha_{|x|})$ has an accepting computation path, where M_i is the i-th nondeterministic Turing machine. We take the family of circuits $\{C_n\}$ as above. Thus we have $C_{|x,\alpha_{|x|},i_L|}(x,\alpha_{|x|},i_L)$ iff $x \in L$. Therefore, by partially fixing the inputs we obtain the desired family of circuits computing L of size at most $(|x|+|\alpha_{|x|}|+|i_L|)^c \leq (2n+|i_L|)^{c+1}$. □

2.5 The Notion of Infinitely Often Classes

The notion of *infinitely often* is also quite basic and essential in the structural complexity theory. Roughly speaking, given a complexity class C, the infinitely often version of C is arbitrary language agreeing with some language from C on infinitely many inputs. For example, for a language $L \in \mathsf{C}$, $L' = \{x : x \in L, |x| = 3n-2, n \in \mathbb{N}\}$ is in the infinitely often version of C.

Definition 2.24. Let C be a complexity class. Define the class $\mathsf{io\text{-}C}$ to contain any language L for which there is some language $L' \in \mathsf{C}$ and an infinite set $I \subseteq \mathbb{N}$, such that for any $n \in I, L \cap \{0,1\}^n = L' \cap \{0,1\}^n$.

It is not hard to prove the following lemma.

Lemma 2.25. Let $\mathsf{C}_1, \mathsf{C}_2$ be two complexity classes. Then,

$$\mathsf{C}_1 \subseteq \mathsf{C}_2 \Rightarrow \mathsf{io\text{-}C}_1 \subseteq \mathsf{io\text{-}C}_2$$

We will also make use of the following lemma.

Lemma 2.26. For any fixed $c \in \mathbb{N}$ it holds that $\mathsf{EXP} \not\subseteq \mathsf{io\text{-}SIZE}[n^c]$

Proof. Firstly, we take a language in EXP, which is hard for $\mathsf{io\text{-}SIZE}[n^c]$. Next, we give a contradiction to the size hierarchy theorem.

By the size hierarchy theorem, there exists $n_0 = n_0(c)$, for any $n > n_0$, there exists a function f_n on n inputs such that **(1)** f_n can not be computed by circuits of size n^c and **(2)** f_n yet can be computed by circuits of size at most $4n^c$. For given input length n, we can find the lexicographically minimum function of all functions satisfying this statement. *Moreover, we can simulate the function in exponential time.* Let L_c be the resulting language $\bigcup_{n \geq n_0} f_n^{-1}(1)$.

If $L_c \in \mathsf{io\text{-}SIZE}[n^c]$ then there exists a family of circuits $\{C_n\}$ of size at most n^c, where C_n and f_n are equivalent for infinitely many lengths of input strings, that is, L_c on the respective input length validly. This contradicts the fact that all circuits except the first n_0 ones can not compute L_c. □

Corollary 2.27. If $\mathsf{NEXP} \subseteq \mathsf{P/poly}$ then for any fixed $a \in \mathbb{N}$ it holds that

$$\mathsf{EXP} \not\subseteq \mathsf{io\text{-}[NTIME}[2^{n^a}]/n].$$

Proof. By the assumption and Lemma 2.23, there is some $b = b(a)$ such that the following inclusion holds.

$$\mathsf{NTIME}[2^{n^a}]/n \subseteq \mathsf{SIZE}[n^b]$$

By Lemma 2.25, it holds the following statement

$$\text{io-}[\mathsf{NTIME}[2^{n^a}]/n] \subseteq \text{io-}\mathsf{SIZE}[n^b].$$

This is, however, a contradiction to Lemma 2.26. $\square$

2.6 NEXP ⊆ P/poly implies NEXP = EXP

We prove the following deterministic simulation of NEXP assuming NEXP ⊆ P/poly.

Theorem 2.28.

$$\mathsf{NEXP} \subseteq \mathsf{P/poly} \Rightarrow \mathsf{NEXP} = \mathsf{EXP}.$$

Indeed, we give a proof to show that both NEXP ⊆ P/poly and NEXP ≠ EXP cannot hold. By Corollary 2.27, it holds that if we suppose NEXP ⊆ P/poly then the following holds.

$$\forall a \in \mathbb{N}, \mathsf{EXP} \not\subseteq \text{io-}[\mathsf{NTIME}[2^{n^a}]/n]$$

Thus, all we have to prove is the following statement.

Lemma 2.29. If NEXP ≠ EXP then, the following holds.

$$\forall a \in \mathbb{N}, \mathsf{MA} \subseteq \text{io-}[\mathsf{NTIME}[2^{n^a}]/n]$$

Proof. By the assumption that NEXP ≠ EXP, we can take some NEXP complete language $L^* \in \mathsf{NEXP} \setminus \mathsf{EXP}$ under the polynomial time many to one reduction. Because of $L^* \in \mathsf{NEXP}$, there are some constant c^* depending on L^* and some a nondeterministic Turing machine M^* such that that runs in time $O(2^{n^{c^*}})$ on inputs of length n, such that

$$x \in L \iff \exists y \in \{0,1\}^{2^{|z|^{c^*}}} M^*(z, y) = 1$$

We consider what happens, if $L^* \in \mathsf{EXP}$. Any Turing machine running in deterministic exponential time cannot get a success to decide L^*. We will take only a particular Truing machine to decide L^* in deterministic exponential time, and this specification is useful for us. Trivially, it take a double exponential time for the simulation of the nondeterministic Turing machine M^* by just enumerating over all potential witnesses y.

It is an important idea that we consider only witness which is "easy" in some sense [28]. We consider a witness y regarded as a truth table of functions having small circuit size complexity. We mention this formal.

For any constant d, consider the following deterministic Turing machine M_d: On input z of length n, it list all circuits of size n^d with n^{c^*} input variables. For each circuit C, take the truth table $y = \mathsf{tt}(C)$ of C, which is a string of length $2^{n^{c^*}}$. Then, check if $M^*(z, y) = 1$. If we found no such y, the machine rejects z, otherwise the machine accepts z.

We note that there is no witness to be in L^* for z, if $z \notin L^*$. Thus there is no easy witness for this false claim. Hence, M_d rejects z. We also note that the running time of M_d is:

$$O\left(((n^{2d})n^d) \cdot (2^{n^{c^*}} \cdot n^d) \cdot (2^{n^{c^*}})\right).$$

Therefore, for every fixed constant d, the Turing machine M_d runs in exponential time. Thus it cannot compute L^*. We can assume that M_d wrongfully decides L^* for infinitely many inputs. The reason is: if not, correcting the error of M_d, we can add the finite number of inputs for which M_d wrongfully decides to the description of M_d. That is, for every d, there is some infinite sequence of input strings $B_d = \{z_i^{(d)}\}_{i \in I_d}$ for which $M_d(z_i^{(d)}) = 1$ if and only if $z_i^{(d)} \notin L^*$, where $I_d \subseteq \mathbb{N}$ is the set of lengths for which there are bad inputs in some sense.

We also note that M_d makes only one-sided error, that is, if $z \notin L^*$ then M_d can validly reject z for any fixed d. The error of this Turing machine is false-negative, rejecting inputs which should have been accepted. This will happen, only when the inputs have only hard witnesses. Here we say hard witnesses as the witnesses that cannot be computed by circuits of size $|z|^d$.

Thus, for any d there exists a nondeterministic Turing machine such that for given n it runs in time $2^{n^{c^*}}$ with an advice string of length n and outputs the truth table of a function having no circuit of size n^d. The machine M'_d nondeterministically guess a string $y \in \{0,1\}^{2^{n^{c^*}}}$ and verifies if $M^*(z_n^{(d)}, y) = 1$. If the verification results in affirmative decision, then the machine M'_d prints y.

Note that the machine computes with n bits of advice string $z_n^{(d)}$ and runs in time $O(2^{n^{c^*}})$. If $n \in I_d$ then $z_n^{(d)}$ is an input which M_d rejects wrongfully. Thus, by the above arguments, $z_n^{(d)}$ while any witness for this fact, and there are such witnesses whose boolean functions associated with truth tables have no circuits of size n^d. Therefore, the machine M'_d are nondeterministically able to output a string $y \in \{0,1\}^{2^{n^{c^*}}}$ as the truth table of a boolean function without circuits of size n^d.

Let $L \in \mathsf{MA}$ be a language. Then, there is some constant $d = d(L)$ such that Merlin sends Arthur a proof $y \in \{0,1\}^{|x|^d}$ to claim "$x \in L$". Arthur then take $|x|^d$ random bits and decides in time $|x|^d$ if he accepts x for given y.

We consider to derandomize Arthur's random computation. We only take the case satisfying $n = |x| \in I_d$, that is, we can consider hard boolean function for the purpose of derandomization. There exists a Turing machine M'_d running in time $O(2^{n^{c^*}})$, which is exponential with the constant c^* not depending on d. The machine $M_{d'}$ outputs the truth table of an n^d-hard function. We can use Nisan-Wigderson PRG and this hard function, and this gives us a derandomized Arthur.

This simulation of Arthur takes time $n^{O(d)}$. Because we consider the machines with n bits advice strings, runs in nondeterministic time $O(2^{n^{c^*}}) + n^{O(d)}$, and correctly computes L for any input string of length $n \in I_d$. Thus, we can execute an infinitely derandomization i.e. $L \in$ io-$[\mathsf{NTIME}[2^{n^{c*}}]/n]$. $\square$

2.7 Universal Witness Circuits

We formally define the notion of easy witness in the proof of Lemma 2.29. This notion is essential to get circuit lower bounds from satisfiability algorithms.

Definition 2.30. A language $L \in \mathsf{NTIME}[t(n)]$ has $S(n)$-size *universal witness circuits(U.W.C)*, if ; For any verifier V, there is some circuit family $\{C_n\}_{n\in\mathbb{N}}$ such that the following **(1)** and **(2)** hold, where $l = n + \lceil \log_2 t(n) \rceil + 1$.

(1) Size$(C_n) = O(S(n))$,
(2) For any input string x of length n, $w(x)$ is a witness i.e. $x \in L \iff V(x, w(x)) = 1$,

where each bit of $w(x)$ is $C_n(\langle x, \text{bit-index}\rangle)$.

Note that $w(x)$ can be written as follows.

$$\begin{aligned} w(x) &= w_{0\cdots 00}(x) w_{0\cdots 01}(x) w_{0\cdots 10}(x) \cdots w_{1\cdots 11}(x) \\ &= C_l(\langle x, 0\cdots 00\rangle) C_l(\langle x, 0\cdots 01\rangle) C_l(\langle x, 0\cdots 10\rangle) \cdots C_l(\langle x, 1\cdots 11\rangle) \end{aligned}$$

We also note that the length of the binary string $w(x)$ is $\lceil \log_2 t(n) \rceil + 1$. The integer k is introduced for an infinite sequence of inputs to design an infinitely often simulation of Merlin-Arthur protocols.

Lemma 2.31. If $\mathsf{NEXP} \subseteq \mathsf{P/poly}$, then NEXP has *universal witness circuits* with size $S(n) = \mathsf{poly}(n)$.

Proof. We use the following known theorem for derandomization.

Theorem 2.32. $\forall \varepsilon > 0 \exists \delta < \varepsilon \exists e \in \mathbb{Z}$ such that for given boolean function with n^δ variables whose circuit complexity is at least $n^{\delta e}$, there exists a pseudo random generator $G : \{0,1\}^{n^\varepsilon} \to \{0,1\}^n$ computable in $2^{O(n^\varepsilon)}$ time which fools circuits of size n.

Is there a language which *does not* have U.W.C. of small size? If not, we complete the proof. If so, we can construct a *hard* function in the sense that it has strong circuit complexity lower bounds. By derandomization from the hardness via Theorem 2.32, we prove $\mathsf{MA} \subseteq$ io-$\mathsf{NTIME}[2^n]/n$. However, this contradicts to the assumption $\mathsf{NEXP} \subseteq \mathsf{P/poly}$ and the following known facts which is from Meyer's Theorem, Lemma 2.26 and Theorem 2.28:

$$\mathsf{NEXP} \subseteq \mathsf{P/poly} \Rightarrow \mathsf{NEXP} = \mathsf{EXP} = \mathsf{MA}$$

and

$$\mathsf{EXP} \subseteq \mathsf{P/poly} \Rightarrow \mathsf{EXP} \not\subseteq \text{io-}\mathsf{NTIME}[2^n]/n.$$

Suppose that there is a language which have U.W.C. of small size, the following holds by the definition of U.W.C.

$$\begin{aligned}&\exists V \text{verifier for } L \forall k (\geq 1) \forall \{C_n\}_{n\in\mathbb{N}} \text{ s.t. } \mathbf{(2)} \Rightarrow \neg\mathbf{(1)},\\ &\neg\mathbf{(1)} \text{Size}(C_n) \neq O(S(n)),\\ &\mathbf{(2)} \text{For any input string } x \text{ of length } n\ w(x) \text{ is a witness i.e. } x \in L \iff V(x, w(x)) = 1,\end{aligned} \tag{2.2}$$

where $w(x) = C_l(\langle x, z_0\rangle)\ C_l(\langle x, z_1\rangle) \cdots$.

We note that the length of z_i is polynomial in n for each i because the length of arbitrary witness string for NEXP language is at most $2^{n^{O(1)}}$.

There is a infinite sequence of inputs $S = \{x_{i_k}\}_{k\in\mathbb{N}}$ such that;

1. $\forall k, x_{i_k} \in L$, and
2. $\exists k_0 \forall k (k \geq k_0) \forall y (|y| = 2^{|x_{i_k}|^c}) \forall d \geq 1$,
   $V(x_{i_k}, y) = 1 \Rightarrow$ circuit size for f is greater than $|x_{i_k}|^d$, where $f : \{0,1\}^{n^{\varepsilon ac}} \ni (x_{i_k}, i) \mapsto y_i \in \{0,1\}$

Arthur can be simulated by a polynomial size n^q circuit A, since $\mathsf{BPP} \subseteq \mathsf{P/poly}$. What is written to the advice? We set the advice for n-bit inputs to be the string $x_{i_l} \in S$, where x_{i_l} has length $n^{\varepsilon a}$ and to be the string 0^n with no existence of such string $x_{i_l} \in S$. Our purpose is to simulate *infinitely often*, thus we do *not* have to choose n *successively*. For any fixed l, we can take some (sufficiently large) n such that $|x_{i_l}| = n^{\varepsilon a}$. We can construct the following algorithm for the simulation.

1. Nondeterministically guess the following two strings:
   - a witness y of length $2^{|x_{i_k}|^c}$ for the verifier V, and *REJECT* if $V(x_{i_k}, y) = 1$ does not hold
   - the message which is sent by Merlin and has polynomial length.
2. Simulate the Arthur:
   - Evaluate G on all $n^{\varepsilon a}$ possible seeds
   - Evaluate the output of the circuit A on the outputs of G regarding y as a truth table of a hard funciton
   - Take the majority of all outputs of A

The pseudo random generator G fools Arthur. Treat y as a hard function : the number of valuables is $n^{c\varepsilon a}$ and circuit complexity is at least $n^{\varepsilon ad}$. Since $n^{\varepsilon ad/\delta e} \geq n^a$ as setting d to be arbitrary large, we can fool circuits of size n^a. Running time is bounded above: $O(2^{n^{c\varepsilon a}} + 2^{n^{\varepsilon a}})$ and setting $\varepsilon(= \varepsilon(c,a)) > 0$ to be arbitrary small accomplish the desired inclusion $\mathsf{MA} \subseteq \text{io-}\mathsf{NTIME}[2^{n^{\varepsilon}}]/n^{\varepsilon}$, for $\forall \varepsilon > 0$. □

Note that the assumption $\mathsf{NEXP} \neq \mathsf{EXP}$ is the source of hardness for derandomization to prove Lemma 2.29 and denying the existence of small U.W.C is the one in this proof. We also note that the number of input variables in witness circuits which we consider is smaller than the one of the witness circuit which is obtained by the exhaustive search running in exponential time in the proof of Theorem 2.29. This difference about the number of input variables is critical to prove Theorem 2.3.

2.8 A faster algorithm rules out small U.W.C

We give the proof of the following theorems connecting slight improvements of satisfiability to circuit lower bounds.

Theorem 2.3(restated) Suppose there is a super polynomial function $s(n)$ such that CIRCUIT SAT on circuits with n variables and n^k gates can be solved in $2^n \cdot \mathsf{poly}(n^k)/s(n)$ time by a (co-non)deterministic algorithm , for all k. Then $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$.

Theorem 2.33. Let $c \geq 1$. Let $a(n)$ be a monotone increasing and unbounded function. Let $S(n), T(n)$ be functions such that

$$T(n)/(S(n) + n^8) \geq \Omega(n^4 a(n)), \text{ and } n \leq S(n) \leq O(2^n/n \cdot 1/a(n)). \tag{2.3}$$

If Circuit SAT on n variables and m gates can be solved in $O(2^n m^m/T(n))$ co -nondeterministic time then $\mathsf{NTIME}[2^n]$ does not have $S(n)$-size U.W.C. .

We use the following known results about efficient local reduction.

Theorem 2.34 (eg: [12, 35]). $L \in \mathsf{NTIME}[n]$ can be reduced to 3SAT instances of $n(\log n)^d$ size. Moreover there is an algorithm that given instance of L and an integer $i \in [cn(\log n)^d]$ in binary, outputs the i-th clause of the resulting 3SAT formula in $O((\log n)^d)$ time.

We obtain the following corollary by Lemma 2.5.

