Arjan Cornelissen Quantum multivariate estimation and span program algorithms



# Quantum multivariate estimation and span program algorithms








Arjan Cornelissen

# Quantum multivariate estimation and span program algorithms

ILLC Dissertation Series DS-2023-02

Institute for Logic, Language and Computation

For further information about ILLC-publications, please contact

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Science Park 107
1098 XG Amsterdam
phone: +31-20-525 6051
e-mail: illc@uva.nl
homepage: http://www.illc.uva.nl/



Printed and bound by Druk. Tan Heck

ISBN: 978–90–6824–076–4

# Quantum multivariate estimation and span program algorithms

Academisch Proefschrift

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. P.P.C.C. Verbeek ten overstaan van een door het
College voor Promoties ingestelde
commissie, in het openbaar te verdedigen in de Agnietenkapel
op vrijdag 17 februari 2023, te 16.00 uur

door

Adriaan Jaco Cornelissen

geboren te Delft

**Promotiecommissie**

| | | |
|---|---|---|
| Promotor: | Prof. Dr. R.M. de Wolf | Universiteit van Amsterdam |
| Co-promotor: | Dr. M. Ozols | Universiteit van Amsterdam |
| Overige leden: | Prof. Dr. A. Ambainis | Latvijas Universitāte |
| | Prof. Dr. H.M. Buhrman | Universiteit van Amsterdam |
| | Prof. Dr. C. Schaffner | Universiteit van Amsterdam |
| | Dr. S. Jeffery | Centrum Wiskunde & Informatica |
| | Dr. F. Speelman | Universiteit van Amsterdam |

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

*to QuSoft*

# Contents

## Part II: Span programs

# Acknowledgments

Of course I did not do my PhD in solitary confinement (even though now and then covid-19 made it feel that way), and there have been several people along the way that helped me tremendously. I could not write this thesis without extending my sincerest gratitude to everyone that was involved in one way or the other.

First and foremost, I would specifically like to thank Māris, who has supervised me throughout these four years. I know I have not always been the easiest student to supervise, but you were always there for me regardless and you were very supportive throughout. I am very grateful for your supervision and the time we spent together. I think after all we can conclude we were quite *constructively aligned*, don't you think?

I would also like to thank Ronald, my promotor. You have always been periodically checking up on how I was doing, and on numerous occasions you provided me with very valuable advice, regarding research in particular as well as about the academic world in general. I am very grateful for these lessons learned, and I'm sure I will take them with me throughout the rest of my academic career. And of course if I ever again have an oh-how-did-hyphenation-work-again-moment, then I know who to ask.

Next, I would like to extend my gratitude to the members of my committee, Andris, Harry, Chris, Stacey and Florian. Thanks for taking the time to take part in my committee, and assess the quality of my thesis. I hope reading this thesis did not consume as much effort as it did writing it.

Of course I am also very grateful to have been given the chance to cooperate with many great researchers on various research projects. In particular, I would like to thank my co-authors, Álvaro, Māris, Stacey, András, Johannes, Sofiene, Vedran, Yassine, Nikhil, Ronald, Subha, Joran and Giacomo. I had a lot of fun working with all of you, and I hope that many more projects will ensue in the future. Besides that, I have to say that M&M's will never taste the same again.

Through the conferences and symposia I attended, I got to know quite some people from all kinds of corners in the world. I'm not going to be able to mention

everyone here, but I would like to mention in particular Yassine, Yixin, João, Alessandro, Armando, Kianna, Ana, Mathias, Alicja, Emiel and Casper. It was a pleasure to spend time with you guys, and I hope to bump into all of you many times again.

During my PhD, I had the chance to do an internship at IBM in New York. I would like to thank Giacomo for notifying me of the opportunity, and Kristan for accommodating my stay there. I would also like to thank Vikesh for providing me with ample gin tonic, and Vojtech for giving me ample rides back to White Plains. And, of course, I would like to thank all the interns that made my stay a lot of fun. In particular, I would like to mention Uma, Christophe, Harshitha, Sai Sree and Weiqiu for several hikes and a wonderful Niagara Falls trip. Similarly, I would like to thank Matt, Priyanka, Tergel, Vincent, Chhavi, Vikram, Alexandra, Clara, Alex, Adou and Yadav for various activities in and around New York. I also had tremendous fun with Nick, Ronak, Timur, Majo, Michelle and Jay-U, not only in New York but also in Mexico. In particular I would like to thank Majo for hosting us all in Mexico. Finally, special mention goes out to Fatih and Utku, because of you two I can never think the same way about gummy bears anymore.

I am extremely grateful to have been given the opportunity to spend four years at QuSoft. Before I started my PhD I had no clue how wonderful a group of people QuSoft actually is, and I would like to thank all its members for the incredible time it has been. It felt a bit like second family – or at times maybe even a bit closer than that.

In particular, I would like to thank my fellow members of the peanut butter room, Farrokh, Álvaro, Harold and Subha. I will never forget how we always started the day with an hour of pointless banter, I will forever realize that I can never beat Harold's word plays, and I hope no-one will ever again have to reinstate our covid infection counter on the whiteboard. Also, Subha, I particularly enjoyed your random questions on parenting, but I'm afraid I still don't have a reasonable answer to until what age you can get away with lying to your kid. Jokes aside, you are the best example of a power woman I know, and I will cherish the time we were able to spend together.

Next, I would like to thank all the members of QuSoft that have invited and/or joined me on various trips around the globe, of which I mention a few here. First, I thank Joran and Sebastian for a very memorable wintersport trip, even though I managed to break my wrist on the first day. Then, I would like to thank Nikhil for inviting me to his wedding, Subha, Quinten, Hema, Llorenç, Sebastian and Chelsea for making wonderful memories there, and Lynn and Chris for an eventful extra trip through India – I'm afraid I will have to adopt a puppy somewhere in the future though to make up for my mistakes. Also, I would like to thank Chris and Fran for roaming the state of Illinois together, similarly I have to thank Marten, Jordi, Emiel, Alicja, Garazi, Fran and Llorenç for a proper detached-from-the-world experience in two RV's through the deserts and nature

parks of California, and I thank Marten, Garazi and Fran for helping me avoid the covid winter by going on a two-week "working" trip to Tenerife. Finally, I would like to thank Garazi for some wonderful getaways in and around the Netherlands and all the time spent together. I hope you don't mind that I will forever wonder about the similarity between trees and standing rocks.

One thing about life at QuSoft that cannot be left unmentioned is the daily habit of playing a game of foosball after lunch, and oftentimes several more later in the afternoon. I have to thank Álvaro for very passionately teaching me the basic strategy of the game, as well as Nikhil for providing by far the best audio feedback to any game I've ever played. Due to foosball, for me *de klassieker* will forever refer to a foosball match of Sander & Mathé vs. Farrokh & me, and I'm afraid I will never again be able to use the words *snake*, *flash* and *German score* in any serious context. Finally, thanks to Tom there is a camera system that tracks the ball, keeps the score, and keeps track of everyone's defense and attack rating. I have supplied a graph of the progression of my rating over time in Figure 0.0.1. Am I the only one wondering if this graph will receive new data points in the years to come?

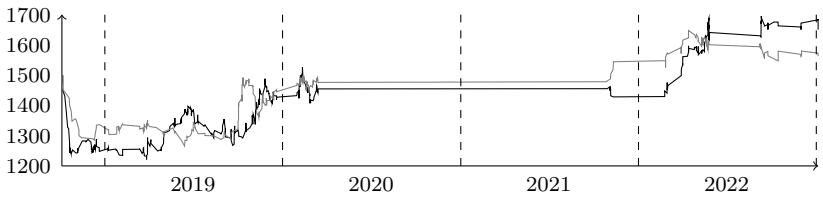



Figure 0.0.1: The progression of my foosball rating throughout my PhD, as of January 16th, 2023, over a total of 730 (rated) games. The black and gray lines are the attack and defense scores, respectively. One can clearly observe the covid period, as well as my internship at IBM.

Then there are still a lot of people of QuSoft that I haven't explicitly thanked, but all of you deserve a mention here. Thanks a lot to Jana for pointing out to me that my dance moves are maybe not on point – I will continue to own it, don't worry – as well as to Yinan for guiding several of us around Shenzhen. I also thank Har&Kar, Jop, Michael, Florian, Jonas, Jeroen, András, Koen, Peter, Joris, Jan, Freek, Yfke, Philip, René, Dmitry, Randy, Galina, Yanlin, Jelena, Dyon, Niels, Bas, Sebastian de Bone, Arie, Chanelle, Crownie, Simona, Mathys, Alex, Chris Majenz, Ido, Davi, Léo, Mert, Yaroslav, Mehrdad, Simon, Seenivasan, John, Adam, Akshay, Victor, Yvonne and Susanne for a wonderful time at QuSoft. Finally, I would like to thank Doutzen in particular for enthusiastically and tirelessly making everyone feel like a part of the family, and organizing many social get togethers, both during and outside of covid. Those were what made it so much fun to be in QuSoft, and even through the covid period we had quite some memorable moments in the digital QuTeas.

Not only was I a part of QuSoft, but I felt very much at home at CWI as well, and became an active member of the activity committee. First, I would like to thank all the people that regularly showed up to the events that we organized, ranging from riddle and casino nights to chess, table tennis and, of course, foosball tournaments. In particular, Dick, pubquizzing will never be the same without you, and a big tanks to Wessel, Sander, Alexander, Pieter, Chris Wesseling, Adrian, Christian, and all the others that joined our events.

Of course I also need to extend a big thanks to all the members of the committe that made organizing all these events so much fun. A big shout out to Giulia, Esteban, Muriel, Ruben, Mark, Hema, Léon, Mathé, Shane, Ludo, Vlad, Nikhil, Max and Sanne, you are all awesome! In particular, I have to thank Isabella for being an extremely devoted and enthusiastic organizer of events. I know I was not always the most organized person when it came to committee things, so I'm very happy that we were able to chair together – I don't think I would have managed on my own. And besides that, I seem to remember that you enjoyed our first house party a lot, don't you agree?

Lastly, I would like to thank the members of the CWI Choir, together with Francien, Jelle, Māris, Joris, Aljosja, Léon, Steven, Ute, Lynda, Stacey, Kareljan and Rianne we performed quite some interesting pieces. In particular, though, I have to thank Doutzen for organizing it and convincing me to join, as I have to say I would have never expected that I would like it as much as I did.

Of course these last four years have not only revolved about work-related things, as there was also a life outside of the realm of PhD. First and foremost I would like to thank my parents and sister for letting me do my thing as much as I like, but simultaneously always being there for me and unconditonally helping out wherever possible. Of course, Maaike, I also have to thank you for organizing all these trips we took – I would have never seen as much of China, Poland, France, Switzerland, Italy, Guatemala and Mexico if it weren't for your planning and dedication.

And finally, we reach the part where I thank all my friends that were there for me throughout my PhD, which is arguably the most important category of all. First, I would like to mention *de Fissagroep* for many barbecues, game nights, chill sessions, and I think we can be quite proud that we managed to maintain biweekly virtual board game nights throughout almost all of the covid period. So a big thanks to Maurice for all the fun we had (*is that chicken curry?*), Erwin for tireless barbecuing, Anne for all the results of ample baking sessions, and Jenny and Mike for all the memorable moments, like climbing Djevelporten in winter and *Arie Pannenkoek*. Special thanks go out to Hugo for playing a major role in organizing so many getaway trips, most notably twice to Norway and twice to Lancaster, and of course for coming all the way to New York to pay me a visit and take me on a weekend trip to Washington DC. Also special thanks go out to Stan for being an awesome person and never allowing any dull moment to happen – and no, I still don't really have an answer to what you should do if you

accidentally break someone else's couch in half. Finally, special thanks go out to Hazel for taking me on an impromptu and epic Curaçao trip and always being down for the most yolo option available – I don't know anyone who I share that sentiment more convincingly with.

Next, I would like to thank Alex, Luke and Mirela for several musical distractions, and on top of that together with Joey and Tom for some good away-from-the-science-world drinks. Besides all that I would like to thank Mathijs, Anda and Erik for some very fun and successful programming sessions – I'm sure that I'm never again going to forget to make my Github repo's private.

Then I would like to thank Tom, David, Mago, Jesse and Ilin for two very memorable trips, to the Czech republic and to Macedonia. I'm still not sure if it was to be expected to find Dutch music in a night club in Ohrid, but I am sure that the song *Gekke boys* will never sound the same to me again.

Of course I will also have to thank all the members of Team Krekhut, Ilin, Jesse and Gera, for joining the Baltic Sea Circle rally. There are too many memories of this trip to list here, but I'm sure that the song *On the road again* will always bring all of them back. And in the unlikely event that you guys are ever considering a surströmming tasting, I'm in.

Next I would like to thank Erik, Matthias and Ilin for the regular drinks sessions. Even though all our lives went into very different directions over the last four years, I thoroughly enjoyed every single time we got back together – and I still cannot think of a better way to spend the night before going to a conference in the US than by getting a few too many beers with you guys.

Lastly, there is no better way to end this section than by thanking my roommates. Through all my adventures, *de Krekhut* has always been the place that I could call home, and that is completely because of the three of you. First, Jesse, we have lived together for almost six years, and even though recently we have mostly been living our separate lives, we still have quite some memorable house parties and board game nights to share. And I will never be able to enjoy the game of *pesten* as much as I did with you.

Then, Jente, I could literally not imagine a nicer roommate to live with. My tea consumption absolutely skyrocketed after you moved in, and it facilitated many very chill moments that allowed for proper unwinding of the craziness of life and work. And, of course, I will not ever forget our impromptu trip to Vienna – if I ever desperately need to figure out where to sleep at an airport, I know who to call.

And finally, Ilin, I recently realized that I know you for well over half of my life. We have had so many chess nights, TrackMania sessions, sporcle adventures, and so many other forms of chilling that I cannot list them all. I hope you don't mind all the random WhatsApp messages I sent you about anything and everything, especially all the audio fragments in the middle of the night. A big thanks to all the fun we had, and whenever I see the wind blow through the trees, I will forever wonder whether this is the correct meaning of *шушка молика*.

As you can probably tell by the length of this acknowledgment section, I thoroughly enjoyed my time as a PhD student. Even though I tried in this text, I cannot put into words how fortunate I feel to have experienced all this. Ultimately, all I can conclude with is that I am very grateful to all of you who were a part of this journey, and that I hope I have been able to bring some happiness to you somewhere along the way as well.

*Het is mooi geweest, op naar iets nieuws.*

January 16th, 2023.

# Chapter 1

# Introduction

The ability to perform computations has been a catalyst in many developments. Numerous accounts indicate that a computational device called an *abacus* was used already in ancient times to perform simple arithmetic, for instance for accounting purposes in ancient Greece and the Roman empire, as well as during the Chinese dynasties. Later, during the second world war, the development of the *electrical computer* played a crucial role in the ability of the allied forces to decrypt communication between German commanders, and over the decades afterwards, similar computational devices revolutionized society with the advent of the information age. Even more recently, the development of more efficient and more powerful *graphical processing units* have opened up the ability for a boom in artificial intelligence, with countless emerging applications ranging from self-driving cars to computer-aided diagnosis.

Along with the aforementioned physical advances in the ability to perform computations, came a deeper understanding of the complexity of computational problems of interest. At first, this understanding mainly manifested in the development of *algorithms* that solve these problems, i.e., explicit descriptions of sequences of operations to be performed on these computational devices. For instance, *Euclid's algorithm* describes how one can calculate the greatest common divisor of two integers using an abacus. Similarly, *Dijkstra's algorithm* can be used to find the shortest distance between two cities in a road network using a conventional computer. And finally, the *backtracking algorithm*, which tells us how we can optimize the parameters of a neural network, has a particularly efficient implementation on graphical processing units.

In parallel to the aforementioned development of algorithms came a better understanding of the hardness of computational problems of interest. For instance, Alan Turing exhibited a problem, known as the halting problem, that cannot be solved using *any* algorithm run on a conventional computer [Tur37]. Similarly, a large class of problems, i.e., the NP-complete problems, has been discovered with the conjectured property that any algorithm that solves them on a conventional

computer necessarily requires a number of steps that is exponential in the size of the input [Coo71; Lev73].

When we specify a computational problem, a computational device, and a way of measuring the cost of an algorithm run on this device, then we can look for an *optimal* algorithm that solves the given problem on the given device with minimal cost. Surprisingly, one can show that such an optimal algorithm does not always exist [Blu67], but if it does then its cost is referred to as the *computational complexity* of the given problem on the given device. A large area within computer science is dedicated to characterizing these computational complexities of all sorts of problems, and this field of research is called *complexity theory*.

Over the past decades, the idea of a *quantum computer* rapidly caught traction in the scientific community. Rather than *classical computers* whose physics can be described by classical mechanics, which includes laptops, graphical processing units and modern-day supercomputers, the proposed quantum computers rely on the principles of quantum mechanics to perform computations. The idea arose during the cold war independently on both sides of the iron curtain, by Yuri Manin in 1980 in the Soviet Union [Man80], and by Paul Benioff in the USA [Ben80]. Later, the idea was popularized in the west by Richard Feynman, who asserted that in order to simulate complicated quantum mechanical systems, we need a computational device that natively supports quantum operations.

Broadly speaking, the study of performing computations on such quantum devices is called *quantum computing*. Within this field, one can investigate similar questions about the complexity of computational problems of interest, under all kinds of cost models. We typically refer to the complexity of a computational problem on a quantum computer as the *quantum computational complexity* of the given problem.

In some cases, there are surprisingly big discrepancies between the classical and quantum complexities. Probably the most well-known result of this type is Shor's algorithm [Sho97], which describes a way to find divisors of a composite integer with exponentially fewer elementary quantum operations than the number of steps that the current best-known algorithm on a classical computer performs to solve the same problem. Similarly, a wide range of search problems feature a quadratic cost reduction between the classical and quantum setting due to Grover's algorithm [Gro96].

Broadly speaking, the field of quantum computing lies at the intersection between physics, mathematics, and computer science. In order to grasp the underlying theory that governs the computational foundations of a quantum computer, a thorough understanding of quantum mechanics is of paramount importance. Additionally, to analyze the behavior of the newly-developed quantum algorithms, a fair amount of non-trivial mathematics is typically involved. Finally, many of the algorithmic techniques used to design quantum algorithms find their origin in (classical) computer science.

In this thesis, we make progress on understanding the quantum complexity

of several computational problems. In the process, we develop new algorithmic techniques, as well as new proof techniques to arrive at hardness results in the quantum setting.

This thesis is the result of a four-year PhD program, conducted from 2018 to 2022 at QuSoft, which is a collaboration between University of Amsterdam (UvA) and the Center of Mathematics and Computer Science (CWI).

## 1.1 Overview

This thesis is structured as follows. In Chapter 2, we define some notation, and we explain the computational model we will be presenting our work in. The idea is that the quantum devices that are being designed and built at the moment, can in the future efficiently implement the algorithms that are defined in the computational model we describe here. Subsequently, we recall a couple of well-known algorithmic primitives that we will be using throughout the remainder of this thesis to build new algorithms upon.

After that the thesis is subdivided into two parts. In Part I, we introduce several quantum algorithms that solve separate computational problems of interest. The common theme among all of these is that they estimate some mathematical object up to a given accuracy, and typically the quantum algorithm's cost has a better dependence on the desired accuracy than its classical counterpart.

We now individually describe the quantum algorithms in Part I in more detail. In Chapter 3, we introduce a quantum algorithm that solves the *quantum mean estimation problem*, i.e., that computes the mean of a multivariate random variable $X$ taking values in $\mathbb{R}^d$. In our model, we assume to have access to a quantum routine that prepares a coherent superposition over the underlying events, and we measure the cost of our algorithm by counting the number of inverse and/or controlled calls to this routine. This model closely mimics the classical setting in which we count the sample complexity of a random process. The crucial observation that we make in this chapter is that one can only obtain a quantum speed-up if the number of calls $n$ to the oracle exceeds the dimension of the random variable $d$, i.e., we have a quantum speed-up if and only if $n > d$.

In Chapter 4, we describe a quantum algorithm that solves the *quantum state-tomography problem*, i.e., that finds an approximation to a state $\rho$ given access to a unitary that prepares a purification of it. We assume that $\rho \in \mathbb{C}^{d\times d}$, and that we know a priori that the rank of $\rho$ is at most $r \leq d$. In the model we consider, we measure the cost of our algorithm as the number of inverse and/or controlled queries made to the state-preparation unitary. We characterize the quantum complexity of this problem to be $\widetilde{\Theta}(rd/\varepsilon)$ queries to the state-preparation oracle, by explicitly describing a quantum algorithm and subsequently showing a matching lower bound.

In Chapter 5, we describe a quantum algorithm that solves the *partition func-*

*tion estimation problem*, i.e., it finds an approximation to the partition function at a given inverse temperature. The technique employed here is called simulated annealing, and the approach outlined in this chapter improves over a long line of previous work. In the process, we develop several new algorithmic techniques, and showcase our result with a wide range of applications.

Subsequently, in Part II, we present the *span program formalism*, which is a particular technique that can be used to design quantum algorithms. Parts of the theory developed in this part are present in various different parts of the scientific literature, so the main contribution of this part is to provide a self-contained and unified introduction into the existing theory. On top of that, we extend the framework in several new directions, and improve some of the existing constructions along the way. Throughout, we focus on the special case where the function being computed is a decision problem, i.e., its output is a single bit.

In Chapter 6, we provide a self-contained introduction to the formalism. The central objects are called *span programs*, and the formalism provides a way to constructively convert such a span program into a quantum algorithm. We define span programs slightly more generally compared to the existing literature, and spend significant effort in explaining formalism in a pictorial and intuitive manner. We also show how these objects relate to feasible solutions to a semidefinite program known as the adversary bound, that characterizes the quantum query complexity of a boolean function.

In Chapter 7, we discuss several composition results. Broadly speaking, these constructions allow for taking many smaller span programs, and combining them into bigger ones that can potentially solve much more complicated problems. We present two types of composition results, *logical* and *graph compositions*, and we show that the latter is a generalization of the former. We also discuss that such compositions are much more efficient when performed on the span program level, compared to when one would perform them directly on the quantum algorithm level. Additionally, we discover a new property of span programs that neatly relates the operational properties of span programs and their logical compositions through the theory of complex analytic functions. Finally, we show how graph composition can be used to turn a classical algorithm into a quantum one, and sometimes obtain a quantum speed-up in the process.

In Chapter 8, we consider a slight generalization of the span program framework known as *approximate span programs*. We use the theory developed in the previous chapters to devise an improved quantum algorithm that evaluates such approximate span programs, and we show its optimality up to polylogarithmic factors. Finally, we show how approximate span programs are complete for a specific class of quantum algorithms.

Finally, in Chapter 9, we end this thesis with some final remarks and potential topics of future research.

## 1.2 Relation to the literature

The following papers form the basis for this thesis:

- The following paper forms the basis for Chapter 3:

  [CHJ22] Arjan Cornelissen, Yassine Hamoudi, and Sofiène Jerbi. "Near-optimal Quantum algorithms for multivariate mean estimation". In: *54th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2022)*. ACM, 2022, pp. 33–43. Presented at: 25th Annual Conference on Quantum Information Processing (QIP 2022). `arXiv:2111.09787`

- The following paper forms the basis for Section 7.3:

  [CMP22] Arjan Cornelissen, Nikhil S. Mande, and Subhasree Patro. *Improved Quantum Query Upper Bounds Based on Classical Decision Trees*. 2022. Presented at: 17th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2022). Accepted to: 42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2022). `arXiv:2203.02968`

- The following paper forms the basis for Chapter 4:

  [vACG+22] Joran van Apeldoorn, Arjan Cornelissen, András Gilyén, and Giacomo Nannicini. *Quantum tomography using state-preparation unitaries*. 2022. Accepted to: Symposium on Discrete Algorithms (SODA 2023) and the 26th Conference on Quantum Information Processing (QIP 2023). `arXiv:2207.08800`

- The following paper forms the basis for Chapter 5:

  [CH22] Arjan Cornelissen and Yassine Hamoudi. *A Sublinear-Time Quantum Algorithm for Approximating Partition Functions*. 2022. Accepted to: Symposium on Discrete Algorithms (SODA 2023) and the 26th Conference on Quantum Information Processing (QIP 2023). `arXiv:2207.08643`

We also provide a summary of the results obtained in the following paper, but we don't include all the details.

- A summary of the following paper is provided in Section 8.3:

[CJO+20] Arjan Cornelissen, Stacey Jeffery, Māris Ozols, and Alvaro Piedrafita. "Span Programs and Quantum Time Complexity". In: *Proceedings of the 45th International Symposium on Mathematical Foundations of Computer Science, (MFCS 2020)*. Vol. 170. 2020, 26:1–26:14. arXiv:2005.01323

On top of that, several components of the span program part are new research, but are not contained in the above-mentioned papers. Specifically, we mention the following new results here:

1. We discover a new property of span programs called the *characteristic function*, and show how it can be used to turn span programs into approximation algorithms in Section 7.1.3.
2. We generalize the existing *st*-connectivity construction for span programs to a composition result in Section 7.2.
3. We present an algorithm that evaluates approximate span programs in Section 8.2. It improves over the best-known approximate span program algorithm and achieves tight complexity bounds, up to polylogarithmic factors.

Even though these last parts are not yet turned into self-contained papers, it is my expectation that they will form the basis of future publications.

Additionally, the following papers also appeared over the course of my PhD, but they are not part of this PhD thesis:

[Cor19] Arjan Cornelissen. *Quantum gradient estimation of Gevrey functions*. 2019. arXiv:1909.13528

[CBG21] Arjan Cornelissen, Johannes Bausch, and András Gilyén. *Scalable Benchmarks for Gate-Based Quantum Computers*. 2021. arXiv:2104.10698

[CJ21] Arjan Cornelissen and Sofiene Jerbi. *Quantum algorithms for multivariate Monte Carlo estimation*. 2021. arXiv:2107.03410

[CMO+21] Arjan Cornelissen, Nikhil S. Mande, Maris Ozols, and Ronald de Wolf. *Exact quantum query complexity of computing Hamming weight modulo powers of two and three*. 2021. arXiv:2112.14682

In all papers mentioned, the amount of work was divided equally among all authors.

# Chapter 2

# Preliminaries

In this chapter, we introduce the necessary background information that the rest of the thesis is built upon. We start by introducing some notational conventions in Section 2.1. After that, we describe the basics of quantum mechanics in Section 2.2. Next, we explain how the theory of quantum computing fits into the more general framework provided by quantum mechanics, in Section 2.3. Finally, we present some well-known algorithmic primitives that are used in several places throughout this thesis in Section 2.4.

## 2.1 Notation

We start by listing some notational conventions that we will be using throughout this thesis. For all natural numbers $n \in \mathbb{N}$, we denote $[n] = \{1, \ldots, n\}$, and we let $[n]_0 = [n] \cup \{0\}$. For all expressions $a, b$, we define the *Kronecker delta* as

$$\delta_{a,b} = \begin{cases} 1, & \text{if } a = b, \\ 0, & \text{otherwise.} \end{cases}$$

We use specific notation to perform truncations, i.e., for $a, b, x \in \mathbb{R}$ with $a < b$, we let

$$[\![x]\!]_a^b = \begin{cases} x, & \text{if } a \leq x \leq b, \\ 0, & \text{otherwise.} \end{cases} \tag{2.1.1}$$

Setting the value to 0, rather than the closest endpoint, is mostly done to simplify the exposition of the analyses later on. One could also set $x$ to the closest endpoint, i.e., let $[\![x]\!]_a^b = a$ whenever $x < a$, and similarly when $x > b$.

We let $\mathbb{R}_{>0} = \{x \in \mathbb{R} : x > 0\}$ and $\mathbb{R}_{\geq 0} = \{x \in \mathbb{R} : x \geq 0\}$ be the sets of positive and non-negative reals, respectively. Vectors in $\mathbb{R}^d$ are often represented in boldface and we typically assume the indexing to these vectors to be 1-based, i.e., $(x_1, \ldots, x_d) = \mathbf{x} \in \mathbb{R}^d$.

*Hilbert spaces* are complete inner product spaces. In this text, we assume all Hilbert spaces to be finite-dimensional and over the complex field $\mathbb{C}$, unless otherwise stated. As is customary in quantum mechanics, we assume inner products to be linear in the second variable, and antilinear in the first.

Unit vectors in Hilbert spaces are referred to as *states* for short. We denote them with the *ket*-notation, i.e., $|\cdot\rangle$. Therefore, unless stated otherwise, we assume in particular that every ket denotes a *unit* vector, i.e., a vector with norm 1. If the Hilbert space $\mathcal{H}$ is defined explicitly over some finite basis set $\Omega$, i.e., $\mathcal{H} = \mathbb{C}^{\Omega}$, then we associate the corresponding standard basis vectors with the notation $|\omega\rangle$, for all $\omega \in \Omega$. For any $n \in \mathbb{N}$, we let $\mathbb{C}^n$ denote $\mathbb{C}^{[n-1]_0}$, which in turn is equal to $\mathrm{Span}\{|j\rangle : j \in [n-1]_0\}$. In particular, this means that $\mathbb{C}^2 = \mathrm{Span}\{|0\rangle, |1\rangle\}$.

We use the *bra*-notation, i.e., $\langle\cdot|$, to denote vectors in the dual of a Hilbert space. Formally, a bra is a linear functional on the Hilbert space, defined by taking the inner product. That is, a dual vector $\langle\psi| : \mathcal{H} \to \mathbb{C}$ evaluated at a vector $|\phi\rangle \in \mathcal{H}$ equals the inner product between $|\psi\rangle$ and $|\phi\rangle$, denoted by $\langle\psi|\phi\rangle$. The latter is referred to as a *braket*.

Let $\{|e_1\rangle, \ldots, |e_n\rangle\}$ and $\{|f_1\rangle, \ldots, |f_m\rangle\}$ be orthonormal bases for two Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively. Then, we denote the *tensor product* $\mathcal{H}_1 \otimes \mathcal{H}_2$ to be the space that is spanned by tuples of the basis states, i.e., $\mathrm{Span}\{|e_j\rangle \otimes |f_k\rangle : j \in [n], k \in [m]\}$. The resulting space does not explicitly depend on the choice of basis, and we can easily observe that the Hilbert space dimension of the tensor product is the product of the Hilbert space dimensions, i.e., $\dim(\mathcal{H}_1 \otimes \mathcal{H}_2) = \dim(\mathcal{H}_1)\dim(\mathcal{H}_2)$.

All spaces of linear operators between two Hilbert spaces, we endow with the *operator norm* by default. For a linear operator $A$, we denote its *spectrum* by $\sigma(A) \subseteq \mathbb{C}$. We denote its adjoint by $A^\dagger$, and we say that an operator is *Hermitian* if it equals its adjoint. For Hermitian operators, we recall that their spectrum is contained in the reals, and that the operator norm is the same as the maximum absolute value of its eigenvalues. A Hermitian operator is *positive semidefinite* if all its eigenvalues are non-negative, and it is *positive definite* if all its eigenvalues are positive. If $A$ is positive semidefinite, or PSD for short, we write $A \succeq 0$, and if $A$ is positive definite, or PD, we write $A \succ 0$.

Following common conventions in the computer science domain, we take all logarithms to be base 2, and we denote the *natural logarithm*, i.e., the logarithm base $e$, by ln.

Let $f, g : \mathcal{D} \to \mathbb{R}$, with $\mathcal{D} \subseteq \mathbb{R}^d$. Suppose that we can find constants $C \geq 0$ and $C_1, \ldots, C_d \in \mathbb{R}$ such for all $(x_1, \ldots, x_d) = \mathbf{x} \in \mathbb{R}^d$,

$$|f(\mathbf{x})| \leq C|g(\mathbf{x})|, \tag{2.1.2}$$

if there exists a $j \in [d]$ such that $x_j \geq C_j$. Then, we write $f = \mathcal{O}(g)$. This is known as the big-$\mathcal{O}$-notation, and it is important to stress that the notation demands that the bound in Equation (2.1.2) holds whenever we take the limit to $\infty$ in *any* parameter, even when keeping the others fixed.

Oftentimes, we will not define the above functions $f$ and $g$ very formally. As an example, suppose that we write

$$\frac{\sin^2(\varepsilon)}{t} = \mathcal{O}\left(\frac{\varepsilon^2}{t}\right), \qquad (\varepsilon \downarrow 0, t \to \infty). \tag{2.1.3}$$

What we formally mean is that we consider the two functions $f, g : \mathbb{R}^2_{>0} \to \mathbb{R}$, defined as

$$f(x_1, x_2) = \frac{\sin^2(1/x_1)}{x_2}, \qquad \text{and} \qquad g(x_1, x_2) = \frac{1/x_1^2}{x_2},$$

and we claim that $f = \mathcal{O}(g)$ as defined above. The choice of replacing $\varepsilon$ by $1/x_1$ is justified by the fact that $x_1 \to \infty$ indeed coincides with $\varepsilon \downarrow 0$, as indicated in Equation (2.1.3). Similarly, $\lambda \uparrow 1$ could get replaced by $1 - 1/x_j$, for instance, but other choices are also possible. Oftentimes, we will also drop the limiting regimes that we mean in our usage of the big-$\mathcal{O}$-notation, when they are clear from context.

Next, we use the big-$\widetilde{\mathcal{O}}$-notation if we want to additionally suppress polylogarithmic factors. Formally, if we have two functions $f, g : \mathcal{D} \to \mathbb{R}$, with $\mathcal{D} \subseteq \mathbb{R}^d$, then we say that $f = \widetilde{\mathcal{O}}(g)$, if we can find constants $p_1, \ldots, p_d > 0$ such that

$$f(\mathbf{x}) = \mathcal{O}\left(g(\mathbf{x}) \cdot \prod_{j=1}^{d} \log^{p_j}(x_j)\right),$$

where $x_j$ is the $j$th entry of a vector $\mathbf{x} \in \mathbb{R}^d$. As an example, we observe that

$$x_1 \log(x_1 x_2) + x_2 \log(x_1) = \widetilde{\mathcal{O}}(x_1 + x_2).$$

Sometimes one cannot infer from the notation directly what space $\mathbb{R}^d$ we mean when we use the big-$\widetilde{\mathcal{O}}$-notation. For instance, if we write

$$\frac{\log(1/\delta)}{\varepsilon} = \widetilde{\mathcal{O}}\left(\frac{1}{\varepsilon}\right),$$

in the limit where $\varepsilon, \delta \downarrow 0$, then from the expression on the right-hand side it is not clear that there is a logarithmic factor in $1/\delta$ that has been suppressed. Therefore, if there can be any confusion about the parameters involved, we explicitly mention the parameters in which polylogarithmic factors have been suppressed.

The big-$\mathcal{O}$-notation is useful for providing upper bounds on functions in some limits of the parameters' range. Sometimes, we want to similarly provide lower bounds, for which we employ the big-$\Omega$-notation. We simply say that $f = \Omega(g)$ if and only if $g = \mathcal{O}(f)$, and similarly $f = \widetilde{\Omega}(g)$ if and only if $g = \widetilde{\mathcal{O}}(f)$. Finally, if we have both, i.e., $f = \mathcal{O}(g)$ and $f = \Omega(g)$, then we say that $f = \Theta(g)$, and similarly with $\widetilde{\Theta}$.

We conclude this section with the formal definition of the *cyclic distance*.

**2.1.1.** DEFINITION (Cyclic distance). Let $a, b, p \in \mathbb{R}$. We define the *cyclic distance between $a$ and $b$ with period $p$* as:

$$\text{cyclic-dist}_p(a, b) = \min\{|a - b + zp| : z \in \mathbb{Z}\}. \quad \blacktriangleleft$$

We can think of the cyclic distance as representing the shortest distance between two points when they are placed on a circle of length $p$. If we omit $p$, we assume that $p = 1$.

## 2.2 Basics of quantum mechanics

Quantum mechanics is a theory that describes the laws that govern the physical world around us. It provides this description through four postulates, which are referred to as the *postulates of quantum mechanics*. We briefly recap them here, and we refer the interested reader to [NC00] for a more elaborate introduction into quantum mechanics.

**Postulate I: State space** The first postulate of quantum mechanics asserts that every physical system has an associated *state space*, which is simply a Hilbert space. By physical system we mean *any* set of particles in the real world, e.g., your neighbor's bicycle, or a cell in your body. A *quantum state*, or *state* for short, of a physical system is a unit vector in its state space. It captures all the properties of the physical system, i.e., the energy levels of the electrons in its atoms, the position of all its particles in the universe, etc.

As far as we know, there is no reason why we could not use quantum mechanics to describe everyday phenomena. However, if we tried to describe the quantum state of an everyday object, like your neighbor's bike, we would end up with a very unwieldy object in an extremely large Hilbert space. Consequently, quantum mechanics finds its application predominantly in describing very small physical systems, on the level of atoms and molecules.

**Postulate II: System composition** The second postulate of quantum mechanics states that the state space of a composite physical system is the tensor product of the individual systems. Specifically, suppose that we have two physical systems with corresponding state spaces $\mathcal{H}_1$ and $\mathcal{H}_2$. Then, the state space of the *combined* physical system is $\mathcal{H}_1 \otimes \mathcal{H}_2$.

An interesting observation to make here is that if we describe the state of a composite physical system, which is simply a unit vector in $\mathcal{H}_1 \otimes \mathcal{H}_2$, then this state cannot always be written as a tensor product $|h_1\rangle \otimes |h_2\rangle$, with $|h_1\rangle \in \mathcal{H}_1$ and $|h_2\rangle \in \mathcal{H}_2$. Thus, there is a very important conceptual distinction to make. If we can write the state of a composite system as a tensor product of states of the individual systems, then we say that this state is a *separable state*, or *product*

*state*. On the other hand, if the state of a composite system is not separable, then we say that it is *entangled*, in which case the quantum state of either of its individual subsystems is not a well-defined mathematical object.[1]

**Postulate III: Time evolution** The third postulate of quantum mechanics postulates that the time-evolution of a physical system's state can be modeled as a unitary operation acting on its state space. The quantum state of a physical system prior to the operation is then related to its posterior state by a single application of this unitary operator.

It is important to note here that this postulate inherently assumes that the physical system is isolated, i.e., that during the time evolution it does not interact with other physical systems. Indeed, we have already seen in the composition postulate that the quantum state of a composite system can become entangled, in which case the quantum states of the two individual physical systems becomes an ill-defined object. If we consider a composite system, with state spaces $\mathcal{H}_1$ and $\mathcal{H}_2$, and we apply a unitary operation $U$ to the first system, and do not perform any operation on the second, then the corresponding unitary operation on the composite system is $U \otimes I$.

**Postulate IV: Measurement** The final postulate of quantum mechanics describes how we can extract information from a quantum state. To that end, we let $\mathcal{H}$ be the state space of some physical system, and we let $\Omega$ be a set, referred to as the set of *measurement outcomes*. To every outcome $\omega \in \Omega$, we associate a subspace $S_\omega \subseteq \mathcal{H}$, and we demand that all these subspaces form an orthogonal decomposition of $\mathcal{H}$, i.e.,

$$\mathcal{H} = \bigoplus_{\omega \in \Omega} S_\omega.$$

Then, the set $\Omega$ with its associated subspaces defines a *measurement operation*. If the quantum state of the system is $|\psi\rangle \in \mathcal{H}$, then for every $\omega \in \Omega$, the probability of obtaining measurement outcome $\omega$ is given by

$$\mathbb{P}(\omega) = \left\| \Pi_{S_\omega} |\psi\rangle \right\|^2,$$

where $\Pi_{S_\omega}$ is the operation that projects onto the subspace $S_\omega$. Furthermore, if the measurement outcome is $\omega$, then the post-measurement quantum state of the system is

$$|\psi'\rangle = \frac{\Pi_{S_\omega} |\psi\rangle}{\sqrt{\mathbb{P}(\omega)}}.$$

We say that the quantum state has *collapsed* to the subspace $S_\omega$.

[1] If instead of state vectors, as we do here, one uses density matrices to describe quantum states, then one can can always describe the reduced density matrix on a subsystem. We do not elaborate on that here, and refer the interested reader to [NC00].

The above definition teaches us something very important: performing a measurement on a physical system inherently affects the state. Indeed, the resulting state will always be in one of the subspaces $S_\omega$, and therefore if we started with a quantum state that had overlap with multiple subspaces $S_\omega$, it will definitely be altered by the measurement operation itself. Indeed, somewhat surprisingly, it turns out that this is indeed an accurate description of the real world, i.e., at the atomic level it is impossible to obtain information about the system without affecting it. The most well-known example of this is Heisenberg's uncertainty principle [Hei27], that asserts that one cannot measure the position of a particle without affecting its momentum, and vice versa.

In the special case where the set of measurement outcomes is contained in the reals, i.e., $\Omega \subseteq \mathbb{R}$, we can very neatly combine the set $\Omega$ and its associated projectors into a single operator on the state space, i.e., we can write

$$O = \sum_{\omega \in \Omega} \omega \Pi_{S_\omega}.$$

The resulting operator $O$ is a Hermitian operator on the state space $\mathcal{H}$, and it is referred to as an *observable*. Conversely, we also observe that the eigendecomposition of any Hermitian operator uniquely defines a set of outcomes $\Omega \subseteq \mathbb{R}$ with corresponding subspaces, and as such there exists a one-to-one correspondence between observables and Hermitian operators on the state space.

As a final remark, we state that the *expectation* of an observable measurement can be very succinctly written. Suppose that we have a quantum system in state $|\psi\rangle$, and a measurement operation described by an observable $O$, then the expectation of the outcome is given by $\langle\psi| O |\psi\rangle$. It is a nice exercise for the reader to verify that this is indeed the case, given the measurement postulate that we defined here.

## 2.3 Basics of quantum computing

In this section, we shift gears a little bit and narrow our scope to quantum computing. That is, we explain how the principles of quantum mechanics give rise to a model of computation.

**Qubits** The fundamental carrier of information in quantum computing is a *qubit*, i.e., a physical system whose state space is $\mathbb{C}^2 = \text{Span}\{|0\rangle, |1\rangle\}$. Qubit is short for *quantum bit*, and it is the quantum equivalent of a *bit*, or binary digit, in classical computing. Indeed, we see that a qubit can be in the states $|0\rangle$ or $|1\rangle$. In addition, its state can also be a linear combination of these two, i.e., $\alpha |0\rangle + \beta |1\rangle$ with $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. We refer to such a state as a *superposition* over $|0\rangle$ and $|1\rangle$. If $\alpha = \beta$, we say that it is a *uniform superposition*.

The third postulate of quantum mechanics tells us that every operation we can perform on a qubit is a unitary acting on $\mathbb{C}^2$. We refer to these unitaries as *quantum gates*, and we define a few particular elementary operations, which we refer to as the *elementary single-qubit gates*. To that end, we associate $|0\rangle$ and $|1\rangle$ with the vectors $[1 \;\; 0]^T$ and $[0 \;\; 1]^T$, so that we can conveniently write these operations as $2 \times 2$-matrices. They are listed in Table 2.1.

| Name | Symbol | Matrix |
|---|---|---|
| Pauli-$X$ | $X$ | $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ |
| Pauli-$Y$ | $Y$ | $\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$ |
| Pauli-$Z$ | $Z$ | $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ |
| Hadamard | $H$ | $\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ |
| Phase gate | $S$ | $\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$ |
| $T$-gate | $T$ | $\begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{\pi i}{4}} \end{bmatrix}$ |

Table 2.1: Elementary single-qubit quantum operations.

We also define a very simple measurement. We let $\Omega = \{0, 1\}$, $S_0 = |0\rangle\langle 0|$ and $S_1 = |1\rangle\langle 1|$. We refer to the resulting measurement operation as the *standard basis measurement*. It returns 0 when we are in state $|0\rangle$, and 1 when we are in state $|1\rangle$, and hence it allows us to recover a classical bit of information from the quantum state stored in a qubit. If the qubit state is some superposition over $|0\rangle$ and $|1\rangle$, then the measurement outcome will be probabilistic, as described by the measurement postulate.

When we combine several qubits, we obtain a physical system with a larger dimension. As such, we can also apply larger unitary operations to these systems. For instance, if we have a system containing two qubits, we can perform a controlled-NOT operation on them, denoted by CNOT, and defined as

$$\text{CNOT} = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

The above operation can be interpreted as a conditional operation: if the first qubit is in state $|0\rangle$, then we apply the identity operation to the second qubit, and if the first qubit is in state $|1\rangle$, we apply an $X$-operation to the second qubit. Furthermore, note that the $X$-operation indeed plays the role of the NOT-operation, because it maps $|0\rangle$ to $|1\rangle$ and vice versa.

The state space of an $n$-qubit system is $(\mathbb{C}^2)^{\otimes n}$, which has $2^n$ dimensions. To every integer $j \in [2^n - 1]_0$, with binary representation $j = j_1 j_2 \dots j_n$, we can associate the $n$-qubit standard basis vector $|j\rangle = |j_1\rangle \otimes \cdots \otimes |j_n\rangle$. As such, a computational basis measurement on an $n$-qubit system is a measurement with $2^n$ projections onto these standard basis states, and corresponding measurement outcomes in $[2^n - 1]_0$.

To be able to reason about qubits and their states more easily, we typically group several qubits into a *register*, and then talk about the state of this register rather than the state of the individual qubits themselves. For instance, an *$n$-qubit register* is the system we used in the previous paragraph, which has state space $(\mathbb{C}^2)^{\otimes n}$.

Additionally, we usually take the liberty to think of the state space of a register not necessarily as a tensor products of qubits, but as an arbitrary Hilbert space, and our operations merely as unitaries on this Hilbert space. This means that if we want to actually implement our operations on a physical device, we would need to embed the Hilbert space into the state space of an $n$-qubit register, and subsequently we would need to decompose our unitaries into elementary single-qubit and two-qubit operations. Oftentimes, we will not give this embedding of the Hilbert space and decomposition of the unitary explicitly, but merely rely on results that this can in principle always be done, like [Kit97], or the construction presented in [NC00, Chapter 4].

**Quantum algorithms** A *quantum computation*, or *quantum algorithm* is a sequence of unitary operations and measurements performed on a set of qubits. For the purposes of this thesis, we will always assume that the initial state of the system is the all-zeros state, i.e., every qubit is initialized to the state $|0\rangle$ at the start of the computation.

In this thesis, we make a distinction between a *quantum circuit* and a *quantum algorithm*. By a *quantum circuit*, we mean a sequence of unitary operations applied to a sequence of qubits. Hence, a quantum circuit does not consist of any measurement operations. A *quantum algorithm*, on the other hand, is allowed to perform unitary operations, as well as measurement operations on a set of qubits.

The benefit of making this distinction is that quantum circuits have a few properties that don't in general hold true for quantum algorithms. First, since a circuit consists of just unitary operations, its overall action can be described as a unitary as well. We say that the circuit *implements* this overall unitary operation.

Furthermore, all unitaries are *invertible*, and as such a quantum circuit admits an operational inverse. That is, we can implement the inverse of a quantum circuit by reversing the order of all the gates that make up the circuit, and then also inverting all the gates individually as well. One can easily verify that if we run a quantum circuit, followed by its inverse, then we end up implementing the identity operation, i.e., we *do nothing*.

For many computational problems, we naturally have an input that we would like to feed into an algorithm that solves the computational problem for that particular input. For instance, if we want to add two integers $a, b \in \mathbb{N}$, then we typically design an algorithm that can add any two integers, and then feed it the integers $a$ and $b$ as input. Similarly, if we consider Dijkstra's algorithm that computes the smallest distance between two cities in a road network, the input to the algorithm is a description of the cities in the network, and the roads connecting them.

Since we start our quantum computation in the all-zeros state, one might wonder how we supply this input to the quantum algorithm in this computational model. The typical assumption is to provide a circuit that explicitly depends on the input, known as an *oracle*, that the algorithm can *call* or *query*. It is important to realize that oracles can themselves call other oracles, and hence this model allows for nesting several layers of computation, which mimics the setting of classical programming.

To give an example of a quantum algorithm that uses such an oracle, we present *Deutsch's algorithm*. Given a function $f : \{0,1\} \to \{0,1\}$, it decides whether $f(0) = f(1)$. We can encode this function in an oracle $O_f$, acting on one qubit, that for all $b \in \{0,1\}$ acts as

$$O_f : |b\rangle \mapsto (-1)^{f(b)} |b\rangle .$$

Now, the quantum algorithm proceeds as follows. We use a single qubit, and then apply the Hadamard gate to obtain the uniform superposition, i.e., the state

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle).$$

Then, we apply the oracle, which by linearity will map the state to

$$\frac{1}{\sqrt{2}}((-1)^{f(0)} |0\rangle + (-1)^{f(1)} |1\rangle) = \frac{(-1)^{f(0)}}{\sqrt{2}}(|0\rangle + (-1)^{f(0)\oplus f(1)} |1\rangle),$$

where by $a \oplus b$ we denote binary addition, i.e., $a \oplus b = 0$ if and only if $a = b$, and 1 otherwise. Next, we apply the Hadamard gate again, and we can easily verify that we end up in the state

$$(-1)^{f(0)} |f(0) \oplus f(1)\rangle .$$

Thus, we are either in a scalar multiple of the state $|0\rangle$, if $f(0) = f(1)$, or $|1\rangle$ if $f(0) \neq f(1)$. Hence, if we now perform a computational basis measurement on our qubit, we obtain 0 when $f(0) = f(1)$, and 1 when $f(0) \neq f(1)$ with certainty.

Note that in the above algorithm, we only made a single query to the input, i.e., to the oracle $O_f$. This is remarkable, since one could naively expect that comparing two function values, i.e., $f(0)$ and $f(1)$, would also require two calls

to an operation that computes the function. Thus, here we see the first signs of quantum effects coming into play, i.e., we managed to solve this problem with fewer queries than we would need if we were to solve the same problem in a similar setting on a classical computer. More precisely, we observe the *quantum interference* effect, where a positive and a negative contribution to an amplitude result in it vanishing altogether.

If we have access to some oracle $O_x$, encoding some input $x$, that acts on a Hilbert space $\mathcal{H}$, then it will often be very convenient to assume that we also have access to circuits that derive from $O_x$. For instance, the *inverse* of $O_x$ is the circuit $O_x^\dagger$ that acts on $\mathcal{H}$ as well. Furthermore, the *controlled* version of $O_x$ is the circuit that acts on $\mathbb{C}^2 \otimes \mathcal{H}$ as $|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes O_x$.

Throughout this thesis we will always assume that if we have access to $O_x$, then we also have access to its inverse, its controlled version, and the controlled version of its inverse. This assumption is not very restrictive once one observes that it is recursively satisfied. That is, suppose that $O_x$ itself queries some input oracle $O'_y$. Then, having access to the inverse of $O'_y$, and the controlled version of $O'_y$ and its inverse, can easily be seen to imply that we also have access to $O_x$, its inverse, and the controlled versions. Thus, once we satisfy this assumption at the lowest level, i.e., where the input circuit is simply a collection of elementary gates, it automatically carries over through all consecutive layers of nesting that we encounter.

Finally, we remark that since the outcome of a quantum measurement is inherently probabilistic, we sometimes end up implementing algorithms that output the correct answer only with high probability. Specifically, when an algorithm outputs the correct answer to the computational problem with certainty, we say that the algorithm is *exact*. On the other hand, if it outputs the correct answer with probability at least $2/3$, then we say that the algorithm solves the computational problem *with bounded error*.

**Cost** The cost of a quantum algorithm can be measured in many different ways. One relatively simple way is to count the number of elementary operations that are performed throughout the execution of the algorithm. This cost measure we refer to as the *gate cost*, or *time cost*. In line with the discussion in the introduction, we now define the *gate complexity*, or *time complexity* of a particular problem to be the minimal gate cost of an algorithm that solves it.

Another relatively simple cost measure is known as the *space cost*, and simply counts the number of qubits that the algorithm uses throughout its execution. We similarly define the *space complexity* of a problem as the minimum number of qubits one needs to solve it.

If an algorithm uses input that is supplied to it by means of oracles, then there is a very natural third measure that we can consider, which is to simply count the number of times we call the input oracles. This measure we refer to as

the *query cost*, and similarly we define the *query complexity* of a problem as the minimum number of queries that one needs in order to solve the problem.

Note that these complexities can be very different based on whether we want to solve our computational problem exactly, or with bounded error. For instance, the *search* problem takes as input a bit string $x \in \{0,1\}^n$, and finds an index $j \in [n]$ such that $x_j = 1$, or outputs that no such index exists. The input bit string is encoded in an oracle $O_x$ acting on $\mathbb{C}^{[n]}$, that for all $j \in [n]$ acts as

$$O_x : |j\rangle \mapsto (-1)^{x_j} |j\rangle .$$

There is a trivial *exact* algorithm that solve this problem in $n$ queries. For any $j \in [n]$, observe that the controlled oracle $O_x$, on the subspace $\mathrm{Span}\{|0\rangle |j\rangle , |1\rangle |j\rangle\}$, acts as the oracle operation in Deutsch's algorithm, with the function $f : \{0,1\} \to \{0,1\}$ defined as $f(0) = 0$ and $f(1) = x_j$. Thus, by running Deutsch's algorithm for every $j \in [n]$, we can recover the whole bit string $x$ with $n$ queries to $O_x$ in total, and consequently also output the answer to the search problem.

Surprisingly, though, in the *bounded-error* setting there exists a quantum algorithm that solves this problem with $\mathcal{O}(\sqrt{n})$ queries [Gro96]. We also know that these exact and bounded-error algorithms are optimal, since we have lower bounds on the query complexity that match these expressions up to constants [BBB+97]. Thus, the search problem has a quadratic separation between its exact and bounded-error query complexity.

Throughout this thesis we will predominantly consider this third complexity measure, i.e., the *query complexity*. One reason is that there exists a beautiful theory and a lot of prior work on upper and lower bounding query complexity, paving the way for tightly characterizing it for many different computational problems. Another reason is that the resulting expressions are usually easier to state in the text, easier to parse by the reader, and hence more effective in generating understanding of the difficulty of a computational problem. A third reason is that it circumvents having to be explicit about the embedding of the registers and unitaries onto the level of qubits and their elementary operations. But most importantly, both space and gate complexity likely will not accurately reflect the total amount of physical operations that need to be executed on an actual quantum device, due to architecture constraints and error reduction and mitigation considerations, all of which are still very active research at the moment. Query complexity is not affected by these implementation details, and as such is a more relevant quantity to compute at the moment.

**Physical implementation** The physical realization of a quantum computational device essentially comes down to developing a physical system whose state space is $\mathbb{C}^2$, or at least has a subspace into which $\mathbb{C}^2$ can be embedded, so that can be used to represent a single qubit. The device must then be able to apply the elementary operations defined in Table 2.1 to these qubits, and it must also

be able to perform some operation like the CNOT that acts on multiple qubits at once, so that their states can be entangled. Finally, it must be able to perform computational basis measurements on the individual qubits, so that we can learn properties of the quantum state it stores. If the device is able to perform all these things, then we can run our quantum computations on the given device.

Of course there are many challenges that designers of these quantum chips face. To hint at a few of them, we mention here that they have to find ways to deal with errors that occur during the computations, or cannot always perform two-qubit gates on pairs of qubits that are physically separated far from one another.

On the other hand, there are several things that might be possible to realize in the lab that are not captured well by the model presented in this section. For instance, suppose that we have a register that holds the index of some qubit in a larger array of data qubits, and we would like to use the quantum state stored in this particular data qubit. This is commonly known as a *random-access gate*, and in the model presented in this section would require at least a number of *elementary gates* that scales linearly in the number of data qubits. However, there are several proposed constructions of such random-access gates that can be implemented in significantly fewer *physical operations*, or at least in total execution time that scales sub-linearly in the number of data qubits used. This data structure is usually called QRAM (Quantum Random Access Memory), and it is at this moment not clear whether quantum devices ultimately will have access to such features or not.

Even though the challenges mentioned in these paragraphs are very interesting topics of research in their own right, they are beyond the scope of this thesis.

## 2.4 Algorithmic primitives

Now that we have introduced the model of computation that we will be using throughout the remainder of this thesis, we can describe several well-known algorithmic primitives that we will be using. This is not meant to be a full-fledged introduction into all these techniques, instead we refer to [NC00; dWol22; Chi22] for more complete introductions.

We start by defining a particular unitary operation on an $n$-qubit register, known as the quantum Fourier transform.

**2.4.1.** Definition. Let $n \in \mathbb{N}$. The quantum Fourier transform on $n$ qubits is an operation that for all $j \in [2^n - 1]_0$ performs the mapping

$$\mathrm{QFT}_{2^n} : |j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi i j k}{2^n}} |k\rangle . \tag{2.4.1}$$

For ease of notation, for all $j \in \mathbb{R}$ we define the state

$$|\mathrm{QFT}_{2^n}(j)\rangle = \frac{1}{\sqrt{2^n}} \sum_{j'=0}^{2^n-1} e^{\frac{2\pi i j k}{2^n}} |k\rangle \,. \tag{2.4.2}$$

We refer to this as the QFT-*state of $j$ on $n$ qubits.* ◀

Note that unlike in many branches of physics, in quantum computing it is customary to define the quantum Fourier transform without a minus sign in the exponent. Thus, when comparing the above definition with other disciplines, this one is closer to the inverse Fourier transform in many other fields.

Furthermore, note that for all $j \in [2^n - 1]_0$, we indeed have $\mathrm{QFT}_{2^n} |j\rangle = |\mathrm{QFT}_{2^n}(j)\rangle$, as the notation suggests. The benefit of the newly-introduced notation is that $|\mathrm{QFT}_{2^n}(j)\rangle$ is also defined whenever $j$ is not integer, whereas $\mathrm{QFT}_{2^n} |j\rangle$ is not.

Next, we prove a very useful lemma about computational basis measurements of QFT-states. The proof here is very similar in flavor to the analysis provided in [NC00, Section 5.2.1].

**2.4.2.** Lemma (Measurement of a QFT-state in the QFT-basis). *Let $k \in \mathbb{N}$ and $j \in \mathbb{R}$. If we measure $\mathrm{QFT}_{2^k}^\dagger |\mathrm{QFT}_{2^k}(j)\rangle$ in the computational basis and denote the outcome by $\widetilde{j} \in [2^n - 1]_0$, then for all $\ell \in [2^n - 1]_0$,*

$$\mathbb{P}[\widetilde{j} = \ell] = \frac{\sin^2(\pi(j-\ell))}{2^{2n} \sin^2(\pi(j-\ell)/2^n)},$$

*where we take limits if the denominator evaluates to 0. In particular, for all integer $m \geq 2$,*

$$\mathbb{P}[\text{cyclic-dist}_{2^n}(j, \widetilde{j}) > m] \leq \frac{1}{2(m-1)}.$$

**Proof:**
We directly write out the state before the computational basis measurement, and obtain

$$\mathrm{QFT}_{2^n}^\dagger |\mathrm{QFT}_{2^n}(j)\rangle = \frac{1}{2^n} \sum_{\ell=0}^{2^n-1} \sum_{k=0}^{2^n-1} e^{\frac{2\pi i (j-\ell) k}{2^n}} |\ell\rangle = \frac{1}{2^n} \sum_{\ell=0}^{2^n-1} \frac{1 - e^{2\pi i (j-\ell)}}{1 - e^{\frac{2\pi i (j-\ell)}{2^n}}} |\ell\rangle \,,$$

where we used a formula for the partial sums of the geometric series. Using that $|1 - e^{2ix}| = 2|\sin(x)|$ for all $x \in \mathbb{R}$, we obtain for all $\ell \in [2^n - 1]_0$:

$$\begin{aligned} \mathbb{P}[\widetilde{j} = \ell] &= \left| \langle \ell | \, \mathrm{QFT}_{2^n}^\dagger |\mathrm{QFT}_{2^n}(j)\rangle \right|^2 = \frac{1}{2^{2n}} \left| \frac{1 - e^{2\pi i (j-\ell)}}{1 - e^{\frac{2\pi i (j-\ell)}{2^n}}} \right|^2 \\ &= \frac{\sin^2(\pi(j-\ell))}{2^{2n} \sin^2(\pi(j-\ell)/2^n)}. \end{aligned}$$

Since the functions $\sin^2(\pi x)$ and $\sin^2(\pi x/2^n)$ are periodic with period $2^n$, we can substitute $j-\ell$ with cyclic-dist$_{2^n}(j,\ell)$. We also observe that among all possible choices for $\ell \in \{0,\ldots,2^n-1\}$, there are at most 2 such that cyclic-dist$_{2^n}(j,\ell) \in (k,k+1]$, for all $k \in [2^{n-1}-1]_0$. Furthermore, observe that $\sin^2(\pi x)$ is increasing on the interval $[0,1/2]$, and thus

$$\mathbb{P}[\text{cyclic-dist}_{2^n}(\widetilde{j},j) > m] = \sum_{\substack{\ell=0 \\ \text{cyclic-dist}_{2^n}(j,\ell)>m}}^{2^n-1} \mathbb{P}[\widetilde{j}=\ell]$$

$$\leq \sum_{\substack{\ell=0 \\ \text{cyclic-dist}(j,\ell)>m}}^{2^n-1} \frac{1}{2^{2n}\sin^2(\pi\,\text{cyclic-dist}_{2^n}(j,\ell)/2^n)} \leq \frac{2}{2^{2n}}\sum_{\ell=m}^{2^{n-1}-1}\frac{1}{\sin^2(\pi\ell/2^n)}.$$

Now, we use the numerical inequality $\sin^2(\pi x) \geq (2x)^2$, for all $x \in [0,1/2]$, and obtain

$$\mathbb{P}[\text{cyclic-dist}_{2^n}(\widetilde{j},j) > m] \leq \frac{2}{2^{2n}}\sum_{\ell=m}^{2^{n-1}-1}\frac{2^{2n}}{4\ell^2} \leq \sum_{\ell=m}^{\infty}\frac{1}{2\ell^2}$$

$$\leq \int_{m-1}^{\infty}\frac{1}{2\ell^2}\,\mathrm{d}\ell = \left[-\frac{1}{2\ell}\right]_{m-1}^{\infty} = \frac{1}{2(m-1)}.$$

This completes the proof. □

The lemma presented above plays a crucial role in the analysis of the *phase estimation algorithm*, which is the algorithm we present next. It was first introduced in [Kit96].

**Algorithm 2.4.3:** Phase estimation

**Input:**

1: $k \in \mathbb{N}$: the number of bits of precision.
2: $\mathcal{U}$: a quantum circuit acting on $\mathcal{H}$ that implements a unitary $U$.
3: $C_{|\psi\rangle}$: a quantum circuit acting on $\mathcal{H}$ that implements $|0\rangle \mapsto |\psi\rangle$.

**Derived objects:**

1: Let the eigendecomposition of $U$ be

$$U = \sum_{\ell=1}^{r} e^{2\pi i\theta_\ell}\,|\psi_\ell\rangle\langle\psi_\ell|\,,$$

where for all $\ell \in [r]$, $\theta_\ell \in (-1/2,1/2]$.

2: Let $\Phi$ be a random variable taking values in $\{\theta_\ell : \ell \in [r]\}$, such that for all $j \in [r]$,

$$\mathbb{P}[\Phi=\theta_\ell] = |\,\langle\psi|\psi_\ell\rangle\,|^2 =: p_\ell.$$

**Output:** A number $\overline{\phi} \in (-1/2, 1/2]$ such that for all $\ell \in [r]$ and integer $m \geq 2$,

$$\mathbb{P}[\text{cyclic-dist}(\theta_\ell, \overline{\phi}) \leq m \cdot 2^{-k}] \geq p_\ell \left(1 - \frac{1}{2(m-1)}\right).$$

**Queries:**

1: Number of queries to $\mathcal{U}$: $2^k - 1$.
2: Number of queries to $C_{|\psi\rangle}$: 1.

**Procedure:** PHASE-EST($k$, $\mathcal{U}$, $C$):

We start in the state $|0\rangle \otimes |0\rangle \in (\mathbb{C}^2)^{\otimes k} \otimes \mathcal{H}$.

1: Apply $C_{|\psi\rangle}$ to the second register.
2: Apply $H^{\otimes k}$ to the first register.
3: Conditioned on the first register being in state $|j\rangle$, apply $U^j$ on the second register.
4: Apply $\mathrm{QFT}^\dagger_{2^k}$ to the first register.
5: Measure the first register in the computational basis. Denote the outcome by $j \in \{-2^{k-1}+1, \ldots, 2^{k-1}\}$.
6: Output $\overline{\phi} = j/2^k$.

---

**Proof of the properties of Algorithm 2.4.3:**
We easily obtain the claimed number of calls to $\mathcal{U}$ and $C_{|\psi\rangle}$ from the description of the procedure. Thus, it remains to check the claimed outcome probabilities.

To that end, we track the state throughout the algorithm. After step 2, we are in the state

$$\frac{1}{\sqrt{2^k}} \sum_{j=0}^{2^k-1} |j\rangle \otimes |\psi\rangle .$$

Since $U$ is unitary, the set $\{|\psi_j\rangle : j \in [r]\}$ is an orthonormal basis of $\mathcal{H}$. Thus, we can decompose $|\psi\rangle$ in this basis, and obtain that after step 2 we have the following state:

$$\frac{1}{\sqrt{2^k}} \sum_{j=0}^{2^k-1} |j\rangle \otimes \sum_{\ell=1}^{r} \langle\psi_\ell|\psi\rangle \, |\psi_\ell\rangle .$$

After step 3, we obtain the state

$$\sum_{\ell=1}^{r} \frac{\langle\psi_\ell|\psi\rangle}{\sqrt{2^k}} \sum_{j=0}^{2^k-1} e^{2\pi i j \theta_\ell} \, |j\rangle \otimes |\psi_\ell\rangle = \sum_{\ell=1}^{r} \langle\psi_\ell|\psi\rangle \left|\mathrm{QFT}_{2^k}(2^k\theta_\ell)\right\rangle \otimes |\psi_\ell\rangle ,$$

where we used the definition of the QFT-state, i.e., Definition 2.4.1. Thus, by applying the inverse Fourier transform to the first register, we obtain the state

$$\sum_{\ell=1}^{r} \langle\psi_\ell|\psi\rangle \, \mathrm{QFT}^\dagger \left|\mathrm{QFT}_{2^k}(2^k\theta_\ell)\right\rangle \otimes |\psi_\ell\rangle .$$

Hence, the probability that at the end of step 5 we obtain outcome $j' \in \{-2^{k-1}+1, \ldots, 2^{k-1}\}$, becomes

$$\mathbb{P}[j = j'] = \sum_{\ell=1}^{r} p_\ell \cdot \frac{\sin^2(\pi(2^k\theta_\ell - j'))}{2^{2k}\sin^2(\pi(2^k\theta_\ell - j')/2^k)}, \tag{2.4.3}$$

where we used Lemma 2.4.2. Using the same analysis as in said lemma, we obtain that, for all integer $m \geq 2$,

$$\mathbb{P}[\text{cyclic-dist}_{2^k}(2^k\theta_\ell, j) \leq m] \geq p_\ell \cdot \left(1 - \frac{1}{2(m-1)}\right),$$

and so we obtain that at the end of step 6,

$$\mathbb{P}[\text{cyclic-dist}(\theta_\ell, \overline{\phi}) \leq m \cdot 2^{-k}] \geq p_\ell \cdot \left(1 - \frac{1}{2(m-1)}\right).$$

This completes the proof. □

Note in particular that if $U|\psi\rangle = e^{2\pi i\phi}|\psi\rangle$, then we recover the result from Nielsen and Chuang, i.e., [NC00, Equation (5.34)].

Next, we simply recall a couple of algorithms and their properties, but refer for their specific implementation, and the proof of their properties to their corresponding papers.

We continue with the *amplitude estimation algorithm*, which was first introduced in [BHM+02].

---

**Algorithm 2.4.4:** Amplitude estimation [BHM+02, Theorem 12]

---

**Input:**

1: $M \in \mathbb{N}$: a parameter that decides the number of iterations.
2: $C_{|\psi\rangle}$: a quantum circuit acting on $\mathcal{H}$ that implements $|0\rangle \mapsto |\psi\rangle$.
3: $R$: a quantum circuit acting on $\mathcal{H}$ that reflects through a subspace $\mathcal{A} \subseteq \mathcal{H}$.

**Derived objects:** $p = \|\Pi_{\mathcal{A}}|\psi\rangle\|^2$.

**Output:** A number $\widetilde{p} \in [0, 1]$ that satisfies

$$|\widetilde{p} - p| \leq \frac{2\pi\sqrt{p(1-p)}}{M} + \frac{\pi^2}{M^2}.$$

**Success probability:** Lower bounded by $8/\pi^2$.

**Queries:**

1: Number of calls to $C_{|\psi\rangle}$: $M$.
2: Number of calls to $R$: $\mathcal{O}(M)$.

**Procedure:** AMP-EST($M$, $C_{|\psi\rangle}$, $R$).

---

Next, we continue with fixed-point amplitude amplification, first introduced in [YLC14], and later reproved via more general techniques in [GSL+19, Theorem 27].

**Algorithm 2.4.5:** Fixed-point amplitude amplification [YLC14; GSL+19]

**Input:**
1: $\varepsilon > 0$: a precision parameter.
2: $\delta > 0$: a proximity parameter.
3: $R_{\mathcal{A}}$: a quantum circuit that reflects through a subspace $\mathcal{A} \subseteq \mathcal{H}$.
4: $C_{|\psi\rangle}$: a quantum circuit that implements $|0\rangle \mapsto |\psi\rangle$, acting on $\mathcal{H}$.
**Assumption:** $\|\Pi_{\mathcal{A}} |\psi\rangle\| \geq \delta$.
**Output:** A quantum state $|\psi'\rangle$ such that

$$\left\| |\psi'\rangle - \frac{\Pi_{\mathcal{A}} |\psi\rangle}{\|\Pi_{\mathcal{A}} |\psi\rangle\|} \right\| \leq \varepsilon.$$

**Queries:**
1: Number of calls to $R_{\mathcal{A}}$: $\mathcal{O}(\log(1/\varepsilon)/\delta)$.
2: Number of calls to $C_{|\psi\rangle}$: $\mathcal{O}(\log(1/\varepsilon)/\delta)$.
**Procedure:** FIXED-POINT-AMPL($\varepsilon$, $\delta$, $R_{\mathcal{A}}$, $C_{|\psi\rangle}$).

Now, we provide *linear amplitude amplification*, which is slightly different from *fixed-point amplitude amplification*. It follows from the techniques introduced in [GSL+19], with a particular choice of polynomial that is for instance constructed in [GL20, Lemma 11].

**Algorithm 2.4.6:** Linear amplitude amplification [GSL+19; GL20]

**Input:**
1: $f \in [0, 1]$: the multiplication factor.
2: $\delta > 0$: the norm error tolerance parameter.
3: $C$: a quantum circuit acting on $\mathcal{H}$ that implements the mapping

$$C : |0\rangle \mapsto \sqrt{p}\,|\psi\rangle + \sqrt{1-p}\,|\perp\rangle\,,$$

where $|\psi\rangle \in \mathcal{A} \subseteq \mathcal{H}$, $|\perp\rangle \in \mathcal{A}^{\perp}$.
4: $R_{\mathcal{A}}$: a circuit that reflects around $\mathcal{A}$.
**Assumption:** $fp < 1/4$.
**Output:** A circuit that implements the mapping

$$|0\rangle \mapsto \sqrt{fp}\,|\psi\rangle + \sqrt{1-fp}\,|\perp\rangle\,,$$

up to norm error $\delta$.
**Queries:** Number of calls to $C$ and $R_{\mathcal{A}}$: $\mathcal{O}(\log(1/\delta)\sqrt{f})$.
**Procedure:** LINEAR-AMPL-AMPL($f$, $\delta$, $C$, $R_{\mathcal{A}}$).

Finally, we recall two more results from quantum signal processing. First, we describe an oracle conversion technique that first appeared in [GAW19, Theorem 14].

**Algorithm 2.4.7:** Probability to phase oracle conversion [GAW19]

**Input:**

1: $\delta > 0$: the norm error tolerance parameter.

2: $U$: a circuit acting on $\mathbb{C}^{\mathcal{D}} \otimes \mathcal{H} \otimes \mathbb{C}^2$ that for all $x \in \mathcal{D}$ implements the operation

$$U : |x\rangle |0\rangle |0\rangle \mapsto |x\rangle \otimes \left( \sqrt{p_x} |\psi_{x,1}\rangle |1\rangle + \sqrt{1 - p_x} |\psi_{x,0}\rangle |0\rangle \right),$$

where for all $x \in \mathcal{D}$, $p_x \in [0, 1]$, and $|\psi_{x,0}\rangle, |\psi_{x,1}\rangle \in \mathcal{H}$.

**Output:** A circuit acting on $\mathbb{C}^{\mathcal{D}}$, that implements the operation

$$O : |x\rangle \mapsto e^{ip_x} |x\rangle,$$

up to norm error $\delta$.

**Queries:** Number of queries to $U$: $\mathcal{O}(\log(1/\delta))$.

**Procedure:** Probability-to-phase($U$, $\delta$).

Finally, we reference a very special case of a much more general result on Hamiltonian simulation. The specific version we are using here is [GSL+19, Corollary 63].

**Algorithm 2.4.8:** Special case of Hamiltonian simulation [GSL+19]

**Input:**

1: $C$: a quantum circuit acting on $\mathbb{C}^X \otimes \mathcal{H}$, with $X$ some finite set, that performs the mapping

$$|x\rangle |0\rangle \mapsto |x\rangle \otimes (p(x) |0\rangle + |\perp\rangle),$$

where for all $x \in X$, $p(x) \in [0, 1]$, and $|\perp\rangle$ is some unnormalized state that is orthogonal to $|0\rangle$, and may depend on $x$.

2: $t > 0$: the amplification parameter.

3: $\delta > 0$: the norm error tolerance.

**Output:** A quantum circuit acting on $\mathbb{C}^X$ that performs the mapping

$$|x\rangle \mapsto e^{itp(x)} |x\rangle,$$

up to norm error $\delta$.

**Queries:** Number of calls to $C$: $\mathcal{O}(t + \log(1/\delta))$.

**Procedure:** Ham-sim($C$, $t$, $\delta$).

# Part I

# Quantum algorithms

# Chapter 3

# Quantum mean estimation

In this chapter, we consider the fundamental problem in statistics of estimating the mean of a random variable. We consider the setting where we can take quantum samples, i.e., where we can generate coherent superpositions over all possible values that the random variable can attain, weighted by their probabilities. We also specifically allow for the case where the random variable is multivariate, and do not require any additional information besides that its covariance matrix is well-defined. The quantum algorithm we construct estimates the mean using a number of samples set a priori, and with this constraint performs optimally up to polylogarithmic factors. This tight characterization leads to the interesting observation that quantum computers only offer an advantage over classical ones in terms of precision, if the number of samples exceeds the dimension of the random variable.

This chapter is based on [CHJ22]. We start by introducing prior work on this topic in Section 3.1. Then, in Section 3.2, we precisely define how we are given access to the random variable, and state the naive classical algorithm that solves this problem essentially optimally. Next, in Section 3.3, we introduce a useful subroutine that solves the slightly less general *bounded mean estimation* problem. After that, in Section 3.4, we show how this subroutine can be used to solve the general mean estimation problem. Finally, in Section 3.5, we prove the optimality of the approach outlined in this chapter. We end with some general remarks in Section 3.6.

## 3.1 Introduction

Monte Carlo methods are used extensively in various fields of science and engineering, such as statistical physics [BH10], finance [Gla03], and machine learning [AdFD+03]. At the core of these methods is a Monte Carlo process, e.g., a randomized algorithm, whose expected outcome is to be estimated via repeated random executions. Quantum computers can speed up this approach at two dif-

ferent levels [Mon15]. First, novel algorithmic techniques such as Hamiltonian simulation [Fey82] or quantum walks [Sze04] provide faster Monte Carlo simulation processes. Second, quantum metrology algorithms (such as phase estimation [Kit96]) give better error rates for computing statistics on these processes. This chapter focuses on this second point through the lens of the mean estimation problem. In this problem, the objective is to compute the closest possible estimate $\widetilde{\mu}$ to the mean $\mu = \mathbb{E}[X]$ of a random variable $X$ representing the output of some black-box process. Given the ability to repeat this process $n$ times (the sample complexity), one seeks to minimize the error $\|\widetilde{\mu} - \mu\|_2$ made with high probability.

In the classical setting, a beautiful theory [LM19] has been developed to solve the mean estimation problem in Euclidean norm. Under the sole assumption that the covariance matrix $\Sigma$ of $X$ exists, it turns out that the optimal non-asymptotic error behaves as if $X$ followed the Gaussian distribution $N(\mu, \Sigma)$. This motivated the use of the adjective *sub-Gaussian* to qualify the optimal classical estimators. In one dimension, the most well-known sub-Gaussian estimator is arguably the median-of-means [NY83; JVV86; AMS99]. The first computationally efficient sub-Gaussian estimator in high dimension was only found recently by Hopkins [Hop20]. These estimators achieve an optimal error of $\|\widetilde{\mu} - \mu\|_2 = \mathcal{O}(\sqrt{\mathrm{Tr}(\Sigma)/n} + \sqrt{\|\Sigma\| \log(1/\delta)/n}$ with probability at least $1 - \delta$.

In the quantum setting, the univariate case $X \in \mathbb{R}$ has been studied since the early works on quantum counting [BBH+98]. The celebrated amplitude estimation algorithm [BHM+02] provides a smaller error rate for estimating the mean of any Bernoulli random variable compared to the classical estimators. For general univariate distributions, a series of quantum estimators [Gro98; Ter99; AW99; Hei02; WCN+09; BDG+11; Mon15; HM19; Ham21] culminated into a near-optimal algorithm that outperforms any classical estimator. On the other hand, the multivariate case $X \in \mathbb{R}^d$, appearing notably in machine learning applications, remains largely unaddressed by quantum algorithms. Classically, it admits a simple near-optimal approach: the $d$ coordinates of $\mu$ can all be estimated simultaneously with $d$ univariate sub-Gaussian estimators run in parallel (i.e., using the same samples from $X$) with only a logarithmic overhead $\log(d)$ in sample complexity (due to the Hoeffding bound and a union bound over the $d$ coordinates). In the quantum scenario however, this simultaneous evaluation of several univariate expectation values is more complicated. Indeed, the quantum algorithms for the univariate case rely on quantum amplitude estimation [BHM+02], which involves as a critical step an encoding of the expectation value in the relative phase of a quantum register. At first sight, it is unclear how a vector of $d$ phases could be encoded simultaneously into $d$ registers without requiring a linear overhead in $d$. In fact, a lower bound proved by Heinrich [Hei04] rules out the possibility of simply a $\log(d)$ overhead for the quantum multivariate mean estimation problem.

This chapter develops near-optimal and computationally efficient quantum mean estimators for vector-valued random variables of arbitrary dimension with binary oracle access. Unlike in the univariate setting ($d = 1$), where the optimal quantum estimator [Ham21] is strictly more efficient than any classical estimator, we identify two different regimes in higher dimension: (i) if a quantum estimator is limited to accessing the input at most $d$ times (i.e., $n \leq d$) then no advantage can be gained over the classical sub-Gaussian estimators, (ii) if it can access the input at least $d$ times (i.e., $n \geq d$) then the approximation error can be reduced by a near-optimal factor of $\sqrt{d/n}$ compared to classical sub-Gaussian estimators.

## 3.2 Preliminaries

In this section, we introduce the model that we will be using throughout this chapter. We start by formally defining a random variable.

**3.2.1.** Definition (Mean and covariance matrix).
Let $\Omega$ be a finite set, $(\Omega, 2^\Omega, \mathbb{P})$ be a probability space, and let $R \subseteq \mathbb{R}^d$ be a finite set. Then $X : \Omega \to R$ is a *$d$-dimensional random variable*, and we define

$$\mathbb{E}[X] = \sum_{\omega \in \Omega} \mathbb{P}(\omega) X(\omega), \qquad \text{and} \qquad \mathrm{Cov}(X) = \mathbb{E}[XX^T] - \mathbb{E}[X]\mathbb{E}[X]^T.$$

We refer to $\mathbb{E}[X] \in \mathbb{R}^d$ as the *mean* or *expectation*, and frequently denote it by $\mu$. Similarly, we refer to $\mathrm{Cov}(X) \in \mathbb{R}^{d \times d}$ as the *covariance matrix*, and frequently denote it by $\Sigma$. ◀

We assume here that $\Omega$ and $R$ are finite sets, which has two benefits. First, it ensures that the expectation and covariance matrix are always well-defined, which is not necessarily the case when $\Omega$ and $R$ can be any sets. Second, to construct a quantum algorithm we will need to embed $\Omega$ and $R$ into finite-dimensional Hilbert spaces, so assuming that $\Omega$ and $R$ are finite from the start saves us from having to use discretizations later on.

Next, we define how a quantum algorithm can access the random variable. This is the objective of the next definition.

**3.2.2.** Definition (Random variable access model).
Let $\Omega$ be a finite set, $(\Omega, 2^\Omega, \mathbb{P})$ be a probability space, and $X : \Omega \to R \subseteq \mathbb{R}^d$ be a $d$-dimensional random variable.

1. The *probability distribution oracle* $U_\mathbb{P}$ acts on $\mathbb{C}^\Omega$ and implements the operation
$$U_\mathbb{P} : |0\rangle \mapsto \sum_{\omega \in \Omega} \sqrt{\mathbb{P}(\omega)}\, |\omega\rangle .$$

2. The *random variable oracle* $O_X$ acts on $\mathbb{C}^\Omega \otimes \mathbb{C}^R$ and implements the operation, for all $\omega \in \Omega$,

$$O_X : |\omega\rangle |0\rangle \mapsto |\omega\rangle |X(\omega)\rangle ,$$

where $|0\rangle$ is some arbitrary reference state in $\mathbb{C}^R$. ◀

In the above definition, we associate a computational basis state $|r\rangle$ to every element $r \in R$. In general we don't require any particular embedding of the states $|r\rangle$ into qubit registers, as the most convenient way of designing this embedding might depend on the architecture of the device on which the algorithm is to be implemented. The only requirement is that two states $|r\rangle$ and $|r'\rangle$, for $r, r' \in R$, are orthogonal if $r \neq r'$.

If $R \subseteq \mathbb{R}^d$, then every element $r \in R$ is a $d$-dimensional vector $(r_1, \ldots, r_d)^T$, where for each $j \in [d]$, $r_j \in \mathbb{R}$ is specified up to some finite precision. In that case we can for instance think of $|r\rangle$ as

$$|r\rangle = |r_1\rangle \otimes \cdots \otimes |r_d\rangle ,$$

i.e., the state $|r\rangle$ contains a full binary description of all the elements of $r$. This particular embedding corresponds with the typical way in which vectors in $\mathbb{R}^d$ are encoded in classical computers too.

We also remark that one can easily obtain a classical sample of the random variable $X$, by performing the following sequence of operations. First, execute $U_\mathbb{P}$ to make a superposition over all events $\omega \in \Omega$, then call $O_X$ to compute the random variable, and subsequently measure the final register. Thus, obtaining classical samples from $X$ can be done by making one call to each of the two routines that provide quantum access.

The question that we seek to answer in this chapter is how precisely we can estimate the mean of the random variable $X$, if we are allowed to make at most $n$ queries to both $U_\mathbb{P}$ and $O_X$. In the classical setting, we can very easily construct an algorithm that achieves a precision of $\mathcal{O}(\sqrt{\mathrm{Tr}[\Sigma]/n})$ with high probability, where $\Sigma$ is the covariance matrix of $X$. For ease of reference, we state this algorithm below, and prove its properties.

**Algorithm 3.2.3:** Classical mean estimation

**Input:**

1: $X : \Omega \to \mathbb{R}^d$: a random variable on a probability space $(\Omega, 2^\Omega, \mathbb{P})$.
2: $n \in \mathbb{N}$: the number of samples (up to logarithmic factors).
3: $\delta > 0$: the failure probability tolerance.
4: $U_\mathbb{P}$: the probability distribution oracle, as defined in Definition 3.2.2.
5: $O_X$: the random variable oracle, as defined in Definition 3.2.2.

**Derived objects:**

1: $N = \lceil 18 \log(2/\delta) \rceil$.

**Output:** A vector $\widetilde{\mu} \in \mathbb{R}^d$ that satisfies $\|\widetilde{\mu} - \mathbb{E}[X]\|_2 \leq 2\sqrt{3\,\mathrm{Tr}[\Sigma]/n}$.
**Success probability:** Lower bounded by $1 - \delta$.
**Queries:** $\mathcal{O}(n \log(1/\delta))$ queries to $U_{\mathbb{P}}$ and $O_X$.
**Procedure:** CLASSICAL-MEAN-EST($X$, $n$, $\delta$, $U_{\mathbb{P}}$, $O_X$):
1: For $j = 1, \ldots, N$, let $\widetilde{\mu}_j \in \mathbb{R}^d$ be the average of $n$ independent classical samples of $X$.
2: Output $\widetilde{\mu} \in \mathbb{R}^d$ such that for the strict majority of indices $j \in [N]$, we have $\|\widetilde{\mu}_j - \mathbb{E}[X]\|_2 \leq \sqrt{3\,\mathrm{Tr}[\Sigma]/n}$, or FAILURE if such a vector $\widetilde{\mu}$ does not exist.

