Succinct representations of Boolean functions and the Circuit-SAT problem

by

© *Shadab Romani*

A thesis submitted to the

School of Graduate Studies

in partial fulfilment of the

requirements for the degree of

Master of *Science*

Department of *Computer Science*

Memorial University of Newfoundland

April 2016

St. John's Newfoundland

Abstract

We study the question whether there is a computational advantage in deciding properties of Boolean functions given a succinct description of the function (such as a Boolean circuit) as opposed to black-box access to the function. We argue that a significant computational advantage for a large class of properties implies a non-trivial algorithm for the Circuit Satisfiability (Circuit-SAT) problem. In particular, we show that if there is a property with strong black-box lower bounds yet decidable in **BPP**, which also has a highly sensitive instance computable by a small circuit, then there is a non-uniform sub-exponential algorithm for the Circuit-SAT problem. Additionally, we analyze variants of this question for other computational models.

Acknowledgements

I am greatly indebted to my supervisor Antonina Kolokolova for giving me the opportunity to study complexity theory. I have learned a lot from her. Antonina is unbelievably kind, generous and patient with students. Remembering all my struggles at the beginning of grad school, without her help, I couldn't finish grad school!

Working on this thesis wouldn't have been possible without the guidance of Russell Impagliazzo, Valentine Kabanets, and Pierre Mckenzie. In particular, this thesis is based on a beautiful conjecture by Russell Impagliazzo. I was really privileged to collaborate with them.

And finally, I dedicate this thesis to my parents for their unconditional Love and support.

Contents

Chapter 1

Introduction

Solving a problem aside from the implications of the problem itself can reveal several facts about the solver. The complexity of a task shows the ability to perform tasks of lesser or equal complexity. The celebrated Rice's Theorem, proved by Henry Gordon Rice[Ric53], gives such a picture about Turing machines (TM) as problem solvers.

Rice's Theorem states that any non-trivial semantic property of TMs is undecidable. Semantic properties indicate something about the functionality of a TM and are independent of machine's syntactic description. Undecidability of semantic properties implies that the only useful thing that one can do with the description of a TM is simply running it.

Rice's Theorem elegantly reduces the *Halting Problem* to any non-trivial semantic property of TMs. Therefore, before deciding anything interesting about the functionality of programs, we must face the Halting Problem as if it is the simplest undecidable problem!

Rice's theorem cannot be straightforwardly extended to models other than TMs. For instance, a polynomial time TM is guaranteed to use a finite amount of time in its computation. Therefore, halting is not a problem for such machines and the idea of reduction from the Halting Problem is not valid for them. Understanding the functionality of these models becomes easier, in the sense that it is decidable. But, we face the next barrier: intractability!

An analog of Rice's Theorem can be particularly useful for generalizing and abstracting proofs of hardness in models of finite computation. Proofs of hardness for circuit analysis problems use a great variety of techniques from all areas of mathematics. For the TM model, Rice's Theorem eliminates the need for different proofs of undecidability by presenting a general clean template.

Boolean circuits are a powerful model of finite computation. If a property about the underlying Boolean functions is hard, then no efficient way is known to benefit from the circuit description. Essentially brute-force is the best algorithm we can design to decide such properties. Hard circuit problems show a similarity with undecidable problems in TM Model. In the sense that, for none of these problems the syntax of a program helps in understanding its semantics.

Additionally in the circuit model, the status of satisfiability among other hard problems resembles that of the Halting Problem among other undecidable problems. As if it is *the simplest hard problem*! Obviously, **NP**-completeness of the satisfiability problem means that if any other problem **NP**-complete has an efficient algorithm so does satisfiability. But by "simplest hard problem", we mean a more fine-grained perspective of the complexity of the satisfiability problem and its relationship with other **NP**-hard problems.

TMs are succinct representations of languages and Rice's Theorem rules out the possibility of any advantage provided by a succinct representation for deciding semantic properties of TMs. Circuits are succinct representations of Boolean functions. We study whether there is an advantage in looking into the circuit description compared to accessing it in a *black-box* manner for deciding semantic properties of circuits. Informally, black box is a type of access in which we know the size of the circuit, and we can ask input-output queries.

This thesis is based on the line of research initiated in the 90's ([BS96, BGI$^+$01b]) studying the existence of Rice's Theorem analogues for Boolean circuits, the first of which was formulated by Borchert and Stephan[BS96]. They focused on counting properties, that is, properties that only depend on the number of solutions, and have shown the following;

> *"Any non-empty counting property is* **UP***-hard with respect to polynomial time Turing reductions. A property is a counting property if it only depends on the number of solutions. "*

In the early 2000's, Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, and Yang [BGI$^+$01a] suggested a different analogue of Rice's Theorem for Boolean circuits. Motivated by cryptographic applications, they differentiated between algorithms analysing circuit descriptions and algorithms accessing circuits as a black-box. They made the following conjecture which they called "Scaled-down Rice's Theorem";

> *"Every property of Boolean functions F that can be computed in* **BPP** *given a circuit for function f, can be computed in comparable probabilistic polynomial time and with comparable success probability by a black-box algorithm with an oracle access to f and an upper bound on its circuit complexity. "*

If Scaled-down Rice's Theorem holds, then any property decidable in **BPP**, has at most polynomial randomized query complexity in black-box. Contrapositively, assuming a super-polynomial randomized query complexity in black-box, a **BPP** algorithm doesn't exist. Here, we will work with the following equivalent statement which we call "black-box hypothesis" :[1]

> *"**Black-box Hypothesis:** Let F be a property of Boolean functions such that even a randomized algorithm has to examine a large fraction of bits of the truth table of a function f to decide if $f \in F$. Then the complexity of deciding F is not significantly less when a circuit computing f is given as an input. "*

Barak et al. disproved a variant of this conjecture for promise-**BPP** under the assumption that one-way functions exist. However, the main statement remains open. The Scaled-down

[1]We are using the word "hypothesis" at Russell Impagliazzo's suggestion, since both proving and disproving it would have interesting consequences.

Rice's Theorem is more general than the approach based on counting properties and can explain the hardness of those properties as well.

Our Results

We study the implications of the black-box hypothesis for the hardness of Circuit-SAT. Our result supports the intuition of satisfiability being the simplest **NP**-hard problem.

> *"**Main Theorem:** We show that Circuit-SAT has a non-trivial algorithm if for some circuit analysis problem (satisfying a very general condition) there is a super-polynomial gap in complexity between the algorithms accessing the circuit description and the algorithms treating circuits as black-box. "*

Additionally, we show that in restricted models such as read-once branching programs, algorithm that accesses a description may gain a super-polynomial advantage over any algorithm that access the read-once branching programs as black-box.

More specifically, our main result is as follows. Let F be a property of Boolean functions. Let succinct-F be a language $L_F = \{\langle C \rangle \mid [C] \in F\}$, where $[C]$ is a function computed by a circuit C. If F is hard for black-box algorithms, but $L_F \in BPP$, then Circuit-SAT is solvable by a sub-exponential non-uniform family of circuits, provided that F either 1) contains only easy functions (i.e. functions computable by circuits of $\approx Size(2^{o(n)})$), or 2) F is sensitive on some easy inputs. In other words, there exists an easy Boolean function f such that there is a significant number boolean functions f's disagreeing with f on a single point such that $F(f) \neq F(f')$. In particular, non-empty counting properties satisfy the second condition, with an easy sensitive string of the form $tt(f) = 0\ldots01\ldots1$ such that adding or removing a 1 flips the value of $F(f)$. If that decision tree complexity is polynomially related to sensitivity, the second condition can be replaced by a lower bound on the decision tree complexity for F (on easy inputs), however it is not clear how to do that. In particular, it seems to involve giving a positive answer to a (generalization of) the well-studied Sensitivity Conjecture.

Organization

- Chapter 2 contains the necessary definitions and notational conventions for introducing the complexity theoretic analogues of Rice's Theorem.
- In Chapter 3 we overview the two previous approaches for translating Rice's Theorem to circuit complexity theory.
- In Chapter 4, we discuss several ways to formulate analogues of Rice's Theorem with respect to different computational models and complexity classes. We will also present the black-box hypothesis which can be viewed as an analogue of Rice's Theorem for general Boolean circuits and **BPP** time.
- Chapter 5 discusses the implications of the black-box hypothesis for hardness of Circuit-SAT.
- In the last chapter, we present our conclusions, intuitions, and future work directions.

Chapter 2

Preliminaries

In the world of circuits, we can make a distinction between circuit as syntax and the computed Boolean function as semantics. *White-box* access to a program informally means accessing the syntax of a program. The syntax of a circuit is a directed acyclic graph encoded as a binary string. On the other hand, *black-box* access informally means using only the semantics of a program by asking oracle queries. Our goal is correlating the hardness of circuit decision problems in black-box and white-box. To make a meaningful comparison between these two settings, we must define the circuit representation and the type of oracle access carefully. So first, we need several basic definitions from circuit complexity theory about encoding and representation in white-box and black-box.

