論文 / 著書情報
Article / Book Information

Title	Local Restrictions from the Furst-Saxe-Sipser Paper
Authors	Suguru Tamaki, Osamu Watanabe
Citation	Theory of Computing Systems, Vol. 60, No. 1, Page 20-32
Pub. date	2017, 1
Note	This is a post-peer-review, pre-copyedit version of an article published in Theory of Computing Systems. The final authenticated version is available online at: http://dx.doi.org/10.1007/s00224-016-9730-0.

Noname manuscript No.
(will be inserted by the editor)

Local Restrictions from the Furst-Saxe-Sipser Paper

Suguru Tamaki · Osamu Watanabe

Received: date / Accepted: date

Abstract In their celebrated paper [Mathematical Systems Theory 17(1): 13–27 (1984)], Furst, Saxe and Sipser used *random restrictions* to reveal the weakness of Boolean circuits of bounded depth, establishing that constant-depth and polynomial-size circuits cannot compute the parity function. Such *local restrictions* have played important roles and found many applications in complexity analysis and algorithm design over the past three decades. In this article, we give a brief overview of two intriguing applications of local restrictions: the first one is for the Isomorphism Conjecture and the second one is for moderately exponential time algorithms for the Boolean formula satisfiability problem.

Keywords local restriction · random restriction · computational complexity theory · algorithm design

1 Introduction

A *local restriction* on an n-ary function $f(x_1, \ldots, x_n)$ is in general a way to reduce the domain of f by restricting the range of its input variables locally.

The first author is supported in part by MEXT KAKENHI (24106003); JSPS KAKENHI (26330011, 16H02782), and the second author is supported in part by MEXT KAKENHI (24106008).

S. Tamaki
Kyoto University, Yoshida Honmachi, Sakyo-ku, Kyoto 606-8501, Japan
Tel.: +81-75-753-5392
Fax: +81-75-753-5972
E-mail: tamak@kuis.kyoto-u.ac.jp

O. Watanabe
Tokyo Institute of Technology, Meguro-ku Ookayama, Tokyo 152-8552, Japan
Tel.: +81-3-5734-3210
Fax: +81-3-5734-2688
E-mail: watanabe@is.titech.ac.jp

In this article, we consider the simplest one, that is, a restriction obtained by assigning values to some of its variables. Such local restrictions may be one of the most commonly used techniques in mathematics and computer science; for example, inductive arguments are often based on local restrictions. Here we focus on Boolean functions and the usage of local restrictions in the field of computational complexity and algorithm design. In particular, we explain local restriction techniques in relation to the celebrated paper by M. Furst, J. Saxe, and M. Sipser [12]. In that paper they demonstrated the usefulness of "random restriction" for analyzing Boolean functions by showing their famous circuit lower bound result[1]. Since then, the analysis of the complexity of Boolean functions by using their random restriction technique has been developed to one of the main streams of computational complexity theory. Naturally there have been several excellent expository materials on this topic, such as a survey paper by P. Beame [7], a book by R. O'Donnell [11], and so on. Thus, in this short article, we would like to explain rather unique examples of local restriction techniques that might have been less focused in the literature; another interesting examples related to the Furst-Saxe-Sipser paper.

We briefly review the Furst-Saxe-Sipser paper and its subsequent research. Recall some notions on circuit complexity. A *circuit* consists of input gates, NOT-gates, and unbounded fan-in AND- and OR-gates, where one of these gates with no fan-out wire is regarded as an output gate. Consider any circuit C, and let n be the number of its input gates. Then for any binary sequence $a_1, \ldots, a_n$ the value computed at its output gate when each a_i is given at each input gate is regarded as the output of C on the input $a_1, \ldots, a_n$ and denoted by $C(a_1, \ldots, a_n)$. In computational complexity we would usually consider a function that takes a binary sequence of any length, and in order to treat this formally, we need to consider a family of Boolean functions such as $\{\mathrm{parity}_n(x_1, \ldots, x_n)\}_{n\geq 1}$ and a family $\{C_n\}_{n\geq 1}$ of circuits for computing each member of such a function family. But here we argue more simply by using a function parity_n and a circuit C_n as representatives of such families. Note that n is regarded as an input size, and it is used as a size parameter in the following. The *size* of circuits C_n's is a function mapping n to the number of gates of C_n. We may assume that a circuit is structured so that it consists of layers of OR- and AND-gates with *literal gates* (i.e., input gates or their negations) at the bottom layer. We can modify a given circuit without increasing circuit size so much; all negation gates can be moved to the bottom by using De Morgan's law, and any two consecutive AND- (resp., OR-) gates can be merged because the fan-in of OR- and AND-gates is unbounded. Thus, the *depth* of a circuit is simply the number of its layers. Then the class AC^0 is defined as the class of Boolean functions that can be computed by constant-depth and polynomial-size circuits.