Corollary 2.35. $L \in \mathsf{NTIME}[2^n]$ can be reduced to 3SAT instances of $c2^n n^4$ size. Moreover there is an algorithm that given instance of L and an integer $i \in [c2^n n^4]$ in binary, outputs the i-th clause of the resulting 3SAT formula in $O(n^4)$ time.

Proof of Theorem 2.33. We give a proof by contradiction to the nondeterministic hierarchy theorem, which states that the separation $\mathsf{NTIME}[f(n)] \subsetneq \mathsf{NTIME}[g(n-1)]$ holds for any functions $f, g : \mathbb{N} \to \mathbb{N}$ satisfying that $f(n) = o(g(n-1))$.

Assume that $\mathsf{NTIME}[2^{\mathsf{n}}]$ has a family of U.W.C of $S(n)$-size, determining whether the variable corresponding to a given bit-index is 0 or 1. Note that L has U.W.C i.e. $\forall x \in L \exists y (|y| \leq c2^n n^4)$ s.t. $V(x,y) = 1$ and y can be encoded with a circuit of size $S(|x|)$. That is, we can obtain *a kind of compressed representation*: $\forall x \in L \exists C_y$ such that C_y is an U.W.C with input length $l = \log(c2^{|x|}|x|^4)$ and size $S(|x|)$.

We construct a nondeterministic algorithm N for L which is an arbitrary $\mathsf{NTIME}[2^{\mathsf{n}}]$ language.

1. For an input string x, existentially guess the circuit C_y
   - Given index of a variable as an input string, C_y outputs the 0-1value of the variable
2. Construct a circuit D with l input wires such that

   $D(X)$ outputs 1 $\iff$ the X-th clause is *not* satisfied. (see Fig1. and Fig2.)
3. Recall the assumption that there is a faster C-SAT algorithm. We can call the algorithm to solve circuit satisfiability on D, and $ACCEPT$ if and only if D is *unsatisfiable*

Using bounds on $S(n), T(n)$, we prove that the running time of this nondeterministic algorithm is at most $O(2^n/a(n))$. This is, however, a contradiction to the hierarchy theorem, since $2^n/a(n) = o(2^{n-1})$ and $\mathsf{NTIME}[2^n] \subseteq \mathsf{NTIME}[2^n/a(n)]$.

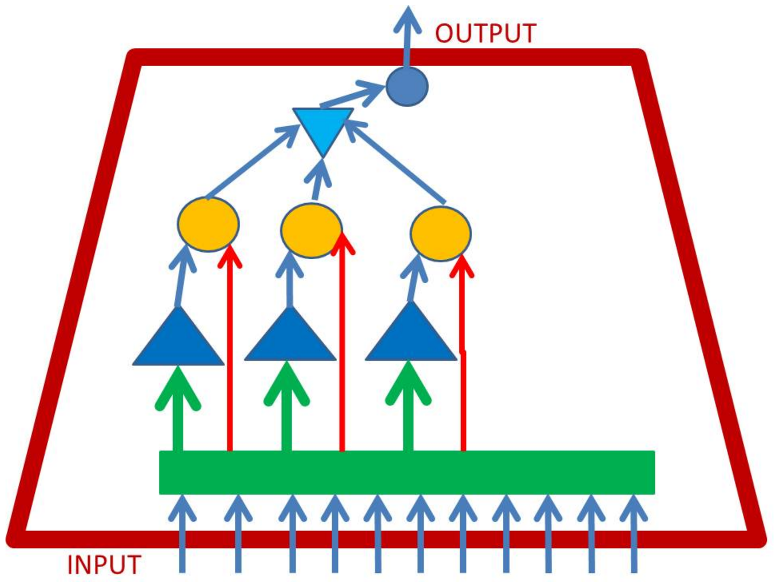

Fig. 2.1:

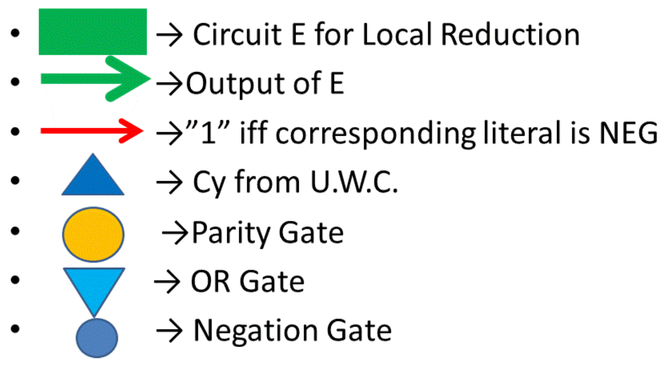
→ Circuit E for Local Reduction
→Output of E
→"1" iff corresponding literal is NEG
→ Cy from U.W.C.
→Parity Gate
→ OR Gate
→ Negation Gate

Fig. 2.2:

The running time of the C-SAT algorithm on the circuit D is ;

$$\begin{aligned} O(2^n/a(n)) + 2^l(n^8 + S(n))^c/T(n) &\leq O(2^n/a(n) + 2^n n^4 (n^8 + S(n))^c/T(n)) \\ &\leq O(2^n/a(n) + 2^n n^4/(n^4 a(n))) \\ &\leq O(2^n/a(n)) \end{aligned}$$

□

Finally, we show that designing a faster C-SAT algorithm is a way to prove NEXP $\not\subseteq$ P/poly.

Proof of Theorem 2.3. Proving this theorem is straight forward. Let $S(n) = n^k, T(n) = s(n)$ in Theorem 2.3. Then a $2^n \cdot \mathsf{poly}(n^k)/s(n)$ time algorithm for C-SAT implies that NEXP does not have n^k-size U.W.C. Since k can be arbitrary, NEXP does not have n^k-size U.W.C, thus NEXP also does not. Taking contrapositive of Lemma 2.31, we complete the proof. □

Chapter 3

Satisfiability for a Restricted Class of Depth Two Threshold Circuits

The structure of this chapter is as follows. Firstly, we define several notions being necessary in other sections in Section 3.1 and formally state our results in Section 3.3. We give an overview of the entire algorithm in Section 3.4. In Section 3.4, we also define a problem: for given a circuit and a graph containing information about dependency of the circuit output YES if and only if the circuit is satisfiable. We give a constructive proof of a reduction from our original circuit satisfiability problem to this problem in Section 3.5. In Section 3.6, we solve the problem defined in Section 3.4. In Section 3.7, we give an algorithm whose subroutine is to solve the problem in Section 3.5 to solve the original satisfiability problem.

3.1 Preliminaries

A threshold gate which outputs a boolean value has the label $w_1x_1 + \cdots + w_nx_n \geq t$, where each of $w_1, ..., w_m$ and t is a real number, and $x_1, ..., x_n$ are input boolean variables. For all boolean inputs $(x_1, ..., x_m)$, it outputs 1 if and only if the statement of the label holds. We give a more precise definition as follows.

Definition 3.1. Let $x_1, ..., x_n$ be boolean variables. Let $w_1, ..., w_n, t$ be real numbers. We define a *threshold gate* as a gate computing a boolean function $THR_{w_1,...,w_n,t}(x_1, ..., x_n)$ such that $THR_{w_1,...,w_n,t}(x_1, ..., x_n) = 1 \iff \Sigma_{i=1}^{n} w_ix_i \geq t$. A *depth two threshold circuit* is a circuit which has two layers of threshold gates: the top gate and bottom gates. We assume that there may be some wire from an input variable to the top gate, and we call such wire a direct wire.

Definition 3.2.
(1) Two gates G_1, G_2 at the bottom level have *dependency*, if $\forall x \in \{0,1\}^n[G_1(x) \leq G_2(x)] \vee \forall x \in \{0,1\}^n[G_2(x) \leq G_1(x)]$, where n is the number of input variables. In other words, one of two preimages $G_1^{-1}(1), G_2^{-1}(1)$ is a subset of the other one.
(2) A subset of bottom gates is called *independent gate set*, if any two gates in the set do not have dependency. A circuit may contain several independent gate sets.

Definition 3.3. A depth two threshold circuit C is *sparse*, if $\sum_{G \in \mathcal{B}}$ fan-in of $G \leq dn$, where d is a constant and $\mathcal{B}$ is the set of all bottom level gates in C.

We define an extension of this notion which involves dependency in circuits.

Definition 3.4. A depth two threshold circuit C is *sparse in independent gates*, if there exists some constant d for an arbitrary independent gate sets I in C (at the bottom level) $\sum_{G \in I}$ fan-in of $G \leq dn$. The constant d is called a *sparse constant* of C.

3.2 Problems We Consider

Let's consider the following parameterized circuit SAT problems.

Definition 3.5.
Name of Problem: k-THR-SAT
Given: Depth two threshold circuit C of size n^c which is sparse in independent gates, where c is a constant.
Parameter: k : Maximum size of independent gate sets of C which may depend on the number of input variables n.
Compute: YES iff C is satisfiable.

Definition 3.6.
Name of Problem: k-THR-SAT with unique exception
Given: Depth two threshold circuit C of size n^c which is sparse in independent gates such that there exists an unique *maximal* independent gate set I' of size greater than k, where c is a constant.
Parameter: k: Maximum size of independent gate sets of C except I'.
Compute: YES iff C is satisfiable.

Note that we obtain an instance of k-THR-SAT by eliminating all gates in I' and wires connecting to them from C.

3.2.1 Motivation of our setting

In this section, we describe several facts on circuits with high dependency. We think that these explain why the investigation of threshold circuits parameterized by its dependency is interesting. One may think that circuits with bounded size independent gate sets seems to have very weak computational ability and seems to compute only boolean functions in a narrow class. However, the class of depth two threshold circuits is universal, even if there is *no* pair of independent gates.

Let k-THR be a layer of threshold gates whose maximum independent gate set size is k, and let $\mathsf{THR} \circ k$-THR denote the class of depth two circuits where the top gate is an arbitrary threshold gate and the bottom level is k-THR.

It is clear that the class $\mathsf{THR} \circ k\text{-}\mathsf{THR}$ with $k = 2^n$ can compute all boolean functions by emulating DNF formulas. A bit surprisingly, $\mathsf{THR} \circ k\text{-}\mathsf{THR}$ is universal even for $k = 1$. We describe below the construction of such a circuit for an arbitrary given function.

Let $f(x_{n-1}, \ldots, x_1, x_0)$ be a boolean function on n variables. For simplicity, we assume $f(0, 0, \ldots, 0) = 0$. For $0 \le j \le 2^n - 1$, let y_j denote the binary representation of j of length n, i.e., $y_j := (x_{n-1}, \ldots, x_1, x_0)$ with $\sum_{i=0}^{n-1} x_i 2^i$. Let G_j be the threshold gate whose output is 1 iff $\sum_{i=0}^{n-1} 2^i x_i \ge j$. The bottom level of a circuit is consisting of $\mathcal{G} = \{G_j \mid f(y_j) \ne f(y_{j-1}) \ (1 \le y \le 2^n - 1)\}$. Obviously, there is no pair of independent gates in $\mathcal{G}$. The top gate outputs 1 iff $\sum_{G_j \in \mathcal{G}} w_j G_j \ge 1$ where the weight w_j is $f(y_j) - f(y_{j-1})$ which is 1 or -1. In fact, the value of f is *equal* to $\sum_{G_j \in \mathcal{G}} w_j G_j$. We note that the top gate can be replaced by a symmetric gate (i.e., a gate whose output depends only on the sum of its inputs) that outputs 1 iff $\sum_{G_j \in \mathcal{G}} G_j$ is odd. This says that $\mathsf{SYM} \circ k\text{-}\mathsf{THR}$ is also universal.

We see by these examples that limiting dependency affects not the universality but the complexity, i.e., the number of gates or wires in a circuit. As was described in Introduction, for every $\delta < 1$, the existence of $2^{\delta n}$ time algorithm for k-THR-SAT of superlinear size for $k = \omega(n)$ would refute SETH. It is clear from the definition that the class of functions that can be computed by $\mathsf{THR} \circ k\text{-}\mathsf{THR}$ circuits of size $s(n)$ contains the one computed by $\mathsf{THR} \circ k'\text{-}\mathsf{THR}$ circuits of the same size for every $k' \le k$. Hence it is interesting to see the largest value of k such that k-THR-SAT admits an algorithm with constant savings as well as to study how the time complexity of circuit satisfiability problem varies as the dependency of input circuits does.

3.3 Results on Satisfiability Algorithms

We show the following main theorem, which is about a construction of nontrivial satisfiability algorithm for depth two threshold circuits with bounded size of independent gate sets.

Theorem 3.7. There is a satisfiability algorithm for k-THR-SAT with unique exception that runs in time $O(2^{(1-s)n})$, where $s = 1/d^{O(d^2)}$, and $k \le n^{\gamma}$ for an arbitrary real constant $0 < \gamma < 1$ and d is a sparse constant of a given circuit.

In the rest of the sections, our main goal is to prove the following Lemma 3.8 and we obtain Theorem 3.7 from Lemma 3.8.

Lemma 3.8. There is a randomized satisfiability algorithm for k-THR-SAT with unique exception *in which all random bits are created by independently tossing a coin*, and the algorithm runs in time $O(2^{(1-s)n})$, where $E[s] = 1/d^{O(d^2)}$, and $k \le n^{\gamma}$ for an arbitrary real constant $0 < \gamma < 1$ and d is a sparse constant of a given circuit.

In what follows, we give a way to obtain Theorem 3.7 from Lemma 3.8. A way to get a deterministic algorithm from a two sided error algorithm with error probability at most 1/3 is given as follows. This method is generally called the conditional expectation method (pessimistic estimator).

We consider a randomized algorithm that uses m random bits. We can regard all its sequences of coin tosses as corresponding to a binary tree of depth m. We know that most paths from the root to the leaf are *good*, that is, give a correct answer. It is natural and simple thought to try and find such a path by walking down from the root and making *good* choices at each step. Equivalently, we try to find a good sequence of coin flips with considering each *single bit*.

We consider formally this intuition. Fix a randomized algorithm A and an input x, and let m be the number of random bits used by A on input x. For $1 \leq i \leq m$ and $r_1, r_2, ..., r_m \in \{0, 1\}$, we define $P(r_1, ..., r_i)$ as the fraction of continuations of a randomized computation that are good sequences of coin tosses. A precise definition is as follows: if $R_1, ..., R_m$ are uniform and independent random bits, then $P(r_1, ..., r_i)$ is defined as $\underset{R_1,...,R_m}{Pr}$ $[A(x, R_1, ..., R_m)$ is correct $\mid \bigwedge_{j=1}^{i}$ "$R_j = r_j$"$] = \underset{R_{i+1}}{E}[P(r_1, ..., r_i, R_{i+1})]$.

By averaging argument, there exists an $r_{i+1} \in \{0, 1\}$ such that $P(r_1, ..., r_i, r_{i+1}) \geq P(r_1, r_2, ..., r_i)$. Thus, at node $(r_1, ..., r_i)$, *we pick* r_{i+1} *which maximizes* $P(r_1, ..., r_{i+1})$. Finally, we obtain $r_1, ..., r_m$ such that

$$\begin{aligned} P(r_1, r_2, ..., r_m) &\geq P(r_1, r_2, ..., r_{m-1}) \\ &\vdots \\ &\geq P(r_1) \\ &\geq P \geq 2/3, \end{aligned}$$

where P is the fraction of good path from the root. Therefore, we have $P(r_1, ..., r_m) = 1$, because $P(r_1, ..., r_m)$ is either 0 or 1.

For an implementation of this argument, we just construct a deterministic algorithm to compute $P(r_1, r_2, ..., r_i)$ for each i. Note that if we show an algorithm in which all random bits are created by independently tossing a biased coin then an implementation is given. By using Chernoff bound, we can construct a randomized algorithm which repeats the algorithm in Lemma 3.8 constant times and runs with error probability at most 1/3. Thus, using the conditional expectation method, we obtain Theorem 3.7 by repeating the algorithm in Lemma 3.8 constant times.

Note that the statement of Theorem 3.7 improves the following previous result by Impagliazzo et al. [25] in the sense that we can construct a nontrivial algorithm when we relax some condition on sparsity in input circuits.

Theorem 3.9 ([25])**.** There is a depth two threshold circuit SAT algorithm with n variables and dn wires that runs in time $O(2^{(1-s)n})$, where $s = 1/d^{O(d^2)}$ and d is an arbitrary constant.

We first give a rough and qualitative sketch of the outline of the algorithm in [25]. For given depth two sparse threshold circuits, they give three reductions: the first one transforms an arbitrary ILP instance with small number of inequalities to an instance of vector domination problem, and the second one transforms any depth two circuit with small number of gates to an union of ILP instances with small number of inequalities, and the third one transforms a depth two threshold circuit with linear number of wires to an union of depth two circuits with small number of gates. Constructing an algorithm with constant savings in the exponent, we can test satisfiability of depth two sparse threshold circuits with our setting. Our meaning of *small number* is the one that the number of gates or inequalities is less than the number of variables.

The random restriction technique is used to construct these reduction procedures. The second reduction uses restrictions to output wires of bottom gates in depth two threshold circuits with small number of gates. The third reduction uses restrictions to input variables, and we can decrease the number of bottom gates because of this restriction. For each restriction to output wires of bottom gates we obtain an ILP instance with small number of inequalities by the second reduction, and for each restriction to input variables we obtain a depth two circuit with small number of gates by the third one.

Restricting to the output wire of a bottom gate means obtaining a linear inequality whose variables and coefficients and the threshold value agree with the label of the bottom gate. Let's consider when we obtain a depth two threshold circuit with small number of gates. The number of brute force restrictions to bottom gates of which the number is less than the number of variables is still less than the number of brute force restrictions to input variables. Because of this saving, we can constantly save the complexity of the exponent of the running time.

