Randomness and Quantumness in Space-Bounded Computation

Wei Zhan

a dissertation
presented to the faculty
of princeton university
in candidacy for the degree
of Doctor of Philosophy

recommended for acceptance
by the Department of
Computer Science

Adviser: Ran Raz

September 2023

Abstract

In the field of computational complexity theory, we study the power and limits of different computational resources and the interplay between them. The constraints on space complexity provide a natural and interesting setting, that is often more tractable than the time-restricted counterparts. In this dissertation, we specifically study how randomness and quantumness interact with space complexity.

Our results consist of two parts. In the first part, we present our algorithmic results. We show that randomness used for BPL algorithms can be reduced to logarithmic with the access to untrusted random bits. Consequentially, every BPL algorithm can be certifiably derandomized using presumably hard functions. For quantum computing, we show how to eliminate intermediate measurement in logspace quantum circuits, and simulate general quantum algorithms in BQL with only unitaries.

In the second part, we present our lower bound results. For decision problems, we propose the coupon-collector model where one receives random coordinates of the input, and prove a quadratic time-space tradeoff lower bound in the model. For computing multi-output functions, we prove the first polynomial separation between randomized and deterministic oblivious computation for total functions. And for learning, we prove an exponential time lower bound against classical-quantum hybrid learners with sub-quadratic classical memory and sublinear quantum memory.

Contents

To the light of my life.

Acknowledgments

First and foremost, I want to express my gratitude to my PhD adviser, Ran Raz, who patiently walked me through the basics in my junior years when I was weak in the fundamentals, who always encouraged me on my unorthodox research ideas while being very careful about the correctness, and who was extremely forgiving of my chaotic sleep schedules and my poor memory for not appearing in meetings. His extensive knowledge, rigorous attitude and easygoing personality has positively influenced me throughout the years. I am so indebted, yet so honored to have Ran as my mentor.

I was fortunate to collaborate with some of the most brilliant researchers during my PhD, including Sumegha Garg, Uma Girish, Justin Holmgren, Qipeng Liu, Kunal Mittal, Edward Pyne, Ran Raz and Huacheng Yu. I want to thank Jian Li and Seth Pettie for introducing me to the theory world in my undergraduate years, and also Zeev Dvir and Emmanuel Abbe for guiding me in my first year when I was unclear about what type of research I want to do. I cannot overstate how much I learned from discussing with them, which not only led to the results presented in this dissertation, but overall helped me find the direction I like. Besides my collaborators and advisors, there are countless other people who generously shared their knowledge with me via in-person discussions, emails or online forums. I could not list all of you here, but I am sincerely grateful for your inputs.

I want to thank the computer science department and the theory group, including the staff, the faculty and the students, for creating such a wonderful working environment. In particular, I am thankful to Nicki Mahler and Mitra Kelly for their prompt support on administrative issues; to Mark Braverman, Gillat Kol, Ran Raz, Zeev Dvir and Huacheng Yu for taking their time being my committee members; and to Fermi Ma and Clayton Thomas for helping me when I was in need.

I am grateful to Princeton University for offering me the Francis Robbins Upton Fellowship. My research was also funded by Ran's grants from Simons foundation and NSF.

The journey would not be so fun without the help and supports from my friends. As a partial list, let me share my thanks to Zhiyuan Li and Yuping Luo, who tried out the dishes I made; to Jiaxin Guan, who shared with me the dishes he made; and to Runzhe Yang, who collaborated dishes with me. As well as thanks to Dingli Yu for letting me petting his dog, and to Xiaoqi Chen for his constant Chanel shipments. Special thanks to the friends I met in the Werewolf board games, where I spent some of the most enjoyable Saturday nights.

Finally, I want to give my heartfelt thanks to two of my lifelong friends: Kody Wang and Zui Tao, along with my parents and my cousin, for their continued company and support.

1

Introduction

How much computational resources do we need for a computation task? This is the fundamental problem in computational complexity theory, where the resources of interest vary from model to model, including but not limited to time, space, randomness, communication with other parties, etc. Usually we could utilize multiple dimensions of resources which demonstrate interesting interplay: sometimes there is a smooth tradeoff between two resources, and other times one overshadows another. Studying these phenomena not only provides insights on how to optimize resource allocation for different tasks in practice, but also has great theoretical value for understanding the roles played by these resources in computation.

Ever since the introduction of computational complexity [HS65], a majority of the efforts from theorists have been made in exploring the interactions between different resources and time complexity. Indeed, the most influential open problems in classical complexity theory, derandomization and quantum computing respectively, P versus NP, BPP

and BQP, can be viewed as the questions of whether nondeterminism, randomness and quantum mechanics enhance the power of time-bounded computation. Decades of research lead to significant advancements in our knowledge on these questions (some of which we will mention in the corresponding parts of this dissertation), yet it is still widely open whether any of NP, BPP or BQP could be equal to P or much larger classes, say, EXP.

In comparison, despite it receiving relatively less attention, we still saw plenty of results proved exclusively for space-bounded computation, especially the algorithmic ones showing that certain resources can efficiently simulate others. Generally speaking, the space constraint provides extra structures that are useful in designing simulation algorithms. And specifically for logspace computations, operations like going over all possible configurations (which in general would take exponential time) are enabled. For instance, many computational resources can be replaced by a quadratic increase in space, and as a result the space-counterpart of the aforementioned classes, NL, BPL and BQL, are all easily shown to be contained in L^2. In fact, much more are known about these space-bounded classes, and people are quite optimistic about the resolution of some fundamental problems such as L vs BPL. We will review the developments on related topics later in this chapter.

On the other hand, to show that certain resources are essential and cannot be replaced at a small cost, we need to prove corresponding lower bounds. Lower bounds results seems easier to prove with the additional space constraint, but in reality this is mostly not the case. Most of the lower bounds in space complexity we know today are proved via communication complexity, which applies only to the streaming model. For more general computation models, the same method leads to resorting to multi-party communication complexity,

where we do not know any non-trivial result for a large number of parties. Even worse, Barrington's Theorem implies that without huge breakthroughs in circuit complexity, even with extreme space constraints, we cannot expect to prove any lower bound better than some small polynomial for decision problems. Fortunately, it turns out that we can circumvent the obstacles and prove meaningful lower bounds by modifying the computation model, or by considering non-decision problems. But even then, there are still plentiful challenging open problems that await to be resolved.

This dissertation is devoted to the study of the above problems, in particular to understand more about the power of randomness and quantumness when interacting with space complexity. Below we briefly motivate and summarize the results included in this dissertation.

1.1 Randomness with Bounded Space

To what extend could randomness help reducing other computational resources? When the resource of interest refers to time, it is widely believed that randomness could not provide exponential speed up in general sequential computation models, and algorithms using randomness can be efficiently *derandomized*, yielding deterministic algorithms with the same functionality. Indeed, by the *hardness-vs-randomness* paradigm [Sha81, Yao82, BM84, NW94, IW97, STV01], $\mathsf{BPP} = \mathsf{P}$ under plausible cryptographic or hardness assumptions. As noted in [KvM02], the framework also works in the space-bounded regime, that $\mathsf{BPL} = \mathsf{L}$ assuming the existence of space-efficient hard functions.

Unlike the polynomial-time counterpart, there are also considerable progress towards proving *unconditional* efficient derandomization of BPL, and one can refer to [Sak96] and

[Hoz22] respectively for earlier and more recent developments in the area. Most notably, it was proved that $\mathsf{BPL} \subseteq \mathsf{L}^{3/2}$ by Saks and Zhou [SZ99], and the result was slightly improved to $\mathsf{BPL} \subseteq \mathsf{DSPACE}(\log^{3/2} n/\sqrt{\log\log n})$ by Hoza [Hoz21].

Their results are based on simpler derandomization objects such as pseudorandom generators (PRGs). Improving the constructions of these objects is the most direct approach to achieve the final goal of proving $\mathsf{BPL} = \mathsf{L}$: If we can construct a PRG with seed length $O(\log n)$ that fools BPL machines, then by going over all possible seeds we can fully derandomize BPL. Actually, it turns out that PRG is not even necessary and a hitting-set generator with seed length $O(\log n)$ suffices to imply that $\mathsf{BPL} = \mathsf{L}$ [CH22]. However, currently the best known seed length for all these objects are $O(\log^2 n)$ [Nis90, INW94, BCG18, Hoz21], which only yield the trivial $\mathsf{BPL} \subseteq \mathsf{L}^2$ when applied naively. The results in [SZ99] and [Hoz21] applied these objects in much more sophisticated ways.

In this dissertation we present several results that supplement this line of work, and practically derandomize all problems in BPL. To start with, we show that $O(\log n)$ *trusted* randomness suffices for computation in BPL. In contrast to true randomness, we allow the algorithm to use *untrusted* randomness which is supposed to be randomness but could have arbitrary distribution.

Theorem 1.1. *For every problem in* BPL*, there is a randomized logspace algorithm that uses* $O(\log n)$ *truly random bits and an unlimited number of untrusted random bits, such that:*

1. *If the untrusted random bits are perfectly random, then the algorithm must output the correct answer with high probability.*

2. *For every possibility of the untrusted random bits, even adversarially chosen after seeing*

the input, the algorithm must with high probability either output the correct answer or abort the computation.

Theorem 1.1 is formally stated as Theorem 4.1.11 and proved in Section 4.2.2. The second guarantee in Theorem 1.1 makes our algorithm literally error-free: we can use anything as the untrusted randomness, such as the digits of π, and in the worst case the computation just gets aborted with high probability. In addition, in most cases the digits of π should be irrelevant to the computation that we execute and thus look indistinguishable from perfectly random numbers, and we could hope to receive the correct answer with high probability.

After all, the digits of π are not designed to fool randomized computation, but there are things designed to do so, namely the pseudorandom generators. In particular, when we use the outputs of the PRGs based on hard functions [BM84, NW94, IW97] as the untrusted randomness, we either obtain the correct answer and successfully derandomize, or know for a fact that the PRG is distinguishable from perfect randomness when the algorithm aborts. The latter case further implies that the hard function, which the PRG is based on, is not really hard.

Theorem 1.2. *For every family of functions f decidable in linear space, with the hardness assumption that it cannot be computed by circuits of size $2^{\varepsilon n}$ for some $\varepsilon > 0$, there is a deterministic logspace process that, given a* BPL *algorithm, outputs either*

1. *The correct derandomized answer of the algorithm; Or*
2. *A small circuit computing f that refutes the hardness assumption.*

Theorem 1.2 is formally stated as Theorem 5.1.1. We can view it as a practical derandomization of BPL: For instance, we can use SAT as the hard function f whose hardness is based on the non-uniform version of Exponential Time Hypothesis (ETH) [IP99], and choose a scale of diminishing ε. Previously we believe this works because we believe in ETH, but the derandomized result cannot be fully trusted as there is no guarantee on the result in a world where ETH is false. In comparison, no matter ETH is true or not, the derandomized result in Theorem 1.2, if outputted, is always correct.

Another way to think of using PRG as untrusted randomness is that, Theorem 1.1 effectively provides an efficient way to check whether a PRG is working (for the specific derandomization instance) or not. Suppose now that an optimal PRG exists, in the form of a Turing machine which can be described by a constant number. Therefore, to derandomize a BPL algorithm, we can enumerate all Turing machines and plug in each one of them until a derandomized answer is outputted, which is guaranteed to be correct. In other words, without knowing the actual PRG, we have a explicit algorithm that universally derandomize all computation.

In fact, we can prove something even stronger: We only need to assume that $\mathsf{BPL} = \mathsf{L}$, and then such universal derandomization exists. This is formally stated as Theorem 5.3.1, and here we give a informal description.

Theorem 1.3. *There exists a explicitly described deterministic Turing machine that derandomizes every* BPL *algorithm, and runs in space $O(S)$ if and only if* $\mathsf{BPL} \subseteq \mathsf{DSPACE}(S)$.

Now let us consider a more fine-grained problem: Assuming $\mathsf{BPL} = \mathsf{L}$, how large would the derandomization overhead be? For the space usage, it is recently shown by Doron and Tell [DT23] that under proper assumptions, we can manage to blow up the space only by a

very small constant factor. What about time usage? By definition the overhead in time is at most polynomial, whereas in practice we would like the polynomial to be as small as possible. For derandomizing BPP, Chen and Tell [CT21] showed that an $O(n)$ overhead factor in time is both possible and necessary under plausible assumptions, and one could expect a similar conditional result holds for derandomizing BPL. Proving it unconditionally would be hard as we will explain in Chapter 7, and instead we can show an unconditional polynomial lower bound for derandomizing logspace computation for *multi-output* functions rather than decision problems:

Theorem 1.4. *There is a total function on n inputs and $O(n)$ outputs, such that:*

- *There exists a randomized oblivious algorithm with space $O(\log n)$, time $O(n \log n)$ and one-way access to randomness, that computes the function with high probability.*
- *Any deterministic oblivious algorithm with space S and time T that computes the function must satisfy $T^2 S \geq \widetilde{\Omega}(n^{2.5})$.*

Theorem 1.4 will be formally stated as Theorem 9.2.2. Its implies that black-box derandomization, which keeps the oblivious query pattern for any randomized oblivious algorithm, of logspace computation requires at least an $\widetilde{\Omega}(n^{1/4})$ overhead in time. An interesting open problem is that if we can prove a stronger lower bound to push the overhead to $\widetilde{\Omega}(n)$ to match the lower bound in [Wil16, CT21]. A matching linear overhead upper bound is also not completely out of scope: [Hoz19] even showed that, if we just want to reduce the number of random bits to $O(\log n)$ (for decision problems and in average-case), it can be done even with only constant overhead.

1.2 Quantum computation with Bounded Space

Unlike the case of randomness, it is widely believed that quantum mechanics provides super-polynomial speed-ups against classical computation, i.e. $\mathsf{BQP} \neq \mathsf{P}$, with candidate separating problems such as Factoring [Sho94]. The space constraints emerges naturally in quantum computing, since at the time of writing the largest-scale circuit-based quantum processor consists of only hundreds of qubits, and the number is not expected to drastically increase in the near future without theoretical breakthroughs. Therefore, near-term quantum devices by themselves have limited space, at least for the quantum part of the memory.

However, quantum computational complexity with bounded space was relatively understudied, mostly due to the confusion and complication in definition. The space-bounded quantum complexity classes, such as BQL, were first formally defined by Watrous [Wat99] via quantum Turing machines, similar to the way BQP and related time-bounded classes were defined by Bernstein and Vazirani [BV97]. There is a huge caveat in this definition though: the time-evolution of the states of the machine must be unitary, which rules out operations like intermediate measurements (measuring some qubits during the computation) or classical erasure (in general, physically feasible operations on quantum states are characterized by quantum channels). For BQP this is not a problem, since we have the principle of deferred measurements (see e.g. [NC10, Section 4.4]) and in general every quantum channel can be perceived as a unitary in a larger Hilbert space by Stinespring's dilation theorem [Sti55].

But with limited space, this becomes a severe problem. It is highly non-trivial to even simulate classical deterministic machines and show the containment $\mathsf{L} \subseteq \mathsf{BQL}$, without

using the classical reversible computation result by Lange, McKenzie and Tapp [LMT00]. Even worse, it was entirely not clear whether BPL is contained in BQL (which was raised as an open problem in [Wat99]), as the trick of simulating random bits with qubits and Hadamard gates does not work in logspace without the ability to reset qubits. Consequently, a lot of later works on space-bounded quantum computation, such as [vMW12, Ta-13, FL18], use the more general notion of BQL which allows the usage of any quantum channel, and the previous one that allows only unitaries are changed to be denoted as $\mathsf{BQ_UL}$ [FL18]. The discrepancy in notation is often confusing, and one may hope to unify these notations by showing that they are actually the same class. As a starting point, Vidick [Vid18] explicitly asked whether intermediate measurements could be eliminated without increasing the space too much.

Vidick's question was answered positively in [GRZ21b], which showed that in general all *unital* quantum channels can be simulated with unitary quantum computers with only constant blow-up in space. These channels include intermediate measurements but not the classical operations like erasure. The full equality $\mathsf{BQL} = \mathsf{BQ_UL}$ was finally proved by Fefferman and Remscrim [FR21], and here we presented the strongest statement of simulating general quantum channels with unitaries, using the conclusion of [FR21] and the techniques of [GRZ21b]:

Theorem 1.5. *Given a general quantum algorithm $\mathcal{A}$ with time T and space S, represented by the quantum channels applied in each step, we can compute in time* $\mathrm{poly}(2^S T)$ *and space* $O(S + \log T)$ *a unitary quantum circuit U with the same time and space, such that the final state of $\mathcal{A}$ and U are polynomially close when measured under the computational basis.*

Theorem 1.5 will be formally stated as Theorem 6.2.6. Note that in [FR21] the result

was stated for approximating a single coordinate, and the corresponding result in [FL18] that it relies on has constant error probability. As the simulation result with constant error in [GRZ21b] is entirely subsumed by that of [FR21], we will focus on the error reduction part with techniques developed in [GRZ21b].

As a by-product, the series of work [FL18, GRZ21b, FR21] also provides multiple complete problems for BQL, which works as candidate problems separating BQL and BPL. Examples of such problems include approximating powers of unitary matrices (with additive error), or determinants of well-conditioned matrices (with multiplicative error). The best classical upper bound for these problems is NC^2, hence no classical polynomial-time algorithm with $O(\log^{2-\varepsilon} n)$ space is known. The equality $\mathsf{BQL} = \mathsf{BQ_UL}$ also helped characterizing quantum space complexity using span programs [Jef22].

Finally, we would like to mention that in the large space regime, a highly-efficient intermediate measurement elimination scheme was proposed by Girish and Raz [GR22]. When the original quantum algorithm consists of only unitaries and measurements, the intermediate measurements can be removed in time $T \cdot \mathrm{poly}(S)$ and space $O(S \cdot \log T)$. Thus multiple extensions of Theorem 1.5 remains open: To show a unitary simulation of general quantum algorithms matching the bounds of [GR22], simulation with one-sided error ($\mathsf{RQL} = \mathsf{RQ_UL}$) [FR21], and similar simulation result for state synthesis [RY22].

1.3 Learning with Bounded Space

Even with the oracle separation $\mathsf{BQP}^{\mathcal{O}} \not\subseteq \mathsf{PH}^{\mathcal{O}}$ by Raz and Tal [RT22] and numerous reports on experimental quantum supremacy, proving unconditional, unrelativized superpolynomial quantum advantage still has a long way to go. The closest result, by Yamakawa

and Zhandry [YZ22], is still relative to a random oracle. This is most due to the fact that we do not know how to prove unconditional super-polynomial classical lower bounds in general computation models, and the situation is not bettered when adding space constraints (see Chapter 7 for a discussion).

It turns out that our current best hope in proving such advantage is on learning problems, for which we do have exponential classical lower bounds. Indeed, in [Raz18] Raz showed that parity learning, the problem of learning an unknown parity function on n bits, requires $2^{\Omega(n)}$ samples when the space usage is sub-quadratic. A long line of follow-up works [KRT17, Raz17, MM18, BGY18, GRT18, GRT19, GKR20, GKLR21] extend this result to many different learning problems and settings. In particular, [GRT18] showed that a similar exponential sample lower bound with bounded space holds whenever the learning problem exhibits the extractor property, which includes problems such as learning low-degree polynomials and learning error correcting codes.

With these strong classical lower bounds, it seems that proving an exponential separation between quantum and classical learning in the space-bounded setting is not out of reach. However, we show that in certain regime, this task as hard as proving separations for decision problems:

Theorem 1.6. *Every quantum learning algorithm with time T and space $S = O(\log T)$ can be simulated classically with time* $\mathrm{poly}(T)$ *and space $O(\log T)$, if and only if* $\mathsf{BQL} = \mathsf{BPL}$.

Theorem 1.6 will be proved in Section 6.3. It indicates that if we want to prove super polynomial separations, we need to look for candidate problems where the quantum learning algorithm uses more than logarithmic number of qubits. In fact, such a result has been given by Chen, Cotler, Huang and Li [CCHL21]. They showed that for tasks like shadow

tomography [Aar20] on an n-qubit state, classical learning algorithms need to measure $2^{\Omega(n)}$ copies of the state, while $O(n)$ copies suffices when there are n qubits of quantum memory. Moreover, when the shadow tomography is on Pauli observables, they proved a smooth tradeoff that with k qubits of quantum memory, $2^{\Omega(n-k)}$ samples are required.

Yet, the result in [CCHL21] is not a desired proof of quantum advantage, because the learning problems there are inherently quantum, and a classical learning algorithm does not have full access to a sample which is a copy of the quantum state. Therefore, the question of demonstrating exponential quantum vs. classical separation for classical learning problems is still open. Given the results of [GRT18], the natural candidates for this separation are the learning problems with the extractor property. We study the plausibility of these candidate problems and show that their classical lower bounds could not be improved with a small amount of quantum memory. Using the parity learning as an example, our result states as follows.

Theorem 1.7. *Any learning algorithm for parity learning on n bits requires either:*

- $\Omega(n^2)$ *bits of classical memory; or,*
- $\Omega(n)$ *qubits of quantum memory; or,*
- $2^{\Omega(n)}$ *samples.*

Theorem 1.7 will be formally stated as Theorem 10.3.1. It implies that if any such problems works in demonstrating the separation, the usage of quantum memory in the quantum upper bound will not be small. It is in sharp contrast with the situation of problems like Sorting, where $O(\log n)$ qubits of quantum memory suffices to defy the classical

lower bound [Kla03]. The result also gives a direct lift on the bounded-storage cryptography [Mau92] based on parity learning [Raz18, GZ19, LV21, DQW22]. For a cryptographic protocol using parity learning on n bits (which is treated as a security parameter), Theorem 1.7 shows that the security holds even in the presence of a quantum adversary with at most $O(n^2)$ bits of classical memory and $O(n)$ qubits of quantum memory.

We also note that the lower bound in [GRT18] is not always tight for all parameters. For instance, when the task is to learn $x \in \{0,1\}^n$ with samples being parity equations on x with sparsity ℓ, they gave lower bound $2^{\Omega(\ell \log \ell)}$ on the number of samples, which is sublinear in n when ℓ is small and clearly not optimal. At an extreme, consider the case when $\ell = 1$, that is every sample provides a random coordinate in x. This is the standard coupon-collector scenario and we know that $O(n \log n)$ samples suffices to recover the full information about x. But when the memory is much less than n, we cannot store all the samples, and the situation becomes much more interesting if the goal is to answer some question about x, e.g. computing its parity, instead of outputting x as a whole which is impossible.

Theorem 1.8. *Any algorithm that computes $x_1 \oplus \cdots \oplus x_n$ in the above coupon-collector scenario with T samples and S space with zero-error must satisfy $TS \geq \widetilde{\Omega}(n^2)$.*

Theorem 1.8 will be formally stated and generalized to other computation problems in Theorem 8.1.1. We conjecture that the same bound holds for bounded-error computation (as this dissertation is finishing, we are glad to know that Dinur [Din23] affirmatively proved our conjecture), which will serve as the first step to tighten up the time-space tradeoffs of learning in the polynomial regime. This result is also closely related to strong lower bounds on deterministic branching programs, which is a major open problem that will be

discussed in Chapter 7.

1.4 Dissertation Organization and Bibliographic Details

In Chapter 2 we provide some background knowledge, definitions of notations and useful inequalities required for the reading. The rest of this dissertation will be divided into two parts: Part I on algorithmic results and Part II on lower bound results. At the beginning of each part, we will give an overview (Chapter 3 and Chapter 7) on related topics and techniques used in proving our results. Note that the two parts are not isolated, and there are many intersections and interactions between the two parts.

In Chapter 4, we study the robust notion of randomness and prove Theorem 1.1, based on [GRZ23]:

> Uma Girish, Ran Raz, and Wei Zhan. Is untrusted randomness helpful? In *14th Innovations in Theoretical Computer Science Conference, ITCS 2023*.

In Chapter 5, we revise the hardness vs. randomness paradigm in logspace and prove Theorem 1.2, base on [PRZ23]:

> Edward Pyne, Ran Raz, and Wei Zhan. Certified hardness vs. randomness for log-space. *Electronic Colloquium on Computational Complexity: ECCC, 2023*.

In Chapter 6, we show the power of unitary quantum logspace, proving Theorem 1.5 and Theorem 1.6, based on [GRZ21b]:

> Uma Girish, Ran Raz, and Wei Zhan. Quantum logspace algorithm for powering matrices with bounded norm. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021*.

In Chapter 8, we propose the coupon-collector model and prove Theorem 1.8 based on [RZ20]:

> Ran Raz and Wei Zhan. The random-query model and the memory-bounded coupon collector. In *11th Innovations in Theoretical Computer Science Conference, ITCS 2020*.

In Chapter 9, we examine randomized vs. deterministic separations in time-space tradeoffs and prove Theorem 1.4 based on [YZ23]:

> Huacheng Yu and Wei Zhan. Randomized vs. deterministic separation in timespace tradeoffs of multi-output functions. *In Preparation, 2023*.

And finally, in Chapter 10, we give time-space lower bound for learning with classical-quantum hybrid memory, proving Theorem 1.7 based on [LRZ23]:

> Qipeng Liu, Ran Raz, and Wei Zhan. Memory-sample lower bounds for learning with classical-quantum hybrid memory. In *55th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2023*.

Other works the author completed during the doctoral study, that are not included in this dissertation because of thematic relevance, are the series of works on parallel repetition [GHM+21, GHM+22, GMRZ22], and the work on the Forrelation function [GRZ21a].

2

Preliminaries

Let us begin with some general notations. Let $\mathbb{N}, \mathbb{N}_+, \mathbb{R}, \mathbb{R}_+, \mathbb{C}$ be the set of natural numbers, positive integers, real numbers, positive numbers and complex numbers. For $n \in \mathbb{N}_+$, let $[n]$ be the set $\{0, 1, \ldots, n\}$. The logarithm in base 2 is simply denoted as log, and e is the base of natural logarithm ln.

We use $x \sim \mathcal{D}$ to denote drawing x from a distribution $\mathcal{D}$. Let the support $\mathrm{supp}(\mathcal{D})$ be the elements d such that $\Pr_{x\sim\mathcal{D}}[x = d] > 0$. When $\mathcal{D}$ is a uniform distribution over its support D, we abuse the notation and use $x \sim D$ to stand for $x \sim \mathcal{D}$. Denote the uniform distribution over $\{0,1\}^n$ by U_n.

For a random variable X, the expectation is denoted as $\mathbb{E}[X]$, and the variance is $\mathrm{Var}\, X = \mathbb{E}[X^2] - \mathbb{E}[X]^2$. The covariance of two random variables X and Y is $\mathrm{Cov}(X, Y) = \mathbb{E}[XY] - \mathbb{E}[X]\,\mathbb{E}[Y]$. Note that we have $\mathrm{Var}(X + Y) = \mathrm{Var}\, X + \mathrm{Var}\, Y + 2\,\mathrm{Cov}(X, Y)$.

The asymptotic notations $O(\cdot)$ and $\Omega(\cdot)$ are used to bound the growth of functions. For any two quantities $f, g \geq 0$ that varies depending on other variables, we say $f = O(g)$ if $f \leq c \cdot g$ always holds for some absolute constant c, and similarly $f = \Omega(g)$ if $f \geq c \cdot g$ always

holds. We use asymptotic notations $\widetilde{O}$ and $\widetilde{\Omega}$ to hide poly-logarithmic factors in $O(\cdot)$ and $\Omega(\cdot)$. When the input size n is clear from context, the poly-logarithmic factors are always on n, e.g. $\widetilde{O}(1)$ always stands for $O(\mathrm{polylog}(n))$.

2.1 Vectors and Matrices

We claim the following notations for complex vectors and matrices, but they are defined the same way on all other base sets whenever applicable. Let $v \in \mathbb{C}^n$ be a vector, we use v_i to denote the i-th coordinate of v. For every $p \in [1, \infty]$, let the ℓ_p norm of v be

$$\|v\|_p = \left(\sum_{i=1}^{d} |v_i|^p \right)^{1/p}.$$

In particular, when $p = \infty$, $\|v\|_\infty = \max_{i \in [n]} |v_i|$. Whenever $p \geq p'$, we have

$$\|v\|_p \leq \|v\|_{p'} \leq n \cdot \|v\|_p .$$

For two vectors $u, v \in \mathbb{C}^d$, define their inner product as

$$\langle u, v \rangle = u^\dagger v = \sum_{i=1}^{d} \overline{u_i} v_i.$$

Hence $\|v\|_2^2 = \langle v, v \rangle$. We also abuse the notation to identify every distribution $\mathcal{D}$ over $[n]$ as a non-negative real vector in $\mathbb{R}^n$ with $\|\mathcal{D}\|_1 = 1$. For every $i \in [n]$, let e_i be the special vector whose i-th coordinate is 1 and all other coordinates are 0. The all-zero vector is denoted as $\vec{0}$.

For an m by n matrix $M \in \mathbb{C}^{m \times n}$, let $M_{i,j}$ be the entry of M at row i and column j.

We sometimes also use $M[i,j]$ for the same meaning. We use M_i to denote the vector in $\mathbb{C}^n$ corresponding to the i-th row of M. Let $\mathrm{vec}(M)$ be the vectorization of M, which is a vector of dimension mn formed by stacking the columns of M on top of each other, that is

$$\mathrm{vec}(M)_{i+(j-1)m} = M_{i,j}, \quad \forall i \in [m], j \in [n].$$

For every vector $v \in \mathbb{C}^n$, let $\mathrm{Diag}\, v \in \mathbb{C}^{n\times n}$ be the diagonal matrix whose diagonal entries represent v. Conversely, for every square matrix $M \in \mathbb{C}^{n\times n}$, let $\mathrm{diag}\, M$ be the vector consisting of the diagonal entries of M. The trace of M is the sum of elements in $\mathrm{diag}\, M$, that is,

$$\mathrm{Tr}[M] = \sum_{i=1}^{n} M_{i,i}.$$

We use $\mathbb{I}_n \in \mathbb{C}^{n\times n}$ to denote the identity matrix, and a matrix with only entries 0 is simply denoted as 0. Let $M^\dagger$ be the conjugate transpose of M. When $m = n$, we say M is unitary if $M^\dagger M = \mathbb{I}_n$, and Hermitian if $M = M^\dagger$. A Hermitian M is further positive semi-definite (PSD) if for every $v \in \mathbb{C}^n$, $v^\dagger M v \geq 0$, and is a projection if $M^2 = M$. We say $M \leq N$ for two Hermitian matrices if $N - M$ is PSD.

Matrix Norms

We will encounter a lot of different matrix norms. Let us start with the operator norms, which for every $p \in [1,\infty]$ is defined as

$$\|M\|_p = \max_{v\in\mathbb{C}^n, v\neq\vec{0}} \frac{\|Mv\|_p}{\|v\|_p}.$$

This norm is sub-multiplicative, i.e. for any two matrices M, N we have

$$\|MN\|_p \leq \|M\|_p \|N\|_p .$$

In particular, when $p = 2$, we usually omit the subscript and use $\|M\|$ directly, which is called the spectral norm of M. Other matrix norms we consider include the Frobenius norm $\|M\|_{\mathrm{F}} = \|\mathrm{vec}(M)\|_2$, and the trace norm $\|M\|_{\mathrm{Tr}} = \mathrm{Tr}\left[\sqrt{M^\dagger M}\right]$. We have the following inequalities: For every $M \in \mathbb{C}^{m\times n}$,

$$\|M\| \leq \|M\|_{\mathrm{F}} \leq \|M\|_{\mathrm{Tr}} .$$

Finally, we say a matrix $M \in \{-1,1\}^{m\times n}$ is a (k,ℓ)-extractor with error 2^{-r}, if for every distribution P over $[n]$ with $\|P\|_2 \leq 2^\ell/\sqrt{n}$, there are at most $2^{-k}m$ rows $i \in [m]$ such that

$$|\langle M_i, P\rangle| \geq 2^{-r}.$$

This definition will specifically be used in Chapter 10.

2.2 Quantum Information

We use the Dirac notation to denote a pure quantum state $|v\rangle$, which is a vector in the $\mathbb{C}^n$ with $\||v\rangle\|_2 = 1$, and $\langle v| = |v\rangle^\dagger$. The state evolves by unitaries $|v\rangle \mapsto U_t|v\rangle = e^{-iHt}|v\rangle$ where H is a Hermitian matrix called Hamiltonian. For a non-zero vector $u \in \mathbb{C}^n$, we use $|v\rangle \sim u$ to denote a quantum state in the same direction as u, that is, $|v\rangle = u/\|u\|_2$.

A mixed state is a probability distribution of pure states, which can be described as a

density operator

$$\rho = \sum_i p_i |v_i\rangle\langle v_i|,$$

where $\sum_i p_i = 1$. The maximally-mixed state is $\frac{1}{n}\mathbb{I}_n$. Another way to formulate the density operator is a PSD matrix $\rho \in \mathbb{C}^{n\times n}$ with $\mathrm{Tr}[\rho] = 1$. By comparison, a partial density operator is a PSD matrix $\tau \in \mathbb{C}^{n\times n}$ with $\mathrm{Tr}[\tau] \leq 1$. All the notions below on density operators also works on partial density operators.

The magnitude of ρ on a pure state $|v\rangle$ is given by

$$\langle v|\rho|v\rangle = \mathrm{Tr}[\rho|v\rangle\langle v|].$$

More generally, a measurement on quantum states is represented by a PSD matrix $M \in \mathbb{C}^{n\times n}$ such that $M \leq \mathbb{I}_n$, and the probability of getting this measurement outcome on ρ is $\mathrm{Tr}[\rho M]$. A positive operator-valued measure (POVM) is a set of measurements $\{M_i\}$ such that $\sum_i M_i = \mathbb{I}_n$.

We use the notation ρ_V to stress the case when ρ describes the quantum state of a system V. If V consists of two parts X and Y, that is, the space of classical base states is a tensor product $\mathcal{V} = \mathcal{X} \otimes \mathcal{Y}$, then the quantum state describing the sub-system on X is given by the partial trace:

$$\rho_X = \mathrm{Tr}_Y[\rho_{XY}], \text{ where } \langle x|\mathrm{Tr}_Y[\rho_{XY}]|x\rangle = \sum_{y\in\mathcal{Y}} \langle x,y|\rho_{XY}|x,y\rangle, \quad \forall x \in \mathcal{X}.$$

For every $y \in \mathcal{Y}$, let the conditional system on X given $Y = |y\rangle$ be

$$\rho_{X|y} = (\mathbb{I}_X \otimes \langle y|)\rho_{XY}(\mathbb{I}_X \otimes |y\rangle),$$

which is a partial density operator on X. Here $\mathbb{I}_X = \sum_x |x\rangle\langle x|$ is the identity operator on X, which is an identity matrix in the matrix form. We can show the equality

$$\rho_X = \sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} |x\rangle\langle x, y|\rho_{XY}|x, y\rangle\langle x| = \sum_{y\in\mathcal{Y}} \rho_{X|y}.$$

The fidelity between two quantum states ρ and σ in the same underlying space is

$$F(\rho, \sigma) = \mathrm{Tr}[\sqrt{\rho\sigma}]^2.$$

The von Neumann entropy of state ρ is defined as

$$\mathbf{S}(\rho) = -\mathrm{Tr}[\rho \ln \rho],$$

and the quantum mutual information between two states is defined as

$$\mathbf{S}(\rho \parallel \sigma) = \mathrm{Tr}[\rho \ln \rho - \rho \ln \sigma].$$

We can then write the quantum mutual information between the two parts X and Y of the system ρ_{XY} as

$$\mathbf{I}_\rho(X; Y) = \mathbf{S}(\rho_X) + \mathbf{S}(\rho_Y) - \mathbf{S}(\rho_{XY}) = \mathbf{S}\left(\rho_{XY} \parallel \rho_X \otimes \rho_Y\right).$$

Quantum Channels

Let $\mathcal{L}(\mathbb{C}^n)$ be the space of linear operators on $\mathbb{C}^n$. A quantum channel Φ is a completely-positive trace-preserving (CPTP) linear map $\Phi : \mathcal{L}(\mathbb{C}^m) \to \mathcal{L}(\mathbb{C}^n)$, which maps every density operator $\rho \in \mathcal{L}(\mathbb{C}^m)$ to a density operator $\Phi(\rho) \in \mathcal{L}(\mathbb{C}^n)$.

There are several ways to represent a quantum channel. The Kraus representation of the quantum channel Φ is a set of matrices $E_1, \ldots, E_k \in \mathbb{C}^{n \times m}$ such that

$$\sum_{i=1}^{k} E_i^\dagger E_i = \mathbb{I}_m \quad \text{and} \quad \Phi(\rho) = \sum_{i=1}^{k} E_i \rho E_i^\dagger.$$

The natural representation of Φ, denoted as $K(\Phi)$, is a matrix in $\mathbb{C}^{n^2 \times m^2}$ such that

$$\operatorname{vec}(\Phi(\rho)) = K(\Phi) \operatorname{vec}(\rho)$$

for any density operator $\rho \in \mathcal{L}(\mathbb{C}^m)$. Given the Kraus representation $E_1, \ldots, E_k$ of Φ, one can easily compute the natural representation

$$K(\Phi) = \sum_{i=1}^{k} \overline{E_i} \otimes E_i.$$

We focus on quantum channels from $\mathcal{L}(\mathbb{C}^n)$ to itself. An important property of quantum channels is the contractivity under trace norms, that is, $\|\Phi(\pi)\|_{\mathrm{Tr}} \leq \|\pi\|_{\mathrm{Tr}}$ for any $\pi \in \mathcal{L}(\mathbb{C}^n)$. A channel Φ is unital if it is also contractive under Frobenius norms, i.e. $\|\Phi(\pi)\|_{\mathrm{F}} \leq \|\pi\|_{\mathrm{F}}$, or equivalently, $\|K(\Phi)\| \leq 1$. Another equivalent definition of unital channels is the channels that map the maximally-mixed state to the maximally-mixed

state:

$$\Phi\left(\frac{1}{n}\mathbb{I}_n\right) = \frac{1}{n}\mathbb{I}_n.$$

The Kraus representation of a unital channel additionally satisfies $\sum_{i=1}^{k} E_i E_i^\dagger = \mathbb{I}_n$.

Quantum channels represents all the physically feasible operations on quantum states. Every unitary operator $U : |v\rangle \mapsto U|v\rangle$ is a quantum channel $\Phi(\rho) = U\rho U^\dagger$. A distribution over unitary operators is a mixed-unitary channel

$$\Phi(\rho) = \sum_i p_i U_i \rho U_i^\dagger.$$

A POVM $\{M_i\}$ is also a quantum channel after specifying the post-measurement states. In particular, if the post-measurement state of ρ with outcome M_i is $\sqrt{M_i}\rho\sqrt{M_i}/\mathrm{Tr}[\rho M_i]$, it corresponds to the channel

$$\Phi(\rho) = \sum_i \sqrt{M_i}\rho\sqrt{M_i}.$$

Unitary channels, mixed-unitary channels and POVM specified this way are all unital. An example of non-unital channels is resetting, that is, $\Phi(\rho) = |0^n\rangle\langle 0^n|$.

2.3 Some Useful Inequalities

2.3.1 Concentration Bounds

The Chernoff Bound will be used countless number of times. Here we state a general complex version of it.

Lemma 2.3.1 (Chernoff-Hoeffding). *Let X be a random complex number with $|X| \leq 1$,*

and $X_1, \ldots, X_n$ are n independent copies of X. Then

$$\Pr\left[\left|\frac{1}{n}(X_1 + \cdots + X_n) - \mathbb{E}[X]\right| \geq \varepsilon\right] \leq 4e^{-2n\varepsilon^2}.$$

Besides, we will use the following concentration and anti-concentration bounds.

Lemma 2.3.2 (Chebyshev). *Let X be a real random variable with variance $\operatorname{Var} X = \sigma^2$. For every $k > 0$,*

$$\Pr\left[|X - \mathbb{E}[X]| \geq k\sigma\right] \leq \frac{1}{k}.$$

Lemma 2.3.3 (Paley–Zygmund [PZ32]). *Let $X \geq 0$ be a real random variable, and $\theta \in [0, 1]$. Then*

$$\Pr[X \geq \theta\,\mathbb{E}[X]] \geq (1 - \theta)^2 \frac{\mathbb{E}[X]^2}{\mathbb{E}[X^2]}.$$

Lemma 2.3.4 (Laurent-Massart [LM00]). *Let $g \sim \mathcal{N}(0, 1)^n$ be the n-dimensional standard Gaussian. Then $X = \|g\|_2^2 \sim \chi^2_n$ follows the chi-square distribution, and for every $x > 0$,*

$$\Pr\left[X \geq n + 2x + 2\sqrt{nx}\right] \leq e^{-x}.$$

Lemma 2.3.5 (Carbery–Wright [CW01]). *There is a absolute constant $c > 0$, such that for every $n \in \mathbb{N}_+$, $\varepsilon > 0$, Hermitian $H \in \mathbb{C}^{n \times n}$ and the standard Gaussian $g \sim \mathcal{N}(0, 1)^n$,*

$$\Pr\left[|g^\dagger H g|^2 \leq \varepsilon \operatorname{Var}[g^\dagger H g]\right] \leq c\varepsilon.$$

2.3.2 Quantum Inequalities

We first present two bounds on the norm of quantum operators.

Proposition 2.3.6. *Every quantum channel* $\Phi : \mathcal{L}(\mathbb{C}^n) \to \mathcal{L}(\mathbb{C}^n)$ *satisfies* $\|K(\Phi)\| \leq \sqrt{n}$.

The proof for Proposition 2.3.6 can be found in [RPRŻ13].

Proposition 2.3.7. *Every measurement* $M \in \mathbb{C}^{n \times n}$ *satisfies* $\|\mathrm{vec}(M)\|_2 = \|M\|_{\mathrm{F}} \leq \sqrt{n}$.

Proof. Since M is PSD and $M \leq \mathbb{I}_n$, we have $M^2 \leq \mathbb{I}_n$, and thus

$$\|M\|_{\mathrm{F}}^2 = \mathrm{Tr}[M^\dagger M] = \mathrm{Tr}[M^2] \leq \mathrm{Tr}[\mathbb{I}_n] = n. \qquad \square$$

The Fuchs-van de Graaf inequality [FvdG99] states that for two density operators ρ and σ, we have

$$\frac{1}{2}\|\rho - \sigma\|_{\mathrm{Tr}} \leq \sqrt{1 - F(\rho, \sigma)}.$$

Here we prove a variant of this inequality on general PSD operators.

Lemma 2.3.8. *Let* ρ, σ *be two PSD operators. Assume* $\mathrm{Tr}[\rho] \geq \mathrm{Tr}[\sigma]$. *Then*

$$\frac{1}{2}\big\|\rho - \sigma\big\|_{\mathrm{Tr}} \leq \sqrt{\frac{1}{4}(\mathrm{Tr}[\rho] + \mathrm{Tr}[\sigma])^2 - F(\rho, \sigma)} \leq \sqrt{\mathrm{Tr}[\rho]^2 - F(\rho, \sigma)}.$$

Proof. Let u and v be purifications of ρ and σ, that is, $u, v \in \mathbb{C}^n$ with $\rho = \mathrm{Tr}_A[uu^\dagger]$ and $\sigma = \mathrm{Tr}_A[vv^\dagger]$ where A is some additional system. Let U be a unitary that diagonalizes $uu^\dagger - vv^\dagger$, that is there is a diagonal matrix $\Lambda \in \mathbb{C}^{n \times n}$ such that $uu^\dagger - vv^\dagger = U\Lambda U^\dagger$. Let $p =$

diag $U^\dagger uu^\dagger U$ and $q = \operatorname{diag} U^\dagger vv^\dagger U$. Then we have

$$\begin{aligned}
u^\dagger u = \operatorname{Tr}[U^\dagger uu^\dagger U] &= \sum_{i=1}^n p_i, \\
v^\dagger v = \operatorname{Tr}[U^\dagger vv^\dagger U] &= \sum_{i=1}^n q_i, \\
\left\| uu^\dagger - vv^\dagger \right\|_{\mathrm{Tr}} = \|\Lambda\|_{\mathrm{Tr}} &= \sum_{i=1}^n |p_i - q_i|, \\
|\langle u, v\rangle| = |\langle U^\dagger u, U^\dagger v\rangle| &\le \sum_{i=1}^n \sqrt{p_i q_i}.
\end{aligned}$$

Therefore, by Cauchy-Schwarz inequality,

$$\begin{aligned}
\left\| uu^\dagger - vv^\dagger \right\|_{\mathrm{Tr}}^2 &= \left(\sum_{i=1}^n |p_i - q_i| \right)^2 \\
&= \left(\sum_{i=1}^n \left|\sqrt{p_i} - \sqrt{q_i}\right| \cdot \left|\sqrt{p_i} + \sqrt{q_i}\right| \right)^2 \\
&\le \left(\sum_{i=1}^n \left|\sqrt{p_i} - \sqrt{q_i}\right|^2 \right) \left(\sum_{i=1}^n \left|\sqrt{p_i} + \sqrt{q_i}\right|^2 \right) \\
&= \left(\sum_{i=1}^n p_i + \sum_{i=1}^n q_i \right)^2 - 4 \left(\sum_{i=1}^n \sqrt{p_i q_i} \right)^2 \\
&\le \left(u^\dagger u + v^\dagger v \right)^2 - 4 |\langle u, v\rangle|^2.
\end{aligned}$$

Notice that $\|\rho - \sigma\|_{\mathrm{Tr}} \le \left\| uu^\dagger - vv^\dagger \right\|_{\mathrm{Tr}}$, $\operatorname{Tr}[\rho] = u^\dagger u$ and $\operatorname{Tr}[\sigma] = v^\dagger v$. By Uhlmann's theorem [Uhl76], we can also choose u and v such that $F(\rho, \sigma) = |\langle u, v\rangle|^2$. Plugging them into the above inequality concludes the proof. □

Corollary 2.3.9. *For every partial density operator ρ and projection operator Π on ρ, we have*

$$\|\rho - \Pi\rho\Pi\|_{\mathrm{Tr}}^2 \leq 4\mathrm{Tr}[\rho]^2 - 4\mathrm{Tr}[\Pi\rho]^2.$$

Proof. By Lemma 2.3.8, it suffices to prove the following bound on fidelity:

$$F(\rho, \Pi\rho\Pi) \geq \mathrm{Tr}[\Pi\rho]^2.$$

Let u be a purification of ρ, that is, $\rho = \mathrm{Tr}_A[uu^\dagger]$ for some system A. Then $\left(\Pi \otimes \mathbb{I}_A\right)u$ is a purification of $\Pi\rho\Pi$. By Uhlmann's theorem we have

$$\begin{aligned} F(\rho, \Pi\rho\Pi) &\geq \left|u^\dagger\left(\Pi \otimes \mathbb{I}_A\right)u\right|^2 = \mathrm{Tr}\left[\left(\Pi \otimes \mathbb{I}_A\right)uu^\dagger\right]^2 \\ &= \mathrm{Tr}\left[\Pi \cdot \mathrm{Tr}_A[uu^\dagger]\right]^2 = \mathrm{Tr}[\Pi\rho]^2. \end{aligned}$$

□

Finally, we prove the anti-concentration bound on uniformly random pure states.

Lemma 2.3.10. *There exists an absolute constant c such that following holds. Let $|v\rangle$ be a uniformly random pure state in $\mathbb{C}^n$, and let $H \in \mathbb{C}^{n\times n}$ be a Hermitian. Then for every $\varepsilon > 0$, we have*

$$\Pr\left[|\langle v|H|v\rangle| \leq \varepsilon n^{-1}\|H\|\right] \leq c\sqrt{\varepsilon} + e^{-n}.$$

Proof. Let $g = (g_1, \ldots, g_n) \sim \mathcal{N}(0,1)^n$ be the standard Gaussian. Notice that $|g^\dagger \sigma g|/\|g\|_2^2$

is equidistributed as $|\langle v|H|v\rangle|$. Therefore by union bound we have

$$\begin{aligned}\Pr_v\left[|\langle v|H|v\rangle| \le \varepsilon n^{-1}\|H\|\right] &= \Pr_g\left[|g^\dagger Hg| \le \varepsilon\|H\| \cdot \frac{\|g\|_2^2}{n}\right] \\ &\le \Pr_g\left[|g^\dagger Hg| \le 5\varepsilon\|H\|\right] + \Pr_g\left[\|g\|_2^2 \ge 5n\right].\end{aligned}$$

For the first term, notice that $\mathrm{Var}[g^\dagger Hg] = 2\mathrm{Tr}[H^2]$ (see e.g. [RS08, Chapter 5]) which is no smaller than $2\|H\|^2$. Therefore, by Carbery–Wright inequality Lemma 2.3.5, there exists an absolute constant c such that

$$\Pr\left[|g^\dagger Hg| \le 5\varepsilon\|H\|\right] \le \Pr\left[|g^\dagger Hg| \le 4\varepsilon\,\mathrm{Var}[g^\dagger Hg]^{1/2}\right] \le c\sqrt{\varepsilon}.$$

And the second term is bounded by Lemma 2.3.4 with $x = n$:

$$\Pr\left[\|g\|_2^2 \ge 5n\right] \le e^{-n}. \qquad \square$$

2.4 Computational Models and Complexity Classes

2.4.1 Branching Programs

Universally in this dissertation, the computational models we consider are branching programs. These are the most general sequential models when considering both time and space complexity, and our results can all be easily translated to more restrictive models such as Turing machines or RAMs.

We start with the definition of classical branching programs. A deterministic branching program B with *space* S and *time* T is a layered directed acyclic graph (DAG) that consists of

at most $T+1$ layers of vertices (or states) $V_0, \ldots, V_T$, each contains at most 2^S vertices. The first layer V_0 contains one unique state called the initial state v_0, and each state in the last layer V_T represents an output. Let $V(B)$ be the collection of vertices in B. For each vertex $v \in V(B)$, let $B_{\to v}$ be the branching program that cuts off all the layers in B after v, and instead let v output 1 while all other vertices in the same layer of v output 0.

When the inputs (or samples) of the problem are from a domain D, each vertex $v \notin V_T$ has $|D|$ edges going out towards the next layer labeled with elements in D. If the problem allows querying specific coordinates of the input from D^n, v is also label with some $i \in [n]$ indexing the coordinate that the algorithm queries at the state v. We say the branching program is *oblivious*, if the query pattern is independent of the input. Specifically, if $T = n$ and the branching program always queries coordinates $1, \ldots, n$ in order, then we way it is an ordered branching program (OBP).

A randomized branching program with space S and time T is a distribution over deterministic branching programs with the same space and time bound. If the deterministic branching programs in the distribution are all oblivious, the randomized branching program is also oblivious. We say that a randomized branching program has *one-way access* to random bits, if in each layer the labels on the vertices and outgoing edges are independent of the rest of the branching program.

For both deterministic and randomized branching programs, the *computation path* is the path that starts from the initial vertex v_0, at each vertex following the edges labeled by the query answers or samples it receives until reaching the last layer. The computation path is subject to the randomness of the branching program, and samples in the learning case, but for learning we can actually without loss of generality assume that the branching program is

deterministic because of linearity of expectation.