While circuits can be used as a simple and concrete computation model for discussing computational complexity, it is very difficult to prove nontrivial

[1] It should be noted at this point that M. Ajtai independently obtained the corresponding result by using a similar notion in a different contest [6]. Also we note here that Subbotovskaya [24] used random restrictions in computational complexity in early 60's.

circuit size lower bounds in general. For example, it is a long standing open problem until now to show even a nonlinear lower bound on the size of circuits computing a function not only in P but also in a much higher class such as NP. Furst, Saxe, and Sipser [12] showed that the situation could be improved drastically if we introduce the constant-depth restriction. They proved a superpolynomial-size lower bound for computing the parity, the majority, etc by constant-depth circuits. For the proof, they used random restrictions.

We recall the notion of random restriction and explain how they proved the lower bound. For any Boolean function $f(x_1, \ldots, x_n)$, a local restriction on f we consider in this article can be expressed as a function ρ from $\{1, \ldots, n\}$ to $\{0, 1, *\}$; for each i, if $\rho(i) \in \{0, 1\}$, then assign $\rho(i)$ to the variable x_i, and otherwise (i.e., if $\rho(i) = *$), then *unset* the variable, that is, leave it as it is. Let $f|_\rho$ denote the function obtained by this restriction. We can apply this random restriction ρ also to a circuit C having n input gates and *simplify* it, which is denoted by $C|_\rho$; for example, if an AND-gate having an input gate that is assigned 1 by ρ can be simplified to the constant 1 in $C|_\rho$, which can be used for the further simplification. For a given parameter p, $0 < p < 1$, a *random restriction* used in [12] is a random function ρ that, for each i, independently assigns $\rho(i)$ with probabilities $\Pr[\rho(i) = *] = p$ and $\Pr[\rho(i) = 0] = \Pr[\rho(i) = 1] = (1-p)/2$. The key fact shown in [12] is that one can choose p appropriately so that an AND of small OR's can be written as an OR of small AND's (and vice versa) while keeping enough number of unset input gates. This fact was used to derive a contradiction from an assumption that the function parity_n is computable by a depth d and polynomial-size circuit C_n. By applying some appropriate restriction ρ to C_n, it can be shown that the AND and OR of the bottom two layers can be switched in $C_n|_\rho$, which gives a depth $d-1$ circuit $C'_{n'}$ that can be used as a polynomial-size circuit for $\mathrm{parity}_{n'}$ because $\mathrm{parity}_n|_\rho$ is still a parity function, i.e., $\mathrm{parity}_{n'}$ (or its negation) for a smaller input size n'. Then by induction on d we can derive a contradiction because we know that there is no polynomial-size depth 2 circuits computing the parity [18].

Their result is one of few superpolynomial circuit size lower bounds, and it is important by itself. But their proof technique may be more influential, and it has been developed to a fruitful research area in computational complexity. Soon after their paper, the essence of their proof technique has been formulated into a more sophisticated and stronger form by several researchers, and the most powerful one was given by Håstad [13], which is now called as *Switching Lemma*. Roughly speaking, by the Switching Lemma we can show that any small depth and small size circuit gets simplified very much by a random restriction with non small probability. For example, for any constant-depth and polynomial-size circuit C_n, consider a random restriction ρ w.r.t. the probability parameter $p = n^{-1+\epsilon}$ for some appropriate choice of ϵ, $0 < \epsilon < 1$. Then it can be shown (see, e.g., [4]) that with probability close to 1, the simplified circuit $C_n|_\rho$ depends on only constant number of input gates. Note, on the other hand, $\mathrm{parity}_n|_\rho$ is still a parity function on approximately n^ϵ variables. Thus, obviously, $C_n|_\rho \neq \mathrm{parity}_n|_\rho$, and hence, $C_n \neq \mathrm{parity}$. From this observation, it is clear that no constant-depth and polynomial-size circuit computes the par-

ity. Furthermore, it suggests that every constant-depth and polynomial-size circuit can approximate a parity function very poorly; for example, $C_n|_\rho$ cannot give any hint to the value of $\mathrm{parity}_n|_\rho$ because the value of $\mathrm{parity}_n|_\rho$ can be either 0 or 1 even if we fix the values of variables that $C_n|_\rho$ can see. Linial, Mansour, and Nisan [17] strengthened this argument and succeeded to show the "simplicity" of AC^0 functions in the framework of the Fourier analysis of Boolean functions suggested in [15]. For a given set $X_n = \{x_1, \ldots, x_n\}$ of variables, one can show that the set of parity functions over all subsets of X_n forms a Fourier basis of Boolean functions; that is, any Boolean function can be expressed by a linear combination of some of such parity functions. In this framework, the number k of variables of a parity function $\mathrm{parity}_k(x_{i_1}, \ldots, x_{i_k})$ is called its *degree*, and a function that can be expressed by using only low degree parity functions is regarded as a *simple* function. In [17] they proved by using the Switching Lemma that every AC^0 function has small correlation to any parity function $\mathrm{parity}_t(x_{i_1}, \ldots, x_{i_t})$ if t is larger than $\mathrm{poly}(\log n)$. In other words, all AC^0 functions can be approximated well by a linear combination of polylog-degree parity functions. Based on this analysis, they proposed a quasi-polynomial-time algorithm for learning AC^0 functions. This became a starting point of various investigations on the Fourier analysis of Boolean functions; the reader can find these investigations in an excellent text book on this subject [11].