---

**Proof:**
For all $j \in [N]$, let $X^{(1)}, \ldots, X^{(n)}$ be the random variables that describe the classical samples of $X$ obtained in the $j$th iteration of step 1 of the algorithm. Furhermore, let $X_\ell$ be the $\ell$th entry of the $d$-dimensional random variable $X$. Then,

$$\begin{aligned}
\mathbb{E}[\|\widetilde{\mu}_j - \mathbb{E}[X]\|_2^2] &= \mathbb{E}\left[\left\|\frac{1}{n}\sum_{k=1}^{n} X^{(k)} - \mathbb{E}[X]\right\|_2^2\right] \\
&= \mathbb{E}\left[\left(\frac{1}{n}\sum_{k=1}^{n} X^{(k)} - \mathbb{E}[X]\right)^T \left(\frac{1}{n}\sum_{k=1}^{n} X^{(k)} - \mathbb{E}[X]\right)\right] \\
&= \frac{1}{n^2}\mathbb{E}\left[\sum_{k=1}^{n}\sum_{\ell=1}^{n}(X^{(k)} - \mathbb{E}[X])^T(X^{(\ell)} - \mathbb{E}[X])\right] = \frac{1}{n^2}\sum_{k=1}^{n}\mathbb{E}\left[\left\|X^{(k)} - \mathbb{E}[X]\right\|_2^2\right] \\
&= \frac{1}{n}\mathbb{E}[\|X - \mathbb{E}[X]\|_2^2] = \frac{1}{n}\sum_{\ell=1}^{n}\mathbb{E}\left[(X_\ell - \mathbb{E}[X_\ell])^2\right] = \sum_{\ell=1}^{n}\mathrm{Var}(X_\ell) = \mathrm{Tr}[\Sigma],
\end{aligned}$$

and hence we find by Markov's inequality that for all $j \in [N]$,

$$\mathbb{P}\left[\|\widetilde{\mu}_j - \mathbb{E}[X]\|_2 > \sqrt{\frac{3\,\mathrm{Tr}[\Sigma]}{n}}\right] \leq \frac{\mathbb{E}[\|\widetilde{\mu}_j - \mathbb{E}[X]\|_2^2]n}{3\,\mathrm{Tr}[\Sigma]} = \frac{1}{3}.$$

Now, let $B_j$ be the Bernoulli random variable that is 1 if and only if we have $\|\widetilde{\mu}_j - \mathbb{E}[X]\|_2 \leq \sqrt{3\,\mathrm{Tr}[\Sigma]/n}$. Let $p := \mathbb{E}[B_j] \geq 2/3$. By Hoeffding's inequality, we find that

$$\begin{aligned}
\mathbb{P}\left[\sum_{j=1}^{N} B_j \leq \frac{N}{2}\right] &\leq \mathbb{P}\left[\left|\sum_{j=1}^{N} B_j - pN\right| \leq N\left|p - \frac{1}{2}\right|\right] \leq 2\exp\left(-\frac{2N^2}{N}\left|p - \frac{1}{2}\right|^2\right) \\
&\leq 2\exp\left(-\frac{N}{18}\right) \leq \delta,
\end{aligned}$$

where the last inequality follows from the choice of $N$ in the algorithm statement. Thus, with probability at least $1 - \delta$, $\mathbb{E}[X]$ is a possible choice for $\widetilde{\mu}$ that satisfies

the conditions of step 2 in the algorithm. Moreover, by the pigeonhole principle, any $\widetilde{\mu}$ we choose has the property that at least one $j \in [N]$ satisfies $\|\widetilde{\mu} - \widetilde{\mu}_j\|_2 \leq \sqrt{3\operatorname{Tr}[\Sigma]/n}$ and $\|\mathbb{E}[X] - \widetilde{\mu}_j\|_2 \leq \sqrt{3\operatorname{Tr}[\Sigma]/n}$. Thus, by the triangle inequality, we find that $\|\widetilde{\mu} - \mathbb{E}[X]\|_2 \leq 2\sqrt{3\operatorname{Tr}[\Sigma]/n}$. This completes the proof. □

Note that the above algorithm can be easily rephrased into a form where we set the precision guarantee as a parameter. Indeed, with $\mathcal{O}(n\log(1/\delta))$ samples we obtain precision

$$\varepsilon = \mathcal{O}\left(\sqrt{\frac{\operatorname{Tr}[\Sigma]}{n}}\right),$$

with probability at least $1-\delta$, and so if we want to obtain precision $\varepsilon$, it suffices to use a number of samples $n$ that satisfies

$$n = \Theta\left(\frac{\operatorname{Tr}[\Sigma]}{\varepsilon^2}\log\left(\frac{1}{\delta}\right)\right).$$

However, the downside of the latter formulation is that we need to know (an upper bound on) $\operatorname{Tr}[\Sigma]$ when we choose the number of samples to run in our algorithm. This is why the first formulation, where we express the obtained precision in terms of $\varepsilon$, is a bit more flexible.

If one carefully compares the logarithmic overhead in the above algorithm to the best classical algorithms in the literature, then one finds that the above algorithm can be slightly improved [LM19]. However, in this chapter we only focus on the polynomial dependence on the parameters $d$, $n$, $\operatorname{Tr}[\Sigma]$, $1/\varepsilon$ and $1/\delta$. As such, we will present all subsequent results in $\widetilde{\mathcal{O}}$-notation, where the tilde hides polylogarithmic factors in the five above-mentioned quantities.

## 3.3 Bounded mean estimation

As a first stepping stone towards developing a quantum algorithm that solves the general mean estimation problem, we consider the restricted case where the range of the random variable is contained in the unit $\ell_2$-ball in $\mathbb{R}^d$, i.e., we assume that $X : \Omega \to R \subseteq \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_2 \leq 1\}$. Later, in Section 3.4, we generalize the results obtained in this section to the setting where the random variable has unbounded range.

We start by defining the *dual grid*, which plays a central role in many quantum algorithms that compute multivariate objects. This technique was first introduced in [Jor05], and later developed in [GAW19; vApe21; CJ21].

**3.3.1.** Definition (Dual grid). Let $d, n \in \mathbb{N}$. Then, for every *index vector* $\mathbf{j} \in \{-2^{n-1}, \ldots, 2^{n-1}-1\}^d$, we define a corresponding *grid vector*

$$\mathbf{x}_{\mathbf{j}} = \frac{\mathbf{j} + \frac{1}{2}\mathbf{1}}{2^n} \subseteq \left[-\frac{1}{2}, \frac{1}{2}\right]^d,$$

where $\mathbf{1} \in \mathbb{R}^d$ is the all-ones vector. We let

$$G_n^d = \{\mathbf{x}_\mathbf{j} : \mathbf{j} \in \{-2^{n-1}, \ldots, 2^{n-1} - 1\}^d\},$$

be the set of all these grid vectors, and we refer to $G_n^d$ as the *d-dimensional dual grid on n qubits.* ◀

We include a graphical representation of the dual grid in Figure 3.3.1. Note that $G_n^d$ has $2^n$ grid points in every direction, and thus contains $2^{nd}$ grid points in total.

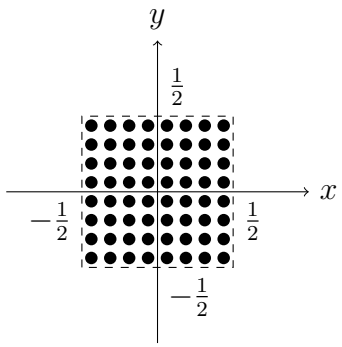



Figure 3.3.1: The dual grid for $d = 2$ and $n = 3$.

For every grid vector $\mathbf{x} \in G$, there exists a unique $\mathbf{j} \in \{-2^{n-1}, \ldots, 2^{n-1} - 1\}^d$ such that $\mathbf{x} = \mathbf{x}_\mathbf{j}$. We will use this correspondence to define the quantum state that represents $\mathbf{x}$, i.e., if $\mathbf{x} = \mathbf{x}_\mathbf{j}$, we let

$$|\mathbf{x}\rangle := |\mathbf{j}\rangle := \bigotimes_{k=1}^{d} |j_k\rangle\,, \tag{3.3.1}$$

where every $|j_k\rangle$ is an $n$-qubit computational basis state representing the integer $j_k \in \{-2^{n-1}, \ldots, 2^{n-1} - 1\}$.

Using the above identification of $|\mathbf{x}\rangle$ with a $d$-fold tensor product of $n$-qubit states, we can use the grid vectors to obtain a more intuitive understanding of the $d$-fold tensor product of the quantum Fourier transform over $n$ qubits, as defined in Definition 2.4.1. This is the objective of the following lemma.

**3.3.2.** Lemma. *Let $d, n \in \mathbb{N}$, and let $\mathbf{k} \in \{-2^{n-1}, \ldots, 2^{n-1} - 1\}^d$. Then,*

$$\mathrm{QFT}_{2^n}^{\otimes d} |\mathbf{k}\rangle = \frac{\alpha_\mathbf{k}}{\sqrt{2^{nd}}} \sum_{\mathbf{x} \in G_n^d} e^{2\pi i \mathbf{x}^T \mathbf{k}} |\mathbf{x}\rangle\,,$$

*where $\alpha_\mathbf{k} \in \mathbb{C}$ and $|\alpha_\mathbf{k}| = 1$.*

**Proof:**
From the definition of the vector $|\mathbf{k}\rangle$ in Equation (3.3.1) and the quantum Fourier transform in Definition 2.4.1, we observe that

$$\begin{aligned}
\mathrm{QFT}_{2^n}^{\otimes d}|\mathbf{k}\rangle &= \bigotimes_{\ell=1}^{d} \mathrm{QFT}_{2^n}|k_\ell\rangle = \bigotimes_{\ell=1}^{d} \frac{1}{\sqrt{2^n}} \sum_{j_\ell=0}^{2^n-1} e^{\frac{2\pi i j_\ell k_\ell}{2^n}} |j_\ell\rangle \\
&= \frac{1}{\sqrt{2^{nd}}} \bigotimes_{\ell=1}^{d} e^{\frac{-2\pi i(\frac{1}{2}+2^{n-1})k_\ell}{2^n}} \sum_{j_\ell=-2^{n-1}}^{2^{n-1}-1} e^{\frac{2\pi i(j_\ell+\frac{1}{2})k_\ell}{2^n}} |j_\ell\rangle \\
&= \frac{1}{\sqrt{2^{nd}}} \underbrace{e^{\frac{-2\pi i(\frac{1}{2}+2^{n-1})\sum_{\ell=1}^{d} k_\ell}{2^n}}}_{\alpha_{\mathbf{k}}} \sum_{\mathbf{j}\in\{-2^{n-1},\ldots,2^{n-1}-1\}^d} e^{2\pi i \mathbf{x}_{\mathbf{j}}^T \mathbf{k}} |\mathbf{j}\rangle \\
&= \frac{\alpha_{\mathbf{k}}}{\sqrt{2^{nd}}} \sum_{\mathbf{x}\in G_n^d} e^{2\pi i \mathbf{x}^T \mathbf{k}} |\mathbf{x}\rangle .
\end{aligned}$$

This completes the proof. □

Thus, the above lemma tells us that up to a global phase, we can think of the multidimensional quantum Fourier transform applied to the state $|\mathbf{k}\rangle$ as making a uniform superposition over all dual grid vectors $\mathbf{x}$, and then computing the inner product of a dual grid vector with the objective vector $\mathbf{k}$ in the phase.

Crucially, we now observe that the multidimensional quantum Fourier transform is a unitary operation, and as such can be inverted. Hence, intuitively, if we have the ability to compute the inner product of a given dual grid vector $\mathbf{x}$ with some vector $\mathbf{k}$ in the phase, then we can recover $\mathbf{k}$ using the *inverse* multidimensional quantum Fourier transform. More specifically, if we have access to an operation that implements the mapping

$$O_{\mathbf{k}} : |\mathbf{x}\rangle \mapsto e^{2\pi i \mathbf{x}^T \mathbf{k}} |\mathbf{k}\rangle , \tag{3.3.2}$$

then we can recover $\mathbf{k}$ by first making a uniform superposition over all vectors in the dual grid, then applying this operation $O_{\mathbf{k}}$, and finally performing the inverse multidimensional quantum Fourier transform.

The core idea of the multivariate quantum mean estimation algorithm is to apply the approach outlined above to the vector $\mathbf{k} = \mathbb{E}[X]$. To that end, we observe that it suffices to build an operation that computes the inner product with a dual grid vector $\mathbf{x} \in G_n^d$ and $\mathbb{E}[X]$ in the phase.

We slowly work towards constructing this operation. The high-level idea is to first encode the inner product we are after, i.e., $\mathbf{x}^T\mathbb{E}[X]$, in the amplitude of a quantum state that we can efficiently prepare, and then to use existing techniques from [GAW19] to compute this inner product in the phase.

One detail we have to take into account is that for any element $\omega \in \Omega$, the inner product $\mathbf{x}^T X(\omega)$ might be as large as $\sqrt{d}$, and as such we cannot prepare

a state that encodes $\mathbf{x}^T X(\omega)$ as an amplitude. To resolve this, we introduce a truncation parameter $M > 0$, and prepare the amplitudes $\sqrt{[\![\mathbf{x}^T X(\omega)]\!]_0^M / M}$ and $\sqrt{-[\![\mathbf{x}^T X(\omega)]\!]_{-M}^0 / M}$ instead, where the notation $[\![\cdot]\!]$ denotes truncation, and is defined in Equation (2.1.1). In the following subroutine, we explain how we can construct a state that encodes these amplitudes using a single call to the probability distribution and random variable oracles $U_{\mathbb{P}}$ and $O_X$.

**Algorithm 3.3.3:** Probability inner product oracle

**Input:**

1: $X : \Omega \to R \subseteq \mathbb{R}^d$: a random variable on a probability space $(\Omega, 2^\Omega, \mathbb{P})$.
2: $M > 0$: the truncation parameter.
3: $U_{\mathbb{P}}$: the probability distribution oracle as defined in Definition 3.2.2.
4: $O_X$: the random variable oracle as defined in Definition 3.2.2.

**Output:** A circuit acting on four registers with state spaces $\mathbb{C}^{G_n^d}$, $\mathbb{C}^\Omega$, $\mathbb{C}^{\mathbb{R}^d}$, $\mathbb{C}^2$, respectively. The operation implemented is, for all $\mathbf{x} \in G_n^d$,

$$|\mathbf{x}\rangle |0\rangle |0\rangle |0\rangle \mapsto |\mathbf{x}\rangle \sum_{\omega \in \Omega} \sqrt{\mathbb{P}(\omega)}\, |\omega\rangle\, |X(\omega)\rangle \otimes \left( \sqrt{\frac{[\![\mathbf{x}^T X(\omega)]\!]_0^M}{M}}\, |1\rangle + \sqrt{1 - \frac{[\![\mathbf{x}^T X(\omega)]\!]_0^M}{M}}\, |0\rangle \right),$$

**Queries:** 1 call to $U_{\mathbb{P}}$ and $O_X$.

**Procedure:** INNER-PRODUCT-PROBABILITY($X$, $M$, $U_{\mathbb{P}}$, $O_X$):

1: Apply $U_{\mathbb{P}}$ to the second register.
2: Apply $O_X$ to the second and third registers.
3: For vectors $\mathbf{x} \in G_n^d$ and $\mathbf{y} \in R$, apply

$$|\mathbf{x}\rangle |\mathbf{y}\rangle |0\rangle \mapsto |\mathbf{x}\rangle |\mathbf{y}\rangle \otimes \left( \sqrt{\frac{[\![\mathbf{x}^T \mathbf{y}]\!]_0^M}{M}}\, |1\rangle + \sqrt{1 - \frac{[\![\mathbf{x}^T \mathbf{y}]\!]_0^M}{M}}\, |0\rangle \right)$$

to the first, third and fourth registers.

**Proof of the properties of Algorithm 3.3.3:**
We track the state throughout the execution of the circuit. For all $\mathbf{x} \in G_n^d$, we find

$$\begin{aligned} |\mathbf{x}\rangle |0\rangle |0\rangle |0\rangle &\mapsto |\mathbf{x}\rangle \sum_{\omega \in \Omega} \sqrt{\mathbb{P}(\omega)}\, |\omega\rangle |0\rangle |0\rangle \mapsto |\mathbf{x}\rangle \sum_{\omega \in \Omega} \sqrt{\mathbb{P}(\omega)}\, |\omega\rangle\, |X(\omega)\rangle |0\rangle \\ &\mapsto |\mathbf{x}\rangle \sum_{\omega \in \Omega} \sqrt{\mathbb{P}(\omega)}\, |\omega\rangle\, |X(\omega)\rangle \otimes \left( \sqrt{\frac{[\![\mathbf{x}^T X(\omega)]\!]_0^M}{M}}\, |1\rangle + \sqrt{1 - \frac{[\![\mathbf{x}^T X(\omega)]\!]_0^M}{M}}\, |0\rangle \right). \end{aligned}$$

This completes the proof. □

Now, we observe via Algorithm 2.4.7 that we can turn this probability oracle into a phase oracle. This is the objective of the next algorithm we present here.

**Algorithm 3.3.4:** Phase inner product oracle

**Input:**
1: $X : \Omega \to R \subseteq \mathbb{R}^d$: a random variable on a probability space $(\Omega, 2^\Omega, \mathbb{P})$.
2: $M \in 3\mathbb{N}$: the truncation parameter.
3: $\delta > 0$: the norm error tolerance.
4: $U_{\mathbb{P}}$: the probability distribution oracle, as defined in Definition 3.2.2.
5: $O_X$: the random variable oracle, as defined in Definition 3.2.2.

**Derived objects:**
1: $\mathcal{A} = \textsc{Inner-product-probability}(X, 1/M, U_{\mathbb{P}}, O_X)$.

**Output:** A circuit acting on a single register with state space $\mathbb{C}^{G_n^d}$, implementing the mapping

$$|\mathbf{x}\rangle \mapsto e^{2\pi i \cdot \frac{1}{3}\mathbb{E}[[\![\mathbf{x}^T X]\!]_{-M}^{M}]} |\mathbf{x}\rangle ,$$

up to norm error $\delta$.

**Queries:** Number of calls to $U_{\mathbb{P}}$ and $O_X$: $\mathcal{O}(M \log(M/\delta))$.

**Procedure:** $\textsc{Inner-product-phase}(X, M, \delta, U_{\mathbb{P}}, O_X)$:
1: Run $\textsc{Probability-to-phase}(\mathcal{A}, 3\delta/(2M))$ $M/3$ times.
2: Map $|\mathbf{x}\rangle \mapsto |-\mathbf{x}\rangle$.
3: Run $\textsc{Probability-to-phase}(\mathcal{A}, 3\delta/(2M))$ $M/3$ times in reverse.
4: Map $|\mathbf{x}\rangle \mapsto |-\mathbf{x}\rangle$.

**Proof of the properties of Algorithm 3.3.4:**
Since $\mathcal{A}$ performs exactly one call to $U_{\mathbb{P}}$ and $O_X$, the total number of calls we make to these operations is $\mathcal{O}(M \log(M/\delta))$, by the properties of Algorithm 2.4.7. Moreover, since we make $2M/3$ calls to Algorithm 2.4.7, and in each make a norm error of at most $3\delta/(2M)$, we end up making a norm error of at most $\delta$.

It remains to check that we approximately implement the right operation. To that end, we track the state throughout the algorithm, i.e., for all $\mathbf{x} \in G_n^d$,

$$\begin{aligned} |\mathbf{x}\rangle &\mapsto e^{2\pi i \frac{1}{3}\mathbb{E}[[\![\mathbf{x}^T X]\!]_0^M]} |\mathbf{x}\rangle \mapsto e^{2\pi i \frac{1}{3}\mathbb{E}[[\![\mathbf{x}^T X]\!]_0^M]} |-\mathbf{x}\rangle \\ &\mapsto e^{2\pi i \frac{1}{3}(\mathbb{E}[[\![\mathbf{x}^T X]\!]_0^M] - \mathbb{E}[[\![-\mathbf{x}^T X]\!]_0^M])} |-\mathbf{x}\rangle \mapsto e^{2\pi i \frac{1}{3}(\mathbb{E}[[\![\mathbf{x}^T X]\!]_0^M] - \mathbb{E}[[\![-\mathbf{x}^T X]\!]_0^M])} |\mathbf{x}\rangle , \end{aligned}$$

where we can rewrite the expression within the brackets in the exponent as

$$\mathbb{E}[[\![\mathbf{x}^T X]\!]_0^M] - \mathbb{E}[[\![-\mathbf{x}^T X]\!]_0^M] = \mathbb{E}[[\![\mathbf{x}^T X]\!]_0^M + [\![\mathbf{x}^T X]\!]_{-M}^0] = \mathbb{E}[\![\mathbf{x}^T X]\!]_{-M}^M .$$

This completes the proof. □

Now, we can give the full bounded quantum mean estimation algorithm that estimates the mean of a random variable $X : \Omega \to R \subseteq \{\|\mathbf{x}\|_2 \leq 1\}$ up to precision $\varepsilon > 0$ in $\ell_\infty$-norm.

**Algorithm 3.3.5:** Bounded quantum mean estimation

**Input:**
1: $X : \Omega \to R \subseteq \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_2 < 1\}$: a random variable on a probability space $(\Omega, 2^\Omega, \mathbb{P})$.
2: $\varepsilon > 0$: the precision parameter.
3: $\delta > 0$: the failure probability tolerance.
4: $U_\mathbb{P}$: the probability distribution oracle, as defined in Definition 3.2.2.
5: $O_X$: the random variable oracle, as defined in Definition 3.2.2.
**Derived objects:**
1: $N = \lceil 18 \log(2d/\delta) \rceil$.
2: $n = \lceil \log(9/\varepsilon) \rceil$.
3: $M = 3\lceil \sqrt{\ln(2^{2n} 512\pi^2 d)/18} \rceil$.
**Output:** a vector $\widetilde{\mu} \in \mathbb{R}^d$ such that $\|\widetilde{\mu} - \mathbb{E}[X]\|_\infty \leq \varepsilon$.
**Success probability:** Lower bounded by $1 - \delta$.
**Queries:** Number of calls to $U_\mathbb{P}$ and $O_X$: $\widetilde{\mathcal{O}}(1/\varepsilon)$.
**Procedure:** Bounded-mean-estimation-ell-infty($X$, $\varepsilon$, $\delta$, $U_\mathbb{P}$, $O_X$):
1: For $k = 1, \ldots, N$, run the following operations on a single register with state space $\mathbb{C}^{G_n^d}$:

1. Prepare the uniform superposition over the grid vectors.
2. Call Inner-product-phase($M$, $2^{-n}/24$, $U_\mathbb{P}$, $O_X$) $2^n$ times.
3. Apply the inverse multidimensional quantum Fourier transform, i.e., $(\mathrm{QFT}_{2^n}^\dagger)^{\otimes d}$.
4. Measure in the computational basis.
   Denote the outcome by $\mathbf{m}_k \in \{-2^{n-1}, \ldots, 2^{n-1} - 1\}^d$.

2: Let $\mathbf{m}$ be the coordinate-wise median of $\mathbf{m}_1, \ldots, \mathbf{m}_N$.
3: Output $\widetilde{\mu} = 3\mathbf{m}/2^n$.

---

**Proof of the properties of Algorithm 3.3.5:**
Since we make $N \cdot 2^n$ calls to Algorithm 3.3.4 with truncation parameter $M$ and norm error parameter $\delta = 2^{-n}/24$, we calculate the total number of calls to $U_\mathbb{P}$ and $O_X$ to be

$$\mathcal{O}\left(N \cdot 2^n M \log\left(\frac{M}{\delta}\right)\right) = \mathcal{O}\left(\frac{\sqrt{\log(d)}}{\varepsilon} \log\left(\frac{\sqrt{\log(d)}}{\varepsilon}\right) \log\left(\frac{d}{\delta}\right)\right),$$

which by suppressing all the logarithmic overhead indeed becomes $\widetilde{O}(1/\varepsilon)$.

It remains to check the lower bound on the success probability. We track the state throughout the algorithm, and obtain that step 2 implements the mapping

$$\frac{1}{\sqrt{2^{nd}}} \sum_{\mathbf{x} \in G_n^d} |\mathbf{x}\rangle \mapsto \frac{1}{\sqrt{2^{nd}}} \sum_{\mathbf{x} \in G_n^d} e^{2\pi i \frac{2^n}{3} \mathbb{E}[[\![\mathbf{x}^T X]\!]_{-M}^M]} |\mathbf{x}\rangle =: |\psi\rangle .$$

We calculate the norm difference between $|\psi\rangle$ and $|\phi\rangle$, where the latter is defined as

$$|\phi\rangle := \frac{1}{\sqrt{2^{nd}}} \sum_{\mathbf{x}\in G_n^d} e^{2\pi i \frac{2^n}{3}\mathbb{E}[\mathbf{x}^T X]} |\mathbf{x}\rangle = \bigotimes_{j=1}^{d} |\mathrm{QFT}_{2^n}(2^n \mathbb{E}[X]_j/3)\rangle ,$$

where the notation $|\mathrm{QFT}(\cdot)\rangle$ is defined in Definition 2.4.1. To that end, observe that

$$\begin{aligned} \||\psi\rangle - |\phi\rangle\|^2 &= \frac{1}{2^{nd}} \sum_{\mathbf{x}\in G_n^d} \left| e^{2\pi i \cdot \frac{2^n}{3}\mathbb{E}[\mathbf{x}^T X]} - e^{2\pi i \cdot \frac{2^n}{3}\mathbb{E}[\![\mathbf{x}^T X]\!]_{-M}^{M}} \right|^2 \\ &\leq \frac{2^{2n}4\pi^2}{9} \mathop{\mathbb{E}}_{\mathbf{x}\sim U(G_n^d)} \left[ \mathbb{E}\left[ \mathbf{x}^T X - [\![\mathbf{x}^T X]\!]_{-M}^{M} \right]^2 \right], \end{aligned} \tag{3.3.3}$$

where we used that $|e^{ix} - e^{iy}| \leq |x - y|$, for all $x, y \in \mathbb{R}$. Since for all $\omega \in \Omega$ and $\mathbf{x} \in G_n^d$, Cauchy–Schwarz implies that $|\mathbf{x}^T X(\omega)| \leq \|\mathbf{x}\|_2 \cdot \|X(\omega)\|_2 \leq \sqrt{d}$, we can bound for any $\mathbf{x} \in G_n^d$,

$$\mathbb{E}\left[ \mathbf{x}^T X - [\![\mathbf{x}^T X]\!]_{-M}^{M} \right]^2 \leq \sqrt{d} \cdot \mathbb{E}\left| \mathbf{x}^T X - [\![\mathbf{x}^T X]\!]_{-M}^{M} \right| .$$

Plugging this bound into Equation (3.3.3) and swapping the expectations yields

$$\begin{aligned} \||\psi\rangle - |\phi\rangle\|^2 &\leq \frac{2^{2n}4\pi^2\sqrt{d}}{9} \mathbb{E}\left[ \mathop{\mathbb{E}}_{\mathbf{x}\sim U(G_n^d)} \left| \mathbf{x}^T X - [\![\mathbf{x}^T X]\!]_{-M}^{M} \right| \right] \\ &\leq \frac{2^{2n}4\pi^2 d}{9} \max_{\omega\in\Omega} \mathop{\mathbb{P}}_{\mathbf{x}\sim U(G_n^d)} \left[ |\mathbf{x}^T X(\omega)| > M \right], \end{aligned} \tag{3.3.4}$$

where the final inequality holds by observing that for all $\omega \in \Omega$, $|\mathbf{x}^T X(\omega)| \leq \sqrt{d}$, and the fact that $\mathbf{x}^T X(\omega) - [\![\mathbf{x}^T X(\omega)]\!]_{-M}^{M}$ is zero whenever $|\mathbf{x}^T X(\omega)| \leq M$. Now, observe by Hoeffding's inequality, for all $\omega \in \Omega$,

$$\mathop{\mathbb{P}}_{\mathbf{x}\sim U(G_n^d)} \left[ \left| \mathbf{x}^T X(\omega) \right| > M \right] \leq 2\exp\left( -\frac{2M^2}{\|X(\omega)\|_2^2} \right) \leq 2\exp(-2M^2) \leq \frac{1}{2^{2n}256\pi^2 d},$$

where the last inequality follows from the choice of $M$. Plugging this into Equation (3.3.4) yields

$$\||\psi\rangle - |\phi\rangle\|^2 \leq \frac{2^{2n}4\pi^2 d}{9} \cdot \frac{1}{2^{2n}256\pi^2 d} = \left(\frac{1}{24}\right)^2 .$$

Hence, the norm difference between $|\psi\rangle$ and $|\phi\rangle$ is at most $1/24$. Since we also make an additional global norm error of at most $1/24$ in the calls to the inner product phase oracle, we construct $|\phi\rangle$ up to norm error $1/12$.

Thus, we can assume that we are in the state $|\phi\rangle$ at the end of step 2 in the loop, at the expense of $1/12$ in the measurement probabilities. Furthermore, if we apply the multidimensional inverse quantum Fourier transform to $|\phi\rangle$, we obtain

$$(\mathrm{QFT}_{2^n}^\dagger)^{\otimes d} |\phi\rangle = \bigotimes_{j=1}^{d} \mathrm{QFT}_{2^n}^\dagger |\mathrm{QFT}_{2^n}(2^n \mathbb{E}[X]_j/3)\rangle .$$

Thus, according to Lemma 2.4.2, we now have for every $k \in [N]$ and $j \in [d]$,

$$\mathbb{P}\left[\text{cyclic-dist}_{2^n}\left((m_k)_j, \frac{2^n}{3}\mathbb{E}[X]_j\right) \leq 3\right] \geq \frac{3}{4} - \frac{1}{12} = \frac{2}{3},$$

where cyclic-dist is defined in Definition 2.1.1, and the $1/12$ comes from the imperfections in the constructed state. Since $\mathbb{E}[X]_j \in [-1, 1]$, we obtain that $2^n \mathbb{E}[X]_j/3 \in [-2^n/3, 2^n/3]$. Moreover, since $2^n/3 + 3 \leq 2^{n-1}$, we find that $\text{cyclic-dist}((m_k)_j, 2^n \mathbb{E}[X]_j/3) \leq 3 \Leftrightarrow |(m_k)_j - 2^n \mathbb{E}[X]_j/3| \leq 3$, and so

$$\mathbb{P}\left[\left|(m_k)_j - \frac{2^n}{3}\mathbb{E}[X]_j\right| \leq 3\right] = \mathbb{P}\left[\text{cyclic-dist}_{2^n}\left((m_k)_j, \frac{2^n}{3}\mathbb{E}[X]_j\right) \leq 3\right] \geq \frac{2}{3}.$$

Now, let $j \in [d]$ and for all $k \in [N]$ let $A_{kj}$ be the Bernoulli random variable that is 1 whenever $|(m_k)_j - 2^n \mathbb{E}[X]_j/3| \leq 3$. Since the different runs are independent and identical, we find that $p := \mathbb{E}[A_{kj}] \geq 2/3$ is independent of $k$ and $j$, and the sequence of random variables $(A_{kj})_{k=1}^N$ is independent and identically distributed. Moreover, since by the properties of the coordinate-wise median, if $|m_j - 2^n \mathbb{E}[X]_j/3| > 3$, then $A_{kj}$ must be 0 for at least half of the $k$'s, i.e., the sum of the $A_{kj}$'s over $k$ is at most $N/2$. The probability that this happens can be upper bounded by Hoeffding's inequality, as

$$\begin{aligned}
\mathbb{P}\left[\sum_{k=1}^{N} A_{kj} \leq \frac{N}{2}\right] &= \mathbb{P}\left[\left|\sum_{k=1}^{N} A_{kj} - Np\right| \leq N\left|p - \frac{1}{2}\right|\right] \\
&\leq 2\exp\left(-\frac{2N^2}{N}\left|p - \frac{1}{2}\right|^2\right) \leq 2\exp\left(-2N \cdot \left(\frac{1}{6}\right)^2\right) = 2\exp(-N/18) \leq \frac{\delta}{d},
\end{aligned}$$

where the last inequality follows from the choice of $N$. Thus, by the union bound over all $j \in [d]$, we obtain that

$$\mathbb{P}\left[\left\|\mathbf{m} - \frac{2^n}{3}\mathbb{E}[X]\right\|_\infty \leq 3\right] \geq 1 - \delta,$$

and if this event happens, then we also have

$$\|\widetilde{\mu} - \mathbb{E}[X]\|_\infty = \frac{3}{2^n} \cdot \left\|\mathbf{m} - \frac{2^n}{3}\mathbb{E}[X]\right\|_\infty \leq \frac{9}{2^n} \leq \varepsilon,$$

where the last inequality follows by the choice of $n$. This completes the proof. □

Thus, we have now constructed a quantum algorithm that computes the mean of a $d$-dimensional random variable up to precision $\varepsilon$ in $\ell^\infty$-norm.

To better understand how this algorithm relates to the classical procedure, Algorithm 3.2.3, which naturally estimates the mean up $\varepsilon$-error in the $\ell_2$-norm, we turn the bounded mean estimation algorithm into one that approximates the mean up to $\ell_2$-norm as well, using standard norm conversions. Hölder's inequality tells us that

$$\|\widetilde{\mu} - \mathbb{E}[X]\|_2 \leq \sqrt{d}\, \|\widetilde{\mu} - \mathbb{E}[X]\|_\infty ,$$

and thus if we want to obtain precision $\varepsilon$ w.r.t. the $\ell_2$-norm, we can run the bounded mean estimation algorithm with precision $\varepsilon/\sqrt{d}$ in the $\ell_\infty$-norm. This induces an extra factor $\sqrt{d}$ in the number of queries to both $U_{\mathbb{P}}$ and $O_X$.

Interestingly, this leads to the observation that the quantum algorithm does not beat the classical one in all regimes. Indeed, with Algorithm 3.2.3 and the observation that $\mathrm{Tr}[\Sigma] \leq 1$ whenever the range of $X$ is contained in the unit $\ell_2$-ball, we can approximate the mean of a bounded random variable $X : \Omega \to \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_2 < 1\}$ up to precision $\varepsilon$ in $\ell_2$-norm with $n = \mathcal{O}(1/\varepsilon^2)$ classical samples, which is smaller than the resources required to run the quantum algorithm whenever $\varepsilon = \Omega(1/\sqrt{d})$.

Thus, if $\varepsilon$ is large, i.e., if it is $\Omega(1/\sqrt{d})$, then it is more beneficial to run the classical algorithm, i.e., Algorithm 3.2.3. We refer to this regime as the *low-precision regime*. On the other hand, if $\varepsilon = \mathcal{O}(1/\sqrt{d})$, then the quantum algorithm is more efficient in terms of the number of queries to $U_{\mathbb{P}}$ and $O_X$ required, and we refer to this regime as the *high-precision regime*.

For convenience, we state the full algorithm that makes a distinction and handles both regimes below.

**Algorithm 3.3.6:** Bounded quantum mean estimation in $\ell_2$-norm

**Input:**

1: $X : \Omega \to R \subseteq \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_2 < 1\}$: a random variable on a probability space $(\Omega, 2^\Omega, \mathbb{P})$.
2: $\varepsilon > 0$: the precision parameter.
3: $\delta > 0$: the failure probability tolerance.
4: $U_{\mathbb{P}}$: the probability distribution oracle, as defined in Definition 3.2.2.
5: $O_X$: the random variable oracle, as defined in Definition 3.2.2.

**Output:** a vector $\widetilde{\mu} \in \mathbb{R}^d$ such that $\|\widetilde{\mu} - \mathbb{E}[X]\|_2 \leq \varepsilon$.
**Success probability:** Lower bounded by $1 - \delta$.
**Queries:** Number of calls to $U_{\mathbb{P}}$ and $O_X$:

$$K := \widetilde{\mathcal{O}}\left(\min\left\{\frac{1}{\varepsilon^2}, \frac{\sqrt{d}}{\varepsilon}\right\}\right). \tag{3.3.5}$$

**Procedure:** BOUNDED-MEAN-ESTIMATION($X$, $\varepsilon$, $\delta$, $U_{\mathbb{P}}$, $O_X$):
1: **if** $\varepsilon > 1/\sqrt{d}$ **then**,
2: Output $\widetilde{\mu} =$ CLASSICAL-MEAN-EST($X$, $1/\varepsilon^2$, $\delta$, $U_{\mathbb{P}}$, $O_X$).
3: **else**
4: Output $\widetilde{\mu} =$ BOUNDED-MEAN-ESTIMATION-ELL-INFTY($X$, $\varepsilon/\sqrt{d}$, $\delta$, $U_{\mathbb{P}}$, $O_X$).
5: **end if**

**Proof of the properties of Algorithm 3.3.6:**
It follows directly from the properties of Algorithms 3.2.3 and 3.3.5 that if we either results in an estimate of $\mathbb{E}[X]$ with precision $\varepsilon$ in $\ell_2$-norm, with probability at least $1 - \delta$.

By comparing the number of calls made to $U_{\mathbb{P}}$ and $O_X$ in both approaches, we obtain that in the classical approach, we make $\widetilde{O}(1/\varepsilon^2)$ calls, whereas in the quantum approach, the number of queries we make is $\widetilde{O}(\sqrt{d}/\varepsilon)$. We easily verify that by choosing the classical approach whenever $\varepsilon > 1/\sqrt{d}$, we indeed choose the minimum of these two quantities. This completes the proof. □

Somewhat surprisingly, as we will see in Section 3.5, the above algorithm is indeed essentially optimal. In particular this implies that the distinction between the low-precision and the high-precision regimes is something fundamental – it is not specifically a property of the algorithm that we design here, but rather it reflects a qualitative difference in the difficulty of the mean estimation problem between these two regimes. Moreover, it implies that there is no "hybrid" version of the bounded mean estimation algorithm, i.e., one cannot cleverly combine classical samples with a multidimensional phase estimation technique to interpolate between the high- and low-precision regimes and smoothen out the apparent discontinuity in the query complexity.

## 3.4 General mean estimation

Next, we show how the bounded mean estimation subroutine that we developed in the previous subsection can be used to construct a mean estimation algorithm for general random variables $X : \Omega \to R \subseteq \mathbb{R}^d$, without any restriction on their range. First, in Section 3.4.1, we consider the setting where we know an upper bound $S \geq \mathrm{Tr}[\Sigma]$, where $\Sigma$ is the covariance matrix of $X$. Afterwards, in Section 3.4.2, we generalize our approach so that we can also handle the case where such an upper bound is not available a priori.

### 3.4.1 Known upper bound on $\mathrm{Tr}[\Sigma]$

We first consider the case where we know an upper bound $S$ on $\mathrm{Tr}[\Sigma]$, where $\Sigma$ is the covariance matrix of the $d$-dimensional random variable $X$. The first

observation that we make is that this upper bound gives us some idea of the *spread* of $X$, i.e., how far apart one can expect realizations of $X$ to be from its mean. The following lemma makes this more precise.

**3.4.1.** Lemma. *Let $\Omega$ be a finite set, $(\Omega, 2^\Omega, \mathbb{P})$ be a probability space, and $X : \Omega \to R \subseteq \mathbb{R}^d$ be a random variable with $R$ finite as well. Let $\mu = \mathbb{E}[X]$ and $\Sigma = \mathrm{Cov}(X)$, and let $S > 0$ such that $\mathrm{Tr}[\Sigma] \leq S$. Furthermore, let $\overline{\mu} \in \mathbb{R}^d$ such that $\|\overline{\mu} - \mu\|_2 \leq \sqrt{S}$. Then, with $t > 0$,*

$$\mathbb{E}\left[\|X - \overline{\mu}\|_2^2\right] \leq 2S \qquad \text{and} \qquad \mathbb{P}\left[\|X - \overline{\mu}\|_2 \geq \sqrt{2tS}\right] \leq \frac{1}{t}.$$

**Proof:**
For the left claim, we observe by standard probability theory that

$$\begin{aligned}
\mathbb{E}[\|X - \overline{\mu}\|_2^2] &= \mathbb{E}[\|X - \mu + \mu - \overline{\mu}\|_2^2] = \mathbb{E}[(X - \mu + \mu - \overline{\mu})^T (X - \mu + \mu - \overline{\mu})] \\
&= \mathbb{E}[(X - \mu)^T (X - \mu)] + (\mu - \overline{\mu})^T \mathbb{E}[X - \mu] \\
&\quad + \mathbb{E}[(X - \mu)^T](\mu - \overline{\mu}) + (\mu - \overline{\mu})^T (\mu - \overline{\mu}) \\
&= \mathbb{E}[X^T X] - \mu^T \mu + \|\mu - \overline{\mu}\|_2^2 = \mathrm{Tr}[\mathbb{E}[XX^T] - \mu\mu^T] + \|\mu - \overline{\mu}\|_2^2 \\
&= \mathrm{Tr}[\Sigma] + \|\mu - \overline{\mu}\|_2^2 \leq S + S = 2S.
\end{aligned}$$

Then, by Markov's inequality, we obtain that

$$\mathbb{P}\left[\|X - \overline{\mu}\|_2 \geq \sqrt{2tS}\right] = \mathbb{P}\left[\|X - \overline{\mu}\|_2^2 \geq 2tS\right] \leq \frac{\mathbb{E}[\|X - \overline{\mu}\|_2^2]}{2tS} \leq \frac{1}{t}.$$

This completes the proof. □

Next, we introduce the concept of a ring variable.

**3.4.2.** Definition (Ring variable). Let $\Omega$ be a finite set, $(\Omega, 2^\Omega, \mathbb{P})$ be a probability space, and $X : \Omega \to R \subseteq \mathbb{R}^d$ be a random variable with $R$ a finite set as well. Let $\overline{\mu} \in \mathbb{R}^d$ and $0 < a < b$. We define

$$p_{X,\overline{\mu}}^{(a,b)} = \mathbb{P}[a \leq \|X - \overline{\mu}\|_2 < b].$$

We refer to $p_{X,\overline{\mu}}^{(a,b)}$ as the *ring probability around $\overline{\mu}$ with inner radius $a$ and outer radius $b$.*

Furthermore, let $R' = \{(x - \overline{\mu})/b : x \in R, a \leq \|x - \overline{\mu}\|_2 < b\} \cup \{0\}$ and let $R_{X,\overline{\mu}}^{(a,b)} : \Omega \to R' \subseteq \mathbb{R}^d$ be defined as

$$R_{X,\overline{\mu}}^{(a,b)}(\omega) = \begin{cases} \frac{X(\omega) - \overline{\mu}}{b}, & \text{if } a \leq \|X(\omega) - \overline{\mu}\|_2 < b, \\ 0, & \text{otherwise.} \end{cases}$$

We refer to $R_{X,\overline{\mu}}^{(a,b)}$ as the *ring variable around $\overline{\mu}$ with inner radius $a$ and outer radius $b$.* ◀

We observe that newly-defined range $R'$ in the above definition is automatically contained in the unit $\ell_2$-ball in $\mathbb{R}^d$, and as such any ring variable satisfies the condition required in the bounded mean estimation algorithm, i.e., Algorithm 3.3.6. Thus, this opens up the possibility of a routine that estimates the mean of a ring variable, which is the objective of the following algorithm.

**Algorithm 3.4.3:** Mean estimation of a ring variable

**Input:**

1: $X : \Omega \to R \subseteq \mathbb{R}^d$: a random variable on a probability space $(\Omega, 2^\Omega, \mathbb{P})$.
2: $\overline{\mu} \in \mathbb{R}^d$: a vector in $\mathbb{R}^d$.
3: $a > 0$: the inner radius of the ring.
4: $b > 0$: the outer radius of the ring.
5: $\varepsilon > 0$: the desired precision.
6: $\overline{p} \in (0, 1]$: an upper bound on the ring probability $p^{(a,b)}_{X,\overline{\mu}}$.
7: $\delta > 0$: the failure probability tolerance.
8: $U_{\mathbb{P}}$: the probability distribution oracle, as defined in Definition 3.2.2.
9: $O_X$: the random variable oracle, as defined in Definition 3.2.2.

**Derived objects:**

1: $C$: A circuit acting on $\mathbb{C}^R$, implementing the mapping

$$C : |x\rangle \mapsto \begin{cases} |x\rangle, & \text{if } a \leq \|x - \overline{\mu}\|_2 < b, \\ -|x\rangle, & \text{otherwise.} \end{cases}$$

2: $U' = \textsc{Linear-ampl-ampl}(1/(4\overline{p}),\ \delta/(2K),\ O_X(U_{\mathbb{P}} \otimes I),\ I \otimes C)$, acting on $\mathbb{C}^\Omega \otimes \mathbb{C}^R$, where $K$ is as in Equation (3.3.5).
3: $O' = (I \otimes Z)(O_X \otimes I)$, acting on $\mathbb{C}^\Omega \otimes \mathbb{C}^R \otimes C^{R'}$, where $Z$ is a circuit acting on $\mathbb{C}^R \otimes \mathbb{C}^{R'}$ as

$$|x\rangle |0\rangle \mapsto \begin{cases} |x\rangle \left|\frac{x-\overline{\mu}}{b}\right\rangle, & \text{if } a \leq \|x - \overline{\mu}\|_2 < b, \\ |x\rangle |0\rangle, & \text{otherwise.} \end{cases}$$

**Output:** A vector $\widetilde{\mu} \in \mathbb{R}^d$ that satisfies

$$\left\| \widetilde{\mu} - \mathbb{E}\left[ R^{(a,b)}_{X,\overline{\mu}} \right] \right\|_2 \leq \varepsilon.$$

**Success probability:** Lower bounded by $1 - \delta$.
**Queries:** Number of calls to $U_{\mathbb{P}}$ and $O_X$:

$$\widetilde{\mathcal{O}}\left( \min\left\{ \frac{\overline{p}}{\varepsilon^2}, \frac{\sqrt{d}}{\varepsilon} \right\} \cdot \sqrt{\overline{p}} \right). \tag{3.4.1}$$

**Procedure:** $\textsc{Ring-mean-estimation}(X,\ \overline{\mu},\ a,\ b,\ \varepsilon,\ \overline{p},\ \delta,\ U_{\mathbb{P}},\ O_X)$:

1: Let $\widetilde{\mu}^{(a,b)}_{X,\overline{\mu}} = \textsc{Bounded-mean-estimation}(R^{(a,b)}_{X,\overline{\mu}},\ \varepsilon/(4\overline{p}),\ \delta/2,\ U',\ O')$.

2: Output $\widetilde{\mu} = 4\overline{p}\widetilde{\mu}_{X,\overline{\mu}}^{(a,b)}$.

---

**Proof of the properties of Algorithm 3.4.3:**
First, we check that we satisfy the assumption in the linear amplitude amplification procedure, i.e., Algorithm 2.4.6. To that end, observe that the operator $O_X(U_{\mathbb{P}} \otimes I)$ performs the mapping

$$|0\rangle |0\rangle \mapsto \sum_{\omega\in\Omega} \sqrt{\mathbb{P}(\omega)}\, |\omega\rangle |X(\omega)\rangle ,$$

and hence the total squared overlap of the state on the right-hand side with the subspace that is left invariant by $C$ is exactly

$$\sum_{\substack{\omega\in\Omega \\ a\leq \|X(\omega)-\overline{\mu}\|_2 < b}} \mathbb{P}(\omega) = \mathbb{P}[a \leq \|X - \overline{\mu}\|_2 < b] = p_{X,\overline{\mu}}^{(a,b)} \leq \overline{p}.$$

Thus, multiplying by $1/(4\overline{p})$ gives at most $1/4$, which means that the assumption required by the linear amplitude amplification routine is satisfied.

Next, observe from Algorithm 2.4.6 that $U'$ calls $U_{\mathbb{P}}$ and $O_X$ a total of $\widetilde{O}(1/\sqrt{\overline{p}})$ times, and that the other derived circuits don't query $U_{\mathbb{P}}$ and $O_X$ at all. Furthermore, from Algorithm 3.3.6, we observe that the number of calls to $U'$ is $K$ defined in Equation (3.3.5). Multiplying the two indeed proves the number of queries claimed in Equation (3.4.1).

Thus, it remains to check the claimed lower bound on the success probability. To that end, we observe directly from Definition 3.4.2 that $R_{X,\overline{\mu}}^{(a,b)}$ indeed is a random variable whose range is bounded by 1 in $\ell_2$-norm. We also observe that $U'$, acting on $\mathbb{C}^\Omega \otimes \mathbb{C}^R$, prepares the superposition

$$U' : |0\rangle |0\rangle \mapsto \sum_{\substack{\omega\in\Omega \\ a\leq \|X(\omega)-\overline{\mu}\|_2 < b}} \sqrt{\frac{\mathbb{P}(\omega)}{4\overline{p}}}\, |\omega'\rangle + |\perp\rangle ,$$

up to norm precision $\delta/(2K)$, where we used the shorthand notation $|\omega'\rangle = |\omega\rangle |X(\omega)\rangle$, and $|\perp\rangle$ only has support on the states $|\omega\rangle |X(\omega)\rangle$ where we have $\|X(\omega) - \overline{\mu}\|_2 \notin [a, b)$. Furthermore, the circuit $O'$ acting on $\mathbb{C}^\Omega \otimes \mathbb{C}^R \otimes \mathbb{C}^{R'}$, implements the operation

$$O' : |\omega'\rangle |0\rangle = |\omega\rangle |X(\omega)\rangle |0\rangle \mapsto |\omega\rangle |X(\omega)\rangle \left|R_{X,\overline{\mu}}^{(a,b)}(\omega)\right\rangle = |\omega'\rangle \left|R_{X,\overline{\mu}}^{(a,b)}(\omega)\right\rangle .$$

Thus, $U'$ is a probability oracle of the probability distribution $\mathbb{P}'$ with an extra multiplicative factor of $1/(4\overline{p})$ for all $\omega$ for which $R_{X,\overline{\mu}}^{(a,b)}(\omega)$ is non-zero, i.e., $\mathbb{P}'(\omega) = \mathbb{P}(\omega)/(4\overline{p})$ for all $\omega \in \Omega$ that satisfy $a \leq \|X(\omega) - \overline{\mu}\|_2 < b$. Furthermore, $O'$ is a random variable oracle for the ring variable $R_{X,\overline{\mu}}^{(a,b)}$.

Since we call $U'$ a total of $K$ times in the bounded mean estimation algorithm, i.e., Algorithm 3.3.6, the total norm error caused by the imperfect implementation of $U'$ is at most $\delta/2$. Since Algorithm 3.3.6 succeeds with probability at least $1 - \delta/2$, we find with probability at least $1 - \delta$ that

$$\left\| \widetilde{\mu}_{X,\overline{\mu}}^{(a,b)} - \frac{1}{4\overline{p}} \cdot \mathbb{E}[R_{X,\overline{\mu}}^{(a,b)}] \right\|_2 \leq \frac{\varepsilon}{4\overline{p}},$$

and from there it follows directly that

$$\left\| \widetilde{\mu} - \mathbb{E}[R_{X,\overline{\mu}}^{(a,b)}] \right\|_2 \leq \varepsilon.$$

This completes the proof. □

Note that the number of queries performed by the ring estimation algorithm is better than what one would naively expect from the bounded mean estimation routine. Indeed, if one were to run the bounded mean estimation routine directly with precision $\varepsilon$, then one would end up making $\widetilde{\mathcal{O}}(\min\{1/\varepsilon^2, \sqrt{d}/\varepsilon\})$ queries, whereas the ring estimation routine attains an improvement of at least a factor $\sqrt{\overline{p}}$. Hence, the ring estimation procedure is especially efficient in the settings where we can choose $\overline{p}$ to be very small.

To exploit the above observation, the mean estimation algorithm divides up $\mathbb{R}^d$ into several concentric rings centered at a crude estimation of the mean, $\overline{\mu}$, with an increasing sequence of radii $0 = a_0 < a_1 < \cdots < a_k$. We then find that

$$\mathbb{E}[X] \approx \overline{\mu} + \sum_{j=1}^{k} a_j \mathbb{E}[R_{X,\overline{\mu}}^{(a_{j-1},a_j)}],$$

where we have to choose our last radius $a_k$ big enough so that the approximation is sufficiently close. We then approximate each of the above terms individually. A visualization of the concentric rings and their radii are depicted in Figure 3.4.1.

The crucial insight is now that we can use Lemma 3.4.1 to upper bound the ring probabilities $p_{X,\overline{\mu}}^{(a,b)}$, which can subsequently be used to obtain more efficient ring estimation routines, i.e., Algorithm 3.4.3.

It remains to show how to choose the radii $a_1, \ldots, a_k$. We do this in the algorithm statement below.

**Algorithm 3.4.4:** Mean estimation, with a known upper bound on $\mathrm{Tr}[\Sigma]$

**Input:**

1: $X : \Omega \to R \subseteq \mathbb{R}^d$: a random variable.
2: $S > 0$: an upper bound on $\mathrm{Tr}[\Sigma]$, where $\Sigma = \mathrm{Cov}(X)$.
3: $\varepsilon > 0$: the desired precision.
4: $\delta > 0$: the failure probability tolerance.

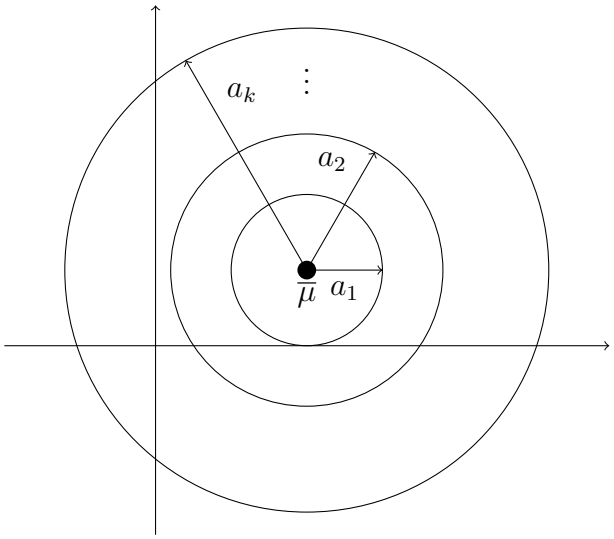



Figure 3.4.1: Visualization of the algorithm that performs mean estimation. The space $\mathbb{R}^d$ is divided up into rings that are separated by concentric circles around a crude estimate of the mean, $\overline{\mu}$.

5: $U_{\mathbb{P}}$: the probability distribution oracle, as defined in Definition 3.2.2.
6: $O_X$: the random variable oracle, as defined in Definition 3.2.2.

**Derived objects:**
1: $k = \lceil \log(8S/\varepsilon^2) \rceil$.
2: $a_0 = 0$.
3: For all $j \in [k]$, we let $a_j = \sqrt{2^j \cdot 2S}$.

**Output:** A vector $\widetilde{\mu} \in \mathbb{R}^d$ such that $\|\widetilde{\mu} - \mathbb{E}[X]\|_2 \leq \varepsilon$.

**Queries:** The number of calls to $U_{\mathbb{P}}$ and $O_X$ is

$$\widetilde{\mathcal{O}}\left(\min\left\{\frac{S}{\varepsilon^2}, \frac{\sqrt{dS}}{\varepsilon}\right\}\right). \tag{3.4.2}$$

**Procedure:** Mean-estimation($X$, $S$, $\varepsilon$, $\delta$, $U_{\mathbb{P}}$, $O_X$):
1: Let $\overline{\mu}$ = Classical-mean-est($X$, 12, $\delta/2$, $U_{\mathbb{P}}$, $O_X$).
2: For $j = 1, \ldots, k$, let $\widetilde{\mu}_j$ = Ring-mean-estimation($X$, $\overline{\mu}$, $a_{j-1}$, $a_j$, $\varepsilon/(2ka_j)$, $1/2^{j-1}$, $\delta/(2k)$, $U_{\mathbb{P}}$, $O_X$).
3: Output $\widetilde{\mu} = \overline{\mu} + a_1\widetilde{\mu}_1 + \cdots + a_k\widetilde{\mu}_k$.

---

**Proof of the properties of Algorithm 3.4.4:**
We start by checking the claim on the number of queries. To that end, observe that in the first step, only $\widetilde{O}(1)$ queries to $U_{\mathbb{P}}$ and $O_X$ are performed.

In the second step, we call Algorithm 3.4.3 a total of $k$ times. According to

Equation (3.4.1), the number of calls in the $j$th iteration, with $j \in [k]$, is

$$\widetilde{\mathcal{O}}\left(\min\left\{\frac{8k^2a_j^2}{2^j\varepsilon^2}, \frac{2ka_j\sqrt{d}}{\varepsilon}\right\} \cdot \sqrt{\frac{2}{2^j}}\right).$$

Since $\min\{a,b\} + \min\{c,d\} \leq \min\{a+c, b+d\}$, for all $a,b,c,d \in \mathbb{R}$, we obtain that the total number of queries can be written as $\widetilde{O}(\min\{A,B\})$, where

$$A = \sum_{j=1}^{k} \frac{8\sqrt{2}k^2a_j^2}{2^{3j/2}\varepsilon^2} \leq \frac{16\sqrt{2}k^2S}{\varepsilon^2} \cdot \sum_{j=1}^{\infty}\left(\frac{1}{\sqrt{2}}\right)^j = \frac{16\sqrt{2}k^2S}{\varepsilon^2(\sqrt{2}-1)} = \widetilde{\mathcal{O}}\left(\frac{S}{\varepsilon^2}\right),$$

and similarly

$$B = \sum_{j=1}^{k} \frac{2\sqrt{2}ka_j\sqrt{d}}{2^{j/2}\varepsilon} = \frac{2\sqrt{2}k\sqrt{2dS}}{\varepsilon} \cdot \sum_{j=1}^{k} \frac{2^{j/2}}{2^{j/2}} = \frac{2\sqrt{2}k^2\sqrt{2dS}}{\varepsilon} = \widetilde{\mathcal{O}}\left(\frac{\sqrt{dS}}{\varepsilon}\right).$$

Thus, we indeed obtain the expression from Equation (3.4.2).

It remains to prove the lower bound on the claimed success probability. To that end, we can assume that all subroutines succeed, since their cumulative failure probability is at most $\delta$. In particular, from the properties of Algorithm 3.2.3, we find that $\|\overline{\mu} - \mathbb{E}[X]\|_2 \leq 2\sqrt{3\,\mathrm{Tr}[\Sigma]/12} = \sqrt{\mathrm{Tr}[\Sigma]} \leq \sqrt{S}$, and thus the assumption in Lemma 3.4.1 is satisfied.

Next, we check that for all $j \in [k]$, the ring probability $p_{X,\overline{\mu}}^{(a_{j-1},a_j)}$ is indeed upper bounded by $1/2^{j-1}$, as we asserted by choosing this as our upper bound when we call the ring estimation algorithm. To that end, observe that the bound is trivial for $j = 1$, and whenever $j > 1$, we find by Lemma 3.4.1 that

$$p_{X,\overline{\mu}}^{(a_{j-1},a_j)} \leq \mathbb{P}\left[\|X - \overline{\mu}\|_2 \geq a_{j-1}\right] = \mathbb{P}\left[\|X - \overline{\mu}\|_2 \geq \sqrt{2^{j-1} \cdot 2S}\right] \leq \frac{1}{2^{j-1}}.$$

Now, it remains to check that $\widetilde{\mu}$ indeed approximates $\mathbb{E}[X]$ precisely enough. To that end, observe by the triangle inequality that

$$\|\widetilde{\mu} - \mathbb{E}[X]\|_2 \leq \sum_{j=1}^{k} a_j \left\|\widetilde{\mu}_j - \mathbb{E}[R_{X,\overline{\mu}}^{(a_{j-1},a_j)}]\right\|_2 + \left\|\mathbb{E}[X] - \overline{\mu} - \sum_{j=1}^{k} a_j\mathbb{E}[R_{X,\overline{\mu}}^{(a_{j-1},a_j)}]\right\|_2.$$

We observe by the properties of Algorithm 3.4.3 that the first terms are all bounded by $\varepsilon/(2k)$, and thus it remains to check that the last term in the above expression is upper bounded by $\varepsilon/2$. To that end, observe by the Cauchy–Schwarz

inequality that

$$\begin{aligned}
\left\|\mathbb{E}[X] - \overline{\mu} - \sum_{j=1}^{k} a_j \mathbb{E}[R_{X,\overline{\mu}}^{(a_{j-1},a_j)}]\right\|_2 &= \left\| \sum_{\substack{\omega\in\Omega \\ \|X(\omega)-\overline{\mu}\|_2 \geq a_k}} \mathbb{P}(\omega)\,(X(\omega)-\overline{\mu}) \right\|_2 \\
&\leq \sum_{\substack{\omega\in\Omega \\ \|X(\omega)-\overline{\mu}\|_2 \geq a_k}} \mathbb{P}(\omega)\,\|X(\omega)-\overline{\mu}\|_2 \\
&\leq \sqrt{\sum_{\substack{\omega\in\Omega \\ \|X(\omega)-\overline{\mu}\|_2 \geq a_k}} \mathbb{P}(\omega) \cdot \sum_{\substack{\omega\in\Omega \\ \|X(\omega)-\overline{\mu}\|_2 \geq a_k}} \mathbb{P}(\omega)\,\|X(\omega)-\overline{\mu}\|_2^2} \\
&\leq \sqrt{\mathbb{P}[\|X-\overline{\mu}\|_2 \geq \sqrt{2^k \cdot 2S}] \cdot \mathbb{E}[\|X-\overline{\mu}\|_2^2]} \leq \sqrt{\frac{\varepsilon^2}{8S} \cdot 2S} = \frac{\varepsilon}{2},
\end{aligned}$$

where in the last inequality, we used Lemma 3.4.1. This completes the proof. □

In the above algorithm, we have chosen $k$ rings, where $k$ is logarithmic in all the relevant parameters. In the high-precision regime, the number of queries performed to estimate the mean of each of the ring variables is roughly equal, namely roughly $\widetilde{\mathcal{O}}(\sqrt{dS}/\varepsilon)$. However, intuitively, the contribution of these queries is different. In the inner-most rings, we don't have a very strong upper bound on the ring probability, so we can spend all the effort in estimating the mean of the ring variable through the bounded mean estimation routine. On the other hand, in the outer-most rings, we have a very strong upper bound on the ring probability, and as such we spend a lot of work in the linear amplitude amplification routine in Algorithm 3.4.3, and a bit less in the bounded mean estimation routine.

### 3.4.2 Unknown upper bound on $\mathrm{Tr}[\Sigma]$

In the previous subsection, we developed a quantum algorithm that estimates the mean of a random variable, if we a priori know an upper bound on the trace of the covariance matrix, i.e., we know an $S > 0$ such that $\mathrm{Tr}[\Sigma] \leq S$. However, this result is qualitatively different from the classical result, Algorithm 3.2.3, since there we don't need any prior knowledge about $\mathrm{Tr}[\Sigma]$, other than that it is well-defined. So, one might wonder whether there is a way to improve on the mean estimation algorithm from the previous subsection, to remove the necessity of knowing an upper bound on $\mathrm{Tr}[\Sigma]$ a priori.

In this subsection, we answer this question affirmatively. The core idea is to estimate the radii of the rings, i.e., the $a_j$'s in Algorithm 3.4.4, using a routine called *quantile estimation*. The approach we present here is a slightly modified version of the approach that was first introduced in [Ham21].

First, we define quantiles formally.

**3.4.5.** Definition (Quantiles and approximate quantiles).
Let $(\Omega, 2^\Omega, \mathbb{P})$ be a probability space, and let $X : \Omega \to \mathbb{R}$ be a random variable.

1. Let $p \in [0,1]$, and let $x \in \mathbb{R}$ be a real number such that $\mathbb{P}[X \leq x] = p$. Then $x$ is a *$p$-quantile of $X$*.
2. Let $p \in [0,1]$, and let $\varepsilon > 0$. Let $x \in \mathbb{R}$ be a real number such that $|p - \mathbb{P}[X \leq x]| \leq \varepsilon$. Then $x$ is an *$\varepsilon$-approximate $p$-quantile* of $X$. ◀

To make the exposition of the algorithm we develop in this section easier, we assume that the probability distribution is sufficiently smooth, so that every quantile that we refer to below actually exists. With some extra work this assumption can most likely be removed, but it would make the analyses somewhat more involved. We leave this for future work.[1]

Now, we present a quantum algorithm that finds an approximate $p$-quantile.

**Algorithm 3.4.6:** Quantile estimation

**Input:**
1: $X : \Omega \to \mathbb{R}$: a random variable on a probability space $(\Omega, 2^\Omega, \mathbb{P})$.
2: $p \in (0,1)$: the probability value of the quantile.
3: $\delta > 0$: the failure probability tolerance parameter.
4: $\varepsilon > 0$: the proximity parameter.
5: $U_\mathbb{P}$: the probability distribution oracle, as defined in Definition 3.2.2.
6: $O_X$: the random variable oracle, as defined in Definition 3.2.2.
**Derived objects:**
1: $N = \lceil \ln(4/\delta)/(2\varepsilon^2) \rceil$.
**Output:** An $\varepsilon$-approximate $p$-quantile.
**Success probability:** Lower bounded by $1 - \delta$.
**Queries:** Number of calls to $U_\mathbb{P}$ and $O_X$: $\widetilde{\mathcal{O}}(1/\varepsilon^2)$.
**Procedure:** Quant-Est($X$, $p$, $\varepsilon$, $\delta$, $U_\mathbb{P}$, $O_X$):
1: Generate $N$ classical samples: $x_1, \ldots, x_N$.
2: Sort the generated samples.
3: Return $x_\ell$, where $\ell = \text{round}(pN - 1/2) + 1/2$. Denote the outcome by $q$.

**Proof of the properties of Algorithm 3.4.6:**
The claim on the number of queries to $U_\mathbb{P}$ and $O_X$ follows immediately from the choice of $N$. Thus, it remains to verify the claimed lower bound on the success probability.

To that end, let $z \in \mathbb{R}$ arbitrarily, and for all $j \in [N]$, let $X_j$ be the Bernoulli variable that is 1 if and only if $x_j \leq z$. Then, $\mathbb{E}[X_j] = \mathbb{P}[X \leq z]$.

[1] One particularly hideous way of removing this assumption is to take the convolution product of the distribution of $X$ with an infinitesimally small symmetric kernel function – this does not change the mean and allows us to control the smoothness of our distribution easily. We leave it for future work to find more elegant ways of dealing with this detail.

Next, let $z \in \mathbb{R}$ be the smallest value for which $\mathbb{P}[X \leq z] \geq p - \varepsilon$. If $q \leq z$, then at least $pN$ of the $x_j$'s must have been at most $z$, and so by Hoeffding's inequality,

$$\begin{aligned}\mathbb{P}[q \leq z] &\leq \mathbb{P}\left[\sum_{j=1}^{n} X_j \geq pN\right] \\ &\leq \mathbb{P}\left[\left|\sum_{j=1}^{N} X_j - (p-\varepsilon)N\right| \geq N\left|p - (p-\varepsilon)\right|\right] \\ &\leq 2\exp\left(-2N\varepsilon^2\right) \leq \delta/2.\end{aligned}$$

A similar argument shows that when we let $z \in \mathbb{R}$ be the largest value such that $\mathbb{P}[X \leq z] \leq p + \varepsilon$, then $\mathbb{P}[q \geq z] \leq \delta/2$ as well. Thus, the probability that $q$ is an $\varepsilon$-approximate $p$-quantile of $X$ is at least $1 - \delta$. This completes the proof. □

Note that the above analysis can be improved by substituting Hoeffding's inequality by the Chernoff bound. The improved analysis especially makes a difference in the limits where $p$ is close to 0 or 1. Since we will only use Algorithm 3.4.6 in the setting where $p$ is far away from the endpoints of the interval $[0, 1]$, we refrain from giving that improvement here.

The idea, now, is to start the algorithm the same way as in the previous section, i.e., by finding a $\overline{\mu} \in \mathbb{R}^d$ that approximates $\mu = \mathbb{E}[X]$ up to precision $\sqrt{\mathrm{Tr}[\Sigma]}$ in $\ell_2$-norm. Then, the idea is to find a value for $a_1$, such that $\mathbb{P}[\|X - \overline{\mu}\|_2 \geq a_1] \leq 1/2$. Subsequently, one can use the value for $a_1$ to build a circuit $U'_{\mathbb{P}}$ that samples over all $\omega$'s under the constraint that $\|X(\omega) - \overline{\mu}\|_2 \geq a_1$. Using this new $U'_{\mathbb{P}}$, one can find a value $a_2 > a_1$, such that $\mathbb{P}[\|X - \overline{\mu}\|_2 \geq a_2] \leq 1/4$. This sequence of $a_j$'s generated in this way then replaces the sequence of $a_j$'s used in Algorithm 3.4.4.

The following algorithm makes this sketch precise.

**Algorithm 3.4.7:** Circle radius estimation

**Input:**

1: $X : \Omega \to R \subseteq \mathbb{R}$: a random variable on a probability space $(\Omega, 2^{\Omega}, \mathbb{P})$.
2: $n \in \mathbb{N}$: a parameter governing the number of samples taken.
3: $\delta > 0$: the failure probability tolerance.
4: $U_{\mathbb{P}}$: the probability distribution oracle, as defined in Definition 3.2.2.
5: $O_X$: the random variable oracle, as defined in Definition 3.2.2.

**Derived objects:**

1: $k = \lceil 2\log(n) \rceil$.
2: $a_0 = 0$.

**Output:** A sequence $a_1, \ldots, a_k$, such that for all $j \in [k]$,

$$1/2^{j+1} \leq \mathbb{P}[X \geq a_j] \leq 1/2^j.$$

**Success probability:** Lower bounded by $1 - \delta$.
**Queries:** Number of queries to $U_{\mathbb{P}}$ and $O_X$: $\widetilde{O}(n)$.
**Procedure:** CIRCLE-RADII-EST($X$, $n$, $\delta$, $U_{\mathbb{P}}$, $O_X$):
1: For $j = 1, \ldots, k$:

1. Let $C^{(j)}$ be a circuit acting on $\mathbb{C}^R$ that performs the operation

$$|x\rangle \mapsto \begin{cases} |x\rangle\,, & \text{if } x \geq a_{j-1} \\ -|x\rangle\,, & \text{otherwise.} \end{cases}$$

2. Define $U_{\mathbb{P}}^{(j)} = \text{FIXED-POINT-AMPL}(U_{\mathbb{P}}, O_X^{\dagger}(I \otimes C^{(j)})O_X, 1/\sqrt{2^j}, \delta/(2kK))$, where $K = \widetilde{O}(k^2)$ is the number of times $U_{\mathbb{P}}^{(j)}$ is called in the next step.
3. Let $a_j = \text{QUANT-EST}(X, 1/2 + 1/(8k), 1/(8k), \delta/(2k), U_{\mathbb{P}}^{(j)}, O_X)$.

---

**Proof of the properties of Algorithm 3.4.7:**
First, we check the claimed query complexities. To that end, we observe from the properties of the fixed-point amplitude amplification algorithm, Algorithm 2.4.5, that in the $j$th iteration, the operation $U_{\mathbb{P}}^{(j)}$ makes $\widetilde{O}(\sqrt{2^j})$ calls to $U_{\mathbb{P}}$ and $O_X$. Moreover, we observe from Algorithm 3.4.6 that naive quantile estimation algorithm makes $\mathcal{O}(k^2) = \widetilde{\mathcal{O}}(1)$ queries to $U_{\mathbb{P}}^{(j)}$ and $O_X$. By multiplying both together and summing over $j \in [k]$, we obtain that the total number of queries to $U_{\mathbb{P}}$ and $O_X$ is $\widetilde{\mathcal{O}}(A)$, with

$$A = \sum_{j=1}^{k} \sqrt{2^j} \leq k\sqrt{2^k} = \widetilde{\mathcal{O}}(n).$$

Thus, it remains to check the claimed lower bound on the success probability. To that end, we first of all assume that all the subroutines succeed, since their cumulative failure probability is at most $\delta$. Thus, it remains to show that the algorithm indeed produces a sequence of quantiles $a_1, \ldots, a_k$ such that for all $j \in [k]$, we have $1/2^{j+1} \leq \mathbb{P}[X \geq a_j] \leq 1/2^j$.

To that end, we prove a slightly stronger statement by induction. We prove that

$$\left(\frac{1}{2} - \frac{1}{4k}\right)^j \leq \mathbb{P}[X \geq a_j] \leq \left(\frac{1}{2}\right)^j. \tag{3.4.3}$$

Indeed, when $j = 1$, we find immediately from the properties of Algorithm 3.4.6 that $1/2 \leq \mathbb{P}[X \leq a_1] \leq 1/2 + 1/(4k)$, from which the statement follows. This provides the basis of induction.

Now, suppose that Equation (3.4.3) holds for some $j \in [k-1]$. Then, we find by the properties of Algorithm 3.4.6 that

$$\frac{1}{2} \leq \mathbb{P}[X \leq a_{j+1} | X \geq a_j] \leq \frac{1}{2} + \frac{1}{4k}.$$

Thus, we find that

$$\left(\frac{1}{2}-\frac{1}{4k}\right)^{j+1} \leq \mathbb{P}[X > a_j] \cdot \mathbb{P}[X > a_{j+1} | X \geq a_j] = \mathbb{P}[X > a_{j+1}],$$

and similarly

$$\mathbb{P}[X > a_{j+1}] = \mathbb{P}[X > a_j] \cdot \mathbb{P}[X > a_{j+1} | X \geq a_j] \leq \frac{1}{2^j} \cdot \frac{1}{2} = \frac{1}{2^{j+1}}.$$

This proves the induction hypothesis. Finally, we observe that for all $j \in [k]$,

$$\left(\frac{1}{2}-\frac{1}{4k}\right)^{j} \geq \frac{1}{2^j} \cdot \left(1-\frac{1}{2k}\right)^{k} > \frac{1}{2^j} \cdot \frac{1}{\sqrt{e}} > \frac{1}{2^{j+1}}.$$

This completes the proof. □

Now, we show how using the above quantiles of the random variable $\|X - \overline{\mu}\|_2$ gives rise to a quantum algorithm that estimates the mean of $X$ without any prior knowledge on $\mathrm{Tr}[\Sigma]$.

**Algorithm 3.4.8:** Mean estimation without prior knowledge of $\mathrm{Tr}[\Sigma]$

**Input:**

1: $X : \Omega \to R \subseteq \mathbb{R}^d$: a random variable on a probability space $(\Omega, 2^\Omega, \mathbb{P})$.
2: $n \in \mathbb{N}$: a parameter governing the number of samples taken.
3: $\delta > 0$: the failure probability tolerance.
4: $U_{\mathbb{P}}$: the probability distribution oracle, as defined in Definition 3.2.2.
5: $O_X$: the random variable oracle, as defined in Definition 3.2.2.

**Derived objects:**

1: $k = \lceil 2\log(n) \rceil$.
2: $a_0 = 0$.
3: $U' = O_X(U_{\mathbb{P}} \otimes I)$.

**Output:** A vector $\widetilde{\mu} \in \mathbb{R}^d$ such that

$$\|\widetilde{\mu} - \mathbb{E}[X]\|_2 = \widetilde{\mathcal{O}}\left(\min\left\{\sqrt{\frac{\mathrm{Tr}[\Sigma]}{n}}, \frac{\sqrt{d\,\mathrm{Tr}[\Sigma]}}{n}\right\}\right).$$

**Queries:** Number of calls to $U_{\mathbb{P}}$ and $O_X$: $\widetilde{O}(n)$.

**Procedure:** MEAN-EST($X$, $n$, $\delta$, $U_{\mathbb{P}}$, $O_X$):

1: Let $\overline{\mu}$ = CLASSICAL-MEAN-EST($X$, 12, $\delta/3$, $U_{\mathbb{P}}$, $O_X$).
2: Let $(a_1, \ldots, a_k)$ = CIRCLE-RADII-EST($\|X - \overline{\mu}\|_2$, $n$, $\delta/3$, $U'$, $C_{\overline{\mu}}$), where $R'_{\overline{\mu}} = \{\|x - \overline{\mu}\|_2 : x \in R\}$ and $C_{\overline{\mu}}$ be the circuit that acts on $\mathbb{C}^R \otimes \mathbb{C}^{R'}$, and implements the operation

$$|x\rangle\,|0\rangle \mapsto |x\rangle\,|\|x - \overline{\mu}\|_2\rangle\,.$$

3: For $j = 1, \ldots, k$, let $\widetilde{\mu}_j = \text{RING-MEAN-ESTIMATION}(X, \overline{\mu}, a_{j-1}, a_j, \varepsilon_j/k, 1/2^{j-1}, \delta/(3k), U_{\mathbb{P}}, O_X)$, with

$$\varepsilon_j = \frac{1}{\sqrt{2^j}} \cdot \min\left\{\sqrt{\frac{1}{n}}, \frac{\sqrt{d}}{n}\right\}.$$

4: Output $\widetilde{\mu} = \overline{\mu} + \widetilde{\mu}_1 + \cdots + \widetilde{\mu}_k$.

---

**Proof of the properties of Algorithm 3.4.8:**
First we check the number of queries performed throughout the algorithm. In the first step, from the properties of Algorithm 3.2.3, we deduce that we make only $\widetilde{\mathcal{O}}(1)$ queries to $U_{\mathbb{P}}$ and $O_X$. In the second step, we find by the properties of Algorithm 3.4.7 that we make a total of $\widetilde{\mathcal{O}}(n)$ queries to $U'$, which in turn is constructed using a single query to both $U_{\mathbb{P}}$ and $O_X$. Then, for all $j \in [k]$, in the $j$th iteration in the third step, we make a number of queries to $U_{\mathbb{P}}$ and $O_X$ that is

$$\widetilde{\mathcal{O}}\left(\min\left\{\frac{2}{2^j\varepsilon_j^2}, \frac{\sqrt{d}}{\varepsilon_j}\right\} \cdot \sqrt{\frac{2}{2^j}}\right).$$

Hence, if we choose $\varepsilon_j = \sqrt{1/(2^j n)}$, then the first term in the above expressions ensures that the resulting number of queries is $\widetilde{\mathcal{O}}(n)$. On the other hand, if we choose $\varepsilon_j = \sqrt{d/2^j}/n$, then the second term ensures that the resulting number of queries is $\widetilde{\mathcal{O}}(n)$. This completes the proof of the claimed number of queries.

Thus, it remains to prove the lower bound on the success probability. To that end, we first assume that all the subroutines succeed, since their cumulative failure probabilities sum to $\delta$. We also observe that assuming that the first step succeeds implies that the preconditions for Lemma 3.5.2 are satisfied, and assuming that the second step succeeds provides the guarantee that the assumptions required in step 3 are satisfied.

Next, let $j \in [k]$. Suppose that $a_j > \sqrt{2\operatorname{Tr}[\Sigma] \cdot 2^j}$. Then, by choosing $S = \operatorname{Tr}[\Sigma]$ in Lemma 3.5.2, we find that

$$\mathbb{P}\left[\|X - \overline{\mu}\|_2 \geq a_j\right] \leq \mathbb{P}\left[\|X - \overline{\mu}\|_2 > \sqrt{2\operatorname{Tr}[\Sigma] \cdot 2^{j+1}}\right] < \frac{1}{2^{j+1}}.$$

This contradicts the choice of the $a_j$'s, and as such we observe that for every $j \in [k]$, we have $a_j \leq \sqrt{2\operatorname{Tr}[\Sigma] \cdot 2^j}$.

Next, as in the analysis of Algorithm 3.4.4, we use the triangle inequality to bound

$$\|\widetilde{\mu} - \mathbb{E}[X]\|_2 \leq \sum_{j=1}^{k} a_j \left\|\widetilde{\mu}_j - \mathbb{E}[R_{X,\overline{\mu}}^{(a_{j-1},a_j)}]\right\|_2 + \left\|\mathbb{E}[X] - \overline{\mu} - \sum_{j=1}^{k} a_j \mathbb{E}[R_{X,\overline{\mu}}^{(a_{j-1},a_j)}]\right\|_2.$$

From the properties of the ring variable estimation routine, Algorithm 3.4.3, we obtain that the the $j$th term is at most $a_j\varepsilon_j/k$, and so we obtain.

$$\begin{aligned}\sum_{j=1}^{k} a_j \left\|\widetilde{\mu}_j - \mathbb{E}[R_{X,\overline{\mu}}^{(a_{j-1},a_j)}]\right\|_2 &\leq \sum_{j=1}^{n} \frac{a_j\varepsilon_j}{k} \leq \sqrt{2\operatorname{Tr}[\Sigma]}\cdot\min\left\{\sqrt{\frac{1}{n}}, \frac{\sqrt{d}}{n}\right\} \\ &= \widetilde{\mathcal{O}}\left(\min\left\{\sqrt{\frac{\operatorname{Tr}[\Sigma]}{n}}, \frac{\sqrt{d\operatorname{Tr}[\Sigma]}}{n}\right\}\right).\end{aligned}$$

Thus, it remains to prove that the final term is bounded by the same expression. To that end, we obtain using the Cauchy–Schwarz's inequality that

$$\begin{aligned}\left\|\mathbb{E}[X] - \overline{\mu} - \sum_{j=1}^{k} a_j\mathbb{E}[R_{X,\overline{\mu}}^{(a_{j-1},a_j)}]\right\|_2 &\leq \sum_{\substack{\omega\in\Omega \\ \|X(\omega)-\overline{\mu}\|_2 > a_k}} \mathbb{P}(\omega)\,\|X(\omega)-\overline{\mu}\|_2 \\ &\leq \sqrt{\mathbb{P}\left[\|X-\overline{\mu}\|_2 \geq a_k\right]\cdot\mathbb{E}\left[\|X-\overline{\mu}\|_2^2\right]} \leq \sqrt{\frac{1}{2^k}\cdot 2\operatorname{Tr}[\Sigma]} \leq \frac{\sqrt{2\operatorname{Tr}[\Sigma]}}{n},\end{aligned}$$

where in the last line we used Lemma 3.5.2 with $S = \operatorname{Tr}[\Sigma]$. The resulting expression is smaller than both expressions in the accuracy claim, and thus the proof is complete. □

This completes the construction of the mean estimation algorithm. In the next section, we will prove that the approach presented here is optimal, up to polylogarithmic factors.

## 3.5 Lower bound

In this section, we prove that the quantum mean estimation algorithm that we designed in this chapter is optimal in terms of the number of calls to the probability distribution oracle $U_{\mathbb{P}}$, up to polylogarithmic factors.

As is customary with lower bounding, we would like to embed a problem whose hardness has already been shown before in the setting we consider here, in order to conclude that the mean estimation problem must be at least as hard to solve. We start by considering the problem of recovering a constant fraction of the bits in a bit string when we are given access to it by means of a fractional phase oracle.

**3.5.1.** Lemma. *Let $\varepsilon \in (0,\pi]$, $d \in \mathbb{N}$, and suppose that we have access to a bit string $\mathbf{b} \in \{0,1\}^d$ through controlled calls to a fractional phase oracle $F_\varepsilon : |j\rangle \mapsto e^{i\varepsilon b_j}|j\rangle$. Then, in order to find a bit string $\widetilde{\mathbf{b}} \in \{0,1\}^d$ such that $\|\widetilde{\mathbf{b}} - \mathbf{b}\|_1 \leq d/4$ with probability at least $2/3$, we must make at least $\Omega(d/\varepsilon)$ calls to $F_\varepsilon$.*

**Proof:**
First, we argue that it is sufficient to consider the case where $\varepsilon = \pi$. Indeed, in general, the query complexity of any problem is increased by a multiplicative factor of $\Theta(1/\varepsilon)$, when one changes the input model from a regular phase oracle $F_\pi$ to a fractional phase oracle $F_\varepsilon$. In Appendix B of [LMR+11], this is proven for problems that can be phrased as computing a binary function. However, since the problem we consider here does not have a unique correct output on every given input, we must combine their technique with the general adversary bound for relations, as derived by [Bel15], to arrive at the desired result. More details can be found in [CJ21].

Thus, it remains to focus on the case where $\varepsilon = \pi$. Suppose that we have an algorithm $\mathcal{A}$ that finds a bit string $\widetilde{\mathbf{b}} \in \{0,1\}^d$ such that with probability at least $2/3$, we have $\|\widetilde{\mathbf{b}} - \mathbf{b}\|_1 \leq d/4$, i.e., $\widetilde{\mathbf{b}}$ and $\mathbf{b}$ differ in at most $d/4$ bits. Then, we can let $\mathcal{B}$ be the quantum algorithm that first runs $\mathcal{A}$ to obtain such a bit string $\widetilde{\mathbf{b}}$, and then selects uniformly at random a bit string $\overline{\mathbf{b}} \in \{0,1\}^d$ that satisfies $\|\overline{\mathbf{b}} - \widetilde{\mathbf{b}}\|_1 \leq d/4$. We have $M = \sum_{t=0}^{\lfloor d/4 \rfloor} \binom{d}{t}$ possible choices, which implies that the probability of this algorithm outputting $\mathbf{b}$ exactly is lower bounded by $2/3 \cdot 1/M$. By the information-theoretic lower bound, i.e., Equation (4) in [FGG+99], the number of queries to $F_\pi$, performed by $\mathcal{B}$ and hence also by $\mathcal{A}$, denoted by $Q$, satisfies

$$2^d \leq \frac{3}{2} \cdot \sum_{t=0}^{\lfloor \frac{d}{4} \rfloor} \binom{d}{t} \sum_{t=0}^{Q} \binom{d}{t} \leq \frac{3}{2} \cdot 2^{d\left(H\left(\frac{1}{4}\right) + H\left(\frac{Q}{d}\right)\right)},$$

where in the final inequality, we used a well-known upper bound on sums of binomial coefficients, as proven for instance in Lemma 16.19 in [FG06], and $H(x) = -x\log(x) - (1-x)\log(1-x)$ is the binary entropy function. Taking logarithms on both sides yields that $H(Q/d) \geq 1 - H(1/4) + o(1)$, which implies that $Q = \Omega(d)$, completing the proof. □

We now show how the hardness of the problem considered in the previous lemma can be used to lower bound the query complexity in the mean estimation problem.