For quantum branching programs, we can use a similar definition by replacing edges with transitions on superpositions of states. Here we present a simpler but equivalent definition. A unitary quantum algorithm with space S and time T starts from the initial state $|v_0\rangle = |0^S\rangle$, and applies the unitary operator $U_t \in \mathcal{L}(\mathbb{C}^{2^S})$ controlled by either the input query or the sample in each time step $t \in [T]$. After T steps the final state becomes

$$|v_T\rangle = U_T \cdot \cdots \cdot U_1 |v_0\rangle$$

which is measured in the computational basis, and outputs the answer according to the measurement result.

In general, a quantum algorithm could allow non-unitary quantum channels, and thus we instead describe the state using density operators. Starting from the initial state $\rho_0 = |0^S\rangle\langle 0^S|$, in each step t a channel $\Phi : \mathcal{L}(\mathbb{C}^{2^S}) \to \mathcal{L}(\mathbb{C}^{2^S})$ is applied, and hence the final state is

$$\rho_T = \Phi_T \circ \cdots \circ \Phi_1(\rho_0).$$

In addition, for decision problems we can assume that the final measurement is some two-outcome measurement $\{M, \mathbb{I} - M\}$, so that the probability of the branching program outputting 0 is $\mathrm{Tr}[\rho_T M]$.

For all branching program models presented above, we say the branching program with space S and time T is uniform, if the branching program itself can be printed by a deterministic Turing machine within the same space and time bound for every input size.

2.4.2 Space-Bounded Complexity Classes

For the sake of succinctness, we only define the complexity classes that will appear substantially in subsequent chapters, and we only define their space bounded versions. These include:

$\mathsf{DSPACE}(S)$ The set of decision problems solvable by uniform deterministic branching programs with space $O(S)$ and time $O(2^S)$. In particular, $\mathsf{L} = \mathsf{DSPACE}(\log n)$ and $\mathsf{L}^2 = \mathsf{DSPACE}(\log^2 n)$.

$\mathsf{BPTISP}(T, S)$ The set of decision problems solvable by uniform randomized branching programs with space $O(S)$ and time $O(T)$, with one way access to random bits. On every input, the output is in $\{0, 1\}$ and must be correct with probability at least $2/3$ (which is called *bounded error*).

BPL The union of $\mathsf{BPTISP}(n^c, \log n)$ for all $c > 0$. Equivalently, It is the set of decision problems solvable with bounded error by OBPs on the random bits, that uniformly depends on the input, with space $O(\log n)$ and time $\mathrm{poly}(n)$.

ZPL The same as BPL, but on every input, the output is in $\{0, 1, \perp\}$ where $\perp$ stands for giving up. The correct answer must be outputted with probability at least $1/2$, and the wrong answer is never outputted (this is called *zero error*).

BQL The set of decision problems solvable with bounded error by uniform quantum branching programs with space $O(\log n)$ and time $\mathrm{poly}(n)$.

$\mathsf{BQ_UL}$ The same as BQL, but the quantum branching programs are unitary.

All the above classes are usually defined for languages, which are total functions on $\{0,1\}^*$. In this dissertation we instead use these notations to denote the more general promised versions, which are defined for all partial boolean functions on $\{0,1\}^*$. Most of our results still holds for languages, and the exceptions are clearly identifiable from context. Finally, we note that all these classes can be naturally extended to their non-uniform versions.

Part I

Algorithmic Results

3

Overview of Part I

In Part I, we present our results that could be classified as algorithmic. A shared theme for all these results is that certain classes of space-bounded computation can be simulated by some other classes with less resources while the space usage increases by at most a constant factor. For instance, we partially derandomized BPL, and eliminate intermediate measurements for BQL, while keeping the space logarithmic.

Space-Bounded Computation and Matrix Powering

A particularly interesting perspective of space-bounded computation is that all the classes with space constraint can be characterized as powering (or iteratively multiplying) matrices with bounded dimensions. This is because when the space is limited to $O(S)$ (qu)bits, the computation model contains at most $2^{O}(S)$ base states, and the actual state could be a probability distribution or a superposition on the base states, depending on which resource the model possesses. Then the transition between the states naturally corresponds to $2^{O}(S) \times 2^{O(S)}$ matrices. Specifically we have the following correspondences:

- Classical deterministic computation $\leftrightarrow$ Deterministic Transition matrices ($0,1$-matrices that has exactly one entry 1 in each column)
- Classical reversible computation $\leftrightarrow$ Permutation matrices
- Classical randomized computation $\leftrightarrow$ Stochastic matrices
- Quantum computation $\leftrightarrow$ Unitary matrices

The actual computation is just applying these transition matrices, which are determined by the input x (and uniformly determined if the model is uniform), for the time limit number of times. Therefore, the computation corresponds to repeatedly multiplying such matrices, or even powering the matrices where we integrate the time stamps if the time limit $T \leq 2^{O}(S)$. Since matrix powering is a problem in NC^2, this fact alone is powerful enough to put almost all logspace complexity classes (L, NL, BPL, BQL, $\#\mathsf{L}$ and GapL, etc.) in NC^2. This also provides a very simple characterization of these complexity classes via the complete problems of powering corresponding matrices, for instance:

Theorem 3.1. *The Stochastic Matrix Powering problem is* BPL*-complete. The input includes an* $n \times n$ *stochastic matrix* M *and a parameter* T *such that* $T \leq \mathrm{poly}(n)$*. The promise is that* $M^T[n,1] \geq 4/5$ *or* $M^T[n,1] \leq 1/5$*, and the output is* 1 *in the former case and* 0 *in the latter.*

Theorem 3.2. *The Unitary Matrix Powering problem is* BQL*-complete. The input include a unitary matrix* $M \in \mathbb{C}^{n\times n}$*, a parameter* $T \leq \mathrm{poly}(n)$ *and a projective measurement* $\Pi \in \mathbb{C}^{n\times n}$*. The promise on the input is that* $\|\Pi M^T e_1\|_2^2 \geq 4/5$ *or* $\|\Pi M^T e_1\|_2^2 \leq 1/5$*, and the output is* 1 *in the former case and* 0 *in the latter.*

Albeit being simple observations, characterizations like Theorem 3.1 and Theorem 3.2 turn out to be very fundamental and have found numerous applications, such as designing new complete problems for space-bounded complexity classes [DST17, FL18], and developing new derandomization algorithms [CDST22]. In the rest of Part I we will see more applications, which we give a brief overview below.

Robust Algorithms with Untrusted Randomness

One benefit of the matrix powering characterization is that linear algebra is efficiently verifiable. To verify that $M \cdot x = y$, where the matrix M is easily accessible but the vectors x and y are costly to access, one could sample a random vector a and check whether $(a \cdot M) \cdot x$ equals $a \cdot y$. This way, each entry of x and y is accessed only once. Even better, it suffices to use a pseudorandom vector a (specifically k-wise independent when the field is $\mathbb{R}$), so that the randomness usage and space usage are both only logarithmic in the dimension.

This implies an efficient way to verify space-bounded computation: Since all we care about is $M^T e_1$ for some easily computable transition matrix M and initial vector e_1, the proof could just be $Me_1, M^2e_1, \ldots, M^Te_1$, and we check whether each vector after multiplied by M equals the next. The proof could be generated by simply repeatedly simulating the original computation, and therefore the proof protocol could be realized by a pair of efficient prover and verifier where the prover has unlimited randomness but the verifier has only logarithmic randomness.

As we show in Chapter 4, this protocol is closely related to the notion of robust randomness. We consider algorithms that uses two parts of randomness, one is trusted and is guaranteed to be perfectly random, while the other is untrusted and could be adversarially

chosen. In particular, if we run the prover's algorithm from the above protocol but feed it with untrusted random bits, the generated proof either passes the verification and yields the correct answer or gets rejected, and the verification uses only $O(\log n)$ trusted random bits. In order words, we reduce the trusted random bit usage to logarithmic while keeping the small space usage. This gives rise to the result of Theorem 4.1.11, which in the language of complexity classes is translated to $\mathsf{BPL} = \mathsf{RPL}(\log n)$.

Certified Hardness vs. Randomness

Assume that we have a function that is supposed to be a PRG, for instance the Nisan-Wigderson PRG [NW94] constructed from a presumably hard function f. We use its outputs as the untrusted random bits fed to the prover above. This provides an efficient way to utilize the pseudorandomness that is in some sense certified: Either the algorithm gives an output that is certified to be correct, or rejects the proof and aborts so that the function f is certified to be not hard, and a certification of the latter fact, which is a small circuit that average-case computes f, could be easily generated.

What we do in Chapter 5 is even stronger. Instead of using the average case hardness assumption in [NW94], we can start with the worse-case assumption in [IW97] and implement the hardness amplification process. In details, starting with a boolean function f that we assume no small circuit could compute correctly on all inputs, [IW97] shows that how to construct another boolean function f' that no small circuit could compute correctly even on a little bit more than half of the inputs, which is the hard function we want in the Nisan-Wigderson PRG. It was further checked in [KvM02] that the construction from f to f' is realizable in logspace, which gives rise to a RPG for BPL assuming f is computable in

small (actually linear) space.

In their proof for the hardness amplification process, [IW97] showed that if there exists a small circuit $\mathcal{B}$ that computes f' in the average case, then there exists a small circuit $\mathcal{C}$ computing f in the worst case (i.e. on all inputs). For our purpose, we want the result to be certifiable so we need to explicitly *reconstruct* the circuit $\mathcal{C}$ from $\mathcal{B}$, deterministically in small space. However, even though [IW97] did provide the reconstruction somewhat explicitly, certain steps in the process is not clearly implementable within small space. Moreover, the reconstruction heavily depends on taking the majority vote over repeated random procedures, in the form that if $\mathcal{B}(x, r)$ equals $f(x)$ with high probability over random r, then taking polynomially many independent r_i we have that $\mathtt{MAJ}_i(\mathcal{B}(x, r_i))$ equals $f(x)$ with error probability exponentially small. By the union bound over all possible inputs x, there must exists a selection of r_i which makes the majority equals $f(x)$ on all x. This is basically the same way that $\mathsf{BPP} = \mathsf{P/poly}$ is proved [Adl78], but we cannot assort to non-uniformity, nor could we store that many random bits and test all possibilities.

In our actual proof, we use an amplification and reconstruction process that is a bit different from the one in [IW97]. In particular, the amplification from a worst-case hard function to a constant-average-case hard (hard on a constant fraction of inputs) function in [IW97], which relies on Impagliazzo's Hardcore-Set Lemma [Imp95] and is not clearly doable in logspace, is replaced with the low-degree polynomial extension process in [STV01]. The results in [STV01] actually showed that the entire hardness amplification in [IW97] could be replaced with low-degree extension, and the reconstruction corresponds to list-decoding Reed-Muller codes. However, it is also not clear that list-decoding could be done deterministically in logspace, so we only use low-degree extension up to constant er-

ror so that it can be directly decoded with Berlekamp-Welch algorithm [WB86]. The other technical issue mentioned above of error reduction by repetition, is solved by not repeating the randomness independently but using pseudorandomness, generated by *samplers* (Definition 5.2.2), which has logarithmic seed length and is computable in logspace [RVW01].

Unitary Quantum Computing

Notice that Theorem 3.2 does not hold trivially. In fact, since the problem is only powering unitary matrices, the corresponding computation should be unitary quantum logspace, i.e. $\mathsf{BQ_UL}$. The reason it holds is because we have the results that general quantum computing can be simulated by unitary quantum computing, with constant error in logspace [FR21].

Without the logspace constraint, it was known how to simulate any quantum channel with unitary quantum circuits [ICC17, SNA+17], in which intermediate measurements are allowed and can be deferred at the cost of space. The key idea to perform the simulation in logspace is that instead of simulating the channels Φ themselves, we simulate their natural representations $K(\Phi)$. The problem with this approach is that $K(\Phi)$ could have large spectral norms, and if we force them into unitary matrices we have to scale them down, and that introduces a exponentially large factor in error after powering. In [FR21], they found out that we actually do not need to scale each individual $K(\Phi)$ down: The problem of computing the power $K(\Phi)^T$ actually reduces to computing the inverse Z^{-1} for some matrix Z polynomially related to $K(\Phi)$, and we just need to scale Z down which only introduces polynomially large factor in error.

In Chapter 6 we give a detailed review of this approach, and enhance the result of [FR21] by showing that not only decision problems can be simulated, we can actually use unitary

quantum circuits to output the entire final state of any general quantum algorithm up to polynomial accuracy, and thus could simulate the results of any multi-part measurement. We also show that any algorithm that simulates quantum computing for decision problem can also be used to simulate quantum learning with the same efficiency. Both these results rely heavily on producing consistent $\{0, 1\}$-bits so that the results on decision problems could be used, similar to the concept of pseudo-determinism for randomized computation, and indeed we adapt the *shift and truncate* method by Saks and Zhou [SZ99] to prove our results.

4

Robustly Randomized Algorithms

The goal of this chapter is to introduce a new type of randomized algorithms called *robustly randomized algorithms*, define the related complexity classes and prove its relations with previously-studied classes.

We first recall the definition of robustly randomized algorithms. Note that for simplicity, we define here robustly randomized algorithms only for total functions, but similar definitions can be given for partial functions, search problems, etc. Similar definitions can also be given in essentially all other settings where random strings are used, for example, query complexity, interactive proofs, etc.

Definition 4.1. *Let $f = \{f_n : \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$ be a family of functions. Let $k : \mathbb{N} \to \mathbb{N}$ be a monotone computable function. Let $\mathcal{A}$ be a randomized algorithm that uses two separate (read-once) random strings R_1, R_2. We say that $\mathcal{A}$ is a robustly randomized algorithm for f, with $O(k)$ trusted random bits, if on every input x of length n, the algorithm A reads at most $O(k(n))$ bits from R_1 and the output $\mathcal{A}(x)$ satisfies the following two requirements:*

1. *With the uniform distribution over R_1, R_2,*

$$\Pr_{R_1,R_2}[\mathcal{A}(x) = f_n(x)] \geq 3/4$$

2. *For every r (even adversarially chosen after seeing the input x),*

$$\Pr_{R_1}[\mathcal{A}(x) \in \{f_n(x), \perp\} | R_2 = r] \geq 3/4$$

(where the probability is over the uniform distribution over R_1).

4.1 Simple Relations with Other Classes

In this section we show that many previously-studied complexity classes can be revisited and redefined in light of our new definition. We first consider polynomial-time algorithms.

Definition 4.1.1. *Let $k : \mathbb{N} \to \mathbb{N}$ be a monotone computable function. The class $\mathsf{RPP}(k)$ is the class of all languages computable by a polynomial-time robustly randomized algorithm with $O(k)$ trusted random bits.*

Let ZPP be the class of problems solvable by zero-error probabilistic polynomial-time algorithms, and $\mathsf{BPP}(k)$ be the class of problems solvable by bounded-error probabilistic polynomial-time algorithms that are limited to reading $O(k)$ random bits. We have the following relations:

Proposition 4.1.2.

$$\mathsf{RPP}(0) = \mathsf{RPP}(\log n) = \mathsf{ZPP}.$$

Proof. Given a language L in ZPP, consider the zero-error randomized algorithm for L that outputs the correct answer with probability at least $\frac{3}{4}$. This algorithm can be directly viewed as a robustly-randomized algorithm where all the random bits are untrusted, and thus $\mathsf{ZPP} \subseteq \mathsf{RPP}(0)$.

On the other hand, given a language L in $\mathsf{RPP}(\log n)$, consider its robustly randomized algorithm A with $O(\log n)$ trusted random bits R_1 and (polynomially many) untrusted random bits R_2. We design a zero-error randomized algorithm for L as follows. After sampling $r \sim R_2$ and storing r, it iterates through all $2^{O(\log n)}$ possible values of R_1. In each iteration it runs A based on the chosen value of R_1 and r, and takes the majority vote (breaking ties arbitrarily) over all $2^{O(\log n)}$ outputs as the final output.

To see the correctness of the algorithm, first notice that it is zero-error as for every r, any incorrect answer can be outputted by at most $\frac{1}{4}$ fraction of R_1 and thus cannot win the majority vote. Also notice that the correct answer wins the majority vote if it is outputted for more than half of R_1. Since the correct answer is outputted with probability at least $\frac{3}{4}$ over R_1, R_2, it must win the majority vote (and be the final output) with probability at least $\frac{1}{2}$ over R_2.

Therefore we showed $\mathsf{RPP}(\log n) \subseteq \mathsf{ZPP}$, and together it holds that $\mathsf{RPP}(0) = \mathsf{ZPP} = \mathsf{RPP}(\log n)$. □

Proposition 4.1.3. *For every k,*

$$\mathsf{BPP}(k) \subseteq \mathsf{RPP}(k).$$

Proof. Given a language L in $\mathsf{BPP}(k)$, consider the bounded-error randomized algorithm

for L that outputs the correct answer with probability at least $\frac{3}{4}$. This algorithm can be directly viewed as a robustly randomized algorithm where all the $O(k)$ random bits are trusted, and thus $\mathsf{BPP}(k) \subseteq \mathsf{RPP}(k)$. □

Corollary 4.1.4. *For every k,*

$$\mathsf{BPP}(k)^{\mathsf{ZPP}} \subseteq \mathsf{RPP}(k).$$

Proof. Consider an algorithm in $\mathsf{BPP}(k)^{\mathsf{ZPP}}$ with error probability $\frac{7}{8}$. By Proposition 4.1.2 and Proposition 4.1.3, we can simulate such an algorithm with a $\mathsf{RPP}(k)$ algorithm that uses only trusted random bits itself, but answers the oracle calls with $\mathsf{RPP}(0)$ subroutines which uses only untrusted random bits. The overall number of trusted random bits is $O(k)$.

Suppose there are m oracle calls. We can assume each $\mathsf{RPP}(0)$ subroutine outputs the correct answer to the oracle call with probability at least $1-\frac{1}{8m}$ by repetition. The algorithm aborts whenever one of the subroutines outputs $\perp$, so the overall success probability is at least $\frac{3}{4}$ by union bound. □

Proposition 4.1.5.

$$\bigcup_c \mathsf{RPP}(n^c) = \mathsf{BPP}.$$

Proof. For every c, given a language L in $\mathsf{RPP}(n^c)$ and its robustly randomized algorithm. We modify it so that whenever it is supposed to output $\perp$, it simply outputs 0 instead. Then the algorithm can be viewed as a randomized algorithm for L with error bounded

by $\frac{1}{4}$, and thus $\mathsf{RPP}(n^c) \subseteq \mathsf{BPP}$. Combined with Proposition 4.1.3 we have

$$\mathsf{BPP} = \bigcup_c \mathsf{BPP}(n^c) = \bigcup_c \mathsf{RPP}(n^c). \qquad \square$$

Proposition 4.1.6. *For every k, if $\mathsf{BPP}(k)$ and ZPP do not contain one another, then*

$$\mathsf{BPP}(k) \cup \mathsf{ZPP} \neq \mathsf{RPP}(k).$$

Proof. Take $L_1 \in \mathsf{BPP}(k) \setminus \mathsf{ZPP}$, and $L_2 \in \mathsf{ZPP} \setminus \mathsf{BPP}(k)$. We claim that $L = L_1 \oplus L_2$ (the symmetric difference of L_1 and L_2) gives the desired separation.

Since $L \in \mathsf{BPP}(k)^{\mathsf{ZPP}}$, by Proposition 4.1.2 we have $L \in \mathsf{RPP}(k)$. On the other hand, $L \notin \mathsf{BPP}(k)$ since otherwise so does $L_2 = L \oplus L_1$. Similarly $L \notin \mathsf{ZPP}$. Notice that here we crucially use the fact that both classes are closed under symmetric difference. Therefore $\mathsf{BPP}(k) \cup \mathsf{ZPP} \neq \mathsf{RPP}(k)$. $\square$

Next, we show that every efficient robustly randomized algorithms is equivalent to an interactive proof with efficient prover and verifier, where the prover is allowed an unlimited number of random bits, but the random bits for the verifier (public or private) are limited. That is, unlike standard interactive proofs, the prover and the verifier here have the same computational power and only differ in the number of random bits that they are allowed to use, and the verifier is allowed to abort. We give the formal definition below.

Definition 4.1.7. *A language L is recognized by an interactive proof protocol $(\mathcal{P}, \mathcal{V})$ which outputs 1, 0 or $\perp$, if the following requirements are satisfied:*

1. *With the honest prover $\mathcal{P}$, $(\mathcal{P}, \mathcal{V})$ outputs 1 if $x \in L$ and outputs 0 if $x \notin L$ with*

probability at least $\frac{3}{4}$.

2. *With any (computationally unbounded) prover* $\mathcal{P}'$, $(\mathcal{P}, \mathcal{V})$ *outputs* 1 *or* $\perp$ *if* $x \in L$ *and outputs* 0 *or* $\perp$ *if* $x \notin L$ *with probability at least* $\frac{3}{4}$.

Such an interactive proof can be viewed as a proof of randomness: The prover sends a random string to the verifier and tries to convince the verifier that the random string is sufficiently random to perform the computation on a particular input x (which they both know). We are now ready to show the equivalence.

Proposition 4.1.8. *A language* L *is in* $\mathsf{RPP}(k)$ *if and only if it is recognizable by an interactive proof (which outputs* 1, 0 *or* $\perp$*) with a probabilistic polynomial-time prover and a probabilistic polynomial-time verifier, where the verifier is limited to at most* $O(k)$ *random bits.*

Proof. Given a robustly randomized algorithm, we design an interactive proof where the verifier simulates the algorithm, but with the untrusted random bits provided by the prover as proof. The verifier also aborts when the length of the proof does not equal to the number of untrusted random bits it needs. On the other hand, given a interactive proof protocol, we can simulate the protocol with a robustly randomized algorithm where the random bits used by the verifier are trusted and the random bits used by the prover are untrusted. The equivalence follows from Definition 4.1.1 and Definition 4.1.7. □

Finally, we consider robustly randomized algorithms in the logspace regime.

Definition 4.1.9. *Let* $k : \mathbb{N} \to \mathbb{N}$ *be a monotone computable function. The class* $\mathsf{RPL}(k)$ *is the class of all languages computable by a logarithmic-space and polynomial-time robustly-randomized algorithm with* $O(k)$ *trusted random bits.*

Similar to the polynomial-time cases, both ZPL (the class of problems solvable by zero-error probabilistic logspace algorithms) and $\mathsf{BPL}(k)$ (the class of problems solvable by bounded-error probabilistic logspace algorithms that are limited to reading $O(k)$ random bits) are closely related to $\mathsf{RPL}(k)$. In fact, we can show the following analog of Proposition 4.1.2:

Proposition 4.1.10.

$$\mathsf{RPL}(0) = \mathsf{RPL}(1) = \mathsf{ZPL}.$$

Proof. The proof is almost the same as the one for Proposition 4.1.2. The proof for $\mathsf{ZPL} \subseteq \mathsf{RPL}(0)$ goes in exactly the same way. However for the other direction $\mathsf{RPL}(1) \subseteq \mathsf{ZPL}$, there is a caveat: a logarithmic-space algorithm cannot afford to store all the untrusted random bits. Therefore, instead of sequential iterations over all possible values of the trusted bits, we do it in parallel, so that the untrusted random bits are still read-once. As there are $O(1)$ trusted random bits, this only increases the space by an $O(1)$ factor. □

However, rather surprisingly, unlike Proposition 4.1.5, we do not need to characterize BPL using RPL with polynomially many trusted random bits. As shown in the next theorem, $O(\log n)$ trusted random bits suffice to accomplish all the jobs in BPL.

Theorem 4.1.11.

$$\mathsf{RPL}(\log n) = \mathsf{BPL}.$$

We will prove Theorem 4.1.11 in the next section.

4.2 Streaming Proof for BPL

To prove Theorem 4.1.11, we will first show the equivalence of RPL algorithms with a special type of non-interactive proofs called streaming proofs, in a similar way to the equivalence in Proposition 4.1.8.

4.2.1 Streaming Proofs

A streaming proof consists of a pair of randomized Turing machines (a randomized prover and a randomized verifier) which share a common stream tape. We assume that the verifier is a logspace machine. The prover doesn't have a separate output tape, instead, it has write-once access to the stream tape onto which it writes a proof Π. The verifier has read-once access to the stream tape from which it can read Π. Both the verifier and the prover have read-many access to the input $x \in \{0,1\}^*$. We allow the prover and verifier to output a special symbol $\perp$. Upon outputting this symbol, the algorithm stops all further processing and we say that the algorithm aborts.

Definition 4.2.1. *Let $\mathcal{F} = \{f_n : \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$ be a family of functions. Let $P : \mathbb{N} \to \mathbb{N}$ be a monotone computable function. We say that $\mathcal{F}$ has a streaming proof of length P if there exists a prover $\mathcal{P}$ using a random string R_2 and a randomized logspace verifier $\mathcal{V}$ using a random string R_1 such that on input $x \in \operatorname{supp}(f_n)$,*

1. *The honest prover $\mathcal{P}$, with at least $\frac{3}{4}$ probability over R_2, outputs a (randomized) proof $\Pi \in \{0,1\}^{P(n)}$ such that*

$$\Pr_{R_1}[\mathcal{V}(x,\Pi) = f_n(x)] \geq \tfrac{3}{4}$$

(where the probability is over the uniform distribution over R_1, R_2).

2. *For an arbitrary* $\Pi \in \{0,1\}^{P(n)}$ *(even adversarially chosen after seeing the input* x*),*

$$\Pr_{R_1}[\mathcal{V}(x,\Pi) \in \{f_n(x), \perp\}] \geq \tfrac{3}{4}$$

(where the probability is over the uniform distribution over R_1*).*

Let $k : \mathbb{N} \to \mathbb{N}$ *be a monotone computable function. If the verifier* $\mathcal{V}$ *never reads more than* $O(k(n))$ *random bits from* R_1*, we say that the verifier uses at most* $O(k(n))$ *random bits.*

We sometimes omit mentioning the length $P(n)$ of the proof and it is understood that it is always bounded by the running time of the prover, which is polynomial in n.

Lemma 4.2.2. *A family of functions is in* $\mathsf{RPL}(k)$ *if it has a streaming proof between a logspace prover and a logspace verifier where the verifier uses* $O(k)$ *random bits.*

Proof. Given a streaming proof between a logspace verifier and a logspace prover where the verifier uses $O(k)$ random bits, consider a $\mathsf{RPL}(k)$ algorithm that simulates the verifier using trusted randomness. For each bit of the stream that the verifier wants to read, we have the algorithm simulate the honest prover using untrusted randomness. The completeness follows as both the verifier and the honest prover simultaneously succeed with probability at least $(3/4)^2 > 1/2$, furthermore, the probability that the verifier incorrectly answers given any proof is at most $1/4$. By standard error-reduction techniques, we can amplify the completeness to be at least $3/4$. This completes the proof. □

What remains is to show that BPL can be solved by streaming proof protocols with $k = O(\log n)$. This is reflected in the following lemma, which will be proved in the next subsection. For now, let us see how the two lemmas imply Theorem 4.1.11.

Lemma 4.2.3. *Every family of functions in* BPL *has a streaming proof between a logspace prover and a logspace verifier where the verifier uses* $O(\log n)$ *random bits.*

Proof of Theorem 4.1.11 from Lemma 4.2.2 and Lemma 4.2.3. Firstly, it is easy to see the containment $\mathsf{RPL}(\log n) \subseteq \mathsf{BPL}$. We consider any $\mathsf{RPL}(\log n)$ algorithm and modify it so that whenever it is supposed to output $\perp$, it outputs 0 instead. This can be viewed as a BPL algorithm with error at most $1/4$ and thus, $\mathsf{RPL}(\log n) \subseteq \mathsf{BPL}$.

The conclusion that $\mathsf{BPL} \subseteq \mathsf{RPL}(\log n)$ follows immediately from Lemma 4.2.2 and Lemma 4.2.3. □

4.2.2 Proof of Lemma 4.2.3

It suffices to develop a streaming proof for the Stochastic Matrix Powering problem, which is logspace-complete for BPL. Towards this, we define a notion of a δ-good sequence of vectors for a stochastic matrix M.

Definition 4.2.4. *Let M be any $n \times n$ stochastic matrix and $T \leq \mathrm{poly}(n)$ be a natural number. Let $v_i = M^i(e_1)$ for all $i \leq T$. Let $\delta \in [0,1]$. A sequence of vectors $v'_0, v'_1, \ldots, v'_T \in \mathbb{R}^n$ is said to be δ-good for M if for all $i \in [T]$, we have $\|v'_i - v_i\|_1 \leq \delta$ and $v'_0 = e_1$.*

We make use of the following two claims.

Claim 4.2.5. *There is a randomized logspace prover which given an $n \times n$ stochastic matrix M and parameters $T \leq \mathrm{poly}(n)$, $\delta \geq 1/\mathrm{poly}(n)$ as input, outputs a δ-good sequence of vectors for M with probability at least $3/4$.*

Claim 4.2.6. *Let $0 < \delta \leq (10^4 n T^3)^{-1}$. There is a randomized logspace verifier which given any $n \times n$ stochastic matrix M and parameters $T \leq \mathrm{poly}(n)$, $\delta \geq 1/\mathrm{poly}(n)$ as input and*

read-once access to a stream of vectors $v'_0, \ldots, v'_T \in \mathbb{R}^n$ *(where each vector is specified up to* $\Theta(\log(n))$ *bits of precision), does the following.*

- *If the sequence is* δ*-good for* M*, then the verifier aborts with probability at most* $1/4$*.*
- *If* $\|v'_T - v_T\|_1 \geq 1/4$*, then the verifier aborts with probability at least* $3/4$*.*

Furthermore, this verifier only uses $O(\log(n))$ *bits of randomness.*

We now complete the proof of Lemma 4.2.3 using Claim 4.2.5 and Claim 4.2.6. Consider the Stochastic Matrix Powering Problem. Given an $n \times n$ stochastic matrix M as input and a parameter $T \leq \text{poly}(n)$, set $\delta = \min\{(10^4 n T^3)^{-1}, 1/10\}$. Run the prover's algorithm from Claim 4.2.5 using this value of δ to produce a stream $v'_0, \ldots, v'_T$. Run the verifier's algorithm from Claim 4.2.6 on this stream to verify, and return $\perp$ whenever it aborts. If the verifier does not abort, we have it return 1 if $v'_T(n) \geq 2/3$, return 0 if $v'_T(n) \leq 1/3$ and return $\perp$ otherwise.

Completeness: Claim 4.2.5 implies that an honest prover outputs a δ-good sequence with probability at least $\frac{3}{4}$. Claim 4.2.6 implies that a verifier aborts an honest proof with very small probability. Since $\|v'_T - v_T\|_1 \leq \delta \leq 1/10$ by assumption, if $v_T(n) \geq 4/5$, then $v'_T(n) \geq 2/3$ and if $v_T(n) \leq 1/5$ then $v'_T(n) \leq 1/3$. Since $M^T[n, 1] = v_T(n)$, the verifier returns the correct answer whenever the subroutine does not abort.

Soundness: Consider the behavior of this verifier on an arbitrary proof. If the verifier makes a mistake and returns an incorrect answer, it must be the case that either $v_T(n) \geq 4/5$ and $v'_T(n) \leq 1/3$ or $v_T(n) \leq 1/5$ and $v'_T(n) \geq 2/3$. In either case, we must have

$\|v'_T - v_T\|_k \geq |v'_T(n) - v_T(n)| \geq 1/4$. Claim 4.2.6 implies that such a proof is aborted with probability at least $3/4$.

This completes the proof of Lemma 4.2.3. We now proceed to prove Claim 4.2.5 and Claim 4.2.6.

Proof of Claim 4.2.5. The prover's algorithm is formally described in Algorithm 4.1, which uses multiple samples to estimate the entries of $M^i(e_1)$. It is clear that Algorithm 4.1 can be performed in space $O(\log n)$.

Algorithm 4.1: Algorithm for Prover in Claim 4.2.5.

```
Output v_0 = e_1.
let C ← Θ(n^2 log(nT)/δ^2).
for i ← 1, ..., T, j ← 1, ..., n do
    for count ← 1, ..., C do
        let estimate_{i,j} ← 0, k ← 1.
        for t ← 1, ..., i do
            Sample k' ∈ [n] with probability M[k, k'].;
            let k ← k'.
        let estimate_{i,k} ← estimate_{i,k} + 1.
    Output vector v_i where v'_i(j) = estimate_{i,j}/C.
```

To see the correctness of the algorithm, observe that by evolving the state k from the initial state 1 for i steps according to M, the algorithm produces a state which is j with probability exactly $M^i[1,j] = v_i(j)$ for each $j \in [n]$. By repeated sampling and taking the empirical average, the algorithm can estimate each $v_i(j)$ up to δ/n accuracy with probability at least $1 - 1/(4nT)$ using $C = \Theta(n^2 \log(nT)/\delta^2)$ samples by Chernoff bound. This, along with a union bound over $i \in [T], j \in [n]$ implies that the algorithm with probability at least $3/4$, estimates each $v_i(j)$ up to δ/n precision, and hence produces v'_i such that $\|v'_i - v_i\|_1 \leq \delta$. □

Proof of Claim 4.2.6. The verifier's algorithm is formally described in Algorithm 4.2, which tries to check that $M \cdot v'_{i-1}$ is approximately equal to v'_i for all $i \in [T]$. However, to do this in a streaming fashion, the verifier will instead test that a random linear combination of these approximate equations holds. To reduce the randomness from T to $O(\log n)$, instead of using a truly random combination of the equations the verifier uses a pseudorandom combination drawn using a 4-wise independent collection of $\{-1, 1\}$-random variables. This is similar to the ℓ_2-frequency estimation algorithm in [AMS99].

Algorithm 4.2: Algorithm for Verifier in Claim 4.2.6

Abort if $v'_0 \neq e_1$.
for $t \leftarrow 1, \ldots, 11$ **do**
  Sample $\alpha_{i,j} \in \{-1, 1\}$ for $i \in [T], j \in [n]$ from a collection of 4-wise independent $\{-1, 1\}$-random variables with mean 0;
  let
$$\Delta \leftarrow \sum_{i \in [T], j \in [n]} \alpha_{i,j} \cdot \left[(M \cdot v'_{i-1})(j) - v'_i(j) \right].$$
  Abort if $|\Delta| > 20T\partial$.

One can sample from a collection of 4-wise independent $\{-1, 1\}$-random variables of size $O(nT)$ in logspace using only $O(\log(nT))$ bits of randomness [AMS99]. Note that the quantity Δ can be expressed as $\sum_{\substack{i \in \{0,\ldots,T\} \\ j \in [n]}} \beta_{i,j} v'_i(j)$ where $\beta_{i,j}$ are coefficients that depend only on the entries of M and α, and can be computed in logspace. Thus, a logspace algorithm can read the stream of $v'_i(j)$ for $i = 0, \ldots, T$ and $j \in [n]$ once from left to right and compute $\Delta = \sum_{i,j} \beta_{i,j} v'_i(j)$ in a streaming fashion.

We now move on to the completeness and soundness. Let $w \in \mathbb{R}^{nT}$ be defined at $i \in [T], j \in [n]$ by $w_{i,j} = (M \cdot v'_{i-1})(j) - v'_i(j)$. Let $v_0, \ldots, v_T$ be defined as before.

Completeness of the Algorithm: Suppose $v'_0, \ldots, v'_T$ is a δ-good sequence, then $\|v'_i - v_i\|_1 \leq \delta$ for all $i \in [T]$ and $v'_0 = e_1$. Since M is a contraction map with respect to $\|\cdot\|_1$, for all $i \in [T]$

$$\begin{aligned}\left\|M \cdot v'_{i-1} - v'_i\right\|_1 &\leq \left\|M \cdot v'_{i-1} - M \cdot v_{i-1}\right\|_1 + \left\|M \cdot v_{i-1} - v_i\right\|_1 + \|v_i - v'_i\|_1 \\ &\leq \|v_{i-1} - v'_{i-1}\|_1 + \|v_i - v'_i\|_1 \leq 2\delta.\end{aligned}$$

Thus, $\|w\|_2 \leq \|w\|_1 \leq 2T\delta$. Consider the quantity $\Delta = \langle \alpha, w \rangle = \sum_{i,j} \alpha_{i,j} w_{i,j}$ that the algorithm estimates. Note that $\mathbb{E}\langle \alpha, w \rangle = 0$, and

$$\mathrm{Var}\langle \alpha, w \rangle = \mathbb{E}\langle \alpha, w \rangle^2 = \sum_{i,j} w_{i,j}^2 = \|w\|_2^2.$$

Chebyshev's Inequality implies that with probability at least 0.99, we have $|\langle \alpha, w \rangle| \leq 20T\delta$. This implies that with probability at least $(0.99)^{11} \geq 0.8$, every iteration in Algorithm 4.2 does not abort.

Soundness of the Algorithm: Suppose a dishonest prover produces a stream of vectors $v'_0, \ldots, v'_T$ such that $\|v'_T - v_T\|_1 \geq 1/4$. The verifier always returns $\perp$ if $v'_0 \neq e_1$, so we may assume that $v'_0 = e_1$. Let $\varepsilon = 1/(10T)$. We argue that for some $i \in [T]$, we must have $\|w_i\|_1 > \varepsilon$. Assume by contradiction that $\left\|M \cdot v'_{i-1} - v'_i\right\|_1 \leq \varepsilon$ for all $i \in [T]$. Then

by triangle inequality we have

$$\begin{aligned}\|v_T - v'_T\|_1 = \left\|M^T(v'_0) - v'_T\right\|_1 &\le \sum_{i=1}^{T} \left\|M^{T-(i-1)} \cdot v'_{i-1} - M^{T-i} \cdot v'_i\right\|_1 \\ &\le \sum_{i=1}^{T} \left\|M^{T-i}\right\|_1 \cdot \left\|M \cdot v'_{i-1} - v'_i\right\|_1 \\ &\le \sum_{i=1}^{T} \varepsilon = T\varepsilon = 1/10.\end{aligned}$$

which contradicts the assumption that $\|v'_T - v_T\|_1 \ge 1/4$. Thus, we must have $\|w\|_2 \ge \|w\|_1/(nT) \ge \varepsilon/(nT)$. Note that $\mathbb{E}\langle \alpha, w\rangle = 0$ and $\mathbb{E}\langle \alpha, w\rangle^2 = \|w\|_2^2$. Furthermore,

$$\mathbb{E}\langle \alpha, w\rangle^4 = \mathbb{E}\sum_{i,j,k,l} w_i w_j w_k w_l \alpha_i \alpha_j \alpha_k \alpha_l \le 6\sum_{i,j} w_i^2 w_j^2 \le 6\|w\|_2^4.$$

Here, we used the fact that the random variables are 4-wise independent. By the Paley-Zygmund Inequality [PZ32], it implies that

$$\Pr\left[\langle \alpha, w\rangle^2 \ge \frac{1}{10} \cdot \|w\|_2^2\right] \ge (1 - 1/10)^2 \cdot \frac{(\mathbb{E}\langle \alpha, w\rangle^2)^2}{\mathbb{E}\langle \alpha, w\rangle^4} \ge \frac{1}{8}.$$

This, along with the fact that $\|w\|_2 \ge \varepsilon/(nT)$ implies that $\Pr\left[|\langle \alpha, w\rangle| \ge \frac{\varepsilon}{10nT}\right] \ge \frac{1}{8}$. By repeating this experiment 11 times, we can ensure that with probability at least $1 - (1 - 1/8)^{11} \ge 3/4$, we find at least one instance so that

$$|\langle \alpha, w\rangle| \ge \frac{\varepsilon}{10nT} = \frac{1}{100nT^2} > 20T\delta.$$

The last inequality is because $\delta \leq (10^4 n T^3)^{-1}$. This implies that the algorithm aborts with probability at least $3/4$. □

Remark. *In our algorithm, the verifier essentially checks if a certain random linear combination of the $M \cdot v'_{i-1}$ is close to the same linear combination of the v'_i. The verifier could have instead checked if for every $i \in [T]$, $M \cdot v'_{i-1}$ was close to v'_i. This could have been done by testing for each $i \in [T]$ whether $\langle M \cdot v'_{i-1}, w \rangle$ was close to $\langle v'_i, w \rangle$ for a certain random vector w. While the latter test is conceptually simpler, the former test has the additional useful property that the verifier simply computes a linear function in the variables $\{v_i(j)\}$.*

4.3 Query-Complexity Separations

In this section we consider the query-complexity model, where one can fully prove that the untrusted random string is helpful.

Proposition 4.3.1. *For any $n, k \in \mathbb{N}$, there are two disjoint subsets $A, B \subset \{0,1,2\}^{2^{n+k}}$ (where $\{0,1,2\}^{2^{n+k}}$ is viewed as the set of all functions $f : \{0,1\}^{n+k} \to \{0,1,2\}$), such that, given a black box access to a function $f : \{0,1\}^{n+k} \to \{0,1,2\}$, the following three are satisfied:*

1. *There is a robustly randomized protocol that uses only k trusted random bits and only one query to f and distinguishes between the cases $f \in A$ and $f \in B$ with probability at least $3/4$.*

2. *For any constant $\varepsilon > 0$, any randomized protocol that uses at most $(1-\varepsilon)n$ random bits and distinguishes between the cases $f \in A$ and $f \in B$ with probability at least $3/4$, requires at least $2^{\Omega(n)}$ queries to f.*

3. *Any zero-error randomized protocol that distinguishes between the cases $f \in A$ and $f \in B$ with probability at least $3/4$ (and outputs do not know with probability at most $1/4$) requires at least $2^{\Omega(k)}$ queries to f.*

Specifically, taking $k = \log n$ in Proposition 4.3.1, we observe that there are promise problems that can be solved by robustly-randomized protocols with only one query and just a logarithmic number of trusted random bits, whereas any randomized protocol requires either a linear number of random bits or an exponential number of queries, and any zero-error randomized protocol requires a polynomial number of queries.

Proof. We define the subsets $A, B \subset \{0,1,2\}^{2^{n+k}}$ as follows. Let R_1 and R_2 be uniformly distributed over $\{0,1\}^k$ and $\{0,1\}^n$ respectively. For every function $f : \{0,1\}^{n+k} \to \{0,1,2\}$, we say that

- $f \in A$ if and only if $\Pr[f(R_1, R_2) = 0] \geq \frac{3}{4}$, and for every $r \in \{0,1\}^n$,

$$\Pr[f(R_1, r) \in \{0,2\}] \geq \frac{3}{4}.$$

- $f \in B$ if and only if $\Pr[f(R_1, R_2) = 1] \geq \frac{3}{4}$, and for every $r \in \{0,1\}^n$,

$$\Pr[f(R_1, r) \in \{1,2\}] \geq \frac{3}{4}.$$

Clearly A and B are disjoint as $\Pr[f(R_1, R_2) = 0] + \Pr[f(R_1, R_2) = 1] \leq 1$.

1. A robustly randomized protocol that distinguishes between $f \in A$ and $f \in B$ uses k trusted random bits R_1 and n untrusted random bits R_2. It makes a single query

for $f(R_1, R_2)$ and based on the query result being $0, 1$ or 2, outputs A, B or $\perp$ accordingly. The correctness of the protocol is guaranteed by Definition 4.1.

2. For any constant $\varepsilon > 0$, consider a randomized protocol using at most $(1-\varepsilon)n$ random bits and making at most q queries to f. Fix all the query answers to be 2, and let $Q \subseteq \{0,1\}^{n+k}$ be all the possible positions being queried over all choices of the random bits. Assuming $q \leq \frac{1}{4} \cdot 2^{\varepsilon n}$, we have $|Q| \leq 2^{(1-\varepsilon)n} q \leq \frac{1}{4} \cdot 2^n$.

   Now pick $f, g : \{0,1\}^{n+k} \to \{0,1,2\}$ to be:

$$f(r) = \begin{cases} 0 & \text{if } r \notin Q \\ 2 & \text{if } r \in Q \end{cases} \qquad g(r) = \begin{cases} 1 & \text{if } r \notin Q \\ 2 & \text{if } r \in Q \end{cases}$$

   It is straightforward to check that $f \in A$ and $g \in B$. The protocol behaves exactly the same on f and g, thus being incorrect with probability at least $\frac{1}{2}$ on either one of them. Therefore, a successful protocol must have $q \geq 2^{\Omega(n)}$.

3. Consider a zero-error randomized protocol making at most q queries to f. Fix f to be the constant function 0 and clearly $f \in A$. We arbitrarily fix a choice of the random bits where the protocol outputs $f \in A$, and let Q be the positions queried under the fixing. Assume that $|Q| \leq q \leq \frac{1}{4} \cdot 2^k$.

   Now pick $g : \{0,1\}^{n+k} \to \{0,1,2\}$ to be:

$$g(r) = \begin{cases} 1 & \text{if } r \notin Q \\ 0 & \text{if } r \in Q \end{cases}$$

   It is straightforward to check that $g \in B$. Under the fixed choice of random bits, the

protocol behaves exactly the same on f and g, and thus incorrectly decides $g \in A$. Therefore, a successful protocol must have $q \geq 2^{\Omega(k)}$. □

5

Certified Hardness vs. Randomness for Logspace

Given Theorem 4.1.11 from last chapter, that every BPL problem can be solved with robustly randomized algorithms with $O(\log n)$ trusted random bits, the follow-up question is how to manipulate the untrusted random bits for our benefit. It turns out that the natural choice is to feed pseudorandom bits to the untrusted part. Say we have an $\mathsf{RPL}(\log n)$ algorithm $\mathcal{A}(x, R_1, R_2)$ for some BPL problem, where $|R_1| = O(\log n)$ and R_2 is untrusted, and we are given an alleged pseudorandom generator G of seed length $O(\log n)$. In logspace we can find the majority of outputs, among $\{0, 1\perp\}$, over $\mathcal{A}(x, y, G(z))$ for all y and z:

- If the majority is 0 or 1, it must be the correct answer on x by soundness of the robustly randomized algorithm.
- On the other hand, if the majority is $\perp$, it means that the pseudorandomness is defective. After all, if we replace $G(z)$ with truly random bits, by completeness the majority of outputs cannot be $\perp$. Therefore in this case we know for a fact that G is not

really pseudorandom, and $\mathcal{A}(x, y, \cdot)$ is a proof for this fact.

The goal of this chapter is to work more effectively with the above intuition, and prove two main results: Theorem 5.1.1 and Theorem 5.3.1.

5.1 Logspace Verifier for PRG

This and the next section are devoted to prove the following theorem:

Theorem 5.1.1. *For every family of boolean functions $f \in \mathsf{DSPACE}[n]$ and $\varepsilon > 0$, there is a deterministic algorithm that, given as the input an OBP B of length n and width n, runs in space $O(\log n)$, and either*

1. *Outputs $\mathbb{E}[B]$ with $1/4$ error; Or*
2. *Outputs a circuit C of size $2^{\varepsilon m}$ that computes f on $\{0,1\}^m$ where $m = \Theta(\log n)$.*

Before getting into the proof, let us first compare it to the result of Klivans and van Melkebeek [KvM02].

Theorem 5.1.2 ([KvM02])**.** *If there is a family of boolean functions $f \in \mathsf{DSPACE}[n]$ that is not computable by circuits of size $2^{\varepsilon n}$ for some $\varepsilon > 0$, then $\mathsf{BPL} = \mathsf{L}$.*

Their proof is based on the worst-case hardness vs. randomness results by Imagliazzo and Wigderson [IW97], and shows how every step in the construction of the Imagliazzo-Wigderson pseudorandom generator can be executed in deterministic logspace. To obtain our desire result, we need to provide two additional features. First, give an algorithm that distinguishes between the PRG and real randomness, we need to explicitly construct a next-bit predictor for the PRG. Second, we need to reconstruct the circuit that computes f from

the next-bit predictor in a space-efficient and uniform manner. In this section, we address the first feature. In order to do so let us first formally define pseudorandom generators and next-bit predictors.

Definition 5.1.3. *A pseudorandom generator (PRG) is a function $G : \{0,1\}^s \to \{0,1\}^n$, where s is usually referred to as seed length. We say that G ε-fools a function $f : \{0,1\}^n \to \mathbb{R}$ if*

$$\left|\mathbb{E}[f] - \mathop{\mathbb{E}}_{G}[f]\right| \leq \varepsilon,$$

where we use $\mathbb{E}[f]$ and $\mathbb{E}_G[f]$ to denote the expectation of f under uniformly distributed inputs and pseudorandom inputs generated by G respectively, that is,

$$\mathbb{E}[f] = \mathop{\mathbb{E}}_{x\sim U_n}[f(x)], \qquad \mathop{\mathbb{E}}_{G}[f] = \mathop{\mathbb{E}}_{y\sim U_s}[f(G(y))].$$

We use the same convention in all of the rest of this section.

Definition 5.1.4. *Given a generator $G : \{0,1\}^s \to \{0,1\}^n$, a function $T : \{0,1\}^i \to \{0,1\}$ with $i < n$ is an ε-next-bit predictor for G if*

$$\Pr_{x\sim U_s}[T(G(x)_{1..i}) = G(x)_{i+1}] > 1/2 + \varepsilon.$$

Note that a next-bit predictor is naturally not fooled by the pseudorandom generator.

The main lemma in this section is the following, which shows that there is a logspace verifier for PRGs (with logarithmic seed lengths against logspace OBPs), that detects when a PRG fails and outputs a next-bit predictor for the PRG.

Lemma 5.1.5. *For every error function $\varepsilon(n)$ computable in space $O(\log n)$, there is a deterministic algorithm that, given as input an OBP B of length n and width w, and the black-box oracle access to a PRG $G : \{0,1\}^s \to \{0,1\}^n$, runs in space $O(s + \log(nw))$, and either*

1. *Confirms that G $\varepsilon \cdot n$-fools B; Or*
2. *Outputs an OBP T of length at most n and width w that is an $\varepsilon/2$-next-bit predictor for G.*

Lemma 5.1.5 can be proved using Theorem 4.1.11, as we described at the start of this chapter. Here we present a simpler and more direct proof. We first define a series of potential distinguishers, with the property that each can be evaluated in logspace. Each distinguisher measures the bias of the next bit in the PRG upon reaching a particular state.

Definition 5.1.6. *Given an OBP B of length n, for every $i < n$ and $v \in V_i$, let $N_v : \{0,1\}^{i+1} \to \{-1,0,1\}$ be the function defined as:*

$$N_v(x) = \begin{cases} 1 & \text{if } B_{\to v}(x) = 1 \text{ and } x_{i+1} = 1 \\ -1 & \text{if } B_{\to v}(x) = 1 \text{ and } x_{i+1} = 0 \\ 0 & \text{otherwise.} \end{cases}$$

Furthermore, N_v is computable in logspace given B and v.