2.1 Boolean Functions and Types of Access

An *n-ary Boolean function* is a function that maps the domain $\{0,1\}^n$ to the co-domain $\{0,1\}$. Notation: $tt(f)$ denotes the truth table of a Boolean function f.

Definition 2.1. A property of Boolean functions F is a subset of Boolean functions, where each slice $F_n \subseteq \{0,1\}^{2^n}$ is a subset of (truth tables of) Boolean functions of n variables. F is *non-trivial* if for every (large enough) n, both F_n and its complement $\overline{F_n}$ are non-empty.

Definition 2.2 (Boolean Circuit)**.** A Boolean circuit is a directed acyclic graph with m nodes $g_1, g_2, \ldots, g_m$ (in some fixed topological order) called gates, such that the first n nodes $g_1, g_2, \ldots, g_n$ take their value from the input $g_1 = x_1, g_2 = x_2, \ldots, g_n = x_n$. Each of the subsequent gates g_i computes a Boolean function from a set of Boolean functions $\mathcal{B}$ called basis gates which take their argument from the set of previously evaluated gates i.e.,

$$g_i = \phi(g_{i_1}, g_{i_2}, \ldots, g_{i_d<i}),$$

where ϕ is a function in $\mathcal{B}$ and d is the number of its arguments. The last gate g_m is distinguished as the output gate.

A circuit C computes a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ if for every input string $x_1 \ldots x_n \in \{0,1\}^n$, $f(x_1 \ldots x_n)$ is equal to the value of gate g_m in $C(x_1 \ldots x_n)$. $[C]$ denotes the Boolean function computed by circuit C. We equate a circuit (or an algorithm) outputting 1 with accepting and outputting 0 with rejecting. A circuit is *satisfiable* if there is at least one input on which it outputs 1. We refer to the language of all satisfiable circuits as **Circuit-SAT**.

$|C|$ refers to the size of the circuit. In general, there are several related ways to define circuit size: size of the string describing the circuit, number of edges, number of gates or the sum of the number of edges and gates. Note that all these measures are polynomially related. The number of edges is at most square of the number of gates, and description length is:

$$O((\#edges + \#gates)\log(\#edges + \#gates))$$

Here we take $|C|$ to be the length of the description of the circuit using the direct connection language of [Ruz81] denoted $desc(C)$. The size of a circuit on n inputs is at least $O(n \log n)$. The circuit size of a Boolean function f, denoted by $CircuitSize(f)$, is the size of the smallest circuit computing f.

In this thesis, we mainly discuss circuits defined over the following basis, which we refer to as general Boolean circuits: all the possible Boolean functions defined over two bits, the constant 0, the constant 1 and the NOT gate. Since we are working with general circuits, we assume constant fan-in without loss of generality.

Definition 2.3 (Semantic Property of Circuits)**.** A *semantic property* F of circuits is a property of Boolean functions that circuits compute. A language of circuits that satisfy a semantic property F is called a metalanguage i.e. a metalanguage L_F for F is:

$$L_F = \{desc(C) \mid [C] \in F\}$$

Based on the distinction between a circuit and the function it computes, two access types are defined:

Definition 2.4. An algorithm A decides property F in *white-box* if A decides the corresponding metalanguage L_F. That is, given as input a string $desc(C)$ it accepts iff $[C] \in F$.

An algorithm in black-box model accesses the circuit using an oracle and the knowledge of the size of the circuit description. An oracle O_L for a language L is an external device that is capable of reporting whether any string w is a member of L. An oracle Turing machine M^{O_L} is a modified TM that has the additional capability of querying an oracle. Whenever M^{O_L} writes a string on a special oracle tape, it is informed whether that string is a member of A in a single computational step. An oracle for a Boolean function f is the language $O_f : \{x \mid f(x) = 1\}$. To simplify the notation, we often write M^L, M^f to mean M^{O_L}, M^{O_f}.

Definition 2.5. An algorithm A decides F in *black-box* if $A^f(1^n, 1^m)$ accepts iff $f \in F$, where $f\colon \{0,1\}^n \to \{0,1\}$, m is an upper bound on circuit size of f and O_f is an oracle for the Boolean function f. Here, 1^n and 1^m represent n and m in unary.

We will be comparing the complexity of algorithms in white-box vs. black-box. As the inputs to the black-box algorithm are given in unary, the size of the input to black-box and white-box algorithms are essentially the same. Note that given $desc(C)$, it is easy to compute number of inputs n. In particular, we compare metalanguages in **BPP** with their black-box counterparts.

Definition 2.6. A language L is in **BPP** if there exists a randomized algorithm A running in probabilistic polynomial-time and such that for every $x \in \{0,1\}^n$,

$$x \in L \Rightarrow Pr[A(x) \text{ accepts }] \geq 2/3$$

$$x \notin L \Rightarrow Pr[A(x) \text{ rejects}] \geq 2/3$$

We say that a property F is decidable in **BPP** in the white-box setting iff $L_F \in$ **BPP**.

A property F is in black-box-**BPP** if there exists a randomized polynomial time algorithm A, such that for every $n, f\colon \{0,1\}^n \to \{0,1\}$ and $m \geq CircuitSize(f)$,

$$f \in F \Rightarrow Pr[A^f(1^n, 1^m) \text{ accepts}] \geq 2/3$$

$$f \notin F \Rightarrow Pr[A^f(1^n, 1^m) \text{ rejects}] \geq 2/3.$$

2.2 Boolean Function Lower-Bounds in Black-Box Model

A Boolean function is a *point function* if it outputs 1 on exactly one input. We refer to the Boolean function that outputs zero on every input as constant **Zero** function.

Proposition 2.7. *Checking satisfiability of n-ary Boolean functions needs 2^{n-1} oracle queries by a randomized black-box algorithm.*

Proof. This can be proved by a simple adversary argument. Alice thinks of a point function f and Bob asks for values of f on different inputs. In the worst case, Alice answers 0 to all the $2^n - 1$ first queries and 1 to the last query. To distinguish a point function from the constant zero function with probability $p > 1/2$, at least half of the inputs must be queried. Circuit complexity of point functions and all zero function is linear in n. Even though the black-box algorithm is given the upper-bound $cn \log n$ for constant c on the size of the circuit, it still needs to make exponentially many queries. □

For a thorough discussion on lower bounds using different types of oracle queries see [Ang88].

The query complexity of randomized and deterministic algorithms are polynomially related. So the black-box lower bound of Proposition 2.7 also implies a lower bound for deterministic query complexity. Decision trees are a standard model for proving oracle lower bounds. Presentation below follows [BDW02].

Definition 2.8 (Decision tree)**.** A *decision tree* is a rooted ordered binary tree T. Each internal node of T is labeled with a variable x_i and each leaf is labeled with a value 0 or 1. Given an input $x \in \{0,1\}^n$, the tree is evaluated as follows. Start at the root. If this is a leaf then stop. Otherwise, query the variable x_i that labels the root. If $x_i = 0$, then recursively evaluate the left sub-tree, if $x_i = 1$ then recursively evaluate the right sub-tree. The output of the tree is the value of the leaf that is reached eventually.

Definition 2.9 (Randomized decision tree)**.** A *randomized decision tree* is defined like a decision tree but each internal node is associated with a bias probability $p \in [0, 1]$. On input

$x_1x_2\dots x_n$, tree evaluation is similar to a regular decision tree, but when an internal node is reached, we follow the edge whose label agrees with x_i with probability p.

Definition 2.10. Deterministic query complexity $D(f)$ is the minimal depth of a decision tree computing f. Randomized query complexity $R_2(f)$ is the minimal depth of a randomized decision tree for f, where the subscript 2 indicates 2-sided error.

$R_2(f)$ and $D(f)$ are polynomially related.

Theorem 2.11. *[Nis91]* $D(f) < 27R_2(f)^3$

Therefore, randomized algorithms cannot do significantly better than deterministic algorithms in the black-box setting.

Another important algorithm is *Circuit Evaluation.* In black-box, we only need one oracle query. In white-box, there exists a linear time algorithm.

Proposition 2.12. *Let F be the property of outputting 1 on the all zero input ($00\dots0$). Then the complexity of white-box and black-box is $O(m)$, where n is the number of inputs and m is the size of the given circuit abd $m \geq n$.*

Proof. The white-box algorithm evaluates the given circuit on $00\dots0$. Circuit evaluation takes linear time with respect to the size of the circuit. The black-box algorithm also takes $O(m)$ (to write the query) time because one oracle query in enough. □

2.3 The Original Rice's Theorem: A Brief Look

Rice's Theorem [Ric53] gives a general test for determining the undecidability of certain properties of TMs. Here, we only outline the proof to make an analogy with the black-box hypothesis. For the sake of simplicity we use a different terminology from the original proof. To see a detailed proof [HU79] is a good source.