The results of Furst-Saxe-Sipser paper have been also greatly advanced. We only mention here some of the latest work so that the reader can get the other information from the references therein. For the size of depth $d+1$ circuits computing the parity, the superpolynomial lower bound of Furst-Saxe-Sipser paper has been improved to $2^{n^{\Omega(1/d)}}$ lower bound in [14]. More recently, Rossman [19] considered formula size and extended this lower bound result by showing depth $d+1$ formulas computing the parity have size $2^{\Omega(dn^{1/d})}$. Recall that a formula is a special type of circuit where each gate (except for literal gates) has at most one fan-out. He used an interesting variation of the random restriction. Note that Furst-Saxe-Sipser paper has another important result. Based on their superpolynomial circuit size lower bound, they constructed an oracle that defines a relativized world where the polynomial-time hierarchy is a strict subclass of PSPACE. Recently this oracle construction has been improved greatly, and we now have a "random oracle" that makes the polynomial-time hierarchy infinite [20]. Here again an interesting extension of the random restriction has been introduced to show a lower bound result needed for the oracle construction; see [21] for a detail survey on this topic.

These are major investigations after the Furst-Saxe-Sipser paper. Clearly there are many other interesting topics related to that paper. We explain two such examples in the following sections. The first one is yet another interesting application of the random restriction. In the context of the Isomorphism Conjecture, the random restriction has been used to show a beautiful observation that gives a new insight to the conjecture. The Isomorphism Conjecture claims that all NP-complete sets share a similar structure in common. Suppose

hypothetically that any pair of NP-complete sets are reducible to each other via computationally very weak reductions; then we could expect that all NP-complete sets have a similar structure because such weak reductions cannot change the structure of NP-complete sets. Agrawal, Allender, and Rudich [4] used the random restriction to prove that this situation indeed holds among all sets that are shown NP-complete via AC^0-reductions. The second one is about an application of a *deterministic* local restriction for designing an efficient algorithm for the linear-size formula satisfiability problem. (In this context, fan-in and gate types are important. We assume here that the fan-in of all gates are at most two. A formula defined by using only AND-, OR-, and literal gates are called a *De Morgan formula* whereas a *general formula* can use any gate type such as XOR-gate.) Santhanam [22] developed a new powerful approach for analyzing deterministic local restrictions, by which he obtained a subexponential-time satisfiability algorithm for linear-size De Morgan formulas. Following the work of Santhanam, Seto and Tamaki [23] proposed a similar subexponential-time algorithm for linear-size general formulas. Recall that the parity function (which is essentially a sequence of XOR-gates) is resistant against simplifications based on local restrictions, which in fact is the key to obtain the lower bound result of Furst-Saxe-Sipser. From this view point, their analysis of local restrictions to formulas with XOR-gates is unique and interesting. We explain these two examples a bit more in detail in the following two sections.

2 The Isomorphism Conjecture

Those who have studied several NP-completeness proofs would feel that NP-complete sets have a somewhat similar structure in common. Berman and Hartmanis [9] made this intuition precise by the following conjecture that is now called the *Isomorphism Conjecture* (also known as the Berman-Hartmanis Conjecture).

Conjecture *All NP-complete sets are polynomial-time isomorphic to each other.*

They showed the following powerful tool for proving a polynomial-time isomorphism; this tool is strong enough to convince us that all *known* (at that time) NP-complete problems are polynomial-time isomorphic.

Theorem 1 *For any sets A and B, if there are polynomial-time many-one reductions both from A to B and from B to A that are one-to-one, length-increasing, and polynomial-time invertible, then A and B are polynomial-time isomorphic.*

Clearly, if the Isomorphism Conjecture holds, then $\mathrm{P} \neq \mathrm{NP}$ because if P were equal to NP, then any set in P could be NP-complete, and then, for example, both $\{0\}$ and $\{0,1\}$ would be NP-complete, implying the Isomorphism

Conjecture fails since no one-to-one reduction exists between them. Thus, proving the Isomorphism Conjecture is at least as hard as proving P $\neq$ NP. Nevertheless, researchers study the Isomorphism Conjecture because they believe that understanding the conjecture would provide us with some important clue why should P $\neq$ NP hold.

Though there are very good surveys, e.g., [16] for early research and [1] for more recent developments[2], we explain this topic because of an interesting application of the random restriction for proving one of the beautiful results on this topic.

2.1 Background: Basics and early research on the Isomorphism Conjecture

For discussing the Isomorphism Conjecture, we recall several key notions and introduce necessary notation. In this section we take a Turing machine based framework for computational complexity. A computational problem we consider here is a set recognition problems, that is, problems of deciding a given input string belongs to a specified set. A complexity class $\mathcal{C}$ is a class of subsets of $\{0,1\}^*$ whose recognition problems are solvable by using Turing machines of a certain class. We will abuse this notation for describing the corresponding machineries themselves. For example, we say "P-computable function", "AC^0 circuits", and so on, by which we mean a function computable by a polynomial-time Turing machine, constant-depth and polynomial-size circuits, etc. We will discuss the circuit complexity for computing a function on $\{0,1\}^*$, and for this, we need to revise our circuit model that is defined for computing Boolean functions. But since the revision is natural, we omit explaining it here.

