On the Black-Box Impossibility of Hardness in TFNP from One-Way Functions

by

Anas A. Ibrahim

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2026

© Anas A. Ibrahim 2026

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The complexity class TFNP contains all NP search problems for which a solution is guaranteed to exist for every instance. A proof of totality is needed for every problem in TFNP, so syntactic subclasses (such as PLS, PPA, PPAD, PPP, and PWPP) categorize many problems in TFNP according to the argument needed for totality. Although these subclasses contain many natural complete problems, the status of whether all of TFNP can be solved efficiently remains unknown, especially noting that TFNP is unlikely to have a complete problem. Furthermore, it has been shown that hardness of TFNP cannot be based on $\mathsf{P} \neq \mathsf{NP}$ unless $\mathsf{NP} = \mathsf{coNP}$.

A major line of work has sought to base TFNP hardness on cryptographic assumptions. Existing positive results obtain hardness for subclasses of TFNP from cryptographic primitives such as one-way permutations, collision-resistant hashing, indistinguishability obfuscation, factoring, and various others. However, the status of basing the hardness on a general assumption such as the existence of one-way functions remains unsuccessful.

There have been several attempts at providing evidence for this difficulty through black-box impossibility. Existing barriers rule out either restricted constructions of TFNP problems or restricted classes of reductions. In particular, [RSS16] rules out constructions whose instances have a bounded number of solutions, while [HKKS20, FGH+23] establish impossibility results for reductions satisfying various non-adaptivity and obliviousness restrictions.

In this work, we prove a semi-black-box impossibility result under a restriction on the construction: the construction can only query the one-way function once. Subject to this restriction, we allow the reduction to make polynomially many adaptive queries to the solver, as well as adaptive queries to the one-way function.

Acknowledgements

I want to thank my supervisor, Mohammad Hajiabadi, for his support, patience, and flexibility; he never imposed the slightest bias on my explorations, and always gave me the freedom to explore what I wanted at the pace I wanted, giving me the autonomous maturity I was seeking in this stage of my mathematical journey.

I thank my committee, Elena Grigorescu and Rafael Oliveira, for their helpful comments and feedback.

I extend warm thanks to my family and my (previous and current) close friends; your unconditional support, at any point in my life, helped get me here.

My first pure math professor, Daoud Siniora, told me once "when we write mathematical texts, we always use the pronoun 'we' instead of 'I,' because it's not only you who's writing, it's also all those who came before you whose work you build on."

My deepest gratitude, consequently, goes to the people addressed in the use of the pronoun "we" in this text: all the brilliant minds whose work I build on, and especially to those who taught said work to me. My dear pure mathematics professors, your close mentorship and support mattered greatly, and I would credit this work to you first.

Warm gratitude goes unforgotten to anyone who helped directly or indirectly at any point: the musical composers whose music I listened to while thinking about this work and writing it, acquaintances and friends who exchanged greetings with me on any day and supported me, and anyone who smiled at me one day and made me feel welcome far away from my home.

Table of Contents

Author's Declaration ii

Abstract iii

Acknowledgements iv

List of Figures vii

List of Tables viii

List of Abbreviations ix

1 Introduction 1

1.1 Prior Work . . . 3

1.1.1 Positive Results . . . 3

1.1.2 Negative Results . . . 5

1.2 This Work . . . 6

2 Background 9

2.1 Search Problems, Reductions, and Completeness . . . 9

2.2 Black-Box Constructions of TFNP hardness from One-Way Functions . . . 14

2.3 Main Open Problem . . . 18

3 Black-Box Separation Techniques in the Literature 24

3.1 Introduction and Motivation . . . 25

3.2 Properties of Good Oracles . . . 27

3.3 Techniques of Proving Sufficiency . . . 29

3.3.1 Encoding . . . 30

3.3.2 Coupling . . . 34

4 Main Contributions **37**

4.1 Towards Achieving Stability 37

4.1.1 Motivation and Challenges 38

4.1.2 Decision Tree Viewpoint 40

4.1.3 Simulatable Circuits 44

4.2 Adaptive Queries with One Oracle Gate 50

4.2.1 The Combinatorial Underworld 50

4.2.2 Combinatorial Structure of 1-Turán Sets 55

4.2.3 Simulating Circuits with One Oracle Gate 59

4.2.4 Lost in Translation: Challenges in Extending to Multiple Oracle Gates 63

4.3 One Round of Non-Adaptive Queries 71

References **74**

APPENDICES **78**

A Notations and Terminology **79**

B Known Theorems **82**

List of Figures

4.1 Edges of a complete bipartite graph as points a grid 51

4.2 All edges of $\mathcal{K}^2_{3,4}$ representing every 2-matching in $\mathcal{G}_{3,4}$. 53

4.3 A 3-uniform hypergraph on a 3×4 grid, with hyperedges drawn as smooth closed curves. 53

4.4 The three colored hyperedges viewed as 3-matchings in the bipartite graph $\mathcal{G}_{3,4}$. 54

4.5 Grid view of a rectangle $S \times H \subseteq [N] \times [M]$. 58

4.6 Bipartite view of a rectangle $S \times H$ 59

4.7 1-Turán set induced by the circuit that leaks y on the least satisfying assignment. 60

4.8 Visualizing Neighborhoods for $t = 1$ and $t = 2$. 65

4.9 Forward direction of 2. 66

4.10 2-Turán set of a non-simulatable circuit with two oracle gates 70

List of Tables

1.1 Selected positive results relating cryptographic assumptions to hardness in Total NP-Search Problems (TFNP) and its subclasses. 4

1.2 Comparison of the restrictions imposed by previous black-box barriers and by this work. 7

List of Abbreviations

CLS Continuous Local Search 4

EOL End-of-Line 4

FNP NP-*Search* Problems 1, 2, 9, 10, 12–14

PLS Polynomial Local Search 2

PPAD Polynomial Parity Argument for Directed graphs 2, 4

PPA Polynomial Parity Argument 2, 4

PPP Polynomial Pigeonhole Principle 2–4, 11, 15, 17

PWPP Polynomial Weak Pigeonhole Principle 2–4

SVL Sink-of-Verifiable-Line 4

TFNP Total NP-Search Problems viii, 1–7, 9–14, 16–18, 20

iO Indistinguishability Obfuscation 4

CRHF Collision-Resistant Hash Functions 3

OWF One-Way Functions 3–6

OWP One-Way Permutation 3, 14

PPT probabilistic polynomial time 6, 16, 18–20, 22, 33, 34, 45, 63, 80

Chapter 1

Introduction

I have discovered a truly marvelous proof of this proposition, which this margin is too narrow to contain.

Pierre de Fermat, 1637.
This begs the question: does knowing a proof exists make it easier to find?

One of the central goals of computational complexity theory is studying the class NP – the class of decision problems where "yes" instances have short proofs/certificates that are efficiently verifiable. Finding such certificates is captured by the class of NP-*Search* Problems (FNP) (F stands for "Function"). On the other hand, the class coNP is the class of decision problems where "no" instances have short proofs/certificates that are efficiently verifiable. If "no" instances of problems in NP have proofs/certificates, this would imply NP = coNP, which is widely believed to be false.

For some problems in FNP, deciding whether a certificate exists for an instance can be as difficult as finding the certificate. For example, it is well-known that if we can efficiently decide whether a boolean formula is satisfiable, then we can find the satisfying assignment by repetitively guessing the truth value of one variable and checking if the resulting simplified formula is satisfiable. In many cases, however, the aspects that go into solving a search problem in FNP are not captured by deciding the underlying decision problem. For example, it is currently unknown whether finding the prime factorization of a positive integer can be done efficiently, but the existence of one is well-known.

Consequently, for problems where it is known that a short proof exists for every instance, it is interesting to study the difficulty of finding such proof. In other words, does knowing that a theorem is true make it easier to find the proof?

Megiddo and Papadimitriou [MP91] introduced the class of TFNP – problems where existence of solutions to every instance is guaranteed. That is, TFNP is a subclass of FNP with

the additional requirement of totality. TFNP is usually considered as the search analogue of NP ∩ coNP, since decision problems in NP ∩ coNP have both "yes" and "no" short proofs/certificates.

Inclusion of a problem in TFNP needs a proof of totality, which is not necessarily easy to obtain. In particular, it is believed that no TFNP-complete problem exists (under oracles it does not, as shown in [Pud14]).[1]

Thus, the study of problems in TFNP is classically done by clustering them into syntactic subclasses, each with non-constructive arguments to prove totality as introduced by [JPY85, MP91, Pap94] – more modern expositions can be found in [GHJ+22, GP17, Das19, Hol21]. For example,

- the argument that "*every directed acyclic graph has a sink*" gives rise to the class "Polynomial Local Search (PLS)" [JPY85, Pap94],
- the pigeonhole principle gives rise to the classes "Polynomial Pigeonhole Principle (PPP)" and "Polynomial Weak Pigeonhole Principle (PWPP)" [Pap94, Jeř16],
- the argument that "*every undirected graph has an even number of odd-degree nodes*" gives rise to the class "Polynomial Parity Argument (PPA)" [Pap94], and
- the argument that "*every directed graph containing an unbalanced node (in-degree ≠ out-degree) must contain another*" gives rise to the class "Polynomial Parity Argument for Directed graphs (PPAD)" [Pap94].

These classes attracted attention for connecting seemingly distant domains. For example, PPAD is of particular interest, as a natural complete problem is the problem of finding a Nash equilibrium (which is provably total, [Nas51]). Additionally, many cryptographic assumptions can be phrased in terms of problems in TFNP (e.g., integer factoring, discrete log, ...etc.).

However, the place for TFNP in the hierarchy of complexity classes is still not well understood. If P = NP, then all problems in FNP (and by extension, all problems in TFNP) can be solved efficiently. It is not known whether P ≠ NP implies that some problem in TFNP is hard (i.e., cannot be solved in polynomial time). For that, such an implication would be shown through a reduction from an FNP-complete problem, but if a problem in TFNP is FNP-complete, then this would imply NP = coNP.[2] It is therefore unlikely that such a problem exists. Establishing completeness for the syntactic subclasses, while interesting, still leaves the possibility that all problems in TFNP can be solved efficiently. Thus, it is natural to seek evidence of hardness for some problem in TFNP.

[1] TFNP was referred to informally in [MP91] as a semantic class (as opposed to a syntactic class), and it was claimed that such semantic classes seem to have no complete problems.

Defining syntactic and semantic classes precisely is challenging, but it informally means that establishing that $P \in$ TFNP requires proving the defining totality condition, and there is no known single theory that is able to prove that $P \in$ TFNP for a given description of P.

One possible definition given by Sasha Razborov as a comment to Pavel's talk in [RPBD21] is that for semantic classes, "there is no known enumeration of turing machines solving problems in this class."

[2] To be more precise, a problem in TFNP is FNP-complete under *Turing reductions* if and only if NP = coNP. See **Theorem 2.1.6**.

1.1 Prior Work

Papadimitriou [Pap94] was the first to indicate that a natural potential source of evidence of hardness for problems in TFNP is by basing it on cryptographic assumptions, thus starting a line of research on this path. In particular, [Pap94] pointed out that the existence of a One-Way Permutation (OWP)[3] implies average-case hardness of PPP. Additionally, the existence of collision-resistant hashing (which can be implied by hardness of integer factoring or other primitives) gives rise to average-case hardness in PWPP.

These observations, however, rely on primitives with more structure than a general One-Way Functions (OWF).[4] While OWPs imply OWFs, known black-box separations show that an OWP cannot in general be constructed from an OWF in a black-box manner [Rud89, KSS11]. Likewise, Simon [Sim98] showed that Collision-Resistant Hash Functions (CRHF)s cannot be constructed from general OWFs in a black-box manner. This motivates the more general question of whether the existence of an arbitrary OWF alone can be used to establish hardness for some problem in TFNP.

1.1.1 Positive Results

A number of positive results establish hardness for problems in TFNP, or in its syntactic subclasses, from stronger or additional cryptographic assumptions. Some of the results most relevant to this line of work are summarized in **Table 1.1**.

[3] Loosely speaking, a permutation on $\{0,1\}^n$ which is easy to compute but hard to invert.

[4] Loosely speaking, a function on binary strings that is easy to compute but hard to invert.

Work	Assumption	Hardness statement	Relevance here
[BPR14, GPS15]	OWF + Indistinguishability Obfuscation (iO)	Hard-on-average distributions of End-of-Line (EOL) instances, and hence average-case hardness for PPAD, under the stated security assumptions.	Introduces and develops the use of Sink-of-Verifiable-Line (SVL) as an intermediate problem.
[Jeř16]	Hardness of integer factoring	A randomized polynomial-time reduction from general integer factoring to a problem in PPA.	Bases hardness of PPA on a concrete number-theoretic problem.
[HY17]	Hard distributions of SVL	Reduces SVL to a problem in Continuous Local Search (CLS), transferring average-case hardness of suitable SVL distributions to CLS.	Extends the SVL-based approach to another subclass of TFNP.
[HNY17]	OWF + additional assumptions	Constructs hard-on-average problems in TFNP from the average-case hardness associated with one-way functions together with an additional "fooling assumption"; an alternative result uses injective OWFs and Zaps [DN07].	Gets closer to basing TFNP hardness on OWFs, but still requires additional assumptions.
[HV21]	Discrete logarithm	Shows variants of the discrete logarithm problem complete for PPP and PWPP.	A notable positive result for PPP and PWPP.
[BCH+22]	LWE and iterated squaring	Shows that an efficient randomized algorithm solving EOL with non-negligible probability would yield an efficient algorithm breaking either LWE or iterated squaring.	Provides further evidence for hardness of PPAD from concrete cryptographic assumptions.

Table 1.1: Selected positive results relating cryptographic assumptions to hardness in TFNP and its subclasses.

The work of [BPR14] is particularly significant because they set the scene by introducing a technique by defining a promise search problem, SVL, that admits a reduction to EOL. Thus, constructing a hard-on-average distribution of SVL instances gives a corresponding source of average-case hardness for EOL. This idea was introduced in [BPR14] and subsequently used and refined in much of the later work.

1.1.2 Negative Results

Despite these positive results, no analogous result is known that bases worst-case hardness of a problem in TFNP solely on the existence of a general OWF. This has motivated a complementary line of work arised to explain this lack of success by asking whether such a result can be obtained at all using black-box techniques (informally, without access to the code implementing the one-way function).

To describe these results, we informally describe a black-box construction of a hard TFNP problem from a one-way function as follows:

1. **A construction:** an efficient oracle verifier $\mathbf{V}^{(\cdot)}$ that defines the TFNP problem with black-box access to the one-way function.

   In other words, for any function f, $\mathbf{V}^f$ is total. That is, for every instance i, there exists a solution s where $\mathbf{V}^f(i, s) = 1$.

2. **A reduction:** an efficient oracle algorithm $\mathbf{R}^{(\cdot)}$ where, for any function f and any oracle Φ^f that solves the constructed TFNP problem (i.e., $\Phi^f(i) = s$ where $\mathbf{V}^f(i, s) = 1$), $\mathbf{R}^{f,\Phi^f}$ can invert f (on a non-negligible fraction of images of f).

Basically, this notion is aimed to encapsulate the informal notion that, in a relativized world with access to a function f, if f is one-way then $\mathbf{V}^f$ is hard. This is equivalent to saying that if $\mathbf{V}^f$ can be efficiently solved, then f can be inverted.

So, one evidence that a construction from a hard TFNP problem from one-way functions would be to rule out such black-box constructions. That is, show that for every construction $\mathbf{V}^{(\cdot)}$ and reduction $\mathbf{R}^{(\cdot)}$, there exists a function f and an oracle $\Phi^{(\cdot)}$ that solves $\mathbf{V}^{(\cdot)}$ where $\mathbf{R}^{f,\Phi^f}$ cannot invert f.

The order of quantifiers here is important: if Φ is allowed to be implemented according to the code of $\mathbf{R}$, then such an impossibility result rules out *fully black-box constructions*. If one Φ works for any reduction $\mathbf{R}$, then it will rule out a larger class of *semi black-box constructions* [RTV04].

In the works of [RSS16, HKKS20, FGH+23], Φ was instead described to be an oracle that provides a satisfying assignment to satisfiable circuits. That is, instead of $\mathbf{R}$ providing an instance i to Φ^f, $\mathbf{R}$ can encode the verifier on input i, $\mathbf{V}^{(\cdot)}(i, \cdot)$, through a circuit $C^{(\cdot)}(\cdot)$ where the requirement that for every f and i there exists s where $\mathbf{V}^f(i, s) = 1$ translates to the requirement that for every f there exists u where $C^f(u) = 1$. Then, Φ^f can be redefined so that, on receiving a satisfiable circuit $C^{(\cdot)}$, $\Phi^f(C)$ gives u where $C^f(u) = 1$.[5]

It turns out that the problem of fully ruling out said black-box constructions is non-trivial. Starting with [RSS16] and followed by [HKKS20] and [FGH+23], each work proves the result either for

[5] We can define Φ where on an input circuit C where there exists some g where every u has $C^g(u) = 0$ (that is, C^g is unsatisfiable for some function g), $\Phi^f(C) = \perp$ regardless of f.

This doesn't give $\mathbf{R}$ any information about the hidden f, as if $\mathbf{R}$ was only bounded by the amount and length of queries to its oracles (but computationally unbounded), it can check satisfiability by brute force (over possible functions and assignments).

1. a certain class of constructions $\mathbf{V}^{(\cdot)}$ or
2. a certain class of reductions $\mathbf{R}^{(\cdot)}$.

Among other results, [RSS16] ruled out black-box constructions where the construction $\mathbf{V}^{(\cdot)}$ describes a problem in TFNP with a bounded number of solutions. [6]Thus, they showed that a black-box attempt to base TFNP hardness on OWFs may produce instances having nearly exponentially many solutions.

Then, [HKKS20] ruled out a class of reductions $\mathbf{R}^{(\cdot)}$ (in *fully black-box* constructions) by showing that for any reduction $\mathbf{R}^{(\cdot)}$ such that

- $\mathbf{R}^{f,\Phi^f}$ makes one round of non-adaptive queries to Φ and
- $\mathbf{R}^{f,\Phi^f}$ makes no queries to f prior to that round of non-adaptive queries (but possibly makes queries after)

there exists a Φ and an injective function f where $\mathbf{R}^{f,\Phi^f}$ cannot invert f.

The result of [FGH+23] strengthens this separation. In particular, it rules out a larger class of *semi-black-box* reductions, where the constructed solver Φ does not depend on the code of $\mathbf{R}$; so, one solver works simultaneously against every reduction. Their proof is also simplified and more combinatorial in nature.

To compare, [RSS16] allowed $\mathbf{R}$ to submit adaptive queries to Φ but the queries themselves are restricted (i.e., the construction $\mathbf{V}$ is restricted).

The problem of ruling out all fully black-box constructions for any construction $\mathbf{V}^{(\cdot)}$ or a TFNP problem and any reduction $\mathbf{R}$ that submits *adaptive* queries to Φ still remains open.

1.2 This Work

This work takes one more step in the direction of ruling out fully-black box constructions. In particular, we rule out *semi-black box constructions* of TFNP problems from one-way functions where the construction $\mathbf{V}^{(\cdot)}$ makes one query to the one-way function.

In particular, we show that there exists a function f and an oracle Φ where the following holds:

1. f is an injective function that maps n bits to $2n$ bits,
2. $\Phi^{(\cdot)}$ is an oracle that takes input a circuit $C^{(\cdot)}$ that satisfies the following:
   - $C^{(\cdot)}$ is promised to be satisfiable (for every g, there exists u where $C^g(u) = 1$).
   - $C^{(\cdot)}$ has one oracle gate.
3. For any probabilistic polynomial time (PPT) reduction $\mathbf{R}$ that submits satisfiable circuits with one oracle gate, $\mathbf{R}^{f,\Phi^f}$ cannot invert f except with negligible probability.

[6]That is, if for every f and i there exist subexponentially many s satisfying $\mathbf{V}^f(i,s) = 1$, then $\mathbf{R}^{f,\Phi^f}$ cannot invert f. Equivalently, if all submitted satisfiable circuits by $\mathbf{R}$ have subexponentially many satisfying assignments for every f, then $\mathbf{R}^{f,\Phi^f}$ cannot invert f.

Our proof employs ideas related to techniques appearing in [RSS16]. We show that every satisfiable oracle circuit containing only one oracle gate admits the structural properties needed so that queries to the TFNP solver can be fully simulated by queries to the one-way function (except with negligible probability). Thus, $\mathbf{R}^{f,\Phi^f}$ can be simulated by an algorithm $\mathbf{R}^f_\star$ that only makes queries to the one-way function.

To clarify the distinctions, we summarize all negative results in **Table 1.2** (in the language of satisfiable circuits).

Work	**Restricted object**	**Solver queries**	**Queries to f**	**Queries to Φ**
[RSS16]	Construction	Adaptive	Allowed	Bounded number of solutions
[HKKS20]	Reduction	One round of non-adaptive queries	None before solver queries	Any satisfiable circuit
[FGH+23]	Reduction	One	None before solver query	Any satisfiable circuit
This work	Construction	Polynomially many, adaptive	Allowed	One oracle gate per submitted circuit

Table 1.2: Comparison of the restrictions imposed by previous black-box barriers and by this work.

We remark that all submitted circuits must have a polynomial number of oracle gates since the reduction is polynomial-time.

We also make several additional contributions toward understanding the general open problem. We present techniques from the literature and discuss different approaches for attacking the conjectured black-box separation. We investigate the combinatorial structure underlying these approaches, including the structure that enables our result for circuits with one oracle gate and the difficulties that arise when moving beyond this restriction.

We also include an explicit extension of [FGH+23] to one round of non-adaptive queries, as [FGH+23] explicitly only allowed $\mathbf{R}$ to submit one query to Φ for the sake of simplicity, we show that the same argument extends naturally to a polynomial-size non-adaptive round.

The work is structured as follows:

1. **Chapter 2** provides background about search problems, fully black-box constructions, and provides a formulation of the main open problem that is studied by this work.
2. **Chapter 3** mainly investigates insights and techniques in the literature in black box separation.

   We provide two standard techniques: *encoding* and *coupling*. We note that *coupling* is simpler and requires fewer conditions than *encoding*, but we provide both for a full understanding of the literature.

3. **Chapter 4** studies the combinatorial structure of the problem, and provides the non-adaptive extension described above.

Chapter 2

Background

In this chapter, we introduce the relevant formal definitions and explain the background needed to motivate the main results.

We assume the reader is familiar with standard concepts and formalisms of computational complexity theory, including the Turing machines and oracle Turing machines. For a comprehensive background on these topics, we refer the reader to [Sip96] and [AB09].

We refer the reader to **Appendix A** for common notation and terminology used in this text.

The structure of this chapter is as follows:

1. In **section 2.1**, we introduce definitions of search problems, FNP, and TFNP and prove the first obstruction into showing hardness in TFNP: absence of FNP-complete problems in TFNP unless NP = coNP.
2. In **section 2.2**, we introduce the notion of black box constructions of TFNP problems from one-way functions and discuss the method used to rule out such reductions, motivating the next section.
3. **Section 2.3** formulates the main open problem (or conjecture) that is intended to be resolved and is studied by this work.

2.1 Search Problems, Reductions, and Completeness

In this section, we introduce definitions – adapted from [MP91] and [Jeř16] – to formally describe classes of search problems and reductions between them.

One important goal of this section is to show an important obstruction into proving hardness in TFNP: basing the hardness of TFNP on $\mathsf{P} \neq \mathsf{NP}$ would imply a collapse in the polynomial hierarchy.

We start with the formal definitions of search problems.

Definition 2.1.1 (Search Problems and FNP)**.** A *search problem* R is a relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$. An algorithm $\mathbf{A}$ is said to *solve* R if given $x \in \{0,1\}^*$ outputs $y \in \{0,1\}^*$ where $(x,y) \in R$ or $\perp$ if no such y exists.

An algorithm $\mathbf{V}$ is said to *decide* R if for any $x, y \in \{0,1\}^*$, $\mathbf{V}(x,y) = 1$ if $(x,y) \in R$ and $\mathbf{V}(x,y) = 0$ otherwise.

We define FNP to contain all search problems R that are decidable in polynomial time and there exists $c \in \mathbb{N}$ where $|y| \leqslant |x|^c$ for any $(x,y) \in R$.

So, the formal definition of a search problem is a relation, but it represents the computational problem: given x, find y where $(x,y) \in R$.

An example of a problem in FNP is the problem FSAT: given a boolean circuit $C : \{0,1\}^m \to \{0,1\}$ (with logic and/or/not gates), find a satisfying assignment (i.e., $u \in \{0,1\}^m$ where $C(u) = 1$) or report that none exists.

One can also define FNP as follows: for every language $L \in \mathsf{NP}$, collect all possible verifiers $\mathbf{V}$ and add the relation $R_{\mathbf{V}} := \{(x,y) : \mathbf{V}(x,y) = 1\}$ to FNP.

An important observation from this definition is that one language can have multiple verifiers. For example, if $L \in \mathsf{P}$, one possible verifier is $\mathbf{V}(x,y) = 1$ if y is the empty string and $x \in L$ (i.e., $\mathbf{V}$ decides L).

In that case, the search problem corresponding to that verifier is easy: if $x \in L$, output the empty string, otherwise output $\perp$. However, some verifiers of a language $L \in \mathsf{P}$ may not induce easy search problems. This motivates the definition of TFNP: collect all verifiers for the language of all binary strings $\{0,1\}^*$.

Definition 2.1.2 (The Class TFNP)**.** We define $\mathsf{TFNP} \subseteq \mathsf{FNP}$ such that for any $R \in \mathsf{FNP}$, $R \in \mathsf{TFNP}$ if for every $x \in \{0,1\}^*$ there exists $y \in \{0,1\}^*$ where $(x,y) \in R$ (i.e., R is *total*).

An example of a problem in TFNP is the problem PIGEON, where given a circuit $C : \{0,1\}^n \to \{0,1\}^n$, it asks to find either a pair $u \neq v$ where $C(u) = C(v)$ or a u such that $C(u) = 0^n$.

More formally, $(\langle C \rangle, z) \in$ PIGEON if and only if for some $n \in \mathbb{N}$, $C : \{0,1\}^n \to \{0,1\}^n$ is a circuit, and $z = u$ such that $C(u) = 0^n$ or $z = \langle (x,y) \rangle$ where $x, y \in \{0,1\}^n$, $x \neq y$, and $C(x) = C(y)$.

Observe that PIGEON $\in$ TFNP since if no u exists where $C(u) = 0^n$, then C is a circuit that maps 2^n inputs to $2^n - 1$ outputs, and the pigeonhole principle dictates that there must be two distinct inputs with the same output.[1]

[1] We remark that collecting the encodings $\langle C \rangle$ for circuits $C : \{0,1\}^n \to \{0,1\}^n$ may not strictly produce $\{0,1\}^*$, but invalid/malformed strings can be mapped to a trivial circuit (e.g., a circuit that maps a bit to itself), and such decoding is efficient as deciding whether an encoding is valid can be done efficiently.

To that end, another possible convention is to define TFNP (as treated in [JPY85]) by collect relations $R \in \mathsf{FNP}$ where $\mathrm{Dom}(R) \in \mathsf{P}$. The difference here is a matter of encoding.

Next, we define two standard types of reductions between search problems: *many-one reductions* (also known as Karp reductions) and *Turing reductions*.

Definition 2.1.3 (Reductions)**.** Let R, S be two search problems. We say that R is *many-one reducible* to S and write $R \leq_m S$ if there are polynomially computable functions $f, g : \{0,1\}^* \to \{0,1\}^*$ such that

1. for any $x \in \{0,1\}^*$, $x \in \mathrm{Dom}(R)$ implies $f(x) \in \mathrm{Dom}(S)$, and
2. for any $x, y \in \{0,1\}^*$, if $(f(x), y) \in S$, then $(x, g(x,y)) \in R$.

We say that R is *Turing reducible* to S, and write $R \leq_T S$ if there exists a polynomial time oracle Turing machine $\mathbf{R}^{(\cdot)}$ such that for any oracle $\Psi : \{0,1\}^* \to \{0,1\}^*$ solving S, $\mathbf{R}^{\Psi}$ solves R. We call $\mathbf{R}$ a *Turing reduction* from R to S.

Here, Ψ solves S if for any $x \in \{0,1\}^*$, we have $(x, \Psi(x)) \in S$ whenever $x \in \mathrm{Dom}(S)$ and $\Psi(x) = \perp$ otherwise.

The first property of many-one reductions says that f maps instances of R to instances of S, and the second one says that g maps solutions of S on images of f back to solutions of R.

One example of the use of many-one reductions is in defining syntactic subclasses of TFNP by setting canonical complete problems. For example,

$$\mathsf{PPP} := \{R \in \mathsf{FNP} : R \leq_m \textsc{Pigeon}\}.$$

In other words, PPP is the subclass of TFNP with the canonical complete problem Pigeon. Observe that for $R, S \in \mathsf{FNP}$, if $R \leq_m S$ and $S \in \mathsf{TFNP}$, then $R \in \mathsf{TFNP}$ (by **Property 2** of many-one reductions). This implies $\mathsf{PPP} \subseteq \mathsf{TFNP}$.