When x is uniformly random, $B_{\to v}(x)$ and x_{i+1} are independent, and therefore $\mathbb{E}[N_v] = 0$ for all v. Consequentially, our verifier checks that $|\mathbb{E}_G[N_v]|$ is small for all v, where we feed the first $i+1$ bits of the PRG output to N_v. We first show its soundness:

Lemma 5.1.7. *Given an OBP B of length n, suppose that for every i, $\sum_{v \in V_i} |\mathbb{E}_G[N_v]| \leq \varepsilon$. Then $G\,\varepsilon \cdot n$-fools B.*

Proof. As every edge from layer V_i goes into layer V_{i+1}, for every $i < n$ we have

$$\begin{aligned}
&\sum_{v \in V_{i+1}} \left| \mathbb{E}[B_{\to v}] - \mathop{\mathbb{E}}_{G}[B_{\to v}] \right| \\
&\leq \sum_{v \in V_i} \sum_{b \in \{0,1\}} \left| \Pr_{x \sim U_n}[B_{\to v}(x) = 1 \wedge x_{i+1} = b] - \Pr_{x \sim G(U_s)}[B_{\to v}(x) = 1 \wedge x_{i+1} = b] \right|.
\end{aligned}$$

Notice that by the definition of N_v, we have

$$\begin{aligned}
\mathbb{E}[N_v] &= \Pr_{x \sim U_n}[B_{\to v}(x) = 1 \wedge x_{i+1} = 1] - \Pr_{x \sim U_n}[B_{\to v}(x) = 1 \wedge x_{i+1} = 0] \\
&= 2 \Pr_{x \sim U_n}[B_{\to v}(x) = 1 \wedge x_{i+1} = 1] - \mathbb{E}[B_{\to v}] \\
&= \mathbb{E}[B_{\to v}] - 2 \Pr_{x \sim U_n}[B_{\to v}(x) = 1 \wedge x_{i+1} = 0],
\end{aligned}$$

and the above holds similarly under pseudorandomness generated by G. Therefore we further have

$$\begin{aligned}
\sum_{v \in V_{i+1}} \left| \mathbb{E}[B_{\to v}] - \mathop{\mathbb{E}}_{G}[B_{\to v}] \right| &\leq \sum_{v \in V_i} \left| \mathbb{E}[B_{\to v}] - \mathop{\mathbb{E}}_{G}[B_{\to v}] \right| + \sum_{v \in V_i} \left| \mathbb{E}[N_v] - \mathop{\mathbb{E}}_{G}[N_v] \right| \\
&= \sum_{v \in V_i} \left| \mathbb{E}[B_{\to v}] - \mathop{\mathbb{E}}_{G}[B_{\to v}] \right| + \sum_{v \in V_i} \left| \mathop{\mathbb{E}}_{G}[N_v] \right|.
\end{aligned}$$

With the assumption that $\sum_{v \in V_i} |\mathbb{E}_G[N_v]| \leq \varepsilon$ and the fact that $\mathbb{E}[B_{\to v_0}] = \mathbb{E}_G[B_{\to v_0}] = 1$, we conclude that $\sum_{v \in V_n} |\mathbb{E}[B_{\to v}] - \mathbb{E}_G[B_{\to v}]| \leq \varepsilon \cdot n$. As the output labels are binary, this means that $|\mathbb{E}[B] - \mathbb{E}_G[B]| \leq \varepsilon \cdot n$, i.e. $G\,\varepsilon \cdot n$-fools B. □

Proof of Lemma 5.1.5. For every $i < n$, the algorithm iterates through every $v \in V_i$ and all the possible seeds for G, computes $\sum_{v \in V_i} |\mathbb{E}_G[N_v]|$ and checks if it is at most ε. This can be done in space $O(s + \log(nw))$. If all such checks pass, we have by Lemma 5.1.7 that G $\varepsilon \cdot n$-fools B.

Otherwise, we find some $i < n$ such that $\sum_{v \in V_i} |\mathbb{E}_G[N_v]| > \varepsilon$. Let T be an OBP of length i that is the same as B from layer V_0 to V_i, such that the output label on each $v \in V_i$ is 1 if $\mathbb{E}_G[N_v] \geq 0$, and 0 if $\mathbb{E}_G[N_v] < 0$. Such an OBP is of size at most that of B, and can be constructed in space $O(s + \log(nw))$. We have

$$
\begin{aligned}
&\Pr_{x \sim G(U_s)}[T(x_{1..i}) = x_{i+1}] \\
&= \sum_{\substack{v \in V_i \\ \mathbb{E}_G[N_v] \geq 0}} \Pr_{x \sim G(U_s)}[B_{\to v}(x) = 1 \wedge x_{i+1} = 1] + \sum_{\substack{v \in V_i \\ \mathbb{E}_G[N_v] < 0}} \Pr_{x \sim G(U_s)}[B_{\to v}(x) = 1 \wedge x_{i+1} = 0] \\
&= \sum_{v \in V_i} \frac{1}{2} \left(\mathop{\mathbb{E}}_G[B_{\to v}] + \left| \mathop{\mathbb{E}}_G[N_v] \right| \right) \\
&> \frac{1}{2}(1 + \varepsilon).
\end{aligned}
$$

□

What remains is to construct an alleged PRG from the alleged hard function f, such that given its next-bit predictor we can efficiently reconstruct the circuit that computes f. Formally, we state the following theorem, which will be proved in the next section.

Theorem 5.1.8. *Given $\varepsilon > 0$, and a family of functions $f \in \mathsf{DSPACE}[m]$ where $f = \{f_m : \{0,1\}^m \to \{0,1\}\}_{m \in \mathbb{N}}$, there is a family of explicit generators $G : \{0,1\}^s \to \{0,1\}^n$ with $s = O(\log n)$ computable in space $O(\log n)$, and a deterministic logspace algorithm that, given $n \in \mathbb{N}$ and a $1/(8n)$-next-bit predictor $\mathcal{B}$ for G of size at most n^2 which is evaluable in space $O(\log n)$, outputs a circuit C of size $2^{\varepsilon \cdot m_0}$ for f_{m_0} with $m_0 = \Theta(\log n)$.*

For now, let us see how Theorem 5.1.1 follows from Lemma 5.1.5 and Theorem 5.1.8.

Proof of Theorem 5.1.1. Let $G : \{0,1\}^s \to \{0,1\}^n$ be the generator of Theorem 5.1.8 with error ε and function family f, and let $m = m_0$ be the instance size of f used to construct G.

We then apply Lemma 5.1.5 on B and G with $\varepsilon = 1/(4n)$. Of the two possible outcomes:

1. If it is certified that G $\varepsilon \cdot n$-fools B. In this case the algorithm computes and outputs $\mathbb{E}_G[B]$ which approximates $\mathbb{E}[B]$ within additive error $1/4$.

2. Otherwise we get for some $i < n$ an explicit OBP T of length i and width w, such that $\Pr_{x \sim U_s}[T(G(x)_{1..i}) = G(x)_{i+1}] > \frac{1}{2}(1+\varepsilon)$. In other words, T is an $\varepsilon/2 = 1/8n$ next-bit predictor against G of size at most n^2, and T can be evaluated in space $O(\log n)$. Then by Theorem 5.1.8, we can construct in space $O(\log n)$ a circuit C for f on inputs of size $m = \Theta(\log n)$ of size at most $2^{\varepsilon m}$. □

5.2 Efficiently Reconstructive Derandomization

We prove Theorem 5.1.8 in four stages. Following the framework of [IW97], we first assume that f is a (worst-case) hard function, and construct a PRG via hardness amplifications and the Nisan-Wigderson PRGs [NW94]. The detailed steps are slightly different from those in [IW97], and we adapt the following strategy:

1. From f, construct (by low-degree extension) a function f' that is hard-on-average on a 0.99 fraction of inputs.

2. From f', construct (by derandomized XOR Lemma) a function f'' (with multiple bits of output) that is hard-on-average on a $2^{-\Omega(m)}$ fraction of inputs.

3. From f', construct (by Goldreich-Levin) a function f'' with single-bit output that is hard-on-average on a $1/2 + 2^{-\Omega(m)}$ fraction of inputs.

4. Use f'' to instantiate a Nisan-Wigderson pseudorandom generator $G : \{0,1\}^s \to \{0,1\}^n$ for $s = O(m)$.

We make sure that f, f', f'' and G are all computable within $O(\log n)$ space.

Furthermore, we prove that every step can be made logspace reconstructive, in the sense that given a counterexample to the conclusion (i.e. a small circuit that obtains some advantage) we can produce a counterexample to the assumption in deterministic logspace. This requires modifying the standard reconstruction algorithms for the first three steps, all of which use randomness-inefficient applications of the probabilistic method. Over the next four subsections, we state and prove the necessary components of the reconstructive PRG, and in Section 5.2.6, combine these results to conclude Theorem 5.1.8.

5.2.1 Derandomization Toolbox

First, we recall some notation related to the advantage of circuits.

Definition 5.2.1. *Given $f : \{0,1\}^n \to \{0,1\}^m$ and a circuit C, let*

$$\mathtt{SUC}(C, f) = \Pr_{x \sim U_n}[C(x) = f(x)].$$

For $m = 1$, let $\mathtt{ADV}(C, f) = 2\mathtt{SUC}(C, f) - 1$.

We will repeatedly make use of an averaging sampler in order to make probabilistic method arguments randomness efficient. We first recall the definition of an averaging sam-

pler, and then recall the classical result in [RVW01] that there exist highly efficient averaging samplers, even with exponentially small error.

Definition 5.2.2. *Given $m \in \mathbb{N}$ and $\varepsilon, \delta > 0$, we say that $\mathtt{SAMP} : \{0,1\}^\ell \to (\{0,1\}^m)^t$ is a t-query (m, ε, δ)-averaging sampler with seed length ℓ if for every $g : \{0,1\}^m \to [0,1]$ we have*

$$\Pr_{q_1,\ldots,q_t \sim \mathtt{SAMP}(U_\ell)} \left[\left| \mathop{\mathbb{E}}_{i \in [t]} [g(q_i)] - \mathbb{E}[g] \right| \leq \varepsilon \right] \geq 1 - \delta.$$

Proposition 5.2.3 ([RVW01])**.** *Given $m \in \mathbb{N}$ and $\varepsilon > 0$, there exists $t = \mathrm{poly}(m/\varepsilon)$ and a t-query $(m, \varepsilon, 2^{-2m})$-averaging sampler with seed length $4m$. Moreover, the sampler is evaluable in space $O(m)$.*

Another tool that is repeatedly used in our proof is the combinatorial design, which is a family of subsets $S_1, \ldots, S_n \subseteq [s]$ such that $|S_i| = \alpha s$ for some constant $\alpha \in (0,1)$ and all $i \in [n]$, while $|S_i \cap S_j| \leq 2\alpha^2 s$ for all $i \neq j$. The design will be used at two places: once in derandomized XOR Lemma (Section 5.2.3) and once in the Nisan-Wigderson PRG (Section 5.2.5). While the application in Section 5.2.3 only requires a linear-sized design, the application in Section 5.2.5 requires an exponential-sized design that is deterministically constructible in linear pace. The later was formally given in [KvM02], so we concurrently use it for both applications.

Proposition 5.2.4 ([KvM02])**.** *For every $\alpha \in (0,1)$, there is $\beta \in (0,1)$ such that for $s \in \mathbb{N}$ one can deterministically generate in space $O(s)$ a combinatorial design of size $n = 2^{\beta s}$ over $[s]$, that is, a family of subsets $S_1, \ldots, S_n \subseteq [s]$ such that $|S_i| = \alpha s$ and $|S_i \cap S_j| \leq 2\alpha^2 s$ for all $1 \leq i < j \leq n$.*

5.2.2 Derandomizing the Polynomial Decoder

For step (a) in Theorem 5.1.8, we need to convert a worst-case hard function to one with constant average-case hardness.

Lemma 5.2.5. *Given $f : \{0,1\}^m \to \{0,1\}$, there is $g : \{0,1\}^{m'} \to \{0,1\}$ where $m' = \Theta(m)$ such that, for every circuit $\mathcal{B}$ such that $\mathtt{SUC}(\mathcal{B}, g) > 0.99$, there is a circuit $\mathcal{C}$ of size $m^{O(1)} \cdot |\mathcal{B}|$ such that*

$$\mathcal{C}(x) = f(x), \ \forall x \in \{0,1\}^m.$$

Moreover, when f is computable in space $O(m)$, g is also computable in space $O(m)$, and there is a deterministic $O(m)$-space algorithm that, given the circuit $\mathcal{B}$ which is evaluable in space $O(m)$, prints $\mathcal{C}$, and $\mathcal{C}$ is also evaluable in space $O(m)$.

The proof for Lemma 5.2.5 is inspired by [STV01], where we encode f through Reed-Muller codes and switch to boolean domain via Hadamard codes. However, since we only need the resulting function to be average-case hard on a constant fraction of inputs, the code can be directly decoded instead of list-decoded, and we derandomize the decoding procedure with samplers.

We need the following two facts. The first is a folklore fact on constructing low-degree extension, whose proof can be found at [GKR15, Proposition 2.2]:

Proposition 5.2.6. *Given a finite field $\mathbb{F}$ and a subset $H \subseteq \mathbb{F}$, and oracle access to a function $f : H^\ell \to \{0,1\}$, one can compute in space $O(\log |\mathbb{F}| + \log \ell)$ an ℓ-variable polynomial $p : \mathbb{F}^\ell \to \mathbb{F}$ that coincides with f on H^ℓ, and the degree of p in each variable is smaller than $|H|$.*

The second fact concerns decoding Reed-Solomon codes:

Proposition 5.2.7. *Given a finite field $\mathbb{F}$ with $|\mathbb{F}| = N$, whose elements can be canonically listed as $a_1, \ldots, a_N$ where $a_1 = 0$, there exists a circuit $\mathtt{DEC} : \mathbb{F}^N \to \mathbb{F}^N$ that satisfies the following: If there exists a univariate polynomial $q : \mathbb{F} \to \mathbb{F}$ of degree at most $d < N$, such that $q(a_i) = b_i$ for at least $(N+d)/2$ of $i \in [N]$, then*

$$\mathtt{DEC}(b_1, \ldots, b_N) = (q(a_1), \ldots, q(a_N)).$$

Furthermore, DEC *is of size* $\mathrm{poly}(N)$ *and depth* $\mathrm{polylog}(N)$*, and can be uniformly constructed in space $O(\log N)$ given the arithmetics in $\mathbb{F}$.*

Proof. The circuit DEC instantiates the Berlekamp-Welch algorithm [WB86, GS92]. The algorithm involves solving systems of $O(N)$ linear equations on $O(N)$ variables, for which Csanky's algorithm [Csa76] can be implemented in logspace-uniform-NC. □

Proof of Lemma 5.2.5. We assume without loss of generality that m is a power of 2. Let $\ell = m/\log m$, and $\mathbb{F}$ be a finite field of characteristic 2 and size m^2. Take $H \subset \mathbb{F}$ to be a subset of size m, and we identify the domain $\{0,1\}^m$ of f with H^ℓ as $2^m = |H|^\ell$. The arithmetics in $\mathbb{F}$ can be done in time $O(|\mathbb{F}|)$ and space $O(m)$, and so does the bijection between $\{0,1\}^m$ and H^ℓ (and its reverse).

Let $p : \mathbb{F}^\ell \to \mathbb{F}$ be the polynomial in Proposition 5.2.6, and let $g : \mathbb{F}^{\ell+1} \to \{0,1\}$ be the function defined as

$$g(x_1, \ldots, x_\ell, y) = \langle p(x_1, \ldots, x_\ell), y \rangle,$$

where $\langle \cdot, \cdot \rangle$ stands for inner product in $\mathbb{F}_2$ when taking the binary representation of the

two arguments in $\mathbb{F}$. It is clear that g can be computed in space $O(m)$, and the input of g has length $(\ell+1)\log|\mathbb{F}| = O(m)$ when represented in binary.

Now assume there is a circuit $\mathcal{B}$ such that $\mathtt{SUC}(\mathcal{B}, g) > 0.99$. We first construct the circuit $\mathcal{B}' : \mathbb{F}^\ell \to \mathbb{F}$ such that the i-th bit of the output is

$$\mathcal{B}'_i(x_1, \ldots, x_\ell) = \mathtt{MAJ}_{z\in\mathbb{F}}(\mathcal{B}(x_1, \ldots, x_\ell, e_i + z) - \mathcal{B}(x_1, \ldots, x_\ell, z)).$$

Here e_i is the element in $\mathbb{F}$ whose binary representation has 1 on the i-th bit and 0 elsewhere.

Claim 5.2.8. $\mathtt{SUC}(\mathcal{B}', p) \geq 0.96$.

Proof. Since $\mathtt{SUC}(\mathcal{B}, g) > 0.99$, there are at least a 0.96-fraction of $(x_1, \ldots, x_\ell) \in \mathbb{F}^\ell$ such that $\mathcal{B}$ coincide with g on more than $3/4$ of $y \in \mathbb{F}$, which contains both z and $(e_i + z)$ with probability larger than $1/2$ for a random $z \in \mathbb{F}$. In such cases we have $\mathcal{B}'_i(x_1, \ldots, x_\ell) = \langle p(x_1, \ldots, x_\ell), e_i\rangle$ for every i, and thus $\mathcal{B}'(x_1, \ldots, x_\ell) = p(x_1, \ldots, x_\ell)$. □

From $\mathcal{B}'$, we reconstruct the circuit $\mathcal{C} : \{0,1\}^m \to \{0,1\}$ as follows. Let $\mathtt{SAMP} : \{0,1\}^{8m} \to (\mathbb{F}^\ell)^t$ be the sampler in Proposition 5.2.3 with $\varepsilon = 0.01$ and thus $t = \mathrm{poly}(m)$. We think of $\mathtt{SAMP}$ as sampling t random vectors $v = (v_1, \ldots, v_\ell) \in \mathbb{F}^\ell$, and given the input $x = (x_1, \ldots, x_\ell) \in H^\ell$ for $\mathcal{C}$, each vector v represents a line $\{x + \lambda v \mid \lambda \in \mathbb{F}\}$. On each line, $p(x + \lambda v)$ is a univariate polynomial on λ of degree at most $\ell|H| = m^2/\log m$, and we use the decoder circuit $\mathtt{DEC}$ in Proposition 5.2.7 to decode the Reed-Solomon code given by $\mathcal{B}'$ on the line. We let the value of $\mathcal{C}(x)$ to be the most common (breaking ties arbitrarily) decoded value among the t lines. Notice that this process depends on the seed of the sam-

pler, and we actually go through all the seeds and choose the one that makes $\mathcal{C}(x)$ correctly compute f on all $x \in H^\ell$.

Formally, we present this linear space reconstruction algorithm as Algorithm 5.1.

Algorithm 5.1: Reconstruction Algorithm in Lemma 5.2.5

Let $\mathtt{SAMP} : \{0,1\}^{8m} \to (\mathbb{F}^\ell)^t$ be the sampler of Proposition 5.2.3 with $\varepsilon = 0.01$.
for $y \in \{0,1\}^{8m}$ **do**
  let $v_1, \ldots, v_t \leftarrow \mathtt{SAMP}(y)$.
  Let $\mathcal{C} : \{0,1\}^m \to \{0,1\}$ be the circuit

$$\mathcal{C}(x) = \mathtt{MAJ}_{i\in[t]}(\mathtt{DEC}_1((\mathcal{B}'(x+\lambda v_i))_{\lambda\in\mathbb{F}})).$$

  if $\mathcal{C}(x) = f(x)$ *for all* $x \in \{0,1\}^m$ **then return** $\mathcal{C}$.

The circuit $\mathcal{C}$ constructed in the algorithm is of size $2t|\mathbb{F}|^2|\mathcal{B}| + m^{O(1)} = m^{O(1)} \cdot |\mathcal{B}|$, and has additional depth $\mathrm{polylog}(m)$ compared to that of $\mathcal{B}$. Therefore $\mathcal{C}$ can be evaluated in space $O(m)$.

Now we prove that the algorithm always returns a valid circuit $\mathcal{C}$. Notice that for uniformly random $v \in \mathbb{F}^\ell$, $x + \lambda v$ is also uniformly random after given x and $\lambda \neq 0$. Since $\mathtt{SUC}(\mathcal{B}', p) \geq 0.96$, it means that there are at least a 0.84-fraction of $v \in \mathbb{F}^\ell$ such that $\mathcal{B}'$ coincide with p on $x + \lambda v$ for at least $3/4$ of $\lambda \in \mathbb{F}, \lambda \neq 0$. Recall that the degree of $q(\lambda) = p(x + \lambda v)$ is at most $\ell|H| = |\mathbb{F}|/\log m$, and therefore by Proposition 5.2.7 we conclude that for every $x \in \{0,1\}^\ell$,

$$\Pr_{v\in\mathbb{F}^\ell}[\mathtt{DEC}((\mathcal{B}'(x+\lambda v))_{\lambda\in\mathbb{F}}) = (p(x+\lambda v))_{\lambda\in\mathbb{F}}] \geq 0.84,$$

in which case we have $\mathtt{DEC}_1((\mathcal{B}'(x+\lambda v))_{\lambda\in\mathbb{F}}) = p(x)$. Viewing this probability as an expectation of the indicator function on v, by the guarantee of the sampler in Proposition 5.2.3

we have

$$\Pr_{v_1,\ldots,v_t \sim \mathtt{SAMP}(U_{8m})}\left[\Pr_{i\in[t]}\left[\mathtt{DEC}_1((\mathcal{B}'(x+\lambda v_i))_{\lambda\in\mathbb{F}}) = p(x)\right] \geq 0.51\right] \geq 1 - 2^{-4m}.$$

By a union bound over $x \in \{0,1\}^m$, there must exist a $y \in \{0,1\}^{8m}$ such that $\mathcal{C}(x) = p(x) = f(x)$ for all $x \in \{0,1\}^m$. Therefore the algorithm always returns such a circuit $\mathcal{C}$. Moreover, the algorithm can be implemented to run in space $O(m)$, as we can enumerate over seeds to the sampler and construct the circuit (as a function of the sampler output) in space $O(m)$, and test if the circuit correctly computes f in this space bound. □

5.2.3 Derandomizing the Derandomized XOR Lemma

Our next step follows the approach of Impagliazzo and Wigderson [IW97], who use a derandomized XOR lemma to produce from a function that is hard on a constant fraction of inputs, a function that is hard on any exponentially small fraction of inputs. The construction is identical to the one in [IW97], except that we modify the reconstruction algorithm and analysis to make the circuit $\mathcal{C}$ constructible in deterministic space $O(m)$.

Lemma 5.2.9. *For every $\gamma \in (0,1)$, there is an $O(m)$-space computable function $G : \{0,1\}^{m'} \to (\{0,1\}^m)^m$, where $m' = \Theta(m/\gamma)$, that satisfies the following: Given $f : \{0,1\}^m \to \{0,1\}$, and a circuit $\mathcal{B}$ satisfying $\mathtt{SUC}(\mathcal{B}, f^n \circ G) \geq 2^{-\gamma m}$, there exists a circuit $\mathcal{C}$ of size $2^{O(\gamma m)} \cdot |\mathcal{B}|$ such that*

$$\mathtt{SUC}(\mathcal{C}, f) > 0.99.$$

Moreover, when f is computable in space $O(m)$, there is a deterministic $O(m)$-space algorithm

that, given the circuit $\mathcal{B}$ which is evaluable in space $O(m)$, prints $\mathcal{C}$, and $\mathcal{C}$ is also evaluable in space $O(m)$.

We first give the construction of the function G, which is called a *direct-product generator* in [IW97]. As in [IW97], it consists of two components: an expander walk and a combinatorial design. For the expander walk, we need an explicit expander where the neighbors of a vertex can be efficiently computed:

Proposition 5.2.10 (see e.g. [LPS88]). *There is a constant $\lambda \in (0,1)$, such that for every $m \in \mathbb{N}$, there exists a 4-regular graph E_m on the vertex set $\{0,1\}^m$ with spectral expansion (second largest eigenvalue of the normalized adjacency matrix) at most λ, such that given any vertex $v \in \{0,1\}^m$, its neighbors can be computed in time $\mathrm{poly}(m)$ and space $O(\log m)$.*

Define the expander walk function $\mathtt{EW} : \{0,1\}^{3m} \to (\{0,1\}^m)^m$ as follows: Given the input $v \in \{0,1\}^m$ and $d = (d_1, \ldots, d_m) \in [4]^m$, the output is sequence of vertices $v_1, \ldots, v_m$ in E_m that starts with $v_1 = v$, and take v_{i+1} to be the d_i-th neighbor of v_i. On the other hand, let $S_1, \ldots, S_m \subseteq [s]$ be the first m sets in the combinatorial design from Proposition 5.2.4 with $\alpha = \gamma/2$ and $s = m/\alpha$. Then we defined the function $G : \{0,1\}^{3m+s} \to (\{0,1\}^m)^m$ as:

$$G(r, v, d) = \big((r|_{S_1}) \oplus \mathtt{EW}(v,d)_1, \ldots, (r|_{S_m}) \oplus \mathtt{EW}(v,d)_m\big).$$

Here $r|_S$ is the part of $r \in \{0,1\}^s$ on indices S, and $\oplus$ is bit-wise XOR. From the definition we have that G can be computed in time $\mathrm{poly}(m)$ and space $O(m)$. The input length of G is $m' = 3m + 2m/\gamma = O(m/\gamma)$.

Now given $f : \{0,1\}^m \to \{0,1\}$, assume there is a circuit $\mathcal{B}$ such that $\mathtt{SUC}(\mathcal{B}, f^n \circ G) \geq 2^{-\gamma m}$. Before we move on and show how to reconstruct the circuit $\mathcal{C}$ efficiently and deterministically from $\mathcal{B}$, let us first review the reconstruction step in [IW97]. For $i \in [m]$, $x \in \{0,1\}^m$, $a \in \{0,1\}^{s-m}$, $v \in \{0,1\}^m$ and $d \in [4]^m$, let $h(i,x,a,v,d) = (r,v,d)$ where $r \in \{0,1\}^s$ such that

$$r|_{S_i} = x \oplus \mathtt{EW}(v,d)_i \text{ and } r|_{\overline{S_i}} = a.$$

The function h is called the *restricting function* of G. Given $x \in \{0,1\}^m$, with i, a, v and d chosen uniformly at random, they build a circuit $\mathcal{F}$ that first simulates $\mathcal{B}$ to compute $\mathcal{B}(h(i,x,a,v,d)) = (y_1, \ldots, y_m)$. Then it computes a number t defined as

$$t = \left|\left\{j \neq i \mid y_j \neq f\big(G_j \circ h(i,x,a,v,d)\big)\right\}\right|,$$

and outputs y_i with probability 2^{-t}, while outputting a random bit with probability $1 - 2^{-t}$. To compute t, for each $j \neq i$, $f(G_j \circ h(i,x,a,v,d))$ is computed through a non-uniformly constructed look-up table for f of size $2^{\gamma m}$, containing the values of $f(x_j)$ for all possible j-th output x_j of $G \circ h$ with the fixed i, a, v and d.

We could not resort to non-uniformity to construct the look-up table. Nevertheless, when f is computable in space $O(m)$, we can compute the entire table in space $O(m)$ and hardwire it to the circuit. Even better, when i, a, v and d are given, each output x_j of $G \circ h$ is fixed except for γm bits (corresponding to the coordinates in $S_i \cap S_j$), so we only need to go through all $2^{\gamma m}$ possibilities for these bits to compute the table.

The circuit $\mathcal{F}$ presented above uses a string R of $|R| = O(m)$ random bits, including i, a, v, d along with $w \in \{0,1\}^{m+1}$, the randomness used to decide the final output. It was

proved in [IW97] that:

Proposition 5.2.11 ([IW97, Theorem 15]). *Suppose that* $\mathtt{SUC}(\mathcal{B}, f^n \circ G) \geq 2^{-\gamma m}$. *There exists* $c > 0$ *(that depends on* γ*), such that the fraction of inputs* $x \in \{0,1\}^m$ *with*

$$\Pr_R[\mathcal{F}(x,R) = f(x)] \geq 1/2 + 2^{-\gamma m}/c$$

is more than 0.99.

Therefore, the final circuit $\mathcal{C}$ takes $O(m \cdot 2^{2\gamma m})$ independent copies of $\mathcal{F}$ and outputs their majority, and there exists a fixing of the randomness that provides the final deterministic circuit $\mathcal{C}$. We could not afford to store exponentially many random bits if they are independently sampled. Instead, we employ the efficient sampler in Proposition 5.2.3 that uses only $O(m)$ random bits as the seed to generate $2^{O(\gamma m)}$ samples, and we can enumerate over all the seeds to find the one that makes $\mathtt{SUC}(\mathcal{C}, f) > 0.99$. As shown in the proof below, such seed always exists.

Algorithm 5.2: Reconstruction Algorithm in Lemma 5.2.9

1 Let $\mathtt{SAMP} : \{0,1\}^{4|R|} \to (\{0,1\}^{|R|})^t$ be the sampler of Proposition 5.2.3 with $\varepsilon = 2^{-\gamma m}/(2c)$.
2 **for** $y \in \{0,1\}^{4|R|}$ **do**
3 Let $R_1, \ldots, R_t \leftarrow \mathtt{SAMP}(y)$.
4 Let $\mathcal{C} : \{0,1\}^m \to \{0,1\}$ be the circuit
$$\mathcal{C}(x) = \mathtt{MAJ}(\mathcal{F}(x,R_1), \ldots, \mathcal{F}(x,R_t)).$$
5 **if** $\mathtt{SUC}(\mathcal{C}, f) > 0.99$ **then return** $\mathcal{C}$

Proof of Lemma 5.2.9. Let $\mathcal{F} : \{0,1\}^{m+|R|} \to \{0,1\}$ be the circuit described above, and

$c > 0$ be the constant in Proposition 5.2.11. We give the formal description of the linear-space algorithm for the reconstruction procedure as Algorithm 5.2.

By Proposition 5.2.3 we have $t = \mathrm{poly}(m/\varepsilon) = 2^{O(\gamma m)}$ for $\varepsilon = 2^{-\gamma m}/(2c)$. From the description we know that $\mathcal{F}$ has size $|\mathcal{B}| + 2^{\gamma m} \cdot m^{O(1)}$, and therefore $\mathcal{C}$ has size $t|\mathcal{F}| + m^{O(1)} = 2^{O(\gamma m)} \cdot |\mathcal{B}|$. When $\mathcal{B}$ is evaluable in space $O(m)$, $\mathcal{C}$ is clearly also evaluable in space $O(m)$.

By the guarantee of the averaging sampler in Proposition 5.2.3, for every $x \in \{0,1\}^m$:

$$\Pr_{R_1,\ldots,R_t \sim \mathtt{SAMP}(U_{4|R|})}\left[\left|\mathop{\mathbb{E}}_{i\in[t]}[\mathcal{F}(x,R_i)] - \mathop{\mathbb{E}}_{R}[\mathcal{F}(x,R)]\right| \le \varepsilon\right] \ge 1 - 2^{-2|R|}.$$

By Proposition 5.2.11, there exists a subset $V \subseteq \{0,1\}^m$ such that $|V| > 0.99 \cdot 2^m$, such that for every $x \in V$:

$$\left|\mathop{\mathbb{E}}_{R}[\mathcal{F}(x,R)] - f(x)\right| \le 1/2 - 2^{-\gamma m}/c = 1/2 - 2\varepsilon.$$

Therefore for every $x \in V$, it is implied that

$$\Pr_{R_1,\ldots,R_t \sim \mathtt{SAMP}(U_{4|R|})}\left[\left|\mathop{\mathbb{E}}_{i\in[t]}[\mathcal{F}(x,R_i)] - f(x)\right| \le 1/2 - 2\varepsilon + \varepsilon\right] \ge 1 - 2^{-2|R|},$$

which means that

$$\Pr_{R_1,\ldots,R_t \sim \mathtt{SAMP}(U_{4|R|})}[\mathtt{MAJ}(\mathcal{F}(x,R_1),\ldots,\mathcal{F}(x,R_t)) = f(x)] \ge 1 - 2^{-2|R|} > 1 - 1/|V|.$$

By a union bound over $x \in V$, there must exist a $y \in \{0,1\}^{4|R|}$ such that $\mathcal{C}(x) = f(x)$ for all $x \in V$, which satisfies $\mathtt{SUC}(\mathcal{C}, f) > 0.99$. Therefore the algorithm always returns a

valid $\mathcal{C}$. Moreover, the algorithm runs in space $O(m)$, as it enumerates the seeds of length $O(|R|) = O(m)$, constructs and evaluates the circuit $\mathcal{C}$ and makes oracle calls to f, which all can be done in space $O(m)$. □

5.2.4 Derandomizing the Goldreich-Levin Theorem

Lemma 5.2.12. *Given $f : \{0,1\}^m \to \{0,1\}^m$, let $g : \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ be defined as $g(x,r) = \langle f(x), r\rangle$. Then, given $\delta > 0$, there is $\delta' \geq \Omega(\delta^3/m)$ so that, for every $\mathcal{B}$ satisfying $\mathtt{ADV}(\mathcal{B}, g) > \delta$, there is a circuit $\mathcal{C}$ of size at most $|\mathcal{B}| \cdot (m/\delta)^{O(1)}$ satisfying*

$$\mathtt{SUC}(\mathcal{C}, f) > \delta'.$$

Moreover, when f is computable in space $O(m)$, there is a deterministic $O(m)$-space algorithm that, given the circuit $\mathcal{B}$ which is evaluable in space $O(m)$, prints $\mathcal{C}$, and $\mathcal{C}$ is also evaluable in space $O(m)$.

Note that the original Goldreich-Levin theorem [GL89] does not guarantee (and in fact does not give) an efficient deterministic reconstructor, as it is not randomness efficient. A later work of Hoza and Klivans [HK18] achieves this, though with a significantly more involved proof. As such, we directly show this using small-bias spaces, which we define now:

Definition 5.2.13. *A function $G : \{0,1\}^t \to \{0,1\}^k$ is an ε-biased generator if $G(U_t)$ is a ε-biased probability space over $\{0,1\}^k$, which formally means that for every $T \in \{0,1\}^k$,*

$$\Pr_{y \sim U_t}[\langle T, G(y)\rangle = 1] \in [1/2 - \varepsilon, 1/2 + \varepsilon].$$

We recall that small-bias generators exist with good seed length, and moreover these generators can be evaluated in small space:

Proposition 5.2.14 ([NN93]). *Given $k \in \mathbb{N}$ and $\varepsilon > 0$, there is an $O(t)$-space evaluable ε-biased generator $\mathtt{BIAS} : \{0,1\}^t \to \{0,1\}^k$ with seed length $t = O(\log(k/\varepsilon))$.*

We require a basic Fourier-analytic lemma, that states that a small-bias space fools the conjunction of k parities.

Lemma 5.2.15. *Let $\mathtt{BIAS} : \{0,1\}^t \to \{0,1\}^k$ be an ε-biased generator. Then for every collection $T_1, \ldots, T_d \in \{0,1\}^k$ and $v_1, \ldots, v_d \in \{0,1\}$ we have*

$$\left| \mathop{\mathbb{E}}_{r \sim \mathtt{BIAS}(U_t)} \left[\bigwedge_{i \in [d]} \left(\langle T_i, r \rangle \oplus v_i \right) \right] - \mathop{\mathbb{E}}_{r \sim U_k} \left[\bigwedge_{i \in [d]} \left(\langle T_i, r \rangle \oplus v_i \right) \right] \right| \leq 2\varepsilon.$$

Proof. We have

$$\begin{aligned} \bigwedge_{i \in [d]} \left(\langle T_i, r \rangle \oplus v_i \right) &= 1 - 2 \cdot 2^{-d} \sum_{S \subseteq [d]} \bigoplus_{i \in S} \neg \left(\langle T_i, r \rangle \oplus v_i \right) \\ &= 1 - 2 \cdot 2^{-d} \sum_{S \subseteq [d]} \left(\left\langle \bigoplus_{i \in S} T_i, r \right\rangle \oplus \bigoplus_{i \in S} \neg v_i \right) \end{aligned}$$

and as $\mathtt{BIAS}$ fools all such parities to error ε in the summation over $S \subseteq [d]$, we have that the total error is at most 2ε. □

Proof of Lemma 5.2.12. If $\delta < 2^{-m}$, we can choose $\delta' = 2^{-m}$ and the lemma trivially holds for a circuit C outputting a constant. Therefore, from now on we assume that $\delta \geq 2^{-m}$. We formally state our algorithm as Algorithm 5.3, with δ' to be determined later. Note that

$\ell = O(m)$, and therefore in the ε-biased generator $\mathtt{BIAS} : \{0,1\}^t \to \{0,1\}^{\ell \times m}$ we have $t = O(\log(\ell m/\varepsilon)) = O(m)$ with $\varepsilon = 2^{-4m-1}$, and the algorithm runs in space $O(t + \ell + m) = O(m)$.

Algorithm 5.3: Reconstruction Algorithm in Lemma 5.2.12

Let $\ell \leftarrow \lceil \log_2(128m/\delta^2 + 1) \rceil$.
Let $\mathtt{BIAS} : \{0,1\}^t \to \{0,1\}^{\ell \times m}$ be the generator of Proposition 5.2.14 with $\varepsilon = 2^{-4m-1}$.
for $y \in \{0,1\}^t$ **do**
    Let $r_1, \ldots, r_\ell \leftarrow \mathtt{BIAS}(y)$.
    for $(b_1, \ldots, b_\ell) \in \{0,1\}^\ell$ **do**
        Let $\mathcal{C} : \{0,1\}^m \to \{0,1\}^m$ be the circuit that for each $i \in [m]$:

$$\mathcal{C}_i(x) = \mathtt{MAJ}_{J \subseteq [\ell] : J \neq \emptyset}(b^J \oplus \mathcal{B}(x, r^J \oplus e_i)).$$

        if $\mathtt{SUC}(\mathcal{C}, f) > \delta'$ **then return** $\mathcal{C}$.

We view the output of $\mathtt{BIAS}$ as a tuple of ℓ vectors:

$$\mathtt{BIAS}(y) = (r_1, \ldots, r_\ell), \quad r_i \in \{0,1\}^m.$$

For convenience, let $\vec{r} := (r_1, \ldots, r_\ell)$ and $\vec{b} := (b_1, \ldots, b_\ell)$. For every $J \subseteq [\ell]$, let:

$$r^J := \bigoplus_{i \in J} r_i, \qquad b^J = \bigoplus_{i \in J} b_i.$$

Note that in the original GL algorithm, all r_i's are i.i.d. uniformly over $\{0,1\}^m$. We first argue that our distribution over r^J's satisfies (approximately) the two properties used in the analysis of the original algorithm:

Claim 5.2.16. *The following two properties hold:*

1. *For every non-empty J, r^J is 2^{-2m}-close to U_m in ℓ_1-distance.*

2. *For every non-empty J and J' where $J \neq J'$, $(r^J, r^{J'})$ is 2^{-2m} close to U_{2m} in ℓ_1-distance.*

Proof. For $i \in [m]$, the i-th bit of r^J can be written as $\langle T_{i,J}, \mathtt{BIAS}(y)\rangle$ where $T_{i,J}$ indicates a non-empty subset of bits. From Lemma 5.2.15 we know that for every $v \in \{0,1\}^m$,

$$\left| \Pr_{r\sim \mathtt{BIAS}(U_t)}[r^J = v] - \Pr_{r\sim U_{\ell m}}\left[\bigwedge_{i\in[m]} (\langle T_{i,J}, r\rangle = v_i)\right]\right| \leq 2\varepsilon.$$

Notice that $\{T_{i,J}\}_{i\in[m]}$ are linearly independent, and thus $(\langle T_{i,J}, r\rangle)_{i\in[m]}$ is uniformly distributed over $\{0,1\}^m$. Therefore taking the sum over $v \in \{0,1\}^m$ we have that r^J is $2\varepsilon \cdot 2^m \leq 2^{-2m}$-close to U_m in ℓ_1 distance.

When $J \neq J'$ are both non-empty, $\{T_{i,J}\}_{i\in[m]} \cup \{T_{i,J'}\}_{i\in[m]}$ are still linearly independent. For the same reason as above, $(r^J, r^{J'})$ is $2\varepsilon \cdot 2^{2m} = 2^{-2m}$-close to U_{2m} in ℓ_1 distance. □

Now recall that for $i \in [m]$ the ith bit of the output of $\mathcal{C}$ is

$$\mathcal{C}_i(x) = \mathtt{MAJ}_{J:J\neq\emptyset}(b^J \oplus \mathcal{B}(x, r^J \oplus e_i)).$$

Thus $\mathcal{C}$ has size $|\mathcal{C}| \leq (|\mathcal{B}| + O(\ell)) \cdot 2^\ell m \leq |\mathcal{B}| \cdot O(2^\ell \ell m) = |\mathcal{B}| \cdot (m/\delta)^{O(1)}$ as claimed. To analyze the performance of $\mathcal{C}$, let

$$S := \{x \in \{0,1\}^m : \Pr_{z\sim U_m}[\mathcal{B}(x,z) = g(x,z)] \geq 1/2 + \delta/2\}.$$

By a standard averaging argument, $|S| \geq (\delta/2) \cdot 2^m$.

Claim 5.2.17. *For every $x \in S$ and $i \in [m]$,*

$$\Pr_{(r_1,\ldots,r_\ell)\sim \mathtt{BIAS}(U_t)}\left[\left|\{J : \mathcal{B}(x, r^J \oplus e_i) = g(x, r^J \oplus e_i)\}\right| \leq \frac{1}{2}(2^\ell - 1)\right] \leq \frac{1}{2m}.$$

Proof. For the remainder of the proof we fix x and i. Let $A \subset \{0,1\}^m$ be the set of values r on which $\mathcal{B}(x, r) = g(x, r)$. By the fact that $x \in S$ we have $|A| \geq (1/2 + \delta/2) \cdot 2^m$. Furthermore, for each $y \in \{0,1\}^t$ (where y is the input to BIAS) let

$$\zeta_J(y) = \mathbb{I}[r^J \oplus e_i \in A]$$

and observe that $\zeta_J = 1$ is equivalent to $\mathcal{B}(x, r^J \oplus e_i) = g(x, r^J \oplus e_i)$, i.e. $\mathcal{B}$ computes the inner product with $f(x)$ correctly on that input. Now observe that by Claim 5.2.16,

$$\mathop{\mathbb{E}}_{y}[\zeta_J] = \Pr_y[\zeta_J(y) = 1] \geq 1/2 + \delta/2 - 2^{-2m} \geq 1/2 + \delta/4.$$

We now bound the variance of the number of such places where we compute the inner product correctly. Let

$$\begin{aligned}\sigma^2 = \mathrm{Var}\left(\sum_J \zeta_J\right) &= \sum_{J,J'} \mathrm{Cov}(\zeta_J, \zeta_{J'}) \\ &\leq \sum_J \mathrm{Var}(\zeta_J) + \sum_{J,J'} 2^{-2m} \\ &\leq 2^\ell + 2^{2\ell} \cdot 2^{-2m} \leq 2^{\ell+1}\end{aligned}$$

where the first inequality follows from Claim 5.2.16. Now the result follows by Cheby-

shev's inequality and a union bound. For convenience let $d = 2^\ell - 1$, and the probability in the claim equals:

$$\Pr_y\left[\sum_J \zeta_J \leq \frac{d}{2}\right] \leq \Pr_y\left[\left|\sum_J \zeta_J - \mathbb{E}[\zeta_J]\cdot d\right| \geq \left(\frac{d\delta}{4\sigma}\right)\cdot\sigma\right]$$
$$\leq \frac{16\sigma^2}{\delta^2(2^\ell-1)^2} \leq \frac{32\sigma^2}{\delta^2 2^{2\ell}} \leq \frac{64}{\delta^2 2^\ell} \leq \frac{1}{2m}. \qquad \square$$

Notice that when $\mathcal{B}(x, r^J \oplus e_i) = g(x, r^J \oplus e_i)$ and for every $j \in [\ell]$, $b_j = g(x, r_j)$, we have

$$b^J \oplus \mathcal{B}(x, r^J \oplus e_i) = g(x, r^J) \oplus g(x, r^J \oplus e_i) = g(x, e_i) = f_i(x).$$

Therefore using a union bound over $i \in [m]$ on Claim 5.2.17, we have that for every $x \in S$,

$$\begin{aligned}
&\Pr_{\substack{\vec{r}\sim \mathtt{BIAS}(U_t)\\ \vec{b}\sim U_l}}[\mathcal{C}(x) = f(x)] \\
&\geq \Pr_{\vec{r}\sim \mathtt{BIAS}(U_t)}\left[\forall i \in [m], \left|\{J : \mathcal{B}(x, r^J \oplus e_i) = g(x, r^J \oplus e_i)\}\right| > \frac{1}{2}(2^l - 1)\right] \\
&\quad \cdot \Pr_{\vec{b}\sim U_\ell}\left[\forall j \in [\ell], b_j = g(x, r_j)\right] \\
&\geq \frac{1}{2}\cdot \Pr_{\vec{b}\sim U_\ell}\left[\forall j \in [\ell], b_j = g(x, r_j)\right] \geq 2^{-\ell-1}.
\end{aligned}$$

Thus, there is an assignment of y and $\vec{b}$ such that $\mathcal{C}$ computes f correctly on at least $|S| \cdot 2^{-\ell-1} \geq 2^m \cdot \delta 2^{-\ell-2}$ inputs. Moreover, we can find such a circuit by enumerating the assignments to y and $\vec{b}$, and verifying the success probability by evaluating $\mathcal{C}$ and f over all

$x \in \{0,1\}^m$. Therefore letting

$$\delta' = \delta 2^{-\ell-2} = \Omega(\delta^3/m)$$

completes the proof. □

5.2.5 Space-Efficient Nisan-Wigderson PRG

We recall the argument of [KvM02] that there is a space-efficient implementation of the Nisan-Wigderson [NW94] PRG, using the linear-space constructible combinatorial design (Proposition 5.2.4). While we rephrase their result in our notation, we make no changes to the construction, as (in contrast to all other steps) the existing implementation satisfies our desired reconstruction property.

Lemma 5.2.18. *Given $\rho > 0$ and $n \in \mathbb{N}$ and a family of functions $f_m : \{0,1\}^m \to \{0,1\} \in \mathsf{DSPACE}[m]$, there exists an $m = \Theta(\log n)$ and $G : \{0,1\}^s \to \{0,1\}^n$ with $s = O(m)$ such that, given a circuit $\mathcal{B}$ which is a next-bit predictor for G with advantage ε, there is a circuit $\mathcal{C}$ of size $|\mathcal{B}| + O(n2^{\rho m})$ satisfying*

$$\mathtt{ADV}(\mathcal{C}, f_m) > \varepsilon.$$

Moreover, there is a deterministic $O(m)$-space algorithm that, given the circuit $\mathcal{B}$ which is evaluable in space $O(m)$, prints $\mathcal{C}$, and $\mathcal{C}$ is also evaluable in space $O(m)$.

Proof of Lemma 5.2.18. Fix $\alpha \in (0,1)$ such that $\alpha \leq \rho/2$, and let $\beta \in (0,1)$ be the constant in Proposition 5.2.4. Choose $s = O(\log n)$ such that $2^{\beta s} = n$, and let $m = \alpha s$.

Let $\mathcal{S} = (S_1, \ldots, S_n)$ be the design of Proposition 5.2.4 over $[s]$ with parameter α, and let $f_m : \{0,1\}^m \to \{0,1\}$ be the function on inputs of size $m = O(\log n)$.

We let $G(x) := f(x_{S_1})f(x_{S_2}) \ldots f(x_{S_n})$. Now suppose $\mathcal{B}$ is an ε-next-bit predictor for bit i of G, i.e.

$$\Pr_{x \sim U_s}[\mathcal{B}(G(x)_{1..i}) = G(x)_{i+1}] > \frac{1}{2} + \varepsilon.$$

Then let $S := S_{i+1}$ and $T := [s] \setminus S_{i+1}$ and write the above inequality as

$$\Pr_{(x_S, x_T) \sim U_s}[\mathcal{B}(G(x_S \cup x_T)_{1..i}) = f(x_S)] > \frac{1}{2} + \varepsilon.$$

For each fixing of x_T, we let the circuit $\mathcal{C}$ to be $\mathcal{C}(x_S) = \mathcal{B}(G(x_S \cup x_T)_{1..i})$. Then we have

$$\mathop{\mathbb{E}}_{x_T}[\mathtt{ADV}(\mathcal{C}, f_m)] > \varepsilon.$$

Thus, the algorithm can enumerate over all possible assignments to x_T in space $|T| = O(m)$, and for each assignment check the advantage of $\mathcal{C}$. Once the algorithm has found the fixing of x_T such that the restricted circuit has advantage at least ε, for every $j \leq i$, the j-th bit of the output of $G(x_S \cup x_T)$, which is $f(x_{S_j})$, depends on $|S \cap S_j| \leq 2\alpha^2 s = \rho m$ bits of x_S, and hence we can output a ($O(m)$-space constructible) circuit for $f(x_{S_j})$ of size at most $O(2^{\rho m})$, and hence the total size of $\mathcal{C}$ is at most $|\mathcal{B}| + O(n2^{\rho m})$. □

5.2.6 Putting It All Together

Proof of Theorem 5.1.8. Given ε, we first do the construction steps. For each $m \in \mathbb{N}$:

1. Let $f' : \{0,1\}^{m_1} \to \{0,1\}$ be the function g of Lemma 5.2.5 applied to f_m.

2. Let $f' : \{0,1\}^{m_2} \to \{0,1\}^{m_1}$ be the function $f^{m_1} \circ G$ of Lemma 5.2.9 applied to f_{m_1} with the constant γ to be chosen later.

3. Let $f'' : \{0,1\}^{m_3} \to \{0,1\}$ be the function g of Lemma 5.2.12 applied to f'_{m_2} with the constant δ to be chosen later.

4. Let $G : \{0,1\}^{s} \to \{0,1\}^{n}$ be the function of Lemma 5.2.18 applied to f''_{m_3} and $\mathcal{B}$ with the constant ρ to be chosen later.

Notice that m_1, m_2, m_3 and s are all $\Theta(m)$, and the functions f', f'', f''' and G are all computable in space $O(m)$.

Suppose now we are given a $1/(8n)$ next-bit predictor $\mathcal{B}$ for G of size n^2. As n is given, we decide the value of $m_3 = \Theta(\log n)$ through Lemma 5.2.18, which in turn decides the value of $m = \Theta(\log n)$. The reconstruction steps go as follows:

4. By Lemma 5.2.18, we can construct in space $O(m)$ a circuit $\mathcal{C}_3$ such that $\mathtt{ADV}(\mathcal{C}_3, f''_{m_3}) > 1/(8n)$, and $\mathcal{C}_3$ has size $s_3 = n^2 + O(n2^{\rho m_3}) \leq 2^{c_3 \rho m}$ for some constant $c_3 > 0$.

3. By Lemma 5.2.12, where we now set $\delta = 1/(8n)$, we can construct in space $O(m)$ a circuit $\mathcal{C}_2$ such that $\mathtt{SUC}(\mathcal{C}_2, f'_{m_2}) > \Omega(\delta^3/m_2) \geq 2^{-c_2 \rho m}$, and $\mathcal{C}_2$ has size $s_2 = s_3 \cdot (m_2/\delta)^{O(1)} \leq 2^{c_2 \rho m}$ for some constant $c_2 > 0$.