For any two sets A and B, a *many-one reduction* from A to B is a total function $h : \{0,1\}^* \to \{0,1\}^*$ such that $x \in A \iff h(x) \in B$ holds for all $x \in \{0,1\}^*$; a reduction is further called, e.g., *one-to-one*, *length-increasing*, P-*computable* based on its property. The most basic one is a *polynomial-time many-one reduction* (in short, $\leq^{\mathrm{P}}_{\mathrm{m}}$-reduction) that is a P-computable many-one reduction. We say that A is *polynomial-time many-one reducible* (in short, $\leq^{\mathrm{P}}_{\mathrm{m}}$-reducible) to B if there exists a $\leq^{\mathrm{P}}_{\mathrm{m}}$-reduction from A to B. In general, for any complexity class $\mathcal{C}$ and for any $\leq^{r}_{\mathrm{m}}$-reducibility, we say that a set L is $\leq^{r}_{\mathrm{m}}$-*complete for* $\mathcal{C}$ if (i) all sets in $\mathcal{C}$ are $\leq^{r}_{\mathrm{m}}$-reducible to L and (ii) L is indeed in $\mathcal{C}$. For example, the NP-complete sets considered in the Isomorphism Conjecture are in this notation $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete sets for NP. We say that two sets A and B are *polynomial-time isomorphic* if A is $\leq^{\mathrm{P}}_{\mathrm{m}}$-reducible to B via f that is also one-to-one, onto, and P-invertible. (Note that this definition of isomorphism is symmetric because f^{-1} is indeed a one-to-one, onto, and P-invertible $\leq^{\mathrm{P}}_{\mathrm{m}}$-reduction from B to A.) Here again in order to specify a resource bound of reductions, we say that A and B are $\mathcal{C}$-*isomorphic* if there is a many-one reduction from A to B that is one-to-one, onto, and both $\mathcal{C}$-computable and $\mathcal{C}$-

[2] The technical exposition part of this article is mainly from this survey.

invertible. From the condition of Theorem 1, we introduce the following equivalence relation: we say that A and B are $\equiv^{\mathcal{C}}_{1,\mathrm{li},\mathrm{inv}}$-*equivalent* if they are many-one reducible to each other by using one-to-one, length-nondecreasing, and $\mathcal{C}$-computable and $\mathcal{C}$-invertible reductions. Theorem 1 shows that the $\equiv^{\mathrm{P}}_{1,\mathrm{li},\mathrm{inv}}$-equivalence is a sufficient condition for the P-isomorphism. We also define a slightly weaker equivalence relation by dropping the invertibility condition: We say that A and B are $\equiv^{\mathcal{C}}_{1,\mathrm{li}}$-*equivalent* if they are many-one reducible to each other by using one-to-one, length-nondecreasing, and $\mathcal{C}$-computable reductions. We will see below that this equivalence relation plays an important role in the investigation of the Isomorphism Conjecture.

Two approaches have been taken for investigating the Isomorphism Conjecture. The first approach is to consider a similar conjecture for a higher complexity class or in a relativized setting where we can investigate the conjecture technically without solving challenging open problems such as $\mathrm{P} \neq \mathrm{NP}$. The second approach, which has been studied more recently, is to consider reduction types weaker than the $\leq^{\mathrm{P}}_{\mathrm{m}}$-reducibility by restricting the resource for computing reductions. Though the second approach is where the random restriction has been used, let us review below the first approach briefly.

Many of the early results on the Isomorphism Conjecture have been obtained by studying the class EXP. For example, Berman [8] proved the following (see also [25] for a simpler proof and related discussions).

Theorem 2 *All* $\leq^{\mathrm{P}}_{\mathrm{m}}$*-complete sets for* EXP *are* $\equiv^{\mathrm{P}}_{1,\mathrm{li}}$*-equivalent.*

Recall that a function is called *one-way* in general if it is easy to compute but its inverse is hard to compute. To be specific, we say (in this article) that a length-nondecreasing and one-to-one function is $\mathcal{C}$*-one-way* if it is $\mathcal{C}$-computable and computing its inverse is not $\mathcal{C}$-computable. The $\equiv^{\mathrm{P}}_{1,\mathrm{li}}$-equivalence of the above theorem is quite close to the P-isomorphism because *if* no P-one-way function exits, then the $\equiv^{\mathrm{P}}_{1,\mathrm{li}}$-equivalence is the same as the $\equiv^{\mathrm{P}}_{1,\mathrm{li},\mathrm{inv}}$-equivalence, and hence from the above theorem and Theorem 1, the P-isomorphism holds for all $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete sets for EXP. On the other hand, if there exists a P-one-way function, then the $\equiv^{\mathrm{P}}_{1,\mathrm{li}}$-equivalence of Theorem 2 could be the best we can hope. To see this more clearly, suppose that there exists a one-to-one and length-nondecreasing P-one-way function f. Then for any $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete set C for EXP, it is easy to show that $f(C)$ is $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for EXP. The function f itself is a $\leq^{\mathrm{P}}_{\mathrm{m}}$-reduction from C to $f(C)$ that is indeed one-to-one and length-nondecreasing. But it seems difficult to have a many-one reduction from C to $f(C)$ that is one-to-one, length-nondecreasing, polynomial-time computable, *and also* polynomial-time invertible. That is, we have $C \equiv^{\mathrm{P}}_{1,\mathrm{li}} f(C)$ but it seems difficult to have $C \equiv^{\mathrm{P}}_{1,\mathrm{li},\mathrm{inv}} f(C)$. Note that the same argument holds for the class NP. In fact, Kurtz, Mahaney, and Royer [16] showed a relativized world where there exist a P-one-way function and an NP-complete set C such that $C \not\equiv^{\mathrm{P}}_{1,\mathrm{li},\mathrm{inv}} f(C)$ holds, and hence the Isomorphism Conjecture fails. We also believe that P-one-way functions exist. Then would it be better to revise the Isomorphism Conjecture by replacing the P-