**3.5.2.** Lemma. *Let $d \in \mathbb{N}$, $0 < \varepsilon \leq 1/16$, and suppose that we have a quantum algorithm that finds an approximation $\widetilde{\mu}$ to the mean $\mu$ of any $d$-dimensional random variable with values contained in the unit ball in $\ell_2$ norm, using $n$ queries to $U_{\mathbb{P}}$, such that $\|\widetilde{\mu} - \mu\|_1 \leq \varepsilon$, with probability at least $2/3$. Then,*

$$n = \Omega\left(\frac{d}{\varepsilon}\right).$$

**Proof:**
Let $\varepsilon' = \arcsin(16\varepsilon)$, and suppose that we have access to some hidden bit string

$\mathbf{b} \in \{0,1\}^d$ by means of controlled calls to a fractional phase oracle $F_{\varepsilon'} : |j\rangle \mapsto e^{i\varepsilon' b_j} |j\rangle$. We know from Lemma 3.5.1 that it takes $\Omega(d/\varepsilon')$ calls to find a bit string $\widetilde{\mathbf{b}}$ such that $\|\widetilde{\mathbf{b}} - \mathbf{b}\|_1 \leq d/4$.

Now, let $\Omega = [d] \times \{0,1\}$, and for every $\mathbf{b} \in \{0,1\}^d$, let the probability measure $\mathbb{P}_{\mathbf{b}}$ on $\Omega$ be defined as

$$\mathbb{P}_{\mathbf{b}}(j,x) = \frac{1}{d}\cos^2\left(\frac{\pi}{4} + (-1)^x \frac{\varepsilon' b_j}{2}\right), \qquad \text{for all } j \in [d], x \in \{0,1\}.$$

Observe that with one call to $F_{\varepsilon'}$, we can implement

$$\frac{1}{\sqrt{2d}}\sum_{j=1}^{d}(|j\rangle|0\rangle + i|j\rangle|1\rangle) \overset{F_{\varepsilon'}}{\mapsto} \frac{1}{\sqrt{2d}}\sum_{j=1}^{d}(|j\rangle|0\rangle + ie^{i\varepsilon' b_j}|j\rangle|1\rangle) \tag{3.5.1a}$$

$$= \sum_{j=1}^{d} \frac{e^{i\frac{\pi}{4}+i\frac{\varepsilon' b_j}{2}}}{\sqrt{2d}}\left(e^{-i\frac{\pi}{4}-i\frac{\varepsilon' b_j}{2}}|j\rangle|0\rangle + e^{i\frac{\pi}{4}+i\frac{\varepsilon' b_j}{2}}|j\rangle|1\rangle\right) \tag{3.5.1b}$$

$$\overset{I\otimes(SH)}{\mapsto} \sum_{j=1}^{d} \frac{e^{i\frac{\pi}{4}+i\frac{\varepsilon' b_j}{2}}}{\sqrt{d}}\left(\cos\left(\frac{\pi}{4}+\frac{\varepsilon' b_j}{2}\right)|j\rangle|0\rangle + \sin\left(\frac{\pi}{4}+\frac{\varepsilon' b_j}{2}\right)|j\rangle|1\rangle\right) \tag{3.5.1c}$$

$$= \sum_{(j,x)\in\Omega} \frac{e^{i\frac{\pi}{4}+i\frac{\varepsilon' b_j}{2}}}{\sqrt{d}}\cos\left(\frac{\pi}{4}+(-1)^x\frac{\varepsilon' b_j}{2}\right)|j\rangle|x\rangle \tag{3.5.1d}$$

$$= \sum_{(j,x)\in\Omega} \sqrt{\mathbb{P}_{\mathbf{b}}(j,x)}e^{i\frac{\pi}{4}+i\frac{\varepsilon' b_j}{2}}|j\rangle|x\rangle, \tag{3.5.1e}$$

and hence we can implement $U_{\mathbb{P}_{\mathbf{b}}}$ with one call to $\mathcal{F}_{\varepsilon'}$.[2]

Next, let the random variable $X : \Omega \to \mathbb{R}^d$ be defined as $X(j,x) = x\mathbf{e}_j$. Then,

$$\mu = \mathbb{E}[X] = \frac{1}{d}\sum_{j=1}^{d}\cos^2\left(\frac{\pi}{4}-\frac{\varepsilon' b_j}{2}\right)\mathbf{e}_j = \frac{1}{2d}\mathbf{1} + \frac{\sin(\varepsilon')}{2}\mathbf{b} = \frac{1}{2d}\mathbf{1} + \frac{8\varepsilon}{d}\mathbf{b},$$

Thus, if we find an approximation $\widetilde{\mu}$ to $\mu$ such that $\|\widetilde{\mu} - \mu\|_1 \leq \varepsilon$, then

$$\begin{aligned}\min_{\widetilde{\mathbf{b}}\in\{0,1\}^d}\left\|\frac{d}{8\varepsilon}\left(\widetilde{\mu}-\frac{1}{2d}\mathbf{1}\right)-\widetilde{\mathbf{b}}\right\|_1 &= \frac{d}{8\varepsilon}\min_{\widetilde{\mathbf{b}}\in\{0,1\}^d}\left\|\widetilde{\mu}-\frac{1}{2d}\mathbf{1}-8\varepsilon\widetilde{\mathbf{b}}\right\|_1 \\ &\leq \frac{d}{8\varepsilon}\|\widetilde{\mu}-\mu\|_1,\end{aligned}$$

[2]Note that we don't have to worry about the extra global phase here – we can absorb it in the definition of the state $|\omega\rangle$, i.e., if $\omega = (j,x)$ we can define $|\omega\rangle = e^{i\pi/4+i\varepsilon' b_j/2}|j\rangle|x\rangle$, and then simply use the resulting probability distribution oracle $U_{\mathbb{P}}$.

and hence if we let $\widetilde{\mathbf{b}}$ be the bit string for which the minimum in the above expression is attained, then we find that

$$\begin{aligned}\left\|\widetilde{\mathbf{b}} - \mathbf{b}\right\|_1 &\leq \left\|\frac{d}{8\varepsilon}\left(\widetilde{\mu} - \frac{1}{2d}\mathbf{1}\right) - \widetilde{\mathbf{b}}\right\|_1 + \left\|\frac{d}{8\varepsilon}\left(\widetilde{\mu} - \frac{1}{2d}\mathbf{1}\right) - \mathbf{b}\right\|_1 \\ &\leq \frac{d}{8\varepsilon}\left\|\widetilde{\mu} - \mu\right\|_1 + \frac{d}{8\varepsilon}\left\|\widetilde{\mu} - \mu\right\|_1 = \frac{d}{4\varepsilon}\left\|\widetilde{\mu} - \mu\right\|_1 \leq \frac{d}{4}.\end{aligned}$$

We know that constructing such a bit string $\widetilde{\mathbf{b}}$ requires $\Omega(d/\varepsilon)$ queries to $F_{\varepsilon'}$, and hence we find that in order to find an $\varepsilon$-precise $\ell_1$-approximation of the mean of a random variable, we need to make at least $\Omega(d/\varepsilon)$ calls to $U_{\mathbb{P}}$ as well. This completes the proof. □

We now add a final norm conversion to the lemma we proved above, to obtain a lower bound for estimating the mean of a random variable up to $\ell_2$-norm. This yields the main theorem in this section.

**3.5.3.** Theorem. *Let $d \in \mathbb{N}$, $0 < \varepsilon \leq 1/16$, and suppose that we have a quantum algorithm $\mathcal{A}$ that finds an approximation $\widetilde{\mu}$ to the mean $\mu$ of any $d$-dimensional random variable with values contained in the unit ball in $\ell_2$ norm, using $n$ queries to $U_{\mathbb{P}}$, such that $\left\|\widetilde{\mu} - \mu\right\|_2 \leq \varepsilon$, with probability at least $2/3$. Then,*

$$n = \Omega\left(\min\left\{\frac{1}{\varepsilon^2}, \frac{\sqrt{d}}{\varepsilon}\right\}\right).$$

**Proof:**
First, suppose that $\varepsilon < 1/(16\sqrt{d})$. Then, we know from Hölder's inequality that if we can find a $\widetilde{\mu} \in \mathbb{R}^d$ such that $\left\|\widetilde{\mu} - \mu\right\|_2 \leq \varepsilon$, then we additionally have, with $\varepsilon' = \sqrt{d}\varepsilon$,

$$\left\|\widetilde{\mu} - \mu\right\|_1 \leq \sqrt{d}\left\|\widetilde{\mu} - \mu\right\|_2 \leq \sqrt{d}\varepsilon = \varepsilon'.$$

Since $\varepsilon' \leq 1/16$, we can now use Lemma 3.5.2 to obtain

$$n = \Omega\left(\frac{\sqrt{d}}{\varepsilon}\right).$$

On the other hand, suppose that $1/(16\sqrt{d}) < \varepsilon \leq 1/16$. Then, let $d' = \lfloor 1/(16\varepsilon) \rfloor$. Now, we have $d' \geq 1$, and we can use $\mathcal{A}$ to solve the mean estimation problem on $d'$ dimensions, by embedding any $d'$-dimensional random variable into $d$ dimensions by simply padding with zeros. We know from Lemma 3.5.2 that this requires a number of queries that scales at least as

$$n = \Omega\left(\frac{d'}{\varepsilon}\right) = \Omega\left(\frac{1}{\varepsilon^2}\right).$$

This completes the proof. □

Thus, we have proved the optimality of the bounded mean estimation algorithm, Algorithm 3.3.6. Since the bounded mean estimation problem is an easier problem than the general mean estimation problem, this lower bound naturally extends to an optimality proof for the general mean estimation problem. Thus, in particular, we find that Algorithms 3.3.6, 3.4.4 and 3.4.8 are all optimal up to polylogarithmic factors.

## 3.6 Discussion

In this chapter, we have developed a quantum mean estimation algorithm, and showed that it is essentially optimal. We sketch the graphs showing the precision that one can obtain with $n$ samples in the classical and quantum case in Figure 3.6.1. We can see that there is only a difference between the classical and quantum performance in the high-precision regime, i.e., where $n = \Omega(d)$. Thus, we observe that we can only obtain a quantum advantage for mean estimation if the number of samples we are allowed to use, $n$, exceeds the dimension of the random variable whose mean we try to estimate.

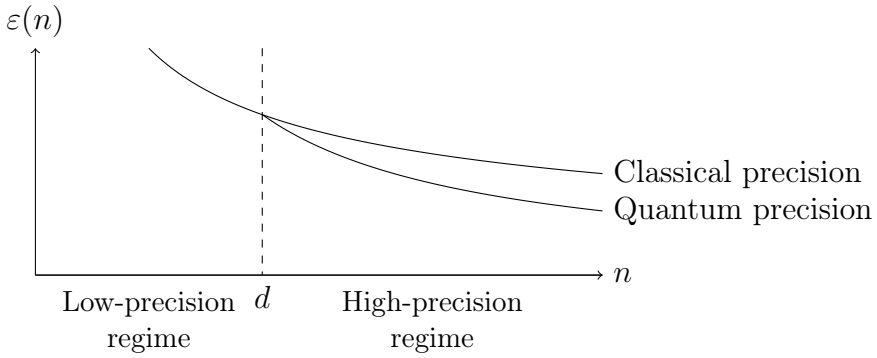



Figure 3.6.1: A graph of the precision $\varepsilon$ that can be obtained using $n$ samples, with a classical algorithm, as well as with a quantum one. The two graphs differ only in the high-precision regime, i.e., where $n = \Omega(d)$.

There are quite some natural follow-up questions to consider. First, the construction we present is only optimal up to polylogarithmic factors. It would be nice to figure out how many of the polylogarithmic factors that show up in the expressions for the number of queries can be removed. Very recently, Kothari and O'Donnell showed that in the univariate case, all of them can be removed if we set the failure probability $\delta$ to a constant, say 1/3 [KO22]. It would be nice to see if their approach can be generalized to the multivariate setting as well.

Another interesting question is whether the approach outlined in this section can be shown to be optimal in other $\ell_p$-norms as well. The most general question we can pose in this regard is the following. Let $p, q \in [1, \infty]$ and let $X : \Omega \to R \subseteq \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_q \leq 1\}$, i.e., let $X$ have values bounded by the unit ball w.r.t. $\ell_q$-norm, and suppose that we want to estimate $\mu = \mathbb{E}[X]$ up to precision $\varepsilon$ in $\ell_p$-norm. What is the query complexity of this problem? In this chapter we answer this question in the case where $p = q = 2$. Some subsequent insights indicate that it is likely possible to answer this question for all values of $p$ and $q$, and would most probably only require somewhat more intricate norm conversion results than those used in this chapter.

Finally, it is not clear whether the parameter $\mathrm{Tr}[\Sigma]$ is always the most convenient parameter to characterize the complexity of the mean estimation problem, especially when approximating the mean in norms other than the $\ell_2$-norm. For instance, in the $\ell_\infty$-norm, there exists a classical algorithm that attains precision $\varepsilon$ with $\widetilde{O}(\max_{j\in[d]} \sqrt{\mathrm{Var}(X_j)}/\varepsilon^2)$ samples. It would be interesting to figure out whether quantumly one can also attain an algorithm that similarly depends on the maximum of the variances, whilst improving over the dependence in $1/\varepsilon$ quadratically. We leave this for future work.

# Chapter 4

# State tomography

In the previous chapter, we took a look at the *classical* problem of mean estimation, and developed a quantum algorithm that solves it more efficiently than the optimal classical algorithm. As such, the input model that we used in the quantum algorithm represented the corresponding access model in the classical case as best as possible.

In contrast, one can also consider estimation problems that are inherently quantum, i.e., that don't necessarily bear a classical counterpart. In this chapter, we take a look at one particularly useful problem of this form, known as quantum state tomography, where our task is to output an approximate classical description of some quantum state that we can only access through a black box that prepares it.

We end up tightly characterizing the required number of calls to such a black box up to polylogarithmic factors. Without proper introduction of the relevant terminology, we briefly mention the takeaway message here: for a mixed state $\rho$ of rank $r$ and dimension $d$, it costs $\widetilde{\Theta}(dr/\varepsilon)$ (controlled, inverse) calls to the unitary that prepares its purification, to obtain an $\varepsilon$-approximate description of $\rho$ in trace norm. We additionally show a series of consequences of this tight characterization, in Section 4.8.

This chapter is based on [vACG+22] and contains some subsequent improvements obtained recently. We start with some introductory remarks in Section 4.1, and explain the model we are working with in more detail in Section 4.2. Then, we develop three separate techniques, all of which are of independent interest:

1. In Section 4.3 we introduce a norm conversion lemma that beats Hölder's inequality in the low-precision regime.
2. In Section 4.4 we introduce an unbiased version of the phase estimation algorithm, i.e., Algorithm 2.4.3.
3. In Section 4.5 we show how we can use it to estimate many non-commuting observables simultaneously, improving over a result from [HWM+21].

Finally, we show in Section 4.6 how all these ideas can be used in conjunction to obtain an essentially optimal mixed-state tomography algorithm. The corresponding lower bound is proved in Section 4.7. Finally, we give some immediate consequences of our work in Section 4.8.

## 4.1 Introduction

Quantum state tomography is the process of obtaining a classical description of a quantum state. Tomography is a fundamental tool in quantum information science, where it finds numerous applications. In the context of quantum algorithms, pure quantum state tomography can be used to retrieve a classical description of the final state of the algorithm, e.g., the solution of a linear system [HHL09] or the evolution of a quantum system [Llo96].

The more general mixed quantum state tomography finds applications in quantum information theory, and in the simulation of quantum many-body and thermodynamic systems. In some settings we are not interested in the full state, but only in its expectation value under a certain set of (possibly overlapping) measurements. This was first introduced by Aaronson [Aar20] under the name shadow tomography, and has since received a lot of attention in the literature, e.g., [HKP20; ASS21; HLY+22].

Most of the existing work on this topic has focused on the sample complexity of these problems: how many copies of the state are needed to perform tomography? In this chapter we consider the problem under a different input model: we assume access to a unitary (and its inverse) that prepares the state. This model is very natural when the state is the output of a quantum algorithm, but it has received little attention so far. The main improvements in this model come from the ability to use techniques related to amplitude estimation to reduce the dependence on the error parameter, but attaining such quadratic improvements requires the development of several new tools, and the analysis does not follow from a simple application of amplitude estimation.

Throughout this chapter we consider either a $d$-dimensional pure state $|\psi\rangle = \sum_{j=0}^{d-1} \alpha_j |j\rangle$ or a rank-$r$ mixed state $\rho \in \mathbb{C}^{d\times d}$. We are interested in learning the state up to error $\varepsilon$ in some $\ell_q$-norm or Schatten $q$-norm, often with some probability of failure $\leq \delta$. In the introduction we often use $\widetilde{\mathcal{O}}(\cdots)$ notation to hide polylogarithmic factors in the parameters $d$, $r$, $1/\varepsilon$, and $1/\delta$, even if these parameters do not appear polynomially in the $\widetilde{\mathcal{O}}(\cdots)$. For more precise complexity statements we refer to the relevant theorems in the main text.

**Related work.** Classical algorithms that estimate probabilities generally depend quadratically on $1/\varepsilon$, as that many samples are required to bring down the variance. In certain settings quantum algorithms can improve on this classical

complexity. Brassard et al. [BHM+02] introduced the amplitude estimation algorithm, and showed that it can estimate an amplitude (or probability) with a $1/\varepsilon$ dependence, if a state-preparation unitary and its inverse are available.

Van Apeldoorn [vApe21] generalized this for finding an $\ell_\infty$-norm estimate of a discrete probability distribution. In the model of van Apeldoorn, access to the distribution is given by a state-preparation oracle (and its inverse), such that the probability distribution corresponds to computational-basis measurements of the prepared state. Van Apeldoorn [vApe21] showed that $\widetilde{\mathcal{O}}(1/\varepsilon)$ applications of the input unitary are sufficient to compute the desired $\ell_\infty$-norm estimate. In the same paper the question was posed whether one can also speed up the estimation of multiple expectation values over the same distribution. A lower bound of $\Omega(\min\{\sqrt{m}/\varepsilon, 1/\varepsilon^2\})$ was given when $m$ expectation values need to be estimated op to precision $\varepsilon$. It was later shown by Huggins et al. [HWM+21] that $\widetilde{\mathcal{O}}(\sqrt{m}/\varepsilon)$ queries are sufficient even when estimating expectation values of observables on a pure quantum state.

Besides these few results for pure quantum state tomography, the most frequently studied setting is that of mixed-state tomography. In this setting we want to determine how many copies of $\rho$ are necessary to obtain a classical description with a given maximum error $\varepsilon$ in trace norm; it is often assumed that some upper bound $r$ on the rank of the state is known (if the state is pure, $r = 1$). An algorithm of Gross et al. [GLF+10], that applies measurements on one copy of the state at once, achieves $\mathcal{O}(dr^2/\varepsilon^2)$ sample complexity. Haah et al. [HHJ+17] show that the bound is optimal when the measurements are on a single copy at a time, and Chen et al. [CHL+22] complete our understanding of this setting by showing that the bound cannot be improved even with adaptive measurement schemes, as long as we require single-copy measurements. A better sample complexity can be achieved if we allow joint measurements on multiple copies of the state: with this more powerful access model, the best algorithm for tomography is also due to Haah et al. [HHJ+17] and O'Donnell and Wright [OW16], and it requires $\widetilde{\mathcal{O}}(dr/\varepsilon^2)$ copies of the quantum state; see also [OW16]. Haah et al. also show matching lower bounds up to polylogarithmic factors (these polylogarithmic factors are eliminated by Yuen [Yue22]), therefore their algorithm is essentially optimal. The main drawback of their approach is that it does not only require joint measurements on many states at once, but it also has time complexity exponential in $d$.

## 4.2 Preliminaries

In this section, we define the objects that we will be using in the remainder of this chapter. First, we formally define density matrices.

**4.2.1.** Definition (Density matrix). Let $\mathcal{H}$ be a Hilbert space of dimension $d \in \mathbb{N}$. A *density matrix* on a Hilbert space is a positive semidefinite operator $\rho$ acting

on $\mathcal{H}$, i.e., $0 \preceq \rho \in \mathcal{L}(\mathcal{H})$, such that $\mathrm{Tr}[\rho] = 1$. Its *rank* $1 \leq r \leq d$ is the number of non-zero eigenvalues, counting multiplicity. If $r = 1$, then we say that $\rho$ is a *pure state*, otherwise we say that $\rho$ is a *mixed state*. ◀

We can embed a density matrix into a quantum state through the concept of a purification. This is the topic of our next definition.

**4.2.2.** Definition (Purification). Let $|\psi\rangle \in \mathcal{H} \otimes \mathcal{W}$ be a quantum state. We define the linear operator $\mathrm{Tr}_{\mathcal{W}} : \mathcal{L}(\mathcal{H} \otimes \mathcal{W}) \to \mathcal{L}(\mathcal{H})$ as the unique map that for all $\rho \in \mathcal{L}(\mathcal{H})$ and $\sigma \in \mathcal{L}(\mathcal{W})$ acts as

$$\mathrm{Tr}(\rho \otimes \sigma) = \mathrm{Tr}[\sigma]\rho.$$

Next, let $\rho \in \mathcal{L}(\mathcal{H})$ be a density matrix, and let $|\psi\rangle \in \mathcal{H} \otimes \mathcal{W}$ be a quantum state. If $\mathrm{Tr}_{\mathcal{W}}(|\psi\rangle\langle\psi|) = \rho$, then we say that $|\psi\rangle$ is a *purification of* $\rho$. ◀

Note that every density matrix $\rho \in \mathcal{L}(\mathcal{H})$ always admits a purification, since we can write the eigendecomposition of the rank-$r$ density matrix $\rho$ as

$$\rho = \sum_{j=1}^{r} \lambda_j |\psi_j\rangle\langle\psi_j| ,$$

which implies that the state $|\psi\rangle \in \mathcal{H} \otimes \mathbb{C}^{[r]}$

$$\sum_{j=1}^{r} \sqrt{\lambda_j} |\psi_j\rangle \otimes |j\rangle$$

is a purification of $\rho$.

In this chapter, we consider the problem of estimating $\rho \in \mathcal{L}(\mathcal{H})$, given access to a unitary that prepares its purification.

**4.2.3.** Definition (State-preparation unitary). Let $\rho \in \mathcal{L}(\mathcal{H})$ be a density matrix, and let $U$ be a unitary that maps

$$U : |0\rangle \mapsto |\psi\rangle ,$$

where $|\psi\rangle$ is a purification of $\rho$. Then, $U$ is a *state-preparation unitary of* $\rho$. ◀

When we say that we have access to a state-preparation unitary, we mean that we can apply this operation $U$, and its inverse $U^\dagger$. Moreover, we assume that we can also apply these operations $U$ controlled on an auxiliary qubit being in the state $|1\rangle$. Similarly, when we count the number of queries to $U$, we mean the total number of calls to any of these routines.

In order to measure the distance of our approximation $\widetilde{\rho}$ to $\rho$, we specify a family of distance measures on $\mathcal{L}(\mathcal{H})$.

**4.2.4.** Definition (Schatten norms). Let $A \in \mathcal{L}(\mathcal{H})$ be a linear operator on $\mathcal{H}$. Let $\lambda_1, \ldots, \lambda_r$ be its singular values. For every $p \in [1, \infty]$, we let $\|A\|_p$ denote the $\ell_p$-norm of this vector of singular values, i.e.,

$$\|A\|_p = \left( \sum_{j=1}^{r} |\lambda|^p \right)^{\frac{1}{p}},$$

and we refer to $\|A\|_p$ as the *Schatten p-norm of A*. Specifically,

1. We refer to $\|A\|_1$ as the *trace norm*.
2. We refer to $\|A\|_2$ as the *Frobenius norm*.
3. We refer to $\|A\|_\infty$ as the *operator norm*. ◀

## 4.3 Bounded norm conversion

If we have an estimate of a vector with error at most $\varepsilon$ in, for example, the $\ell_\infty$-norm, then we can use Hölder's inequality to show that this is also an estimate with error at most $\varepsilon d^{1/q}$ in the $\ell_q$-norm, where $q \in [1, \infty]$. However, this bound is poor for large $d$. Here, we show that we can do better if we know that the vector we are estimating is normalized in some $\ell_s$-norm, for $s \in [1, \infty]$, using the fact that such a vector cannot have too many large entries.

We start by proving a "norm sandwiching" lemma, i.e., a lemma that tells us how we can bound the $\ell_q$-norm of a vector, when we know bounds on its $\ell_\infty$-norm and $\ell_s$-norm, with $1 \leq s < q < \infty$.

**4.3.1.** Lemma (Norm sandwiching). *Let $\varepsilon > 0$, $s \in [1, \infty]$, $\mathbf{v} \in \mathbb{R}^d$, and suppose that $\|\mathbf{v}\|_s \leq 1$, and $\|\mathbf{v}\|_\infty \leq \varepsilon$. Then, for all $q \in (s, \infty)$, we have*

$$\|\mathbf{v}\|_q \leq \min\left\{ \varepsilon \cdot d^{\frac{1}{q}}, \varepsilon^{1-\frac{s}{q}} \right\}.$$

**Proof:**
The first part of the minimum follows directly from Hölder's inequality. On the other hand, we have

$$\|\mathbf{v}\|_q^q = \sum_{j=1}^{d} |v_j|^q \leq \sum_{j=1}^{d} |v_j|^s \cdot |v_j|^{q-s} \leq \varepsilon^{q-s}.$$

The claim follows by taking the power of $1/q$ on both sides. This completes the proof. □

A visualization of the above lemma in the special case where $s = 1$ and $q = 2$ is provided in Figure 4.3.1. We can see that there are indeed inherently two

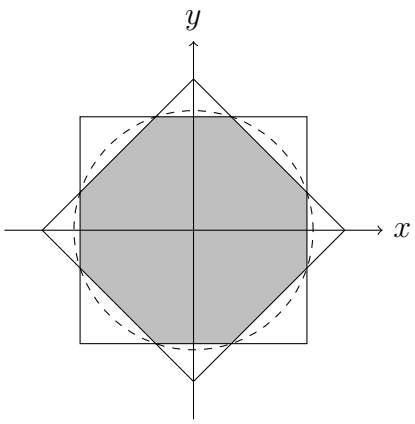



Figure 4.3.1: Visualization of the norm conversion lemma, in the case where $s = 1$ and $q = 2$, in two dimensions with $\varepsilon = 3/4$. The vectors $\mathbf{v}$ are constrained to being in the gray area. Its maximal $\ell_2$-norm is the radius of the smallest circle we can draw around this set, which is indicated by the dashed line.

regimes: either $\varepsilon$ is big enough that some part of the $\ell_\infty$-cube with radius $\varepsilon$ sticks out of the $\ell_s$-ball, in which case the best upper bound on the $\ell_q$-norm is $\varepsilon^{1-s/q}$, or the $\ell_\infty$-cube with radius $\varepsilon$ is completely contained in the unit $\ell_s$-ball, in which case one cannot do any better than Hölder's inequality.

The above lemma can be used in cases where we are given an $\ell_\infty$-estimate of a vector which we a priori know to be normalized in some $\ell_s$-norm. This is the objective of the following lemma.

**4.3.2.** Lemma (Norm conversion lemma). *Let $s \in [1, \infty]$, and let $\mathbf{v} \in \mathbb{R}^d$ be such that $\|\mathbf{v}\|_s = 1$. Suppose furthermore that we have a vector $\widetilde{\mathbf{v}} \in \mathbb{R}^d$ such that $\|\widetilde{\mathbf{v}} - \mathbf{v}\|_\infty \leq \varepsilon$. Let $\overline{\mathbf{v}} \in \mathbb{R}^d$ be such that $\|\overline{\mathbf{v}}\|_s = 1$, and the difference $\|\overline{\mathbf{v}} - \widetilde{\mathbf{v}}\|_q$ is minimized. Then, for all $q \in (s, \infty)$, we have*

$$\|\overline{\mathbf{v}} - \mathbf{v}\|_q \leq 4 \min\left\{\varepsilon \cdot d^{\frac{1}{q}}, \varepsilon^{1-\frac{s}{q}}\right\}.$$

**Proof:**
First, observe that $\overline{\mathbf{v}} \in \mathbb{R}^d$ is the vector that is closest to $\widetilde{\mathbf{v}}$ w.r.t. $\ell_q$-norm, while being normalized in $\ell_s$-norm. In particular, this implies that $\|\overline{\mathbf{v}} - \widetilde{\mathbf{v}}\|_q \leq \|\mathbf{v} - \widetilde{\mathbf{v}}\|_q$, and thus by the triangle inequality we obtain that

$$\|\mathbf{v} - \overline{\mathbf{v}}\|_q \leq \|\overline{\mathbf{v}} - \widetilde{\mathbf{v}}\|_q + \|\mathbf{v} - \widetilde{\mathbf{v}}\|_q \leq 2\|\mathbf{v} - \widetilde{\mathbf{v}}\|_q.$$

We also know by the triangle inequality that

$$\|\mathbf{v} - \widetilde{\mathbf{v}}\|_s \leq \|\mathbf{v}\|_s + \|\widetilde{\mathbf{v}}\|_s = 2,$$

and so we can apply Lemma 4.3.1 to the vector $(\mathbf{v} - \widetilde{\mathbf{v}})/2$, and with $\varepsilon' = \varepsilon/2$. Thus, we obtain

$$\|\mathbf{v} - \overline{\mathbf{v}}\|_q \leq 2\,\|\mathbf{v} - \widetilde{\mathbf{v}}\|_q \leq 4\min\{\varepsilon' \cdot d^{\frac{1}{q}}, (\varepsilon')^{1-\frac{s}{q}}\} \leq 4\min\{\varepsilon \cdot d^{\frac{1}{q}}, \varepsilon^{1-\frac{s}{q}}\}.$$

This completes the proof. □

We remark here that this result similarly applies to density matrices, since the Schatten $p$ norms are simply $\ell_p$-norms on the eigenvalue vectors. As such, we can simply apply the above lemma to those vectors of eigenvalues to obtain the same statement in terms of Schatten norms.

In high-level terms, the lemma presented here characterizes the optimal norm error we can achieve, if we have a procedure that naturally produces $\ell_\infty$-estimates of the vector that we are interested in. It is tempting to ponder over the applicability of this norm conversion lemma to the mean estimation setting. However, the situation in the mean estimation setting is subtly different: there we have two possible algorithmic techniques to obtain our estimate, either we use a quantum technique that naturally yields an $\ell_\infty$-estimate, or we use a classical technique to naturally obtain an $\ell_2$-estimate. Thus, if one were to generalize the above lemma to also include the case where the approximation $\widetilde{\mathbf{v}} \in \mathbb{R}^d$ of $\mathbf{v} \in \mathbb{R}^d$ is assumed to satisfy $\|\mathbf{v} - \widetilde{\mathbf{v}}\|_2 \leq \varepsilon$, then one could probably close the open question from the previous chapter asking for the optimal sample complexity of bounded mean estimators w.r.t. other $\ell_q$-norms. Initial attempts seem to indicate that the resulting expressions become very messy, but this is definitely a very interesting direction of future research.

## 4.4 Unbiased phase estimation

In this section, we describe a method for phase estimation that is unbiased. More precisely, it is symmetric in the sense that for a phase $\phi$ it provides an estimate $\varphi$ such that the probability of getting estimate $\phi + \varepsilon$ is the same as getting estimate $\phi - \varepsilon$ (modulo $2\pi$) for all $\varepsilon$. Note that this is not satisfied by ordinary phase estimation, but this property is highly desirable, as we showcase in our applications. In particular, we need unbiased phase estimation to recover unbiased estimates of the entries of a density matrix, allowing us to give tighter error bounds with high probability.

We start this section by introducing the basic algorithm, in Section 4.4.1. Subsequently, in Section 4.4.2, we show how the tail bounds of this basic algorithm can be improved. Then, in Section 4.4.3, we show an application of this algorithm to unbiased probability estimation.

### 4.4.1 Basic algorithm

Our method is based on adding and later subtracting a random phase shift; this idea is not new, see, e.g., [LdW21, Section 3]. We present the algorithm and its analysis below.

---

**Algorithm 4.4.1:** Unbiased phase estimation

---

**Input:**

1: $k \in \mathbb{N}$: the number of bits of precision.

2: $\mathcal{U}$: a quantum circuit acting on $\mathcal{H}$ that implements a unitary $U$.

3: $C_{|\psi\rangle}$: a quantum circuit acting on $\mathcal{H}$ that implements $|0\rangle \mapsto |\psi\rangle$.

**Derived objects:**

1: $r = \dim(\mathcal{H})$.

2: Let the eigendecomposition of $U$ be

$$U = \sum_{j=1}^{r} e^{2\pi i \theta_j} |\psi_j\rangle \langle\psi_j| ,$$

where for all $j \in [r]$, $\theta_j \in (-1/2, 1/2]$.

3: Let $\Phi$ be a random variable taking values in $\{\theta_j : j \in [r]\}$, such that for all $j \in [r]$,

$$\mathbb{P}[\Phi = \theta_j] = |\langle\psi|\psi_j\rangle|^2 =: p_j.$$

4: $C_k = 1 - 2^{-k}$.

**Output:** A random variable $\overline{\phi} \in (-1/2, 1/2]$ such that

$$\mathbb{E}[e^{2\pi i \overline{\phi}}] = C_k \sum_{j=1}^{r} p_j e^{2\pi i \theta_j},$$

and for all $\ell \in [r]$ and integer $m \geq 2$,

$$\mathbb{P}[\text{cyclic-dist}(\theta_\ell, \overline{\phi}) \leq m \cdot 2^{-k}] \geq p_\ell \left(1 - \frac{1}{2(m-1)}\right).$$

**Queries:**

1: Number of calls to $\mathcal{U}$: $2^k - 1$.

2: Number of calls to $C_{|\psi\rangle}$: 1.

**Procedure:** UNBIASED-PHASE-EST($k$, $\mathcal{U}$, $C_{|\psi\rangle}$):

1: Choose $\phi \in (-1/2, 1/2]$ uniformly at random.

2: Let $\widetilde{\phi} =$ PHASE-EST($k$, $e^{2\pi i \phi}\mathcal{U}$, $C_{|\psi\rangle}$).

3: Let $\overline{\phi} = \widetilde{\phi} - \phi$, and add or subtract an integer such that $\overline{\phi} \in (-1/2, 1/2]$.

4: Output $\overline{\phi}$.

---

**Proof of the properties of Algorithm 4.4.1:**
The claims on the number of calls to $\mathcal{U}$ and $C_{|\psi\rangle}$ follow immediately from the properties of Algorithm 2.4.3. Moreover, the tail bound also follows directly. Thus, it remains to compute the expectation of $e^{2\pi i\overline{\phi}}$.

To that end, observe from the analysis of the phase estimation algorithm, i.e., Equation (2.4.3), that conditioned on the choice of $\phi$ in step 1, the measurement outcome $\widetilde{\phi}$ equals $\phi' \in \{-2^{k-1}+1, \ldots, 2^{k-1}\}$ with probability

$$\mathbb{P}\left[\widetilde{\phi} = \phi' \middle| \phi\right] = \sum_{\ell=1}^{r} p_\ell \cdot \frac{\sin^2(\pi \cdot 2^k(\theta_\ell + \phi - \phi'))}{2^{2k}\sin^2(\pi(\theta_\ell - \phi'))}.$$

Hence, the probability distribution for $\overline{\phi}$ becomes

$$\mathbb{P}\left[\overline{\phi} = \phi' \middle| \phi\right] = \sum_{\ell=1}^{r} p_\ell \cdot \frac{\sin^2(\pi \cdot 2^k(\theta_\ell - \phi'))}{2^{2k}\sin^2(\pi(\theta_\ell + \phi - \phi'))},$$

where the dependence on $\phi$ only comes back in the possible choices for $\phi'$, since the above relation only holds whenever $\phi' + \phi \in \{-2^{k-1}-1, \ldots, 2^{k-1}\}$. Thus, averaging over $\phi$ will only smoothen out the possible values that $\phi'$ can attain[1], and thus we obtain that the probability density function for $\overline{\phi}$ is

$$f_{\overline{\phi}}(\phi') = \sum_{\ell=1}^{r} p_\ell \cdot \frac{\sin^2(\pi \cdot 2^k(\theta_j - \phi'))}{2^k \sin^2(\theta_j - \phi')}.$$

Thus, we can directly compute the expectation of $\overline{\phi}$, as

$$\mathbb{E}\left[e^{2\pi i\overline{\phi}}\right] = \sum_{\ell=1}^{r} p_\ell \mathbb{E}\left[e^{2\pi i\phi_\ell}\right],$$

where $\phi_\ell$ is a random variable taking values in $(-1/2, 1/2]$ with probability density function

$$f_{\phi_\ell}(\phi') = \frac{\sin^2(\pi \cdot 2^k(\theta_j - \phi'))}{2^k \sin^2(\theta_j - \phi')}.$$

It remains to prove that for all $\ell \in [r]$, the expectation of $e^{2\pi i\phi_\ell}$ is equal to $C_k e^{2\pi i\theta_\ell}$. To that end, observe that

$$\begin{aligned}\mathbb{E}[e^{2\pi i(\phi_\ell - \theta_\ell)}] &= \int_{-1/2}^{1/2} e^{2\pi i(x-\theta_\ell)} \cdot \frac{\sin^2(\pi \cdot 2^k(x-\theta_\ell))}{2^k \sin^2(\pi(x-\theta_\ell))}\,\mathrm{d}x \\ &= \int_0^1 e^{2\pi i x} \cdot \frac{\sin^2(\pi \cdot 2^k x)}{2^k \sin^2(\pi x)}\,\mathrm{d}x,\end{aligned}$$

[1] A proper proof should not be very difficult, but it would require some measure theory. We will omit it here for readability.

where in the last equality we substituted $x$ for $x+\theta_j$, and used that the integrand is periodic with period 1. Thus, it remains to prove that the final integral expression equates to $C_k$. To that end, observe that the integrand is periodic with period 1, and that the factor $\sin^2(\pi \cdot 2^k x)/(2^k \sin^2(\pi x))$ is symmetric around 0. Thus,

$$\begin{aligned}
&\int_0^1 e^{2\pi i x} \cdot \frac{\sin^2(\pi \cdot 2^k x)}{2^k \sin^2(\pi x)} \, \mathrm{d}x = \int_0^{\frac{1}{2}} \left[e^{2\pi i x} + e^{-2\pi i x}\right] \frac{\sin^2(\pi \cdot 2^k x)}{2^k \sin^2(\pi x)} \, \mathrm{d}x \\
&= \int_0^{\frac{1}{2}} 2\cos(2\pi x) \frac{\sin^2(\pi \cdot 2^k x)}{2^k \sin^2(\pi x)} \, \mathrm{d}x = \int_0^{\frac{1}{2}} \left(2 - 4\sin^2(\pi x)\right) \frac{\sin^2(\pi \cdot 2^k x)}{2^k \sin^2(\pi x)} \, \mathrm{d}x \\
&= 1 - \frac{4}{2^k} \int_0^{\frac{1}{2}} \sin^2(\pi \cdot 2^k x) \, \mathrm{d}x = 1 - \frac{4}{2^k} \int_0^{\frac{1}{2}} \left[\frac{1}{2} - \frac{1}{2}\cos(2\pi \cdot 2^k x)\right] \, \mathrm{d}x \\
&= 1 - \frac{4}{2^k} \left[\frac{x}{2}\right]_0^{\frac{1}{2}} = 1 - \frac{1}{2^k} = C_k.
\end{aligned}$$

This completes the proof. □

### 4.4.2 Improved tail bounds

Note that the tail bound in Algorithm 4.4.1 is only hyperbolic. Thus, we conclude that the random variable $\overline{\phi}$ might be quite heavy-tailed, which can be a problem for potential applications. Fortunately, we can obtain a much more narrow distribution by a median trick on the unit circle, which is what we introduce in the algorithm below.

**Algorithm 4.4.2:** Unbiased phase estimation with boosted precision

**Input:**

1: $k \in \mathbb{N}$: the number of bits of precision.
2: $k' \in \mathbb{N}$: the boosting parameter.
3: $\mathcal{U}$: a quantum circuit acting on $\mathcal{H}$ that implements a unitary $U$.
4: $C_{|\psi\rangle}$: a quantum circuit acting on $\mathcal{H}$ that implements $|0\rangle \mapsto |\psi\rangle$.

**Derived objects:**

1: Let the eigendecomposition of $U$ be

$$U = \sum_{j=1}^{r} e^{2\pi i \theta_j} \, |\psi_j\rangle \langle\psi_j| \, ,$$

where for all $j \in [r]$, $\theta_j \in (-1/2, 1/2]$.

2: Let $\Phi$ be a random variable taking values in $\{\theta_j : j \in [r]\}$, such that for all $j \in [r]$,

$$\mathbb{P}[\Phi = \theta_j] = |\langle\psi|\psi_j\rangle|^2 =: p_j.$$

**Output:** A random variable $\overline{\phi} \in (-1/2, 1/2]$ such that

$$\mathbb{E}[e^{2\pi i\overline{\phi}}] = C_{k,k'} \sum_{j=1}^{r} p_j e^{2\pi i\theta_j},$$

where $1 - 2^{-k} \leq C_{k,k'} \leq 1$, and for all $\ell \in [r]$ and $m \geq 6$,

$$\mathbb{P}[\text{cyclic-dist}(\theta_\ell, \overline{\phi}) \leq m \cdot 2^{-k}] \geq p_\ell \left(1 - (m/6)^{-k'/2}\right).$$

**Queries:**

1: Number of calls to $\mathcal{U}$: $k'(2^k - 1)$.
2: Number of calls to $C_{|\psi\rangle}$: 1.

**Procedure:** UNBIASED-BOOSTED-PHASE-EST($k$, $k'$, $\mathcal{U}$, $C_{|\psi\rangle}$):

1: Choose $\phi \in (-1/2, 1/2]$ uniformly at random.
2: For $j = 1, \ldots, k'$, let $\widetilde{\phi}_j = $ PHASE-EST($k$, $e^{2\pi i\phi}\mathcal{U}$, $C_{|\psi\rangle}$), where all of the runs are performed *without* re-initializing the state $|\psi\rangle$.
3: Find the smallest arc of the unit circle that contains at least a strict majority of the points $e^{2\pi i\widetilde{\phi}_1}, \ldots, e^{2\pi i\widetilde{\phi}_{k'}}$. Let $\widetilde{\phi} \in (-1/2, 1/2]$ be such that $e^{2\pi i\widetilde{\phi}}$ is the midpoint of this arc.
4: Let $\overline{\phi} = \widetilde{\phi} - \phi$, and add or subtract integers until $\overline{\phi} \in (-1/2, 1/2]$.
5: Output $\overline{\phi}$.

---

In step 2 of the algorithm, we stress that we re-use the output state of the previous run of phase estimation. This way we ensure that all our outcomes belong to the same $\theta_\ell$, and thus it makes sense to take the median on the unit circle. Coincidentally, this is similar to the multiple runs of phase estimation we will come back to in Definition 8.2.2.

**Proof of the properties of Algorithm 4.4.2:**
The number of calls to $\mathcal{U}$ and $C_{|\psi\rangle}$ follow immediately from the description of the algorithm. Thus, it remains to check the properties of the output.

To that end, observe that since we don't reinitialize $|\psi\rangle$, we can analyze this setting as if $|\psi\rangle$ is exactly an eigenvector of $U$, i.e., $U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$. It remains to prove that in this case, the expectation of $e^{2\pi i\overline{\phi}}$ equals $C_{k,k'}e^{2\pi i\theta}$, and that the cyclic distance between $\overline{\phi}$ and $\theta$ is at most $m \cdot 2^{-k}$ with probability at least $1 - \exp(-k'm)$.

We start with the latter claim. To that end, suppose that the cyclic distance between $\theta$ and $\overline{\phi}$ is bigger than $m \cdot 2^{-k}$. Then, less than half of the points $\widetilde{\phi}_j$ are located in the interval $(-m/2 \cdot 2^{-k}, m/2 \cdot 2^{-k})$ around $\theta$, because if a strict majority were in this interval, by the pigeonhole principle there would be one point both in this interval and in the interval centered around $m \cdot 2^{-k}$, but then that interval would have to be wider than $m/2 \cdot 2^{-k}$ in radius. Thus, we find that

$$\mathbb{P}[\text{cyclic-dist}(\theta, \overline{\phi}) > m \cdot 2^{-k}] \leq \binom{k'}{\lceil k'/2 \rceil} \mathbb{P}[\text{cyclic-dist}(\theta, \widetilde{\mu}_j) > \lfloor m/2 \rfloor \cdot 2^{-k}]^{k'/2}$$
$$\leq \frac{2^{k'/2}}{2^{k'/2}(m/2-2)^{k'/2}} \leq (m/6)^{-k'/2}.$$

Finally, the post-processing method in step 3 of the algorithm is easily seen to be symmetric, and so the expectation of $e^{2\pi i \overline{\phi}}$ is a constant times $e^{2\pi i \theta}$. Moreover, we observe that the concentration we achieve here is tighter than in Algorithm 4.4.1, and thus $C_{k,k'} \geq 1 - 2^{-k}$. This completes the proof. □

### 4.4.3 Unbiased probability estimation

One particularly neat application of the unbiased phase estimation routine that we developed in the previous subsections is to a problem called unbiased probability estimation. In this problem, we are given access to a unitary that implements the mapping

$$C : |0\rangle \mapsto \sqrt{p}\,|\psi_1\rangle\,|1\rangle + \sqrt{1-p}\,|\psi_0\rangle\,|0\rangle =: |\psi\rangle\,,$$

for some unknown value $p \in [0, 1]$, and we are asked to estimate it.

The standard algorithm that solves this is known as *amplitude estimation*, and it first appeared in [BHM+02]. It performs phase estimation on an operation that is known as the Grover operator, which is defined by

$$G = C(2\,|0\rangle\langle 0| - I)C^\dagger(I \otimes (2\,|1\rangle\langle 1| - I)).$$

If in the algorithm proposed by [BHM+02], we substitute normal phase estimation with the unbiased version we developed in this section, then it turns out we can exactly calculate the total bias of the estimator of $p$. Surprisingly, the expectation of the resulting algorithm's outcome is a linear function of $p$, and hence we can invert it and obtain an unbiased estimator for $p$.

We state the resulting algorithm and prove its properties below.

**Algorithm 4.4.3:** Unbiased probability estimation

**Input:**

1: $k \in \mathbb{N}$: the precision parameter.

2: $C$: a quantum circuit that implements the mapping

$$|0\rangle \mapsto \sqrt{p}\,|\psi_1\rangle\,|1\rangle + \sqrt{1-p}\,|\psi_0\rangle\,|0\rangle\,.$$

**Derived objects:**

1: $\theta = \arcsin(\sqrt{p})/\pi$.

2: $G = C(2\left|0\right\rangle\left\langle 0\right| - I)C^\dagger(I \otimes (2\left|1\right\rangle\left\langle 1\right| - I))$.

**Output:** A random variable $\widetilde{p}$ that satisfies $\mathbb{E}[\widetilde{p}] = p$, and if we let $\overline{p} = 1/2 - (1 - 2^{-k})(1/2 - \widetilde{p})$, then for all integer $m \geq 2$,

$$\mathbb{P}\left[|\overline{p} - p| > \frac{4\pi^2\sqrt{p(1-p)}m^2}{2^k} + \left(\frac{2\pi m^2}{2^k}\right)^2\right] \leq \frac{1}{2(m-1)}. \tag{4.4.1}$$

**Queries:** Number of queries to $C$: $2^{k+1} - 1$.

**Procedure:** UNBIASED-PROBABILITY-EST($k$, $C$):

1: Let $\phi =$ UNBIASED-PHASE-EST($k$, $G$, $C$).

2: Let $\overline{p} = \frac{1}{2} - \frac{\cos(2\pi\phi)}{2}$

3: Output $\widetilde{p} = \frac{1}{2} - \frac{\cos(2\pi\phi)}{2(1-2^{-k})}$.

---

**Proof of the properties of Algorithm 4.4.3:**
The claim about the number of queries follows directly from the properties of unbiased phase estimation, i.e., Algorithm 4.4.1.

It can be easily shown that $G$ has eigenvectors

$$G\left|\psi_\pm\right\rangle = e^{\mp 2\pi i\theta}\left|\psi_\pm\right\rangle, \qquad \text{where} \qquad \left|\psi_\pm\right\rangle = (\left|\psi_1\right\rangle\left|1\right\rangle \pm i\left|\psi_0\right\rangle\left|0\right\rangle),$$

and where $\theta = \arcsin(\sqrt{p})$. Since we can write

$$\frac{i}{\sqrt{2}}(-e^{-\pi i\theta}\left|\psi_+\right\rangle + e^{\pi i\theta}\left|\psi_-\right\rangle) = \left|\psi\right\rangle,$$

we obtain in Algorithm 4.4.1 that

$$\mathbb{P}[\Phi = \pm\theta] = |\left\langle\psi_\pm|\psi\right\rangle|^2 = \frac{1}{2}.$$

Thus, we find that

$$\begin{aligned}\mathbb{E}[e^{2\pi i\phi}] &= C_k \cdot \frac{e^{2\pi i\theta} + e^{-2\pi i\theta}}{2} = C_k\cos(2\pi\theta) = C_k\cos(2\arcsin(\sqrt{p})) \\ &= C_k(1-2p).\end{aligned}$$

Therefore, by taking the real part on both sides, we obtain that

$$\mathbb{E}[\widetilde{p}] = \frac{1}{2} - \frac{\mathbb{E}[\cos(2\pi\phi)]}{2(1-2^{-k})} = \frac{1}{2} - \frac{C_k(1-2p)}{2C_k} = p.$$

Thus, it remains to check the tail bound from Equation (4.4.1). To that end, observe that the distributions arising from the phases $2\pi\theta$ and $-2\pi\theta$ are mirrored in phase 0, and so we can take the absolute value of $\phi$ without changing the shape

of its distribution. Thus, from the properties of Algorithm 4.4.1, we find that for all $m \geq 2$,

$$\mathbb{P}\left[\text{cyclic-dist}(|\phi|, \theta) > m \cdot 2^{-k}\right] \leq \frac{1}{2(m-1)}.$$

Next, suppose that $\text{cyclic-dist}(\phi, \theta) \leq t$, for some $t > 0$. Then,

$$\begin{aligned}
|\overline{p} - p| &\leq \frac{|\cos(2\pi\phi) - \cos(2\pi\theta)|}{2} \leq 2|\sin(\pi(\phi-\theta))\sin(\pi(\phi+\theta))| \\
&= 2|\sin(\pi(\phi-\theta))| \cdot |\sin(2\pi\theta - \pi(\theta-\phi))| \\
&= 2|\sin(\pi(\phi-\theta))| \cdot |\cos(\pi(\theta-\phi))\sin(2\pi\theta) - \cos(2\pi\theta)\sin(\pi(\theta-\phi))| \\
&\leq 4\sin|\pi(\phi-\theta)|\sqrt{p(1-p)} + 2\sin|\pi(\theta-\phi)|^2 \\
&\leq 4\pi\sqrt{p(1-p)}t + 2\pi^2 t^2,
\end{aligned}$$

where we used that $\text{cyclic-dist}(\phi, \theta) \leq t$ implies that $\sin|\pi(\theta-\phi)| \leq \pi t$. The above bound on $|\overline{p} - p|$ also holds if we exchange $\phi$ for $-\phi$, and thus we obtain that $\text{cyclic-dist}(|\phi|, \theta) \leq t$ implies that $|\overline{p} - p| \leq 4\pi\sqrt{p(1-p)}t + 2\pi^2 t^2$. Hence, for all $m \geq 2$, we have

$$\mathbb{P}\left[|\overline{p} - p| > \frac{4\pi\sqrt{p(1-p)}m}{2^k} + \left(\frac{2\pi m}{2^k}\right)^2\right] \leq \frac{1}{2(m-1)}.$$

This completes the proof. □

We observe that the boosted routine achieves the same precision as in the seminal work [BHM+02], up to a multiplicative constant. Thus, our technique is essentially a black-box improvement over the classic amplitude estimation algorithm, since it provides an estimate with the same accuracy, and with the extra property that it is unbiased. A very interesting direction for future research is to see what potential implications this has for algorithms that use amplitude estimation as a subroutine.

One additional remark to make here is that even though the expectation of $\widetilde{p}$ is $p \in [0, 1]$, that does not mean that every realization of $\widetilde{p}$ will be contained in the interval $[0, 1]$. Hence, any algorithm that would like to use our routine as a black box, will have to deal with these exceptions.

One way to circumvent this out-of-bounds behavior is by outputting $\overline{p}$ instead, i.e., to disregard the normalization factor $1/C_k$. This would ensure that $\widetilde{p}$ is always contained in the interval $[0, 1]$, at the expense of having a small bias on the order of $\mathcal{O}(2^{-k})$.

We also briefly remark here that one can also use the boosted version of the unbiased phase estimation routine to obtain an unbiased probability with an improved tail bound. Using the exact same proof techniques as displayed above, we observe that if we substitute the boosted unbiased phase estimation routine

with parameters $k$, $k'$, i.e., Algorithm 4.4.2, in step 1 of Algorithm 4.4.3, then for any $m \geq 6$,

$$\mathbb{P}\left[|\overline{p} - p| > \frac{4\pi\sqrt{p(1-p)}m}{2^k} + \left(\frac{2\pi m}{2^k}\right)^2\right] \leq \left(\frac{6}{m}\right)^{k'/2}. \tag{4.4.2}$$

In particular, the boosting parameter, $k'$, allows for regulating the polynomial dependence on the tail bound – if we take $k' = 2$, we get a linear dependence on $1/m$, whereas if we ramp up $k'$ a bit higher, the exponent of the decay is increased accordingly. In particular, if we choose $k' > 4$, then the decay is bigger than inverse quadratic in $m$, which is favorable if we want to compute the *variance* of $\widetilde{p}$, a property that we haven't touched on yet in this chapter. Setting $k'$ even higher will also ensure that higher-order central moments of $\widetilde{p}$ are small. We develop these ideas further in Section 5.2.

## 4.5 Estimating multiple observables with a state-preparation oracle

To perform efficient mixed-state tomography we rely on an algorithm that estimates multiple observables simultaneously. Specifically, suppose we have access to some density matrix $\rho \in \mathcal{L}(H)$ through a unitary that prepares its purification, as defined in Definitions 4.7.1 and 4.2.3. Next, we define $m$ Hermitian operators $E_1, \ldots, E_m \in \mathcal{L}(H)$, and we define the vector $\mathbf{z} \in \mathbb{R}^m$ such that for all $j \in [m]$, $z_j = \mathrm{Tr}[\rho E_j]$. The question we address in this section is how many (inverse, controlled) calls to the state-preparation unitary are required to obtain an $\varepsilon$-precise estimate of $\mathbf{z}$ in $\ell_\infty$-norm.

We assume that the observables $E_1, \ldots, E_m$ are known a priori. For simplicity, we don't count the cost of implementing these in a quantum circuit. See [vACG+22] for a more detailed account on the number of gates in the algorithm designed in this section.

The algorithm is based on constructing the phase oracle for a function whose gradient is the vector of the desired expectation values, similarly to the approach presented in Section 3.3. As was the case before, to ensure that the function is properly normalized we need to bound the weighted combination of expectation values, where the weights are taken from the dual grid, as defined in Definition 3.3.1. In this setting, we require some results on random matrices, which we use by translating properties that hold for uniformly random matrices into properties that hold for all but a constant fraction of the points in the dual grid.

This task was recently studied in [HWM+21]. They constructed an algorithm that solves this problem using $\mathcal{O}(\sqrt{\sum_{j=1}^m \|E_j\|^2}/\varepsilon)$ applications of the state-preparation unitary and its inverse. This is $\mathcal{O}(\sqrt{m}/\varepsilon)$ under the assumption $\|E_j\| \leq 1$. We present a subtle but crucial improvement over [HWM+21], since

we give an algorithm with a sample complexity of $\widetilde{\mathcal{O}}(\sqrt{\|\sum_{j=1}^m E_j^2\|}/\varepsilon)$, i.e., we move the summation and square inside the norm. Ultimately, as we will see in the next section, this leads to a saving of a factor $d$ when applied to mixed-state tomography. For a discussion of other existing approaches to solve the problem of computing expectation values, we refer to the excellent introduction in [HWM+21].

### 4.5.1 Tail bounds on uniform matrix series

As mentioned above, we first need to prove some properties of uniform random matrices. We do this by adapting a result on Gaussian / Rademacher random matrices given below. Here and in the remainder of this section, for a random matrix $Y$ we define $v(Y) := \|\mathbb{E}[Y^2] - (\mathbb{E}[Y])^2\|$ as its variance.

**4.5.1.** Theorem ([Tro15, Theorem 4.6.1]). *Let $E_1, \ldots, E_m$ be $d \times d$ Hermitian matrices. Let $\lambda_1, \ldots, \lambda_m$ be drawn from iid standard normal distributions and let $Y = \sum_{\ell=1}^m \lambda_\ell E_\ell$. Then $\mathbb{E}[Y] = 0, v(Y) = \|\sum_{\ell=1}^m E_\ell^2\|$ and*

$$\mathbb{P}[\|Y\| \geq t] \leq 2de^{-\frac{t^2}{2v(Y)}}.$$

*The same bounds hold when $\{\lambda_j\}$ is iid uniformly random over $\{-1, 1\}$.*

In order to adapt the above result to our setting, we invoke a technical statement from [Tro15].

**4.5.2.** Proposition ([Tro15, Theorem 3.6.1]).
*Consider a finite sequence $(E_\ell)_{\ell=1}^m$ of independent, random, Hermitian matrices of the same size. Then for all $t \in \mathbb{R}$ we have*

$$\mathbb{P}\left[\lambda_{\max}\left(\sum_{\ell=1}^m E_\ell\right) \geq t\right] \leq \inf_{\theta>0} e^{-\theta t} \operatorname{Tr}\left[\exp\left(\sum_{\ell=1}^m \log \mathbb{E}\left[e^{\theta E_\ell}\right]\right)\right].$$

With the help of this result we prove the following variant of Theorem 4.5.1 for bounded random variables.

**4.5.3.** Theorem (Bounded Matrix series inequality). *Let $E_1, \ldots, E_m$ be $d \times d$ Hermitian matrices. Let $\lambda_1, \ldots, \lambda_m$ be independent symmetrically distributed random variables supported on $[-1, 1]$ and let $Y = \sum_{\ell=1}^m \lambda_\ell E_\ell$. Then $\mathbb{E}[Y] = 0, v(Y) \leq \|\sum_{\ell=1}^m E_\ell^2\|$ and*

$$\mathbb{P}[\|Y\| \geq t] \leq 2de^{-\frac{t^2}{2v(Y)}}.$$

**Proof:**
We follow the proof of [Tro15, Theorem 4.6.1] and modify it where necessary. First we note that for all $\ell \in [m]$,

$$\begin{aligned}\mathbb{E}[e^{\lambda_\ell E_\ell}] = \mathbb{E}\left[\sum_{k=0}^{\infty} \frac{\lambda_\ell^k}{k!} E_\ell^k\right] = \sum_{k=0}^{\infty} \frac{\mathbb{E}[\lambda_\ell^k]}{k!} E_\ell^k = \sum_{q=0}^{\infty} \frac{\mathbb{E}[\lambda_\ell^{2q}]}{(2q)!} E_\ell^{2q} \\ \preceq \sum_{q=0}^{\infty} \frac{1}{(2q)!} E_\ell^{2q} \preceq \sum_{q=0}^{\infty} \frac{1}{q!} (E_\ell^2/2)^q = e^{E_\ell^2/2}, \qquad (4.5.1)\end{aligned}$$

where we used that $\mathbb{E}[\lambda_\ell^k] = 0$ for all odd integers $k$, and in the last line we used that $(2q)! \geq 2^q q!$. This implies that

$$\sum_{\ell=1}^{m} \log\left(\mathbb{E}[e^{\lambda_\ell E_\ell}]\right) \preceq \sum_{\ell=1}^{m} \log\left(e^{\frac{E_\ell^2}{2}}\right) = \sum_{\ell=1}^{m} \frac{E_\ell^2}{2},$$

where we used $0 \prec A \preceq B \Rightarrow \log(A) \preceq \log(B)$, since the logarithm is a monotone function on $(0, \infty)$. Similarly, the exponent function is monotone on all of $\mathbb{R}$, and so we can take the matrix exponential on both sides to obtain

$$\exp\left(\sum_{\ell=1}^{m} \log\left(\mathbb{E}[e^{\lambda_\ell E_\ell}]\right)\right) \preceq \exp\left(\sum_{\ell=1}^{m} \frac{E_\ell^2}{2}\right).$$

Now, taking the trace on both sides leads to

$$\mathrm{Tr}\left[\exp\left(\sum_{\ell=1}^{m} \log\left(\mathbb{E}[e^{\lambda_\ell E_\ell}]\right)\right)\right] \leq \mathrm{Tr}\left[\exp\left(\sum_{\ell=1}^{m} \frac{E_\ell^2}{2}\right)\right],$$

where we used that $A \preceq B$ implies $B - A \succeq 0$, and so $\mathrm{Tr}[B] - \mathrm{Tr}[A] = \mathrm{Tr}[B - A] \geq 0$. Now, using Proposition 4.5.2, we obtain

$$\begin{aligned}\mathbb{P}[\lambda_{max}(Y) \geq t] &\leq \inf_{\theta>0} e^{-\theta t} \,\mathrm{Tr}\left[\exp\left(\sum_{\ell=1}^{m} \log \mathbb{E}[e^{\theta \lambda_\ell E_\ell}]\right)\right] \\ &\leq \inf_{\theta>0} e^{-\theta t} \,\mathrm{Tr}\left(\exp\left(\frac{\theta^2}{2} \sum_{\ell=1}^{m} E_\ell^2\right)\right) \leq \inf_{\theta>0} e^{-\theta t} d \cdot \left\|\exp\left(\frac{\theta^2}{2} \sum_{\ell=1}^{m} E_\ell^2\right)\right\| \\ &= \inf_{\theta>0} e^{-\theta t} d \cdot \exp\left(\frac{\theta^2}{2} \left\|\sum_{\ell=1}^{m} E_\ell^2\right\|\right) = d \inf_{\theta>0} \exp\left(-\theta t + \frac{\theta^2}{2} v(Y)\right).\end{aligned}$$

Plugging in $\theta = \frac{t}{v(Y)}$ yields

$$\mathbb{P}[\lambda_{\max}(Y) \geq t] \leq d e^{-\frac{t^2}{2v(t)}}.$$

By symmetry we get the same bound for the smallest eigenvalue. This completes the proof. □

### 4.5.2 Algorithm for estimating multiple observables

With the ideas used in the construction of the unbiased phase estimation algorithm, Algorithm 4.4.2, and the tail bounds on the operator norm of linear combinations of the operators derived in the previous subsection, we can now construct an algorithm that estimates multiple observables simultaneously. The idea is to take the bounded mean estimation algorithm, Algorithm 3.3.6, make it unbiased using the techniques introduced in Section 4.4, and analyze the truncation error throughout the algorithm using the techniques from Section 4.5.1.

We state the full algorithm below. To improve the clarity of the proof of its properties, we refer to the analyses of the algorithms introduced earlier.

**Algorithm 4.5.4:** Multiple observable estimation

**Input:**

1: $(E_\ell)_{\ell=1}^m$: a finite sequence of observables on $\mathcal{H}$.
2: $0 < \varepsilon < \max_{\ell\in[m]} \|E_\ell\|_\infty / 2$: the precision parameter.
3: $\delta > 0$: the failure probability parameter.
4: $U$: a state-preparation oracle acting on $\mathcal{H} \otimes \mathcal{W}$, that implements $|0\rangle \mapsto |\psi\rangle$.

**Derived objects:**

1: $d = \dim(\mathcal{H})$.
2: $\rho = \mathrm{Tr}_{\mathcal{W}}[|\psi\rangle \langle\psi|]$.
3: $\mathbf{z} \in \mathbb{R}^m$, where for all $j \in [m]$, $z_j = \mathrm{Tr}[\rho E_j]$.
4: $S = \max_{j\in[m]} \|E_j\|_\infty$.
5: $k = \lceil \log\left(\frac{36S}{\varepsilon}\right) \rceil$.
6: $k' = \lceil 2\log\left(\frac{4m}{\delta}\right) \rceil$.
7:

$$M = \sqrt{2 \cdot \left\| \sum_{\ell=1}^m E_\ell^2 \right\| \cdot \ln\left(\frac{4k'2^{2k}dm^2}{9\delta}\right)}.$$

**Output:** A random variable $\widetilde{\mathbf{z}}$ that is $\delta$-close in total variation distance to a random variable $\mathbf{z}' \in \mathbb{R}^m$ that has independent entries, satisfies $\|\mathbf{z}' - \mathbf{z}\|_\infty \leq \varepsilon$ almost surely, and satisfies $\mathbb{E}[\mathbf{z}'] = \mathbf{z}$.

**Queries:** Number of calls to $U$:

$$\widetilde{\mathcal{O}}\left(\frac{\sqrt{\|\sum_{\ell=1}^m E_\ell^2\|}}{\varepsilon}\right).$$

**Procedure:** MULTIPLE-OBSERVABLE-EST($(E_\ell)_{\ell=1}^m$, $\varepsilon$, $\delta$, $U$)

1: For $j' = 1, \ldots, k'$,

1. Choose a vector $\phi \in (-1/2, 1/2]^m$ uniformly at random.
2. Prepare the superposition

$$\frac{1}{\sqrt{2^{km}}} \sum_{\mathbf{x}\in G_k^m} e^{2\pi i \mathbf{x}^T \phi} |\mathbf{x}\rangle,$$

where $G_k^m$ is the dual grid, defined in Definition 3.3.1.

3. Apply the operation HAM-SIM($A$, $2^k M/(3S)$, $\delta/(4k')$), where $A$ is the circuit acting on $\mathbb{C}^{G_k^m} \otimes \mathcal{H} \otimes \mathcal{W} \otimes \mathbb{C}^2$ that consists of the following operations:
   (a) Apply $U$ to the second and third registers.
   (b) Apply the following operation to the first, second and fourth registers:

$$|\mathbf{x}\rangle |\chi\rangle |0\rangle \mapsto |\mathbf{x}\rangle \left( \left( \frac{I}{2} + \frac{1}{2M} \left[\!\!\left[ \sum_{\ell=1}^{m} x_\ell E_\ell \right]\!\!\right]_0^M \right) |\chi\rangle |0\rangle + |\perp\rangle \right),$$

   where $|\perp\rangle$ is an unnormalized state that is orthogonal to $|\chi\rangle |0\rangle$, and

$$[\![A]\!]_0^M = \begin{cases} A, & \text{if } \|A\|_\infty \leq M, \\ 0, & \text{otherwise.} \end{cases}$$

   (c) Apply $U^\dagger$ to the second and third registers.
4. Apply the inverse $m$-dimensional quantum Fourier transform.
5. Measure in the computational basis, and denote the output by $\mathbf{j} \in \{-2^{k-1}+1, \ldots, 2^{k-1}\}^m$.
6. Let $\widetilde{\phi}^{(j')} = \mathbf{j}/2^k - \phi$, and add or subtract integers element-wise until all entries of $\phi_{j'}$ are in the interval $(-1/2, 1/2]$.

2: For all $j \in [m]$, find the shortest arc on the unit circle that contains a strict majority of the points $e^{2\pi i \widetilde{\phi}_j^{(j')}}$, with $j' \in [k']$. Let $\widetilde{\phi} \in (-1/2, 1/2]^m$ be such that for all $j \in [m]$, $e^{2\pi i \widetilde{\phi}_j}$ is the midpoint of this arc.

3: Output $\widetilde{\mathbf{z}} = 3S\widetilde{\phi}$.

---

**Proof of the properties of Algorithm 4.5.4:**
We start by counting the number of queries to $U$ that the algorithm performs. From the properties of Algorithm 2.4.8, we obtain that in step 1.3, we perform $\mathcal{O}(2^k M/(3S) + \log(\delta/(4k')))$ calls to $U$, and we execute this step $k' = \widetilde{\mathcal{O}}(1)$ times. Thus, the total number of times we call $U$ is the product of these, which evaluates to the expression from the algorithm statement.

It remains to check the claimed properties on the output of the algorithm. To that end, we first investigate the action of the operation defined in step 1.3. Let's first write the Schmidt decomposition of $|\psi\rangle \in \mathcal{H} \otimes \mathcal{W}$, as

$$|\psi\rangle = \sum_{j=1}^{r} \sqrt{\lambda_j} |\psi_j\rangle |\chi_j\rangle,$$

where $\{|\psi_j\rangle : j \in [r]\}$ and $\{|\chi_j\rangle : j \in [r]\}$ are orthonormal sets of vectors in $\mathcal{H}$ and $\mathcal{W}$, respectively. This implies that

$$\rho = \sum_{j=1}^{r} \lambda_j |\psi_j\rangle \langle\psi_j| .$$

Now, observe that if we apply the operations in steps 1.3a–1.3c, to the state $|\mathbf{x}\rangle |0\rangle |0\rangle |0\rangle$, then after step 1.3a we have the state

$$|\mathbf{x}\rangle \sum_{j=1}^{r} \sqrt{\lambda_j} |\psi_j\rangle |\chi_j\rangle |0\rangle .$$

Then, after step 1.3b, we obtain the state

$$|\mathbf{x}\rangle \sum_{j=1}^{r} \sqrt{\lambda_j} \left( \frac{I}{2} + \frac{1}{2M} \left[\!\!\left[ \sum_{\ell=1}^{m} x_\ell E_\ell \right]\!\!\right]_0^M \right) |\psi_j\rangle |\chi_j\rangle |0\rangle + |\perp\rangle ,$$

where $|\perp\rangle$ is orthogonal to $|\psi_j\rangle |\chi_j\rangle |0\rangle$, for, for all $j \in [r]$. Hence, after step 1.3c, the overlap of the state with $|0\rangle |0\rangle |0\rangle$ in the second, third and fourth register, respectively, is

$$\sum_{j=1}^{r} \lambda_j \langle\psi_j| \left( \frac{I}{2} + \frac{1}{2M} \left[\!\!\left[ \sum_{\ell=1}^{m} x_\ell E_\ell \right]\!\!\right]_0^M \right) |\psi_j\rangle = \frac{1}{2} + \frac{1}{2M} \operatorname{Tr} \left[ \rho \left[\!\!\left[ \sum_{\ell=1}^{m} x_\ell E_\ell \right]\!\!\right]_0^M \right] .$$

Thus, from the properties of Algorithm 2.4.8, we obtain that in step 1.3 altogether, we implement the operation

$$|\mathbf{x}\rangle \mapsto \exp\left( 2\pi i \cdot \frac{2^k}{3S} \operatorname{Tr} \left[ \rho \left[\!\!\left[ \sum_{\ell=1}^{m} x_\ell E_\ell \right]\!\!\right]_0^M \right] \right) |\mathbf{x}\rangle ,$$

where we neglect a global phase $e^{\pi i 2^k M}$, which doesn't depend on $\mathbf{x}$ and hence does not influence the measurement probabilities. Thus, when we apply step 1.3 to the state prepared in step 1.2, then we obtain the state

$$\frac{1}{\sqrt{2^{km}}} \sum_{\mathbf{x} \in G_k^m} \exp\left( 2\pi i \left( \mathbf{x}^T \phi + \frac{2^k}{3S} \operatorname{Tr} \left[ \rho \left[\!\!\left[ \sum_{\ell=1}^{m} x_\ell E_\ell \right]\!\!\right]_0^M \right] \right) \right) |\mathbf{x}\rangle =: |\chi'\rangle .$$

As in the proof of Algorithm 3.3.6, we analyze the influence of the truncation on the state, i.e., we analyze the norm difference between $|\chi'\rangle$ and

$$|\chi\rangle := \frac{1}{\sqrt{2^{km}}} \sum_{\mathbf{x} \in G_k^m} \exp\left( 2\pi i \left( \mathbf{x}^T \phi + \frac{2^k}{3S} \operatorname{Tr} \left[ \rho \sum_{\ell=1}^{m} x_\ell E_\ell \right] \right) \right) |\mathbf{x}\rangle .$$

To that end, observe that we can express the norm difference as

$$\begin{aligned}
&\left\||\chi\rangle - |\chi'\rangle\right\|^2 \\
&\quad= \frac{1}{2^{km}} \sum_{\mathbf{x}\in G_k^m} \left| \exp\left(2\pi i \left(\mathbf{x}^T\phi + \frac{2^k}{3S}\operatorname{Tr}\left[\rho \left[\!\!\left[\sum_{\ell=1}^m x_\ell E_\ell\right]\!\!\right]_0^M\right]\right)\right)\right. \\
&\qquad\left. - \exp\left(2\pi i \left(\mathbf{x}^T\phi + \frac{2^k}{3S}\operatorname{Tr}\left[\rho \sum_{\ell=1}^m x_\ell E_\ell\right]\right)\right)\right|^2 \\
&\quad\leq \frac{2^{2k}4\pi^2}{9S^2} \mathop{\mathbb{E}}_{\mathbf{x}\sim U(G_k^m)} \left[\operatorname{Tr}\left[\rho\left(\sum_{\ell=1}^m x_\ell E_\ell - \left[\!\!\left[\sum_{\ell=1}^m x_\ell E_\ell\right]\!\!\right]_0^M\right)\right]^2\right] \\
&\quad\leq \frac{2^{2k}4\pi^2}{9S^2} \mathop{\mathbb{E}}_{\mathbf{x}\sim U(G_k^m)} \left[\left\|\sum_{\ell=1}^m x_\ell E_\ell - \left[\!\!\left[\sum_{\ell=1}^m x_\ell E_\ell\right]\!\!\right]_0^M\right\|_\infty^2\right],
\end{aligned}$$

where we used that $|e^{ix} - e^{iy}| \leq |x-y|$, for all $x, y \in \mathbb{R}$, and $|\operatorname{Tr}[AB]| \leq \operatorname{Tr}[A] \cdot \|B\|_\infty$ for Hermitian operators $A, B \in \mathcal{L}(\mathcal{H})$, with $A \succeq 0$.

Next, remember that $S = \max_{j\in[d]} \|E_j\|$, and so the operator norm of $\sum_{\ell=1}^m x_\ell E_\ell$ is always upper bounded by $mS/2$. Thus,

$$\begin{aligned}
\left\||\chi\rangle - |\chi'\rangle\right\|^2 &\leq \frac{2^{2k}\pi^2 m^2 S^2}{9S^2} \cdot \mathop{\mathbb{P}}_{\mathbf{x}\sim U(G_k^m)} \left[\left\|\sum_{\ell=1}^m x_\ell E_\ell\right\|_\infty \geq M\right] \\
&\leq \frac{2^{2k}2\pi^2 dm^2}{9} \cdot \exp\left(-\frac{M^2}{2\left\|\sum_{\ell=1}^m E_\ell^2\right\|}\right) \leq \frac{\delta}{4k'},
\end{aligned}$$

where in the last line we used Theorem 4.5.3, and the choice of $M$ in the algorithm statement.

Thus, in the remainder of the proof, we assume that we are in the state $|\chi\rangle$ after step 1.3. Moreover, we let $\overline{\phi}^{(j')}$, $\overline{\phi}$ and $\overline{\mathbf{z}}$ be the variables $\widetilde{\phi}^{(j')}$, $\widetilde{\phi}$ and $\widetilde{\mathbf{z}}$ one would have obtained if the state after step 3 would have been $|\chi\rangle$. Since we make a total norm error of $\delta/4$ in the construction of $|\chi\rangle$, as well as a total norm error of $\delta/4$ in the calls to Hamiltonian simulation, every pair of these variables is $\delta/2$-close in total variation distance.

Next, observe that we can rewrite

$$\begin{aligned}
|\chi\rangle &= \frac{1}{\sqrt{2^{km}}} \sum_{\mathbf{x}\in G_k^m} \exp\left(2\pi i \left(\mathbf{x}^T\phi + \frac{2^k}{3S}\operatorname{Tr}\left[\rho\sum_{\ell=1}^m x_\ell E_\ell\right]\right)\right)|\mathbf{x}\rangle \\
&= \frac{1}{\sqrt{2^{km}}} \sum_{\mathbf{x}\in G_k^m} \exp\left(2\pi i \left(\mathbf{x}^T\phi + \frac{2^k}{3S}\sum_{\ell=1}^m x_\ell \operatorname{Tr}\left[\rho E_\ell\right]\right)\right)|\mathbf{x}\rangle \\
&= \frac{1}{\sqrt{2^{km}}} \sum_{\mathbf{x}\in G_k^m} \exp\left(2\pi i \left(\mathbf{x}^T\left(\phi + \frac{2^k}{3S}\mathbf{z}\right)\right)\right)|\mathbf{x}\rangle \\
&= \bigotimes_{\ell=1}^m \left|\mathrm{QFT}_{2^k}\left(\phi_j + \frac{2^k}{3S}z_j\right)\right\rangle,
\end{aligned}$$

where the notation $|\mathrm{QFT}_{2^k}(\cdot)\rangle$ is introduced in Definition 2.4.1. From the analysis of Algorithm 4.4.2, we now obtain that all the coordinates of $\overline{\phi}$, and hence also all the entries of $\overline{\mathbf{z}}$, are independent from one another. Moreover, for all $j \in [m]$, $\mathbb{E}[e^{2\pi i\overline{\phi}_j}] = e^{2\pi i z_j/(3S)}$, and the probability distribution of $e^{2\pi i\overline{\phi}_j}$ is symmetric around $e^{2\pi i z_j/(3S)}$.

Furthermore, we find that

$$\frac{|z_j|}{3S} = \frac{|\operatorname{Tr}[\rho E_j]|}{3S} \leq \frac{\|E_j\|_\infty}{3S} \leq \frac{1}{3}.$$

Since we required that $\varepsilon < S/2$, the interval $[z_j/(3S) - \varepsilon/(3S), z_j/(3S) + \varepsilon/(3S)]$ is contained in the interval $(-1/2, 1/2)$. Thus, from the properties of Algorithm 4.4.2, we obtain that for all $j \in [m]$,

$$\begin{aligned}
\mathbb{P}[|\overline{z}_j - z_j| > \varepsilon] &= \mathbb{P}\left[\text{cyclic-dist}\left(\overline{\phi}_j, \frac{z_j}{3S}\right) > \frac{2^k\varepsilon}{2^k\cdot 3S}\right] \\
&\leq \left(\frac{2^k\varepsilon}{18S}\right)^{-k'/2} \leq 2^{-k'/2} \leq \frac{\delta}{4m},
\end{aligned}$$

where the last two inequalities hold by the choices of $k$ and $k'$. Thus, by the union bound, we find that

$$\mathbb{P}\left[\|\overline{\mathbf{z}} - \mathbf{z}\|_\infty \leq \varepsilon\right] \geq 1 - \frac{\delta}{4}.$$

Now, we define the random variable $\mathbf{z}' \in \mathbb{R}^d$ as the random variable that is equal to $\overline{\mathbf{z}}$ if $\|\overline{\mathbf{z}} - \mathbf{z}\|_\infty \leq \varepsilon$, and $\mathbf{z}$ otherwise. Since we are only changing the random variable $\overline{\mathbf{z}}$ with probability at most $\delta/4$, $\mathbf{z}'$ and $\overline{\mathbf{z}}$ are $\delta/2$-close in total variation distance. In particular, this implies that $\widetilde{\mathbf{z}}$ and $\mathbf{z}'$ are $\delta$-close in total variation distance.

Finally, observe that $\mathbf{z}'$ satisfies $\|\mathbf{z}' - \mathbf{z}\|_\infty \leq \varepsilon$ by assumption, its coordinates are independent and its distribution is symmetric around $\mathbf{z}$. In particular, this implies that $\mathbb{E}[\mathbf{z}'] = \mathbf{z}$. This completes the proof. □

## 4.6 Mixed-state tomography

In this section, we show how the algorithm that estimates multiple observables simultaneously can be used to construct an algorithm that performs mixed-state tomography. The core idea is remarkably straightforward – if the dimension of the density matrix is $d$, then we simply define $\mathcal{O}(d^2)$ observables that estimate all of the entries of the matrix individually, and plug them into Algorithm 4.5.4.

This naive approach naturally only gives us coordinate-wise guarantees on the precision of our estimate of the density matrix we are after. Thus, we need a lemma that relates entry-wise estimators of matrices to their operator norm. We present this lemma first, and afterwards state the resulting algorithm.

**4.6.1.** Definition (Subgaussian random variable [RV10, Definition 2.2]).
A random variable $X$ is *subgaussian* if there exists a $K > 0$, called the subgaussian moment of $X$, such that for all $t \geq 0$,

$$\mathbb{P}[|X| > t] \leq 2e^{-t^2/K^2}. \qquad \blacktriangleleft$$

We easily observe that every bounded random variable $X \in [-B, B]$ has subgaussian moment $\leq \sqrt{\ln(2)}B$. Next, we recall a concentration inequality on the operator norm of a matrix, if all of its entries are independent and subgaussian random variables.

**4.6.2.** Lemma (Operator norm of subgaussian matrices [RV10, Proposition 2.4]).
*Let $X$ be an $N \times n$ random matrix whose entries are independent mean-zero subgaussian random variables whose subgaussian moments are bounded by $K$. Then, for all $t \geq 0$, we have*

$$\mathbb{P}\left[\frac{\|X\|}{K} > C(\sqrt{N} + \sqrt{n}) + t\right] \leq 2e^{-ct^2},$$

*where $C$ and $c$ denote positive absolute constants.*

Now, for all $j, j' \in [d]$, consider the $d$-dimensional observables

$$E_{j,j'} = \frac{|j'\rangle\langle j| + |j\rangle\langle j'|}{2}, \qquad \text{and} \qquad E'_{j,j'} = \frac{|j'\rangle\langle j| - |j\rangle\langle j'|}{2i}. \tag{4.6.1}$$

For any density matrix $\rho \in \mathbb{C}^{d\times d}$, we find that

$$\mathrm{Tr}[\rho E_{j,j'}] = \frac{\rho_{j,j'} + \rho_{j',j}}{2} = \mathrm{Re}[\rho_{j,j'}], \quad \text{and} \quad \mathrm{Tr}[\rho E'_{j,j'}] = \frac{\rho_{j,j'} - \rho_{j',j}}{2i} = \mathrm{Im}[\rho_{j,j'}],$$

where we use that $\rho$ is Hermitian. Thus, if we run Algorithm 4.5.4 with the observables $E_{j,j'}$ for all integer $1 \leq j' \leq j \leq d$, then we obtain the real part of $\rho$, and similarly if we run it with the observables $E'_{j,j'}$ for all integer $1 \leq j' < j \leq d$, then we obtain the imaginary part of $\rho$.

We develop this approach in more detail in the algorithm below.

**Algorithm 4.6.3:** Mixed-state tomography w.r.t. the operator norm

**Input:**

1: $U$: a quantum circuit acting on $\mathcal{H}\otimes\mathcal{W}$, implementing the operation $|0\rangle \mapsto |\psi\rangle$.
2: $\varepsilon > 0$: the precision tolerance parameter.
3: $\delta > 0$: the failure probability tolerance parameter.
4: $1 \leq r \leq d$: an upper bound on the rank of the density matrix $\rho$.

**Derived objects:**

1: $\rho = \mathrm{Tr}_{\mathcal{W}}(|\psi\rangle\langle\psi|)$.
2: For all $j, j' \in [d]$, we let $E_{j,j'}$ and $E'_{j,j'}$ be as in Equation (4.6.1).
3: Let $c$ and $C$ be as in Lemma 4.6.2, and

$$\varepsilon' = \frac{\varepsilon}{4\ln(2)\left(\sqrt{\ln\left(\frac{8}{\delta}\right)/c + 2C\sqrt{d}}\right)}.$$

**Output:** A density matrix $\widetilde{\rho} \in \mathbb{C}^{d\times d}$ with rank at most $r$ such that $\|\widetilde{\rho} - \rho\|_\infty \leq \varepsilon$.
**Success probability:** Lower bounded by $1 - \delta$.
**Queries:** Number of calls to $U$: $\widetilde{\mathcal{O}}(d/\varepsilon)$.
**Procedure:** Mixed-state-tomography($U$, $\varepsilon$, $\delta$, $r$):

1: Let $\mathbf{z}$ = Multiple-observable-est($(E_{j,j'})_{1\leq j'\leq j\leq d}$, $\varepsilon'$, $\delta/4$, $U$).
2: Let $\mathbf{z}'$ = Multiple-observable-est($(E'_{j,j'})_{1\leq j'<j\leq d}$, $\varepsilon'$, $\delta/4$, $U$).
3: Let

$$\overline{\rho}_{\mathrm{Re}} = \sum_{j=1}^{d}\left[z_{j,j}|j\rangle\langle j| + \sum_{j'=1}^{j-1} z_{j,j'}|j\rangle\langle j'| + z_{j,j'}|j'\rangle\langle j|\right].$$

4: Let

$$\overline{\rho}_{\mathrm{Im}} = \sum_{j=1}^{d}\sum_{j'=1}^{j-1} z'_{j,j'}|j\rangle\langle j'| - z'_{j,j'}|j'\rangle\langle j|.$$

5: Let $\overline{\rho} = \overline{\rho}_{\mathrm{Re}} + i\overline{\rho}_{\mathrm{Im}}$.
6: Among all density matrices $\widetilde{\rho} \in \mathbb{C}^{d\times d}$ of rank at most $r$, output the one that minimizes the operator distance $\|\widetilde{\rho} - \overline{\rho}\|_\infty$.