2. By Lemma 5.2.9, where we now set $\gamma = c_2 \rho$, we can construct in space $O(m)$ a circuit $\mathcal{C}_1$ such that $\mathtt{SUC}(\mathcal{C}_1, f_{m_1}) > 0.99$ and $\mathcal{C}_1$ has size $s_1 = s_2 \cdot 2^{O(\gamma m_1)} \leq 2^{c_1 \rho m}$ for some constant $c_1 > 0$.

1. By Lemma 5.2.5, we can construct in space $O(m)$ a circuit $\mathcal{C}$ such that $\mathcal{C}(x) = f_m(x)$ for every $x \in \{0,1\}^m$, and $\mathcal{C}$ has size $s = s_1 \cdot m^{O(1)} \leq 2^{c_0 \rho m}$ for some constant $c_0 > 0$.

By choosing $\rho = \varepsilon/c_0$, we obtain the final result. □

5.3 Universal Derandomization of BPL

Here we state the main theorem of this section, that there exists a universal derandomizer for logspace computation.

Theorem 5.3.1. *There is a deterministic Turing machine $\mathcal{U}$ such that:*

- *On input 1^n and an OBP B of length and width at most n, the output $\mathcal{U}(1^n, B)$ satisfying $|\mathcal{U}(1^n, B) - \mathbb{E}[B]| < n^{-1}$.*
- *For every space-constructible function $S : \mathbb{N} \to \mathbb{N}$ satisfying $S(n) \geq \log n$, $\mathcal{U}$ runs in space $O(S(n))$ if and only if $\mathsf{BPL} \subseteq \mathsf{DSPACE}(S)$.*

Notice that the condition $\mathsf{BPL} \subseteq \mathsf{DSPACE}(S)$ is stronger than say, there exists a PRG for BPL with seed length $O(S)$. Therefore we cannot simply enumerate Turing machines and use their outputs blindly as pseudorandomness. Instead, we use them as the outputs of a specific BPL-complete search problem.

5.3.1 A Pseudo-Deterministic BPL-Complete Problem

Definition 5.3.2. *The problem OBPRounding$_c$ with parameter $c \in \mathbb{N}$ is a promise search problem that outputs a real number in $[0, 1]$, defined as follows. Given an ordered branching program B of length n and width $\mathrm{poly}(n)$, and a shifting parameter r with the promise that*

$$\left|\mathbb{E}[B] - k \cdot n^{-c+2} + r\right| > \frac{n^{-c}}{6}, \ \forall k \in \mathbb{Z},$$

the problem asks to output a real number δ that satisfies $|\mathbb{E}[B] - \delta| \leq n^{-c+2}$. Further more, we require δ be pseudo-deterministic, that is, on every fixed input, the randomized algorithm that computes OBPRounding must output the same δ with probability at least $2/3$.

The promise means that $\mathbb{E}[B] + r$ is polynomially bounded away from every multiple of n^{-c+2}. We introduce the promise and the shifting parameter r to prevent the case when $\mathbb{E}[B]$ is very close to some multiple of n^{-c+2}, and it becomes hard to determining whether the expectation is above or below the cutoff. This is inspired by the approach of Saks and Zhou [SZ99].

Proposition 5.3.3. *OBPRounding$_c$ is* BPL*-complete under* L *reductions.*

Proof. Fix arbitrary $c \geq 3$. We first prove that OBPRounding$_c$ can be computed in BPL. We sample n^{2c+1} random walks on the branching program B, and let γ be the fraction of these walks which reach the acceptance state. Let $k \in \mathbb{Z}$ be the largest value such that $\gamma + r \geq k \cdot n^{-c+2}$, and return $\delta = k \cdot n^{-c+2}$. Since this algorithm clearly runs in randomized logspace, it suffices to show that, for B and r that satisfy the promise, there is some fixed k that the above algorithm identifies with probability over $2/3$. Note that by the promise, we have that for some $k_0 \in \mathbb{Z}$,

$$k_0 \cdot n^{-c+2} + \frac{n^{-c}}{6} < \mathbb{E}[B] + r < (k_0 + 1) \cdot n^{-c+2} - \frac{n^{-c}}{6}.$$

On the other hand, using concentration bounds we can show that with probability at least $2/3$,

$$|(\mathbb{E}[B] + r) - (\gamma + r)| = |\mathbb{E}[B] - \gamma| \leq \frac{n^{-c}}{6}.$$

In this case the algorithm always identifies $k = k_0$ since $k_0 \cdot n^{-c+2} < \gamma + r < (k_0 + 1) \cdot n^{-c+2}$.

We now prove that OBPRounding$_c$ is BPL-hard. Recall the standard BPL-complete problem: Given an OBP B of length and width n, determine if $\mathbb{E}[B] < 1/3$ or $\mathbb{E}[B] > 2/3$, where the promise is that one of these cases holds. We reduce this problem to OBPRounding as follows. Let $T_B : \{0,1\}^{dn} \to \{0,1\}$ be the OBP defined as

$$T_B(x_1, \ldots, x_d) = \mathtt{MAJ}(B(x_1), \ldots, B(x_d))$$

where $d = O(c \log n)$ such that if $\mathbb{E}[B] < 1/3$ then $\mathbb{E}[T_B] < n^{-c}/6$, and if $\mathbb{E}[B] > 2/3$ then $\mathbb{E}[T_B] > 1 - n^{-c}/6$. Observe that T_B has length and width polynomial in n, and is constructible in deterministic logspace given B. Let the input to OBPRounding$_c$ be (T_B, n^{-c}), which satisfies the promise. Hence if the answer is less than $1/2$ we determine that $\mathbb{E}[B] < 1/3$, and otherwise determine that $\mathbb{E}[B] > 2/3$. □

5.3.2 Universal Derandomization Algorithm

Intuitively, our universal derandomizer $\mathcal{U}$ enumerates over deterministic Turing machines $\langle i \rangle$, space bounds j, and shifting parameters r. At each step, it simulate $\langle i \rangle$ on input $(B_{\to v}, r)$ for every state v in the OBP B. If $\langle i \rangle$ ever touches more than j spaces on the work tape, $\mathcal{U}$ halts and increments i or j. Otherwise, we have a set of estimates $\{p_{\to v}\} = \{\langle i \rangle(1^n, B_{\to v})\}$ which can be generated on the fly in space $O(j + \log n)$. We then verify whether these estimates are close to the actual probabilities $\mathbb{E}[B_{\to v}]$, and return the estimate of the probability of reaching the accepting state if they pass the check.

Note that given the OBP B, we can construct in logspace the probability transition matrix corresponds to B. Therefore the verifier Algorithm 4.2 in Lemma 4.2.3 is able to accomplish the verification job. The verifier there was randomized, but we can make it deter-

ministic by going over all possibilities of the $O(\log n)$ random bits, as the estimates $\{p_{\to v}\}$ are not read-once. A simpler verifier with even better parameters can be found in [CH22], which we state below.

Proposition 5.3.4 ([CH22]). *There is a deterministic logspace algorithm $\mathcal{V}$ that takes as an OBP B of length n and width* $\mathrm{poly}(n)$*, and the estimates $\{p_{\to v}\}$. If for every state v of B, $|p_{\to v} - \mathbb{E}[B_{\to v}]| \leq n^{-3}$ the algorithm accepts, and moreover if the algorithm accepts, $|p_{\to v} - \mathbb{E}[B_{\to v}]| \leq n^{-1}$ for every v.*

We now give the formal description of the algorithm as Algorithm 5.4. By the soundness of the verifier $\mathcal{V}$, if $\mathcal{U}$ returns a value, it must be a good approximation of the acceptance probability of B. Therefore it suffices to show that the machine halts and runs in the desired space bound.

Algorithm 5.4: Universal derandomizer $\mathcal{U}(B)$

```
for j ← 0, 1 . . . , do
    for i ← 0, 1, . . . , j do
        for t ← 1, . . . , 2n^2 do
            let r ← t · n^{-5}/2;
            Compute b ← V(B, {⟨i⟩(1^n, B_{→v}, r)}_{v∈V(B)});
            whenever ⟨i⟩ uses more than j space or more than 2^j time do
                Abort the simulation of ⟨i⟩ and pass to the next r.
            if b = 1 then return ⟨i⟩(B, r).
```

Lemma 5.3.5. *For every space-constructible function $S : \mathbb{N} \to \mathbb{N}$ with $S(n) \geq \log n$, if* $\mathsf{BPL} \subseteq \mathsf{DSPACE}(S)$*, then $\mathcal{U}$ halts and runs in space $O(S(n))$.*

Proof. We prove that $\mathcal{U}(1^n, B)$ halts and returns a value with $i + j \leq c \cdot S(n)$ for an absolute constant c (in particular, $i, j < \infty$), which suffices to establish the lemma by the composition of space-bounded algorithms.

By Proposition 5.3.3, there is a Turing machine $\langle i \rangle$ that computes OBPROUNDING_c for $c = 5$ in $\mathsf{DSPACE}(S)$. We now show that there exists $r \in \{1 \cdot n^{-5}/2, 2 \cdot n^{-5}/2, \ldots, 2n^2 \cdot n^{-5}/2\}$ such that

$$\left|\mathbb{E}[B_{\to v}] - k \cdot n^{-3} + r\right| > n^{-5}/6 \tag{5.1}$$

for every k and v. There are n^2 different values $\mathbb{E}[B_{\to v}]$ over v in the vertex set $V(B)$ of the branching program, and for each v, there is at most one assignment to r such that (5.1) fails to hold for some $k \in \mathbb{Z}$. As there are $2n^2$ possible values for r, there must be one such that (5.1) holds for all k and v.

Finally, let $j = O(S(|B|))$ be such that $\langle i \rangle(B_{\to v}, r)$ halts using at most j space for every v. Such a j exists per assumption and the fact that the input $(B_{\to v}, r)$ satisfies the promise of Proposition 5.3.3 for every v. Thus, upon reaching the tuple (i, j, r), the set of estimates $p_{\to v} = \langle i \rangle(B_{\to v}, r)$ must satisfy $|p_{\to v} - \mathbb{E}[B_{\to v}]| \leq n^{-3}$ for every $v \in V(B)$. Then running $\mathcal{V}(B, \{p_{\to v}\}_{v \in V(B)})$ (where we wait for the test to request a particular value $p_{\to v}$ and then recompute it from $\langle i \rangle$, avoiding the need to store all n^2 values) will result in $\mathcal{V}$ accepting, and hence $\mathcal{U}$ halts in the claimed space bound. Moreover, the returned value $\delta = \langle i \rangle(B, r)$ satisfies that $|\delta - \mathbb{E}[B]| \leq n^{-1}$. □

We can now conclude the proof of Theorem 5.3.1.

Proof of Theorem 5.3.1. Let $\mathcal{U}$ be Algorithm 5.4. The fact that $|\mathcal{U}(1^n, B) - \mathbb{E}[B]| < n^{-1}$ follows from the soundness in Proposition 5.3.4 applied to $p_{\to v} = \langle i \rangle(B_{\to v}, r)$.

The direction that $\mathcal{U}$ runs in space $O(S(n))$ if $\mathsf{BPL} \subseteq \mathsf{DSPACE}(S)$ was proved in Lemma 5.3.5. To prove the other direction, notice that $\mathcal{U}$ actually deterministically solves OBPROUNDING_c for $c = 3$, which by Proposition 5.3.3 is BPL-hard. Therefore if $\mathcal{U} \in$

$\mathsf{DSPACE}(S)$ then $\mathsf{BPL} \subseteq \mathsf{DSPACE}(S)$. $\square$

6

Unitary Quantum Simulation

In this chapter we study the power of unitary quantum computing with bounded space. Most of the results stated here will be focusing on logarithmic space, but they generally work for any larger space bound as well.

The main problem we consider in this chapter is the simulation of space-bounded quantum computing with general quantum channels, i.e. BQL. This is captured by the following problem of powering matrices that represents quantum channels.

Definition 6.1. *In the* ChannelPowering *problem, given $n = 2^S$, a quantum channel $\Phi : \mathcal{L}(\mathbb{C}^n) \to \mathcal{L}(\mathbb{C}^n)$ in its natural representation $K(\Phi) \in \mathbb{C}^{n^2 \times n^2}$ and a positive integer T in unary, it is promised that $\mathrm{Tr}[\rho_0 \cdot \Phi^T(\rho_0)]$ is either in $[0, 1/3]$ or $[2/3, 1]$ for $\rho_0 = |0^S\rangle\langle 0^S|$, and the goal is to distinguish between the two cases.*

Notice that in general a quantum algorithm is specified by T channels $\Phi_1, \ldots, \Phi_T$ instead of one, but we could simply add a register of $O(\log T)$ qubits that stores the time stamp, and make

$$\Phi(\rho \otimes |t\rangle\langle t|) = \Phi_t(\rho) \otimes |t+1\rangle\langle t+1|.$$

The final measurement that measures the $|0^S\rangle$ state is also general enough for BQL, as mentioned in Section 2.2. Therefore we have:

Theorem 6.2. *ChannelPowering is* BQL*-hard.*

In the following sections, we will show how the ChannelPowering problem is computed with only unitary operators, how the error can be reduced in unitary logspace even for the problem of outputting the measurement probabilities, and its relations with quantum learning.

6.1 Unitary Quantum Logspace Algorithms

In this section we survey some of the important problems that can be solved in $\mathsf{BQ_UL}$, which at the end leads to the resolution of $\mathsf{BQL} = \mathsf{BQ_UL}$. Note that in all these results, the unitary quantum circuits we construct are uniform, and contains only unitary operators from a fixed universal gate set. We first state a simple but useful lemma regarding error propagation in quantum computing:

Lemma 6.1.1. *If $A_1, \ldots, A_k, B_1, \ldots, B_k \in \mathbb{C}^{n\times n}$ satisfy $\|A_i - B_i\| \leq \varepsilon$ for every i, and has bounded spectral norm for all partial products, that is*

$$\left\|A_iA_{i+1}\cdots A_j\right\|, \left\|B_iB_{i+1}\cdots B_j\right\| \leq \kappa, \quad \forall i \leq j$$

for some $\kappa \geq 1$, then $\|A_1\cdots A_k - B_1\cdots B_k\| \leq k\varepsilon\kappa^2$. In particular, when A_i and B_i are all unitary, $\|A_1\cdots A_k - B_1\cdots B_k\| \leq k\varepsilon$.

Proof. We have

$$\begin{aligned}\|A_1\cdots A_k - B_1\cdots B_k\| &\le \sum_{i=1}^{k} \|A_1\cdots A_i B_{i+1}\cdots B_k - A_1\cdots A_{i-1}B_i\cdots B_k\| \\ &\le \sum_{i=1}^{k} \|A_1\cdots A_{i-1}\| \cdot \|A_i - B_i\| \cdot \|B_{i+1}\cdots B_k\| \\ &\le k\varepsilon\kappa^2.\end{aligned}$$

□

State Preparation We start from the task of preparing any given pure quantum state from the initial state $|0^S\rangle$. A considerable amount of study has been conducted trying to reduce the time complexity for state preparation, and these works usually requires huge amount of ancilla qubits (and therefore requires large space); see [ZLY22] for a exposure on this topic. Here we provide a simple space-efficient construction, relying on the result of [Ta-13] which uses the space-efficient Solovay-Kitaev Theorem [vMW12]:

Lemma 6.1.2. *Given $n = 2^S$, a unit vector $v \in \mathbb{C}^n$ and $\varepsilon > 0$, we can construct a unitary quantum circuit Q_v on S qubits with time $O(n \cdot \mathrm{polylog}(1/\varepsilon))$ and space $O(\log(n/\varepsilon))$ such that $\|Q_v|0^S\rangle - v\|_2 \le \varepsilon$.*

Proof. The circuit is a composition of $n-1$ two-level unitaries, that is, unitaries that operate on two dimensions of the computational basis). More specifically, starting from the initial state $|0^S\rangle$, in the i-th step the unitary $U_{a_i} = \begin{pmatrix} a_i & \sqrt{1-|a_i|^2} \\ \sqrt{1-|a_i|^2} & \overline{a_i} \end{pmatrix}$ is applied on the i-th and $(i+1)$-th dimension, where $a_i = v_i/\sqrt{|v_i|^2 + |v_{i+1}|^2 + \cdots + |v_n|^2}$, so that

the state after the i-th step is

$$(v_1, \ldots, v_i, \sqrt{|v_{i+1}|^2 + \cdots + |v_n|^2}, 0, \ldots, 0).$$

By [Ta-13], each two-level unitary can be implemented up to error ε/m with time $O(m \cdot \mathrm{polylog}(1/\varepsilon))$ and space $O(\log(m/\varepsilon))$. By Lemma 6.1.1 the total error is at most ε. □

We also note that the result in Lemma 6.1.2 was known in [MVBS05] with a slightly different proof, and while the space complexity is not originally stated there, it is implicit from their construction.

Hamiltonian simulation Given a Hamiltonian H, which is a Hermitian matrix that describes the evolution of the quantum system, the Hamiltonian simulation problem asks to simulate the evolution e^{iHt} for arbitrary $t > 0$. Ta-Shma showed in [Ta-13] how to perform Hamiltonian simulation with a space-efficient unitary quantum circuit, and it is in-place (without any ancillas). We restate the result for Hermitian contractions:

Theorem 6.1.3 ([Ta-13]). *Given a Hamiltonian $H \in \mathbb{C}^{n\times n}$ and $\varepsilon > 0$, we can construct a unitary quantum circuit U with time $\mathrm{poly}(n/\varepsilon)$ and space $O(\log(n/\varepsilon))$, such that $\|U - e^{iH}\| \leq \varepsilon$.*

For Hamiltonians with bounded spectral norms, the dependence on ε was improved in a series of works [BCC+14, BCC+15, BCK15]. Formally we have:

Theorem 6.1.4 ([BCK15]). *Given a Hamiltonian $H \in \mathbb{C}^{n\times n}$ with $\|H\| \leq \mathrm{poly}(n)$ and $\varepsilon > 0$, we can construct a unitary quantum circuit U with time $\mathrm{poly}(n, \log(1/\varepsilon))$ and space $O(\log(n/\varepsilon))$, such that $\|U - e^{iH}\| \leq \varepsilon$.*

Matrix-Vector Product and Matrix Inverse Using the Hamiltonian simulation algorithm and phase estimation (see e.g. [NC10, Section 5.2]), Ta-shma [Ta-13] gives an efficient way to compute $H^{-1}|u\rangle$ for any well-conditioned matrix H and unit vector $|u\rangle$, based on the framework of [HHL09]. Assuming that H is Hermitian and $\|H\| \geq 1$, we sketch how the algorithm works as follows.

- First apply the phase estimation over the unitary e^{iH} so that it maps $|u_\lambda\rangle$ to $|u_\lambda\rangle|\lambda\rangle$, where u_λ is an eigenvector of H with eigenvalue λ.
- For each eigenvector apply the unitary transformation

$$|\lambda\rangle \to \lambda^{-1}|0\rangle|\lambda\rangle + \sqrt{1-\lambda^{-2}}|1\rangle|\lambda\rangle$$

  according to the eigenvalue $\lambda \geq 1$.
- Uncompute the eigenvalues by reversing the phase estimation over e^{iH}.

When $\|H\| \geq \kappa^{-1}$ for some $\kappa > 1$, we can simply apply the algorithm on κH. Also when H is not Hermitian, the matrix $\begin{pmatrix} 0 & H \\ H^\dagger & 0 \end{pmatrix}$ does the trick. Finally, notice that by replacing λ^{-1} with λ while assuming $\|H\| \leq \kappa$, the algorithm also works on computing $H|u\rangle$.

Theorem 6.1.5. *Given a matrix $H \in \mathbb{C}^{n\times n}$ with $\kappa = \mathrm{poly}(n)$ such that $\|H\|, \|H^{-1}\| \leq \kappa$, and an error parameter $\varepsilon > 0$, we can construct unitary quantum circuits Q_H and Q'_H with time $\mathrm{poly}(n/\varepsilon)$ and space $O(\log(n/\varepsilon))$, such that*

$$\left\| (\mathbb{I}_n \otimes |0^\ell\rangle\langle 0^\ell|) Q_H (\mathbb{I}_n \otimes |0^\ell\rangle\langle 0^\ell|) - (\kappa^{-1}H) \otimes |0^\ell\rangle\langle 0^\ell| \right\| \leq \varepsilon,$$

$$\left\|(\mathbb{I}_n \otimes |0^\ell\rangle\langle 0^\ell|)Q'_H(\mathbb{I}_n \otimes |0^\ell\rangle\langle 0^\ell|) - (\kappa H^{-1}) \otimes |0^\ell\rangle\langle 0^\ell|\right\| \leq \varepsilon,$$

where $\ell = O(\log(1/\varepsilon))$ *is the number of ancilla.*

In other words, Q_H and Q'_H are efficiently constructed *block encodings* [CGJ19] of $\kappa^{-1}H$ and κH^{-1}.

Based on Theorem 6.1.5, Fefferman and Lin [FL18] showed that Poly-Conditioned MatrixInverse, the problem of approximating one entry of H^{-1} to constant error, is in fact a complete problem for $\mathsf{BQ_UL}$.

Matrix Powering When H is a contraction, that is when $\|H\| \leq \kappa = 1$, a directly corollary of Theorem 6.1.5 is that we can actually compute the powers of H, by simply repeatedly applying H. Each application produces $\mathrm{poly}(n/\varepsilon)$ dimensions of junks and requires 2^ℓ additional clean dimensions, and thus to compute H^T for $T = \mathrm{poly}(n)$ the overall space usage is still $O(\log(n/\varepsilon))$.

Notice that the channel powering problem is essentially powering the natural representation matrix of a channel, as we have

$$\mathrm{Tr}[|0^S\rangle\langle 0^S| \cdot \Phi^T(|0^S\rangle\langle 0^S|)] = \langle 0^{2S}|K(\Phi)^T|0^{2S}\rangle. \tag{6.1}$$

And since the natural representation $K(\Phi)$ being a contraction corresponds to the channel Φ being unital, taking $\varepsilon = O(1/T)$, by Lemma 6.1.1 we get a $\mathsf{BQ_UL}$ algorithm for a restricted version of ChannelPowering on unital channels.

Theorem 6.1.6. *UnitalChannelPowering* $\in \mathsf{BQ_UL}$.

When κ is larger, repeated applying $\kappa^{-1}H$ does not work because the error will also be multiplies by κ^T which is too large. Instead, observed by Fefferman and Remscrim [FR21] is that there is a reduction from matrix powering to matrix inverse that maintains the well-conditioned property: Let $Z \in \mathbb{C}^{n(T+1)\times n(T+1)}$ such that

$$Z = \begin{pmatrix} \mathbb{I}_n & -H & 0 & \cdots & 0 \\ 0 & \mathbb{I}_n & -H & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \mathbb{I}_n & -H \\ 0 & \cdots & \cdots & 0 & \mathbb{I}_n \end{pmatrix},$$

then

$$Z^{-1} = \begin{pmatrix} \mathbb{I}_n & H & H^2 & \cdots & H^T \\ 0 & \mathbb{I}_n & H & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & H^2 \\ \vdots & & \ddots & \mathbb{I}_n & H \\ 0 & \cdots & \cdots & 0 & \mathbb{I}_n \end{pmatrix}.$$

Moreover, $\|Z\| \leq 1 + \kappa$ and $\|Z^{-1}\| \leq 1 + T\kappa$. Use the fact that Poly-Conditioned MatrixInverse is in $\mathsf{BQ_UL}$ [FL18], this gives a $\mathsf{BQ_UL}$ algorithm for powering matrices with $\kappa = \mathrm{poly}(n)$.

Theorem 6.1.7 ([FR21]). *Given a matrix $H \in \mathbb{C}^{n\times n}$ with $\kappa = \mathrm{poly}(n)$ such that $\|H\| \leq \kappa$, a positive integer T, two unit vectors $v, w \in \mathbb{C}^n$ and an error parameter $\varepsilon > 0$, we can construct a unitary quantum circuit W with time $\mathrm{poly}(n/\varepsilon)$ and space $2^S = O(\log(n/\varepsilon))$*

such that

$$\left| |\langle 0^S | W | 0^S \rangle|^2 - |w^\dagger H^T v|^2 \right| \leq \varepsilon.$$

When $H = K(\Phi)$ for a quantum channel on S qubits, by Proposition 2.3.6 we have $\|H\| \leq 2^S = \sqrt{n}$, and thus by taking $\kappa = \sqrt{n}$ they conclude that ChannelPowering $\in$ $\mathsf{BQ_UL}$. Combined with Theorem 6.2, this implies that

Theorem 6.1.8 ([FR21]). $\mathsf{BQL} = \mathsf{BQ_UL}$.

6.2 Error Reduction in $\mathsf{BQ_UL}$

Just like in classical randomized computation, quantum algorithms in BQL can also be amplified to reduce the error rates. One example of such techniques is the Marriott-Watrous amplification [MW05].

Theorem 6.2.1 ([MW05]). *Given a unitary quantum circuit U with time $T = \mathrm{poly}(n)$ and space $S = \log n$, with a final measurement that outputs 0 with probability p, and an error parameter $\varepsilon > 0$, we can construct a circuit U' with time $\mathrm{poly}(n/\varepsilon)$ and space $O(\log(n/\varepsilon))$, with intermediate measurements, that outputs a value $\tilde{p}$ that $|\tilde{p} - p| \leq \varepsilon$ with probability $1 - 1/\mathrm{poly}(n/\varepsilon)$.*

The drawback of the Marriott-Watrous amplification is that it uses intermediate measurements. In [FKL+16], Fefferman et al. proposed a unitary amplification algorithm for decision problems, which also obtains the optimal exponential error reduction.

Theorem 6.2.2 ([FKL+16]). $\mathsf{BQ_UL} = \mathsf{Q_UL}(1 - 2^{-\mathrm{poly}(n)}, 2^{-\mathrm{poly}(n)})$, *which stands for unitary quantum logspace with exponentially small error.*

Corollary 6.2.3. $\mathsf{BQL} = \mathsf{Q_UL}(1 - 2^{-\mathrm{poly}(n)}, 2^{-\mathrm{poly}(n)})$.

Built on these previous works on error reduction, here we present a simulation of BQL with unitary quantum circuits in a strong sense, that is not limited to decision problems and does not require promises on the measurement probabilities. We start with the numerical form of ChannelPowering:

Lemma 6.2.4. *Given a matrix $H \in \mathbb{C}^{n\times n}$ with $\kappa = \mathrm{poly}(n)$ such that $\|H\| \leq \kappa$, a positive integer T, two unit vectors $v, w \in \mathbb{C}^n$ and an error parameter $\varepsilon > 0$, we can construct a unitary quantum circuit with time $\mathrm{poly}(n/\varepsilon)$ and space $O(\log(n/\varepsilon))$ such that with probability $1 - 1/\mathrm{poly}(n/\varepsilon)$, it outputs $|w^\dagger H^T v|^2$ with additive error ε.*

Proof. Let W be the circuit in Theorem 6.1.7, which outputs 0 with probability p such that

$$\left|p - |w^\dagger H^T v|^2\right| \leq \varepsilon/2,$$

and the problem here is to output the numerical value p. By Theorem 6.2.1, we can construct a quantum circuit W' with time $\mathrm{poly}(n/\varepsilon)$ and space $O(\log(n/\varepsilon))$ with intermediate measurements, which with probability $1 - \delta = 1 - 1/\mathrm{poly}(n/\varepsilon)$ outputs a value $\tilde{p}$ such that $|\tilde{p} - p| \leq \varepsilon/4$.

Since the resulting circuit W' is not unitary, we would like to use Corollary 6.2.3 to compute unitarily each bit in the output value $\tilde{p}$ of W' while reducing the error. Assume that every bit in $\tilde{p}$ is 0 with probability either in $[0, 1/3]$ or $[2/3, 1]$, then for $1 \leq i \leq \lceil \log(1/\varepsilon) \rceil + 2$, we let W_i be the unitary quantum circuit that computes the i-th bit of $\tilde{p}$ with exponentially small error. Ideally, the outputs of W_i combined together would ε-approximate $|w^\dagger H^T v|^2$.

However, the value $\tilde{p}$ outputted by the Marriott-Watrous amplification might be different in each W_i, so the final approximation assembled can be totally wrong (for instance, when $p = 0.5$, the outputs $\tilde{p} = \overline{0.1000\ldots}$ and $\tilde{p} = \overline{0.0111\ldots}$ might be assembled to $\overline{0.1111\ldots}$). Moreover, the error reduction in [FKL+16] may have unpredictable results, as the promises on the distributions of the bits in $\tilde{p}$ are not guaranteed (again when $p = 0.5$, the most significant bit of $\tilde{p}$ is equally distributed on 0 and 1).

Fortunately, we can solves both problems by computing from the most significant bit to the least significant bit. We maintain a value $q \in [0,1]$ which is initialized to 0. For each $i = 1$ to $\lceil \log(1/\varepsilon) \rceil + 2$ do the following: Run the modified circuit W_i which outputs the i-th bit of $(\tilde{p} - q)$ instead of $\tilde{p}$. To deal with case when $\tilde{p} - q$ is outside of $[0, 2^{-i+1})$, if $\tilde{p} - q < 0$ it outputs 0, and if $\tilde{p} - q \geq 2^{-i+1}$ it outputs 1. Let the output bit be b_i and update q to $q + b_i \cdot 2^{-i}$.

We claim that with probability $1 - \mathrm{poly}(n/\varepsilon)$, $|q - p| \leq \varepsilon/2$. First notice that, if every bit in $\tilde{p}$ is 0 with probability in $[0, 2\delta] \cup [1 - 2\delta, 1]$, then the error reduction will work as intended, while with probability $1 - O(\delta \log(1/\varepsilon)) = 1 - 1/\mathrm{poly}(n/\varepsilon)$ the value $\tilde{p}$ is the same in each circuit W_i, so that q is also the same as $\tilde{p}$.

Now let i be the first index such that the i-th bit of $\tilde{p}$ is 0 with probability in $[2\delta, 1 - 2\delta]$. As the Marriott-Watrous amplification outputs incorrectly with probability at most δ, it means that there are two valid outputs $\tilde{p}_1$ and $\tilde{p}_2$, both are $\varepsilon/4$-close to p, and they coincide in the first $i - 1$ bits but differs at the i-th bit. Let q_i be the value of q at that step, which consists of the first $i-1$ bits of $\tilde{p}_1$ and $\tilde{p}_2$, then $|q_i + 2^{-i} - p| \leq \varepsilon/4$. Therefore the remaining bits of q could only be $\overline{011\ldots11}$, $\overline{100\ldots00}$ or $\overline{100\ldots01}$, which means $|q_i + 2^{-i} - q| \leq \varepsilon/4$ and thus $|q - p| \leq \varepsilon/2$. Notice that on the i-th (and the last bit when $b_i = 1$) the error

reduction may fail and arbitrarily output 0 or 1, but it does not matter as both 0 and 1 are viable in these cases.

As a conclusion, the value q is an ε-approximation of $|w^\dagger H^T v|^2$ with probability $1 - 1/\mathrm{poly}(n/\varepsilon)$. The circuit that outputs q is clearly constructible with time $\mathrm{poly}(n/\varepsilon)$ and space $O(\log(n/\varepsilon))$, and the circuit is unitary since the $O(\log(1/\varepsilon))$ measurements that output b_i can be deferred directly, while each W_i can be uncomputed by implementing the circuit in reverse. □

Corollary 6.2.5. *Given a matrix $H \in \mathbb{C}^{n\times n}$ with $\kappa = \mathrm{poly}(n)$ such that $\|H\| \leq \kappa$, a positive integer T, two unit vectors $v, w \in \mathbb{C}^n$ and an error parameter $\varepsilon > 0$, we can construct a unitary quantum circuit with time $\mathrm{poly}(n/\varepsilon)$ and space $O(\log(n/\varepsilon))$ such that with probability $1 - 1/\mathrm{poly}(n/\varepsilon)$, it outputs $w^\dagger H^T v$ with additive error ε.*

Proof. Let $H_1 = \begin{pmatrix} H & \\ & 1 \end{pmatrix}$, $v_1 = \begin{pmatrix} v/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$, $v_1' = \begin{pmatrix} v/\sqrt{2} \\ \mathrm{i}/\sqrt{2} \end{pmatrix}$ and $w_1 = \begin{pmatrix} w/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$. Notice that $\|H_1\| \leq \max(\kappa, 1)$. Since we have

$$w^\dagger H^T v = \frac{1}{2}\left(4|w_1^\dagger H_1^T v_1|^2 - |w^\dagger H^T v|^2 - 1\right) + \frac{\mathrm{i}}{2}\left(4|w_1^\dagger H_1^T v_1'|^2 - |w^\dagger H^T v|^2 - 1\right),$$

computing $|w^\dagger H^T v|^2$ up to error $\varepsilon/2$, and computing $|w_1^\dagger H_1^T v_1|^2$ and $|w_1^\dagger H_1^T v_1'|^2$ each up to error $\varepsilon/8$ gives $w^\dagger A^T v$ with error ε. □

Notice that one can instead achieve $1/\mathrm{poly}(n/\varepsilon)$ error probability without using the exponential error reduction in [FKL+16], by simply repeating the decision circuit in $\mathsf{BQ_UL}$ for $O(\log(n/\varepsilon))$ rounds. Nevertheless, it is enough for proving the following theorem,

which states that unitary quantum circuits can simulate any quantum algorithm in logspace by computing its output distribution with polynomially small error.

Theorem 6.2.6. *Given a quantum algorithm with time $T = \mathrm{poly}(n)$ and space $S = \log n$ specified by the natural representations $K(\Phi_1), \ldots, K(\Phi_T) \in \mathbb{C}^{n^2 \times n^2}$, where*

$$\rho_T = \Phi_T \circ \Phi_{T-1} \circ \cdots \circ \Phi_1(|0^S\rangle\langle 0^S|)$$

is its final state, a multi-outcome measurement $\{M_1, \ldots, M_r\}$ with $r = \mathrm{poly}(n)$, and an error parameter $\varepsilon > 0$, we can construct a unitary quantum circuit U with time $\mathrm{poly}(n/\varepsilon)$ and space $S' = O(\log(n/\varepsilon))$ such that for every $j \in [r]$ it holds that

$$\left| |\langle j| W |0^{S'}\rangle|^2 - \mathrm{Tr}[\rho_T M_j] \right| \leq \varepsilon.$$

Proof. As shown at the start of this chapter, we can define a unified channel Φ such that

$$\Phi^T(\rho_0 \otimes |0\rangle\langle 0|) = \rho_T \otimes |T\rangle\langle T|,$$

and $K(\Phi)$ is simply a block matrix consists of $K(\Phi_1), \ldots, K(\Phi_T)$. Note that $\|K(\Phi)\| \leq \kappa = \mathrm{poly}(n)$ by Proposition 2.3.6. Now we have

$$\mathrm{Tr}[\rho_T M_j] = \mathrm{vec}(M_j)^\dagger \,\mathrm{vec}(\rho_T) = \left(\mathrm{vec}(M_j)^\dagger \otimes \langle T^2|\right) K(\Phi)^T \left(\mathrm{vec}(\rho_0) \otimes |0\rangle\right).$$

For every $j \in [r]$, let $m_j = \|\mathrm{vec}(M_j)\|_2 \leq \sqrt{n}$ by Proposition 2.3.7. Therefore by letting

$$w = \mathrm{vec}(M_j) \otimes |T^2\rangle / m_j \quad \text{and} \quad v = \mathrm{vec}(\rho_0) \otimes |0\rangle$$

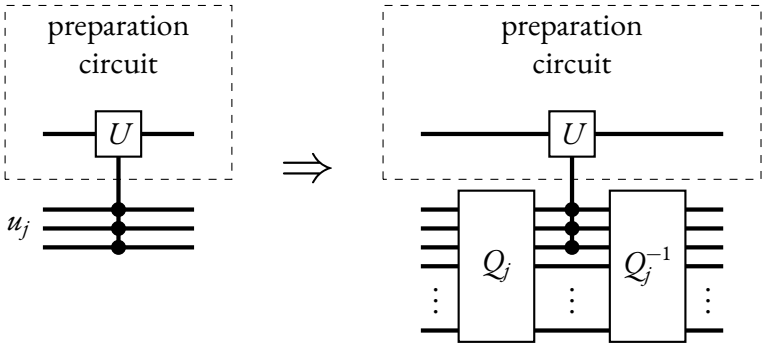

Figure 6.1: The quantum operator U in the preparation circuit controlled by an entry u_j of u, in binary representation with classical bits. We replace the classical control by first implementing the circuit Q_j, applying the controlled-U operator, and implementing Q_j in reverse.

in Corollary 6.2.5, we get a unitary quantum circuit Q_j with time $\text{poly}(n/\varepsilon)$ and space $O(\log(n/\varepsilon))$ such that with probability $1 - 1/\text{poly}(n/\varepsilon)$, it outputs a value $\varepsilon/(2\sqrt{n}r)$-close to $\text{Tr}[\rho_T M_j]/m_j$, which implies an $\varepsilon/(2r)$-approximation of $\text{Tr}[\rho_T M_j]$.

Consider the preparation circuit constructed in Lemma 6.1.2 which prepares the unit vector

$$u = \left(\sqrt{\text{Tr}[\rho_T M_0]}, \sqrt{\text{Tr}[\rho_T M_1]}, \ldots, \sqrt{\text{Tr}[\rho_T M_{r-1}]}\right).$$

with error $\varepsilon/3$. By construction, the preparation circuit can be viewed as a composition of $r - 1$ unitary operators, each controlled by a different entry in u. Since u is not explicitly given, we instead control these unitary operators with the output qubits of Q_j, but without measurements. Each circuit Q_j is applied in reverse after the control, so that the space can be reused. See Figure 6.1 for an illustration.

It is clear that the entire circuit is with time $\text{poly}(n/\varepsilon)$ and space $O(\log(n/\varepsilon))$. The error introduced by replacing each of th $r - 1$ unitary operators with circuits Q_j and Q_j^{-1} is at

most $\varepsilon/(2r) + 1/\mathrm{poly}(n/\varepsilon)$, therefore by Lemma 6.1.1 the total error is at most

$$\varepsilon/3 + (r-1)(\varepsilon/(2r) + 1/\mathrm{poly}(n/\varepsilon)) < \varepsilon. \qquad \square$$

6.3 Equivalence of Learning and Deciding

In this section we show that, the existence of efficient classical simulation of quantum learning is equivalent to that on decision problems, which by our beliefs are highly unlikely. This is formally stated as the following theorem.

Theorem 6.3.1. *Every quantum learning algorithm with time T and space S can be simulated classically with time* $\mathrm{poly}(2^S T)$ *and space* $O(S + \log T)$*, if and only if* $\mathsf{BQL} = \mathsf{BPL}$.

Note that here we state the theorem with respect to uniform learning algorithms. If we want to extent the result to all non-uniform algorithms, the corresponding decision classes also need to be changed to non-uniform ones, i.e. $\mathsf{BQL/poly}$ and $\mathsf{BPL/poly}$. Below we prove both directions of this equivalence.

Lemma 6.3.2. *If there are functions $t(\cdot,\cdot)$ and $s(\cdot,\cdot)$, such that every unitary quantum learning algorithm with time T and space S can be simulated classically with time $t(T,S)$ and space $s(T,S)$, then*

$$\mathsf{BQ_UL} \subseteq \mathsf{BPTISP}(t(\mathrm{poly}(n), O(\log n)), s(\mathrm{poly}(n), O(\log n))).$$

Specifically, if every unitary quantum learning algorithm in time T and space S can be simulated classically with time $\mathrm{poly}(2^S T)$ *and space* $O(S + \log T)$*, then* $\mathsf{BQ_UL} = \mathsf{BPL}$.

Proof. Suppose that we have a unitary quantum circuit with time $T(n) = \text{poly}(n)$ and space $S(n) = O(\log n)$ that decides a partial function $f : X \to \{0,1\}$, where $X \subseteq \{0,1\}^n$. Let $U(x,i)$ be the unitary gate at the i-th step of the decision algorithm with input x, which can be constructed in time $\text{poly}(n)$ and space $O(\log n)$.

We can convert the quantum circuit to a learning algorithm as follows. Use X directly as the sample space, while the samples are always constant x for some fixed $x \in X$. The learning task is to distinguish between $x \in f^{-1}(0)$ or $x \in f^{-1}(1)$. Upon receiving the sample x, the learning algorithm simply applies the following unitary operator on $\mathbb{C}^{2^{S(n)}} \otimes \mathbb{C}^{T(n)}$:

$$|\psi\rangle|i\rangle \to \big(U(x,i)|\psi\rangle\big)|(i+1) \bmod T(n)\rangle$$

so that after $T(n)$ steps it computes in the first register the same state as in the quantum circuit. Therefore it computes $f(x)$ and distinguishes between the two cases. Using the premises, we have a classical learning algorithm with time $t(\text{poly}(n), O(\log n))$ and space $s(\text{poly}(n), O(\log n))$ that accomplishes the same task. The classical learning algorithm can be viewed as a randomized decision algorithm that computes $f(x)$ by self-constructing the stochastic matrices in the same time and space.

Alternatively, in the learning task we can have a potentially much smaller sample space $\Gamma = \{0,1\} \times [n]$, by viewing the learning problem as computing f in the random-query model (see Chapter 8). For each $x \in X$, let distribution D_x be the one that uniformly draws $i \in [n]$ and outputs (x_i, i). Let $\mathcal{X} = \{D_x \mid x \in f^{-1}(0)\}$ and $\mathcal{Y} = \{D_x \mid x \in f^{-1}(1)\}$. Each x_i can be retrieved within $O(n \log n)$ samples with high probability, therefore the quantum learning algorithm can compute each $U(x,i)$ with high probability in time $\text{poly}(n)$ and space $O(\log n)$, and the rest of the proof is the same as above. □

Lemma 6.3.3. *If* ChannelPowering $\in \mathsf{BPTISP}(t(n), s(n))$*, where* $t(n) \geq \Omega(n)$ *and* $s(n) \geq \Omega(\log n)$*, then every quantum learning algorithm with time* T *and space* S *can be simulated classically with time* $t(\mathrm{poly}(2^S T))$ *and space* $s(\mathrm{poly}(2^S T))$*.*

Proof. Suppose that we have a quantum learning algorithm with time T and space $S = \log n$ that distinguishes between two distribution families $\mathcal{X}$ and $\mathcal{Y}$. Let $\Phi(z)$ be the channel applied when receiving the sample z, and let M be the final measurement. With the sample distribution D, let

$$A = \mathop{\mathbb{E}}_{z\sim D}[K(\Phi(z))].$$

Note that A is also a natural representation (specifically, for the channel $\mathbb{E}_{z\sim D}[\Phi(z)]$). Similar to proof of Theorem 6.2.6, the probability of the learning algorithm outputting 0 is

$$\mathop{\mathbb{E}}_{z\sim D^T}\left[\mathrm{vec}(M_0)^\dagger K(\Phi_T)\cdots K(\Phi_1)\,\mathrm{vec}(\rho_0)\right] = \mathrm{vec}(M)^\dagger A^T \mathrm{vec}(\rho_0).$$

What's different from Theorem 6.2.6 is that here A is not explicitly given. Instead, by Chernoff bound each time an entry of A is requested, it takes $\mathrm{poly}(nT)$ samples z to approximate the entry to at most $1/\mathrm{poly}(nT)$ error, so that the approximated matrix $\widetilde{A}$ differs from the actual matrix A by at most $\|\widetilde{A} - A\| \leq O((n^2T)^{-1})$. By Lemma 6.1.1 and Proposition 2.3.6, it means that $\|\widetilde{A}^T - A^T\| \leq O(n^{-1})$. Therefore applying the channel powering algorithm on $\widetilde{A}$ gives a classical learning algorithm that distinguishes $\mathcal{X}$ and $\mathcal{Y}$ in time $t(\mathrm{poly}(nT))$ and space $s(\mathrm{poly}(nT))$.

The above scheme has two problems. First, a fixed matrix $\widetilde{A}$ cannot be directly stored, and if every time the same entry is requested, the entry is approximated as the average of a different batch of samples, it may result in different requested values for the same en-

try (even though the difference is small with high probability), similar to the problem in Lemma 6.2.4. However, unlike the case in Lemma 6.2.4, here the classical powering algorithm is not explicitly given, and may not be robust against changing inputs.

The solution to this problem is the shift and truncate method by Saks and Zhou [SZ99], which has found numerous applications in space-bounded algorithms [Ta-13] and derandomization [CCvM06, GL19]. Concretely, let $P = t(\mathrm{poly}(nT))$ be the largest number of possible requests to entries of A in the powering algorithm, and take a uniform random number $\zeta \in [8P]$. For simplicity let $L = 12n^2T$ and $N = 24n^4T$. When the entry A_{jk} is requested, the algorithm takes $t(\mathrm{poly}(nT))$ samples z_i and calculate the average value a of the (j,k)-entries of $K(\Phi(z_i))$, so that $|a - A_{jk}| < \frac{1}{8NP}$ with probability at least $1 - 2^{-P}$. The value fed back for the request is

$$\widetilde{A}_{jk} = \frac{1}{N}\left\lfloor N \cdot \mathrm{Re}(a) + \frac{\zeta}{8P} \right\rfloor + \frac{\mathrm{i}}{N}\left\lfloor N \cdot \mathrm{Im}(a) + \frac{\zeta}{8P} \right\rfloor .$$

We claim that with high probability, this value coincides with the fixed value

$$\frac{1}{N}\left\lfloor N \cdot \mathrm{Re}(A_{jk}) + \frac{\zeta}{8P} \right\rfloor + \frac{\mathrm{i}}{N}\left\lfloor N \cdot \mathrm{Im}(A_{jk}) + \frac{\zeta}{8P} \right\rfloor .$$

For the real part, as $|N \cdot \mathrm{Re}(a) - N \cdot \mathrm{Re}(A_{jk})| < \frac{1}{8P}$, there is at most one possibility for ζ such that $\left\lfloor N \cdot \mathrm{Re}(a) + \frac{\zeta}{8P} \right\rfloor \neq \left\lfloor N \cdot \mathrm{Re}(A_{jk}) + \frac{\zeta}{8P} \right\rfloor$, which is of probability $\frac{1}{8P}$, and the same holds for the imaginary part. By the union bound on the bad events during all P requests, with probability

$$1 - \left(2^{-P} + \frac{1}{8P} + \frac{1}{8P}\right) P > \frac{2}{3}$$

for every (j,k) the value $\widetilde{A}_{jk}$ are always the same, and $|\widetilde{A}_{jk} - A_{jk}| \leq \frac{\sqrt{2}}{N} \leq (n^2L)^{-1}$, so $\|\widetilde{A} - A\| \leq L^{-1}$.

Now by Proposition 2.3.6 we have $\|A\| \leq \sqrt{n}$, and thus $\|\widetilde{A}\| \leq \|A\| + \|\widetilde{A} - A\| \leq \sqrt{n} + L^{-1}$. Since $\|\operatorname{vec}(M)\|_2 \leq \sqrt{n}$ by Proposition 2.3.7 and $\|\operatorname{vec}(\rho_0)\|_2 = 1$, by Lemma 6.1.1 we have

$$\left|\operatorname{vec}(M)^\dagger (\widetilde{A}'^T - A^T) \operatorname{vec}(\rho_0)\right| \leq \sqrt{n} \cdot \frac{T}{L} \cdot (\sqrt{n} + L^{-1})^2 < \frac{1}{12}.$$

Since the error of the original quantum learning algorithm can be amplified to $1/4$ so that $\operatorname{vec}(M)^\dagger A^T \operatorname{vec}(\rho_0)$ is in $[0, 1/4]$ or $[3/4, 1]$, we conclude that with probability $2/3$,

$$\operatorname{vec}(M)^\dagger \widetilde{A}^T \operatorname{vec}(\rho_0) \in [0, 1/3] \text{ or } [2/3, 1].$$

and the two cases can be distinguished by the classical ChannelPowering algorithms on $\widetilde{A}$. □

Corollary 6.3.4. *If* ChannelPowering $\in$ BPL, *then every quantum learning algorithm with time T and space S can be simulated classically with time* $\operatorname{poly}(2^S T)$ *and space* $O(S + \log T)$.

Theorem 6.3.1 follows as we already know ChannelPowering $\in \mathsf{BQ_UL}$. In addition, as we also know that $\mathsf{BQ_UL} \subseteq \mathsf{uniformNC}^2 \subseteq \mathsf{TISP}(n^{O(1)}, \log^2 n)$, we have the following unconditional result:

Corollary 6.3.5. *Every quantum learning algorithm with time T and space S can be simulated classically with time* $\operatorname{poly}(2^S T)$ *and space* $O(S^2 + \log^2 T)$.

Finally, we show that there are indeed cases when the quantum learning algorithm can be

simulated classically with efficiency. Specifically, when the learning task is to distinguish an arbitrary family of distributions with a single distribution, we have the following result.

Theorem 6.3.6. *If $\mathcal{Y} = \{Y\}$, then any quantum learning algorithm that distinguishes between $\mathcal{X}$ and $\mathcal{Y}$ within time T and space S can be simulated classically in time* $\mathrm{poly}(2^S T)$ *and space* $O(S + \log T)$.

Proof. Suppose that we have a quantum learning algorithm with time T and space $S = \log m$ that distinguishes between $\mathcal{X}$ and $\{Y\}$. Let $\Phi(z)$ be the unital channel applied when receiving the sample z. We already know from Lemma 6.3.3 that with the sample distribution D, the output probability is $\mathrm{vec}(M)^\dagger A^T \mathrm{vec}(\rho_0)$ where $A = \mathbb{E}_{z\sim D}[K(\Phi(z))] \in \mathbb{C}^{n^2\times n^2}$. Since the distribution Y is fixed, the corresponding matrix $B = \mathbb{E}_{z\sim Y}[K(\Phi(z))]$ is also fixed. Now for any $D \in \mathcal{X}$, we know that

$$\left|\mathrm{vec}(M)^\dagger (A^T - B^T)\,\mathrm{vec}(\rho_0)\right| \geq 1/3.$$

By Proposition 2.3.6, we have $\|A^i\|, \|B^i\| \leq \sqrt{n}$ for every $i \leq T$. Thus by Lemma 6.1.1,

$$\|A - B\|_{\mathrm{F}} \geq \|A - B\| \geq \frac{1}{\sqrt{n}T}\|A^T - B^T\| \geq \frac{1}{3nT}.$$

which means there must exist $i, j \in [n^2]$ such that $|A_{i,j} - B_{i,j}| \geq \frac{1}{9}n^{-6}T^{-2}$.