isomorphism with the $\equiv^{\mathrm{P}}_{1,\mathrm{li}}$-equivalence? More recently, a new insight has been obtained by considering weaker reduciblities, that is, the second approach.

2.2 The Gap Theorem: Another interesting example of random restrictions

We explain the following beautiful result obtained by Agrawal, Allender, and Rudich [4]. (Further improvements have been made in [5,2].)

Theorem 3 *All* $\leq^{\mathrm{AC}^0}_{\mathrm{m}}$*-complete sets for* NP *are* AC^0*-isomorphic.*

This theorem gives two important messages on the Isomorphism Conjecture. Firstly, the existence of P-one-way functions may not immediately kill the conjecture. The theorem gives an example; the AC^0-isomorphism holds as claimed although we *know* that AC^0-one-way functions exist [10]. Secondly, the theorem suggests one approach toward the Isomorphism Conjecture. We can prove the conjecture (nonuniform version) by showing that all $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete sets for NP are indeed $\leq^{\mathrm{AC}^0}_{\mathrm{m}}$-complete. Note that we know that AC^0 is (in a sense) much weaker than P. But as we will see below, by using the power of NP-complete sets we may be able to simplify many-one reductions considerably. Unfortunately, it has been found later [5] that there exists some $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete set for NP that cannot be $\leq^{\mathrm{AC}^0}_{\mathrm{m}}$-complete for NP. But still, this approach may be worth considering.

We explain how the theorem was proved. As a result of a sequence of investigations following the second approach, Agrawal and Allender [3] proved that all $\leq^{\mathrm{NC}^0}_{\mathrm{m}}$-complete sets for NP are $\equiv^{\mathrm{AC}^0}_{1,\mathrm{li}}$-equivalent. Here NC^0 is[3] the class of constant-depth and polynomial-size circuits consisting of 2-fan-in Boolean gates. As a corollary, they also showed that all $\leq^{\mathrm{NC}^0}_{\mathrm{m}}$-complete sets for NP are AC^0-isomorphic by using a technique similar to Theorem 1. Then soon after the following remarkable theorem was given as a final step of the proof of Theorem 3.

Theorem 4 *Any* $\leq^{\mathrm{AC}^0}_{\mathrm{m}}$*-complete set for* NP *is also* $\leq^{\mathrm{NC}^0}_{\mathrm{m}}$*-complete.*

This theorem is called the *Gap Theorem*. Note that by the constant-depth and 2-fan-in restriction, an output of an NC^0 circuit cannot depend on more than some constant number of input gates. Surprisingly the Gap Theorem claims that such very weak circuits can replace AC^0 circuits. We can show this by using the random restriction of Furst-Saxe-Sipser in a unique way.

Consider any $\leq^{\mathrm{AC}^0}_{\mathrm{m}}$-complete set K for NP. Let A be any set in NP. Then we have a reduction from A to K that is computable by AC^0 circuits $\{C_n\}_{n\geq 1}$. For any sufficiently large n, consider the random restriction ρ to the circuit C_n with a parameter $p = n^{-1+\epsilon}$ for some appropriate ϵ that is determined by C_n. Then by the Switching Lemma, we can show that with some constant

[3] We abuse the notation here; NC^0 should be defined as a complexity class, that is, the class of *Boolean functions* computable by circuits described above.

probability each subcircuit of $C_n|_\rho$ computing an output gate of C_n becomes a very simple circuit that depends on only some constant number of input gates. Thus, $C_n|_\rho$ is computable by a fan-in two circuit with some fixed constant depth, that is, $C_n|_\rho$ can be converted to some NC^0 circuit. But obviously $C_n|_\rho$ cannot be used as a reduction because of the random restriction. In [4], they took a clever method. Adding some redundancy to A, we define a new set $\widehat{A}$. Note that we can still assume an $\leq_{\mathrm{m}}^{\mathrm{AC}^0}$-reduction from $\widehat{A}$ to K so long as $\widehat{A}$ is in NP. Then consider the random restriction to this reduction circuit. Again the circuit simplified by the random restriction cannot be used as a reduction from $\widehat{A}$ to K; nevertheless, thanks to the redundancy, it can be used as a reduction from A to K.