To formally address complete problems, we define what it means for a problem to be *complete* for some subclass of FNP:

Definition 2.1.4 (Completeness)**.** Let $\mathsf{C} \subseteq \mathsf{FNP}$. We say that $S \in \mathsf{C}$ is C*-complete* (under many-one reductions) if for any $R \in \mathsf{C}$, $R \leq_m S$.

Furthermore, we say that $S \in \mathsf{C}$ is C*-complete under Turing reductions* if for any $R \in \mathsf{C}$, $R \leq_T S$.

Observe that many-one reductions imply Turing reductions (the reduction $\mathbf{R}$, on input x, can query the oracle on $f(x)$ to obtain y or $\perp$ and return $g(x,y)$ or $\perp$, respectively), but the converse is not necessarily true. Thus, completeness under many-one reductions implies completeness under Turing reductions.

Observe that this definition shows Pigeon is PPP-complete.

The very first problem that was found to be complete for FNP was FSAT. We state this foundational theorem as it plays part in proving the intended theorem that follows it.

Theorem 2.1.5 (Cook-Levin)**.** Let FSAT $\in$ FNP be the relation linking boolean circuits with their satisfying assignments.

Then, FSAT is FNP-complete.

Proof. See [AB09, Sip96] for a full proof.

The idea is that for any verifier $\mathbf{V}$ defining a problem in FNP, one can encode an instance i into a polynomially-sized boolean circuit C_i whose satisfying assignments encode accepting computation tableaux of $\mathbf{V}(i, s)$ for some witness s. Since $\mathbf{V}$ runs in polynomial time, the validity of such a tableau can be enforced by a polynomial number of local constraints, and C_i can therefore be constructed in polynomial time.

Thus, any satisfying assignment of C_i yields a witness s such that $\mathbf{V}(i, s) = 1$, while any valid witness s gives rise to a satisfying assignment of C_i. Hence, this gives a polynomial-time reduction from any problem in FNP to FSAT. □

Now, we are ready to provide a proof of the first barrier in showing TFNP-hardness. The proof is adapted from [MP91] with technical changes to adopt the formal definitions presented.

Theorem 2.1.6. There exists $S \in$ TFNP that is FNP-complete *under Turing reductions* if and only if NP = coNP.

Proof. Suppose NP = coNP. Then, there exists a relation $\overline{\text{FSAT}}$ linking unsatisfiable boolean circuits with short proofs of unsatisfiability. Consequently, $\overline{\text{FSAT}} \in$ FNP.

Define $S = \text{FSAT} \cup \overline{\text{FSAT}}$. That is, S links satisfiable boolean circuits with their satisfying assignments and unsatisfiable circuits with their short proof of unsatisfiability.

Observe that $\text{Dom}(S) = \{0,1\}^*$ so S is total (every boolean circuit with one output bit is either satisfiable or unsatisfiable). Thus, $S \in$ TFNP. Furthermore, we claim that S is FNP-complete under Turing reductions.

To see this, pick $R \in$ FNP. By Cook-Levin, there are polynomially computable f, g that reduce R to FSAT.

Then, we construct a Turing reduction $\mathbf{R}$ from R to S that queries the S-solver on $f(x)$ for an input x to get an output y. If y is a satisfying assignment for $f(x)$ (which $\mathbf{R}$ can check in polynomial time), then $\mathbf{R}$ will output $g(x, y)$. Otherwise, $\mathbf{R}$ will output $\perp$. It is clear that $\mathbf{R}$ solves R.

Conversely, assume that some $S \in$ TFNP is FNP-complete under Turing reductions. We construct short proofs for unsatisfiable circuits and a polynomial time verifier, showing that the language of unsatisfiable circuits, $\overline{\text{SAT}}$ is in NP, implying that NP = coNP.

Let $\mathbf{R}$ be the reduction from FSAT to S, and let $\mathbf{J}$ be the decider for S.

For a circuit C, we know that if $\langle C \rangle \notin \text{Dom}(\text{FSAT})$ (i.e. $\langle C \rangle \in \overline{\text{SAT}}$), then $\mathbf{R}^{\Phi}(C)$ outputs $\perp$ (given oracle access to an S-solver Φ).

Thus, the transcript of interaction of $\mathbf{R}$ with Φ would be the proof of unsatisfiability (which is clearly of polynomial length since $\mathbf{R}$ is polynomial time).

The verifier (call it $\mathbf{V}$), given a circuit C and a transcript τ, will do the following.

1. For every query-answer pair (x, y), check if $y = \perp$. If so, return 0, since S is total, we can't have the answer of $\Phi(x)$ be $\perp$.

   Now, run $\mathbf{J}(x, y)$, and if it returns 0 then return 0. Otherwise, proceed to the next step.

2. Simulate $\mathbf{R}^{\Phi}(C)$ using the transcript τ sequentially.

   If at any point, the query made by $\mathbf{R}$ disagrees with the subsequent x in a query-answer pair (x, y) in τ, return 0.

   Furthermore, if the number of queries made by $\mathbf{R}$ disagree with the number of query-answer pairs (x, y) in τ, return 0.

3. Finally, check if the output of $\mathbf{R}$ is $\perp$ (meaning that the circuit is unsatisfiable). If so, return 1. Otherwise, return 0.

Basically, $\mathbf{V}$ checks if the given certificate τ is a valid transcript first, and then simulates $\mathbf{R}$ to see if it returns $\perp$. The input C will be unsatisfiable if and only if $\mathbf{R}$ returns $\perp$, creating a valid transcript along the way. Also, $\mathbf{V}$ checks that whenever the same query appears more than once, it receives the same answer

This shows $\overline{\text{FSAT}} \in \mathsf{NP}$, concluding the proof. □

Note that in the converse direction, if S were not total, then $\mathbf{J}$ will be useless in verifying a query-answer pair (x, y) where $y = \perp$, since $\perp$ cannot be verified as a certificate in polynomial time without deciding $\text{Dom}(S)$. In other words, the totality of S here gives us an efficient way to decide $\text{Dom}(S)$.[2]

As $\mathsf{FP} \neq \mathsf{FNP}$ is equivalent[3] to $\mathsf{P} \neq \mathsf{NP}$ (where FP is the class of search problems that can be solved in polynomial time), showing hardness in TFNP from $\mathsf{P} \neq \mathsf{NP}$ is equivalent to showing this implication:

- If some problem in FNP cannot be solved in polynomial time, then there exists a problem in TFNP that cannot be solved in polynomial time.
- If any problem in TFNP can be solved in polynomial time, then any problem in FNP can be solved in polynomial time.

[2] This makes the above theorem equally true if TFNP was defined to have problems with polynomially-decidable domains as opposed to total domains.

[3] The implication from $\mathsf{FP} = \mathsf{FNP}$ to $\mathsf{P} = \mathsf{NP}$ is easy, as any algorithm that solves a search problem decides its underlying language.

For the other implication, observe that if we can decide whether a boolean circuit of length n is satisfiable in time n^k, then we can plug in 0/1 in one of the variables and check whether the resulting circuit of length $\leqslant n - 1$ is satisfiable in time $2(n-1)^k$, then recursively do the same for the satisfiable choice. Thus, one can find the satisfying assignment in time $\leqslant 2(n-1)^k + 2(n-2)^k + \cdots + 2 \leqslant 2n^{k+1}$ which is polynomial time.

Showing any of the two implications usually requires finding a problem in $P \in \mathsf{TFNP}$ where, if there is a polynomial time algorithm $\mathbf{A}$ that solves it, then one can use that fact to solve any problem in FNP in polynomial time. The way to use it is usually to construct an algorithm that solved an FNP-complete problem (e.g., FSAT) in polynomial time by making polynomially many calls to $\mathbf{A}$.

This is where the significance of Turing reductions come into play: treat $\mathbf{A}$ as a black-box correct solver and the calls to $\mathbf{A}$ as constant-time as opposed to polynomial time, and solve an FNP-complete problem.

This is exactly what **Theorem 2.1.6** rules out. It says that if such a proof of TFNP hardness is possible, it would imply that the chosen TFNP problem is FNP-complete under Turing reductions, making $\mathsf{NP} = \mathsf{coNP}$. So, the likely assumptions of both $\mathsf{P} \neq \mathsf{NP}$ and $\mathsf{NP} \neq \mathsf{coNP}$ cannot imply TFNP hardness in a black-box manner. That is, we are not looking at the code of $\mathbf{A}$, or the fact that it exists, to solve all of NP.

So, a slightly stronger assumption is to assume that some problem in FNP cannot be solved except with low probability for a random instance. That is, some problem in FNP is *average-case hard*.[4] Thus, the next general assumption to consider is the existence of a *one-way function*. Furthermore, similar to Turing reductions, we investigate a similar (probabilistic) notion in cryptography called *black-box constructions*.

2.2 Black-Box Constructions of TFNP hardness from One-Way Functions

The main aim of this section is to define and develop understanding of black-box constructions of a TFNP hard problem from one-way functions. We start by an informal discussion of showing how one would usually go about a proof that the security of some cryptographic primitive would imply the hardness of a problem in TFNP. Then, we introduce the notion of *fully black-box constructions* of a worst-case hard problem in TFNP from one-way functions, and we show that this notion collects common aspects of previous informal constructions. Then, we study the definition closely and define *semi black-box constructions*.

Similar to a Turing reduction, to show hardness in TFNP from any cryptographic primitive, one shows that with access to a solver of a problem in TFNP, one can break the hardness or security of that primitive, usually through a randomized experiment. This establishes that the security of the primitive would imply the hardness of the chosen problem.

Let's consider for example the existence of a OWP. Suppose $f : \{0,1\}^* \to \{0,1\}^*$ is a one-way permutation. That is, f is a one-way function where $f|_{\{0,1\}^n} : \{0,1\}^n \to \{0,1\}^n$ is a permutation for all n.

We can consider the problem $\textsc{Invert}^f$: given $y \in \{0,1\}^*$, find $x \in \{0,1\}^*$ where $f(x) = y$.

[4] That is, informally speaking, instead of "for every efficient algorithm $\mathbf{A}$ there exists an instance on which $\mathbf{A}$ fails" we assume "for every efficient algorithm $\mathbf{A}$, $\mathbf{A}$ fails on most intances."

We observe that this problem is total: for any $y \in \{0,1\}^*$, if $n = |y|$, there must exist $x \in \{0,1\}^n$ where $f(x) = y$ by assumption.

Clearly, if one has access to a solver of $\textsc{Invert}^f$, one can break the one-wayness of f; an algorithm, given an image y, feeds it directly to the solver to obtain the preimage.

In fact, [Pap94] indicated that one can also show from one-wayness of a permutation f the hardness of the problem $\textsc{Pigeon}^f$: given a boolean circuit $C^f : \{0,1\}^m \to \{0,1\}^m$ with f oracle gates find $u \in \{0,1\}^m$ with $C^f(u) = 0^m$ or find $u, v \in \{0,1\}^m$ with $u \neq v$ and $C^f(u) = C^f(v)$.

The idea was simple: if Φ^f solves $\textsc{Pigeon}^f$ (for every permutation f), then for the circuit $C^f : \{0,1\}^n \to \{0,1\}^n$ defined as

$$C^f(u) := f(u) \oplus y,$$

we have $\Phi^f(C) = f^{-1}(y)$ since the fact that f is a permutation implies that there cannot exist $u \neq v$ with $C^f(u) = C^f(v)$ (as this would imply $f(u) = f(v)$). Thus, the only solution would be x where $C^f(x) = 0$ or $f(x) = y$.

This shows that the existence of one-way permutations imply the hardness of the subclass PPP.

Two common components of both of these constructions are the following:

1. a description of a problem P^f (that depends on f in the description) that is total
2. a description of an oracle algorithm $\mathbf{R}$ that, with access to a solver Φ^f of P^f, can break the one-wayness of f.

We formalize these ideas in the next definition, inspired by [RTV04] and partly by [HKKS20] and [BH13]: (refer to **Appendix A** for conventions/terminology about families of functions)

Definition 2.2.1 (Fully Black-Box Constructions)**.** Let $\mathcal{J} \subseteq \mathcal{B}$ be a family of functions.

A *fully black-box construction* of a worst-case hard TFNP problem from one-way functions *relative to the family* $\mathcal{J}$ consists of the following:

1. **A construction:** a polynomial-time oracle algorithm $\mathbf{V}^{(\cdot)}$ that defines a problem in TFNP (the verifier).

   That is, there exists a constant $c \in \mathbb{N}^+$ where for every $n \in \mathbb{N}$, $g \in \mathcal{J}_n$, and instance $i \in \{0,1\}^*$, there exists a solution $s \in \{0,1\}^*$ with $|s| \leqslant |i|^c$ and $\mathbf{V}^g(1^n, i, s) = 1$.

2. **A reduction:** a PPT algorithm $\mathbf{R}^{(\cdot),(\cdot)}$ where for every $n \in \mathbb{N}$, $f \in \mathcal{J}_n$, and oracle $\Phi^{(\cdot)}$ that satisfies

   - $\Phi^f(i) = s$ where $\mathbf{V}^f(1^n, i, s) = 1$ for any instance $i \in \{0,1\}^*$

   we have

   $$\Pr_{x \leftarrow \{0,1\}^n}\left[\mathbf{R}^{f,\Phi^f}(1^n, f(x)) = x' \text{ where } f(x) = f(x')\right] \geqslant \lambda(n)$$

   for some non-negligible function λ.

If we omit "relative to the family $\mathcal{J}$," then we consider $\mathcal{J} = \mathcal{B}$.

Now, before we discuss the intuition of this definition, a few important notes to clarity or emphasize about this definition are as follows:

- The oracle verifier $\mathbf{V}^{(\cdot)}$ must satisfy totality *for every choice of* $g \in \mathcal{J}_n$.

  So, the quantifier order here is $\forall g \forall i \exists s\, (\mathbf{V}^g(1^n, i, s) = 1)$.

- $\mathbf{R}^{f,\Phi^f}$ has access to an *arbitrary* function $f \in \mathcal{J}_n$. So, $\mathbf{R}$ must be able to invert every $f \in \mathcal{J}_n$.

- The oracle $\Phi^{(\cdot)}$ is defined for every choice of hidden function. One can view it formally a function $\Phi^{(\cdot)} : \mathcal{J}_n \times \{0,1\}^* \to \{0,1\}^*$ with two inputs.

  However, once the hidden $f \in \mathcal{J}_n$ is fixed, $\mathbf{R}^{f,\Phi^f}$ has access to Φ^f *on the same* f. So, if $\mathbf{R}$ queries i to Φ, s where $\mathbf{V}^f(1^n, i, s) = 1$ can feed $\mathbf{R}$ information about the hidden f.

- This is a *fixed-parameter* fully black-box construction (following conventions of [BH13] and [FGH+23]).

  This means that both the construction and the reduction can query the one-way function on inputs of length n. Malformed inputs will have the oracles output $\perp$.

The intuition of this definition is meant to encapsulate a proof of TFNP hardness from one-way functions that doesn't assume have access to the code of the one-way function.

In order to do that, one does the following:

1. Define a problem in TFNP but in a relativized world with access to f (this is meant to model the assumption that a one-way function can be implemented efficiently).

   The requirement is that the definition must be a problem in TFNP independent of the choice of the function.

2. Show that for any f, if f is one-way, then $\mathbf{V}^f$ is worst-case hard.

   Equivalently, one can show that if $\mathbf{V}^f$ is not worst-case hard, then f is not one-way. This is where the reduction $\mathbf{R}$ that makes use of a solver Φ^f of $\mathbf{V}^f$ comes into play.

   The reduction is meant to show that if $\mathbf{V}^f$ can be solved efficiently (modeled by access to Φ^f), then f can be inverted. This is where the similarity to Turing reductions shows up.

So in summary, the primitives exist and are efficiently implementable, but we don't have access to the code of the implementation.

This construction is not too restrictive and is common when attempting to construct cryptographic primitives from others. In fact, hardness of PPP from the existence of one-way permutations pointed out by [Pap94] is a fully black-box construction. We walk through the definition through this example as follows:

1. The chosen family $\mathcal{J} \subseteq \mathcal{B}$ is such that $f \in \mathcal{J}_n$ if and only if $f : \{0,1\}^n \to \{0,1\}^n$ is a permutation.
2. $\mathbf{V}^{(\cdot)}$ is meant to define the problem Pigeon.

   So, given an instance $C^{(\cdot)} : \{0,1\}^m \to \{0,1\}^m$ which is a circuit with oracle gates with input length n and output length $\ell(n)$ (the output length of functions in $\mathcal{J}_n$) and a solution s, $\mathbf{V}$ works as follows:

   - If $s = (u, v)$, then $\mathbf{V}^f\left(1^n, C^{(\cdot)}, u, v\right)$ simulates $C^f(u)$ and $C^f(v)$ by forwarding the queries by C to f, then checks if $C^f(u) = C^f(v)$ and $u \neq v$.
   - If $s = u$, then $\mathbf{V}^f\left(1^n, C^{(\cdot)}, u\right)$ simulates $C^f(u)$ similarly and checks if $C^f(u) = 0^m$.

   Observe that this verifier satisfies totality independent of the one-way function.[5]
3. $\mathbf{R}^{f,\Phi^f}(1^n, y)$ feeds the oracle circuit $C^{(\cdot)} : \{0,1\}^n \to \{0,1\}^n$ where $C^g(u) := g(u) \oplus y$ to Φ^f and returns the answer of $\Phi^f(C)$. This succeeds with 100% probability.

We observe that this reduction $\mathbf{R}$ doesn't make any queries to f and makes only one query to Φ. We remark that other reductions are allowed to submit adaptive queries to both Φ and f.

An example of a reduction that is *not black-box* is a Karp reduction (as pointed out in [Fis12]). For example, the Cook-Levin theorem makes access to the code of the algorithm M_A that decides the language A to construct the circuit.

[5] Checking if the input is a valid encoding of an oracle circuit where oracle gates have the specific fixed length of input/output can be done efficiently, and malformed strings can be mapped to a trivial circuit. Thus, issues of encoding are ignored.

In *semi black-box constructions*, the order of quantifiers between $\mathbf{R}$ and Φ and f is reversed. So, loosely speaking,

$$\exists \mathbf{R} \forall f, \Phi (\Phi^f \text{ solves } \mathbf{V}^f \implies \mathbf{R}^{f,\Phi^f} \text{ inverts } f),$$

we have

$$\forall f, \Phi \ \exists \mathbf{R} (\Phi^f \text{ solves } \mathbf{V}^f \implies \mathbf{R}^{f,\Phi^f} \text{ inverts } f).$$

More formally, one may say that for any sequence of functions $(f_n)_{n\in\mathbb{N}}$ where $f_n \in \mathcal{J}_n$, and any sequence of oracles $(\Phi_n^{(\cdot)})_{n\in\mathbb{N}}$ where $\Phi_n^{f_n}$ solves $\mathbf{V}^{f_n}$, there exists a PPT $\mathbf{R}$ where for every n,

$$\Pr_{x\leftarrow\{0,1\}^n}\left[\mathbf{R}^{f_n,\Phi_n^{f_n}}(1^n, f_n(x)) = x' \text{ where } f_n(x) = f_n(x')\right] \geqslant \lambda(n).$$

We now turn to an important consideration: the choice of family $\mathcal{J}$. If one wishes to show that a fully black-box construction exists of a worst-case TFNP hard problem from one-way functions, one must show it relative to the family $\mathcal{B}$ (all polynomially bounded and length-regular functions).

The main focus of this work is to *rule out* black-box constructions. One can do so by ruling out constructions relative to a subfamily $\mathcal{J} \subseteq \mathcal{B}$.

For that, observe that for any families of functions $\mathcal{N}, \mathcal{M} \subseteq \mathcal{B}$ with $\mathcal{N} \subseteq \mathcal{M}$ if a fully (respectively, semi) black-box construction exists relative to $\mathcal{M}$, then a fully (respectively, semi) black-box construction exists relative to $\mathcal{N}$. This is because the construction for $\mathcal{M}$ remains total when quantifying over functions in $\mathcal{N}$ and the reduction for $\mathcal{M}$ can invert any function in $\mathcal{M}$ and therefore any function in $\mathcal{N}$.

[FGH+23] ruled out a restricted class semi black-box constructions relative to the family $\mathcal{J} \subseteq \mathcal{B}$ where $\mathcal{J}_n$ collects all injective functions mapping n bits to $n+1$ bits. This is a step towards ruling out black-box constructions relative to the full family $\mathcal{B}$.

In this work, we mainly consider ruling out semi black-box constructions relative to the family $\mathcal{F}$, where $\mathcal{F}_n$ collects injective functions $f : \{0,1\}^n \to \{0,1\}^{2n}$ mapping n bits to $2n$ bits.

In general, we can define $\mathcal{F}^k$ to collect all injective functions mapping n bits to $k(n)$ bits for some polynomial $k(n) \geqslant n+1$, and since $\mathcal{F}^k \subseteq \mathcal{B}$, this would also be sufficient. Thus, we will consider $f \in \mathcal{F}^k$ to be functions from $f : [N] \to [M]$, where $N = 2^n$ and $M = 2^{k(n)}$ (the usual choice throughout this work is $M = N^2$).

2.3 Main Open Problem

Loosely speaking the order of quantifiers in a fully black-box reduction is

$$\exists \mathbf{V}^{(\cdot)}, \mathbf{R}^{(\cdot)} \ \forall \Phi \ \forall f \in \mathcal{F} (\Phi^f \text{ solves } \mathbf{V}^f \implies \mathbf{R}^{f,\Phi^f} \text{ inverts } f).$$

Our aim is to rule out fully black-box reductions. Thus, we wish to show

$$\forall \mathbf{V}^{(\cdot)}, \mathbf{R}^{(\cdot)} \ \exists \Phi \ \exists f \in \mathcal{F} (\Phi^f \text{ solves } \mathbf{V}^f \text{ but } \mathbf{R}^{f,\Phi^f} \text{ does not invert } f).$$

Observe here that the oracle Φ can be chosen according to the code of $\mathbf{R}$.

A stronger result would be to rule out semi black-box reductions:

$$\forall \mathbf{V}^{(\cdot)} \exists \Phi \ \exists f \in \mathcal{F} \forall \mathbf{R}^{(\cdot)} (\Phi^f \text{ solves } \mathbf{V}^f \text{ but } \mathbf{R}^{f,\Phi^f} \text{ does not invert } f).$$

In other words, one oracle Φ and function f are chosen so that Φ and f are one-way over any PPT $\mathbf{R}$.

Following the footsteps of [FGH+23] and [HKKS20], it would suffice to choose an oracle Φ that has the power to solve all of (relativized) TFNP.

To that end, we observe that for any chosen verifier $\mathbf{V}^{(\cdot)}$ in a black-box construction, and any instance $i \in \{0,1\}^*$, one can encode the algorithm $\mathbf{V}^{(\cdot)}(i, -)$ into a circuit $C_i^{(\cdot)}(-)$, so for any $g \in \mathcal{F}_n$ and any s, $\mathbf{V}^g(i, s) = C_i^g(s)$. This idea is shown in the following theorem.

Theorem 2.3.1 (Relativized Cook-Levin)**.** Let $\mathcal{J} \subseteq \mathcal{B}$ be a family of functions, and let $\mathbf{V}^{(\cdot)}$ be a polynomial-time oracle verifier where $\mathbf{V}^f$ defines a problem in FNP for every $f \in \mathcal{J}_n$ (for every n).

Then, for every $n \in \mathbb{N}$, every instance $i \in \{0,1\}^*$, there exists a polynomial-size oracle circuit $C_i^{(\cdot)}$ such that for every $g \in \mathcal{J}_n$ and every valid witness s,

$$C_i^g(s) = \mathbf{V}^g(1^n, i, s).$$

Moreover, $C_i^{(\cdot)}$ can be constructed in polynomial time.

Proof. We cite [For94] (specifically Lemma 2.1), and [GJ86]. Furthermore, [AB09] mentions in the notes of Chapter 3 that Cook-Levin does relativize.

We provide a sketch for completeness. Fix n and an instance i. Since $\mathbf{V}^{(\cdot)}$ runs in polynomial time, its computation on (i, s) can be encoded by a polynomial-size oracle circuit $C_i^{(\cdot)}$, where every oracle query made by $\mathbf{V}$ is represented by an oracle gate.

Thus, for every $g \in \mathcal{J}_n$ and every valid witness s,

$$C_i^g(s) = \mathbf{V}^g(1^n, i, s).$$

The construction depends only on i and the code of $\mathbf{V}^{(\cdot)}$, and hence can be carried out in polynomial time without depending on the particular choice of g. □

Thus, the requirement of totality, which states that for any $g \in \mathcal{F}_n$ and any i there exists s where $\mathbf{V}^g(i, s) = 1$ translates to the requirement of satisfiability:

- for any $g \in \mathcal{F}_n$, there exists s where $C^g(s) = 1$.

Thus, it suffices that the oracle Φ solves a promise version of boolean satisfiability:

- given an oracle circuit $C^{(\cdot)}$ where C^f is satisfiable for every f, $\Phi^f(C)$ outputs a satisfying assignment u where $C^f(u) = 1$.

We formulate this idea in the following statement of the conjecture, then we show that this statement is sufficient to rule out all semi black-box constructions of a worst-case hard problem in TFNP from one-way functions.

Here, we use the notation $\mathcal{C}_n$ to denote the set of all *satisfiable* oracle circuits.

That is, $\mathcal{C}_n$ collects all $C^{(\cdot)} : \{0,1\}^m \to \{0,1\}$ where for every $f \in \mathcal{F}_n$, [6]there exists $u \in \{0,1\}^m$ where $C^f(u) = 1$ (where the input length m is some polynomial in n).

Now, the formal statement of the main conjecture is as follows:

Conjecture 2.3.2 (Main Conjecture)**.** There exist oracles f and Φ that satisfy the following:

1. $f \in \mathcal{F}$, so f can be interpreted as a sequence of functions $f_n \in \mathcal{F}_n$ where f_n is an injective function that maps n bits to $2n$ bits,
2. $\Phi = (\Phi_n^{(\cdot)})_{n\in\mathbb{N}}$ where for all n, $\Phi_n^{(\cdot)}$ is defined as follows:
   - Φ_n^g outputs $\perp$ on any invalid input.

     That is, on any input that does not encode an oracle circuit $C^{(\cdot)} : \{0,1\}^m \to \{0,1\}$ with oracle gates that have input length n and output length $2n$.
   - If $C^{(\cdot)}$ describes a valid oracle circuit (as described above), and $C^{(\cdot)}$ is *not satisfiable* (or $C^{(\cdot)} \notin \mathcal{C}_n$), then $\Phi_n^f(C) = \perp$ for any $f \in \mathcal{F}_n$.

     In other words, Φ_n first checks if there exists $g \in \mathcal{F}_n$ where $C^g(u) = 0$ for all $u \in \{0,1\}^m$. If so, $\Phi_n^f(C)$ outputs $\perp$ (disregarding f).
   - If $C^{(\cdot)} \in \mathcal{C}_n$, then $\Phi_n^f(C)$ outputs u where $C^f(u) = 1$.

     That is, $\Phi_n^f(C)$ outputs a satisfying assignment of C^f for any $f \in \mathcal{F}_n$.
3. For any PPT $\mathbf{R}$, there exists a negligible function λ where for any n,

$$\Pr_{\mathbf{R},x\leftarrow\{0,1\}^n}\left[\mathbf{R}^{f_n,\Phi_n^{f_n}}(f_n(x)) = x\right] \leqslant \lambda(n).$$

Usually, we fix the parameter n and work with $\mathcal{F}_n$ and Φ_n. Now, this conjecture is sufficient to rule out semi black-box constructions by the following theorem.

Theorem 2.3.3. Suppose **Conjecture 2.3.2** holds. Then, there are no semi black-box constructions of a worst-case hard problem in TFNP from one-way functions.

Proof. Suppose, towards a contradiction, that there exists a semi black-box construction of a worst-case hard problem in TFNP from one-way functions.

By the discussion above, it suffices to consider the construction relative to the family $\mathcal{F}$. Let $\mathbf{V}^{(\cdot)}$ denote the verifier of this construction.

[6]Recall that $\mathcal{F}$ denotes the family of injective functions mapping n bits to $2n$ bits.

Let $f = (f_n)_{n\in\mathbb{N}} \in \mathcal{F}$ and $\Phi = (\Phi_n^{(\cdot)})_{n\in\mathbb{N}}$ be the pair of oracles whose existence is guaranteed by **Conjecture 2.3.2**.

We construct a solver Ψ for the problem specified by $\mathbf{V}^{(\cdot)}$.