**Proof of the properties of Algorithm 4.6.3:**
To verify the claim on the number of queries, we must calculate the sum of the squares of the observables. To that end, observe that for every $j, j' \in [d]$ with $j \neq j'$, we have

$$E_{j,j'}^2 = \frac{|j\rangle\langle j| + |j'\rangle\langle j'|}{4} = (E'_{j,j'})^2,$$

and if $j = j'$, we have $E_{j,j}^2 = |j\rangle\langle j|$. Thus,

$$\sum_{j=1}^{d}\sum_{j'=1}^{j} E_{j,j'}^2 = \sum_{j=1}^{d}|j\rangle\langle j| + \frac{d-1}{4}\sum_{j=1}^{d}|j\rangle\langle j| = \frac{d+3}{4}I,$$

and similarly

$$\sum_{j=2}^{d}\sum_{j'=1}^{j-1}(E'_{j,j'})^2 = \frac{d-1}{4}\sum_{j=1}^{d}|j\rangle\langle j| = \frac{d-1}{4}I.$$

Thus, the norms of the right-hand sides are $\mathcal{O}(\sqrt{d})$. Moreover, observe that $\varepsilon' = \widetilde{\Omega}(\varepsilon/\sqrt{d})$, and so the number of queries in steps 1 and 2 are both

$$\widetilde{\mathcal{O}}\left(\frac{\sqrt{d}}{\varepsilon'}\right) = \widetilde{\mathcal{O}}\left(\frac{d}{\varepsilon}\right).$$

Hence, it remains to check the validity of the output. To that end, observe that up to total variance distance $\delta/2$, both random variables $\mathbf{z}$ and $\mathbf{z}'$ are unbiased and contained in an $\ell_\infty$-cube of radius $\varepsilon$ around the expectation values of the observables. Hence, $\overline{\rho}_{\mathrm{Re}}$ and $\overline{\rho}_{\mathrm{Im}}$ are unbiased estimators of $\mathrm{Re}[\rho]$ and $\mathrm{Im}[\rho]$, with independent entries, and all the entries of $\overline{\rho}_{\mathrm{Re}} - \mathrm{Re}[\rho]$ and $\overline{\rho}_{\mathrm{Im}} - \mathrm{Im}[\rho]$ are contained in the interval $[-\varepsilon', \varepsilon']$, and as such are subgaussian with subgaussian moment $K = \ln(2)\varepsilon'$. Next, from Lemma 4.6.2, we obtain that

$$\begin{aligned}\mathbb{P}\left[\left\|\overline{\rho}_{\mathrm{Re}} - \mathrm{Re}[\rho]\right\|_\infty \geq \frac{\varepsilon}{4}\right] &= \mathbb{P}\left[\frac{\left\|\overline{\rho}_{\mathrm{Re}} - \mathrm{Re}[\rho]\right\|_\infty}{K} \geq 2C\sqrt{d} + \frac{\varepsilon}{4K} - 2C\sqrt{d}\right] \\ &\leq 2\exp\left(-c\left(\frac{\varepsilon}{4K} - 2C\sqrt{d}\right)^2\right) = \delta/4,\end{aligned}$$

where the last equality holds by the choice of $\varepsilon'$ in the algorithm statement. Similarly we find that with probability at most $\delta/4$, $\left\|\overline{\rho}_{\mathrm{Im}} - \mathrm{Im}[\rho]\right\|_\infty \geq \varepsilon/4$. Putting everything together, we obtain that with probability at least $1 - \delta$

$$\left\|\overline{\rho} - \rho\right\|_\infty \leq \left\|\overline{\rho}_{\mathrm{Re}} - \mathrm{Re}[\rho]\right\|_\infty + \left\|\overline{\rho}_{\mathrm{Im}} - \mathrm{Im}[\rho]\right\|_\infty \leq \frac{\varepsilon}{2}.$$

Finally, observe that $\rho$ is itself a density matrix of rank at most $r$ that is at most $\varepsilon/2$ away from $\overline{\rho}$. Hence, $\left\|\widetilde{\rho} - \overline{\rho}\right\|_\infty \leq \varepsilon/2$, and thus we find from the triangle inequality that

$$\left\|\widetilde{\rho} - \rho\right\|_\infty \leq \left\|\widetilde{\rho} - \overline{\rho}\right\|_\infty + \left\|\overline{\rho} - \rho\right\|_\infty \leq \varepsilon.$$

This completes the proof. □

Thus, we have constructed an algorithm that performs state tomography up to precision $\varepsilon$ w.r.t. the operator norm. Now, as the final step, we use the norm conversion results from Section 4.3 to analyze how well this algorithm performs w.r.t. other Schatten norms.

**Algorithm 4.6.4:** Mixed-state tomography w.r.t. Schatten norms

**Input:**

1: $U$: a quantum circuit acting on $\mathcal{H}\otimes\mathcal{W}$, implementing the operation $|0\rangle \mapsto |\psi\rangle$.
2: $q \in [1, \infty]$: the Schatten norm.
3: $\varepsilon > 0$: the precision tolerance parameter.
4: $\delta > 0$: the failure probability tolerance parameter.
5: $1 \leq r \leq d$: an upper bound on the rank of the density matrix $\rho$.

**Derived objects:**

1: $\rho = \mathrm{Tr}_{\mathcal{W}}(|\psi\rangle \langle\psi|)$.
2: $\varepsilon' = \max\{\varepsilon/r^{1/q}, \varepsilon^{1/(1-1/q)}\}$.

**Output:** A density matrix $\widetilde{\rho}$ with rank at most $r$, such that $\|\widetilde{\rho} - \rho\|_q \leq \varepsilon$.
**Success probability:** Lower bounded by $1 - \delta$.
**Queries:** Number of calls to $U$:

$$\widetilde{\mathcal{O}}\left(\min\left\{\frac{dr^{\frac{1}{q}}}{\varepsilon}, \frac{d}{\varepsilon^{\frac{1}{1-\frac{1}{q}}}}\right\}\right).$$

**Procedure:** Mixed-state-tomography($U$, $q$, $\varepsilon$, $\delta$, $r$):
1: Output $\widetilde{\rho} =$ Mixed-state-tomography($U$, $\varepsilon'$, $\delta$, $r$).

---

**Proof of the properties of Algorithm 4.6.4:**
Observe from the properties of Algorithm 4.6.3 that $\|\widetilde{\rho} - \rho\|_\infty \leq \varepsilon'$. Thus, we obtain from Lemma 4.3.2, with $s = 1$ and applied to the $r$-dimensional vector of eigenvalues of $\widetilde{\rho} - \rho$, that

$$\|\widetilde{\rho} - \rho\|_q \leq \min\{\varepsilon' \cdot r^{\frac{1}{q}}, (\varepsilon')^{1-\frac{1}{q}}\}.$$

Hence, if $\varepsilon' = \varepsilon/r^{1/q}$, then the first term becomes $\varepsilon$, and similarly if $\varepsilon' = \varepsilon^{1/(1-1/q)}$, then the second term becomes $\varepsilon$. Thus, we indeed obtain an $\varepsilon$-precise approximation of $\widetilde{\rho}$ w.r.t. the Schatten $q$-norm, and plugging in $\varepsilon'$ into the number of calls to $U$ in Algorithm 4.6.3 yields

$$\widetilde{\mathcal{O}}\left(\frac{d}{\varepsilon'}\right) = \widetilde{\mathcal{O}}\left(\min\left\{\frac{dr^{\frac{1}{q}}}{\varepsilon}, \frac{d}{\varepsilon^{\frac{1}{1-\frac{1}{q}}}}\right\}\right).$$

This completes the proof. □

This completes the construction of the mixed-state tomography algorithm. In the next section, we turn to proving a matching lower bound on the query complexity of the mixed-state tomography problem, and hence proving optimality up to polylogarithmic factors of Algorithm 4.6.4. Afterwards, in Section 4.8, we list several direct implications of these algorithms.

## 4.7 Lower bounds

In this section, we prove tight lower bounds on the problem of mixed state tomography. More specifically, we prove that the algorithm outlined in Section 4.6 is

essentially optimal in all Schatten $q$-norms. The optimality in all other Schatten norms follows directly from it. The results obtained in this section are summarized in Theorem 4.7.7.

The core idea is to take a bit string $b \in \{0,1\}^{dr}$, and hide it in a density matrix $\rho_b \in \mathbb{C}^{d\times d}$, with rank at most $r$, such that its purification can be constructed with exactly one fractional phase query $O_b^\varepsilon$. Next, we show that if we learn a classical description $\widetilde{\rho}$ of $\rho_b$ such that $\|\widetilde{\rho} - \rho_b\|_1 \leq \varepsilon$, we narrow down the size of the set of bit strings $b' \in \{0,1\}^{dr}$ that satisfy $\|\widetilde{\rho} - \rho_{b'}\|_1 \leq \varepsilon$ to $O(2^{(1-c)dr})$, for some constant $c > 0$. Finally, we show that this must require $\Omega(dr/\varepsilon)$ queries to $O_b^\varepsilon$, and hence the state tomography algorithm must make at least this number of queries to the state preparation unitary too.

We start by defining the density matrices $\rho_b \in \mathbb{C}^{d\times d}$, in which we hide the bit string $b \in \{0,1\}^{dr}$, in the following definition.

**4.7.1.** Definition. Let $\varepsilon \in [0,1]$, $d \in \mathbb{N}$, $r \in [d]$, and $U^{(0)}, \ldots, U^{(r-1)} \in \mathbb{C}^{d\times d}$ unitaries to be fixed later. Let $b \in \{0,1\}^{dr}$ be a bit string of length $dr$. We write $b = (b^{(0)}, \ldots, b^{(r-1)})$, where $b^{(j)}$ is the $j$th block of size $d$. Let $|\psi_b\rangle \in \mathbb{C}^2 \otimes \mathbb{C}^{[d-1]_0} \otimes \mathbb{C}^{[r-1]_0}$ be explicitly defined as

$$|\psi_b\rangle = \frac{1}{\sqrt{dr}} \sum_{j=0}^{r-1} \sum_{k=0}^{d-1} \sum_{c\in\{-1,1\}} \sqrt{\frac{1}{2} + \frac{1}{2} c\varepsilon (-1)^{b_k^{(j)}}}\, |c\rangle\, U^{(j)} |k\rangle\, |j\rangle\,,$$

and let $\rho_b$ be the density matrix that is obtained by tracing out the last register of $|\psi_b\rangle\langle\psi_b|$. ◀

We easily compute that the norm of $|\psi_b\rangle$ is 1, and so it defined a valid quantum state. We can directly derive some interesting properties of the density matrices $\rho_b$ defined above, without fixing the unitaries $U^{(j)}$, as we discuss in Lemma 4.7.2. We will choose the unitaries explicitly in Definition 4.7.3.

**4.7.2.** Lemma. *Let $\varepsilon$, $d$, $r$ and $U^{(0)}, \ldots, U^{(r-1)}$ as in Definition 4.7.1. Let $b, \bar{b} \in \{0,1\}^{dr}$. Let $(\delta^{(0)}, \ldots, \delta^{(r-1)}) = \delta \in \{-2, 0, 2\}^{dr}$ be such that for every $j \in [r-1]_0$ and $k \in [d-1]_0$, $\delta_k^{(j)} = (-1)^{b_k^{(j)}} - (-1)^{\bar{b}_k^{(j)}}$. Furthermore, let $X, Y \in \mathbb{C}^{[d-1]_0 \times [r-1]_0}$ be defined as*

$$X = \begin{bmatrix} U^{(0)}\delta^{(0)} & \cdots & U^{(r-1)}\delta^{(r-1)} \end{bmatrix}, \qquad \text{and} \qquad Y = \begin{bmatrix} U^{(0)}\mathbf{1} & \cdots & U^{(r-1)}\mathbf{1} \end{bmatrix},$$

*where $\mathbf{1} \in \mathbb{C}^{[d-1]_0}$ is the all-ones vector in $d$ dimensions. Then*

$$\|\rho_b - \rho_{\bar{b}}\|_1 \geq \frac{\varepsilon}{rd} \left\|XY^\dagger\right\|_1 - 4\varepsilon^2.$$

**Proof:**
By taking the partial trace, we obtain that

$$\rho_b = \frac{1}{dr} \sum_{j=0}^{r-1} \sum_{k,k'=0}^{d-1} \sum_{c,c'\in\{-1,1\}} \sqrt{\frac{1}{2} + \frac{1}{2} c\varepsilon (-1)^{b_k^{(j)}}} \cdot \sqrt{\frac{1}{2} + \frac{1}{2} c'\varepsilon (-1)^{b_{k'}^{(j)}}}\, |c\rangle\langle c'| \otimes U^{(j)} |k\rangle\langle k'| \left(U^{(j)}\right)^\dagger.$$

Next, we approximate both square roots by their tangents around 1/2, i.e., we write $\sqrt{1/2+x} \approx (1+x/2)/\sqrt{2}$, and subsequently we neglect the the cross term, i.e., we write $(1+x)(1+y) \approx 1+x+y$ when $x, y \in \mathbb{R}$ are small. We denote the resulting matrix by $\overline{\rho}_b$, and we can express it directly as

$$\overline{\rho}_b = \frac{1}{2dr} \sum_{j=0}^{r-1} \sum_{k,k'=0}^{d-1} \sum_{c,c' \in \{-1,1\}} \left(1 + \frac{1}{2} c\varepsilon (-1)^{b_k^{(j)}} + \frac{1}{2} c'\varepsilon (-1)^{b_{k'}^{(j)}}\right) |c\rangle \langle c'| \otimes U^{(j)} |k\rangle \langle k'| \left(U^{(j)}\right)^\dagger .$$

Next, we characterize the total error introduced by the above approximations. To that end, observe that if $c(-1)^{b_k^{(j)}}$ and $c'(-1)^{b_{k'}^{(j)}}$ are equal, then both expressions under the square root are equal, and hence their product becomes exactly the product of the linearizations. Thus, the only entries in which the matrices $\rho_b$ and $\overline{\rho}_b$ differ are those for which $c(-1)^{b_k^{(j)}}$ and $c'(-1)^{b_{k'}^{(j)}}$ differ. This allows us to write the difference between $\rho_b$ and $\overline{\rho}_b$ succinctly, as

$$\rho_b - \overline{\rho}_b = \frac{1-\sqrt{1-\varepsilon^2}}{2dr} \sum_{j=0}^{r-1} \sum_{c,c' \in \{-1,1\}} \sum_{\substack{k,k'=0 \\ c(-1)^{b_k^{(j)}} \neq c'(-1)^{b_{k'}^{(j)}}}}^{d-1} |c\rangle \langle c'| \otimes U^{(j)} |k\rangle \langle k'| \left(U^{(j)}\right)^\dagger .$$

Thus, by taking the trace norm, applying the triangle inequality, removing the unitary transformations on the last register, and using $1 - \sqrt{1-\varepsilon^2} \leq \varepsilon^2$, we obtain that

$$\|\rho_b - \overline{\rho}_b\|_1 \leq \frac{\varepsilon^2}{2dr} \sum_{j=0}^{r-1} \left[ \left\| \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \sum_{\substack{k,k'=0 \\ (-1)^{b_k^{(j)}} \neq (-1)^{b_{k'}^{(j)}}}}^{d-1} |k\rangle \langle k'| \right\|_1 + \left\| \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes \sum_{\substack{k,k'=0 \\ (-1)^{b_k^{(j)}} = (-1)^{b_{k'}^{(j)}}}}^{d-1} |k\rangle \langle k'| \right\|_1 \right].$$

Since the $2 \times 2$ matrices both have 2 eigenvalues that are both of modulus 1, we can factor them out at the expense of a factor of 2. Then, after reordering rows and columns such that all $k$'s with $b_k^{(j)} = 1$ are at the upper-left corner, we obtain two big diagonal all-ones blocks, i.e.,

$$\|\rho_b - \overline{\rho}_b\|_1 \leq \frac{\varepsilon^2}{dr} \sum_{j=0}^{r-1} \left[ \left\| \begin{array}{c|cc} & b_k^{(j)}=1 & b_k^{(j)}=0 \\ \hline b_{k'}^{(j)}=1 & 1 & 0 \\ b_{k'}^{(j)}=0 & 0 & 1 \end{array} \right\|_1 + \left\| \begin{array}{c|cc} & b_k^{(j)}=1 & b_k^{(j)}=0 \\ \hline b_{k'}^{(j)}=1 & 0 & 1 \\ b_{k'}^{(j)}=0 & 1 & 0 \end{array} \right\|_1 \right].$$

The trace norm of the left term is $d$, since the two diagonal blocks have the single eigenvalues $|b^{(j)}|$ and $d - |b^{(j)}|$. On the right-hand side, the two eigenvalues are $\pm\sqrt{|b^{(j)}|(d - |b^{(j)}|)}$, and so the trace norm is $2\sqrt{|b^{(j)}|(d - |b^{(j)}|)} \leq 2\sqrt{d^2/4} = d$. Plugging these bounds into the above equation yields $\|\rho_b - \overline{\rho}_b\|_1 \leq 2\varepsilon^2$. Thus, we have

$$\|\overline{\rho}_b - \overline{\rho}_{\overline{b}}\|_1 \leq \|\rho_b - \overline{\rho}_b\|_1 + \|\rho_b - \rho_{\overline{b}}\|_1 + \|\rho_{\overline{b}} - \overline{\rho}_{\overline{b}}\|_1 \leq \|\rho_b - \rho_{\overline{b}}\|_1 + 4\varepsilon^2,$$

and hence it suffices to prove that $\|\overline{\rho}_b - \overline{\rho}_{\overline{b}}\|_1 = \varepsilon \left\|XY^\dagger\right\|_1 / (dr)$.

To that end, observe that we can write the difference as

$$\overline{\rho}_b - \overline{\rho}_{\overline{b}} = \frac{\varepsilon}{4dr} \sum_{j=0}^{r-1} \sum_{k,k'=0}^{d-1} \sum_{c,c' \in \{-1,1\}} \left( c\delta_k^{(j)} + c'\delta_{k'}^{(j)} \right) |c\rangle \langle c'| \otimes U^{(j)} |k\rangle \langle k'| \left(U^{(j)}\right)^\dagger.$$

Next, some of the indices can be decoupled from one another, which results in

$$\begin{aligned}\overline{\rho}_b - \overline{\rho}_{\overline{b}} = \frac{\varepsilon}{4dr} \Bigg[ &\sum_{j=0}^{r-1} \sum_{c \in \{-1,1\}} c\,|c\rangle \sum_{c' \in \{-1,1\}} \langle c'| \otimes U^{(j)} \sum_{k=0}^{d-1} \delta_k^{(j)} |k\rangle \sum_{k'=0}^{d-1} \langle k'| \left(U^{(j)}\right)^\dagger \\ &+ \sum_{j=0}^{r-1} \sum_{c \in \{-1,1\}} |c\rangle \sum_{c' \in \{-1,1\}} c'\,|c'\rangle \otimes U^{(j)} \sum_{k=0}^{d-1} |k\rangle \sum_{k'=0}^{d-1} \delta_{k'}^{(j)} \langle k'| \left(U^{(j)}\right)^\dagger \Bigg].\end{aligned}$$

Now, we can let

$$V = \begin{bmatrix} X \\ -X \end{bmatrix}, \qquad \text{and} \qquad W = \begin{bmatrix} Y \\ Y \end{bmatrix},$$

from which we can directly observe that the expression for $\overline{\rho}_b - \overline{\rho}_{\overline{b}}$ simplifies to

$$\overline{\rho}_b - \overline{\rho}_{\overline{b}} = \frac{\varepsilon}{4rd} \left[ VW^\dagger + WV^\dagger \right].$$

To evaluate the trace norm, we observe that

$$\|\overline{\rho}_b - \overline{\rho}_{\overline{b}}\|_1 = \operatorname{Tr}\left[\sqrt{(\overline{\rho}_b - \overline{\rho}_{\overline{b}})^2}\right] = \frac{\varepsilon}{4dr} \operatorname{Tr}\left[\sqrt{(VW^\dagger + WV^\dagger)^2}\right].$$

By direct calculation, we find that $V^\dagger W = W^\dagger V = 0$, and so when we expand the square, two of the terms cancel, and we end up with

$$\|\overline{\rho}_b - \overline{\rho}_{\overline{b}}\|_1 = \frac{\varepsilon}{4rd} \operatorname{Tr}\left[\sqrt{VW^\dagger WV^\dagger + WV^\dagger VW^\dagger}\right].$$

Again, since $V^\dagger W = W^\dagger V = 0$, we find that the two terms underneath the square root are only acting non-trivially on mutually orthogonal subspaces. Therefore, the square root of the sum is the sum of the square roots, and we end up with

$$\|\overline{\rho}_b - \overline{\rho}_{\overline{b}}\|_1 = \frac{\varepsilon}{4rd} \left[ \operatorname{Tr}\left[\sqrt{VW^\dagger WV^\dagger}\right] + \operatorname{Tr}\left[\sqrt{WV^\dagger VW^\dagger}\right] \right].$$

We rewrite the term on the left within parentheses as

$$\operatorname{Tr}\left[\sqrt{VW^\dagger WV^\dagger}\right] = \operatorname{Tr}\left[\sqrt{\begin{bmatrix} 2XY^\dagger YX^\dagger & -2XY^\dagger YX^\dagger \\ -2XY^\dagger YX^\dagger & 2XY^\dagger YX^\dagger \end{bmatrix}}\right]$$
$$= \operatorname{Tr}\left[\sqrt{\begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix} \otimes XY^\dagger YX^\dagger}\right].$$

The (non-zero) spectrum of the expression inside the square root is $4\sigma(XY^\dagger YX^\dagger)$, and hence the trace becomes $2\operatorname{Tr}[\sqrt{XY^\dagger YX^\dagger}] = 2\left\|XY^\dagger\right\|_1$. Analogously, the term on the right yields $2\left\|YX^\dagger\right\|$, and since $\left\|A^\dagger\right\|_1 = \|A\|_1$, for any matrix $A$, we obtain the expression from the statement of the lemma. □

It is worth noting that the matrix $XY^\dagger$ from the previous lemma statement is in general not Hermitian. Therefore, it is important to stress that the trace norm has to be interpreted as the sum of the singular values, rather than merely the sum of the absolute eigenvalues.

Next, we fix the unitary matrices $U^{(j)}$, for $j \in [r-1]_0$.

**4.7.3.** Definition. Let $d, r \in \mathbb{N}$, and let $j \in [r]$. Define $U^{(j)} \in \mathbb{C}^{d\times d}$ as

$$U^{(j)}_{k\ell} = \frac{1}{\sqrt{d}}\omega_d^{(j-k)\ell}.$$ ◀

The benefit of this particular choice of unitaries $U^{(j)}$ is that they further simplify the expressions that appear in Lemma 4.7.2. The details are presented in the lemma below.

**4.7.4.** Lemma. *Let $\varepsilon$, $d$ and $r$ as in Definition 4.7.1. Then, for all $j \in [r-1]_0$, $U^{(j)}$ is unitary,*

$$Y = \begin{bmatrix} U^{(0)}\mathbf{1} & \cdots & U^{(r-1)}\mathbf{1} \end{bmatrix} = \sqrt{d}\begin{bmatrix} I_r \\ 0 \end{bmatrix},$$

*where $\mathbf{1} \in \mathbb{C}^{[d-1]_0}$ is the all-ones vector in $d$ dimensions, and we have $\Delta \in \mathbb{C}^{[d-1]_0\times[r-1]_0}$ such that*

$$\|\rho_b - \rho_{\bar{b}}\|_1 \geq \frac{\varepsilon}{r\sqrt{d}}\|\Delta\|_1 - 2\varepsilon^2, \qquad \text{with} \qquad \Delta_{kj} = \omega_d^{jk}\delta_k^{(j)}.$$

**Proof:**
Let $j \in [r-1]_0$, and $\ell, \ell' \in [d-1]_0$. Then,

$$\left[\left(U^{(j)}\right)^\dagger U^{(j)}\right]_{\ell,\ell'} = \sum_{k=0}^{d-1}\overline{U}^{(j)}_{k\ell}U^{(j)}_{k\ell'} = \frac{1}{d}\sum_{k=0}^{d-1}\omega_d^{-(j-k)\ell}\omega_d^{(j-k)\ell'} = \frac{1}{d}\sum_{k=0}^{d-1}\omega_d^{(\ell'-\ell)k} = 1_{\ell=\ell'},$$

and thus indeed $\left(U^{(j)}\right)^\dagger U^{(j)} = I$. Next, let $j \in [r-1]_0$ and $k \in [d-1]_0$. We observe that

$$Y_{jk} = \left(U^{(j)}\mathbf{1}\right)_k = \sum_{\ell=0}^{d-1} U^{(j)}_{k\ell} = \frac{1}{\sqrt{d}} \sum_{\ell=0}^{d-1} \omega_d^{(j-k)\ell} = \sqrt{d} \cdot 1_{j=k},$$

as claimed. Finally, for the trace norm, observe that it suffices to show that $\|X\|_1 = \|\Delta\|_1$. To that end, for all $j \in [r-1]_0$ and $k \in [d-1]_0$,

$$\begin{aligned}\left[\left(U^{(0)}\right)^\dagger X\right]_{kj} = \left[\left(U^{(0)}\right)^\dagger U^{(j)}\delta^{(j)}\right]_k &= \frac{1}{d}\sum_{\ell,\ell'=0}^{d-1} \omega_d^{k\ell}\omega_d^{(j-\ell)\ell'}\delta^{(j)}_{\ell'} \\ &= \frac{1}{d}\sum_{\ell'=0}^{d-1}\left[\sum_{\ell=0}^{d-1}\omega_d^{(k-\ell')\ell}\right]\omega_d^{j\ell'}\delta^{(j)}_{\ell'}.\end{aligned}$$

The inner summation vanishes if $k \neq \ell'$, and evaluates to $d$ when $k = \ell'$. Therefore, the whole expression simplifies to $\omega_d^{jk}\delta^{(j)}_k = \Delta_{kj}$. Thus, $\Delta = \left(U^{(0)}\right)^\dagger X$, and since $U^{(0)}$ is unitary, $X$ and $\Delta$ have the same singular values, and hence the same trace norm as well. This completes the proof. □

Next, we choose two bit strings $b, \bar{b} \in \{0,1\}^{dr}$ uniformly at random, and we analyze the probability of the trace norm of the difference $\rho_b - \rho_{\bar{b}}$ being small. On a high level, the smaller this probability is, the fewer $\rho_{\bar{b}}$'s are close to a $\rho_b$ that is chosen uniformly at random.

**4.7.5.** Lemma. *Let $\varepsilon \in [0, 1/128]$, $d \in \mathbb{N}$ and $r \in [d]$. Let $b, \bar{b} \in \{0,1\}^{dr}$ uniformly at random. Then, there exist constants $c, d_0 > 0$ such that for all $d > d_0$,*

$$\mathbb{P}\left[\|\rho_b - \rho_{\bar{b}}\|_1 \leq \frac{\varepsilon}{64}\right] \leq e^{-crd}.$$

**Proof:**
By the previous lemma, we know that if $\|\Delta\|_1 \geq r\sqrt{d}/32$, then $\|\rho_b - \rho_{\bar{b}}\|_1 \geq \varepsilon/32 - 2\varepsilon^2 \geq \varepsilon/32 - \varepsilon/64 = \varepsilon/64$. Thus, it suffices to prove that there exist constants $c, d_0 > 0$ such that for all $d > d_0$,

$$\mathbb{P}\left[\|\Delta\|_1 \leq \frac{r\sqrt{d}}{32}\right] \leq e^{-crd}.$$

Next, let $A \in \mathbb{C}^{d\times d}$ be such that $\|A\|_\infty \leq 1$, and let $\Delta = U\Sigma V$ be the singular value decomposition of $\Delta$, with the singular values $\sigma_1, \ldots, \sigma_r$ (where we allow the $\sigma$'s to be 0 if the rank turns out to be strictly smaller than $r$). For technical reasons, we pad the matrices $\Sigma$ and $V$ with zeros, such that we have

$U, \Sigma, V \in \mathbb{C}^{d\times d}$. Let $\mathbf{u}_j$ and $\mathbf{v}_j$ denote the $j$th columns of $U$ and $V$, respectively. Then,

$$\begin{aligned}|\mathrm{Tr}\left[A\Delta\right]| = |\mathrm{Tr}\left[AU\Sigma V\right]| = |\mathrm{Tr}\left[VAU\Sigma\right]| &\leq \sum_{j=1}^{r} \left|\left[VAU\right]_{jj}\right| \sigma_j = \sum_{j=1}^{r} \left|\mathbf{v}_j^\dagger A \mathbf{u}_j\right| \sigma_j \\ &\leq \sum_{j=1}^{r} \sigma_j = \left\|\Delta\right\|_1,\end{aligned}$$

where we used that $|\mathbf{v}_j^\dagger A \mathbf{u}_j| \leq \|\mathbf{v}_j\| \cdot \|A\|_\infty \cdot \|\mathbf{u}_j\| \leq 1$. Thus, it is sufficient to find some matrix $A \in \mathbb{C}^{d\times d}$ such that $\|A\| \leq 1$, which is allowed to depend on $\Delta$, and prove that there exists a constant $c > 0$ such that

$$\mathbb{P}\left[|\mathrm{Tr}\left[A\Delta\right]| \leq \frac{r\sqrt{d}}{32}\right] \leq e^{-crd}.$$

The core idea is to turn the matrix $\Delta$ into an upper triangular matrix by modifying it column by column. To that end, we find a unitary matrix $A_1$ such that the first column of $A_1\Delta$ is a multiple of the standard basis vector $\mathbf{e}_1$. Next, we find a unitary matrix $A_2$ such that the second column of $A_2A_1\Delta$ has only zeros below the diagonal. We continue this process, until we reached the $r$th and last column of $\Delta$. Then, we let $A = A_r \cdots A_1$.

Since all the matrices $A_j$ are unitary, we find $\|A_r \cdots A_1\|_\infty \leq 1$. Moreover, we can analyze the probability distribution over the diagonal entries of $A\Delta$, and show an exponential tail bound on the trace in its lower limit.

Now, we formalize the above idea. To that end, for any vector $\mathbf{v}$, we let the matrix $A_\mathbf{v}$ be the operation that reflects through the subspace $\mathrm{Span}\{\mathbf{v}\}^\perp$, i.e.,

$$A_\mathbf{v} = I - 2\frac{\mathbf{v}\mathbf{v}^\dagger}{\|\mathbf{v}\|^2}.$$

Since $A_\mathbf{v}$ is a reflection, $A_\mathbf{v}$ is a unitary matrix and in particular we have $\|A_\mathbf{v}\| \leq 1$. Additionally, suppose we have $\mathbf{v}$ and $\mathbf{w}$ with $\|\mathbf{v}\| = \|\mathbf{w}\| = 1$ and $\mathrm{Im}(\mathbf{v}^\dagger\mathbf{w}) = 0$, then since $(\mathbf{v}-\mathbf{w})^\dagger(\mathbf{v}+\mathbf{w}) = \|\mathbf{v}\|^2 - \|\mathbf{w}\|^2 = 0$, we obtain

$$A_{\mathbf{v}-\mathbf{w}}\mathbf{v} = A_{\mathbf{v}-\mathbf{w}} \cdot \frac{1}{2}\left[\mathbf{v}-\mathbf{w}+\mathbf{v}+\mathbf{w}\right] = \frac{1}{2}\left[-\mathbf{v}+\mathbf{w}+\mathbf{v}+\mathbf{w}\right] = \mathbf{w}.$$

Now, for all $j \in [r]$, we denote the $j$th column of $\Delta$ by $\mathbf{x}_j$. We let $\mathbf{y}_1 = \mathbf{x}_1$, and for all $j \in [r]$ recursively define

$$\mathbf{z}_j = \begin{cases} \frac{\mathbf{y}_j}{\|\mathbf{y}_j\|} \cdot \frac{\overline{(y_j)_1}}{|(y_j)_1|}, & \text{if } (y_j)_1 \neq 0, \\ \frac{\mathbf{y}_j}{\|\mathbf{y}_j\|}, & \text{if } \mathbf{y}_j \neq \mathbf{0}, \\ \mathbf{e}_1, & \text{if } \mathbf{y}_j = \mathbf{0}, \end{cases} \qquad \varphi_j = \begin{cases} \frac{\overline{(y_j)_1}}{|(y_j)_1|} & \text{if } (y_j)_1 \neq 0, \\ 1, & \text{if } (y_j)_1 = 0, \end{cases},$$

and

$$A_j = \left[\begin{array}{c|c} I_{j-1} & 0 \\ \hline 0 & \varphi_j A_{\mathbf{z}_j - \mathbf{e}_1} \end{array}\right],$$

and furthermore

$$A_j \cdots A_1 \mathbf{x}_{j+1} = \left[\begin{array}{c} \mathbf{w}_{j+1} \in \mathbb{C}^j \\ \hline \mathbf{y}_{j+1} \in \mathbb{C}^{d-j} \end{array}\right], \qquad \text{and} \qquad A = A_r \cdots A_1.$$

From the construction, it is clear that all $A_j$'s are unitary, and so $\|A\| \leq 1$. Additionally,

$$A\mathbf{x}_j = A_r \cdots A_1 \mathbf{x}_j = A_r \cdots A_j \left[\begin{array}{c} \mathbf{w}_j \\ \hline \mathbf{y}_j \end{array}\right] = A_r \cdots A_{j+1} \left[\begin{array}{c} \mathbf{w}_j \\ \hline \varphi_j A_{\mathbf{z}_j - \mathbf{e}_1} \mathbf{y}_j \end{array}\right].$$

We can directly observe that $\mathrm{Im}(\mathbf{e}_1^\dagger \mathbf{z}_j) = 0$, for all $j \in [r]$. If $(y_j)_1 \neq 0$, then we find $\varphi_j A_{\mathbf{z}_1 - \mathbf{e}_1} \mathbf{y}_j = |(y_j)_1| \, \|\mathbf{y}_j\| \, \varphi_j A_{\mathbf{z}_1 - \mathbf{e}_1} \mathbf{z}_1 / \overline{(y_j)_1} = \|\mathbf{y}_1\| \, \mathbf{e}_1$. Similarly, if $(y_j)_1 = 0$ but $\mathbf{y}_j \neq \mathbf{0}$, then $\varphi_j A_{\mathbf{z}_1 - \mathbf{e}_1} \mathbf{y}_j = \|\mathbf{y}_1\| \, A_{\mathbf{z}_1 - \mathbf{e}_1} \mathbf{z}_1 = \|\mathbf{y}_1\| \, \mathbf{e}_1$. Finally, if $\mathbf{y}_j = \mathbf{0}$, then $\varphi_j A_{\mathbf{z}_1 - \mathbf{e}_1} \mathbf{y}_j = \mathbf{0} = \|\mathbf{y}_j\| \, \mathbf{e}_1$. Thus, in all cases we find

$$A\mathbf{x}_j = A_r \cdots A_{j+1} \left[\begin{array}{c} \mathbf{w}_j \\ \hline \|\mathbf{y}_j\| \, \mathbf{e}_1 \end{array}\right] = \left[\begin{array}{c} \mathbf{w}_j \\ \hline \|\mathbf{y}_j\| \, \mathbf{e}_1 \end{array}\right].$$

Moreover, observe that $A_j \cdots A_1$ is unitary and it only depends on the vectors $\mathbf{x}_1, \ldots, \mathbf{x}_j$. Therefore, $\mathbf{y}_{j+1}$ is a projection of $\mathbf{x}_{j+1}$ onto an $(d-j)$-dimensional subspace, which we denote by $S_{j+1}$, and we observe that $S_{j+1}$ does not depend on the vector $\mathbf{x}_{j+1}$ itself.

Finally, we have

$$\|\Delta\|_1 \geq |\operatorname{Tr}[A\Delta]| = \sum_{j=1}^{r} \|\mathbf{y}_j\| \geq \sum_{j=1}^{\min\{r, \lfloor d/2 \rfloor\}} \|\mathbf{y}_j\|.$$

Now, if there are at least $r/4$ $j$s in $1, \ldots, \min\{r, \lfloor d/2 \rfloor\}$ for which $\|\mathbf{y}_j\| > \sqrt{d}/8$, then this would imply that $\|\Delta\|_1 > r\sqrt{d}/32$. Thus, by the contrapositive, if $\|\Delta\|_1 \leq r\sqrt{d}/32$, then there must be at least $\min\{r, \lfloor d/2 \rfloor\} - r/4 + 1 \geq r/4$ $j$s for which $\|\mathbf{y}_j\| \leq \sqrt{d}/8$. Let us label them by $j_1 < \cdots < j_k \leq d/2$, where we now know that $k \geq r/4$. Then,

$$\mathbb{P}\left[\bigwedge_{\ell=1}^{k} \|\mathbf{y}_{j_\ell}\| \leq \frac{\sqrt{d}}{8}\right] = \prod_{\ell=1}^{k} \mathbb{P}\left[\|\mathbf{y}_{j_\ell}\| \leq \frac{\sqrt{d}}{8} \middle| \bigwedge_{m=1}^{\ell-1} \|\mathbf{y}_{j_m}\| \leq \frac{\sqrt{d}}{8}\right].$$

Now, recall that $\mathbf{y}_{j_\ell} = \Pi_{S_{j_\ell}} \mathbf{x}_{j_\ell}$. Since the entries of $\mathbf{x}_{j_\ell}$ are independent and sub-Gaussian with constant $K = 2$, we know from [RV13, Corollary 3.1], combined

with the complexification techniques outlined in the last paragraph of said paper, that there exists a constant $c' > 0$ such that

$$\mathbb{P}\left[\left\|\Pi_{S_{j_\ell}}\mathbf{x}_{j_\ell}\right\| \leq \frac{\sqrt{d}}{8}\right] \leq \mathbb{P}\left[\left|\left\|\Pi_{S_{j_\ell}}\mathbf{x}_{j_\ell}\right\| - \sqrt{d-j_\ell}\right| \leq \left|\sqrt{d-j_\ell} - \frac{\sqrt{d}}{8}\right|\right] \leq 2e^{-c'd}.$$

Moreover, since this bound holds for all subspaces $S_{j_\ell}$ of dimension $d - j_\ell$, we obtain

$$\mathbb{P}\left[\|\Delta\|_1 \leq \frac{r\sqrt{d}}{32}\right] \leq \prod_{\ell=1}^{k} \mathbb{P}\left[\left\|\Pi_{S_{j_\ell}}\mathbf{x}_{j_\ell}\right\| \leq \frac{\sqrt{d}}{8}\middle| \bigwedge_{m=1}^{\ell-1} \|\mathbf{y}_{j_m}\| \leq \frac{\sqrt{d}}{8}\right] \leq \left(2e^{-c'd}\right)^k.$$

Thus, whenever $d \geq \ln(2)/c' =: d_0$, we can choose $c = (c' - \ln(2)/d)/4$. Then, $c > 0$, and

$$\mathbb{P}\left[\|\Delta\|_1 \leq \frac{r\sqrt{d}}{32}\right] \leq \left(2e^{-c'd}\right)^k = \left(2e^{-\left(4c + \frac{\ln(2)}{d}\right)d}\right)^k = e^{-4cdk} \leq e^{-cdr},$$

where in the last step, we used that $k \geq r/4$. This completes the proof. □

Now, we are able to finish the lower bound proof for mixed-state tomography w.r.t. the trace norm. The proof strategy followed from here onward is very similar to the lower bound proof presented in Section 3.5.

**4.7.6.** Theorem. *Let $\varepsilon \in [0, 1/128]$, $d \in \mathbb{N}$, and $r \in [d]$. Suppose that we have a $Q$-query quantum algorithm that given access to an (inverse) state-preparation unitary for a purification of a $d \times d$ density matrix $\rho$ of rank at most $r$, outputs an approximation $\widetilde{\rho}$ such that $\|\widetilde{\rho} - \rho\|_1 \leq \varepsilon/128$, with probability at least $2/3$. Then $Q = \Omega(dr/\varepsilon)$.*

**Proof:**
Let $G$ be a bipartite graph with $2 \cdot 2^{rd}$ nodes, labeled by the bit strings $b \in \{0,1\}^{rd}$ on one side, and $\overline{b} \in \{0,1\}^{rd}$ on the other. Let there be an edge between $b$ and $\overline{b}$, if $\|\rho_b - \rho_{\overline{b}}\|_1 \leq \varepsilon/64$. From the previous lemma, we obtain that there exist constants $c, d_0 > 0$ such that whenever $d \geq d_0$, the number of edges $m$ in $G$ satisfies

$$m \leq 2^{2rd} \cdot e^{-crd}.$$

We abbreviate $f = e^{-crd}$, and observe that

$$\sum_{b \in \{0,1\}^{rd}} \deg(b) = m \leq f \cdot 2^{2rd},$$

where $\deg(b)$ denotes the degree of $b$ in $G$. Next, let $B = \{b \in \{0,1\}^{rd} : \deg(b) \geq 2^{rd}\sqrt{f}\}$, i.e., the set of nodes on the left side that have high degree. Then, by the pigeonhole principle, we obtain that $|B| \leq 2^{rd}\sqrt{f}$.

Let $b \in \{0,1\}^{rd} \setminus B$, and suppose that we have access to $b$ through the phase oracle

$$O_{b,\varepsilon} : |j\rangle \mapsto e^{i\varepsilon b_j} |j\rangle .$$

We now use our $Q$-query mixed-state tomography algorithm to construct a new algorithm that recovers $b$ with some very low probability.

The first step is to implement the unitary $U_b$ that maps

$$U_b : |0\rangle \mapsto \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} \left|\psi_b^{(j)}\right\rangle |j\rangle .$$

Using the same construction as in Equation (3.5.1), we can construct a circuit implementing this unitary $U_b$ with $K$ calls to $O_{b,\varepsilon}$, where $K = \Theta(1)$. Next, since this unitary $U_b$ prepares a purification of $\rho_b$, we can use $Q$ queries to it to obtain an estimate $\widetilde{\rho}$, such that $\|\widetilde{\rho} - \rho_b\|_1 \leq \varepsilon/128$, with probability at least $2/3$.

Next, suppose that $\overline{b} \in \{0,1\}^{rd}$ satisfies $\|\widetilde{\rho} - \rho_{\overline{b}}\|_1 \leq \varepsilon/128$. Then, by the triangle inequality, we have that $\|\rho_b - \rho_{\overline{b}}\|_1 \leq \|\rho_b - \widetilde{\rho}\|_1 + \|\widetilde{\rho} - \rho_{\overline{b}}\|_1 \leq \varepsilon/64$, and hence we find that $b$ and $\overline{b}$ are neighbors in $G$. Since we chose $b$ to be in $\{0,1\}^{rd} \setminus B$, we know that $\deg(b) \leq 2^{rd}\sqrt{f}$, and hence there are at most $2^{rd}\sqrt{f}$ choices for $\overline{b}$, among which is $b$ itself. Thus, if we uniformly choose one such $\overline{b}$, it will be equal to $b$ with probability at least $2/3 \cdot 2^{-rd} f^{-1/2}$.

The procedure above uses $KQ$ queries to $O_{b,\varepsilon}$, and recovers $b$ with probability at least $2/3 \cdot 2^{-rd} f^{-1/2}$. It is known that if we can solve this problem with $KQ$ queries to the fractional phase oracle $O_{b,\varepsilon}$, we can also solve it with at most $K'KQ$ queries to the regular phase oracle $O_b$, with $K' = \Theta(\varepsilon)$.[2] According to [FGG+99], Equation 4, this implies that

$$2^{rd} - |B| \leq \frac{3}{2} \cdot 2^{rd}\sqrt{f} \cdot \sum_{k=0}^{K'KQ} \binom{rd}{k} \leq \frac{3}{2} \cdot 2^{rd}\sqrt{f} \cdot 2^{rdH\left(\frac{K'KQ}{rd}\right)},$$

where $H(x) = -x\log(x) - (1-x)\log(1-x)$ is the binary entropy function, and the rightmost inequality can be found in several text books, e.g., [FG06], Lemma 16.19.

We can now plug everything into the above equation. Since $|B| \leq 2^{rd}\sqrt{f} \leq 2^{rd}e^{-1/2} \leq 2^{rd}/\sqrt{2}$, and hence we write

$$2^{rd + \log\left(1 - \frac{1}{\sqrt{2}}\right)} \leq 2^{\log(3) - 1 + rd - crd\log(e) + rdH\left(\frac{K'KQ}{dr}\right)}.$$

Comparing the exponents we obtain

$$\log(3) - 1 - \log\left(1 - \frac{1}{\sqrt{2}}\right) + rd\left(-c\log(e) + H\left(\frac{K'KQ}{dr}\right)\right) \geq 0,$$

[2]The argument for this is sketched in the proof of Lemma 3.5.1.

and thus $H(\frac{K'KQ}{dr}) = \Omega(1)$. Since the binary entropy function is monotonously increasing from 0 to 1 in the interval $[0, 1/2]$, we find that $K'KQ = \Omega(dr)$, and hence $Q = \Omega(rd/\varepsilon)$. □

We can now derive the lower bounds for all other Schatten norms too.

**4.7.7.** Theorem. *Let $\varepsilon \in [0, 1/128]$, $d \in \mathbb{N}$, $r \in [d]$ and $q \in [1, \infty]$. Suppose that we have a quantum algorithm that estimates a density matrix $\rho$ up to precision $\varepsilon$ in Schatten $q$-norm, using $Q$ (inverse) queries to a unitary preparing its purification. Then,*

$$Q = \Omega\left(\min\left\{\frac{dr^{\frac{1}{q}}}{\varepsilon}, \frac{d}{\varepsilon^{\frac{1}{1-\frac{1}{q}}}}\right\}\right).$$

**Proof:**
First, suppose that $\varepsilon \leq 1/(256r^{1-1/q})$. Then, using Hölder's inequality, we can obtain a $r^{1-1/q}\varepsilon$-precise approximation of $\rho$ in trace norm in $Q$ queries. Since $r^{1-1/q}\varepsilon \leq 1/128$, using Theorem 4.7.6 we find that

$$Q = \Omega\left(\frac{dr}{r^{1-1/q}\varepsilon}\right) = \Omega\left(\frac{dr^{\frac{1}{q}}}{\varepsilon}\right).$$

On the other hand, if $1/(128r^{1-1/q}) < \varepsilon \leq 1/128$, then we can choose an integer $1 \leq r' < r$, such that $1/(256(r')^{1-1/q}) < \varepsilon \leq 1/(128(r')^{1-1/q})$. By the previous argument for the case $\varepsilon \leq 1/(256r^{1-1/q})$, we can now use $Q$ queries to obtain a $2\varepsilon$-precise Schatten $q$-approximation of any density matrix of rank at most $r'$, using our algorithm for density matrices of rank $r$. We already know that this takes $\Omega(d(r')^{1/q}/\varepsilon)$ queries, and since $r' = \Theta(1/\varepsilon^{1/(1-1/q)})$, we obtain that

$$Q = \Omega\left(\frac{d(r')^{\frac{1}{q}}}{\varepsilon}\right) = \Omega\left(\frac{d}{\varepsilon^{1+\frac{1/q}{1-1/q}}}\right) = \Omega\left(\frac{d}{\varepsilon^{\frac{1}{1-\frac{1}{q}}}}\right).$$

Finally, observe that $d/\varepsilon^{1/(1-1/q)} < dr^{1/q}/\varepsilon$ is equivalent to $\varepsilon^{1/(q-1)} > 1/r^{1/q}$, and hence to $\varepsilon > 1/r^{1-1/q}$, so indeed the minimum picks out the right branch of the lower bound. This completes the proof. □

Note that this exactly matches the complexity that we obtained in Algorithm 4.6.4. Thus, up to polylogarithmic factors, we have completely characterized the query complexity of state tomography in this model.

## 4.8 Implications

Over the course of the previous sections, we developed a mixed-state tomography algorithm, and proved its optimality up to polylogarithmic factors. In particular,

we conclude from Algorithm 4.6.4 and Theorem 4.7.7 that in order to estimate a density operator $\rho \in \mathbb{C}^{d\times d}$ of rank at most $r$ up to precision $\varepsilon > 0$ in Schatten $q$-norm, with $q \in [1,\infty]$, requires

$$\widetilde{\Theta}\left(\min\left\{\frac{dr^{\frac{1}{q}}}{\varepsilon}, \frac{d}{\varepsilon^{\frac{1}{1-\frac{1}{q}}}}\right\}\right)$$

(inverse, controlled) queries to a unitary that prepares a purification of $\rho$.

In this section, we take a look at several immediate implications of this result. We start with the problem of pure-state tomography, which is the special case of the mixed-state tomography problem where we know ahead of time that the rank is 1. In the pure case, we have the following well-known connection between the Euclidean distance between two state vectors, and the operator distance between the corresponding density matrices.

**4.8.1.** Lemma (Connection between operator distance and Euclidean distance). *Let $|\psi\rangle, |\phi\rangle \in \mathcal{H}$ be quantum states. Then,*

$$\frac{1}{\sqrt{2}} \min_{\chi\in\mathbb{R}} \left\| |\psi\rangle - e^{i\chi} |\phi\rangle \right\| \leq \left\| |\psi\rangle\langle\psi| - |\phi\rangle\langle\phi| \right\|_\infty = \sqrt{1 - |\langle\psi|\phi\rangle|^2} \leq \left\| |\psi\rangle - |\phi\rangle \right\|.$$

**Proof:**
The statement is trivial for $|\psi\rangle = |\phi\rangle$, so suppose that $|\psi\rangle \neq |\phi\rangle$. We start by proving the inner equality. To that end, if $a|\psi\rangle + b|\phi\rangle$ is an eigenvector of this difference operator with eigenvalue $\lambda$, then

$$\lambda \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1 & -\langle\psi|\phi\rangle \\ \langle\phi|\psi\rangle & -1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}.$$

Thus, we find that the eigenvalues of $|\psi\rangle\langle\psi| - |\phi\rangle\langle\phi|$ satisfy $\lambda^2 - 1 + |\langle\psi|\phi\rangle|^2 = 0$, from which we obtain that $|\lambda| = \sqrt{1 - |\langle\psi|\phi\rangle|^2}$. This proves the equality.

For the remaining inequalities, observe that for all $\chi \in \mathbb{R}$,

$$\left\| |\psi\rangle - e^{i\chi}|\phi\rangle \right\|^2 = \langle\psi|\psi\rangle + \langle\phi|\phi\rangle - e^{i\chi}\langle\psi|\phi\rangle - e^{-i\chi}\langle\phi|\psi\rangle = 2 - 2\mathrm{Re}\left[e^{i\chi}\langle\psi|\phi\rangle\right].$$

Thus, the choice of $\chi$ that minimizes the above expression is the one for which $e^{i\chi}\langle\psi|\phi\rangle \geq 0$, in which case

$$\begin{aligned} \left\| |\psi\rangle - e^{i\chi}|\psi\rangle \right\|^2 &= 2(1 - |\langle\psi|\phi\rangle|) \leq 2(1 - |\langle\psi|\phi\rangle|)(1 + |\langle\psi|\phi\rangle|) \\ &= 2(1 - |\langle\psi|\phi\rangle|^2) = 2\left\| |\psi\rangle\langle\psi| - |\phi\rangle\langle\phi| \right\|_\infty^2. \end{aligned}$$

On the other hand, we have

$$1 - |\langle\psi|\phi\rangle|^2 = (1 - |\langle\psi|\phi\rangle|)(1 + |\langle\psi|\phi\rangle|) \leq 2(1 - \mathrm{Re}\langle\psi|\phi\rangle) = \left\| |\psi\rangle - |\phi\rangle \right\|^2.$$

This completes the proof. □

The above lemma tells us how we can use our mixed-state tomography algorithm to perform pure state tomography. Indeed, if we want to approximate a state $|\psi\rangle$ up to global phase in Euclidean norm with precision $\varepsilon$, the inequality on the left in Lemma 4.8.1 tells us that it suffices to approximate the density operator $|\psi\rangle\langle\psi|$ with another rank-1 density matrix up to operator norm distance $\varepsilon/\sqrt{2}$. On the other hand, the inequality on the right in Lemma 4.8.1 tells us that this is tight. Thus, we conclude that estimating a pure state up to Euclidean distance $\varepsilon$ costs $\widetilde{\Theta}(d/\varepsilon)$ calls to its state-preparation oracle.

Another interesting problem to consider is classical probability distribution reconstruction. Suppose that we have a $d$-dimensional probability distribution $p \in \Delta_d = \{p \in \mathbb{R}^d_{\geq 0} : \sum_{j=1}^d p_j = 1\}$, and that we are given access to it through a quantum circuit $U$ that implements the mapping

$$U : |0\rangle \mapsto \sum_{j=1}^{d} \sqrt{p_j}\,|j\rangle =: |\psi\rangle\,. \tag{4.8.1}$$

How many (inverse, controlled) calls to $U$ suffice to find an approximate distribution $\widetilde{p} \in \Delta_d$ such that $\|p - \widetilde{p}\|_1 \leq \varepsilon$? We can also readily answer this question due to the following lemma.

**4.8.2.** Lemma. *Let $p, \widetilde{p} \in \Delta_d$, and let*

$$|\psi\rangle = \sum_{j=1}^{d} \sqrt{p_j}\,|j\rangle\,, \qquad \textit{and} \qquad |\psi'\rangle = \sum_{j=1}^{d} \sqrt{\widetilde{p}_j}\,|j\rangle\,.$$

*Then,*

$$\|p - \widetilde{p}\|_1 \leq 2\,\||\psi\rangle - |\psi'\rangle\|\,.$$

**Proof:**
We use the Cauchy–Schwarz inequality to obtain

$$\begin{aligned}
\|p - \widetilde{p}\|_1 &= \sum_{j=1}^{d} |p_j - \widetilde{p}_j| = \sum_{j=1}^{d} |\sqrt{p_j} - \sqrt{\widetilde{p}_j}| \cdot (\sqrt{p_j} + \sqrt{\widetilde{p}_j}) \\
&= \sum_{j=1}^{d} |\sqrt{p_j} - \sqrt{\widetilde{p}_j}| \cdot \sqrt{p_j} + \sum_{j=1}^{d} |\sqrt{p_j} - \sqrt{\widetilde{p}_j}| \cdot \sqrt{\widetilde{p}_j} \\
&\leq \sqrt{\sum_{j=1}^{d} |\sqrt{p_j} - \sqrt{\widetilde{p}_j}|^2} \cdot \sqrt{\sum_{j=1}^{d} p_j} + \sqrt{\sum_{j=1}^{d} |\sqrt{p_j} - \sqrt{\widetilde{p}_j}|^2} \cdot \sqrt{\sum_{j=1}^{d} \widetilde{p}_j} \\
&= 2\,\||\psi\rangle - |\psi'\rangle\|\,.
\end{aligned}$$

This completes the proof. □

Hence, we can solve the probability distribution reconstruction problem using our techniques as well. We can use $\widetilde{\mathcal{O}}(d/\varepsilon)$ calls to the state-preparation unitary to obtain an estimate $|\psi'\rangle$ of $|\psi\rangle$, defined in Equation (4.8.1), with Euclidean distance $\varepsilon/4$. Then, we find the probability distribution $\widetilde{p} \in \Delta_d$ such that its quantum state approximates $|\psi'\rangle$ as well as possible, and then we find by Lemma 4.8.2 that $\|\widetilde{p} - p\|_1$ is at most $\varepsilon$. Thus, recovering a probability distribution up to $\ell_1$-norm $\varepsilon$ can be done with $\widetilde{\mathcal{O}}(d/\varepsilon)$ queries to the state-preparation oracle as well. This recovers a result from [vApe21].

Using simplified version of the techniques employed in this chapter, we can also consider the above questions in more general $\ell_p$-norms. More details about the results in these settings can be found in [vACG+22]. We mention here that approximating a probability distribution $p \in \Delta_d$ with precision $\varepsilon > 0$ in $\ell_p$-norm, with $p \in [1, \infty]$, costs

$$\widetilde{\Theta}\left(\min\left\{\frac{d^{\frac{1}{p}}}{\varepsilon}, \frac{1}{\varepsilon^{\frac{1}{1-\frac{1}{p}}}}\right\}\right)$$

(inverse, controlled) queries to the state-preparation unitary defined in Equation (4.8.1). The upper bound proof can be found by combining the algorithm from [vApe21] with Lemma 4.3.2. The lower bound follows by combining Lemmas 48 and 53 from [vACG+22]. This result generalizes the construction in [vApe21], and the special case where $p = 2$ resolves an open question from that paper.

We conclude this chapter by remarking that the state-tomography problem seems to be completely resolved up to polylogarithmic factors. The most natural follow-up question is whether these polylogarithmic factors can be removed. After all, the lower bounding technique yields "nice" complexities without any trailing logarithmic factors, so it would be very elegant if there exists an algorithm that could meet these lower bound results up to constants. In the algorithmic construction that we presented in this chapter, though, we obtain many logarithmic factors in many different places, so such an algorithm would at the very least have to look quite different from the approach displayed here. We leave this question for future research.

# Chapter 5

# Partition function estimation

In this chapter, we develop a quantum algorithm that estimates partition functions using a number of Gibbs state reflections that is sub-linear in the logarithm of the size of the state space.

Partition functions play a key role in statistical mechanics, and as such being able to estimate them helps understanding the behavior of physical systems governed by the laws of statistical physics. Additionally, several computational problems can be embedded in partition functions, and consequently our algorithm provides a new state-of-the-art solution to these problems.

Along the way, we come up with several new algorithmic techniques. First, we come up with a new way of removing the bias from the phase estimation routine, which is different from the technique introduced in Section 4.4. We also construct an unbiased, low-variance amplitude estimation routine, which has the remarkable property of being *non-destructive*, i.e., it restores the initial state at the end of the procedure.

This chapter is based on [CH22]. We start with an introduction to the topic of partition functions in Section 5.1. Then, we derive some new algorithmic techniques, in Section 5.2. After that, we show how these subroutines can be used to perform unbiased and non-destructive mean estimation in Section 5.3. And finally, we tie everything together and show how to compute partition functions, in Section 5.4.

## 5.1 Introduction

The Boltzmann–Gibbs distribution is a paradigmatic tool for modeling systems that obey the principle of maximum entropy. It arises in several fields of research such as statistical mechanics [Geo11; FV17; Sin82], economic modeling [DY00], image processing [GG84; Bes86], statistical learning theory [Cat04], etc. The probability assigned by the Gibbs distribution to each possible configuration of a system is inversely proportional to the exponential of its energy multiplied by the

inverse temperature. Mathematically, for a classical Hamiltonian of degree $n$, i.e., a function $H : \Omega \to \{0, \ldots, n\}$, specifying the energy level of each configuration $x \in \Omega$, the Gibbs distribution at inverse temperature $\beta$ is given by $\pi_\beta(x) = \frac{1}{Z(\beta)} e^{-\beta H(x)}$ where the normalization factor

$$Z(\beta) = \sum_{x \in \Omega} e^{-\beta H(x)} \tag{5.1.1}$$

is called the *partition function*. While it is often straightforward to evaluate the partition function at high temperature (when $\beta = 0$, it is just the number of possible configurations), the low-temperature regime captures ground state properties that are challenging to compute. For instance, $Z(\infty)$ can represent the cardinalities of exponentially large combinatorial structures (such as the number of colorings of a graph [Jer95; SVV09], the volume of convex bodies [DFK91; DF91] or the permanent of non-negative matrices [JSV04]) which are generally #P-hard to compute exactly [Val79; DF88; JS93].

The standard approach for evaluating partition functions at low temperature is to resort to Markov chain Monte Carlo methods [JS96]. A celebrated line of works [VC72; JVV86; DF91; BSV+08; SVV09] has shown how to turn the ability of efficiently *sampling* from the Gibbs distribution into that of efficiently *approximating* the partition function. At a high level, these works rely on the same two-stage simulated annealing algorithm. First, they compute a short *cooling schedule*, which is an increasing sequence of inverse temperatures $0 = \beta_1 < \cdots < \beta_\ell = \infty$ with limited fluctuations in Gibbs distributions between two consecutive values. Next, the partition function at low temperature is expressed as a telescoping product

$$Z(\infty) = Z(0) \prod_{i=0}^{\ell-1} \frac{Z(\beta_{i+1})}{Z(\beta_i)} \tag{5.1.2}$$

which is approximated by using a suitable *product estimator*.

As an example, the seminal algorithm of Štefankovič, Vempala and Vigoda [SVV09] generates a so-called "Chebyshev cooling schedule" with schedule length $\ell = \widetilde{\mathcal{O}}(\sqrt{\log |\Omega|})$ (ignoring logarithmic dependences on the degree $n$, which is often on the order of $n \sim \log |\Omega|$) where each ratio $Z(\beta_{i+1})/Z(\beta_i)$ is expressed as the expectation value of a random variable $X_i$ with bounded relative second moment $\mathbb{E}[X_i^2] = \mathcal{O}(\mathbb{E}[X_i]^2)$. Such schedules are known to admit a product estimator that requires $\mathcal{O}(\ell^2/\varepsilon^2)$ classical Gibbs samples to estimate $Z(\infty) = Z(0) \cdot \mathbb{E}[X_1] \cdots \mathbb{E}[X_\ell]$ with relative error $\varepsilon$. Thus, if we let $\delta$ denote the spectral gap of a (ergodic reversible) Markov chain generating samples from the considered Gibbs distributions, the overall cost of the algorithm presented in [SVV09] is $\widetilde{\mathcal{O}}(\log |\Omega|/(\varepsilon^2 \delta))$.

The theory of quantum algorithms provides several directions for accelerating the computation of partition functions. Quantum Markov chains [Sze04;

MNR+11] can prepare coherent "qsample" encodings $|\pi_\beta\rangle = \sum_x \sqrt{\pi_\beta(x)}\,|x\rangle$ of the Gibbs distribution with a quadratic improvement in spectral gap for the rate of convergence (but an increase dependence on other parameters). Quantum phase estimation [Kit96] and amplitude estimation [BHM+02] lead to quadratically better convergence rates for estimating expectation values [Ter99; AW99; Hei02; BDG+11; Mon15; HM19; Ham21]. Yet, while this may hint at the existence of an $\widetilde{\mathcal{O}}(\sqrt{\log|\Omega|}/(\varepsilon\sqrt{\delta}))$ quantum algorithm for estimating partition functions, the best-known algorithms [HW20; AHN+21] still require a linear scaling $\widetilde{\mathcal{O}}(\log|\Omega|/(\varepsilon\sqrt{\delta}))$ in the logarithm of the size of the state space. This bottleneck is due to additional challenges posed by current quantum algorithmic techniques. It is for instance significantly harder to prepare the qsample $|\pi_\beta\rangle$ (at low temperature) than to implement the reflection $I - 2\,|\pi_\beta\rangle\langle\pi_\beta|$ through it. This obstacle requires using *non-destructive* procedures [MW05; WA08; WCN+09; TOV+11; ORR13; HW20] to recycle the same qsamples all along the algorithm, and to rely mostly on the reflection operator. Another fundamental limitation faced by current best quantum mean estimators [Mon15; HM19; Ham21] is the presence of *biases* in the estimates that degrade the convergence guarantee of the product estimators.

Our main contribution is to develop the first quantum algorithm for approximating Gibbs partition functions with a complexity scaling *sublinearly* with respect to the logarithm of the size of the state space. More precisely, we prove the next theorem in Section 5.4.

**5.1.1.** Theorem (Informal statement of Theorem 5.4.1).
*There is a quantum algorithm that, given a Gibbs distribution generated by a Markov chain with spectral gap $\delta$, computes an estimate $\widetilde{Z}$ of the partition function at zero temperature that satisfies $|\widetilde{Z} - Z(\infty)| \leq \varepsilon Z(\infty)$ using*

$$\widetilde{\mathcal{O}}\left(\log^{3/4}(|\Omega|)\log^{3/2}(n)/(\varepsilon\sqrt{\delta})\right)$$

*steps of the quantum walk operator.*

Our result reduces the polynomial dependence on $\log|\Omega|$ by a factor of $1/4$ and it achieves state-of-the-art dependence on the spectral gap $\delta$ and the accuracy $\varepsilon$ up to logarithmic factors. We provide a comparison with prior work in Table 5.1.

## 5.2 Modified quantum subroutines

Our starting point is the amplitude estimation algorithm, as first introduced in [BHM+02] and restated in Algorithm 2.4.4 for convenience. We note that the amplitude estimation algorithm can naturally be used to estimate probabilities, by simply squaring the output of the algorithm.

| Source | Schedule generation | Mean-value estimation | **Total cost** |
|---|---|---|---|
| [DF91; BSV+08] | 0 (non-adaptive) | $\widetilde{\mathcal{O}}(k^2/(\varepsilon^2\delta))$ | $\widetilde{\mathcal{O}}(k^2/(\varepsilon^2\delta))$ |
| [SVV09; Hub15; Kol18] | $\widetilde{\mathcal{O}}(k/\delta)$ | $\widetilde{\mathcal{O}}(k/(\varepsilon^2\delta))$ | $\widetilde{\mathcal{O}}(k/(\varepsilon^2\delta))$ |
| [WCN+09] | Uses [BSV+08] | $\widetilde{\mathcal{O}}(k^2/(\varepsilon\sqrt{\delta}))$ | $\widetilde{\mathcal{O}}(k^2/(\varepsilon\sqrt{\delta}))$ |
| [Mon15] | Uses [SVV09] | $\widetilde{\mathcal{O}}(k/(\varepsilon\sqrt{\delta}))$ | $\widetilde{\mathcal{O}}(k(1/\delta + 1/\varepsilon\sqrt{\delta}))$ |
| [HW20; AHN+21] | $\widetilde{\mathcal{O}}(\sqrt{\log\lvert\Omega\rvert/\delta})$ | Uses [Mon15] | $\widetilde{\mathcal{O}}(k/(\varepsilon\sqrt{\delta}))$ |
| **This chapter** | Uses [HW20; AHN+21] | $\widetilde{\mathcal{O}}(k^{3/4}/(\varepsilon\sqrt{\delta}))$ | $\widetilde{\mathcal{O}}(k^{3/4}/(\varepsilon\sqrt{\delta}))$ |

Table 5.1: Comparison of the complexity (in terms of Markov chain steps) needed to compute partition functions over a state space $\Omega$, where $\delta$ is the spectral gap of the Markov chain and $\varepsilon$ is the accuracy parameter. We use the shorthand notation $k = \log|\Omega|$, and we omit polylogarithmic dependencies on the degree $n$ of the partition function, which is typically on the order of $n \sim \log|\Omega|$, as well as $k$, $1/\varepsilon$ and $1/\delta$. The first two rows are classical algorithms only.

In the previous chapter, we already saw how we can modify the approach from [BHM+02] to make the probability estimation procedure unbiased, which resulted in Algorithm 4.4.3. The core idea, introduced for the first time in [LdW21], is to add a random phase to the phase estimation subroutine, and subsequently argue by symmetry that any bias is automatically removed.

In this section, we modify Algorithm 4.4.3 in such a way that it has the following three properties:

1. *Unbiased*: The difference between the expectation of the subroutine's outcome and the estimated quantity can be exponentially suppressed with polynomial overhead.
2. *Low-variance*: The variance of the subroutine's outcome is of the same order as the size of the interval of high-concentration.
3. *Non-destructive*: The subroutine restores the initial state at the end of its execution.

We proceed by giving the algorithm and proving its properties.

**Algorithm 5.2.1:** Unbiased, low-variance, non-destructive probability estimation

**Input:**

1: $k \in \mathbb{N}$: the precision parameter.

2: $k' \geq 9$: the boosting parameter.

3: $R_{|\psi\rangle}$: a quantum circuit that reflects through

$$|\psi\rangle = \sqrt{p}\,|\psi_1\rangle\,|1\rangle + \sqrt{1-p}\,|\psi_0\rangle\,|0\rangle \in \mathcal{H} \otimes \mathbb{C}^2,$$

i.e., it acts on $\mathcal{H} \otimes \mathbb{C}^2$ and performs the operation $2\,|\psi\rangle\langle\psi| - I$.

4: A single copy of the state $|\psi\rangle$.

**Derived objects:**

1: $\theta = \arcsin(\sqrt{p})$.
2: $C_{k,k'}$ as in Algorithm 4.4.2.
3: $G = R_{|\psi\rangle}(I \otimes (2\,|1\rangle\langle 1| - I))$.

**Output:**

1: A random variable $\widetilde{p}$ that satisfies $\mathbb{E}[\widetilde{p}] = p$,

$$\mathrm{Var}[\widetilde{p}] = \mathcal{O}(p(1-p)/2^{2k} + 1/2^{4k}). \tag{5.2.1}$$

and with $\overline{p} = 1/2 - C_{k,k'}(1/2 - \widetilde{p})$, we have for all $m \geq 6$,

$$\mathbb{P}\left[|\overline{p} - p| > \frac{4\pi^2\sqrt{p(1-p)}m}{2^k} + \left(\frac{2\pi m}{2^k}\right)^2\right] \leq \left(\frac{6}{m}\right)^{k'/2}, \tag{5.2.2}$$

2: A copy of the state $|\psi\rangle$.

**Queries:** Expected number of queries to $R_{|\psi\rangle}$: $\mathcal{O}(2^k - 1)$.

**Procedure:** PROB-EST($k$, $k'$, $R_{|\psi\rangle}$, $|\psi\rangle$):

1: Repeat

1. Let $\phi =$ UNBIASED-BOOSTED-PHASE-EST($k$, $k'$, $G$, $\cdot$), run without the state-preparation routine, but on the state $|\psi\rangle$ directly instead.
2. Set $\widetilde{p} = \frac{1}{2} - \frac{\cos(2\pi\phi)}{2C_{k,k'}}$.
3. Measure the observable $|\psi\rangle\langle\psi|$ and denote the result by $b \in \{0, 1\}$.

Until the measurement outcome $b = 1$.

2: Return the $\widetilde{p}$ that was obtained in the first iteration.

---

**Proof of the properties of Algorithm 5.2.1:**

First, we observe from the properties of Algorithm 4.4.2 that we perform $\mathcal{O}(2^k - 1)$ calls to $R_{|\psi\rangle}$, every time we execute step 1. Thus, in order to verify the claimed number of queries, it suffices to compute the expected number of iterations that we perform in step 1.

To that end, it is important to notice that the Grover iterate $G$ acts on a two-dimensional space containing $|\psi\rangle$, i.e., there exists another vector $\left|\psi^\perp\right\rangle$ such that the state vector remains in the space $\mathrm{Span}\{|\psi\rangle, \left|\psi^\perp\right\rangle\}$. Hence, if we end up with a measurement outcome 0 at the end of step 1.3, then we know that we are in the state $\left|\psi^\perp\right\rangle$.

The crucial observation, now, is that since the states $|\psi\rangle$ and $\left|\psi^\perp\right\rangle$ are orthogonal, the resulting states we obtain right before the measurement in step 1.3 when we start with $|\psi\rangle$ or $\left|\psi^\perp\right\rangle$ are orthogonal as well. Thus, if we start in $|\psi\rangle$ at step 1.1, and then get output 1 with some probability $q \in [0, 1]$ in step 1.3, then we would get output 1 with probability $1 - q$ if we started in the state $\left|\psi^\perp\right\rangle$ instead. In other words, the algorithm remains in the state $|\psi\rangle$ or $\left|\psi^\perp\right\rangle$ with

probability $q$, and transitions from one state to the other with probability $1-q$. This phenomenon first appeared in [MW05].[1]

Thus, given this value $q$, we either stop directly after the first iteration, which happens with probability $q$, or we stop after $t \geq 2$ iterations if we transition from $|\psi\rangle$ to $|\psi^\perp\rangle$ in the first iteration, then stay in $|\psi^\perp\rangle$ for $t-2$ iterations, and then transition back to $|\psi\rangle$ in the $t$th iteration, which happens with probability $(1-q)^2 q^{t-2}$. In other words, we can express the probability distribution of $T$, i.e., the number of iterations that we make, by

$$\mathbb{P}[T=t] = \begin{cases} q, & \text{if } t=1, \\ (1-q)^2 q^{t-2}, & \text{if } t>1. \end{cases}$$

Next, we compute the expectation of $T$ to obtain

$$\begin{aligned} \mathbb{E}[T] &= 1 + \mathbb{E}[T-1] = 1 + \sum_{t=2}^{\infty} (t-1)\mathbb{P}[T=t] = 1 + (1-q)^2 \sum_{t=2}^{\infty} (t-1) q^{t-2} \\ &= 1 + (1-q)^2 \frac{\mathrm{d}}{\mathrm{d}q}\left[\sum_{t=2}^{\infty} q^{t-1}\right] = 1 + (1-q)^2 \frac{\mathrm{d}}{\mathrm{d}q}\left[\frac{q}{1-q}\right] \\ &= 1 + (1-q)^2 \frac{\mathrm{d}}{\mathrm{d}q}\left[\frac{1}{1-q} - 1\right] = 1 + \frac{(1-q)^2}{(1-q)^2} = 2. \end{aligned}$$

This completes the proof of the number of queries. It also proves that we get back to the state $|\psi\rangle$ almost surely.

Hence, it remains to check the claimed properties of the output of the algorithm. To that end, we observe from the properties of Algorithm 4.4.2 that $\mathbb{E}[\widetilde{p}] = p$. Furthermore, recall from Equation (4.4.2) that

$$\mathbb{P}\left[|\overline{p} - p| > \frac{4\pi^2\sqrt{p(1-p)}m}{2^k} + \left(\frac{2\pi m}{2^k}\right)^2\right] \leq \left(\frac{6}{m}\right)^{k'/2},$$

with $\overline{p} = 1/2 - C_{k,k'}(1/2 - \widetilde{p})$. Thus, we find

$$\widetilde{p} = \frac{1}{2} - \frac{1}{C_{k,k'}}\left(\frac{1}{2} - \overline{p}\right),$$

and so $\mathrm{Var}[\widetilde{p}] = \mathrm{Var}[\overline{p}]/C_{k,k'}^2$. Finally, we observe that

$$\mathrm{Var}[\overline{p}] \leq \mathbb{E}[(\overline{p}-p)^2] = \int_0^\infty 2t\mathbb{P}\left[|\overline{p}-p| \geq t\right] \, \mathrm{d}t.$$

[1] The idea of using the techniques from [MW05] in the amplitude estimation setting was also coined in [HW20]. However, the authors seem to assume that $q = 1/2$, which we could not find any justification for. Here, we provide the missing details.

In the case where $\sqrt{p(1-p)} \geq 6\pi/2^k$, we analyze the resulting integral by splitting it up in three regimes. First, if $t \geq 8p(1-p)$, then let $m = 2^k\sqrt{t/2}/(2\pi) \geq 2^k\sqrt{p(1-p)}/\pi$. We observe that $|\overline{p}-p| \geq t$ implies

$$\begin{aligned}|\overline{p}-p| \geq t = 2\left(\frac{2\pi m}{2^k}\right)^2 &\geq \frac{2\pi m}{2^k}\cdot\frac{2\pi\cdot 2^k\sqrt{p(1-p)}/\pi}{2^k} + \left(\frac{2\pi m}{2^k}\right)^2 \\ &= \frac{4\pi m\sqrt{p(1-p)}}{2^k} + \left(\frac{2\pi m}{2^k}\right)^2,\end{aligned}$$

and hence in this regime we have

$$\mathbb{P}\left[|\overline{p}-p| > t\right] \leq \left(\frac{6\cdot 2\pi}{2^k\sqrt{t/2}}\right)^{k'/2} = \left(\frac{12\sqrt{2}\pi}{2^k}\right)^{k'/2}\cdot t^{-k'/4}.$$

Similarly, if $48\pi\sqrt{p(1-p)}/2^k < t \leq 8p(1-p)$, then we assign the value $m = 2^k t/(8\pi\sqrt{p(1-p)}) \leq 2^k\sqrt{p(1-p)}/\pi$. Similarly as before, we observe that $|\overline{p}-p| \geq t$ implies that

$$\begin{aligned}|\overline{p}-p| \geq t &= \frac{8\pi m\sqrt{p(1-p)}}{2^k} \\ &\geq \frac{4\pi m\sqrt{p(1-p)}}{2^k} + \frac{4\pi m\sqrt{p(1-p)}}{2^k}\cdot\frac{m}{2^k\sqrt{p(1-p)}/\pi} \\ &= \frac{4\pi m\sqrt{p(1-p)}}{2^k} + \left(\frac{2\pi m}{2^k}\right)^2,\end{aligned}$$

and hence in this regime, we have

$$\mathbb{P}\left[|\overline{p}-p| > t\right] \leq \left(\frac{6\cdot 8\pi\sqrt{p(1-p)}}{2^k t}\right)^{k'/2} = \left(\frac{48\pi\sqrt{p(1-p)}}{2^k}\right)^{k'/2}\cdot t^{-k'/2}.$$

Finally, if $t \leq 48\pi\sqrt{p(1-p)}/2^k$, we just use the trivial upper bound that $\mathbb{P}[|\overline{p}-p| > t] \leq 1$. Putting everything together yields,

$$\begin{aligned}\mathrm{Var}[\overline{p}] &\leq \int_0^{48\pi\sqrt{p(1-p)}/2^k} 2t\,\mathrm{d}t + \left(\frac{48\pi\sqrt{p(1-p)}}{2^k}\right)^{k'/2}\int_{48\pi\sqrt{p(1-p)}/2^k}^{8p(1-p)} 2t\cdot t^{-k'/2}\,\mathrm{d}t \\ &\quad + \left(\frac{12\sqrt{2}\pi}{2^k}\right)^{k'/2}\int_{8p(1-p)}^{\infty} 2t\cdot t^{-k'/4} \\ &= \left[t^2\right]_0^{48\pi\sqrt{p(1-p)}/2^k} + \left(\frac{48\pi\sqrt{p(1-p)}}{2^k}\right)^{k'/2}\left[-\frac{2t^{-k'/2+2}}{k'/2-2}\right]_{48\pi\sqrt{p(1-p)}/2^k}^{8p(1-p)} \\ &\quad + \left(\frac{12\sqrt{2}\pi}{2^k}\right)^{k'/2}\left[-\frac{t^{-k'/4+2}}{k'/4-2}\right]_{8p(1-p)}^{\infty} = \mathcal{O}\left(\frac{p(1-p)}{2^{2k}}\right).\end{aligned}$$

On the other hand, if $\sqrt{p(1-p)} < 6\pi/2^k$, then the middle term in the above integral drops out, and we can resort to using the trivial bound in the regime where $t \leq 288\pi/2^{2k}$. Thus,

$$\begin{aligned}\mathrm{Var}[\overline{p}] &\leq \int_0^{288\pi/2^{2k}} 2t \, \mathrm{d}t + \left(\frac{12\sqrt{2}\pi}{2^k}\right)^{k'/2} \int_{288\pi/2^{2k}}^{\infty} 2t \cdot t^{-k'/4} \, \mathrm{d}t \\ &\leq \left[t^2\right]_0^{288\pi/2^{2k}} + \left(\frac{12\sqrt{2}\pi}{2^k}\right)^{k'/2} \left[-\frac{2t^{-k'/4+2}}{k'/4-2}\right]_{288\pi/2^{2k}}^{\infty} \\ &= \mathcal{O}\left(\frac{1}{2^{2k}}\right).\end{aligned}$$

Finally, recall that $\mathrm{Var}[\widetilde{p}] = \mathcal{O}(\mathrm{Var}[\overline{p}])$, and so $\mathrm{Var}[\widetilde{p}]$ is $\mathcal{O}(p(1-p)/2^{2k} + 1/2^{4k})$. This completes the proof. □

We have now constructed an *unbiased, low-variance, non-destructive* variant of probability estimation, achieving the same precision as the well-known algorithm of [BHM+02], as well as achieving the same query complexity, albeit only in expectation. Since the amplitude estimation routine derived in [BHM+02] is very frequently used in other quantum algorithms, we expect that there are settings where convenient immediate improvements can be obtained by simply substituting our result in a black-box way.

The second term in the variance of the resulting probability estimator, as in Equation (5.2.1), is a bit mysterious. After all, in the classical case, if we toss a biased coin $n$ times, we can estimate its success probability $p$ unbiasedly, with an estimator that has variance $p(1-p)/n$. Hence, in order to get a proper quadratic improvement in sample complexity over the classical case, we would expect a quantum routine obtaining variance $p(1-p)/n^2$. In Equation (5.2.1), we can see that we are almost there, but the second term makes the estimator perform slightly worse in the regime where $p$ is either very close to 0 or very close to 1.

Below, we showcase a routine that circumvents this problem by running a two-stage algorithm. First, we run the above algorithm to get a good estimate of $p$, and if it turns out to be very close to 0 or 1, then we first use linear amplitude amplification, Algorithm 2.4.6, to get out of this regime. The downside of this approach is that we lose exact unbiasedness, and can only ensure a very small bias instead.

**Algorithm 5.2.2:** Low-bias, low-variance, non-destructive probability estimation

**Input:**

1: $t \in \mathbb{N}$: the precision parameter.
2: $\delta > 0$: the bias tolerance parameter.

3: $R_{|\psi\rangle}$: a quantum circuit that reflects through

$$|\psi\rangle = \sqrt{p}\,|\psi_1\rangle\,|1\rangle + \sqrt{1-p}\,|\psi_0\rangle\,|0\rangle \in \mathcal{H} \otimes \mathbb{C}^2,$$

i.e., it acts on $\mathcal{H} \otimes \mathbb{C}^2$ and performs the operation $2\,|\psi\rangle\langle\psi| - I$.

4: A single copy of the state $|\psi\rangle$.

**Derived objects:**

1: $k = \lceil \log(48\pi t) \rceil$.

2: $k' = \lceil 2\log(64/\delta) \rceil$.

**Output:**

1: A random variable $\widetilde{p}$ that satisfies $|\mathbb{E}[\widetilde{p}] - p| \leq \delta$, and

$$\mathrm{Var}[\widetilde{p}] = \mathcal{O}(p/t^2 + \delta). \tag{5.2.3}$$

2: A copy of the state $|\psi\rangle$, up to norm error $\delta/16$.

**Queries:** Expected number of queries to $R_{|\psi\rangle}$: $K := \widetilde{\mathcal{O}}(t)$.

**Procedure:** LOW-BIAS-PROB-EST($t$, $\delta$, $R_{|\psi\rangle}$, $|\psi\rangle$):

1: Let $q' =$ PROB-EST($k$, $R_{|\psi\rangle}$, $|\psi\rangle$), and let $q = 1/2 - C_{k,k'}(1/2 - q')$, where $C_{k,k'}$ is as in Algorithm 5.2.1.

2: If $q \leq 1/t^2$, then

1. Prepare the state $|\psi'\rangle$ by applying $L$, where $L =$ LINEAR-AMPL-AMPL($t^2/64$, $\delta/(16K)$, $R_{|\psi\rangle}$, $I \otimes 2(|1\rangle\langle 1| - I)$).
2. Repeat
   (a) Measure the final qubit, denote the outcome with $b' \in \{0, 1\}$.
   (b) Measure the operator $|\psi'\rangle\langle\psi'|$, using the reflection $LR_{|\psi\rangle}L^\dagger$. Denote the outcome by $b \in \{0, 1\}$.

   until the measurement outcome $b = 1$.
3. Restore $|\psi\rangle$ using $L^\dagger$.
4. Output $64b'/t^2$, where $b'$ was obtained in the first iteration.

3: Else, output $\widetilde{p} =$ PROB-EST($k$, $R_{|\psi\rangle}$, $|\psi\rangle$).

---

**Proof of the properties of Algorithm 5.2.2:**
We observe that the total number of calls to $R_{|\psi\rangle}$ in steps 1 and 3 follows easily from the properties of Algorithm 5.2.1. In step 2, we perform again a state reconstruction scheme based on the protocol from [MW05], which we presented already in the proof of Algorithm 5.2.1. The proof in this setting is identical, and we spend $\mathcal{O}(1)$ iterations in step 2.2 in expectation. Therefore, we perform $\mathcal{O}(1)$ calls to $L$ in expectation, and the number of calls to $R_{|\psi\rangle}$ in $L$ follows from the properties of Algorithm 2.4.6. Putting everything together verifies the claim on the number of queries.

Thus, it remains to check the validity of the output of the algorithm. To that end, observe that the output of the algorithm is always in the interval $[-2, 2]$, and

so if $\widetilde{p}$ differs only by $\delta/8$ from an unbiased estimator in total variation distance, then the total bias in the algorithm is at most $\delta$. Thus, it suffices to ensure that the total variation distance picked up throughout the analysis is at most $\delta/8$.

To that end, we first observe that the total norm error picked up by the imperfect implementation of $L$ is at most $K \cdot \delta/(16K) = \delta/16$.