Let us see their idea more technically. Here we simplify their argments by omitting many technical details. For the set $\widehat{A}$ we define the following set.

$$\widehat{A} \;=\; \{\; u_1u_2\cdots u_n \,|\, u_i \in \{0,1\}^{n^a},\, \text{and}\, \mathrm{par}(u_1)\mathrm{par}(u_2)\cdots\mathrm{par}(u_n) \in A' \;\},$$

where $\mathrm{par}(u)$ is the parity of all bits of u. The parameter a should not be a constant because it needs to depend on the ϵ for defining the random restriction; neverthless, let us assume here that a is some constant. Then clearly $\widehat{A}$ is also in NP, and hence there is a $\leq_{\mathrm{m}}^{\mathrm{AC}^0}$-reduction $\{H_m\}_{m\geq 1}$ from $\widehat{A}$ to K. For any sufficiently large n, let $m = n^{a+1}$ be the length of strings of the form $u_1\cdots u_n$ such that $|u_i| = n^a$ for each i, $1 \leq i \leq n$. Consider the circuit H_m that reduces $\widehat{A}$ to K for length m inputs. H_m has m input gates corresponding to input instances $u_1u_2\cdots u_n$ for $\widehat{A}$; let $U_1,\ldots,U_n$ denote the *blocks* of input gates such that each U_i is a set of input gates for the part u_i of input instances. Now apply a random restriction ρ to H_m with a parameter $p = n^{-1+\epsilon}$. Then as discussed above by the Switching Lemma we can show that (i) $H_m|_\rho$ becomes an NC^0 circuit with high probability. On the other hand, we can also show that (ii) each block with n^a input gates has at least two unset input gates with high probability (by choosing a sufficiently large w.r.t. $1/\epsilon$). Thus, there exists some restriction for which both (i) and (ii) holds. Let ρ_0 denote one of such restrictions. We define further restrictions for each block U_i of the input gates of H_m. Consider any block U_i. From (ii), ρ_0 leaves at least 2 input gates unassigned. Keep the first two such input gates and assign 0 to the others. Furthermore, assign one of the remaining two to either 0 or 1 so that the value of the last unset input gate becomes the same as the value of $\mathrm{par}(U_i)$ in $H_m|_{\rho_0}$ after the restriction. Let H'_m denote the NC^0 circuit obtained by applying ρ_0 and these additional restrictions at all U_i's. Then it is easy to see that H'_m is a NC^0 reduction from A to K.

3 Deterministic Restrictions and Formula Satisfiability Algorithms

We explain some examples of using restrictions for analyzing the running time of formula satisfiability algorithms.

A formula is a special type of a circuit such that the fan-out of each gate is at most 1. We further assume in this section that the fan-in is at most two.

Then it would be convenient for our explanation to regard a formula as a rooted binary tree, where leaf nodes are labeled with a literal (i.e., a Boolean variable or its negation) and the other internal nodes are labeled with a logical operator. We may assume that without loss of generality a logical operator is either AND, OR, or XOR. A formula is called a *De Morgan formula* if it uses no XOR label; otherwise, it is called *general formula*. For a formula F, let $L(F)$ denote the number of leaves of F, which is regarded as the "size" of F in this section.

A formula satisfiability problem is, for a given formula, to decide whether there exists an assignment to the variables such that the formula evaluates 1. For a formula F in n variables, the problem can be solved in time $\mathrm{poly}(L(F))2^n$ by the exhaustive search. Here we show algorithms that run in time exponentially faster than 2^n if a given formula F is sparse, i.e., $L(F) = O(n)$. We explain how to give nontrivial upper bounds by using the analysis of local restrictions.

3.1 Background: A satisfiability algorithm for De Morgan formulas

Subbotovskaya [24] analyzed the effect of random restrictions on the size of De Morgan formulas and obtained a bound of the form

$$\mathrm{E}_\rho[\, L(F|_\rho)\,] \;=\; O(p^{3/2}L(F)). \tag{1}$$

Santhanam [22] refined Subbotovskaya's analysis and gave a satisfiability algorithm for linear-size De Morgan formulas. More specifically, for any formula F with $L(F) = cn$, the running time of this algorithm is of the form $\mathrm{poly}(L(F))2^{(1-1/\mathrm{poly}(c))n}$.

In what follows, we often consider restrictions that set the values of a small number of variables. It is convenient to introduce the following notation for such restrictions: For a formula F, a variable x_i, and $a \in \{0,1\}$, $F|_{x_i=a}$ denotes a formula obtained from F by assigning a to the variable x_i; the notation is extended naturally for the case where multiple variables are assigned.

Santhanam's algorithm is a simple branching algorithm based on greedy variable selection. For a given formula F, it works as follows.

(a) If $L(F) \le n/2$, then check the satisfiability of F by the exhaustive search.
(b) Otherwise, select a variable, say x_i, that appears (as x_i or $\neg x_i$) most frequently in F and check the satisfiability of $F|_{x_i=0}$ and $F|_{x_i=1}$ recursively.