A property of languages is defined as a set of languages. *Semantic properties* of TMs are properties of languages they accept. A semantic property is non-trivial if there is a TM that satisfies the property and one that violates the property. Informally, Rice's Theorem shows that every property about the language recognized by a TM is either trivial or undecidable.

Theorem 2.13 (Rice's Theorem)**.** *Let $\mathcal{P}$ be any non-trivial semantic property of TMs. Then, the following language is undecidable;*

$$L_{\mathcal{P}} = \{\langle M \rangle : L(M) \in \mathcal{P}\}$$

The idea is a classic proof by contradiction using a clever reduction from the Halting Problem. Suppose that $\mathcal{P}$ is decidable. Since $\mathcal{P}$ is non-trivial there must be some machine M_L that has this property. Without loss of generality, assume that $\mathcal{P}$ does not include the empty language: otherwise, consider complement of $\mathcal{P}$. Given machine M and input x, we want to decide if M halts on x. We construct another machine R using M_L:

R description: On input w do the following steps:

1. Run M on x
2. Run M_L on input w

Now R satisfies $\mathcal{P}$ if and only if M halts on x. Because only in this case R behaves like M_L and if M doesn't halt then R only accepts the empty language which by assumption does not satisfy $\mathcal{P}$.

Chapter 3

Previous Work

The naive way to formulate an analogue of Rice theorem is by simply replacing the words "Turing machine" with "circuit" and the word "undecidable" with "intractable";

"Every nontrivial semantic property of Boolean circuits is intractable. "

This formulation is wrong. Some nontrivial semantic properties are easy, such as the property of outputting zero on the all zero input (see Proposition 2.12). Therefore, any analogue must present a more sophisticated characterization of hard semantic properties. There are two lines of work for constructing a complexity theoretic analogue of Rice's Theorem for the circuit model: one based on hardness of *Counting* problems and one based on the notion of obfuscation. In the next two sections, we overview these approaches.

3.1 Approach Based on Counting Problems

The quest for an analogue of Rice's Theorem for circuits began in the 90s by Borchert and Stephan[BS96] and was further developed by Hemaspaandra, Rothe and Thakur[HT01, HR00]. In this line of research, *Counting Properties* were the main focus to prove some analogue of Rice's Theorem. These properties only depend on the number of satisfying assignments of the circuit–not their particular distribution. Generally, three specific types of counting properties are defined:

Definition 3.1. Counting Properties: The notation $\#_1$ (and $\#_0$) denotes the total number of satisfying (respectively falsifying) assignments of a circuit.

- *Absolute Counting:* Let S be a subset of $\mathbb{N}$

  $Absolute - Counting(S) = \{C \mid \#_1(C) = n, n \in S\}$

- *Gap Counting:* Let S be a subset of $\mathbb{Z}$

  $Gap - Counting(S) = \{C \mid \#_1(C) - \#_0(C) = n, n \in S\}$

- *Relative Counting:* Let S be a subset of $\mathbb{D}$ (dyadic numbers):

  $RelativeCounting(S) = \{C \mid \frac{\#_1(C)}{\#_1(C)+\#_0(C)} = n, n \in S\}$

Borchert and Stephan proved the following hardness result for these problems:

Theorem 3.2. *[BS96] Any nontrivial absolute (gap, relative) counting property of circuits is* **UP***-hard with respect to Turing reductions, where* **UP** *is the class of decision problems solvable by an* **NP** *machine that for all inputs has at most one accepting path.*

Moreover, absolute and gap counting problems have polynomial many-one reductions to **UP**. Combined with **NP** and **coNP** being polynomial randomized reducible to **UP**, it follows that

Theorem 3.3. *[BS96] Satisfiability or its complement is randomized polynomial-time reducible to any nontrivial absolute and gap counting problem.*

Following this work, Hemaspaandra and Rothe improved the above result for Absolute Counting problem.

Theorem 3.4. *[HT01] Every non-trivial absolute counting property of circuits is* $\mathbf{UP}_{o(1)}$*-hard, where* $\mathbf{UP}_{o(1)}$ *is defined similar to* **UP** *but allows the* **NP** *machine to have at most* $o(1)$ *accepting paths.*

They argued that under plausible complexity-theoretic assumptions, this lower bound cannot be improved to **SPP** hardness. **SPP** is the class of languages recognized by **NP** machines such that for NO instances the number of accepting computation paths exactly equals the number of rejecting paths, and for YES instances these numbers differ by 2.

Theorem 3.4 was further improved by Hemaspaandra and Thakur [HR00] to **FewP**-hardness which is defined similar to $\mathbf{UP}_{o(1)}$, but the accepting TM can have polynomial ambiguity or non-determinism.

Counting problems are generally hard and intractable. However, it is not hard to find hard properties of Boolean functions which cannot be described as counting problems. In the next section, we introduce an analogue of Rice's Theorem which is more general than counting problems.

3.2 Approach Based on Obfuscation

In early 2000s, Barak et al. [BGI$^+$01b] in their seminal work on the impossibility of black-box obfuscation formulated a different analogue of Rice's Theorem for circuits which they named *"Scaled-down Rice's Theorem"*. Obfuscation is a fundamental cryptographic concept. Informally, obfuscation is a compiler that modifies the syntax of programs to make it unintelligible to an adversary. Barak et al proved that the strongest notion of obfuscation is impossible.

Definition 3.5. A *Virtual Black-box* (VBB) circuit obfuscator is a Probabilistic Polynomial Time (**PPT**) algorithm that given a circuit C outputs another circuit $\mathcal{O}(C)$ which satisfies the following conditions,

1. *Functionality*: C and $\mathcal{O}(C)$ implement exactly the same Boolean function.

2. *Polynomial slow-down*: Size of $\mathcal{O}(C)$ is at most polynomially larger than C, i.e. $|\mathcal{O}(C)| \leq poly(|C|)$

3. *Virtual black-box*: Anything that can be learned efficiently about $\mathcal{O}(C)$ can also be learned by using only black-box access to C. More formally, for any **PPT** algorithm A there is another **PPT** algorithm S and a negligible function α (growing slower than the inverse of any polynomial) such that for all circuits C:

$$|Pr[A(\mathcal{O}(C)) = 1] - Pr[S^{[C]}(1^n, 1^{|C|}) = 1]| \leq \alpha(|C|)$$

A virtual black-box TM obfuscator is also defined similarly just by replacing circuits with TMs. Barak et al. proved that TM virtual black-box obfuscation is impossible unconditionally. Assuming that one-way function exists, virtual black-box circuit obfuscation is also impossible. In particular, because of the virtual black-box condition. The main idea of the proof is constructing a family of un-obfuscatable Boolean functions $\mathcal{F}$. There is a property π for family F that given any circuit computing the function $f \in \mathcal{F}$, this property can be efficiently learned. On the other hand, using only black-box access it is impossible to learn π.

Motivated by the impossibility of VBB, Barak et al. proposed a weaker notion of obfuscation called *Indistinguishibility Obfuscation* (IO) which is still very useful (see [SW14]). IO requires that for every two functionally equivalent circuits of equal size, their obfuscation be indistinguishable. In 2013, Garg et al[GGH$^+$13] proposed the first construction of efficient IO.

VBB is a strong notion. To satisfy the conditions of VBB, for every property and every function, it should be possible to find reasonably small circuits computing that function from which it is hard to decide the property. The impossibility of VBB obfuscation rules out this possibility. However, it may still be possible to have circuits that are somewhat unintelligible. But, are there any circuits of reasonably small size from which it is hard to decide a property? This observation led Barak et al. to suggest the following conjecture:

Conjecture 3.6 (Scaled-down Rice's Theorem)**.** Let $L \subset \{0,1\}^*$ be a language such that for circuits C and C', $[C] = [C']$ implies that $C \in L \Leftrightarrow C' \in L$. If $L \in \mathbf{BPP}$ then L is trivial in the sense that there exist a **PPT** algorithm S such that,

$$C \in L \to Pr[S^{[C]}(1^n, 1^{|C|}) = 1] \geq 2/3$$

$$C \notin L \to Pr[S^{[C]}(1^n, 1^{|C|}) = 0] \geq 2/3$$

Barak et al. left this conjecture as an open problem. However, they considered another variant by generalizing the statement to promise problems. Promise problems are decision problems where the input is promised to be from a specific subset. More formally, a decision problem Π consists of a pair (Π_Y, Π_N), corresponding to Yes and No instances respectively. They showed that the unobfuscatable circuit family is a counter example for the promise version of the conjecture.

Scaled-down Rice's Theorem has been the main motivation of this thesis. We will present an equivalent formulation of it in the next chapter.