Next, suppose that $p \leq 1/(16t^2)$. Then, $q \geq 1/t^2$ implies

$$|q-p| \geq \frac{1}{2t^2} \geq \frac{\sqrt{p}}{t} + \left(\frac{1}{2t}\right)^2 \geq \frac{4\pi\sqrt{p(1-p)}\cdot 12}{2^k} + \left(\frac{24\pi}{2^k}\right)^2$$

and so, with $m = 12$, we obtain through Equation (5.2.2) that

$$\begin{aligned}\mathbb{P}\left[q \geq \frac{1}{t^2}\right] &\leq \mathbb{P}\left[|q-p| \geq \frac{1}{2t^2}\right] \leq \mathbb{P}\left[|q-p| \geq \frac{4\pi\sqrt{p(1-p)}\cdot m}{2^k} + \left(\frac{2\pi m}{2^k}\right)^2\right] \\ &\leq \left(\frac{1}{2}\right)^{k'/2} \leq \frac{\delta}{64}.\end{aligned}$$

On the other hand, suppose that $p \geq 16/t^2$. Then, $q \leq 1/t^2$ implies

$$\begin{aligned}|q-p| &\geq p - \frac{1}{t^2} \geq p - \frac{p}{16} \geq \frac{p}{2} = \frac{p}{4} + \frac{p}{4} \geq \frac{\sqrt{p}}{4}\cdot\frac{4}{t} + \frac{4}{t^2} \geq \frac{\sqrt{p}}{t} + \left(\frac{1}{2t}\right)^2 \\ &\geq \frac{4\pi\sqrt{p(1-p)}\cdot 12}{2^k} + \left(\frac{24\pi}{2^k}\right)^2,\end{aligned}$$

and so again through Equation (5.2.2) with $m = 12$, we find that

$$\begin{aligned}\mathbb{P}\left[q \leq \frac{1}{t^2}\right] &\leq \mathbb{P}\left[|q-p| \geq \frac{1}{2t^2}\right] \leq \mathbb{P}\left[|q-p| \geq \frac{4\pi\sqrt{p(1-p)}\cdot m}{2^k} + \left(\frac{2\pi m}{2^k}\right)^2\right] \\ &\leq \left(\frac{1}{2}\right)^{k'/2} \leq \frac{\delta}{64}.\end{aligned}$$

Thus, by assuming that $p \leq 1/(16t^2) \Rightarrow q \leq 1/t^2$ and $p \geq 16/t^2 \Rightarrow q \geq 1/t^2$, we incur at most $2\cdot(\delta/64+\delta/64) = \delta/16$ error in total variation distance. Combined with the total variation distance picked up from the imperfect implementation of $L$, we conclude that the total variation distance picked up throughout the analysis is $\delta/8$.

Thus, it suffices to prove that the output of step 2 is correct as long as $p \leq 16/t^2$, and that the output of step 3 is correct as long as $p \geq 1/(16t^2)$. The latter claim follows easily from Algorithm 5.2.1, and so we verify the correctness of step 2.

To that end, first of all observe that with the same idea from [MW05], the output state is again $|\psi\rangle$. Moreover, the amplification factor in $L$ is $t^2/64$, and

hence from $p \cdot t^2/64 \leq 16/t^2 \cdot t^2/64 = 1/4$ we find that the assumption in the linear amplitude amplification algorithm, Algorithm 2.4.6, is satisfied.

Finally, observe that in the first iteration, we obtain output $b' = 1$ with probability $t^2/64 \cdot p$. Thus, the expectation $\mathbb{E}[b'] = pt^2/64$ and hence the expectation of $\widetilde{p}$ is exactly $p$. Furthermore, the variance of $b'$ is upper bounded by $pt^2/64$, from which we find that the variance of $\widetilde{p}$ is upper bounded by $64p/t^2 = \mathcal{O}(p(1-p)/t^2)$. This completes the proof. □

We can easily improve the first term in the variance of the above construction to $\mathcal{O}(p(1-p)/t^2)$, rather than $\mathcal{O}(p/t^2)$. The trick is to perform a slight modification of step 2 in Algorithm 5.2.2, in case $q \geq 1 - 1/t^2$. We omit the details here, because we don't need this particular feature in subsequent sections.

As a demonstration of the applicability of the low-bias, low-variance, non-destructive probability estimation algorithm, we continue this chapter by developing a partition function estimator, which uses the probability estimation procedure outlined above, and crucially relies on its non-destructive, low-variance and low-bias properties.

## 5.3 Unbiased and non-destructive mean estimation

In the previous section, we constructed a low-variance, non-destructive probability estimation routine. In this section, we show how this routine can be used to perform mean estimation non-destructively, with small bias and low variance. We employ very similar ideas to those in Chapter 3, but in a slightly simpler setting since we only consider univariate random variables here.

We start with developing a bounded mean estimation routine.

---

**Algorithm 5.3.1:** Low-bias, low-variance, non-destructive bounded mean estimation

---

**Input:**

1: $X : \Omega \to R \subseteq [0, 1]$: a random variable on a probability space $(\Omega, 2^\Omega, \mathbb{P})$.

2: $t > 0$: a precision parameter.

3: $\delta > 0$: the bias tolerance parameter.

4: A quantum state

$$|\psi\rangle = \sum_{\omega \in \Omega} \sqrt{\mathbb{P}(\omega)}\, |\omega\rangle \in \mathbb{C}^\Omega.$$

5: $R_{|\psi\rangle}$: a quantum circuit acting on $\mathbb{C}^\Omega$ that reflects through $|\psi\rangle$.

6: $O_X$: a quantum circuit that acts on $\mathbb{C}^\Omega \otimes \mathbb{C}^R$, and implements the mapping, for all $\omega \in \Omega$,

$$|\omega\rangle\, |0\rangle \mapsto |\omega\rangle\, |X(\omega)\rangle\,.$$

**Derived objects:**

1: $U$: a quantum circuit acting on $\mathbb{C}^R \otimes \mathbb{C}^2$, as

$$|r\rangle |0\rangle \mapsto |r\rangle \otimes \left(\sqrt{r}\,|1\rangle + \sqrt{1-r}\,|0\rangle\right).$$

2: $C = (I \otimes U)(O_X \otimes I)$, acting on $\mathbb{C}^\Omega \otimes \mathbb{C}^R \otimes \mathbb{C}^2$.

**Output:**

1: A random variable $\widetilde{\mu}$ such that $|\mathbb{E}[\widetilde{\mu}] - \mathbb{E}[X]| \leq \delta$, and

$$\mathrm{Var}[\widetilde{\mu}] \leq \mathcal{O}(\mathbb{E}[X]/t^2 + \delta).$$

2: The state $|\psi\rangle$ up to norm error $\delta$.

**Queries:** Number of queries to $R_{|\psi\rangle}$ and $O_X$: $\widetilde{\mathcal{O}}(t)$.

**Procedure:** BOUNDED-MEAN-EST($X$, $t$, $\delta$, $|\psi\rangle$, $R_{|\psi\rangle}$, $O_X$):

1: Apply $C$ to $|\psi\rangle |0\rangle |0\rangle$ to obtain $|\psi'\rangle \in \mathbb{C}^\Omega \otimes \mathbb{C}^R \otimes \mathbb{C}^2$.

2: Output $\widetilde{\mu} =$ LOW-BIAS-PROB-EST($t$, $\delta$, $CR_{|\psi\rangle}C^\dagger$, $|\psi'\rangle$).

3: Uncompute step 1.

---

**Proof of the properties of Algorithm 5.3.1:**
The claim on the number of queries follows immediately from the properties of Algorithm 5.2.2. Hence, it remains to check the validity of the output. To that end, we must check that the state $|\psi'\rangle$ indeed encodes $\mathbb{E}[X]$ as a probability. Indeed,

$$\begin{aligned}|\psi'\rangle &= \sum_{\omega\in\Omega} \sqrt{\mathbb{P}(\omega)}\,|\omega\rangle\,|X(\omega)\rangle \left(\sqrt{X(\omega)}\,|1\rangle + \sqrt{1-X(\omega)}\,|0\rangle\right), \\ &= \sqrt{\mathbb{E}[X]}\,|\psi_1\rangle\,|1\rangle + \sqrt{1-\mathbb{E}[X]}\,|\psi_0\rangle\,|0\rangle\,,\end{aligned}$$

for suitably chosen unit vectors $|\psi_0\rangle, |\psi_1\rangle$. Thus, $|\psi'\rangle$ satisfies the assumptions of Algorithm 5.2.2, and hence the proof is complete. □

Now, we use the bounded estimator to calculate the mean in the more general setting where we merely know an upper bound on the variance of the random variable, and have access to crude estimate of the mean beforehand.

---

**Algorithm 5.3.2:** Mean estimation with upper bound on variance

---

**Input:**

1: $X : \Omega \to R \subseteq [0,1]$: a random variable on a probability space $(\Omega, 2^\Omega, \mathbb{P})$.

2: $t > 0$: a precision parameter.

3: $\delta > 0$: the variance tolerance parameter.

4: $S > 0$: an upper bound on the variance of $X$.

5: $\overline{\mu} \in \mathbb{R}$: a crude estimate of the mean.

6: A quantum state

$$|\psi\rangle = \sum_{\omega\in\Omega} \sqrt{\mathbb{P}(\omega)}\,|\omega\rangle \in \mathbb{C}^\Omega.$$

7: $R_{|\psi\rangle}$: a quantum circuit acting on $\mathbb{C}^\Omega$ that reflects through $|\psi\rangle$.
8: $O_X$: a quantum circuit that acts on $\mathbb{C}^\Omega \otimes \mathbb{C}^R$, and implements the mapping, for all $\omega \in \Omega$,

$$|\omega\rangle\,|0\rangle \mapsto |\omega\rangle\,|X(\omega)\rangle\,.$$

**Derived objects:**
1: $k = \lceil \log(8S/\delta) \rceil$.
2: $\delta' = \min\{\delta/(8(2^k-1)\sqrt{S}), 3/(4(4^k-1)t^2)\}$.
3: $a_0 = 0$.
4: For every $\ell \in [k]$, let $a_\ell = 2^\ell \sqrt{S}$.
5: For every $\ell \in [k]$, let $R_\ell^\pm = \{(\pm x - \overline{\mu})/a_\ell : x \in R, a_{\ell-1} \leq \pm x - \overline{\mu} < a_\ell\} \cup \{0\}$.
6: For every $\ell \in [k]$, let $X_\ell^\pm : \Omega \to R_\ell$ be defined as

$$X_\ell^\pm(\omega) = \begin{cases} \pm\frac{X(\omega)-\overline{\mu}}{a_\ell}, & \text{if } a_{\ell-1} \leq \pm(X(\omega)-\overline{\mu}) < a_\ell, \\ 0, & \text{otherwise.} \end{cases}$$

7: $U_\ell^\pm$: a circuit acting on $\mathbb{C}^R \otimes \mathbb{C}^{R_\ell^\pm}$, implementing the operation, for all $r \in R$,

$$|r\rangle\,|0\rangle \mapsto |r\rangle \otimes \begin{cases} \left|\pm\frac{r-\overline{\mu}}{a_\ell}\right\rangle, & \text{if } a_{\ell-1} \leq \pm(x-\overline{\mu}) < a_\ell \\ |0\rangle\,, & \text{otherwise.} \end{cases}$$

**Assumption:** $|\overline{\mu} - \mathbb{E}[X]| \leq \sqrt{3S}$.
**Output:**
1: A random variable $\widetilde{\mu}$ such that

$$|\mathbb{E}[\widetilde{\mu}] - \mathbb{E}[X]| \leq \delta \quad \text{and} \quad \mathrm{Var}[\widetilde{\mu}] = \mathcal{O}\left(\frac{S}{t^2}\right).$$

2: The state $|\psi\rangle$ up to norm error $2\delta'$.
**Queries:** Number of queries to $R_{|\psi\rangle}$ and $O_X$: $\widetilde{\mathcal{O}}(t)$.
**Procedure:** MEAN-EST($X$, $t$, $\delta$, $S$, $\overline{\mu}$, $|\psi\rangle$, $R_{|\psi\rangle}$, $O_X$):
1: For $\ell = 1, \ldots, k$,

1. Let $\widetilde{\mu}_\ell^+ =$ BOUNDED-MEAN-EST($X_\ell^+$, $t$, $\delta'$, $|\psi\rangle$, $R_{|\psi\rangle}$, $(I \otimes U_\ell^+)(O_X \otimes I)$).
2. Let $\widetilde{\mu}_\ell^- =$ BOUNDED-MEAN-EST($X_\ell^-$, $t$, $\delta'$, $|\psi\rangle$, $R_{|\psi\rangle}$, $(I \otimes U_\ell^-)(O_X \otimes I)$).

2: Output $\widetilde{\mu} = \overline{\mu} + \sum_{\ell=1}^k a_\ell(\widetilde{\mu}_\ell^+ - \widetilde{\mu}_\ell^-)$.

---

**Proof of the properties of Algorithm 5.3.2:**
The claimed number of queries to $R_{|\psi\rangle}$ and $O_X$ follows immediately from the properties of Algorithm 5.3.1. Thus, it remains to check the claimed properties of the algorithm's output.

To that end, we start by calculating the bias of the estimator $\widetilde{\mu}$. Observe that

$$\begin{aligned}|\mathbb{E}[\widetilde{\mu}] - \mathbb{E}[X]| &= \left|\overline{\mu} - \mathbb{E}[X] + \sum_{\ell=1}^{k} a_\ell(\mathbb{E}[\widetilde{\mu}_\ell^+] - \mathbb{E}[\widetilde{\mu}_\ell^-])\right| \\ &= \left|\overline{\mu} - \mathbb{E}[X] + \sum_{\ell=1}^{k} a_\ell(\mathbb{E}[X_\ell^+] - \mathbb{E}[X_\ell^-])\right| + 2\sum_{\ell=1}^{k} a_\ell\delta'.\end{aligned}$$

The second term can easily be shown to be upper bounded by $\delta/2$, since

$$2\delta'\sum_{\ell=1}^{k} a_\ell = 2\delta'\sqrt{S}\sum_{\ell=1}^{k} 2^\ell = 4\delta'\sqrt{S}\cdot(2^k - 1) \leq \frac{\delta}{2}.$$

Thus, it remains to upper bound the first term by $\delta/2$. To that end, observe that

$$\begin{aligned}&\sum_{\ell=1}^{k} a_\ell\left(\mathbb{E}\left[\frac{X-\overline{\mu}}{a_\ell}1_{a_{\ell-1}\leq X-\overline{\mu}<a_\ell}\right] + \mathbb{E}\left[\frac{X-\overline{\mu}}{a_\ell}1_{a_{\ell-1}\leq -(X-\overline{\mu})<a_\ell}\right]\right) \\ &\quad= \sum_{\ell=1}^{k}\mathbb{E}[(X-\overline{\mu})\cdot 1_{a_{\ell-1}\leq|X-\overline{\mu}|<a_\ell}] = \mathbb{E}[(X-\overline{\mu})1_{|X-\overline{\mu}|<a_k}],\end{aligned}$$

and so

$$\begin{aligned}&\left|\overline{\mu} - \mathbb{E}[X] + \sum_{\ell=1}^{k} a_\ell(\mathbb{E}[X_\ell^+] - \mathbb{E}[X_\ell^-])\right| = \left|\mathbb{E}[(X-\overline{\mu})1_{|X-\overline{\mu}|\geq a_k}]\right| \\ &\quad\leq \frac{1}{a_k}\mathbb{E}\left[(X-\overline{\mu})^2\right] \leq \frac{4S}{a_k} = \frac{4S}{\sqrt{S}\cdot 2^k} \leq \frac{\delta}{2},\end{aligned}$$

where the final inequality holds due to our choice of $k$. Thus, we indeed conclude that the bias in the algorithm is at most $\delta$.

Next, we turn our attention to the variance of the estimator $\widetilde{\mu}$. We have

$$\mathrm{Var}[\widetilde{\mu}] = \sum_{\ell=1}^{k} a_\ell^2(\mathrm{Var}[\widetilde{\mu}_\ell^+] + \mathrm{Var}[\widetilde{\mu}_\ell^-]) = \mathcal{O}\left(\sum_{\ell=1}^{k} a_\ell^2\cdot\left(\frac{\mathbb{E}[X_\ell^+] + \mathbb{E}[X_\ell^-]}{t^2} + 2\delta'\right)\right).$$

We first focus on the final term within the parenthesis. Summing over $\ell$ yields

$$2\delta'\sum_{\ell=1}^{k} a_\ell^2 = 2\delta'\sum_{\ell=1}^{k} S\cdot 4^\ell = \frac{4}{3}S\delta'(4^k - 1) \leq \frac{S}{t^2}.$$

On the other hand, for all $\ell \in [k]$ with $\ell \geq 2$,

$$\mathbb{E}[X_\ell^+] = \frac{1}{a_\ell}\mathbb{E}\left[(X-\overline{\mu})1_{a_{\ell-1}\leq X-\overline{\mu}<a_\ell}\right] \leq \frac{1}{a_\ell a_{\ell-1}}\mathbb{E}\left[(X-\overline{\mu})^2\cdot 1_{a_{\ell-1}\leq X-\overline{\mu}<a_\ell}\right],$$

and similarly

$$\mathbb{E}[X_\ell^-] = \frac{1}{a_\ell}\mathbb{E}\left[-(X-\overline{\mu})1_{a_{\ell-1}\le -(X-\overline{\mu})<a_\ell}\right] \le \frac{1}{a_\ell a_{\ell-1}}\mathbb{E}\left[(X-\overline{\mu})^2\cdot 1_{a_{\ell-1}\le -(X-\overline{\mu})<a_\ell}\right].$$

Finally, when $\ell = 1$, we have

$$\mathbb{E}[X_\ell^+] \le \frac{1}{a_1}\mathbb{E}[(X-\overline{\mu})1_{0\le X-\overline{\mu}<a_1}] \le 1,$$

and similarly $\mathbb{E}[X_\ell^-] \le 1$. Thus,

$$\begin{aligned}\sum_{\ell=1}^{k} a_\ell^2 \cdot \frac{\mathbb{E}[X_\ell^+]+\mathbb{E}[X_\ell^-]}{t^2} &\le \frac{2S}{t^2} + \frac{2}{t^2}\sum_{\ell=2}^{k}\mathbb{E}\left[(X-\overline{\mu})^2\cdot 1_{a_{\ell-1}\le |X-\overline{\mu}|<a_\ell}\right] \\ &\le \frac{2S}{t^2} + \frac{2}{t^2}\mathbb{E}\left[(X-\overline{\mu})^2\right] \le \frac{4S}{t^2}.\end{aligned}$$

This completes the proof. □

The general mean estimation algorithm that we constructed here is very similar to the multivariate mean estimation algorithm that we developed in Section 3.4.1. However, there is one intricate qualitative difference, namely in Section 3.4.1 we didn't need a crude estimate of the mean as the input to the algorithm, since we could start the algorithm by obtaining a few classical samples to compute such an estimate.

The issue with this approach in Algorithm 5.3.2 is that taking classical samples is an inherently destructive operation. For instance, suppose that $|\psi\rangle$ is a uniform superposition over all possible outcomes $\omega \in \Omega$. Performing a measurement will collapse this state to one of the outcomes. Moreover, the resulting state inherently has very small overlap with the initial state $|\psi\rangle$, and as such it takes a lot of work to reconstruct $|\psi\rangle$. Thus, taking classical samples is not an available algorithmic tool in the setting considered in this chapter.

This leaves the question how one obtains a crude estimate $\overline{\mu}$ to the mean of a random variable, to be used as input to Algorithm 5.3.2. We observe that the quantiles of our random variable tell us something about the spread. In particular, we can relate the quantiles at 1/3 and 2/3 to the mean and variance by means of the following lemma.

**5.3.3.** Lemma. *Let $X : \Omega \to \mathbb{R}$ be a univariate random variable, with well-defined mean $\mu$ and variance $\sigma^2$. Let $x \in \mathbb{R}$. Then,*

$$\begin{aligned}\mathbb{P}[X \ge x] > \frac{1}{3} &\qquad \Rightarrow x - \mu < \sqrt{3}\sigma, \\ \mathbb{P}[X \ge x] < \frac{2}{3} &\qquad \Rightarrow x - \mu > -\sqrt{3}\sigma.\end{aligned}$$

**Proof:**
Suppose that $x - \mu \geq \sqrt{3}\sigma$. Then, by Chebyshev's inequality,

$$\mathbb{P}\left[X \geq x\right] \leq \mathbb{P}\left[X - \mu \geq \sqrt{3}\sigma\right] \leq \frac{1}{3}.$$

Thus, taking the contrapositive of the statement above yields the first claim. The second claim follows similarly. This completes the proof. □

Thus, if we can find an $x \in \mathbb{R}$ such that $\mathbb{P}[X \geq x] \in (1/3, 2/3)$, then we find that $|x - \mu| \leq \sqrt{3}\sigma$. In other words, in order to generate a crude estimate of the mean that we can use as input to the non-destructive mean estimation algorithm, Algorithm 5.3.2, it suffices to be able to find quantiles of the random variable.

In the special case where the support of the random variable contains only finitely many values, say $n$, we can use binary search and probability estimation to find a suitable quantile with $\mathcal{O}(\log(n))$ steps. Below we sketch an algorithm that achieves this.

**Algorithm 5.3.4:** Non-destructive quantile estimation with finite support

**Input:**
1: $X : \Omega \to R = \{x_1, \ldots, x_n\}$: a univariate random variable on a probability space $(\Omega, 2^\Omega, \mathbb{P})$.
2: $p \in [0, 1]$: a quantile.
3: $\varepsilon > 0$: the precision parameter.
4: $\delta > 0$: the failure probability tolerance.
5: A copy of the state

$$|\psi\rangle = \sum_{\omega \in \Omega} \sqrt{\mathbb{P}(\omega)}\, |\omega\rangle\,.$$

6: $R_{|\psi\rangle}$: a quantum circuit acting on $\mathbb{C}^\Omega$ that reflects through $|\psi\rangle$.
7: $O_X$: a quantum circuit acting on $\mathbb{C}^\Omega \otimes \mathbb{C}^R$, that implements the operation

$$O_X : |\omega\rangle\,|0\rangle \mapsto |\omega\rangle\,|X(\omega)\rangle\,.$$

**Derived objects:**
1: $k = \lceil \log(\frac{96\pi}{\varepsilon}) \rceil$.
2: $k' = \lceil 2\log(\log(n)/\delta) \rceil$.
**Output:**
1: An $\varepsilon$-approximate $p$-quantile of $X$.
2: The state $|\psi\rangle$.
**Success probability:** Lower bounded by $1 - \delta$.
**Queries:** Number of calls to $O_X$: $\widetilde{\mathcal{O}}(\log(n)/\varepsilon)$.
**Procedure:** QUANT-EST($X$, $p$, $\varepsilon$, $\delta$, $|\psi\rangle$, $O_X$):
1: Set $a = 0$ and $b = n$.
2: Repeat

1. Set $k = \lceil (a+b)/2 \rceil$.
2. Apply the operation $U$ to $|\psi\rangle |0\rangle \in \mathbb{C}^R \otimes \mathbb{C}^2$, where
$$U : |r\rangle |0\rangle \mapsto |r\rangle \otimes \begin{cases} |1\rangle, & \text{if } r \geq x_k, \\ |0\rangle, & \text{otherwise.} \end{cases}$$
Denote the resulting state $|\psi'\rangle$.
3. Let $\overline{p} = \textsc{Prob-est}(k, k', U(R_{|\psi\rangle} \otimes I)U^\dagger, |\psi'\rangle)$, where we take the output $\overline{p}$ rather than $\widetilde{p}$.
4. Uncompute step 2.2.
5. If $\overline{p} \geq p$, set $a = k$, else set $b = k - 1$.

Until $a = b$.

3: Output $x_a$.

**Proof of the properties of Algorithm 5.3.4:**
The algorithm runs binary search, and as such performs $\mathcal{O}(\log(n))$ iterations. Furthermore, the cost per iteration is $\widetilde{\mathcal{O}}(1/\varepsilon)$, as follows directly from the properties of Algorithm 5.2.1.

Thus, it remains to check the validity of the output. To that end, we must ensure that throughout the algorithm, the failure probability is at most $\delta$. To that end, observe that if in all iterations $\mathbb{P}[X \geq x_k] \leq p - \varepsilon$ implies that $\overline{p} < p$, and similarly $\mathbb{P}[X \geq x_k] \geq p + \varepsilon$ implies that $\overline{p} \geq p$, then we find a suitable outcome. Thus, we bound the probability of either of these implications not holding in any of the iterations. To that end, observe that in any iteration
$$\mathbb{P}[|\overline{p} - p| \geq \varepsilon] \leq \mathbb{P}\left[|\overline{p} - p| \geq \frac{48\pi}{2^k} + \left(\frac{24\pi}{2^k}\right)^2\right] \leq \frac{1}{2^{k'/2}} \leq \frac{\delta}{\log(n)}.$$
Thus, the total failure probability is at most $\delta$. This completes the proof. □

Thus, the idea for non-destructive mean estimation is to first find a crude estimate of the mean using this quantile estimator, and subsequently run Algorithm 5.3.2 to obtain a proper low-bias estimate of the mean. In the next section, we will see how the low-variance, low-bias and non-destructive properties play a crucial role in estimating partition functions.

## 5.4 Partition function estimation

In this section, we showcase an application of our newly-constructed quantum mean estimation algorithm. Specifically, we show how it can be used to speed up existing quantum algorithms for estimating partition functions. In Section 5.4.1, we elaborate on the generic algorithm for partition function estimation and how our results provide a speed-up. Subsequently, in Section 5.4.2, we discuss how this gives rise to more efficient quantum algorithms for several applications.

### 5.4.1 Algorithm overview

In this subsection, we first describe our partition function estimation algorithm on a high level, and define the required notation along the way. We follow the exposition in [HW20], but use slightly different notation to match the rest of this document more closely.

Let $\Omega$ be a state space, and let $H : \Omega \to \{0, \dots, n\}$ be a classical Hamiltonian, i.e., a function associating a numerical value to each of the states. Our goal will be to compute the number of states whose value is 0, i.e., $|H^{-1}(0)|$. To that end, we define the partition function $Z : [0, \infty] \to \mathbb{R}$ as

$$Z(\beta) = \sum_{x \in \Omega} e^{-\beta H(x)},$$

and we observe that $Z(0) = |\Omega|$, and $Z(\infty) = |H^{(-1)}(0)|$.

Next, we define a sequence of inverse temperatures $0 = \beta_0 < \beta_1 < \cdots < \beta_\ell = \infty$, and express $Z(\infty)$ as the product

$$Z(\infty) = Z(0) \cdot \frac{Z(\beta_1)}{Z(0)} \cdot \frac{Z(\beta_2)}{Z(\beta_1)} \cdot \ldots \cdot \frac{Z(\infty)}{Z(\beta_\ell)} = Z(0) \prod_{i=0}^{\ell-1} \frac{Z(\beta_{i+1})}{Z(\beta_i)}. \tag{5.4.1}$$

Since throughout the sequence of $\beta_i$'s, we are increasing the inverse temperature, and hence decreasing the temperature, this sequence is referred to as a *cooling schedule*. Its length $\ell$ is called the *schedule length*. The core idea for estimating $Z(\infty)$ is to evaluate each of the factors in the product on the right-hand side of Equation (5.4.1) individually.

Thus, let $i \in [\ell]$. We endow $\Omega$ with a probability distribution $\pi_{\beta_i}$, called the Gibbs distribution in statistical physics, and define a random variable $X_i : \Omega \to \mathbb{R}$ on it, with

$$\pi_{\beta_i}(x) = \frac{e^{-\beta_i H(x)}}{Z(\beta_i)} \qquad \text{and} \qquad X_i(x) = e^{-(\beta_i - \beta_{i-1})H(x)}.$$

It follows immediately that

$$\mathbb{E}[X_i] = \sum_{x \in \Omega} \frac{e^{-\beta_i H(x)}}{Z(\beta_i)} \cdot e^{-(\beta_{i+1} - \beta_i)H(x)} = \sum_{x \in \Omega} \frac{e^{-\beta_{i+1} H(x)}}{Z(\beta_i)} = \frac{Z(\beta_{i+1})}{Z(\beta_i)}.$$

The idea is now to devise a procedure that samples from the Gibbs distribution and obtains an estimator $\widetilde{\mu}_i$ of $\mathbb{E}[X_i]$.

In order to be able to sample from the Gibbs distribution, we encode it in a quantum state. To that end, we define the Gibbs state $|\pi_{\beta_i}\rangle \in \mathbb{C}^\Omega$ at inverse temperature $\beta_i$ as

$$|\pi_{\beta_i}\rangle = \frac{1}{\sqrt{Z(\beta_i)}} \sum_{x \in \Omega} \sqrt{e^{-\beta_i H(x)}} \, |x\rangle \, .$$

In typical applications (see the next subsection), it can be much easier to reflect through the Gibbs state than to prepare it. Therefore, we estimate $\mathbb{E}[X_i]$ using a nondestructive mean estimation algorithm, which we constructed in the previous sections. This has the benefit of enabling access to the Gibbs state even after the computation of $\widetilde{\mu}_i$, which makes it possible to reuse it for computing the next factor in the product on the right-hand side of Equation (5.4.1).

It remains to choose the cooling schedule, i.e., the sequence of inverse temperatures $\beta_1, \ldots, \beta_{\ell-1}$. For reasons that are sketched below on a high level, we want our cooling schedule to have the following two properties.

1. *$B$-Chebyshev.* For any $B \geq 1$, a cooling schedule is called $B$-Chebyshev if for any two subsequent $\beta_{i-1}$ and $\beta_i$, we have

$$\frac{\mathbb{E}[X_i^2]}{\mathbb{E}[X_i]^2} = \frac{Z(2\beta_{i+1} - \beta_i)Z(\beta_i)}{Z(\beta_{i+1})^2} \leq B.$$

   This requirement can be viewed as a tail bound – it tells us that $X_i$ concentrates well around its mean $\mathbb{E}[X_i]$.

2. *$B$-slowly varying.* For any $B \geq 1$, a cooling schedule is called $B$-slowly varying if

$$\left|\left\langle \pi_{\beta_i} \middle| \pi_{\beta_{i+1}} \right\rangle\right|^2 = \frac{Z\left(\frac{\beta_i + \beta_{i+1}}{2}\right)^2}{Z(\beta_i)Z(\beta_{i+1})} \geq 1/B.$$

   This requirement can be seen as a proximity requirement – it tells us that the Gibbs states $|\pi_{\beta_i}\rangle$ and $\left|\pi_{\beta_{i+1}}\right\rangle$ have some non-negligible overlap. It ensures that we can *anneal*, i.e., transform, $|\pi_{\beta_i}\rangle$ into $\left|\pi_{\beta_{i+1}}\right\rangle$ without having to do too much work, and hence circumvents having to prepare $\left|\pi_{\beta_{i+1}}\right\rangle$ from scratch for estimating the next mean $\mathbb{E}[X_{i+1}]$.

When we set $B = e^2$, it is shown in [SVV09] that there exists a cooling schedule of length $\ell = \widetilde{\mathcal{O}}(\sqrt{\log|\Omega| \log(n)})$, which is $B$-Chebyshev. Subsequently, it was shown [HW20] that one can modify the construction so that the resulting cooling schedule is not only $B$-Chebyshev, but also $B$-slowly varying. Moreover, computing this cooling schedule can be done on the fly, with associated cost scaling approximately linearly in the schedule length.

Combining previous work [SVV09; HW20] with our new unbiased and nondestructive mean estimation techniques gives rise to the following result.

**5.4.1.** Theorem (Partition function estimator). *Let $H : \Omega \to \{0, \ldots, n\}$ be a Hamiltonian, and let $Z$ be its partition function. Suppose that, for every inverse temperature $\beta$, the associated Gibbs distribution $\pi_\beta$ is the stationary distribution of an ergodic reversible Markov chain with spectral gap at least $\delta$. Then,*

*we can estimate $Z(\infty)$ up to multiplicative error $\varepsilon$, with probability at least $2/3$, by using*

$$\widetilde{\mathcal{O}}\left(\log^{3/4}|\Omega|\log^{3/4}(n)/\varepsilon+\sqrt{\log|\Omega|}\log^{3/2}(n)/\sqrt{\delta}\right)$$

*steps of the quantum walk operator in expectation.*

**Proof:**
We run the algorithm from [HW20] to compute the cooling schedule on the fly. The total number of Gibbs-state reflections used for that is $\widetilde{\mathcal{O}}(\sqrt{\log|\Omega|}\log^{3/2}n)$, as proved in [HW20, Theorem 13], and the resulting cooling schedule is both $e^2$-slowly varying and $e^2$-Chebyshev, and of length $\ell=\widetilde{\mathcal{O}}(\sqrt{\log|\Omega|\log n})$. For computing the product in Equation (5.4.1), we estimate each of the factors individually with Algorithm 5.3.2, with relative variance $v_i=\varepsilon^2/(2\ell)$ for the $i$th factor. The resulting relative variance $v$ of the product estimator then satisfies

$$1+v=\prod_{i=1}^{\ell}(1+v_i)=\left(1+\frac{\varepsilon^2}{2\ell}\right)^{\ell}\leq e^{\varepsilon^2/2}\leq 1+\varepsilon^2,$$

where we used that $1+v$ with $v$ the relative variance is multiplicative for product estimators. The total number of Gibbs state reflections used in this algorithm is $\widetilde{\mathcal{O}}(\log^{3/4}|\Omega|\log^{3/4}(n)/\varepsilon+\sqrt{\log|\Omega|}\log^{3/2}n)$. With standard failure probability reduction techniques, we obtain that the number of Gibbs-state reflections performed by the resulting algorithm scales as the sum of the two complexities. Finally, by a well-known result [Sze04; MNR+11], each reflection through a Gibbs state $\pi_\beta$ can be implemented with $\mathcal{O}(1/\sqrt{\delta})$ steps of the quantum walk operator corresponding to a Markov chain with spectral gap at least $\delta$ generating $\pi_\beta$. □

The core of our improvement lies in our product estimator – the one used in [HW20] is quadratic in the schedule length, whereas ours is subquadratic. Thus, more generally, if we are in a setting where a cooling schedule that is shorter than $\widetilde{\mathcal{O}}(\sqrt{\log|\Omega|\log(n)})$ suffices, then our resulting algorithm scales with the schedule length to the power of $3/2$. A typical setting where this is the case is when we a priori know a lower bound on $Z(\infty)$ – then we can terminate the annealing at a lower inverse temperature, and hence obtain a shorter cooling schedule. We leave working out the details in this setting for future work.

### 5.4.2 Applications

The quantum partition function estimator can be applied to several counting problems. See [HW20] for more possible applications. In all applications we list in this section, $n$ scales at most polylogarithmically in $|\Omega|$, so we can hide any dependence on $\log(n)$ in the tilde of the big-$O$ notation.

**Counting $k$-colorings.** Let $G = (V, E)$ be an undirected simple graph with degree at most $\Delta$, and suppose we have $k > 2\Delta$ colors. We would like to count the number of ways we can color the vertices such that no two adjacent ones have the same color. We let $\Omega$ be the set of all colorings $x \in [k]^V$, which means $Z(0) = |\Omega| = k^{|V|}$. We define $H(x)$ to be the number of monochromatic edges in $G$ when each vertex $v \in V$ is colored with $x_v$. Then, $Z(\infty) = |H^{(-1)}(0)|$ is exactly the number of $k$-colorings. Jerrum [Jer95] constructed a Markov chain whose stationary distribution is the Gibbs state at infinite temperature, and showed that its mixing time is $O(|V| \log |V|)$. With a Metropolis sampling correction this can be turned into Gibbs states at arbitrary temperatures, and thus performing an approximate Gibbs state reflection takes time $\widetilde{\mathcal{O}}(\sqrt{|V|})$. Multiplying by the complexity in Theorem 5.4.1, we can obtain an $\varepsilon$-multiplicative approximation to $Z(\infty)$ in $\widetilde{\mathcal{O}}(\log^{3/4}(|\Omega|)\sqrt{|V|}/\varepsilon) = \widetilde{\mathcal{O}}(|V|^{5/4}/\varepsilon)$ steps in the graph.

**Ising model.** Let $G = (V, E)$ be an undirected simple graph of maximum degree $\Delta$, and assign a sign $\pm 1$ to each of its vertices. We let $\Omega$ be the set of all assignments $x \in \{-1, 1\}^V$ of signs to the vertices, which readily implies that $|\Omega| = 2^{|V|}$. We let $H(x)$ be the number of edges whose endpoints have the same sign, when the sign of each vertex $v \in V$ is $x_v$. Then, Mossel and Sly [MS13] constructed a Markov chain whose stationary distribution is the Gibbs state at arbitrary temperature. They also show that it mixes in time $O(|V| \log |V|)$ as long as $(\Delta - 1)\tanh(\beta) < 1$. Thus, for any inverse temperature $\beta$ satisfying such a condition, we can evaluate $Z(\beta)$ up to relative error $\varepsilon$ in $\widetilde{\mathcal{O}}(|V|^{5/4}/\varepsilon)$ steps in the graph.

**Counting matchings.** Let $G = (V, E)$ be an undirected simple graph. A matching $x \subseteq E$ is a subset of disjoint edges. We let $\Omega$ be the set of all matchings, and we set $H(x) = |x|$ to be the number of edges in the matching. Then $Z(0) = |\Omega|$ is the number of matchings, and $Z(\infty) = 1$. Now, the setting is slightly different than before, since the partition function is easy to calculate at zero temperature ($\beta = \infty$), rather than at infinite temperature ($\beta = 0$). The core idea is to run the same algorithm as outlined before, but anneal in the opposite direction. Previous works [HW20; Mon15] justify this idea in more detail. Jerrum and Sinclair [JS89] constructed a Markov chain whose stationary distribution is the Gibbs state at zero temperature, and analyzed that the mixing time of this Markov chain is $O(|V||E|)$. Similarly as before, the Markov chain can be easily adapted by a metropolis sampling correction so that its stationary distribution is a Gibbs state at an arbitrary temperature, and so we can perform Gibbs state reflections in time $\widetilde{\mathcal{O}}(\sqrt{|V||E|})$. The state space of this random walk is of size $|\Omega| = O(|V|!2^{|V|})$, which implies that $\log |\Omega| = O(|V| \log |V|)$. Thus, combining with the complexity in Theorem 5.4.1, we can find an $\varepsilon$-multiplicative estimate of the number of matchings in $\widetilde{\mathcal{O}}(|V|^{5/4}|E|^{1/2}/\varepsilon)$ steps in the graph.

**Computing the volume of a convex body.** Finally, we describe a quantum algorithm for estimating the volume of convex bodies that provides an improvement over the algorithm given in [CCH+19]. The problem statement is as follows. Let $K \subseteq \mathbb{R}^d$ be a convex body, and let $R > 0$ be such that $B(1) \subseteq K \subseteq B(R)$, where $B(r) = \{x \in \mathbb{R}^d : \|x\|_2 \leq r\}$ is the ball of radius $r$ centered at 0. We wish to estimate the volume $\mathrm{Vol}(K)$ of $K$ up to multiplicative precision $\varepsilon$, i.e., to output $\widetilde{V}$ such that $(1-\varepsilon)\,\mathrm{Vol}(K) \leq \widetilde{V} \leq (1+\varepsilon)\,\mathrm{Vol}(K)$. We assume to have access to the convex body by means of a membership oracle $O_K$, i.e., for any point $x \in \mathbb{R}^d$, we can query whether $x \in K$. We are interested in the query complexity of this problem, i.e., we wish to minimize the number of queries made to the membership oracle. We note here that previous work also consider time-efficient implementations of algorithms that solve this problem [CCH+19]. We expect that similar techniques could also be used to implement our algorithm time-efficiently, but we leave this for future work.

First, observe that if the precision $\varepsilon$ is smaller than $\varepsilon \leq (3/4)^d$ then any polylogarithmic overhead in $1/\varepsilon$ is polynomial in the dimension $d$. Thus, in this regime we can run a simple adaption of approximate counting on a suitably dense discretization of $B(R)$ – this will run with $O(1/\varepsilon)$ queries, which, when adding polylogarithmic overhead in $1/\varepsilon$, already achieves the desired $\widetilde{\mathcal{O}}(d^3 + d^{2.25}/\varepsilon)$ query complexity. Thus, without loss of generality, we can focus on the regime in which $\varepsilon > (3/4)^d$.

Previous works make use of the *pencil construction*, introduced in [LV06], where the idea is to define a new convex body $K' \in \mathbb{R}^{d+1}$ with one extra dimension, as

$$K' = \{\mathbf{x} = (x_0, x) \in \mathbb{R}^{d+1} : x \in K \wedge x_0 \in [0, 2R] \wedge \|x\|_2 \leq x_0\}.$$

The algorithm now consists of two steps. First, the volume of $K'$ is estimated up to multiplicative precision $\varepsilon/2$ using the Markov Chain Monte Carlo framework, with the partition function defined as

$$Z(\beta) = \int_{K'} e^{-\beta x_0}\, \mathrm{d}\mathbf{x}.$$

For sufficiently large values of $\beta$, the value of the partition function is almost completely determined by the tip of the pencil, whose shape is known to us a priori. On the other hand, for sufficiently small values of $\beta$, the partition function essentially captures the volume of $K'$. These claims are made more precise in [CCH+19] and the references therein, resulting in Algorithm 4 in said paper which uses a cooling schedule of length $m = \widetilde{\Theta}(\sqrt{d})$. We can speed up this part by using our unbiased product estimator in Step 2 of said algorithm. This reduces the number of steps of the quantum walk by a factor of $\sqrt{m} = \widetilde{\Theta}(d^{1/4})$. Hence, we can estimate the volume of $K'$ with $\widetilde{\mathcal{O}}(d^3 + d^{2.25}/\varepsilon)$ calls to the membership oracle.

The second step of the algorithm relates the volumes of $K$ and $K'$. It relies on the observation that $R\,\mathrm{Vol}(K) \leq \mathrm{Vol}(K') \leq 2R\,\mathrm{Vol}(K)$ since $[R, 2R] \times K \subseteq K' \subseteq [0, 2R] \times K$. The idea is then to use *rejection sampling* to obtain an $\varepsilon/2$-precise multiplicative estimate of the ratio between $\mathrm{Vol}(K')$ and $\mathrm{Vol}(K)$. The procedure outlined in [CCH+19, Page 29] prepares approximately uniform samples from $K$ in $\widetilde{\mathcal{O}}(d^{2.5})$ calls to the membership oracle. These samples can be trivially converted to samples from $[0, 2R] \times K$ by sampling uniformly from the interval $[0, 2R]$ and adding the result as an extra dimension. Classically, one could now take $O(1/\varepsilon^2)$ uniform samples from $[0, 2R] \times K$, and count the fraction that is contained in $K'$. This yields an $\varepsilon$-precise multiplicative estimate of the ratio between $\mathrm{Vol}(K)$ and $\mathrm{Vol}(K')$. Quantum approximate counting speeds up this step and only requires $O(1/\varepsilon)$ quantum samples, yielding a total of $\widetilde{\mathcal{O}}(d^{2.5}/\varepsilon)$ queries to the membership oracle in this step of the algorithm.

We claim that the second step can be done with $\widetilde{\mathcal{O}}(d^2/\varepsilon)$ instead of $\widetilde{\mathcal{O}}(d^{2.5}/\varepsilon)$ queries, essentially because sampling from $K$ requires in fact only $\widetilde{\mathcal{O}}(d^2)$ quantum queries. For simplicity in the proof, we describe a slightly different rejection sampling strategy achieving this complexity. Instead of sampling from $[0, 2R] \times K$ and checking whether the sample is contained in $K'$, we sample from $K'$ and check whether the resulting sample is contained in $[R, 2R] \times K$. Using the same analysis as above, we obtain an $\varepsilon/2$-precise multiplicative estimate of the ratio between $\mathrm{Vol}(K)$ and $\mathrm{Vol}(K')$, using $O(1/\varepsilon)$ quantum samples from $K'$. Sampling approximately uniformly from $K'$ can be achieved with the simulated annealing schedule from [CCH+19, Algorithm 3] that requires $\widetilde{\mathcal{O}}(d^2)$ calls to the membership oracle. It remains to check that the obtained samples from $K'$ are sufficiently close to uniform, which they are if we marginally increase the length of the cooling schedule.

Let $\varepsilon_1 > 0$ be fixed later, and let $\beta' \leq \varepsilon_1/(2R)$. We show that the Gibbs state at this inverse temperature, denoted by $|\pi'\rangle = \left|\pi'_{\beta'}\right\rangle$, is close to the uniform distribution over $K'$, denoted by $|\pi\rangle$. To that end, recall that $\mathrm{Vol}(K') \geq Z(\beta')$, and so

$$\begin{aligned}\mathrm{Vol}(K')(1 - \langle\pi|\pi'\rangle) &= \int_{K'} \left(1 - \sqrt{\frac{\mathrm{Vol}(K')}{Z(\beta')}} e^{-\beta' x_0}\right) \mathrm{d}\mathbf{x} \leq \int_{K'} \left(1 - e^{-\beta' x_0}\right) \mathrm{d}\mathbf{x} \\ &\leq \int_{K'} \beta' x_0 \,\mathrm{d}\mathbf{x} \leq 2\beta' R\,\mathrm{Vol}(K') \leq \varepsilon_1\,\mathrm{Vol}(K'),\end{aligned}$$

which implies that $\||\pi\rangle - |\pi'\rangle\| = 2\sqrt{1 - \langle\pi|\pi'\rangle} \leq \sqrt{\varepsilon_1}$.

Next, we can prepare $|\pi'\rangle$ by choosing $m = \lceil \sqrt{d} \log(4dR/\varepsilon_1) \rceil$ in [CCH+19, Algorithm 3], where $m$ denotes the length of the cooling schedule. The final inverse temperature then satisfies $\beta' \leq 2d(1 - 1/\sqrt{d})^{\sqrt{d}\log(4dR/\varepsilon_1)} \leq 2de^{-\log(4dR/\varepsilon_1)} = \varepsilon_1/(2R)$. Since in every annealing step we are performing $N$ Gibbs state reflections, where $N = O(1)$, we can implement these reflections with precision

$\sqrt{\varepsilon_1}/(mN)$, to ensure that the total norm error from the imperfections of these Gibbs state reflections amounts to $\sqrt{\varepsilon_1}$ as well. Thus, we end up preparing a uniform quantum sample up to precision $2\sqrt{\varepsilon_1}$, with a total number of reflections through Gibbs states that satisfies $O(m \log(mN/\sqrt{\varepsilon_1})) = \widetilde{\mathcal{O}}(\sqrt{d} \log(1/\varepsilon_1))$.

Now, with $\varepsilon_1 = 1/A^2(2N')^2 = \Theta(\varepsilon^2)$, where $N' = \Theta(1/\varepsilon)$ is the number of calls to the state-preparation unitary in the approximate counting algorithm, we obtain that the total accumulated error throughout the whole procedure is at most $1/A$. With $A$ a large enough constant, we ensure that the error probability of this step is negligible. Thus, we obtain that the total number of Gibbs state reflections in this second part of the procedure becomes $\widetilde{\mathcal{O}}(N'\sqrt{d} \log(1/\varepsilon_1)) = \widetilde{\mathcal{O}}(\sqrt{d}/\varepsilon)$. Since every reflection through a Gibbs state can be performed with $\widetilde{\mathcal{O}}(d^{1.5}\text{polylog}(1/\varepsilon))$ membership queries, the total number of calls to $O_K$ becomes $\widetilde{\mathcal{O}}(d^2/\varepsilon)$. Thus, this second step is less costly than the first step, and the resulting total query complexity becomes $\widetilde{\mathcal{O}}(d^3 + d^{2.25}/\varepsilon)$.

# Part II

# Span programs

# Chapter 6

# The span program formalism

The span program formalism is a method to design quantum algorithms. On very high level, one takes a computational problem, and embeds it in the geometry of a Hilbert space by constructing a mathematical object called a span program. The formalism, then, provides a way to turn this mathematical object into a quantum algorithm, and analyzes its cost.

Span programs were first introduced in the classical literature in the study of logspace complexity [KW93]. Later, they were introduced to quantum computing through the study of evaluating boolean formulas [RŠ12][1]. They gained most attention when they were used in a landmark result to prove that the quantum adversary bound is a tight characterization of quantum query complexity of boolean functions, first up to polylogarithmic factors [Rei09], and later up to constants [Rei11].

Afterwards, progress was made in two directions. First, the formalism was extended in several ways, e.g., to include functions over non-boolean alphabets [Jef14], and an approximate version of span programs was introduced [IJ19]. Later, the formalism was also extended to accommodate functions with non-binary output [BT19].

Second, several explicit constructions for span programs were introduced, most notably for *st*-connectivity [BR12; JJK+18], cycle detection and bipartiteness testing [Āri15; CMB18], $k$-distinctness [Bel12], formula evaluation [RŠ12; Rei09; JK17], and oracle identification [Tag22]. Additionally, conversions from classical algorithms to span programs exist [BT20; BTT22; CMP22], as well as conversions from quantum algorithms to approximate span programs [Jef14; Jef20; CJO+20].

In this chapter, we give a self-contained introduction to span programs. Alongside presenting formal proofs, we specifically emphasize building intuition for the mathematical objects defined, theorems stated and proofs given. Most of this intuition is conveyed through visualizations, and to accommodate those, we change

[1]The first version of this work appeared in 2007, so it indeed predates the other results mentioned here.

the notation in this text slightly from the notation used in the existing literature. We highlight along the way how the new notation relates to that used in prior works.

Furthermore, in this text, we constrain ourselves to the case of computing functions with boolean output, i.e., decision problems. It might be possible to generalize many of the results presented here to the more general setting where the function output can take more than two values, however such generalizations are likely not straightforward. We leave these for future work.

This chapter is structured as follows. In Section 6.1, we formally define the formalism and show how it can be used to build quantum algorithms. Subsequently, in Section 6.2, we show how span programs relate to the quantum adversary bound.

## 6.1 Definition and basic properties

In this first section, we formally define span programs, and present their basic properties. The analysis presented here can be found in several previous works, albeit with different notation [Rei09; Jef14; IJ19; BT20]. The emphasis in this section is on developing a more intuitive approach and a pictorial interpretation of the mathematical objects that we define along the way. These interpretations are the core novelty in this section.

### 6.1.1 Span programs and witnesses

We start by formally defining span programs.

**6.1.1.** Definition (Span program). A span program consists of the following mathematical objects:

1. The *state space*: A Hilbert space $\mathcal{H}$ of finite dimension.
2. The *domain*: A finite set $\mathcal{D}$, whose elements are referred to as *inputs*.
3. The *input-dependent subspaces*: to every *input* $x \in \mathcal{D}$, we associate an *input-dependent subspace* $\mathcal{H}(x) \subseteq \mathcal{H}$.
4. The *input-independent subspace*: $\mathcal{K} \subseteq \mathcal{H}$.
5. The *initial state*: $|w_0\rangle \in \mathcal{K}^\perp$, with $\||w_0\rangle\| = 1$.

Then $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ is a span program on $\mathcal{D}$. We make a distinction between positive and negative inputs, as such:

1. $x \in \mathcal{D}$ is a *positive input*, if $|w_0\rangle \in \mathcal{K} + \mathcal{H}(x)$, i.e., if there exist $|k\rangle \in \mathcal{K}$ and $|h\rangle \in \mathcal{H}(x)$ such that $|w_0\rangle = |k\rangle + |h\rangle$.
2. $x \in \mathcal{D}$ is a *negative input*, if the projection of $|w_0\rangle$ onto $\mathcal{K}^\perp \cap \mathcal{H}(x)^\perp$ is non-zero.

We let $f : \mathcal{D} \to \{0,1\}$ with $f(x) = 1$ if and only if $x$ is a positive input of $\mathcal{P}$, and we say that $\mathcal{P}$ computes $f$. We say that $\mathcal{P}$ is constant if the function it computes is constant. ◀

The definition presented here differs subtly from definitions used in previous works. Most notably, in all prior works one of the ingredients of a span program is a linear operator $A$ called the *span program operator*, mapping the Hilbert space $\mathcal{H}$ into some auxiliary vector space $\mathcal{V}$ [Rei09; Jef14; IJ19; BT20]. The analysis presented in these works then crucially depends on the null-space of this operator, i.e., $\mathrm{Ker}(A) \subseteq \mathcal{H}$. Our input-independent subspace $\mathcal{K} \subseteq \mathcal{H}$ plays the same role as $\mathrm{Ker}(A)$ in previous works, hence the choice to use the letter $\mathcal{K}$ to denote this subspace. Using the space $\mathcal{K}$ directly instead of having to define the span program operator $A$ removes the necessity of using an auxiliary space $\mathcal{V}$, and thus simplifies the exposition and proofs that accompany the span program formalism.

Furthermore, in previous works it was always assumed that the domain $\mathcal{D}$ is a subset of strings containing characters from some alphabet, i.e., $\mathcal{D} \subseteq S^n$, where $S$ is some finite set of symbols. The subspaces $\mathcal{H}(x)$ were then required to be decomposable into $n$ mutually orthogonal subspaces, each related to one symbol from the input string $x \in S^n$ [Rei09; Jef14; IJ19; BT20]. However, the analysis of most of the results within the span program formalism goes through without this assumption, so we will only explicitly require this assumption when we need to.

In order to build some intuition for the span program definition, Figure 6.1.1 provides a visual depiction of the components that make up a span program, and showcases the difference between positive and negative inputs. It is a simple linear-algebraic exercise to show that $(\mathcal{K} + \mathcal{H}(x))^\perp = \mathcal{K}^\perp \cap \mathcal{H}(x)^\perp$. Thus, we can easily observe from the picture that $x$ is indeed either a positive or a negative input.

The visualization in Figure 6.1.1 already gives us a first rough idea of how a quantum algorithm can distinguish between positive and negative inputs. Suppose one starts in the state $|w_0\rangle$, and then iteratively applies two reflections, first through $\mathcal{H}(x)$ and then through $\mathcal{K}$. If $x$ is a positive input, then the visualization suggests that the entire vector $|w_0\rangle$ starts rotating, whereas if $x$ is a negative input, a non-zero component of $|w_0\rangle$ remains stationary. It is this dichotomy between positive and negative inputs that the span program algorithm exploits, as we will see in Section 6.1.3.

Furthermore, as is apparent from Definition 6.1.1 as well as from Figure 6.1.1, span programs naturally encode a decision problem, i.e., there is a natural distinction between positive and negative inputs. There exist generalizations of span programs to functions with more than two outcomes, most notably in [LMR+11; BT20].

Informally speaking, some inputs are more positive or negative than others,

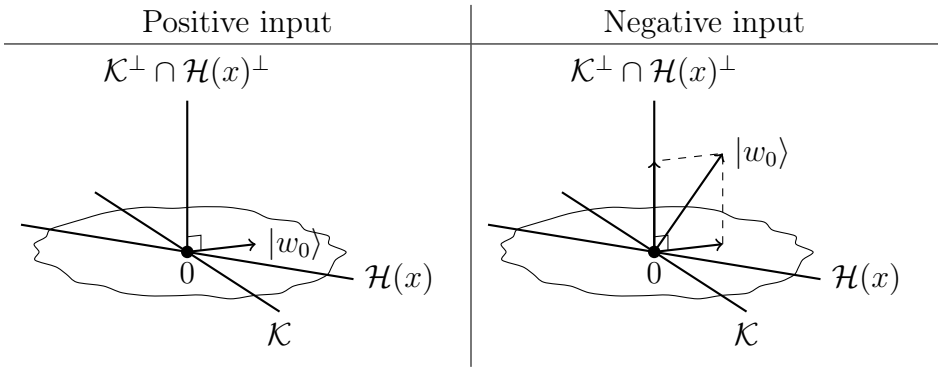



Figure 6.1.1: The difference between positive and negative inputs. If $x$ is a positive input, $|w_0\rangle \in \mathcal{K} + \mathcal{H}(x)$, i.e., $|w_0\rangle$ lies in the ground plane of the picture on the left-hand side. Alternatively, if $x$ is a negative input, then $|w_0\rangle$ has a non-trivial overlap with $\mathcal{K}^\perp \cap \mathcal{H}(x)^\perp$, and $|w_0\rangle$ sticks out of the ground plane in the picture on the right-hand side.

as we can intuitively see in Figure 6.1.1. For instance in the negative case, if the overlap of $|w_0\rangle$ with the stationary subspace $\mathcal{K}^\perp \cap \mathcal{H}(x)^\perp$ is very small, then $|w_0\rangle$ is very close to the ground plane, and hence we can say that the input is "almost" positive. Similarly, in the positive case, if $\mathcal{K}$ and $\mathcal{H}(x)$ are very close together, i.e., the angle between them is very small, then the rotation of $|w_0\rangle$ will be very slow, and thus we can say that it is "almost" stationary. With this intuition, then, we can say that $x$ is very close to being a negative input.

The objects we introduce next, dubbed witnesses, serve a dual purpose. First, they can be used as a certificate for either positivity or negativity, i.e., exhibiting a witness can be used to prove that a particular input is either positive or negative. Second, analyzing the size of these witnesses can be used to formally quantify the above intuition about "how positive" or "how negative" the input is.

**6.1.2.** Definition (Span program witnesses). Let $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ be a span program on $\mathcal{D}$, and let $x \in \mathcal{D}$.

1. Suppose that $|w\rangle \in \mathcal{H}(x)$, and $|w\rangle - |w_0\rangle \in \mathcal{K}$. Then, we refer to $|w\rangle$ as a *positive witness for* $x$. We refer to $\||w\rangle\|^2$ as the *size* of the witness. For every input $x$ for which at least one positive witness exists, we call the unique positive witness for $x$ with smallest size the *minimal positive witness for* $x$. Consequently, the *minimal positive witness size for* $x$ is defined as

$$w_+(x, \mathcal{P}) = \min\{\||w\rangle\|^2 : |w\rangle \text{ is a positive witness for } x\}. \tag{6.1.1}$$

2. Suppose that $|w\rangle \in \mathcal{K}^\perp \cap \mathcal{H}(x)^\perp$ such that $\langle w_0|w\rangle = 1$. Then, we refer to $|w\rangle$ as a *negative witness for* $x$, and we refer to $\||w\rangle\|^2$ as its *size*. For every input $x$ for which at least one negative witness exists, we call the unique

negative witness for $x$ with smallest size the *minimal negative witness for* $x$. Consequently, the *minimal negative witness size for* $x$ is defined as

$$w_-(x, \mathcal{P}) = \min\{\||w\rangle\|^2 : |w\rangle \text{ is a negative witness for } x\}. \tag{6.1.2}$$

In both cases above, we use the convention that the minimum of an empty set is $\infty$. Finally, we define the *positive* and *negative witness complexity of* $\mathcal{P}$, respectively, as

$$W_+(\mathcal{P}) = \max_{\substack{x \in \mathcal{D} \\ x \text{ positive}}} w_+(x, \mathcal{P}), \qquad \text{and} \qquad W_-(\mathcal{P}) = \max_{\substack{x \in \mathcal{D} \\ x \text{ negative}}} w_-(x, \mathcal{P}),$$

where we use the convention that the maximum of an empty set is 0. We define the *span program complexity* of $\mathcal{P}$ as $C(\mathcal{P}) = \sqrt{W_+(\mathcal{P})W_-(\mathcal{P})}$. ◀

In order to check the well-definedness of the above definition, observe that both in the positive and the negative case, the constraints on the vectors $|w\rangle$ are linear, and thus the sets of positive and negative witnesses form affine subspaces of $\mathcal{H}$. Since $\mathcal{H}$ is finite-dimensional, these sets are closed, and since taking the squared norm is continuous, both sets on the right-hand side of Equations (6.1.1) and (6.1.2) are closed as well. Thus, the minima are well-defined, and are attained at the unique vectors in the affine subspaces.

Observe that the above well-definedness crucially relies on the Hilbert space $\mathcal{H}$ having finite dimension. It might be possible to generalize Definition 6.1.1 to allow for infinite-dimensional Hilbert spaces, but it is not clear what benefits that would bring. Therefore, we will restrict ourselves to the finite-dimensional case in this text.

We can add these newly-defined objects to our visualization in Figure 6.1.1, which yields Figure 6.1.2. On the left-hand side, we can see how the existence of a positive witness certifies that $|w_0\rangle$ is indeed in the ground plane, and hence that $x$ is a positive input. Similarly, on the right-hand side, we observe that the existence of a negative witness implies that $|w_0\rangle$ is indeed not in the ground plane, and hence $x$ must be a negative input.

Furthermore, from Figure 6.1.2, we can now build some intuition on how the size of the witnesses plays a role in quantifying "how positive" or "how negative" a particular input is. Indeed, in the positive case, if the angle between $\mathcal{K}$ and $\mathcal{H}(x)$ is very small, then the size of $|w\rangle$ will be very large. Thus, intuitively speaking, bigger positive witnesses indicate more difficult positive instances. Similarly, in the negative case, if $|w_0\rangle$ has a very small overlap with the stationary subspace $\mathcal{K}^\perp \cap \mathcal{H}(x)^\perp$, then the vector $|w\rangle$ becomes very long. Thus, a higher negative witness size indicates a more difficult negative instance.

We proceed by formally proving some basic properties of the positive and negative witnesses, in the following lemma. We also formalize some of the intuitions introduced in the previous paragraphs.

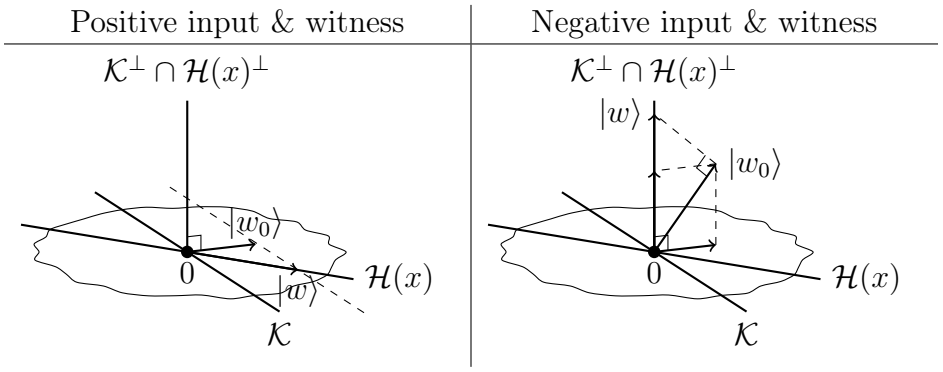



Figure 6.1.2: Graphical depiction of the positive and negative witnesses. Positive witnesses are vectors in $\mathcal{H}(x)$ that can be reached from $|w_0\rangle$ by adding an element from $\mathcal{K}$, as can be seen in the picture on the left-hand side. Negative witnesses are vectors in $|w\rangle \in \mathcal{K}^\perp \cap \mathcal{H}(x)^\perp$, such that the difference vector $|w\rangle - |w_0\rangle$ is orthogonal to $|w_0\rangle$ itself, as is shown on the right-hand side.

**6.1.3.** Lemma (Basic properties of span program witnesses).
*Let $\mathcal{P}$ be a span program on $\mathcal{D}$, and let $x \in \mathcal{D}$. Then,*

1. $w_+(x,\mathcal{P}) \geq 1$ *and* $w_-(x,\mathcal{P}) \geq 1$.
2. $w_+(x,\mathcal{P}) < \infty \quad \Leftrightarrow \quad x$ *is positive* $\quad \Leftrightarrow \quad w_-(x,\mathcal{P}) = \infty$.
3. $w_+(x,\mathcal{P}) = \infty \quad \Leftrightarrow \quad x$ *is negative* $\quad \Leftrightarrow \quad w_-(x,\mathcal{P}) < \infty$.
4. $W_+(\mathcal{P})$, $W_-(\mathcal{P})$ *and* $C(\mathcal{P})$ *are all finite.*
5. *If $\mathcal{P}$ is non-constant, then $W_+(\mathcal{P})$, $W_-(\mathcal{P})$ and $C(\mathcal{P})$ are all at least* 1. *If $\mathcal{P}$ is constant, then $C(\mathcal{P}) = 0$.*

**Proof:**
For the first claim, suppose that $|w\rangle$ is the minimal positive witness for $x$. Then,

$$w_+(x,\mathcal{P}) = \||w\rangle\|^2 = \||w\rangle - |w_0\rangle\|^2 + \||w_0\rangle\|^2 \geq 0 + 1 = 1.$$

Similarly, suppose that $|w\rangle$ is the minimal negative witness for $x$. Then,

$$w_-(x,\mathcal{P}) = \||w\rangle\|^2 = \||w\rangle\|^2 \, \||w_0\rangle\|^2 \geq |\langle w|w_0\rangle|^2 = 1,$$

where we used the Cauchy–Schwarz inequality. This completes the proof of the first claim.

For the second claim, we supply a list of equivalences that proves the left-hand equivalence, as follows:

$$\begin{aligned} w_+(x,\mathcal{P}) < \infty &\Leftrightarrow \min\{\||w\rangle\|^2 : |w\rangle \in \mathcal{H}(x) \cap (|w_0\rangle + \mathcal{K})\} < \infty \\ &\Leftrightarrow \mathcal{H}(x) \cap (|w_0\rangle + \mathcal{K}) \neq \varnothing \\ &\Leftrightarrow \exists\, |w\rangle \in \mathcal{H}(x), |k\rangle \in \mathcal{K} : |w_0\rangle = |w\rangle + |k\rangle \\ &\Leftrightarrow |w_0\rangle \in \mathcal{K} + \mathcal{H}(x) \\ &\Leftrightarrow x \text{ is a positive input.} \end{aligned}$$

Similarly, for the right-hand equivalence, we have

$$\begin{aligned} w_-(x,\mathcal{P}) = \infty &\Leftrightarrow \min\{\||w\rangle\|^2 : |w\rangle \in \mathcal{K}^\perp \cap \mathcal{H}(x)^\perp, \langle w|w_0\rangle = 1\} = \infty \\ &\Leftrightarrow \{|w\rangle \in \mathcal{K}^\perp \cap \mathcal{H}(x)^\perp, \langle w|w_0\rangle = 1\} = \varnothing \\ &\Leftrightarrow \forall\, |w\rangle \in \mathcal{K}^\perp \cap \mathcal{H}(x)^\perp, \langle w|w_0\rangle = 0, \\ &\Leftrightarrow |w_0\rangle \in (\mathcal{K}^\perp \cap \mathcal{H}(x)^\perp)^\perp \\ &\Leftrightarrow |w_0\rangle \in \mathcal{K} + \mathcal{H}(x) \\ &\Leftrightarrow x \text{ is a positive input.} \end{aligned}$$

In the third equivalence, we used that $\mathcal{K}^\perp \cap \mathcal{H}(x)^\perp$ is a linear subspace. Indeed, if any $|w\rangle \in \mathcal{K}^\perp \cap \mathcal{H}(x)^\perp$ satisfies $\langle w|w_0\rangle = c \neq 0$, then $|w'\rangle := |w\rangle / c \in \mathcal{K}^\perp \cap \mathcal{H}(x)^\perp$ satisfies $\langle w'|w_0\rangle = \langle w|w_0\rangle / c = 1$. The contrapositive provides the $\Rightarrow$-part of the third equivalence above. This completes the proof of the second claim.

The third claim is simply the negation of the second claim, and therefore follows easily. The fourth claim also follows easily from claims 2 and 3, and the definition of the objects $W_+(\mathcal{P})$, $W_-(\mathcal{P})$, and $C(\mathcal{P})$ in Definition 6.1.1. Finally, for the fifth claim, we observe that if $\mathcal{P}$ is non-constant, then there exists at least one positive and negative input, and thus the positive and negative witness complexity are at least 1. Similarly, if $\mathcal{P}$ is constant, then the positive or negative witness complexity is 0, and thus so is $C(\mathcal{P})$. This completes the proof. □

We end this introductory section with the construction of some particular span programs, which we will use as running examples through the rest of this chapter. The reader is encouraged to verify the claimed properties of these span programs in the statements below.

**6.1.4.** Example (Identity span program).
Let $\mathcal{D} = \{0,1\}$. Let $\mathcal{H} = \mathrm{Span}\{|*\rangle\}, \mathcal{K} = \{0\}, |w_0\rangle = |*\rangle$, and for all $x \in \mathcal{D}$,

$$\mathcal{H}(x) = \begin{cases} \mathcal{H}, & \text{if } x = 1, \\ \{0\}, & \text{if } x = 0. \end{cases}$$

Then $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ is a span program that evaluates the identity function $f : \mathcal{D} \to \{0,1\}$ with $f(x) = x$, and $W_+(\mathcal{P}) = W_-(\mathcal{P}) = C(\mathcal{P}) = 1$. ◁

**6.1.5.** Example (Span program evaluating the AND-function).
Let $\mathcal{D} = \{0,1\}^n$. Let $\mathcal{H} = \mathbb{C}^n$, $\mathcal{K} = \{0\}$, for all $x \in \mathcal{D}$, $\mathcal{H}(x) = \mathrm{Span}\{|j\rangle : x_j = 1\}$, and

$$|w_0\rangle = \frac{1}{\sqrt{n}} \sum_{j=1}^{n} |j\rangle .$$

Then $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ is a span program that evaluates the AND-function on $n$ bits, and for all $x \in \mathcal{D}$,

$$w_+(x,\mathcal{P}) = \begin{cases} \infty, & \text{if } |x| < n, \\ 1, & \text{if } |x| = n, \end{cases} \qquad \text{and} \qquad w_-(x,\mathcal{P}) = \begin{cases} \frac{n}{n-|x|}, & \text{if } |x| < n, \\ \infty, & \text{if } |x| = n, \end{cases}$$

which implies that $W_+(\mathcal{P}) = 1$, $W_-(\mathcal{P}) = n$ and $C(\mathcal{P}) = \sqrt{n}$. ◁

**6.1.6.** Example (Span program evaluating the OR-function).
Let $\mathcal{D} = \{0,1\}^n$. Let $\mathcal{H} = \mathbb{C}^n$, for all $x \in \mathcal{D}$, $\mathcal{H}(x) = \mathrm{Span}\{|j\rangle : x_j = 1\}$,

$$|w_0\rangle = \frac{1}{\sqrt{n}} \sum_{j=1}^{n} |j\rangle ,$$

and $\mathcal{K} = \mathrm{Span}\{|w_0\rangle\}^\perp$. Then, $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ is a span program that evaluates the OR-function on $n$ bits, and for all $x \in \mathcal{D}$,

$$w_+(x,\mathcal{P}) = \begin{cases} \infty, & \text{if } |x| = 0, \\ \frac{n}{|x|}, & \text{if } |x| \geq 1, \end{cases} \qquad \text{and} \qquad w_-(x,\mathcal{P}) = \begin{cases} 1, & \text{if } |x| = 0, \\ \infty, & \text{if } |x| \geq 1, \end{cases}$$

which implies that $W_+(\mathcal{P}) = n$, $W_-(\mathcal{P}) = 1$ and $C(\mathcal{P}) = \sqrt{n}$. ◁

This concludes the definition of span programs and their witnesses. At this point, we have only introduced some vague intuitive idea about the hardness of instances, and how it relates to the size of the corresponding witnesses. The following subsection develops this idea further, and formalizes it into an *operational interpretation* of the witness sizes.

## 6.1.2 Operational interpretation

In the previous subsection we already hinted at the existence of a quantum algorithm that distinguishes between positive and negative inputs of a span program. In this subsection, we further develop this idea, and prove the core theoretical result that makes this algorithm possible. The main result of this section is Theorem 6.1.13, which provides an operational interpretation of the witnesses defined in Definition 6.1.2.

We start by introducing a purely linear-algebraic concept, which we refer to as an *inverse projection*.

**6.1.7.** Definition (Inverse projection). Let $\mathcal{H}$ be a Hilbert space, with a linear subspace $A \subseteq \mathcal{H}$. Let $|\psi\rangle \in \mathcal{H}$, such that $|\psi\rangle \notin A^\perp$. We define the *inverse projection of* $|\psi\rangle$ *onto* $A$ as

$$\Pi_A^{(-1)}(|\psi\rangle) = \frac{\||\psi\rangle\|^2}{\|\Pi_A |\psi\rangle\|^2} \cdot \Pi_A |\psi\rangle .$$

We refer to $\Pi_A^{(-1)} : \mathcal{H} \setminus A^\perp \to A$ as the *inverse projective map onto* $A$. ◀

Since we assumed that $|\psi\rangle \notin A^\perp$, we have $\Pi_A |\psi\rangle \neq 0$, and thus $\Pi_A^{(-1)}(|\psi\rangle)$ is well-defined. Moreover, $\Pi_A^{(-1)}$ is not a linear operator, which can already be observed from the fact that it is only well-defined on the set $\mathcal{H} \setminus A^\perp$, which is not a linear subspace of $\mathcal{H}$. To emphasize that the inverse projective map onto $A$ is merely a function, and not a linear operator, we write the argument of the function in parentheses behind $\Pi_A^{(-1)}$, i.e., we write $\Pi_A^{(-1)}(|\psi\rangle)$ rather than $\Pi_A^{(-1)} |\psi\rangle$.

Next, we prove a variational characterization of inverse projections. The statement of the following lemma already indicates a connection to the definition of the witnesses, in Definition 6.1.2.

**6.1.8.** Lemma (Variational characterization of inverse projections). *Let $\mathcal{H}$ be a Hilbert space, with a linear subspace $A \subseteq \mathcal{H}$. Let $|\psi\rangle \in \mathcal{H}$, such that $|\psi\rangle \notin A^\perp$. Then,*

$$\||\psi\rangle\|^2 \, \|\Pi_A |\psi\rangle\|^{-1} = \|\Pi_A^{(-1)}(|\psi\rangle)\| = \min\{\||\phi\rangle\| : |\phi\rangle \in A, \langle\phi|\psi\rangle = \||\psi\rangle\|^2\},$$

*and $|\phi\rangle = \Pi_A^{(-1)}(|\psi\rangle)$ is the unique vector attaining the minimum on the right-hand side.*

Before we give a formal proof of Lemma 6.1.8, we provide a slightly naive geometric depiction of inverse projections in Figure 6.1.3. In the picture, we can easily deduce that the inner product between $\Pi_A^{(-1)}(|\psi\rangle)$ and $|\psi\rangle$ is equal to the inner product of $|\psi\rangle$ with itself, which is equal to $\||\psi\rangle\|^2$. We also have two similar triangles, i.e., $\triangle OQP \sim \triangle OPR$. Thus, using elementary geometry, we find $OP/OQ = OR/OP$, and so

$$\||\psi\rangle\|^2 \, \|\Pi_A |\psi\rangle\|^{-1} = \frac{OP^2}{OQ} = OP \cdot \frac{OP}{OQ} = OP \cdot \frac{OR}{OP} = OR = \|\Pi_A^{(-1)}(|\psi\rangle)\|.$$

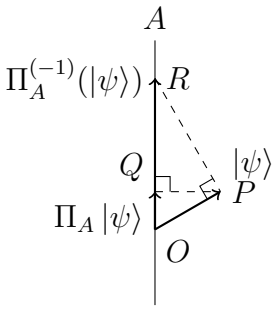



Figure 6.1.3: Graphical depiction of the inverse projection.

We now proceed by formalizing this proof sketch of Lemma 6.1.8.

**Proof of Lemma 6.1.8:**

The left equality follows directly from the definition of the inverse projection, so

we focus on the right equality. To that end, for any vector $|\phi\rangle$ in the set on the right-hand side, we have

$$\||\psi\rangle\|^2 = \langle\phi|\psi\rangle = \langle\phi|\,\Pi_A\,|\psi\rangle + \langle\phi|\,\Pi_{A^\perp}\,|\psi\rangle = \langle\phi|\,\Pi_A\,|\psi\rangle\,,$$

where in the last equality, the last term vanishes because $|\phi\rangle \in A$ by definition. Thus, requiring that $\langle\phi|\psi\rangle = \||\psi\rangle\|^2$ is equivalent to imposing the constraint that $\langle\phi|\,\Pi_A\,|\psi\rangle = \||\psi\rangle\|^2$. We can directly verify that one such vector $|\phi\rangle$ is indeed $\Pi_A^{(-1)}(|\psi\rangle)$, and it follows that all vectors $|\phi\rangle$ are in the affine subspace

$$\Pi_A^{(-1)}(|\psi\rangle) + (A \cap \mathrm{Span}\{\Pi_A\,|\psi\rangle\}^\perp).$$

Since $\Pi_A^{(-1)}(|\psi\rangle) \in \mathrm{Span}\{\Pi_A\,|\psi\rangle\}$, it is orthogonal to all vectors in the subspace $A \cap \mathrm{Span}\{\Pi_A\,|\psi\rangle\}^\perp$. Thus, the unique shortest vector in the above affine subspace is indeed $\Pi_A^{(-1)}(|\psi\rangle)$ itself, completing the proof. □

With the variational characterization in mind, we can now prove a connection between the inverse projection and the witnesses of a span program.

**6.1.9.** Lemma. *Let $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ be a span program on $\mathcal{D}$, and let $x \in \mathcal{D}$ be an input.*

1. *If $x$ is a negative input for $\mathcal{P}$, then the minimal negative witness for $x$ is the vector $\Pi^{(-1)}_{\mathcal{K}^\perp \cap \mathcal{H}(x)^\perp}(|w_0\rangle)$.*
2. *If $x$ is a positive input for $\mathcal{P}$, then the minimal positive witness for $x$ is the vector $\Pi^{(-1)}_{\mathcal{H}(x)\cap(\mathcal{K}\oplus\mathrm{Span}\{|w_0\rangle\})}(|w_0\rangle)$.*

**Proof:**
The first claim follows directly from the variational characterization of inverse projections, i.e., Lemma 6.1.8, the definition of negative witnesses in Definition 6.1.2, and the property that $\||w_0\rangle\| = 1$, as defined in Definition 6.1.1.

For the positive case, using that $|w_0\rangle \in \mathcal{K}^\perp$ and $\||w_0\rangle\| = 1$, we observe that for any vector $|w\rangle \in \mathcal{H}$, we have

$$\begin{aligned} &|w\rangle \text{ is a positive witness for } x \\ &\quad\Leftrightarrow |w\rangle \in \mathcal{H}(x) \wedge |w\rangle - |w_0\rangle \in \mathcal{K} \\ &\quad\Leftrightarrow |w\rangle \in \mathcal{H}(x) \cap (\mathcal{K} \oplus \mathrm{Span}\{|w_0\rangle\}) \wedge \langle w|w_0\rangle = 1. \end{aligned}$$

Then, the second claim also follows from the variational characterization of inverse projections, i.e., Lemma 6.1.8. This completes the proof. □

Now that we have introduced the concept of inverse projections, we proceed by developing the technique we employ to differentiate between positive and negative inputs. The core idea is to alternate reflections through $\mathcal{H}(x)$ and $\mathcal{K}$. We formally define the unitary that performs these reflections consecutively, and we take a look at its eigendecomposition.

**6.1.10.** Definition (Span program unitary and ideal phase variable).
Let $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ be a span program on $\mathcal{D}$, and let $x \in \mathcal{D}$. We define the *span program unitary* for input $x$ as

$$U(x, \mathcal{P}) = (2\Pi_{\mathcal{K}} - I)(2\Pi_{\mathcal{H}(x)} - I).$$

Next, let $S = \{\phi \in (-1/2, 1/2] : e^{2\pi i\phi} \in \sigma(U(x, \mathcal{P}))\}$, where $\sigma(U(x, \mathcal{P}))$ denotes the spectrum of $U(x, \mathcal{P})$. We write $E_\phi = \{|\psi\rangle \in \mathcal{H} : U(x, \mathcal{P}) |\psi\rangle = e^{2\pi i\phi} |\psi\rangle\}$, for all values $\phi \in (-1/2, 1/2]$, i.e., $E_\phi = \{0\}$ when $\phi \notin S$. The eigendecomposition of the span program unitary becomes

$$U(x, \mathcal{P}) = \sum_{\phi \in S} e^{2\pi i\phi} \Pi_{E_\phi}.$$

Finally, we define the random variable $\Phi$, referred to as the *ideal phase variable on input* $x$ as

$$\mathbb{P}[\Phi = \phi] = \left\| \Pi_{E_\phi} |w_0\rangle \right\|^2.$$

We refer to $S$ as the *support of* $\Phi$, and denote it by $\mathrm{supp}(\Phi)$. ◀

First, we check that $\Phi$ is well-defined. Since $U(x, \mathcal{P})$ is acting on a finite-dimensional Hilbert space, $S$ is finite, and thus $\Phi$ is a discrete random variable with support on $S$. Moreover, the eigenspaces $E_\phi$, with $\phi \in S$ form an orthogonal decomposition of the Hilbert space $\mathcal{H}$, and thus if we sum all the probabilities $\mathbb{P}[\Phi = \phi]$ with $\phi \in S$, we obtain $\||w_0\rangle\|^2$, which is 1 by definition. Thus, $\Phi$ is indeed a well-defined random variable.

Note that the span program unitary $U(x, \mathcal{P})$ depends on the input $x$, and hence so do the eigenspaces $E_\phi$ and, in particular, the ideal phase variable $\Phi$. Therefore, perhaps the notation $\Phi(x, \mathcal{P})$ would be more apt, but it has the downside of making the expressions more cumbersome later on. Thus, in the remainder of this text, we will leave the ideal phase variable's dependence on the input $x$ implicit.

Since the span program unitary is a product of two reflections, we take a closer look at the properties of such unitaries. To that end, we recall a classic linear-algebraic result known as Jordan's lemma [Jor75]. A more modern statement of this lemma and a proof sketch can also be found in [Sze04, Theorem 1]. For our purposes, however, we need some more subtle consequences of Jordan's lemma than those proved in [Sze04], and therefore we provide a self-contained proof here.

**6.1.11.** Lemma (Jordan's lemma). *Let $\mathcal{H}$ be a finite-dimensional Hilbert space, and let $A, B \subseteq \mathcal{H}$ be linear subspaces. Let $U = (2\Pi_B - I)(2\Pi_A - I)$. For every $\phi \in (-1/2, 1/2]$, let $E_\phi = \{|\psi\rangle \in \mathcal{H} : U |\psi\rangle = e^{2\pi i\phi} |\psi\rangle\}$. Then, we can find a $k \in \mathbb{N}$, such that for all $j \in [k-1]$, we can find a two-dimensional subspace $R_j \subseteq \mathcal{H}$ and an angle $\theta_j \in (0, \pi)$, such that the following properties hold:*

1. *The* 1*-eigenspace of* $U$ *is* $R_0 := E_0 = (A \cap B) \oplus (A^\perp \cap B^\perp)$. *We let* $\theta_0 = 0$.
2. *The* $-1$*-eigenspace of* $U$ *is* $R_k := E_{1/2} = (A^\perp \cap B) \oplus (A \cap B^\perp)$. *We let* $\theta_k = \pi$.
3. *For all* $j \in [k]_0$, $U$ *leaves* $R_j$ *invariant and it acts as a rotation over angle* $\theta_j$ *on it.*
4. *Let* $\phi \in (0, 1/2)$. *We have the decompositions*

$$\mathcal{H} = \bigoplus_{j=0}^{k} R_j, \qquad E_\phi \oplus E_{-\phi} = \bigoplus_{\substack{j=1 \\ \theta_j = 2\pi\phi}}^{k-1} R_j,$$

$$A = \bigoplus_{j=0}^{k} (A \cap R_j), \qquad B = \bigoplus_{j=0}^{k} (B \cap R_j),$$

*and similarly for* $A^\perp$ *and* $B^\perp$. *Moreover, for all* $j \in [k-1]$, *we can decompose* $R_j$ *into sets of one-dimensional components, i.e.,*

$$R_j = (A \cap R_j) \oplus (A^\perp \cap R_j) = (B \cap R_j) \oplus (B^\perp \cap R_j).$$

*The angle between* $A \cap R_j$ *and* $B \cap R_j$ *is* $\theta_j/2$, *and their intersection is* $\{0\}$.

**Proof:**
We consider the operator $\Pi_B \Pi_A$, and write its singular value decomposition as

$$\Pi_B \Pi_A = \sum_{j=1}^{r} s_j \left|\psi_j\right\rangle \left\langle \phi_j \right|,$$

where we order the terms such that $0 < s_1 \leq \cdots \leq s_r \leq 1$. Next, we define

1. $k = |\{j \in [r] : s_j \neq 1\}| + 1$.
2. For all $j \in [k-1]$, we define the subspace $R_j = \mathrm{Span}\{\left|\psi_j\right\rangle, \left|\phi_j\right\rangle\}$.
3. For all $j \in [k-1]$, we let $\theta_j = 2 \arccos(s_j) \in (0, \pi)$.

We now check that all claims from the lemma statement are satisfied. For the first claim, observe that we have, for all $\left|\psi\right\rangle \in \mathcal{H}$,

$$\begin{aligned}
&(2\Pi_B - I)(2\Pi_A - I)\left|\psi\right\rangle = \left|\psi\right\rangle \Leftrightarrow 4\Pi_B\Pi_A\left|\psi\right\rangle - 2\Pi_A\left|\psi\right\rangle - 2\Pi_B\left|\psi\right\rangle + \left|\psi\right\rangle = \left|\psi\right\rangle \\
&\quad \Leftrightarrow \Pi_B(\Pi_A - I)\left|\psi\right\rangle + (I - \Pi_B)\Pi_A\left|\psi\right\rangle = 0 \Leftrightarrow \Pi_B\Pi_{A^\perp}\left|\psi\right\rangle = -\Pi_{B^\perp}\Pi_A\left|\psi\right\rangle \\
&\quad \Leftrightarrow \Pi_B\Pi_{A^\perp}\left|\psi\right\rangle = \Pi_{B^\perp}\Pi_A\left|\psi\right\rangle = 0.
\end{aligned}$$

Now, observe that $\Pi_B\Pi_{A^\perp}\left|\psi\right\rangle = 0$ is equivalent to either $\left|\psi\right\rangle \in A$, or $0 \neq \Pi_A\left|\psi\right\rangle \in B^\perp$. This can be concisely rephrased to $\left|\psi\right\rangle \in A \oplus (A^\perp \cap B^\perp)$. Similarly $\Pi_{B^\perp}\Pi_A\left|\psi\right\rangle = 0$ is equivalent to $\left|\psi\right\rangle \in A^\perp \oplus (A \cap B)$. Thus,

$$\begin{aligned}
&U\left|\psi\right\rangle = \left|\psi\right\rangle \\
&\quad \Leftrightarrow \left|\psi\right\rangle \in (A \oplus (A^\perp \cap B^\perp)) \cap (A^\perp \oplus (A \cap B)) = (A \cap B) \oplus (A^\perp \cap B^\perp).
\end{aligned}$$

This proves the first claim. Next, observe that $(2\Pi_A - I)(2\Pi_{B^\perp} - I) = -U$, and thus we obtain the second claim from the first one by substituting $B$ for $B^\perp$.