To analyze its running time, let us consider a branching tree associated with the execution of the algorithm where we assume that only the case (b) occurs (until some depth). In this tree, a node is labeled with a formula examined by the algorithm; in particular, a node at depth i is labeled with a formula in $n-i$ variables. For the size of each of $2^{n-n'}$ formulas at depth $n-n'$ in the tree, Santhanam gave the following analysis.

Lemma 1 *There exist universal constants $\alpha, \beta > 0$ such that for any formula F in n variables, at least a $1 - 2^{-n'}$ fraction of the formulas at depth $n - n'$ in the branching tree for F have size at most $n'/2$, where $n' = \alpha n/c^{\beta}$.*

This implies that the running time is at most

$$\mathrm{poly}(L(f)) \left\{ 2^{n-n'}(1-2^{-n'})2^{n'/2} + 2^{n-n'}2^{-n'}2^{n'} \right\} \ < \ \mathrm{poly}(L(F))2^{(1-1/\mathrm{poly}(c))n}.$$

The intuition behind Lemma 1 is as follows. At the case (b) in Santhanam's algorithm, we have, for some $a \in \{0,1\}$,

$$L(F|_{x_i=a}) \ \leq \ \left(1 - \frac{1}{n}\right)^{3/2} L(F), \tag{2}$$

$$L(F|_{x_i=\neg a}) \ \leq \ \left(1 - \frac{1}{n}\right) L(F) \tag{3}$$

through the argument for establishing (1). Say $F|_{x_i=a}$ and $F|_{x_i=\neg a}$ a *good* branch and a *bad* branch respectively. Then the standard concentration inequality tells us that a "typical path" from the root to a node at depth $n - n'$ in the tree goes through a good branch and a bad branch almost alternatingly. This in turn implies a "concentrated version" of (1), that is,

$$\Pr_{\rho}\left[\, L(F|_{\rho}) \leq (n'/n)^{1+\varepsilon} L(F) \,\right] \ \geq \ 1 - 2^{-n'}, \tag{4}$$

where $\varepsilon > 0$ is a positive constant independent of F and a random restriction ρ is sampled by selecting a node at depth $n - n'$ in the tree uniformly at random and regarding the path from the root to the node as a restriction.

3.2 A satisfiability algorithm for general formulas

In a general formula, arbitrary binary logical operators are allowed. In particular, the existence of XOR makes general formulas much more powerful than De Morgan formulas. It also prevents us from obtaining a satisfiability algorithm for general formulas based on Santhanam's algorithm in the straightforward manner.

For example, let F be a formula that consists of only XOR nodes and let x_i be the most frequent variable in the formula. Then it can be the case that $L(F|_{x_i=a}) = \left(1 - \frac{1}{n}\right) L(F)$ holds for all $a \in \{0,1\}$. This implies that after $n - n'$ times of branchings, we always obtain a formula F' with $L(F') = (n'/n)L(F)$ and that $L(F') \leq n'/2$ cannot be achieved regardless of the choice of n'. Of course, we have a fix for this F; namely, F just computes the parity function or its negation in some variables and is obviously satisfiable. However, the situation can be more complicated.

Nevertheless, Seto and Tamaki [23] were able to give a satisfiability algorithm for general formulas such that the running time is exponentially faster than 2^n for linear-size formulas. More specifically, for any formula F with

$L(F) = cn$, the running time of their algorithm is of the form $\mathrm{poly}(L(F))2^{(1-\mu(c))n}$, where $\mu(c) = 1/2^{O(c^3)}$. Let us briefly describe the idea of [23].

We introduce the notion of *parity node* that plays a key role in Seto-Tamaki's algorithm. It is defined inductively as follows.

- A leaf node is parity.
- A node is parity if both of its children are parity and it is labeled with XOR.

We say that a node is *maximal parity* if it is parity and its parent is not parity.

Intuitively, nice things happen both when the number t of maximal parity nodes is small and when t is large. In the former case, once we fix the output values of t maximal parity nodes, then the output of the formula is also fixed. In addition, we can determine whether there exists an assignment to the variables such that the outputs of maximal parity nodes are fixed in a specified way by solving a system of linear equations over GF(2) using the Gaussian Elimination. To summarize, we can check the satisfiability in time $\mathrm{poly}(L(F))2^t$ in this case.

In the latter case, we can find a constant number of variables, say, $x_1, x_2, \ldots, x_k$, such that for at least 2^{k-1} assignments $a = a_1 \cdots a_k \in \{0,1\}^k$, we have

$$L(F|_{x_1=a_1,\ldots,x_k=a_k}) \;\leq\; \left(1 - \frac{k}{n}\right) L(F) - 1 \tag{5}$$

and for every assignment $a = a_1 \cdots a_k \in \{0,1\}^k$, we have

$$L(F|_{x_1=a_1,\ldots,x_k=a_k}) \;\leq\; \left(1 - \frac{k}{n}\right) L(F). \tag{6}$$

In this case, we check the satisfiability of $F|_{x_1=a_1,\ldots,x_k=a_k}$ for all $a = a_1 \cdots a_k \in \{0,1\}^k$ recursively.