Chapter 4

Black-Box Hypothesis and Its Variants

Intuitively, to decide properties about the input-output behaviour of circuits or programs, it seems easier to just run the program on some inputs instead of analyzing the syntax. Based on Conjecture 3.6 from [BGI$^+$01b], we formulate the black-box hypothesis to capture this intuition:

Conjecture 4.1 (Black-box hypothesis)**.** Let F be a property of Boolean functions. If $F \in \mathbf{BPP}$ in the white-box setting over circuits, then $F \in$ black-box $\mathbf{BPP}$.

Informally, this hypothesis conveys that the complexity of deciding a semantic property of circuits in white-box and black-box is comparable and white-box access does not really provide a significant computational advantage.

It is an interesting question how the black-box hypothesis is related to $\mathbf{P}$ vs. $\mathbf{NP}$ problem. Could it be the same problem only expressed in a different language and disguise? If $\mathbf{P} = \mathbf{NP}$, then satisfiability (as explained in Proposition 2.7) has a exponential lower bound in black-box and a P-time (also **BPP**) algorithm in white-box. Therefore, Circuit-SAT would be a counterexample to the black-box hypothesis. But if $\mathbf{P} \neq \mathbf{NP}$, would it imply that the conjecture holds? This is not an easy question. To answer it, the exact relationship between hardness in white-box and black-box must be explained.

4.1 Black-box Hypothesis in Other Models

The concept of white-box and black-box can be extended to other models of computation. Based on the distinction between access types, an analogue of the black-box hypothesis can be formulated. An analogue would generally follow this template:

> *"Let $\mathcal{M}$ be a model of computation and $\mathcal{C}$ a time complexity class. If a semantic property of $\mathcal{M}$ is decidable in $\mathcal{C}$, it can also be decided in $\mathcal{C}$ using only black-box access. "*

The black-box hypothesis says that for general Boolean Circuits, if Φ is semantic property decidable in class **BPP** then deciding Φ in white-box is as hard as black-box. The above template can also be modified for other models and time complexities. For instance, we can only focus on some computationally limited subclass of general Boolean circuits or deterministic polynomial time. Even by assuming TMs as the model $\mathcal{M}$ and the class of undecidable languages as complexity class $\mathcal{C}$, we can view the original Rice's Theorem as some variant of the black-box hypothesis.

4.1.1 Read-once Models

In this section we consider an analogue of the black-box hypothesis for models with read-onceness property. A very important and useful class of models with this property is *Read-Once Branching Program.*

Definition 4.2 (Read-Once Branching Program)**.** A branching program on the variable set $X = \{x_1, x_2, \ldots x_n\}$ is a finite directed acyclic graph with one source node and sink nodes partitioned into two sets, Accept and Reject. Each non-sink node is labeled by a variable x_i and has two outgoing edges labeled 0 and 1 respectively. In a read-once branching program (roBP), on each path from source to sink each variable occurs at most once.

Claim 4.3. *Black-box hypothesis for roBPs is false, with satisfiability and equivalence as counterexamples.*

Proof. Testing equivalence of read-once branching programs is known to be in **BPP** [BCW80]. Given two roBPs, using Schwartz-Zippel lemma [Zip79, Sch80] we convert them to equivalent polynomials and then evaluate them at some random point, the two are equal with high probability if they agree on a random point.

Testing equivalence of programs is generally a hard problem, for TMs it is undecidable and for circuits and general branching programs (without read-onceness) it is **coNP**-complete. Satisfiability is also easy for this model of computation. Given the description of the graph, to decide satisfiability we check if there exist a path from the source node to the sink 1.

Now consider how we test both satisfiability and equivalence in black-box. To check equivalence, all the bits of the truth tables must be queried in the worst case. Because of lower-bounds for point functions (see Proposition 2.7), exponentially many oracle queries are necessary to find a satisfiable assignment. Therefore, equivalence checking and satisfiability are two counterexamples for the black-box hypothesis in roBP model. □

Note. Equivalence can be expressed as a semantic property by considering a specific family of functions. For instance, the property of equivalence to all zero functions.

Another weaker subclass of read-once models is *read-once CNF*. That is a CNF formula in which each variable occurs at most once. Checking satisfiability in white-box is really easy for read-once CNFs. These CNFs cannot have any conflicting occurrences of variables because each variable occurs at most once. Given a description of the formula, it is easy to find a satisfying assignment. So the white-box algorithm just needs to check if the CNF is non-empty. On the other hand, black-box satisfiability requires at least exponentially many queries in the worst case (because of point function lower-bounds Proposition 2.7). Generally, for models of computation capable of computing point functions, if checking satisfiability is easy, then the black-box hypothesis (for **BPP**) does not hold.

However, if we restrict the computational model to some very weak class that is exactly learnable by oracle queries, then black-box hypothesis holds trivially. Learning with membership queries is a learning model in which a learner requests examples. A circuit class $\mathcal{C}$ is exactly learnable in polynomial time (with membership queries), if after submitting polynomially many oracle queries to the circuit $C \in \mathcal{C}$ the learner outputs a circuit C' such that $[C] = [C']$. The learning is *proper* if $C' \in \mathcal{C}$.

Observation 4.4. *The black-box hypothesis is true for a class $\mathcal{C}$ that is properly exactly learnable in* **P***-time.*

The black-box algorithm first learns the function by submitting polynomially many queries and gains white-box access to the $\mathcal{C}$-representation of a function. After learning, the black-box algorithm can compute any property by simply applying the white-box algorithm to what it has learned.

4.2 Black-Box hypothesis in Other Complexity Classes

In the previous section, we considered Rice's Theorem analogues for models other than general Boolean circuits. In this section, we discuss a different variant by changing the complexity class from **BPP** to deterministic sub-exponential time. We analyze if every property of general Boolean circuits that takes sub-exponential time in white-box has a sub-exponential time black-box algorithm. More precisely:

Conjecture 4.5 (Black-box hypothesis for sub-exponential time)**.** Let F be a property of Boolean functions and L_F a corresponding metalanguage over circuits. If $L_F \in Time(2^{o(n)})$ in the white-box setting, then $F \in Time(2^{o(n)})$ black-box.

We don't intend to prove or disprove Conjecture 4.5, but we analyze its connection to the black-box hypothesis for **BPP** and prove that a deterministic sub-exponential time counterexample can imply a polynomial time counterexample. Since $\mathbf{P} \subseteq \mathbf{BPP}$, a polynomial time counterexample also violates the original black-box hypothesis.

Padding is a technique for proving equality or inequality of complexity classes by transforming classes to bigger classes. The transformation works by adding a certain amount of *dummy* symbols to every language in a class and showing that equality scales up or enequality scales down. We define a padding scheme for circuits–specifically designed for our conjecture.

Definition 4.6 (Padded Language)**.** Let L be a metalanguage. L_{pad} is defined as the language of all circuits with $N = pad(n) > n$ many input that satisfy two conditions:

- $pad(n)$ and its inverse $pad^{-1}(n)$ are computable in **P**.
- If the last $pad(n) - n$ many variables are replaced with 1 the resulting circuit is in L.

L_{pad} is the language of circuits with $pad(n)$ input whose restriction to the first n inputs is in L. This method of padding by adding dummy inputs makes L_{pad} a different metalanguage. Note that similar to a classic padding argument, if the dummy symbols were added to the circuit description, then the underlying Boolean function stays the same. The black-box lower bound and white-box upper bound of L_{pad} can be computed with respect to lower/upper bounds of L. Let C be a circuit with n variables and size m. Assume that,

- L is decidable in $T_w(n)p(m)$ time in the white-box, where $p(m) \in poly(m)$.
- L takes at least $T_b(n)p'(m)$ time in the black-box, where $p'(m) \in poly(m)$.

Lemma 4.7 (White-box upper-bound of padded language)**.** *L_{pad} is decidable in $T_w(n)q(M)$ for circuit of size M and $pad(n)$ many inputs, where $q(M) \in poly(M)$.*

Proof. First we have to eliminate the dummy variables and then decide if the resulting circuit is in L. Let C_{pad} be the input circuit with N inputs and of size m. So the algorithm takes 3 steps:

1. Compute $n = pad^{-1}(N)$ and plug in $11\dots1$ in circuit for the last $N-n$ variables.
2. Simplify the circuit to get a new one with only n variables.
3. Apply the algorithm that decides L to the simplified circuit.