Fix $n \in \mathbb{N}$ and an instance $i \in \{0,1\}^*$. By **Theorem 2.3.1**, we can construct an oracle circuit $C_i^{(\cdot)}$ such that, for every $g \in \mathcal{F}_n$ and every valid solution s,

$$C_i^g(s) = \mathbf{V}^g(1^n, i, s).$$

Since $\mathbf{V}^{(\cdot)}$ satisfies the totality requirement of a black-box construction, for every $g \in \mathcal{F}_n$ there exists a solution s such that $\mathbf{V}^g(1^n, i, s) = 1$. Therefore, $\forall g \in \mathcal{F}_n \; \exists s \, C_i^g(s) = 1$, and hence $C_i^{(\cdot)} \in \mathcal{C}_n$.

Thus, $\Psi_n^f(i) := \Phi^f(C_i)$ is an oracle that solves $\mathbf{V}^{(\cdot)}$. In particular, $\Psi_n^{f_n}$ is a valid solver relative to the function f_n supplied by **Conjecture 2.3.2**.

Since $\mathbf{V}^{(\cdot)}$ is assumed to be a semi black-box construction, applied to the sequence $(f_n)_{n\in\mathbb{N}}$ and the sequence of solvers $(\Psi_n^{(\cdot)})_{n\in\mathbb{N}}$, there exists a PPT algorithm $\mathbf{R}$ which inverts f with non-negligible probability. In particular, for the required values of n,

$$\Pr_{x\leftarrow\{0,1\}^n}\left[\mathbf{R}^{f_n,\Psi_n^{f_n}}(1^n, f_n(x)) = x\right] \geq \varepsilon(n)$$

for some non-negligible function ε.

We now construct a PPT algorithm $\widetilde{\mathbf{R}}^{(\cdot),(\cdot)}$ having oracle access directly to f_n and $\Phi_n^{f_n}$. On input $f_n(x)$, the algorithm $\widetilde{\mathbf{R}}^{f_n,\Phi_n^{f_n}}$ simulates $\mathbf{R}^{f_n,\Psi_n^{f_n}}$. Queries to f_n are answered directly. Whenever $\mathbf{R}$ makes a query i to $\Psi_n^{f_n}$, $\widetilde{\mathbf{R}}$ constructs the circuit $C_i^{(\cdot)}$ and answers the query with $\Phi_n^{f_n}\left(C_i^{(\cdot)}\right)$.

By the definition of $\Psi_n^{f_n}$, this is exactly the answer that $\mathbf{R}$ would have received from its oracle $\Psi_n^{f_n}$. Hence the two executions are identical, and therefore

$$\Pr_{x\leftarrow\{0,1\}^n}\left[\widetilde{\mathbf{R}}^{f_n,\Phi_n^{f_n}}(f_n(x)) = x\right] = \Pr_{x\leftarrow\{0,1\}^n}\left[\mathbf{R}^{f_n,\Psi_n^{f_n}}(1^n, f_n(x)) = x\right].$$

Consequently, $\widetilde{\mathbf{R}}^{f_n,\Phi_n^{f_n}}$ inverts f_n with non-negligible probability.

This contradicts property 3 of **Conjecture 2.3.2**, which completes the proof. □

Now, a stronger and sufficient result would be sampling f uniformly at random for each n:

Theorem 2.3.4. Suppose there exists an oracle Φ such that the following holds for every $n \in \mathbb{N}$

1. For every $C \in \mathcal{C}_n$ and $g \in \mathcal{F}_n$, $\Phi^g(C)$ outputs $u \in \{0,1\}^m$ such that $C^g(u) = 1$.
2. For every PPT $\mathbf{R}$,

$$\Pr_{\mathbf{R}}\left[\mathbf{R}^{f,\Phi^f}(f(x)) = x : \begin{array}{l} f \leftarrow \mathcal{F}_n \\ x \leftarrow \{0,1\}^n \end{array}\right] \leqslant \varepsilon(n)$$

   for some negligible function ε.

Then, **Conjecture 2.3.2** holds.

The main challenge in this theorem is reversing the order of quantifiers: if for every $\mathbf{R}$ there exists a choice of $f \in \mathcal{F}$, we want to show that we can get one $f \in \mathcal{F}$ that works for all $\mathbf{R}$.

Proof. Since there are countably many oracle Turing machines, enumerate all PPT inverters $\mathbf{R}_1, \mathbf{R}_2, \ldots$, and let ε_i be a negligible function where

$$\Pr_{\mathbf{R}_i}\left[\mathbf{R}_i^{f,\Phi^f}(f(x)) = x : \begin{array}{l} f \leftarrow \mathcal{F}_n \\ x \leftarrow \{0,1\}^n \end{array}\right] \leqslant \varepsilon_i(n).$$

It suffices to find $f^* \in \mathcal{F}$ such that for every $i, c \in \mathbb{N}^+$, there exists $N_{i,c} \in \mathbb{N}^+$ where

$$\Pr_{\mathbf{R}_i, x \leftarrow \{0,1\}^n}\left[\mathbf{R}_i^{f^*,\Phi^{f^*}}(f^*(x)) = x\right] \leqslant n^{-c}$$

for every $n \geqslant N_{i,c}$.

To simplify the notation, denote $I_{i,n}(f) := \Pr_{\mathbf{R}_i, x \leftarrow \{0,1\}^n}\left[\mathbf{R}_i^{f,\Phi^f}(f(x)) = x\right]$ for $f \in \mathcal{F}_n$.

For $i, c \in \mathbb{N}^+$, we choose $N_{i,c}$ as follows:

1. Since $\mathbb{E}_{f \leftarrow \mathcal{F}_n}[I_{i,n}(f)] \leqslant \varepsilon_i(n)$, Markov (see **B.0.1**) implies

$$\Pr_{f \leftarrow \mathcal{F}_n}[I_{i,n}(f) > n^{-c}] \leqslant n^c \varepsilon_i(n).$$

2. Since ε_i is negligible, we choose $N_{i,c}$ large enough so that
   - $N_{i,c} \geqslant i + c$ and
   - $\varepsilon_i(n) \leqslant \frac{1}{2} \cdot 2^{-(i+c)} \cdot n^{-c}$ or $n^c \varepsilon_i(n) \leqslant \frac{1}{2} \cdot 2^{-(i+c)}$.

Recall that we need to construct a sequence $\{f_n\}_{n \in \mathbb{N}}$ where $f_n \in \mathcal{F}_n$ so that $I_{i,n}(f_n) \leqslant n^{-c}$ for all $i, c \in \mathbb{N}^+$ and $n \geqslant N_{i,c}$.

To that end, fix some $k \in \mathbb{N}^+$, and we will choose $f_k \in \mathcal{F}_k$. We employ the probabilistic

method:

$$\begin{aligned}
&\Pr_{g \leftarrow \mathcal{F}_k}\left[I_{i,k}(g) \leqslant k^{-c} \text{ for all } i, c \in \mathbb{N}^+ \text{ where } N_{i,c} \leqslant k\right] \\
&= 1 - \Pr_{g \leftarrow \mathcal{F}_k}\left[I_{i,k}(g) > k^{-c} \text{ for some } i, c \in \mathbb{N}^+ \text{ where } N_{i,c} \leqslant k\right] \\
&\geqslant 1 - \sum_{N_{i,c} \leqslant k} \Pr_{g \leftarrow \mathcal{F}_k}\left[I_{i,k}(g) > k^{-c}\right] && \text{(by the union bound)} \\
&\geqslant 1 - \frac{1}{2} \cdot \sum_{N_{i,c} \leqslant k} 2^{-(i+c)} && \text{(by choice of } N_{i,c}) \\
&\geqslant 1 - \frac{1}{2} \cdot \sum_{i+c \leqslant k} 2^{-(i+c)} && \text{(by choice of } N_{i,c}) \\
&\geqslant \frac{1}{2} > 0.
\end{aligned}$$

This implies there exists $f_k \in \mathcal{F}_k$ where $I_{i,k}(f_k) \leqslant k^{-c}$ for all $i, c \in \mathbb{N}^+$ where $N_{i,c} \leqslant k$.

If we take $f^* = \bigcup_{k \in \mathbb{N}+} f_k$ we get the desired result. □

This is a standard technique, as used by [Sim98], [FGH+23], [BD19], and many more. This proof is inspired partially by [MF21].

[FGH+23] showed **Conjecture 2.3.2** but with a restricted class of reductions that submit only one query to Φ and prior to any queries to f.

Our work can be viewed as restricting reductions only by making all submitted circuits to Φ have one oracle gate. This corresponds to restricting constructions $\mathbf{V}$ to only make one query to the one-way function.

Before describing our methods, the next chapter will be dedicated to investigating techniques in the literature providing ways to attack **Conjecture 2.3.2** by attacking **Theorem 2.3.4**.

Chapter 3

Black-Box Separation Techniques in the Literature

The aim of this chapter is to study what makes a good choice of oracle Φ, covering the most important ideas in the literature and different ways to attack the problem. It is divided into four sections:

1. We start in **section 3.1** by walking the reader through the intuition of the main difficulty of attempting to attack **Conjecture 2.3.2**, explaining that naive constructions of Φ fail.
2. Then, we consider approaches in the literature for general black box separations for constructing a good candidate Φ in **section 3.2**.

   In particular, we contextualize the framework of [BH13].
3. **Section 3.3** formally provides two central techniques that are used in the literature to transition towards full black-box separation: *encoding* and *coupling*.

   While coupling is stronger than encoding, we provide both for a more thorough investigation of the literature. We show using coupling that a property called *stability* is sufficient to obtain to resolve **Conjecture 2.3.2**.

Before we proceed, we list notation/conventions that we will use throughout this chapter and the next:

1. For the remaining portion of this text, we will usually fix $n \in \mathbb{N}$ and define parameters with respect to n.

   Thus, we will replace $\mathcal{F}_n$ by $\mathcal{F}$ and $\mathcal{C}_n$ by $\mathcal{C}$.
2. All circuits $C^{(\cdot)} \in \mathcal{C}$ will be of input length m and will have at most t oracle gates (where $m(n), t(n)$ are polynomials in terms of n).

   Thus, an oracle Φ is a function $\Phi : \mathcal{F} \times \mathcal{C} \to \{0,1\}^m$.
3. $\mathrm{Tr}(C^f(u))$ denotes the transcript (set of query/answer pairs) of the interaction of the oracle circuit $C^{(\cdot)}$ with the oracle gate f on input u.

4. $N = 2^n$, $M = 2^{k(n)}$, and we often use $[N]$ to replace $\{0,1\}^n$ and $[M]$ to replace $\{0,1\}^{k(n)}$. So, now for $f \in \mathcal{F}$ has domain $[N]$ and codomain $[M]$ instead.

   We will replace $k(n) \geqslant n$ when needed, but for now we address the size of the codomain M abstractly for the sake of generality (often, we use $M = N^2$).

5. $\mathcal{F}^<$ is the set of partial assignments $T : [N] \to [M]$ such that there exists some $f \in \mathcal{F}$ where $T \subseteq f$ (i.e., T is injective on its domain).
6. For $T \in \mathcal{F}^<$, we define $\mathrm{Cube}(T) := \{f \in \mathcal{F} : T \subseteq f\}$.
7. For $f \in \mathcal{F}$, $x \in [N], y \in [M]$, we define $f_{x \to y} \in \mathcal{F}$ as follows:
   (a) If $y \in \mathrm{Im}(f)$ with preimage $x^\star$, then we swap $f(x)$ with y: $f_{x\to y}(z) := \begin{cases} f(x) & \text{if } z = x^\star \\ y & \text{if } z = x \\ f(z) & \text{otherwise} \end{cases}$.
   (b) If $y \notin \mathrm{Im}(f)$, then $f_{x\to y}(z) := \begin{cases} y & \text{if } z = x \\ f(z) & \text{otherwise} \end{cases}$.
8. For $T \in \mathcal{F}^<$ and $f \in \mathcal{F}$, we define $f_T \in \mathcal{F}$ inductively (on $|T|$) as follows:[1]
   (a) If $T = \varnothing$, then $f_T := f$.
   (b) Let $(x, y) \in T$ and let $T^\star := T \setminus \{(x, y)\}$. If $g := f_{x\to y}$, we define $f_T := g_{T^\star}$.

3.1 Introduction and Motivation

In **section 2.2**, we observed that the inverter $\mathbf{R}$ is defined so that in an execution of $\mathbf{R}^{f,\Phi^f}$, the oracle Φ has access to the same f given to $\mathbf{R}$, and $\mathbf{R}$ does not know about f except the queries made to f and to Φ.

Thus, the problem of inverting f for $\mathbf{R}$ can be thought of as an interactive game (very similar to the famous Wordle), where $\mathbf{R}$, given y, tries to guess a hidden property $f^{-1}(y)$ of a hidden object f.

In doing so, $\mathbf{R}$ interacts with an oracle Φ that provides partial information about f through the satisfying assignment.[2] Generally, our aim is then to find an oracle Φ that leaks little information about f.

As pointed out in [FGH+23], the naive constructions of Φ fail:

- If Φ returns the lexicographically smallest (or largest) satisfying assignment to C^f, then $\mathbf{R}$, given y, can query Φ on the circuit

$$D^f(u) := \mathbf{1}[u = 1^n] \vee \mathbf{1}[f(u) = y]$$

  (that is, observe, satisfiable for every $f \in \mathcal{F}$) to obtain the preimage of y.

[1] This definition implicitly assumes that the resulting function is the same independent of the order of T. One would need to prove for $x \neq x'$ and $y \neq y'$ that $(f_{x\to y})_{x'\to y'} = (f_{x'\to y'})_{x\to y}$. We remark that this is true, and we leave its proof as an exercise for the reader.

[2] $\mathbf{R}$ interacts with f too, but it can be shown that a query to f can be done using two queries with one gate or one query with two gates Φ with some specific circuits.

- If Φ returns a uniformly random satisfying assignment to C^f, then the same circuit D^f yields a preimage of y with probability $\geqslant 1/2$.

We remark importantly that Φ does not know which y the reduction is trying to invert. How to precisely state what it means for Φ to not help $\mathbf{R}$ invert y? We give an idea on how to formulate that in the rest of this discussion.

In Wordle, a particularly committed player can parse the list of words, and each time the game gives information about the right letter, the player can execute a script that eliminates all invalid words. Additionally, the final answer provided by the player is counted as a query.

These two points motivate the following two useful assumptions that we make about $\mathbf{R}$:

1. We imagine $\mathbf{R}$ as computationally unbounded but only bounded by the queries it makes to its oracles and the size of such queries.[3]

   Thus, when we say $\mathbf{R}$ is of runtime at most t, we mean that there are at most t circuits submitted to Φ of size at most t, and at most t queries to f.

2. We assume without loss of generality that if $\mathbf{R}$ wishes to submit $x^\star$ as its final answer, then it submits a dummy query to Φ prior to termination that is intended to query $f(x^\star)$. This doesn't affect $\mathbf{R}$ being bounded by polynomially many queries.[4]

From these two assumptions, we can model the interactive game as follows:

- Every time $\mathbf{R}$ submits a circuit $C^{(\cdot)}$ to Φ and Φ answers with u, then $\mathbf{R}$ can brute force every $g \in \mathcal{F}$ and see if $\Phi^g(C) = u$.[5]

  This restricts the range of possibilities of the hidden f to $\mathcal{U}_u^C := \{g \in \mathcal{F} : \Phi^g(C) = u\}$.

- Suppose $\mathbf{R}^{\Phi^f}$ submits a sequence of circuits $C_1, C_2, \ldots, C_k$ getting answers $u_1, u_2, \ldots, u_k$ respectively then terminates.

  Let $\mathcal{H} := \mathcal{U}_{u_1}^{C_1} \cap \mathcal{U}_{u_2}^{C_2} \cap \cdots \cap \mathcal{U}_{u_k}^{C_k}$.

  Then, $\mathbf{R}$ wins the interactive game on input $f \in \mathcal{F}$ and $y \in \mathrm{Im}(f)$ if and only if

  $$\mathrm{leak}_y(\mathcal{H}) := \left\{f^{-1}(y) : f \in \mathcal{H}\right\}$$

  has size 1 (otherwise the final query submitted doesn't hit $f^{-1}(y)$).

It is natural to explore what properties of $\mathcal{U}_u^C$ or the answer provided by an oracle Φ to provide sufficient conditions for Φ to satisfy the conditions of **Theorem 2.3.4**.

We will examine in the next section approaches in the literature to describe good oracles for general black box separations.

[3] This is often useful in the literature of black box separations, cause you want to know if the black box reduction really just narrows down to the computational power of $\mathbf{R}$ or is inherently impossible even with unbounded computation.

[4] To formally address this assumption, one proves that any $\mathbf{R}$ succeeds with probability ε if and only if a simulator $\mathbf{S}$ that adds this extra query succeeds with probability ε.

[5] in ruling out semi-black box reductions, we assume $\mathbf{R}$ has access to the code of Φ.

3.2 Properties of Good Oracles

One important property that is used often in the literature is by looking at the *transcript* of interaction of the oracle circuit C with f on input u (i.e., set of query/answer pairs to f in computing $C^f(u)$).

If that transcript has y in its image, then $\mathbf{R}$ would win the interactive game since, given u, it can manually execute $C^f(u)$ and query f when it needs.

It turns out that having the oracle consistently not leak y in the transcript of $C^f(u)$ is a slightly stronger property than what is usually sufficient.

In this section, we provide a formal and useful framework adapted from [BH13] to analyze properties that make a good candidate oracle Φ:

1. determining sets
2. stability, almost-stability, and bending sets

We will later on prove that for an oracle to be sufficient, we only need "stability," but we provide the other properties as well to give a complete understanding of the literature, as well as showing that one can obtain stability by adding an extra property of leaking images with low probability (**Theorem 3.2.4**).

Let's start by defining determining sets.

Definition 3.2.1 (Determining Sets)**.** Let Φ be an oracle and $C^{(\cdot)} \in \mathcal{C}$ be a satisfiable circuit.

For $f \in \mathcal{F}, T \in \mathcal{F}^{<}$, we say that T *breaks* f if $\Phi^{f_T}(C) \neq \Phi^f(C)$.

A collection of sets $\{I_f\}_{f\in\mathcal{F}}$ are called *t-determining* (relative to Φ and C) if the following holds:

1. $I_f \subseteq [N]$ for every $f \in \mathcal{F}$ and $|I_f| \leqslant t$.
2. For any $f, g \in \mathcal{F}$ where $\Phi^f(C) \neq \Phi^g(C)$, then either $f\,|_{I_f}$ breaks g or $g\,|_{I_g}$ breaks f.

One nice property that easily follows from the definition is described in the following lemma.

Lemma 3.2.2. Let Φ be an oracle, $C^{(\cdot)} \in \mathcal{C}$ be a satisfiable circuit, and let $\{I_f\}_{f\in\mathcal{F}}$ be t-determining.

Suppose for some $f, g \in \mathcal{F}$ we have $f\,|_{I_f\cup I_g} = g\,|_{I_f\cup I_g}$. Then, $\Phi^f(C) = \Phi^g(C)$.

Proof. Suppose the opposite. Then, either $f\,|_{I_f}$ breaks g or $g\,|_{I_g}$ breaks f. Suppose WLOG that $T = f\,|_{I_f}$ breaks g. Since $f\,|_{I_f\cup I_g} = g\,|_{I_f\cup I_g}$, then $g\,|_{I_f} = f\,|_{I_f} = T$. So, $g_T = g$ which contradicts that T breaks g. □

The next important property is *stability* (i.e., small "bending sets").

Definition 3.2.3 (Stable Oracles)**.** Let Φ be an oracle and $C^{(\cdot)} \in \mathcal{C}$ be a satisfiable circuit.

We say that Φ is *t-stable* relative to C, $f \in \mathcal{F}$, and $y \in [M]$ if

$$\Pr_{x \leftarrow [N]} \left[\Phi^f(C) \neq \Phi^{f_{x \to y}}(C) \right] \leqslant \frac{t}{N}.$$

We collect the witnesses $x \in [N]$ of instability in a set $B^C_{f,y}$ called the *bending set*. That is,

$$B^C_{f,y} := \left\{ x \in [N] : \Phi^f(C) \neq \Phi^{f_{x \to y}}(C) \right\}$$

and so equivalently, $\left| B^C_{f,y} \right| \leqslant t$.

We say that Φ is *almost t-stable* relative to C and $y \in [M]$

$$\mathop{\mathbb{E}}_{f \leftarrow \mathcal{F}} \left[\Pr_{x \leftarrow [N]} \left[\Phi^f(C) \neq \Phi^{f_{x \to y}}(C) \right] \right] \leqslant \frac{t}{N}.$$

We say that Φ is *t-stable* (resp. almost t-stable) if it's t-stable (resp. almost t stable) relative to every $C \in \mathcal{C}$, $f \in \mathcal{F}$, and $y \in [M]$.

Inspired by [FGH+23], from the nice properties of t-determining sets (hence the name) is that when you don't leak an image y too often in your determining sets, you get almost stability:

Theorem 3.2.4 (Sufficient conditions to become almost stable)**.** Let Φ be an oracle, $C^{(\cdot)} \in \mathcal{C}$ be a satisfiable circuit, and $y \in [M]$.

Suppose the following:

1. There are t-determining sets $\{I_f\}_{f \in \mathcal{F}}$.
2. $\Pr_{f \leftarrow \mathcal{F}} [y \in f(I_f)] \leqslant \frac{t}{M}$.

Then, Φ is almost $4t$-stable relative to C and y.

Proof. Observe if $\Phi^f(C) \neq \Phi^{f_{x \to y}}(C)$ for some $x \in [N]$, then by **Lemma 3.2.2**, f and $f_{x \to y}$ disagree on some $a \in I_f \cup I_{f_{x \to y}}$, so $f(a) \neq f_{x \to y}(a)$.

Then either $a \in \{x, f^{-1}(y)\}$. This implies one of four cases: $x \in I_f, x \in I_{f_{x \to y}}$ $y \in f(I_f)$, and $y \in f(I_{f_{x \to y}})$. In other words,

$$\Pr_{f,x} \left[\Phi^f(C) \neq \Phi^{f_{x \to y}}(C) \right] \leqslant \Pr_{f,x}[x \in I_f] + \Pr_{f,x}[x \in I_{f_{x \to y}}] + \Pr_{f}[y \in f(I_f)] + \Pr_{f,x}[y \in f(I_{f_{x \to y}})].$$

Two of these terms are easy to bound: $\Pr_{f,x}[x \in I_f] \leqslant t/N$ as $|I_f| \leqslant t$, and $\Pr_f[y \in f(I_f)] \leqslant t/M$ by assumption.

For $x \in I_{f_{x\to y}}$, observe that

$$\begin{aligned}
\Pr_{f,x}[x \in I_{f_{x\to y}}] &\leqslant \Pr_{f,x}[y \in f_{x\to y}(I_{f_{x\to y}})] \\
&= \Pr_{g}[y \in g(I_g) \mid y \in \mathrm{Im}(g)] \quad (\text{as } f_{x\to y} \text{ is uniform in } \{g \in \mathcal{F} : y \in \mathrm{Im}(g)\}) \\
&\leqslant \frac{\Pr_g[y \in g(I_g)]}{\Pr_g[y \in \mathrm{Im}(g)]} \\
&\leqslant \frac{t/M}{N/M} = \frac{t}{N}.
\end{aligned}$$

For the last term, we observe that $(f, x) \mapsto (f_{x\to y}, f^{-1}(y))$ is a bijection on

$$\{(f, x) : x \in [N], f \in \mathcal{F}, y \in \mathrm{Im}(f)\},$$

that is its own inverse. Thus,

$$\Pr_{f,x}[y \in f(I_{f_{x\to y}})] \leqslant \Pr_{f,x}[f^{-1}(y) \in I_{f_{x\to y}} \mid y \in \mathrm{Im}(f)] \leqslant \Pr_{g,a}[a \in I_g \mid y \in \mathrm{Im}(g)] \leqslant \frac{t}{N}$$

by fixing g and taking the average.

Combining the four bounds, we get

$$\Pr_{f,x}\left[\Phi^f(C) \neq \Phi^{f_{x\to y}}(C)\right] \leqslant \frac{4t}{N}$$

as desired. □

In the next section, we show that being stable relative to all circuits submitted by $\mathbf{R}$ is sufficient for satisfying the conditions of **Theorem 2.3.4**.

We use the notions of determining sets and bending sets we defined in this section for *encoding*, and we use a more simple argument using only the notion of stability in *coupling*.

3.3 Techniques of Proving Sufficiency

There are two central techniques in the literature when moving from some of the properties discussed in the previous section towards a full-fledged black box separation.

One technique is called *encoding*, where you collect all the invertible functions in a set, and describe an injective function that encodes them using smaller amounts of information/bits, thus showing their rarity. The heart of the proof lies in showing injectivity by comparing different runs of $\mathbf{R}$ on the same input $y^\star$.

This elegant paradigm was used by [GT00, HHRS07, RS08], and generalized in [BH13]. It is also known as *compression* or *reconstruction*. We provide a statement and a proof in **Theorem 3.3.1**, which says that stability and having determining sets (which can already follow by exhaustive search) are enough for a complete black box separation.

We follow up on it in **Corollary 3.3.2**, providing the requirements and parameters needed to use invoke this paradigm.

The other, more modern, technique is called *coupling*, introduced by [BD19], expanded upon in [MF21] and used in [FGH+23]. This technique is slightly more powerful in our context, as it exempts the requirement of determining sets and strengthens the parameters.

The idea is similar in spirit: transition using a hybrid argument to comparing two different runs of $\mathbf{R}$ on the same input y with access to two different functions. We provide a statement and proof in **Theorem 3.3.3**

3.3.1 Encoding

Theorem 3.3.1 (Encoding Theorem)**.** Let Φ be an oracle, and let $\mathbf{R}$ be a deterministic reduction with access to Φ with runtime at most t.

For $f \in \mathcal{F}$ and $y \in [M]$, let $\mathrm{Stab}^{\mathbf{R}}_{f,y}$ denote the event that on input f and y, every circuit C that $\mathbf{R}$ submitted, we have Φ is b-stable relative to f, y and C (where $b \geqslant t$).

Define $f \in \mathcal{F}$ to be *c-invertible* by $\mathbf{R}$ if

$$\Pr_{y \leftarrow \mathrm{Im}(f)} \left[\mathbf{R}^{f,\Phi^f}(y) = f^{-1}(y) \text{ and } \mathrm{Stab}^{\mathbf{R}}_{f,y} \right] > 1/c.$$

Let $\mathcal{F}^c$ collect all c-invertible functions.

If Φ has t-determining sets (relative to every C, f, y) and $r := \left\lfloor \frac{N}{4ctb} \right\rfloor$, then assuming $r \geqslant 1$, we have

$$|\mathcal{F}^c| \leqslant \binom{M}{N} \binom{N}{r}^2 (N-r)!.$$

We partition the long proof of this theorem using horizontal lines and bold face font to denote key observations.

Proof. For $f \in \mathcal{F}^c$, define $K_f := \left\{ y \in \mathrm{Im}(f) \mid \mathbf{R}^{f,\Phi^f}(y) = f^{-1}(y) \text{ and } \mathrm{Stab}^{\Phi}_{f,y} \right\}$ (all y that $\mathbf{R}$ can invert). Since f is c-invertible, then, $|K_f| > N/c$.

We encode $f \in \mathcal{F}^c$ in the following steps:

1. Set $Z_f := \varnothing$ and $U := K_f$.
2. While $U \neq \varnothing$, do the following:
   (a) Let y be the lexicographically least element of U.
   (b) Add y to Z_f.
   (c) Run $x^* \leftarrow \mathbf{R}^{f,\Phi^f}(y)$, saving queries $C_1, C_2, \ldots, C_q$ to Φ and $x_1, x_2, \ldots, x_\ell$ to f.
   (d) Delete from U the following:
      i. y itself

ii. $f(x_1), f(x_2), \ldots, f(x_\ell)$
iii. $f(I_{f,C_1}), f(I_{f,C_2}), \ldots, f(I_{f,C_q})$, where I_{f,C_i} is the determining set of Φ relative to C_i,
iv. $B_1, B_2, \ldots, B_q$ where for any $1 \leqslant i \leqslant q$,

$$B_i := \left\{ f(x) : x \in [N] \text{ satisfies } \Phi^f(C_i) \neq \Phi^{f_{x \to y}}(C_i) \right\}.$$

3. Encode f in the following:
   (a) $J_f := \mathrm{Im}(f)$
   (b) $Y_f \subseteq J_f$ which is the first r elements of Z_f (in lexicographical order)
   (c) $X_f := f^{-1}(Y_f)$
   (d) f restricted on $[N] \setminus X_f$ (as a set of pairs).

Observation 1: (Counting) First, note that we must have $|Z_f| \geqslant r$ so the definition of Y_f is valid. To see this, we note that with each element added to Z_f, we remove from U at most

1	(for y)
$+\, t$	(for the $\ell \leqslant t$ queries made to f)
$+\, t^2$	(for the images of the $q \leqslant t$ determining sets, each of which has size $\leqslant t$)
$+\, bt$	(for each B_i, of which there are $\leqslant t$ and the sizes, by b-stability, are $\leqslant b$)

which is at most $4tb$. Since $|K_f| > N/c$, then $|Z_f| \geqslant \dfrac{N}{4ctb} \geqslant r$.