Now, for all $j \in [k-1]$, we have $|\phi_j\rangle \in A$ and $|\psi_j\rangle \in B$, and so

$$\begin{aligned}
&(2\Pi_B - I)(2\Pi_A - I)\,|\phi_j\rangle = (2\Pi_B - I)\,|\phi_j\rangle = 2\Pi_B\Pi_A\,|\phi_j\rangle - |\phi_i\rangle \\
&\quad = 2s_j\,|\psi_j\rangle - |\phi_j\rangle\,, \\
&(2\Pi_B - I)(2\Pi_A - I)\,|\psi_j\rangle = (2\Pi_B - I)(2\Pi_A\Pi_B\,|\psi_j\rangle - |\psi_j\rangle) \\
&\quad = (2\Pi_B - I)(2s_j\,|\phi_j\rangle - |\psi_j\rangle) = 4\Pi_B\Pi_A\,|\phi_j\rangle - 2s_j\,|\phi_j\rangle - |\psi_j\rangle \\
&\quad = (4s_j^2 - 1)\,|\psi_j\rangle - 2s_j\,|\phi_j\rangle\,.
\end{aligned}$$

This implies that

$$\begin{aligned}
&U(|\psi_j\rangle - e^{\pm i\theta_j/2}\,|\phi_j\rangle) = (4s_j^2 - 1 - 2e^{\pm i\theta_j/2}s_j)\,|\psi_j\rangle - (2s_j - e^{\pm i\theta_j/2})\,|\phi_j\rangle \\
&= \left[4\left(\frac{e^{i\theta_j/2} + e^{-i\theta_j/2}}{2}\right)^2 - 1 - 2e^{\pm i\theta_j/2}\left(\frac{e^{i\theta_j/2} + e^{-i\theta_j/2}}{2}\right)\right]|\psi_j\rangle \\
&\quad - \left[2\left(\frac{e^{i\theta_j/2} + e^{-i\theta_j/2}}{2}\right) - e^{\pm i\theta_j/2}\right]|\phi_j\rangle \\
&= \left[2 + e^{i\theta_j} + e^{-i\theta_j} - 1 - 1 - e^{\pm i\theta_j}\right]|\psi_j\rangle - e^{\mp i\theta_j/2}\,|\phi_j\rangle \\
&= e^{\mp i\theta_j}\,|\psi_j\rangle - e^{\mp i\theta_j/2}\,|\phi_j\rangle = e^{\mp i\theta_j}\left[|\psi_j\rangle - e^{\pm i\theta_j/2}\,|\phi_j\rangle\right].
\end{aligned}$$

Thus, for all $j \in [k-1]$, we have found two eigenvectors of $U$ in $R_j$, with eigenvalues $e^{\pm i\theta_j}$. Therefore, indeed $U$ acts as a rotation on $R_j$ over angle $\theta_j$. This proves claim 3.

Next, we observe that

$$\begin{aligned}
\sum_{j=1}^{r} s_j^3\,|\phi_j\rangle\langle\psi_j| &= (\Pi_B\Pi_A)(\Pi_B\Pi_A)^\dagger\Pi_B\Pi_A = \Pi_B\Pi_A\Pi_B\Pi_A = (\Pi_B\Pi_A)^2 \\
&= \sum_{j,k=1}^{r} s_j s_k\,|\psi_j\rangle\langle\phi_j|\psi_k\rangle\langle\phi_k|\,.
\end{aligned}$$

Thus, for all $j, k \in [r]$, we obtain that $\langle\phi_j|\psi_k\rangle = s_j\delta_{j,k}$. Since $\langle\phi_j|\phi_k\rangle = \langle\psi_j|\psi_k\rangle = \delta_{j,k}$ by the properties of the singular value decomposition, we deduce that $R_j$'s are mutually orthogonal.

Next, extend the right singular vectors $\{|\phi_1\rangle, \ldots, |\phi_r\rangle\}$ of $\Pi_B\Pi_A$ to a basis $\{|\phi_1\rangle, \ldots, |\phi_n\rangle\}$ for $A$. Observe that now $\{|\psi_k\rangle, \ldots, |\psi_r\rangle\}$ is a basis for $A \cap B$,

and $\{|\psi_{r+1}\rangle, \ldots, |\psi_n\rangle\}$ is a basis for $A \cap B^\perp$. Moreover,

$$\begin{aligned}\Pi_{B^\perp}\Pi_A = \Pi_A - \Pi_B\Pi_A &= \sum_{j=1}^{n} |\phi_j\rangle\langle\phi_j| - \sum_{j=1}^{r} s_j |\psi_j\rangle\langle\phi_j| \\ &= \sum_{j=1}^{k-1} (|\phi_j\rangle - s_j|\psi_j\rangle)\langle\phi_j| + \sum_{j=r+1}^{n} |\phi_j\rangle\langle\phi_j| \\ &= \sum_{j=1}^{k-1} \sqrt{1-s_j^2}\frac{|\phi_j\rangle - s_j|\psi_j\rangle}{\sqrt{1-s_j^2}}\langle\phi_j| + \sum_{j=r+1}^{n} |\phi_j\rangle\langle\phi_j|,\end{aligned}$$

where the final expression is a singular value decomposition of $\Pi_{B^\perp}\Pi_A$. We can also express $\Pi_B\Pi_{A^\perp}$ and $\Pi_{A^\perp}\Pi_{B^\perp}$ in a similar way. All of the resulting expressions start with $k-1$ terms, each of which acts on the rotation space $R_j$ with $j \in [k-1]$, and then add a projection onto the intersection of the two spaces projected upon. Since $\Pi_A\Pi_B + \Pi_{A^\perp}\Pi_B + \Pi_A\Pi_{B^\perp} + \Pi_{A^\perp}\Pi_{B^\perp} = I$, no non-zero element of $\mathcal{H}$ is in the kernel of all four operators, and thus we find that $\mathcal{H}$ can be decomposed into the spaces $R_j$, for $j \in [k]_0$.

Next, for all $\phi \in (0, 1/2)$, we know that the eigenspaces $E_\phi$ and $E_{-\phi}$ must be orthogonal to $R_0$ and $R_k$, since eigenspaces with different eigenvalues are orthogonal to one another. Therefore, we have that $E_\phi \oplus E_{-\phi}$ must be contained in the vector sum of the spaces $R_j$, where $j \in [k-1]$. Since every space $R_j$ is spanned by two eigenvectors of $U$ with a conjugate pair of eigenvalues, these spaces must coincide and we find the expression from the lemma statement.

Using the basis we constructed for $A$, the decomposition of $A$ from the lemma statement follows directly. Using similar constructions of bases, the same decomposition results can be shown for $B$, $A^\perp$ and $B^\perp$. Next, observe that $\mathrm{Span}\{|\phi_j\rangle\} = A \cap R_j$, and $\mathrm{Span}\{|\phi_j\rangle - s_j|\psi_j\rangle\} = A^\perp \cap R_j$. Similarly, we can find a decomposition of $R_j$ into one-dimensional spaces $B \cap R_j$ and $B^\perp \cap R_j$. Finally, observe that $\langle\phi_j|\psi_j\rangle = s_j$, which implies that the angle between the two subspaces $A \cap R_j$ and $B \cap F_j$ is $\arccos(s_j) = \theta_j/2$. Finally, $s_j \neq 0$, and so $(A \cap R_j) \cap (B \cap R_j) = \{0\}$. □

Note that the angles $\theta_1, \ldots, \theta_{k-1}$ in Jordan's lemma need not be distinct, i.e., there might be several rotation spaces $R_j$ that rotate over the same angle. In that case, the choice of these rotation spaces is not unique. Furthermore, observe that the spaces $R_1, \ldots, R_{k-1}$ are all two-dimensional, but this is not necessarily the case for $R_0$ and $R_k$. These could even be the trivial subspace, $\{0\}$.

We now use Jordan's lemma to understand the span program operator, as defined in Definition 6.1.10. To that end, recall that we defined the span program unitary as $U(x, \mathcal{P}) = (2\Pi_{\mathcal{K}} - I)(2\Pi_{\mathcal{H}(x)} - I)$, and so if we choose $B = \mathcal{K}$ and $A = \mathcal{H}(x)$, then we can apply Jordan's lemma to this unitary. Thus, using the notation from Lemma 6.1.11, we observe that the span program unitary decomposes $\mathcal{H}$ into $k+1$ subspaces $R_0, \ldots, R_k$, and for each $j \in [k]_0$, the operator acts on $R_j$ as a

rotation over angle $\theta_j \in [0, \pi]$. Moreover, $\mathcal{K}$ and $\mathcal{H}(x)$ can be decomposed into the subspaces $\mathcal{K} \cap R_j$ and $\mathcal{H}(x) \cap R_j$, respectively, where $j$ ranges from 0 to $k$. A visualization of these decompositions is presented in Figure 6.1.4.

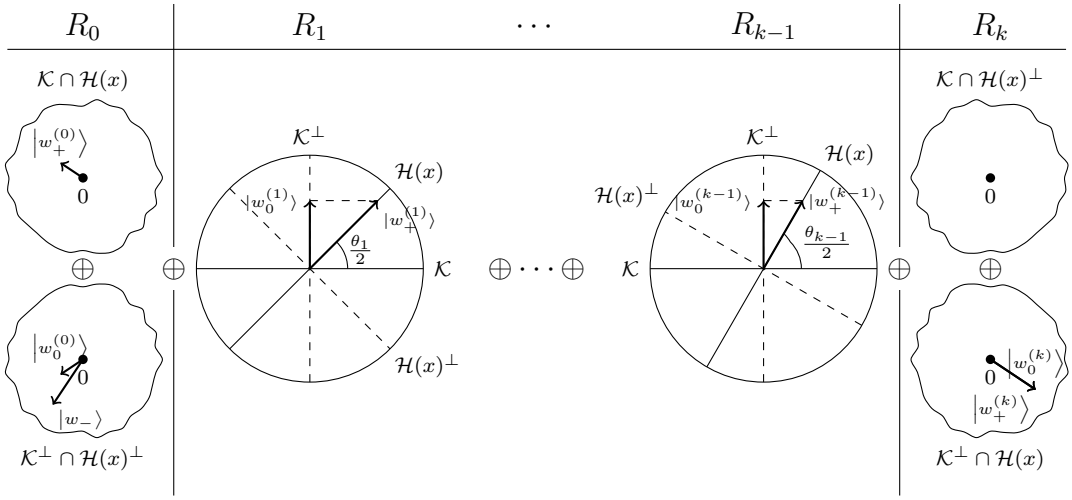



Figure 6.1.4: Visualization of $\mathcal{H}$, $\mathcal{K}$, $\mathcal{H}(x)$, $|w_0\rangle$, a positive witness $|w_+\rangle$ and a negative witness $|w_-\rangle$, in light of the decompositions provided by Jordan's lemma.

Next, we investigate how the initial state $|w_0\rangle$, and the negative and positive witnesses can be visualized in Figure 6.1.4. To that end, observe that $|w_0\rangle \in \mathcal{K}^\perp$. Since Jordan's lemma tells us that we can decompose $\mathcal{K}^\perp$ into the subspaces $\mathcal{K}^\perp \cap R_j$ for $j \in [k]_0$, we can decompose $|w_0\rangle$ into its components $|w_0^{(j)}\rangle \in \mathcal{K}^\perp \cap R_j$. Additionally, we observe by Definition 6.1.1 that $|w_0^{(0)}\rangle$ is zero if and only if the input $x$ is a positive input. Additionally, every negative witness $|w_-\rangle$ lives in $\mathcal{K}^\perp \cap \mathcal{H}(x)^\perp$, which is a subspace of $R_0$, as can be seen in Figure 6.1.4.

Furthermore, every positive witness $|w_+\rangle$ is an element of $\mathcal{H}(x)$, and as such can be decomposed into its components $|w_+^{(j)}\rangle \in \mathcal{H}(x) \cap R_j$. Moreover, $|w_+\rangle - |w_0\rangle \in \mathcal{K}$ by the definition of positive witnesses, and therefore we know that in each of the subspaces $R_j$, $|w_+^{(j)}\rangle - |w_0^{(j)}\rangle \in \mathcal{K}$. We can see from the visualization, i.e., Figure 6.1.4, that there is only one possible choice for $|w_+^{(j)}\rangle$ for all $j \in [k]$, which is the inverse projection of $|w_0^{(j)}\rangle$ onto $\mathcal{H}(x) \cap R_j$. Hence, any freedom in choosing positive witnesses relies on the choice of $|w_+^{(0)}\rangle$, which can be any vector in $\mathcal{K} \cap \mathcal{H}(x)$. It then follows immediately that we minimize the size of a positive witness by choosing $|w_+^{(0)}\rangle = 0$.

Finally, observe that for all $j \in [k-1]$, the angle between the subspace $\mathcal{K} \cap R_j$ and $\mathcal{H}(x) \cap R_j$ provide an interesting relation between the norms of $|w_0^{(j)}\rangle$ and $|w_+^{(j)}\rangle$. Indeed, we can see using Figure 6.1.4 and some elementary geometry that

$$\sin(\theta_j/2) = \frac{\||w_0^{(j)}\rangle\|}{\||w_+^{(j)}\rangle\|}.$$

These observations give us a handle on expressing the size of the minimal witnesses in terms of the projections of $|w_0\rangle$ on the subspaces $R_j$. This is the objective of the following lemma.

**6.1.12.** LEMMA. *Let $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ be a span program on $\mathcal{D}$, and let $x \in \mathcal{D}$ be an input. Let $R_0, \ldots, R_k$ and $\theta_0, \ldots, \theta_k \in [0, \pi]$ be as in Jordan's lemma, i.e., Lemma 6.1.11, applied to the span program operator $U(x, \mathcal{P})$. For all $j \in [k]_0$, let $|w_0^{(j)}\rangle = \Pi_{R_j} |w_0\rangle$. Then,*

1. *Suppose $x$ is a negative input. Then $|w_0^{(0)}\rangle \neq 0$, the minimal negative witness is exactly $\Pi_{R_0}^{(-1)}(|w_0\rangle)$, and its size is*

$$w_-(x, \mathcal{P}) = \frac{1}{\||w_0^{(0)}\rangle\|^2}.$$

2. *Suppose $x$ is a positive input. Then $|w_0^{(0)}\rangle = 0$. Let $|w_+\rangle \in \mathcal{H}$, and for all $j \in [k]_0$, let $|w_+^{(j)}\rangle = \Pi_{R_j} |w_+\rangle$. Then, $|w_+\rangle$ is a positive witness if and only if $|w_+^{(0)}\rangle \in \mathcal{K} \cap \mathcal{H}(x)$, and for all $j \in [k]$,*

$$|w_+^{(j)}\rangle = \Pi_{R_j \cap \mathcal{H}(x)}^{(-1)}(|w_0^{(j)}\rangle).$$

*Moreover, the minimal positive witness satisfies $|w_+^{(0)}\rangle = 0$, and its size is*

$$w_+(x, \mathcal{P}) = \sum_{j=1}^{k} \frac{\||w_0^{(j)}\rangle\|^2}{\sin^2(\theta_j/2)}.$$

**Proof:**
We start by proving claim 1. To that end, suppose that $x$ is a negative input. From Definition 6.1.1, we obtain that $|w_0^{(0)}\rangle \neq 0$. Next, we use Jordan's lemma to express $R_0 = (\mathcal{K} \cap \mathcal{H}(x)) \oplus (\mathcal{K}^\perp \cap \mathcal{H}(x)^\perp)$. Since $|w_0\rangle \in \mathcal{K}^\perp$, we find the simplification $\Pi_{R_0} |w_0\rangle = \Pi_{\mathcal{K}^\perp \cap \mathcal{H}(x)^\perp} |w_0\rangle$. Thus,

$$\begin{aligned}\Pi_{R_0}^{(-1)}(|w_0\rangle) &= \frac{1}{\|\Pi_{R_0} |w_0\rangle\|^2} \Pi_{R_0}(|w_0\rangle) = \frac{1}{\left\|\Pi_{K^\perp \cap \mathcal{H}(x)^\perp} |w_0\rangle\right\|^2} \Pi_{K^\perp \cap \mathcal{H}(x)^\perp} |w_0\rangle \\ &= \Pi_{\mathcal{K}^\perp \cap \mathcal{H}(x)^\perp}^{(-1)}(|w_0\rangle),\end{aligned}$$

which by Lemma 6.1.9 is the minimal negative witness. It then follows from Lemma 6.1.8 that $w_-(x, \mathcal{P}) = \|\Pi_{R_0}^{(-1)}(|w_0\rangle)\|^2 = 1/\|\Pi_{R_0} |w_0\rangle\|^2 = 1/\||w_0^{(0)}\rangle\|^2$, proving claim 1.

For the second claim, suppose that $x$ is a positive input. Then, from Definition 6.1.1 we immediately obtain that $|w_0^{(0)}\rangle = 0$, and we have the following

sequence of equivalences:

$$\begin{aligned}
&|w_+\rangle \text{ is a positive witness for } x \\
&\quad \Leftrightarrow |w_+\rangle \in \mathcal{H}(x) \wedge |w_+\rangle - |w_0\rangle \in \mathcal{K} \\
&\quad \Leftrightarrow \forall j \in [k]_0, |w_+^{(j)}\rangle \in \mathcal{H}(x) \cap R_j \wedge |w_+^{(j)}\rangle - |w_0^{(j)}\rangle \in \mathcal{K} \cap R_j \\
&\quad \Leftrightarrow |w_+^{(0)}\rangle \in (\mathcal{H}(x) \cap R_0) \cap (\mathcal{K} \cap R_0) = \mathcal{K} \cap \mathcal{H}(x) \\
&\qquad \wedge \forall j \in [k], |w_+^{(j)}\rangle \in \mathcal{H}(x) \cap R_j \wedge \langle w_+^{(j)} | w_0^{(j)}\rangle = \| |w_0^{(j)}\rangle \|^2,
\end{aligned}$$

where in the final step, we used that $\mathcal{K} \cap R_j$ is one-dimensional, and hence equal to $\text{Span}\{|w_0^{(j)}\rangle\}^\perp \cap R_j$. From Lemma 6.1.8, we obtain that for $j \in [k]$, the inverse projection $\Pi^{(-1)}_{\mathcal{H}(x)\cap R_j}(|w_0^{(j)}\rangle)$ is a vector that satisfies these criteria. Furthermore, any other vector that satisfies these criteria, must also satisfy the relation $|w_+^{(j)}\rangle - \Pi^{(-1)}_{\mathcal{H}(x)\cap R_j}(|w_0^{(j)}\rangle) \in (\mathcal{H}(x) \cap R_j) \cap (\mathcal{K} \cap R_j)$. From Lemma 6.1.11, we obtain that this subspace is $\{0\}$ for all $j \in [k]$, and thus we obtain that the inverse projection is the only possible choice for $|w_+^{(j)}\rangle$. This completes the proof for the characterization of the positive witnesses.

The norm of such a positive witness $|w_+\rangle$ is

$$\| |w_+\rangle \|^2 = \sum_{j=0}^{k} \| |w_+^{(j)}\rangle \|^2 = \| |w_+^{(0)}\rangle \|^2 + \sum_{j=1}^{k} \| \Pi^{(-1)}_{\mathcal{H}(x)\cap R_j}(|w_0^{(j)}\rangle) \|^2.$$

It is clear that this norm is minimized when we choose $|w_+^{(0)}\rangle = 0$, and using Lemma 6.1.8, the resulting expression then becomes

$$w_+(x, \mathcal{P}) = \sum_{j=1}^{k} \frac{\| |w_0^{(j)}\rangle \|^4}{\| \Pi_{\mathcal{H}(x)\cap R_j} |w_0^{(j)}\rangle \|^2} = \sum_{j=1}^{k} \frac{\| |w_0^{(j)}\rangle \|^2}{\sin^2(\theta_j/2)}.$$

In the last equality, we used that $|w_0^{(j)}\rangle \in \mathcal{K}^\perp \cap R_j$, and that for all $j \in [k-1]$, the angle between $\mathcal{K} \cap R_j$ and $\mathcal{H}(x) \cap R_j$ is $\theta_j/2$ by Lemma 6.1.11. On the other hand, when $j = k$, we have that $|w_0^{(k)}\rangle \in \mathcal{K}^\perp \cap \mathcal{H}(x)$, and thus we easily verify that the same relation holds. This completes the proof. □

The final observation in this section is that the size of $|w_0^{(j)}\rangle = \Pi_{R_j} |w_0\rangle$ is related to the probability of the ideal phase variable $\Phi$ attaining the value $\theta_j/(2\pi)$. More precisely, we can think of the ideal phase variable as the result of the following randomized procedure: we select a subspace $R_j$ with probability $\| \Pi_{R_j} |w_0\rangle \|^2$, and then output the $\theta_j/(2\pi)$. With this connection in mind, we can translate the expressions for the witness sizes derived in Lemma 6.1.12, as statistical properties of $\Phi$. This is the objective of the following theorem.

**6.1.13.** THEOREM (Operational interpretation of the witness sizes).
*Let $\mathcal{P}$ be a span program, and $\Phi$ its ideal phase variable on input $x$. Then,*

$$w_-(x,\mathcal{P}) = \frac{1}{\mathbb{P}\left[\Phi = 0\right]}, \qquad \textit{and} \qquad w_+(x,\mathcal{P}) = \mathbb{E}\left[\frac{1}{\sin^2\left(\pi\Phi\right)}\right].$$

**Proof:**
Let $U(x,\mathcal{P})$ be the span program unitary, and decompose $\mathcal{H}$ by applying Jordan's lemma, i.e., Lemma 6.1.11, to the span program unitary $U(x,\mathcal{P})$. That is, let the subspaces $R_0,\ldots,R_k$ be such that for all $j \in [k]_0$, $U(x,\mathcal{P})$ acts on $R_j$ as a rotation over angle $\theta_j \in [0,\pi]$, where $0 = \theta_0 < \theta_1 \leq \cdots \leq \theta_{k-1} < \theta_k = \pi$. As before, we write $|w_0^{(j)}\rangle = \Pi_{R_j}|w_0\rangle$, for all $j \in [k]_0$.

We first focus on the claim on the left-hand side. To that end, observe that

$$\mathbb{P}\left[\Phi = 0\right] = \|\Pi_{E_0}|w_0\rangle\|^2 = \|\Pi_{R_0}|w_0\rangle\|^2 = \||w_0^{(0)}\rangle\|^2.$$

If $x$ is a positive input, then $\mathbb{P}[\Phi = 0]$ evaluates to 0, and indeed we have that $w_-(x,\mathcal{P}) = \infty$ by Lemma 6.1.3. If $x$ is a negative input, then the equation follows directly from claim 1 in Lemma 6.1.12.

We now focus on the claim on the right-hand side. Suppose that $x$ is a negative input. Then, $\mathbb{P}[\Phi = 0] \neq 0$, and thus the expectation evaluates to $\infty$. From Lemma 6.1.3, we obtain that $w_+(x,\mathcal{P}) = \infty$ too, proving the validity of the equation in the negative case.

Thus, it remains to focus on the situation where $x$ is a positive input. Then, $\mathbb{P}[\Phi = 0] = 0$, and so

$$\mathbb{E}\left[\frac{1}{\sin^2(\pi\Phi)}\right] = \sum_{\phi\in\mathrm{supp}(\Phi)} \frac{\mathbb{P}[\Phi=\phi]}{\sin^2(\pi\Phi)} = \sum_{\substack{\phi\in\mathrm{supp}(\Phi)\\ \phi>0}} \frac{\|\Pi_{E_\phi\oplus E_{-\phi}}|w_0\rangle\|^2}{\sin^2(\pi\phi)}.$$

From Lemma 6.1.11, we know that we can decompose $E_\phi \oplus E_{-\phi}$ into the rotation spaces $R_j$ that satisfy $\phi = 2\pi\theta_j$. Thus, we rewrite the right-hand side to

$$\mathbb{E}\left[\frac{1}{\sin^2(\pi\Phi)}\right] = \sum_{j=1}^{k} \frac{\|\Pi_{R_j}|w_0\rangle\|^2}{\sin^2(\theta_j/2)} = \sum_{j=1}^{k} \frac{\||w_0^{(j)}\rangle\|^2}{\sin^2(\theta_j/2)} = w_+(x,\mathcal{P}),$$

where we used claim 2 from Lemma 6.1.12 to conclude the final equality. This completes the proof. □

On a broader level, we can interpret Theorem 6.1.13 as a connection between a purely linear-algebraic object, i.e., the witnesses, and an operational object, i.e., the ideal phase variable. The result we presented here is the minimum depth in which we must understand this connection in order to develop the span program algorithm.

We remark here that the above intuitive picture directly leads to other interesting characterizations as well, e.g., it is an instructive exercise for the reader to show that the following relations hold:

$$\left\|\Pi_{\mathcal{H}(x)}\left|w_0\right\rangle\right\|^2 = \mathbb{E}[\sin^2(\pi\Phi)], \qquad \left\|\Pi_{\mathcal{K}}\Pi_{\mathcal{H}(x)}\left|w_0\right\rangle\right\|^2 = \mathbb{E}[\cos^2(\pi\Phi)\sin^2(\pi\Phi)], \tag{6.1.3a}$$

$$\left\|\Pi_{\mathcal{H}(x)^\perp}\left|w_0\right\rangle\right\|^2 = \mathbb{E}[\cos^2(\pi\Phi)], \quad \left\|\Pi_{\mathcal{K}^\perp}\Pi_{\mathcal{H}(x)^\perp}\left|w_0\right\rangle\right\|^2 = \mathbb{E}[\cos^4(\pi\Phi)]. \tag{6.1.3b}$$

In Section 7.1.3, we delve more deeply into the connections between the linear-algebraic decomposition of $|w_0\rangle$ and the operational interpretation via the ideal phase variable $\Phi$.

Operationally, we can interpret the ideal phase variable $\Phi$ as the outcome of an ideal run of phase estimation, i.e., a run of phase estimation where the precision is infinite. Hence, the size of the witnesses tell us something about the outcome probabilities of an ideal run of phase estimation. This is ultimately the fundamental idea that enables us to develop a quantum algorithm that distinguishes between positive and negative inputs, which is the objective of the next subsection.

### 6.1.3 Span program algorithm

In the previous subsection, we have seen how the size of the witnesses can be interpreted in terms of the ideal phase variable. Operationally, the ideal phase variable is the outcome of phase estimation if we were to run it with infinite precision.

In any explicit quantum algorithm, however, we can only run phase estimation with finite precision, which means that truncation errors come into play. In the next theorem we analyze these finite precision effects in more detail.

**6.1.14.** Theorem (Operational interpretation with finite precision). *Let $\mathcal{P}$ be a span program, and $\Phi$ its ideal phase variable on input $x$. Next, let $k \in \mathbb{N}$ and let $\Phi_k$ be the outcome of a run of phase estimation with $k \in \mathbb{N}$ bits of precision, unitary $U(x,\mathcal{P})$ and initial state $|w_0\rangle$. Then,*

$$\frac{1}{w_-(x,\mathcal{P})} = \mathbb{P}\left[\Phi = 0\right] \leq \mathbb{P}\left[\Phi_k = 0\right] \leq \frac{1}{2^{2k}}\mathbb{E}\left[\frac{1}{\sin^2(\pi\Phi)}\right] = \frac{w_+(x,\mathcal{P})}{2^{2k}}.$$

**Proof:**
First, we decompose $|w_0\rangle$ into the eigenbasis of $U(x,\mathcal{P})$, i.e., we write

$$|w_0\rangle = \sum_{\phi\in\mathrm{supp}(\Phi)} \alpha_\phi |w_0^{(\phi)}\rangle, \qquad \text{where} \qquad U(x,\mathcal{P})|w_0^{(\phi)}\rangle = e^{2\pi i\phi}|w_0^{(\phi)}\rangle.$$

Now, recall the phase estimation algorithm, Algorithm 2.4.3. We observe from its analysis that the probability of obtaining the measurement outcome 0 is

$$\mathbb{P}\left[\Phi_k = 0\right] = \sum_{\phi \in \mathrm{supp}(\Phi)} |\alpha_\phi|^2 \frac{\sin^2(\pi 2^k \phi)}{2^{2k} \sin^2(\pi\phi)},$$

where we can take limits if the denominator equates to 0. Now, by the definition of the ideal phase variable in Definition 6.1.10, for all $\phi \in \mathrm{supp}(\Phi)$, we have $|\alpha_\phi|^2 = \left\|\Pi_{E_\phi} |w_0\rangle\right\|^2 = \mathbb{P}[\Phi = \phi]$. Thus, the above expression simplifies to

$$\mathbb{P}\left[\Phi_k = 0\right] = \mathbb{E}\left[\frac{\sin^2(\pi 2^k \Phi)}{2^{2k} \sin^2(\pi\Phi)}\right]. \tag{6.1.4}$$

Since if $\Phi = 0$, the argument of the expectation evaluates to 1, we obtain that $\mathbb{P}[\Phi_k = 0] \geq \mathbb{P}[\Phi = 0]$. On the other hand, we use that the numerator is always upper bounded by 1 to obtain that $\mathbb{P}[\Phi_k = 0] \leq \mathbb{E}[\sin^{-2}(\pi\Phi)]/2^{2k}$. This completes the proof of both the inequalities. The equalities on the outside both follow from Theorem 6.1.13. This completes the proof. □

Now that we have analyzed the outcome probabilities of phase estimation on the span program unitary, we can make a first attempt at designing an algorithm that distinguishes the positive and negative inputs. The result is Algorithm 6.1.15.

**Algorithm 6.1.15:** The naive span program algorithm

**Input:**

1: $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w - 0\rangle)$: a span program on $\mathcal{D}$.
2: $\overline{W}_+$: an upper bound on $W_+(\mathcal{P})$.
3: $\overline{W}_-$: an upper bound on $W_-(\mathcal{P})$.
4: $C_{|w_0\rangle}$: a quantum circuit acting on $\mathcal{H}$ that implements $|0\rangle \mapsto |w_0\rangle$.
5: $U$: a quantum circuit acting on $\mathcal{H}$ that implements $U(x, \mathcal{P})$.

**Derived objects:**

1: $k = \lceil \log(\sqrt{\overline{W}_+ \overline{W}_-}) \rceil + 1/2$.

**Output:** 1 if $x$ is positive for $\mathcal{P}$, 0 otherwise.

**Success probability:** Lower bounded by $\min\{1/\overline{W}_-, 1 - 1/(2\overline{W}_-)\}$.

**Queries:**

1: Number of calls to $C_{|w_0\rangle}$: 1
2: Number of calls to $U$: $\mathcal{O}(\sqrt{\overline{W}_+ \overline{W}_-})$.

**Procedure:** Naive-Span($\mathcal{P}$, $\overline{W}_+$, $\overline{W}_-$, $C_{|w_0\rangle}$, $U$):

1: Run Phase-Est($k$, $U$, $C_{|w_0\rangle}$). Denote the outcome by $\Phi_k$.
2: If $\Phi_k = 0$, output 0. Else output 1.

**Proof of properties of Algorithm 6.1.15:**

The claims about the number of queries all follow directly from the properties of

phase estimation, as stated in Algorithm 2.4.3. Therefore, we are left with checking the claimed success probability. To that end, observe from Theorem 6.1.14 that if $x$ is negative, then

$$\mathbb{P}\left[\Phi_k = 0\right] \geq \mathbb{P}\left[\Phi = 0\right] = \frac{1}{w_-(x,\mathcal{P})} \geq \frac{1}{W_-(\mathcal{P})} \geq \frac{1}{\overline{W}_-}.$$

On the other hand, if $x$ is positive, then

$$\mathbb{P}[\Phi_k = 0] \leq \frac{w_+(x,\mathcal{P})}{2^{2k}} \leq \frac{W_+(\mathcal{P})}{2\overline{W}_+\overline{W}_-} \leq \frac{1}{2\overline{W}_-},$$

where we used the value of $k$ as stated in the algorithm description. This completes the proof. □

Note that the number of calls to $U(x,\mathcal{P})$ is what we expect – if we know the witness complexities $W_-(\mathcal{P})$ and $W_+(\mathcal{P})$ exactly, then we can design a quantum algorithm that makes $\mathcal{O}(\sqrt{W_+(\mathcal{P})W_-(\mathcal{P})}) = \mathcal{O}(C(\mathcal{P}))$ controlled queries to $U(x,\mathcal{P})$. Thus, the span program complexity indeed coincides with the number of calls made to $U(x,\mathcal{P})$ in the naive span program algorithm.

However, the success probability is not quite what we expected. If $W_-(\mathcal{P})$ is big, then the best attainable success probability $1/W_-(\mathcal{P})$ can indeed become very small.

There are several approaches to deal with this problem. The easiest option is to simply run the naive span program algorithm a couple of times to gather statistics on the outcome being 0 or 1. From these statistics one could then try to infer whether the probability of obtaining outcome 1 is bigger than $1/\overline{W}_-$, or smaller than $1/(2\overline{W}_-)$. This can be done using $\mathcal{O}(\overline{W}_-)$ iterations, bringing the total number of queries to $U(x,\mathcal{P})$ to $\mathcal{O}(\overline{W}_-\sqrt{\overline{W}_-\overline{W}_+})$.

A slightly more involved approach is to run amplitude estimation on the output of the naive span program algorithm being 0. With this approach, one would attain a multiplicative overhead of only $\mathcal{O}(\sqrt{\overline{W}_-})$, bringing the total number of queries to $U(x,\mathcal{P})$ to $\mathcal{O}(\overline{W}_-\sqrt{\overline{W}_+})$.

However, there exists a way to circumvent having any overhead at all, obtaining an algorithm with success probability at least $2/3$, and a mere $\mathcal{O}(\sqrt{\overline{W}_-\overline{W}_+})$ controlled calls to $U(x,\mathcal{P})$. The fundamental idea is to use *span program renormalization*, which is a technique that already exists in several different forms in previous literature. The core idea is to simultaneously decrease $W_-(\mathcal{P})$ and increase $W_+(\mathcal{P})$, in such a way that their product stays the same up to constants. Here, we follow the exposition in [IJ19], and give slight improvements of both the construction and the analysis.

**6.1.16.** Definition (Span program renormalization).
Let $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ be a span program on $\mathcal{D}$. We let $|*\rangle$ be a unit

vector orthogonal all vectors in $\mathcal{H}$, and $\beta > 0$. Then, with $x \in \mathcal{D}$, we define

$$\mathcal{H}' = \mathcal{H} \oplus \mathrm{Span}\{|*\rangle\}, \qquad \mathcal{H}'(x) = \mathcal{H}(x),$$
$$\mathcal{K}' = \mathcal{K} \oplus \mathrm{Span}\{|w_0\rangle - \beta\,|*\rangle\}, \qquad |w_0'\rangle = \frac{\beta\,|w_0\rangle + |*\rangle}{\sqrt{1+\beta^2}}.$$

Then the span program $\mathcal{P}' = (\mathcal{H}', x \mapsto \mathcal{H}'(x), \mathcal{K}', |w_0'\rangle)$ is $\mathcal{P}$ *renormalized with parameter* $\beta$. ◀

Note that this construction is a direct simplification of [IJ19, Theorem 2.14], since we require one dimension less.

Next, we investigate how the witnesses of the original span program are related to those in the renormalized version. This is the objective of the following theorem.

**6.1.17.** Theorem (Witness anatomy of renormalized span programs). *Let $\mathcal{P}$ be a span program, and let $\mathcal{P}'$ be $\mathcal{P}$ renormalized with parameter $\beta > 0$. Then,*

1. *$\mathcal{P}$ and $\mathcal{P}'$ compute the same function.*
2. *$|w\rangle$ is a positive witness for $x$ in $\mathcal{P} \Leftrightarrow (\sqrt{1+\beta^2}/\beta)\,|w\rangle$ is a positive witness for $x$ in $\mathcal{P}'$.*
3. *$|w\rangle$ is a negative witness for $x$ in $\mathcal{P} \Leftrightarrow (\beta\,|w\rangle + |*\rangle)/(\sqrt{1+\beta^2})$ is a negative witness for $x$ in $\mathcal{P}'$.*
4. *$w_+(x, \mathcal{P}') = (1 + 1/\beta^2) w_+(x, \mathcal{P})$.*
5. *$w_-(x, \mathcal{P}') = (\beta^2 w_-(x, \mathcal{P}) + 1)/(\beta^2 + 1)$.*
6. *$W_+(\mathcal{P}') = (1 + 1/\beta^2) W_+(\mathcal{P})$.*
7. *If $\mathcal{P}$ has at least one negative input, then $W_-(\mathcal{P}') = (\beta^2 W_-(\mathcal{P}) + 1)/(\beta^2 + 1)$.*
8. *If $\mathcal{P}$ has at least one negative input, then $C(\mathcal{P}')^2 = C(\mathcal{P})^2 + W_+(\mathcal{P})/\beta^2$.*

It is possible to give a direct proof to the above theorem, as is for instance done in [IJ19, Theorem 2.14]. However, such a proof is quite tedious and not so insightful. Instead, we will present several span program composition results, in Section 7.1, which allow for a much more direct and elegant proof of Theorem 6.1.17. We only note here that claims 1 and 4-8 follow directly from claims 2 and 3, the remainder of the proof can be found in Theorem 7.1.8.

Now, we are able to complete the construction of the span program algorithm.

**Algorithm 6.1.18:** The span program algorithm

**Input:**

1: $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$: a span program.
2: $\overline{W}_+$: an upper bound on $W_+(\mathcal{P})$.
3: $\overline{W}_-$: an upper bound on $W_-(\mathcal{P})$.
4: $C_{|w_0\rangle}$: a quantum circuit acting on $\mathcal{H}$ implementing $|0\rangle \mapsto |w_0\rangle$.
5: $U$: a quantum circuit acting on $\mathcal{H}$ implementing $U(x, \mathcal{P})$.

**Derived objects:**
1: $\beta = 1/\sqrt{2\overline{W}_-}$.
2: $\overline{W}'_+ = 3\overline{W}_+\overline{W}_-$.
3: $\mathcal{P}' = (\mathcal{H}', x \mapsto \mathcal{H}'(x), \mathcal{K}', |w_0'\rangle)$: $\mathcal{P}$ renormalized with parameter $\beta$.
4: $R_\beta$: a quantum circuit acting on $\mathcal{H}'$, implementing the mapping

$$|0\rangle \mapsto \frac{\beta|0\rangle + |*\rangle}{\sqrt{1+\beta^2}}, \qquad \text{and} \qquad |*\rangle \mapsto \frac{|0\rangle - \beta|*\rangle}{\sqrt{1+\beta^2}}.$$

5: $C_{|w_0'\rangle}$: the quantum circuit $C_{|w_0\rangle}R_\beta$, acting on $\mathcal{H}'$.
6: $U'$: the quantum circuit $C_{|w_0\rangle}R_\beta(2|*\rangle\langle*| - I)R_\beta^\dagger C_{|w_0\rangle}^\dagger U$, acting on $\mathcal{H}'$.
**Output:** 1 if $x$ is positive for $\mathcal{P}$, 0 otherwise.
**Success probability:** Lower bounded by 2/3.
**Queries:**
1: Number of calls to $U$: $\mathcal{O}(\sqrt{\overline{W}_+\overline{W}_-})$.
2: Number of calls to $C_{|w_0\rangle}$: $\mathcal{O}(\sqrt{\overline{W}_+\overline{W}_-})$.
**Procedure:** SPAN($\mathcal{P}$, $\overline{W}_+$, $\overline{W}_-$, $C_{|w_0\rangle}$, $U$):
1: Run NAIVE-SPAN($\mathcal{P}'$, $\overline{W}'_+$, 3/2, $C_{|w_0'\rangle}$, $U'$).

---

**Proof of the properties of Algorithm 6.1.18:**
We easily verify the claims on the number of queries, hence it remains to prove the lower bound on the success probability of the algorithm. To that end, observe that if the inputs we supply to the naive span program algorithm satisfy the assumptions listed in Algorithm 6.1.15, then we succeed with probability at least

$$\max\left\{\frac{1}{\frac{3}{2}}, 1 - \frac{1}{2\cdot\frac{3}{2}}\right\} = \frac{2}{3}.$$

Thus, it remains to prove that we satisfy the assumptions from Algorithm 6.1.15. To that end, we first prove that $\overline{W}'_+$ and 3/2 are indeed upper bounds for $W_+(\mathcal{P}')$ and $W_-(\mathcal{P}')$, respectively. To that end, observe that by Theorem 6.1.17,

$$W_+(\mathcal{P}') = \left[1 + \frac{1}{\beta^2}\right] W_+(\mathcal{P}) \leq \left[1 + 2\overline{W}_-\right]\overline{W}_+ \leq 3\overline{W}_-\overline{W}_+ = \overline{W}'_+,$$

where we used that $\overline{W}_- \geq 1$ by Lemma 6.1.3. Similarly,

$$W_-(\mathcal{P}') = \frac{\beta^2 W_-(\mathcal{P}) + 1}{\beta^2 + 1} \leq \frac{\frac{1}{2\overline{W}_-}\cdot\overline{W}_- + 1}{\frac{1}{2\overline{W}_-} + 1} \leq \frac{3}{2}.$$

Hence, it remains to check that the routines $C_{|w_0'\rangle}$ and $U(x, \mathcal{P}')$ implement the right operations. To that end, observe that we have

$$C_{|w_0'\rangle}|0\rangle = C_{|w_0\rangle}\frac{\beta|0\rangle + |*\rangle}{\sqrt{1+\beta^2}} = \frac{\beta|w_0\rangle + |*\rangle}{\sqrt{1+\beta^2}} = |w_0'\rangle,$$

where we used the convention that $C_{|w_0\rangle}$ acts as identity on $|*\rangle$. Next, using the same convention, we observe that $U$ acts as identity on $|*\rangle$. Thus, also on $\mathcal{H}'$ it acts as $U(x,\mathcal{P}) = (2\Pi_{\mathcal{K}} - I)(2\Pi_{\mathcal{H}(x)} - I)$. Now, recall that $\mathcal{K}' = \mathcal{K} \oplus \mathrm{Span}\{|w_0\rangle - \beta\,|*\rangle\}$, and thus we can write

$$\begin{aligned} U(x,\mathcal{P}') &= (2\Pi_{\mathcal{K}'} - I)(2\Pi_{\mathcal{H}(x)} - I) \\ &= \left(I - 2\frac{(|w_0\rangle - \beta\,|*\rangle)(|w_0\rangle - \beta\,|*\rangle)}{1+\beta^2}\right)(2\Pi_{\mathcal{K}} - I)(2\Pi_{\mathcal{H}(x)} - I) \\ &= -\left(2\frac{(|w_0\rangle - \beta\,|*\rangle)(|w_0\rangle - \beta\,|*\rangle)}{1+\beta^2} - I\right) U(x,\mathcal{P}). \end{aligned}$$

Thus, up to global phase, it remains to implement a reflection through the state $|\psi\rangle = (|w_0\rangle - \beta\,|*\rangle)/\sqrt{1+\beta^2}$. To that end, we observe that if $C : |*\rangle \mapsto |\psi\rangle$ constructs this state, then $C(2\,|*\rangle\langle*| - I)C^\dagger$ reflects through it. Thus, it remains to check that $C = C_{|w_0\rangle}R_\beta$ indeed prepares $|\psi\rangle$. To that end, observe that

$$C_{|w_0\rangle}R_\beta\,|*\rangle = C_{|w_0\rangle}\frac{|0\rangle - \beta\,|*\rangle}{\sqrt{1+\beta^2}} = \frac{|w_0\rangle - \beta\,|*\rangle}{\sqrt{1+\beta^2}} = |\psi\rangle\,,$$

where we again used the convention that $C_{|w_0\rangle}$ acts as identity on $|*\rangle$. This completes the proof. □

The algorithm presented here makes use of the observation that it is not necessary to run amplitude estimation with the phase estimation routine as the state-preparation circuit, as is for instance proposed in [IJ19]. In fact, the renormalization procedure is sufficient to boost the probability of measuring 0 in the phase estimation part to 2/3 already, so there is no more post-processing required on the outcome of phase estimation.

Now that we have completed the construction of the span program algorithm, we can take a step back and investigate whether improvements are possible. The most obvious question to address in this context is whether the renormalization procedure is really necessary. At a high level it appears to be difficult to answer this question.

One possibility is that one might have to interpret the extra dimension $|*\rangle$ that is added in the renormalization procedure, as an output register. This is similar to the construction in [LMR+11], where the output register in fact has multiple values, which enables their construction to evaluate non-boolean functions. It would be an interesting future research direction to investigate whether the span program renormalization step indeed is related to, or even maybe a special case of the construction in [LMR+11].

## 6.2 Relation to the quantum adversary method

Span programs bear very close connection to the quantum adversary method. In fact, this connection provides a very prominent appeal for investigating span programs in the first place. In this section, we showcase this connection. We first introduce the quantum adversary method and then show how span programs are related to it.

In the literature, there exists a long line of work that attempts to lower bound the quantum query complexity of boolean functions. By and large, there are two main methods to do this. One is known as the polynomial method [BBC+01; ABP19], which has for instance been used to provide tight bounds for the element distinctness problem [AS04]. The other well-known lower bound technique is the quantum adversary method [Amb02; Amb06; HLŠ07; Rei09; Bel15]. It provides a characterization of the quantum query complexity of boolean functions, in terms of a semidefinite program (SDP) referred to as the *adversary bound*. The characterization is tight up to constants.

Feasible solutions to the maximization version, known as the *primal adversary bound*, give rise to lower bounds on the quantum query complexity of boolean functions. Similarly, feasible solutions to the minimization version, known as the *dual adversary bound*, can be used to upper bound the quantum query complexity. Moreover, as we will see in this section, the adversary bound satisfies strong duality, which means that optimal solutions of both versions have the same objective value.

In this section, we first introduce the primal adversary bound, in Section 6.2.1, and we show that it is indeed an SDP. Then, we take the SDP dual to arrive at the dual adversary bound in Section 6.2.2, and we reformulate it slightly. Finally, in Section 6.2.3, we show how feasible solutions to this reformulated dual adversary bound are in one-to-one correspondence with span programs of a specific type.

### 6.2.1 The primal adversary bound

The central objects in the primal adversary bound are *adversary matrices*, so we start by defining these.

**6.2.1.** Definition (Adversary matrix). Let $f : \mathcal{D} \to \{0,1\}$. Then, a Hermitian matrix $\Gamma \in \mathbb{R}^{\mathcal{D}\times\mathcal{D}}$ is called an adversary matrix for $f$, if for all $x, y \in \mathcal{D}$, $\Gamma[x,y] = 0$ whenever $f(x) = f(y)$. ◀

In Figure 6.2.1, one can see a graphical depiction of an adversary matrix.

Adversary matrices have the fundamental property that their spectra are contained in the real line and symmetric around 0. We prove this in a small lemma below.

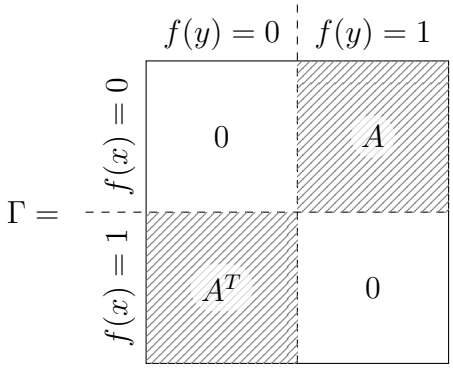



Figure 6.2.1: Graphical depiction of an adversary matrix $\Gamma \in \mathbb{R}^{\mathcal{D}\times\mathcal{D}}$ for a function $f : \mathcal{D} \to \{0,1\}$. Only the hatched areas can have non-zero entries.

**6.2.2.** Lemma (Spectrum of adversary matrices). *Let $f : \mathcal{D} \to \{0,1\}$ be a function, and let $\Gamma$ be an adversary matrix for $f$. Then, the spectrum of $\Gamma$ is contained in the real line, and symmetric around 0.*

**Proof:**
We can characterize the spectrum of $\Gamma$ in terms of the singular values of $A \in \mathbb{R}^{f^{-1}(0)\times f^{-1}(1)}$, in Figure 6.2.1. To that end, write the singular value decomposition of $A$ as

$$A = \sum_{j=1}^{r} \mathbf{u}_j s_j \mathbf{v}_j^T,$$

where $r$ is the rank of $A$, $0 < s_1 \leq \cdots \leq s_r$, and $\{\mathbf{u}_1, \ldots, \mathbf{u}_r\}$ and $\{\mathbf{v}_1, \ldots, \mathbf{v}_r\}$ are orthonormal sets of vectors in $\mathbb{R}^{f^{-1}(0)}$ and $\mathbb{R}^{f^{-1}(1)}$, respectively. Then, we can find $2r$ mutually orthogonal eigenvectors of $\Gamma$ since,

$$\begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_j \\ \mathbf{v}_j \end{bmatrix} = s_j \begin{bmatrix} \mathbf{u}_j \\ \mathbf{v}_j \end{bmatrix}, \qquad \text{and} \qquad \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_j \\ -\mathbf{v}_j \end{bmatrix} = -s_j \begin{bmatrix} \mathbf{u}_j \\ -\mathbf{v}_j \end{bmatrix}.$$

Finally, since rank is subadditive, the rank of $\Gamma$ is at most $2r$. Therefore, we have found all the eigenvectors corresponding to non-zero eigenvalues, and thus we conclude that the spectrum of $\Gamma$ is $\{-s_j, s_j : s_j \in \sigma(A)\}$, which is indeed contained in the real line and symmetric around 0. □

We can now state the primal adversary bound as it is most commonly found in the literature, e.g., in [HLŠ07].

**6.2.3.** Theorem (Primal adversary bound [HLŠ07, Theorem 2]). *Let $f : \mathcal{D} \to \{0,1\}$, with $\mathcal{D} \subseteq \{0,1\}^n$. For all $i \in [n]$, let $\Delta_i \in \{0,1\}^{\mathcal{D}\times\mathcal{D}}$ with*

$$\Delta_i[x,y] = \begin{cases} 1, & \text{if } x_i \neq y_i, \\ 0, & \text{otherwise.} \end{cases}$$

*The adversary bound is defined as*

$$\mathrm{ADV}^{\pm}(f) = \max_{\Gamma \text{ adversary matrix for } f} \frac{\|\Gamma\|}{\max_{j\in[n]} \|\Gamma \circ \Delta_j\|},$$

*where $\circ$ denotes the entry-wise product, or Hadamard product. The query complexity of $f$ is lower bounded by the adversary bound up to constants, i.e., $Q(f) = \Omega(\mathrm{ADV}^{\pm}(f))$.*

The matrices that show up in the denominator of the objective function in Theorem 6.2.3, $\Gamma \circ \Delta_j$, can also be visualized. This is achieved in Figure 6.2.2.

$\Gamma \circ \Delta_j =$

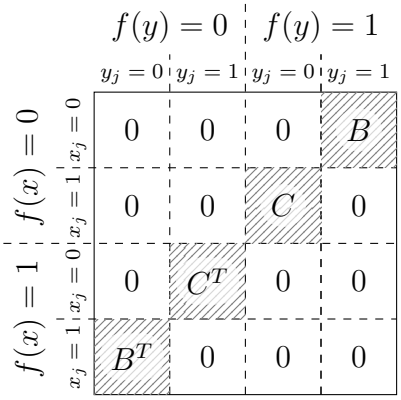



Figure 6.2.2: The matrix that appears in the denominator of the primal adversary bound. Since we take the entry-wise product with a boolean matrix, we are basically setting elements of $\Gamma$ to 0. The hatched blocks in the above visualization are the parts that remain untouched.

Using this visualization, we can now attempt to build an intuitive understanding for the primal adversary bound. To that end, observe that in the denominator, we are selecting some submatrices of $\Gamma$. The adversary matrix $\Gamma$ that maximizes the fraction, minimizes the denominator, and hence computing the adversary bound comes down to finding a $\Gamma$ where all of these submatrices have a norm which is as small possible, while keeping the overall norm of $\Gamma$ large.

It is not immediately clear why the above is a semidefinite program, i.e., how we can construct a semidefinite program that has $\mathrm{ADV}^{\pm}(f)$ as its optimal value. However, explicitly formulating an SDP that computes $\mathrm{ADV}^{\pm}(f)$ is very beneficial, because it enables us to use solvers to find or approximate the optimal adversary matrix $\Gamma$. We use ideas from [Rei09, Theorem 6.2], and reformulate the above optimization problem into a more favorable form for implementation using conventional solvers.

**6.2.4.** Theorem (Reformulated primal adversary bound). *Let $f : \mathcal{D} \to \{0,1\}$, with $\mathcal{D} \subseteq \{0,1\}^n$. Then,* $\mathrm{ADV}^{\pm}(f)$ *is the optimal value of the following SDP:*

$$\begin{aligned}
\max \quad & \sum_{x,y\in\mathcal{D}} \Gamma[x,y], && && \text{(6.2.1a)}\\
s.t. \quad & \mathrm{diag}(\beta) - \Gamma \circ \Delta_j \succeq 0, && \forall j \in [n], && \text{(6.2.1b)}\\
& \Gamma[x,y] = 0, && \forall x,y \in \mathcal{D}, f(x) = f(y), && \text{(6.2.1c)}\\
& \sum_{x\in f^{-1}(1)} \beta[x] = \frac{1}{2}, && && \text{(6.2.1d)}\\
& \sum_{y\in f^{-1}(0)} \beta[y] = \frac{1}{2}, && && \text{(6.2.1e)}
\end{aligned}$$

*where* $\mathrm{diag}(\beta) \in \mathbb{R}^{\mathcal{D}\times\mathcal{D}}$ *denotes the diagonal matrix with diagonal entries given by $\beta$, and the optimization is over all symmetric matrices $\Gamma \in \mathbb{R}^{\mathcal{D}\times\mathcal{D}}$, and entry-wise non-negative vectors $\beta \in \mathbb{R}^{\mathcal{D}}$.*

**Proof:**
First, observe that we can write the original formulation as

$$\begin{aligned}
\max \quad & \|\Gamma\| && \\
\text{s.t.} \quad & \|\Gamma \circ \Delta_j\| \leq 1, && \forall j \in [n],\\
& \Gamma[x,y] = 0, && \forall x,y \in \mathcal{D}, f(x) = f(y),
\end{aligned}$$

where the maximization is over all symmetric matrices $\Gamma \in \mathbb{R}^{\mathcal{D}\times\mathcal{D}}$. Note that by virtue of Lemma 6.2.2, we can change $\|\Gamma\|$ into $\lambda_{\max}(\Gamma)$. Furthermore, note that $\Gamma \circ \Delta_j$ is itself an adversary matrix again, and hence we can also exchange $\|\Gamma \circ \Delta_j\|$ for $\lambda_{\max}(\Gamma \circ \Delta_j)$. Moreover, $\lambda_{\max}(\Gamma \circ \Delta_j) \leq 1$ is equivalent to requiring that $I - \Gamma \circ \Delta_j \succeq 0$. Thus, we obtain that the above optimization problem can be equivalently formulated as

$$\begin{aligned}
\max \quad & \lambda_{\max}(\Gamma) && \\
\text{s.t.} \quad & I - \Gamma \circ \Delta_j \succeq 0, && \forall j \in [n],\\
& \Gamma[x,y] = 0, && \forall x,y \in \mathcal{D}, f(x) = f(y).
\end{aligned}$$

Next, we use that $\lambda_{\max}(\Gamma)$ is the maximum of $\mathbf{v}^T\Gamma\mathbf{v}$ over all vectors $\mathbf{v} \in \mathbb{R}^{\mathcal{D}}$ with $\|\mathbf{v}\| = 1$. Moreover, without loss of generality we can require that $\mathbf{v}$ is entry-wise non-negative, since we can always absorb the signs in $\mathbf{v}$ into $\Gamma$ by multiplying the corresponding rows and columns by $-1$, without changing its objective value or feasibility. But this implies that we can also define an entry-wise non-negative vector $\beta \in \mathbb{R}^{\mathcal{D}}$ such that $\sqrt{\beta[x]} = v[x]$. With this change of variables, the constraint $\|\mathbf{v}\| = 1$ becomes linear in terms of $\beta$, and we are left

with the following formulation of the optimization problem:

$$\begin{aligned}
\max \quad & \sum_{x,y\in\mathcal{D}} \sqrt{\beta[x]}\Gamma[x,y]\sqrt{\beta[y]} && && \text{(6.2.2a)}\\
\text{s.t.} \quad & I - \Gamma \circ \Delta_j \succeq 0, && \forall j \in [n], && \text{(6.2.2b)}\\
& \Gamma[x,y] = 0, && \forall x,y \in \mathcal{D}, f(x) = f(y), && \text{(6.2.2c)}\\
& \sum_{x\in\mathcal{D}} \beta[x] = 1, && && \text{(6.2.2d)}
\end{aligned}$$

where the maximization is over all symmetric matrices $\Gamma \in \mathbb{R}^{\mathcal{D}\times\mathcal{D}}$, and entry-wise non-negative vectors $\beta \in \mathbb{R}^{\mathcal{D}}$. Next, suppose that we have a feasible solution $(\Gamma, \beta)$, and let

$$s = \sum_{x\in f^{-1}(1)} \beta[x], \qquad \text{and} \qquad \beta'[x] = \begin{cases} \frac{1}{2s}\cdot\beta[x], & \text{if } f(x)=1,\\ \frac{1}{2(1-s)}\cdot\beta[x], & \text{if } f(x)=0. \end{cases}.$$

Then,

$$\sum_{x\in\mathcal{D}} \beta'[x] = \frac{1}{2s}\sum_{x\in f^{-1}(1)} \beta[x] + \frac{1}{2(1-s)}\sum_{y\in f^{-1}(0)} \beta[y] = \frac{s}{2s} + \frac{1-s}{2(1-s)} = 1,$$

and so $(\Gamma, \beta')$ is also a feasible solution to Equation (6.2.2). Moreover, since $\Gamma[x,y]$ is only non-zero whenever $f(x) \neq f(y)$, the objective value becomes bigger as

$$\begin{aligned}
\sum_{x,y\in\mathcal{D}} \sqrt{\beta'[x]}\Gamma[x,y]\sqrt{\beta'[y]} &= \frac{1}{2\sqrt{s(1-s)}} \sum_{x,y\in\mathcal{D}} \sqrt{\beta[x]}\Gamma[x,y]\sqrt{\beta[y]},\\
&\geq \sum_{x,y\in\mathcal{D}} \sqrt{\beta[x]}\Gamma[x,y]\sqrt{\beta[y]},
\end{aligned}$$

where we used that $2\sqrt{s(1-s)} \leq 1$ for $s \in [0,1]$. Finally, we observe that

$$\sum_{x\in f^{-1}(1)} \beta'[x] = \frac{1}{2s}\sum_{x\in f^{-1}(1)} \beta[x] = \frac{1}{2} = 1 - \sum_{x\in f^{-1}(1)} \beta'[x] = \sum_{y\in f^{-1}(0)} \beta'[x],$$

and thus we can rewrite the SDP from Equation (6.2.2) into the following equivalent form:

$$\begin{aligned}
\max \quad & \sum_{x,y\in\mathcal{D}} \sqrt{\beta[x]}\Gamma[x,y]\sqrt{\beta[y]} && && \text{(6.2.3a)}\\
\text{s.t.} \quad & I - \Gamma \circ \Delta_j \succeq 0, && \forall j \in [n], && \text{(6.2.3b)}\\
& \Gamma[x,y] = 0, && \forall x,y \in \mathcal{D}, f(x) = f(y), && \text{(6.2.3c)}\\
& \sum_{x\in f^{-1}(1)} \beta[x] = \frac{1}{2}, && && \text{(6.2.3d)}\\
& \sum_{y\in f^{-1}(0)} \beta[y] = \frac{1}{2}. && && \text{(6.2.3e)}
\end{aligned}$$

Next, we show that we can take any feasible solution to the optimization problem in Equation (6.2.3), and turn it into a feasible solution of the optimization problem in Equation (6.2.1). To that end, let $(\Gamma, \beta)$ be a feasible solution to Equation (6.2.3), and let $D \in \mathbb{R}^{\mathcal{D}\times\mathcal{D}}$ be the diagonal matrix where the $[x,x]$-element contains $\sqrt{\beta[x]}$. We define $\Gamma' = D\Gamma D$. Then, for all $\mathbf{v} \in \mathbb{R}^{\mathcal{D}}$, we have

$$\begin{aligned}\mathbf{v}^T(\mathrm{diag}(\beta) - \Gamma' \circ \Delta_j)\mathbf{v} &= \mathbf{v}^T D(I - \Gamma \circ \Delta_j)D\mathbf{v} \\ &= (D\mathbf{v})^T(I - \Gamma \circ \Delta_j)(D\mathbf{v}) \geq 0,\end{aligned}$$

and hence $(\Gamma', \beta)$ is a feasible solution to Equation (6.2.1). Moreover, the objective value becomes

$$\sum_{x,y\in\mathcal{D}} \Gamma'[x,y] = \sum_{x,y\in\mathcal{D}} \sqrt{\beta[x]}\Gamma[x,y]\sqrt{\beta[y]},$$

which completes the proof that the objective value of the SDP from the theorem statement is at least $\mathrm{ADV}^{\pm}(f)$.

It remains to prove that we can take any feasible solution to Equation (6.2.1) and turn it into a feasible solution to Equation (6.2.3) with the same objective value. To that end, let $(\Gamma', \beta)$ be a feasible solution to Equation (6.2.1). We let $D' \in \mathbb{R}^{\mathcal{D}\times\mathcal{D}}$ be a diagonal matrix, with

$$D'[x,x] = \begin{cases} 0, & \text{if } \beta[x] = 0, \\ \frac{1}{\beta[x]}, & \text{otherwise,} \end{cases}$$

and we let $\Gamma = D'\Gamma' D'$. With a similar argument as before, we obtain that $(\Gamma, \beta)$ is a feasible solution to Equation (6.2.3).

It remains to check that the resulting objective values are the same, for which we need to check that the entries in $\Gamma'$ are 0 whenever either their row or column label is $x$ and $\beta[x] = 0$. To that end, let $x, y \in \mathcal{D}$ be such that $x \neq y$ and $\beta[x] = 0$. Let $j$ be such that $x_j \neq y_j$. We consider the $2 \times 2$ submatrix from $\mathrm{diag}(\beta) - \Gamma' \circ \Delta_j$, indexed by rows and columns $x$ and $y$. Since taking submatrices preserves positive semidefiniteness, we obtain that

$$\begin{bmatrix} 0 & -\Gamma'[x,y] \\ -\Gamma'[x,y] & \beta[y] \end{bmatrix} \succeq 0.$$

Since the determinant of this matrix, which is the product of its eigenvalues, is $-\Gamma'[x,y]^2 \leq 0$, the only way for this matrix to be positive semidefinite is when $\Gamma'[x,y] = 0$. Thus, the objective values of the two optimization problems are the same, and the proof is complete. □

Note that even though the optimal value in Theorem 6.2.4 equals the quantity defined in Theorem 6.2.3, the adversary matrices $\Gamma$ that attain the optimal

solutions in both formulations are not the same. The proof of Theorem 6.2.4 reveals how to convert the latter into the former. If $(\Gamma, \beta)$ is an optimal solution for Theorem 6.2.4, then one must define $D = \mathrm{diag}(\sqrt{\beta})$, so that $\Gamma' = D\Gamma D$ is an optimal solution for Theorem 6.2.3. Thus, we can use Theorem 6.2.4 not only to compute $\mathrm{ADV}^{\pm}(f)$, but also to compute explicit adversary matrices that are optimal in the original formulation of the adversary bound, Theorem 6.2.3.

There are two benefits to the formulation of Theorem 6.2.4. First, the formulation is much simpler. The objective function and all constraints are either linear or semidefinite. This makes the conversion of this SDP to standard form much simpler and as such it can be much more easily implemented in a solver. Second, it provides a new way to visualize the adversary bound, as illustrated by Figure 6.2.3.

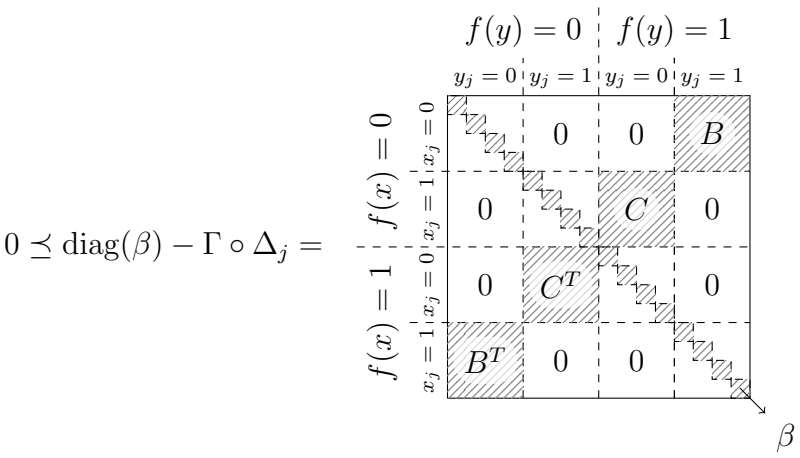



Figure 6.2.3: Visual interpretation of the semidefinite constraints in the reformulated primal adversary bound, i.e., Equation (6.2.1).

The objective function suggests to put as much weight on the (off-diagonal) entries of the adversary matrix $\Gamma$ as possible. By Lemma 6.2.2, though, this necessarily creates negative eigenvalues for $\Gamma \circ \Delta_j$, which makes it non-PSD. To compensate for that, one can put some values $\beta[x]$ on the diagonal, with the constraint that the total budget spent on both parts of the diagonal is $1/2$, i.e., the part where $f(x) = 0$ and the part where $f(x) = 1$. Thus, solving the reformulated primal adversary bound comes down to playing a balancing game between adding weight on the off-diagonal entries of $\Gamma$ and compensating for that on the diagonal, i.e., in the vector $\beta$.

### 6.2.2 The dual adversary bound

Since we have now obtained a simple SDP that computes the adversary bound, we can relatively straightforwardly arrive at its dual using standard techniques. In

the literature, one can find several equivalent formulations of the resulting SDP, all referred to as the dual adversary bound[2]. The formulation that we present here is essentially the form that appears in [Rei09].˙

**6.2.5.** THEOREM (Dual adversary bound [Rei09, Theorem 6.2]). *Let* $f : \mathcal{D} \to \{0,1\}$*, with* $\mathcal{D} \subseteq \{0,1\}^n$*. Then,* $\mathrm{ADV}^{\pm}(f)$ *is the optimal value of the following SDP:*

$$\min \quad \max_{x \in \mathcal{D}} \sum_{j=1}^{n} X_j[x,x] \tag{6.2.4a}$$

$$s.t. \quad \sum_{\substack{j=1 \\ x_j \neq y_j}}^{n} X_j[x,y] = 1, \qquad \forall x,y \in \mathcal{D}, f(x) \neq f(y), \tag{6.2.4b}$$

$$X_j \succeq 0, \qquad \forall j \in [n], \tag{6.2.4c}$$

*where the optimization ranges over all positive semidefinite matrices* $X_1, \ldots, X_n \in \mathbb{R}^{\mathcal{D} \times \mathcal{D}}$*.*

We can immediately see from the above formulation of the dual adversary bound, that if we have a feasible solution $X_1, \ldots, X_n$, we can obtain a new feasible solution by adding the identity matrix to each of the components, i.e., $X_1 + I, \ldots, X_n + I$. We find for all $j \in [n]$ that $X_j + I \succeq I \succ 0$, and hence we have found a strictly feasible solution. This implies that our dual SDP satisfies the Slater's conditions, which in turn ensures that the adversary bound satisfies strong duality.

As was the case with the primal adversary bound, we give a slightly different but equivalent formulation of the dual adversary bound. This formulation facilitates the connection with span programs in the next subsection.

**6.2.6.** THEOREM (Reformulated dual adversary bound). *Let* $f : \mathcal{D} \to \{0,1\}$*, with* $\mathcal{D} \subseteq \{0,1\}^n$*. Then,* $\mathrm{ADV}^{\pm}(f)$ *is the optimal value of the following min-*

[2]Not all formulations are exactly equivalent. For instance, the formulation used in [Bel15], differs by a factor of 2 from most of the other formulations one can find in the literature. In most cases this is of little concern, since the characterization of the quantum query complexity is up to constants anyway. However, in this thesis we also care about exact optimal values, so for us this difference is relevant to point out.

*imization program:*

$$\min \quad \sqrt{\max_{x\in f^{-1}(0)} \sum_{j=1}^{n} X_j[x,x] \cdot \max_{x\in f^{-1}(1)} \sum_{j=1}^{n} X_j[x,x]} \tag{6.2.5a}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} X_j[x,y] = 1, \qquad \forall x,y \in \mathcal{D}, f(x) \neq f(y), \tag{6.2.5b}$$

$$X_j[x,y] = 0, \qquad \forall x,y \in \mathcal{D}, x_j = y_j \wedge f(x) \neq f(y), \tag{6.2.5c}$$

$$X_j[x,y] = 0, \qquad \forall x,y \in \mathcal{D}, x_j \neq y_j \wedge f(x) = f(y), \tag{6.2.5d}$$

$$X_j \succeq 0, \qquad \forall j \in [n], \tag{6.2.5e}$$

*where the optimization ranges over all positive semidefinite matrices* $X_1, \ldots, X_n \in \mathbb{R}^{\mathcal{D}\times\mathcal{D}}$.

**Proof:**
We show that the optimal values of the optimization problems in Equations (6.2.4) and (6.2.5) are the same. To that end, we show that we can take a feasible solution to one, and turn it into a feasible solution to the other with an objective value that is the same or smaller, and vice versa.

Thus, let $X_1, \ldots, X_n$ be a feasible solution to the SDP from Equation (6.2.4). For every $j \in [n]$, we define the matrix $D_j \in \{0,1\}^{\mathcal{D}\times\mathcal{D}}$ as

$$D_j[x,y] = \begin{cases} 1, & \text{if } x_j \neq y_j \quad \text{and} \quad f(x) \neq f(y), \\ 1, & \text{if } x_j = y_j \quad \text{and} \quad f(x) = f(y), \\ 0, & \text{otherwise.} \end{cases}$$

After rearranging rows and columns, this matrix is a diagonal block matrix, with two all-ones blocks. Therefore, it is positive semidefinite, from which we find that $X'_j = X_j \circ D_j$ is positive semidefinite too. We easily verify that $X'_1, \ldots, X'_n$ is a feasible solution to Equation (6.2.5), and since the diagonal elements of the new $X'_j$'s didn't change, we observe that this objective value is smaller than the previous one by the AM-GM inequality. Thus, the optimal value of Equation (6.2.5) is at most the optimal value of Equation (6.2.4).

On the other hand, let $X_1, \ldots, X_n$ be a feasible solution to Equation (6.2.5). We define

$$\overline{W}_+ = \max_{x\in f^{-1}(1)} \sum_{j=1}^{n} X_j[x,x], \qquad \text{and} \qquad \overline{W}_- = \max_{x\in f^{-1}(0)} \sum_{j=1}^{n} X_j[x,x].$$

Now, we define the matrix $D \in \mathbb{R}^{\mathcal{D}\times\mathcal{D}}$,

$$D[x,y] = \begin{cases} \sqrt{\frac{\overline{W}_-}{\overline{W}_+}}, & \text{if } f(x) = f(y) = 1, \\ \sqrt{\frac{\overline{W}_+}{\overline{W}_-}}, & \text{if } f(x) = f(y) = 0, \\ 1, & \text{if } f(x) \neq f(y). \end{cases},$$

Moreover, observe that for all $\alpha > 0$,

$$\begin{bmatrix} \alpha & 1 \\ 1 & \frac{1}{\alpha} \end{bmatrix} = \begin{bmatrix} \sqrt{\alpha} \\ \sqrt{\frac{1}{\alpha}} \end{bmatrix} \begin{bmatrix} \sqrt{\alpha} & \sqrt{\frac{1}{\alpha}} \end{bmatrix} \succeq 0.$$

Since we can freely duplicate rows and columns without affecting positive semidefiniteness, we observe by setting $\alpha = (\overline{W}_-/\overline{W}_+)^{1/2}$ that $D$ is positive semidefinite. Next, for every $j \in [n]$, we let $X'_j = X_j \circ D$, where $\circ$ denotes the Hadamard product, and consequently $X'_j$ is positive semidefinite as well. Finally, the entries $X'_j[x, y]$ where $f(x) \neq f(y)$ are left unchanged compared to $X_j$, and therefore the constraints of Equation (6.2.5) still hold. These constraints are more stringent than the ones in Equation (6.2.4), from which we find that $X'_1, \ldots, X'_n$ is a feasible solution to said SDP as well.

It remains to compute the objective values. In this maximization program, the objective value for $X_1, \ldots, X_n$ is:

$$\sqrt{\max_{x \in f^{-1}(0)} \sum_{j=1}^{n} X_j[x, x] \cdot \max_{x \in f^{-1}(1)} \sum_{j=1}^{n} X_j[x, x]} = \sqrt{\overline{W}_- \overline{W}_+}.$$

Furthermore, we have

$$\begin{aligned} &\max_{x \in \mathcal{D}} \sum_{j=1}^{n} X'_j[x, x] \\ &\quad = \max \left\{ \max_{x \in f^{-1}(0)} \sum_{j=1}^{n} X_j[x, x] \cdot \sqrt{\frac{\overline{W}_+}{\overline{W}_-}}, \max_{x \in f^{-1}(1)} \sum_{j=1}^{n} X_j[x, x] \cdot \sqrt{\frac{\overline{W}_-}{\overline{W}_+}} \right\} \\ &\quad = \max \left\{ \sqrt{\overline{W}_+ \overline{W}_-}, \sqrt{\overline{W}_+ \overline{W}_-} \right\} = \sqrt{\overline{W}_+ \overline{W}_-}. \end{aligned}$$

Thus, we can change any solution to Equation (6.2.5) into a feasible solution to Equation (6.2.4) with the same objective value. We conclude that both problems must have the same optimal value. □

The reformulated dual adversary bound also admits a visual interpretation, which we provide in Figure 6.2.4.

Using the visual interpretation, we can now intuitively understand how to find the optimal solution to the reformulated dual adversary bound. One has to make sure that summations along all the rods where $f(x) \neq f(y)$ become exactly 1, so it is necessary to put some weights on the off-diagonal entries. However, as we saw in the intuitive understanding of the reformulated primal adversary bound, these off-diagonal entries will necessarily create negative eigenvalues, which have to be compensated for on the on-diagonal block. In contrast to the primal adversary

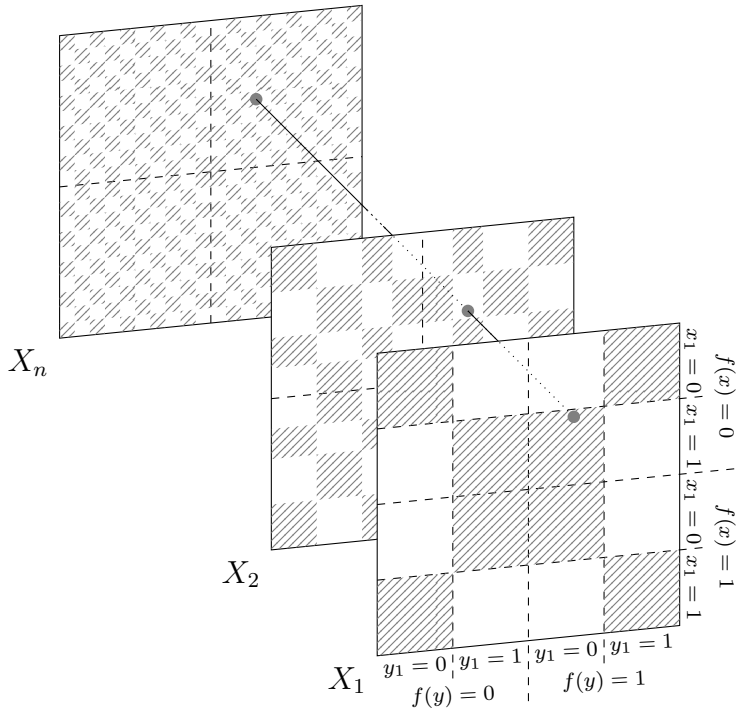



Figure 6.2.4: Graphical depiction of the dual adversary bound. The matrices $X_1, \ldots, X_n$ are arranged in an array. The equality constraints can be viewed as a summation along a rod, poking through all the matrices at entries $[x, y]$ where $f(x) \neq f(y)$. The sparsity constraints force given hatched pattern on the matrices, i.e., only the hatched areas in the picture can be non-zero. Since the area where $x_1 \neq y_1$ is different from the area where $x_2 \neq y_2$, etc., we have different sparsity patterns across the matrices $X_1, \ldots, X_n$.

bound, though, here we can use the whole block on the diagonal to achieve this compensation, rather than only the diagonal elements. To compute the objective value, one finds the maximal summations along a rod which pokes through the matrices at a diagonal entry $[x, x]$, where $f(x) = 0$, and $f(x) = 1$, respectively. The resulting objective value is the geometric mean of these two maxima.

Note that even though we have proven that the objective values of both SDPs are the same, this doesn't imply that the solutions themselves are the same too. However, the proof of Theorem 6.2.6 suggests how one can take a solution of the reformulated version, and turn it into one for the original dual adversary bound with the same objective value. Moreover, as we will see shortly, solutions to the reformulated dual adversary bound have a very nice connection to span programs.

### 6.2.3 Conversion between span programs and the dual adversary bound

In this subsection, we proceed to prove the connection between span programs and the dual adversary bound. To that end, we introduce a particular type of span program.

**6.2.7.** Definition (Query-efficient span programs).
Let $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ be a span program on $\mathcal{D} \subseteq \{0,1\}^n$. Suppose that we can orthogonally decompose $\mathcal{H}$ into subspaces $\mathcal{H}_1, \ldots, \mathcal{H}_n \subseteq \mathcal{H}$, i.e., that we can write

$$\mathcal{H} = \mathcal{H}_1 \oplus \cdots \oplus \mathcal{H}_n.$$

Furthermore, suppose that for every $j \in [n]$, we can orthogonally decompose $\mathcal{H}_j$ into $\mathcal{H}_j(0), \mathcal{H}_j(1)$, i.e., that we can write

$$\mathcal{H}_j = \mathcal{H}_j(0) \oplus \mathcal{H}_j(1).$$

Finally, suppose that for every $x \in \mathcal{D}$, we have

$$\mathcal{H}(x) = \bigoplus_{j=1}^{n} \mathcal{H}_j(x_j).$$

Then, we say that $\mathcal{P}$ is a *query-efficient span program.* ◀

Many prior works require these query-efficient conditions in their standard definition of span programs [Rei09; Jef14; Bel15]. We post it as a separate condition, because as we have seen in Section 6.1, it is not a necessary assumption for the construction of the span program algorithm.

By calling these span programs query-efficient, we suggest that the span program algorithm compiled from these span programs can be implemented query-efficiently. The following lemma warrants this claim. More precisely, we prove that the span program unitary, $U(x, \mathcal{P})$, can be implemented using only a single call to the a phase oracle that encodes $x$, i.e., an operation $O_x$ that performs the mapping $|j\rangle \mapsto (-1)^{x_j} |j\rangle$, for all $j \in [n]$.

**6.2.8.** Lemma (Query-efficiency of query-efficient span programs).
*Let $\mathcal{P}$ be a query-efficient span program on $\mathcal{D} \subseteq \{0,1\}^n$, and let $x \in \mathcal{D}$ be an input. Then $U(x, \mathcal{P})$ can be implemented with one query to a binary oracle $O_x$, which for all $j \in [n]$ acts as*

$$O_x : |j\rangle \mapsto (-1)^{x_j} |j\rangle .$$

**Proof:**
Recall that $U(x,\mathcal{P}) = (2\Pi_{\mathcal{K}} - I)(2\Pi_{\mathcal{H}(x)} - I)$. Since $2\Pi_{\mathcal{K}} - I$ doesn't depend on the specific input $x$, we can implement this part without making any queries to $O_x$. Thus, it remains to implement $2\Pi_{\mathcal{H}(x)} - I$ with one query to $O_x$.

To that end, observe that by our assumptions we can find a basis of $\mathcal{H}$ such that

$$\mathcal{H} = \mathrm{Span}\{|h_{j,b,k}\rangle : j \in [n], b \in \{0,1\}, k \in [\dim(\mathcal{H}_j(b))]\},$$

with $|h_{j,b,k}\rangle \in \mathcal{H}_j(b)$ for all $j \in [n]$, $b \in \{0,1\}$ and $k \in [\dim(\mathcal{H}_j(b))]$. Without making any queries, we can implement a basis transformation $U$ which for all $j \in [n]$, $b \in \{0,1\}$ and $k \in \dim(\mathcal{H}_j(b))$ performs the mapping

$$U : |j\rangle\,|b\rangle\,|k\rangle \mapsto |h_{j,b,k}\rangle\,.$$

We now claim that the operation $U(O_x \otimes Z \otimes I)U^\dagger$ implements $2\Pi_{\mathcal{H}(x)} - I$. To that end, let $j \in [n]$, $b \in \{0,1\}$, and $k \in [\dim(\mathcal{H}_j(b))]$. Then,

$$\begin{aligned} U(O_x \otimes Z \otimes I)U^\dagger\,|h_{j,b,k}\rangle &= U(O_x\,|j\rangle \otimes Z\,|b\rangle \otimes |k\rangle) = (-1)^{x_j+b}U(|j\rangle\,|b\rangle\,|k\rangle) \\ &= (-1)^{x_j+b}\,|h_{j,b,k}\rangle\,. \end{aligned}$$

Thus, indeed, we add a minus sign only when $x_j \neq b$, i.e., when $|h_{j,b,k}\rangle \in \mathcal{H}(x)^\perp$. This completes the proof. □

Note that the above lemma only makes a claim about the query complexity of implementing $U(x,\mathcal{P})$. In general it is much more involved to make claims about the time complexity of implementing this operation. Whether a time-efficient implementation for $U(x,\mathcal{P})$ exists usually depends heavily on the specific span program itself.

Now that we have introduced this specific class of span programs, we can show how feasible solutions to the reformulated dual adversary bound are related to query-efficient span programs.

**6.2.9.** Theorem (Span program to dual adversary bound solution).
*Let $\mathcal{P}$ be a query-efficient span program over a real Hilbert space, with notation as in Definition 6.2.7. Let $f : \mathcal{D} \to \{0,1\}$ be the function computed by $\mathcal{P}$. For every $x \in \mathcal{D}$, let $|w_x\rangle$ be a corresponding witness (positive if $x$ is positive, negative if not). For all $j \in [n]$, we define the matrix $X_j \in \mathbb{R}^{\mathcal{D}\times\mathcal{D}}$ as*

$$X_j[x,y] = \langle w_x|\,\Pi_{\mathcal{H}_j}\,|w_y\rangle\,.$$

*Then, $X_1, \ldots, X_n$ is a feasible solution to the reformulated dual adversary bound for $f$, with objective value equal to $\sqrt{\overline{W}_+\overline{W}_-}$, where*

$$\overline{W}_+ = \max_{x\in f^{-1}(1)} \||w_x\rangle\|^2\,, \qquad \text{and} \qquad \overline{W}_- = \max_{x\in f^{-1}(0)} \||w_x\rangle\|^2\,.$$

**Proof:**
First, we show that $X_1, \ldots, X_n$ forms a feasible solution to the reformulated dual adversary bound. To that end, we let $|w_{x,j}\rangle = \Pi_{\mathcal{H}_j} |w_x\rangle$, and we observe that we can write

$$X_j = W_j^\dagger W_j, \qquad \text{with} \qquad W_j = \begin{bmatrix} \cdots & |w_{x,j}\rangle & \cdots \end{bmatrix}_x : \mathbb{R}^{\mathcal{D}} \to \mathcal{H}_j.$$

Thus, $X_j \succeq 0$ and $W_j$ is a Gram vectorization of $X_j$.