The classical simulation algorithm iterates over all $i, j \in [n^2]$. For each choice of i, j, the algorithm approximates $\mathbb{E}_z[K(\Phi(z))_{i,j}]$, compares it to $B_{i,j}$, and claims the samples are drawn from a distribution in $\mathcal{X}$ if there exists i, j such that

$$\left|\mathbb{E}_z[K(\Phi(z))_{i,j}] - B_{i,j}\right| \geq \frac{1}{10b^6T^2}.$$

Since each entry of $K(\Phi(z))$ can be computed in time $O(T)$ and space $O(S)$ and has magnitude at most 1, the Chernoff bound asserts that $\mathrm{poly}(nT)$ samples are enough for accuracy $(100n^6T^2)^{-1}$ with probability $2/3$, in which case the algorithm correctly distinguish $\mathcal{X}$ and $\{Y\}$. □

6.4 Bonus: Streaming Proof for BQL

Now that we have the simulation results of BQL, we note that the proof in Section 4.2.2 can be easily modified to work with BQL as well. The definition of a streaming proof is the same as in Definition 4.2.1, and the only difference here is that we allow the prover to have quantum computation power (but still in logspace and outputs a classical proof).

Theorem 6.4.1. *A language is in* BQL *if and only if it has a streaming proof between a quantum logspace prover and a classical logspace verifier where the verifier uses $O(\log n)$ random bits.*

The rest of this section is to prove Theorem 6.4.1. First, it is clear that any streaming proof between a quantum logspace prover and a classical logspace verifier can be implemented by a BQL algorithm, which simulates the honest prover and the verifier, with success probability at least $(3/4)^2 > 1/2$ which can be amplified. It suffices to argue that Unitary Matrix Powering Theorem 3.2, the logspace-complete problem for BQL, can be solved by a streaming proof between a quantum logspace prover and a classical logspace verifier, where the verifier uses $O(\log n)$ random bits. We recall the problem definition here.

Definition 6.4.2 (Unitary Matrix Powering)**.** *The input include a unitary matrix $M \in \mathbb{C}^{n\times n}$, a parameter $T \leq \mathrm{poly}(n)$ and a projective measurement $\Pi \in \mathbb{C}^{n\times n}$. The promise*

on the input is that $\|\Pi M^T(e_1)\|_2^2 \geq 4/5$ *or* $\|\Pi M^T(e_1)\|_2^2 \leq 1/5$*, and the output is* 1 *in the former case and* 0 *in the latter.*

Towards this, we change the notion of δ-good sequence of vectors for a matrix M to be defined with ℓ_2-norm.

Definition 6.4.3. *Let M be any $n \times n$ matrix and $T \leq \mathrm{poly}(n)$ be a natural number. Let $v_i = M^i(e_1)$ for all $i \leq T$. Let $\delta \in [0,1]$. A sequence of vectors $v'_0, v'_1, \ldots, v'_T \in \mathbb{R}^n$ is said to be δ-good for M if for all $i \in [T]$, we have $\|v'_i - v_i\|_2 \leq \delta$ and $v_0 = e_1$.*

We make use of the following claims.

Claim 6.4.4. *There is a quantum logspace prover which given an $n \times n$ unitary matrix M and parameters $T \leq \mathrm{poly}(n)$, $\delta \geq 1/\mathrm{poly}(n)$ as input, outputs a δ-good sequence of vectors for M with probability at least* $3/4$*.*

Claim 6.4.5. *Let $1/\mathrm{poly}(n) < \delta \leq (10^4 T^2)^{-1}$. There is a randomized logspace verifier which given any $n \times n$ unitary matrix M and parameters $T \leq \mathrm{poly}(n)$ and δ as input and read-once access to a stream of vectors $v'_0, \ldots, v'_T \in \mathbb{R}^n$ (where each vector is specified up to $\Theta(\log(n))$ bits of precision), does the following.*

- *If the sequence is δ-good for M, then the verifier aborts with probability at most* $1/4$*.*
- *If $\|v'_T - v_T\|_2 \geq 1/5$, then the verifier aborts with probability at least* $3/4$*.*

Furthermore, this algorithm only uses $O(\log(n))$ bits of randomness.

Let us see how to complete the proof using Claim 6.4.4 and Claim 6.4.5. Given an $n \times n$ unitary matrix M as input and a parameter $T \leq \mathrm{poly}(n)$, set $\delta = \min\{(10^4 T^2)^{-1}, 1/10\}$.

Run the prover's algorithm from Claim 6.4.4 using this value of δ to produce a stream $v'_0, \ldots, v'_T$. Run the verifier's algorithm from Claim 4.2.6 on this stream to verify. If the verifier does not abort, we have it return 1 if $\|\Pi v'_T\|_2^2 \geq 0.6$, return 0 if $\|\Pi v'_T\|_2^2 \leq 0.4$ and return $\perp$ otherwise. With the access to read Π from the input, this computation can be easily done in classical logspace when v'_T is given as a stream.

Completeness: Claim 6.4.4 implies that an honest prover outputs a δ-good sequence with probability at least $3/4$. Claim 6.4.5 implies that an honest proof is aborted with probability at most $1/4$. Since $\|v'_T - v_T\|_2 \leq \delta \leq 1/10$ by assumption and Π is a projection, $\|\Pi v'_T - \Pi v_T\|_2 \leq 1/10$. Hence, if $\|\Pi v_T\|_2^2 \geq 4/5$, then $\|\Pi v'_T\|_2^2 \geq (\sqrt{4/5} - 0.1)^2 \geq 0.6$ and if $\|\Pi v_T\|_2^2 \leq 1/5$ then $\|\Pi v'_T\|_2^2 \leq (\sqrt{1/5} + 0.1)^2 \leq 0.4$. Thus, the verifier will return the correct answer whenever the subroutine does not abort.

Soundness: Consider the behavior of this verifier on an arbitrary proof. If the verifier makes a mistake and returns the incorrect answer, it must be the case that either $\|\Pi v_T\|_2^2 \geq 4/5$ and $\|\Pi v'_T\|_2^2 \leq 0.4$, or $\|\Pi v_T\|_2^2 \leq 1/5$ and $\|\Pi v'_T\|_2^2 \geq 0.6$. In either case, we must have $\|v'_T - v_T\|_2 \geq \min\left\{\sqrt{4/5} - \sqrt{0.4}, \sqrt{0.6} - \sqrt{1/5}\right\} \geq 1/5$. Claim 6.4.5 implies that such a proof is aborted with probability at least $3/4$.

This completes the proof of Theorem 6.4.1. We now proceed to prove Claim 6.4.4 and Claim 6.4.5.

Proof of Claim 6.4.4. The prover starts by outputting $v_0 = e_1$. To output the intermediate v_i, note that $v_i(j) = e_j^\dagger \mathcal{M}^i e_1$, so the prover can use Corollary 6.2.5 to estimate $v_i(j)$ up to error ε in $\mathrm{poly}(n/\varepsilon)$ time and $O(\log(n/\varepsilon))$ space with success probability $1 - p$, where $p = 1/\mathrm{poly}(n/\varepsilon)$. Taking the average over $O(p^{-1} \log n)$ trials gives an estimate of $v_i(j)$ up

to error $\varepsilon + p$ with probability $1 - (4nT)^{-1}$, and thus by union bound, we get a quantum logspace prover which with probability at least $3/4$, estimate each $v_i(j)$ up to $\varepsilon + p$ additive accuracy for all $i \in [T]$ and $j \in [n]$.

Take $\varepsilon = 1/\mathrm{poly}(n)$ so that $\varepsilon + p \leq \delta/n$, and we have $\|v'_i - v_i\|_2 \leq \|v'_i - v_i\|_\infty \cdot n \leq \delta$. This completes the proof of Claim 6.4.4. □

Proof of Claim 6.4.5. The verifier's algorithm is exactly the same as Algorithm 4.2, and the proofs for completeness and soundness are almost identical.

Completeness of the Algorithm: Suppose $v'_0, \ldots, v'_T$ is a δ-good sequence, then $\|v'_i - v_i\|_2 \leq \delta$ for all $i \in [T]$ and $v'_0 = e_1$. Since M is unitary, for all $i \in [T]$

$$\begin{aligned}\left\|M \cdot v'_{i-1} - v'_i\right\|_2 &\leq \left\|M \cdot v'_{i-1} - M \cdot v_{i-1}\right\|_2 + \left\|M \cdot v_{i-1} - v_i\right\|_2 + \|v_i - v'_i\|_2 \\ &\leq \|v_{i-1} - v'_{i-1}\|_2 + \|v_i - v'_i\|_2 \leq 2\delta.\end{aligned}$$

Thus, $\|w\|_2 \leq 2T\delta$. Consider the quantity $\Delta = \langle \alpha, w \rangle$ that the algorithm estimates. Similarly by Chebyshev's Inequality, with probability at least 0.99, we have $|\langle \alpha, w \rangle| \leq 20T\delta$. This implies that with probability at least $(0.99)^{11} \geq 0.8$, every iteration in Algorithm 4.2 does not abort.

Soundness of the Algorithm: Suppose a dishonest prover produces a stream of vectors $v'_0, \ldots, v'_T$ such that $\|v'_T - v_T\|_2 \geq 1/5$. Assume that $v'_0 = e_1$. Let $\varepsilon = 1/(10T)$. We argue that for some $i \in [T]$, we must have $\|w_i\|_2 > \varepsilon$. Assume by contradiction that

$\left\|M \cdot v'_{i-1} - v'_i\right\|_2 \leq \varepsilon$ for all $i \in [T]$. Then by triangle inequality we have

$$\begin{aligned}\|v_T - v'_T\|_2 = \left\|M^T(v'_0) - v'_T\right\|_2 &\leq \sum_{i=1}^{T} \left\|M^{T-(i-1)} \cdot v'_{i-1} - M^{T-i} \cdot v'_i\right\|_2 \\ &\leq \sum_{i=1}^{T} \left\|M^{T-i}\right\|_2 \cdot \left\|M \cdot v'_{i-1} - v'_i\right\|_2 \\ &\leq \sum_{i=1}^{T} \varepsilon = T\varepsilon = 1/10.\end{aligned}$$

which contradicts the assumption that $\|v'_T - v_T\|_2 \geq 1/5$. Thus, we must have $\|w\|_2 \geq \varepsilon$. Similarly by 4-wise independence and the Paley-Zygmund Inequality [PZ32] we conclude that $\Pr\left[|\langle \alpha, w\rangle| \geq \varepsilon/10\right] \geq \frac{1}{8}$. By repeating this experiment 11 times, we can ensure that with probability at least $1 - (1 - 1/8)^{11} \geq 3/4$, we find at least one instance so that

$$|\langle \alpha, w\rangle| \geq \varepsilon/10 > 20T\delta,$$

as $\delta \leq (10^4 T^2)^{-1}$. This implies that the algorithm aborts with probability at least $3/4$. □

Part II

Lower Bound Results

7

Overview of Part II

In Part II, we present our results that proves lower bounds for space-bounded computation. These results are roughly in two regimes: Polynomial lower bounds for traditional computational problems with some fixed input, and exponential lower bounds for learning problems.

Time-Space Tradeoffs for Decision Problems

Proving strong lower bounds for explicit decision problems is always hard. The current best lower bound on circuit size barely passes $3n$ after decades of research [Blu84, LY22], and the current best formula size lower bound is still under $\Omega(n^3)$ [Tal17, Bog18].

To get stronger unconditional lower bounds, we need to add other restrictions, and the space constraint usually is a natural assumption. However, by Barrington's theorem [Bar89], even for width-5 branching programs, which corresponds to an unreasonably small space less than 3 bits, the best lower bound for time could not pass the lower bound formula size which is less than cubic. Actually, even this close-to-cubic barrier has not been

obtained when we allow mildly larger space. Under uniform models, Williams [Wil08] proved that SAT cannot be solved in $O(n^c)$ time and $O(n^\varepsilon)$ space with $c = 2\cos(\pi/7) \approx 1.8$ and some $\varepsilon > 0$, which is the best lower bound up to date. On the other hand, for non-uniform models is it still open even to obtain something polynomially better than linear:

Open Problem 7.1. *Find an explicit family of decision problems $F : \{0,1\}^n \to \{0,1\}$, such that any branching program with space $S \leq \text{polylog}(n)$ that computes F requires time $T = n^{1+\Omega(1)}$.*

In fact, Open Problem 7.1 is so notoriously hard that in the past decades, remarkable effort were put into works just to prove $T = \Omega(n \cdot \text{polylog}(n))$ with better poly-logarithmic factors. Even for the most restrictive model of deterministic oblivious branching programs, the best lower bound we have is $T = \Omega(n \log^2 n)$ for $S = O(n^{1-\varepsilon})$ by Babai, Nisan and Szegedy [BNS92], and for the general randomized, non-oblivious branching programs, we have the sophisticated lower bound $T = \Omega\left(n\sqrt{\frac{\log n}{\log\log n}}\right)$ for $S = O(n^{1-\varepsilon})$ by Beame, Saks, Sun and Vee [BSSV03].

Our results in Chapter 8 represents an attempt to improve these lower bounds and answer Open Problem 7.1. We propose a new computation model called the coupon-collector model, in reminiscence of the famous coupon-collector problem. In this model, the input $x \in \{0,1\}^n$ is not given by querying specific coordinates, but is given as samples (i, x_i) for uniformly random i. Clearly any algorithm in the standard model can be converted to one in the coupon-collector model with an $O(n)$ overhead in time by waiting for the desired coordinate, but for certain problems like Gap-Hamming we can do much better. As a result, it is completely non-trivial to prove superlinear lower bounds in this model.

In fact, we showed that if we allow equality dependencies among the input samples, any

time-space lower bound in the coupon-collector model can be directly translated to the same lower bound on deterministic oblivious branching programs, up to logarithmic factors. Therefore, proving strong lower bounds in the our model is potentially a way to the resolution of Open Problem 7.1. As a first step, we proved such bounds for a restricted class of branching programs, along with a quadratic lower bound for general branching programs with zero error when the samples are independent. The next reasonable step is to improve the latter bound to the bounded-error case, which was recently proved in [Din23].

Time-Space Tradeoffs for Multi-Output Functions

Beside decision problems, we also consider the computation of multi-output functions, which maps $\{0,1\}^n$ to $\{0,1\}^m$ for some $m = \text{poly}(n)$. Borodin and Cook [BC82] gave a powerful method to prove polynomial time-space lower bounds for these problems, and in particular showed that Sorting requires a tradeoff of $TS \geq \Omega(n^2)$. Actually all previous classical time-space lower bounds for multi-output functions, whether their authors were aware or not, are applications of this method by Borodin and Cook.

The Borodin-Cook method has a crucial drawback that it is a counting method over some fixed distribution, and there for by Yao's Minimax Principle it provides the same lower bound for deterministic and randomized computation. On the other hand, if we ignore the Borodin-Cook method, we essentially only have lower bounds for decision problems and thus are subject to Open Problem 7.1, which means that we do not have any randomized vs. deterministic separation better than $O(\log^2 n)$.

In Chapter 9 we resolve this discrepancy and prove a polynomial randomized vs. deterministic separation. We present an explicit multi-output function on $[n]^n$, that can be

computed by a randomized oblivious branching program with linear time and logarithmic space, but requires $\widetilde{\Omega}(n^{1.25})$ time for deterministic oblivious logspace. The function is a total function and thus prevents the trivial separation with sublinear randomized algorithm [Mon10], and our lower bound proof is a combination of adversarial method and the Borodin-Cook method.

Our results also suggest that the reason for this separation not being known is that the candidate problem requires careful design. We show that for many natural candidates for which we know better randomized algorithms, proving a separation could lead to the surprising resolution of Open Problem 7.1. One such example is the SetIntersection problem (given two sets A and B, output elements in $A \cap B$), whose optimal randomized algorithm uses random [BCM13] (or pseudo-random [CJWW22, LZ23]) hash functions. A time and space-efficient reduction from SetIntersection to a decision problem shows that, if we prove fully derandomization of these algorithms must introduce polynomial overhead, then Open Problem 7.1 is solved.

Finally, we would like to mention that a quantum vs. randomized separation for multi-output functions has been known for long, due to the quantum algorithm for Sorting by Klauck [Kla03] with $T^2S = \widetilde{O}(n^3)$. The separation is extra strong in the sense that the quantum algorithm uses only polylog(n) quantum memory for any S. On the other hand, lower bounds for quantum time-space tradeoffs are more scarce and the proofs are more ad-hoc. Currently, only two proof methods are known for quantum lower bounds: via direct-product theorems [KŠdW07] and via the recording query technique [HM21].

Time-Space Lower Bounds for Learning

In a sharp contrast to traditional computational problems where it is hard even to prove lower bounds with large polynomials, for learning problems we do have lower bounds exponential in the size of samples. One reason for that is because learning problems naturally has an exponentially large space of inputs (albeit heavy with redundancy), and the streaming nature of the problems also provides huge advantage for proving lower bounds.

Our result in Chapter 10 is a quantum extension of the result of [GRT18], which is also where our techniques originates from. Therefore, it helps to give a quick recap of how the classical lower bound in [GRT18] works. We using parity learning [Raz18] as an example, so $M(a,x)$ means the inner product of a and x in $\mathbb{F}_2$.

Consider a classical branching program that tries to learn an unknown and uniformly random $x \in \{0,1\}^n$ from samples (a,b), where $a \in \{0,1\}^n$ is uniformly random and $b = M(a,x)$. We associate every state v with a distribution $P_{X|v}$ over $\{0,1\}^n$, indicating the distribution of x conditioned on reaching that state, and examine the evolution of the inner product

$$\langle P_{X|v}, P\rangle = \sum_{x\in\{0,1\}^n} P_{X|v}(x)\cdot P(x)$$

with some target distribution P. Receiving a sample (a,b) implies that $M(a,x) = b$, hence only the part of $P_{X|v}$ supported on such x proceeds. If this part is close to $\frac{1}{2}$ probability, we say that a divides $P_{X|v}$ evenly. Denoting the new distribution as $P^{(a,b)}_{X|v}$, after proper normal-

ization the new inner product is

$$\langle P_{X|v}^{(a,b)}, P\rangle = \sum_{\substack{x\in\{0,1\}^n \\ M(a,x)=b}} P_{X|v}(x)\cdot P(x) \Big/ \sum_{\substack{x\in\{0,1\}^n \\ M(a,x)=b}} P_{X|v}(x). \tag{7.1}$$

Ideally, both $P_{X|v}$ and the point-wise product vector $P_{X|v}\cdot P$ should have reasonably small ℓ_2 norms. Due to the extractor property of M, most of $a \in \{0,1\}^n$ should divide both vectors evenly, and thus the denominator is close to $\frac{1}{2}$ while the enumerator is close to $\frac{1}{2}\langle P_{X|v}, P\rangle$. That means, given a uniformly random a, we get limited progress on the inner product. On the other hand, from $\langle U, P\rangle = 2^{-n}$ with uniform distribution U to $\langle P, P\rangle = 2^{2\varepsilon n}\cdot 2^{-n}$, the branching program needs to make multiple steps of progression. Therefore it happens with an extremely small probability.

To ensure that the above statement goes smoothly, we require the following properties for every state v in the branching program:

- The ℓ_2 norm $\left\|P_{X|v}\right\|_2$ is small.
- The ℓ_2 norm $\left\|P_{X|v}\cdot P\right\|_2$ is small, which is implied if the ℓ_∞-norm $\left\|P_{X|v}\right\|_\infty$ is small.
- The denominator in (7.1) is bounded away from 0 for every sample (a, b).

These properties do not hold by themselves. Instead, we execute a *truncation* procedure to the branching program stop whenever one of the properties fails. The proof then boils down to proving a $2^{-\Omega(n^2)}$ bound on the probability of reaching a state with large $\left\|P_{X|v}\right\|_2$, from which by a standard union bound, we can prove that either $2^{\Omega(n)}$ samples or $\Omega(n^2)$ bits of memory are necessary.

To bound the probability of reaching a state with a large ℓ_2-norm, the basic idea is to fix its distribution as the target distribution P, and bound the increment of the inner product $\langle P_{X|v}, P\rangle$. Define a *bad event* to be a pair (v, a) of the state v and the upcoming part of the sample a, such that $\langle P_{X|v}, P\rangle \geq 2^{-n}$, and for one of the two possible outcomes b,

$$\sum_{\substack{x\in\{0,1\}^n \\ M(a,x)=b}} P_{X|v}(x)\cdot P(x) \geq \left(\frac{1}{2}+2^{-\delta n}\right)\cdot \langle P_{X|v}, P\rangle \tag{7.2}$$

with some small constant δ. In other words, the inner product $\langle P_{X|v}, P\rangle$ is large enough, while not being divided evenly by a. From (7.1) we know that the inner product gets at most roughly doubled through a bad event. In contrast, in the good case, the inner product either gets a mere $(1+2^{-\delta n})$ multiplicative factor or is already smaller than the baseline 2^{-n}. Also, the extractor property of M ensures that for every state v, over uniformly random a, the bad event happens with at most $2^{-\Omega(n)}$ probability.

We then define the badness level $\beta(v)$ of a state v keeps track of how many times the computational path went through bad events before reaching v. The above observations on the bad events imply that (omitting the smaller factors):

- For every state v, $\langle P_{X|v}, P\rangle$ is bounded by $2^{\beta(v)}\cdot 2^{-n}$;
- Heading to the next stage, $\beta(v)$ increases by 1 with probability $2^{-\Omega(n)}$.

Therefore at each stage, the total weight of states with badness level β is at most $2^{-\Omega(\beta n)}$. Thus any state with $\langle P_{X|v}, P\rangle \geq 2^{2\varepsilon n}\cdot 2^{-n}$ must have $2^{-\Omega(n^2)}$ probability, which concludes the proof for the classical lower bound.

Now, to prove our quantum lower bound, some of the notions above could be easily

transferred to the scenario with quantum memory:

- The state v is a quantum state in the complex linear space of quantum memory;
- The distribution $P_{X|v}$ is still well-defined: It is the distribution of x when the quantum memory is measured to v (see (10.1));
- We are still able to implement ℓ_2 truncation: If $P_{X|v}$ has large ℓ_2-norm, project the entire system to the orthogonal subspace $v^\perp$ of v and repeat, until there is no such state v.
- We are also able to implement sample truncation, in a similar manner to ℓ_2 truncation. As the criteria here depends on a, we separately create a copy of the current system for each a, truncate the states v using projection when $P_{X|v}$ is not evenly divided by a in each copy, and then merge them back together. We prove that the error introduced by this truncation is small.

However, their are two major problems that prevent us from copying the proof verbatim into the quantum case. The first problem is the ℓ_∞ truncation. When we try to emulate the classical implementation of ℓ_∞ truncation with quantum truncation, that is, to only project to $v^\perp$ the system conditioned on the specific x where $P_{X|v}(x)$ is large, instead of for every x, it may lead to huge changes to the distributions $P_{X|u}$ on states u non-orthogonal to v, even eliminating the entire system.

And the second problem is the definition of badness levels. If we define the badness level $\beta(v)$ for each state v individually by examining the bad events over the historical states, then it is not clear how to measure the total weight of a badness level β. Or we could have a more operational definition of badness levels, but since the bad event in (7.2) is not linear in v,

such an definition, which is a linear operator, inevitably introduces error that escalates fast with the number of stages.

These problems occur when we are shooting for the quadratic quantum memory lower bound. It turns out that if we limit the quantum memory to a small linear number, both problems could be solved with ease.

ℓ_∞ Truncation. When there is only small quantum memory and no classical memory, the treatment for ℓ_∞ truncation is straightforward. We remove all quantum states v with distributions of large ℓ_∞ norm, by projecting the system to the orthogonal subspace $v^\perp$, just like the process of ℓ_2 truncation. As the overall distribution on x is uniform, any state v with $\|P_{X|v}\|_\infty \geq 2^{\delta n} \cdot 2^{-n}$ must have weight at most $2^{-\delta n}$. Therefore, as long as the dimension of the Hilbert space is much smaller than δn, the error introduced in this truncation is small.

With classical memory in presence, the actual ℓ_∞ truncation step is a bit more complicated. We first apply the original classical ℓ_∞ truncation on the classical memory W. Now that $\|P_{X|w}\|_\infty$ is bounded for each classical memory state w, we can remove the quantum states v with large $\|P_{X|v,w}\|_\infty$ by projection as stated above.

Badness Levels. We are able to avoid the problems of defining the badness level on quantum memory altogether, by keeping it a property on the classical memory only. To do so we need to alter the definition of a bad event: it is now a pair (w, a) of classical memory state w and sample a, such that there exists some quantum memory state v with $P_{X|v,w}$ satisfying (7.2).

For each fixed classical memory state w, we still need to ensure that bad events happen

with a small probability. We prove it by showing that, if there are many different samples a, each associated with some quantum state v_a satisfying (7.2), then there is some quantum state v that simultaneously satisfies (7.2) with most of such a (which is impossible because of the extractor property). This is ultimately due to the continuous nature of (7.2): Under some proper congruent transformation, (7.2) becomes a simple threshold inequality on quadratic forms over v. Now if it is satisfied by some v_a, it is going to be satisfied by most v for a much smaller threshold parameter δ, and hence the existence of a simultaneously satisfying v. In this argument, we use Lemma 2.3.10 and crucially relies on the fact that the dimension is at most $2^{\varepsilon n}$ for some small ε.

8

Decision Problems: The Coupon-Collector Model

The goal of this chapter is to introduce the coupon collector model on decision problems, prove a relatively simple lower bound in this model, and relate it to the time-space lower bounds for standard oblivious branching programs.

We first recall the classical coupon-collector problem, which asks how large T should be, so that a uniformly random T-tuple in $[n]^T$ contains every element of $[n]$ with high probability. Generalizing the goal to a subset $A \subseteq [n]$, we have the following answer:

Proposition 8.1. *Given any subset $A \subseteq [n]$, for a uniformly random $i \in [n]^T$, the probability that $A \not\subseteq \{i_1, \ldots, i_T\}$ is at most $n(1 + \log |A|)T^{-1}$.*

The proof follows directly from the fact that the expected waiting time for every element in A to appear is $n \sum_{j=1}^{|A|} j^{-1} \geq n(1 + \log |A|)$, and Markov's inequality.

In the coupon-collector model of computation, at each step $t \in \mathbb{N}_+$ a uniformly random index $i_t \in [n]$ (corresponding to a coupon) is provided. When the problem specifies an

input $x \in \{0,1\}^n$, at each step t the value of the bit $x_{i_t} \in \{0,1\}$ is also given along with the random index i_t. In this paper, we consider two cases for the joint distribution of the indices:

Independent The indices $i_1, i_2, \ldots$ are mutually independent.

Recurring The only dependencies allowed among $i_1, i_2, \ldots$ are equalities. More formally, there is a partition $p : \mathbb{N}_+ \to \mathbb{N}_+$, such that $i_t = i'_{p(t)}$ for every $t \in \mathbb{N}_+$, where $i'_1, i'_2, \ldots$ are mutually independent and uniformly random over $[n]$.

For the rest of the paper, we refer to the two cases as *independent distribution* and *recurring distributions*. Notice that the independent distribution is a special case of the recurring ones.

8.1 Zero-Error Coupon Collector

Consider the task of computing a function $f : \{0,1\}^n \to \{0,1\}$ with zero-error in the coupon-collector model, where the algorithm is allowed to output $0, 1$ or a special symbol $\perp$. For every $x \in \{0,1\}^n$, we require that the probability of outputting $f(x)$ is at least $1/2$, and the probability of outputting $1 - f(x)$ is zero, where the probability is over the randomness used by the algorithm and the randomness from the coupon-collector model itself.

At the end of this section, we will prove the following theorem:

Theorem 8.1.1. *Let $f : \{0,1\}^n \to \{0,1\}$ be a boolean function with sensitivity $s(f)$. Under the coupon-collector model with the independent distribution, every branching program of length T and width $2^S \geq n$ which computes f with zero-error must satisfy $TS \geq \frac{1}{8} n \cdot s(f)$ for sufficiently large n.*

Recall the definition of sensitivity: Let $s(f, x)$ the number of coordinates $i \in [n]$ such that $f(x \oplus e_i) \neq f(x)$, and the the sensitivity of f is $s(f) = \max_x s(f, x)$. Applying Theorem 8.1.1 to functions with large sensitivities yields the quadratic time-space lower bound:

Corollary 8.1.2. *Let f be a boolean function on n-bits with sensitivity $\Omega(n)$ (For instance, AND, XOR, Majority, s-t connectivity, etc.). Under the coupon-collector model with the independent distribution, every branching program of length T and width $2^S \geq n$ which computes f with zero-error must have satisfy $TS \geq \Omega(n^2)$.*

Remark. *Theorem 8.1.1 is tight up to logarithmic factors, in the sense that for every $m \leq n$, the function $x_1 \oplus \cdots \oplus x_m$ with sensitivity m can be computed with zero-error within S space and $O(nmS^{-1} \log n)$ time. We briefly sketch the algorithm here: Equally partition $[m]$ into $O(mS^{-1})$ parts, each of size $O(S)$. For each part P, use $O(n \log n)$ steps to record the values x_i for all indices $i \in P$. If any $i \in P$ does not appear within these $O(n \log n)$ steps, output $\perp$. Otherwise compute the partial parity $\bigoplus_{i \in P} x_i$, and accumulate the partial parities.*

In order to prove Theorem 8.1.1, we first look at the task of solving the original coupon-collector problem with zero-error, that is, the collector can choose to claim success or failure when it stops. Whenever it claims success it must be the case that all coupons in a target set $A \subseteq [n]$ have been collected, and this happens with probability at least $1/2$. Note that in Proposition 8.1, if the collector blindly claims success at $O(n \log n)$ time, there is still probability that the claim is false.

8.1.1 Set-Labeled Branching Program

To show a lower bound for the zero-error coupon-collector problem, we first restrict the computation model to a certain type of branching programs called *set-labeled* branching

programs.

Definition 8.1.3. *In a set-labeled branching program, every non-leaf vertex has n outgoing edges, labeled with each element in $[n]$ exactly once which corresponds to the index of the coupon it receives. Furthermore, every vertex v is labeled with a set $H(v) \subseteq [n]$, satisfying the following soundness condition: if an edge from vertex u to vertex v is labeled with $i \in [n]$, it must hold that $H(v) \subseteq H(u) \cup \{i\}$. The start vertex must be labeled with $\varnothing$. For a target set $A \subseteq [n]$, a leaf v claims success if $A \subseteq H(v)$.*

Theorem 8.1.4. *Under any recurring distribution and for any set $A \subseteq [n]$, every set-labeled branching program of length n and width $2^S \geq |A|$ that solves the zero-error coupon-collector problem for the target set A must satisfy $TS \geq \frac{1}{8}n|A|$ for sufficiently large n.*

Fix such a set-labeled branching program. We first prove an upper bound on the probability of the computation path reaching two given vertices:

Lemma 8.1.5. *For any two vertices u, v in a set-labeled branching program, where $u \in V_i$, $v \in V_j$ and $i < j$. Under the coupon-collector model with any recurring distribution,*

$$\Pr[\textit{reaching } u \wedge \textit{reaching } v] \leq \left(\frac{j-i}{n}\right)^{|H(v)\setminus H(u)|}.$$

Proof. Let $p : \mathbb{N}_+ \to \mathbb{N}_+$ be the partition in the recurring distribution. Let G be the random variable that represents the set of indices received between layer V_i and layer V_j, and let $\ell = |\{p(k) \mid i < k \leq j\}|$. Note that G is uniformly distributed over $[n]^\ell$. By the soundness requirement of set-labeled branching programs, if the computation path reaches u and then v, the set G corresponding to this path must satisfy $H(v) \subseteq H(u) \cup G$.

Therefore,

$$\Pr[\text{reaching } u \wedge \text{reaching } v] \leq \Pr[H(v) \subseteq H(u) \cup G] = \Pr[H(v) \setminus H(u) \subseteq G].$$

If $\ell < |H(v) \setminus H(u)|$ then the above probability is zero. Otherwise by (over)counting the positions where the elements of $|H(v) \setminus H(u)|$ appear and the union bound we have

$$\begin{aligned}\Pr[H(v) \setminus H(u) \subseteq G] &\leq \frac{\ell!}{(\ell - |H(v) \setminus H(u)|)!} \cdot n^{-|H(v)\setminus H(u)|} \\ &\leq \left(\frac{\ell}{n}\right)^{|H(v)\setminus H(u)|} \\ &\leq \left(\frac{j-i}{n}\right)^{|H(v)\setminus H(u)|}.\end{aligned}$$

□

Remark. *For the independent distribution, the above argument yield:*

$$\Pr[\textit{reaching } v \mid \textit{reaching } u] \leq \left(\frac{j-i}{n}\right)^{|H(v)\setminus H(u)|}.$$

The weaker result in Lemma 8.1.5, however, holds more generally for any recurring distribution. It is also strong enough for proving Theorem 8.1.4.

Now we are ready to prove Theorem 8.1.4.

Proof of Theorem 8.1.4. Suppose the length of the set-labeled branching program is $\mathcal{B}$. Define the weight of a vertex v as $W(v) = \mathbb{E}[\mathcal{B}_{\to v}] = \Pr[\text{reaching } v]$. For a set of vertices $\mathcal{A}$, let $W(\mathcal{A}) = \sum_{v\in\mathcal{A}} W(v)$. Since the leaves are all in V_T, for every $0 \leq i \leq T$ we have $W(V_i) = 1$. The fact that the branching program succeeds on the target set $A \subseteq [n]$

translates to:

$$\sum_{\substack{v\in\mathcal{L}_T\\ A\subseteq H(v)}} W(v) \geq 1/2. \tag{8.1}$$

We divide the branching program into $\frac{|A|}{2S}$ stages, each consists of a consecutive part of the layers. For every $0 \leq k \leq \frac{|A|}{2S}$, let $i(k)$ be the smallest index i of a layer V_i such that

$$\sum_{\substack{v\in V_i\\ |H(v)|\geq 2kS}} W(v) \geq \frac{kS}{|A|}.$$

By (8.1) we know such a layer must exist. Now the k-th stage consists of the layers from $V_{i(k)}$ to $V_{i(k+1)-1}$. Let

$$A_k = \{u \in V_{i(k)} \mid |H(u)| \geq 2kS\}, \qquad B_k = \{u \in V_{i(k-1)} \mid |H(u)| < 2kS\}.$$

By the definitions of $i(k)$, we know that $W(A_k) \geq kS/|A|$, $W(B_k) > 1 - kS/|A|$.

Now we show that every stage contains at least $(n/3 - 1)$ layers. Suppose for contradiction that for some k, it holds that $i(k+1) - i(k) < n/3 - 1$. For any two vertices $u \in B_k$ and $v \in A_{k+1}$, by Lemma 8.1.5 we have

$$\Pr[\text{reaching } u \wedge \text{reaching } v] \leq \left(\frac{i_{k+1} - i_k + 1}{n}\right)^{|H(v)\setminus H(u)|} < 3^{-2S}.$$

Since each layer consists of 2^S vertices, we have

$$\Pr[\text{reaching } B_k \wedge \text{reaching } A_{k+1}] \leq \sum_{\substack{u\in B_k\\ v\in A_{k+1}}} \Pr[\text{reaching } u \wedge \text{reaching } v] \leq 2^S \cdot 2^S \cdot 3^{-2S}.$$

Therefore, applying the union bound gives:

$$\begin{aligned}
&\Pr[\text{reaching } V_{i(k)-1} \wedge \text{reaching } V_{i(k+1)}] \\
\leq\ &\Pr[\text{reaching } V_{i(k)-1} \setminus B_k] + \Pr[\text{reaching } V_{i(k+1)} \setminus A_{k+1}] \\
&+ \Pr[\text{reaching } B_k \wedge \text{reaching } A_{k+1}] \\
\leq\ &1 - W(B_k) + 1 - W(A_{k+1}) + 2^S \cdot 2^S \cdot 3^{-2S} \\
<\ &\frac{kS}{|A|} + 1 - \frac{(k+1)S}{|A|} + (2/3)^{2S} = 1 - \frac{S}{|A|} + (2/3)^{2S} < 1.
\end{aligned}$$

The last step is because $1/|A| \leq 2^{-S}$. However, since the computation path must pass through both $V_{i(k)-1}$ and $V_{i(k+1)}$, the probability above must be 1, which is a contradiction.

Thus we conclude that, for n large enough, $i_{k+1} - i_k \geq n/3 - 1 \geq n/4$. Therefore,

$$T \geq \sum_{0 \leq k < |A|/2S} (i_{k+1} - i_k) \geq \frac{n|A|}{8S}. \qquad \square$$

8.1.2 Reduction to Set-Labeled Branching Programs

We will prove Theorem 8.1.1 by reducing every decision problem to a set-labeled branching program solving the coupon-collector problem with zero-error. The first step is to show that in general, any such branching program is set-labeled.

Lemma 8.1.6. *Under the independent distribution, for any set $A \subseteq [n]$, every branching program of length n and width $2^S \geq |A|$ that solves the zero-error coupon-collector problem for the target set A can be assigned on each vertex a label $H(v) \subseteq [n]$ so that the branching program is set-labeled.*

Proof. Let $P(v)$ be the collection of directed paths from the starting vertex to v. For every directed path p let $h(p)$ be the collection of indices labeled on the edges of p. Then we define $H(v) = \cap_{p \in P(v)} h(p)$.

The starting vertex is clearly labeled with the empty set. To check the soundness, consider an edge e from vertex u to vertex v labeled with i. For every path $p \in P(u)$, the concatenation pe is a path in $P(v)$, and $h(pe) = h(p) \cup \{i\}$. Therefore,

$$H(v) \subseteq \bigcap_{p \in P(u)} h(pe) = H(u) \cup \{i\}.$$

Notice that every path from the starting vertex to a leaf corresponds to a collection of indices $i_1, \ldots, i_T$, that are given with probability $n^{-T} > 0$ under the independent distribution. Since the branching program collects A with zero-error, for every path to a leaf that claim success, it must hold that $A \subseteq \{i_1, \ldots, i_T\}$. Thus every successful leaf v is now labeled with $H(v) \supseteq A$. □

Now we are able to prove Theorem 8.1.1.

Proof of Theorem 8.1.1. Suppose there is a branching program $\mathcal{P}$ of length T and width 2^S that computes f with zero-error. Let $x \in \{0,1\}^n$ be an input such that $s(f) = s(f,x)$, and let $A = \{i \in [n] \mid f(x) \neq f(x^{(i)})\}$. We show below that from $\mathcal{P}$, one can extract a simple branching program $\mathcal{P}'$ for the coupon-collector problem of width at most 2^S and length T, which solves the zero-error coupon-collector problem for the target set A. Since $|A| = s(f)$, by Lemma 8.1.6 we know that $TS \geq = \frac{1}{8} n \cdot s(f)$ for sufficiently large n.

We construct $\mathcal{P}'$ inductively to simulate $\mathcal{P}$ on input x. For vertex v in $\mathcal{P}$ we use v' to denote its corresponding vertex in $\mathcal{P}'$. The start vertex v'_0 in $\mathcal{P}'$ corresponds to the start

vertex v_0 in $\mathcal{P}$. If in $\mathcal{P}$ there exists an edge from u to v labeled with (i, x_i), and u' is in $\mathcal{P}'$, then add v' to $\mathcal{P}'$ (if v' is not already there), and add an edge from u' to v' labeled with i. Finally, every leaf v' in $\mathcal{P}'$ claims success if the output on v is $f(x)$, and otherwise claims failure.

First notice that under the independent distribution, the probability of reaching a vertex v' in $\mathcal{P}'$ is exactly the same as the probability of reaching v in $\mathcal{P}$ with the input x. Since the probability that $\mathcal{P}$ outputs $f(x)$ on input x is at least $1/2$, the probability that $\mathcal{P}'$ claims success is also at least $1/2$.

We now show that conditioned on reaching a successful leaf v' in $\mathcal{P}'$, it must hold that $A \subseteq \{i_1, \ldots, i_T\}$. Suppose not, then for some index $i \in A$ there is a path p' from the start vertex to v' where no edge is labeled with i. Consider the corresponding path p in $\mathcal{P}$. On input $x^{(i)}$, the computation follows the path p with non-zero probability and outputs $f(x) \neq f(x \oplus e_i)$, which contradicts the zero-error property of $\mathcal{P}$. That concludes the proof that $\mathcal{P}'$ solves the zero-error coupon-collector problem for the target set A. □

8.2 Relation with Oblivious Branching Programs

Note that the zero-error guarantee is crucial to Theorem 8.1.1, since for instance, the n-bit AND function can be computed with constant error by a branching program of length $O(n)$ and width $O(1)$. However, when specified to the parity function, the best trade-off seems to be still quadratic even in the bounded-error setting. We propose the following conjecture, which is still open at the time of writing this thesis:

Conjecture 8.2.1. *Under the coupon-collector model with the independent distribution, any branching program of length T and width 2^S which computes $x_1 \oplus \cdots \oplus x_n$ with error* $1/3$

must satisfy $TS = \widetilde{\Omega}(n^2)$.

Interestingly, there is another algorithm for computing parity (which actually computes the Hamming weight) with bounded error, which is essentially different from the algorithm mentioned in the previous remark: Equally partition $[n]$ into $O(S/\log n)$ parts. For each part P, record the number of steps t when a pair (i, x_i) such that $i \in P$ and $x_i = 1$ is received, and finally approximate the partial sum $\sum_{i \in P} x_i$ with the integer closest to tn/T. By Chernoff bound, $T = O(n^2 S^{-1} \log^2 n)$ is enough so that the approximation of each part is wrong with probability $O(n^{-1})$.

Notice that this algorithm does not work in the zero-error setting. While the previous algorithm corresponds directly to a set-labeled branching program, it is not clear whether this approximation algorithm is related to set-labeled branching programs or not.

We note that by the time this dissertation is finished, Dinur [Din23] proved our Conjecture 8.2.1 affirmatively. In fact, for any total boolean function an lower bound holds with respect to its total influence:

Theorem 8.2.2 ([Din23]). *Let* $f : \{0,1\}^n \to \{0,1\}$ *be a boolean function with total influence* $I(f)$. *Under the coupon-collector model with the independent distribution, every branching program of length* T *and width* $2^S \geq n$ *which computes* f *with at most* $1/3$ *error must satisfy* $TS \geq \widetilde{\Omega}(n) \cdot I(f)$.

The conjecture (now a theorem) is also interesting as it could be seen as the first step to prove strong lower bounds for oblivious branching programs. In this section, we present two potential directions, both via proving lower bounds in the coupon-collector model. Let $\mathrm{SURJ}_{n,m} : [n]^m \to \{0,1\}$ be the surjectivity function: $\mathrm{SURJ}_{n,m}(i) = 1$ if and only $\{i_1, \ldots, i_m\} = [n]$.

Theorem 8.2.3. *For any $m \geq 2n(\log n + 1)$, any deterministic oblivious branching program computing $\mathrm{SURJ}_{n,m}$ is also a branching program for zero-error coupon-collector problem with the target set $[n]$ under some recurring distribution.*

Proof. Suppose at level $t-1$ the oblivious branching program reads $i_{p(t)}$, for some function $p : \mathbb{Z}_+ \to [m]$. Use p as the partition in the recurring distribution, then the computation of the branching program for the coupon-collector problem is exactly the same as in the oblivious branching program with a uniformly random input $i \in [n]^m$. Proposition 8.1 shows that the probability of $\mathrm{SURJ}_{n,m}(i) = 1$ is at least $1/2$. As the deterministic oblivious branching program always outputs correctly, as a branching program for the coupon-collector problem it succeeds with zero-error. □

For any boolean function $f : \{0,1\}^n \to \{0,1\}$ and $m \geq n$, let $f^* : [n]^m \times \{0,1\}^m \to \{0,1\}$ be a partial function defined as follows: $f^*(i,y)$ is well-defined for $i \in [n]^m$ and $y \in \{0,1\}^m$, if and only if $\mathrm{SURJ}_{n,m}(i) = 1$, and whenever $i_j = i_k$ it must hold $y_j = y_k$. When $f^*(i,y)$ is well-defined, the value of $f^*(i,y)$ is $f(y_{j_1}, \ldots, y_{j_n})$, where for every $\tau \in [n]$, j_τ is some $j \in [m]$ such that $i_j = \tau$.

Theorem 8.2.4. *Given any function $f : \{0,1\}^n \to \{0,1\}$, and for any $m \geq 3n(\log n + 1)$, if there is a deterministic oblivious branching program computing f^* of length T and width 2^S (on the inputs where f^* is well-defined), then there is a branching program of the same length and width, that computes f with error $1/3$ in the coupon-collector model under some recurring distribution.*

Proof. Add dummy levels to the oblivious branching program to double the length, such that if originally at level t the branching program reads either i_j or y_j, now it reads i_j at level

$2t$ and y_j at level $2t+1$. The oblivious branching program now can be regarded as the one of length T and width 2^S that at each level t reads a pair $(i_{p(t)}, y_{p(t)})$, for some function $p : \mathbb{N}_+ \to [m]$.

Use p as the partition in the recurring distribution. For any fixed $x \in \{0,1\}^n$, the computation in the coupon-collector model on input x is exactly the same as in the oblivious branching program with a uniformly random $i \in [n]^m$, and input $y \in \{0,1\}^m$ defined as $y_j = x_{i_j}$. For such i and y, $f^*(i,y)$ is well-defined if and only if $\mathrm{SURJ}_{n,m}(i) = 1$, and Proposition 8.1 indicates the probability that $f^*(i,y)$ is well-defined is at least $2/3$. Since whenever $f^*(i,y)$ is well-defined, the deterministic oblivious branching program correctly outputs $f(y_{j_1}, \ldots, y_{j_n}) = f(x)$, as a branching program under the coupon-collector model it computes f with error $1/3$. □

As a corollary, if in the coupon-collector model we were able to prove a time-space lower bound that holds under any recurring distribution, either for the coupon-collector problem with zero-error, or for any bounded-error computation, we would immediately have the same lower bound (up to logarithmic factors) on deterministic oblivious branching programs, and thus solves Open Problem 7.1.

9

Multi-Output Functions: A Polynomial Separation

In this chapter we study the time-space tradeoffs for multi-output functions. Previously all lower bounds on such tradeoffs were proved by the Borodin-Cook method [BC82], which we will revise in Section 9.1. However, as the Borodin-Cook method inherently give the same lower bound for deterministic and randomized computation, the following question remains unanswered for a long time:

> *Is there a polynomial separation between randomized and deterministic branching programs for time-space tradeoffs of multi-output functions?*

Here we answer this question in the affirmative for oblivious branching programs, where the queries made in the branching programs are independent of the input. In particular, we design a total function (n, p)-Non-Occurring Elements (Definition 9.2.1) on $[n]^n$ which outputs a subset of $[n]$, such that:

- There exists a randomized oblivious algorithm with space $O(\log n)$, time $O(n \log n)$

and one-way access to randomness, that computes the function with probability $1 - O(1/n)$;

- Any deterministic oblivious branching program with space S and time T that computes the function must satisfy $T^2 S \geq \Omega(n^{2.5}/\log n)$.

This will be proved in Section 9.2. The proof is is not technically hard, and the difficulty in proving such a separation actually lies mostly in finding a proper total function where the adversarial method works. We demonstrate this difficulty by showing that, for several natural candidate problems whose best known deterministic algorithms are polynomially worse than randomized algorithms, proving a polynomial separation will lead to the resolution of Open Problem 7.1.

9.1 The Borodin-Cook Method

In this section we revise the Borodin-Cook method which was used in all previous works for proving time-space lower bounds of multi-output functions. The original method by Borodin and Cook [BC82], used on the sorting problem, is quite complicated and not modular enough to be applied to other problem. It was simplified by Beame [Bea91], and from then on used to provide lower bounds for a variety of multi-output problems, including algebraic problems like matrix multiplication and inversion [Abr91], frequency moments over sliding windows [BCM13], and more recently, the memory game [CC17], printing and counting SAT assignments [MW18] and multiple collision finding [Din20], just to name a few. For a formal but restrictive description of the method and more applications, see [Sav98, Chapter 10].

Here we give a brief framework of the proof method:

1. Fix a distribution $\mathcal{D}$ over the inputs (often uniform), and find a suitable number $a(S)$ such that given $a(S)$ bits in the input, only S bits of output are revealed on average.

2. Prove that, given any decision tree of depth $a(S)$ and $c \cdot S$ bits of output assigned to each path in the tree, these outputs are correct with probability $2^{-\Omega(S)}$ under $\mathcal{D}$, for some large $c > 1$.

3. Now split the branching program into stages of length $a(S)$, and by a union bound over the 2^S starting nodes of each stage, the above argument shows that most inputs under $\mathcal{D}$ cannot generate $c \cdot S$ bits output within a stage. This implies a lower bound of the form $ST/a(S) \geq \Omega(n)$.

Step 2 is the technical part of the proof, which usually involves certain counting arguments as the distribution $\mathcal{D}$ is often a uniform distribution on some support. To give a concrete taste of the method, we prove a lower bound for the following problem.

Definition 9.1.1. *In the 2-StepPointerChasing (2-PC for short) problem, the input is a function $f : [n] \to [n]$, and the output consists of $(x, f(f(x)))$ for all $x \in [n]$.*

Proposition 9.1.2. *Any randomized oblivious branching program with space S and time T that computes 2-PC must satisfy $T^2 S \geq \widetilde{\Omega}(n^3)$.*

Proof. Consider a deterministic oblivious branching program $\mathcal{A}$ computing 2-PC. Divide the branching program into ℓ stages where each stage consists of $a = \frac{1}{6}\sqrt{nS}$ consecutive layers, for $\ell = T/a$.

Take any such stage k, and assume that the oblivious queries within the stage are on $f(x_1), \ldots, f(x_a)$ where $x_1, \ldots, x_a$ are distinct elements in $[n]$. Let $f : [n] \to [n]$ be a uniformly random permutation, and let

$$I = \left\{ i \in [a] \mid \exists j \in [a], f(x_i) = x_j \right\}.$$

Notice that $|I| = |f(A) \cap A|$ where $A = \{x_1, \ldots, x_a\}$ has size a, and $f(A)$ is a uniformly random subset of $[n]$ of size a. Therefore,

$$\begin{aligned}
\Pr_{f \sim S_n}[|I| \geq S] &\leq \binom{n}{a}^{-1} \cdot \binom{a}{S}\binom{n}{a-S} \\
&= \frac{a!a!(n-a)!}{(a-S)!(a-S)!S!(n-a+S)!} \\
&\leq \frac{a^{2S}}{S!(n-a)^S} \leq \left(\frac{ea^2}{S(n-a)}\right)^S \leq 5^{-S}.
\end{aligned}$$

Now consider the probability

$$\Pr_{f \sim S_n}[\text{On input } f, \text{ at least } 3S \text{ distinct pairs of } (x, f(f(x))) \text{ are outputted in stage } k]. \tag{9.1}$$

When $|I| < S$, let us fix the query answers in stage k and examine (9.1). Among the $3S$ pair of outputs $(x, f(f(x)))$ at least S of them satisfies that $x \notin I$ and $f(f(x)) \notin f(I)$. For those x, $f(f(x))$ is simply uniformly distributed over the not-revealed $n - a$ elements, which means that these answers are correct with probability at most $(n - a - S)^{-S}$.