Step 1 takes polynomial time $T_1(N)$ because $pad(n)$, $pad^{-1}(N)$ are computable in **P** by definition. Step 2 takes polynomial time $T_2(M)$ because circuit simplification by injecting values is also in P. Simplification is a very simple algorithm similar to circuit evaluation but only partial. The simplification algorithm greedily progresses by evaluating all the gates that based on their available input, the output can be computed. And leaving the others un-evaluated. Finally, step 3 takes at most $T_w(n)p(M)$. All these steps sums up to: $T_1(N) + T_2(N) + T_w(n)p(M) = T_w(n)q(M)$ for some $q(n) \geq T_1(n), T_2(n), p(n)$ □

Lemma 4.8 (Black-box lower-bound of padded language)**.** *Deciding L_{pad} on inputs of length $N = pad(n)$ in black-box setting requires at least as many queries as deciding L on inputs of length n. If L_{pad} is decidable in time $T(N, M)$, then L can be decided in time $T(pad(n), (m - n + pad(n)) \log(m + pad(n)) / \log(m)) + O((t_{pad}(n) + m) * \log(m - n + pad(n)) / \log(m)$, where $t_{pad}(n)$ is the time complexity of computing $pad(n)$.*

Proof. Suppose there is an algorithm $A(N, M)$ which decides L_{pad} using $Q(N, M)$ queries and time $T(N, M)$. Consider the following algorithm $A'(n, m)$ deciding L:

Let C of size M on N inputs. Compute $N = pad(n)$ and create an instance C' of L_{pad} by adding $pad(n)$ dummy input gates to C. The circuit C' has $N = pad(n)$ and $M = SizePad(n, m) = (m - n + pad(n)) \log(m - n + pad(n)) / \log(m)$. Now, run $A(C')$; if $A(C')$ accepts, accept C.

Now, if $A(N, M)$ makes $Q(N, M)$ queries, then so does the algorithm for L described above. Thus, if L_{pad} is decidable with $Q(N, M)$ queries, then the algorithm above decides L using $Q'(n, m) = Q(pad(n), SizePad(n, m))$ queries.

The time complexity of the algorithm consists of complexity of creating C' plus the complexity of running $A(C')$. To compute the former, note that there are $pad(n) - n$ new gates and all gate names are now $log(m - n + pad(n))$-bit strings rather than $\log(m)$-bit strings, and also that the new circuit can be constructed in time linear to the output size. The time complexity of running $A(C')$ is $A(N, M)$ for N, M as above.

Therefore, if there is no algorithm deciding L in time at most $T_b(n, m)$ with $Q(n, m)$ queries, then there is no algorithm for L_{pad} with time complexity less than

$T_b(pad^{-1}(N), SizePad^{-1}(M,N))$ (where inverse of SizePad returns m).

□

Corollary 4.9. *Suppose there is a language L with $T_b(n) = 2^{\Omega(n)}$ and $T_w(n) \in 2^{o(n)}$, where T_w and its inverse are polynomially computable. Then there is another language L_{pad} that takes polynomial time in the white-box and super-polynomial time in the black-box.*

Proof of Corollary 4.9. Take the amount of dummy inputs in padded language to be T_w. Using Lemma 4.8 and Lemma 4.7 white-box complexity is $T_w(n)q(M)$ and black-box is $T_b(n)q'(M)$.

Because the circuit has $T_w(n)$ inputs, white-box complexity $T_w(n)q(M)$ is a polynomial with respect to the circuit size. But for the black-box complexity $T_b(n)q'(M)$ we have: $T_w(n) = \omega(T_b(n))$. Because, $T_b(n)$ as an exponential function cannot be bounded by any polynomial of a sub-exponential function $T_w(n)$. Therefore, we can say that $T_b(n)$ is at least super-polynomial with respect to number of variables $n + T_w(n)$. □

In conclusion, the existence of a metalanguage with the following conditions violates black-box hypothesis,

1. Exponential black-box lower-bound.
2. Sub-exponential white-box upper bound.
3. Upper bound of the white-box algorithm and its inverse be computable in **P**.

This variant for exponential time mainly helps in understanding the complexity of languages like satisfiability which have exponential lower-bounds in black-box. Even a slight improvement in the white-box algorithm will have consequences such as violating the black-box hypothesis or the Exponential Time Hypothesis.

Chapter 5

Connections with Circuit-SAT

Rice's Theorem shows that any counterexample to its statement implies an algorithm deciding the Halting Problem. Following the approach of Rice's Theorem, we study the implications of the black-box hypothesis counterexamples for the hardness of Circuit-SAT–which may be a good candidate for the title of the easiest hard problem. To investigate the possibility of this intuition, we construct a non-trivial algorithm for Circuit-SAT assuming a counterexample to Conjecture 4.1.

For several problems with exponential complexity in black-box the existence of a **BPP** algorithm would immediately give a **BPP** algorithm for Circuit-SAT. However, no general way to prove such connection is known.

Consider Parity-SAT: the language of all the circuits that the number of their satisfying assignments is odd. Circuit-SAT is reducible to Parity-SAT by the following well-known reduction.

Theorem 5.1 (Valiant-Vazirani Reduction [VV86])**.** *There exists a* **PPT** *algorithm that on input C, where C is a circuit of n variables, outputs a list of circuits $C_1, C_2, \ldots, C_n$ such* *that*

- *If C is unsatisfiable then all C_i are unsatisfiable.*
- *If C is satisfiable then with probability at least $1/2$ some C_i is uniquely satisfiable.*

Example 5.2 (Algorithm for Circuit-SAT Using Parity-SAT [VV86])**.** Let A_P be the algorithm for Parity-SAT and let C be the circuit for which we want to decide satisfiability.

Apply the Valiant-Vazirani reduction to get a list $C_1, C_2, \ldots, C_n$. If $A_P(C_i) = 1$ for some i, then $C \in \mathit{Circuit\text{-}SAT}$, and $C \notin \mathit{Circuit\text{-}SAT}$ otherwise.

As another example, consider the language that for every n, circuits on n bits in the language compute a fixed Boolean function on n bits for which there exists a polynomial size circuit. Circuit-SAT is reducible to this language.

Example 5.3 (Algorithm for Circuit-SAT using fixed easy function)**.** Let f_{fixed} be a fixed Boolean function such that $size(f_{fixed}) \in poly(n)$. Let A_{fixed} be a **BPP** algorithm deciding $\{C \mid [C] = f_{fixed}\}$. The **BPP** algorithm with advice for Circuit-SAT works as follows- assuming circuit C as input and circuit $C_{f_{fixed}}$ as advice.

- $C' = C \oplus C_{f_{fixed}}$
- if $A_{fixed}(C') = 1$ then output 1 and 0 otherwise.

Because f_{fixed} is a fixed Boolean function changing any bits of the truth table of $C_{f_{fixed}}$ results in A_{fixed} not accepting it. So XORing the circuit $C_{f_{fixed}}$ with a circuit that has at least one satisfying assignment gives a new circuit that is not in the language.

5.1 Properties of Easy Functions

The metalanguage defined in Example 5.3 consists of only one single easy Boolean function for every n. The idea of this example can be extended to metalanguages that contain only easy functions; an example of such is *Succinct Minimum Circuit Size Problem* (Succinct MCSP). There are two ways to define Succinct MCSP, one with size given as a parameter, and one with fixed size; we explain both cases.

Definition 5.4 (Succinct MCSP)**.** Given a circuit C on n inputs and a number $t \in [0, \dots, 2^n/n]$ given in unary, decide whether there exists a circuit for $[C]$ of size less than t (Size is the number of gates).

Circuit-SAT can be trivially decided by calling $\mathit{SuccinctMCSP}(C,t)$ with $t = n+1$. Circuits of such complexity can only compute constants $\{0,1\}$ and input variables; the only of these cases when $C(\bar{1}) = 0$ is the constant 0 case, corresponding to unsatisfiable circuits. Thus, if $\mathit{SuccinctMCSP}(C,t) \in BPP$, then $\mathit{Circuit\text{-}SAT} \in BPP$.

The converse is also true. If Circuit-SAT $\in$ **BPP** (or even $\mathit{CNF\text{-}SAT} \in \mathbf{BPP}$), then polynomial time hierarchy collapses to **BPP**. $\mathit{SuccinctMCSP}(C,t) \in \Sigma_2^p$, which is in second level of the polynomial hierarchy, and consists of languages solvable in **NP** with access to a **coNP** oracle. $\mathit{SuccinctMCSP}(C,t)$ can be decided by non-deterministically generating a circuit of the required size and then using the **coNP** oracle to verify equivalence. As $\mathbf{BPP} \subset \Sigma_2^p$, the polynomial hierarchy collapses to **BPP**. The case when t is not an input parameter is more involved.

Example 5.5. A **BPP** algorithm deciding $MCSP_t$ gives a $BPTIME(\mathit{poly}(t(n)), \mathit{poly}(|C|))$ algorithm for Circuit-SAT. In particular if $t(n)$ is polynomial, this is a **BPP** algorithm.

Proof. Let $F = \{tt(f) \mid Size(f) < t(n)\}$, where $t(n) \in \omega(n), t(n) \in 2^{o(n)}$ is a fixed non-decreasing time-constructable function. We want to use it to decide satisfiability of a circuit C on n variables.