Next, let $x, y \in \mathcal{D}$ be such that $f(x) = 1$ and $f(y) = 0$. Then, $|w_x\rangle$ and $|w_y\rangle$ are positive and negative witnesses for $x$ and $y$, respectively. From their properties in Definition 6.1.2, we observe that

$$\begin{aligned} 1 &= \langle w_0 | w_y \rangle + 0 = \langle w_x | \Pi_{\mathcal{K}^\perp} |w_y\rangle + \langle w_x | \Pi_{\mathcal{K}} |w_y\rangle = \langle w_x | w_y \rangle \\ &= \sum_{j=1}^n \langle w_x | \Pi_{\mathcal{H}_j} |w_y\rangle = \sum_{j=1}^n X_j[x,y]. \end{aligned}$$

Furthermore, if $f(x) = 1$, $f(y) = 0$ and $x_j = y_j$, then we have that

$$X_j[x,y] = \langle w_x | \Pi_{\mathcal{H}_j} |w_y\rangle = \langle w_x | \Pi_{\mathcal{H}_j(x_j)} |w_y\rangle = 0,$$

where in the second equality we used that $|w_{x,j}\rangle \in \mathcal{H}(x) \cap \mathcal{H}_j = \mathcal{H}_j(x_j)$, and in the last one that $|w_{y,j}\rangle \in \mathcal{H}(y)^\perp \cap \mathcal{H}_j \subseteq \mathcal{H}_j(x_j)^\perp$. Similarly if $f(x) = f(y) = 1$, $x_j = 1$ and $y_j = 0$, then

$$X_j[x,y] = \langle w_x | \Pi_{\mathcal{H}_j} |w_y\rangle = \langle w_x | \Pi_{\mathcal{H}_j(1)} \Pi_{\mathcal{H}_j(0)} |w_y\rangle = 0,$$

and if $f(x) = f(y) = 0$, $x_j = 1$ and $y_j = 0$, then

$$X_j[x,y] = \langle w_x | \Pi_{\mathcal{H}_j} |w_y\rangle = \langle w_x | \Pi_{\mathcal{H}_j(1)^\perp \cap \mathcal{H}_j} \Pi_{\mathcal{H}_j(0)^\perp \cap \mathcal{H}_j} |w_y\rangle = 0.$$

Thus $X_1, \ldots, X_n$ is a feasible solution to the reformulated dual adversary bound, and it remains to compute the objective value. To that end, observe that for all $x \in \mathcal{D}$,

$$\sum_{j=1}^n X_j[x,x] = \sum_{j=1}^n \langle w_x | \Pi_{\mathcal{H}_j} |w_x\rangle = \langle w_x | w_x \rangle = \| |w_x\rangle \|^2 .$$

Thus, indeed, the objective value becomes

$$\begin{aligned} \sqrt{\max_{x \in f^{-1}(1)} \sum_{j=1}^n X_j[x,x] \cdot \max_{x \in f^{-1}(0)} \sum_{j=1}^n X_j[x,x]} &= \sqrt{\max_{x \in f^{-1}(1)} \| |w_x\rangle \|^2 \cdot \max_{x \in f^{-1}(0)} \| |w_x\rangle \|^2} \\ &= \sqrt{\overline{W}_+ \overline{W}_-}. \end{aligned}$$

This completes the proof. □

We have now shown how one can take a span program and turn it into a solution to the reformulated dual adversary bound. Moreover, if we take the optimal witnesses in Theorem 6.2.9, then we observe that the objective value of the reformulated dual adversary bound solution is $C(\mathcal{P})$. Therefore, we have shown that for any query-efficient span program $\mathcal{P}$ over a real Hilbert space computing $f$, we have

$$\mathrm{ADV}^{\pm}(f) \leq C(\mathcal{P}).$$

The natural question that comes to mind is whether we can similarly construct a query-efficient span program $\mathcal{P}$ over the reals from a feasible solution to the reformulated adversary bound, such that the above inequality becomes an equality. This can indeed be done, as the following theorem shows.

**6.2.10.** Theorem (Dual adversary bound solution to span program). *Let $f : \mathcal{D} \to \{0,1\}$, with $\mathcal{D} \subseteq \{0,1\}^n$. Let $X_1, \ldots, X_n \in \mathbb{R}^{\mathcal{D}\times\mathcal{D}}$ be a feasible solution to the reformulated dual adversary bound for $f$. Then, for all $j \in [n]$, we take Gram vectorizations*

$$X_j = W_j^\dagger W_j, \qquad \text{where} \qquad W_j = \begin{bmatrix} \cdots & \left|w'_{x,j}\right\rangle & \cdots \end{bmatrix}_x \in \mathbb{R}^{\mathcal{D}} \to \mathcal{H}_j.$$

*We write*

$$\mathcal{H}_j(0) = \mathrm{Span}\{\left|w'_{x,j}\right\rangle : x \in f^{-1}(1), x_j = 0\},$$
$$\mathcal{H}_j(1) = \mathrm{Span}\{\left|w'_{x,j}\right\rangle : x \in f^{-1}(1), x_j = 1\},$$

*$\mathcal{H}_j = \mathcal{H}_j(0) \oplus \mathcal{H}_j(1)$, and $\mathcal{H} = \bigoplus_{j=1}^n \mathcal{H}_j$. Furthermore, for every $x \in \mathcal{D}$, we let $|w'_x\rangle = \bigoplus_{j=1}^n \left|w'_{x,j}\right\rangle$, and*

$$\mathcal{K} = \mathrm{Span}\{|w'_x\rangle - \left|w'_y\right\rangle : x, y \in f^{-1}(1)\}.$$

*Finally, we let $|w'_0\rangle$ be the shortest vector in the affine subspace $|w'_x\rangle + \mathcal{K}$, for any $x \in f^{-1}(1)$, and write $N = \||w'_0\rangle\|$. Now, we let*

$$|w_0\rangle = \frac{1}{N}|w'_0\rangle, \qquad |w_x\rangle = \frac{1}{N}|w'_x\rangle, \qquad \text{and} \qquad |w_y\rangle = N\left|w'_y\right\rangle,$$

*for all $x \in f^{-1}(1)$ and $y \in f^{-1}(0)$. Then, the span program $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ computes $f$, uses a real Hilbert space $\mathcal{H}$, has witnesses $|w_x\rangle$ for every $x \in \mathcal{D}$, is query-efficient, and its complexity $C(\mathcal{P})$ is at most the objective value of the reformulated dual adversary bound.*

**Proof:**
Observe that every input $x \in f^{-1}(1)$ is indeed a positive input for $\mathcal{P}$, since by definition, $|w'_0\rangle$ can be written as a sum of $|w'_x\rangle \in \mathcal{H}(x)$, and a vector in $\mathcal{K}$. Furthermore, observe that $|w'_0\rangle$ can be written as

$$|w'_0\rangle = \sum_{x\in f^{-1}(1)} \alpha_x |w'_x\rangle, \qquad \text{where} \qquad \sum_{x\in f^{-1}(1)} \alpha_x = 1,$$

for some real numbers $\alpha_x \in \mathbb{R}$. Thus for every $y \in f^{-1}(0)$, we have

$$\langle w_0|w_y\rangle = \langle w'_0|w'_y\rangle = \sum_{x\in f^{-1}(1)} \alpha_x \langle w'_x|w'_y\rangle = \sum_{x\in f^{-1}(1)} \alpha_x \sum_{j=1}^{n} \langle w'_x| \Pi_{\mathcal{H}_j} |w'_y\rangle$$
$$= \sum_{x\in f^{-1}(1)} \alpha_x \sum_{j=1}^{n} X_j[x,y] = \sum_{x\in f^{-1}(1)} \alpha_x = 1.$$

Thus, it remains to check that $|w_y\rangle \in \mathcal{K}^\perp \cap \mathcal{H}(y)^\perp$. Since $(\langle w'_x| - \langle w'_{x'}|)\,|w'_y\rangle = 1 - 1 = 0$ for $x, x' \in f^{-1}(1)$, we easily obtain that $|w_y\rangle \in \mathcal{K}^\perp$, hence it remains to check that $|w_y\rangle \in \mathcal{H}(y)^\perp$. It suffices to show that for every $j \in [n]$, we have $|w'_{y,j}\rangle \in \mathcal{H}_j(y_j)^\perp \cap \mathcal{H}_j$. Thus, we must show that $|w'_{y,j}\rangle$ is orthogonal to all $|w'_{x,j}\rangle$'s, where $x \in f^{-1}(1)$ and $x_j = y_j$. But we know that

$$\langle w'_{x,j}|w'_{y,j}\rangle = X_j[x,y] = 0,$$

since $X_1, \ldots, X_n$ is a feasible solution to the reformulated dual adversary bound. Thus, $\mathcal{P}$ indeed computes the same function $f$, and for every $x \in \mathcal{D}$, $|w_x\rangle$ is a witness.

It is easily seen that $\mathcal{P}$ is indeed query-efficient over a real Hilbert space, so it remains to show that its complexity is at most the objective value of the reformulated adversary bound. To that end, observe that

$$C(\mathcal{P})^2 = W_+(\mathcal{P})W_-(\mathcal{P}) \leq \max_{x\in f^{-1}(1)} \||w_x\rangle\|^2 \cdot \max_{y\in f^{-1}(0)} \||w_y\rangle\|^2$$
$$= \max_{x\in f^{-1}(1)} \||w'_x\rangle\|^2 \cdot \max_{y\in f^{-1}(0)} \big\||w'_y\rangle\big\|^2$$
$$= \max_{x\in f^{-1}(1)} \sum_{j=1}^{n} X_j[x,x] \cdot \max_{y\in f^{-1}(0)} \sum_{j=1}^{n} X_j[y,y].$$

This completes the proof. □

Thus, we have now proven that given the optimal solution to the reformulated dual adversary bound, we can generate a query-efficient span program whose complexity equals $\mathrm{ADV}^{\pm}(f)$. For completeness, we show how this observation can be used to reprove [Rei11, Theorem 1.3].

**6.2.11.** Corollary. *For every $f : \mathcal{D} \to \{0,1\}$, with $\mathcal{D} \subseteq \{0,1\}^n$, we have*

$$Q(f) = \Theta(\mathrm{ADV}^{\pm}(f)).$$

**Proof:**
We have already stated the lower bound, in Theorem 6.2.3. For the upper bound, we can take an optimal solution to the reformulated dual adversary

bound, and convert it into a query-efficient span program $\mathcal{P}$ with complexity $C(\mathcal{P}) = \mathrm{ADV}^{\pm}(f)$. Since according to Lemma 6.2.8 we can implement the span program unitary $U(x, \mathcal{P})$ with one query, we find from Algorithm 6.1.18 that we can implement an algorithm that evaluates $\mathcal{P}$ using $\mathcal{O}(C(\mathcal{P}))$ queries to the input. Thus, $Q(f) = \mathcal{O}(\mathrm{ADV}^{\pm}(f))$, completing the proof. □

Note that in the above two conversions, we restrict ourselves to span programs over real Hilbert spaces. This is because the adversary bound formulations are all over the reals as well. One could equivalently define all the optimization problems considered in this section over complex vector spaces, and achieve a similar correspondence between reformulated dual complex adversary bound solutions and query-efficient span programs.

There is one very important benefit of doing everything over the reals, though. Since we have been able to show that the adversary bound over the reals is already a tight characterization, we have shown that there is no need to generalize all previous results to complex spaces. In particular, it means that if we are interested in the query complexity of a particular function, it suffices to solve the adversary bound over the reals and we don't need to bother about complex numbers at all.

We have shown a one-to-one correspondence between feasible solutions to the reformulated dual adversary bound, and query-efficient span programs over the reals. This correspondence gives us a very powerful and elegant way to interpret dual adversary bound solutions, which at face value are nothing more than $n$ tableaux of numbers. However, by turning these tableaux into span programs, one can use the span program framework and its subsequent analysis to understand better how a quantum algorithm actually computes the function at hand with the given cost. Moreover, we will see in the next section that one can also use the span program framework to modify and combine dual adversary bound solutions.

There is one final thing to observe. In Theorem 6.2.10, the dimension of the space $\mathcal{H}$ turns out to be the sum of the dimensions of the vectors in the Gram vectorization of all $X_j$'s. But these dimensions are the ranks of the $X_j$'s, and hence we find that any feasible solution $X_1, \ldots, X_n$ to the reformulated dual adversary bound gives rise to a span program with Hilbert space $\mathcal{H}$, where

$$\dim(\mathcal{H}) = \sum_{j=1}^{n} \mathrm{rank}(X_j).$$

This provides a connection between the rank of dual adversary bound solutions and the space requirements of span programs, and consequently also quantum algorithms.

The construction we present here starts with a solution to the *reformulated* dual adversary bound. It should be noted that one can also start from a solution to the original dual adversary bound, i.e., Theorem 6.2.5, as is for instance done in [Bel14, Theorem 3.39]. The construction presented here saves a factor of 2 in

the dimension of the Hilbert space, though.

We remark that the space requirements of span programs are further developed in [Jef20], where it is shown that space-efficient query-efficient span programs always exist. Thus, we might be able to show a more substantial connection between the space requirements of a quantum algorithm, $\dim(\mathcal{H})$, and the sum of the ranks of an optimal solution to the reformulated dual adversary bound. We leave these connections for future work.

Finally, we emphasize that there exist generalizations of the dual adversary bound to functions with non-boolean input and output, e.g., in [Rei09, Theorem 6.4]. It has also been shown that these solutions can be turned into efficient quantum query algorithms [LMR+11]. There has also been some progress in generalizing the concept of span programs to the setting where the function input is non-boolean [Jef14], and where the output is non-boolean [BT19]. Thus, there is quite some existing literature already on the non-boolean case, and it would be interesting to phrase all those results in a similar language as presented here, and to see which of the upcoming parts of the theory follow through in this generalized setting. We leave all this for future work.

# Chapter 7

# Compositions of span programs

One property of span programs that makes them particularly appealing is the relative ease with which they can be composed through a series of composition constructions. In this chapter, we introduce two classes of these. First, in Section 7.1, we introduce logical compositions, that allow for evaluating ANDs, ORs and NOTs of the decision problems that are evaluated by individual span programs. Afterwards, in Section 7.2, we introduce a more general class of such compositions, known as graph compositions.

## 7.1 Logical composition of span programs

First, we formally state and prove the logical composition results, in Section 7.1.1. Then, we study how these results can be interpreted on the level of dual adversary bound solutions, in Section 7.1.2. Finally, we more deeply analyze the operational implications, in Section 7.1.3.

### 7.1.1 Definition and basic properties

We start with a simple observation. Observe that the dual adversary bound in Theorem 6.2.5, and its reformulation in Theorem 6.2.6, are invariant under taking the negation of $f$. That is, a feasible solution to the minimization problem for $f$ is also a feasible solution to the minimization problem for $\neg f$, and the corresponding objective value is the same. From this we find that $\mathrm{ADV}^{\pm}(f) = \mathrm{ADV}^{\pm}(\neg f)$. This is not surprising, since the adversary bound tightly characterizes quantum query complexity and computing $f$ and $\neg f$ should be equally difficult.

Since there is a direct correspondence between dual adversary bound solutions and query-efficient span programs, every query-efficient span program $\mathcal{P}$ computing $f$ should also have a counterpart that computes $\neg f$ with the same complexity. However, from the intuitive picture developed in Figure 6.1.2, it is not immediately clear how such a span program should look like. After all, there

is a qualitative difference between positive and negative inputs, i.e., in the positive case each component of $|w_0\rangle$ starts rotating while in the negative case it has a non-negative overlap with the stationary subspace.

From the construction in Theorem 6.2.10, we can get some idea of how we can convert a query-efficient span program computing a function $f$, into one that computes the function $\neg f$. Indeed, if one carefully investigates what happens when we run the construction presented in Theorem 6.2.10 for the function $\neg f$ rather than $f$, and if we assume that we use the matrices $W_j$ that are of minimal dimensions, then we observe that the new spaces $\mathcal{H}'_j(0)$ and $\mathcal{H}'_j(1)$ become $\mathcal{H}_j(1) = \mathcal{H}_j(0)^\perp \cap \mathcal{H}_j$ and $\mathcal{H}_j(0) = \mathcal{H}_j(1)^\perp \cap \mathcal{H}_j$, respectively, and the new space $\mathcal{K}'$ becomes $(\mathcal{K} \oplus \mathrm{Span}\{|w_0\rangle\})^\perp$. In short, this gives us a direct way to convert a query-efficient span program $\mathcal{P}$ computing a function $f$, into one that computes the function $\neg f$.

Surprisingly, it turns out that this conversion can be generalized to the setting where the span program is not necessarily query-efficient. We refer to this technique as *span program negation*, and we formally present it in the following definition. This construction was first introduced in [Rei09, Lemma 4.1], where it was called span program complementation instead. The existence of such a construction was already hinted at in [RŠ12, Definition 2.7], but only a special case was presented there. A construction in a more restricted setting was already present in [CF02; NNP04].

**7.1.1.** Definition. Let $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ be a span program, computing a function $f : \mathcal{D} \to \{0,1\}$. We let

$$\begin{aligned} \mathcal{H}'(x) &= \mathcal{H}(x)^\perp, && \forall x \in \mathcal{D}, \\ \mathcal{K}' &= (\mathcal{K} \oplus \mathrm{Span}\{|w_0\rangle\})^\perp. \end{aligned}$$

Then the *negation of* $\mathcal{P}$ is the span program $\neg\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}'(x), \mathcal{K}', |w_0\rangle)$. ◀

We analyze its properties in the theorem below.

**7.1.2.** Theorem (Properties of span program negation). *Let $\mathcal{P}$ be a span program on $\mathcal{D}$ that computes the function $f : \mathcal{D} \to \{0,1\}$. Then, with $x \in \mathcal{D}$,*

1. *$\neg\mathcal{P}$ computes $\neg f$.*
2. *The positive/negative witnesses for $x$ in $\neg\mathcal{P}$ are exactly the negative/positive witnesses for $x$ in $\mathcal{P}$, respectively.*
3. *The minimal positive/negative witness for $x$ in $\neg\mathcal{P}$ is the minimal negative/positive witness for $x$ in $\mathcal{P}$, respectively.*
4. *$w_+(x, \neg\mathcal{P}) = w_-(x, \mathcal{P})$ and $w_-(x, \neg\mathcal{P}) = w_+(x, \mathcal{P})$.*
5. *$W_+(\neg\mathcal{P}) = W_-(\mathcal{P})$ and $W_-(\neg\mathcal{P}) = W_+(\mathcal{P})$.*
6. *$C(\neg\mathcal{P}) = C(\mathcal{P})$.*
7. *$\mathcal{P}$ is query-efficient if and only if $\neg\mathcal{P}$ is query-efficient.*

8. $U(x, \neg\mathcal{P}) = -(2\,|w_0\rangle\langle w_0| - I)U(x, \mathcal{P})$.

**Proof:**
Let $x \in \mathcal{D}$, and $|w\rangle \in \mathcal{H}$ arbitrarily. We derive the following sequence of equivalences.

$$\begin{aligned} |w\rangle \in (\mathcal{K}')^\perp \cap \mathcal{H}'(x)^\perp &\Leftrightarrow |w\rangle \in (\mathcal{K} \oplus \mathrm{Span}\{|w_0\rangle\}) \cap \mathcal{H}(x) \\ &\Leftrightarrow |w\rangle \in \mathcal{H}(x) \wedge \exists \alpha \in \mathbb{C} : |w\rangle - \alpha\,|w_0\rangle \in \mathcal{K}. \end{aligned}$$

Using these equivalences, we obtain by elementary linear algebra that

$$\begin{aligned} &|w\rangle \text{ is a negative witness in } \neg\mathcal{P} \\ &\quad\Leftrightarrow |w\rangle \in (\mathcal{K}')^\perp \cap \mathcal{H}'(x)^\perp \wedge \langle w_0|w\rangle = 1 \\ &\quad\Leftrightarrow |w\rangle \in \mathcal{H}(x) \wedge \exists \alpha \in \mathbb{C} : |w\rangle - \alpha\,|w_0\rangle \in \mathcal{K} \wedge \langle w_0|w\rangle = 1 \\ &\quad\Leftrightarrow |w\rangle \in \mathcal{H}(x) \wedge |w\rangle - |w_0\rangle \in \mathcal{K} \\ &\quad\Leftrightarrow |w\rangle \text{ is a positive witness in } \mathcal{P}. \end{aligned}$$

Since we easily check that $\neg(\neg\mathcal{P}) = \mathcal{P}$, we can apply the above sequence of equivalences to $\neg\mathcal{P}$ to obtain that $|w\rangle$ is a negative witness in $\mathcal{P}$ if and only if $|w\rangle$ is a positive witness is $\neg\mathcal{P}$. Claims 1 and to 6 in the theorem follow directly from these equivalences of the witnesses. Claim 7 follows immediately from $\mathcal{H}'(x) = \mathcal{H}(x)^\perp$ and the definition of query-efficiency, i.e., Definition 6.2.7. Claim 8 follows from

$$\begin{aligned} U(x, \neg\mathcal{P}) &= (2\Pi_{(\mathcal{K}\oplus\mathrm{Span}\{|w_0\rangle\})^\perp} - I)(2\Pi_{\mathcal{H}(x)^\perp} - I) \\ &= (2(\Pi_{\mathcal{K}} + |w_0\rangle\langle w_0|) - I)(2\Pi_{\mathcal{H}(x)} - I) \\ &= (I - 2\,|w_0\rangle\langle w_0|)(2\Pi_{\mathcal{K}} - I)(2\Pi_{\mathcal{H}(x)} - I) \\ &= -(2\,|w_0\rangle\langle w_0| - I)U(x, \mathcal{P}). \end{aligned}$$

This completes the proof. □

Even though the above proof is very short and direct, it is difficult to develop intuition for how $\mathcal{P}$ and $\neg\mathcal{P}$ relate to one another. For instance, if we consider the visualization of the subspaces $\mathcal{K}$ and $\mathcal{H}(x)$ from Figure 6.1.4, it is difficult to imagine how the new spaces $\mathcal{K}'$ and $\mathcal{H}'(x)$ can be visualized on top of this same figure. Developing a more visual interpretation of the above result would be an interesting topic of further research.[1]

Now that we know how to compute the negation of a span program, the immediate question that arises is whether we can similarly find constructions

[1] I did make several attempts to visualize this in three dimensions, one of which involved poking an object called a "satéprikker" through a piece of paper. Unsurprisingly, the results were not very impressive.

that compute other logical compositions. For instance, if we have span programs $\mathcal{P}$ and $\mathcal{P}'$, computing the functions $f$ and $f'$, can we construct a span program that computes $f \vee f'$, or $f \wedge f'$? The answer is yes, as was first shown in [RŠ12] in the special case where $f$ and $f'$ themselves are given by formulas, and later showed more generally for span programs in [Rei09, Theorem 4.3]. The AND-composition we construct below is based on the direct-sum construction in [Rei09, Definition 4.5].

The core idea of the AND-composition is as follows. Let $\mathcal{P}^{(1)}, \dots, \mathcal{P}^{(n)}$ be $n$ span programs, all on the same domain $\mathcal{D}$, and for all $j \in [n]$, we write the components of the span program explicitly as $\mathcal{P}^{(j)} = (\mathcal{H}^{(j)}, x \mapsto \mathcal{H}^{(j)}(x), \mathcal{K}^{(j)}, |w_0^{(j)}\rangle)$. Recall that for $x \in \mathcal{D}$ and $j \in [n]$, $x$ is a positive input in $\mathcal{P}^{(j)}$ if and only if $|w_0^{(j)}\rangle \in \mathcal{K}^{(j)} + \mathcal{H}^{(j)}(x)$, i.e., we can reach the vector $|w_0^{(j)}\rangle$ by taking a linear combination of vectors in $\mathcal{K}^{(j)}$ and $\mathcal{H}^{(j)}(x)$. Now, we define a new vector space $\mathcal{H}$ by simply taking the direct sum of all $\mathcal{H}^{(j)}$'s, and similarly define $\mathcal{K}$ and $\mathcal{H}(x)$, and we let $|w_0\rangle$ be a linear combination of the $|w_0^{(j)}\rangle$'s with non-zero coefficients. Then, reaching $|w_0\rangle$ by taking a linear combination of vectors in $\mathcal{K}$ and $\mathcal{H}(x)$, can only be done if we can reach each of the individual components $|w_0^{(j)}\rangle$ in each of the spaces $\mathcal{K}^{(j)} + \mathcal{H}^{(j)}(x)$ separately. Thus, we find that $x$ is a positive input in this composed span program $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ exactly when it is a positive instance for all the individual span programs $\mathcal{P}^{(j)}$, with $j \in [n]$.

We now formalize this idea in the following definition.

**7.1.3.** Definition (AND-composition of span programs). Let $\mathcal{P}^{(1)}, \dots, \mathcal{P}^{(n)}$ be span programs on $\mathcal{D}$, and we write $\mathcal{P}^{(j)} = (\mathcal{H}^{(j)}, x \mapsto \mathcal{H}^{(j)}(x), \mathcal{K}^{(j)}, |w_0^{(j)}\rangle)$ for all $j \in [n]$. Let $\alpha_1, \dots, \alpha_n > 0$ be some coefficients. We define the normalized coefficients $\alpha'_1, \dots, \alpha'_n > 0$ as

$$\alpha'_j = \frac{\alpha_j}{\sum_{k=1}^n \alpha_k},$$

for all $j \in [n]$. Then, we let

$$\mathcal{H} = \bigoplus_{j=1}^n \mathcal{H}^{(j)}, \qquad \mathcal{H}(x) = \bigoplus_{j=1}^n \mathcal{H}^{(j)}(x),$$
$$\mathcal{K} = \bigoplus_{j=1}^n \mathcal{K}^{(j)}, \qquad |w_0\rangle = \sum_{j=1}^n \sqrt{\alpha'_j}|w_0^{(j)}\rangle,$$

and we let $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$. We call this span program $\mathcal{P}$ the *AND-composition* of $\mathcal{P}^{(1)}, \dots, \mathcal{P}^{(n)}$ with weights $\alpha_1, \dots, \alpha_n$, and refer to it using the shorthand notations

$$\alpha_1\mathcal{P}^{(1)} \wedge \alpha_2\mathcal{P}^{(2)} \wedge \cdots \wedge \alpha_n\mathcal{P}^{(n)}, \qquad \text{and} \qquad \bigwedge_{j=1}^n \alpha_j\mathcal{P}^{(j)}.$$

In these shorthand notations, we omit $\alpha_j$ if it is 1, i.e., we write $\mathcal{P}^{(j)}$ instead of $1\mathcal{P}^{(j)}$. ◀

Note that in the shorthand notations, we do not require the coefficients to be properly normalized, i.e., if we write

$$\bigwedge_{j=1}^{n} \alpha_j \mathcal{P}^{(j)},$$

we don't require the $\alpha_j$'s to sum to 1. Instead, in the AND-composition itself, the coefficients are first normalized to a sequence $\alpha'_1, \ldots, \alpha'_n$ that sums to 1, and are subsequently used in the definition of $|w_0\rangle$. The reason for this is that it makes the notation significantly less cumbersome later on.

Similarly as we did with span program negation, we now analyze the properties of the AND-composition of span programs.

**7.1.4.** Theorem (Properties of the AND-composition of span programs). *Let $\mathcal{P}^{(1)}, \ldots, \mathcal{P}^{(n)}$ be span programs on $\mathcal{D}$, where for all $j \in [n]$, $\mathcal{P}^{(j)}$ computes the function $f^{(j)} : \mathcal{D} \to \{0,1\}$. Let $\alpha_1, \ldots, \alpha_n > 0$ be strictly positive coefficients, and let $\alpha'_1, \ldots, \alpha'_n > 0$ be their normalized counterparts, as in Definition 7.1.3. Let*

$$\mathcal{P} = \bigwedge_{j=1}^{n} \alpha_j \mathcal{P}^{(j)}.$$

*Let $x \in \mathcal{D}$. Then, $\mathcal{P}$ has the following properties:*

1. *$\mathcal{P}$ computes $\bigwedge_{j=1}^{n} f^{(j)}$.*

2a. *If $x$ is a positive input, then the positive witnesses for $x$ in $\mathcal{P}$ are exactly the vectors $|w\rangle \in \mathcal{H}$ of the form*

$$|w\rangle = \sum_{j=1}^{n} \sqrt{\alpha'_j} |w^{(j)}\rangle, \tag{7.1.1}$$

*where for all $j \in [n]$, $|w^{(j)}\rangle$ is a positive witness for $x$ in $\mathcal{P}^{(j)}$.*

2b. *If $x$ is a negative input, then the negative witnesses for $x$ in $\mathcal{P}$ are exactly the vectors $|w\rangle \in \mathcal{H}$ of the form*

$$|w\rangle = \sum_{\substack{j=1 \\ \beta_j \neq 0}}^{n} \frac{\beta_j}{\sqrt{\alpha'_j}} |w^{(j)}\rangle + \sum_{\substack{j=1 \\ \beta_j = 0}}^{n} |\overline{w}^{(j)}\rangle, \tag{7.1.2}$$

*where for all $j \in [n]$, $\beta_j \in \mathbb{C}$ such that $\sum_{j=1}^{n} \beta_j = 1$, $|w^{(j)}\rangle$ is a negative witness for $x$ in $\mathcal{P}^{(j)}$, and $|\overline{w}^{(j)}\rangle \in (\mathcal{K}^{(j)})^{\perp} \cap \mathcal{H}^{(j)}(x)^{\perp} \cap \mathrm{Span}\{|w_0^{(j)}\rangle\}^{\perp}$.*

*3a. If $x$ is a positive input for $\mathcal{P}$, then the minimal positive witness for $x$ in $\mathcal{P}$ is $|w\rangle$ in Equation 7.1.1, where for every $j \in [n]$, we choose $|w^{(j)}\rangle$ to be the minimal positive witness for $x$ in $\mathcal{P}^{(j)}$.*

*3b. If $x$ is a negative input for $\mathcal{P}$, then the minimal negative witness for $x$ in $\mathcal{P}$ is $|w\rangle$ in Equation 7.1.2, where for all $j \in [n]$, we choose*

$$\beta_j = \frac{\alpha'_j}{w_-(x, \mathcal{P}^{(j)})} \cdot \left[ \sum_{j=1}^{n} \frac{\alpha'_j}{w_-(x, \mathcal{P}^{(j)})} \right]^{-1},$$

*$|w^{(j)}\rangle$ to be the minimal negative witness for $x$ in $\mathcal{P}^{(j)}$ and $|\overline{w}^{(j)}\rangle = 0$.*

*4. We have*

$$w_+(x, \mathcal{P}) = \sum_{j=1}^{n} \alpha'_j w_+(x, \mathcal{P}^{(j)}) \quad \text{and} \quad w_-(x, \mathcal{P}) = \left[ \sum_{j=1}^{n} \frac{\alpha'_j}{w_-(x, \mathcal{P}^{(j)})} \right]^{-1}.$$

*5. Suppose that none of the $\mathcal{P}^{(j)}$'s nor $\mathcal{P}$ compute a constant function. Then,*

$$W_+(\mathcal{P}) \leq \sum_{j=1}^{n} \alpha'_j W_+(\mathcal{P}^{(j)}) \qquad \text{and} \qquad W_-(\mathcal{P}) \leq \max_{j \in [n]} \frac{W_-(\mathcal{P}^{(j)})}{\alpha'_j},$$

*and in particular,*

$$C(\mathcal{P})^2 \leq \sum_{j=1}^{n} \alpha_j W_+(\mathcal{P}^{(j)}) \cdot \max_{j \in [n]} \frac{W_-(\mathcal{P}^{(j)})}{\alpha_j}.$$

*6. Suppose that none of the $\mathcal{P}^{(j)}$'s compute a constant function, and suppose furthermore that we can decompose $\mathcal{D} = \bigtimes_{j=1}^{n} \mathcal{D}^{(j)}$, and that for every input $(x^{(1)}, \ldots, x^{(n)}) = x \in \mathcal{D}$, $\mathcal{H}^{(j)}(x)$ only depends on $x^{(j)}$. Then, all inequalities from item 5 turn into equalities. Moreover, if for all $j \in [n]$, we choose $\alpha_j = W_-(\mathcal{P}^{(j)})$, then we minimize $C(\mathcal{P})$, and*

$$C(\mathcal{P})^2 = \sum_{j=1}^{n} C(\mathcal{P}^{(j)})^2.$$

*7. $\mathcal{P}$ is query-efficient if and only if for every $j \in [n]$, $\mathcal{P}^{(j)}$ is.*

*8. $U(x, \mathcal{P}) = \prod_{j=1}^{n} U(x, \mathcal{P}^{(j)})$.*

**Proof:**
Claims 1 and 5-8 follow easily from claims 2-4, so we focus on proving those. We start by checking claims 2a, 3a and 4a, by characterizing the positive witnesses for an input $x \in \mathcal{D}$ and analyzing their sizes. To that end, we let $|w\rangle \in \mathcal{H}$,

and we write $|w^{(j)}\rangle = (\alpha'_j)^{-1/2}\Pi_{\mathcal{H}^{(j)}}|w\rangle$. Then, $|w\rangle$ is indeed decomposed as in Equation 7.1.1. Furthermore,

$$\begin{aligned}
&|w\rangle \text{ is a positive witness for } x \text{ in } \mathcal{P}\\
&\quad\Leftrightarrow |w\rangle \in \mathcal{H}(x) \wedge |w\rangle - |w_0\rangle \in \mathcal{K}\\
&\quad\Leftrightarrow \forall j \in [n], \Pi_{\mathcal{H}^{(j)}}|w\rangle \in \mathcal{H}^{(j)}(x) \wedge \Pi_{\mathcal{H}^{(j)}}(|w\rangle - |w_0\rangle) \in \mathcal{K}^{(j)}\\
&\quad\Leftrightarrow \forall j \in [n], \sqrt{\alpha'_j}|w^{(j)}\rangle \in \mathcal{H}^{(j)}(x) \wedge \sqrt{\alpha'_j}|w^{(j)}\rangle - \sqrt{\alpha'_j}|w_0^{(j)}\rangle \in \mathcal{K}^{(j)}\\
&\quad\Leftrightarrow \forall j \in [n], |w^{(j)}\rangle \in \mathcal{H}^{(j)}(x) \wedge |w^{(j)}\rangle - |w_0^{(j)}\rangle \in \mathcal{K}^{(j)}\\
&\quad\Leftrightarrow \forall j \in [n], |w^{(j)}\rangle \text{ is a positive witness for } x \text{ in } \mathcal{P}^{(j)}.
\end{aligned}$$

This proves claim 2a. For claim 3a, we observe that we minimize the size of the positive witness if we minimize the size of each of its components, i.e., to that end we must choose the minimal positive witnesses for $x$ in each of the $\mathcal{P}^{(j)}$'s. The resulting size, then, becomes

$$w_+(x,\mathcal{P}) = \||w\rangle\|^2 = \sum_{j=1}^{n} \alpha_j \||w^{(j)}\rangle\|^2 = \sum_{j=1}^{n} \alpha_j w_+(x,\mathcal{P}^{(j)}),$$

proving claim 4a too.

Next, we turn to claims 2b, 3b and 4b. To that end, we characterize the negative witnesses in $\mathcal{P}$ for an input $x \in \mathcal{D}$. Let $|w\rangle \in \mathcal{H}$, and for all $j \in [n]$, write $|\overline{w}^{(j)}\rangle = \Pi_{\mathcal{H}^{(j)}}|w\rangle$ and $\beta_j = (\alpha'_j)^{1/2}\langle w_0^{(j)}|\overline{w}^{(j)}\rangle$. If $\beta_j \neq 0$, we also define $|w^{(j)}\rangle = \beta_j^{-1}(\alpha'_j)^{1/2}|\overline{w}^{(j)}\rangle$. Then, we observe that we have indeed decomposed $|w\rangle$ in the form of Equation 7.1.2.

For all $j \in [n]$ for which $\beta_j \neq 0$, by definition $\langle w_0^{(j)}|w^{(j)}\rangle = 1$, which implies that $|w^{(j)}\rangle$ is a negative witness for $x$ in $\mathcal{P}^{(j)}$ if and only if it is in $(\mathcal{K}^{(j)})^\perp \cap \mathcal{H}^{(j)}(x)^\perp$. It follows that

$$\begin{aligned}
&|w\rangle \text{ is a negative witness for } x \text{ in } \mathcal{P}\\
&\quad\Leftrightarrow |w\rangle \in \mathcal{K}^\perp \cap \mathcal{H}(x)^\perp \wedge \langle w_0|w\rangle = 1\\
&\quad\Leftrightarrow \underbrace{\forall j \in [n], |\overline{w}^{(j)}\rangle \in (\mathcal{K}^{(j)})^\perp \cap \mathcal{H}^{(j)}(x)^\perp}_{A} \wedge \underbrace{\sum_{j=1}^{n} \sqrt{\alpha'_j}\langle w_0^{(j)}|\overline{w}^{(j)}\rangle = 1.}_{B}
\end{aligned}$$

By the definition of $\beta_j$, we easily see that

$$B \Leftrightarrow \sum_{j=1}^{n} \beta_j = 1,$$

so it remains to further evaluate statement $A$. To that end, observe that if $\beta_j \neq 0$,

$|\overline{w}^{(j)}\rangle$ and $|w^{(j)}\rangle$ differ only by scalar multiplication, and so

$$A \Leftrightarrow \forall j \in [n], \begin{cases} |w^{(j)}\rangle \in (\mathcal{K}^{(j)})^\perp \cap \mathcal{H}^{(j)}(x)^\perp, & \text{if } \beta_j \neq 0, \\ |\overline{w}^{(j)}\rangle \in (\mathcal{K}^{(j)})^\perp \cap \mathcal{H}^{(j)}(x)^\perp \cap \mathrm{Span}\{|w_0^{(j)}\rangle\}^\perp, & \text{if } \beta_j = 0, \end{cases}$$

$$\Leftrightarrow \forall j \in [n], \begin{cases} |w^{(j)}\rangle \text{ is a negative witness in } \mathcal{P}^{(j)}, & \text{if } \beta_j \neq 0, \\ |\overline{w}^{(j)}\rangle \in (\mathcal{K}^{(j)})^\perp \cap \mathcal{H}^{(j)}(x)^\perp \cap \mathrm{Span}\{|w_0^{(j)}\rangle\}^\perp, & \text{if } \beta_j = 0. \end{cases}$$

This proves claim 2b.

For claim 3b, it remains to compute the minimal vector $|w\rangle \in \mathcal{H}$ that satisfies the constraints above. To that end, observe that if $f^{(j)}(x) = 1$, then there does not exist a negative witness $|w^{(j)}\rangle$ for $x$ in $\mathcal{P}^{(j)}$, and so the only possibility is that $\beta_j = 0$. Thus,

$$\begin{aligned} \||w\rangle\|^2 &\geq \sum_{\substack{j=1 \\ \beta_j \neq 0}}^{n} \frac{|\beta_j|^2}{\alpha'_j} \left\||w^{(j)}\rangle\right\|^2 \geq \sum_{\substack{j=1 \\ f^{(j)}(x)=0}}^{n} \frac{|\beta_j|^2}{\alpha'_j} w_-(x, \mathcal{P}^{(j)}) \\ &\geq \left|\sum_{j=1}^{n} \beta_j\right|^2 \cdot \left[\sum_{\substack{j=1 \\ f^{(j)}(x)=0}}^{n} \frac{\alpha'_j}{w_-(x, \mathcal{P}^{(j)})}\right]^{-1} = \left[\sum_{j=1}^{n} \frac{\alpha'_j}{w_-(x, \mathcal{P}^{(j)})}\right]^{-1}, \end{aligned}$$

where we used the Cauchy–Schwarz inequality[2], and $w_-(x, \mathcal{P}^{(j)}) = \infty$ whenever $f^{(j)}(x) = 1$. Moreover, we observe that we have equality in all inequalities above, when we choose all $|\overline{w}^{(j)}\rangle$'s to be 0, all $|w^{(j)}\rangle$'s to be the minimal negative witnesses for $x$ in $\mathcal{P}^{(j)}$, and

$$\beta_j = \frac{\alpha'_j}{w_-(x, \mathcal{P}^{(j)})} \cdot \left[\sum_{j=1}^{n} \frac{\alpha'_j}{w_-(x, \mathcal{P}^{(j)})}\right]^{-1},$$

for all $j \in [n]$. This completes the proof of claim 3b and 4b, and hence the proof is complete. □

Note that in item 5 of the above theorem, we use the *unnormalized* coefficients $\alpha_j$, rather than the normalized ones $\alpha'_j$, to upper bound the span program complexity $C(\mathcal{P})$. This is not a typo – rather, we observe here that the normalized coefficients $\alpha'_j$ arise in the numerator and denominator in the bound on the positive and negative witness complexity, respectively. Therefore, the normalization prefactor cancels in the upper bound of the span program complexity.

In the AND-construction in Definition 7.1.3, we require that all span programs have the same domain $\mathcal{D}$. However, the construction can also be used if we have

[2]This particular form of the Cauchy–Schwarz inequality is also known as Sedrakyan's inequality.

several span programs on different domains. For instance, suppose we have a span program $\mathcal{P}^{(1)}$ on $\mathcal{D}^{(1)}$ and $\mathcal{P}^{(2)}$ on $\mathcal{D}^{(2)}$, computing the boolean functions $f^{(1)} : \mathcal{D}^{(1)} \to \{0,1\}$ and $f^{(2)} : \mathcal{D}^{(2)} \to \{0,1\}$, respectively. Note that we can reinterpret the span program $\mathcal{P}^{(1)}$ to act on $\mathcal{D}^{(1)} \times \mathcal{D}^{(2)}$, where to every input $(x^{(1)}, x^{(2)}) \in \mathcal{D}^{(1)} \times \mathcal{D}^{(2)}$, we simply associate $\mathcal{H}((x^{(1)}, x^{(2)})) = \mathcal{H}(x^{(1)})$. If we do the same with $\mathcal{P}^{(2)}$ we now have two span programs that act on the same domain, and plugging them in the AND-composition yields a span program that computes the function $f : \mathcal{D}^{(1)} \times \mathcal{D}^{(2)} \to \{0,1\}$, defined as $f((x^{(1)}, x^{(2)})) = f^{(1)}(x^{(1)}) \wedge f^{(2)}(x^{(2)})$. Moreover, in this case the assumptions from claim 6 in Theorem 7.1.4 hold, and thus, if we choose the $\alpha_j$'s appropriately, the span program complexity becomes the squared sum of the individual ones.

One might wonder whether the above composition result admits a converse, i.e., if we have a span program that evaluates the function $f = \bigwedge_{j=1}^{n} f^{(j)}$, can we decompose the span program into $n$ individual span programs computing the functions $f^{(j)}$, for $j \in [n]$? And if we then plug these span programs back into the AND-composition, do we get back the original span program computing $f = \bigwedge_{j=1}^{n} f^{(j)}$ again? This is indeed an interesting question for future research, and we briefly revisit this question at the end of the next subsection.

Now that we know how to compute "AND"s and "NOT"s, we can use De Morgan's formula to compute "OR"s as well, as is presented in the following definition.

**7.1.5.** Definition (OR-composition of span programs).
Let $\mathcal{P}^{(1)}, \ldots, \mathcal{P}^{(n)}$ be span programs on some common domain $\mathcal{D}$, and we write $\mathcal{P}^{(j)} = (\mathcal{H}^{(j)}, x \mapsto \mathcal{H}^{(j)}(x), \mathcal{K}^{(j)}, |w_0^{(j)}\rangle)$ for all $j \in [n]$. Let $\alpha_1, \ldots, \alpha_n > 0$ be coefficients, and for all $j \in [n]$, we define the normalized coefficients

$$\alpha'_j = \frac{\alpha_j}{\sum_{j=1}^{n} \alpha_j}.$$

Now, we define the span program $\mathcal{P}$ by

$$\mathcal{P} = \neg \bigwedge_{j=1}^{n} \alpha_j(\neg \mathcal{P}^{(j)}),$$

and we refer to it as the OR-composition of span programs $(\mathcal{P}^{(j)})_{j=1}^{n}$ with coefficients $(\alpha_j)_{j=1}^{n}$. We denote $\mathcal{P}$ by the shorthand notations

$$\alpha_1 \mathcal{P}^{(1)} \vee \alpha_2 \mathcal{P}^{(2)} \vee \cdots \vee \alpha_n \mathcal{P}^{(n)}, \qquad \text{and} \qquad \bigvee_{j=1}^{n} \alpha_j \mathcal{P}^{(j)}.$$

Similarly as in the AND-composition, we omit $\alpha_j$ if it is 1, i.e., we write $\mathcal{P}^{(j)}$ instead of $1\mathcal{P}^{(j)}$. ◀

Most of the properties of the OR-composition follow directly from the properties of the AND-composition, proved in Theorem 7.1.4. For completeness and ease of reference, we include the full statement of its properties here.

**7.1.6.** Theorem (Properties of the OR-composition of span programs). *Let $\mathcal{P}^{(1)}, \dots, \mathcal{P}^{(n)}$ be span programs on a domain $\mathcal{D}$, where for all $j \in [n]$, $\mathcal{P}^{(j)}$ computes the function $f^{(j)} : \mathcal{D} \to \{0,1\}$. Let $\alpha_1, \dots, \alpha_n > 0$ be coefficients, and let their normalized counterparts $\alpha'_1, \dots, \alpha'_n$ be as in Definition 7.1.5. Let*

$$\mathcal{P} = \bigvee_{j=1}^{n} \alpha_j \mathcal{P}^{(j)},$$

*and we write $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$. Let $x \in \mathcal{D}$. Then, $\mathcal{P}$ has the following properties:*

1. *$\mathcal{P}$ computes $\bigvee_{j=1}^{n} f^{(j)}$.*

2a. *If $x$ is a positive input, then the positive witnesses for $x$ in $\mathcal{P}$ are exactly the vectors $|w\rangle \in \mathcal{H}$ of the form*

$$|w\rangle = \sum_{\substack{j=1 \\ \beta_j \neq 0}}^{n} \frac{\beta_j}{\sqrt{\alpha'_j}} |w^{(j)}\rangle + \sum_{\substack{j=1 \\ \beta_j = 0}}^{n} |\overline{w}^{(j)}\rangle, \tag{7.1.3}$$

*where for all $j \in [n]$, $\beta_j \in \mathbb{C}$ such that $\sum_{j=1}^{n} \beta_j = 1$, $|w^{(j)}\rangle$ is a positive witness for $x$ in $\mathcal{P}^{(j)}$, and $|\overline{w}^{(j)}\rangle \in \mathcal{K}^{(j)} \cap \mathcal{H}^{(j)}(x)$.*

2b. *If $x$ is a negative input, then the negative witnesses for $x$ in $\mathcal{P}$ are exactly the vectors $|w\rangle \in \mathcal{H}$ of the form*

$$|w\rangle = \sum_{j=1}^{n} \sqrt{\alpha'_j} |w^{(j)}\rangle, \tag{7.1.4}$$

*where for all $j \in [n]$, $|w^{(j)}\rangle$ is a negative witness for $x$ in $\mathcal{P}^{(j)}$.*

3a. *If $x$ is a positive input for $\mathcal{P}$, then the minimal positive witness for $x$ in $\mathcal{P}$ is $|w\rangle$ in Equation 7.1.3, where for all $j \in [n]$, we choose*

$$\beta_j = \frac{\alpha'_j}{w_+(x, \mathcal{P}^{(j)})} \cdot \left[ \sum_{j=1}^{n} \frac{\alpha'_j}{w_+(x, \mathcal{P}^{(j)})} \right]^{-1},$$

*$|w^{(j)}\rangle$ to be the minimal positive witness for $x$ in $\mathcal{P}^{(j)}$, and $|\overline{w}^{(j)}\rangle = 0$.*

3b. *If $x$ is a negative witness for $\mathcal{P}$, then the minimal negative witness for $x$ in $\mathcal{P}$ is $|w\rangle$ in Equation 7.1.4, where for every $j \in [n]$, we choose $|w^{(j)}\rangle$ to be the minimal negative witness for $x$ in $\mathcal{P}^{(j)}$.*

4. *We have*

$$w_+(x,\mathcal{P}) = \left[\sum_{j=1}^{n} \frac{\alpha'_j}{w_+(x,\mathcal{P}^{(j)})}\right]^{-1} \quad \text{and} \quad w_-(x,\mathcal{P}) = \sum_{j=1}^{n} \alpha'_j w_-(x,\mathcal{P}^{(j)}).$$

5. *Suppose that none of the $\mathcal{P}^{(j)}$'s nor $\mathcal{P}$ compute a constant function. Then,*

$$W_+(\mathcal{P}) \leq \max_{j\in[n]} \frac{W_+(\mathcal{P}^{(j)})}{\alpha'_j} \qquad \text{and} \qquad W_-(\mathcal{P}) \leq \sum_{j=1}^{n} \alpha'_j W_-(\mathcal{P}^{(j)}),$$

*and in particular,*

$$C(\mathcal{P})^2 \leq \sum_{j=1}^{n} \alpha_j W_-(\mathcal{P}^{(j)}) \cdot \max_{j\in[n]} \frac{W_+(\mathcal{P}^{(j)})}{\alpha_j}.$$

6. *Suppose that none of the $\mathcal{P}^{(j)}$'s compute a constant function, and suppose furthermore that we can decompose $\mathcal{D} = \bigtimes_{j=1}^{n} \mathcal{D}^{(j)}$, and that for every input $(x^{(1)},\ldots,x^{(n)}) = x \in \mathcal{D}$, $\mathcal{H}^{(j)}(x)$ only depends on $x^{(j)}$. Then, all inequalities from item 5 turn into equalities. Moreover, if for all $j \in [n]$, we choose $\alpha_j = W_+(\mathcal{P}^{(j)})$, then we minimize $C(\mathcal{P})$, and*

$$C(\mathcal{P})^2 = \sum_{j=1}^{n} C(\mathcal{P}^{(j)})^2.$$

7. *$\mathcal{P}$ is query-efficient if and only if for every $j \in [n]$, $\mathcal{P}^{(j)}$ is.*
8. *$U(x,\mathcal{P}) = (-1)^{n+1}(2\left|w_0\right\rangle\left\langle w_0\right| - I)\prod_{j=1}^{n}(2|w_0^{(j)}\rangle\langle w_0^{(j)}| - I)U(x,\mathcal{P}^{(j)})$.*

One interesting thing to note is that a distributive law does not seem to hold on the level of span programs. For instance, if $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$ are all span programs on some common domain $\mathcal{D}$, then

$$\mathcal{P}_1 \vee (\mathcal{P}_2 \wedge \mathcal{P}_3), \qquad \text{and} \qquad (\mathcal{P}_1 \vee \mathcal{P}_2) \wedge (\mathcal{P}_1 \vee \mathcal{P}_3)$$

do not have the same witnesses or witness sizes in general. On the contrary, in many situations one formula results in span programs with strictly smaller complexity compared to the other. Thus, even though we can in principle evaluate any boolean formula using the techniques introduced here, the specific form of this formula influences the complexity of the resulting span program non-trivially.

An immediate consequence of these composition results is that we can recover [Rei10, Corollary 1.6], i.e., any boolean function that can be represented as a formula of length $k$ has query complexity $\mathcal{O}(\sqrt{k})$.

**7.1.7.** Theorem. *Let $f : \{0,1\}^n \to \{0,1\}$ be a boolean function that can be represented as a formula of length $k$, i.e., it can be represented as a formula with $\wedge$, $\vee$'s and $\neg$'s, and $k$ variables. Then, $Q(f) = \mathcal{O}(\sqrt{k})$.*

**Proof:**
First, observe that any ¬'s that appear in the formula are insignificant, since if we have a span program that evaluates any given formula, we can build a span program with the same cost that evaluates the negation of this formula, using Definition 7.1.1. Moreover, by item 7 of Theorem 7.1.2, query-efficiency is preserved in this conversion.

We give a proof using induction to the length of the formula. To that end, we start by proving that the statement holds when the length of the formula is 1.

For all $j \in [n]$, we can use the construction from Example 6.1.4 to define a span program that computes the bit $x_j$, i.e., we let $\mathcal{P}^{(j)} = (\mathcal{H}^{(j)}, x \mapsto \mathcal{H}^{(j)}(x), \mathcal{K}, |w_0^{(j)}\rangle)$ on $\{0,1\}^n$, where

$$\mathcal{H} = \mathrm{Span}\{|j\rangle\}, \qquad \mathcal{H}(x) = \begin{cases} \mathrm{Span}\{|j\rangle\}, & \text{if } x_j = 1, \\ \{0\}, & \text{otherwise,} \end{cases}$$
$$\mathcal{K}^{(j)} = \{0\}, \qquad |w_0^{(j)}\rangle = |j\rangle\,.$$

Then, $\mathcal{P}^{(j)}$ is query-efficient, evaluates the length-1 formula $x_j$, and $W_+(\mathcal{P}^{(j)}) = W_-(\mathcal{P}^{(j)}) = C(\mathcal{P}^{(j)}) = 1$. This provides our basis for induction.

Next, suppose that for any formula of length $\ell$, where $\ell \leq k$, we can construct a query-efficient span program that computes this formula with complexity at most $\sqrt{\ell}$. Next, suppose that we have a formula $f$ of length $k+1$. We can break up this formula in two parts, i.e., we can write the formula as $f$ as $f_1 \wedge f_2$, or $f_1 \vee f_2$, with lengths $\ell_1$ and $\ell_2$, respectively, such that $\ell_1 + \ell_2 = k+1$. The two cases are very similar, so we only provide the proof for the case where we have $f = f_1 \wedge f_2$.

Now, by our induction hypothesis, we can find query-efficient span programs $\mathcal{P}_1$ and $\mathcal{P}_2$ that evaluate $f_1$ and $f_2$, with complexities at most $\sqrt{\ell_1}$ and $\sqrt{\ell_2}$, respectively. Then, we let $\alpha_1 = W_-(\mathcal{P}_1)$ and $\alpha_2 = W_-(\mathcal{P}_2)$, and from item 5 in Theorem 7.1.4 we find that with $\mathcal{P} = \alpha_1\mathcal{P}_1 \wedge \alpha_2\mathcal{P}_2$,

$$\begin{aligned} C(\mathcal{P})^2 &\leq (\alpha_1 W_+(\mathcal{P}_1) + \alpha_2 W_+(\mathcal{P}_2)) \cdot \max\left\{\frac{W_-(\mathcal{P}_1)}{\alpha_1}, \frac{W_-(\mathcal{P}_2)}{\alpha_2}\right\} \\ &= W_-(\mathcal{P}_1)W_+(\mathcal{P}_1) + W_-(\mathcal{P}_2)W_+(\mathcal{P}_2) = C(\mathcal{P}_1)^2 + C(\mathcal{P}_2)^2 \\ &= \ell_1 + \ell_2 = k+1. \end{aligned}$$

Moreover, by item 7 of Theorem 7.1.4, we find that $\mathcal{P}$ is query-efficient. Thus, we have constructed a query-efficient span program that evaluates a formula of length $k+1$ with complexity $\sqrt{k+1}$, completing the proof of the induction step. Thus, we find that for any formula of length $k \in \mathbb{N}$, we can find a query-efficient span program that computes this formula with complexity at most $\sqrt{k}$.

Finally, We know from Algorithm 6.1.18 and Lemma 6.2.8 that for query-efficient span programs, the query complexity of the function it computes is upper bounded by the span program complexity, up to constants. Thus, we find that

any formula of length $k$ can be computed using a quantum algorithm making at most $\mathcal{O}(\sqrt{k})$ queries. This completes the proof. □

As a further showcase of these techniques, we give the missing proof of Theorem 6.1.17.

**7.1.8.** Theorem (Witness anatomy of renormalized span programs).
*Let $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ be a span program, and $\beta > 0$. Let $\mathcal{P}'$ be $\mathcal{P}$ renormalized with constant $\beta$, as defined in Definition 6.1.16. Let $x \in \mathcal{D}$. Then:*

1. *$\mathcal{P}$ and $\mathcal{P}'$ compute the same function.*
2. *If $x$ is a positive input, then the positive witnesses for $x$ in $\mathcal{P}'$ are the vectors $\sqrt{1 + 1/\beta^2}\,|w\rangle$, where $|w\rangle$ is a positive witness for $x$ in $\mathcal{P}$.*
3. *If $x$ is a negative input, then the negative witnesses for $x$ in $\mathcal{P}'$ are the vectors $(\beta\,|w\rangle + |*\rangle)/\sqrt{1+\beta^2}$, where $|w\rangle$ is a negative witness for $x$ in $\mathcal{P}$.*

**Proof:**
Let $x \in \mathcal{D}$, and define

$$\mathcal{H} = \mathrm{Span}\{|*\rangle\}, \quad \mathcal{H}(x) = \{0\}, \quad \mathcal{K} = \{0\}, \quad |w_0\rangle = |*\rangle\,.$$

Let $\mathcal{T} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ be a span program. Then $\mathcal{T}$ computes the constant function $x \mapsto 0$.

By comparing Definition 6.1.16 with the composition results introduced in this section, we observe that $\mathcal{P}' = \beta^2\mathcal{P} \vee \mathcal{T}$. Thus, by claim 1 of Theorem 7.1.6, $\mathcal{P}'$ computes the same function as $\mathcal{P}$, and by claims 2a and 2b of said theorem, the positive witnesses for $\mathcal{P}'$ are the vectors $\sqrt{1 + 1/\beta^2}\,|w\rangle$, where $|w\rangle$ is a positive witness in $\mathcal{P}$, and the negative witnesses are the vectors $(\beta\,|w\rangle + |*\rangle)/\sqrt{1+\beta^2}$, where $|w\rangle$ is a negative witness in $\mathcal{P}$. This completes the proof. □

Note that in the proof of the previous theorem, we could not use the results from claims 5 and 6 of Theorem 7.1.6, since the OR-composition involved the span program $\mathcal{T}$, which computes a constant function. Indeed, if we compute the complexity of $\mathcal{P}'$, we can see that it does not satisfy the formulae in these claims.

Now that we know how to compute AND's and OR's of functions, it is a natural question to ask whether we can also compute the XOR of two functions. The following theorem shows that this can indeed be done, using both the AND and OR constructions that were introduced earlier.

**7.1.9.** Theorem (XOR-composition of span programs). *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be span programs on a common domain $\mathcal{D}$, computing functions $f_1, f_2 : \mathcal{D} \to \{0,1\}$, respectively. Let $\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2 > 0$ be defined by*

$$\alpha_1 = \sqrt{W_-(\mathcal{P}_1)W_-(\mathcal{P}_2)}, \qquad \alpha_2 = \sqrt{W_+(\mathcal{P}_1)W_+(\mathcal{P}_2)},$$
$$\beta_1 = \sqrt{W_+(\mathcal{P}_1)W_+(\mathcal{P}_2)}, \qquad \beta_2 = \sqrt{W_-(\mathcal{P}_1)W_-(\mathcal{P}_2)},$$
$$\gamma_1 = \alpha_1 W_+(\mathcal{P}_1) + \alpha_2 W_-(\mathcal{P}_2), \qquad \gamma_2 = \beta_1 W_-(\mathcal{P}_1) + \beta_2 W_+(\mathcal{P}_2).$$

*Next, let*

$$\mathcal{P} = \gamma_1(\alpha_1\mathcal{P}_1 \wedge \alpha_2(\neg\mathcal{P}_2)) \vee \gamma_2(\beta_1(\neg\mathcal{P}_1) \wedge \beta_2\mathcal{P}_2) =: \mathrm{XOR}(\mathcal{P}_1, \mathcal{P}_2).$$

*Then, $\mathcal{P}$ computes $f : \mathcal{D} \to \{0,1\}$,*

$$f(x) = \begin{cases} 1, & \text{if } f_1(x) \neq f_2(x), \\ 0, & \text{otherwise}, \end{cases}$$

*and*

$$C(\mathcal{P}) \leq C(\mathcal{P}_1) + C(\mathcal{P}_2).$$

**Proof:**
We verify immediately from the first claims of Theorems 7.1.4 and 7.1.6 that $\mathcal{P}$ indeed computes $f$. Thus, it remains to check the claim on its complexity, i.e., it remains to upper bound $C(\mathcal{P})$. First, we define

$$N_\gamma = \gamma_1 + \gamma_2, \qquad \text{and} \qquad N_{\alpha\beta} = \alpha_1 + \alpha_2 = \beta_1 + \beta_2.$$

Now, let $x \in \mathcal{D}$. Using the formulae from claims 4 in Theorems 7.1.2, 7.1.4 and 7.1.6, we find that

$$\begin{aligned} w_+(x,\mathcal{P}) &= N_\gamma \left[\frac{\gamma_1}{w_+(x,\alpha_1\mathcal{P}_1 \wedge \alpha_2(\neg\mathcal{P}_2))} + \frac{\gamma_2}{w_+(x,\beta_1(\neg\mathcal{P}_1) \wedge \beta_2\mathcal{P}_2)}\right]^{-1} \\ &= N_\gamma N_{\alpha\beta}^{-1}\left[\frac{\gamma_1}{\alpha_1 w_+(x,\mathcal{P}_1) + \alpha_2 w_-(x,\mathcal{P}_2)} + \frac{\gamma_2}{\beta_1 w_-(x,\mathcal{P}_1) + \beta_2 w_+(x,\mathcal{P}_2)}\right]^{-1}, \end{aligned}$$

and similarly we find that

$$\begin{aligned} w_-(x,\mathcal{P}) &= N_\gamma^{-1}[\gamma_1 w_-(x,\alpha_1\mathcal{P}_1 \wedge \alpha_2(\neg\mathcal{P}_2)) + \gamma_2 w_-(x,\beta_1(\neg\mathcal{P}_1) \wedge \beta_2\mathcal{P}_2)] \\ &= N_\gamma^{-1}N_{\alpha\beta}\left[\gamma_1\left[\frac{\alpha_1}{w_-(x,\mathcal{P}_1)} + \frac{\alpha_2}{w_+(x,\mathcal{P}_2)}\right]^{-1} + \gamma_2\left[\frac{\beta_1}{w_+(x,\mathcal{P}_1)} + \frac{\beta_2}{w_-(x,\mathcal{P}_2)}\right]^{-1}\right]. \end{aligned}$$

Now, we distinguish four cases. First, suppose that $f_1(x) = f_2(x) = 1$. Then,

$$\begin{aligned} w_-(x,\mathcal{P}) &= N_\gamma^{-1}N_{\alpha\beta}\left[\gamma_1\left[0 + \frac{\alpha_2}{w_+(x,\mathcal{P}_2)}\right]^{-1} + \gamma_2\left[\frac{\beta_1}{w_+(x,\mathcal{P}_1)} + 0\right]^{-1}\right] \\ &\leq N_\gamma^{-1}N_{\alpha\beta}\left[\frac{\gamma_1}{\alpha_2}W_+(\mathcal{P}_2) + \frac{\gamma_2}{\beta_1}W_+(\mathcal{P}_1)\right] \\ &= N_\gamma^{-1}N_{\alpha\beta}\left[\left[\frac{\alpha_1}{\alpha_2} + \frac{\beta_2}{\beta_1}\right]W_+(\mathcal{P}_1)W_+(\mathcal{P}_2) + W_-(\mathcal{P}_2)W_+(\mathcal{P}_2) + W_-(\mathcal{P}_1)W_+(\mathcal{P}_1)\right] \\ &= N_\gamma^{-1}N_{\alpha\beta}\left[2C(\mathcal{P}_1)C(\mathcal{P}_2) + C(\mathcal{P}_1)^2 + C(\mathcal{P}_2)^2\right] = N_\gamma^{-1}N_{\alpha\beta}[C(\mathcal{P}_1) + C(\mathcal{P}_2)]^2. \end{aligned}$$

Similarly, if $f_1(x) = f_2(x) = 0$, then

$$\begin{aligned}
w_-(x,\mathcal{P}) &= N_\gamma^{-1} N_{\alpha\beta} \left[ \gamma_1 \left[ \frac{\alpha_1}{w_-(x,\mathcal{P}_1)} + 0 \right]^{-1} + \gamma_2 \left[ 0 + \frac{\beta_2}{w_-(x,\mathcal{P}_2)} \right]^{-1} \right] \\
&\leq N_\gamma^{-1} N_{\alpha\beta} \left[ \frac{\gamma_1}{\alpha_1} W_-(\mathcal{P}_1) + \frac{\gamma_2}{\beta_2} W_-(\mathcal{P}_2) \right] \\
&= N_\gamma^{-1} N_{\alpha\beta} \left[ \left[ \frac{\alpha_2}{\alpha_1} + \frac{\beta_1}{\beta_2} \right] W_-(\mathcal{P}_1) W_-(\mathcal{P}_2) + W_-(\mathcal{P}_2) W_+(\mathcal{P}_2) + W_-(\mathcal{P}_1) W_+(\mathcal{P}_1) \right] \\
&= N_\gamma^{-1} N_{\alpha\beta} \left[ 2C(\mathcal{P}_1)C(\mathcal{P}_2) + C(\mathcal{P}_1)^2 + C(\mathcal{P}_2)^2 \right] = N_\gamma^{-1} N_{\alpha\beta} [C(\mathcal{P}_1) + C(\mathcal{P}_2)]^2.
\end{aligned}$$

On the other hand, if $f_1(x) = 0$ and $f_2(x) = 1$, then

$$\begin{aligned}
w_+(x,\mathcal{P}) &= N_\gamma N_{\alpha\beta}^{-1} \left[ 0 + \frac{\gamma_2}{\beta_1 w_-(x,\mathcal{P}_1) + \beta_2 w_+(x,\mathcal{P}_2)} \right]^{-1} \\
&\leq N_\gamma N_{\alpha\beta}^{-1} \cdot \frac{\beta_1 W_-(\mathcal{P}_1) + \beta_2 W_+(\mathcal{P}_2)}{\gamma_2} = N_\gamma N_{\alpha\beta}^{-1},
\end{aligned}$$

and similarly, if $f_1(x) = 1$ and $f_2(x) = 0$, then

$$\begin{aligned}
w_+(x,\mathcal{P}) &= N_\gamma N_{\alpha\beta}^{-1} \left[ \frac{\gamma_1}{\alpha_1 w_+(x,\mathcal{P}_1) + \alpha_2 w_-(x,\mathcal{P}_2)} + 0 \right]^{-1} \\
&\leq N_\gamma N_{\alpha\beta}^{-1} \cdot \frac{\alpha_1 W_+(\mathcal{P}_1) + \alpha_2 W_-(\mathcal{P}_2)}{\gamma_1} = N_\gamma N_{\alpha\beta}^{-1}.
\end{aligned}$$

Thus, we arrive at

$$\begin{aligned}
C(\mathcal{P}) &= \sqrt{W_+(\mathcal{P}) W_-(\mathcal{P})} \leq \sqrt{N_\gamma N_{\alpha\beta}^{-1} \cdot N_\gamma^{-1} N_{\alpha\beta} [C(\mathcal{P}_1) + C(\mathcal{P}_2)]^2} \\
&= C(\mathcal{P}_1) + C(\mathcal{P}_2).
\end{aligned}$$

This completes the proof. □

As a final showcase of these composition techniques, we demonstrate how one can compute the parity of span programs. That is, if we have span programs $\mathcal{P}^{(1)}, \ldots, \mathcal{P}^{(n)}$ on a common domain $\mathcal{D}$. We devise a span program that computes whether on any given input $x \in \mathcal{D}$, the number of these $n$ span programs for which this is a positive input, is even or odd.

**7.1.10.** Corollary (PARITY-composition of span programs).
*Let $\mathcal{P}_1, \ldots, \mathcal{P}_n$ be span programs on a common domain $\mathcal{D}$, where for all $j \in [n]$, $\mathcal{P}_j$ computes the function $f_j : \mathcal{D} \to \{0,1\}$. We define $f : \mathcal{D} \to \{0,1\}$, such that for all $x \in \mathcal{D}$,*

$$f(x) = \begin{cases} 1, & \text{if } |\{j \in [n] : f_j(x) = 1\}| \text{ is odd,} \\ 0, & \text{if } |\{j \in [n] : f_j(x) = 1\}| \text{ is even.} \end{cases}$$

*Then, we define* $\mathcal{T}_1 = \mathcal{P}_1$*, and we inductively define* $\mathcal{T}_{j+1} = \mathrm{XOR}(\mathcal{P}_{j+1}, \mathcal{T}_j)$*, for all* $j \in [n-1]$*. We label* $\mathcal{P} = \mathcal{T}_n$*. It follows by the principle of mathematical induction that* $\mathcal{P}$ *computes* $f$*, with complexity*

$$C(\mathcal{P}) = \sum_{j=1}^{n} C(\mathcal{P}_j).$$

We conclude this subsection by making a couple of final observations. The logical composition results we developed here are all on the level of span programs. Using common techniques, one can also achieve such logical compositions directly on the level of quantum algorithms. For instance, if we want to compute the parity of $n$ functions $f_j : \mathcal{D} \to \{0,1\}$, and for every $j \in [n]$ we have a quantum algorithm $\mathcal{A}_j$ that computes the function $f_j$, then one approach we could take is to simply run each of the algorithms $\mathcal{A}_j$ once, count the number of 1's we obtained in the process, and subsequently output whether this number is even or odd.

If all the algorithms $\mathcal{A}_j$ are exact, i.e., they output the right answer with certainty, then this yields a similar result to Corollary 7.1.10, i.e., we can compute the parity of the $n$ function $f_j$, using the sum of the number of queries required to compute each of the individual functions. Requiring that all the algorithms $\mathcal{A}_j$ are exact, though, is quite a restrictive assumption, since for many functions it is known that exact computation is significantly harder than performing the same task with bounded error.

The situation slightly changes, however, when we assume that each of the algorithms $\mathcal{A}_j$ merely computes the function $f_j$ with bounded error, say with success probability at least $2/3$. Then, running each of the algorithms $\mathcal{A}_j$ independently means that we can expect all of the algorithms $\mathcal{A}_j$ to succeed simultaneously only with a probability that is exponentially small in $n$. Thus, if we use separate independent runs of the individual algorithms $\mathcal{A}_j$, we can merely guarantee that we correctly compute the parity of all the outcomes with probability at least $1/2 + \mathcal{O}(\exp(-n))$, i.e., we only obtain an exponentially small advantage in terms of the success probability over randomly guessing 0 or 1.

The typical solution, then, is to wrap each of the algorithms $\mathcal{A}_j$ in an error-reduction routine. For instance, one can run the algorithm $\mathcal{A}_j$ a number of times, and take the majority of the observed outcomes. This exponentially suppresses the failure probability, i.e., if we run $\mathcal{A}_j$ a total of $N$ times, and take the majority of the outcomes, the failure probability is brought down to $\mathcal{O}(\exp(-N))$. From this, it follows that if we run each of the algorithms $\mathcal{A}_j$ $N = \mathcal{O}(\log(n))$ times, we can compute the parity of the $n$ functions with bounded error. In short, the error reduction techniques introduce a multiplicative logarithmic overhead in the query complexity.[3]

[3]For the specific setting where we want to know the parity of the outcomes of $n$ algorithms, the extra $\log(n)$ overhead can be avoided using different techniques, as shown in [BNR+07].

The beauty of span program composition is that this multiplicative logarithmic overhead can be avoided. If instead of having quantum algorithms $\mathcal{A}_j$ computing the functions $f_j$, we have query-efficient span programs $\mathcal{P}_j$ that evaluate $f_j$, then we can use the parity composition from Corollary 7.1.10, and subsequently the conversion from span programs to bounded-error quantum query algorithms, Algorithm 6.1.18, to obtain a quantum algorithm that computes the parity of the $n$ functions with a number of queries that truly is the sum of the span program complexities, without multiplicative logarithmic overhead. This highlights exactly the true higher-level benefit of using span programs, as they ease the process of taking individual algorithmic components, and stitching them together to compute more involved problems.

In the above discussion, we used computing the parity of function outcomes as an example, but using the AND- and OR-constructions presented in this chapter, similar arguments can be made for computing the AND and OR on $n$ function outputs. Furthermore, there are most likely more intricate logical composition results to be discovered. For instance, it would be interesting to figure out how to build a threshold-composition, i.e., given $n$ span programs $\mathcal{P}_j$ evaluating functions $f_j$, can we build a span program that decides whether at least $k$ of the function outputs are 1? This is a very interesting question for future research.

The inherent precondition for all these composition results, though, is that we can construct these individual span programs $\mathcal{P}_j$, computing the functions $f_j$, in the first place. This is in general a difficult task, but it motivates why, even if we have efficient quantum algorithms that compute a particular boolean function $f : \mathcal{D} \to \{0, 1\}$, there is still inherent value in coming up with an explicit construction of a span program that evaluates the given function with optimal complexity. For many functions, it is not completely understood what an optimal span program computing it looks like. We make some progress on this question for several specific functions later on in this chapter, but we already mention here that this is a very interesting direction for future research too.

### 7.1.2 Relation to dual adversary bound solutions

Recall that the NOT-, AND- and OR-compositions introduced in the previous subsection preserve query-efficiency, and that by virtue of Theorems 6.2.9 and 6.2.10, query-efficient span programs over real Hilbert spaces have a one-to-one correspondence with solutions to the reformulated dual adversary bound, as defined in Theorem 6.2.6. This naturally leads to the realization that we can just as well interpret these composition results on the dual adversary bound level, and perform these compositions with its solutions.

More specifically, let $f^{(1)}, \ldots, f^{(m)} : \mathcal{D} \to \{0, 1\}$ be $m$ boolean functions with a common domain $\mathcal{D}$, and suppose that we want to generate a solution to the dual adversary bound for the function $f = f^{(1)} \wedge \cdots \wedge f^{(m)}$. Given feasible solutions to the reformulated dual adversary bound for the functions $f^{(j)}$ with $j \in [m]$, we

can convert these into query-efficient span programs, compose the span programs using the AND-composition, i.e., Theorem 7.1.4, and then convert the resulting span program back into a feasible solution to the reformulated dual adversary bound for $f$.

The core thing to realize is that the connection between dual adversary bound solutions and query-efficient span programs is through their Gram vectorization and witnesses, as proved in Theorems 6.2.9 and 6.2.10. Thus, by understanding how the witnesses compose during the composition results derived in the previous subsection, we can understand how the dual adversary bound solutions can be composed as well.

We start by stating the span program negation result, i.e., Theorem 7.1.2, in terms of feasible solutions to the dual adversary bound. From said theorem, we observe that the witnesses are not changed during the negation process, so therefore we also don't need to change the adversary bound solution itself. Thus, we straightforwardly arrive at the following theorem.

**7.1.11.** Theorem (Negation of dual adversary bound solutions).
*Let $f : \mathcal{D} \to \{0,1\}$, with $\mathcal{D} \subseteq \{0,1\}^n$. Let $X_1, \dots, X_n$ be a feasible solution to the reformulated dual adversary bound for $f$. Then $X_1, \dots, X_n$ is also a solution to the dual adversary bound for $\neg f$, with the same objective value.*

**Proof:**
The result can be directly observed from the statement of the reformulated dual adversary bound, i.e., Theorem 6.2.6. □

Strictly speaking, the above result proves $\mathrm{ADV}^{\pm}(\neg f) \leq \mathrm{ADV}^{\pm}(f)$. However, if we again apply the same result to the function $\neg f$ and observe that $\neg\neg f = f$, we also obtain the reverse inequality, and hence have proved that

$$\mathrm{ADV}^{\pm}(\neg f) = \mathrm{ADV}^{\pm}(f). \tag{7.1.5}$$

Next, we turn to the AND-composition, i.e., Theorem 7.1.4. The goal is to phrase this theorem completely in terms of feasible solutions to the reformulated dual adversary bound. The resulting composition construction on the level of dual adversary bound solutions is implicitly implied through the results in [Rei09; Bel14; Bel15], but to the best of our knowledge they are not explicitly exhibited.

Before stating and proving the result formally, we first sketch the resulting construction. To that end, let $\mathcal{D} \subseteq \{0,1\}^n$ and for all $j \in [m]$, let $f^{(j)} : \mathcal{D} \to \{0,1\}$, and let $X_1^{(j)}, \dots, X_n^{(j)}$ be a feasible solution to the dual adversary bound for the function $f^{(j)}$. We wish to find a feasible solution $X_1, \dots, X_n$ for the reformulated dual adversary bound for the function $f = \bigwedge_{j=1}^{m} f^{(j)}$.

To that end, we let $\alpha_1, \dots, \alpha_m > 0$, like in Theorem 7.1.4. Next, we write the Gram vectorization $X_k^{(j)} = (W_k^{(j)})^\dagger W_k^{(j)}$, and we recall that the columns of $W_k^{(j)}$ define the witnesses of a query-efficient span program compiled from the

dual adversary bound solution. Then, we observe that Equation 7.1.1 and Equation 7.1.2 characterize the positive and negative witnesses for the composed span program. Inspired by these relations and using the notation from there, we multiply the columns of $W_k^{(j)}$ by the constants that appear in the characterizations, as displayed in Figure 7.1.1, to obtain the matrix $(W_k^{(j)})'$. Next, we define $(X_k^{(j)})' = ((W_k^{(j)})')^\dagger (W_k^{(j)})'$, which is displayed in the same figure. Finally, we define the matrices that feature the solution to the reformulated dual adversary bound as $X_k = \sum_{j=1}^m (X_k^{(j)})'$.

$(W_k^{(j)})' =$

| | $f(x)=1$, $f^{(j)}(x)=1$ | $f(x)=0$, $f^{(j)}(x)=1$ | $f(x)=0$, $f^{(j)}(x)=0$ |
|---|---|---|---|
| | Multiply by $\sqrt{\alpha'_j}$ | Set to 0 | Multiply by $\frac{\beta_j[x]}{\sqrt{\alpha'_j}}$ |

$(X_k^{(j)})' =$

| | $f(y)=1$, $f^{(j)}(y)=1$ | $f(y)=0$, $f^{(j)}(y)=1$ | $f(y)=0$, $f^{(j)}(y)=0$ |
|---|---|---|---|
| $f(x)=1$, $f^{(j)}(x)=1$ | Multiply by $\alpha'_j$ | Set to 0 | Multiply by $\beta_j[y]$ |
| $f(x)=0$, $f^{(j)}(x)=1$ | Set to 0 | Set to 0 | Set to 0 |
| $f(x)=0$, $f^{(j)}(x)=0$ | Multiply by $\beta_j[x]$ | Set to 0 | Multiply by $\frac{\beta_j[x]\beta_j[y]}{\alpha'_j}$ |

Figure 7.1.1: Schematic overview of how to obtain $(W_k^{(j)})'$ and $(X_k^{(j)})'$ from $W_k^{(j)}$ and $X_k^{(j)}$, respectively. The matrices displayed are $W_k^{(j)}$ and $X_k^{(j)}$, and after applying the operations indicated in the figure, one obtains $(W_k^{(j)})'$ and $(X_k^{(j)})'$. The constants $\alpha'_1, \ldots, \alpha'_m > 0$ sum to 1, and the constants $\beta_j[x] \in \mathbb{R}$ are chosen such that $\sum_{j \in J} \beta_j[x] = 1$, for all $x \in f^{-1}(0)$, where $J = \{j \in [n] : f^{(j)}(x) = 0\}$.

Now, we present the result formally.

**7.1.12.** THEOREM (AND-composition of dual adversary bound solutions).
*Let $f^{(1)}, \dots, f^{(m)} : \mathcal{D} \to \{0,1\}$ be $m$ boolean functions on a common domain $\mathcal{D} \subseteq \{0,1\}^n$. For all $j \in [m]$, let $X_1^{(j)}, \dots, X_n^{(j)}$ be a feasible solution to the reformulated dual adversary bound for the function $f^{(j)}$, with objective value $A^{(j)}$. Let*

$$f = \bigwedge_{j=1}^{m} f^{(j)} : \mathcal{D} \to \{0,1\}.$$

*Next, let $\alpha_1, \dots, \alpha_m > 0$, and define, for all $j \in [m]$,*

$$\alpha'_j = \frac{\alpha_j}{\sum_{k=1}^m \alpha_k}.$$

*Furthermore, for all $j \in [m]$ and $y \in (f^{(j)})^{-1}(0)$, let $\beta_j[y] \in \mathbb{R}$ be such that for all $x \in f^{-1}(0)$,*

$$\sum_{\substack{j=1 \\ f^{(j)}(x)=0}}^{m} \beta_j[x] = 1.$$

*For all $j \in [m]$ and $k \in [n]$, we define the matrix $(X_k^{(j)})' \in \mathbb{R}^{\mathcal{D}\times\mathcal{D}}$ as*

$$(X_k^{(j)})'[x,y] = \begin{cases} \alpha'_j \cdot X_k^{(j)}[x,y], & \text{if } f(x) = 1 \wedge f(y) = 1, \\ 0, & \text{if } f(x) = 0 \wedge f^{(j)}(x) = 1, \\ 0, & \text{if } f(y) = 0 \wedge f^{(j)}(y) = 1, \\ \beta_j[x] \cdot X_k^{(j)}[x,y], & \text{if } f(y) = 1 \wedge f(x) = 0 \wedge f^{(j)}(x) = 0, \\ \beta_j[y] \cdot X_k^{(j)}[x,y], & \text{if } f(x) = 1 \wedge f(y) = 0 \wedge f^{(j)}(y) = 0, \\ \frac{\beta_j[x]\beta_j[y]}{\alpha'_j} \cdot X_k^{(j)}[x,y], & \text{if } f(x) = f(y) = 0 \wedge f^{(j)}(x) = f^{(j)}(y) = 0. \end{cases}$$

*Finally, for all $k \in [n]$, we define $X'_k \in \mathbb{R}^{\mathcal{D}\times\mathcal{D}}$ as*

$$X_k = \sum_{j=1}^{m} (X_k^{(j)})'.$$

*Then,*

1. *$X_1, \dots, X_n$ is a solution to the reformulated dual adversary bound for $f$.*
2. *The optimal choice for the coefficients $\beta_j[x]$ is, for all $j \in [m]$ and all $x \in (f^{(j)})^{-1}(0)$,*

$$\beta_j[x] = \frac{\alpha'_j}{\sum_{k=1}^n X_k^{(j)}[x,x]} \cdot \left[ \sum_{j=1}^{m} \frac{\alpha'_j}{\sum_{k=1}^n X_k^{(j)}[x,x]} \right]^{-1}.$$

*We assume that we use these $\beta_j[x]$'s in subsequent claims.*

3. *If for all $j \in [m]$, $f^{(j)}$ are all non-constant, and $f$ is non-constant either, then the objective value $A$ of the solution to the reformulated dual adversary bound for $f$ satisfies*

$$A^2 \leq \left[\sum_{j=1}^{m} \alpha_j \max_{x \in (f^{(j)})^{-1}(1)} \sum_{k=1}^{n} X_k^{(j)}[x,x]\right] \cdot \left[\max_{j \in [m]} \frac{1}{\alpha_j} \max_{x \in (f^{(j)})^{-1}(0)} \sum_{k=1}^{n} X_k^{(j)}[x,x]\right].$$

4. *Suppose that for all $j \in [m]$, $f^{(j)}$ is non-constant, and that we can decompose $n = n_1 + \cdots + n_m$, for positive integers $n_j$. Furthermore, suppose that $\mathcal{D} = \bigtimes_{j=1}^{m} \mathcal{D}^{(j)}$ where for all $j \in [m]$, $\mathcal{D}^{(j)} \subseteq \{0,1\}^{n_j}$, such that $f^{(j)}$ only depends on $\mathcal{D}^{(j)}$. Then the above inequality becomes an equality, and the optimal choice for the $\alpha_j$'s becomes*

$$\alpha_j = \max_{x \in (f^{(j)})^{-1}(0)} \sum_{k=1}^{n} X_k^{(j)}[x,x],$$

*for all $j \in [m]$, in which case*

$$A^2 = \sum_{j=1}^{m} (A^{(j)})^2.$$

**Proof:**
For every $j \in [m]$, we first use Theorem 6.2.10 to generate a query-efficient span program $\mathcal{P}^{(j)}$ that computes $f^{(j)}$ from the dual adversary bound solution $X_1^{(j)}, \ldots, X_n^{(j)}$. Next, we compose the span programs using Theorem 7.1.4, with coefficients $\alpha_1, \ldots, \alpha_m$, and we denote the resulting span program by $\mathcal{P}$. From claim 7 in Theorem 7.1.4, we find that $\mathcal{P}$ is query-efficient, and hence we can use Theorem 6.2.9 to turn $\mathcal{P}$ into a solution to the reformulated dual adversary bound $X_1, \ldots, X_n$.

We can check by direct calculation from claims 3a and 3b in Theorem 7.1.4 that the resulting matrices $X_k$ defined in the theorem statement indeed equal those that come out of the span program construction. Then, claims 1, 2, 3 and 4 above are simply claims 1, 3, 5 and 6 of Theorem 7.1.4, reformulated in the new language. This completes the proof. □

We have now seen two statements of the AND-composition result, i.e., one on the span program level, Theorem 7.1.4, and one on the dual adversary bound level Theorem 7.1.12. Even though they both have the same theoretical implications, one could argue that the composition result looks a bit more natural on the span program level, since the conversion from $X_k^{(j)}$ to $(X_k^{(j)})'$ in Theorem 7.1.12 looks fairly non-trivial.

By feeding the optimal solutions to the reformulated dual adversary bound for $f^{(j)}$ in the above composition result, we find that for non-constant functions

$f^{(j)}$,

$$\mathrm{ADV}^{\pm}\left(\bigwedge_{j=1}^{m} f^{(j)}\right)^2 \leq \sum_{j=1}^{m} \mathrm{ADV}^{\pm}(f^{(j)})^2. \tag{7.1.6}$$

One might wonder whether the above inequality can be turned into an equality, i.e., whether the span-program-based construction is indeed optimal regardless of the choice of the functions $f^{(j)}$. This turns out to be the case as long as these functions $f^{(j)}$ have mutually disjoint support, as imposed by claim 3 of Theorem 7.1.12.

In order to show the reverse inequality of Equation (7.1.6), we turn to the reformulated *primal* adversary bound, i.e., Theorem 6.2.4. Specifically, we show that the reformulated primal adversary bound admits a similar AND-construction as the result in Theorem 7.1.12. The resulting theorem, Theorem 7.1.13, can already implicitly be obtained from the results presented in [HLŠ07; Rei09; Bel14; Bel15], but here we give an explicit construction.