Thus by union bound over the 2^S starting points of stage k, we can bound the probabil-

ity in (9.1) by

$$2^S \left(\Pr_{f \sim S_n} [|I| \geq S] + (n - a - S)^{-S} \right) < 2^{-S}.$$

Without loss of generality, assume that $S > \log n$, then the above bound is smaller than $1/(2T) \leq 1/(2\ell)$. But on the other hand, since throughout the ℓ stages, n correct pairs of answers are outputted, it means that as long as $3\ell S < n$ there must be a stage k such that this probability is at least $1/\ell$, which is a contraction. Hence we have $3\ell S \geq n$, which translates to $T^2 S \geq \Omega(n^3)$.

Finally, notice that we actually proved that no deterministic oblivious branching program could succeed with probability $1/2$ on computing 2-PC when the input f is uniformly drawn over permutations S_n. By Yao's Minimax Principle, we conclude that no distribution over deterministic oblivious branching programs could computing 2-PC with probability $1/2$. □

Note that the final application of Yao's Minimax Principle is not a coincidence. In fact, by the design of Borodin-Cook method, we always first prove a lower bound on deterministic branching programs over some input distribution $\mathcal{D}$, and thus Yao's Minimax Principle is always applicable. Therefore, the Borodin-Cook method itself could not demonstrate a randomized vs. deterministic separation. In the next section, we show how to combine the Borodin-Cook method with the adversarial method to bypass this weakness.

9.2 Polynomial Separation for Oblivious Computation

Here we define our total function (n, p)-NOE that demonstrate the polynomial separation between randomized and deterministic oblivious branching programs.

Definition 9.2.1. *Let $n > 1$ and p be a prime factor of n. In the (n,p)-Non-Occurring Elements ((n,p)-NOE for short) problem, the input is an unordered list of n numbers $X = (x_1, \ldots, x_n) \in [n]^n$. The output is a set $Y \subseteq [n]$ such that:*

- *If for every $c \in [n]$, the number of times that c occurs in X is a multiple of p (0 included, so there are at most n/p distinct occurring elements), then Y consists of the (at least $n - n/p$) elements in $[n]$ that do not occur in X;*
- *Otherwise $Y = \varnothing$.*

Theorem 9.2.2. *There is a randomized oblivious branching program with space $O(\log n)$ and time $\max\{1, n/p^2\} \cdot O(n \log n)$, that computes (n,p)-NOE with probability at least $1 - 2/n$. Moreover, the algorithm can be implemented with one-way access to random bits. On the other hand, any deterministic oblivious branching program with space S and time T that correctly computes (n,p)-NOE must satisfy $T^2(S + \log T) \geq \Omega(n^3/p)$.*

Taking $n = p^2$, we get a polynomial separation with randomized upper bound $S = O(\log n)$, $T = O(n \log n)$ and deterministic lower bound $T^2 S \geq \widetilde{\Omega}(n^{2.5})$.

We note that our (n,p)-NOE problem could be perceived as a "promised" version of the Non-Occurring Elements problem (in which the output at all times consists of the elements not occurring in X), and the latter problem has time-space tradeoff $TS = \Theta(n^2)$ for both deterministic and randomized branching programs [MW18]. The promise that every elements occurs a multiple of p times can be efficiently checked with randomness (Lemma 9.2.3), however there may as well be a deterministic algorithm that verifies the promise in almost-linear time and poly-log space (subject to Open Problem 7.1). The above

facts imply that neither the Non-Occurring Elements problem nor the promise itself could demonstrate the desired separation.

We first prove the randomized upper bound for (n, p)-NOE, which consists of two parts: the algorithm for checking the promise Lemma 9.2.3, and the algorithm for solving Non-Occurring Elements under the promise Lemma 9.2.4.

Lemma 9.2.3. *There is a randomized algorithm using $O(\log n)$ space and $O(\log n)$ random bits that reads $X = (x_1, \ldots, x_n) \in [n]^n$ as a one-pass stream and satisfies that:*

- *If every $c \in [n]$ occurs in X a multiple of p times, the algorithm always accepts;*
- *Otherwise, the algorithm rejects with probability at least $1 - 2p^{-1/2} \log n$.*

Proof. The algorithm maintains a linear sketch of the frequencies of elements in $[n]$. Specifically, let $\alpha_1, \ldots, \alpha_n$ be uniformly and independently drawn from $\mathbb{F}_p$. The algorithm computes $\sum_i \alpha_{x_i}$ and accepts if the sum equals 0. If some $c \in [n]$ occurs not a multiple of p times, the factor before α_c in the sum is non-zero, and the sum equals 0 with probability $1/p$.

To reduce the random bit usage (the naive approach uses $n \log p$ random bits) we use Reed-Muller codes. Instead of drawing $\alpha_1, \ldots, \alpha_n$ independently, the algorithm draws $\beta_1, \ldots, \beta_m \in \mathbb{F}_p$ uniformly and independently, and let $\alpha_1, \ldots, \alpha_n$ be the values of monomials

$$\beta_1^{d_1} \beta_2^{d_2} \cdots \beta_m^{d_m}, \quad 0 \le d_1, \ldots, d_m < d.$$

By taking $d = p^{1/2}$ and $m = 2 \log n / \log p$, the number of such monomials is at least $d^m \ge n$. Since $m \log p = O(\log n)$, the algorithm can draw and store $\beta_1, \ldots, \beta_m$ directly.

After reading $x_i = c \in [n]$, $(c-1)$ is decomposed in base d to obtain $d_1, \ldots, d_m$ in sequence, while the algorithm computes $\alpha_c = \beta_1^{d_1}\beta_2^{d_2}\cdots\beta_m^{d_m}$ and accumulates it to the sum $\sum_i \alpha_{x_i}$.

Now the sum $\sum_i \alpha_{x_i}$ is a total degree md polynomial in $\mathbb{F}_p$ on variables $\beta_1, \ldots, \beta_m$, where the the coefficients are the frequencies of elements in $[n]$ occurring in X. If every $c \in [n]$ occurs in X a multiple of p times, the polynomial is always zero; Otherwise, the polynomial is non-zero, and by the Schwartz-Zippel Lemma, the probability that the polynomial evaluates to zero is at most $md/p \leq 2p^{-1/2}\log n$. □

Lemma 9.2.4. *Suppose $X = (x_1, \ldots, x_n) \in [n]^n$ satisfies that every $c \in [n]$ occurs in X either 0 or at least p times. Then there is a randomized oblivious algorithm using $O(\log n)$ space and $O(n^2p^{-2}\log n)$ time, with one-way access to random bits, that solves* Non-Occurring Elements *on X with probability at least $1 - 1/n$.*

Proof. Let $R \subseteq [n]$ be a random multi-set of size $r = 3np^{-1}\ln n$. As every occurring element occurs at least p times, the probability that $\{x_i \mid i \in R\}$ does not contain all occurring elements in X is at most

$$n \cdot (1 - p/n)^r \leq n \cdot e^{-3\ln n} = n^{-2}. \tag{9.2}$$

The algorithm goes for n/p rounds, in each round independently samples such a multi-set R of size r, and queries x_i for $i \in R$. The algorithm also stores an number j, which is initialized as 0, and in each round j is updated to

$$j' = \min\{x_i > j \mid i \in R\} \cup \{n+1\}.$$

At the end of each round, the algorithm outputs every number strictly between the pre-

updated j and j'. By (9.2) and a union bound, with probability at least $1 - 1/n$, in every round $\{x_i \mid i \in R\}$ contains all occurring elements (where there are at most n/p of them). In this case j goes through all occurring elements in order, and thus the outputs are exactly the non-occurring ones.

The overall time complexity is $rn/p = O(n^2 p^{-2} \log n)$, and since elements in R can be sampled sequentially to compute j' and no need to be stored, the only space usage is for storing j and j' which is $O(\log n)$. □

Note that Lemma 9.2.3 solves a decision problem and thus can be repeated for $O(\log n)$ times to amplify the success probability to $1 - 1/n$. Then combined with Lemma 9.2.4, we obtain the desired randomized oblivious upper bound of space $O(\log n)$ and time $O((1 + n/p^2) \cdot n \log n)$. We now show the deterministic lower bound, from which Theorem 9.2.2 follows.

Lemma 9.2.5. *Any deterministic oblivious branching program with space S and time T that correctly computes (n, p)-NOE must satisfy $T^2(S + \log T) \geq \Omega(n^3/p)$.*

Proof. Divide the branching program into $\ell = 2T/n$ stages, each of which contains $T/\ell = n/2$ queries. We first construct a partition $\mathcal{P}$ on $[n]$ that consists of n/p parts of size p as follows:

1. Initially, let $\mathcal{P} = \varnothing$.

2. For each stage of the branching program, let Q_k be the set of indices queried in the k-th stage. Arbitrarily pick $r = n^2/(4Tp)$ disjoint sets of size p outside $\bigcup\{P \in \mathcal{P}\} \cup Q_k$, and add them into $\mathcal{P}$.

3. Finally after going through all the stages, arbitrarily partition the remaining elements in $[n]$ into sets of size p.

Notice that during Step 2, the total number of elements in $\bigcup\{P \in \mathcal{P}\}$ never exceeds

$$r\ell p = \frac{n^2}{4Tp} \cdot \frac{2T}{n} \cdot p = n/2.$$

As $|Q_k| \leq n/2$, this implies that Step 2 is always feasible.

We define a distribution $\mathcal{D}$ of $X \in [n]^n$ as follows: For every part $P \in \mathcal{P}$, uniformly and independently pick $c \in [n]$ and let $x_i = c$ for all $i \in P$. Notice that the (n, p)-NOE problem is identical to Non-Occurring Elements on $\mathrm{supp}(\mathcal{D})$. Now consider the probability

$$\Pr_{X \sim \mathcal{D}}[\text{On input } X\text{, at least } m \text{ distinct elements are outputted in stage } k]. \tag{9.3}$$

Since for each input $X \in \mathrm{supp}(\mathcal{D})$ there are at least $n - n/p \geq n/2$ non-occurring elements, there must exist a stage k such that the above probability is at least $1/\ell$ for $m = n/(2\ell)$.

On the other hand, Step 2 in the construction of $\mathcal{P}$ implies that, given the query answers in stage k (i.e. x_i for all $i \in Q_k$), there are at least r parts in $\mathcal{P}$ whose values in X are still uniformly random. When the query answers are given, there are at most 2^S different collections of outputs in stage k (dictated by the starting state of the stage), and if m distinct elements are outputted and thus non-occurring, each one of the r parts is consistent with these outputs with probability $1 - m/n$. Therefore the probability in (9.3) is upper

bounded by

$$2^S \cdot \left(1 - \frac{m}{n}\right)^r = 2^S \cdot \left(1 - \frac{n}{4T}\right)^{\frac{n^2}{4Tp}} \leq 2^S \cdot e^{-\frac{n^3}{16T^2p}}.$$

As the probability is at least $1/\ell \geq 1/T$, we have

$$S - \frac{\log e \cdot n^3}{16T^2p} \geq -\log T \quad \Rightarrow \quad T^2(S + \log T) \geq \Omega(n^3/p). \qquad \square$$

9.3 Separations that Imply Decision Lower Bounds

In this section we present several natural candidates of multi-output function for randomized vs. deterministic separations, and show that actually proving such separations will lead to answering Open Problem 7.1. These results can be perceived in two ways: On one hand, these are currently barrier results implying that proving separations for these natural problems is difficult, which is where the (n, p)-NOE problem in our main result stands out; On the other hand, one may hope that future developments in proving lower bounds for multi-output functions will help towards the final resolution of Open Problem 7.1.

Before getting into the concrete examples, we note that every multi-output function $F : \{0,1\}^n \to \{0,1\}^m$ can be converted to a decision problem $F' : \{0,1\}^n \times [m] \to \{0,1\}$ defined as $F'(x, i) = F(x)_i$. Therefore, if F' can be computed in space $O(\log n)$ and time $\widetilde{O}(n)$, then F can trivially be computed in space $O(\log n)$ and time $\widetilde{O}(mn)$. Our results in this section holds non-trivially with better time complexity than $\widetilde{O}(mn)$. However, this implication is still useful as it makes decision problems and single-output functions (whose outputs are in $[n]$, or generally have length $m = \text{polylog}(n)$) morally equivalent with respect to Open Problem 7.1: Any lower bound for a single-output function that is polyno-

mially better than trivial implies a corresponding lower bound for a decision problem.

9.3.1 Pointer Chasing and Expanders

Recall the definition of the 2-StepPointerChasing problem, where the input is a function $f : [n] \to [n]$, and the output consists of $(x, f(f(x)))$ for all $x \in [n]$. For non-oblivious algorithms, 2-PC can be easily solved in deterministic space $O(\log n)$ and time $O(n)$, by querying f on each x and adaptively on $f(x)$. On the other hand for oblivious algorithms, we have the $T^2S \geq \widetilde{\Omega}(n^3)$ lower bound in Proposition 9.1.2, which provides an example of polynomial separation between oblivious and non-oblivious time-space tradeoffs of total functions.

The bound is also tight and can be achieved via the following simple algorithm: In each round pick two random subsets $X, Y \subseteq [n]$ with $|X| = |Y| = \sqrt{nS}$. We store at most $\widetilde{O}(S)$ pairs of $(x, f(x)) \in X \times Y$ by querying f on X, and output the corresponding $(x, f(f(x)))$ by querying f on Y. Each pair in a round is found with probability close to S/n, and thus $\widetilde{O}(n/S)$ rounds suffices.

The above algorithm heavily relies on the fact that Y is decided entirely by randomness and hardwired into the branching programs. A natural question is whether the same time-space tradeoff holds for oblivious computation with weaker notions of randomness, or even without randomness at all. In Theorem 9.3.4 below, we show that proving impossibility results to this question will give answers to Open Problem 7.1. We first need to introduce the single-output function, ExpanderMatching based on the explicit unbalanced bipartite expanders by Guruswami, Umans and Vadhan [GUV09].

Definition 9.3.1. *A bipartite graph $\Gamma \subseteq [n] \times [m]$ is a (k, a)-expander if for every subset*

$L \subseteq [n]$ with $|L| \leq k$, the number of the neighbors of L is at least $a \cdot |L|$.

Theorem 9.3.2 ([GUV09]). *For every constant $\alpha > 0$, given $n \in \mathbb{N}$ and $k \leq n$, there is an explicitly constructed bipartite graph $\Gamma_{\alpha,n,k} \subseteq [n] \times [m]$ which is a $(k, 1)$-expander, with $|\Gamma_{\alpha,n,k}| = \widetilde{O}(n)$ and $m \leq \widetilde{O}(k^{1+\alpha})$.*

The original result in [GUV09] is stronger than stated in Theorem 9.3.2, with the expansion factor a arbitrarily close to the degree $|\Gamma_{\alpha,n,k}|/n = \text{polylog}(n)$. For our application, we only need expansion to be no less than 1. We use the graph to construct an explicit single-output function as follows:

Definition 9.3.3. *The (α, n, k)-ExpanderMatching problem is a function $[n]^k \times [m] \to [n] \cup \{\perp\}$, with m decided by Theorem 9.3.2. Given the input $L \in [n]^k$ and $y \in [m]$, we think of L as a subset of $[n]$ with $|L| \leq k$. There exists a matching for L in $\Gamma_{\alpha,n,k}$ because of the $(k, 1)$-expander property, and we consider the lexicographically smallest matching $\mathcal{M} : L \to [m]$ in $\Gamma_{\alpha,n,k}$. The output of the problem is $\mathcal{M}^{-1}(y)$ if it exists, or $\perp$ if not.*

Theorem 9.3.4. *For every constant $\alpha > 0$, if (α, n, k)-ExpanderMatching can be solved by deterministic oblivious branching programs with space $\widetilde{O}(1)$ and time $\widetilde{O}(k)$, then for every $S \leq n$, there is a deterministic oblivious branching program solving 2-PC with space $\widetilde{O}(S)$ and time $\widetilde{O}(\sqrt{n^{3+\alpha}/S})$.*

Proof. We partition $[n]$ into blocks $B_1 \sqcup \cdots \sqcup B_{n/k}$ of size k, with k to be optimally chosen later. The deterministic oblivious algorithm for 2-PC consists of $n/(kS)$ stages, where in each stage we output $(x, f(f(x)))$ all x in S consecutive blocks. In order to do so we need to query f on $f(B_i)$, but as the queries are oblivious, we instead query f on the neighbors of y

for each $y \in [m]$. Since $|f(B_i)| \leq k$, the matching for $f(B_i)$ provides all the answers for $x \in B_i$. More concretely, the algorithm is described in Algorithm 9.1.

Algorithm 9.1: The deterministic oblivious algorithm for 2-PC

```
for ℓ ← 0, ..., n/(kS) − 1 do
    for y ∈ [m] do
        for i ∈ [S] do
            Apply (α, n, k)-ExpanderMatching on f(B_{i+ℓS}) ∈ [n]^k and y;
            Store the answer u_i ∈ [n] ∪ {⊥}.
        foreach v ∈ [n] such that (v, y) ∈ Γ_{α,n,k} do
            Query f(v);
            if v = u_i for some i ∈ [S] then attach f(u_i) to u_i.
        for x ∈ B_{ℓS+1} ⊔ ··· ⊔ B_{(ℓ+1)S} do
            Query f(x);
            if f(x) = u_i for some i ∈ [S] then output (x, f(u_i)).
```

To prove the correctness, it suffices to show that every $x \in [n]$ is outputted. This is guaranteed in every block $B_{i+\ell S}$, as when y goes through $[m]$, every element in $f(B_{i+\ell S})$ is matched and appears as u_i at some point.

The space complexity is clearly $\widetilde{O}(S)$ as the bottleneck is storing u_i and $f(u_i)$ for $i \in [S]$. To identify the time complexity, notice that f is queried in all three inner loops. For each ℓ and y, the (α, n, k)-Expander Matching algorithm makes $\widetilde{O}(kS)$ queries in total, while querying $f(x)$ for $x \in B_{\ell S+1} \sqcup \cdots \sqcup B_{(\ell+1)S}$ also takes $O(kS)$ time. Besides, for each ℓ, querying $f(v)$ for every edge $(v, y) \in \Gamma_{\alpha,n,k}$ takes up $|\Gamma_{\alpha,n,k}| = \widetilde{O}(n)$ time. Therefore the total number of oblivious queries is

$$\frac{n}{kS}\left(m \cdot \widetilde{O}(kS) + \widetilde{O}(n)\right) = \widetilde{O}\left(k^{1+\alpha}n + \frac{n^2}{kS}\right).$$

Taking $k = \sqrt{n/S}$, the above expression is upper bounded by $\widetilde{O}(\sqrt{n^{3+\alpha}/S})$. □

As a direct corollary of Theorem 9.3.4, if we managed to prove a polynomial separation between randomized and deterministic oblivious time-space tradeoffs of 2-PC, it would imply a strong lower bound for (α, n, k)-ExpanderMatching for some $\alpha > 0$ and thus would answer Open Problem 7.1.

9.3.2 Element Distinctness and Collision Finding

We recall the definition of the ElementDistinctness problem.

Definition 9.3.5. *In the ElementDistinctness (ED for short) problem, the input is a list of n elements from a fixed domain D, with $|D| = \text{poly}(n)$. The output is 1 if all elements are distinct, and 0 otherwise.*

A randomized algorithm for ED with $T^2S = \widetilde{O}(n^3)$ was given in [BCM13], and it was later improved to use only one-way access to randomness in [CJWW22, LZ23]. Based on the same algorithm, they also showed that the SetIntersection problem (given two sets A and B of size n, output $A \cap B$) can be solved $T^2S = \widetilde{O}(n^3)$, and the tradeoff is known to be tight [Din20]. Different variants of this problem was also studied, such as memory games [CC17] and n-collision finding [Din20], which share the same tight tradeoff for randomized algorithms.

Here we present a general form of SetIntersection, that covers all the variants when two sets that contains no duplicates are given, and show its black-box relationship with ED:

Definition 9.3.6. *In the SetCollision problem, the input contains two sets $A, B \subseteq D$ given as unordered lists $(a_1, \ldots, a_n)$ and $(b_1, \ldots, b_n)$ that contain no duplicated elements in each list itself. The output consists of all collisions, that are triples (i, j, x) such that $a_i = b_j = x$.*

Theorem 9.3.7. *If ED can be solved deterministically with space $\widetilde{O}(1)$ and time $\widetilde{O}(n)$, then* SetCollision *can be solved deterministically with space $\widetilde{O}(1)$ and time $\widetilde{O}(n^{3/2})$. Furthermore, if the algorithm for ED is oblivious, then for every $S \leq n$, there is a deterministic (non-oblivious) algorithm that solves* SetCollision *with space $\widetilde{O}(S)$ and time $\widetilde{O}(\sqrt{n^3/S})$.*

Proof. We first present a simple divide-and-conquer algorithm $\mathcal{A}$ for solving SetCollision. The algorithm $\mathcal{A}(\ell, s, s')$ is described recursively as Algorithm 9.2, where for the sake of simplicity we assume that ℓ is a power of 2:

Algorithm 9.2: Recursive algorithm $\mathcal{A}(\ell, s, s')$ for SetCollision

```
1 if ℓ = 1 then
2 |   if a_s = b_{s'} then Output (s, s', a_s);
3 |   return.
4 if ED(a_s, ..., a_{s+ℓ-1}, b_{s'}, ..., b_{s'+ℓ-1}) = 1 then return.
5 let ℓ' ← ℓ/2;
6 Sequentially execute A(ℓ', s, s'), A(ℓ', s + ℓ', s'), A(ℓ', s, s' + ℓ') and
    A(ℓ', s + ℓ', s + ℓ').
```

It is easy to see that $\mathcal{A}(\ell, s, s')$ outputs all the collisions between the two intervals

$$a[s, \ldots, s + \ell - 1] \quad \text{and} \quad b[s', \ldots, s' + \ell - 1],$$

since whenever $\ell > 1$ and there exists at least one collision (which is checked by the ED call), the algorithm splits each interval into two sub-intervals of half length, and solve all four pairs of sub-intervals with the four recursive calls. Hence $\mathcal{A}(n, 1, 1)$ solves SetCollision.

The space usage of $\mathcal{A}(n, 1, 1)$ is $\widetilde{O}(1)$, since there are $O(\log n)$ levels of recursion and each recursive call locally uses $\widetilde{O}(1)$ space. To bound the time usage, the key observation is

that there are at most n collisions. Therefore, although there could be as much as $(n/\ell)^2$ possible recursive calls to $\mathcal{A}$ at the level of recursion with interval length ℓ, there are in fact at most $O(n)$ actual calls within each level, while the rest are prematurely stopped because of the ED check. Taking the summation over $\ell = 2^t$ for $t = 0, \ldots, \log n$, the total time usage of $\mathcal{A}(n, 1, 1)$ bounded by

$$\sum_{t \le \frac{1}{2}\log n} O(n) \cdot \widetilde{O}(2^t) + \sum_{t > \frac{1}{2}\log n} \left(\frac{n}{2^t}\right)^2 \cdot \widetilde{O}(2^t) = \widetilde{O}(n^{3/2}).$$

When the space S is larger, in order to leverage the space advantage and reduce the time usage we need to *parallelize* the algorithm $\mathcal{A}$. However, the core of algorithm $\mathcal{A}$ is the black-box ED algorithm, whose instances cannot be parallelized if they are highly adaptive. Therefore from now on, we assume that the space-$\widetilde{O}(1)$ and time-$\widetilde{O}(n)$ ED algorithm is oblivious.

To understand how oblivious ED algorithm helps parallelization, consider the recursion level with $\ell = \sqrt{n}$. At this level, we need to answer $\mathrm{ED}(a_s, \ldots, a_{s+\ell-1}, b_{s'}, \ldots, b_{s'+\ell-1})$ for all n pairs of $s, s' \in \{1, \sqrt{n}+1, \ldots, n-\sqrt{n}+1\}$. We can call the oblivious ED algorithm to solve the instance with $s = s' = 1$, and call it again to solve another instance with $s = \sqrt{n}, s' = 1$. Because the algorithm is oblivious, whenever a_i (resp. b_i) is queried in the first instance, $a_{i+\sqrt{n}}$ (resp. b_i) is queried in the second instance at the exact same time step. That means the two algorithm instance can be interleaved, using double the space while the queries to B do not need to be repeated. Take a step further, we can interleave the 4 instances of ED with $s, s' \in \{1, \sqrt{n}+1\}$, using 4 times the space but only *double* the time.

In our actual algorithm, we partition $\{1, \sqrt{n}+1, \ldots, n-\sqrt{n}+1\}$ into $\sqrt{n/S}$ groups,

each of size $\sqrt{S}$. With the idea stated above, for each pair of groups of s and s', we can solve all the ED instances within this pair (where there are S instances) with space $\widetilde{O}(S)$ and time $\widetilde{O}(\sqrt{nS})$. As there are n/S pairs of groups, the overall time usage all the ED instances at level $\ell = \sqrt{n}$ is $\widetilde{O}(\sqrt{n^3/S})$. More generally, using the same idea, we design a parallelized version of $\mathcal{A}$, which is the algorithm $\mathcal{A}^*(\ell, (s_i, s'_i)_{i\in I})$ described in Algorithm 9.3, that takes as an argument a list of $|I| \leq S$ pairs of s and s'.

Algorithm 9.3: Recursive algorithm $\mathcal{A}^*(\ell, (s_i, s'_i)_{i\in I})$ for SetCollision

```
1  if ℓ = 1 then
2  |  for i ∈ I do
3  |  |  if a_{s_i} = b_{s'_i} then Output (s_i, s'_i, a_{s_i});
4  |  return.
5  Solve e_i ← ED(a_{s_i}, ..., a_{s_i+ℓ−1}, b_{s'_i}, ..., b_{s'_i+ℓ−1}) for all i ∈ I in parallel;
6  let ℓ' ← ℓ/2, Q ← ∅;
7  foreach i ∈ I such that e_i = 0 do
8  |  for (Δs, Δs') ← (0,0), (ℓ',0), (ℓ',0), (ℓ',ℓ') do
9  |  |  Add (s_i + Δs, s'_i + Δs') to the queue Q;
10 |  |  if |Q| = S or reaching the end of the algorithm then
11 |  |  |  Execute A*(ℓ', Q);
12 |  |  |  Q ← ∅.
```

It is clear from the description that $\mathcal{A}^*(\ell, (s_i, s'_i)_{i\in I})$ functions the same as the sequential execution of $\mathcal{A}(\ell, s_i, s'_i)$ for all $i \in I$. Our final algorithm for SetCollision is to run sequentially $\mathcal{A}^*(\sqrt{n}, G \times G')$, for all G and G' chosen from the $\sqrt{n/S}$ groups of size $\sqrt{S}$ that partitions $\{1, \sqrt{n}+1, \ldots, n-\sqrt{n}+1\}$, and thus it correctly outputs all collisions between set A and B. Each recursive call of $\mathcal{A}^*$ uses $O(S \log n)$ space locally, plus the $\widetilde{O}(S)$ space to compute at most S instances of ED in parallel. As there are $O(\log n)$ levels of recursion, the overall space usage is $\widetilde{O}(S)$.

To bound the time usage, we first examine how much time is used to solve S instances

of ED in parallel. Fix the initial argument G and G' at the start of the recursion and focus on one level of recursion with interval length ℓ. At this level, one instance of the ED algorithm takes $\widetilde{O}(\ell)$ time. Since the input intervals for these ED instances are either the same or disjoint, each query is repeated for at most $|G| \cdot \sqrt{n}/\ell$ times at its parallel places after the interleaving parallelization. Thus the time usage for solving ED is $\widetilde{O}(|G| \cdot \sqrt{n}) = \widetilde{O}(\sqrt{nS})$. As the rest of steps take $O(S) \leq O(\sqrt{nS})$ time, altogether each recursive call of $\mathcal{A}^*$ locally takes $\widetilde{O}(\sqrt{nS})$ time, regardless of the level of recursion.

On the other hand, let us call a recursive call $\mathcal{A}^*(\ell, (s_i, s'_i)_{i \in I})$ *complete* if $|I| = S$, and *incomplete* if $|I| < S$. Since there are at most n collisions, at each level of the recursion there are at most $O(n/S)$ complete calls, while each call produces at most one incomplete call in the next level. Initially there are n/S calls, and therefore the total number of calls to $\mathcal{A}^*$ in our final algorithm is $\widetilde{O}(n/S)$. So the total running time is $\widetilde{O}(\sqrt{n^3/S})$. □

Since SetCollision has the randomized lower bound $T^2S = \widetilde{\Omega}(n^3)$, Theorem 9.3.7 implies that any polynomial separation between randomized and deterministic time-space tradeoffs of SetCollision (or its variants such as SetIntersection) would answer Open Problem 7.1 on ElementDistinctness.

Notice that in the reduction of Theorem 9.3.7, the input guarantee that both lists A and B are sets is only used so that ED decides the distinctness between the two lists. Without the guarantee, we can instead resort to the ListDistinctness problem studied in [BGNV18].

Definition 9.3.8. *In the ListDistinctness problem (LD for short), the input contains two unordered lists $(a_1, \ldots, a_n)$ and $(b_1, \ldots, b_n)$ from a fixed domain D, with $|D| = \text{poly}(n)$. The output is 1 if there exist $i, j \in [n]$ such that $a_i = b_j$, and 0 otherwise.*

LD is at least as harder as ED, and while ED can be solved in $\widetilde{O}(1)$ space and $\widetilde{O}(n^{3/2})$ time, no algorithm even with $n^{o(1)}$ space and $n^{2-\Omega(1)}$ time was known for LD. The proof of Theorem 9.3.7 can be altered to show that the problem of n-Collision reduces deterministically to LD:

Definition 9.3.9. *In the n-Collision problem, the input is an unordered list $(a_1, \ldots, a_n)$ of elements in D. The output consists of n distinct collisions, that are triples (i, j, x) such that $i \neq j$ and $a_i = a_j = x$, or all of the collisions if there are less than n of them.*

Strictly speaking, the n-Collision problem is not a function, but rather a relational problem, as the collection of outputted collisions is not uniquely determined. However, a time-space lower bound of $T^2S = \widetilde{\Omega}(n^3)$ is still know for n-Collision [Din20].

Theorem 9.3.10. *If LD can be solved deterministically with space $\widetilde{O}(1)$ and time $\widetilde{O}(n)$, then n-Collision can be solved deterministically with space $\widetilde{O}(1)$ and time $\widetilde{O}(n^{3/2})$. Furthermore, if the algorithm for LD is oblivious, then for every $S \leq n$, there is a deterministic (non-oblivious) algorithm that solves n-Collision with space $\widetilde{O}(S)$ and time $\widetilde{O}(\sqrt{n^3/S})$.*

Proof. Notice that the collisions found in the algorithms in Theorem 9.3.7 are all distinct. By setting a global counter for the number of collisions already found and outputted, the algorithms and proofs in Theorem 9.3.7 can be copied verbatim to show a reduction to LD from the problem k-ListCollision, where the input consists of two unordered lists of size n that may contain duplicates, and the output contains k collisions (if exist) between the two lists. If LD can be solved deterministically with space $\widetilde{O}(1)$ and time $\widetilde{O}(n)$, then the deterministic algorithm for k-ListCollision works in space $\widetilde{O}(S)$ and time $\max\{m, n\} \cdot \widetilde{O}(\sqrt{n/S})$, where m is the actual number of collisions outputted (S can be arbitrary when the algorithm for LD is oblivious, and $S = O(1)$ in the general case).

Now notice that the complete graph over n vertices can be partitioned into a set of complete bipartite graphs, 2^{t-1} of which being of size $(n/2^t, n/2^t)$ for $t = 1, \ldots, \log n$. We apply n-ListCollision on each pairs of lists of size $n/2^t$ defined by these bipartite graphs, until n collisions are found. This clearly solves the n-Collision problem with space $\widetilde{O}(S)$.

Suppose that the number of collisions actually outputted on each bipartite graph is $m_1, m_2, \ldots$ respectively, then the total time usage is

$$\begin{aligned}
&\max\left\{m_1, \frac{n}{2}\right\} \cdot \widetilde{O}\left(\sqrt{\frac{n}{2S}}\right) + \max\left\{m_2, \frac{n}{4}\right\} \cdot \widetilde{O}\left(\sqrt{\frac{n}{4S}}\right) \\
&+ \max\left\{m_3, \frac{n}{4}\right\} \cdot \widetilde{O}\left(\sqrt{\frac{n}{4S}}\right) + \cdots \\
\leq\ & (m_1 + m_2 + \cdots) \cdot \widetilde{O}\left(\sqrt{\frac{n}{2S}}\right) + \sum_{t=1}^{\log n} 2^{t-1} \cdot \frac{n}{2^t} \cdot \widetilde{O}\left(\sqrt{\frac{n}{2^t S}}\right) \\
=\ & \widetilde{O}\left(\sqrt{n^3/S}\right).
\end{aligned}$$

□

Similarly, we have the corollary of Theorem 9.3.10 that any polynomial separation between randomized and deterministic time-space tradeoffs of n-Collision (or its variants such as MemoryGame [CC17]) would answer Open Problem 7.1 on ListDistinctness.

10

Learning with Classical-Quantum Hybrid Memory

In the final chapter of this dissertation, we study the learning problems with the presence of quantum memory. Consider the parity learning problem: Let $x \in \{0,1\}^n$ be uniformly random an hidden from the learner, the goal is to learn x from samples (a, b), where $a \in \{0,1\}^n$ is uniformly random and $b = \langle a, x \rangle$ under $\mathbb{F}_2$. In [Raz18] it was shown that either $\Omega(n^2)$ space or $2^{\Omega(n)}$ time (samples) is required to learn x classically. We will show that when we have both classical and quantum memory, we then need either $\Omega(n^2)$ classical space, or $\Omega(n)$ quantum space, or $2^{\Omega(n)}$ time.

More generally, following previous works [Raz18, KRT17, Raz17, GRT18, GRT19, GKLR21], we use a matrix M to represent the following learning problem. There is an unknown element $x \in \mathcal{X}$ that was chosen uniformly at random. A learner tries to learn x from samples (a, b), where $a \in \mathcal{A}$ is chosen uniformly at random and $b = M(a, x)$. That is, the learning algorithm is given a stream of samples, $(a_1, b_1), (a_2, b_2), \ldots$, where each

a_t is uniformly distributed and for every t, $b_t = M(a_t, x)$. When M is a (k, ℓ)-extractor with error 2^{-r}, we show in Theorem 10.3.1 that either $\Omega(k \cdot \ell)$ bits of classical memory, or $\Omega(r)$ qubits of quantum memory, or $2^{\Omega(r)}$ time is required for the learning problem corresponding to M.

10.1 Classical-Quantum Hybrid Model

10.1.1 Classical-Quantum Systems

Let us first rigorously define what it means by having classical and quantum hybrid memory. Consider a two-part quantum system on registers X and Y represented by the density operator ρ_{XY}. We say X is classical, if for every $|x\rangle \neq |x'\rangle$ in the computational basis of X, we have

$$(\langle x'| \otimes \mathbb{I}_Y)\rho_{XY}(|x\rangle \otimes \mathbb{I}_Y) = 0.$$

In this case, we can identify the space of X with its computational basis, and remove the Dirac notation when we talk about the values of X. The system ρ_{XY} can also be written as a direct sum

$$\rho_{XY} = \bigoplus_x \rho_{Y|x}.$$

Also notice that when X is classical, both ρ_X and $\rho_{X|y}$ for all quantum states $|y\rangle$ on Y are diagonal. If $\mathrm{Tr}[\rho_{X|y}] > 0$, it induces a distribution over the computation basis of X, defined as

$$P^{\rho}_{X|y} = \mathrm{diag}\rho_{X|y}/\mathrm{Tr}[\rho_{X|y}]. \tag{10.1}$$

From now on whenever we use this notation, it is always implicitly assumed that the corresponding $\mathrm{Tr}[\rho_{X|y}]$ is non-zero and the distribution exists.

We will typically consider the following scenario: There is a quantum memory register V in the complex linear space $\mathcal{V}$, and a classical memory register W ranging in the set of classical memory states $\mathcal{W}$, along with some classical information $X \in \mathcal{X}$ (the concept to be learned) that is correlated with V and W. We will often make use of the following fact:

Claim 10.1.1. *Let ρ_{XVW} be a classical-quantum system over classical X, W and quantum V. For every $w \in \mathcal{W}$, $P^{\rho}_{X|w}$ is a convex combination of $P^{\rho}_{X|v,w}$ for some $\{|v\rangle\} \subseteq \mathcal{V}$.*

Proof. Let $\mathcal{B}$ be an orthogonal basis of $\mathcal{V}$, so that we have (from the end of last section)

$$\rho_{X|w} = \sum_{|v\rangle \in \mathcal{B}} \rho_{X|v,w}.$$

Therefore $P^{\rho}_{X|w}$ is a linear combination of $P^{\rho}_{X|v,w}$ for $|v\rangle \in \mathcal{B}$, with non-negative coefficients. Since they are all distributions, it must be a convex combination. □

Now we identify all possible operators on the classical-quantum hybrid memory space $\mathcal{V} \otimes \mathcal{W}$. A priori to the assumption that W is classical, we think of a quantum channel operating on the system as working on the underlying space $\mathcal{V} \otimes \mathbb{C}^{|\mathcal{W}|}$. Now we denote $\mathscr{T}_{\mathcal{V}\otimes\mathcal{W}}$ to be the set of all such quantum channels Φ that satisfy the following: for every classical-quantum system ρ_{VW} in $\mathcal{V} \otimes \mathcal{W}$, W is still classical in $\Phi(\rho_{VW})$. That is, for every two states $|v\rangle, |v'\rangle \in \mathcal{V}$ and every pair of distinct $w, w' \in \mathcal{W}$, we have

$$\langle v, w|\Phi(\rho_{VW})|v', w'\rangle = 0.$$

Note that not all channels in $\mathcal{T}_{\mathcal{V}\otimes\mathcal{W}}$ are physically realizable. For instance, with one-bit classical memory and no quantum memory, the channel

$$\begin{pmatrix} a & c \\ \bar{c} & b \end{pmatrix} \mapsto \begin{pmatrix} a & ic \\ -i\bar{c} & b \end{pmatrix}$$

is not a classical operator. However, since we are constrained to classical quantum systems, this channel is effectively equivalent to an identity channel on one-bit classical memory. Generally speaking, every channel in $\mathcal{T}_{\mathcal{V}\otimes\mathcal{W}}$ is equivalent to a channel controlled by $\mathcal{W}$ that maps $\mathcal{V}$ to $\mathcal{V}\otimes\mathcal{W}$. Below, we prove this observation and use it to show the following claim:

Claim 10.1.2. *Let ρ_{XVW} be a classical-quantum system over classical X, W and quantum V. Let $\Phi \in \mathcal{T}_{\mathcal{V}\otimes\mathcal{W}}$, and we use $\Phi(\rho)$ to denote the system after applying Φ to VW and identity to X. Then for every $|v\rangle \in \mathcal{V}$ and $w \in \mathcal{W}$, $P^{\Phi(\rho)}_{X|v,w}$ is a convex combination of $P^{\rho}_{X|v',w'}$ for some $\{|v'\rangle\} \subseteq \mathcal{V}$ and $\{w'\} \subseteq \mathcal{W}$.*

Notice that unlike Claim 10.1.1, in Claim 10.1.2 it is not always possible to write $P^{\Phi(\rho)}_{X|v,w}$ as a convex combination of $P^{\rho}_{X|v',w'}$ for $|v'\rangle$ from an orthogonal basis of $\mathcal{V}$.

Proof. Since $\Phi \in \mathcal{T}_{\mathcal{V}\otimes\mathcal{W}}$, the following channel is functionally equivalent to Φ on classical-quantum systems:

$$\Phi' : \rho \to \sum_{w\in\mathcal{W}} \Phi(\rho_{V|w} \otimes |w\rangle\langle w|).$$

The physical meaning of Φ' is to measure W under the computational basis (which should not change the functionality we care about) and apply Φ.

By defining the channel $\Phi_w(\cdot) := \Phi(\cdot \otimes |w\rangle\langle w|)$, the above can be alternatively written as:

$$\Phi' : \rho \to \sum_{w \in \mathcal{W}} \Phi_w(\rho_{V|w}).$$

Now consider the Kraus representation of each Φ_w, that is, a finite set of linear operators $E_{w,k} : \mathcal{V} \to \mathcal{V} \otimes \mathcal{W}$ such that

$$\Phi_w(\rho_{V|w}) = \sum_k E_{w,k} \rho_{V|w} E_{w,k}^\dagger, \qquad \sum_k E_{w,k}^\dagger E_{w,k} = \mathbb{I}_V.$$

We can write

$$\begin{aligned}
\Phi(\rho)_{X|v,w} = \Phi'(\rho)_{X|v,w} &= (\mathbb{I}_X \otimes \langle v, w|)\Phi'(\rho)(\mathbb{I}_X \otimes |v, w\rangle) \\
&= \sum_{w' \in \mathcal{W}} \sum_k (\mathbb{I}_X \otimes \langle v, w|E_{w',k})\rho_{XV|w'}(\mathbb{I}_X \otimes E_{w',k}^\dagger|v, w\rangle) \\
&= \sum_{w' \in \mathcal{W}} \sum_k \left\| E_{w',k}^\dagger |v, w\rangle \right\|_2 \cdot \rho_{X|v',w'}
\end{aligned}$$

where in each term of the summation, $|v'\rangle \sim E_{w',k}^\dagger |v, w\rangle$. Similar to the arguments in Claim 10.1.1, $P_{X|v,w}^{\Phi(\rho)}$ is a convex combination of $P_{X|v',w'}^{\rho}$. □

10.1.2 Branching Program with Hybrid Memory

For a learning problem that corresponds to the matrix M, a branching program of hybrid memory with m-bit classical memory, q-qubit quantum memory and length T is specified as follows.

At each stage $0 \leq t \leq T$, the memory state of the branching program is described as a classical-quantum system $\rho_{VW}^{(t)}$ over quantum memory space $\mathcal{V} = (\mathbb{C}^2)^{\otimes q}$ and classical memory space $\mathcal{W} = \{0,1\}^m$. The memory state evolves based on the samples that the branching program receives, and therefore depends on the unknown element $x \in_R \mathcal{X}$. We can then interpret the overall systems over XVW, in which X consists of an unknown concept x, resulting in a classical-quantum system $\rho_{XVW}^{(t)}$. It always holds that the distribution of x is uniform, i.e.,

$$\rho_X^{(t)} = \mathrm{Tr}_{VW}[\rho_{XVW}^{(t)}] = \frac{1}{2^n}\mathbb{I}_X.$$

Initially the memory VW is independent of X and can be arbitrarily initialized. We assume that it starts from the maximally mixed state

$$\rho_{XVW}^{(0)} = \frac{1}{2^n}\mathbb{I}_X \otimes \frac{1}{2^q}\mathbb{I}_V \otimes \frac{1}{2^m}\mathbb{I}_W.$$

At each stage $0 \leq t < T$, the branching program receives a sample (a,b), where $a \in_R \mathcal{A}$ and $b = M(a,x)$, and applies an operation $\Phi_{t,a,b} \in \mathcal{T}_{\mathcal{V}\otimes\mathcal{W}}$ over its memory state. Thus the evolution of the entire system can be written as

$$\rho_{XVW}^{(t+1)} = \mathop{\mathbb{E}}_{a\in_R\mathcal{A}} \left[\sum_{x\in\mathcal{X}} |x\rangle\langle x| \otimes \Phi_{t,a,M(a,x)}\left(\rho_{VW|x}^{(t)}\right) \right].$$

Finally, at stage $t = T$, a measurement over the computational bases is applied on $\rho_{VW}^{(T)}$, and the branching program outputs an element $\widetilde{x} \in X$ as a function of the measurement result $(v,w) \in \{0,1\}^{q+m}$. The success probability of the program is the probability that $\widetilde{x} = x$

which can be formulated as

$$\sum_{\substack{x\in\mathcal{X},v\in\{0,1\}^q,w\in\mathcal{W}\\ \widetilde{x}(v,w)=x}} \langle x,v,w|\rho^{(T)}_{XVW}|x,v,w\rangle.$$

10.2 Linear Quantum Lower Bound

Let us first prove a linear quantum memory lower bound for learning algorithms without classical memory. This will not be used as a part of our actual proof of the main theorem, but since the proof is fully information theoretical, it is simple enough to be included.

We first define the quantum extractor property that we need, which is a simplified version of the ones considered in [KK12]. Given a matrix $M : \mathcal{A}\times\mathcal{X}\to\{-1,1\}$, consider two independent sources A and X uniformly distributed over $\mathcal{A}$ and $\mathcal{X}$ respectively. Suppose there is some quantum register V whose state depends on A and X, and they together form a classical-quantum system

$$\rho_{AXV} = \bigoplus_{a\in\mathcal{A},x\in\mathcal{X}} \rho_{V|a,x},$$

where $\rho_{V|a,x}$ is the state of V when $A=a$ and $X=x$. For any function f on $A\times X$, we say that V depends only on $f(A,X)$ if for any $a,a'\in\mathcal{A}$ and $x,x'\in\mathcal{X}$, whenever $f(a,x)=f(a',x')$ we have $\rho_{V|a,x}=\rho_{V|a',x'}$. In particular, V depending only on A is equivalent to V being independent of X, or $\rho_{XV}=\rho_X\otimes\rho_V$.

We say that M is an X-strong (q,r)-quantum extractor, if for every classical-quantum system ρ_{AXV}, as above, with the q-qubit quantum subsystem V that depends only on A, it

holds that

$$\left\| \rho_{M(A,X)XV} - U \otimes \rho_X \otimes \rho_V \right\|_{\mathrm{Tr}} \leq 2^{-r}.$$

Here $U = \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix}$ is the uniform operator over one bit, and $\rho_{M(A,X)XV}$ is the classical-quantum system constructed by adding a new classical register which stores the value of $M(A,X)$, and then tracing out A. In other words,

$$\rho_{M(A,X)XV} = \bigoplus_{b \in \{-1,1\}, x \in \mathcal{X}} \sum_{\substack{a \in \mathcal{A} \\ M(a,x)=b}} \rho_{V|a,x}.$$

Notice that if we choose V to be trivial, the above inequality immediately implies that $|\mathbb{E}[M(A,X)]| \leq 2^{-r}$.

As an example, the results in [KK12] imply that the inner product function on n bits, where $\mathcal{A} = \mathcal{X} = \mathbb{F}_2^n$ and

$$M(a,x) = (-1)^{a \cdot x},$$

is an X-strong $(k, n-k)$-quantum extractor for every $2 \leq k \leq n$.

In this section we prove the following theorem, which provides a linear lower bound on quantum memory when applied to :

Theorem 10.2.1. *Let $\mathcal{X}, \mathcal{A}$ be two finite sets with $n = \log_2 |\mathcal{X}|$. Let $M : \mathcal{A} \times \mathcal{X} \to \{-1,1\}$ be a matrix which is a X-strong (q,r)-quantum extractor. Let ρ be a branching program for the learning problem corresponding to M, described by classical-quantum systems $\rho_{XV}^{(t)}$, with $q/2$-qubit quantum memory V and length T, and without classical memory. Then the success*

probability of ρ is at most

$$2^{-n} + 8T\sqrt{n+q} \cdot 2^{-r/4}.$$

The proof of Theorem 10.2.1 is significantly simpler than the proof of our main theorem for hybrid memory, which will be presented in the next section. We first need to define the following measure of dependency:

Definition 10.2.2. *Let ρ_{XV} be a classical-quantum system over classical X and quantum V. The dependency of V on X in ρ_{XV} is defined as*

$$\xi(X;V) = \min_{\tau_V} \left\|\rho_{XV} - \rho_X \otimes \tau_V\right\|_{\mathrm{Tr}}$$

where τ_V is taken over all density operators on V. Notice that in this definition taking $\tau_V = \rho_V$ is almost optimal as we have

$$\left\|\rho_{XV} - \rho_X \otimes \rho_V\right\|_{\mathrm{Tr}} \leq \left\|\rho_{XV} - \rho_X \otimes \tau_V\right\|_{\mathrm{Tr}} + \left\|\rho_V - \tau_V\right\|_{\mathrm{Tr}} \leq 2\left\|\rho_{XV} - \rho_X \otimes \tau_V\right\|_{\mathrm{Tr}}. \quad (10.2)$$

When V consists of q qubits, we have the following relationship between our dependency measure and quantum mutual information:

Lemma 10.2.3. $\frac{1}{2}\xi(X;V)^2 \leq \mathbf{I}_\rho(X;V) \leq q \cdot \xi(X;V) + 2\sqrt{\xi(X;V)}.$

Proof. On one hand, using the inequality on quantum relative entropy and trace distance (see e.g. [OP04, Theorem 1.15]), we have

$$\mathbf{I}_\rho(X;V) = \mathbf{S}\left(\rho_{XV} \,\|\, \rho_X \otimes \rho_V\right) \geq \frac{1}{2}\left\|\rho_{XV} - \rho_X \otimes \rho_V\right\|_{\mathrm{Tr}}^2 \geq \frac{1}{2}\xi(X;V)^2.$$

On the other hand, Fannes-Audenaert inequality [Aud07] tells us that for every $x \in \mathcal{X}$, the difference between the von-Neumann entropies of any two states ρ and τ on V is bounded by

$$|\mathbf{S}(\rho) - \mathbf{S}(\tau)| \leq q \cdot \frac{1}{2}\left\|\rho - \tau\right\|_{\mathrm{Tr}} + h\left(\frac{1}{2}\left\|\rho - \tau\right\|_{\mathrm{Tr}}\right)$$

where $h(\varepsilon) = -\varepsilon \log_2 \varepsilon - (1-\varepsilon)\log_2(1-\varepsilon)$ is the binary entropy function. Since the state of V conditioned on $X = x$ is $\rho_{V|x}/\Pr[X = x] = 2^n \rho_{V|x}$, we have

$$\begin{aligned}
\mathbf{I}_\rho(X; V) &= \mathop{\mathbb{E}}_{x\sim X}\left[\mathbf{S}(\rho_V) - \mathbf{S}(2^n\rho_{V|x})\right] \\
&\leq \frac{1}{2}q \cdot \mathop{\mathbb{E}}_{x\sim X}\left\|\rho_V - 2^n\rho_{V|x}\right\|_{\mathrm{Tr}} + \mathop{\mathbb{E}}_{x\sim X} h\left(\frac{1}{2}\left\|\rho_V - 2^n\rho_{V|x}\right\|_{\mathrm{Tr}}\right) \\
&\leq \frac{1}{2}q \cdot \left\|\rho_{XV} - \rho_X \otimes \rho_V\right\|_{\mathrm{Tr}} + h\left(\frac{1}{2}\left\|\rho_{XV} - \rho_X \otimes \rho_V\right\|_{\mathrm{Tr}}\right) \\
&\leq \frac{1}{2}q \cdot \left\|\rho_{XV} - \rho_X \otimes \rho_V\right\|_{\mathrm{Tr}} + \sqrt{2\left\|\rho_{XV} - \rho_X \otimes \rho_V\right\|_{\mathrm{Tr}}},
\end{aligned}$$

as h is concave and $h(\varepsilon) \leq 2\sqrt{\varepsilon}$. Now let τ_V be the optimal density operator in the definition of $\mathcal{E}(X; V)$. Plugging in (10.2), we conclude that

$$\mathbf{I}_\rho(X; V) \leq q \cdot \mathcal{E}(X; V) + 2\sqrt{\mathcal{E}(X; V)}. \qquad \square$$

Lemma 10.2.4. *For every classical-quantum system ρ_{AXV} with the q-qubit quantum subsystem V that depends only on A, we have*

$$\mathbf{I}_\rho(X; M(A, X), V) \leq 2(n+q) \cdot 2^{-r/2}.$$

Proof. Since $\mathbf{I}_\rho(X; V) = 0$, it suffices to bound $\mathbf{I}_\rho(X; M(A,X) \mid V) \leq \mathbf{I}_\rho(M(A,X); X, V)$.