Restricting variables for the third reduction involves a little technical argument. We take a random subset of variables and assign a boolean string to these variables and let the other unchosen variables remaining. When for a random subset of input variables these variables are fixed, we consider the following two cases for an arbitrary bottom gate. In the first case the gate has at most one unfixed fan-in. In the second case the gate has at least two unfixed input fan-ins. In the former case, such kind of gates do not cause any trouble for the reduction, because we can eliminate these gates and decrease the number of bottom gates. In the latter case, however, we cannot take such straight forward argument. It is the sparsity of circuits that gives the nice property that there are not so many such bad gates.

We mention how we obtain an extension of [25] from our setting. In our setting, the number of restrictions to bottom level gates is bounded above because of dependency of bottom gates. We first define some partial order on the set of bottom gates. Hasse diagrams of this relation are useful to formalize the notion of dependency of bottom gates in a circuit. Next, we define a mid-point problem: for given a pair of a circuit and a Hasse diagram relating with the circuit, output YES if and only if the circuit is satisfiable. Our main subroutine is a randomized algorithm solving this problem. Because of an upper bound on the expected number of restrictions to bottom level gates, the running time of the randomized algorithm is faster than the complexity of the trivial exhaustive search. In other words, the expected exponent of the running time is faster than the one of the trivial exhaustive search. Our main subgoal is to obtain an upper bound on the expected exponent of the running time of a randomized algorithm to check satisfiability. We show several lemmas about the bounds on the number of restrictions. We finally design a randomized algorithm contains several subroutines: the reduction procedure from satisfiability problem for given depth two threshold circuits to the mid-point problem and the algorithm solving an intermediate problem. We obtain a deterministic algorithm by repeating this randomized algorithm constant times, using the method called the conditional expectation method.

3.4 An Overview of the Entire Algorithm in Lemma 3.8

3.4.1 Partial order on bottom gates

We express structures on dependency of bottom level gates using directed graphs. We first introduce a partial order representing the dependency of threshold gates.

Definition 3.10. Let C be a depth two threshold circuit. The binary relation $\preceq$ on the set of bottom level gates of C is defined as follows: $G_1 \preceq G_2 \underset{\text{def}}{\iff} G_1^{-1}(1) \subseteq G_2^{-1}(1)$ for all $G_1, G_2 \in \mathcal{G}$, where $\mathcal{G}$ is the set of bottom gates.

We define a problem using the partial order stated in the above.

Definition 3.11.
Name of Problem: L'
Given:$\langle C, H \rangle$ satisfying the following conditions.

- C is an instance of k-THR-SAT with unique exception
- There is *no* pair of gates G_1, G_2 s.t. $G_1^{-1}(1) = G_2^{-1}(1)$
- H is a Hasse diagram of a partial ordered set of bottom level gates in C according to the order $\preceq$.

Output:YES iff C is satisfiable

We will give a procedure for the problem L' and this procedure is a critical subroutine of the algorithm constructed in Lemma 3.8. The algorithm in Lemma 3.8 first transforms a given instance of k-THR-SAT to an instance of the problem L', and then solves the satisfiability problem using some structure of Hasse diagram.

The first reduction is given by the following lemma.

Lemma 3.12. There is a reduction which reduces k-THR-SAT with unique exception to the problem L' defined above and the reduction runs in deterministic time $O(poly(n)T_{ZOLP}[2, n])$, where $T_{ZOLP}[m, n]$ is the time complexity of the 0-1 Linear Programming with m constraints and n variables.

Roughly speaking, the reduction described in this lemma generates a Hasse diagram by checking the dependency of every pair of bottom gates by solving an ILP with two constraints. The proof of the lemma is postponed to the next section.

3.4.2 Restriction to the bottom gates and reduction to ILP

Intuitively, when the dependency of a circuit C is limited, the output of C is determined by fixing the output of a small number of bottom gates. By using this property, we can build a set $\mathcal{S}$ of small number of ILP instances such that C is satisfiable iff at least one instance in $\mathcal{S}$ is feasible and each instance has a small number of constraints. In fact, our algorithm solves k-THR-SAT by solving such set of ILP instances. The following lemma whose proof is postponed states this more formally.

Definition 3.13. For a depth two threshold circuit C, the set $X(C)$ is defined as $\{(y_1, ..., y_{n^c}) \in \{0,1\}^{n^c} : y_i$ is the output of the i-th bottom gate in C, when C runs for an arbitrary input $x \in \{0,1\}^n\}$.

Lemma 3.14. Let C be an instance of k-THR-SAT with unique exception. There is a set $\mathcal{S}$ of ILP instances with n variables satisfying the following three conditions: **(1)** It holds that $C^{-1}(1) = \bigcup_{S\in\mathcal{S}} F(S)$, where $F(S)$ is the set of feasible solutions of $S \in \mathcal{S}$, **(2)** the set $\mathcal{S}$ contains at most $|X(C)|$ ILP instances and **(3)** each instance in $\mathcal{S}$ has at most $2k + |I'|$ constraints, where I' is unique exceptional independent gate set in C.

We will use this reduction in the main algorithm which will be described in another section.

3.4.3 The entire overview

The construction of an algorithm in Lemma 3.8 is as follows:
1. Call the reduction procedure in Lemma 3.12 to transform the given instance of k-THR-SAT with unique exception to an instance of the problem L'.
2. Find the exceptional unique independent set I'.
3. Run the main algorithm on the input $\langle C, H\rangle$ and I'.

We note that we can find I' in step 2. as follows. First, let a positive integer l be 1, and repeat increasing l by one until there uniquely exist an independent set of size l. Next, search the unique maximal independent set of size greater than l. Note that the repeating process stops in at most k steps. In step 3, restriction methods to both input variables and outputs of bottom gates and the reduction to ILP in Lemma 3.14 are used.

3.5 Partial Order in Circuits and Reduction Lemma

At first we give the following lemma on a binary relation on the set of bottom gates of a depth two threshold circuit mentioned in the previous section.

Lemma 3.15. Assume that C is an instance of k-THR-SAT with unique exception, and that there is *no* two gates G_1, G_2 in C such that $G_1^{-1}(1) = G_2^{-1}(1)$. Then, there exists some partial ordered set $(\mathcal{G}, \preceq)$, where $\mathcal{G}$ is a set of bottom gates of C such that the maximum size of an independent set of Hasse diagram H of $(\mathcal{G}, \preceq)$ is k.

Proof First, we prove the existence of a partial ordered set. For any instance of k-THR-SAT with unique exception the following holds. If there is *no* two equivalent gates at the bottom level in the instance, then there is a partial ordered set $(\mathcal{G}, \preceq)$ such that for all $G_1, G_2 \in \mathcal{G}$, $G_1 \preceq G_2 \iff G_1^{-1}(1) \subseteq G_2^{-1}(1)$, where $\mathcal{G}$ is a set of bottom gates. To prove this statement we show that the relation $\preceq$ is *reflective*, *asymmetric* and *transitive*.

It is clear that $G_1^{-1}(1) \subseteq G_1^{-1}(1)$ and that $G_1^{-1}(1) \subseteq G_2^{-1}(1) \wedge G_2^{-1}(1) \subseteq G_3^{-1}(1) \Rightarrow G_1^{-1}(1) \subseteq G_3^{-1}(1)$ for all gates G_1, G_2, G_3. Thus the relation is reflective and transitive.

Finally, $G_i^{-1}(1) \subseteq G_j^{-1}(1) \wedge G_j^{-1}(1) \subseteq G_i^{-1}(1) \Rightarrow i = j$, because there is *no* pair of two gates which is equivalent. Thus the relation is asymmetric.

Next we argue the maximum size of independent sets in a Hasse diagram. Let $H = (V, E)$ be a Hasse diagram of C stated above. It holds that $G_i^{-1}(1) \neq G_j^{-1}(1)$ if and only if either **(1)** $G_i^{-1}(1) \subsetneq G_j^{-1}(1)$ or **(2)** $G_j^{-1}(1) \subsetneq G_i^{-1}(1)$ or **(3)** $G_i^{-1}(1) \setminus G_j^{-1}(1) \neq \emptyset \wedge G_j^{-1}(1) \setminus G_i^{-1}(1) \neq \emptyset$. Thus, for any bottom gates G, G', the following three conditions are equivalent.
(i) $\neg(G \preceq G') \wedge \neg(G' \preceq G)$.
(ii) $(G, G') \notin E \wedge (G', G) \notin E$.
(iii) G and G' do not have dependency.
Hence, an arbitrary maximum independent set in H corresponds to some maximum independent gate set in C by the definition of $\preceq$. □

In the rest of this section, we give the proof of the lemma describing the reduction from k-THR-SAT to the problem L'.

Lemma 3.12 (restated) There is a reduction which reduces k-THR-SAT with unique exception to the problem L' defined above and the reduction runs in deterministic time $O(poly(n)T_{ZOLP}[2, n])$, where $T_{ZOLP}[m, n]$ is the time complexity of the 0-1 Linear Programming with m constraints and n variables.

Proof Let $P(x), Q(x)$ be constraints depending on $x = (x_1, ..., x_n) \in \{0, 1\}^n$. Note that $\forall x[P(x) \Rightarrow Q(x)]$ is equivalent to $\forall x[\neg P(x) \vee Q(x)]$ and is equivalent to $\neg[\exists x[P(x) \wedge \neg Q(x)]]$ and that $\exists x[P(x) \wedge \neg Q(x)]$ is the YES-condition of the Integer Linear Programming. Thus we have the following procedure for the reduction. In this procedure, the 0-1 Linear Programming with two constraints and n variables is solved in step 4. and the other steps are computed in polynomial time. Thus, the running time of this procedure is $O(poly(n)T_{ZOLP}[2, n])$. □

The reduction procedure

1. Let $V = \{G_1, ..., G_{n^c}\}$ be a set of bottom gates and let D be $\emptyset$.
2. For all pairs of bottom level gates G, G' do the following steps 3.,4.,5.
    3. Let the labels of G and G' be $\sum_{i \in S} a_i x_i \geq b$ and $\sum_{i \in S'} a'_i x_i \geq b'$ respectively, where S, S' are the sets of indices of variables connecting to G, G' respectively.

4. Solve the following instances of Integer 0-1 Linear Programming with *two* constraints, that is,
   (1) $\sum_{i\in S} a_i x_i \geq b$ and (2) $\sum_{i\in S'} a'_i x_i < b'$.
5. If there does *not* exist $(x_1, ..., x_n) \in \{0,1\}^n$ satisfying the constraints (1) and (2) then $D := D \cup \{(G, G')\}$(because of $G^{-1}(1) \subseteq G'^{-1}(1)$)

6. For all G, G' such that $(G, G') \in D \wedge (G', G) \in D$ do the following steps 7., 8.

   7. for each bottom gate U let y_U be the output wire of U. The label $w_G y_G + w_{G'} y_{G'} + \sum_{U \neq G,G'} w_U y_U \geq t_{TOP}$ in the TOP gate is replaced with $(w_G + w_{G'}) y_G + \sum_{U \neq G,G'} w_U y_U \geq t_{TOP}$.
   8. G' and all input and output wires of G' are removed from C. Replace V with the set of bottom gates in C. For any bottom gate U, if $(U, G') \in D \vee (G', U) \in D$ then $D := D \setminus \{(U, G'), (G', U)\}$.

9. Output the result $\langle C, H = (V, D) \rangle$.

3.6 Main Algorithm

At first we recap a sketch of the outline of the algorithm in [25]. Remind three reductions for given depth two sparse threshold circuits: the first one transforms an arbitrary ILP instance with small number of inequalities to an instance of vector domination problem, and the second one transforms any depth two circuit with small number of gates to an union of ILP instances with small number of inequalities, and the third one transforms given instance to an union of depth two circuits with small number of gates.

We show several lemmas about the bounds of the number of restrictions. We first define several terms being necessary for formal statements of these lemmas.

Definition 3.16. A directed graph $H = (V, E)$ is an *Induced Hasse Diagram* (abbreviated I.H.D) of a circuit C which is an instance of k-THR-SAT with unique exception, if H is the output $\langle C, H \rangle$ of the procedure in the proof of Lemma 3.12.

Definition 3.17.
(1) Let V be a set of gates and let $H = (V, E)$ be I.H.D of V. A coloring $\chi : V \mapsto \{0, 1\}$ is called a *validly ordered restriction*, if $\forall (u, v) \in E$, $\chi(u) \leq \chi(v)$.
(2) Let χ be a validly ordered restriction for an arbitrary I.H.D $H = (V, E)$.
We define the *min-set of* H *for* χ as the set $\{u_{min} \in V \cap \chi^{-1}(1) : \forall v \in V \setminus \{u_{min}\}[v \preceq u_{min} \Rightarrow \chi(v) = 0]\}$.
We define the *max-set of* H *for* χ as the set $\{u_{max} \in V \cap \chi^{-1}(0) : \forall v \in V \setminus \{u_{max}\}[u_{max} \preceq v \Rightarrow \chi(v) = 1]\}$.

Definition 3.18. Let H be an I.H.D and χ be a validly ordered restriction of H. Let I_1, I_0 be independent sets in H. The pair of independent sets (I_1, I_0) satisfies the *covering condition* for H, if the following condition holds.
Condition: For any $v \in V \setminus (I_1 \cup I_0)$ in H, either $\exists u_1 \in I_1, u_1 \preceq v$ or $\exists u_0 \in I_0, v \preceq u_0$ according to the order $\preceq$ of H.

We count the number of validly ordered restrictions, and a lemma in this section is about an upper bound on the number of these restrictions. Bottom gates in min-set or max-set are critical to design our algorithm for satisfiability. Satisfiability of a circuit depends on information about bottom gates which are in min-set or max-set of the circuit, when output of bottom gates are fixed. In other words, we can decide satisfiability, even if we consider only some local information about bottom gates of a circuit and ignore the other gates. The covering condition is a condition stating the concept of min-set and max-set from another viewpoint, and is used in our algorithm in this section.

Definition 3.19. Let X'_H be a set of validly ordered restrictions of H. We define $\mathcal{I}_H$ as a set of pairs of independent sets in H which satisfies the covering condition, that is, $\mathcal{I}_H := \{(I_1, I_0) \subseteq V \times V : I_1, I_0 \text{ are independent sets satisfying the covering condition in } H\}$.

Definition 3.20. Let c be a constant. For a depth two threshold circuit C, the set $X(C)$ is defined as $\{(y_1, ..., y_{n^c}) \in \{0,1\}^{n^c} : y_i$ is the output of the i-th bottom gate in C, when C runs for an arbitrary input $x \in \{0,1\}^n\}$.

The following lemma is a main lemma of this section, and rough meaning of this lemma is that we can construct a satisfiability algorithm using exhaustive search for all independent gate sets.

Lemma 3.21. For an arbitrary instance $\langle C, H \rangle \in L'$, let I' be the unique maximal independent set of size greater than k. Then, $|X(C)| \leq 2^{|I'|} k^2 n^{O(k)}$.

We first show the following lemma to prove Lemma 3.21, which reduces counting the number of restrictions for a circuit to counting the number of structures in a graph.

Lemma 3.22. There is a bijection $\mu_H : X'_H \ni \chi \mapsto (I_1, I_0) \in \mathcal{I}_H$ such that if $\mu_H(\chi) = (I_1, I_0)$ then I_1 is the min-set of H for χ and I_0 is the max-set of H for χ.

First, we show two claims. The lemma easily follows from these claims.

Claim 3.23. Let H be an I.H.D and χ be a validly ordered restriction of H. Let I_1, I_0 be min-set and max-set of H for χ respectively. Then (I_1, I_0) is a pair of independent sets such that the covering condition holds for H.

Proof. We give a proof by contradiction.

For any fixed χ which assign 0 or 1 to vertices of H and for min-set I_0 and max-set I_1 of H for χ, adding all edges which is in $I_0 \times I_1$ preserves validity of χ. In other words, after adding all edges which is in $I_0 \times I_1$, χ is still a valid ordered restriction. Let $H' = (V, E')$ be this I.H.D which is *obtained by adding edges to H*. Thus I_0, I_1 are maximal independent sets in H'.

Assume that I_0, I_1 do not satisfy the covering condition. Then either **case1** or **case2** holds for H'.

case1. $\exists u_1 \in I_1, u_1 \preceq v$ and $\exists u_0 \in I_0, v \preceq u_0$, for some $v \in V \setminus (I_1 \cup I_0)$ in H'.

In this case, $u_1 \preceq u_0$ contradicts to the assumption that $u_1 \in I_1, u_0 \in I_0$.

case2. $\forall u_1 \in I_1, \neg(u_1 \preceq v)$ and $\forall u_0 \in I_0, \neg(v \preceq u_0)$, for some $v \in V \setminus (I_1 \cup I_0)$ in H'.

In this case, since I_1, I_0 are maximal independent sets, for all $v' \in V \setminus (I_1 \cup I_0)$ it holds that $\forall u_1 \in I_1, \neg(u_1 \preceq v') \Rightarrow v' \preceq u_1$ and $\forall u_0 \in I_0, \neg(v' \preceq u_0) \Rightarrow u_0 \preceq v'$. Thus we obtain that there exists a vertex $v \in V \setminus (I_0 \cup I_1)$ such that $u_0 \preceq v \preceq u_1$, contradicting to $u_1 \in I_1$ and $u_0 \in I_0$. In other words, for any $c \in \{0, 1\}$ we obtain that $\chi(v) = c$ contradicts to $u_c \in I_c$. ☐

Claim 3.24. For any (I_1, I_0) which is a pair of independent sets in H satisfying the covering condition, there uniquely exists validly ordered restriction $\chi : V \to \{0, 1\}$ such that I_1, I_0 are min-set and max-set of H for χ, respectively.