Observation 2: (Counting) Note that there are

1. $\binom{M}{N}$ choices for J_f,
2. $\binom{N}{r}$ choices for Y_f,
3. $\binom{N}{r}$ choices for X_f, and
4. $(N-r)!$ possible bijections $[N] \setminus X_f \to J_f \setminus Y_f$.

Thus, the image of such encoding has size $\leqslant \binom{M}{N}\binom{N}{r}^2 (N-r)!$. It remains to show that this encoding is injective.

Suppose $f, g \in \mathcal{F}^c$ have the same encoding, so $J := J_f = J_g$, $Y := Y_f = Y_g$, $X := X_f = X_g$, f and g agree on $[N] \setminus X$. We want to argue that $f = g$.

Suppose for the contrary that $f \neq g$. This implies that there exists some $y \in Y$ where $f^{-1}(y) \neq g^{-1}(y)$ (we must have $y \in Y$ since f and g agree on $[N] \setminus X$). Let $y^\star$ be the lexicographically least such y.

Since $y^\star \in Y \subseteq Z_f \subseteq K_f$, then $\mathbf{R}^{f,\Phi^f}(y^\star) = f^{-1}(y^\star)$ successfully and similarly $\mathbf{R}^{g,\Phi^g}(y^\star) = g^{-1}(y^\star)$ successfully.

This implies that $\mathbf{R}^{f,\Phi^f}(y^\star) \neq \mathbf{R}^{g,\Phi^g}(y^\star)$, so the two runs are different with respect to the same input $y^\star$, implying two cases:

1. some query $x_i \in [N]$ submitted towards f or g has $f(x_i) \neq g(x_i)$
2. some query C submitted towards that had $\Phi^f(C) \neq \Phi^g(C)$

Let's take the first query where the two runs differ, and observe that this query must be the same since the prior transcripts agree and $\mathbf{R}$ is assumed to be deterministic.

We cover the first case, and the rest of the proof will be dedicated to the second case.

Suppose the first time the runs differ, some query x^* is submitted where $f(x^*) \neq g(x^*)$. Since f and g agree on $[N] \setminus X$, we must have $x^* \in X$.

Thus, x^* is selected by the encoding. Observe that $f(x^*) \geqslant y^\star$ and $g(x^*) \geqslant y^\star$ lexicographically by minimality of $y^\star$.

Furthermore, we can't have $f(x^*) > y^\star$ since $f(x^*)$ was deleted in the encoding while x^* is selected in the encoding, and similarly $g(x^*) > y^\star$ is impossible.

This implies $f(x^*) = y^\star = g(x^*)$ which contradicts the choice of x^*.

In the second case, the first query that differed in both runs on input $y^\star$ is some circuit C_i (where $1 \leqslant i \leqslant q$).

By **Lemma 3.2.2**, we can't have f and g agree on $I_{f,C_i} \cup I_{g,C_i}$. Thus, there exists some $\alpha \in I_{f,C_i} \cup I_{g,C_i}$ where $f(\alpha) \neq g(\alpha)$.

So, to recap, we have

1. $y^\star \in Y$ is the minimal such that $f^{-1}(y^\star) \neq g^{-1}(y^\star)$ or $\mathbf{R}^{f,\Phi^f}(y^\star) \neq \mathbf{R}^{g,\Phi^g}(y^\star)$,
2. C_i as the first query that they differ, making $\Phi^f(C_i) \neq \Phi^g(C_i)$, and
3. $\alpha \in I_{f,C_i} \cup I_{g,C_i}$ where $f(\alpha) \neq g(\alpha)$.

Observation 3: We have $f(\alpha) \geqslant y^\star$ and $g(\alpha) \geqslant y^\star$ (lexicographically).

To see that, suppose $f(\alpha) < y^\star$. Since $f(\alpha) \neq g(\alpha)$, we must have $\alpha \in X$, and so $f(\alpha)$ was selected in the encoding of f and before $y^\star$, but this contradicts the minimality of $y^\star$.

The argument for $g(\alpha)$ is identical.

Observation 4: If $\alpha \in I_{f,C_i}$, then $f(\alpha) = y^\star$, and if $\alpha \in I_{g,C_i}$, then $g(\alpha) = y^\star$.

We prove the case when $\alpha \in I_{f,C_i}$ and the other case follows by symmetry.

Suppose for the contrary that $f(\alpha) \neq y^\star$. The previous observation implies $f(\alpha) > y^\star$ lexicographically.

Again, since $f(\alpha) \neq g(\alpha)$, then, $\alpha \in X$ and so $f(\alpha)$ was selected in the encoding of f, and *after* $y^\star$.

But when $y^\star$ was selected, $f(I_{f,C_i})$ was deleted from U, and so was $f(\alpha) > y^\star$, and so it's impossible for it to be selected.

Now, since $\Phi^f(C_i) \neq \Phi^g(C_i)$, $f\mid_{I_{f,C_i}}$ breaks g or $g\mid_{I_{g,C_i}}$ breaks f. Suppose the former without loss of generality.

Let $T := f\mid_{I_{f,C_i}}$, and so we get $\Phi^{g_T}(C_i) \neq \Phi^g(C_i)$.

Observation 5: $g_T = g_{a\to y^\star}$ for some unique $a \in I_{f,C_i}$ where $g(a) > f(a) = y^\star$

Suppose there are $x, x' \in I_{f,C_i}$ where $f(x) \neq g(x)$ and $f(x') \neq g(x')$. By the previous observation, we have $f(x) = y^\star = f(x')$, and so injectivity of f implies $x = x'$.

And since clearly $g_T \neq g$, there exists a unique $a \in I_{f,C_i}$ where $f(a) \neq g(a)$, and $f(x) = T(x) = g(x)$ for any $x \in I_{f,C_i}$ where $x \neq a$. Then, forcing $x \to T(x)$ in g for any $x \neq a$ doesn't have any effect.

Since obtaining g_T from g is independent of the order of T, we can let the pair $(a, f(a))$ be the last one, making $g_T = g_{a\to y^\star}$ (since $f(a) = y^\star$ by the previous observation).

The fact that $g(a) > f(a)$ comes from **Observation 3**.

Thus, we get $\Phi^{g_{a\to y^\star}}(C_i) \neq \Phi^g(C_i)$. Since $g(a) > y^\star$ and $y^\star \in Y$ and $a \in X$ (since $g(a) \neq f(a)$), then $g(a)$ was selected by the encoding of g.

But since $\Phi^{g_{a\to y^\star}}(C_i) \neq \Phi^g(C_i)$, then $g(a) \in B_i$, which was removed when selecting $y^\star < g(a)$ in the encoding of g. This is a contradiction.

This finally implies that the primary assumption that $f \neq g$ is incorrect and so $f = g$ and the encoding is injective. □

The following corollary explicitly spells out a sufficient requirement from an oracle to satisfy the conditions of **Theorem 2.3.4**.

Corollary 3.3.2. Let Φ be an oracle with t-determining sets, and let $b \leqslant \ell t^\ell N^\varepsilon$ for some constants $0 < \varepsilon < \frac{1}{2}$ and $\ell \in \mathbb{N}^+$.

For a PPT reduction $\mathbf{R}$ of runtime at most t, $f \in \mathcal{F}$ and $y \in [M]$, let $\mathrm{Stab}^{\mathbf{R}}_{f,y}$ denote the event that on input f and y, every circuit C that $\mathbf{R}$ submitted, we have Φ is b-stable relative to f, y and C.

Suppose that for every PPT $\mathbf{R}$, there exists a negligible function λ where

$$\mathop{\mathbb{E}}_{f\leftarrow\mathcal{F}}\left[\Pr_{y\leftarrow \mathrm{Im}(f)}\left[\neg \mathrm{Stab}^{\mathbf{R}}_{f,y}\right]\right] \leqslant \lambda(n).$$

Then, **Conjecture 2.3.2** holds.

Proof. We show that the conditions of **Theorem 2.3.4** hold.

We fix the randomness of $\mathbf{R}$ and take the average in the end. Thus, we assume without loss of generality that $\mathbf{R}$ is deterministic.

For simplicity of notation, let $\text{Invert}^{\mathbf{R}}_{f,y}$ be the event that $\mathbf{R}^{f,\Phi^f}(y) = f^{-1}(y)$. Then,

$$\begin{aligned}\Pr_{\substack{f\leftarrow\mathcal{F},\\ y\leftarrow\text{Im}(f)}}\left[\text{Invert}^{\mathbf{R}}_{f,y}\right] &= \Pr_{\substack{f\leftarrow\mathcal{F},\\ y\leftarrow\text{Im}(f)}}\left[\text{Invert}^{\mathbf{R}}_{f,y} \text{ and } \text{Stab}^{\mathbf{R}}_{f,y}\right] + \Pr_{\substack{f\leftarrow\mathcal{F},\\ y\leftarrow\text{Im}(f)}}\left[\text{Invert}^{\mathbf{R}}_{f,y} \text{ and } \neg\text{Stab}^{\mathbf{R}}_{f,y}\right] \\ &\leqslant \Pr_{\substack{f\leftarrow\mathcal{F},\\ y\leftarrow\text{Im}(f)}}\left[\text{Invert}^{\mathbf{R}}_{f,y} \text{ and } \text{Stab}^{\mathbf{R}}_{f,y}\right] + \lambda(n)\end{aligned}$$

by assumption. We bound the remaining term using the Encoding Theorem as follows: set $c = N^{\frac{1}{2}-\varepsilon}/40\ell t^{\ell+1}$, and observe that $r = \left\lfloor \frac{N}{4ctb} \right\rfloor \geqslant \left\lfloor 10\sqrt{N} \right\rfloor \geqslant 5\sqrt{N}$.

So, if we let $\gamma(f) := \Pr_{y\in\text{Im}(f)}\left[\text{Invert}^{\mathbf{R}}_{f,y} \text{ and } \text{Stab}^{\mathbf{R}}_{f,y}\right]$, then by the Encoding Theorem (**Theorem 3.3.1**)

$$\begin{aligned}\mathop{\mathbb{E}}_{f\leftarrow\mathcal{F}}[\gamma(f)] &= \mathop{\mathbb{E}}_{f\leftarrow\mathcal{F}}[\gamma(f) \mid f\in\mathcal{F}^c] \Pr_{f\leftarrow\mathcal{F}}[f\in\mathcal{F}^c] + \mathop{\mathbb{E}}_{f\leftarrow\mathcal{F}}[\gamma(f) \mid f\notin\mathcal{F}^c] \Pr_{f\leftarrow\mathcal{F}}[f\notin\mathcal{F}^c] \\ &\leqslant \Pr_{f\leftarrow\mathcal{F}}[f\in\mathcal{F}^c] + \mathop{\mathbb{E}}_{f\leftarrow\mathcal{F}}[\gamma(f) \mid f\notin\mathcal{F}^c] \\ &\leqslant \frac{|\mathcal{F}^c|}{|\mathcal{F}|} + \frac{1}{c}\end{aligned}$$

and for the choice of c, $1/c$ is clearly negligible. It remains to show that the expression

$$\frac{\binom{M}{N}\binom{N}{r}^2(N-r)!}{M!/(M-N)!} = \binom{N}{r}/r!$$

is negligible for $r \geqslant 5\sqrt{N}$ which is straightforward computation and application of the Stirling approximation/bound. □

The other technique that can be used to directly show a version of **Corollary 3.3.2** but with stronger parameters and exempting the requirement for Φ to be exhaustive search is the *coupling* technique.

3.3.2 Coupling

Theorem 3.3.3 (Hybrid Coupling Argument)**.** Let Φ be an oracle, and let $b \leqslant \ell t^\ell N^{1-\varepsilon}$ for some constants $0 < \varepsilon < 1$ and $\ell \in \mathbb{N}^+$.

For a PPT reduction $\mathbf{R}$ of runtime at most t, $f \in \mathcal{F}$ and $y \in [M]$, let $\text{Stab}^{\mathbf{R}}_{f,y}$ denote the event that on input f and y, every circuit C that $\mathbf{R}$ submitted, we have Φ is b-stable relative to f, y and C.

Suppose that for some negligible function λ we have

$$\mathop{\mathbb{E}}_{f\leftarrow\mathcal{F}}\left[\Pr_{y\leftarrow[M]\setminus\text{Im}(f)}\left[\neg\text{Stab}^{\mathbf{R}}_{f,y}\right]\right] \leqslant \lambda(n).$$

Then, **Conjecture 2.3.2** holds.

Proof. For simplicity, we assume that $\mathbf{R}$ only makes queries to Φ as queries to f can be simulated by queries to Φ (so we replace $\mathbf{R}^{f,\Phi^f}$ by $\mathbf{R}^{\Phi^f}$).

Furthermore, like before, we ignore the internal randomness of $\mathbf{R}$ as we can fix it and in the end average over it.

We define the following hybrid games:

$$\begin{aligned}
H_1 &:= \left\{(f,x) : \mathbf{R}^{\Phi^f}(f(x)) = x, f \in \mathcal{F}, x \in [N]\right\} \\
H_2 &:= \left\{(f,x,y) : \mathbf{R}^{\Phi^{f_{x\to y}}}(y) = x, f \in \mathcal{F}, x \in [N], y \in [M] \setminus \mathrm{Im}(f)\right\} \\
H_3 &:= \left\{(f,x,y) : \mathbf{R}^{\Phi^f}(y) = x, f \in \mathcal{F}, x \in [N], y \in [M] \setminus \mathrm{Im}(f)\right\}
\end{aligned}$$

It is straightforward to show that $\Pr[H_1] = \Pr[H_2]$ and that $\Pr[H_3] = 1/N$ (over $f \leftarrow \mathcal{F}$, $x \leftarrow [N]$, and $y \leftarrow [M] \setminus \mathrm{Im}(f)$). We turn to bound $|\Pr[H_2] - \Pr[H_3]|$.

We fix $y \in [M]$ and we average over its values at the end. We compare the executions $\mathbf{R}^{\Phi^f}(y)$ and $\mathbf{R}^{\Phi^{f_{x\to y}}}(y)$, and we may assume that both executions make exactly $q \leqslant t$ queries (otherwise we can add dummy queries).

To that end, define $Q_i(g)$ and $A_i(g)$ denote the ith query made and answer received in the execution given access to Φ^g (for $g \in \mathcal{F}$). Define

$$\begin{aligned}
\mathcal{Q}_i &:= \{(f,x,y) : f \in \mathcal{F}, x \in [N], y \notin \mathrm{Im}(f), Q_i(f) = Q_i(f_{x\to y})\}, \\
\mathcal{A}_i &:= \{(f,x,y) : f \in \mathcal{F}, x \in [N], y \notin \mathrm{Im}(f), A_i(f) = A_i(f_{x\to y})\}, \\
\mathcal{L}_i &:= \mathcal{Q}_i \cap \mathcal{A}_i.
\end{aligned}$$

Let $\mathcal{G} = \bigcap_{1\leqslant i\leqslant q} \mathcal{L}_i$. Observe that over $x \leftarrow [N]$ and $f \leftarrow \{f \in \mathcal{F} : y \notin \mathrm{Im}(f)\}$, we have

$$\begin{aligned}
\Pr[H_2] &= \Pr[H_2 \cap \mathcal{G}] + \Pr[H_2 \cap \neg\mathcal{G}] \\
&= \Pr[H_3 \cap \mathcal{G}] + \Pr[H_2 \cap \neg\mathcal{G}] \qquad (\mathcal{G} \text{ implies both executions become identical}) \\
&\leqslant \Pr[H_3] + \Pr[\neg\mathcal{G}].
\end{aligned}$$

Similarly, we can show that $\Pr[H_3] \leqslant \Pr[H_2] + \Pr[\neg\mathcal{G}]$. This shows that $|\Pr[H_2] - \Pr[H_3]| \leqslant \Pr[\neg\mathcal{G}]$.

Now,

$$\begin{aligned}
\Pr[\neg\mathcal{G}] &= \Pr\left[\neg\mathcal{G} \text{ and } \mathrm{Stab}^{\mathbf{R}}_{f,y}\right] + \Pr\left[\neg\mathcal{G} \text{ and } \neg\mathrm{Stab}^{\mathbf{R}}_{f,y}\right] \\
&\leqslant \Pr\left[\neg\mathcal{G} \text{ and } \mathrm{Stab}^{\mathbf{R}}_{f,y}\right] + \Pr\left[\neg\mathrm{Stab}^{\mathbf{R}}_{f,y}\right] \\
&\leqslant \Pr\left[\neg\mathcal{G} \text{ and } \mathrm{Stab}^{\mathbf{R}}_{f,y}\right] + \lambda(n)
\end{aligned}$$

as given, by averaging over $y \in [M]$. Thus, it remains to bound $\Pr\left[\neg\mathcal{G} \text{ and } \mathrm{Stab}^{\mathbf{R}}_{f,y}\right]$.

Since the executions are given the same inputs, the first query is the same, and as long as they receive the same answers, they get identical executions, and they diverge when they

receive different answers. Thus,

$$\begin{aligned}\Pr\left[\neg\mathcal{G} \text{ and } \mathrm{Stab}^{\mathbf{R}}_{f,y}\right] &= \sum_{i=1}^{q} \Pr\left[\mathcal{L}_1 \cap \mathcal{L}_2 \cap \cdots \cap \mathcal{L}_{i-1} \cap \neg\mathcal{L}_i \text{ and } \mathrm{Stab}^{\mathbf{R}}_{f,y}\right] \\ &= \sum_{i=1}^{q} \Pr\left[\mathcal{L}_1 \cap \mathcal{L}_2 \cap \cdots \cap \mathcal{L}_{i-1} \cap \mathcal{Q}_i \cap \neg\mathcal{A}_i \text{ and } \mathrm{Stab}^{\mathbf{R}}_{f,y}\right]\end{aligned}$$

since the query answers agree prior to the ith query, the ith query must be identical. Now, fix f, so the probability is only taken over $x \leftarrow [N]$.

Under fixing f and y, $\mathbf{R}^{\Phi^f}(y)$ asks a unique circuit $Q_i(f) = C$ in the ith query. Furthermore, conditioning on $\cap_{j<i}\mathcal{L}_j \cap \mathcal{Q}_i$, $\mathbf{R}^{\Phi^{f_{x\to y}}}(y)$ asks the same circuit $Q_i(f_{x\to y}) = C$.

Yet, $\neg\mathcal{A}_i$ implies $A_i(f) \neq A_i(f_{x\to y})$, or in other words, $\Phi^f(C) \neq \Phi^{f_{x\to y}}(C)$ for the selected C. In other words,

$$\Pr\left[\mathcal{L}_1 \cap \mathcal{L}_2 \cap \cdots \cap \mathcal{L}_{i-1} \cap \mathcal{Q}_i \cap \neg\mathcal{A}_i \text{ and } \mathrm{Stab}^{\mathbf{R}}_{f,y}\right] \leqslant \Pr\left[\Phi^f(C) \neq \Phi^{f_{x\to y}}(C) \text{ and } \mathrm{Stab}^{\mathbf{R}}_{f,y}\right]$$

and since f and y are fixed, and C is a circuit submitted by $\mathbf{R}$ in the run $\mathbf{R}^{\Phi^f}(y)$, the event $\mathrm{Stab}^{\mathbf{R}}_{f,y}$ (being independent from x) implies this probability is at most b/N. In other words,

$$\Pr\left[\neg\mathcal{G} \text{ and } \mathrm{Stab}^{\mathbf{R}}_{f,y}\right] \leqslant \sum_{i=1}^{q} \frac{b}{N} \leqslant \ell t^{\ell+1} N^{-\varepsilon}$$

which is negligible, as desired. □

This technique is called *coupling* because of the subtle step where we show $|\Pr[H_2] - \Pr[H_3]| \leqslant \Pr[\neg\mathcal{G}]$. $\mathcal{G}$ is an event on a joint or *coupled* distribution, and then we observed that in this coupled distribution we get $\Pr[H_2 \cap \mathcal{G}] = \Pr[H_3 \cap \mathcal{G}]$ which is the basis towards concluding the aforementioned inequality.

We note from the previous results that in the case of one query to Φ, it would suffice to bound, as conjectured in [FGH+23], that $\neg\mathrm{Stab}^{\mathbf{R}}_{f,y}$ rarely happens conditioned on prior queries to f.

In other words, we need that an oracle Φ such that for every partial assignment $T \in \mathcal{F}^{<}$ where $|T| \leqslant t$, every $y \in [M]$ where $y \notin \mathrm{Im}(T)$, and every satisfiable circuit $C^{(\cdot)} \in \mathcal{C}_n$,

$$\Pr_{\substack{f\leftarrow\mathcal{F}\\ x\leftarrow[N]}}\left[\Phi^f(C) \neq \Phi^{f_{x\to y}}(C) \mid T \subseteq f, y \notin \mathrm{Im}(f)\right] \leqslant t^{O(1)}/N^{\Omega(1)}.$$

Chapter 4

Main Contributions

The aim of this chapter is to extensively study the ways to attack **Conjecture 2.3.2**, provide partial results, and elaborate on the barriers and challenges towards extending them.

The chapter is structured as follows:

1. We start in **section 4.1** by discussing challenges towards constructing a stable oracle Φ, elaborate on some insights into the structure of satisfiable circuits inspired by [FGH+23], and providing an example of a class of circuits that can be easily made adaptively stable called *simulatable circuits*.
2. In **section 4.2**, we build on the work of the previous section by showing impossibility against reductions that adaptively submit circuits with only one oracle gate.

   It expands on the beautiful combinatorial structure implicit in the problem and the properties of this structure, revealing the challenges facing the generalization beyond one oracle gate per query.
3. **Section 4.3** explicitly extends the result of [FGH+23] to one round of non-adaptive queries.

4.1 Towards Achieving Stability

This section discusses possible approaches and challenges to construct an oracle that satisfies the conditions in **Theorem 2.3.4**. It is divided into three subsections

1. We start in **subsection 4.1.1** by discussing what it means for Φ to satisfy the premise of **Theorem 3.3.3** (i.e., to be stable relative to every circuit submitted by $\mathbf{R}$) and challenges facing the existence of such an oracle.
2. **Subsection 4.1.2** gives more understanding towards the structure satisfiable circuits induce and the kind of information that a satisfying assignment of a circuit may leak.
3. **Subsection 4.1.3** discusses a special kind of circuits that we call *simulatable* where it's easy to choose a satisfying assignment that leaks little information.

4.1.1 Motivation and Challenges

Observe that $\mathbf{R}^{(\cdot),\Phi^{(\cdot)}}(y)$ on a fixed $y \in [M]$ provides the same first query. Let's ignore the queries it makes to the function for moment. If it queries a circuit C_1, then this circuit becomes a random variable defined based on its input y.

Once it receives an output $\Phi^f(C_1)$, this output is now based on f, and the second query C_2 becomes a random variable that relies on both y and f (to be precise, it relies on y and the answer of $\Phi^f(C_1)$).

So, if one asks "why is an almost t-stable oracle not sufficient?" It is sufficient to provide stability of the first query C_1 that doesn't rely on f, which is the result of [FGH+23].

Once it receives an answer to the first query, the second query, that we denote $C_2[f]$, relies on f, so one needs to bound

$$\Pr_{f \leftarrow \mathcal{F}}[\Phi \text{ is stable relative to } C_2[f] \text{ and } y \text{ for all } y \in [M]]$$

which we don't get from Φ being almost t-stable.

Let's call Φ stable relative to $C \in \mathcal{C}$ and $f \in \mathcal{F}$ if it's stable relative to C, f, y for every $y \in [M]$ (i.e., if we don't mention y, then we quantify over it).

Another natural question is: "can we get an oracle that is t-stable relative to every $C \in \mathcal{C}$ and $f \in \mathcal{F}$?"

We show in the next theorem that the answer is no.

Theorem 4.1.1 (Unconditional t-stability is impossible)**.** Consider the satisfiable circuit $C^{(\cdot)} : \{0,1\}^n \to \{0,1\}$ defined as

$$C^f(x) := \mathbf{1}[f(x) \leqslant M - N + 1].$$

Then, there does not exist any oracle Φ that is t-stable relative to C, f, y for every $f \in \mathcal{F}$ and $y \in [M]$ for any $t \in o(N)$.

Proof. Let's denote $A := [M - N + 1]$ and $B := [M] \setminus A$. So A and B partition $[M]$ and $|B| = N - 1$, and $C^{(\cdot)}$ can be redefined as

$$C^f(x) := \mathbf{1}[f(x) \in A].$$

First, observe that this circuit is satisfiable for every $f \in \mathcal{F}$. This is because if $f(x) \in B$ for all $x \in [N]$, pigeonhole would imply that f is not injective, which is a contradiction.

Now, we pick an arbitrary oracle Φ. We want to find some $f \in \mathcal{F}$ and $y \in [M]$ such that Φ is not t-stable relative to C, f, y.

Fix any $y \in A$, and for $a \in [N]$, define $\mathcal{G}_y^a \subseteq \mathcal{F}$ as follows:

$$\mathcal{G}_y^a := \{f \in \mathcal{F} : f(a) = y \text{ and } \mathrm{Im}(f) = \{y\} \cup B\}.$$

In other words, every $f \in \mathcal{G}_y^a$ has $f(a) = y$ and the rest is a bijection from $[N] \setminus \{a\}$ to B.

Observe the following:

1. $\Phi^f(C) = a$ for any $f \in \mathcal{G}_y^a$, since $f(a) \in A$ and no other $x \neq a$ has $f(x) \in A$.
2. For $a, b \in [N]$ where $a \neq b$, if $f \in \mathcal{G}_y^a$, then $f_{b\to y} \in \mathcal{G}_y^b$.

This implies that if we select any $a \in [N]$, any $f \in \mathcal{G}_y^a$,

$$\Pr_{x\leftarrow[N]}\left[\Phi^f(C) \neq \Phi^{f_{x\to y}}(C)\right] = \Pr_{x\leftarrow[N]}[x \neq a] = \frac{N-1}{N}.$$

So choosing any $y \in A$, and choosing any $a \in [N]$ and $f \in \mathcal{G}_y^a$ already gives instability.

Alternative Proof: If we want to disallow $f_{x\to y}$ to be defined when $y \in \mathrm{Im}(f)$ (as in [FGH+23]), we can still obtain a similar construction.

Observe that for any $y, z \in A$ where $y \neq z$ and $a, b \in [N]$ where $a \neq b$ and for any $f \in \mathcal{G}_y^a$, there exists a unique $g \in \mathcal{G}_z^b$ where $f_{b\to z} = g_{a\to y}$.

Thus, $\Phi^{f_{b\to z}}(C)$ is forced to choose either a or b. If it chooses a, then we get $\Phi^{g_{a\to y}}(C) \neq \Phi^g(C)$. If it chooses b, then we get $\Phi^{f_{b\to z}}(C) \neq \Phi^f(C)$. So, there is a symmetric property we wish to exploit.

To that end, define for $y \in A$, $\mathcal{S}_y := \bigcup_{a\in[N]} \mathcal{G}_y^a$, and observe that $a \neq b$ implies $\mathcal{G}_y^a \cap \mathcal{G}_y^b = \varnothing$ (since we can't have $f(a) = f(b) = y$ without $a = b$). Furthermore, if $y, y \in A$ have $y \neq y'$ then $\mathcal{S}_y \cap \mathcal{S}_{y'} = \emptyset$ since every element has either y in the image or y', not both.

Let $\mathcal{S} = \bigcup_{y\in A} \mathcal{S}_y$. Thus, for every $f \in \mathcal{S}$, $f \in \mathcal{S}_y$ for some unique $y \in A$ (we denote it by $\rho(f)$). Furthermore, for $f \in \mathcal{S}_y$, we must have $f \in \mathcal{G}_y^a$ for some unique $a \in [N]$ (we denote it by $\lambda(f)$).

We observe that

$$\mathbb{E}\left[\Pr_{x\leftarrow[N]\setminus\{\lambda(f)\}}\left[\Phi^f(C) \neq \Phi^{f_{x\to y}}(C)\right] : \begin{matrix} f \leftarrow \mathcal{S} \\ y \leftarrow A \setminus \{\rho(f)\}\end{matrix}\right] = \frac{1}{2}$$

by the symmetry concluded above. Thus, there must exist $f \in \mathcal{S}$ and $y \in A \setminus \{\rho(f)\}$ where

$$\Pr_{x\leftarrow[N]\setminus\{\lambda(f)\}}\left[\Phi^f(C) \neq \Phi^{f_{x\to y}}(C)\right] \geqslant \frac{1}{2}$$

which implies

$$\Pr_{x\leftarrow[N]}\left[\Phi^f(C) \neq \Phi^{f_{x\to y}}(C)\right] \geqslant \frac{N-1}{2N} = \frac{1}{2} - \frac{1}{2N}$$

by the law of total probability. □

We note that $|\mathcal{S}| = (M - N + 1) \cdot N!$, meaning that

$$\frac{|\mathcal{S}|}{|\mathcal{F}|} = \frac{(M-N+1)\cdot N!}{M!/(M-N)!} \leqslant M \cdot \binom{M}{N}^{-1} \leqslant \frac{N^N}{M^{N-1}}$$

which is $\leqslant N/2^N$ (very small or negligible) when $M \geqslant 2N$.