Take a random string r of length $t(n)^k$, and construct a circuit C_r on $\log |r|$ variables $y_1, \ldots, y_{\log |r|}$ of size $2^{|r|}/|r| = t(n)^k / k \log t(n)$. C_r is a exponential size circuit that can be constructed without any simplification on truth table. C_r encodes all the position of all the 1 bits in the string r.

Here, k is chosen such that $t(n + k \log t(n)) < t(n)^k / k \log t(n)$ As r is a random string, by Shannon's counting argument r will have exponentially high circuit complexity with high probability, so $[C_r] \notin F$ w.h.p.

Consider a circuit $C' = C \wedge C_r$. If C is unsatisfiable, $[C'] \equiv 0$, and thus $Size([C']) < t(n + k \log t(n))$, so $[C'] \in F$. Otherwise, for each satisfying assignment $a_1, \ldots, a_n$ to the inputs to C, $C'(a_1, \ldots, a_n, y_1, \ldots, y_{|r|}) = C_r(y_1, \ldots, y_{|r|})$. In particular, $Size([C']) \geq Size([C_r]) > t(n + k \log n)$.

Thus, a **BPP** algorithm deciding $MCSP_t$ gives a $BPTIME(poly(t(n), poly(|C|))$ algorithm for Circuit-SAT. If $t(n)$ is polynomial, this is a **BPP** algorithm (assuming the above is repeated with several r's to amplify success probability). $\square$

This case of $MCSP$ can be generalized to any property F which contains only easy functions.

Lemma 5.6. *Let F be a non-empty (for all n) property that contain only functions $f \in Size(t(n))$, for some (computable) $t(n) \in \omega(n)$. If there is a* **BPP** *algorithm A_F for L_F, then there is a non-uniform randomized algorithm for Circuit-SAT that on circuits of size m runs in time polynomial in $t(n)$ with size $t(n)$ advice.*

Proof. The structure of this proof resembles the original Rice's Theorem proof, with the main idea as in Example 5.5. Suppose we are given a circuit C on n variables. Construct circuit C_r on $k \log t(n)$ variables as in Example 5.5. Now, $C(x_1, \ldots, x_n) \wedge C_r(y_1, \ldots, y_{|r|})$ has circuit complexity $\geq t(n + |r|)$ iff C is satisfiable; in this case, $[C \wedge C_r] \notin F$. When C is unsatisfiable, circuit $C \wedge C_r$ is unsatisfiable. However, now it is possible that $\bar{0} \notin F$.

In this case, the algorithm needs to know a circuit C_f on $n + |r|$ variables for some $f \in F$. Given such circuit, either as an advice, circuit $(C(\vec{x}) \wedge C_r(\vec{y})) \oplus C_f(\vec{x}, \vec{y})$ will still have high circuit complexity when C is satisfiable; however when C is unsatisfiable, it will be equivalent to C_f, and so $\in F$. Thus, with high probability $A_F((C \wedge C_r) \oplus C_f)$ is 1 iff C is satisfiable. $\square$

Remark. If for every n there is an algorithm $Adv(n)$ producing in time polynomial in $t(n)$ a circuit C_f on n variables such that $[C_f] \in F$, then non-uniformity is not needed. In the case of $MCSP$ we had an algorithm for constructing advice.

5.2 Properties with Easy Sensitive Instances

Let us recall the Example 5.3: the property of being equivalent to one nice and easy function. This property contains only one function, by changing any bit of its truth table the resulting function is not in the language anymore. This idea could be extended to other properties using the concept of *Sensitivity*. The notion of sensitivity of Boolean functions was introduced by Cook, Dwork, and Reischuk [CDR86].

Definition 5.7 (Sensitivity)**.** Boolean function f is *sensitive* on the ith bit of input x if flipping that bit changes the value of $f(x)$. Sensitivity of f on input x denoted by $S(f, x)$ is the number of bits in x to which f is sensitive. Sensitivity of a function $s(f)$ is defined as $max_x S(f, x)$

View a property F as a Boolean function on strings of length 2^n for all n. First, suppose that F has maximal (2^n) sensitivity, and, moreover, for each n there is a maximally sensitive input $tt(f)$ where f has a small circuit C_f. This is a setting similar to Example 5.3. Let $A_F(C)$ be a **BPP** algorithm deciding, given a circuit C, if $tt(C) \in F$. Now, if C has at most 1 satisfying assignment, it is enough to check whether $A_F(C \oplus C_f) = A_F(C_f)$: if there is a satisfying assignment for C, it flips a sensitive bit of $tt(C_f)$, otherwise $tt(C \oplus C_f) = tt(C_f)$.

Theorem 5.8. *Let $L_F \in$ **BPP**. Suppose that for every n, F has sensitivity $s(F) \geq \mathcal{S}$: that is, there exists a function f such that $F(f) \neq F(f')$ for at least s functions f' which disagree with f on one input. Additionally, suppose that f is computed by a small circuit C_f.*

Then Circuit-SAT can be decided by a probabilistic algorithm with advice C_f in time $poly(|C_f|, |C|)$ with negligible probability of error on unsatisfiable inputs and success probability $1/2 * (\mathcal{S} - 1)/2^n$ on satisfiable inputs.

Proof. To use the idea described above we need to guarantee that the circuit C for which we want to decide satisfiability has at most one satisfiable assignment. This can be done by applying the Valiant-Vazirani reduction. Assuming that f is a highly sensitive input we have a non-trivial chance of hitting one of its sensitive bits. So we check $A_F(C(x \oplus r) \oplus C_f)$ where r is some random value. Repeating this process with more random values reduces the error probability. Below is the algorithm for the process described;

Algorithm AlgCS for CircuitSAT

Input: A circuit C on n inputs.

Advice: A circuit C_f such that $tt(C_f)$ is a $\mathcal{S}$-sensitive string for F, and a truth value of $F([C_f])$

1. Apply the Valiant-Vazirani reduction to C to obtain a list $C_1, \ldots, C_n$ with at least one uniquely satisfiable circuit in this list.
2. Let A_F be a BPP algorithm for L_F; assume without loss of generality that success probability of A_F has been amplified to $1 - p = 1 - 1/2^{m^\alpha}$ for input size m and constant $\alpha > 1$. For each C_i in this list, check if $F([C_f]) \neq A_F(C_i(x \oplus r_i) \oplus C_f)$. If so, accept. (Here, r_i are random strings of length n).

Let A_F run in time $O(n^d)$ for a constant d. Then running time of $AlgCS$ is $O(n * (|C_f| + |C|)^d)$.

Assume without loss of generality that success probability of A_F has been amplified to $1 - p = 1 - 1/2^{m^\alpha}$ for input size m and constant $\alpha > 1$. If C is unsatisfiable, then A_F rejects C with probability $1 - pn$.

If C is satisfiable, then the success probability will be $(1 - p)\mathcal{S}/2^{n+1} = \mathcal{S}/2^{n+1} - \mathcal{S}/2^{n+1+m^2} \approx O(\mathcal{S}/2^n)$. This is the probability of A_F giving the correct answer multiplied by the probability of hitting a sensitive bit of the advice. $\square$

Corollary 5.9. *Under the conditions of Theorem 5.8, Circuit-SAT can be decided by a randomized one-sided error algorithm $AlgCS'$ with advice in time polynomial in $|C| + |advice|$. $AlgCS'$ always rejects unsatisfiable circuits, and accepts satisfiable circuits with success probability $1/2 * (\mathcal{S} - 1)/2^n$, provided the advice is correct.*

Proof. The idea is to eliminate the error probability generated by algorithm A_F using non-uniformity. Because **BPP** $\subset$ **P/poly** [Adl78], the amount of randomness used by algorithm A_F can be at most of polynomial length, so A_F can be transformed to a nonuniform polynomial-time deterministic algorithm. The algorithm $AlgCS$ is modified to $AlgCS'$ which receives an additional advice for A_F. The success probability of $AlgCS'$ on satisfiable circuits is just the probability of hitting exactly one sensitive bit which is $1/2 * \mathcal{S}/2^n$. □

The idea of this Theorem 5.8 could be extended for checking satisfiability of other models. Generally any model closed under the Valiant-Vazirani reduction, $\oplus$ and conjunction supports the algorithm of Theorem 5.8. Examples of such models are branching programs and formulas over arbitrary basis which contains $AC^0[2]$ circuits.