Proof. For any validly ordered restriction χ it holds that I_1 is max-set implies $\chi(I_1) = \{1\}$, and I_0 is min-set implies $\chi(I_0) = \{0\}$. First, we consider vertices in $I_0 \cup I_1$. Since I_1, I_0 are min-set and max-set of H for χ respectively, assign zero-one value to vertices in $I_1 \cup I_0$ such that $\chi(I_1) = \{1\}$ and $\chi(I_0) = \{0\}$. Finally we consider the other vertices. For any $v \in V \setminus (I_1 \cup I_0)$, the value $\chi(v)$ is uniquely determined by the definition of covering condition. ☐

Proof of Lemma 3.22. For any $\chi \in X'_H$, there uniquely exists $(I_1, I_0) \subseteq V \times V$ such that I_1 is the min-set of H for χ and I_0 is the max-set of H for χ. By Claim 3.23, there is a map $\mu_H : X'_H \to \mathcal{I}_H, \mu_H(\chi) = (I_0, I_1)$ such that if $\mu_H(\chi) = (I_1, I_0)$ then I_1 is the min-set of H for χ and I_0 is the max-set of H for χ. By Claim 3.24, this map μ_H is a bijective map. ☐

Proof of Lemma 3.21. Fix an assignment $\chi|_{I'} : I' \to \{0, 1\}$. Let $H = (V, E)$.

We assign 1 to any vertex $u \in V$ such that there is some vertex $v \in \chi|_{I'}^{-1}(1) \subseteq I'$ such that $v \preceq u$. We assign 0 to any vertex $u \in V$ such that there is some vertex $v \in \chi|_{I'}^{-1}(0) \subseteq I'$ such that $u \preceq v$.

Remove all vertices *whose assignment is fixed* and all edges connecting to them. Thus we obtain a subgraph H' of H such that size of any independent vertex set in H' is at most k because we assume the existence of unique exception. The output pattern set $X(C)$ is a subset of validly ordered restrictions of H.

By Lemma 3.22 there is a bijective map $\mu_{H'} : X'_{H'} \to \mathcal{I}_{H'}$. Thus for any H' the cardinality $|X'_{H'}|$ is bounded above by $|\mathcal{I}_{H'}|$. Since $|X(C)| \leq 2^{|I'|} \max_{H'} |X'_{H'}|$ and $|\mathcal{I}_{H'}| \leq \left(\sum_{i=1}^{k} \binom{n^c}{i}\right)^2$ $\leq \left(\sum_{i=1}^{k} n^{ci}\right)^2 \leq \left(kn^{ck}\right)^2 = k^2 n^{O(k)}$, we obtain the desired bound : $|X(C)| \leq 2^{|I'|} k^2 n^{O(k)}$. ☐

The following lemma is similar to a part of work in [25], which gives a reduction from an instance of circuit satisfiability to a union of ILPs.

Lemma 3.14(restated) Let C be an instance of k-THR-SAT with unique exception. There is a set $\mathcal{S}$ of ILP instances with n variables satisfying the following three conditions: **(1)** It holds that $C^{-1}(1) = \bigcup_{S \in \mathcal{S}} F(S)$, where $F(S)$ is the set of feasible solutions of $S \in \mathcal{S}$, **(2)** the set $\mathcal{S}$ contains at most $|X(C)|$ ILP instances and **(3)** each instance in $\mathcal{S}$ has at most $2k + |I'|$ constraints, where I' is unique exceptional independent gate set in C.

Proof. For any circuit C and each element in $X(C)$, we obtain the following transformation from a circuit to an ILP instance, according to the following three kinds of gates.
(i) For gates whose output is fixed to 1 with weights $w_1, ..., w_n$ and threshold t, we have $\sum\limits_{i=1}^{n} w_i x_i \geq t$.
(ii) For gates whose output gate is fixed to 0 we require $\sum\limits_{i=1}^{n} w_i x_i < t$, which is equivalent to $\sum\limits_{i=1}^{n} -w_i x_i \geq -t + \min_i w_i$.
(iii) For the top gate, let $v_1, ..., v_n$ be the weights of the direct wires, and s be the threshold of the top level gate, and w_{FIX} be the sum of the weights of the gates whose output is fixed to 1. Then we require $\sum\limits_{i=1}^{n} v_i x_i \geq s - w_{FIX}$.

Thus, the set of these instances satisfies the conditions **(1)** and **(2)**. Moreover, the following observation and the definition of min-set and max-set implies that the dependency of gates in C gives at most $2k + |I'|$ constraints, and yields the existence of a set $\mathcal{S}$ of ILP instances with n variables satisfying the condition **(3)**, because it holds that for each restriction to gates in I', the circuit C' which is obtained by removing all gates in I' and all wires connecting to them from C has min-set and max-set of size at most k. $\square$

Observation 3.25. For boolean functions $P_1(x), ..., P_m(x) : \{0,1\}^n \to \{0,1\}$, the following statements hold.
(1) If $\forall x \in \{0,1\}^n, P_1(x) = 1 \Rightarrow P_2(x) = 1 \Rightarrow \cdots \Rightarrow P_m(x) = 1$ then, it holds that $\forall x \in \{0,1\}^n, P_1(x) = 1 \land P_2(x) = 1 \land \cdots \land P_m(x) = 1 \iff \forall x \in \{0,1\}^n, P_1(x) = 1$.
(2) If $\forall x \in \{0,1\}^n, P_m(x) = 0 \Rightarrow P_{m-1}(x) = 0 \Rightarrow \cdots \Rightarrow P_1(x) = 0$ then, it holds that $\forall x \in \{0,1\}^n, P_m(x) = 0 \land P_{m-1}(x) = 0 \land \cdots \land P_1(x) = 0 \iff \forall x \in \{0,1\}^n, P_m(x) = 0$.

When for a random subset of input variables these variables are fixed, we consider the following two cases for an arbitrary bottom gate. The gate has at most one unfixed input wire in one case, and the gate has at least two unfixed input wires in the other case. In the former case, such gates are not harmful for our argument about the reduction, because we can eliminate these gates and decrease the number of bottom gates. In the latter case, however, we cannot use such straightforward argument. We define more precisely such gates to which we cannot directly apply gate elimination argument.

Definition 3.26. BAD gate
Let U be a subset of variables. BAD gate on U is a bottom level gate that depends on at least two variables in U.

In other words, if a gate is *not* BAD then we can eliminate it or replace it with a direct wire.

Lemma 3.27 ([25]). Consider a depth two threshold circuit with n variables and dn wires. Let $\delta > 0$ be an arbitrary positive real number and let $\tilde{U}$ be a random set of variables such that each variable is not in $\tilde{U}$ with some probability p independently. There exists a $p = 1/d^{O(d^2)}$ such that *the expected number of BAD gates on* $\tilde{U}$ is at most $3\delta pn$, where d is a sparse constant.

The following corollary is easily obtained from this lemma.

Corollary 3.28 ([25]). Consider a depth two threshold circuit C with n variables, which is a part of $\langle C, H\rangle \in L'$. Let $VAR[I']$ be the set of variables connecting to the gates corresponding to I', which is unique exceptional independent gate set in C. Let $\delta > 0$ be an arbitrary positive real number and let $\tilde{U}$ be a random set of variables such that each variable is not in $\tilde{U}$ with some probability p independently. Let $I'_{\tilde{U}}$ be a subset of I' which are also BAD gates on $\tilde{U}$. There exists a $p = 1/d^{O(d^2)}$ such that $E[|I'_{\tilde{U}}|] \leq 3\delta p|VAR[I']|$, where d is a sparse constant.

Let $C|_{\rho[\tilde{U}]}$ be a circuit obtained by the operation for a circuit C that all variables in $\tilde{U}$ is fixed to an arbitrary assignment $\rho[\tilde{U}] : \tilde{U} \to \{0, 1\}$ and any gate, which is not BAD, is eliminated from C or replaced with direct wires in C.

Corollary 3.29. Consider a depth two threshold circuit with n variables, which is a part of $\langle C, H\rangle \in L'$. Let $\delta > 0$ be an arbitrary positive real number and let $\tilde{U}$ be a random set of variables such that each variable is not in $\tilde{U}$ with some probability $p = 1/d^{O(d^2)}$, where d is a sparse constant. Then, it holds that $E[\log |X(C|_{\rho[\tilde{U}]})|] \leq 3\delta pn + O(k \log_2 n + \log_2 k)$, for any assignment $\rho[\tilde{U}] : \tilde{U} \to \{0, 1\}$.

Proof. By Lemma 3.27, we obtain that $\log |X(C|_{\rho[\tilde{U}]})| \leq |I_{\tilde{U}'}| + O(k \log_2 n + \log_2 k)$. Hence Corrary 3.28 and linearity of expectation give us the desired bound. □

Finally we consider an algorithm to solve L' under given the unique exception of each instance. Let $VAR[I']$ be a set of variables connecting to the gates corresponding to I'.

We give a description of the main algorithm as follows, using for all above ingredients. Note that all random bits are created by tossing a coin independently in the following algorithm.

Description of the main algorithm
Given: An instance $\langle C, H\rangle$, and the exceptional unique independent set I' in H.
Output:YES if and only if $\langle C, H\rangle$ is a YES instance of L'.

1. Choose a random subset $\tilde{U} \subset VAR[I']$ such that each variable is *not* in $\tilde{U}$ with probability p independently.

2. For each boolean assignment to $\tilde{U}$, fix the value of input variables in $\tilde{U}$, and do the following steps 3. and 4.

    3. Eliminate any gate in I' whose output is totally fixed. Replace any gate whose output value is the value of some input variable x with direct wire connecting to x. Let V_D be the set of variable indices directly connecting to the top gate. Let $D_i (i \in V_D)$ be the sum of weights at the top gate such that these weights are coefficients of outputs of bottom gates replaced with direct wires connecting to the i-th variable.

    4. For each restriction μ to outputs of *remaining* bottom gates in I' do the following steps 5., 6., and 7.

        5. Assign 0 to the output of an *arbitrary bottom* gate G, if there is some $G' \in I'$ such that the output of G' is fixed to 0 and $G \preceq G'$. Assign 1 to the output of an *arbitrary bottom* gate G, if there is some $G' \in I'$ such that the output of G' is fixed to 1 and $G' \preceq G$.

        6. Remove all gates whose outputs are totally fixed from the given circuit which is a part of given instance of L'. Let H' be an I.H.D. which is obtained from H by this removing operation.

        7. Find k as follows. Let l be 1. Repeat increasing l by one until there uniquely exist an independent set of size l. Let k be l. For each pair of gate sets (I_0, I_1) in H' such that $|I_0|, |I_1| \leq k$, if (I_0, I_1) is a pair of *independent gate sets* in H' and satisfies *the covering condition*, then solve ILP for an instance which is obtained from the top gate and bottom gates in $I_0 \cup I_1 \cup I'$ and is constituted by the following three kinds of inequalities (i), (ii), and (iii). If satisfying 0-1 vector is found then HALT and return "YES".

            (i) For bottom gates whose output is fixed to 1 with weights $w_1, ..., w_n$ and threshold t, the corresponding inequality is $\sum_{i=1}^{n} w_i x_i \geq t$.

            (ii) For bottom gates whose output gate is fixed to 0, the corresponding inequality is $\sum_{i=1}^{n} -w_i x_i \geq -t + \min_i w_i$.

            (iii) For the top gate, let $v_1, ..., v_n$ be the weights of the direct wires, and s be the threshold of the top level gate, and w_{FIX} be the sum of the weights of the gates whose output is fixed to 1. Then for the top gate, the corresponding inequality is $\sum_{i=1}^{n} v_i x_i \geq s - w_{FIX}$.

8. HALT and return "NO"

Note that three kinds of inequalities in step 7. appear in the proof of Lemma 3.14 and that the iterating increment of l in this step stops in at most k steps. Note also that Lemma 3.21 which gives an upper bound on the number of restrictions in steps 2. and 4. is a key to obtain a constant saving in the exponent in the running time of our algorithm.

3.7 Analysis of the Expected Savings

In this section our goal is the following lemma about the expected savings.

Lemma 3.8(restated) There is a randomized satisfiability algorithm for k-THR-SAT with unique exception *in which all random bits are created by independently tossing a coin*, and the algorithm runs in time $O(2^{(1-s)n})$,where $E[s] = 1/d^{O(d^2)}$, and $k \leq n^{\gamma}$ for an arbitrary real constant $0 < \gamma < 1$ and d is a sparse constant of a given circuit.

We remind the construction of the algorithm in Lemma 3.8 as follows.
1. Call the reduction procedure in Lemma 3.12 to transform the given instance of k-THR-SAT with unique exception to an instance of the problem L'.
2. Find the exceptional unique independent set I'.
3. Run the main algorithm on the input $\langle C, H \rangle$ and I'.

Let $T_{\tilde{U}}(n)$ be the running time of the Main Algorithm. To consider savings of this algorithm, we only analyze the exponent $\log T_{\tilde{U}}(n)$ of the running time of the Main Algorithm, because the time complexity of entire procedure is at most $3\max\{2^{\varepsilon n}, 2^{o(n)}, T_{\tilde{U}}(n)\}$ for some positive constant $\varepsilon < 1$. Note that three quantities $2^{\varepsilon n}$, $2^{o(n)}$, and $T_{\tilde{U}}(n)$ are respectively corresponding to three steps **1.**, **2.**, and **3.** in the above algorithm and that by summing up these three terms we obtain the bound. We also note that we can find I' in step 2. by the exhaustive search for all i $(1 \leq i \leq k+1)$ and for all i-sets of the vertex set of H and searching maximal independent set of size greater than k. We use the following result.

Corollary 3.30 ([25]). Consider a 0-1 Integer Linear Program on n variables and $m(n)$ inequalities. Let λ be $m(n)/n$. Then we can find a solution in time

$$2^{n/2}\left(\binom{(1/2+\lambda)n}{\lambda n} poly(n)\right) \leq 2^{(1/2+\lambda(\log(e)+\log(1+1/2\lambda)))n} \cdot poly(n).$$

Note that this algorithm is faster than 2^n for $\lambda < 0.136$ and has some positive constant saving C' such that running time is $2^{(1-C')n}$.

Firstly we prove Lemma 3.8 assuming the following Claim.

Claim 3.31. There exists a constant N_0 for any $\delta > 0$, $E[\log T_{ZOLP}[2k + |I'_{\tilde{U}}|, |R|]] \leq 0.5pn + 3N_0\delta pn + o(n)$, where R is a set of remaining variables in the main algorithm.

Proof of Lemma 3.8 Let R be a set of remaining variables in the main algorithm. The expectation of the exponent of the time complexity is bounded above as follows.

$E[\log T_{\tilde{U}}] \leq E[\log(|\{0,1\}^{|\tilde{U}|}| \cdot |X(C_{\rho[\tilde{U}]})| \cdot T_{ZOLP}[2k + |I'_{\tilde{U}}|, |R|] \cdot poly(|R|))]$, where $|\{0,1\}^{|\tilde{U}|}|$ is the number of assignments for brute force restriction to a random subset of variables $\tilde{U}$.

We mention how each operation in the main algorithm contributes to the above expectation. We remind that there are two loops in the description of the main algorithm: the inner loop and the outer loop. Note that the term $|X(C_{\rho[\tilde{U}]})|$ corresponds to restricting procedure to bottom gates in the steps 4., 5., and 6. and that $T_{ZOLP}[2k+|I'_{\tilde{U}}|,|R|]$ corresponds to solving ILP problem in the step 7., where $|I'_{\tilde{U}}|$ is the size of the unique exception when a random subset of input variables is chosen and input variables in the subset are fixed. Thus by all these observations we obtain the above bound.

By the linearity of expectation, $E[\log|\{0,1\}^{|\tilde{U}|}|]+\log|X(C_{\rho[\tilde{U}]})|+\log T_{ZOLP}[2k+|I'_{\tilde{U}}|,|R|]] = E[\log|\{0,1\}^{|\tilde{U}|}|]+E[\log|X(C_{\rho[\tilde{U}]})|]+E[\log T_{ZOLP}[2k+|I'_{\tilde{U}}|,|R|]]$.

We show upper bounds on the above three terms. Firstly, note that $E[|\tilde{U}|]=(1-p)n$. By Corrary 3.29, we obtain the following upper bound. $E[\log|X(C|_{\rho[\tilde{U}]})|]\leq 3\delta pn+O(k\log_2 n+\log_2 k)$. By Claim 3.31, we obtain the following bound. $E[T_{ZOLP}[2k+|I'_{\tilde{U}}|,|R|])]\leq 0.5pn+3N_0\delta pn+o(n)$.

By summing up these three bounds, we obtain $E[\log|\{0,1\}^{|\tilde{U}|}|]+E[\log|X(C_{\rho[\tilde{U}]})|]+E[\log T_{ZOLP}[2k+|I'_{\tilde{U}}|,|R|]]\leq(1-p)n+0.5pn+3(N_0+1)\delta pn+o(n)$. There is some δ, which is a sufficient small constant such that for some positive constant C'' we obtain $E[\log|X(C_{\rho[\tilde{U}]})|]+E[\log T_{ZOLP}[2k+|I'_{\tilde{U}}|,|R|]]\leq(1-(C''-o(1)))pn$. Therefore, we obtain a bound $E[\log T_{\tilde{U}}]\leq(1-p)n+(1-(C''-o(1)))pn+O(\log n)=(1-(C''-o(1))p)n$. The lemma follows from $p=1/d^{O(d^2)}$. □

Finally, we give a proof of Claim 3.31 and completes the entire proof of Lemma 3.8.