The "structural lemma" of [23] states as follows: For every formula F in n variables with $L(F) = cn$, if it has at least $3n/4$ maximal parity nodes, then the latter case must occur with $k \leq 8c$. Its proof is too technical to fit in this survey; see the original paper for details. Intuitively, the structural lemma says that if the number of maximal parity nodes is large, then a general sparse formula behaves like a De Morgan formula with respect to a local restriction, in the sense that (5) and (6) can be used as weaker counterparts of (2) and (3) for the analysis of the running time of the recursive execution.

Combining the above two cases yields Seto-Tamaki's algorithm. Its running time analysis is more involved but basically similar to Santhanam's one, i.e., reduces to some concentration inequality of the form (4), but this time ε is not a universal constant but a function of c.

Acknowledgements

On this occassion we would like to express deep appreciation to Alan Selman for his continuous effort for running Mathematical Systems Theory and Theory of Computing Systems as the editor-in-chief.

References

1. M. Agrawal, The Isomorphism Conjecture for NP, in *Computability in Context: Computation and Logic in Real World*, World Scientific, 19–48, 2011.
2. M. Agrawal, The isomorphism conjecture for constant depth reductions, *Journal of Computer System and Sciences*, 77:3–13, 2011.
3. M. Agrawal and E. Allender, An isomorphism theorem for circuit complexity, In Proc. 11th Conference on Computational Complexity, IEEE, 2–11, 1996.
4. M. Agrawal, E. Allender, and S. Rudich, Reductions in circuit complexity: An isomorphism theorem and a gap theorem, *Journal of Computer System and Sciences*, 57:127–143, 1998.
5. M. Agrawal, E. Allender, R. Impagliazzio, T. Pitassi, and S. Rudich, Reducing the complexity of reductions, Computational Complexity, 10:2, 117–138, 2001.
6. M. Ajtai, Σ_1^1-formulae on finite structures, *Ann. Pure Appl. Logic*, 24(1):1–48, 1983.
7. Paul Beame, A switching lemma primer, Department of Computer Science and Engineering, University of Washington, UW-CSE–95-07-01, 1994.
8. L. Berman, Polynomial Reducibilities and Complete Sets, PhD thesis, Cornell University, 1977.
9. L. Berman and J. Hartmanis, On isomorphism and density of NP and other complete sets, *SIAM Journal on Computing*, 1:305–322, 1977.
10. R. Boppana and J. Lagarias, One-way functions and circuit complexity, in *Proceedings of Structure in Complexity Theory*, Lecture Notes in Computer Science 223:51–65, 1986.
11. R. O'Donnell, *Analysis of Boolean Functions*, Cambridge University Press, 2014.
12. M. Furst, J. Saxe, and M. Sipser, Parity, circuits, and the polynomial-time hierarchy, *Mathematical Systems Theory*, 17(1):13–27, 1984.
13. J. Håstad, *Computational Limitations for Small Depth Circuits*, MIT Press, Cambridge, 1986.
14. J. Håstad, Almost optimal lower bounds for small depth circuits, in *Proc. of the 18th Annual ACM Symposium on Theory of Computing* 6–20, 1986.
15. J. Kahn, G. Kalai, and N. Linial, The influence of variables on Boolean functions, in *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, 68–80, 1988.
16. S. Kurtz, S. Mahaney, and J. Royer, The structure of complete degrees, in *Complexity Theory Retrospective* (A. Selman, ed.), Springer-Verlag, 108–146, 1990.
17. N. Linial, Y. Mansour, and N. Nisan, Constant depth circuits, Fourier transform, and learnability, *Journal of the Association for Computing Machinery*, 40(3):607–620, 1993.
18. O. Lupanov, Implementing the algebra of logic functions in terms of constant-depth formulas in the basis $+, *, -$, *Soviet Physics Doklady*, 6(2):107–108, 1961.
19. B. Rossman, The average sensitivity of bounded-depth formulas, in *Proc. the 56th Annual IEEE Symposium on Foundations of Computer Science*, IEEE, 424–430, 2015.
20. B. Rossman, R. Servedio, and L. Tan, An average-case depth hierarchy theorem for Boolean circuits, in *Proc. the 56th Annual IEEE Symposium on Foundations of Computer Science*, IEEE, 1030–1048, 2015.
21. B. Rossman, R. Servedio, and L. Tan, Complexity Theory Column 89: The polynomial hierarchy, random oracles, and Boolean circuits, *SIGACT News*, 46(4): 50–68, 2015.
22. R. Santhanam, Fighting perebor: New and improved algorithms for formula and QBF satisfiability, in *Proc. of the 51th Annual IEEE Symposium on Foundations of Computer Science*, IEEE, 183–192, 2010.
23. K. Seto and S. Tamaki, A satisfiability algorithm and average-case hardness for formulas over the full binary basis, *Computational Complexity*, 22(2):245–274, 2013.
24. B. Subbotovskaya, Realizations of linear functions by formulas using $+, *, -$, *Soviet Mathematics Doklady*, 2:110–112, 1961.
25. O. Watanabe, On one-one polynomial time equivalence relations, *Theoretical Computer Science*, 38:157–165, 1985.