To bound the later, we first notice that since M is a strong (q, r)-quantum extractor,

$$\begin{aligned}\mathscr{E}(M(A,X);X,V) &\leq \left\|\rho_{M(A,X)XV} - \rho_{M(A,X)} \otimes \rho_X \otimes \rho_V\right\|_{\mathrm{Tr}} \\ &\leq \left\|\rho_{M(A,X)XV} - U \otimes \rho_X \otimes \rho_V\right\|_{\mathrm{Tr}} + |\mathbb{E}[M(A,X)]| \\ &\leq 2 \cdot 2^{-r}.\end{aligned}$$

As the total dimension of X and V is 2^{n+q}, by Lemma 10.2.3 we have

$$\begin{aligned}\mathbf{I}_\rho(X; M(A,X), V) &\leq \mathbf{I}_\rho(M(A,X); X, V) \\ &\leq (n+q) \cdot \mathscr{E}(M(A,X);X,V) + 2\sqrt{\mathscr{E}(M(A,X);X,V)} \\ &\leq 5(n+q) \cdot 2^{-r/2}.\end{aligned}$$

□

Lemma 10.2.5. *For every classical-quantum system ρ_{AXV} with $q/2$-qubit quantum subsystem V that depends only on A and $M(A,X)$, we have*

$$\mathscr{E}(X;V) \leq 4\sqrt{n+q} \cdot 2^{-r/4}.$$

Proof. Let $W = \rho_{a,0} \otimes \rho_{a,1}$, where $\rho_{a,b}$ is the density matrix of V when $A = a$ and $M(A,X) = b$. Then W is a q-bit quantum system that depends only on A. Since V can be decided from $M(A,X)$ and W, we have

$$\mathscr{E}(X;V)^2 \leq 2\mathbf{I}_\rho(X;V) \leq 2\mathbf{I}_\rho(X; M(A,X)), W) \leq 10(n+q) \cdot 2^{-r/2}.$$

□

We are now ready to prove Theorem 10.2.1. Let $\Phi_{t,a,b}$ be the quantum channel applied

on V at stage t with sample (a, b), and recall that the evolution of the system $\rho_{XV}^{(t)}$ can be expressed as

$$\rho_{XV}^{(t+1)} = \mathop{\mathbb{E}}_{a\sim A}\left[\sum_{x\in\mathcal{X}} |x\rangle\langle x| \otimes \Phi_{t,a,M(a,x)}\left(\rho_{V|x}^{(t)}\right)\right].$$

Proof of Theorem 10.2.1. We are going to bound the increment of ξ_t, which is the shorthand for $\mathcal{E}^{(t)}(X;V)$. For now let us focus on some stage t, and let τ be the density operator that minimizes $\xi_t = \left\|\rho_{XV}^{(t)} - \rho_X^{(t)} \otimes \tau\right\|_{\mathrm{Tr}}$. Notice that $\rho_X^{(t)} = \rho_X = 2^{-n}\mathbb{I}_X$ for every t.

Since τ is a fixed quantum state, we can prepare τ and apply $\Phi_{A,M(A,X)}$ on τ to obtain a new quantum register V', which depends only on A and $M(A,X)$. Notice that

$$\rho_{XV'} = \mathop{\mathbb{E}}_{a\sim A}\left[\sum_{x\in\mathcal{X}} |x\rangle\langle x| \otimes \Phi_{t,a,M(a,x)}(\tau)\right],$$

and therefore by contractivity of quantum channels under trace norms, we can show that

$$\begin{aligned}\left\|\rho_{XV}^{(t+1)} - \rho_{XV'}\right\|_{\mathrm{Tr}} &\le \mathop{\mathbb{E}}_{a\sim A}\sum_{x\in\mathcal{X}}\left\|\Phi_{t,a,M(a,x)}\left(\rho_{V|x}^{(t)}\right) - \Phi_{t,a,M(a,x)}(\tau)\right\|_{\mathrm{Tr}} \\ &\le \sum_{x\in\mathcal{X}}\left\|\rho_{V|x}^{(t)} - \tau\right\|_{\mathrm{Tr}} \le \left\|\rho_{XV}^{(t)} - \rho_X \otimes \tau\right\|_{\mathrm{Tr}} = \xi_t.\end{aligned}$$

Hence we have

$$\begin{aligned}\xi_{t+1} &\le \left\|\rho_{XV}^{(t+1)} - \rho_X \otimes \rho_{V'}\right\|_{\mathrm{Tr}} \\ &\le \left\|\rho_{XV}^{(t+1)} - \rho_{XV'}\right\|_{\mathrm{Tr}} + \left\|\rho_{XV'} - \rho_X \otimes \rho_{V'}\right\|_{\mathrm{Tr}} \\ &\le \xi_t + 2\mathcal{E}(X;V') \\ &\le \xi_t + 8\sqrt{n+q}\cdot 2^{-r/4}.\end{aligned}$$

Since $\xi_0 = 0$, we conclude that

$$\xi_T \leq 8T\sqrt{n+q} \cdot 2^{-r/4}.$$

This value bounds the difference of the success probability of ρ, and that of a quantum branching program whose memory is independent of X. The later is clearly at most 2^{-n}, which finishes the proof. □

10.3 Truncation of Classical-Quantum Branching Programs

Now we start to prove our actual lower bound, which states as follows for any branching program using both classical and quantum memory. Since the inner product function on n bits is a $(\Omega(n), \Omega(n))$-extractor with error $2^{-\Omega(n)}$ [GRT18], the theorem implies either $\Omega(n^2)$ classical space or $\Omega(n)$ quantum space is necessary for sub-exponential-time parity learning.

Theorem 10.3.1. *Let $\mathcal{X}, \mathcal{A}$ be two finite sets with $n = \log_2 |\mathcal{X}|$. Let $M : \mathcal{A} \times \mathcal{X} \to \{-1, 1\}$ be a matrix which is a (k', ℓ')-L_2 extractor with error $2^{-r'}$ for sufficiently large k', ℓ' and r', where $\ell' \leq n$. Let*

$$r = \min\left\{\frac{1}{4}r', \frac{1}{26}\ell' + \frac{1}{6}, \frac{1}{2}(k'-1)\right\}.$$

Let ρ be a branching program for the learning problem corresponding to M, described by classical-quantum systems $\rho^{(t)}_{XVW}$, with q-qubit quantum memory V, m-bit classical memory W and length T. If $m \leq \frac{1}{44}(k'-1)\ell'$, $q \leq r - 7$ and $T \leq 2^{r-2}$, the success probability of ρ is at most $O(2^{q-r})$.

From now on we let $k = k' - 1$ and $\ell = \frac{1}{5}(\ell' - 13r - 2)$. Then we have the following inequalities to be used later:

$$q + r + 1 - r' \leq -2r. \tag{10.3}$$

$$2\ell + 9r - n \leq -r. \tag{10.4}$$

$$(k - r)\ell \geq 2m + 4r + 1. \tag{10.5}$$

Like the proofs in [Raz17, GRT18], our proof heavily depends on the notion of truncating the branching program, which we will explain below.

10.3.1 Truncated Classical-Quantum Systems

Here we describe how to truncate a partial classical-quantum system ρ_{XVW} according to some property $G(v, w)$ of desire on $\rho_{X|v,w}$. The goal is to remove the parts of ρ_{XVW} where G is not satisfied. We execute the following procedure:

1. Maintain a partial system ρ'_{XVW} initialized as ρ_{XVW}, and subspaces $\mathcal{V}_w \subseteq \mathcal{V}$ initialized as $\mathcal{V}$ for each $w \in \mathcal{W}$.

2. Pick $w \in \mathcal{W}$ and $|v\rangle \in \mathcal{V}_w$ such that $\mathrm{Tr}[\rho'_{X|v,w}] > 0$ and $G(v, w)$ is false.

3. Change the partial system ρ'_{XVW} into the following system by projection:

$$\big(\mathbb{I}_X \otimes (\mathbb{I}_{VW} - |v, w\rangle\langle v, w|)\big)\rho'_{XVW}\big(\mathbb{I}_X \otimes (\mathbb{I}_{VW} - |v, w\rangle\langle v, w|)\big),$$

and change $\mathcal{V}_w$ to its subspace orthogonal to $|v\rangle$, that is

$$\{|v'\rangle \in \mathcal{V}_w \mid \langle v|v'\rangle = 0\}.$$

4. Repeat from step 2 until there is no such w and $|v\rangle$. Denote the final system as $\rho^{|G}_{XVW}$.

In step 2 we pick w and $|v\rangle$ arbitrarily as long as it satisfies the requirements, however we could always think of it as iterating over $w \in \mathcal{W}$ and processing each $\rho_{XV|w}$ separately. The choices of $|v\rangle$ for each w do affect the final system $\rho^{|G}_{XVW}$; Yet as we will see later, these choices are irrelevant to our proof.

Below, we give two useful lemmas on truncated systems.

Lemma 10.3.2. *For every $|v\rangle \in \mathcal{V}$ and $w \in \mathcal{W}$ such that $\mathrm{Tr}[\rho^{|G}_{X|v,w}] > 0$, there exists $|v'\rangle$ in the remaining subspace $\mathcal{V}_w$ such that*

$$P^{\rho^{|G}}_{X|v,w} = P^{\rho}_{X|v',w} = P^{\rho^{|G}}_{X|v',w}.$$

Proof. It suffices to prove the lemma with one round of the truncation procedure executed. Suppose the $|v_1, w_1\rangle$ is picked in step 2, resulting in the partial system

$$\rho'_{XVW} = \big(\mathbb{I}_X \otimes (\mathbb{I}_{VW} - |v_1, w_1\rangle\langle v_1, w_1|)\big)\rho_{XVW}\big(\mathbb{I}_X \otimes (\mathbb{I}_{VW} - |v_1, w_1\rangle\langle v_1, w_1|)\big).$$

We can write

$$\begin{aligned}\rho'_{X|v,w} &= \big(\mathbb{I}_X \otimes \langle v, w|\big)\rho'_{XVW}\big(\mathbb{I}_X \otimes |v, w\rangle\big) \\ &= \big(\mathbb{I}_X \otimes (\langle v, w| - \langle v, w|v_1, w_1\rangle\langle v_1, w_1|)\big)\rho_{XVW}\big(\mathbb{I}_X \otimes (|v, w\rangle - |v_1, w_1\rangle\langle v_1, w_1|v, w\rangle)\big).\end{aligned}$$

- If $w \neq w_1$, then

$$\rho'_{X|v,w} = \big(\mathbb{I}_X \otimes \langle v, w|\big)\rho_{XVW}\big(\mathbb{I}_X \otimes |v, w\rangle\big) = \rho_{X|v,w}.$$

  And the lemma holds directly by choosing $|v'\rangle = |v\rangle$.

- If $w = w_1$, then with $\langle v_1, w_1|v, w\rangle = \langle v_1|v\rangle = \lambda$, we have

$$\rho'_{X|v,w} = \big(\mathbb{I}_X \otimes (\langle v| - \overline{\lambda}\langle v_1|)\langle w|\big)\rho_{XVW}\big(\mathbb{I}_X \otimes (|v\rangle - \lambda|v_1\rangle)|w\rangle\big).$$

  By the fact that $\mathrm{Tr}[\rho^{|G}_{X|v,w}] > 0$, we must have $|v\rangle \neq |v_1\rangle$. Therefore if we let $|v'\rangle \sim |v\rangle - \lambda|v_1\rangle$, which is the normalized projection of $|v\rangle$ onto the orthogonal subspace of $|v_1\rangle$, the above equality implies that $P^{\rho'}_{X|v,w} = P^{\rho}_{X|v',w}$. Meanwhile, since $\langle v_1|v'\rangle = 0$ we have $\rho'_{X|v',w} = \rho_{X|v',w}$, which completes the proof. □

A direct corollary of the above lemma is that if $G(v, w)$ only depends on the distribution $P^{\rho}_{X|v,w}$, then $G(v, w)$ holds for every $|v\rangle \in \mathcal{V}$ and $w \in \mathcal{W}$ in the truncated system $\rho^{|G}_{XVW}$, even when $|v\rangle$ is not in the remaining subspace $\mathcal{V}_w$.

Lemma 10.3.3. *For each $w \in W$, let $|v_1\rangle, \ldots, |v_d\rangle$ be the states picked in step 2 within $\mathcal{V}_w$.*

Then

$$\left\|\rho_{XV|w} - \rho^{|G}_{XV|w}\right\|_{\mathrm{Tr}} \leq 3\sum_{i=1}^{d}\sqrt{\mathrm{Tr}[\rho_{X|v_i,w}]\mathrm{Tr}[\rho_{XV|w}]}.$$

Proof. In Corollary 2.3.9, take ρ to be $\rho_{XV|w}$, and Π to be

$$\mathbb{I}_X \otimes \prod_{i=1}^{d}(\mathbb{I}_V - |v_i\rangle\langle v_i|) = \mathbb{I}_X \otimes \left(\mathbb{I}_V - \sum_{i=1}^{d}|v_i\rangle\langle v_i|\right).$$

Then $\Pi\rho\Pi = \rho^{|G}_{XV|w}$ and $\mathrm{Tr}[\Pi\rho] = \mathrm{Tr}[\rho_{XV|w}] - \sum_{i=1}^{d}\mathrm{Tr}[\rho_{X|v_i,w}]$. Therefore we have

$$\begin{aligned}
\left\|\rho_{XV|w} - \rho^{|G}_{XV|w}\right\|_{\mathrm{Tr}} &\leq \sqrt{4\mathrm{Tr}[\rho]^2 - 4\mathrm{Tr}[\Pi\rho]^2} \\
&\leq \sqrt{8(\mathrm{Tr}[\rho] - \mathrm{Tr}[\Pi\rho])\mathrm{Tr}[\rho]} \\
&= \sqrt{8\sum_{i=1}^{d}\mathrm{Tr}[\rho_{X|v_i,w}]\mathrm{Tr}[\rho_{XV|w}]} \\
&\leq 3\sum_{i=1}^{d}\sqrt{\mathrm{Tr}[\rho_{X|v_i,w}]\mathrm{Tr}[\rho_{XV|w}]}.
\end{aligned}$$

□

Since $\mathrm{Tr}[\rho_{XV|w}] \leq 1$ always holds, by summing over all $w \in \mathcal{W}$ we get the following corollary:

Corollary 10.3.4. *Let $|v_1, w_1\rangle, \ldots, |v_d, w_d\rangle$ be all of the memory states picked in step 2. Then*

$$\left\|\rho_{XVW} - \rho^{|G}_{XVW}\right\|_{\mathrm{Tr}} \leq 3\sum_{i=1}^{d}\sqrt{\mathrm{Tr}[\rho_{X|v_i,w_i}]}.$$

10.3.2 Truncated Branching Program

The properties that we desire for the partial system ρ_{XVW} consist of three parts:

- Small L_2 norm: Let $G_2(v, w)$ be the property that

$$\left\|P^{\varnothing}_{X|v,w}\right\|_2 \leq 2^{\ell} \cdot 2^{-n/2}.$$

- Small L_∞ norm: Let $G_\infty(v, w)$ be the property that

$$\left\|P^{\varnothing}_{X|v,w}\right\|_\infty \leq 2^{2\ell+9r} \cdot 2^{-n}.$$

- Even division: For every $a \in \mathcal{A}$, let $G_a(v, w)$ be the property that

$$|\langle M_a, P^{\varnothing}_{X|v,w}\rangle| \leq 2^{-r}.$$

Now we define the truncated branching program, by specifying the truncated partial classical-quantum system $\tau^{(t)}_{XVW}$ for each stage t. Initially let $\tau^{(0)}_{XVW} = \rho^{(0)}_{XVW}$. For each stage $0 \leq t \leq T$, the truncation consists of three ingredients (below we ignore the superscripts on P for convenience):

1. Remove parts where $\left\|P_{X|v,w}\right\|_2$ is large. That is, let $\tau^{(t,\star)}_{XVW} = \tau^{(t)|G_2}_{XVW}$.

2. Remove parts where $\left\|P_{X|v,w}\right\|_\infty$ is large. This is done by two steps.

   - First, let $g \in \{0,1\}^{\mathcal{X}\otimes\mathcal{W}}$ be an indicator vector such that $g(x, w) = 1$ if and only if

   $$\mathrm{Tr}[\tau^{(t,\star)}_{X|w}] > 0 \text{ and } P^{\tau^{(t,\star)}}_{X|w}(x) \leq 2^{2\ell+5r} \cdot 2^{-n}.$$

   Let $\tau^{(t,\circ)}_{XVW} = (gg^\dagger \otimes \mathbb{I}_V)\tau^{(t,\star)}_{XVW}(gg^\dagger \otimes \mathbb{I}_V)$, where $gg^\dagger$ is the projection operator

acting on $\mathcal{X} \otimes \mathcal{W}$.

- To make sure that the distributions did not change a lot after the projection $gg^\dagger$, for each $0 \leq t < T$, let $G_t(v, w)$ be the property that

$$\mathrm{Tr}[\tau^{(t,\circ)}_{X|v,w}] \geq (1 - 2^{-r})\mathrm{Tr}[\tau^{(t,\star)}_{X|v,w}].$$

Let $\tau^{(t,\infty)}_{XVW} = \tau^{(t,\circ)|G_\infty \wedge G_t}_{XVW}$.

3. For each $a \in \mathcal{A}$, remove (only for this a) parts where $P_{X|v,w}$ is not evenly divided by a. That is, for each $a \in \mathcal{A}$, let $\tau^{(t,a)}_{XVW} = \tau^{(t,\infty)|G_a}_{XVW}$.

Then, if $t < T$, for each $a \in_R \mathcal{A}$ we evolve the system by applying the sample operations $\Phi_{t,a,b}$ as the original branching program on $\tau^{(t,a)}_{XVW}$, so that we have

$$\tau^{(t+1)}_{XVW} = \mathop{\mathbb{E}}_{a \in_R \mathcal{A}} \left[\sum_{x \in \mathcal{X}} |x\rangle\langle x| \otimes \Phi_{t,a,M(a,x)} \left(\tau^{(t,a)}_{VW|x} \right) \right].$$

10.3.3 Bounding the Truncation Difference

In order to show that the success probability of the original branching program $\rho^{(t)}$ is low, the plan is to prove an upper bound on the success probability of the truncated branching program $\tau^{(t)}$, and bound the difference between the two probabilities.

Here we bound the difference by the trace distance between the two systems $\rho^{(t)}_{XVW}$ and $\tau^{(t)}_{XVW}$. We will show that the contribution to the trace distance from each one of the truncation ingredients is small, and in addition the evolution preserves the trace distance.

Truncation by G_2

Lemma 10.3.5. *For every $0 \le t \le T$, $|v\rangle \in \mathcal{V}$ and $w \in \mathcal{W}$ such that $G_2(v,w)$ is violated (that is, $\left\|P^{\tau^{(t)}}_{X|v,w}\right\|_2 > 2^{\ell} \cdot 2^{-n/2}$), we must have $\mathrm{Tr}[\tau^{(t)}_{X|v,w}] < 2^{-2m} \cdot 2^{-4r}$.*

The lemma says, for any direction $|v,w\rangle$ picked by the truncation procedure, the weight will be small and the truncation will not change the state significantly.

Proof. This is our main technical lemma and we defer the proof to Section 10.4. □

Since there are at most 2^{q+m} such directions picked in the truncation procedure, we conclude the following corollary.

Corollary 10.3.6. *For every $0 \le t \le T$, we have $\left\|\tau^{(t,\star)}_{XVW} - \tau^{(t)}_{XVW}\right\|_{\mathrm{Tr}} \le 3 \cdot 2^{q-2r}$.*

Proof. Recall that $\tau^{(t,\star)}_{XVW} = \tau^{(t)|G_2}_{XVW}$. Since $\dim(\mathcal{V} \otimes \mathcal{W}) = 2^{q+m}$, the truncation lasts for at most 2^{q+m} rounds. Since in every round the picked $|v,w\rangle$ has $\mathrm{Tr}[\tau^{(t)}_{X|v,w}] < 2^{-2m} \cdot 2^{-4r}$, by Corollary 10.3.4 we have

$$\left\|\tau^{(t,\star)}_{XVW} - \tau^{(t)}_{XVW}\right\|_{\mathrm{Tr}} \le 3 \cdot 2^{q+m} \cdot \sqrt{2^{-2m} \cdot 2^{-4r}} = 3 \cdot 2^{q-2r}.$$ □

Truncation by G_∞

Lemma 10.3.7. *For every $0 \le t \le T$ and $w \in \mathcal{W}$ we have*

$$\sum_{\substack{x \in \mathcal{X} \\ g(x,w)=0}} P^{\tau^{(t,\star)}}_{X|w}(x) \le 2^{-5r}.$$

Proof. By Claim 10.1.1, $P^{\tau^{(t,\star)}}_{X|w}$ is a convex combination of $P^{\tau^{(t,\star)}}_{X|v,w}$. From Lemma 10.3.2 we know that $G_2(P^{\tau^{(t,\star)}}_{X|v,w})$ holds for every $|v\rangle$ and w, and thus by convexity of ℓ_2-norms we know that $G_2(P^{\tau^{(t,\star)}}_{X|w})$ also holds. That means

$$\mathop{\mathbb{E}}_{x\sim P^{\tau^{(t,\star)}}_{X|w}}\left[P^{\tau^{(t,\star)}}_{X|w}(x)\right] = \left\|P^{\tau^{(t,\star)}}_{X|w}\right\|_2^2 \le 2^{2\ell}\cdot 2^{-n}.$$

Therefore, by Markov's inequality we have

$$\sum_{\substack{x\in\mathcal{X}\\ g(x,w)=0}} P^{\tau^{(t,\star)}}_{X|w}(x) = \Pr_{x\sim P^{\tau^{(t,\star)}}_{X|w}}\left[P^{\tau^{(t,\star)}}_{X|w}(x) > 2^{2\ell+5r}\cdot 2^{-n}\right] \le 2^{-5r}. \qquad \square$$

Corollary 10.3.8. *For every $0 \le t \le T$ and every $w \in \mathcal{W}$, we have $\tau^{(t,\circ)}_{XV|w} \le \tau^{(t,\star)}_{XV|w}$, and*

$$\mathrm{Tr}[\tau^{(t,\circ)}_{XV|w}] \ge (1-2^{-5r})\cdot \mathrm{Tr}[\tau^{(t,\star)}_{XV|w}].$$

Moreover, we have $\left\|\tau^{(t,\circ)}_{XVW} - \tau^{(t,\star)}_{XVW}\right\|_{\mathrm{Tr}} \le 2^{-5r}$.

Proof. Since X and W are both classical and $\tau^{(t,\circ)}_{XVW} = (gg^\dagger\otimes\mathbb{I}_V)\tau^{(t,\star)}_{XVW}(gg^\dagger\otimes\mathbb{I}_V)$, we have

$$\tau^{(t,\star)}_{XV|w} - \tau^{(t,\circ)}_{XV|w} = \sum_{\substack{x\in\mathcal{X}\\ g(x,w)=0}} |x\rangle\langle x|\otimes\tau^{(t,\star)}_{V|x,w},$$

which is positive semi-definite. Recalling (10.1) that

$$\mathrm{Tr}[\tau^{(t,\star)}_{V|x,w}] = \langle x,w|\tau^{(t,\star)}_{XW}|x,w\rangle = \mathrm{diag}\,\tau^{(t,\star)}_{X|w}(x) = P^{\tau^{(t,\star)}}_{X|w}(x)\mathrm{Tr}[\tau^{(t,\star)}_{X|w}],$$

we have

$$\mathrm{Tr}[\tau^{(t,\star)}_{XV|w}] - \mathrm{Tr}[\tau^{(t,\circ)}_{XV|w}] = \sum_{\substack{x\in\mathcal{X}\\ g(x,w)=0}} \mathrm{Tr}[\tau^{(t,\star)}_{V|x,w}] = \sum_{\substack{x\in\mathcal{X}\\ g(x,w)=0}} P^{\tau^{(t,\star)}}_{X|w}(x)\cdot \mathrm{Tr}[\tau^{(t,\star)}_{X|w}] \leq 2^{-5r}\cdot \mathrm{Tr}[\tau^{(t,\star)}_{X|w}].$$

And therefore, as $\tau^{(t,\circ)}_{XVW} - \tau^{(t,\star)}_{XVW}$ is positive semi-definite, we have

$$\left\|\tau^{(t,\circ)}_{XVW} - \tau^{(t,\star)}_{XVW}\right\|_{\mathrm{Tr}} = \sum_{w\in\mathcal{W}} \mathrm{Tr}[\tau^{(t,\star)}_{XV|w}] - \mathrm{Tr}[\tau^{(t,\circ)}_{XV|w}] \leq 2^{-5r}\sum_{w\in\mathcal{W}} \mathrm{Tr}[\tau^{(t,\star)}_{X|w}] \leq 2^{-5r}. \qquad \square$$

Lemma 10.3.9. *For every $0 \leq t \leq T$, $|v\rangle \in \mathcal{V}$ and $w \in \mathcal{W}$ such that $G_\infty(v,w)$ is violated (that is, $\left\|P^{\tau^{(t,\circ)}}_{X|v,w}\right\|_\infty > 2^{2\ell+9r}\cdot 2^{-n}$) or $G_t(v,w)$ is violated (that is, $\mathrm{Tr}[\tau^{(t,\circ)}_{X|v,w}] < (1-2^{-r})\mathrm{Tr}[\tau^{(t,\star)}_{X|v,w}]$), we must have $\mathrm{Tr}[\tau^{(t,\circ)}_{X|v,w}] < 2\cdot 2^{-4r}\cdot \mathrm{Tr}[\tau^{(t,\circ)}_{X|w}]$.*

Proof. If $G_\infty(v,w)$ is violated, let $x \in \mathcal{X}$ be the one such that $P^{\tau^{(t,\circ)}}_{X|v,w}(x) > 2^{2\ell+9r}\cdot 2^{-n}$. If $g(x,w) = 0$ then $P^{\tau^{(t,\circ)}}_{X|w}(x) = 0$, while if $g(x,w) = 1$ then by Corollary 10.3.8,

$$P^{\tau^{(t,\circ)}}_{X|w}(x) \leq \frac{\mathrm{Tr}[\tau^{(t,\star)}_{X|w}]}{\mathrm{Tr}[\tau^{(t,\circ)}_{X|w}]}\cdot 2^{2\ell+5r}\cdot 2^{-n} \leq (1-2^{-5r})^{-1}\cdot 2^{2\ell+5r}\cdot 2^{-n}.$$

Hence we always have

$$\mathrm{Tr}[\tau^{(t,\circ)}_{X|v,w}] \leq \frac{P^{\tau^{(t,\circ)}}_{X|w}(x)}{P^{\tau^{(t,\circ)}}_{X|v,w}(x)}\cdot \mathrm{Tr}[\tau^{(t,\circ)}_{X|w}] \leq 2\cdot 2^{-4r}\cdot \mathrm{Tr}[\tau^{(t,\circ)}_{X|w}],$$

where the first inequality comes from the fact that $\tau^{(t,\circ)}_{X|w} \geq \tau^{(t,\circ)}_{X|v,w}$ and (10.1).

If $G_t(v,w)$ is violated, since we know from Corollary 10.3.8 that

$$\begin{aligned}\left|\mathrm{Tr}[\tau^{(t,\circ)}_{X|v,w}] - \mathrm{Tr}[\tau^{(t,\star)}_{X|v,w}]\right| &\le \left\|\tau^{(t,\circ)}_{XV|w} - \tau^{(t,\star)}_{XV|w}\right\|_{\mathrm{Tr}} \le 2^{-5r}\cdot \mathrm{Tr}[\tau^{(t,\star)}_{XV|w}] \\ &\le 2^{-5r}\cdot(1-2^{-5r})^{-1}\cdot \mathrm{Tr}[\tau^{(t,\circ)}_{XV|w}],\end{aligned}$$

therefore from $\mathrm{Tr}[\tau^{(t,\circ)}_{X|v,w}] < (1-2^{-r})\mathrm{Tr}[\tau^{(t,\star)}_{X|v,w}]$ we deduce that

$$\begin{aligned}\mathrm{Tr}[\tau^{(t,\circ)}_{X|v,w}] &< (2^r-1)\cdot\left(\mathrm{Tr}[\tau^{(t,\star)}_{X|v,w}] - \mathrm{Tr}[\tau^{(t,\circ)}_{X|v,w}]\right) \\ &\le (2^r-1)\cdot 2^{-5r}\cdot(1-2^{-5r})^{-1}\cdot\mathrm{Tr}[\tau^{(t,\circ)}_{XV|w}] \\ &< 2\cdot 2^{-4r}\cdot\mathrm{Tr}[\tau^{(t,\circ)}_{X|w}].\end{aligned}$$

□

Corollary 10.3.10. *For every $0 \le t \le T$, we have $\left\|\tau^{(t,\infty)}_{XVW} - \tau^{(t,\circ)}_{XVW}\right\|_{\mathrm{Tr}} \le 5\cdot 2^{q-2r}$.*

Proof. Recall that $\tau^{(t,\infty)}_{XVW} = \tau^{(t,\circ)|G_\infty\wedge G_t}_{XVW}$. For each $w \in \mathcal{W}$, the truncation picks at most $\dim \mathcal{V} = 2^q$ states $|v,w\rangle$, each with $\mathrm{Tr}[\tau^{(t,\circ)}_{X|v,w}] < 2\cdot 2^{-4r}\cdot\mathrm{Tr}[\tau^{(t,\circ)}_{X|w}]$. Therefore by applying Lemma 10.3.3 for each $w \in \mathcal{W}$, we have

$$\left\|\tau^{(t,\infty)}_{XVW} - \tau^{(t,\circ)}_{XVW}\right\|_{\mathrm{Tr}} \le 3\cdot\sum_{w\in\mathcal{W}} 2^q\cdot\sqrt{2\cdot 2^{-4r}}\cdot\mathrm{Tr}[\tau^{(t,\circ)}_{X|w}] \le 5\cdot 2^{q-2r}.$$

□

Truncation by G_a

Notice that in the truncation step from $\tau^{(t,\star)}$ to $\tau^{(t,\circ)}$, the distribution $P^{\tau^{(t,\star)}}_{X|v,w}$ might change and not satisfy G_2 anymore. However, with the truncation by G_t, any such distribution that changes too much is eliminated, and we have the following guarantee.

Lemma 10.3.11. *For every $0 \le t \le T$, $|v\rangle \in \mathcal{V}$ and $w \in \mathcal{W}$, we have*

$$\left\| P^{\tau,(t,\infty)}_{X|v,w} \right\|_2 \le (1-2^{-r})^{-1} \cdot 2^{\ell} \cdot 2^{-n/2}.$$

Proof. By Lemma 10.3.2, there exists $|v'\rangle \in \mathcal{V}$ such that $P^{\tau,(t,\infty)}_{X|v,w} = P^{\tau,(t,\infty)}_{X|v',w} = P^{\tau,(t,\circ)}_{X|v',w}$. The truncation by G_t ensures that $\mathrm{Tr}[\tau^{(t,\circ)}_{X|v',w}] \ge (1-2^{-r})\mathrm{Tr}[\tau^{(t,\star)}_{X|v',w}]$, and therefore

$$\left\| P^{\tau,(t,\infty)}_{X|v,w} \right\|_2 = \left\| P^{\tau,(t,\circ)}_{X|v',w} \right\|_2 = \frac{\left\| \operatorname{diag} \tau^{(t,\circ)}_{X|v',w} \right\|_2}{\mathrm{Tr}[\tau^{(t,\circ)}_{X|v',w}]} \le \frac{\left\| \operatorname{diag} \tau^{(t,\star)}_{X|v',w} \right\|_2}{(1-2^{-r})\mathrm{Tr}[\tau^{(t,\star)}_{X|v',w}]} \le (1-2^{-r})^{-1} \cdot 2^{\ell} \cdot 2^{-n/2}.$$

□

Lemma 10.3.12. *For every partial classical-quantum system τ_{XV} over $\mathcal{X} \otimes \mathcal{V}$ such that $\left\| P^{\tau}_{X|v} \right\|_2 \le 2^{\ell'} \cdot 2^{-n/2}$ holds for every $|v\rangle \in \mathcal{V}$, we have*

$$\Pr_{a \in_R \mathcal{A}} \left[\exists |v\rangle \in \mathcal{V}, |\langle M_a, P^{\tau}_{X|v} \rangle| \ge 2^{-r} \right] \le 2^{-2r}.$$

Proof. Notice that we can think of $\tau_V = \mathrm{Tr}_X[\tau_{XV}]$ to be $\mathbb{I}_V$. This is because we can first assume that τ_V is full rank (otherwise change $\mathcal{V}$ to its subspace and the conclusion in this lemma still holds), and if we have diagonalization $Q^\dagger \tau_V Q = \mathbb{I}_V$ for some non-singular Q, then consider the new system

$$\tau'_{XV} = (\mathbb{I}_X \otimes Q^\dagger)\tau_{XV}(\mathbb{I}_X \otimes Q),$$

and the set of distributions $\{P^{\tau}_{X|v}\}$ and $\{P^{\tau'}_{X|v}\}$ over $|v\rangle \in \mathcal{V}$ are the same, since $P^{\tau'}_{X|v} = P^{\tau}_{X|v'}$ for $|v'\rangle \sim Q|v\rangle$. With $\tau_V = \mathbb{I}_V$ we have $\mathrm{Tr}[\tau_{X|v}] = 1$ for every $|v\rangle \in \mathcal{V}$, and thus $P^{\tau}_{X|v} = \operatorname{diag} \tau_{X|v}$.

Let $\mathcal{A}' \subseteq \mathcal{A}$ be the set of $a \in \mathcal{A}$ such that there exists $|v\rangle \in \mathcal{V}$ with $|\langle M_a, P^{\tau}_{X|v}\rangle| \geq 2^{-r}$. For each $a \in \mathcal{A}'$, let

$$\sigma_a = \mathrm{Tr}_X[(\mathrm{Diag}\, M_a \otimes \mathbb{I}_V)\tau_{XV}]$$

which is a Hermitian operator on $\mathcal{V}$. There exists $|v\rangle \in \mathcal{V}$ such that

$$|\langle v|\sigma_a|v\rangle| = |\langle M_a, \mathrm{diag}\, \tau_{X|v}\rangle| = |\langle M_a, P^{\tau}_{X|v}\rangle| \geq 2^{-r},$$

which means that $\|\sigma_a\|_2 \geq 2^{-r}$. Now let $|u\rangle$ be a uniformly random unit vector in $\mathcal{V}$, and by Lemma 2.3.10 we know that for some absolute constant c,

$$\Pr_{|u\rangle}\left[|\langle u|\sigma_a|u\rangle| \geq 2^{-r'}\right] \geq 1 - 2^{(q+r-r')/2}c - e^{-2^q} \geq 1 - 2^{-r}c - e^{-1} \geq 1/2.$$

The second last inequality comes from Eq. (10.3), while the last inequality is because of the assumption that r is sufficiently large.

Since the above holds for every $a \in \mathcal{A}'$, it implies that $\Pr_{a\in\mathcal{A}',|u\rangle}[|\langle u|\sigma_a|u\rangle| \geq 2^{-r'}]$ is at least $1/2$. It means that there exists some $|u\rangle \in \mathcal{V}$ such that $|\langle u|\sigma_a|u\rangle| \geq 2^{-r'}$ for at least half of $a \in \mathcal{A}'$. On the other hand, since M is a (k', ℓ')-extractor with error $2^{-r'}$, and $\left\|P^{\tau}_{X|u}\right\|_2 \leq 2^{\ell'} \cdot 2^{-n/2}$, there are at most $2^{-k'}$ fraction of $a \in \mathcal{A}$ such that $|\langle u|\sigma_a|u\rangle| = |\langle M_a, P^{\tau}_{X|u}\rangle| \geq 2^{-r'}$. That means

$$\Pr_{a\in_R\mathcal{A}}[a \in \mathcal{A}'] \leq 2 \cdot 2^{-k'} \leq 2^{-2r}.$$

Here $k' - 1 \geq 2r$, by the definition of r. □

Corollary 10.3.13. *For every $0 \leq t \leq T$, we have $\mathbb{E}_{a\in_R\mathcal{A}} \left\|\tau^{(t,a)}_{XVW} - \tau^{(t,\infty)}_{XVW}\right\|_{\mathrm{Tr}} \leq 2^{-2r}$.*

Proof. For each $w \in \mathcal{W}$, the partial system $\tau^{(t,\infty)}_{XV|w}$ satisfies the condition of Lemma 10.3.12 since for every $|v\rangle \in \mathcal{V}$,

$$\left\| P^{\tau^{(t,\infty)}}_{X|v,w} \right\|_2 \leq (1 - 2^{-r})^{-1} \cdot 2^{\ell} \cdot 2^{-n/2} \leq 2^{\ell'} \cdot 2^{-n/2}.$$

Notice that for each $a \in \mathcal{A}$ such that there does not exist $|v\rangle \in \mathcal{V}$ with $|\langle M_a, P^{\tau^{(t,\infty)}}_{X|v,w}\rangle| \geq 2^{-r}$ (that is, when $G_a(v,w)$ holds for every $|v\rangle \in \mathcal{V}$), the sub system $\tau^{(t,\infty)}_{XV|w}$ is not touched in the truncation by G_a and we have $\tau^{(t,a)}_{XV|w} = \tau^{(t,\infty)}_{XV|w}$. Therefore

$$\begin{aligned}
\mathop{\mathbb{E}}_{a\in_R\mathcal{A}} \left\| \tau^{(t,a)}_{XVW} - \tau^{(t,\infty)}_{XVW} \right\|_{\mathrm{Tr}} &= \sum_{w\in\mathcal{W}} \mathop{\mathbb{E}}_{a\in_R\mathcal{A}} \left\| \tau^{(t,a)}_{XV|w} - \tau^{(t,\infty)}_{XV|w} \right\|_{\mathrm{Tr}} \\
&\leq \sum_{w\in\mathcal{W}} \Pr_{a\in_R\mathcal{A}} \left[\exists |v\rangle \in \mathcal{V}, |\langle M_a, P^{\tau^{(t,\infty)}}_{X|v,w}\rangle| \geq 2^{-r} \right] \cdot \mathrm{Tr}[\tau^{(t,\infty)}_{XV|w}] \\
&\leq 2^{-2r} \cdot \sum_{w\in\mathcal{W}} \mathrm{Tr}[\tau^{(t,\infty)}_{XV|w}] \leq 2^{-2r}.
\end{aligned}$$

□

Evolution preserves trace distance

Lemma 10.3.14. *For every $0 \leq t < T$, we have $\left\| \tau^{(t+1)}_{XVW} - \rho^{(t+1)}_{XVW} \right\|_{\mathrm{Tr}} \leq \mathbb{E}_{a\in_R\mathcal{A}} \left\| \tau^{(t,a)}_{XVW} - \rho^{(t)}_{XVW} \right\|_{\mathrm{Tr}}$.*

Proof. Recall that

$$\rho^{(t+1)}_{XVW} = \mathop{\mathbb{E}}_{a\in_R\mathcal{A}} \left[\sum_{x\in\mathcal{X}} |x\rangle\langle x| \otimes \Phi_{t,a,M(a,x)} \left(\rho^{(t)}_{VW|x} \right) \right],$$

$$\tau^{(t+1)}_{XVW} = \mathop{\mathbb{E}}_{a\in_R\mathcal{A}} \left[\sum_{x\in\mathcal{X}} |x\rangle\langle x| \otimes \Phi_{t,a,M(a,x)} \left(\tau^{(t,a)}_{VW|x} \right) \right].$$

Therefore by triangle inequality and contractivity of quantum channels under trace norms,

$$\begin{aligned}
\left\|\tau_{XVW}^{(t+1)} - \rho_{XVW}^{(t+1)}\right\|_{\mathrm{Tr}} &\leq \mathop{\mathbb{E}}_{a\in_R\mathcal{A}} \left\| \sum_{x\in\mathcal{X}} |x\rangle\langle x| \otimes \left(\Phi_{t,a,M(a,x)}\left(\tau_{VW|x}^{(t,a)}\right) - \Phi_{t,a,M(a,x)}\left(\rho_{VW|x}^{(t)}\right)\right)\right\|_{\mathrm{Tr}} \\
&= \mathop{\mathbb{E}}_{a\in_R\mathcal{A}} \sum_{x\in\mathcal{X}} \left\|\Phi_{t,a,M(a,x)}\left(\tau_{VW|x}^{(t,a)}\right) - \Phi_{t,a,M(a,x)}\left(\rho_{VW|x}^{(t)}\right)\right\|_{\mathrm{Tr}} \\
&\leq \mathop{\mathbb{E}}_{a\in_R\mathcal{A}} \sum_{x\in\mathcal{X}} \left\|\tau_{VW|x}^{(t,a)} - \rho_{VW|x}^{(t)}\right\|_{\mathrm{Tr}} \\
&= \mathop{\mathbb{E}}_{a\in_R\mathcal{A}} \left\|\tau_{XVW}^{(t,a)} - \rho_{XVW}^{(t)}\right\|_{\mathrm{Tr}}. \qquad \square
\end{aligned}$$

We are finally ready to prove Theorem 10.3.1.

Proof. First, combining Corollaries 10.3.6, 10.3.8, 10.3.10 and 10.3.13 and Lemma 10.3.14 we have

$$\left\|\tau_{XVW}^{(t+1)} - \rho_{XVW}^{(t+1)}\right\|_{\mathrm{Tr}} \leq \left\|\tau_{XVW}^{(t)} - \rho_{XVW}^{(t)}\right\|_{\mathrm{Tr}} + 8\cdot 2^{q-2r} + 2^{-5r} + 2^{-2r}.$$

Since $\tau_{XVW}^{(0)} = \rho_{XVW}^{(0)}$, by triangle inequality we know that $\left\|\tau_{XVW}^{(T)} - \rho_{XVW}^{(T)}\right\|_{\mathrm{Tr}} \leq T\cdot 10\cdot 2^{q-2r} \leq 10\cdot 2^{q-r}$, and thus

$$\left\|\tau_{XVW}^{(T,\infty)} - \rho_{XVW}^{(T)}\right\|_{\mathrm{Tr}} \leq 10\cdot 2^{q-r} + 8\cdot 2^{q-2r} + 2^{-5r}.$$

This bounds the difference between the measurement probabilities of $\tau_{XVW}^{(T,\infty)}$ and $\rho_{XVW}^{(T)}$ under any measurement, specifically the difference between the success probability of the

branching program ρ and the following value on τ:

$$\sum_{\substack{x\in\mathcal{X},v\in\{0,1\}^q,w\in\mathcal{W}\\ \widetilde{x}(v,w)=x}} \langle x,v,w|\tau_{XVW}^{(T,\infty)}|x,v,w\rangle = \sum_{v\in\{0,1\}^q,w\in\mathcal{W}} \mathrm{Tr}[\tau_{X|v,w}^{(T,\infty)}]\cdot P_{X|v,w}^{\tau,(T,\infty)}(\widetilde{x}(v,w)).$$

Since $\left\|P_{X|v,w}^{\tau,(T,\infty)}\right\|_\infty \le 2^{2\ell+9r}\cdot 2^{-n}$ and $\mathrm{Tr}[\tau_{XVW}^{(T,\infty)}]\le 1$, the above value is at most $2^{2\ell+9r}\cdot 2^{-n}$. Therefore the success probability of the branching program ρ is at most (recall that $2\ell+9r-n\le -r$)

$$10\cdot 2^{q-r}+8\cdot 2^{q-2r}+2^{-5r}+2^{2\ell+9r}\cdot 2^{-n}=O(2^{q-r}). \qquad \square$$

10.4 Target Distribution and Badness

In this section we prove Lemma 10.3.5. The first step is to analyze how $P_{X|v,w}^{\tau,(t)}$ evolves according to the rule

$$\tau_{XVW}^{(t+1)} = \mathop{\mathbb{E}}_{a\in_R\mathcal{A}}\left[\sum_{x\in\mathcal{X}}|x\rangle\langle x|\otimes\Phi_{t,a,M(a,x)}\left(\tau_{VW|x}^{(t,a)}\right)\right].$$

We introduce the following notations. For every $a\in\mathcal{A}$ and $b\in\{-1,1\}$, let

$$\vec{1}_{a,b}=\frac{1}{2}(\vec{1}+b\cdot M_a),$$

which is a 0-1 vector that indicates whether $M(a,x)=b$. Let

$$\tau_{XVW}^{(t,a,b)} = (\mathrm{Diag}\,\vec{1}_{a,b}\otimes\mathbb{I}_{VW})\tau_{XVW}^{(t,a)}, \tag{10.6}$$

so that we can write

$$\tau_{XVW}^{(t+1)} = \mathop{\mathbb{E}}_{a\in_R\mathcal{A}}\left[(\mathbb{I}_X\otimes\Phi_{t,a,1})\left(\tau_{XVW}^{(t,a,1)}\right) + (\mathbb{I}_X\otimes\Phi_{t,a,-1})\left(\tau_{XVW}^{(t,a,-1)}\right)\right]. \tag{10.7}$$

Thus Claim 10.1.2 implies that $P_{X|v,w}^{\tau^{(t+1)}}$ is a convex combination of $P_{X|v',w'}^{\tau^{(t,a,b)}}$ for some a, b, w' and $|v'\rangle$.

10.4.1 Target Distribution

Before considering the target distribution, let us first establish that the ℓ_2-norms of $P_{X|v,w}^{\tau^{(t)}}$ cannot be too large:

Lemma 10.4.1. *For every $0 \le t \le T$, $|v\rangle \in \mathcal{V}$, $w \in \mathcal{W}$, we have*

$$\left\|P_{X|v,w}^{\tau^{(t)}}\right\|_2 \le 4\cdot 2^{\ell}\cdot 2^{-n/2}.$$

Proof. When $t = 0$ the statement is clearly true as $P_{X|v,w}^{\tau^{(0)}}$ is always uniform.

Now assume $t > 0$. By Lemma 10.3.2 and Lemma 10.3.11 we know that

$$\left\|P_{X|v',w'}^{\tau^{(t-1,a)}}\right\|_2 \le (1-2^{-r})^{-1}\cdot 2^{\ell}\cdot 2^{-n/2}$$

for every $w' \in \mathcal{W}$, $|v'\rangle \in \mathcal{V}$ and $a \in \mathcal{A}$, as $\tau_{XVW}^{(t-1,a)}$ is truncated from $\tau_{XVW}^{(t-1,\infty)}$. Since $G_a(P_{X|v',w'}^{\tau^{(t-1,a)}})$ is true, meaning that the distribution is evenly divided by a, we further have

$$\left\|P_{X|v',w'}^{\tau^{(t-1,a,b)}}\right\|_2 = \frac{\left\|\vec{1}_{a,b}\cdot P_{X|v',w'}^{\tau^{(t-1,a)}}\right\|_2}{\left\|\vec{1}_{a,b}\cdot P_{X|v',w'}^{\tau^{(t-1,a)}}\right\|_1} \le 2(1-2^{-r})^{-1}\cdot\left\|P_{X|v',w'}^{\tau^{(t-1,a)}}\right\|_2 \le 4\cdot 2^{\ell}\cdot 2^{-n/2}.$$

Since $P^{\tau^{(t)}}_{X|v,w}$ is a convex combination of $P^{\tau^{(t-1,a,b)}}_{X|v',w'}$, by convexity its ℓ_2-norm is bounded by $4 \cdot 2^{\ell} \cdot 2^{-n/2}$. □

From now on we use P to denote a fixed target distribution (which we will later choose to be the distribution in Lemma 10.3.5), such that

$$2^{\ell} \cdot 2^{-n/2} \leq \|P\|_2 \leq 4 \cdot 2^{\ell} \cdot 2^{-n/2}.$$

We want to bound the progress of $\langle P^{\tau^{(t)}}_{X|v,w}, P\rangle$, which starts off as 2^{-n} at $t = 0$, and becomes at least $2^{2\ell} \cdot 2^{-n}$ when $P^{\tau^{(t)}}_{X|v,w} = P$. Note that by Cauchy-Schwarz we always have

$$\langle P^{\tau^{(t)}}_{X|v,w}, P\rangle \leq \left\|P^{\tau^{(t)}}_{X|v,w}\right\|_2 \|P\|_2 \leq 16 \cdot 2^{2\ell} \cdot 2^{-n}. \tag{10.8}$$

In order to bound the progress, we introduce some new notations. For any superscript (such as (t, a)) on the partial systems, we use σ_{XVW} to denote $\tau_{XVW}(\operatorname{Diag} P \otimes \mathbb{I}_{VW})$. Notice that

$$\operatorname{Tr}[\sigma_{X|v,w}] = \operatorname{Tr}[\tau_{X|v,w} \operatorname{Diag} P] = \operatorname{Tr}[\tau_{X|v,w}] \cdot \langle P^{\tau}_{X|v,w}, P\rangle.$$

Similarly, $P^{\sigma}_{X|v,w}$ can be deduced from $P^{\tau}_{X|v,w}$ via

$$P^{\sigma}_{X|v,w}(x) = \frac{\operatorname{Tr}[\tau_{X|v,w}]}{\operatorname{Tr}[\sigma_{X|v,w}]} \cdot P^{\tau}_{X|v,w}(x) \cdot P(x) = \frac{P^{\tau}_{X|v,w}(x) \cdot P(x)}{\langle P^{\tau}_{X|v,w}, P\rangle}. \tag{10.9}$$

Therefore we can bound the ℓ_2 norm of $P^{\sigma}_{X|v,w}$ as

$$\left\|P^{\sigma}_{X|v,w}\right\|_2 \leq \frac{1}{\langle P^{\tau}_{X|v,w}, P\rangle} \cdot \left\|P^{\tau}_{X|v,w}\right\|_\infty \cdot \|P\|_2.$$

10.4.2 Bad Events

Now we can identity the places where $\langle P^{\tau^{(t)}}_{X|v,w}, P\rangle$ increases by a lot, which happens when the *inner product* is not evenly divided by some $a \in \mathcal{A}$ (we will see the reason in the analysis later). Formally, at stage $0 \le t < T$, we say (w, a) is *bad* if

$$\exists |v\rangle \in \mathcal{V}, \text{s.t. } |\langle M_a, P^{\tau^{(t,a)}}_{X|v,w}\rangle| > 2^{-r} \text{ and } \langle P^{\tau^{(t,a)}}_{X|v,w}, P\rangle \ge \frac{1}{2}\cdot 2^{-n}. \tag{10.10}$$

Lemma 10.4.2. *For every $0 \le t < T$ and $w \in \mathcal{W}$, we have*

$$\Pr_{a\in_R\mathcal{A}}[(w,a) \textit{ is bad}] \le 2^{-k}.$$

Proof. Since $\tau^{(t,a)}_{XVW}$ is truncated from $\tau^{(t,\infty)}_{XVW}$, Lemma 10.3.2 shows that for every $|v\rangle \in \mathcal{V}$, $w \in \mathcal{W}$ and $a \in \mathcal{A}$ there is $|v'\rangle \in \mathcal{V}$ such that

$$P^{\tau^{(t,a)}}_{X|v,w} = P^{\tau^{(t,\infty)}}_{X|v',w}$$

and by (10.9) it also implies that

$$P^{\tau^{(t,a)}}_{X|v,w} = P^{\tau^{(t,\infty)}}_{X|v',w}.$$

Now fix some $w \in \mathcal{W}$, and let $\mathcal{A}' \subseteq \mathcal{A}$ be the set of of $a \in \mathcal{A}$ such that

$$\exists |v\rangle \in \mathcal{V}, \text{s.t. } |\langle M_a, P^{\tau^{(t,\infty)}}_{X|v,w}\rangle| > 2^{-r} \text{ and } \langle P^{\tau^{(t,\infty)}}_{X|v,w}, P\rangle \ge \frac{1}{2}\cdot 2^{-n}.$$

Then $\mathcal{A}'$ contains all a such that (w, a) is bad, and our goal is to bound the fraction of $\mathcal{A}'$ in $\mathcal{A}$.