Proof of Claim 3.31 We will use Corrary 3.30 to obtain the desired upper bound. Note that $|R|$ is the number of remaining variables and that $|I'_{\tilde{U}}|+2k$ is the number of constraints. Recall that λ in Corrary 3.30 is defined as the number of constraints divided by the number of variables. Thus $\lambda=\frac{|I'_{\tilde{U}}|+2k}{|R|}$.

Firstly, we consider the following two cases; (i)$\lambda\geq 1/2$, and (ii) $\lambda<1/2$.

Let's consider the case (i) $\lambda\geq 1/2$. In this case, $1\geq 1/2\lambda$ and then $\log(1+1/2\lambda)\leq 1$.

By Corrary 3.30, $\log T_{ZOLP}[|I'_{\tilde{U}}|+2k,|R|]$ is $0.5|R|+\lambda(\log(e)+1)|R|\leq 0.5|R|+\frac{|I'_{\tilde{U}}|+2k}{|R|}(\log(e)+1)|R|$. Since Corrary 3.28 implies $E[|I'_{\tilde{U}}|]\leq 3\delta pn$, it holds that

$$\begin{aligned}&E[\log T_{ZOLP}[|I'_{\tilde{U}}|+2k,|R|]]\\&\quad\leq 0.5E[|R|]+(\log(e)+1)E[|I'_{\tilde{U}}|]+o(n)\\&\quad\leq 0.5pn+(\log(e)+1)3\delta pn+o(n).\end{aligned}$$

Let's consider the case (ii) $\lambda<1/2$. In this case it holds that $|I'_{\tilde{U}}|\leq\frac{1}{2}|R|-2k$ and $\log(1+1/2\lambda)\leq\log(1/\lambda)$.

Therefore by Corrary 3.30,

$$\begin{aligned}&\log T_{ZOLP}[|I'_{\tilde{U}}|+2k,|R|]\\&\leq 0.5|R|+\lambda(\log(e)+\log(1/\lambda))|R|\\&\leq 0.5|R|+\frac{|I'_{\tilde{U}}|+2k}{|R|}(\log(e)-\log(|I'_{\tilde{U}}|+2k)+\log|R|)|R|\end{aligned}$$

Note that $-\log\lambda = -\log\frac{|I'_{\tilde{U}}|+2k}{|R|} = -\log(|I'_{\tilde{U}}|+2k)+\log|R|$.

In this case, we consider the following two subcases ; (ii-a) for all real constant $\beta > 0$, $|I'_{\tilde{U}}|+2k \leq \beta|R|$, and (ii-b) there exists some real constant β_0 $(0 < \beta_0 < \frac{1}{2})$, $|I'_{\tilde{U}}|+2k > \beta_0|R|$.

(ii-a) for all real constant β $(0 < \beta < \frac{1}{2})$, $|I'_{\tilde{U}}|+2k \leq \beta|R|$.

Then, $\log T_{ZOLP}[|I'_{\tilde{U}}|+2k, |R|] \leq \log T_{ZOLP}[\beta|R|, |R|]$. Thus, we have

$$\begin{aligned}
&E[\log T_{ZOLP}[|I'_{\tilde{U}}|+2k, |R|]] \\
&\leq E[\log T_{ZOLP}[\beta|R|, |R|]] \\
&\leq (1/2+\beta(\log(e)+\log(1+1/2\beta)))E[|R|] \\
&= (0.5+\beta(\log(e)+\log(1+1/2\beta)))pn.
\end{aligned}$$

(ii-b) there exists some real constant β_0 $(0 < \beta_0 < \frac{1}{2})$, $|I'_{\tilde{U}}|+2k > \beta_0|R|$.

In this case, $\beta_0|R| < |I'_{\tilde{U}}|+2k < \frac{1}{2}|R|$. Thus, there exists some real constant α $(\beta_0 < \alpha < \frac{1}{2})$ such that $|I'_{\tilde{U}}|+2k = \alpha|R|$.

Hence $0.5|R| + \frac{|I'_{\tilde{U}}|+2k}{|R|}(\log(e)-\log(|I'_{\tilde{U}}|+2k)+\log|R|)|R| = 0.5|R| + (|I'_{\tilde{U}}|+2k)(\log(e)-\log(\alpha|R|)+\log|R|) = 0.5|R| + (|I'_{\tilde{U}}|+2k)(\log(e)+\log(\alpha^{-1}))$. Remind that $|I'_{\tilde{U}}| \leq \frac{1}{2}|R|-2k$ and $\log(1+1/2\lambda) \leq \log(1/\lambda)$.

Therefore,

$$\begin{aligned}
&E[\log T_{ZOLP}[|I'_{\tilde{U}}|+2k, |R|]] \\
&= 0.5E[|R|] + (\log(e)+\log(\alpha^{-1}))E[|I'_{\tilde{U}}|] + 2k(\log(e)+\log(\alpha^{-1})) \\
&\leq 0.5pn + (\log(e)+\log(\alpha^{-1}))(3\delta p)n + 2k(\log(e)+\log(\alpha^{-1}))
\end{aligned}$$

In the case (ii-a), for any $\delta > 0$, let β be a constant such that $\beta(\log(e)+\log(1+1/2\beta)) \leq 3\delta$. Thus, in the case (ii-a), $T_{ZOLP}[|I'_{\tilde{U}}|+2k, |R|]] \leq 0.5pn + 3\delta pn + o(n)$.

Let N_0 be $\max\{\log(e)+1, \log(e)+\log(\alpha^{-1})\}$. Note that N_0 does not depend on δ because α does not depend on δ. Therefore, there exists some constant N_0 for any $\delta > 0$ it holds that $E[\log T_{ZOLP}[|I'_{\tilde{U}}|+2k, |R|]] \leq 0.5pn + 3N_0\delta pn + o(n)$. $\square$

Chapter 4

A Nonuniform Restricted Circuit Class with Threshold Gates Having Strong Size Lower Bounds

The structure of this chapter is as follows. In Section 4.1, we define several notions being necessary in other sections. In Section 4.2, we mention prior works related with this thesis. We formally state our work in Section 4.2, and we give the proof of our results in sections 4.3 and 4.4. We finally give a result on $\mathsf{ACC} \circ \mathsf{THR} \circ (k\text{-}\mathsf{THR})^d$ lower bounds against NEXP, extending results in [46].

4.1 Preliminaries

In this section, we give several definitions for stating our work.

Definition 4.1. Let $x_1, ..., x_n$ be boolean variables. Let $w_1, ..., w_n, t$ be real numbers.
(1) When all weights are one and the threshold value is the half of fan-in wires, the threshold gate is called *majority gate*.
(2) We define a symmetric gate $SYM_S(x_1, ..., x_n)$ as a gate computing a boolean function $SYM_S(x_1, ..., x_n) = 1 \iff \Sigma_{i=1}^n x_i \in S$ for a subset $S \subseteq \{0, 1, ..., n\}$. We call S the characteristic set of the symmetric gate SYM_S.

Remark of Definition 4.1 In this thesis, we suppose that the absolute value of any weight in threshold gates is at most $2^{poly(n)}$ and any weight are coded by a binary string of length $poly(n)$. We will use the term *source* (*sink*, resp.) to represent an input variable or a gate which is connected to the input terminal (the output terminal, resp.) of a wire.

We remind a notion of "dependency" of gates which was also introduced in Chapter 3.

Definition 4.2. Let C be an arbitrary circuit. For a gate G in C, let $G(x) \in \{0, 1\}$ denote the output value of G when we feed an input string x to the circuit C.

(1) Two gates G_1, G_2 in C have *dependency*, if one of two preimages $G_1^{-1}(1)$ and $G_2^{-1}(1)$ is a subset of the other one. In other words, $\forall x \in \{0,1\}^n[G_1(x) \leq G_2(x)] \vee \forall x \in \{0,1\}^n[G_2(x) \leq G_1(x)]$.
(2) A subset of gates in C is called *independent gate set*, if any two gates in the set do not have dependency. A circuit may contain several independent gate sets.

Definition 4.3. Let $\mathcal{L}_i$ be a type of gates for each $i = 1, 2, ..., d$. Let C be a circuit, and let V_0 and V be respectively the set of input variables of C and the set of gates of C.
(1) A circuit C is a $\mathcal{L}_d \circ \mathcal{L}_{d-1} \circ \cdots \circ \mathcal{L}_1$ circuit, if there exists some partition $V_1, ..., V_d$ of the set V such that **(i)** any wire from $G \in V_i$ to $G' \in V_j$ satisfies that $i < j$ and **(ii)** a type of all gates in V_i is $\mathcal{L}_i$ for each i. We call this partition $V_1, ..., V_d$ a *layering partition* of C. We also call each V_i *the i-th layer*.
(2) We assume that any gate G of a $\mathcal{L}_d \circ \cdots \circ \mathcal{L}_1$ circuit has an integer label i such that G is in the i-th layer, where $1 \leq i \leq d$. We call such labels *layering labels*.

Remark of Definition 4.3 Note that there is no wire connecting two gates belonging to the same layer. For example, any AC^0 circuit C is in $\mathcal{L}_d \circ \mathcal{L}_{d-1} \circ \cdots \circ \mathcal{L}_1$ for some constant d not depending on the number of input variables, where for each i $(1 \leq i \leq d)$ $\mathcal{L}_i \in \{AND, OR, NOT\}$.

Definition 4.4. Let C be a $\mathcal{L}_d \circ \mathcal{L}_{d-1} \circ \cdots \circ \mathcal{L}_1$ circuit. Let $V_1, ..., V_d$ be a layering partition of C.
(1) A set V_i is called *the i-th k-$\mathcal{L}_i$ layer* in C, if the maximum size of an independent gate set $I \subseteq V_i$ in C is at most k.
(2) We call C a k-$\mathcal{L}_d \circ k$-$\mathcal{L}_{d-1} \circ \cdots \circ k$-$\mathcal{L}_1$ circuit, if there exists some layering partition $V_1, ..., V_d$ such that each V_i is the i-th k-$\mathcal{L}_i$ layer in C. Let $(\mathcal{L})^d$ denote an abbreviation of $\mathcal{L}_d \circ \mathcal{L}_{d-1} \circ \cdots \circ \mathcal{L}_1$, if $\mathcal{L}_i$ is the same type $\mathcal{L}$ for all i.
(3) We define $\mathcal{C}_k[d]$ as a class of $\mathsf{SYM} \circ (k\text{-}\mathsf{THR})^d$ circuits.

Remark of Definition 4.4 **(1)** We usually use the term "circuit" as single output circuits, and we particularly mention the use of multi-output circuits. When we consider a class of single output circuits, we write $\mathcal{C}_d \circ k$-$\mathcal{C}_{d-1} \circ \cdots \circ k$-$\mathcal{C}_1$ instead of k-$\mathcal{C}_d \circ k$-$\mathcal{C}_{d-1} \circ \cdots \circ k$-$\mathcal{C}_1$. **(2)** In this thesis, we may use the words *a k-THR layer*, *a layer of k-THR gates* or just *a layer of threshold gates* to mention one of the above sets $V_1, ..., V_d$.

We assume that for an arbitrary $\mathcal{C}_k[d]$ circuit C, each THR gate G in C has an integer label $i(1 \leq i \leq d)$ *called layering label* such that G has the label i if and only if G is in the i-th k-THR layer in C from the bottom level.

In the area of circuit complexity, we sometimes meet the term *layered circuit* that there is a partition $P_1, ..., P_d$ of gate set to d layers such that all wires are put between adjacent gate sets P_i and P_{i+1} from the bottom level P_1 to the top level P_d (e.g. [30]). Any circuit C in a circuit class defined as an accumulated layers of particular gates can have equivalent layered circuit in the same class, because we can *replace wires*, which are not between two adjacent gate sets, *with dummy gates*. However, executing this *replacement for our setting may broke* the condition about the size of maximum independent gate sets. Thus, in our setting, the term *layer* in a circuit is just *corresponding to an element of partition of the gate set* of the circuit. In this thesis, we never use the term layer as any meaning relating with the layered circuit.

Let $\mathcal{A}_t$ be an $\mathcal{A}$ gate with at most t fan-in wires, where $\mathcal{A}$ is a gate type. We will mainly use this notion in sections 4.2 and 4.3 with an explanation of results in [2, 6].

4.2 Prior Work and Our Results

In this section, we firstly review the proof of super quasi-polynomial lower bounds for $\mathsf{ACC} \circ \mathsf{THR}$ by Williams [46] since it is closely related to our work in Subsection 4.2.1. In Subsection 4.2.2, we define a complexity class to separate from NEXP and give a formal statement of our result. In Subsection 4.2.3, we give an intuitive explanation about our proof strategy and formally state notions to understand our proof methods.

4.2.1 Prior work

In [46], the following property of circuit classes plays an important role.

Definition 4.5. Let $\mathcal{C}$ be a circuit class. The class $\mathcal{C}$ is *weakly closed under AND*, if there is a polynomial time procedure such that for given the AND of two $\mathcal{C}$ circuits the procedure produces an equivalent $\mathcal{C}$ circuit.

Remark of Definition 4.5 Note that the time complexity of a procedure in the above definition is a function in the size of a code of two circuits.

We state a meta theorem in [46].

Theorem 4.6 ([**46**])**.** Let $\mathcal{C}$ be a circuit class *weakly closed under AND*. Suppose for any $c \geq 1$, there is an $\varepsilon > 0$ and an algorithm for counting the satisfying assignments in time $2^{n-\Omega(n^{\varepsilon})}$ on $\mathcal{C}$ circuits with n inputs and $n^{\log^c n}$ size. Then NEXP does not have quasi-polynomial size $\mathcal{C}$ circuits.

Note that $\mathsf{ACC} \circ \mathsf{THR}$ clearly satisfies the closure property under AND. By this general theorem, we can derive super quasi-polynomial $\mathsf{ACC} \circ \mathsf{THR}$ lower bounds against NEXP, if we construct a faster counting algorithm. The following theorem achieves this.

Theorem 4.7 ([**46**])**.** For every $m > 1$ and $d > 0$, there is an $\varepsilon > 0$ such that counting satisfying assignments to $\mathsf{ACC} \circ \mathsf{THR}$ circuits of size $2^{n^{\varepsilon}}$, depth d and modulus m gates can be solved in $2^{n-n^{\varepsilon}}$ time.

Remark 4.7. Williams actually constructed a counting algorithm for the class of circuits $\mathsf{ACC} \circ \mathsf{SYM}$. He showed that there is a transformation from an arbitrary threshold gate to a constant depth circuit with single layer of symmetric gates. Using this transformation we can transform an arbitrary $\mathsf{ACC} \circ \mathsf{THR}$ circuit to an $\mathsf{ACC} \circ \mathsf{SYM}$ circuit. Thus, Theorem 4.7 can be proved by giving a counting algorithm for $\mathsf{ACC} \circ \mathsf{SYM}$ circuits. Below we formally state this transformation since we will also use this in the proof of our result.

Claim 4.8 ([46]). An arbitrary THR gate G can be replaced with $\mathsf{AC}^0 \circ \mathsf{MAJ}$ circuit C such that the size of C is at most polynomial in the input size of G. Moreover, time complexity for the replacement is at most polynomial time.

We give a brief overview about a proof of Theorem 4.7.

For an arbitrary $\mathsf{ACC} \circ \mathsf{SYM}$ circuit, a $\mathsf{SYM} \circ \mathsf{ACC} \circ \mathsf{SYM}$ circuit is constructed by partially fixing input variables. The reason why this construction of the $\mathsf{SYM} \circ \mathsf{ACC} \circ \mathsf{SYM}$ circuit is useful to count the satisfying assignments is as follows. The top symmetric gate can be regarded as a *counter device* so that counting the number of satisfying assignments to the $\mathsf{ACC} \circ \mathsf{SYM}$ circuit can be done by evaluating outputs of the $\mathsf{SYM} \circ \mathsf{ACC} \circ \mathsf{SYM}$ circuit for all inputs. Therefore, it is sufficient to show an efficient procedure evaluating output values of given $\mathsf{SYM} \circ \mathsf{ACC} \circ \mathsf{SYM}$ circuit for all input strings. Any $\mathsf{SYM} \circ \mathsf{ACC} \circ \mathsf{SYM}$ circuit can be converted to a $\mathsf{SYM} \circ \mathsf{AND}_u \circ \mathsf{SYM}$ circuit for poly-logarithmic u, because there is a transformation from $\mathsf{SYM} \circ \mathsf{ACC}$ circuits to $\mathsf{SYM} \circ \mathsf{AND}_u$ circuits for poly-logarithmic u [2, 6]. This depth three circuit can be transformed to a depth *two* symmetric circuit. The following claim is applied for this depth reduction, while it is not explicitly mentioned in [46].

Claim 4.9. ([46]) There is a procedure such that for given $\mathsf{AND}_u \circ \mathsf{SYM}$ circuit C with n input variables and $N = N(n)$ wires, where each input variable can have more than one wire connecting to a same symmetric gate in C, the procedure converts the circuit C to *a single* SYM *gate* with $O(N^u)$ wires.

Finally, an extension of matrix rank called *symmetric rank* for a depth two symmetric circuit is defined. An algorithm which evaluates outputs of depth two symmetric circuits for all input strings is constructed by combining the notion of symmetric rank and a fast matrix multiplication algorithm in [14]. For all above arguments a proof of Theorem 4.7 is obtained.