So, the fraction of $f \in \mathcal{F}$ that Φ loses stability to (relative to this specific circuit) is very small.

[BH13] stated a version of **Theorem 3.3.1** that assumed that Φ is t-stable relative to *every* $C \in \mathcal{C}$ *and* $f \in \mathcal{F}$. We wanted to establish that in our context, such a Φ cannot exist, even if the fraction of $f \in \mathcal{F}$ is small.

So, the best one can hope for is a kind of "adaptive t-stability" (conditioned on the previous transcript, Φ remains almost t-stable relative to any circuit in $\mathcal{C}$).

Recalling the discussion in **section 3.1**, one way to phrase "adaptive t-stability" is as follows:

Conjecture 4.1.2. Let Φ be an oracle.

Call $\mathcal{E} \subseteq \mathcal{F}$ (q, Φ)-*generatable* if there are $C_1, C_2, \ldots, C_q \in \mathcal{C}$ and $u_1, u_2, \ldots, u_q \in \{0,1\}^m$ where $\mathcal{E} = \bigcap_{1 \leqslant i \leqslant q} \mathcal{U}_{u_i}^{C_i}$.

We call Φ *adaptively t-stable* if for every $C \in \mathcal{C}$, $q < t$, (q, Φ)-generatable $\mathcal{E}$, $y \in [M]$, we have

$$\Pr_{\substack{f \leftarrow \mathcal{F} \\ x \leftarrow [N]}} \left[\Phi^f(C) \neq \Phi^{f_{x \to y}}(C) \mid f \in \mathcal{E}, y \notin \mathrm{Im}(f) \right] \leqslant \frac{t}{N}.$$

Then, there exists an adaptively t-stable oracle for some polynomial $t(n)$.

This conjecture is sufficient for satisfying the premise of **Theorem 3.3.3** since it would suffice to condition on the previous answers of the queries to Φ and bound the probability of instability.

Theorem 4.1.1 showed us that some circuits induce instability to a fraction of injective functions, but what kind of circuits may leak information?

What would be a strategy on how to choose satisfying assignments, or what does the structure of $\mathcal{U}_u^C$ look like? We consider one way to answer these questions in the next subsection.

4.1.2 Decision Tree Viewpoint

Let $C \in \mathcal{C}$. For $u \in \{0,1\}^m$, the run of $C^{(\cdot)}(u)$ can be viewed as a decision tree of depth at most t, where each node has M branches corresponding to the values of the oracle gates (as observed in [FGH+23]).

Each leaf of such decision tree is labelled with 0 or 1 (depending on whether $C^{(\cdot)}(u) = 0/1$) and with a transcript of the query/answer pairs made along the path from the root. We make these notions more precise in the following discussion.

Before we start our discussion, we make two simplifying assumptions about our circuits from here on out:

1. The oracle gates of our circuits are non-adaptive.

2. The queries made to the oracle gates are always distinct.

The first assumption can be safely made as one can reduce an adaptive gate circuit to a non-adaptive gate by guessing the output of the gates in the satisfying assignment and accepting only when the guess is right.

The second assumption can be safely made as one can always check if one query is repeated before submitting the queries and replacing a repeated query with a fresh dummy query and reusing the previously repeated query.

Thus, any $C^{(\cdot)} \in \mathcal{C}$ with t oracle gates can be factorized into two circuits Q, A where

- $Q : \{0,1\}^m \to \{0,1\}^{tn}$ supplies t **distinct** queries and
- $A : \{0,1\}^{m+t(n+k)} \to \{0,1\}$ receives the input together with the t query/answer pairs and produces the output.

Thus, for any $f \in \mathcal{F}$, we have

$$C^f(u) := A(u, Q(u), f(Q(u)))$$

where $f(Q(u))$ is interpreted as:

$$\text{for } \mathbf{q} = (q_1, q_2, \ldots, q_t) \text{ we define } f(\mathbf{q}) := (f(q_1), f(q_2), \ldots, f(q_t)).$$

It may happen that $Q(u) = Q(u')$ and $u \neq u'$, but regardless the image of Q is a sequence of sets $Q_1, Q_2, \ldots, Q_\ell \subseteq [N]$ for some $\ell \leqslant 2^m$ and $|Q_i| = t$ for $i \in [\ell]$.

This interpretation tells us that for a fixed $u \in \{0,1\}^m$, $C^f(u) = 1$ if and only if $f(Q(u))$ lies in some family of images $\mathcal{A}_u$ where $A \in \mathcal{A}_u$ is a subset of $[M]$ of size t.

Thus, every $u \in \{0,1\}^m$ induces a (possibly empty) family of *transcripts*

$$\mathcal{T}_u := \{T \in \mathcal{F}^< : \mathrm{Dom}(T) = Q(u), |T| = t, A(u, Q(u), T(Q(u))) = 1\}.$$

We call them "transcripts" since they record the interaction of C with its oracle.

Now, since for every $f \in \mathcal{F}$, there exists $u \in \{0,1\}^m$ with $C^f(u) = 1$, then for such f and u, $f(Q(u)) \in \mathcal{A}_u$ or $f\,|_{Q(u)} \in \mathcal{T}_u$.

Thus, importantly, $\mathcal{T} := \bigcup_{u \in \{0,1\}^m} \mathcal{T}_u$ has the property that

$$\text{for every } f \in \mathcal{F}, \text{ there exists } T \in \mathcal{T} \text{ with } T \subseteq f$$

by satisfiability.

We call such sets *t-Turán.*[1] We discuss their structure more extensively in **subsection 4.2.1** and **subsection 4.2.4**, but we provide their definition here as well:

[1] Due to "Turán systems" found in the literature ([Pik25]) that have a similar definition.

Definition 4.1.3 (t-Turán Sets)**.** A subset $\mathcal{T} \subseteq \mathcal{F}^{<}$ is called *t-Turán* if the following holds:

1. $|T| = t$ for all $T \in \mathcal{T}$.
2. For every $f \in \mathcal{F}$, there exists $T \in \mathcal{T}$ where $T \subseteq f$. That is,

$$\bigcup_{T \in \mathcal{T}} \text{Cube}(T) = \mathcal{F}.$$

Therefore, for $f \in \mathcal{F}$, choosing u where $C^f(u) = 1$ can be done in two stages:

1. Choose $T \in \mathcal{T}$ where $T \subseteq f$.
2. Choose $u \in \{0,1\}^m$ where $T \in \mathcal{T}_u$.

Furthermore, if you choose the u directly, we know there exists a unique $T = f\,|_{Q(u)}$ that's implicitly chosen by choosing u. In other words, the two stages above are the only way to choose u.

We want to investigate if the choice of u in the second stage leaks any more information more than the choice of T in the first stage.

Let $\Theta^f(C) := f\,|_{Q(u_f)}$ where $u_f = \Phi^f(C)$. Consider the sets

$$\mathcal{U}_C(u,T) := \left\{f \in \mathcal{F} : \Phi^f(C) = u, \Theta^f(C) = T\right\}$$
$$\mathcal{U}_C(T) := \left\{f \in \mathcal{F} : \Theta^f(C) = T\right\}$$

Clearly, for any $u \in \{0,1\}^m$ and $T \in \mathcal{T}$, $\mathcal{U}_C(u,T) \subseteq \mathcal{U}_C(T)$. In particular, $\mathcal{U}_C(T) = \bigcup_u \mathcal{U}_C(u,T)$.

The reduction $\mathbf{R}$ can always manually compute $C^f(u)$ by querying f after receiving $u = \Phi^f(C)$, we may assume $\mathbf{R}$ also receives $\Theta^f(C)$.

In other words, $\mathbf{R}$ knows that $f \in \mathcal{U}_C(u,T) \subseteq \mathcal{U}_C(T)$ for some u, T after querying C. So, regardless of u, $\mathbf{R}$ would know $f \in \mathcal{U}_C(T)$.

Thus, if we construct Φ so that for one T chosen in the first stage, we choose different $u \neq u'$ in the second stage, we get $\mathcal{U}_C(u,T), \mathcal{U}_C(u',T) \subsetneq \mathcal{U}_C(T)$ and learning either of them provides more information than just $f \in \mathcal{U}_C(T)$.

So, it is not optimal to choose different $u \neq u'$ in the second stage, rather **it is optimal to set the following convention about** Φ:

$\Phi^f(C)$ chooses some $T \in \mathcal{T}$ where $T \subseteq f$, then always chooses the lexicographically least $u \in \{0,1\}^m$ where $T \in \mathcal{T}_u$.

The choice of lexicographically least is any fixed arbitrary (and irrelevant) choice. But this helps us as:

1. if $u \in \{0,1\}^m$ (where $T \in \mathcal{T}_u$) is such fixed choice, we get $\mathcal{U}_C(u,T) = \mathcal{U}_C(T)$, providing $\mathbf{R}$ with the least amount of information.

2. choosing $\Phi^f(C)$ becomes equivalent to choosing $\Theta^f(C)$.

Thus, we often will interchange the choice of $T \in \mathcal{T}$ with the choice of the satisfying assignment u.

We can assume that $\Phi^f(C)$ returns a pair (u, T) instead of only u (which only gives the reduction more information). Since the satisfying assignment associated with each transcript T is fixed by our convention, the pair (u, T) is uniquely determined by T as well.

From this point onward, we slightly abuse notation and, after fixing $f \in \mathcal{F}$, treat the selected transcript T and the corresponding satisfying assignment u as being in one-to-one correspondence. Accordingly, we freely interchange the two and also often regard $\Phi^{(\cdot)}(C) : \mathcal{F} \to \mathcal{T}$.

Now, one method to define such Φ is by taking an ordering $T_1, T_2, \ldots, T_{|\mathcal{T}|}$ of $\mathcal{T}$ and letting $f = T_j$ where $j = \min\{i : T_i \subseteq f\}$.

In other words, $\Phi^f(C)$ chooses the first one in the ordering consistent with f. This was referred to as a *decision list* in [FGH+23]. We will refer to such oracle as an *exhaustive search* oracle as in [BH13].

In fact, defining Φ this way makes it have t-determining sets (relative to C):

> **Theorem 4.1.4** (Exhaustive search oracles have t-determining sets)**.** Let Φ be an exhaustive search oracle and $C \in \mathcal{C}$ be a circuit with t oracle gates.
>
> Let T_f be the transcript chosen by $\Phi^f(C)$.
>
> Then, the collection $\{I_f\}_{f \in \mathcal{F}}$ where $I_f := \mathrm{Dom}(T_f)$ are t-determining relative to Φ and C.

Proof. The fact that $|I_f| \leqslant t$ is easy to see.

For $f \in \mathcal{F}$, denote $\sigma(f) := i$ where $\Phi^f(C)$ chooses T_i.

The main observation is that if for $f, g \in \mathcal{F}$, $T_f \subseteq g$, then we must have $\sigma(g) \leqslant \sigma(f)$ (as otherwise, $\Phi^g(C)$ would choose T_f first).

Now, fix $f, g \in \mathcal{F}$ and suppose $\Phi^f(C) \neq \Phi^g(C)$. This implies that either $\sigma(f) < \sigma(g)$ or $\sigma(f) > \sigma(g)$.

Assume WLOG that $\sigma(f) < \sigma(g)$.

Let $T := f\,|_{I_f}$, and observe that $T = T_f$.

If T does not break g, then $\Phi^{g_T}(C) = \Phi^g(C)$. Furthermore, since $T_f = T \subseteq g_T$, then $\sigma(g) = \sigma(g_T) \leqslant \sigma(f)$, which is a contradiction.

Thus, $f\,|_{I_f}$ breaks g as desired. □

Now, if Φ is exhaustive search, and Φ chooses T_i, this provides $\mathbf{R}$ with the following information about the hidden f:

1. $f \in \mathrm{Cube}(T_i)$,
2. $f \notin \mathrm{Cube}(T_j)$ for all $j < i$.

It turns out that this gives a useful complete characterization of the information provided by Φ.

Lemma 4.1.5 (Information leaked by exhaustive search oracles)**.** Let Φ be an exhaustive search oracle and let $C^{(\cdot)} : \{0,1\}^m \to \{0,1\}$ be a satisfiable circuit.

Suppose C induces a t-Turán set $\mathcal{T}$, and $\Phi^{(\cdot)}(C)$ is defined by an ordering $T_1, T_2, \ldots, T_\ell$ of $\mathcal{T}$.

For $i \in [\ell]$, define $\mathcal{U}_i := \left\{ f \in \mathcal{F} : \Phi^f(C) = T_i \right\}$. Then, for every $i \in [\ell]$,

$$\mathcal{U}_i = \mathrm{Cube}(T_i) \cap \bigcap_{1 \leqslant j < i} (\mathcal{F} \setminus \mathrm{Cube}(T_j))$$

We call $T_1, T_2, \ldots, T_i$ a *leakage prefix*.

Proof. For the forward inclusion, let $f \in \mathcal{U}_i$. Since $\Phi^f(C) = T_i$, then it must be that $f \in \mathrm{Cube}(T_i)$. Furthermore, $f \notin \mathrm{Cube}(T_j)$ for all $j < i$, since $i = \min \{k : f \in \mathrm{Cube}(T_k)\}$.

For the backward inclusion, suppose $f \in \mathrm{Cube}(T_i)$ and $f \notin \mathrm{Cube}(T_j)$ for all $j < i$. This would imply Φ must choose T_i. □

4.1.3 Simulatable Circuits

Constructing Φ is the same as constructing $\Phi^{(\cdot)}(C)$ for every $C \in \mathcal{C}$. It sometimes can be useful to classify $\mathcal{C}$ into subclasses of circuits and define $\Phi^{(\cdot)}$ on every subclass individually.

We saw in **subsection 4.1.1** that some circuits challenge stability. But it turns out that a class of circuits that is easy to define a non-leaking Φ onto is a special class we call *simulatable circuits*.

Definition 4.1.6 (Simulatable Circuits)**.** We call $C \in \mathcal{C}$ (s, t, γ)-*simulatable* if there exists an algorithm $\mathbf{S}^{(\cdot)}$ and a function $\Psi_C : \mathcal{F} \to \{0,1\}^m$ such that the following holds:

1. For every $f \in \mathcal{F}$, $\mathbf{S}^f$ queries f at most s times (on any input).
2. For every $f \in \mathcal{F}$, $\Psi_C(f)$ is a satisfying assignment of C^f (i.e., $C^f(\Psi_C(f)) = 1$).
3. For every $J \in \mathcal{F}^<$ with $|J| \leqslant t$ and $y \in [M] \setminus \mathrm{Im}(J)$, we have

$$\Pr_{f \leftarrow \mathcal{F}} \left[\mathbf{S}^f(J, C) = \Psi_C(f) \mid J \subseteq f, y \in \mathrm{Im}(f) \right] \geqslant 1 - \gamma.$$

We usually say simulatable and eliminate s, t, and γ when we mean that s and t are polynomial and γ is negligible.

The idea behind this definition is that we can choose $\Phi^f(C) = \Psi_C(f)$ and then any inverter $\mathbf{R}$ can find the answer of $\Phi^f(C)$ on a simulatable circuit C just by running $\mathbf{S}^f(J, C)$ where J is the transcript $\mathbf{R}$ has prior to querying C. We formalize these notions in **Theorem 4.1.7**.

An example of a simulatable circuit is circuit that only queries f: for $x \in \{0,1\}^n$, define $C_x^{(\cdot)} : \{0,1\}^k \to \{0,1\}$ as

$$C_x^f(z) := \mathbf{1}[f(x) = z].$$

Clearly, a satisfying assignment to $C_x^f(z)$ is simply $f(x)$. To simulate $C_x^{(\cdot)}$, $\mathbf{S}^f$ only needs to query f at x and succeed with probability 1.

The circuit provided in **Theorem 4.1.1** is also simulatable if $M \geqslant N^2$ (i.e., $k \geqslant 2n$ or f outputs at least $2n$ bits):

1. We can let $\Psi_C(f) := \min\{x \in [N] : f(x) \leqslant M - N + 1\}$.
2. $\mathbf{S}$ queries f at the minimum s entries L in $[N]$ that are outside of $\mathrm{Dom}(J)$.
3. The combined definition of f on L and J gives us the answer of $\Psi_C(f)$ with high probability, since if it lies in J then $\mathbf{S}$ succeeds with probability 1, and if not, then the outputs of L intersect with $[M - N + 1]$ with high probability.

This tells us that some unstable circuits are simulatable.

The next theorem is central to the importance of simulatable circuits. It shows us that if $\mathbf{R}$ only submits simulatable circuits, then $\mathbf{R}$ cannot invert f. We provide the proof in complete detail as the theorem is central for the result in **section 4.2**.

Theorem 4.1.7 (Impossibility Against Simulatable Circuits)**.** Fix some (appropriate) parameters s, t, γ.

Let Φ be such that for any $(s, 2ts, \gamma)$-simulatable circuit C, and $f \in \mathcal{F}$ $\Phi^f(C) := \Psi_C(f)$, and otherwise Φ outputs an arbitrary satisfying assignment of C^f.

Let $\mathbf{R}$ be a PPT reduction with runtime at most t. Define

$$\mathrm{Sim}^{\mathbf{R}} := \left\{(f, y) : f \in \mathcal{F}, y \in \mathrm{Im}(f), \mathbf{R}^{\Phi^f}(y) \text{ only makes } (s, 2ts, \gamma)\text{-simulatable queries}\right\}.$$

Then,

$$\Pr_{\substack{f \leftarrow \mathcal{F} \\ y \leftarrow \mathrm{Im}(f)}}\left[\mathbf{R}^{f,\Phi^f}(y) = f^{-1}(y) \text{ and } (f, y) \in \mathrm{Sim}^{\mathbf{R}}\right] \leqslant t\gamma(n) + \frac{4t^2 s}{N}$$

Proof. As usual, we assume $\mathbf{R}$ is deterministic by fixing the randomness and taking the average over it.

For simplicity of notation, let's denote

$$\mathrm{Inv}^{\mathbf{R}} := \left\{(f, y) : f \in \mathcal{F}, y \in [M], \mathbf{R}^{f,\Phi^f}(y) = f^{-1}(y)\right\}.$$

We want to show that

$$\Pr_{\substack{f\leftarrow\mathcal{F}\\ y\leftarrow\mathrm{Im}(f)}}\left[(f,y)\in \mathrm{Sim}^{\mathbf{R}}\cap \mathrm{Inv}^{\mathbf{R}}\right]\leqslant t\gamma+4t^2s/N.$$

Suppose that $\mathbf{R}$ makes exactly $q\leqslant t$ queries to Φ and $\leqslant t$ queries to f (regardless of the choice of f,y, otherwise we pad useless queries).

The strategy is a hybrid argument, where we construct q algorithms $\mathbf{R}_1,\mathbf{R}_2,\ldots,\mathbf{R}_q$, where $\mathbf{R}_i^{f,\Phi^f}$ makes

1. $q-i$ queries to Φ^f and
2. $\leqslant t+is$ queries to f.

At the end, $\mathbf{R}_q$ makes 0 queries to Φ^f and $t+qs\leqslant 2ts$ queries to f, and thus

$$\Pr_{f,y}\left[\mathbf{R}_q^f(y)=f^{-1}(y)\right]\leqslant\frac{2ts}{N}$$

which can be done naively or with encoding/coupling (we eliminated the Φ^f superscript to emphasize the absence of queries made to it).

Thus, setting $\mathbf{R}_0=\mathbf{R}$, we aim to show that for all i with $0\leqslant i<q$,

$$\Pr_{f,y}\left[(f,y)\in \mathrm{Sim}^{\mathbf{R}_i}\cap \mathrm{Inv}^{\mathbf{R}_i}\right]-\Pr_{f,y}\left[(f,y)\in \mathrm{Sim}^{\mathbf{R}_{i+1}}\cap \mathrm{Inv}^{\mathbf{R}_{i+1}}\right]\leqslant\gamma+\frac{2ts}{N}.$$

If so, we directly get our $\leqslant t\gamma+4t^2s/N$ bound.

The idea is to construct $\mathbf{R}_{i+1}$ based on $\mathbf{R}_i$ by simulating the first query made to Φ by $\mathbf{R}_i$. We proceed with the formal details.

For a fixed $f\in\mathcal{F},y\in\mathrm{Im}(f)$, $\mathbf{R}_i^{f,\Phi^f}(y)$ makes a set of queries $J_{f,y}$ prior to the first query to $\Phi^{(\cdot)}$. Furthermore, $|J_{f,y}|\leqslant t+is\leqslant 2ts$ since $\mathbf{R}_i$ makes $\leqslant t+is$ queries to f.

Let's fix $y\in[M]$ and take the average in the end (so all probabilities over f for the remainder of this argument are conditioned on $y\in\mathrm{Im}(f)$). Thus, we replace $J_{f,y}$ with simply J_f. Thus, the execution will only change depending on the queries made to either f or Φ^f.

Define $\mathcal{K}:=\{J\in\mathcal{F}^{<}:|J|\leqslant 2ts,\Pr_f[J_f=J]>0\}$. Then,

$$\Pr_f\left[(f,y)\in \mathrm{Sim}^{\mathbf{R}_i}\cap \mathrm{Inv}^{\mathbf{R}_i}\right]=\sum_{J\in\mathcal{K}}\Pr_f\left[(f,y)\in \mathrm{Sim}^{\mathbf{R}_i}\cap \mathrm{Inv}^{\mathbf{R}_i}\text{ and }J_f=J\right]$$

This implies that under fixing the queries J prior to the first query to $\Phi^{(\cdot)}$, then the first query $C_J\in\mathcal{C}$ made by both $\mathbf{R}_i$ and $\mathbf{R}_{i+1}$ is fixed and identical, and the execution up to this query is identical.

Thus, there exists an algorithm $\mathbf{S}_J^{(\cdot)}$ that simulates $\Phi^{(\cdot)}(C_J)$, making $\leqslant s$ queries to its oracle from $\mathcal{F}$.

We construct $\mathbf{R}_{i+1}$ identical to $\mathbf{R}_i$ but when it queries the first query C_J to Φ (that solely relies on y and the prior transcript J), $\mathbf{R}_{i+1}$, with access to $f \in \mathcal{F}$, simulates $\mathbf{S}_J^f(J, C_J)$.

Clearly, the amount of queries that $\mathbf{R}_{i+1}$ makes to Φ decreases by 1, and the ones it makes to some $f \in \mathcal{F}$ increase by $\leqslant s$, so the condition on the number of queries mentioned previously is satisfied.

Now, the execution of $\mathbf{R}_{i+1}$ is then identical to $\mathbf{R}_i$ if $\mathbf{S}_J^f(J, C_J) = \Phi^f(C_J)$.

Let Succ_J collect all $f \in \mathcal{F}$ where $\mathbf{S}_J^f(J, C_J) = \Phi^f(C_J)$. We want to show that

$$\sum_{J \in \mathcal{K}} \Pr_{f \leftarrow \mathcal{F}} \left[(f, y) \in \mathrm{Sim}^{\mathbf{R}_i}, f \notin \mathrm{Succ}_J, \text{and } J_f = J \mid y \in \mathrm{Im}(f) \right] \leqslant \gamma + \frac{2ts}{N}. \tag{4.1}$$

If so, then following the same coupling techniques introduced in the proof of **Theorem 3.3.3**, we get

$$\begin{aligned}
&\Pr_f \left[(f, y) \in \mathrm{Sim}^{\mathbf{R}_i} \cap \mathrm{Inv}^{\mathbf{R}_i} \right] \\
&= \sum_{J \in \mathcal{K}} \Pr_{f,J} \left[(f, y) \in \mathrm{Sim}^{\mathbf{R}_i} \cap \mathrm{Inv}^{\mathbf{R}_i} \right] && \text{(subscript } J \text{ adds the event “and } J_f = J\text{”)} \\
&= \sum_{J \in \mathcal{K}} \Pr_{f,J} \left[(f, y) \in \mathrm{Sim}^{\mathbf{R}_i} \cap \mathrm{Inv}^{\mathbf{R}_i} \text{ and } f \in \mathrm{Succ}_J \right] \\
&+ \sum_{J \in \mathcal{K}} \Pr_{f,J} \left[(f, y) \in \mathrm{Sim}^{\mathbf{R}_i} \cap \mathrm{Inv}^{\mathbf{R}_i} \text{ and } f \notin \mathrm{Succ}_J \right] \\
&\leqslant \sum_{J \in \mathcal{K}} \Pr_{f,J} \left[(f, y) \in \mathrm{Sim}^{\mathbf{R}_i} \cap \mathrm{Inv}^{\mathbf{R}_i} \text{ and } f \in \mathrm{Succ}_J \right] + \gamma + \frac{2ts}{N} && \text{(by 4.1)} \\
&\leqslant \sum_{J \in \mathcal{K}} \Pr_{f,J} \left[(f, y) \in \mathrm{Sim}^{\mathbf{R}_{i+1}} \cap \mathrm{Inv}^{\mathbf{R}_{i+1}} \text{ and } f \in \mathrm{Succ}_J \right] + \gamma + \frac{2ts}{N} && \text{(since } \mathbf{S}_J^f \text{ is successful)} \\
&\leqslant \sum_{J \in \mathcal{K}} \Pr_{f,J} \left[(f, y) \in \mathrm{Sim}^{\mathbf{R}_{i+1}} \cap \mathrm{Inv}^{\mathbf{R}_{i+1}} \right] + \gamma + \frac{2ts}{N}. \\
&= \Pr_f \left[(f, y) \in \mathrm{Sim}^{\mathbf{R}_{i+1}} \cap \mathrm{Inv}^{\mathbf{R}_{i+1}} \right] + \gamma + \frac{2ts}{N}. && \text{(eliminating the condition over } J\text{)}
\end{aligned}$$

Taking the average over $y \in [M]$ (noting that we simplified the notation by removing the condition $y \in \mathrm{Im}(f)$), we get our desired bound, completing the proof. It remains to show 4.1.

We observe crucially that for any $J \in \mathcal{K}$, since we required $\Pr_f[J_f = J] > 0$, $J_f = J$ if and only if $J \subseteq f$.

The forward direction here is immediate and the backward direction follows from the fact that $\Pr_f[J_f = J] > 0$ means that for some $g \in \mathcal{F}$ $\mathbf{R}_i^{g, \Phi^g}(y)$ made the queries exactly in J before querying Φ; thus, following the decision tree of queries by $\mathbf{R}_i^{(\cdot), \Phi^{(\cdot)}}(y)$, it will meet the queries made in J and thus must decide to query Φ and make $J_f = J$.

Now, define

$$\mathcal{K}_{\mathrm{Sim}} := \left\{ J \in \mathcal{K} : C_J \text{ is } (s, 2ts, \gamma)\text{-simulatable using } \mathbf{S}_J^{(\cdot)} \right\}$$

Observe that $(f, y) \in \mathrm{Sim}^{\mathbf{R}_i}$ implies $J \subseteq f$ for some $J \in \mathcal{K}_{\mathrm{Sim}}$. Thus,

$$
\begin{aligned}
&\sum_{J \in \mathcal{K}} \Pr_{f \leftarrow \mathcal{F}} \left[(f, y) \in \mathrm{Sim}^{\mathbf{R}_i}, f \notin \mathrm{Succ}_J, \text{and } J_f = J \mid y \in \mathrm{Im}(f)\right] \\
&= \sum_{J \in \mathcal{K}} \Pr_{f \leftarrow \mathcal{F}} \left[(f, y) \in \mathrm{Sim}^{\mathbf{R}_i}, f \notin \mathrm{Succ}_J, \text{and } J \subseteq f \mid y \in \mathrm{Im}(f)\right] \\
&\leqslant \sum_{J \in \mathcal{K}_{\mathrm{Sim}}} \Pr_{f \leftarrow \mathcal{F}} [f \notin \mathrm{Succ}_J \text{ and } J \subseteq f \mid y \in \mathrm{Im}(f)] \\
&= \sum_{J \in \mathcal{K}_{\mathrm{Sim}}} \Pr_{f \leftarrow \mathcal{F}} [f \notin \mathrm{Succ}_J \mid J \subseteq f, y \in \mathrm{Im}(f)] \Pr_{f \leftarrow \mathcal{F}} [J \subseteq f \mid y \in \mathrm{Im}(f)] \\
&= \sum_{J \in \mathcal{K}_{\mathrm{Sim}}} \Pr_{f \leftarrow \mathcal{F}} [f \notin \mathrm{Succ}_J \mid J \subseteq f, y \in \mathrm{Im}(f)]\, a_J && \text{(for simplicity)} \\
&\leqslant \sum_{\substack{J \in \mathcal{K}_{\mathrm{Sim}} \\ y \notin \mathrm{Im}(J)}} \Pr_{f \leftarrow \mathcal{F}} [f \notin \mathrm{Succ}_J \mid J \subseteq f, y \in \mathrm{Im}(f)]\, a_J + \frac{2ts}{N} && \text{(to account for } y \in \mathrm{Im}(J)) \\
&\leqslant \sum_{\substack{J \in \mathcal{K}_{\mathrm{Sim}} \\ y \notin \mathrm{Im}(J)}} \Pr_{f \leftarrow \mathcal{F}} \left[\mathbf{S}_J^f(J, C_J) \neq \Phi^f(C_J) \mid J \subseteq f, y \in \mathrm{Im}(f)\right] a_J + \frac{2ts}{N} \\
&\leqslant \sum_{\substack{J \in \mathcal{K}_{\mathrm{Sim}} \\ y \notin \mathrm{Im}(J)}} \gamma \cdot a_J + \frac{2ts}{N} && \text{(since } J \in \mathcal{K}_{\mathrm{Sim}}, \, y \notin \mathrm{Im}(J), \text{ and by definition of } \mathcal{K}_{\mathrm{Sim}}) \\
&\leqslant \gamma + \frac{2ts}{N}
\end{aligned}
$$

as desired. □

As mentioned in the beginning of this section, we can classify $\mathcal{C}$ into subclasses and define Φ on each subclass. The above theorem tells us that one class where we can easily find sufficient non-leaking assignments is simulatable circuits.