Corollary 5.10. *Let $\mathcal{M}$ be a computational model which allows conjunction, $\oplus$ for two bits and an efficient, possibly randomized, reduction from $\mathcal{M}$-SAT to $\mathcal{M}$-UniqueSAT. Then theorem Theorem 5.8 and corollary Corollary 5.9 apply with circuits replaced by $\mathcal{M}$.*

5.3 Sensitivity of Symmetric Properties

The analogue of Rice's Theorem by Borchert and Stephan [BS96] could be seen as a special case of the black-box hypothesis. Here, using the idea of Theorem 5.8 we show why these properties are hard in general. Counting properties, Definition 3.1, are a special case of *Symmetric Properties.* F is a Symmetric Property of Boolean functions if it only depends on the Hamming weight of the truth table, so permuting the bits of the truth table doesn't change the value of F. Every symmetric property is described by a vector of the form $(F_0, F_1, \ldots, F_{2^n}) \in \{0,1\}^{2^n+1}$, where F_i is the value of F when $\#_1 = i$. One nice property of all symmetric properties (or functions) is their high sensitivity;

Lemma 5.11. *[Tur84] If F is non-trivial symmetric; then $s(F) \geq 2^n/2$.*

Proof. There is a number k such that $F_k \neq F_{k+1}$. Without loss of generality suppose that $k \geq 2^n/2$, then any string with k 1s has sensitivity k. Otherwise, any string with $k+1$ 1s has sensitivity $2^n - (k+1) \geq 2^n/2$. □

Theorem 5.12. *Let F be a symmetric property of Boolean functions and $L_F \in BPP$ be its corresponding metalanguage. Then Circuit-SAT can be decided by a randomized one-sided error polynomial-time algorithm with advice of size poly(n). This algorithm always rejects unsatisfiable circuits and accepts satisfiable circuits with probability at least* $1/4$.

Proof. Without loss of generality, let $k \leq 2^n/2$. Let $k$be the number of 1s in the truth table of f such that flipping a bit from 0 to 1 flips $F(f)$.

A sensitive input is encoded by $C_f =$ "$x > k$" which outputs 0 for inputs $x \leq k$, when x is interpreted as a binary number, and outputs 1 otherwise. The non-uniform algorithm for Circuit-SAT receives an advice consisting of 1) a polynomial-length string r^* to be used as randomness for A_F, and 2) a number $k \leq 2^n$. (size of k is less than n, representing the sensitive input). Given k as advice, we can construct a circuit $C_f(x) =$ "$x \geq k$" of polynomial size which encodes a highly sensitive input.

To decide satisfiability of C, using the advice we first generate C_f. Then, we run Circuit-SAT algorithm $AlgCS'(C)$ with advice C_f several times, and output 1 if on any of these runs $AlgCS'(C)$ outputs 1. Since when C is satisfiable $Prob[AlgCS'(C) = 1] > 1/4$, small constant number of runs is enough to see a 1 with high probability.

□

5.4 Eliminating Randomness for High Sensitivity

The algorithm of Theorem 5.8 always rejects unsatisfiable circuits, but on satisfiable circuits, if the sensitivity is low, it has exponentially small probability of hitting a sensitive bit. In this section, we improve the success probability of our algorithm when sensitivity is $2^{\delta n}$.

Theorem 5.13. *Suppose that there exist a property F with $L_F \in \mathbf{BPP}$, such that for some function f sensitivity $s(tt(f), F) \geq 2^{\delta n}$ and $|C_f| \leq 2^{o(n)}$ for constant $0 < \delta < 1$. Then there is a family of circuits $D_{m,n}$ of size $\leq 2^{o(n)}$ (with different function in $o(n)$) which decides Circuit-SAT.*

In [PP10], Paturi and Pudlak studied OPP algorithms for Circuit-SAT. OPP consists of all the randomized polynomial-time algorithms with one-sided error, but exponentially small success probability on satisfiable circuits. They showed that essentially all the OPP algorithms are unlikely to achieve high success probability for deciding Circuit-SAT. OPP is a very broad class of algorithms and their result rules out the possibility of anything better than brute-force. OPP algorithms can be converted to families of probabilistic circuits parametrized by n and m.

Definition 5.14. A *probabilistic circuit* $C(x, z)$ is a circuit that takes two types of inputs: x as the regular input and z as the required randomness.

Any algorithm with running time $T(n, m)$ can be encoded as a probabilistic circuit of size $T(n, m) \log(T(n, m))$; this can be done by a construction similar to Cook-Levin theorem, see [PF79]. The main result of [PP10], *Exponential amplification lemma*, is a recursive construction to amplify success probability of circuit families. Using the Exponential amplification lemma, Paturi and Pudlak obtained families of deterministic circuits of non-trivially small size that decides Circuit-SAT for several settings of parameters. In particular, if an OPP algorithm has success probability $2^{-\delta n}$ and running time $2^{o(n)} \cdot \tilde{O}(m)$, then deterministic circuits of sub-exponential size can decide Circuit-SAT.

Lemma 5.15 (Exponential amplification lemma[PP10])**.** *Let $\mathcal{G}$ be a family of probabilistic circuits of size bounded by $g(m,n)$ such that $\mathcal{G}$ decides Circuit-SAT with success probability $2^{-\delta n}$. Then there exist a circuit family $\mathcal{G}'$ deciding Circuit-SAT with success probability $2^{-\delta^2 n}$, for all large enough n, where size of circuits in $\mathcal{G}'$ is bounded by $g'(n,m) = O(g(\lceil \delta n \rceil) + 5, \tilde{O}(g(n,m)))$.*

proof of Theorem 5.13. First we convert $AlgCS'(C)$ of Corollary 5.9 to a circuit family. Next, we amplify the success probability of this family using several iterations of the exponential amplification lemma.

Let $G^0_{m,n}$ be the circuit family encoding algorithm $AlgCS'(C)$. $C_f, F([C_f])$ and r^* the required randomness can be easily hard-coded in circuits of $G^0_{m,n}$.

For concreteness, let $desc(C_f) = 2^{n^\gamma}$ denote a bound on the size of $|C_f|$. The size of the complete circuit $G^0_{m,n}$ is $O(2^{kn^\gamma} \cdot n^{k\gamma+1} \cdot m^k)$, where k is the exponent of the running time of A_F. Assuming that $m \leq |C_f|$ to bound smaller factors, $|desc(G^0_{m,n})| = O(2^{kn^\gamma} \cdot n^{(k+1)\gamma+1} \cdot m^k)$.

Now, we apply the Exponential amplification lemma for t iteration to $G^0_{m,n}$, where $t \in \omega(1)$ is a very slow growing function.

If $2^{o(n)} = 2^{\alpha(n)}$ is the bound on advice circuit $|C_f|$, and k is the exponent of the running time of $AlgCS'$, then we need $k^t * \alpha(n) < \beta(n)$, where $\beta(n) \in o(n)$. As t is non-constant, success probability becomes $2^{\delta^t n} \in 2^{o(n)}$. Now, using standard techniques to amplify success probability (with $2^{\delta^t n}+O(n)$ trials and fixing randomness by the averaging argument), obtain a deterministic circuit of sub-exponential size solving Circuit-SAT for circuits of description size m on n variables. □

5.5 Sensitivity and Black-Box Lower Bounds

The efficiency of the non-uniform algorithm in Theorem 5.8 depends on the size of the advice circuit. A small enough advice circuit computing a highly sensitive input can always be found if:

1. High query complexity in black-box setting implies high sensitivity.
2. There exist a high sensitivity instance computable by a small (enough) circuit.

A proof of the first statement would show a polynomial relationship between query-complexity and sensitivity. A proof of the second statement would show a connection between sensitivity and succinct description.

Proving the connection between query complexity and and sensitivity will resolve one closely related open problem: sensitivity conjecture (formulated in [NS94]).

Block sensitivity is a generalization of sensitivity. Let B be a subset of the bits of the input x. B is a *sensitive block* of f on input x, if flipping all the bits of B flips the value of $f(x)$.

Definition 5.16 (Block Sensitivity)**.** Block sensitivity of function f on input x denoted by $bs(f, x)$ is the maximum number of disjoint sensitive blocks of f on input x. Block sensitivity of a function $bs(f)$, is defined as $max_x bs(f, x)$ (The maximum sensitivity over all the possible inputs).

Block sensitivity upper bounds sensitivity (because bs is a generalization of s). The exact relationship between sensitivity and block sensitivity is unknown. Sensitivity conjecture states that sensitivity can be bounded by a polynomial of block sensitivity. It has been shown that $s(f)^2 \leq bs(f)$ for some functions [Rub95]. However, the best known upper-bound for bs in terms of s is still exponential[APV15]. See [HKP11] for a recent survey about progress on this conjecture.

Block sensitivity is polynomially related to several other complexity measures of Boolean functions such as certificate complexity, polynomial degree and quantum query complexity. In particular, it is polynomially related to randomized query complexity (defined in Definition 2.10).

Theorem 5.17. *[NS94]* $bs(f) \leq 2 * R_2(f)$

Therefore black-box query complexity upper bounds sensitivity.

Corollary 5.18. *Low black-box query complexity implies low sensitivity.*

Proof. sketch: Randomized query complexity upper bounds block sensitivity and block sensitivity upper bounds sensitivity. □

However, the converse of the above corollary is hard to prove. Any progress on that would be a great progress on the sensitivity conjecture. But, even assuming that block sensitivity is a polynomial function of sensitivity is not enough for $AlgCs$ to run in polynomial time, because $AlgCS$ needs a highly sensitive instance computable by a small circuit.