4.2.2 Lower Bounds against NEXP for a Circuit Class with Multi Layers of Threshold Gates

In this subsection, we firstly define a class of circuits and a relating class of languages.

Definition 4.10. We define $\tilde{\mathcal{C}}_k[d]$ as $\bigcup\limits_{c>0} \mathcal{C}_{ck}[d]$.

We note that $\tilde{\mathcal{C}}_k[d]$ is a class of circuits with n input variables, where c does not depend on n and k.

Definition 4.11. Let $\mathcal{A}$ be an arbitrary circuit class. We define $\mathcal{A}$-$\mathsf{SIZE}[S(n)]$ as the class of languages having a family of $\mathcal{A}$ circuits with n inputs and of size $O(S(n))$.

We mention our primal goal.

Theorem 4.12. Let $d = poly \log n$ for the number of input variables n. There is some $\gamma > 0$ such that $\mathsf{NEXP} \not\subseteq \tilde{\mathcal{C}}_k[d]$-$\mathsf{SIZE}[2^{poly(\log n)}]$ for $k \leq n^{\gamma}$.

Before proving lower bounds for the class $\tilde{\mathcal{C}}_k[d]$, we see that $\mathcal{C}_k[d]$ is able to compute all boolean functions even when $k = 1$ and $d = 1$. This means that $\mathsf{NEXP} \subseteq \tilde{\mathcal{C}}_1[1]\text{-}\mathsf{SIZE}[2^n]$ and gives the motivation for studying the complexity of circuits $\tilde{\mathcal{C}}_k[d]$ with small values of k and d. Recall that $\mathsf{SYM} \circ k\text{-}\mathsf{THR}$ as well as $\mathsf{THR} \circ k\text{-}\mathsf{THR}$ is universal for $k = 1$.

We prove Theorem 4.12 by applying Theorem 4.6. Apparently, it is sufficient to prove the following two lemmas.

Lemma 4.13. The class of $\tilde{\mathcal{C}}_k[d]$ circuits with n inputs and of quasi-polynomial $2^{\log^{O(1)} n}$ size is weakly closed under AND.

Lemma 4.14. Let d be *poly* $\log n$ in the number of input variables n. There exist some $\varepsilon > 0$ and $\gamma > 0$ such that counting satisfying assignments to $\mathcal{C}_k[d]$ circuits of size $S(n) = 2^{n^{o(1)}}$ can be solved in $2^{n-\Omega(n^{\varepsilon})}$ time for $k \leq n^{\gamma}$.

We will give a proof outline of the lemma about closure property in Section 4.3 We will give a counting algorithm in Section 4.4 The algorithm in Theorem 4.7 is incorporated to our counting algorithm as a subroutine.

4.2.3 Restrictions to Output of Threshold Gates

In this Subsection, we give several notions to understand the reason why we can construct a faster satisfiability or counting algorithm for circuits with bounded size independent gate sets. We will give extensions of these notions to multilayer setting in Section 4.4. Suppose that $\mathcal{G}$ is a set of gates which has independent gate sets of size at most k and has *no pair of equivalent gates*. Then, we can form a partial ordered set $(\mathcal{G}, \preceq)$ by defining the order $\preceq$ on $\mathcal{G}$ so that $G_1 \preceq G_2$ iff $G_1^{-1}(1) \subseteq G_2^{-1}(1)$ for $G_1, G_2 \in \mathcal{G}$. It is easy to observe that the size of an independent vertex set of the Hasse diagram of $(\mathcal{G}, \preceq)$ is at most k. Recall that this Hasse diagram is called an Induced Hasse Diagram (I.H.D, in short) of $\mathcal{G}$. In Chapter 3, depth two threshold circuits are considered, and I.H.D is defined for the set of bottom gates. In this thesis, we will extend the notion of I.H.D for k-THR layer in Section 4.4.

Definition 3.17(restated)
(1) Let V be a set of gates and let $H = (V, E)$ be I.H.D of V. A coloring $\chi : V \mapsto \{0, 1\}$ is called a *validly ordered restriction*, if $\forall(u, v) \in E$, $\chi(u) \leq \chi(v)$.
(2) Let χ be a validly ordered restriction for an arbitrary I.H.D $H = (V, E)$.
We define the *min-set of H for χ* as the set $\{u_{min} \in V \cap \chi^{-1}(1) : \forall v \in V \setminus \{u_{min}\}[v \preceq u_{min} \Rightarrow \chi(v) = 0]\}$.
We define the *max-set of H for χ* as the set $\{u_{max} \in V \cap \chi^{-1}(0) : \forall v \in V \setminus \{u_{max}\}[u_{max} \preceq v \Rightarrow \chi(v) = 1]\}$.

Threshold gates in min-sets and max-sets are regarded as some critical local information about all k-THR layers in a $\mathcal{C}_k[d]$ circuit. We give the following notion stating min-sets and max-sets from a viewpoint of graph theory.

Definition 3.18(restated) Let H be an I.H.D and χ be a validly ordered restriction of H.

Let I_1, I_0 be independent sets in H. The pair of independent sets (I_1, I_0) satisfies the *covering condition* for H, if the following condition holds.
Condition: For any $v \in V \setminus (I_1 \cup I_0)$ in H, either $\exists u_1 \in I_1, u_1 \preceq v$ or $\exists u_0 \in I_0, v \preceq u_0$ according to the order $\preceq$ of H. We define X'_H as the set $\{\chi : \chi$ is a validly ordered restriction of H $\}$. We also define $\mathcal{I}_H$ as follows: $\mathcal{I}_H := \{(I_1, I_0) \subseteq V \times V : I_1, I_0$ are independent sets satisfying the covering condition in H $\}$.

Recall the following lemma in Chapter 3.
Lemma 3.22 There is a bijection $\mu_H : X'_H \ni \chi \mapsto (I_1, I_0) \in \mathcal{I}_H$ such that if $\mu_H(\chi) = (I_1, I_0)$ then I_1 is the min-set of H for χ and I_0 is the max-set of H for χ.

Observation 3.25(restated) For boolean functions $P_1(x), ..., P_m(x) : \{0,1\}^n \to \{0,1\}$, the following statements hold.
(1) If $\forall x \forall i [P_i(x) \leq P_{i-1}(x)]$, then $\forall x [\bigwedge_{i=1}^m P_i(x) = 1 \iff P_1(x) = 1]$.
(2) If $\forall x \forall i [P_i(x) \leq P_{i-1}(x)]$, then $\forall x [\bigvee_{i=1}^m P_i(x) = 0 \iff P_m(x) = 0]$.
We explain the reason why this observation and Lemma 3.22 is useful to prove Lemma 4.14. Intuitively, the output of an arbitrary gate of a circuit is determined by fixing gates in the union of min-set and max-set by Observation 3.25 . By Lemma 3.22, it is also determined by fixing the output of threshold gates in $I_1 \cup I_0$, where $(I_1, I_0) \in \mathcal{I}_H$. Note that for given pair of subsets of the vertex set of H we can efficiently decide whether $(I_1, I_0) \in \mathcal{I}_H$ holds or not. If the maximum size of independent gate sets is small, then the number of threshold gates we have to fix is not so many. We will give a more detailed description of the algorithm in Subsection 4.4.2.

4.3 Closure Property under AND

In this section, we state how to prove the follwing lemma.

Lemma 4.13(restated) The class of $\tilde{\mathcal{C}}_k[d]$ circuits with n inputs and of quasi-polynomial $2^{\log^{O(1)} n}$ size is weakly closed under AND.
We assume that the input size is measured by the number of wires of circuits.

Recall the following claim which is in [46].
Claim 4.9([46]) (restated) There is a procedure such that for given $\mathsf{AND}_u \circ \mathsf{SYM}$ circuit C with n input variables and $N = N(n)$ wires, where each input variable can have more than one wire connecting to a same symmetric gate in C, the procedure converts the circuit C to *a single* SYM *gate* with $O(N^u)$ wires.

Actually, we only use the case $u = 2$ of the above claim. For the completeness, we describe the proof of Claim 4.9 for this case. The extension for genaral u is straightforward (see [46]).

Proof. Let $x_1, ..., x_n$ be input variables of the input circuit C, and let $S_1(x_1, ..., x_n) \wedge S_2(x_1, ..., x_n)$ denote the top AND gate of C, where S_i is the i-th symmetric gate. For $i = 1, 2$, let $f_i : \mathbb{Z} \to \{0,1\}$ be a function such that $f_i(\sum_{j=1}^n c_{i,j} x_j) = S_i(x_1, ..., x_n)$, where

$c_{i,j}$ is the number of wires from the variable x_j to the symmetric gate S_i.

Let $B = 1 + \max_{i=1,2}(\sum_{j=1}^{n} c_{i,j})$. Consider the following linear form.

$$L(x_1, ..., x_n) = B \cdot \left(\sum_{j=1}^{n} c_{2,j} x_j \right) + \left(\sum_{j=1}^{n} c_{1,j} x_j \right). \tag{4.1}$$

For any boolean assignment to the variables $x_1, ..., x_n$, the integer encoded by the linear form $L(x_1, ..., x_n)$ is an integer encoded in $O(\log N)$ bits. Put $S_L = \{bB + a \mid a \in f_1^{-1}(1) \text{ and } b \in f_2^{-1}(1)\}$. It is easy to observe that $L(x_1, \ldots, x_n) \in S_L$ iff $(S_1 \wedge S_2)(x_1, \ldots, x_n) = 1$. Hence, a symmetric gate of characteric set S_L, which takes $B_{2,j} c_{2,j} + c_{1,j}$ wires from the input variable x_j for each j, computes $S_1 \wedge S_2$. Apparently, the total number of wires is bounded by $O(B^2) = O(N^2)$. □

We directly apply the above claim to our setting as follows.

Claim 4.15. There is a procedure such that for given AND of two $\tilde{\mathcal{C}}_k[d]$ circuits C_1 and C_2, where C_1 and C_2 have n input variables and at most $t = t(n)$ top fan-in and at most $N = N(n)$ wires, the procedure transforms the input to a $\mathsf{SYM} \circ (\mathsf{THR})^d$ circuit C_3 having at most $O(t^2 N)$ wires. Moreover, any gate in the l-th layer in C_3 is in either the l-th layer of C_1 or the l-th layer of C_2.

Proof. Let $t_i (\leq t)$ be the top fan-in of the circuit C_i for $i = 1, 2$. Let $v_{i,1}, ..., v_{i,t_i}$ be sources of the top symmetric gate of C_i for $i = 1, 2$.

For each $i = 1, 2$ and for each $1 \leq j \leq t_i$, let $y_{i,j}$ denote the boolean value of the source $v_{i,j}$, when we feed an input string x to the circuit C_i, i.e. if $v_{i,j}$ is a gate G then $y_{i,j} = G(x)$ and if $v_{i,j}$ is the b-th input variable then $y_{i,j} = x_b$. Let $c_{i,j}$ be the number of wires between the source $v_{i,j}$ and the top gate of the circuit C_i for each $i = 1, 2$ and for each $1 \leq j \leq t_i$.

Let B be $1 + \max\limits_{i=1,2}\{\sum_{j=1}^{t_i} c_{i,j}\}$. We consider to replace two symmetric gates with a single symmetric gate. For this purpose, we implement the linear form $B\left(\sum_{j=1}^{t_2} y_{2,j}\right) + \left(\sum_{j=1}^{t_1} y_{1,j}\right)$ as mentioned in the proof of Claim 4.9. The output value of the new symmetric gate is computed by this linear form.

Remove the top symmetric gate from C_1, and regard fan-ins of the top symmetric gate as output wires. Let C_1' be the multi output circuit obtained by this operation. Let C_2' be the circuit which is obtained from C_2 in the same way.

We take B copies of C_2' and let these copies be $C_{2,1}', ..., C_{2,B}'$. Layering labels are also copied. Connect the output wires of $C_{2,1}', ..., C_{2,B}', C_1'$ to the new symmetric gate as defined by the linear form. The number of wires of C_3 is at most $O(B) \times O(t) \times O(N) = O(t^2 N)$. We note that $t_1, t_2 \leq O(t)$ and $B \leq O(t)$. We also note that any sub-circuit whose top gate is the source $v_{i,j}$ has at most N wires, where we regard an input variable as a sub-circuit without wire.

Because layering labels are also copied, any gate in the l-th layer in C_3 is in either the l-th layer of C_1 or the l-th layer of C_2. □

We give the proof of Lemma 4.13 as follows.

Proof of Lemma 4.13. Suppose that C_1, C_2 are respectively a $\mathcal{C}_{c_1k}[d]$ circuit and a $\mathcal{C}_{c_2k}[d]$ circuit. We show that there is a polynomial time procedure such that for given the AND of a $\mathcal{C}_{c_1k}[d]$ circuit C_1 and a $\mathcal{C}_{c_2k}[d]$ circuit C_2, where C_1, C_2 have n input variables and at most N wires and c_1, c_2 do not depend on n and k, the procedure outputs an equivalent $\mathcal{C}_{(c_1+c_2)k}[d]$ circuit C_3 with $poly(N)$ wires.

Note that top fan-in of a circuit is at most the number of wires in the circuit. Hence, the number of wires in the output circuit C_3 in Claim 4.15 is at most $poly(N)$. All that we prove is that the output circuit C_3 in Claim 4.15 is a $\mathcal{C}_{(c_1+c_2)k}[d]$ circuit.

For each $i = 1, 2$, let C_i' be a $(c_ik\text{-}\mathsf{THR})^d$ circuit with n input variables and at most $t = t(n)$ output wires such that C_i' is obtained by removing the top symmetric gate from C_i and by regarding input wires of the top symmetric gate as output wires.

We can construct C_3 by copying C_2'. Let $\mathcal{G}_{2,l}$ be the set $\{G_{i,j} : G_{i,j}$ is the i-th gate in the j-th copy of the l-th c_2k-THR layer in C_2 $\}$, and let $\mathcal{G}_{1,l}$ be the l-th threshold layer in C_1.

By a contradiction, we prove that $\mathcal{G}_{2,l}$ has *no* independent gate set of size greater than c_2k for each fixed l. Let $\mu : \mathcal{G}_{2,l} \to \mathbb{N}$ be a map such that $\mu(G_{i,j}) = i$. Let $M \geq 2$ be an arbitrary integer. Let $I_1, I_2, ..., I_M \subseteq \mathcal{G}_{2,l}$. Suppose that (A) $I_1 \cup I_2 \cup \cdots \cup I_M$ is an independent gate set and (B) all sets $I_1, I_2, ..., I_M$ are distinct singleton sets. Then, any two of images $\mu(I_1), \mu(I_2), ..., \mu(I_M)$ are disjoint, and we have that $|\mu(I_1) \cup \cdots \cup \mu(I_M)| = M$. Note that $G_{i,j}$ and $G_{i,j'}$ are equivalent for any i and for any j, j'. We also note that there is *no* independent gate set in the l-th layer of C_3 which contains both $G_{i,j}$ and $G_{i,j'}$.

For the sake of contradiction, suppose that there exists $k'(> c_2k)$ distinct gates $A_1, ..., A_{k'}$ in $\mathcal{G}_{2,l}$. Let I_m denote a singleton set $\{A_m\}$ for each $1 \leq m \leq k'$. We note that $I_1, ..., I_{k'}$ satisfies the condition (A) and (B). Then, there exist k' gates $B_1, ..., B_{k'} \in \{G_{1,1}, G_{2,1}, ..., G_{s,1}\}$ such that A_i and B_i are equivalent for each i, where s is the number of gates in the l-th c_2k-THR layer in C_2. Thus, there are some two gates which have dependency. Therefore, the size of independent gate set in l-th threshold layers in C_3 is at most $(c_1 + c_2)k$. $\square$

4.4 Transforming of Circuits and a Counting Algorithm

In this section, our goal is to prove the following lemma.

Lemma 4.14(restated) Let d be *poly* $\log n$ in the number of input variables n. There exist some $\varepsilon > 0$ and $\gamma > 0$ such that counting satisfying assignments to $\mathcal{C}_k[d]$ circuits of size $S(n) = 2^{n^{o(1)}}$ can be solved in $2^{n-\Omega(n^{\varepsilon})}$ time for $k \leq n^{\gamma}$.

4.4.1 Notions for bottom up procedures

Let $f : X \to Y$ be a map for finite sets X, Y. For an arbitrary $A \subseteq X$, let $f|_A$ denote the map satisfying that $\forall x \in A, f|_A(x) = f(x)$.

Definition 4.16. Let $\mathcal{C}$ be a circuit class $\mathcal{C}' \circ (k\text{-}\mathsf{THR})^d$ for an arbitrary $\mathcal{C}'$ gate at the top level. We call a $\mathcal{C}' \circ (k\text{-}\mathsf{THR})^d$ circuit an *abbreviated circuit*, if for any threshold layer in the circuit there is *no* pair of equivalent gates in the threshold layer.

Definition 4.17. Let $\mathcal{C}$ be a circuit class $\mathcal{C}' \circ (k\text{-}\mathsf{THR})^d$ for an arbitrary $\mathcal{C}'$ gate at the top level. Let C be an abbreviated $\mathcal{C}$ circuit. Let V_j be the j-th k-THR layer in C for each $1 \leq j \leq d$.