We can then figure out insights about what circuits can be simulated and what circuits can't. Furthermore, we construct a circuit that we prove is *not simulatable* in **subsection 4.2.4**. For now, we provide an example of circuits that are simulatable: circuits that have small hitting sets.

Theorem 4.1.8 (Circuits with small hitting sets are simulatable)**.** Suppose $C \in \mathcal{C}$ is a satisfiable circuit with t oracle gates where there exists $H \subseteq \{0,1\}^m$ with $|H| \leqslant r$ such that

$$\Pr_{f \leftarrow \mathcal{F}} \left[C^f(u) = 1 \text{ for some } u \in H \mid J \subseteq f, y \in \mathrm{Im}(f)\right] \geqslant 1 - \gamma.$$

for every $J \in \mathcal{F}^{<}$ where $|J| \leqslant \ell$, $y \in [M] \setminus \mathrm{Im}(J)$.

Then, C is (rt, ℓ, γ)-simulatable.

Proof. We define Ψ_C as follows:

1. If $C^f(u) = 1$ for some $u \in H$, then set $\Psi_C(f)$ to be the lexicographically least $u \in H$ where $C^f(u) = 1$.
2. Otherwise, $\Psi_C(f)$ is an arbitrary satisfying assignment of C^f.

Then, $\mathbf{S}^f(J, C)$ works as follows:

1. Compute $C^f(u)$ for every $u \in H$, querying f when needed.
2. Output the lexicographically least $u \in H$ where $C^f(u) = 1$ if it exists.
3. Otherwise, output 0^m.

Observe that this makes rt queries (t queries for each $u \in H$) in the worst case.

Clearly, if f satisfies $C^f(u) = 1$ for some $u \in H$, then $\mathbf{S}^f(J, C) = \Psi_C(f)$ when $J \subseteq f$.

This is because after computing each $C^f(u)$, regardless of J, $\mathbf{S}$ has sufficient information to simulate $\Psi_C(f)$ when $C^f(u) = 1$ for some $u \in H$.

Thus,

$$\begin{aligned} & \Pr_{f \leftarrow \mathcal{F}} \left[\mathbf{S}^f(J, C) \neq \Psi_C(f) \mid J \subseteq f, y \in \mathrm{Im}(f) \right] \\ \leqslant \; & \Pr_{f \leftarrow \mathcal{F}} \left[C^f(u) = 0 \text{ for all } u \in H \mid J \subseteq f, y \in \mathrm{Im}(f) \right] \\ & \leqslant \; \gamma \end{aligned}$$

as desired. □

A strong version of the above condition is that if $C^f(u) = 1$ for some $u \in H$ *for every* $f \in \mathcal{F}$ instead.

For example, the circuit that refutes the oracle that outputs the lexicographically least satisfying assignment mentioned in the beginning of **section 3.1**:

$$C^f(u) := \mathbf{1}[f(u) = y] \vee \mathbf{1}[u = 1^n]$$

has $\{1^n\}$ as a hitting set for every $f \in \mathcal{F}$, and thus is simulatable.

This tells us that the circuits that $\mathbf{R}$ submits must have a more sophisticated proof of satisfiability than just a hitting set.

4.2 Adaptive Queries with One Oracle Gate

This section contains the biggest contribution we make. We prove separation against reductions $\mathbf{R}$ that only submit queries with one oracle gate to Φ.

In particular, we show that circuits with one oracle gate are simulatable, and investigate the beautiful combinatorial structure of circuits with one or more oracle gates. Surprisingly, we show that there exists a circuit with two oracle gates that is not simulatable.

This section is structured into four subsections.

1. **Subsection 4.2.1** introduces the beautiful combinatorial structure of the problem using the ideas from **subsection 4.1.2**.

   We introduce different ways to visualize t-Turán sets which can help in investigating their combinatorial structure.

2. **Subsection 4.2.2** further expands on the previous section by showing a simple classification of 1-Turán sets.

3. **Subsection 4.2.3** uses the classification from the previous section to show that circuits with one oracle gate are simulatable.

4. **Subsection 4.2.4** shows how attempts to generalize the classification in **subsection 4.2.2** to t-Turán sets for $t > 1$ fail. In particular, we provide a circuit with two oracle gates that we show is not simulatable.

4.2.1 The Combinatorial Underworld

We have seen in **subsection 4.1.2** that choosing a satisfying assignment for a circuit corresponds to choosing some transcript from a *t-Turán set*.

From that subsection, we know about any t-Turán set $\mathcal{T}$ the following:

1. For every $T \in \mathcal{T}$, $T \in \mathcal{F}^{<}$ with $|T| = t$.
2. Every $f \in \mathcal{F}$ has some $T \in \mathcal{T}$ where $T \subseteq f$.
3. For some $\ell \leqslant 2^m$, then there are pairwise distinct sets $A_1, A_2, \ldots, A_\ell \subseteq [N]$ where
   (a) $|A_i| = t$ for all $i \in [\ell]$ and
   (b) $\mathcal{T}$ can be partitioned as
   $$\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2 \cup \cdots \cup \mathcal{T}_\ell$$
   where $\mathcal{T}_i = \{T \in \mathcal{T} : \mathrm{Dom}(T) = A_i\}$ for all $i \in [\ell]$ ($\mathcal{T}_i$ is possibly empty)

In order to study the structure of t-Turán sets to better inform an oracle Φ, we translate the setting into a combinatorial setting in the following manner:

1. Consider the complete bipartite graph on sets $[N]$ and $[M]$, $\mathcal{G}_{N,M}$.[2]

   For our purposes, edges of $\mathcal{G}_{N,M}$ are *ordered*. That is, every $e \in \mathcal{G}_{N,M}$ can be written as $e = (u, v)$ where $u \in [N]$ and $v \in [M]$.

[2]Recall that $[N]$ and $[M]$ are assumed to be the domain and codomain of functions in $\mathcal{F}$ respectively.

2. Elements of $\mathcal{F}^<$ are *matchings* in $\mathcal{G}_{N,M}$, as implied by injectivity.[3]
3. Elements of $\mathcal{F} \subseteq \mathcal{F}^<$ are matchings that match $[N]$ into $[M]$. We call such matchings *left-perfect*.
4. A t-Turán set $\mathcal{T}$ is a set of t-matchings (matchings having t edges) where every left-perfect matching f has one $T \in \mathcal{T}$ with $T \subseteq f$.

Choosing the values of $\Phi^{(\cdot)}(C)$ is the same as choosing a function $\Phi : \mathcal{F} \to \mathcal{T}$ that sends a left-perfect matching into a t-matching inside it from $\mathcal{T}$.

Thus, since the structure of $\mathcal{F}$ is known, then investigating the structure of $\mathcal{T}$ can better inform the choice of Φ.

Interesting patterns show when investigating different values of t (corresponding to the number of oracle gates in the circuit):

1. On $t = 1$, $\mathcal{T}$ is just a set of edges (i.e. a subgraph of $\mathcal{G}_{N,M}$).
2. On $t = 2$, $\mathcal{T}$ contains pairs of edges. Thus, it can be viewed as a *simple graph on the edges of* $\mathcal{G}_{N,M}$.
3. On $t > 2$, $\mathcal{T}$ is a *t-uniform hypergraph*[4] on the edges of $\mathcal{G}_{N,M}$.

Thus, as an alternative way to visualize, we can imagine the edges of $\mathcal{G}_{N,M}$ as points on an $N \times M$ grid:

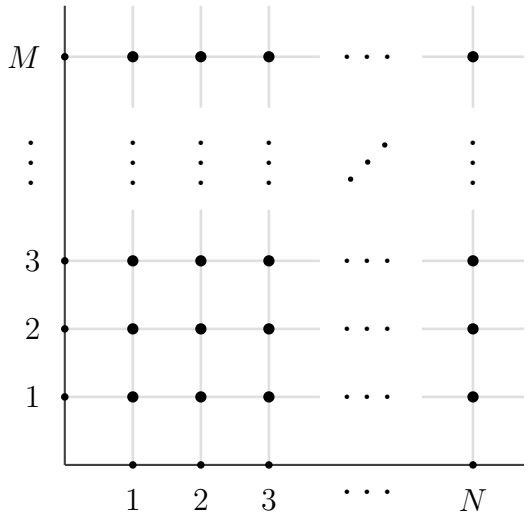


Figure 4.1: Edges of a complete bipartite graph as points a grid

Thus, we can define $\mathcal{V}_{N,M} := [N] \times [M]$ where every $v \in \mathcal{V}_{N,M}$ can be represented by $v = (v_x, v_y)$.

[3] A *matching* is a set of edges that are pairwise disjoint (i.e., they don't touch the same vertex).

[4] A t-uniform hypergraph on a set V is a set E of *t-hyperedges* on V. That is, $e \in E$ has $e \subseteq V$ and $|e| = t$.

To be a 2-matching, an edge $\{u, v\} \subseteq \mathcal{V}_{N,M}$ must have u and v sharing no row or no column, or $u_x \neq v_x$ and $u_y \neq v_y$ (i.e., the ordered edges u and v don't intersect in $[N]$ or in $[M]$).

So, all 2-matchings are edges of the simple graph $\mathcal{K}^2_{N,M} = \{\{u, v\} \subseteq \mathcal{V}_{N,M} : u_x \neq v_x, u_y \neq v_y\}$.

We show the grid visualization of $\mathcal{K}^2_{3,4}$ in **figure 4.2**.

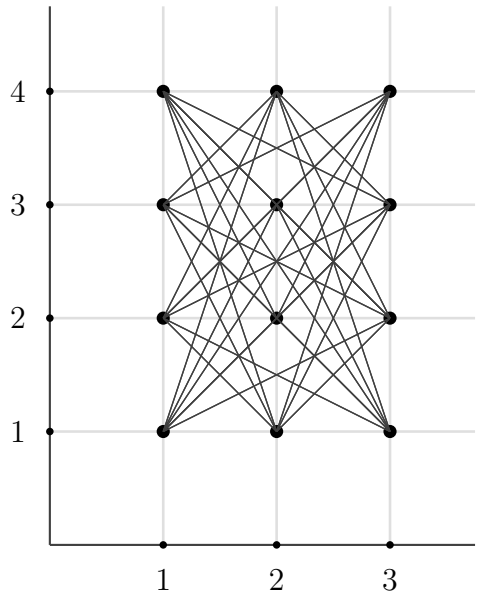


Figure 4.2: All edges of $\mathcal{K}^2_{3,4}$ representing every 2-matching in $\mathcal{G}_{3,4}$.

The generalized version of this visualization is that t-matchings are *t-hyperedges* in the t-uniform hypergraph:

$$\mathcal{K}^t_{N,M} = \left\{ \left\{ u^1, u^2, \ldots, u^t \right\} \subseteq \mathcal{V}_{N,M} : u^i_x \neq u^j_x, u^i_y \neq u^j_y \text{ for all } i, j \in [t] \text{ with } i \neq j \right\}.$$

Figure 4.3 displays a small subset of 3-hyperedges in $\mathcal{K}^3_{3,6}$.

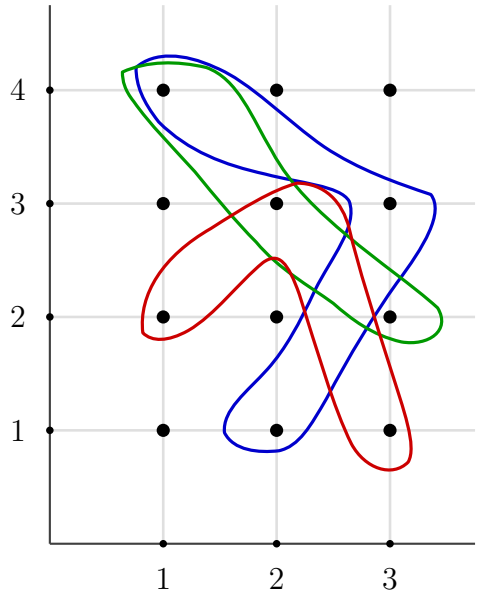


Figure 4.3: A 3-uniform hypergraph on a 3×4 grid, with hyperedges drawn as smooth closed curves.

Thus, we now obtain two beautiful representations of injective functions in $\mathcal{F}$ and partial assignments in $\mathcal{F}^<$ of size t:

1. **The bipartite view**: An injective function in $\mathcal{F}$ is a left-perfect matching in $\mathcal{G}_{N,M}$.

   A partial assignment of size t is a t-matching in $\mathcal{G}_{N,M}$.

2. **The grid view**: An injective function $f \in \mathcal{F}$ is a set of N vertices in $\mathcal{V}_{N,M}$ that share no row or column (i.e., for every $u, v \in f$ with $u \neq v$ then $u_x \neq v_x$ and $u_y \neq v_y$.)

   A partial assignment of size t is a t-uniform hyperedge in $\mathcal{K}^t_{N,M}$.

In summary, the bipartite view looks at partial assignments in $\mathcal{F}^<$ as sets of **edges** that don't intersect pairwise, while the grid view looks at them as sets of **vertices** or points on a grid that share no row or column.

For example, the edges (or 3-matchings in the bipartite view) in the hypergraph in **figure 4.3** are $\{\text{red}, \text{green}, \text{blue}\}$ where

$$\begin{aligned} \text{red} &= \{(1,2),(2,3),(3,1)\}, \\ \text{green} &= \{(1,4),(2,3),(3,2)\}, \text{ and} \\ \text{blue} &= \{(1,4),(2,1),(3,3)\}. \end{aligned}$$

If we look at the bipartite view, we can visualize the previous 3-uniform hypergraph as in **figure 4.4** ($(1,4)$ is drawn with two separate edges since they overlap).

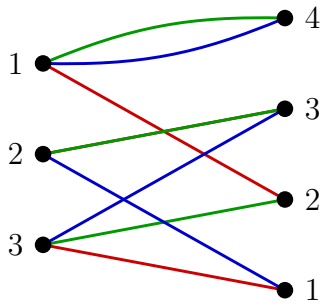


Figure 4.4: The three colored hyperedges viewed as 3-matchings in the bipartite graph $\mathcal{G}_{3,4}$.

We jump between the two views often as needed, specifying which one we talk about. Observe that since we specified that edges in $\mathcal{G}_{N,M}$ are ordered, then the two views set-theoretically are identical.

In other words, $\mathcal{V}_{N,M} = \mathcal{G}_{N,M} = [N] \times [M]$, and elements of $\mathcal{F}^<$ are simply subsets of $[N] \times [M]$ that share no row or column in $\mathcal{V}_{N,M}$ or are a matching in $\mathcal{G}_{N,M}$.

The insightful idea from the second representation is that a set of t-matchings $\mathcal{T}$ is t-Turán if and only if its complement contains no *cliques* of size N: [5]

[5] If $\mathcal{E}$ is a t-uniform hypergraph on $\mathcal{V}$, we call a subgraph $\mathcal{E}' \subseteq \mathcal{E}$ is called a *clique* if there exists $\mathcal{V}' \subseteq \mathcal{V}$ where $\mathcal{E}' = \{e \subseteq \mathcal{V}' : |e| = t\}$ (i.e., $\mathcal{E}'$ collects all t-hyperedges on $\mathcal{V}'$).

Theorem 4.2.1 (Alternative definition of t-Turán sets)**.** Let $\mathcal{T}$ be a set of t-matchings in the complete bipartite graph $\mathcal{G}_{N,M}$ (thus, a subgraph of $\mathcal{K}^t_{N,M}$), and let $\mathcal{L} := \mathcal{K}^t_{N,M} \backslash \mathcal{T}$ be its complement.

For $t \geqslant 2$, we call $\mathcal{L}$ *t-perfect* if it contains a clique f of size N. That is, there exists $f \subseteq \mathcal{V}_{N,M}$ with

1. $|f| = N$ and
2. for every $T \subseteq f$ with $|T| = t$ we have $T \in \mathcal{L}$.

For $t = 1$, we call $\mathcal{L}$ *1-perfect* if, as a bipartite graph, it contains a left-perfect matching.

Then, $\mathcal{T}$ is t-Turán if and only if its complement is not t-perfect.

Proof. Observe that in the grid view, for $f \subseteq \mathcal{V}_{N,M}$ to be a clique of size N inside $\mathcal{K}^t_{N,M}$, the vertices inducing it must exactly represent (in the bipartite view) the edges of a left-perfect matching $f \in \mathcal{F}$.

This is because every pair of edges in f (as a left-perfect matching, bipartite view) cannot touch in $[N]$ or in $[M]$, as in the grid view, they must be contained in some t-hyperedge connecting vertices of f.

In other words, in the grid view, $\mathcal{F}$ collects all possible candidates for cliques of size N that can be contained in $\mathcal{L}$.

Now, if $\mathcal{T}$ is t-Turán, then every $f \in \mathcal{F}$ has $T \in \mathcal{T}$ where $T \subseteq f$.

Thus, for every $f \in \mathcal{F}$, there must exist some $T \subseteq f$ with $|T| = t$ but $T \notin \mathcal{L}$. So, $\mathcal{L}$ cannot contain any clique f of size N.

Conversely, if $\mathcal{L}$ is not t-perfect, then every $f \in \mathcal{F}$ must have some $T \subseteq f$ with $|T| = t$ but $T \notin \mathcal{L}$. Since $\mathcal{T}$ is the complement of $\mathcal{L}$, this implies that such T is in $\mathcal{T}$.

Thus, every $f \in \mathcal{F}$ has some $T \subseteq f$ with $T \in \mathcal{T}$, as desired. □

The only tricky part of this proof is the constant jumping between the bipartite view and the grid view. We call t-perfect sets with this name since they induce left-perfect matchings in the bipartite view.

4.2.2 Combinatorial Structure of 1-Turán Sets

For $t = 1$, the situation is simple and beautiful. First, we note the following:

1. In the grid view, a 1-Turán set $\mathcal{T}$ is a 1-uniform hypergraph on $[N] \times [M]$ (every element is a singleton).

   We simplify this and view it as a subset of $\mathcal{T} \subseteq [N] \times [M]$.[6]

[6] Instead of $\{\{v_1\}, \{v_2\}, \ldots, \{v_h\}\}$, we look at $\{v_1, v_2, \ldots, v_h\}$.

2. In the bipartite view, a 1-Turán set $\mathcal{T}$ is a bipartite graph, or a subgraph of $\mathcal{G}_{N,M}$, whose complement does not contain any left-perfect matching.

   In other words, $\mathcal{G}_{N,M} \setminus \mathcal{T}$ is a bipartite graph with maximum matching of size less than N.

A special kind of 1-Turán sets are 1-*rectangles*. We show a special property about rectangles.

Lemma 4.2.2 (Large 1-rectangles are 1-Turán)**.** Let $\mathcal{T} \subseteq \mathcal{G}_{N,M}$ be nonempty.

We call $\mathcal{T}$ a 1-*rectangle* if there exists nonempty $S \subseteq [N]$ and $H \subseteq [M]$ with $\mathcal{T} = S \times H$.

Then, a 1-rectangle $\mathcal{T} = S \times H$ is 1-Turán if and only if $|S| + |H| > M$.

Proof. Suppose $S \times H$ is 1-Turán, and suppose $|S| + |H| \leqslant M$. Let $R = [M] \setminus H$. Then, $|S| \leqslant |R|$.

Then, some left perfect matching $f \in \mathcal{F}$ can match S inside R (in the language of functions, $f(S) \subseteq R$).

Thus, it cannot contain any edge (u, v) where $u \in S$ and $v \in H$. This contradicts that $S \times H$ is 1-Turán.

Conversely, suppose $|S| + |H| > M$. Let $R = [M] \setminus H$. Then, $|S| > |R|$. Thus, any left-perfect matching $f \in \mathcal{F}$ can't match S into R by pigeonhole (in the language of functions, we can't have $f(S) \subseteq R$).

Thus, every left perfect matching $f \in \mathcal{F}$, must contain some $(u, v) \in f$ where $u \in S$ and $v \notin R$. In other words, $v \in H$. This shows that $S \times H$ is 1-Turán. □

We observe that **Theorem 4.2.1** tells us that 1-Turán sets are bipartite graphs whose complement cannot match $[N]$ into $[M]$.

Looking at a 1-Turán set $\mathcal{T}$ this way gives us an insightful idea: it turns out that minimal 1-Turán sets are "rectangles." In fact, this statement is equivalent to Hall's Theorem (see **Theorem B.0.2**):

Theorem 4.2.3 (Minimal 1-Turán sets are "rectangles")**.** We call a t-Turán set $\mathcal{T} \subseteq \mathcal{K}^t_{N,M}$ *minimal* if every $\mathcal{T}' \subsetneq \mathcal{T}$ is not t-Turán.

Then, the following are equivalent:

1. For every nonempty 1-Turán set $\mathcal{T} \subseteq \mathcal{G}_{N,M}$, if $\mathcal{T}$ is minimal, then $\mathcal{T}$ is a 1-rectangle.
2. Hall's Theorem holds. That is, $\mathcal{L} \subseteq \mathcal{G}_{N,M}$ is 1-perfect if and only if for every $S \subseteq [N]$, $|S| \leqslant |N_{\mathcal{L}}(S)|$.

Here, $N_{\mathcal{L}}(S) := \{v \in [M] : (u, v) \in \mathcal{L} \text{ for some } u \in S\}$.

Proof. Let start with the important direction first: if Hall's Theorem holds, then every minimal 1-Turán set is a 1-rectangle.

Suppose $\mathcal{T}$ is a minimal 1-Turán set. Let $\mathcal{L} = \mathcal{G}_{N,M} \setminus \mathcal{T}$. Then, $\mathcal{L}$ is not 1-perfect, by **Theorem 4.2.1**.

By Hall's Theorem, there must exist some nonempty $S \subseteq [N]$ with $|S| > |N_{\mathcal{L}}(S)|$.

Let $H := [M] \setminus N_{\mathcal{L}}(S)$. Then, $|S| + |H| > M$. Furthermore, every $u \in S$ and $v \in H$ must have $(u, v) \notin \mathcal{L}$ since $u \in S$ and $v \notin N_{\mathcal{L}}(S)$.

In other words, $u \in S$ and $v \in H$ must have $(u, v) \in \mathcal{T}$ and so $S \times H \subseteq \mathcal{T}$. Since $|S| + |H| > M$, then by **Lemma 4.2.2**, $S \times H$ is 1-Turán. By minimality, $\mathcal{T} = S \times H$, as desired.

Now for the converse, suppose every minimal 1-Turán set is a rectangle. We show Hall's Theorem. The forward direction of Hall's is trivial, as $N_{\mathcal{L}}(S)$ must contain the set S is matched to by $\mathcal{L}$.

For the backward direction, we show the contrapositive. Suppose $\mathcal{L}$ is not 1-perfect. Let $\mathcal{T} = \mathcal{G}_{N,M} \setminus \mathcal{L}$ be its complement. By **Theorem 4.2.1**, $\mathcal{T}$ must be 1-Turán.

Since every minimal 1-Turán set is a rectangle, there must exist rectangle inside $\mathcal{T}$. In other words, some $S \subseteq [N]$ and $H \subseteq [M]$ exist with $|S| + |H| > M$ and $S \times H \subseteq \mathcal{T}$.

Now, let $R = [M] \setminus H$. Then, $|S| + |H| > M$ translates to $|S| > |R|$. Since $S \times H \subseteq \mathcal{T}$, then it must be that $N_{\mathcal{L}}(S) \subseteq R$. Thus, $|N_{\mathcal{L}}(S)| \leqslant |R| < S$, as desired. □

This gives us a complete and simple classification of 1-Turán sets:

- $\mathcal{T} \subseteq \mathcal{G}_{N,M}$ is 1-Turán if and only if it contains some rectangle $S \times H \subseteq \mathcal{T}$ where $|S| + |H| > M$.

In the grid view, this looks like **figure 4.5** (with points only inside the rectangle, this is why it's called a rectangle).

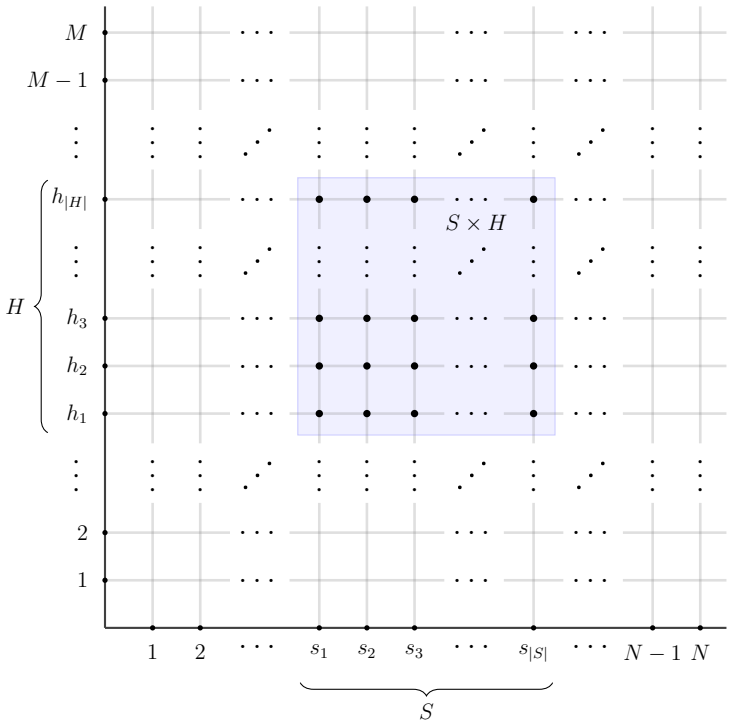


Figure 4.5: Grid view of a rectangle $S \times H \subseteq [N] \times [M]$.

In the bipartite view, this looks like **figure 4.6**.

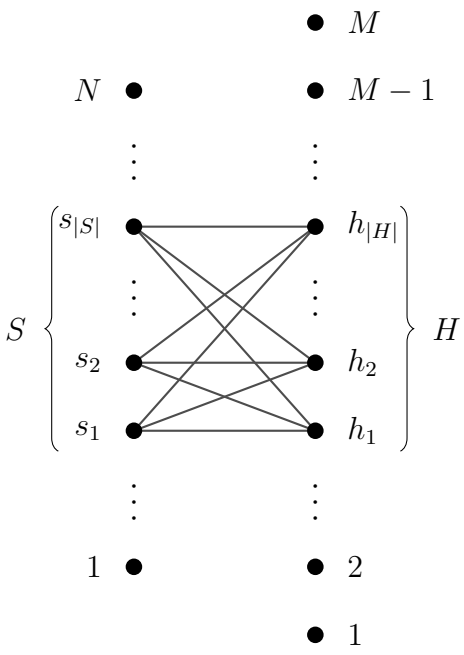


Figure 4.6: Bipartite view of a rectangle $S \times H$

In the next subsection, we show how to use this simple structure to simulate circuits in $\mathcal{C}$ with one oracle gate.

4.2.3 Simulating Circuits with One Oracle Gate

In this section, we show that the nice structure induced by circuits with one oracle gate allows us to simulate them in the following method:

1. Let $\mathcal{T}$ be the 1-Turán set induced by the circuit C given in the input.
2. Find a minimal 1-Turán set $\mathcal{T}' \subseteq \mathcal{T}$ and find $S \subseteq [N]$ and $H \subseteq [M]$ where $\mathcal{T}' = S \times H$.
3. **Lemma 4.2.2** implies that $|S| + |H| > M$ and so $|H| > M - N$ since $|S| \leqslant N$.
4. This means that for a fresh query a and over a uniform f, $f(a) \notin H$ with probability $\leqslant N/M$, which is negligible with $M \geqslant N^2$.
5. Thus, sample some fresh queries from S and if for some fresh query $a \in S$ has $f(a) \in H$ we must have $(a, f(a)) \in S \times H = \mathcal{T}' \subseteq \mathcal{T}$, yielding a satisfying assignment.