Looking back at the black-box hypothesis, we must pay attention that definition of hypothesis is a bit different from pure oracle access. In Conjecture 4.1 the black box algorithm knows the size of the circuit. This gives significant advantage for deciding some semantic properties.

Consider the property of being equivalent to a function f that has exponential circuit lower bounds. The black box algorithm immediately rejects any circuit of small size if it knows a lower bound. And on circuits of exponential size even if it needs to check the value of the circuit on all possible inputs, the whole process takes polynomial time with respect to the length of the input. So, the algorithm always takes polynomial time on all the inputs. Because of such cases, we restrict our notation of a counterexample to properties that have exponential query complexity on small circuits. We define the notion of a strong counter example, as some property that violates the conjecture on small circuits. More formally;

Definition 5.19. A property F is a *t-strong counterexample* to black-box hypothesis if $L_F \in \mathbf{BPP}$, yet any black-box algorithm requires query complexity (and thus running time) of $2^{\Omega(n)}$ on circuits computing functions f of circuit complexity less than t, for any circuit size. When we omit t, assume $t = 2^{o(n)}$.

For example, all point functions are computable by circuits of linear size. If the property of being a point function has a **BPP** algorithm, it is a strong counterexample. With this definition, we conjecture that if a strong counterexample exists then there exists a sensitive input of small size.

Conjecture 5.20 (Easy instance sensitivity conjecture)**.** Let $R_{2,t}(F)$ and $s_t(F)$ be randomized query complexity and sensitivity, respectively, over a subset of inputs f to F such that $f \in Size(t)$. Then there exists integer $k > 0$ and $t'(n) = poly(t(n))$ such that $R_{2,t}(F) \leq s_{t'}(F)^k$.

This is a generalization of sensitivity conjecture, where the original conjecture is for $t = N$. As an example, consider that checking satisfiability has exponential R_2, and its highly sensitive input, the constant zero function, has a small circuit. If this conjecture holds, we can conclude the following:

Corollary 5.21. *If F is a t-strong counterexample to the black-box hypothesis with $t = 2^{o(n)}$, and easy instance sensitivity conjecture holds for this t, then there is a family of circuits of sub-exponential size deciding Circuit-SAT.*

This follows immediately from theorem Theorem 5.13.

Chapter 6

Conclusions

The black-box hypothesis leaves us with several new questions each of which is an interesting direction for future research. Our algorithmic construction relies on the sensitivity of Boolean functions. Nevertheless, our results cannot rule out the possibility of other algorithms that may not use sensitivity and the Valiant-Vazirani reduction.

The non-uniformity of our algorithm for Circuit-SAT, resembles the proof of Rice's theorem. In that proof, a machine satisfying a property must be provided as advice for the argument to work. For uniform computation, the advice is compatible with any other machines. Eliminating the non-uniformity of our algorithm would result in a uniform sub-exponential algorithm for Circuit-SAT which violates ETH.

The further progress of our approach in understanding the connection of the black-box hypothesis and Circuit-SAT, largely depends on the easy instance sensitivity conjecture. The intuition of easy instance sensitivity suggests that a non-uniform randomized polynomial time algorithm is possible if a counterexample to black-box hypothesis exists. Moreover, our definition of strong counter example allows for refinement and weakening of the black-box hypothesis. Easy sensitivity conjecture seems provable for strong counter examples. This would be the first step toward proving this the general black-box hypothesis.

The easy instance sensitivity conjecture connects sensitivity of Boolean function with their descriptive complexity. Even independent of the black-box hypothesis, this is a fascinating open problem. In particular, we don't yet know if sensitivity conjecture implies easy instance sensitivity or vice versa. A good research direction is to prove one of these conjectures assuming that the other holds.

We also briefly discussed variants of the black-box hypothesis for other models of computation and implications of learnability. In fact, restrictions of circuit model are more realistic computational models. A more learning theoretic approach is to correlate the hardness of learning and the hardness of teaching in a model, similar to the correlation between white-box and black-box.

As explained in Chapter 4, black-box hypothesis trivially holds if the model under consideration is exactly learnable by oracle queries. We didn't consider other learning paradigms such as approximate learning or probably approximately correct (PAC) learning and their implications for the black-box hypothesis. Moreover, the interaction with the oracle of a function in many learning problems is not limited to input-output queries; other query types include equivalence, disjointness, subset, superset, etc (for definitions see [Ang88]). Comparing white-box and black box complexity by allowing other query types is interesting.

In this thesis, we used the word "hypothesis" at Russell Impagliazzo's suggestion, since both proving and disproving it would have interesting consequences. Currently, our intuition is not enough to support any of these possibilities. Therefore, it is very useful to prove/disprove the hypothesis under plausible complexity theoretic assumptions for which we already have some intuition. In particular, ETH (or strong ETH) and the existence of one way functions are relevant assumptions to try. The non-uniformity of our Circuit-SAT algorithm doesn't allow to relate our result with ETH easily. Also, we know that if one way functions exist the generalization of this hypothesis to promise problems cannot hold. Considering the recent progress on obfuscation, the existence of efficient indistinguishability obfuscator is also a natural assumption that can be used for proving/disproving the black-box hypothesis.

Bibliography

[Adl78] L. Adleman. Two theorems on random polynomial time. In *Proceedings of the Nineteenth Annual IEEE Symposium on Foundations of Computer Science*, pages 75–83, 1978.

[Ang88] Dana Angluin. Queries and concept learning. *Machine learning*, 2(4):319–342, 1988.

[APV15] Andris Ambainis, Krisjanis Prusis, and Jevgenijs Vihrovs. Sensitivity versus certificate complexity of boolean functions. *CoRR*, abs/1503.07691, 2015.

[BCW80] M. Blum, A.K. Chandra, and M.N. Wegman. Equivalence of free Boolean graphs can be tested in polynomial time. *Information Processing Letters*, 10:80–82, 1980.

[BDW02] Harry Buhrman and Ronald De Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(1):21–43, 2002.

[BGI$^+$01a] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In J. Kilian, editor, *Advances in Cryptology – CRYPTO'2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18. Springer-Verlag, Berlin, Germany, 2001.

[BGI$^+$01b] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. In *Advances in cryptology—CRYPTO 2001*, pages 1–18. Springer, 2001.

[BS96] Bernd Borchert and Frank Stephan. Looking for an analogue of rice's theorem in complexity theory. *Electronic Colloquium on Computational Complexity*, 3, 1996.

[CDR86] Stephen Cook, Cynthia Dwork, and Rüdiger Reischuk. Upper and lower time bounds for parallel random access machines without simultaneous writes. *SIAM Journal on Computing*, 15(1):87–97, 1986.

[GGH$^+$13] Shelly Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Anant Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 40–49. IEEE, 2013.

[HKP11] Pooya Hatami, Raghav Kulkarni, and Denis Pankratov. Variations on the sensitivity conjecture. *Theory of Computing, Graduate Surveys*, 2:1–27, 2011.

[HR00] Lane A Hemaspaandra and Jörg Rothe. A second step towards complexity-theoretic analogs of rice's theorem. *Theoretical computer science*, 244(1):205–217, 2000.

[HT01] L Hemaspaandra and Mayur Thakur. Rice-style theorems for complexity theory. *University of Rochester, Rochester, NY*, 2001.

[HU79] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[Nis91] Noam Nisan. CREW PRAMs and decision trees. *SIAM Journal on Computing*, 20(6):999–1007, 1991.

[NS94] Noam Nisan and Mario Szegedy. On the degree of boolean functions as real polynomials. *Computational complexity*, 4(4):301–313, 1994.

[PF79] Nicholas Pippenger and Michael J Fischer. Relations among complexity measures. *Journal of the ACM (JACM)*, 26(2):361–381, 1979.

[PP10] Ramamohan Paturi and Pavel Pudlák. On the complexity of circuit satisfiability. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 241–250, 2010.

[Ric53] Henry Gordon Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society*, 74(2):358–366, 1953.

[Rub95] David Rubinstein. Sensitivity vs. block sensitivity of boolean functions. *Combinatorica*, 15(2):297–299, 1995.

[Ruz81] Walter L Ruzzo. On uniform circuit complexity. *Journal of Computer and System Sciences*, 22(3):365–383, 1981.

[Sch80] J.T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the Association for Computing Machinery*, 27(4):701–717, 1980.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 475–484. ACM, 2014.

[Tur84] György Turán. The critical complexity of graph properties. *Information Processing Letters*, 18(3):151–153, 1984.

[VV86] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.

[Zip79] R.E. Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of an International Symposium on Symbolic and Algebraic Manipulation (EUROSAM'79)*, Lecture Notes in Computer Science, pages 216–226, 1979.