We call a family of directed graphs $\mathcal{F}_i = \{H_j = (V_j, E_j) : 1 \leq j \leq i\}$ an *i-th Induced Hasse Diagram Family* of C, if for each j the directed graph H_j is the Hasse diagram of the partial ordered set $(V_j, \preceq)$ defined as follows:

$$\forall G_1 \in V_j, \forall G_2 \in V_j [G_1 \preceq G_2 \iff G_1^{-1}(1) \subseteq G_2^{-1}(1)].$$

We call a map $\rho : V_1 \cup \cdots \cup V_d \to \{0,1\}$ a *validly ordered restriction for a family $\mathcal{F}$* , if $\rho|_{V_i}$ is a validly ordered restriction to V_i for any i. We also call a d-th induced Hasse diagram family an *induced Hasse diagram family* of C, and $\mathcal{F}_d$ is simply denoted by $\mathcal{F}$.

4.4.2 Proof of Lemma 4.14

We state the following lemma.

Lemma 4.18. Let $\mathcal{C}$ be a circuit class $\mathcal{C}' \circ (k\text{-}\mathsf{THR})^d$, where $\mathcal{C}'$ is either SYM or THR. There is a procedure such that for given abbreviated $\mathcal{C}$ circuit C of size $S(n)$ and $\mathcal{F}$ which is the induced Hasse diagram family of $\mathcal{C}$, the procedure outputs an $\mathsf{AC}^0 \circ \mathsf{SYM}$ circuit C' of size at most $k^{2d}\binom{S(n)}{O(k)}^d poly(S(n))$ such that C is equivalent to C'. Moreover, this procedure runs in $O\left(k^{2d}\binom{S(n)}{O(k)}^d poly(S(n))\right)$ time.

Proof. Let $\mathcal{F}$ be $\{H_1 = (V_1, E_1), ..., H_d = (V_d, E_d)\}$. We note that $V_1 \cup \cdots \cup V_d$ is the set of threshold gates of C. We define $\mathcal{I}_i$ as follows: $\mathcal{I}_i := \{(I_i, J_i) \subseteq V_i \times V_i : (I_i, J_j)$ is a pair of independent sets satisfying the covering condition in $H_i\}$.

By Lemma 3.22, the definition of min-set and max-set in Definition 3.17 and Observation 3.25 , for each $1 \leq i \leq d$ and for any $(I_i, J_i) \in \mathcal{I}_i$ there uniquely exists $\mu_i : V_i \to \{0,1\}$ such that (A) μ_i is a validly ordered restriction to V_i and (B) the two images $\mu_i(I_i)$ and $\mu_i(J_i)$ are respectively $\{1\}$ and $\{0\}$, and (C) we can fix all outputs of threshold gates in the i-th k-THR layer according to μ_i. Thus, for any $(I_1, J_1) \in \mathcal{I}_1, ..., (I_d, J_d) \in \mathcal{I}_d$ there uniquely exists $\rho : V_1 \cup \cdots \cup V_d \to \{0,1\}$ such that $\rho|_{V_i}$ is equivalent to the restriction μ_i for any i. Let R be the set of all validly ordered restrictions for $\mathcal{F}$. By Lemma 3.22, there is some bijection $\kappa : \mathcal{I}_1 \times \cdots \times \mathcal{I}_d \to R$ such that $\kappa((I_1, J_1), ..., (I_d, J_d)) = \rho$. Then, R and the image $\kappa(\mathcal{I}_1 \times \cdots \times \mathcal{I}_d)$ are the same set. Hence, we have $|R| \leq k^{2d}\binom{S(n)}{O(k)}^d$.

Suppose that $1 \leq \forall i \leq d, (I_i, J_i) \in \mathcal{I}_i$. Then, we can construct an integer linear programming (ILP, in short) instance S_ρ with at most $d \cdot 2k$ constraints such that **(i)** every linear constraint labeled at a gate in $\bigcup_{1 \leq i \leq d} I_i$ holds and **(ii)** *no* linear constraint labeled at a gate in $\bigcup_{1 \leq i \leq d} J_i$ holds. Let $D|^{(\rho)}$ be an $\mathsf{AND} \circ \mathsf{THR}$ circuit corresponding to the ILP instance S_ρ.

We define $T|^{(\rho)}$ as the top gate of C that *all input wires except ones whose sources are input variables* are fixed.

By noticing that the ILP instance can be expressed by an $\mathsf{AND} \circ \mathsf{THR}$ circuit, we obtain an $\mathsf{AND} \circ \{\mathsf{SYM}, \mathsf{THR}\}$ circuit $C'|^{(\rho)} = T|^{(\rho)} \wedge D|^{(\rho)}$ for each restriction ρ to the outputs of threshold gates. Therefore, we obtain the $\mathsf{OR} \circ \mathsf{AND} \circ \{\mathsf{SYM}, \mathsf{THR}\}$ circuit $\bigvee_{\rho \in R} C'|^{(\rho)}$ which is equivalent to C.

Using Claim 4.8, we convert all threshold gates to $\mathsf{AC}^0 \circ \mathsf{MAJ}$ circuits. Note that majority gates are also symmetric ones. Thus, we obtain an $\mathsf{AC}^0 \circ \mathsf{SYM}$ circuit C' of size at most $O\left(k^{2d}\binom{S(n)}{O(k)}^d poly(S(n))\right)$. For the running time analysis, consider the following parts: (1) Checking all d-tuples of a pair of subsets of threshold gates of size at most k, (2) Obtaining an $\mathsf{OR} \circ \mathsf{AND} \circ \{\mathsf{SYM}, \mathsf{THR}\}$ circuit stated in the above, and (3) Running the procedure in Claim 4.8 on this $\mathsf{OR} \circ \mathsf{AND} \circ \{\mathsf{SYM}, \mathsf{THR}\}$ circuit. Thus, the time complexity is $O\left(k^{2d}\binom{S(n)}{O(k)}^d poly(S(n))\right)$. We note that there are efficient ways for listing all pairs of independent gates and implementing Lemma 3.22 than the trivial brute-force. The former can be executed by solving an ILP instance with two linear constraints for each pair of independent gates, and the later can be done by checking all small size subsets of threshold gates. $\square$

Lemma 4.19. Let $d = poly \log n$. There is a procedure such that for given $\mathcal{C}_k[d]$ circuit C of size $S(n) = 2^{n^{o(1)}}$ it outputs an abbreviated circuit C' and an Induced Hasse Diagram Family $\mathcal{F}$ of C' such that C is equivalent to C'. Moreover, there exist some $\varepsilon > 0$ and some $\gamma > 0$ such that it runs in time $2^{n-\Omega(n^\varepsilon)}$ for $k \leq n^\gamma$.

Proof overview. We give an outline of our algorithm.

We first explain a simple procedure which is incorporated to our algorithm. For given $\mathcal{C}_k[d]$ circuit C and two gates $G_1, G_2 \in V_i$, where V_i is the i-th threshold layer in C and $G_1(x) = G_2(x)$ for any input x, it outputs a $\mathcal{C}_k[d]$ circuit which is equivalent to C. Essentially, this procedure replaces the gate G_2 with G_1. We call $V_{i+1} \cup \cdots \cup V_d$ the *upper layers than the i-th threshold layer.* The following is a description of this procedure.

1. For each threshold gate T in the upper layers than the i-th threshold layer, if there is an input wire from G_2 then the label $w_{G_2} y_{G_2} + \sum_{U \neq G_2} w_U y_U \geq t_T$ in the gate T is replaced with $w_{G_2} y_{G_1} + \sum_{U \neq G_2} w_U y_U \geq t_T$, and *a wire is drawn from G_1 to T*, where each U is a source of T.

2. For the top SYM gate S, if there is an input wire from G_2 then the label $y_{G_2} + \sum_{U \neq G_2} y_U \in S_1$ in the gate S is replaced with $y_{G_1} + \sum_{U \neq G_2} y_U \in S_1$, and *a wire is drawn from the gate G_1 to S*, where S_1 is the characteristic set of the symmetric gate S.

3. Remove G_2 and all input and output wires of G_2 from C.

We call this procedure *abbreviation procedure.* We note that eliminating threshold gates in a threshold layer does not increase the size of maximum independent gate sets. We explain our approach to make our algorithm. Our algorithm progresses from the bottom layer to the top layer step by step. For each i, the $(i+1)$-th threshold layer is abbreviated by using an i-th induced Hasse diagram family.

We give the complete proof of Lemma 4.19 as follows.

Proof of Lemma 4.19.

We consider the following procedure about a bottom up construction of Induced Hasse Diagram Family.

1. Let $\mathcal{F}$ be $\emptyset$. For $i = 1, 2, ..., d$, let V_i be the i-th k-THR layer in C, and let E_i be $\emptyset$, and do the following steps **2**., **8**., and **9**..

- **2**. For any $G_1, G_2 \in V_i$, do the following steps **3**., **4**., **5**., **6**., and **7**.
  - **3**. Let C_1, C_2 be $\mathsf{THR} \circ (k\text{-}\mathsf{THR})^{i-1}$ sub-circuits in C whose top gates are G_1 and G_2, respectively, if $i \geq 2$. Let C_1, C_2 be threshold gates G_1, G_2, respectively, if $i = 1$.
  - **4**. For $b = 1, 2$, transform the circuit C_b to an $\mathsf{AC}^0 \circ \mathsf{SYM}$ circuit C'_b, by running the procedure in Lemma 4.18 on the input C_b and $\mathcal{F} = \{H_1, ..., H_{i-1}\}$ for any $i \geq 2$ or by running the procedure in Claim 4.8 on the input threshold gate C_b for $i = 1$.
  - **5**. Call the counting algorithm in Remark 4.7 to check the satisfiability of the two $\mathsf{AC}^0 \circ \mathsf{SYM}$ circuits A_1:$\neg C'_1 \wedge C'_2$ and A_2:$\neg C'_2 \wedge C'_1$.
  - **6**. If it outputs "Unsatisfiable" for A_2 (i.e. $C'_1(x) \leq C'_2(x)$ for any input string x) then $E_i := E_i \cup \{(G_1, G_2)\}$.
  - **7**. Else if it outputs "Unsatisfiable" for A_1 (i.e. $C'_2(x) \leq C'_1(x)$ for any input string x) then $E_i := E_i \cup \{(G_2, G_1)\}$.
- **8**. For each G_1, G_2 in C, if both (G_1, G_2) and (G_2, G_1) are in E_i then run the abbreviation procedure on C, G_1, and G_2. Let C be the resulting circuit (with no pair of equivalent gates in the i-th k-THR layer).
- **9**. Let H_i be (V_i, E_i) and let $\mathcal{F}$ be $\mathcal{F} \cup \{H_i\}$.

10. Output C (with no pair of equivalent threshold gates in any k-THR layer) and $\mathcal{F} = \{H_i : H_i = (V_i, E_i)\ (1 \leq i \leq d)\}$.

Running time analysis is as follows. The most dominant contribution to the entire running time is in the step testing dependency of two circuits. Note that we can construct an $\mathsf{AC}^0 \circ \mathsf{SYM}$ circuit with n inputs and of size $S_1(n) = O\left(k^{2d}\binom{S(n)}{O(k)}^d poly(S(n))\right)$ by Lemma 4.18. By Theorem 4.7 and Remark 4.7, there is some $\varepsilon > 0$ such that an algorithm can count the satisfying assignments to $\mathsf{AC}^0 \circ \mathsf{SYM}$ circuits of size 2^{n^ε} and runs in 2^{n-n^ε} time. We can take sufficiently small constant $\gamma > 0$ such that $S_1(n) \leq 2^{n^\varepsilon}$ for $k \leq n^\gamma$. Thus the running time in step **5**. is at most 2^{n-n^ε}. The entire running time is at most $d \cdot (O\left(\binom{S(n)}{2}\right)$

$\cdot(poly(S(n)) + k^{2d}\binom{S(n)}{O(k)}^d poly(S(n)) + 2^{n-n^{\varepsilon}}))$, for some constant $\varepsilon > 0$. Note that $S(n) = 2^{n^{o(1)}}$. Therefore, there exist some $\varepsilon > 0, \gamma > 0$ such that the entire running time is at most $2^{n-\Omega(n^{\varepsilon})}$ for $k \leq n^{\gamma}$. $\square$

Finally, we give the proof of Lemma 4.14.

Proof of Lemma 4.14. By the procedure in Lemma 4.19, for given input circuit C with depth $d = poly \log n$ and size $S(n) = 2^{n^{o(1)}}$, we compute an Induced Hasse Diagram Family $\mathcal{F}$ and an abbreviated circuit C_1 such that C and C_1 are equivalent. By Lemma 4.18, we obtain an $\mathsf{AC}^0 \circ \mathsf{SYM}$ circuit C_2 with size $S_2(n) = O\left(k^{2d}\binom{S(n)}{O(k)}^d poly(S(n))\right)$. By $S(n) = 2^{n^{o(1)}}$ and $d = poly \log n$, we have $S_2(n) = 2^{n^{o(1)}}$. Thus, there exist $\varepsilon_1, \gamma > 0$ such that this transformation from a $\mathcal{C}_k[d]$ circuit C to C_2 runs in time $2^{n-\Omega(n^{\varepsilon_1})}$ for $k \leq n^{\gamma}$. Finally, run the algorithm in Theorem 4.7 on C_2. There is $\varepsilon_2 > 0$ such that the running time of this algorithm is $2^{n-\Omega(n^{\varepsilon_2})}$. There exist $\varepsilon = \min_{i=1,2} \varepsilon_i$ and $\gamma > 0$ such that counting satisfying assignments to given $\mathcal{C}_k[d]$ circuit C can be done in time $2^{n-\Omega(n^{\varepsilon})}$ for $k \leq n^{\gamma}$. $\square$

4.5 Lower Bounds for $\mathsf{ACC} \circ \mathsf{THR} \circ (O(k)\text{-}\mathsf{THR})^d$ circuits

In this section, we give the proof of the following theorem, extending results of [46].

Theorem 4.20. There is some constant $\gamma > 0$ such that NEXP does not have any family of $\mathsf{ACC} \circ \mathsf{THR} \circ (O(k)\text{-}\mathsf{THR})^d$ circuits with n inputs and size $2^{\log^{O(1)} n}$ for $k \leq n^{\gamma}$.

We note that the parameter k depends on the number of variables n. We also note that it is sufficient to prove the statement of this theorem for the circuit class $\mathsf{ACC} \circ \mathsf{SYM} \circ (O(k)\text{-}\mathsf{THR})^d$, because each threshold gate in the threshold layer without restriction can be replaced with an $\mathsf{AC}^0 \circ \mathsf{MAJ}$ circuit of polynomial size by Claim 4.8 and because the class AC^0 is a subclass of ACC and a MAJ gate is symmetric. Thus, in this section, we will only consider the class $\mathsf{ACC} \circ \mathsf{SYM} \circ (O(k)\text{-}\mathsf{THR})^d$.

Recall two key ideas which are applied to prove Theorem 4.12: (1) closure property of a class of circuits, and (2) a procedure to count the number of satisfying assignments of given circuit which is in the circuit class. We give proofs of the following two lemmas for the circuit class $\mathsf{ACC} \circ \mathsf{SYM} \circ (O(k)\text{-}\mathsf{THR})^d$.

Lemma 4.21. The class of $\mathsf{ACC} \circ \mathsf{SYM} \circ (O(k)\text{-}\mathsf{THR})^d$ circuits with n inputs and size $2^{\log^{O(1)} n}$ is weakly closed under AND.

Lemma 4.22. Let $d = poly \log n$. There is a procedure that runs for given $\mathsf{ACC} \circ \mathsf{SYM} \circ (O(k)\text{-}\mathsf{THR})^d$ circuit of size $2^{n^{\varepsilon}}$ and outputs the number of satisfying assignments of the input circuit. Moreover, there exist some constants $\varepsilon > 0$ and $\gamma > 0$ such that $2^{n-\Omega(n^{\varepsilon})}$ for $k \leq n^{\gamma}$.

We note that Lemma 4.21 and Lemma 4.22 are respectively analogous to Lemma 4.13 and Lemma 4.14.

Proof of Lemma 4.21 We prove that there is a polynomial time procedure such that for given two $\mathsf{ACC} \circ \mathsf{SYM} \circ (k\text{-}\mathsf{THR})^d$ circuits C_1 and C_2, the procedure outputs an $\mathsf{ACC} \circ \mathsf{SYM} \circ (2k\text{-}\mathsf{THR})^d$ circuit C_3. Note that the AND of two ACC circuits is also an ACC circuit. By similar arguments in the proof of Lemma 4.13, the size of independent threshold gate sets in C_3 is at most $2k$, completing the proof of this lemma. $\square$

We give the following lemma about transformation of circuits.

Lemma 4.23. Let $d = \mathit{poly} \log n$. There exists a procedure that converts given $\mathsf{ACC} \circ \mathsf{SYM} \circ (k\text{-}\mathsf{THR})^d$ circuit with n inputs and size $O(S(n))$ to an $\mathsf{ACC} \circ \mathsf{SYM}$ circuit with n inputs and size at most $O(k^{2d}\binom{S(n)}{O(k)}^d \mathit{poly}(S(n)))$. Moreover, this procedure runs in time $O\left(k^{2d}\binom{S(n)}{O(k)}^d \mathit{poly}(S(n))\right)$.

Proof. Let $\mathcal{C}$ and $\mathcal{G}$ be respectively a class of circuits and a type of gates. Let $\{\mathcal{C}, \mathcal{G}\}$ denote a layer of circuits such that $\mathcal{C}$ or $\mathcal{G}$ is in the layer, where we regard a single gate as a circuit. We note that Lemma 4.23 is analogous to Lemma 4.18 and the proof is also similar to one of Lemma 4.18. By using Lemma 3.22, we construct a procedure that converts given $\mathsf{ACC} \circ \mathsf{SYM} \circ (O(k)\text{-}\mathsf{THR})^d$ circuit C_1 with n inputs and size $O(S(n))$ to an $\mathsf{OR} \circ \mathsf{AND} \circ \{\mathsf{ACC} \circ \mathsf{SYM}, \mathsf{THR}\}$ circuit C_2 with n inputs and size at most $O(k^{2d}\binom{S(n)}{O(k)}^d \mathit{poly}(S(n)))$.