Let's try to some examples to see this in action:

1. The circuit that leaks y on the lexicographically least satisfying assignment:

$$C^f(u) = \mathbf{1}[f(u) = y] \vee \mathbf{1}[u = 1^n].$$

We observe that this circuit induces the transcript

$$\mathcal{T} = \left\{(1^n, z) : z \in \{0,1\}^k\right\} \cup \{(u, y) : u \in \{0,1\}^n\}$$

since when $u = 1^n$, all transcripts of size 1 with domain $\{1^n\}$ yields a satisfying assignment (every possible output of $f(1^n)$) and when $u \neq 1^n$, then the transcripts $u \mapsto y$ are only the ones that work.

Since we denote $[N] = \{0,1\}^n$ and $[M] = \{0,1\}^k$, 1^n is denoted by N and y is just denoted by $y \in [M]$, we get a diagram of $\mathcal{T}$ shown in **figure 4.7**. From the figure,

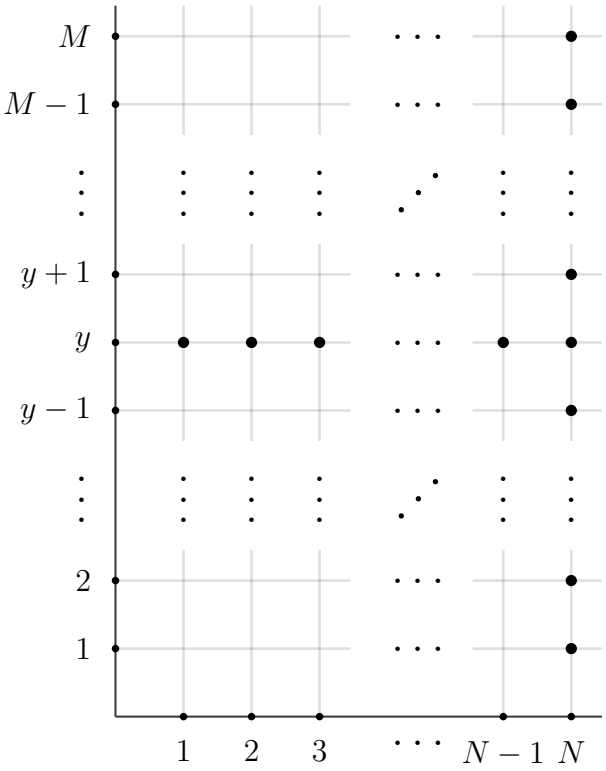


Figure 4.7: 1-Turán set induced by the circuit that leaks y on the least satisfying assignment.

we can see that the only valid choice for a rectangle is $\{N\} \times [M]$.

Thus, we can simulate the circuit by just querying $f(N)$ and the corresponding satisfying assignment will always be 1^n.

2. The circuit in **Theorem 4.1.1**:

$$C^f(x) := \mathbf{1}[f(x) \leqslant M - N + 1]$$

The rectangle for this is clearly $[N] \times [M - N + 1]$, since (x, y) where $x \in [N]$ and $y \in [M - N + 1]$ are exactly the accepting transcripts. So, querying any possible fresh queries will simulate the circuit.

Let's formalize these ideas. We start with this lemma:

Lemma 4.2.4. Let $J \in \mathcal{F}^{<}$, $t = |J|$, and $y \in [M] \setminus \mathrm{Im}(J)$.

Then, for any $a \in [N] \setminus \mathrm{Dom}(J)$ and $H \subseteq [M]$, we have

$$\Pr_{f \leftarrow \mathcal{F}}[f(a) \notin H \mid J \subseteq f, y \in \mathrm{Im}(f)] \leqslant \frac{M - |H|}{M - (t+1)} + \frac{1}{N - t}$$

Proof. This is basically a counting lemma. Let $D = \mathrm{Dom}(J), I = \mathrm{Im}(J)$,

$$A := \{f \in \mathcal{F} : f(a) \notin H, J \subseteq f, y \in \mathrm{Im}(f)\} \text{ and}$$
$$B := \{f \in \mathcal{F} : J \subseteq f, y \in \mathrm{Im}(f)\}$$

And we want to bound $|A| / |B|$. For the denominator,

$$\begin{aligned} |B| &= \sum_{b \in [N] \setminus D} |\mathrm{Cube}\,(J \cup \{(b, y)\})| && \text{(since } y \notin \mathrm{Im}(J)\text{)} \\ &= \sum_{b \in [N] \setminus D} \frac{(M - (t+1))!}{(M - N)!} \\ &= (N - t)\frac{(M - (t+1))!}{(M - N)!}. \end{aligned}$$

For the numerator, we consider different values of $f(a)$ (denoted with z) or preimages of y (denoted with b)

$$\begin{aligned} A &= \{f \in \mathcal{F} : f(a) \notin H, J \subseteq f, y \in \mathrm{Im}(f)\} \\ &= \bigcup_{z \in [M] \setminus H} \bigcup_{b \in [N] \setminus D} \{f \in \mathcal{F} : J \cup \{(a, z), (b, y)\} \subseteq f\} \\ &= \bigcup_{z \in [M] \setminus H} \bigcup_{b \in [N] \setminus D} \mathrm{Cube}\,(J \cup \{(a, z), (b, y)\}) \\ &\subseteq \bigcup_{z \in [M] \setminus H} \bigcup_{b \in [N] \setminus (D \cup \{a\})} \mathrm{Cube}\,(J \cup \{(a, z), (b, y)\}) \cup \mathrm{Cube}\,(J \cup \{(a, y)\}) \end{aligned}$$

So, we get

$$\begin{aligned} |A| &\leqslant \sum_{z \in [M] \setminus H} \sum_{b \in [N] \setminus (D \cup \{a\})} \frac{(M - (t+2))!}{(M - N)!} + \frac{(M - (t+1))!}{(M - N)!} \\ &\leqslant (M - |H|)(N - t)\frac{(M - (t+2))!}{(M - N)!} + \frac{(M - (t+1))!}{(M - N)!}. \end{aligned}$$

So,

$$\frac{|A|}{|B|} \leqslant \frac{(M - |H|)(N - t)\frac{(M-(t+2))!}{(M-N)!} + \frac{(M-(t+1))!}{(M-N)!}}{(N - t)\frac{(M-(t+1))!}{(M-N)!}} = \frac{M - |H|}{M - (t+1)} + \frac{1}{N - t}$$

as desired. □

So now, with $|H| \geqslant M - N$, this probability is $\leqslant 4/N$. Thus, we get the following theorem:

Theorem 4.2.5 (Simulating circuits with one oracle gate)**.** Suppose $M \geqslant N^2$ and let $q, s \in \mathbb{N}^+$ where $q + s \leqslant N/2$. Let $C \in \mathcal{C}$ have one oracle gate.

Then, C is $\left(s, q, \left(\frac{4}{N}\right)^s\right)$-simulatable.

Proof. We consider $[N] = \{1, 2, \ldots, N\}$ as an ordered set (corresponding to lexicographical ordering of $\{0,1\}^n$), and we construct $\Psi_C(f)$ as follows:

1. Let $\mathcal{T}$ be the 1-Turán set induced by the circuit C given in the input.
2. Find a minimal 1-Turán set $\mathcal{T}' \subseteq \mathcal{T}$, $S \subseteq [N]$, and $H \subseteq [M]$ where $\mathcal{T}' = S \times H$. This step is due to **Theorem 4.2.3**
3. Find the least $a \in S$ with $f(a) \in H$ and return the lexicographically least satisfying assignment corresponding to $(a, f(a))$.

Now, to align with Ψ_C, we construct $\mathbf{S}^f(J, C)$ as follows: (noting that $\mathbf{S}$ is only bounded in the queries it makes to f)

1. Let $\mathcal{T}$ be the 1-Turán set induced by the circuit C given in the input.
2. Find a minimal 1-Turán set $\mathcal{T}' \subseteq \mathcal{T}$, $S \subseteq [N]$, and $H \subseteq [M]$ where $\mathcal{T}' = S \times H$.
3. Iterate through S in order, until we queried at least s fresh queries (outside J) or covered all of S.

   So, for $a \in S$, we do the following:

   (a) If $a \in \mathrm{Dom}(J)$ and $J(a) \in H$, output the lexicographically least satisfying assignment corresponding to the transcript $(a, J(a))$.

   (b) If $a \notin \mathrm{Dom}(J)$, query $f(a)$. If $f(a) \in H$, output the lexicographically least satisfying assignment corresponding to the transcript $(a, f(a))$.

Observe that $\mathbf{S}^f(J, C)$ is designed to always choose the **least** valid element of S and the corresponding **lexicographically least** satisfying assignment, to align with Ψ_C as much as possible.

Thus, if we let $a_1, a_2, \ldots, a_s \in S$ be the fresh queried elements, then $\mathbf{S}$ can only fail if $f(a_i) \notin H$ for all $i \in [s]$.

Now, after querying $a_1, a_2, \ldots, a_{i-1}$, we can extend J with the corresponding query-answer pairs. This gives a partial assignment of size at most $q + i - 1 \leqslant q + s \leqslant N/2$.

Thus, by **Lemma 4.2.4**, for every $i \in [s]$, conditioned on all previous queried elements missing H, we have

$$\Pr_{f \leftarrow \mathcal{F}}[f(a_i) \notin H \mid J \subseteq f, y \in \mathrm{Im}(f), f(a_j) \notin H \text{ for all } j < i] \leqslant \frac{4}{N}.$$

Therefore, by the chain rule,

$$\begin{aligned}
&\Pr_{f \leftarrow \mathcal{F}}\left[\mathbf{S}^f(J, C) \neq \Psi_C(f) \mid J \subseteq f, y \in \mathrm{Im}(f)\right] \\
\leqslant\ &\Pr_{f \leftarrow \mathcal{F}}\left[f(a_i) \notin H \text{ for all } i \in [s] \mid J \subseteq f, y \in \mathrm{Im}(f)\right] \\
\leqslant\ &\left(\frac{4}{N}\right)^s,
\end{aligned}$$

as desired. □

So, we prove **Theorem 2.3.4** but only when $\mathbf{R}$ submits circuits with one oracle gate:

Theorem 4.2.6 (Impossibility against circuits with one oracle gate)**.** There exists an oracle Φ such that for every PPT algorithm $\mathbf{R}^{(\cdot),\Phi^{(\cdot)}}$ of runtime at most t that only submits circuits with one oracle gate to Φ we have

$$\Pr_{\mathbf{R}}\left[\mathbf{R}^{f,\Phi^f}(f(x)) = x : \begin{array}{l} f \leftarrow \mathcal{F} \\ x \leftarrow \{0,1\}^n \end{array}\right] \leqslant \frac{4t}{N} + \frac{4t^3}{N}.$$

Proof. Immediate from **Theorem 4.2.5** and **Theorem 4.1.7**. □

We can also allow $\mathbf{R}$ to submit other simulatable circuits in this result.

So, circuits with one oracle gate are simulatable. What about circuits with two oracle gates?

4.2.4 Lost in Translation: Challenges in Extending to Multiple Oracle Gates

Let's see what we did in the $t = 1$ (one oracle gate) case:

1. We observed that some sets of 1-matchings can be *1-rectangles*
2. We showed that $S \times H$ is 1-Turán if and only if $|S| + |H| > M$
3. We showed that minimal 1-Turán sets are rectangles if and only if Hall's Theorem holds, with the choice of the rectangle being $S \times ([M] \setminus N_{\mathcal{L}}(S))$.

Let's attempt to generalize each of these steps naturally.

Property 3 of the t-Turán sets we consider tells us that we only consider t-matchings that match certain subsets $A_1, A_2, \ldots, A_\ell \subseteq [N]$ of size t.

In the $t = 1$ case, $|A_i| = 1$ for each $i \in [\ell]$. Thus, we're considering edges emanating from singletons. In fact, if $\mathcal{T}$ is minimal, then S contains such singletons.

For $t > 1$, it is natural to think of the alternative of S to be a collection of *subsets of* $[N]$ of size t, and similarly for H. What's another name for that? Correct, a *t-uniform hypergraph on* $[N]$.

So, we get the following natural extension of 1-rectangles: (we use the notation $\binom{S}{r}$ for a set S and a positive integer $r \leqslant |S|$ to denote the collection of all subsets of S of size r)

Definition 4.2.7 (t-rectangles)**.** Let $G \subseteq \binom{[N]}{t}$ and $H \subseteq \binom{[M]}{t}$.

We define a *t-rectangle induced by G and H* as

$$G \times_t H := \left\{T \in \mathcal{K}^t_{N,M} : \mathrm{Dom}(T) \in G, \mathrm{Im}(T) \in H\right\}.$$

That is, $G \times_t H$ collects all t-matchings with domains in G and images in H.

To attempt to generalize **Lemma 4.2.2**, we ask: when is $G \times_t H$ t-Turán?

A nice insight that was done in the proof of **Lemma 4.2.2** and is later used in **Theorem 4.2.3** is to let $R := \binom{[M]}{t} \setminus H$, and consider the relation of G with R.

Theorem 4.2.1 tells us that $G \times_t H$ is not t-Turán precisely when the following equivalent statements hold:

1. There exists some $f \in \mathcal{F}$ where every $T \subseteq f$, we have $T \notin G \times_t H$. In other words, $\mathrm{Dom}(T) \notin G$ or $\mathrm{Im}(T) \notin H$.
2. There exists some $f \in \mathcal{F}$ where every $T \subseteq f$ with $\mathrm{Dom}(T) \in G$, we must have $\mathrm{Im}(T) \in R$.
3. There exists some $f \in \mathcal{F}$ where $f(G) := \{f(A) : A \in G\} \subseteq R$.

The last characterization is the definition of *G is isomorphic to a subhypergraph of R*, or *G is embeddable in R*.

In $t = 1$, this worked out nicely to $|G| \leqslant |R|$. So, we got "$G \times H$ is 1-Turán if and only if $|G| + |H| > M$."

Here, we get the following:

Lemma 4.2.8 (Rectangles that are t-Turán)**.** Let $G \subseteq \binom{[N]}{t}$ and $H \subseteq \binom{[M]}{t}$.

Then, $G \times_t H$ is t-Turán if and only if G is not embeddable into the complement of H (i.e., G is not isomorphic to a subhypergraph of $\binom{[M]}{t} \setminus H$).

Proof. The three statements 1, 2, 3 above are all equivalent to $G \times_t H$ is not t-Turán. Thus, the statement follows from **Theorem 4.2.1** and some definition-chasing. □

Now, we turn to generalize **Theorem 4.2.3**. The goal is to give a generalization of Hall's Theorem and see if it holds. If such a theorem holds, it would give us a nice characterization of t-Turán sets.

So, we ask the question: are minimal t-Turán sets t-rectangles? This question is equivalent to saying the following:

- For every t-Turán set $\mathcal{T}$, there exists t-uniform hypergraphs G and H on $[N]$ and $[M]$ respectively, where
  1. $G \times_t H \subseteq \mathcal{T}$ and
  2. G is not embeddable into the complement of H.

This characterizes t-Turán sets as extensions of t-rectangles. However, to characterize t-Turán sets it suffices to characterize t-perfect sets.

In $t = 1$, the characterization was Hall's Theorem, and we used $H = [M] \setminus N_{\mathcal{L}}(S)$ to construct our rectangle, and the property that followed was $G \times H \subseteq \mathcal{T}$.

So, to define $N_{\mathcal{L}}(G)$ for a t-uniform hypergraph G on $[N]$, it suffices to choose $N_{\mathcal{L}}(G)$ to be such that $G \times_t \left(\binom{[M]}{t} \setminus N_{\mathcal{L}}(G) \right) \subseteq \mathcal{T}$.

In other words, any t-matching in $\mathcal{T}$ with a domain in G must have an image that is not in $N_{\mathcal{L}}(G)$. Thus, the natural generalization is

$$N_{\mathcal{L}}(G) := \{\mathrm{Im}(T) : T \in \mathcal{L}, \mathrm{Dom}(T) \in G\}.$$

This also is very intuitive. Looking at the grid view, we can imagine the following:

1. In $t = 1$, you pick one point $a \in [N]$, and you consider all vertices $v \in \mathcal{L}$ that project onto a (i.e., $v_x = a$), then unite their projections on $[M]$, you get the neighborhood of a.
2. In $t = 2$, you pick an edge $e \in \binom{[N]}{2}$, you consider all edges in $\mathcal{L}$ that project onto e, and unite their projects on $[M]$ to get the neighborhood of e

The idea for $t > 2$ generalizes the same way. **Figure 4.8** gives us a way to visualize this idea on a 3×6 grid.

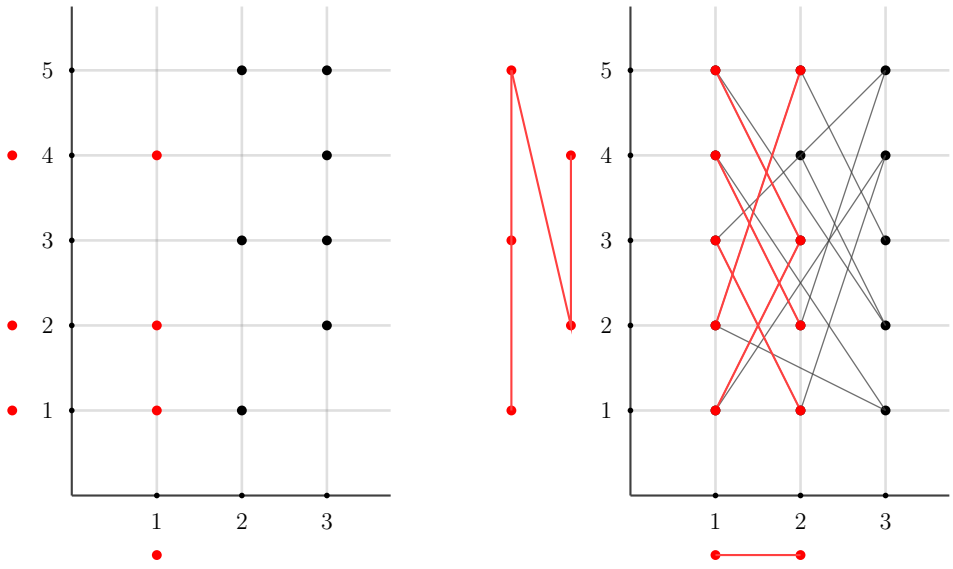


Figure 4.8: Visualizing Neighborhoods for $t = 1$ and $t = 2$.

We obtain the neighborhood of $G \subseteq \binom{[N]}{t}$ by uniting the neighborhoods of the edges inside it.

Thus, we get the following smooth generalization of **Theorem 4.2.3**:

Theorem 4.2.9. The following are equivalent:

1. For every nonempty t-Turán set $\mathcal{T} \subseteq \mathcal{K}^t_{N,M}$, if $\mathcal{T}$ is minimal, then $\mathcal{T}$ is a t-rectangle.
2. For every nonempty $\mathcal{L} \subseteq \mathcal{K}^t_{N,M}$, $\mathcal{L}$ is t-perfect if and only if for every t-uniform hypergraph G on $[N]$, G is embeddable into $N_{\mathcal{L}}(G)$.

Proof. The forward direction of 2 is trivially true, since if there is a clique in $\mathcal{L}$ of size N, then every t-uniform hypergraph G on $[N]$ will have a subgraph of the clique projecting onto it, and therefore G will be embeddable into $N_{\mathcal{L}}(G)$. We show an example in **figure 4.9**.

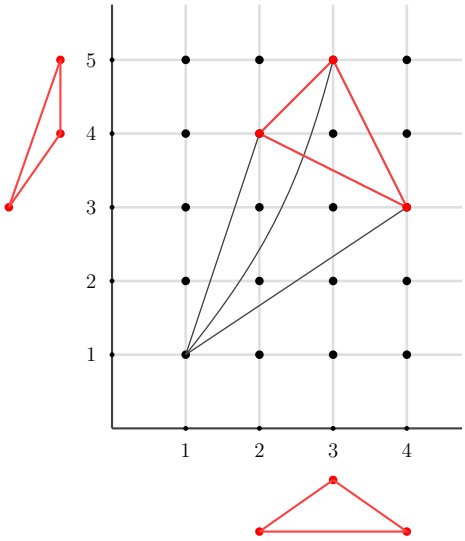


Figure 4.9: Forward direction of 2.

So, we only show that the equivalence for the backward direction of 2.

To that end, we show that if $\mathcal{L}$ is not t-perfect, then there must exist some t-uniform hypergraph G on $[N]$ where G is not embeddable into $N_{\mathcal{L}}(G)$. Let $\mathcal{T} = \mathcal{K}^t_{N,M} \setminus \mathcal{L}$.

By **Theorem 4.2.1**, $\mathcal{T}$ is t-Turán. Then, by **Lemma 4.2.8** for some $G \subseteq \binom{[N]}{t}, H \subseteq \binom{[M]}{t}$ where G is not embeddable into the complement of H we have $G \times_t H \subseteq \mathcal{T}$.

Consequently, $N_{\mathcal{L}}(G) \subseteq \binom{[M]}{t} \setminus H$ and we get that G is not embeddable into $N_{\mathcal{L}}(G)$.

Conversely, suppose 2. Let $\mathcal{T}$ be any t-Turán set. Let $\mathcal{L} = \mathcal{K}^t_{N,M} \setminus \mathcal{T}$. Then, $\mathcal{L}$ is not t-perfect.

Thus, there exists some t-uniform hypergraph G on $[N]$ where that is not embeddable into $N_{\mathcal{L}}(G)$. Take $H = \binom{[M]}{t} \setminus N_{\mathcal{L}}(G)$. By **Lemma 4.2.8**, $G \times_t H$ is t-Turán.

Furthermore, by definition of $N_{\mathcal{L}}(G)$ and H, we must have $G \times_t H \subseteq \mathcal{T}$. By minimality of $\mathcal{T}$, we conclude $G \times_t H = \mathcal{T}$ as desired. □

As much as we would like it to be, it turns out that the generalization of Hall's in 2 is not true:

Lemma 4.2.10 (Generalization to Hall's Theorem is False)**.** Suppose $1 < t < N < M$. Then, there exists $\mathcal{L} \subseteq \mathcal{K}^t_{N,M}$ such that

1. Every t-uniform hypergraph G on $[N]$ is embeddable into its neighborhood $N_{\mathcal{L}}(G)$.
2. $\mathcal{L}$ is not t-perfect.

Proof. The main issue is that the projections leave out the specific ordering.

For example, if you take $A \in \binom{[N]}{t}$ and $B \in \binom{[M]}{t}$ to be two t-hyperedges on $[N]$ and $[M]$ respectively, there are $t!$ possible $T \in \mathcal{K}^t_{N,M}$ with $\mathrm{Dom}(T) = A$ and $\mathrm{Im}(T) = B$.

This allows us to construct a powerful counterexample for $\mathcal{L}$ that makes $N_{\mathcal{L}}(G) = \binom{[M]}{t}$ for every nonempty $G \subseteq \binom{[N]}{t}$ as follows: (working with $t \geqslant 3$)

1. Given $A \in \binom{[N]}{t}$ and $B \in \binom{[M]}{t}$ where $A = \{a_1, a_2, \ldots, a_t\}$ with $a_1 < a_2 < \cdots < a_t$ and $B = \{b_1, b_2, \ldots, b_t\}$ with $b_1 < b_2 < \cdots < b_t$, define $T_{A,B} \in \mathcal{K}^t_{N,M}$

   $$T_{A,B}(a_i) := b_i \text{ for all } i \in [t].$$

   In other words, $T_{A,B}$ assigns A to B but in an ordered manner.
2. Let $\mathcal{L}' := \left\{T_{A,B} : A \in \binom{[N]}{t}, B \in \binom{[M]}{t}\right\}$.
3. Let $\mathcal{L}'' := \mathcal{L}' \setminus \{T \in \mathcal{L}' : \mathrm{Dom}(T) = \{2, 3, \ldots, t+1\}\}$.
4. Define for $B \in \binom{[M]}{t}$ where $B = \{b_1, b_2, \ldots, b_t\}$ with $b_1 < b_2 < \cdots < b_t$ the assignment $T'_B \in \mathcal{K}^t_{N,M}$ where $T(3) = b_1, T(4) = b_2, \ldots, T(t+1) = b_{t-1}$ and $T(2) = b_t$.

   So, T'_B is $T_{\{2,3,\ldots,t+1\},B}$ but cyclically shifted.
5. Set $\mathcal{L} := \mathcal{L}'' \cup \left\{T'_B : B \in \binom{[M]}{t}\right\}$.

First, observe that for any $A \in \binom{[N]}{t}$, either

1. $A \neq \{2, 3, \ldots, t+1\}$, so $N_{\mathcal{L}}(A) = \left\{\mathrm{Im}(T_{A,B}) : B \in \binom{[M]}{t}\right\} = \binom{[M]}{t}$ or
2. $A = \{2, 3, \ldots, t+1\}$, so $N_{\mathcal{L}}(A) = \left\{\mathrm{Im}(T'_B) : B \in \binom{[M]}{t}\right\} = \binom{[M]}{t}$.

Thus, $N_{\mathcal{L}}(G) = \binom{[M]}{t}$ for any nonempty G, making G easily embeddable into $N_{\mathcal{L}}(G)$.

Now, suppose $\mathcal{L}$ is t-perfect. Then, there exists $f \in \mathcal{F}$ with every $T \subseteq f$ in $\mathcal{L}$.

In particular, this happens for T where

1. $\mathrm{Dom}(T) = \{1, 2, \ldots, t-1, t+1\}$ implying $f(2) < f(t+1)$ and

2. $\text{Dom}(T) = \{2, 3, \ldots, t+1\}$ implying $f(t+1) < f(2)$.

This gives us a contradiction, and thus no such $f \in \mathcal{F}$ exists and $\mathcal{L}$ is not t-perfect.

It remains to consider $t = 2$. In this case, orient the pairs in $\binom{[N]}{2}$ such that

$$1 \to 2, \qquad 2 \to 3, \qquad 3 \to 1,$$

and orient all other pairs arbitrarily.

For every $A = \{u, v\} \in \binom{[N]}{2}$ and $B = \{a, b\} \in \binom{[M]}{2}$ where $a < b$, let $T_{A,B}$ map the tail of the oriented pair A to a and its head to b. Define

$$\mathcal{L} := \left\{ T_{A,B} : A \in \binom{[N]}{2}, B \in \binom{[M]}{2} \right\}.$$

Clearly, $N_{\mathcal{L}}(A) = \binom{[M]}{2}$ for every $A \in \binom{[N]}{2}$, and thus $N_{\mathcal{L}}(G) = \binom{[M]}{2}$ for every nonempty G. Therefore, every G is embeddable into $N_{\mathcal{L}}(G)$.

On the other hand, if $\mathcal{L}$ is 2-perfect, then there exists $f \in \mathcal{F}$ with every $T \subseteq f$ in $\mathcal{L}$. In particular, the three oriented pairs above imply

$$f(1) < f(2), \qquad f(2) < f(3), \qquad f(3) < f(1),$$

which is a contradiction. Thus, $\mathcal{L}$ is not 2-perfect. □

So, even though the attempt to generalize fails, this gives some the main insight/technique that characterizing t-Turán sets is equivalent to characterizing t-perfect sets.

For example, minimal 2-Turán sets are exactly complements of maximal K_N-free graphs on the grid points of $[N] \times [M]$ where an edge doesn't connect two vertices of the same row or column.

So, if a Hall-like characterization exists on t-perfect sets, this would tell us a lot about t-Turán sets.

Furthermore, counterexamples like the one in **Lemma 4.2.10** serve as interesting examples of t-Turán sets. So, taking the complement of $\mathcal{L}$ in **Lemma 4.2.10** gives us a t-Turán set $\mathcal{T}$ where every t-rectangle inside it is not t-Turán.

We are not sure if there are other generalizations of Hall that give us meaningful characterizations of t-Turán sets. The complexity of jumping from $t = 1$ to $t = 2$ (points to edges) lies in a theme of connectivity that is needed to be enforced in 2-perfect sets as shown in **Lemma 4.2.10**, as normal Hall's would give us a matching of edges from K_N to $N_{\mathcal{L}}(K_N)$ but the matched edges do not necessarily form a clique.

We do know that if we're seeking simulatability from one of these generalizations, then we will fail. The rest of this section will be dedicated to discussing why.

Recall that **property 3** was the motivation for some of the definitions here, where we take some subsets $A_1, A_2, \ldots, A_\ell \subseteq [N]$ of size t.

What happened in $t = 1$ is that for a small amount $s \leqslant \ell$, there was a large amount of functions $f \in \mathcal{F}$ where $f|_{A_i} \in \mathcal{T}$ for $i \leqslant s$.

To generalize, one desires that for some small $s \leqslant \ell$, the t-hyperedges in $\mathcal{T}$ whose projections (or domains) lie in $A_1, A_2, \ldots, A_s$ cover a large fraction of $\mathcal{F}$.