In the rest of the proof we temporarily omit the super script and write $\tau^{(t,\infty)}$ and $\sigma^{(t,\infty)}$ simply as τ and σ. For the same reason as in Lemma 10.3.12 we can assume that $\tau_{V|w} = \mathbb{I}_V$, and thus

$$\langle v|\sigma_{V|w}|v\rangle = \mathrm{Tr}[\sigma_{X|v,w}] = \langle P^{\tau}_{X|v,w}, P\rangle, \text{ and } \mathrm{Tr}[\sigma_{XV|w}] = \langle P^{\tau}_{X|w}, P\rangle \leq 16 \cdot 2^{2\ell} \cdot 2^{-n}.$$

where the last inequality is by Lemma 10.3.11 and Cauchy-Schwarz, in the same way as (10.8).

Suppose that we have diagonalization $\sigma_{V|w} = U^\dagger DU$, where U is unitary and D is diagonal and non-negative. Let $\mathcal{V}' \subseteq \mathcal{V}$ be the subspace spanned by $U^\dagger|e\rangle$ over the computational basis vectors $|e\rangle \in \mathcal{V}$ such that $\langle e|D|e\rangle \geq 2^{-4r} \cdot 2^{-2\ell} \cdot 2^{-n}$. So for every $|v\rangle \in \mathcal{V}'$ we have

$$\langle P^{\tau}_{X|v,w}, P\rangle = \mathrm{Tr}[\sigma_{X|v,w}] \geq 2^{-4r} \cdot 2^{-2\ell} \cdot 2^{-n}.$$

We claim that for every $a \in \mathcal{A}'$, there exists $|v\rangle \in \mathcal{V}'$ such that $|\langle M_a, P^{\tau}_{X|v,w}\rangle| > \frac{1}{2} \cdot 2^{-r}$. To prove the claim, let Π be the projection operator from $\mathcal{V}$ to $\mathcal{V}'$, and then $(\mathbb{I}_X \otimes \Pi)\sigma_{XV|w}(\mathbb{I}_X \otimes \Pi)$ can be conceptually seen as a truncated partial system $\sigma^{\downarrow G}_{XV|w}$ where $G(v, w)$ holds when $\mathrm{Tr}[\sigma_{X|v,w}] \geq 2^{-4r-2\ell} \cdot 2^{-n}$ for the fixed w. By Lemma 10.3.3 we have

$$\left\|\sigma^{\downarrow G}_{XV|w} - \sigma_{XV|w}\right\|_{\mathrm{Tr}} \leq 3 \cdot 2^q \cdot \sqrt{2^{-4r-2\ell-n} \cdot \mathrm{Tr}[\sigma_{XV|w}]} \leq 12 \cdot 2^{q-2r} \cdot 2^{-n}.$$

Since $a \in \mathcal{A}'$, assume for $|u\rangle \in \mathcal{V}$ we have $|\langle M_a, P^{\tau}_{X|u,w}\rangle| > 2^{-r}$ and $\mathrm{Tr}[\sigma_{X|u,w}] =$

$\langle P^{\sigma}_{X|u,w}, P\rangle \geq \frac{1}{2}\cdot 2^{-n}$. Let $|v\rangle \sim \Pi|u\rangle$, then we have

$$\begin{aligned}
\left\|P^{\sigma}_{X|u,w} - P^{\sigma}_{X|v,w}\right\|_1 = \left\|P^{\sigma}_{X|u,w} - P^{\sigma|^G}_{X|u,w}\right\|_1 &\leq \left\|\frac{\sigma_{X|u,w}}{\mathrm{Tr}[\sigma_{X|u,w}]} - \frac{\sigma^{|G}_{X|u,w}}{\mathrm{Tr}[\sigma^{|G}_{X|u,w}]}\right\|_{\mathrm{Tr}} \\
&\leq \left\|\frac{\sigma_{X|u,w}}{\mathrm{Tr}[\sigma_{X|u,w}]} - \frac{\sigma^{|G}_{X|u,w}}{\mathrm{Tr}[\sigma_{X|u,w}]}\right\|_{\mathrm{Tr}} + \left\|\frac{\sigma^{|G}_{X|u,w}}{\mathrm{Tr}[\sigma_{X|u,w}]} - \frac{\sigma^{|G}_{X|u,w}}{\mathrm{Tr}[\sigma^{|G}_{X|u,w}]}\right\|_{\mathrm{Tr}} \\
&= \left\|\frac{\sigma_{X|u,w}}{\mathrm{Tr}[\sigma_{X|u,w}]} - \frac{\sigma^{|G}_{X|u,w}}{\mathrm{Tr}[\sigma_{X|u,w}]}\right\|_{\mathrm{Tr}} + \left|\frac{1}{\mathrm{Tr}[\sigma_{X|u,w}]} - \frac{1}{\mathrm{Tr}[\sigma^{|G}_{X|u,w}]}\right| \cdot \mathrm{Tr}[\sigma^{|G}_{X|u,w}] \\
&= \frac{\left\|\sigma_{X|u,w} - \sigma^{|G}_{X|u,w}\right\|_{\mathrm{Tr}}}{\mathrm{Tr}[\sigma_{X|u,w}]} + \frac{\left|\mathrm{Tr}[\sigma^{|G}_{X|u,w}] - \mathrm{Tr}[\sigma_{X|u,w}]\right|}{\mathrm{Tr}[\sigma_{X|u,w}]} \\
&\leq \frac{2\left\|\sigma_{X|u,w} - \sigma^{|G}_{X|u,w}\right\|_{\mathrm{Tr}}}{\mathrm{Tr}[\sigma_{X|u,w}]} \\
&\leq \frac{2\left\|\sigma_{XV|w} - \sigma^{|G}_{XV|w}\right\|_{\mathrm{Tr}}}{\mathrm{Tr}[\sigma_{X|u,w}]} \\
&\leq 48\cdot 2^{q-2r} \leq \frac{1}{2}\cdot 2^{-r},
\end{aligned}$$

where the last step is due to $q \leq r-7$. Thus

$$|\langle M_a, P^{\sigma}_{X|v,w}\rangle| \geq |\langle M_a, P^{\sigma}_{X|u,w}\rangle| - \left\|P^{\sigma}_{X|u,w} - P^{\sigma}_{X|v,w}\right\|_1 > \frac{1}{2}\cdot 2^{-r}.$$

Similarly to the proof for Lemma 10.3.12, for each $a \in \mathcal{A}'$ let

$$\pi_a = \mathrm{Tr}_X[(\mathrm{Diag}\, M_a \otimes U^\dagger D^{-1/2} U)\cdot \sigma_{XV|w}\cdot (\mathbb{I}_X \otimes U^\dagger D^{-1/2} U)]$$

which is a Hermitian operator on $\mathcal{V}$. For each $|v\rangle \in \mathcal{V}$, let $|v'\rangle \sim U^\dagger D^{1/2} U|v\rangle$. Recall that

$\sigma_{V|w} = U^\dagger D U$, and therefore

$$\begin{aligned} P^{\sigma}_{X|v,w} &= \frac{\operatorname{diag}\left(\mathbb{I}_X \otimes \langle v|\right)\sigma_{XV|w}\left(\mathbb{I}_X \otimes |v\rangle\right)}{\langle v|\sigma_{V|w}|v\rangle} \\ &= \frac{\operatorname{diag}\left(\mathbb{I}_X \otimes \langle v'|U^\dagger D^{-1/2}U\right)\sigma_{XV|w}\left(\mathbb{I}_X \otimes U^\dagger D^{-1/2}U|v'\rangle\right)}{\langle v'|U^\dagger D^{-1/2}U\sigma_{V|w}U^\dagger D^{-1/2}U|v'\rangle} \\ &= \operatorname{diag}\left(\mathbb{I}_X \otimes \langle v'|U^\dagger D^{-1/2}U\right)\sigma_{XV|w}\left(\mathbb{I}_X \otimes U^\dagger D^{-1/2}U|v'\rangle\right). \end{aligned}$$

And that means

$$\langle v'|\pi_a|v'\rangle = \left\langle M_a, \operatorname{diag}\left(\mathbb{I}_X \otimes \langle v'|U^\dagger D^{-1/2}U\right)\sigma_{XV|w}\left(\mathbb{I}_X \otimes U^\dagger D^{-1/2}U|v'\rangle\right)\right\rangle = \langle M_a, P^{\sigma}_{X|v,w}\rangle.$$

We showed above that there exists $|v\rangle \in \mathcal{V}'$, and thus $|v'\rangle \in \mathcal{V}'$ such that

$$|\langle v'|\pi_a|v'\rangle| = \left|\langle M_a, P^{\sigma}_{X|v,w}\rangle\right| \geq \frac{1}{2}\cdot 2^{-r},$$

which means that for $\Pi\pi_a\Pi$, the restriction of π_a on $\mathcal{V}'$, we have $\|\Pi\pi_a\Pi\|_2 \geq \frac{1}{2}\cdot 2^{-r}$. Now consider a uniformly random unit vector $|v'\rangle$ in $\mathcal{V}'$, and by Lemma 2.3.10 we know that for some absolute constant c,

$$\Pr_{|v\rangle'}\left[|\langle v'|\sigma_a|v'\rangle| \geq 2^{-r'}\right] \geq 1 - 2^{(q+r+1-r')/2}c - e^{-2^q} \geq 1 - 2^{-r}c - e^{-1} \geq \frac{1}{2}.$$

Therefore, for the random vector $|v\rangle \sim U^\dagger D^{-1/2}U|v'\rangle$ where $|v'\rangle$ is uniform in $\mathcal{V}'$, we conclude that

$$\Pr_{|v\rangle}\left[|\langle M_a, P^{\sigma}_{X|v,w}\rangle| \geq 2^{-r'}\right] \geq \frac{1}{2}.$$

On the other hand, as $|v'\rangle \in \mathcal{V}'$, it also holds that $|v\rangle \in \mathcal{V}'$, therefore $\langle P^{\tau}_{X|v,w}, P\rangle \geq 2^{-4r} \cdot 2^{-2\ell} \cdot 2^{-n}$ is always true. Thus there exists a $|v\rangle \in \mathcal{V}$ that simultaneously satisfies

$$\langle P^{\tau}_{X|v,w}, P\rangle \geq 2^{-4r} \cdot 2^{-2\ell} \cdot 2^{-n} \quad \text{and} \quad |\langle M_a, P^{\sigma}_{X|v,w}\rangle| \geq 2^{-r'}$$

for at least $1/2$ of $a \in \mathcal{A}'$. Since

$$\left\|P^{\sigma}_{X|v,w}\right\|_2 \leq \frac{1}{\langle P^{\tau}_{X|v,w}, P\rangle} \cdot \left\|P^{\sigma}_{X|v,w}\right\|_\infty \cdot \left\|P\right\|_2 \leq 4 \cdot 2^{5\ell+13r} \cdot 2^{-n/2} = 2^{\ell'} \cdot 2^{-n/2},$$

and M is a (k', ℓ')-extractor with error $2^{-r'}$, there are at most $2^{-k'}$ fraction of $a \in \mathcal{A}$ such that $|\langle M_a, P^{\sigma}_{X|v',w}\rangle| \geq 2^{-r'}$, which means that

$$\Pr_{a \in_R \mathcal{A}}[(w,a) \text{ is bad}] \leq \Pr_{a \in_R \mathcal{A}}[a \in \mathcal{A}'] \leq 2 \cdot 2^{-k'} = 2^{-k}. \qquad \square$$

10.4.3 Badness Levels

At stage t, for each classical memory state $w \in \mathcal{W}$ we count how many times the path to it has been bad, which is a random variable depending on the previous random choices of $a \in \mathcal{A}$. This is stored in another classical register B, which we call *badness level* and takes values $\beta \in \{0, \ldots, T\}$. It is initially set to be 0, that is, we let

$$\tau^{(0)}_{XVWB} = \tau^{(0)}_{XVW} \otimes |0\rangle\langle 0|_B.$$

We ensure that the distribution of B always only depends on W and is independent of X and V conditioned on W, using the following updating rules on the combined system

τ_{XVWB} for each stage $0 \leq t < T$:

- The truncation steps are executed independently of B. Therefore, for each $a \in \mathcal{A}$ we let

$$\tau^{(t,a)}_{XVWB} = \sum_{w \in \mathcal{W}} \tau^{(t,a)}_{XV|w} \otimes |w\rangle\langle w| \otimes \operatorname{Diag} P^{\tau^{(t)}}_{B|w}. \tag{10.11}$$

- The value of B updates before the evolution step, where for each $a \in \mathcal{A}$ and $b \in \{-1, 1\}$ we let

$$\tau^{(t,a,b)}_{XVWB} = (\operatorname{Diag} \vec{1}_{a,b} \otimes \mathbb{I}_V \otimes U_a)\tau^{(t,a)}_{XVWB}(\mathbb{I}_{XV} \otimes U_a^\dagger).$$

  Here U_a is a permutation operator, depending on $\tau^{(t,a)}_{XVW}$, acting on $\mathcal{W} \otimes \{0, \ldots, T\}$ such that

$$U_a|w\rangle|\beta\rangle = \begin{cases} |w\rangle|(\beta+1) \bmod (T+1)\rangle & \text{if } (w,a) \text{ is bad,} \\ |w\rangle|\beta\rangle & \text{otherwise.} \end{cases}$$

- For the evolution step, we apply the channels $\Phi_{t,a,b}$ on the memories W and V to get

$$\tau^{(t+1)}_{XVWB} = \mathop{\mathbb{E}}_{a \in_R \mathcal{A}} \left[(\mathbb{I}_X \otimes \Phi_{t,a,1} \otimes \mathbb{I}_B)\left(\tau^{(t,a,1)}_{XVWB}\right) + (\mathbb{I}_X \otimes \Phi_{t,a,-1} \otimes \mathbb{I}_B)\left(\tau^{(t,a,-1)}_{XVWB}\right) \right].$$

Notice that the evolution step might introduce dependencies between X, V and B. However, such dependencies are eliminated later due to how we handle the truncation steps (10.11), and thus do not affect our proof.

We can check that the combined partial system $\tau^{(t)}_{XVWB}$ defined above is always consistent with the partial system $\tau^{(t)}_{XVW}$ that we discussed in previous sections, in the sense that

$\mathrm{Tr}_B[\tau^{(t)}_{XVWB}] = \tau^{(t)}_{XVW}$ always holds:

- For the truncation step, it is straightforward to check that

$$\mathrm{Tr}_B[\tau^{(t,a)}_{XVWB}] = \sum_{w\in\mathcal{W}} \tau^{(t,a)}_{XV|w} \otimes |w\rangle\langle w| = \tau^{(t,a)}_{XVW}.$$

- The permutation operator U_a acts on $\mathcal{W}$ as identity since

$$\mathrm{Tr}_B\left[U_a|w,\beta\rangle\langle w,\beta|U_a^\dagger\right] = |w\rangle\langle w|.$$

  Recalling (10.6) that $\tau^{(t,a,b)}_{XVW} = (\mathrm{Diag}\,\vec{1}_{a,b} \otimes \mathbb{I}_V)\tau^{(t,a)}_{XVW}$, we have $\mathrm{Tr}_B[\tau^{(t,a,b)}_{XVWB}] = \tau^{(t,a,b)}_{XVW}$.

- The evolution step can be checked directly from the formula without B by (10.7):

$$\tau^{(t+1)}_{XVW} = \mathop{\mathbb{E}}_{a\in_R\mathcal{A}}\left[(\mathbb{I}_X \otimes \Phi_{t,a,1})\left(\tau^{(t,a,1)}_{XVW}\right) + (\mathbb{I}_X \otimes \Phi_{t,a,-1})\left(\tau^{(t,a,-1)}_{XVW}\right)\right].$$

So all previously proved properties about $\tau^{(t)}_{XVW}$ are preserved. In addition, we prove the following two properties about badness levels.

Lemma 10.4.3. *For every $0 \le t \le T$, $|v\rangle \in \mathcal{V}$ and $w \in \mathcal{W}$, we have*

$$\langle P^{\tau^{(t)}}_{X|v,w}, P\rangle \le \sum_{\beta=0}^{T} P^{\tau^{(t)}}_{B|w}(\beta)\cdot 2^{\beta}\cdot 2^{-n}\cdot(1-2^{-r})^{-3t}.$$

Proof. We prove it by induction on t. For $t = 0$ the lemma is true as $\langle P^{\tau^{(t)}}_{X|v,w}, P\rangle = 2^{-n}$ and $P^{\tau^{(t)}}_{B|w}(0) = 1$.

Suppose the lemma holds for some $t < T$. By an argument similar to Lemma 10.3.11 and applying Lemma 10.3.2 multiple times, we know that for every $|v\rangle \in \mathcal{V}, w \in \mathcal{W}$ and $a \in \mathcal{A}$, there exists $|v'\rangle$and $|v''\rangle \in \mathcal{V}$ such that

$$\langle P^{\tau^{(t,a)}}_{X|v,w}, P\rangle = \langle P^{\tau^{(t,\circ)}}_{X|v',w}, P\rangle \leq (1-2^{-r})^{-1}\langle P^{\tau^{(t,\star)}}_{X|v',w}, P\rangle = (1-2^{-r})^{-1}\langle P^{\tau^{(t)}}_{X|v'',w}, P\rangle,$$

and therefore

$$\langle P^{\tau^{(t,a)}}_{X|v,w}, P\rangle \leq \sum_{\beta=0}^{T} P^{\tau^{(t)}}_{B|w}(\beta)\cdot 2^{\beta}\cdot 2^{-n}\cdot(1-2^{-r})^{-3t-1}. \tag{10.12}$$

Also, the truncation step by G_a implies that $|\langle M_a, P^{\tau^{(t,a)}}_{X|v,w}\rangle| \leq 2^{-r}$. That is, for both $b \in \{-1,1\}$,

$$1-2^{-r} \leq 2\left\|\vec{1}_{a,b}\cdot P^{\tau^{(t,a)}}_{X|v,w}\right\|_1 \leq 1+2^{-r}.$$

Therefore we have, unconditionally

$$\langle P^{\tau^{(t,a,b)}}_{X|v,w}, P\rangle = \frac{\langle \vec{1}_{a,b}\cdot P^{\tau^{(t,a)}}_{X|v,w}, P\rangle}{\left\|\vec{1}_{a,b}\cdot P^{\tau^{(t,a)}}_{X|v,w}\right\|_1} \leq 2(1-2^{-r})^{-1}\cdot\langle P^{\tau^{(t,a)}}_{X|v,w}, P\rangle. \tag{10.13}$$

When the inner product is evenly divided, i.e. $|\langle M_a, P^{\tau^{(t,a)}}_{X|v,w}\rangle| \leq 2^{-r}$, we further have

$$\langle \vec{1}_{a,b}\cdot P^{\tau^{(t,a)}}_{X|v,w}, P\rangle \leq \frac{1}{2}(1+2^{-r})\langle P^{\tau^{(t,a)}}_{X|v,w}, P\rangle \leq \frac{1}{2}(1-2^{-r})^{-1}\langle P^{\tau^{(t,a)}}_{X|v,w}, P\rangle,$$

which means that

$$\langle P^{\tau^{(t,a,b)}}_{X|v,w}, P\rangle = \frac{\langle \vec{1}_{a,b}\cdot P^{\tau^{(t,a)}}_{X|v,w}, P\rangle}{\left\|\vec{1}_{a,b}\cdot P^{\tau^{(t,a)}}_{X|v,w}\right\|_1} \leq (1-2^{-r})^{-2}\cdot\langle P^{\tau^{(t,a)}}_{X|v,w}, P\rangle. \tag{10.14}$$

Now there are three cases to discuss:

- If (w,a) is bad, we have $P_{B|w}^{\tau^{(t,a,b)}}(\beta) = P_{B|w}^{\tau^{(t)}}(\beta-1)$ for every $\beta > 0$. Notice that $P_{B|w}^{\tau^{(t)}}(T) = 0$ as $t < T$, and thus (10.12) and (10.13) imply that

$$\begin{aligned}\langle P_{X|v,w}^{\tau^{(t,a,b)}}, P\rangle &\le \sum_{\beta=0}^{T-1} P_{B|w}^{\tau^{(t)}}(\beta)\cdot 2^{\beta+1}\cdot 2^{-n}\cdot(1-2^{-r})^{-3t-2}\\ &\le \sum_{\beta=0}^{T} P_{B|w}^{\tau^{(t,a,b)}}(\beta)\cdot 2^{\beta}\cdot 2^{-n}\cdot(1-2^{-r})^{-3(t+1)}.\end{aligned}$$

- If (w,a) is not bad and $|\langle M_a, P_{X|v,w}^{\tau^{(t,a)}}\rangle| \le 2^{-r}$, we have $P_{B|w}^{\tau^{(t,a,b)}}(\beta) = P_{B|w}^{\tau^{(t)}}(\beta)$ for every $\beta \ge 0$. Then (10.12) and (10.14) imply that

$$\begin{aligned}\langle P_{X|v,w}^{\tau^{(t,a,b)}}, P\rangle &\le \sum_{\beta=0}^{T} P_{B|w}^{\tau^{(t)}}(\beta)\cdot 2^{\beta}\cdot 2^{-n}\cdot(1-2^{-r})^{-3t-3}\\ &= \sum_{\beta=0}^{T} P_{B|w}^{\tau^{(t,a,b)}}(\beta)\cdot 2^{\beta}\cdot 2^{-n}\cdot(1-2^{-r})^{-3(t+1)}.\end{aligned}$$

- If (w,a) is not bad and $|\langle M_a, P_{X|v,w}^{\tau^{(t,a)}}\rangle| > 2^{-r}$, by the definition of badness (10.10) we must have $\langle P_{X|v,w}^{\tau^{(t,a)}}, P\rangle < \frac{1}{2}\cdot 2^{-n}$. Thus by (10.13),

$$\langle P_{X|v,w}^{\tau^{(t,a,b)}}, P\rangle < (1-2^{-r})^{-1}\cdot 2^{-n} \le \sum_{\beta=0}^{T} P_{B|w}^{\tau^{(t,a,b)}}(\beta)\cdot 2^{\beta}\cdot 2^{-n}\cdot(1-2^{-r})^{-3(t+1)}.$$

The last inequality follows from $\sum_{\beta=0}^{T} P_{B|w}^{\tau^{(t,a,b)}}(\beta)\cdot 2^{\beta}\cdot 2^{-n}\cdot(1-2^{-r})^{-3(t+1)} \ge 2^{-n}(1-2^{-r})^{-3(t+1)}$. Hence we obtain the same conclusion from all three cases.

For the evolution step, since B is classical we can view X and B as a whole and apply

Claim 10.1.2 on $P^{\tau^{(t+1)}}_{XB|v,w}$, which asserts that $P^{\tau^{(t+1)}}_{XB|v,w}$ is a convex combination of $P^{\tau^{(t,a,b)}}_{XB|v',w'}$ for some a, b, w' and $|v'\rangle$. Noted that even though in $\tau^{(t+1)}$, X and B are not independent, we can still use the linearity of partial trace to conclude that

$$\langle P^{\tau^{(t+1)}}_{X|v,w}, P\rangle \leq \sum_{\beta=0}^{T} P^{\tau^{(t+1)}}_{B|w}(\beta) \cdot 2^{\beta} \cdot 2^{-n} \cdot (1-2^{-r})^{-3(t+1)}. \qquad \square$$

Lemma 10.4.4. *For every $0 \leq \beta \leq t \leq T$ we have*

$$\langle \beta | \tau_B^{(t)} | \beta \rangle \leq 2^{-k\beta} \binom{t}{\beta}.$$

Proof. We prove it by induction on t. For $t = 0$ the lemma holds as $\tau_B^{(0)} = |0\rangle\langle 0|_B$. Also notice that the lemma is trivially true for every t when $\beta = 0$.

Now suppose the lemma holds for some t. By definition we have

$$\tau_B^{(t+1)} = \mathop{\mathbb{E}}_{a \in_R \mathcal{A}} [\tau_B^{(t,a,1)} + \tau_B^{(t,a,-1)}] = \mathop{\mathbb{E}}_{a \in_R \mathcal{A}} \mathrm{Tr}_W [U_a \tau_{WB}^{(t,a)} U_a^{\dagger}].$$

Therefore

$$\langle \beta | \tau_B^{(t+1)} | \beta \rangle = \sum_{w \in \mathcal{W}} \mathop{\mathbb{E}}_{a \in_R \mathcal{A}} \left[\langle w, \beta | U_a \tau_{WB}^{(t,a)} U_a^{\dagger} | w, \beta \rangle \right].$$

By Lemma 10.4.2 we know that for every $w \in \mathcal{W}$, the probability that (w, a) is bad for $a \in_R \mathcal{A}$ is at most 2^{-k}. In other words, for every $\beta > 0$,

$$U_a^{\dagger} |w, \beta\rangle = \begin{cases} |w, \beta\rangle, & \text{w.p. } \geq 1 - 2^{-k} \\ |w, \beta - 1\rangle, & \text{w.p. } \leq 2^{-k} \end{cases}$$

where the probability is taken over the random choice of a. It means that

$$\begin{aligned}\langle\beta|\tau_B^{(t+1)}|\beta\rangle &\leq \sum_{w\in\mathcal{W}}\langle w,\beta|\tau_{WB}^{(t,a)}|w,\beta\rangle + 2^{-k}\sum_{w\in\mathcal{W}}\langle w,\beta-1|\tau_{WB}^{(t,a)}|w,\beta-1\rangle \\ &= \langle\beta|\tau_B^{(t,a)}|\beta\rangle + 2^{-k}\cdot\langle\beta-1|\tau_B^{(t,a)}|\beta-1\rangle.\end{aligned}$$

Notice that

$$\tau_B^{(t,a)} = \sum_{w\in\mathcal{W}}\mathrm{Tr}[\tau_{XV|w}^{(t,a)}]\cdot\mathrm{Diag}\,P_{B|w}^{\tau^{(t)}} \leq \sum_{w\in\mathcal{W}}\mathrm{Tr}[\tau_{XV|w}^{(t)}]\cdot\mathrm{Diag}\,P_{B|w}^{\tau^{(t)}} = \tau_B^{(t)},$$

and thus we conclude that

$$\begin{aligned}\langle\beta|\tau_B^{(t+1)}|\beta\rangle &\leq \langle\beta|\tau_B^{(t)}|\beta\rangle + 2^{-k}\cdot\langle\beta-1|\tau_B^{(t)}|\beta-1\rangle \\ &\leq 2^{-k\beta}\binom{t}{\beta} + 2^{-k}\cdot 2^{-k(\beta-1)}\binom{t}{\beta-1} = 2^{-k\beta}\binom{t+1}{\beta}.\end{aligned}$$ □

With the lemmas above in hand, we can finally prove Lemma 10.3.5.

Proof for Lemma 10.3.5. For the target distribution $P = P_{X|v,w}^{\tau^{(t)}}$ we have $\langle P_{X|v,w}^{\tau^{(t)}}, P\rangle > 2^{2\ell}\cdot 2^{-n}$, so by Lemma 10.4.3,

$$\sum_{\beta=0}^{T} P_{B|w}^{\tau^{(t)}}(\beta)\cdot 2^{\beta}\cdot(1-2^{-r})^{-3t} > 2^{2\ell}.$$

Since $t \leq T \leq 2^{r-2}$, we have $(1-2^{-r})^{-3t} \leq 2$, and thus

$$\sum_{\beta=\ell}^{T} P_{B|w}^{\tau^{(t)}}(\beta)\cdot 2^{\beta} > \frac{1}{2}\left(2^{2\ell} - 2\cdot\sum_{\beta=0}^{\ell-1}2^{\beta}\right) > 2^{\ell}.$$

On the other hand, for every $\beta \geq \ell$, by Lemma 10.4.4,

$$\mathrm{Tr}[\tau^{(t)}_{B|w}] \cdot P^{\tau^{(t)}}_{B|w}(\beta) \leq \langle \beta | \tau^{(t)}_B | \beta \rangle \leq (2^{-k}t)^\beta < 2^{-(k-r)\beta},$$

and thus by (10.5),

$$\mathrm{Tr}[\tau^{(t)}_{X|v,w}] \leq \mathrm{Tr}[\tau^{(t)}_{B|w}] < 2^{-\ell} \sum_{\beta=\ell}^{T} 2^{-(k-r)\beta} \cdot 2^\beta \leq 2 \cdot 2^{-(k-r)\ell} \leq 2^{-2m} \cdot 2^{-4r}.$$

□

References

[Aar20] Scott Aaronson. Shadow tomography of quantum states. *SIAM J. Comput.*, 49(5), 2020.

[Abr91] Karl R. Abrahamson. Time-space tradeoffs for algebraic problems on general sequential machines. *J. Comput. Syst. Sci.*, 43(2):269–289, 1991.

[Adl78] Leonard M. Adleman. Two theorems on random polynomial time. In *19th Annual Symposium on Foundations of Computer Science*, pages 75–83. IEEE Computer Society, 1978.

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58:137–147, 1999.

[Aud07] Koenraad MR Audenaert. A sharp continuity estimate for the von neumann entropy. *Journal of Physics A: Mathematical and Theoretical*, 40(28):8127, 2007.

[Bar89] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *J. Comput. Syst. Sci.*, 38(1):150–164, 1989.

[BC82] Allan Borodin and Stephen A. Cook. A time-space tradeoff for sorting on a general sequential model of computation. *SIAM J. Comput.*, 11(2):287–297, 1982.

[BCC+14] Dominic W. Berry, Andrew M. Childs, Richard Cleve, Robin Kothari, and Rolando D. Somma. Exponential improvement in precision for simulating sparse hamiltonians. In *Symposium on Theory of Computing, STOC 2014*, pages 283–292. ACM, 2014.

[BCC+15] Dominic W. Berry, Andrew M. Childs, Richard Cleve, Robin Kothari, and Rolando D. Somma. Simulating hamiltonian dynamics with a truncated taylor series. *Physical review letters*, 114(9):090502, 2015.

[BCG18] Mark Braverman, Gil Cohen, and Sumegha Garg. Hitting sets with near-optimal error for read-once branching programs. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018*, pages 353–362. ACM, 2018.

[BCK15] Dominic W. Berry, Andrew M. Childs, and Robin Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015*, pages 792–809. IEEE Computer Society, 2015.

[BCM13] Paul Beame, Raphaël Clifford, and Widad Machmouchi. Element distinctness, frequency moments, and sliding windows. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013*, pages 290–299. IEEE Computer Society, 2013.

[Bea91] Paul Beame. A general sequential time-space tradeoff for finding unique elements. *SIAM J. Comput.*, 20(2):270–277, 1991.

[BGNV18] Nikhil Bansal, Shashwat Garg, Jesper Nederlof, and Nikhil Vyas. Faster space-efficient algorithms for subset sum, k-sum, and related problems. *SIAM J. Comput.*, 47(5):1755–1777, 2018.

[BGY18] Paul Beame, Shayan Oveis Gharan, and Xin Yang. Time-space tradeoffs for learning finite functions from random evaluations, with applications to polynomials. In *Conference On Learning Theory, COLT 2018*, volume 75 of *Proceedings of Machine Learning Research*, pages 843–856. PMLR, 2018.

[Blu84] Norbert Blum. A boolean function requiring $3n$ network size. *Theor. Comput. Sci.*, 28:337–345, 1984.

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM J. Comput.*, 13(4):850–864, November 1984.

[BNS92] László Babai, Noam Nisan, and Mario Szegedy. Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. *J. Comput. Syst. Sci.*, 45(2):204–232, 1992.

[Bog18] Andrej Bogdanov. Small bias requires large formulas. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018*, volume 107 of *LIPIcs*, pages 22:1–22:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[BSSV03] Paul Beame, Michael E. Saks, Xiaodong Sun, and Erik Vee. Time-space tradeoff lower bounds for randomized computation of decision problems. *J. ACM*, 50(2):154–195, 2003.

[BV97] Ethan Bernstein and Umesh V. Vazirani. Quantum complexity theory. *SIAM J. Comput.*, 26(5):1411–1473, 1997.

[CC17] Amit Chakrabarti and Yining Chen. Time-space tradeoffs for the memory game. *arXiv preprint arXiv:1712.01330*, 2017.

[CCHL21] Sitan Chen, Jordan Cotler, Hsin-Yuan Huang, and Jerry Li. Exponential separations between learning with and without quantum memory. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021*, pages 574–585. IEEE, 2021.

[CCvM06] Jin-yi Cai, Venkatesan T. Chakaravarthy, and Dieter van Melkebeek. Time-space tradeoff in derandomizing probabilistic logspace. *Theory Comput. Syst.*, 39(1):189–208, 2006.

[CDST22] Gil Cohen, Dean Doron, Ori Sberlo, and Amnon Ta-Shma. Approximating iterated multiplication of stochastic matrices in small space. *Electron. Colloquium Comput. Complex.*, TR22-149, 2022.

[CGJ19] Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. The power of block-encoded matrix powers: Improved regression techniques via faster hamiltonian simulation. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019*, volume 132 of *LIPIcs*, pages 33:1–33:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[CH22] Kuan Cheng and William M. Hoza. Hitting sets give two-sided derandomization of small space. *Theory of Computing*, 18(21):1–32, 2022.

[CJWW22] Lijie Chen, Ce Jin, R. Ryan Williams, and Hongxun Wu. Truly low-space element distinctness and subset sum via pseudorandom hash functions. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*, pages 1661–1678. SIAM, 2022.

[Csa76] L. Csanky. Fast parallel matrix inversion algorithms. *SIAM J. Comput.*, 5(4):618–623, 1976.

[CT21] Lijie Chen and Roei Tell. Simple and fast derandomization from very hard functions: eliminating randomness at almost no cost. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 283–291. ACM, 2021.

[CW01] Anthony Carbery and James Wright. Distributional and L^q norm inequalities for polynomials over convex bodies in $\mathbb{R}^n$. *Mathematical Research Letters*, 8(3):233–248, 2001.

[Din20] Itai Dinur. Tight time-space lower bounds for finding multiple collision pairs and their applications. In *EUROCRYPT 2020, 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 12105 of *Lecture Notes in Computer Science*, pages 405–434. Springer, 2020.

[Din23] Itai Dinur. Time-space lower bounds for bounded-error computation in the random-query model. *Electronic Colloquium on Computational Complexity: ECCC*, 2023.

[DQW22] Yevgeniy Dodis, Willy Quach, and Daniel Wichs. Authentication in the bounded storage model. In *EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 13277 of *Lecture Notes in Computer Science*, pages 737–766. Springer, 2022.

[DST17] Dean Doron, Amir Sarid, and Amnon Ta-Shma. On approximating the eigenvalues of stochastic matrices in probabilistic logspace. *Comput. Complex.*, 26(2):393–420, 2017.

[DT23] Dean Doron and Roei Tell. Derandomization with minimal memory footprint. *Electron. Colloquium Comput. Complex.*, TR23-036, 2023.

[FKL+16] Bill Fefferman, Hirotada Kobayashi, Cedric Yen-Yu Lin, Tomoyuki Morimae, and Harumichi Nishimura. Space-efficient error reduction for unitary quantum computations. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016*, volume 55 of *LIPIcs*, pages 14:1–14:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[FL18] Bill Fefferman and Cedric Yen-Yu Lin. A complete characterization of unitary quantum space. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018*, volume 94 of *LIPIcs*, pages 4:1–4:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[FR21] Bill Fefferman and Zachary Remscrim. Eliminating intermediate measurements in space-bounded quantum computation. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1343–1356. ACM, 2021.

[FvdG99] Christopher A. Fuchs and Jeroen van de Graaf. Cryptographic distinguishability measures for quantum-mechanical states. *IEEE Transactions on Information Theory*, 45(4):1216–1227, 1999.

[GHM+21] Uma Girish, Justin Holmgren, Kunal Mittal, Ran Raz, and Wei Zhan. Parallel repetition for the GHZ game: A simpler proof. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021*, volume 207 of *LIPIcs*, pages 62:1–62:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[GHM+22] Uma Girish, Justin Holmgren, Kunal Mittal, Ran Raz, and Wei Zhan. Parallel repetition for all 3-player games over binary alphabet. In *54th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2022*, pages 998–1009. ACM, 2022.

[GKLR21] Sumegha Garg, Pravesh K. Kothari, Pengda Liu, and Ran Raz. Memory-sample lower bounds for learning parity with noise. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021*, volume 207 of *LIPIcs*, pages 60:1–60:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. Delegating computation: interactive proofs for muggles. *Journal of the ACM (JACM)*, 62(4):1–64, 2015.

[GKR20] Sumegha Garg, Pravesh K. Kothari, and Ran Raz. Time-space tradeoffs for distinguishing distributions and applications to security of goldreich's PRG. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020*, volume 176 of *LIPIcs*, pages 21:1–21:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, STOC '89, page 25–32, New York, NY, USA, 1989. Association for Computing Machinery.

[GL19] Ofer Grossman and Yang P. Liu. Reproducibility and pseudo-determinism in log-space. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on DiscreteAlgorithms, SODA 2019*, pages 606–620. SIAM, 2019.

[GMRZ22] Uma Girish, Kunal Mittal, Ran Raz, and Wei Zhan. Polynomial bounds on parallel repetition for all 3-player games with binary inputs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2022*, volume 245 of *LIPIcs*, pages 6:1–6:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[GR22] Uma Girish and Ran Raz. Eliminating intermediate measurements using pseudorandom generators. In *13th Innovations in Theoretical Computer Science Conference, ITCS 2022*, volume 215 of *LIPIcs*, pages 76:1–76:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[GRT18] Sumegha Garg, Ran Raz, and Avishay Tal. Extractor-based time-space lower bounds for learning. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018*, pages 990–1002. ACM, 2018.

[GRT19] Sumegha Garg, Ran Raz, and Avishay Tal. Time-space lower bounds for two-pass learning. In *34th Computational Complexity Conference, CCC 2019*, volume 137 of *LIPIcs*, pages 22:1–22:39. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[GRZ21a] Uma Girish, Ran Raz, and Wei Zhan. Lower bounds for XOR of forrelations. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021*, volume 207 of *LIPIcs*, pages 52:1–52:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[GRZ21b] Uma Girish, Ran Raz, and Wei Zhan. Quantum logspace algorithm for powering matrices with bounded norm. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021*, volume 198 of *LIPIcs*, pages 73:1–73:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[GRZ23] Uma Girish, Ran Raz, and Wei Zhan. Is untrusted randomness helpful? In *14th Innovations in Theoretical Computer Science Conference, ITCS 2023*, volume 251 of *LIPIcs*, pages 56:1–56:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[GS92] Peter Gemmell and Madhu Sudan. Highly resilient correctors for polynomials. *Information processing letters*, 43(4):169–174, 1992.

[GUV09] Venkatesan Guruswami, Christopher Umans, and Salil P. Vadhan. Unbalanced expanders and randomness extractors from parvaresh-vardy codes. *J. ACM*, 56(4):20:1–20:34, 2009.

[GZ19] Jiaxin Guan and Mark Zhandry. Simple schemes in the bounded storage model. In *EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 11478 of *Lecture Notes in Computer Science*, pages 500–524. Springer, 2019.

[HHL09] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.

[HK18] William M. Hoza and Adam R. Klivans. Preserving randomness for adaptive algorithms. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018*, volume 116 of *LIPIcs*, pages 43:1–43:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[HM21] Yassine Hamoudi and Frédéric Magniez. Quantum time-space tradeoff for finding multiple collision pairs. In *16th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2021*, volume 197 of *LIPIcs*, pages 1:1–1:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[Hoz19] William M. Hoza. Typically-correct derandomization for small time and space. In *34th Computational Complexity Conference, CCC 2019*, volume 137 of *LIPIcs*, pages 9:1–9:39. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[Hoz21] William M. Hoza. Better pseudodistributions and derandomization for space-bounded computation. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021*, volume 207 of *LIPIcs*, pages 28:1–28:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[Hoz22] William M. Hoza. Recent progress on derandomizing space-bounded computation. *Bulletin of EATCS*, 138(3), 2022.

[HS65] Juris Hartmanis and Richard Edwin Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.

[ICC17] Raban Iten, Roger Colbeck, and Matthias Christandl. Quantum circuits for quantum channels. *Physical Review A*, 95(5):052316, 2017.

[Imp95] Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *36th Annual Symposium on Foundations of Computer Science*, pages 538–545. IEEE Computer Society, 1995.

[INW94] Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 356–364. ACM, 1994.

[IP99] Russell Impagliazzo and Ramamohan Paturi. Complexity of k-sat. In *Proceedings of the 14th Annual IEEE Conference on Computational Complexity*, pages 237–240. IEEE Computer Society, 1999.

[IW97] Russell Impagliazzo and Avi Wigderson. $\mathsf{P} = \mathsf{BPP}$ if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, pages 220–229. ACM, 1997.

[Jef22] Stacey Jeffery. Span programs and quantum space complexity. *Theory Comput.*, 18:1–49, 2022.

[KK12] Roy Kasher and Julia Kempe. Two-source extractors secure against quantum adversaries. *Theory Comput.*, 8(1):461–486, 2012.

[Kla03] Hartmut Klauck. Quantum time-space tradeoffs for sorting. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 69–76. ACM, 2003.

[KRT17] Gillat Kol, Ran Raz, and Avishay Tal. Time-space hardness of learning sparse parities. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017*, pages 1067–1080. ACM, 2017.

[KŠdW07] Hartmut Klauck, Robert Špalek, and Ronald de Wolf. Quantum and classical strong direct product theorems and optimal time-space tradeoffs. *SIAM J. Comput.*, 36(5):1472–1493, 2007.

[KvM02] Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM J. Comput.*, 31(5):1501–1526, 2002.

[LM00] Beatrice Laurent and Pascal Massart. Adaptive estimation of a quadratic functional by model selection. *The Annals of Statistics*, 28(5), October 2000.

[LMT00] Klaus-Jörn Lange, Pierre McKenzie, and Alain Tapp. Reversible space equals deterministic space. *J. Comput. Syst. Sci.*, 60(2):354–367, 2000.

[LPS88] Alexander Lubotzky, Ralph Phillips, and Peter Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.

[LRZ23] Qipeng Liu, Ran Raz, and Wei Zhan. Memory-sample lower bounds for learning with classical-quantum hybrid memory. In *55th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2023*. To Appear, 2023.

[LV21] Jiahui Liu and Satyanarayana Vusirikala. Secure multiparty computation in the bounded storage model. In *Cryptography and Coding - 18th IMA International Conference, IMACC 2021*, volume 13129 of *Lecture Notes in Computer Science*, pages 289–325. Springer, 2021.

[LY22] Jiatu Li and Tianqi Yang. $3.1n - o(n)$ circuit lower bounds for explicit functions. In *54th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2022*, pages 1180–1193. ACM, 2022.

[LZ23] Xin Lyu and Weihao Zhu. Time-space tradeoffs for element distinctness and set intersection via pseudorandomness. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023*, pages 5243–5281. SIAM, 2023.

[Mau92] Ueli M. Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *J. Cryptol.*, 5(1):53–66, 1992.

[MM18] Dana Moshkovitz and Michal Moshkovitz. Entropy samplers and strong generic lower bounds for space bounded learning. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018*, volume 94 of *LIPIcs*, pages 28:1–28:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[Mon10] Ashley Montanaro. Nonadaptive quantum query complexity. *Inf. Process. Lett.*, 110(24):1110–1113, 2010.

[MVBS05] Mikko Möttönen, Juha J Vartiainen, Ville Bergholm, and Martti M Salomaa. Transformation of quantum states using uniformly controlled rotations. *Quantum Information & Computation*, 5(6):467–473, 2005.

[MW05] Chris Marriott and John Watrous. Quantum arthur-merlin games. *Comput. Complex.*, 14(2):122–152, 2005.

[MW18] Dylan M. McKay and R. Ryan Williams. Quadratic Time-Space Lower Bounds for Computing Natural Functions with a Random Oracle. In *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*, volume 124 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 56:1–56:20, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[NC10] Michael A Nielsen and Isaac L Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[Nis90] Noam Nisan. Psuedorandom generators for space-bounded computation. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 204–212. ACM, 1990.

[NN93] Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, October 1994.

[OP04] Masanori Ohya and Dénes Petz. *Quantum entropy and its use*. Springer Science & Business Media, 2004.

[PRZ23] Edward Pyne, Ran Raz, and Wei Zhan. Certified hardness vs. randomness for log-space. *Electronic Colloquium on Computational Complexity: ECCC*, 2023.

[PZ32] R.E.A.C. Paley and A. Zygmund. A note on analytic functions in the unit circle. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 28, pages 266–272. Cambridge University Press, 1932.

[Raz17] Ran Raz. A time-space lower bound for a large class of learning problems. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*, pages 732–742. IEEE Computer Society, 2017.

[Raz18] Ran Raz. Fast learning requires good memory: A time-space lower bound for parity learning. *J. ACM*, 66(1), dec 2018.

[RPRŻ13] Wojciech Roga, Zbigniew Puchała, Lukasz Rudnicki, and Karol Życzkowski. Entropic trade-off relations for quantum operations. *Physical Review A*, 87(3):032308, 2013.

[RS08] Alvin C. Rencher and G. Bruce Schaalje. *Linear models in statistics*. John Wiley & Sons, 2008.

[RT22] Ran Raz and Avishay Tal. Oracle separation of BQP and PH. *J. ACM*, 69(4):30:1–30:21, 2022.

[RVW01] Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders. *Annals of Mathematics*, 155(1), January 2001.

[RY22] Gregory Rosenthal and Henry Yuen. Interactive proofs for synthesizing quantum states and unitaries. In *13th Innovations in Theoretical Computer Science Conference, ITCS 2022*, volume 215 of *LIPIcs*, pages 112:1–112:4. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[RZ20] Ran Raz and Wei Zhan. The random-query model and the memory-bounded coupon collector. In *11th Innovations in Theoretical Computer Science Conference, ITCS 2020*, volume 151 of *LIPIcs*, pages 20:1–20:11. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[Sak96] Michael Saks. Randomization and derandomization in space-bounded computation. In *Proceedings of the 11th Annual IEEE Conference on Computational Complexity*, pages 128–149, 1996.

[Sav98] John E. Savage. *Models of computation - exploring the power of computing*. Addison-Wesley, 1998.

[Sha81] Adi Shamir. The generation of cryptographically strong pseudo-random sequences. In *CRYPTO*, page 1. U. C. Santa Barbara, Dept. of Elec. and Computer Eng., ECE Report No 82-04, 1981.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science*, pages 124–134. IEEE Computer Society, 1994.

[SNA+17] Chao Shen, Kyungjoo Noh, Victor V Albert, Stefan Krastanov, Michel H Devoret, Robert J Schoelkopf, SM Girvin, and Liang Jiang. Quantum channel construction with circuit quantum electrodynamics. *Physical Review B*, 95(13):134501, 2017.

[Sti55] W Forrest Stinespring. Positive functions on c*-algebras. *Proceedings of the American Mathematical Society*, 6(2):211–216, 1955.

[STV01] Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. Pseudorandom generators without the XOR lemma. *J. Comput. Syst. Sci.*, 62(2):236–266, 2001.

[SZ99] Michael E. Saks and Shiyu Zhou. $\mathsf{BP_H Space}(S) \subseteq \mathsf{DSPACE}(S^{3/2})$. *J. Comput. Syst. Sci.*, 58(2):376–403, 1999.

[Ta-13] Amnon Ta-Shma. Inverting well conditioned matrices in quantum logspace. In *Symposium on Theory of Computing Conference, STOC'13*, pages 881–890. ACM, 2013.

[Tal17] Avishay Tal. Formula lower bounds via the quantum method. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017*, pages 1256–1268. ACM, 2017.

[Uhl76] Armin Uhlmann. The "transition probability" in the state space of a *-algebra. *Reports on Mathematical Physics*, 9(2):273–279, 1976.

[Vid18] Thomas Vidick. The quantum circuit model. *UCSD Summer school notes*, 2018.

[vMW12] Dieter van Melkebeek and Thomas Watson. Time-space efficient simulations of quantum computations. *Theory Comput.*, 8(1):1–51, 2012.

[Wat99] John Watrous. Space-bounded quantum complexity. *J. Comput. Syst. Sci.*, 59(2):281–326, 1999.

[WB86] Lloyd R. Welch and Elwyn R. Berlekamp. Error correction for algebraic block codes, December 30 1986. US Patent 4,633,470.

[Wil08] R. Ryan Williams. Time-space tradeoffs for counting NP solutions modulo integers. *Comput. Complex.*, 17(2):179–219, 2008.

[Wil16] R. Ryan Williams. Strong ETH breaks with merlin and arthur: Short non-interactive proofs of batch evaluation. In *31st Conference on Computational Complexity, CCC 2016*, volume 50 of *LIPIcs*, pages 2:1–2:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *FOCS*, pages 80–91. IEEE Computer Society, 1982.

[YZ22] Takashi Yamakawa and Mark Zhandry. Verifiable quantum advantage without structure. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022*, pages 69–74. IEEE, 2022.

[YZ23] Huacheng Yu and Wei Zhan. Randomized vs. deterministic separation in time-space tradeoffs of multi-output functions, 2023. In Preparation.

[ZLY22] Xiao-Ming Zhang, Tongyang Li, and Xiao Yuan. Quantum state preparation with optimal circuit depth: Implementations and applications. *Physical Review Letters*, 129(23):230504, 2022.