Note that the layer of AND gates in C_2 corresponds to a set of ILP instances such that (1) each ILP instance S is obtained by fixing outputs of threshold gates according to a validly ordered restriction and (2) such validly ordered restriction agrees with the outputs of threshold gates which are obtained by feeding a feasible solution of S to the circuit C_1. We also note that the layer of OR gates in C_2 corresponds to the union of ILP instances for all possible validly ordered restrictions of C_1. Finally, we transform any threshold gate in C_2 to an $\mathsf{AC}^0 \circ \mathsf{MAJ}$ circuit by Claim 4.8. Since the class of $\mathsf{ACC} \circ \mathsf{SYM}$ circuits contains the one of $\mathsf{AC}^0 \circ \mathsf{MAJ}$ circuits, we obtain an $\mathsf{OR} \circ \mathsf{AND} \circ \mathsf{ACC} \circ \mathsf{SYM}$ circuit, i.e., an $\mathsf{ACC} \circ \mathsf{SYM}$ circuit. $\square$

We need another lemma to compute an Induced Hase Diagram Family and an abbreviated circuit for given input circuit. The following lemma corresponds to Lemma 4.19.

Lemma 4.24. Let $d = \mathit{poly} \log n$, where n is the number of input variables. There is a procedure such that for given $\mathsf{ACC} \circ \mathsf{SYM} \circ (O(k)\text{-}\mathsf{THR})^d$ circuit C with n inputs and size $S(n) = 2^{n^{o(1)}}$, the procedure outputs an abbreviated circuit C' and an Induced Hase Diagram Family $\mathcal{F}$ of C' such that C' is equivalent to C. Moreover, there exist some $\varepsilon > 0$ and some $\gamma > 0$ such that it runs in time $2^{n - \Omega(n^{\varepsilon})}$ for $k \leq n^{\gamma}$.

Proof Sketch. The proof is similar to the one of Lemma 4.19. Recall the *abbreviation procedure* in the proof of Lemma 4.19. By abbreviation procedures, pairs of equivalent threshold gates are eliminated from given input circuit. Our algorithm computes from the bottom layer to the top layer step by step. For each i, the $(i+1)$-th threshold layer is abbreviated by using an i-th Hasse diagram family. Remind that we call a d-th induced Hasse diagram family an *induced Hasse diagram family.* We use the algorithm counting the number of satisfying assignments for $\mathsf{ACC} \circ \mathsf{SYM}$ circuits in order to check if a pair of threshold gates in a threshold layer of C is an equivalent one. A direct edge in Hasse diagram corresponds to a pair of threshold gates having dependency, and we can decide if there is a direct edge between two threshold gates which are not equivalent by this counting algorithm for $\mathsf{ACC} \circ \mathsf{SYM}$ circuits.

We note that there is some constant $\varepsilon > 0$ such that the counting the number of assignments for given $\mathsf{ACC} \circ \mathsf{SYM}$ circuit of size $2^{n^{\varepsilon}}$ runs in time $2^{n-n^{\varepsilon}}$. Thus, there is some sufficiently small constant $\gamma > 0$ such that the size of circuits which are outputted by the transformation procedure in Lemma 4.23 is at most $2^{n^{\varepsilon}}$. Thus, we have that there exist some $\varepsilon > 0$ and some $\gamma > 0$ such that the procedure which computes an abbreviated circuit and its induced Hasse diagram family runs in time $2^{n-\Omega(n^{\delta})}$. □

Finally, we give a counting procedure to prove Lemma 4.22.

Proof of Lemma 4.22. We take the same approach to prove Lemma 4.14. By the procedure in Lemma 4.24, for given input circuit C with depth $d = \mathit{poly} \log n$ and size $S(n) = 2^{n^{o(1)}}$, we obtain an Induced Hasse Diagram Family $\mathcal{F}$ and an abbreviated circuit C_1 such that C and C_1 are equivalent. By Lemma 4.23, we have an $\mathsf{ACC} \circ \mathsf{SYM}$ circuit C_2 with size $S_2(n) = O\left(k^{2d}\binom{S(n)}{O(k)}\mathit{poly}(S(n))\right)$.

By $S(n) = 2^{n^{o(1)}}$ and $d = \mathit{poly} \log n$, it holds that $S_2(n) = 2^{n^{o(1)}}$. Hence, there exist $\varepsilon_1, \gamma > 0$ such that this transformation from an $\mathsf{ACC} \circ \mathsf{SYM} \circ (O(k)\text{-}\mathsf{THR})^d$ circuit to C_2 runs in time $2^{n-\Omega(n^{\varepsilon_1})}$ for $k \leq n^{\gamma}$. Running the algorithm in Theorem 4.7 on C_2, there is some $\varepsilon_2 > 0$ such that this algorithm runs in time $2^{n-\Omega(n^{\varepsilon_2})}$. Thus, we have that there is some constant $\varepsilon > 0$ such that counting satisfying assignments to $\mathsf{ACC} \circ \mathsf{SYM} \circ (O(k)\text{-}\mathsf{THR})^d$ can be done in time $2^{n-\Omega(n^{\varepsilon})}$. □

Chapter 5

Conclusion

In this thesis, we give several results about algorithms and lower bounds for Boolean circuit which is one of the most fundamental computation models. In particular, we study threshold circuits and nonuniform computation models, because a basic purpose of computing theory is to grasp differences between uniform computation and nonuniform one.

We show a nontrivial algorithm for a class of depth two threshold circuits, which is larger than the class of depth two sparse threshold circuits in [25]. We also prove lower bounds for nonuniform circuit classes containing threshold gates by using the criteria which Williams developed in [46].

In Lemma 4.18, we implicitly prove that any boolean function computed by restricted circuits of our form can be computed by $\mathsf{OR} \circ \mathsf{AND} \circ \{\mathsf{SYM}, \mathsf{THR}\}$ circuits of exponential size. Currently, there are no known exponential size lower bounds for $\mathsf{OR} \circ \mathsf{AND} \circ \{\mathsf{SYM}, \mathsf{THR}\}$ circuits, and a direct application of [46] is not workable because any exponential function is not a *half-exponential-type* function. We give an explanation to understand the notion of half-exponential-type functions.

The following claim is important to understand the relationship between the concept of half-type-exponential functions and William's lower bound method through witness circuits.

Claim 5.1. Let $\mathcal{C}$ be any circuit class. If P has nonuniform $\mathcal{C}$ circuits of $S(n)^{O(1)}$ size, then there is a constant $c > 0$ such that any circuit family of size $T(n)$ (uniform or not) has an equivalent $\mathcal{C}$ circuit family of size $S(n + O(T(n) \log T(n)))^c$.

Proof. If P has nonuniform $S(n)^{O(1)}$-size $\mathcal{C}$ circuits, then there is some constant $c > 0$ such that the *Circuit Evaluation* problem to decide if $C(x) = 1$ for given pair of a circuit C with N inputs and a boolean string x of length N can be solved by $S(n)^c$-size circuits. Let $\{D_n(\cdot, \cdot)\}_{n \in \mathbb{N}}$ be a circuit family of size $S(n)^c$ for this problem. Now let $\{C_n\}_{n \in \mathbb{N}}$ be an arbitrary circuit family of size $T(n)$. Define $C_{|x|}(x) = D_{n_1}(C_{|x|}, x)$ for an appropriate length $n_1 \leq n + O(T(n) \log T(n))$. Thus, we obtain an equivalent $\mathcal{C}$ circuit family of size $S(n + O(T(n) \log T(n)))^c$. □

The composition of functions $S(n + O(T(n) \log T(n)))^c$ is an origin of the notion of half-type-exponential functions. The notion of half-exponential-type functions is formally defined as follows.

Definition 5.2. A function $f : \mathbb{N} \to \mathbb{N}$ is said to be *sub-half-exponential*, if for any constant k it holds that $f(f(n^k)^k)^k \leq 2^{n^{o(1)}}$.

A function $f : \mathbb{N} \to \mathbb{N}$ is said to be *sub-third-exponential*, if for any constant k it holds that $f(f(f(n^k)^k)^k)^k \leq 2^{n^{o(1)}}$.

We can easily understand that an arbitrary quasi polynomial function $q(n) = 2^{poly \log n}$ is sub-third-exponential. We can also understand that the function $S(n) = 2^{n^{o(1)}}$ is not sub-third-exponential.

The following was originally conjectured by Impagliazzo, and proved in [45].

Theorem 5.3. Let S(n) be an arbitrary sub-half-exponential function such that $S(n) \geq n$ for all n. If $\mathsf{NTIME}[2^n]$ has $S(n)$ size circuits, then any language in NEXP has universal witness circuits of size $S(S(n)^c)^c$ for some constant c depending on the language.

The following is proved in [45] by using this result.

Corollary 5.4. If $\mathsf{NTIME}[2^n]$ has $S(n)$-size ACC circuits, then any NEXP language has universal witness circuits of $S(S(S(n)^c)^c)^c$. for some c depending on the language.

Thus, we currently do not rule out a circuit family of size $S(n) = 2^{n^{o(1)}}$ to compute NEXP languages, and a direct application of [46] is not workable because any exponential function is not a half-exponential-type function.

An obvious direction for the future is to develop methods to rule out $2^{n^{o(1)}}$-size circuits computing NEXP languages.

Another possible direction for the future is to rule out nonuniform circuits for PSPACE languages. Recall Theorem 2.19, which is a critical tool for the lower bound method based on the witness circuits.

Theorem 2.19(restated)

$$\mathsf{PSPACE} \subseteq \mathsf{P/poly} \Rightarrow \mathsf{PSPACE} = \mathsf{MA}$$

In the proof of this theorem, it is important to solve Circuit Evaluation problem for a simulation of the prover in an interactive proof protocol. Circuit evaluation is P-complete if given input circuit is in the class of general Boolean circuits with basis AND, OR, and NOT gates, but we might obtain some meaningful results for some circuit class in which evaluating the output value of an arbitrary circuit can be done with small space complexity. If we can replace the complexity class MA in the statement of Theorem 2.19 with some complexity class related to bounded space computation, we might derive a contradiction to the space hierarchy theorem. We note that the only hierarchy theorem which is applied in [44, 45, 46] is the nondeterministic hierarchy theorem. We might prove different complexity class separations, if we are successful to apply the space hierarchy theorem.

Bibliography

[1] L. Adleman. Two theorems on random polynomial time. In Proc. FOCS'78, pages 75-83, 1978.

[2] E. Allender and Vivek Gore. On strong separations from AC^0. Fundamentals of Computation Theory, 8, 1991.

[3] K. Amano and A. Saito. A satisfiability algorithm for some class of dense depth two threshold circuits, IEICE Trans. Inf. Sys., E98-D, No.1: 108-118, 2015.

[4] K. Amano and A. Saito. A Nonuniform Circuit Class With Multilayer of Threshold Gates Having Super Quasi Polynomial Size Lower Bounds against NEXP, In Proc. LATA'15, to appear.

[5] S. Arora and B Barak, Computational Complexity: A Modern Approach, Cambridge University Press, 2009.

[6] R. Beigel and J. Tarui. On ACC. Computational Complexity 4:350-366, 1994.

[7] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu, Symbolic modelchecking without BDDs, in Tools and Algorithms for the Construction and Analysis of Systems, March 1999, 193-207.

[8] R. B. Boppana and M. Sipser. Handbook of theoretical computer science (vol. a).chapter The complexity of finite functions, 757-804. MIT Press, Cambridge, 1990.

[9] C. Calabro. The exponential complexity of satisfiability problems. PhD thesis, University of California, San Diego, 2009.

[10] C. Calabro, Russell Impagliazzo, and Ramamohan Paturi. The complexity of satisfiability of small depth circuits. In Parameterized and Exact Computation: 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers, 75-85, 2009. Springer-Verlag.

[11] A. K Chandra, Larry Stockmeyer, and Uzi Vishkin. Constant depth reducibility. SIAM J. on Comput, 13(2):423-439, 1984.

[12] S. Cook. Short propositional fomulas represent nondeterministic computations. Information Processing Letters, 26(5):269-270, 1988.

[13] S. Cook. The complexity of theorem-proving procedures. In Proceedings of the Third Annual ACM Symposium on Theory of Computing, 151-158, 1971.

[14] D. Coppersmith. Rapid multiplication of rectangular matrices. SIAM J. Comput. 11(3):467-471, 1982.

[15] E. Dantsin and A. Wolpert. Max-SAT for formulas with constant clause density can be solved faster than in $O(2^n)$ time. In Armin Biere and CarlaP. Gomes, editors, Theory and Applications of Satisfiability Testing - SAT 2006, volume 4121 of Lecture Notes in Computer Science, 266-276. Springer Berlin Heidelberg, 2006.

[16] M. Furst, J.Saxe, and M.Sipser. Parity, circuits, and the polynomial time hierarchy. Mathematical Systems Theory 17:13-27, 1984.

[17] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. In Proc. STOC'85, pages 291-304, 1985.

[18] A. Hajnal, W. Maass, P. Pudlak, M. Szegedy, and G. Turan. Threshold circuits of bounded depth. In Proc. FOCS'87, 99-110, 1987.

[19] J. Håstad. Almost optimal lower bounds for small depth circuits. Advances in Computing Research 5:143-170, 1989.

[20] R. Impagliazzo, V. Kabanets, A. Wigderson. In search of an easy witness: exponential time versus probabilistic polynomial time. J. Comput. and Sys. Sci. 65(4):672-694, 2002.

[21] R. Impagliazzo, W. Matthews, and R. Paturi. A Satisfiability Algorithm for AC0. In Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, 2012.

[22] R. Impagliazzo and R. Paturi. The complexity of k-SAT. Journal of Computer and Systems Sciences, 62(2):367-375, March 2001. Preliminary version in 14th Annual IEEE Conference on Computational Complexity, 237-240, 1999.

[23] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? Journal of Computer and System Sciences, 63:512-530, 1998.

[24] R. Impagliazzo, R. Paturi, and M. E. Saks. Size-depth tradeoffs for thresholdcircuits. SIAM J. Comput. 26(3):693-707, 1997.

[25] R. Impagliazzo, R. Paturi, and Stefan Schneider. A satisfiability algorithm for sparse depth two threshold circuits. In FOCS'13, 479-488, 2013.

[26] R. Impagliazzo and A. Wigderson. Randomness vs. Time: De-randomization under a uniform assumption. J. Comput. and Sys. Sci., pages 734-743, 1998.

[27] K. Iwama and H. Morizumi. An explicit lower bounds of $5n - o(n)$ for Boolean circuits. In Proc. MFCS'02, Springer LNCS:2420:353-364, 2002.

[28] V. Kabanets. Easiness assumptions and hardness tests: Trading time for zero error. In Computational Complexity, 2000. In Proc. CCC'98, pages 150-157, 2000.

[29] R. Karp and R. Lipton. Some connections between nonuniform and uniform complexity classes. In STOC'80, pages 302-309, 1980.

[30] K. Lange, Unambiguity of circuits. Theoretical Computer Science 107(1993)77-94, 1993.

[31] L. Levin. Universal sorting problems. Problems of Information Transmission, 9:265-266, 1973.

[32] I. Lynce and J. Marques-Silva, Efficient haplotype inference with Boolean satisfiability, in National Conference on Artificial Intelligence, July 2006.

[33] N. Nisan and A. Wigderson. Hardness vs randomness. J. Comput. Sys. Sci., 49(2):149-167, October 1994.

[34] I. Oliverira. Algorithms versus circuit lower bounds. ECCC Technical Report TR13-117, 2013.

[35] J. Robson. An O(T log T) reduction from RAM computations to satisfiability. Theoretical Computer Science, 82(1):141-149, 1991.

[36] A. Razborov. Lower bounds on the size of bounded depth networks over a complete basis with logical addition. Mathematical Notes of Academy of Sciences USSR 41(4):598-607, 1987.

[37] R. Santhanam. Fighting perebor: New and improved algorithms for formula and qbf satisfiability. In Proc. FOCS'10, 183-192, 2010.

[38] R. Schuler. An algorithm for the satisfiability problem of formulas in conjunctive normal form. Journal of Algorithms, 54(1):40-44, 2005.

[39] A. Shamir. IP = PSPACE. Journal of the ACM, 39(4):869-877, October 1992.

[40] A. Smith, A. G. Veneris, M. F. Ali, and A. Viglas, Fault diagnosis and logic debugging using Boolean satisfiability, IEEE Transactions on Computer-Aided Design, vol. 24, no. 10, . 1606-1621, 2005.

[41] R. Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In Proc. STOC'87, 77-82, 1987.

[42] L.J. Stockmeyer and A.R. Meyer. Word problems requiring exponential time. In Proc. STOC'73, pages 1-9, 1973.

[43] R. Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. Theoretical Computer Science, 348:357-365, 2005.

[44] R. Williams. Improving exhaustive search implies superpolynomial lower bounds. In Proc. STOC'10, 231-240, 2010.

[45] R. Williams. Non-Uniform ACC Circuit Lower Bounds. Journal of the ACM 61(1), Article 22, January 2014.

[46] R. Williams. New algorithms and lower bounds for circuits with linear threshold gates, In Proc. STOC'14, 194-102, 2014.