So, we want $G \subseteq \binom{[N]}{t}$ where $|G|$ is small enough and $\left|\bigcup_{T\in\mathcal{T},\mathrm{Dom}(T)\in G} \mathrm{Cube}(T)\right|$ is large. But we know that

$$\begin{aligned}\left|\bigcup_{T\in\mathcal{T},\mathrm{Dom}(T)\in G} \mathrm{Cube}(T)\right| &\leqslant \sum_{T\in\mathcal{T},\mathrm{Dom}(T)\in G} |\mathrm{Cube}(T)| \\ &= \sum_{T\in\mathcal{T},\mathrm{Dom}(T)\in G} \frac{(M-t)!}{(M-N)!} \\ &= \frac{(M-t)!}{(M-N)!} \cdot |\{T \in \mathcal{T} : \mathrm{Dom}(T) \in G\}|\end{aligned}$$

Thus,

$$\frac{\left|\bigcup_{T\in\mathcal{T},\mathrm{Dom}(T)\in G} \mathrm{Cube}(T)\right|}{|\mathcal{F}|} \leqslant \frac{|\{T \in \mathcal{T} : \mathrm{Dom}(T) \in G\}|}{M!/(M-t)!}.$$

So, we want a density condition, imagining the grid view: there are some t-hyperedges on $[N]$ where the number of t-hyperedges above them in $\mathcal{T}$ is large, for $\mathcal{T}$ to be t-Turán.

In other words, $\mathcal{T}$ has large enough edges over G to make its complement have no clique.

So, to rule out simulatability, we construct $\mathcal{T}$ that is induced by a circuit where $|\{T \in \mathcal{T} : \mathrm{Dom}(T) \in G\}|$ is small when G is small, implying $\left|\bigcup_{T\in\mathcal{T},\mathrm{Dom}(T)\in G} \mathrm{Cube}(T)\right|$ is small as well.

Theorem 4.2.11 (A circuit with two oracle gates that is not simulatable)**.** Let $\varepsilon \in (0,1)$ and $\delta \in (0,2)$ and suppose s, t are positive integers with $(s+t)^2 \leqslant N^{1-\delta}$.

Then, there exists a circuit $C \in \mathcal{C}$ with two oracle gates that is not (s,t,γ)-simulatable for any $\gamma < 1 - \left(\varepsilon N^{\delta}\right)^{-1}$.

Proof. Partition $[M] = P_1 \cup P_2 \cup \cdots \cup P_{N-1}$ as equally as possible. So,

$$\left||P_i| - \frac{M}{N-1}\right| \leqslant 1 \tag{4.2}$$

for all $i \in [N-1]$. Then, there exists some coloring circuit $\chi : [M] \to [N-1]$ where $\chi(z) = i$ where $z \in P_i$.

Now, define $C^{(\cdot)} : [N] \times [N] \to \{0,1\}$ as

$$C^f(u,v) := \mathbf{1}[u \neq v] \wedge \mathbf{1}[\chi(f(u)) = \chi(f(v))].$$

Clearly, C^f is satisfiable for every $f \in \mathcal{F}$ since $\mathrm{Im}(f)$ has N elements, forcing the existence of $z, z' \in \mathrm{Im}(f)$ with $\chi(z) = \chi(z')$ by pigeonhole.

So, C induces $\mathcal{T}$ where

$$\mathcal{T} := \{\{(u,y),(v,z)\} : \chi(z) = \chi(y), z \neq y, u \neq v\}.$$

In other words, $\mathcal{T}$ looks similar to **figure 4.10** (not drawn to scale).

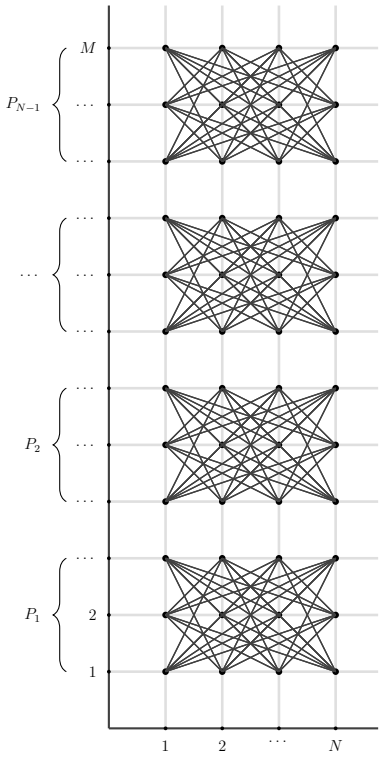


Figure 4.10: 2-Turán set of a non-simulatable circuit with two oracle gates

For $e \in \binom{[N]}{2}$, define $\mathcal{T}[e] := \{T \in \mathcal{T} : \mathrm{Dom}(T) = e\}$. Clearly, if $e = \{u,v\}$, then $\mathcal{T}[e] = \{\{(u,y),(v,z)\} : \chi(z) = \chi(y), z \neq y\}$.

In other words,

$$\begin{aligned} |\mathcal{T}[e]| &= \sum_{i \in [N-1]} 2\binom{|P_i|}{2} \leqslant \sum_{i \in [N-1]} |P_i|^2 \\ &\leqslant (N-1)\left(\frac{M}{N-1} + 1\right)^2 \leqslant \frac{M^2}{(1-\varepsilon)N}. \qquad \text{(by 4.2)} \end{aligned}$$

for a small constant $0 < \varepsilon < 1$.

Thus, for any $G \subseteq \binom{[N]}{2}$, we get

$$\begin{aligned}\frac{|\{T \in \mathcal{T} : \mathrm{Dom}(T) \in G\}|}{M(M-1)} &= \sum_{e \in G} \frac{|\mathcal{T}[e]|}{M(M-1)} \\ &\leqslant |G| \cdot \frac{M^2}{(1-\varepsilon)N} \cdot \frac{1}{M(M-1)} \\ &\leqslant \frac{|G|}{(1-\varepsilon)^2 N}.\end{aligned}$$

Thus, if $|G| \leqslant N^{1-\delta}$, we get that

$$\Pr_{f \leftarrow \mathcal{F}}[T \subseteq f \text{ for some } T \in \mathcal{T} \text{ with } \mathrm{Dom}(T) \in G] \leqslant \frac{1}{\varepsilon' N^{\delta}}.$$

Thus, if we query s points, then $s^2 \leqslant N^{1-\delta}$ implies that $\binom{s}{2} \leqslant N^{1-\delta}$ as well, and the probability of not getting a decisive satisfying assignment is overwhelming $1 - \left(\varepsilon' N^{\delta}\right)^{-1}$.[7] □

This tells us that the jump from $t = 1$ to $t = 2$ requires a deeper understanding of the structure of 2-Turán sets or a more sophisticated oracle than one that relies on simulatability.

4.3 One Round of Non-Adaptive Queries

This section aims to explicitly show that the oracle Φ provided by [FGH+23] is impossible against a starting round of polynomially many non-adaptive[8] queries, as opposed to just one. While it's a simple extension of the results of [FGH+23], we provide it explicitly for the sake of completion.

We use $w : \mathbb{N} \to \mathbb{N}$ as the polynomial number of non-adaptive queries (and we refer to it as w, not $w(n)$, when n is clear from the context).

We denote a tuple of satisfiable circuits $C_1^{(\cdot)}, C_2^{(\cdot)}, \ldots, C_w^{(\cdot)}$ by a bold face $\mathbf{C}^{(\cdot)}$. That is,

$$\mathbf{C}^{(\cdot)} = \left(C_1^{(\cdot)}, C_2^{(\cdot)}, \ldots, C_w^{(\cdot)}\right) \in \mathcal{C}_n^w.$$

For an oracle Φ, we define Φ_w componentwise by

$$\Phi_w^f(\mathbf{C}^{(\cdot)}) := \left(\Phi^f(C_1^{(\cdot)}), \Phi^f(C_2^{(\cdot)}), \ldots, \Phi^f(C_w^{(\cdot)})\right).$$

Thus, one query to Φ_w represents one round of w non-adaptive queries to Φ.

[7] We remark that adaptivity does not significantly change the argument. Reveal f only on fresh points queried by $\mathbf{S}$; conditioned on the transcript so far, the remaining values are still uniform without replacement. Thus, among at most $s+2$ exposed points, including the final output pair, a color collision occurs with probability

$$O\left(\frac{(s+2)^2}{N}\right) = O(N^{-\delta}).$$

Hence the same bound holds for adaptive simulators.

[8] the answer of one query cannot be used in the construction of another

Lemma 4.3.1 (Non-adaptive Stability)**.** Let Φ be an oracle such that for some negligible λ, every satisfiable circuit $C^{(\cdot)} \in \mathcal{C}_n$ of polynomial size and every $y \in [M]$ satisfy

$$\Pr\left[\Phi^f(C) \neq \Phi^{f_{x\to y}}(C) : \begin{matrix} f \leftarrow \mathcal{F}, \\ x \leftarrow [N] \end{matrix}\right] \leqslant \lambda(n).$$

Then, for any tuple of satisfiable circuits $\mathbf{C}^{(\cdot)} \in \mathcal{C}_n^w$,

$$\Pr\left[\Phi_w^f(\mathbf{C}^{(\cdot)}) \neq \Phi_w^{f_{x\to y}}(\mathbf{C}^{(\cdot)}) : \begin{matrix} f \leftarrow \mathcal{F}, \\ x \leftarrow [N] \end{matrix}\right] \leqslant w(n)\lambda(n).$$

Proof. Write $\mathbf{C}^{(\cdot)} = (C_1^{(\cdot)}, \ldots, C_w^{(\cdot)})$. Then, by the union bound,

$$\begin{aligned}
&\Pr\left[\Phi_w^f(\mathbf{C}^{(\cdot)}) \neq \Phi_w^{f_{x\to y}}(\mathbf{C}^{(\cdot)}) : \begin{matrix} f \leftarrow \mathcal{F}, \\ x \leftarrow [N] \end{matrix}\right] \\
&\leqslant \sum_{i=1}^{w} \Pr\left[\Phi^f(C_i^{(\cdot)}) \neq \Phi^{f_{x\to y}}(C_i^{(\cdot)}) : \begin{matrix} f \leftarrow \mathcal{F}, \\ x \leftarrow [N] \end{matrix}\right] \\
&\leqslant \sum_{i=1}^{w} \lambda(n) \\
&\leqslant w(n)\lambda(n)
\end{aligned}$$

as desired. □

So, almost t-stability implies non-adaptive almost t-stability. How do we achieve almost t-stability? **Theorem 3.2.4** says it suffices to find an exhaustive search oracle that leaks every y only with negligible probability.

Lemma 4.3.2. Suppose $M = 2N$ and $C \in \mathcal{C}$. There exists an exhaustive search oracle Φ such that for every $y \in [M]$,

$$\Pr_{f\leftarrow\mathcal{F}}\left[\Phi^f(C) \text{ leaks } y\right] \leqslant \frac{6t}{\sqrt{N}}.$$

Proof. This follows from the construction of the exhaustive search oracle in [FGH+23]. □

Corollary 4.3.3. There exists an exhaustive search oracle Φ that is almost $48t\sqrt{N}$-stable.

Proof. By the previous lemma and $M = 2N$,

$$\Pr_{f\leftarrow\mathcal{F}}\left[\Phi^f(C) \text{ leaks } y\right] \leqslant \frac{6t}{\sqrt{N}} = \frac{12t\sqrt{N}}{M}.$$

Furthermore, since Φ is an exhaustive search oracle, it has t-determining sets, and hence also $12t\sqrt{N}$-determining sets. Thus, **Theorem 3.2.4** implies that Φ is almost $48t\sqrt{N}$-stable. □

Thus, we can show this theorem directly:

Theorem 4.3.4 (Non-adaptive Impossibility)**.** There exists an oracle Φ such that for every reduction $\mathbf{R}$ of runtime at most t that submits one starting round of non-adaptive queries to Φ^f followed up by queries to f we have

$$\Pr\left[\mathbf{R}^{f,\Phi^f}(f(x)) = x : \begin{matrix} f \leftarrow \mathcal{F} \\ x \leftarrow [N] \end{matrix}\right] \leqslant \lambda(n)$$

for some negligible function λ.

Proof. By **Theorem 3.3.3**, it suffices to show that the probability of instability is low on all queries made by $\mathbf{R}$.

If we fix $y \in [M]$ and the randomness of $\mathbf{R}$, then $\mathbf{R}^{f,\Phi^f}(y)$ submits the same non-adaptive round of queries $\mathbf{C}$ for all $f \in \mathcal{F}$ with $y \in \mathrm{Im}(f)$.

By **Corollary 4.3.3** and **Lemma 4.3.1**, $\mathbf{C}$ is stable (with respect to Φ).[9] This completes the proof. □

[9]There is an implicit application of Markov's inequality (**B.0.1**) to pass from negligible average instability over f to a negligible fraction of f violating the b/N-stability threshold required by **Theorem 3.3.3**.

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009. URL: http://www.cs.princeton.edu/theory/complexity/.

[BCH+22] Nir Bitansky, Arka Rai Choudhuri, Justin Holmgren, Chethan Kamath, Alex Lombardi, Omer Paneth, and Ron D. Rothblum. PPAD is as hard as LWE and iterated squaring. Cryptology ePrint Archive, Paper 2022/1272, 2022. URL: https://eprint.iacr.org/2022/1272.

[BD19] Nir Bitansky and Akshay Degwekar. On the complexity of collision resistant hash functions: New and old black-box separations. Cryptology ePrint Archive, Paper 2019/686, 2019. URL: https://eprint.iacr.org/2019/686.

[BH13] Kfir Barhum and Thomas Holenstein. A cookbook for black-box separations and a recipe for uowhfs. In Amit Sahai, editor, *Theory of Cryptography*, pages 662–679, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[BPR14] Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a nash equilibrium. Cryptology ePrint Archive, Paper 2014/1029, 2014. URL: https://eprint.iacr.org/2014/1029.

[Das19] Constantinos Daskalakis. Equilibria, fixed points, and computational complexity. In *Proceedings of the International Congress of Mathematicians (ICM)*. World Scientific, 2019. doi:10.1142/9789813272880_0009.

[DN07] Cynthia Dwork and Moni Naor. Zaps and their applications. *SIAM J. Comput.*, 36(6):1513–1543, 2007.

[FGH+23] Lukas Folwarczny, Mika Göös, Pavel Hubacek, Gilbert Maystre, and Weiqiang Yuan. One-way functions vs. TFNP: Simpler and improved. Cryptology ePrint Archive, Paper 2023/945, 2023. URL: https://eprint.iacr.org/2023/945.

[Fis12] Marc Fischlin. Black-box reductions and separations in cryptography. In Aikaterini Mitrokotsa and Serge Vaudenay, editors, *Progress in Cryptology - AFRICACRYPT 2012*, pages 413–422, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[For94] Lance Fortnow. The role of relativization in complexity theory, 1994.

[GHJ+22] Mika Göös, Alexandros Hollender, Siddhartha Jain, Gilbert Maystre, William Pires, Robert Robere, and Ran Tao. Further collapses in TFNP, 2022. URL: https://arxiv.org/abs/2202.07761, arXiv:2202.07761.

[GJ86] J. Goldsmith and D. Joseph. Three results on the polynomial isomorphism of complete sets. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, pages 390–397, New York, 1986. IEEE.

[GP17] Paul W. Goldberg and Christos H. Papadimitriou. Tfnp: An update. In Dimitris Fotakis, Aris Pagourtzis, and Vangelis Th. Paschos, editors, *Algorithms and Complexity*, pages 3–9, Cham, 2017. Springer International Publishing.

[GPS15] Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. Revisiting the cryptographic hardness of finding a nash equilibrium. Cryptology ePrint Archive, Paper 2015/1078, 2015. URL: https://eprint.iacr.org/2015/1078.

[GT00] R. Gennaro and L. Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 305–313, 2000.

[HHRS07] I. Haitner, J. J. Hoch, O. Reingold, and G. Segev. Finding collisions in interactive protocols – a tight lower bound on the round complexity of statistically-hiding commitments. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 669–679, 2007.

[HKKS20] Pavel Hubáček, Chethan Kamath, Karel Král, and Veronika Slívová. On average-case hardness in TFNP from one-way functions. Cryptology ePrint Archive, Paper 2020/1162, 2020. URL: https://eprint.iacr.org/2020/1162.

[HNY17] P. Hubáček, M. Naor, and E. Yogev. The journey from NP to TFNP hardness. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, pages 60:1–60:21, Berkeley, CA, USA, jan 2017.

[Hol21] Alexandros Hollender. *Structural results for total search complexity classes with applications to game theory and optimisation*. PhD thesis, University of Oxford, 2021. URL: https://ora.ox.ac.uk/objects/uuid:67e2d80b-76bf-4b49-9b7d-8bbd91633dd7.

[HV21] Pavel Hubáček and Jan Václavek. On search complexity of discrete logarithm, 2021. URL: https://arxiv.org/abs/2107.02617, arXiv:2107.02617.

[HY17] Pavel Hubáček and Eilam Yogev. Hardness of continuous local search: Query complexity and cryptographic lower bounds. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*, pages 1352–1371, Barcelona, Spain, jan 2017.

[Jeř16] Emil Jeřábek. Integer factoring and modular square roots. *Journal of Computer and System Sciences*, 82(2):380–394, March 2016. URL: http://dx.doi.org/10.1016/j.jcss.2015.08.001, doi:10.1016/j.jcss.2015.08.001.

[JPY85] David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, SFCS '85, pages 39–42, USA, 1985. IEEE Computer Society. doi:10.1109/SFCS.1985.31.

[KSS11] Jeff Kahn, Michael E. Saks, and Clifford D. Smyth. The dual BKR inequality and Rudich's conjecture. *Combinatorics, Probability & Computing*, 20(2):257–266, 2011.

[MF21] Arno Mittelbach and Marc Fischlin. *The Theory of Hash Functions and Random Oracles: An Approach to Modern Cryptography*. Information Security and Cryptography. Springer Cham, 1 edition, 2021. doi:10.1007/978-3-030-63287-8.

[MP91] Nimrod Megiddo and Christos H. Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, 81(2):317–324, 1991.

[Nas51] John Nash. Non-cooperative games. *The Annals of Mathematics*, 54(2):286–295, 1951.

[Pap94] Christos H. Papadimitriou. On the complexity of parity arguments and other inefficient proofs of existence. In *Proceedings of the 11th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 775 of *Lecture Notes in Computer Science*, pages 23–37. Springer, 1994. doi:10.1007/3-540-57785-8_57.

[Pik25] Oleg Pikhurko. Constructions of turán systems that are tight up to a multiplicative constant, 2025. URL: https://arxiv.org/abs/2406.07443, arXiv:2406.07443.

[Pud14] Pavel Pudlák. On the complexity of finding falsifying assignments for herbrand disjunctions, 2014. URL: https://arxiv.org/abs/1411.3304, arXiv:1411.3304.

[RPBD21] Alexander Razborov, Pavel Pudlák, and Shai Ben-David. SAT and foundations of mathematics. Talk at the workshop "Theoretical Foundations of SAT/SMT Solving", April 2021. Online via Zoom. URL: https://simons.berkeley.edu/talks/sat-foundations-mathematics.

[RS08] Alon Rosen and Gil Segev. Chosen-ciphertext security via correlated products. Cryptology ePrint Archive, Paper 2008/116, 2008. URL: https://eprint.iacr.org/2008/116.

[RSS16] Alon Rosen, Gil Segev, and Ido Shahaf. Can PPAD hardness be based on standard cryptographic assumptions? Cryptology ePrint Archive, Paper 2016/375, 2016. URL: https://eprint.iacr.org/2016/375.

[RTV04] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In *Theory of Cryptography, First Theory of*

Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings, pages 1–20, 2004.

[Rud89] Steven Rudich. *Limits on the Provable Consequences of One-way Functions*. PhD thesis, University of California at Berkeley, 1989.

[Sim98] Daniel R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In Kaisa Nyberg, editor, *Advances in Cryptology — EUROCRYPT'98*, pages 334–345, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[Sip96] Michael Sipser. Introduction to the theory of computation. *ACM Sigact News*, 27:27–29, 1996.

APPENDICES

Appendix A

Notations and Terminology

We provide the notation and/or terminology in the order it appears in the text (starting from **Chapter 2**).

- We use
  - sans-serif font to denote complexity classes (e.g., NP, FNP, TFNP),
  - bold face font to denote algorithms (e.g., $\mathbf{R}, \mathbf{S}, \mathbf{A}$), and
  - small caps to denote problems or languages (e.g. PIGEON, INVERT).
- We adopt the convention that natural numbers include 0. (i.e., $\mathbb{N} = \{0, 1, 2, \ldots\}$), and we use $\mathbb{N}^+$ to denote positive integers (i.e., $\mathbb{N}^+ = \mathbb{N} \setminus \{0\}$).
- For an alphabet A, we use $A^* := \bigcup_{n \in \mathbb{N}} A^n$ to denote all finite strings consisting of symbols from A.
- We use bars $|\cdot|$ to denote the length of a string (e.g., $|x|, |y|$).
- We denote the lengths of the input and output bits of a circuit C by defining it using the notation of functions (e.g., $C : \{0,1\}^n \to \{0,1\}^m$ denotes a circuit with n input bits and m output bits).
- We use $\langle \cdot \rangle$ to denote the encoding of an object into binary. So, $\langle G \rangle$ is the encoding of a graph, $\langle C \rangle$ is the encoding of a circuit.

  We assume encodings are done with respect to known conventions so that the object can be parsed efficiently by the algorithm (e.g. graphs can be encoded as adjacency matrices).

  Furthermore, we assume $\perp$ is a unique recognizable character with respect to such encodings.
- We use Φ to denote an oracle given to an algorithm. To denote that an algorithm $\mathbf{R}$ has an access to an oracle, we sometimes refer to it as $\mathbf{R}^{(\cdot)}$ where $(\cdot)$ is a placeholder for the oracle.

  The same convention is used with circuits with oracle gates (i.e., $C^{(\cdot)}$ denotes an oracle circuit, where for a function f, C^f is a circuit with oracle gates for f).

- We use $[k]$ to denote $\{1, 2, \ldots, k\}$ the set of positive integers up to k.
- We treat functions (and partial functions) with their set-theoretic definition as a set of pairs.

  So, it is natural to say $T \subseteq f$ for a partial assignment T and a function f to say f agrees with T on $\mathrm{Dom}(T)$. Details are provided below:

  - A *relation* R from set A to set B is a subset of $A \times B$ (i.e. $R \subseteq A \times B$).
  - We use

$$\mathrm{Dom}(R) := \{x \in A : (x, y) \in R \text{ for some } y \in B\} \text{ and}$$
$$\mathrm{Im}(R) := \{y \in B : (x, y) \in R \text{ for some } x \in A\}.$$

  - A *partial function* (or *partial assignment*) f from A to B (denoted $f : A \rightharpoonup B$) is a relation from A to B where whenever $(x, y) \in f$ and $(x, y') \in f$, we must have $y = y'$ for any $x \in A$ and $y, y' \in B$. We use $f(x)$ to denote this common value.
  - A partial function f from A to B is said to be a *function* (denoted $f : A \to B$) if for every $x \in A$, there exists $y \in B$ where $(x, y) \in f$. Every partial function is a function if defined on its domain.
  - For $f : A \to B$ and $C \subseteq A$, we define the partial function $f|_C := \{(x, y) \in f : x \in C\}$ to be the restriction of f on C
- We say that a function $\varepsilon : \mathbb{N}^+ \to \mathbb{R}_{\geqslant 0}$ is *negligible* if for every $c \in \mathbb{N}^+$ there exists $n_0 \in \mathbb{N}^+$ where $\varepsilon(n) \leqslant n^{-c}$ for all $n \geqslant n_0$.
- For a finite set X, the notation $x \leftarrow X$ describes sampling x from X uniformly at random. For a distribution $\mathcal{D}$, the notation $x \leftarrow \mathcal{D}$ describes sampling x according to the distribution $\mathcal{D}$.
- For a randomized algorithm $\mathbf{A}$, we frequently write $\Pr_{\mathbf{A}}[\mathbf{A}(\cdot)]$ instead of $\Pr_{r \leftarrow R_{\mathbf{A}}}[\mathbf{A}(r, \cdot)]$ where $R_{\mathbf{A}}$ is the space of randomness of $\mathbf{A}$. We also occasionally eliminate the subscript if clear from the context.
- A function $f : \{0,1\}^* \to \{0,1\}^*$ is *one-way* if and only if
  1. there exists a polynomial time Turing machine $\mathbf{M}_f$ where $\mathbf{M}_f(x) = f(x)$ for all $x \in \{0,1\}^*$, and
  2. for any PPT algorithm $\mathbf{A}$ and there exists a negligible ε where for any $n \in \mathbb{N}$,

$$\Pr_{\mathbf{A}, x \leftarrow \{0,1\}^n}[f(\mathbf{A}(f(x), 1^n)) = f(x)] \leqslant \varepsilon(n).$$

- A family of functions $\mathcal{J}$ is a set $\mathcal{J} \subseteq \{0,1\}^{*\{0,1\}^*}$ where every $g \in \mathcal{J}$ is a function $g : \{0,1\}^* \to \{0,1\}^*$.

  For a family of functions $\mathcal{J}$, we denote $\mathcal{J}_n := \left\{f|_{\{0,1\}^n} : f \in \mathcal{J}\right\}$.

  Thus, $g \in \mathcal{J}_n$ is a function $g : \{0,1\}^n \to \{0,1\}^*$.

  Observe that for two families of functions $\mathcal{Q}$ and $\mathcal{W}$, $\mathcal{Q} \subseteq \mathcal{W}$ if and only if $\mathcal{Q}_n \subseteq \mathcal{W}_n$

for every n.

- We define the family of functions

$$\mathcal{B} := \{f : \{0,1\}^* \to \{0,1\}^* \mid f \text{ is polynomially-bounded and length-regular}\}.$$

  Here, a function $f : \{0,1\}^* \to \{0,1\}^*$ is polynomially bounded if there exists a constant $c \in \mathbb{N}^+$ where $|f(x)| \leqslant |x|^c$ for all $x \in \{0,1\}^*$.

  Furthermore, f is length-regular if there exists a function $\ell : \mathbb{N} \to \mathbb{N}$ where $f(\{0,1\}^n) \subseteq \{0,1\}^{\ell(n)}$ for all n.

- For a partial assignment $T : [N] \to [M]$, $\text{Cube}(T) := \{f \in \mathcal{F} : T \subseteq f\}$

Appendix B

Known Theorems

This lists all the known theorems/results from probability theory.

Theorem B.0.1 (Markov's Inequality)**.** Let X be a non-negative random variable where $\mathbb{E}[X] > 0$ and $t > 0$, then

$$\Pr\left[X \geqslant t \cdot \mathbb{E}[X]\right] \leqslant \frac{1}{t}.$$

Proof. Let $\mu := \mathbb{E}[X]$. Then,

$$\begin{aligned}\mu &= \mathbb{E}[X] \\ &= \mathbb{E}\left[X \mid X \geqslant t\mu\right] \cdot \Pr[X \geqslant t\mu] + \mathbb{E}\left[X \mid X < t\mu\right] \Pr[X < t\mu] \\ &\geqslant t\mu \cdot \Pr[X \geqslant t\mu] + 0.\end{aligned}$$

Dividing by $t\mu$ on both sides since $\mu > 0$ gives the desired conclusion. □

Theorem B.0.2 (Hall's Theorem)**.** Let $\mathcal{L} \subseteq X \times Y$ be a bipartite graph. For $S \subseteq X$, define $N_{\mathcal{L}}(S) := \{y \in Y : (x, y) \in \mathcal{L} \text{ for some } x \in S\}$.

Then, the following are equivalent:

1. X can be matched into Y (i.e., there exists $f \subseteq \mathcal{L}$ where $f : X \to Y$ is an injective function).
2. For every $S \subseteq X$, $|S| \leqslant |N_{\mathcal{L}}(S)|$ (We call this Hall's condition).

Proof. The forward direction is simple: if a matching f exists, then $|S| = |f(S)| \leqslant |N_{\mathcal{L}}(S)|$ as $f(S) \subseteq N_{\mathcal{L}}(S)$.

We show the backward direction by inducting on $|X|$, with base case $|X| = 1$ clearly true since $|N_{\mathcal{L}}(X)| \geqslant 1$ so we can match the single vertex in X.

If some nonempty proper $S \subsetneq X$ with $|S| = |N_{\mathcal{L}}(S)|$ exists, then we can match $S \to N_{\mathcal{L}}(S)$ and $X \setminus S \to Y \setminus N_{\mathcal{L}}(S)$ separately using the induction hypothesis as Hall's condition holds on each of them. Thus, assume $|S| < |N_{\mathcal{L}}(S)|$ for all $S \subsetneq X$.

Pick any vertex $x^\star \in X$ and since $N_{\mathcal{L}}(\{x^\star\})$ must have at least one element, then we can match it into $y^\star \in Y$.

We want to show that Hall's condition holds on $\mathcal{L}' := \{(x, y) \in \mathcal{L} : x \neq x^\star, y \neq y^\star\}$ and then use the induction hypothesis to get our conclusion.

Take any $S \subseteq X \setminus \{x^\star\}$. Observe that $N_{\mathcal{L}}(S) \subseteq N_{\mathcal{L}'}(S) \cup \{y^\star\}$. This implies $|N_{\mathcal{L}'}(S)| \geqslant |N_{\mathcal{L}}(S)| - 1 \geqslant |S|$ as S is a proper subset of X, as desired. $\square$