**7.1.13.** Theorem (AND-composition of primal adversary bound solutions). *Let $n_1, \ldots, n_m$ be such that $n = n_1 + \cdots + n_m$. Next, for every $j \in [m]$, let $\mathcal{D}^{(j)} \subseteq \{0,1\}^{n_j}$ and $f^{(j)} : \mathcal{D}^{(j)} \to \{0,1\}$ be non-constant. Finally, we define $\mathcal{D} = \bigtimes_{j=1}^{m} \mathcal{D}^{(j)}$, and we let*

$$f = \bigwedge_{j=1}^{m} f^{(j)} : \mathcal{D} \to \{0,1\}, \qquad f(x) = \bigwedge_{j=1}^{m} f^{(j)}(x^{(j)}),$$

*for all $(x^{(1)}, \ldots, x^{(m)}) = x \in \mathcal{D}$. Let $\alpha_1, \ldots, \alpha_m > 0$ be strictly positive coefficients, and let their normalized versions be, for all $j \in [m]$,*

$$\alpha'_j = \frac{\alpha_j}{\sum_{k=1}^{m} \alpha_k}.$$

*Next, for all $j \in [m]$, suppose that $(\Gamma^{(j)}, \beta^{(j)})$ is a solution to the reformulated primal adversary bound for $f^{(j)}$, as stated in Theorem 6.2.4, with objective value $A^{(j)}$. We use the abbreviations*

$$P_\beta^{(j)}[x] = \prod_{\substack{j'=1 \\ j' \neq j}}^{m} (2\beta^{(j')}[x^{(j')}]) \qquad \text{and} \qquad P_\beta[x] = \prod_{j=1}^{m} (2\beta^{(j)}[x^{(j)}]).$$

*Then, we define $\Gamma \in \mathbb{R}^{\mathcal{D}\times\mathcal{D}}$, such that for the inputs $(x^{(1)}, \ldots, x^{(m)}) = x \in \mathcal{D}$ and $(y^{(1)}, \ldots, y^{(m)}) = y \in \mathcal{D}$,*

$$\Gamma[x,y] = \begin{cases} \sqrt{\alpha'_j} P_\beta^{(j)}[x] \cdot \Gamma^{(j)}[x^{(j)}, y^{(j)}], & \begin{array}{c} \text{if } [f(x) = 1 \wedge \exists j \in [m] : f^{(j)}(y^{(j)}) = 0 \\ \wedge \forall j' \in [m] \setminus \{j\}, x^{(j')} = y^{(j')}] \\ \vee [f(y) = 1 \wedge \exists j \in [m] : f^{(j)}(x^{(j)}) = 0 \\ \wedge \forall j' \in [m] \setminus \{j\}, x^{(j')} = y^{(j')}], \end{array} \\ 0, & \text{otherwise.} \end{cases}$$

*Furthermore, we let* $\beta \in \mathbb{R}^{\mathcal{D}}$ *be defined such that for all* $(x^{(1)}, \ldots, x^{(m)}) = x \in \mathcal{D}$,

$$\beta[x] = \begin{cases} \frac{1}{2} P_\beta[x], & \text{if } f(x) = 1, \\ \frac{\alpha'_j}{2} P_\beta[x], & \text{if } \exists j \in [m] : f^{(j)}(x^{(j)}) = 0 \\ & \quad \wedge \forall j' \in [m] \setminus \{j\}, f^{(j')}(x^{(j')}) = 1, \\ 0, & \text{otherwise.} \end{cases}$$

*Then,*

1. $(\Gamma, \beta)$ *is a feasible solution to the reformulated primal adversary bound for* $f$.
2. *The objective value is*

$$\sum_{j=1}^{m} \sqrt{\alpha'_j} A^{(j)},$$

*which is maximized by choosing* $\alpha_j = (A^{(j)})^2$, *for all* $j \in [m]$, *in which case the objective value becomes*

$$\sqrt{\sum_{j=1}^{m} (A^{(j)})^2}.$$

**Proof:**
First, observe that for any $(x^{(1)}, \ldots, x^{(m)}) = x \in \mathcal{D}$, $P_\beta^{(j)}[x]$ only depends on the values of $x^{(j')}$, where $j' \neq j$. Thus, in the first condition of the definition of $\Gamma$, we require that all these $x^{(j')}$'s are equal to all these $y^{(j')}$'s, and thus in this case $P_\beta^{(j)}[x] = P_\beta^{(j)}[y]$. Now, the definition preserves symmetry and thus $\Gamma$ is symmetric. We also observe that $\beta$ contains exclusively non-negative entries. Moreover, observe that the entry $\Gamma[x, y]$ can only be non-zero whenever $f(x) \neq f(y)$. We also obtain

$$\begin{aligned} \sum_{x \in f^{-1}(1)} \beta[x] &= \frac{1}{2} \sum_{x \in f^{-1}(1)} \prod_{j=1}^{m} (2\beta^{(j)}[x^{(j)}]) = \frac{1}{2} \prod_{j=1}^{m} \sum_{x^{(j)} \in (f^{(j)})^{-1}(1)} (2\beta^{(j)}[x^{(j)}]) \\ &= \frac{1}{2} \cdot \prod_{j=1}^{m} 1 = \frac{1}{2}, \end{aligned}$$

and similarly,

$$\begin{aligned}\sum_{x\in f^{-1}(0)}\beta[x] &= \frac{1}{2}\sum_{j=1}^{m}\alpha'_j \sum_{x^{(j)}\in(f^{(j)})^{-1}(0)} \sum_{\substack{\forall j'\in[m]\setminus\{j\},\\ x^{(j')}\in(f^{(j')})^{-1}(1)}} \prod_{j'=1}^{m}(2\beta^{(j')}[x^{(j')}]) \\ &= \frac{1}{2}\sum_{j=1}^{m}\alpha'_j \sum_{x^{(j)}\in(f^{(j)})^{-1}(0)} (2\beta^{(j)}[x^{(j)}]) \cdot \prod_{\substack{j'=1\\ j'\neq j}}^{m} \sum_{x^{(j')}\in(f^{(j)})^{-1}(1)} (2\beta^{(j')}[x^{(j')}]) \\ &= \frac{1}{2}\sum_{j=1}^{m}\alpha'_j \cdot 1 \cdot \prod_{\substack{j'=1\\ j'\neq j}}^{m} 1 = \frac{1}{2}.\end{aligned}$$

Next, we check the semidefinite constraint. To that end, let $k \in [n]$, and suppose that $k$ is an index to a bit in the $j$th part of the input, i.e., in $x = (x^{(1)}, \ldots, x^{(m)})$, $x_k$ is a bit of $x^{(j)}$. Next, suppose that $x, y \in \mathcal{D}$ are such that $x_k \neq y_k$ and $\Gamma[x, y] \neq 0$. Then, it must be that for all $j' \in [m]\setminus\{j\}$, $x^{(j')} = y^{(j')}$. As such, we find that we can decompose $\Gamma \circ \Delta_k$ into block-diagonal submatrices where for all $j' \in [m]\setminus\{j\}$, $x^{(j')} = y^{(j')}$. Hence, in order to check that $\operatorname{diag}(\beta) - \Gamma \circ \Delta_k \succeq 0$, it suffices to prove that each of the blocks is positive semidefinite.

Thus, fix inputs $x^{(j')} \in (f^{(j')})^{-1}(1)$, for all $j' \in [m] \setminus \{j\}$, and let $\Gamma'$, $\beta'$ and $\Delta'_k$ be the submatrices of $\Gamma$, $\beta$ and $\Delta_k$ corresponding to these inputs, i.e., for all $x^{(j)}, y^{(j)} \in \mathcal{D}^{(j)}$, $\Gamma'[x^{(j)}, y^{(j)}] = \Gamma[x, y]$, $\beta'[x^{(j)}] = \beta[x]$ and $\Delta'_k[x^{(j)}, y^{(j)}] = \Delta_k[x, y]$. We prove that $\operatorname{diag}(\beta') - \Gamma' \circ \Delta'_k \succeq 0$. To that end, observe that if $f^{(j)}(x^{(j)}) \neq f^{(j)}(y^{(j)})$, then

$$\Gamma'[x^{(j)}, y^{(j)}] = \sqrt{\alpha'_j} P^{(j)}_\beta[x] \Gamma^{(j)}[x^{(j)}, y^{(j)}],$$

and since from the definition of $P^{(j)}_\beta[x]$ we observe that it does not depend on $x^{(j)}$, we obtain that $\Gamma'$ is a scalar multiple of $\Gamma^{(j)}$. Similarly, if $x^{(j)} \in (f^{(j)})^{-1}(1)$, then

$$\beta'[x] = \frac{1}{2} P_\beta[x] = P^{(j)}_\beta[x] \beta^{(j)}[x^{(j)}],$$

and if $x^{(j)} \in (f^{(j)})^{-1}(0)$, then

$$\beta'[x] = \frac{\alpha'_j}{2} P_\beta[x] = \alpha'_j P^{(j)}_\beta[x] \beta^{(j)}[x^{(j)}].$$

Hence, after rearranging rows and columns in such a way that the positive inputs for $f^{(j)}$ are in the first block, and the negative inputs in the second, we can write

$$\Gamma^{(j)} \circ \Delta'_k = \begin{bmatrix} 0 & M^{(j)} \\ (M^{(j)})^\dagger & 0 \end{bmatrix}, \qquad \text{and} \qquad \beta^{(j)} = \begin{bmatrix} \beta^{(j)}_+ \\ \beta^{(j)}_- \end{bmatrix}$$

from which we obtain that

$$\begin{aligned}\operatorname{diag}(\beta') - \Gamma' \circ \Delta'_k &= P_\beta^{(j)}[x] \cdot \begin{bmatrix} \operatorname{diag}(\beta_+^{(j)}) & \sqrt{\alpha'_j} M^{(j)} \\ \sqrt{\alpha'_j}(M^{(j)})^\dagger & \alpha'_j \operatorname{diag}(\beta_-^{(j)}) \end{bmatrix} \\ &= P_\beta^{(j)}[x] \cdot (\operatorname{diag}(\beta^{(j)}) - \Gamma^{(j)} \circ \Delta'_k) \circ \begin{bmatrix} 1 \\ \sqrt{\alpha'_j} \end{bmatrix} \begin{bmatrix} 1 & \sqrt{\alpha'_j} \end{bmatrix} \succeq 0,\end{aligned}$$

where the semidefinite inequality holds since $P_\beta^{(j)}[x] \geq 0$, $(\Gamma^{(j)}, \beta^{(j)})$ forms a feasible solution to the reformulated primal adversary bound for the function $f^{(j)}$, and the right-most matrix is a projection and thus positive semidefinite. This completes the proof of claim 1.

For the second claim, we compute the objective value

$$\begin{aligned}\sum_{x,y \in \mathcal{D}} \Gamma[x,y] &= 2 \sum_{x \in f^{-1}(1)} \sum_{j=1}^{m} \sum_{y^{(j)} \in (f^{(j)})^{-1}(0)} \Gamma[x, (x^{(1)}, \ldots, y^{(j)}, \ldots, x^{(m)})] \\ &= \sum_{j=1}^{m} \sqrt{\alpha'_j} \sum_{\substack{\forall j' \in [m] \setminus \{j\}, \\ x^{(j')} \in (f^{(j')})^{-1}(1)}} P_\beta^{(j)}[x] \sum_{x^{(j)}, y^{(j)} \in \mathcal{D}} \Gamma^{(j)}[x^{(j)}, y^{(j)}] \\ &= \sum_{j=1}^{m} \sqrt{\alpha'_j} \prod_{\substack{j'=1 \\ j' \neq j}}^{m} \sum_{x^{(j')} \in (f^{(j')})^{-1}(1)} (2\beta^{(j')}[x^{(j')}]) \cdot A^{(j)} \\ &= \sum_{j=1}^{m} \sqrt{\alpha'_j} \prod_{\substack{j'=1 \\ j' \neq j}}^{m} 1 \cdot A^{(j)} = \sum_{j=1}^{m} \sqrt{\alpha'_j} A^{(j)}.\end{aligned}$$

Hence, using the Cauchy–Schwarz inequality, we obtain that the objective value satisfies

$$\sum_{j=1}^{m} \sqrt{\alpha'_j} A^{(j)} \leq \sqrt{\sum_{j=1}^{m} \alpha'_j} \cdot \sqrt{\sum_{j=1}^{m} (A^{(j)})^2} = \sqrt{\sum_{j=1}^{m} (A^{(j)})^2},$$

and the inequality becomes an equality when for all $j \in [m]$, we choose $\alpha_j = (A^{(j)})^2$. This completes the proof. □

Thus, if we plug the optimal solutions to the reformulated primal adversary bound for the functions $f^{(j)}$ in the above composition result, we obtain that

$$\operatorname{ADV}^{\pm}\left(\bigwedge_{j=1}^{m} f^{(j)}\right)^2 \geq \sum_{j=1}^{m} \operatorname{ADV}^{\pm}(f^{(j)})^2, \tag{7.1.7}$$

and so we indeed obtain Equation (7.1.6) with the reverse inequality. Hence, both results together prove equality, as we formally state below. This results is

to the best of our knowledge not mentioned explicitly anywhere in the existing literature, however it is considered folklore.

**7.1.14.** COROLLARY. *Let $n_1, \ldots, n_m$ be such that $\{0,1\}^n = \bigtimes_{j=1}^m \{0,1\}^{n_j}$, and for every $j \in [m]$, let $\mathcal{D}^{(j)} \subseteq \{0,1\}^{n_j}$, and $f^{(j)} : \mathcal{D}^{(j)} \to \{0,1\}$ be non-constant. Then,*

$$\mathrm{ADV}^{\pm}\left(\bigvee_{j=1}^m f_j\right)^2 = \mathrm{ADV}^{\pm}\left(\bigwedge_{j=1}^m f_j\right)^2 = \sum_{j=1}^m \mathrm{ADV}^{\pm}(f^{(j)})^2.$$

**Proof:**
The right equality follows directly from Equations (7.1.6) and (7.1.7). This result can be extended to the setting where we take the OR of $m$ functions, rather than AND, by cleverly using De Morgan's law and Equation (7.1.5), to obtain

$$\begin{aligned}\mathrm{ADV}^{\pm}\left(\bigvee_{j=1}^m f^{(j)}\right) &= \mathrm{ADV}^{\pm}\left(\neg \bigwedge_{j=1}^m \neg f^{(j)}\right) = \mathrm{ADV}^{\pm}\left(\bigwedge_{j=1}^m \neg f^{(j)}\right) \\ &= \sum_{j=1}^m \mathrm{ADV}^{\pm}(\neg f^{(j)})^2 = \sum_{j=1}^m \mathrm{ADV}^{\pm}(f^{(j)})^2.\end{aligned}$$

This completes the proof. □

Thus, in this subsection we have exactly characterized the adversary bound of computing the AND or OR of $n$ functions, in terms of the adversary bounds of the individual functions. Moreover, our results prove that in the setting where the individual functions have disjoint support, i.e., non-overlapping inputs, the span program composition constructions presented in Theorems 7.1.4 and 7.1.6 achieve this optimal bound, and as such cannot be improved.

There are many more interesting questions that one could try to tackle from here. The most immediate one is to achieve a similar result for the PARITY-construction that we presented in Corollary 7.1.10. In order to do that, one would have to prove a similar composition result of reformulated primal adversary bound solutions as showcased in Theorem 7.1.13, but now for the parity function rather than the AND-function. It would be interesting to see how this PARITY-construction manifests explicitly on the level of solutions to the primal and dual adversary bounds.

Another interesting question to ask is whether instead of composition results of span programs or dual adversary bound solutions, we can also obtain reverse constructions, i.e., *decomposition constructions*. More specifically, suppose that we have a solution $(\Gamma, \beta)$ to the reformulated dual adversary bound for some function $f = \bigwedge_{j=1}^m f^{(j)}$, where all the $f^{(j)}$'s have disjoint support. Can we somehow decompose this solution to generate individual solutions $(\Gamma^{(j)}, \beta^{(j)})$ to the reformulated dual adversary bound for the functions $f^{(j)}$, in such a way that when

we feed them back into the AND-composition result, i.e., Theorem 7.1.4, we get either back the same original solution, or at least an improved one? And if yes, how does such a decomposition result translate back to span programs? Initial attempts have revealed that these questions are surprisingly tricky, and it would be nice to figure this out somewhere in the future.

Finally, it would also be interesting to broaden the arsenal of composition results that we have available to us, and especially prove the optimality of these compositions. One example for instance is the following setting. Suppose we have $m$ functions $f^{(j)}$ on disjoint domains, and we want to compute the function $f$ that for a tuple of inputs $(x^{(1)}, \ldots, x^{(m)})$ checks whether at least $k$ of the functions $f^{(j)}$ evaluate to 1. We refer to such a composition result as a *threshold composition*. It would be nice to characterize the dual adversary bound of this function $f$ exactly, given the adversary bounds of the individual functions $f^{(j)}$. At this stage, however, we don't even have a candidate expression that we expect to capture this relation optimally. This is a very nice direction for future research, because it will teach us a lot about optimally composing many smaller functions into a bigger one.

### 7.1.3 Characteristic functions

The NOT-, AND-, and OR-composition results introduced in Section 7.1.1, allow for constructing span programs that evaluate logical compositions of functions computed by smaller individual span programs. In this subsection, we investigate how these composition results can be interpreted operationally. Specifically, we derive a way to characterize the probability distribution of the ideal phase variable of composed span programs.

We start by introducing a new property of span programs, called the characteristic function. It is a function defined on the Riemann sphere $C_\infty = \mathbb{C} \cup \{\infty\}$.

**7.1.15.** Definition (Characteristic function). Let $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ be a span program on $\mathcal{D}$, and let $x \in \mathcal{D}$. Next, we define the linear operator on $\mathcal{K}^\perp$ as

$$\mathfrak{p} = \Pi_{\mathcal{K}^\perp} \Pi_{\mathcal{H}(x)^\perp} \Pi_{\mathcal{K}^\perp} \in \mathcal{L}(\mathcal{K}^\perp).$$

Since $\mathfrak{p}$ projects onto $\mathcal{K}^\perp$ as the final operation, its image is indeed contained in $\mathcal{K}^\perp$, and since it is Hermitian we can think of $\mathfrak{p}$ as an operator acting on the Hilbert space $\mathcal{K}^\perp$, rather than on $\mathcal{H}$. Next, we define $\chi_x : \mathbb{C}_\infty \to \mathbb{C}_\infty$. For all $z \in \mathbb{C}$ for which the following expression is well-defined, we let

$$\chi_x(z) = \langle w_0 | \left(\mathfrak{p} - (1-z)I\right)^{-1} |w_0\rangle ,$$

where the inverse is taken in $\mathcal{L}(\mathcal{K}^\perp)$. We then identify $\chi_x$ with its analytic continuation, and say that $\chi_x$ is the *characteristic function of $\mathcal{P}$ for input $x$*. ◀

In order to develop some intuition for the characteristic function, we define a slight modification of the ideal phase variable, which we dub the *modified phase variable*. The ability to express our results in terms of this new quantity leads to less cumbersome expressions later on.

**7.1.16.** Definition (Modified phase variable). Let $\mathcal{P}$ be a span program on $\mathcal{D}$, and let $x \in \mathcal{D}$. Let $\Phi$ be the ideal phase variable for $x$ in $\mathcal{P}$. Then, we define the *modified phase variable for $x$ in $\mathcal{P}$* as $Q = \sin^2(\pi\Phi)$. Similarly as for $\Phi$, we define its support $\mathrm{supp}(Q)$ to be the set of values $q \in [0,1]$ such that $\mathbb{P}[Q = q] > 0$. ◀

Next, we prove several properties of these newly-defined objects. In particular, we discover that there is a surprisingly direct relation between the characteristic function and the probability distribution of the modified phase variable.

**7.1.17.** Lemma (Properties of the characteristic function).
*Let $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ be a span program on $\mathcal{D}$, and let $\chi_x$ be its characteristic function for input $x \in \mathcal{D}$. Let $Q$ be the modified phase variable for $x$ in $\mathcal{P}$. Then,*

1. *$\chi_x$ is a rational, analytic function, and the sum of residues in $\mathbb{C}$ is 1.*
2. *For all $z \in \mathbb{C}_\infty$,*
$$\chi_x(z) = \mathbb{E}\left[\frac{1}{z - Q}\right].$$
3. *For all $q \in \mathbb{C}_\infty$,*
$$\mathbb{P}\left[Q = q\right] = \lim_{z \to q}(z - q)\chi_x(z).$$
4. *The minimal positive and negative witness size for $x$ is*
$$w_+(x, \mathcal{P}) = -\chi_x(0), \qquad \text{and} \qquad w_-(x, \mathcal{P}) = \lim_{z \to 0} \frac{1}{z\chi_x(z)}.$$

**Proof:**
We start with proving claim 2. To that end, let $z \in \mathbb{C}$ be such that the expression for $\chi_x(z)$ in Definition 7.1.15 is well-defined. Furthermore, let $k \in \mathbb{N}$, the subspaces $R_0, \ldots, R_k \subseteq \mathcal{H}$ and $\theta_0, \ldots, \theta_k \in [0, \pi]$ be as in Jordan's lemma, i.e., Lemma 6.1.11. From said lemma, recall that we can write

$$\mathcal{K}^\perp = \bigoplus_{j=0}^{k} (\mathcal{K}^\perp \cap R_j), \qquad \text{and} \qquad \mathcal{H}(x)^\perp = \bigoplus_{j=0}^{k} (\mathcal{H}(x)^\perp \cap R_j),$$

and that for all $j \in [k]_0$, the angle between $\mathcal{K}^\perp \cap R_j$ and $\mathcal{H}(x)^\perp \cap R_j$ is $\theta_j/2$. From these observations, we deduce that we can write

$$\mathfrak{p} = \sum_{j=0}^{k} \cos^2(\theta_j/2)\Pi_{K^\perp \cap R_j} \in \mathcal{L}(\mathcal{K}^\perp),$$

and thus we can write

$$\mathfrak{p} - (1-z)I = \sum_{j=0}^{k} (\cos^2(\theta_j/2) - 1 + z)\Pi_{\mathcal{K}^\perp \cap R_j} = \sum_{j=0}^{k} (z - \sin^2(\theta_j/2))\Pi_{\mathcal{K}^\perp \cap R_j},$$

where both sides of the equality are elements in $\mathcal{L}(\mathcal{K}^\perp)$. Thus, $\mathfrak{p} - (1-z)I$ decomposes into projections with a scalar multiple on several mutually orthogonal subspaces that together span $\mathcal{K}^\perp$. As such, its inverse is the same sum of projections, but with the reciprocal of all the scalar multiples, which allows us to obtain

$$\begin{aligned}
\chi_x(z) &= \langle w_0 | \left( \Pi_{\mathcal{K}^\perp} \Pi_{\mathcal{H}(x)^\perp} \Pi_{\mathcal{K}^\perp} - (1-z)I \right)^{-1} |w_0\rangle \\
&= \sum_{j=0}^{k} \frac{1}{z - \sin^2(\theta_j/2)} \langle w_0 | \Pi_{\mathcal{K}^\perp \cap R_j} |w_0\rangle = \sum_{j=0}^{k} \frac{\left\| \Pi_{R_j} |w_0\rangle \right\|^2}{z - \sin^2(\theta_j/2)} \\
&= \sum_{\phi \in \mathrm{supp}(\Phi)} \frac{\mathbb{P}\left[\Phi = \phi\right]}{z - \sin^2(\pi\phi)} = \sum_{q \in \mathrm{supp}(Q)} \frac{\mathbb{P}[Q = q]}{z - q} = \mathbb{E}\left[\frac{1}{z - Q}\right].
\end{aligned}$$

We can now analytically extend both sides of the equality to all the values of $z$ for which the expression for $\chi_x(z)$ is not well-defined, which completes the proof of claim 2.

Next, we turn to claim 1. To that end, since the spectrum of $\Phi$ is discrete, so is the spectrum of $Q$, and we can take $p_1, \ldots, p_n, q_1, \ldots, q_n \in [0,1]$ such that for all $j \in [n]$, $\mathbb{P}[Q = q_j] = p_j$, and for all other values $q \notin \{q_j : j \in [n]\}$, $\mathbb{P}[Q = q] = 0$. Then, we can write

$$\chi_x(z) = \sum_{j=1}^{n} \frac{p_j}{z - q_j}.$$

It follows that $\chi_x$ is indeed a rational function. Moreover, its residues in $\mathbb{C}$ are at $q_j$, for all $j \in [n]$, and we have $\mathrm{Res}(\chi_x; q_j) = p_j$. Therefore, the sum of the residues in $\mathbb{C}$ is the sum of the $p_j$'s, which is 1 since they form a probability distribution. This completes the proof of claim 1.

For claim 3, observe that for all $z \in \mathbb{C} \setminus \{q_j : j \in [n]\}$, we have that $\chi_x$ does not have a pole, and therefore the limit from claim 3 becomes 0. On the other hand, for all $k \in [n]$,

$$\lim_{z \to q_k} (z - q_k)\chi_x(z) = \lim_{z \to q_k} \sum_{j=1}^{n} (z - q_k)\frac{p_j}{z - q_j} = p_k = \mathbb{P}[Q = q_k].$$

Thus, we have proven claim 3.

Finally, observe from Theorem 6.1.13 that

$$-\chi_x(0) = -\mathbb{E}\left[-\frac{1}{Q}\right] = \mathbb{E}\left[\frac{1}{\sin^2(\pi\Phi)}\right] = w_+(x, \mathcal{P}),$$

and

$$\lim_{z\to 0} \frac{1}{z\chi_x(z)} = \frac{1}{\mathbb{P}[Q=0]} = \frac{1}{\mathbb{P}[\Phi=0]} = w_-(x,\mathcal{P}).$$

This completes the proof. □

From Definition 7.1.15, we have seen that the characteristic function is defined directly in terms of the geometrical objects that make up a span program. On the other hand, we have observed in Lemma 7.1.17 that it bears an interesting connection to the operational analysis of span programs through the modified phase variable. Moreover, it is a rational and analytic function, which opens up the possibility for us to study these objects using tools from both polynomial algebra and complex analysis.

In order to further develop the intuition for characteristic functions and their relation to the modified phase variable, we draw both in Figure 7.1.2. We observe that the function has simple poles of first order exactly at the values that the modified phase variable attains with non-zero probability, and the residues at these poles are exactly equal to these probabilities.

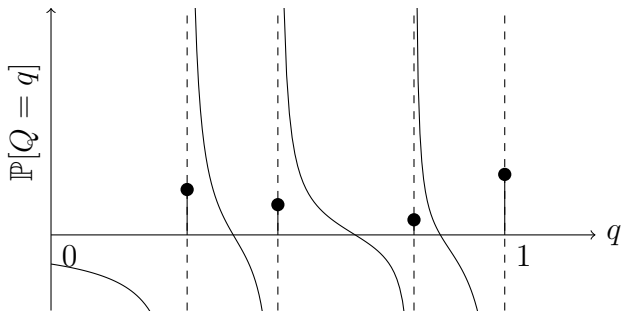



Figure 7.1.2: The relation between the modified phase variable $Q$ and the characteristic function. The latter's asymptotes are exactly at the values that $Q$ attains with non-zero probability.

Recall from Lemma 6.1.3 that $x$ is a negative input for $\mathcal{P}$ if and only if the ideal phase variable is 0 with non-zero probability. Thus, we can easily visually observe whether the characteristic function corresponds to a negative input – it does if and only if it has an asymptote at the left end of the spectrum, i.e., at 0. On the other hand, if $x$ is a positive input, then the $y$-value at which the characteristic function passes through the vertical axis is minus the positive witness size.

One of the most striking properties of the characteristic function is that it relates the properties of the modified phase variable of any span program and its negation. The following theorem characterizes this relation.

**7.1.18.** Theorem. *Let $\mathcal{P}$ be a span program on $\mathcal{D}$, and let $x \in \mathcal{D}$. Let $\chi_x$ and $\chi'_x$ be the characteristic functions for $x$ in $\mathcal{P}$ and $\neg\mathcal{P}$, respectively. Then, for all*

$z \in \mathbb{C}_\infty$,

$$\chi'_x(z) = \frac{1}{z(z-1)\chi_x(z)}.$$

**Proof:**
By analytic continuation, it suffices to prove that the above relation holds for all $z \in \mathbb{C}$ such that $\chi_x(z)$ and $\chi'_x(z)$ are well-defined. Thus, let $z \in \mathbb{C}$ satisfy this assumption, and let $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ and $\neg\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}'(x), \mathcal{K}', |w_0\rangle)$. Then, we define the vectors $|v\rangle \in \mathcal{K}^\perp$ and $|v'\rangle \in (\mathcal{K}')^\perp = \mathcal{K} \oplus \mathrm{Span}\{|w_0\rangle\}$ as

$$|v\rangle = (\mathfrak{p} - (1-z)I)^{-1} |w_0\rangle, \quad \text{and} \quad |v'\rangle = (\mathfrak{p}' - (1-z)I)^{-1} |w_0\rangle,$$

where $\mathfrak{p}' = \Pi_{\mathcal{K}\oplus\mathrm{Span}\{|w_0\rangle\}}\Pi_{\mathcal{H}(x)}\Pi_{\mathcal{K}\oplus\mathrm{Span}\{|w_0\rangle\}} \in \mathcal{L}((\mathcal{K}')^\perp)$. Rewriting yields

$$|w_0\rangle = (\mathfrak{p} - (1-z)I)\,|v\rangle, \tag{7.1.8a}$$
$$|w_0\rangle = (\mathfrak{p}' - (1-z)I)\,|v'\rangle. \tag{7.1.8b}$$

Next, we decompose the vectors $|v\rangle$ and $|v'\rangle$ according to the subspaces generated by Jordan's lemma, i.e., Lemma 6.1.11, with the spaces $A = \mathcal{K}$ and $B = \mathcal{H}(x)$. For all $j \in [k]_0$, we write

$$|w_0^{(j)}\rangle = \Pi_{\mathcal{K}^\perp \cap R_j} |w_0\rangle, \quad |v^{(j)}\rangle = \Pi_{\mathcal{K}^\perp \cap R_j} |v\rangle, \quad \text{and} \quad |(v')^{(j)}\rangle = \Pi_{\mathcal{K} \cap R_j} |v'\rangle,$$

and subsequently we choose $\alpha \in \mathbb{C}$, such that

$$|w_0\rangle = \sum_{j=0}^{k} |w_0^{(j)}\rangle, \quad |v\rangle = \sum_{j=0}^{k} |v^{(j)}\rangle, \quad \text{and} \quad |v'\rangle = \sum_{j=0}^{k} |(v')^{(j)}\rangle + \alpha\,|w_0\rangle. \tag{7.1.9}$$

We can now substitute these decompositions into Equation (7.1.8a), to obtain

$$\begin{aligned}\sum_{j=0}^{k} |w_0^{(j)}\rangle &= \sum_{j=0}^{k} (\Pi_{\mathcal{K}^\perp}\Pi_{\mathcal{H}(x)^\perp}\Pi_{\mathcal{K}^\perp} - (1-z)I)|v^{(j)}\rangle \\ &= \sum_{j=0}^{k} \left[\cos^2\left(\frac{\theta_j}{2}\right) - (1-z)\right] |v^{(j)}\rangle = \sum_{j=0}^{k} \left[z - \sin^2\left(\frac{\theta_j}{2}\right)\right] |v^{(j)}\rangle\end{aligned}$$

from which we find that for all $j \in [k]_0$, by projecting onto $R_j$ we are left with

$$|w_0^{(j)}\rangle = \left[z - \sin^2\left(\frac{\theta_j}{2}\right)\right] |v^{(j)}\rangle.$$

Thus,

$$\chi_x(z) = \langle w_0|v\rangle = \sum_{j=0}^{k} \langle w_0^{(j)}|v^{(j)}\rangle = \sum_{j=0}^{k} \left[z - \sin^2\left(\frac{\theta_j}{2}\right)\right]^{-1} \||w_0^{(j)}\rangle\|^2. \tag{7.1.10}$$

Similarly, we can plug the decompositions from Equation (7.1.9) into Equation (7.1.8b). To that end, we observe that $\mathrm{Span}\{|w_0\rangle\}$ and $\mathcal{K}$ are orthogonal subspaces, and so we can write $\Pi_{\mathcal{K}\oplus\mathrm{Span}\{|w_0\rangle\}} = \Pi_{\mathcal{K}} + |w_0\rangle\langle w_0|$. Thus, we obtain

$$\begin{aligned}|w_0\rangle = \big(&\Pi_{\mathcal{K}}\Pi_{\mathcal{H}(x)}\Pi_{\mathcal{K}} + \Pi_{\mathcal{K}}\Pi_{\mathcal{H}(x)}\,|w_0\rangle\langle w_0| \\ &+ |w_0\rangle\langle w_0|\,\Pi_{\mathcal{H}(x)}\Pi_{\mathcal{K}} + |w_0\rangle\langle w_0|\,\Pi_{\mathcal{H}(x)}\,|w_0\rangle\langle w_0| - (1-z)I\big)\,|v'\rangle\,.\end{aligned}$$

We evaluate the 5 terms individually, to obtain

$$\Pi_{\mathcal{K}}\Pi_{\mathcal{H}(x)}\Pi_{\mathcal{K}}\,|v'\rangle = \sum_{j=0}^{k}\Pi_{\mathcal{K}}\Pi_{\mathcal{H}(x)}|(v')^{(j)}\rangle = \sum_{j=0}^{k}\cos^2\left(\frac{\theta_j}{2}\right)|(v')^{(j)}\rangle, \tag{7.1.11a}$$

$$\Pi_{\mathcal{K}}\Pi_{\mathcal{H}(x)}\,|w_0\rangle\langle w_0|v'\rangle = \alpha\sum_{j=0}^{k}\Pi_{\mathcal{K}}\Pi_{\mathcal{H}(x)}|w_0^{(j)}\rangle, \tag{7.1.11b}$$

$$|w_0\rangle\langle w_0|\,\Pi_{\mathcal{H}(x)}\Pi_{\mathcal{K}}\,|v'\rangle = |w_0\rangle\sum_{j=0}^{k}\langle w_0^{(j)}|\Pi_{\mathcal{H}(x)}|(v')^{(j)}\rangle, \tag{7.1.11c}$$

$$\begin{aligned}|w_0\rangle\langle w_0|\,\Pi_{\mathcal{H}(x)}\,|w_0\rangle\langle w_0|v'\rangle &= \alpha\,|w_0\rangle\sum_{j=0}^{k}\langle w_0^{(j)}|\Pi_{\mathcal{H}(x)}|w_0^{(j)}\rangle \\ &= \alpha\,|w_0\rangle\sum_{j=0}^{k}\sin^2\left(\frac{\theta_j}{2}\right)\||w_0^{(j)}\rangle\|^2,\end{aligned} \tag{7.1.11d}$$

$$-(1-z)\,|v'\rangle = \sum_{j=0}^{k}-(1-z)|(v')^{(j)}\rangle - \alpha(1-z)\,|w_0\rangle\,. \tag{7.1.11e}$$

Summing everything on the right-hand side and projecting onto the space $\mathcal{K}\cap R_j$, for $j\in[k]_0$, yields

$$0 = \left[\cos^2\left(\frac{\theta_j}{2}\right) - (1-z)\right]|(v')^{(j)}\rangle + \alpha\Pi_{\mathcal{K}}\Pi_{\mathcal{H}(x)}|w_0^{(j)}\rangle,$$

which implies that

$$\alpha\Pi_{\mathcal{K}}\Pi_{\mathcal{H}(x)}|w_0^{(j)}\rangle = -\left[z - \sin^2\left(\frac{\theta_j}{2}\right)\right]|(v')^{(j)}\rangle. \tag{7.1.12}$$

Similarly, summing everything on the right-hand side of Equation (7.1.11) and projecting onto $\mathrm{Span}\{|w_0\rangle\}$ yields

$$1 = \sum_{j=0}^{k}\langle w_0^{(j)}|\Pi_{\mathcal{H}(x)}|(v')^{(j)}\rangle + \alpha\sum_{j=0}^{k}\left[\sin^2\left(\frac{\theta_j}{2}\right) - (1-z)\right]\||w_0^{(j)}\rangle\|^2. \tag{7.1.13}$$

For every $j \in [k]_0$, we rewrite the first summand using Equation (7.1.12), to obtain

$$\begin{aligned}\langle w_0^{(j)}|\Pi_{\mathcal{H}(x)}|(v')^{(j)}\rangle &= -\alpha\left[z-\sin^2\left(\frac{\theta_j}{2}\right)\right]^{-1}\langle w_0^{(j)}\Pi_{\mathcal{H}(x)}\Pi_{\mathcal{K}}\Pi_{\mathcal{H}(x)}|w_0^{(j)}\rangle \\ &= -\alpha\left[z-\sin^2\left(\frac{\theta_j}{2}\right)\right]^{-1}\|\Pi_{\mathcal{K}}\Pi_{\mathcal{H}(x)}|w_0^{(j)}\rangle\|^2 \\ &= -\alpha\left[z-\sin^2\left(\frac{\theta_j}{2}\right)\right]^{-1}\cos^2\left(\frac{\theta_j}{2}\right)\sin^2\left(\frac{\theta_j}{2}\right),\end{aligned}$$

where the latter equality can be observed from the visualization of Jordan's lemma in Figure 6.1.4. We can use this relation to rewrite Equation (7.1.13), and obtain

$$\begin{aligned}1 &= -\alpha\sum_{j=0}^{k}\left[z-\sin^2\left(\frac{\theta_j}{2}\right)\right]^{-1}\sin^2\left(\frac{\theta_j}{2}\right)\cos^2\left(\frac{\theta_j}{2}\right)\||w_0^{(j)}\rangle\|^2 \\ &\quad -\alpha\sum_{j=0}^{k}\left[(1-z)-\sin^2\left(\frac{\theta_j}{2}\right)\right]\||w_0^{(j)}\rangle\|^2 \\ &= -\alpha\sum_{j=0}^{k}\left[\left[z-\sin^2\left(\frac{\theta_j}{2}\right)\right]^{-1}\cos^2\left(\frac{\theta_j}{2}\right)-1\right]\sin^2\left(\frac{\theta_j}{2}\right)\||w_0^{(j)}\rangle\|^2 \\ &\quad -\alpha\sum_{j=0}^{k}(1-z)\||w_0^{(j)}\rangle\|^2 \\ &= -\alpha\sum_{j=0}^{k}\left[\left[z-\sin^2\left(\frac{\theta_j}{2}\right)\right]^{-1}(1-z)\sin^2\left(\frac{\theta_j}{2}\right)+(1-z)\right]\||w_0^{(j)}\rangle\|^2 \\ &= -\alpha(1-z)\sum_{j=0}^{k}\left[z-\sin^2\left(\frac{\theta_j}{2}\right)\right]^{-1}\left[\sin^2\left(\frac{\theta_j}{2}\right)+z-\sin^2\left(\frac{\theta_j}{2}\right)\right]\||w_0^{(j)}\rangle\|^2 \\ &= -\alpha z(1-z)\sum_{j=0}^{k}\left[z-\sin^2\left(\frac{\theta_j}{2}\right)\right]^{-1}\||w_0^{(j)}\rangle\|^2 = -\alpha z(1-z)\chi_x(z),\end{aligned}$$

where in the last step we used Equation (7.1.10). Finally, we find

$$\chi_x'(z) = \langle w_0|v'\rangle = \alpha = \frac{1}{z(z-1)\chi_x(z)},$$

completing the proof. □

The above proof is quite non-trivial. Even though we mentioned at the beginning of this chapter that we would lay emphasis on the intuitive interpretation of the results that we present here, coming up with an intuitive understanding of

the above proof has proven futile. It would certainly be an interesting direction of future research to obtain a better geometrical understanding of the above result, and especially if it can be understood in the context of the visualization of Jordan's lemma, i.e., Figure 6.1.4.

One potentially related piece of literature is [BSS14, Section 3.1][4] – especially Figure 3.1 in said paper is very reminiscent of Figure 7.1.2. In the cited paper, the spectrum of the operator $A + |v\rangle\langle v|$ is analyzed, for a Hermitian operator $A$ and a vector $|v\rangle$ from the Hilbert space acted on by $A$. It is not directly clear how their setting relates to ours, though, i.e., it is not clear how one should choose $A$ and $|v\rangle$ to recover the setting we are interested in here. Moreover, the mentioned paper only investigates the spectrum, whereas here the newly-computed characteristic function also captures the overlap with the eigenspaces of the new operator. We leave developing connections between this text and the paper mentioned for future work.

One implication of the above proof is particularly elegant. Where the probability distribution of the original modified phase variable is concentrated on the points where the characteristic function has its asymptotes, the probability distribution of the *negated* modified phase variable, i.e., the modified phase variable of the negated span program, is concentrated on the locations of its zeros. This is illustrated in Figure 7.1.3.

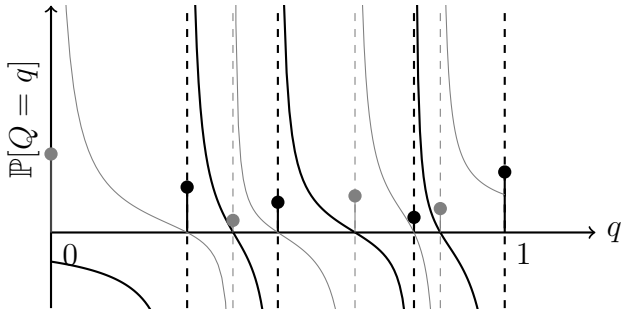



Figure 7.1.3: Relation between the characteristic functions and the modified phase variables of the original and negated span programs, shown in black and gray, respectively. For both endpoints, we see that either the original, or the negated modified phase variable has non-zero probability, and in the interval $(0, 1)$, we see that the zeros of the characteristic functions coincide with the locations of the non-zero probabilities of the negated modified phase variable.

The visualization in Figure 7.1.3 suggests a remarkable property of modified phase variables of span programs and their negations. Suppose that $Q$ is the modified phase variable of a span program $\mathcal{P}$, and let $Q'$ be the modified phase variable of $\neg\mathcal{P}$. We write $q_1, \ldots, q_n$ for the values that $Q$ attains with non-zero probability, and similarly $q'_1, \ldots, q'_{n'}$ for $Q'$. Now, the characteristic function $\chi_x$

[4]This connection was pointed out by Prof. M.C. Veraar in personal communication.

in $\mathcal{P}$ has asymptotes at all $q_j$'s, and it has zeros at all $q'_j$'s. Since the function $\chi_x$ is continuous between two asymptotes, it must pass through the horizontal axis, and thus in between two consecutive values $q_j$ and $q_{j+1}$ there must be a value $q'_{j'}$, and vice versa. In other words, the supports of $Q$ and $Q'$ are interlaced, i.e., if one draws them in the interval $[0, 1]$, one obtains an alternating pattern. We would like to explicitly point out that this property is very easily observed from the visualization of the characteristic function, but is in no way obvious from our geometrical understanding of span programs, e.g., Figure 6.1.2 or Figure 6.1.4.

We also remark that if one knows the probability distribution of the modified phase variable $Q$ of a span program $\mathcal{P}$, i.e., one knows $q_1, \ldots, q_n \in [0, 1]$ and $p_1, \ldots, p_n \in [0, 1]$ such that $\mathbb{P}[Q = q_j] = p_j$ for all $j \in [n]$, then it is very easy to explicitly calculate the probability distribution of the modified phase variable $Q'$ of the negation $\neg\mathcal{P}$. Indeed, one can write the characteristic function

$$\chi_x(z) = \sum_{j=1}^{n} \frac{p_j}{z - q_j},$$

and then find the new probability distribution $q'_1, \ldots, q'_{n'} \in [0, 1]$ and $p'_1, \ldots, p'_{n'} \in [0, 1]$ by performing a partial fraction decomposition on the negated characteristic function $\chi'_x$ and write it in the following form

$$\chi'_x(z) = \frac{1}{z(z-1)\chi_x(z)} = \sum_{j=1}^{n'} \frac{p'_j}{z - q'_j}.$$

One might wonder whether the probabilities in the resulting probability distribution indeed sum to 1 again, i.e., whether in the above equation

$$\sum_{j=1}^{n'} p'_j = 1.$$

From the theory we have developed thus far, we can deduce that this should be the case – after all, $\chi'_x$ is again a characteristic function, and hence it possesses the properties described in Lemma 7.1.17. We can also prove this directly from the relation derived in Theorem 7.1.18, and the proof is too elegant not to include it in this text.

**7.1.19.** THEOREM. *Let $\chi_x$ and $\chi'_x$ be two analytic functions of the form*

$$\chi_x(z) = \sum_{j=1}^{n} \frac{p_j}{z - q_j}, \qquad \textit{and} \qquad \chi'_x(z) = \frac{1}{z(z-1)\chi_x(z)} = \sum_{j=1}^{n'} \frac{p'_j}{z - q'_j},$$

*where $p_j, q_j, p'_{j'}, q'_{j'} \in \mathbb{C}$, for all $j \in [n]$ and $j' \in [n']$. Then,*

$$\sum_{j=1}^{n'} p'_j = \left[\sum_{j=1}^{n} p_j\right]^{-1}.$$

*In particular, if either summation is* 1*, then so is the other.*

**Proof:**
Since $\chi'_x$ is an analytic function, we know that the sum of all its residues on the Riemann sphere must be 0. Using that for every analytic function $f$, we have the formula $-\operatorname{Res}(f;\infty) = \operatorname{Res}(z \mapsto f(1/z)/z^2; 0)$, we obtain that

$$\begin{aligned}\sum_{j=1}^{n'} p'_j &= \sum_{j=1}^{n'} \operatorname{Res}(\chi'_x; q'_j) = -\operatorname{Res}(\chi'_x;\infty) = -\operatorname{Res}\left(z \mapsto \frac{1}{z(z-1)\chi_x(z)};\infty\right)\\ &= \operatorname{Res}\left(z \mapsto \frac{1}{z^2}\cdot\frac{z}{\left(\frac{1}{z}-1\right)\chi_x\left(\frac{1}{z}\right)};0\right) = \lim_{z\to 0}\frac{1}{\left(\frac{1}{z}-1\right)\chi_x\left(\frac{1}{z}\right)}\\ &= \lim_{z\to 0}\left[\sum_{j=1}^{n} p_j \frac{\frac{1}{z}-1}{\frac{1}{z}-q_j}\right]^{-1} = \left[\lim_{z\to 0}\sum_{j=1}^{n} p_j\frac{1-z}{1-zq_j}\right]^{-1} = \left[\sum_{j=1}^{n} p_j\right]^{-1}.\end{aligned}$$

This completes the proof. □

We can use the same techniques to recover relations between the moments of the modified phase variable and its negated counterpart.

**7.1.20.** Theorem. *Let $\mathcal{P}$ be a span program on $\mathcal{D}$, and $x \in \mathcal{D}$. Let $\chi_x$ and $\chi'_x$ be the characteristic functions for $x$ in $\mathcal{P}$ and $\neg\mathcal{P}$, respectively. Let $Q$ and $Q'$ be the modified phase variables for $x$ in $\mathcal{P}$ and $\neg\mathcal{P}$, respectively. Then, we have the power series*

$$\sum_{k=0}^{\infty}\mathbb{E}[(Q')^k]z^k = \left[\left(\frac{1}{z}-1\right)\chi_x\left(\frac{1}{z}\right)\right]^{-1} = \sum_{\ell=0}^{\infty}\left[\sum_{k=1}^{\infty}\mathbb{E}[Q^{k-1}(1-Q)]z^k\right]^{\ell}.$$

*In particular, for all $m \geq 0$, we have*

$$\mathbb{E}[(Q')^m] = \sum_{\substack{\mathbf{k}\in\mathbb{N}^*\\ \Sigma\mathbf{k}=m}}\prod_{k\in\mathbf{k}}\mathbb{E}\left[Q^{k-1}(1-Q)\right],$$

*where $\mathbb{N}^*$ denotes all finite positive integer sequences of arbitrary length, including the empty sequence $\mathbf{k} = \varnothing$, and for every $\mathbf{k} \in \mathbb{N}^*$, we denote the sum of all entries by $\Sigma\mathbf{k}$. For the first values of $m$, this yields*

$$\begin{aligned}\mathbb{E}[(Q')^0] &= 1,\\ \mathbb{E}[Q'] &= \mathbb{E}[1-Q],\\ \mathbb{E}[(Q')^2] &= \mathbb{E}[1-Q]^2 + \mathbb{E}[Q(1-Q)],\\ \mathbb{E}[(Q')^3] &= \mathbb{E}[1-Q]^3 + 2\mathbb{E}[Q(1-Q)]\mathbb{E}[1-Q] + \mathbb{E}[Q^2(1-Q)].\end{aligned}$$

**Proof:**
Just like in Theorem 7.1.19, we write

$$\chi_x(z) = \sum_{j=1}^{n} \frac{p_j}{z - q_j}, \qquad \text{and} \qquad \chi'_x(z) = \frac{1}{z(z-1)\chi_x(z)} = \sum_{j=1}^{n'} \frac{p'_j}{z - q'_j},$$

where $p_j, q_j, p'_{j'}, q'_{j'} \in \mathbb{C}$, for all $j \in [n]$ and $j' \in [n']$. Next, let $k \in [n]$. Then,

$$\begin{aligned}
\mathbb{E}[(Q')^k] &= \sum_{j=1}^{n'} p'_j (q'_j)^k = \sum_{j=1}^{n'} \operatorname{Res}\left(z \mapsto z^k \chi'_x(z); q'_j\right) \\
&= -\operatorname{Res}\left(z \mapsto \frac{z^k}{z(z-1)\chi_x(z)}; \infty\right) \\
&= \operatorname{Res}\left(z \mapsto \frac{1}{z^2 \cdot z^k \cdot \frac{1}{z} \cdot \left(\frac{1}{z} - 1\right) \chi_x\left(\frac{1}{z}\right)}; 0\right) \\
&= \operatorname{Res}\left(z \mapsto \frac{1}{z^{k+1}} \underbrace{\left[\left(\frac{1}{z} - 1\right) \chi_x\left(\frac{1}{z}\right)\right]^{-1}}_{f(z)}; 0\right).
\end{aligned}$$

We already saw in the previous lemma that $f$ is analytic near 0, as $f(z) \to 1$ when $z \to 0$. Thus, we can expand $f$ into its Taylor series around $z = 0$, and then the above relation tells us that the coefficient in this Taylor series in front of the term $z^k$ is exactly $\mathbb{E}[(Q')^k]$. Thus, we have

$$\sum_{k=0}^{\infty} \mathbb{E}[(Q')^k] z^k = \left[\left(\frac{1}{z} - 1\right) \chi_x\left(\frac{1}{z}\right)\right]^{-1}.$$

This proves the first equality.

For the second equality, we observe that

$$\chi_x\left(\frac{1}{z}\right) = \mathbb{E}\left[\frac{1}{\frac{1}{z} - Q}\right] = z\mathbb{E}\left[\frac{1}{1 - zQ}\right] = \sum_{k=0}^{\infty} \mathbb{E}[Q^k] z^{k+1},$$

and hence

$$\left(\frac{1}{z} - 1\right) \chi_x\left(\frac{1}{z}\right) = \sum_{k=0}^{\infty} \mathbb{E}[Q^k] z^k - \sum_{k=1}^{\infty} \mathbb{E}[Q^{k-1}] z^k = 1 - \sum_{k=1}^{\infty} \mathbb{E}[Q^{k-1}(1-Q)] z^k.$$

Thus,

$$\left[\left(\frac{1}{z} - 1\right) \chi_x\left(\frac{1}{z}\right)\right]^{-1} = \sum_{\ell=0}^{\infty} \left[\sum_{k=1}^{\infty} \mathbb{E}[Q^{k-1}(1-Q)] z^k\right]^{\ell},$$

proving the second equality.

Next, we rewrite the right-hand side. To that end, we use the convention that $\mathbb{N}^0 = \{\varnothing\}$, and the empty product equals 1. Then,

$$\begin{aligned}\sum_{k=0}^{\infty} \mathbb{E}[(Q')^k] z^k &= \sum_{\ell=0}^{\infty} \sum_{\mathbf{k}\in\mathbb{N}^\ell} \prod_{k\in\mathbf{k}} z^k \mathbb{E}[Q^{k-1}(1-Q)] \\ &= \sum_{m=0}^{\infty} z^m \sum_{\ell=0}^{\infty} \sum_{\substack{\mathbf{k}\in\mathbb{N}^\ell \\ \Sigma\mathbf{k}=m}} \prod_{k\in\mathbf{k}} \mathbb{E}[Q^{k-1}(1-Q)],\end{aligned}$$

from which we find that for all $m \geq 0$,

$$\mathbb{E}[(Q')^m] = \sum_{\substack{\mathbf{k}\in\mathbb{N}^* \\ \Sigma\mathbf{k}=m}} \prod_{k\in\mathbf{k}} \mathbb{E}[Q^{k-1}(1-Q)].$$

This completes the proof. □

We note here that the above results can also be directly obtained from the geometrical picture presented in Figure 6.1.4. For instance, using relations of the form of Equation (6.1.3), observe that

$$\begin{aligned}\mathbb{E}[(Q')^2] = \mathbb{E}[\sin^4(\pi\Phi')] &= \left\|\Pi_{\mathcal{K}\oplus\mathrm{Span}\{|w_0\rangle\}}\Pi_{\mathcal{H}(x)}\,|w_0\rangle\right\|^2 \\ &= \left|\langle w_0|\,\Pi_{\mathcal{H}(x)}\,|w_0\rangle\right|^2 + \left\|\Pi_{\mathcal{K}}\Pi_{\mathcal{H}(x)}\,|w_0\rangle\right\|^2 \\ &= \left\|\Pi_{\mathcal{H}(x)}\,|w_0\rangle\right\|^4 + \left\|\Pi_{\mathcal{K}}\Pi_{\mathcal{H}(x)}\,|w_0\rangle\right\|^2 \\ &= \mathbb{E}[\cos^2(\pi\Phi)]^2 + \mathbb{E}[\cos^2(\pi\Phi)\sin^2(\pi\Phi)] = \mathbb{E}[1-Q]^2 + \mathbb{E}[Q(1-Q)].\end{aligned}$$

Even though we found direct expressions for the moments of $Q'$, i.e., $\mathbb{E}[(Q')^m]$, these formulae become very cumbersome for higher values of $m$. Thus, in principle, one could from here define the moment-generating function $t \mapsto \mathbb{E}[\exp(tQ')]$ and from there infer the probability distribution of $Q'$. However, the characteristic function provides this probability distribution much more directly.

Now that we have seen how the characteristic function behaves under negation of the span program, we can also investigate how it behaves under the AND- and OR-composition constructions from Theorems 7.1.4 and 7.1.6, respectively. This is the objective of the following theorem.

**7.1.21.** Theorem. *Let $\mathcal{P}_1, \ldots, \mathcal{P}_n$ be span programs on a common domain $\mathcal{D}$. Let $\alpha_1, \ldots, \alpha_n > 0$, and for all $j \in [n]$, let*

$$\alpha'_j = \frac{\alpha_j}{\sum_{k=1}^n \alpha_k}.$$

*Let $x \in \mathcal{D}$, and let $\chi_x^{(j)}$ be the characteristic function for $x$ in $\mathcal{P}_j$.*

1. *Let $\chi_x$ be the characteristic function for $x$ in $\bigwedge_{j=1}^n \alpha_j \mathcal{P}_j$. Then,*

$$\chi_x(z) = \sum_{j=1}^n \alpha_j' \chi_x^{(j)}(z).$$

2. *Let $\chi_x'$ be the characteristic function for $x$ in $\bigvee_{j=1}^n \alpha_j \mathcal{P}_j$. Then,*

$$\chi_x'(z) = \left[\sum_{j=1}^n \frac{\alpha_j'}{\chi_x^{(j)}(z)}\right]^{-1}.$$

**Proof:**
For claim 1, we let $\mathcal{P} = \bigwedge_{j=1}^n \alpha_j \mathcal{P}_j$, and we write out the definition of the characteristic function $\chi_x$ for $x$ in $\mathcal{P}$. To that end, let $z \in \mathbb{C}$ be such that we have the well-defined relation

$$\chi_x(z) = \langle w_0 | \left(\mathfrak{p} - (1-z)I\right)^{-1} |w_0\rangle .$$

For all $j \in [n]$, we write $\mathcal{P}_j = (\mathcal{H}^{(j)}, x \mapsto \mathcal{H}^{(j)}(x), \mathcal{K}^{(j)}, |w_0^{(j)}\rangle)$. Then, we recall from Definition 7.1.3 that $\mathcal{K}$ and $\mathcal{H}(x)$ are simply the vector sum of the $\mathcal{K}^{(j)}$'s and $\mathcal{H}^{(j)}(x)$'s, respectively. Thus, we can decompose the operator

$$\mathfrak{p} = \sum_{j=1}^n \mathfrak{p}^{(j)},$$

where each $\mathfrak{p}^{(j)} \in \mathcal{L}(\mathcal{K}^\perp \cap \mathcal{H}^{(j)})$. As these spaces are mutually orthogonal with $j \in [k]_0$, $\mathfrak{p}$ is a block-diagonal operator, and as such the inverse can be taken term by term as well. Therefore, we obtain that

$$\chi_x(z) = \sum_{j=1}^n \alpha_j' \langle w_0^{(j)}| \left(\mathfrak{p}^{(j)} - (1-z)I\right)^{-1} |w_0^{(j)}\rangle = \sum_{j=1}^n \alpha_j' \chi_x^{(j)}(z).$$

This completes the proof of claim 1.

For claim 2, we recall from Definition 7.1.5 that the definition of $\bigvee_{j=1}^n \alpha_j \mathcal{P}_j$ is given via the NOT- and AND-construction, and as such we obtain that for all $z \in \mathbb{C}_\infty$,

$$\frac{1}{z(z-1)\chi_x'(z)} = \sum_{j=1}^n \frac{\alpha_j'}{z(z-1)\chi_x^{(j)}(z)}.$$

The factor $z(z-1)$ cancels on both sides, and after taking the reciprocal, we are left with the expression from the theorem statement, completing the proof. □

We can now use these characteristic function relations to pictorially understand what happens during the span program renormalization procedure. To

that end, recall from the proof of Theorem 7.1.8 that we can view the renormalization procedure as generating span program $\mathcal{P}' = \beta^2\mathcal{P} \vee \mathcal{T}$, where we choose $\beta = 1/\sqrt{2W_-(\mathcal{P})}$, and $\mathcal{T} = (\mathrm{Span}\{|*\rangle\}, x \mapsto \{0\}, \{0\}, |*\rangle)$. For any input $x$, the characteristic function for $\mathcal{T}$ is

$$\chi_x^{(\mathcal{T})}(z) = \langle *| \left(I - (1-z)I\right)^{-1} |*\rangle = \frac{1}{z},$$

and thus, using Theorem 7.1.18, we find that for any $x \in \mathcal{D}$, the characteristic function for $x$ in $\neg\mathcal{T}$ is

$$\chi_x^{(\neg\mathcal{T})}(z) = \frac{1}{z(z-1)\chi_x^{(\mathcal{T})}(z)} = \frac{1}{z(z-1)\cdot\frac{1}{z}} = \frac{1}{z-1},$$

and we also see that the modified phase variable $Q^{(\neg\mathcal{T})}$ is 1 with probability 1.

Now, observe that $\mathcal{P}' = \neg(\beta^2(\neg\mathcal{P}) \wedge \neg\mathcal{T})$. Thus, using Theorems 7.1.18 and 7.1.21, we obtain that we go from $\chi_x^{(\mathcal{P})}$ to $\chi_x^{(\mathcal{P}')}$ in four steps:

1. Start with the characteristic function $\chi_x^{(\mathcal{P})}$
2. Take the negation to obtain

$$\chi_x^{(\neg\mathcal{P})}(z) = \frac{1}{z(z-1)\chi_x^{(\mathcal{P})}(z)}.$$

3. AND-compose with $\neg\mathcal{T}$ to obtain

$$\chi_x^{(\beta^2(\neg\mathcal{P})\wedge\neg\mathcal{T})}(z) = \frac{\beta^2}{1+\beta^2}\chi_x^{(\neg\mathcal{P})} + \frac{1}{(1+\beta^2)(z-1)}.$$

4. Take the negation to obtain

$$\chi_x^{(\mathcal{P}')}(z) = \frac{1}{z(z-1)\chi_x^{(\beta^2(\neg\mathcal{P})\wedge\neg\mathcal{T})}(z)}.$$

We illustrate this procedure in Figure 7.1.4, and use it to visually interpret the renormalization construction. The core idea is to push almost all the weight of the probability distribution of $Q$ to the left-hand side of the spectrum, while preserving whether the input is positive or negative for $\mathcal{P}$. Thus, we cannot add some weight directly at $q = 0$ (by performing an AND-construction with a span program whose modified phase variable would have a single peak at $q = 0$), because this would also turn every positive input into a negative one. However, what we can do is first negate the span program, then put some weight on the right-hand side, i.e., at $q = 1$ (by performing an AND-construction with $\neg\mathcal{T}$) and then negate again. We can see that this has the effect of making any peak at $q = 0$ larger, if it exists, as desired. It also pushes the peaks that are in the interval $(0, 1)$ closer to 0 in the process.

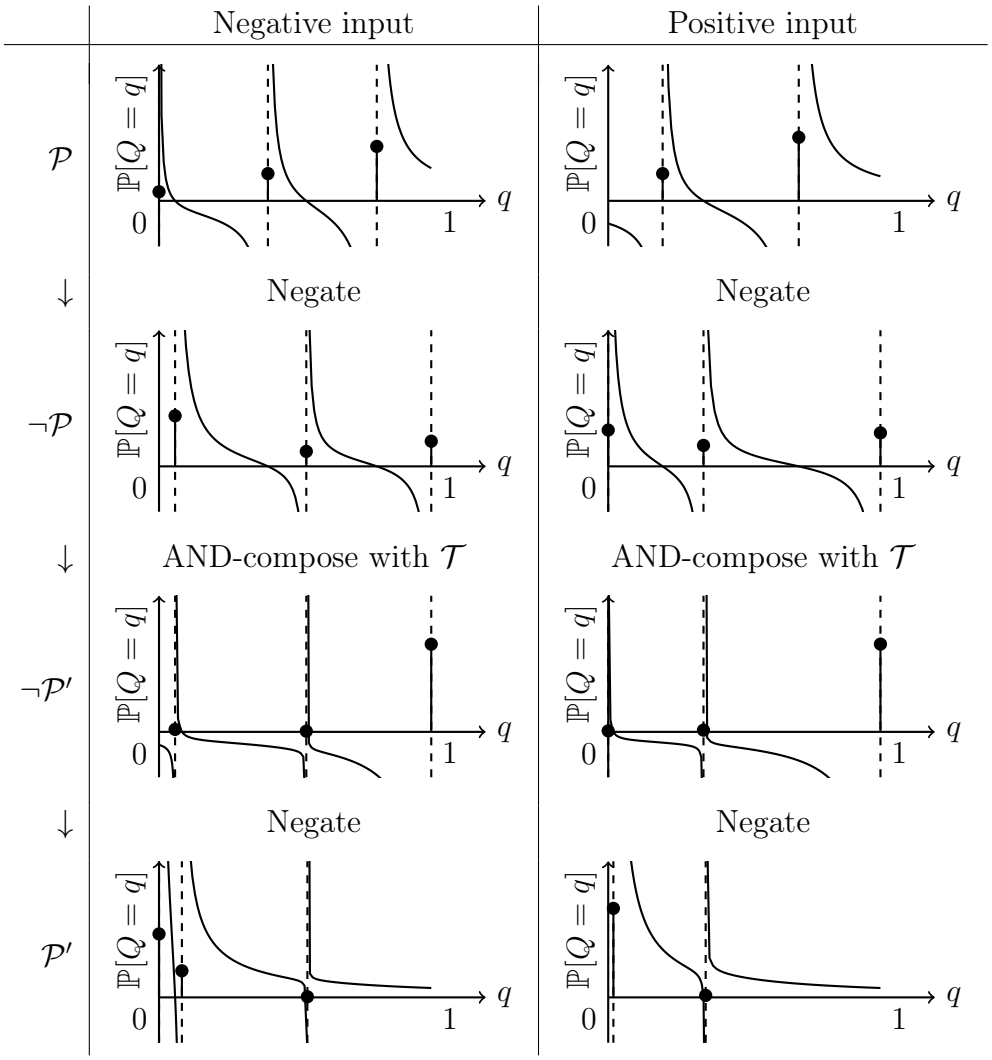



Figure 7.1.4: Illustration of the renormalization procedure. The core idea is to push almost all the "weight" of the probability distribution left-hand side of the spectrum. This is achieved by negating, adding some weight on the right, and negating again.

As a final note, we mention how the quantum approximate counting algorithm ties in with the notions that we have introduced in this section. To that end, we consider the domain $\mathcal{D} = \{0,1\}^n$, and we would like to devise a quantum query algorithm that outputs the hamming weight of any input $x \in \mathcal{D}$. Let $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ be a span program on $\mathcal{D}$, defined as

$$\mathcal{H} = \mathbb{C}^n, \quad \mathcal{H}(x) = \mathrm{Span}\{|j\rangle : x_j = 0\}, \quad \mathcal{K} = \{0\}, \quad |w_0\rangle = \frac{1}{\sqrt{n}} \sum_{j=1}^{n} |j\rangle .$$

Then, we obtain that the ideal and modified phase variables $\Phi$ and $Q$ for an input $x$ in $\mathcal{P}$ capture the hamming weight in the heights of the peaks at 0 or 1, i.e.,

$$\mathbb{P}[Q=0] = \mathbb{P}[\Phi = 0] = \left\| \Pi_{\mathcal{H}(x)^\perp} |w_0\rangle \right\|^2 = \frac{|x|}{n}, \qquad \text{and}$$
$$\mathbb{P}[Q=1] = \mathbb{P}\left[\Phi = \frac{1}{2}\right] = \left\| \Pi_{\mathcal{H}(x)} |w_0\rangle \right\|^2 = 1 - \frac{|x|}{n}.$$

Consequently, the characteristic function for an input $x \in \mathcal{D}$ is

$$\chi_x(z) = \langle w_0 | \left( \Pi_{\mathcal{H}(x)^\perp} - (1-z)I \right)^{-1} |w_0\rangle = \frac{|x|}{nz} + \frac{n-|x|}{n(z-1)}.$$

If we now perform span program negation, the resulting characteristic function $\chi'_x$ in $\neg\mathcal{P}$ becomes

$$\chi'_x(z) = \frac{1}{z(z-1)\chi_x(z)} = \frac{n}{|x|(z-1) + (n-|x|)z} = \frac{1}{z - \frac{|x|}{n}}.$$

Hence, the resulting modified phase variable $Q'$ for input $x$ in $\neg\mathcal{P}$ has exactly one peak, located at $|x|/n$, from which we find that the ideal phase variable $\Phi'$ for input $x$ in $\neg\mathcal{P}$ has exactly one peak at

$$\phi_{|x|} = \arcsin\left( \frac{1}{\pi} \sqrt{\frac{|x|}{n}} \right).$$

Thus, we conclude that we can run phase estimation on the span program unitary $U(x, \neg\mathcal{P})$ to obtain an estimate of $\phi_{|x|}$ with additive error, which we can then use to solve for $|x|$. Even though the analysis is completely different, this exactly recovers the quantum approximate counting algorithm, as first introduced in [BHM+02].

The example of quantum approximate counting hints at a very exciting new direction of potential research – even though we constructed a span program $\mathcal{P}$, which in principle only decides between positive and negative inputs, we used the characteristic function and its relations to the modified and ideal phase variable to arrive at an algorithm that *performs an estimation*, and as such steps out

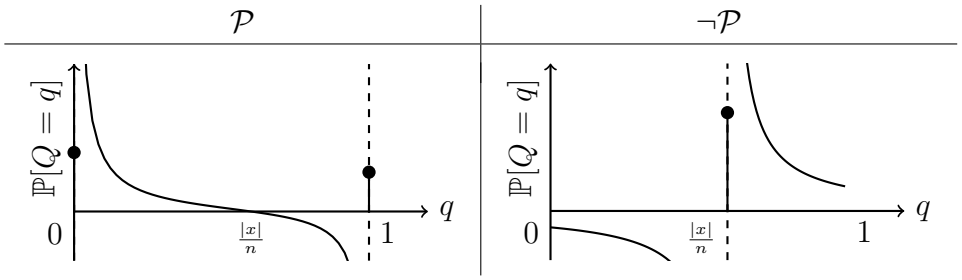



Figure 7.1.5: The quantum approximate counting algorithm. We show the characteristic function $\chi_x$ for an input $x$, in $\mathcal{P}$, on the left, and in $\neg\mathcal{P}$, on the right. We can see that the root of the function on the left-hand side is the location of the peak on the right-hand side, which is at $q = |x|/n$.

of the setting of merely solving decision problems. For now, quantum approximate counting remains the only application for these techniques, but it is very interesting to figure out if the characterization of the ideal phase variable's probability distribution can be used to give estimation algorithms in more complicated settings.

A very interesting follow-up question that comes to mind is whether these ideas can be used to perform mean estimation of random variables, as was the topic of Chapter 3. Somewhat surprisingly, the answer to this question might very well be yes. In a recent work, Kothari and O'Donnell [KO22] gave an optimal quantum mean estimation algorithm that estimates the mean of a univariate random variable, removing the logarithmic overhead of the approach outlined earlier on in this thesis. The analysis of their algorithm is very reminiscent of the techniques presented in this work. It would be very interesting to investigate possible connections further in the near future.

As a final note, we remark that the definition of characteristic functions, Definition 7.1.15, is not very intuitive, that is, it is difficult to capture the objects that appear in the definition concisely in a geometrical picture. Perhaps this is also why the proof of the negation relation, i.e., Theorem 7.1.18, is not very intuitive either. It would be nice to figure out if there is a more elegant way to define the characteristic function. One possibility that comes to mind is whether it is possible to give a variational characterization just like we did in Lemma 6.1.8, i.e., given any $z \in \mathbb{C}$, can we find a specific subset of vectors $|v\rangle \in \mathcal{H}(x)$ such that the shortest vector in this subset has size $\chi_x(z)$? It would be interesting to figure out whether this characterization sheds some new light on the results proved in this subsection. We leave this for future work.

## 7.2 Graph composition of span programs

The attentive reader might have already noticed that there is something peculiar about the formulae for the positive and negative witness sizes in the AND- and OR-constructions. Indeed, these formulas are very reminiscent of those that appear when one computes the effective resistance in an electrical network, when the resistors are parallel or in series. In this section, we argue that this is not a coincidence, and we develop a more thorough connection between span programs and electrical networks.

This connection was first introduced in [BR12], and the analysis was successively improved in [JK17], and later in [JJK+18]. The result presented here is a bit more general compared to the existing literature, since prior works only use a graph to generate a span program from scratch, whereas here we use a graph to compose many smaller span programs into a bigger one.

We start by introducing some background on electrical networks and how they can be interpreted in the Hilbert space setting, in Section 7.2.1. Then, we show how this leads to a construction that composes span programs using these networks, referred to as the graph composition, in Section 7.2.2. After that, we consider a special case of this construction, where the graph is planar, in Section 7.2.3.

### 7.2.1 Electrical networks

Let $G = (V, E)$ be an undirected graph, possibly with double edges and self-loops. To every edge $e \in E$, we associate a "default direction," i.e., when we say that $e$ connects node $u$ to $v$, a default direction from $u$ to $v$ is implied. We also associate a strictly positive weight $r_e > 0$ to every edge, as well as a separate dimension in a Hilbert space $\mathcal{H}_G$, i.e., we have $\mathcal{H}_G = \mathrm{Span}\{|e\rangle : e \in E\}$.

We now build an intuition for the vectors in $\mathcal{H}_G$. To that end, think of the graph $G$ as an electrical network, where every edge $e$ is a wire with a resistance $r_e$. Through a wire $e \in E$, we can send an "edge flow" $f_e \in \mathbb{C}$. If the edge flow $f_e$ on a given edge $e \in E$ is positive, we can think of electrical current flowing through $e$ in its default direction, and similarly if $f_e < 0$, we can think of electrical current flowing through the wire in the opposite direction.

An assignment of such edge flows to all the edges in the graph $(f_e)_{e\in E} \subseteq \mathbb{C}$ is referred to as a "flow", and to this flow we associate the vector $|f\rangle \in \mathcal{H}_G$, defined as

$$|f\rangle = \sum_{e\in E} f_e \sqrt{r_e}\, |e\rangle\,.$$

Since there is a one-to-one relation between the object $(f_e)_{e\in E} \subseteq \mathbb{C}$ and $|f\rangle \in \mathcal{H}_G$, we will refer to both of them as the *flow*, and use them interchangeably. We immediately find that the size of $|f\rangle$, i.e., the squared norm of the vector $|f\rangle$, can

be expressed as

$$\| |f\rangle \|^2 = \sum_{e \in E} |f_e|^2 r_e.$$

Hence, if all the edge flows are real we can think of the squared norm of the vector $|f\rangle$ as the energy dissipated in the electrical network in a unit of time, and as such we refer to $\| |f\rangle \|^2$ as the energy of $|f\rangle$.

The analogy with electrical currents breaks when we take imaginary flows, i.e., when the vector $|f\rangle \in \mathcal{H}_G$ has imaginary entries, since it's not clear how to make sense of imaginary currents in an electrical network. However, for the purpose of developing intuition for the results to come, it usually suffices to think about flows with real entries only.

In electrical networks, we have the "law of conservation of current," also known as "Kirchhoff's current law", which tells us that at every junction, the sum of the incoming flow should be the same as the sum of the outgoing flow. Flows that satisfy this law are called "circulations", and we can think of the space of circulations $\mathcal{C}_G$ as a subspace of the flow space $\mathcal{H}_G$.

The following definition formalizes flows and circulations.

**7.2.1.** Definition (Circulation, flow and effective resistance). Let $G = (V, E)$ be an undirected graph, with strictly positive weights $(r_e)_{e \in E}$. We refer to $\mathcal{H}_G = \mathrm{Span}\{|e\rangle : e \in E\}$ as the *flow space* of $G$. For every vertex $v \in V$, let $N^+(v), N^-(v) \subseteq E$ denote the set of edges with outgoing and incoming default directions, respectively. Then,

1. Let $|f\rangle = \sum_{e \in E} f_e \sqrt{r_e}\, |e\rangle \in \mathcal{H}_G$. If for all vertices $v \in V$ we have *conservation of flow*, i.e.,

$$\forall v \in V, \qquad \sum_{e \in N^+(v)} f_e - \sum_{e \in N^-(v)} f_e = 0, \tag{7.2.1}$$

   then we say that $|f\rangle$ is a *circulation*. The space of all circulations is called the *circulation space* of $G$, and denoted by $\mathcal{C}_G \subseteq \mathcal{H}_G$.

2. Let $|f\rangle = \sum_{e \in E} f_e \sqrt{r_e}\, |e\rangle \in \mathcal{H}_G$. Furthermore, let $s, t \in V$, such that $s \neq t$, and suppose that

$$\forall v \in V, \qquad \sum_{e \in N^+(v)} f_e - \sum_{e \in N^-(v)} f_e = \begin{cases} 1, & \text{if } v = s, \\ -1, & \text{if } v = t, \\ 0, & \text{otherwise.} \end{cases} \tag{7.2.2}$$

   Then we say that $|f\rangle$ is a *unit st-flow*. The set of all unit $st$-flows is denoted by $\mathcal{F}_{st} \subseteq \mathcal{H}_G$.

3. For every $|f\rangle = \sum_{e \in E} f_e \sqrt{r_e}\,|e\rangle$, we refer to the squared norm $\||f\rangle\|^2$ as the *energy of* $|f\rangle$. As such, we refer to the vector $|f\rangle \in F_{st}$ that minimizes $\||f\rangle\|^2$ as the *minimum-energy unit st-flow*. We denote this minimum energy by $R_{s,t}$, and call it the *effective resistance between s and t*, i.e.,

$$R_{s,t}(G, (r_e)_{e \in E}) = \min\{\||f\rangle\|^2 : |f\rangle \in F_{st}\}.$$

For notational convenience, we write that the effective resistance is $\infty$ whenever $s$ and $t$ are not connected in $G$, and 0 whenever $s = t$. ◀

We now prove several properties of these newly-defined objects.

**7.2.2.** Lemma (Properties of flows). *Let $G = (V, E)$ be an undirected graph, with strictly positive weights $(r_e)_{e \in E}$. Then,*

1. *The circulation space $\mathcal{C}_G \subseteq \mathcal{H}_G$ is a linear subspace.*
2. *Let $s, t \in V$, with $s \neq t$. The set $\mathcal{F}_{st} \subseteq \mathcal{H}_G$ is an affine subspace, and can be written as $|f\rangle + \mathcal{C}_G$, where $|f\rangle \in \mathcal{C}_G^\perp$ is the unique minimum-energy unit st-flow.*

**Proof:**
The first claim follows immediately from the observation that the constraints imposed on the circulations in Equation (7.2.1) are all linear and homogeneous. For the second one, if $|f\rangle$ and $|f'\rangle$ are both unit $st$-flows, then from the constraints in Equation (7.2.2) it is directly clear that $|f\rangle - |f'\rangle$ is a circulation. Thus $\mathcal{F}_{st}$ is indeed an affine subspace that can be written as $|f\rangle + \mathcal{C}_G$, with $|f\rangle$ any unit $st$-flow. In particular, we can choose $|f\rangle$ to be the minimum-energy unit $st$-flow, which, since the energy of a flow is the norm squared, must be the unique element in $\mathcal{F}_{st}$ that minimizes the norm. Thus, it must be in the orthogonal complement of $\mathcal{C}_G$, completing the proof. □

The vectors in $\mathcal{C}_G^\perp$, i.e., the orthogonal complement of the circulation space, have the interesting property that they admit the definition of a *potential function*. We formally define this concept below.

**7.2.3.** Definition. Let $G = (V, E)$ be an undirected graph, with strictly positive weights $(r_e)_{e \in E}$. We refer to any function $U : V \to \mathbb{C}$ as a *potential function*. The *flow derived from U with weights* $(r_e)_{e \in E}$ is the vector $\sum_{e \in E} f_e \sqrt{r_e}\,|e\rangle$, where if $e \in E$ connects $v$ to $w$, then

$$f_e = \frac{U_v - U_w}{r_e}.$$ ◀

One can wonder what flows can be defined through these potential functions. It turns out that these are exactly the flows $|f\rangle \in \mathcal{C}_G^\perp$. We prove this in the following lemma.

**7.2.4.** Lemma (Characterization of potential functions). *Let $G = (V, E)$ be an undirected graph, with strictly positive weights $(r_e)_{e\in E}$. Let $|f\rangle \in \mathcal{H}_G$ be a flow. Then $|f\rangle \in \mathcal{C}_G^\perp$ if and only if there exists a potential function $U$ such that $|f\rangle$ is the flow derived from $U$ with weights $(r_e)_{e\in E}$. In that case, any unit st-flow $|f'\rangle$ satisfies $\langle f'|f\rangle = U_s - U_t$.*

**Proof:**
First, suppose that $U$ is a potential function, and that $|f\rangle$ is the flow derived from $U$. Let $\sum_{e\in E} f'_e\sqrt{r_e}\,|e\rangle = |f'\rangle \in \mathcal{C}_G$ be a circulation. Then,

$$\langle f|f'\rangle = \sum_{e\in E} \overline{f_e} r_e f'_e = \sum_{\substack{e\in E\\ e:v\to w}} (\overline{U_v} - \overline{U_w}) f'_e = \sum_{v\in V} \overline{U_v} \left[ \sum_{e\in N^+(v)} f'_e - \sum_{e\in N^-(v)} f'_e \right] = 0,$$

where $e : v \to w$ denotes that $e$ connects $v$ to $w$. Thus, we find that $|f\rangle \in \mathcal{C}_G^\perp$.

Conversely, suppose that $|f\rangle \in \mathcal{C}_G^\perp$. In every connected component of $G$, we take one vertex $v \in V$ arbitrarily, and set $U(v) = 0$. Next, for any vertex $w \in V$, we find a path that reaches $w$ from one of the nodes $v$ for which we set $U(v) = 0$. We denote the edges along this path by $e_1, \ldots, e_n$, with directions $m_1, \ldots, m_n \in \{-1, 1\}$, that is, for all $j \in [n]$, if the path from $v$ to $w$ along the edges $e_1, \ldots, e_n$ traverses the edge $e_j$ in its default direction, we set $m_j = 1$, and otherwise we set $m_j = -1$. Then, let

$$U(w) = \sum_{j=1}^{n} m_j f_{e_j} r_{e_j}.$$

It remains to check that $U$ is well-defined, i.e., independent of the path we took from $v$ to $w$. To that end, suppose that there are two paths $e_1, \ldots, e_n$ and $e'_1, \ldots, e'_{n'}$ with directions $m_1, \ldots, m_n$ and $m'_1, \ldots, m'_{n'}$, that connect $v$ to $w$. Let $U(w)$ and $U'(w)$ be defined as above based on the two different paths, respectively. We must prove that $U(w) = U'(w)$.

To that end, observe that $e_1, \ldots, e_n, e'_{n'}, \ldots, e'_1$ with the associated directions $m_1, \ldots, m_n, -m'_{n'}, \ldots, -m'_1$ is a cycle in $G$. Consequently, we can send unit flow along this cycle and obtain a circulation $|f'\rangle$, defined as

$$|f'\rangle = \sum_{j=1}^{n} m_j \sqrt{r_{e_j}}\,|e_j\rangle - \sum_{j=1}^{n'} m'_j \sqrt{r_{e'_j}}\,|e'_j\rangle\,.$$

Since $|f'\rangle$ is a circulation, it is orthogonal to $|f\rangle$, and thus we find

$$U(w) - U'(w) = \sum_{j=1}^{n} m_j f_{e_j} r_{e_j} - \sum_{j'=1}^{n'} m_{j'} f_{e_{j'}} r_{e_{j'}} = \langle f'|f\rangle = 0.$$

Hence, indeed, $U$ is well-defined.

Finally, let $|f\rangle = \sum_{e\in E} f_e\sqrt{r_e}\,|e\rangle \in \mathcal{C}_G^\perp$ be derived from $U : V \to \mathbb{C}$, and let $|f'\rangle = \sum_{e\in E} f'_e\sqrt{r_e}\,|e\rangle$ be a unit $st$-flow. Then,

$$\begin{aligned}\langle f'|f\rangle &= \sum_{e\in E} \overline{f_e}' r_e f_e = \sum_{\substack{e\in E\\ e:v\to w}} \overline{f_e}'[U_v - U_w] \\ &= \sum_{v\in V} U_v \left[\sum_{e\in N^+(v)} \overline{f_e}' - \sum_{e\in N^-(v)} \overline{f_e}'\right] = U_s - U_t.\end{aligned}$$

This completes the proof. □

Thus, we have proved that the existence of a potential function is a property of a flow being in $\mathcal{C}_G^\perp$. As such, we find from Lemma 7.2.2 that any minimum-energy flow can be associated to a potential function.

In the electrical networks picture, Kirchhoff's second law asserts the existence of such a potential function. Thus, using the above lemma, we can reinterpret Kirchhoff's second law as stating that any flow in an electrical network must necessarily be in the subspace $\mathcal{C}_G^\perp$.

### 7.2.2 Definition and basic properties

Now that we have introduced the necessary background on electrical networks, we can show how this theory is related to the study of quantum algorithms. The primary connection is through the construction of a particular span program, known as $st$-connectivity, i.e., a span program that given an undirected graph $G = (V, E)$ decides whether $s$ and $t$ are connected, where the input $x$ to the span program tells us which of the edges $e \in E$ are present and which are not. This span program was first introduced in [BR12] and also appeared in [Bel14]. Its analysis of the positive and negative witnesses was subsequently improved in [JK17] in the special case where the graph is planar, and finally a complete characterization of the positive and negative witness sizes was given in [JJK+18] in the general case.

In this subsection, we extend the result by not only allowing the construction of a single $st$-connectivity span program, but we show how one can associate individual span programs to each of the edges in the graph, and as such arrive at a novel technique to compose span programs. We refer to the resulting span program as the *graph composition of span programs*, and we give a full characterization of its positive and negative witnesses in Theorem 7.2.7.

The high-level idea is to define an undirected graph $G = (V, E)$, with distinct source and sink nodes $s, t \in V$. We also associate strictly positive weights $r_e > 0$ to every edge in the graph, in such a way that the effective resistance between $s$ and $t$ is 1. Next, to every $e \in E$ we associate a span program $\mathcal{P}^{(e)}$ on some

common domain $\mathcal{D}$. Every input $x \in \mathcal{D}$ defines a subgraph $G^+(x)$ of $G$ in a natural way, i.e., every edge $e \in E$ is present in $G^+(x)$ if and only if $x$ is a positive input for $\mathcal{P}^{(e)}$. We now construct a composed span program $\mathcal{P}$ that for a given input $x \in \mathcal{D}$ computes whether $s$ and $t$ are connected in the subgraph $G^+(x)$. We give an example illustration of such a composition in Figure 7.2.1.

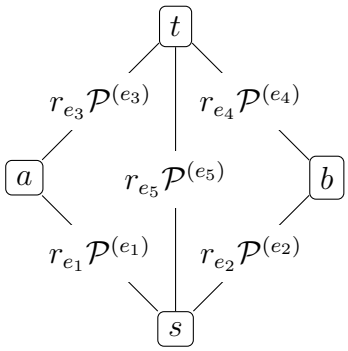



Figure 7.2.1: An example of a graph composition of five span programs, $\mathcal{P}^{(e_1)}, \ldots, \mathcal{P}^{(e_5)}$, with weights $r_{e_1}, \ldots, r_{e_5}$. If span program $\mathcal{P}^{(e_j)}$ computes function $f_j$, then the above graph composition evaluates the function $(f_1 \wedge f_3) \vee (f_2 \wedge f_4) \vee f_5$.

The core idea in the construction of this composed span program $\mathcal{P}$ is to embed the flow space $\mathcal{H}_G$ into the direct sum of the Hilbert spaces of the span programs associated to each of the edges. More concretely, for every $e \in E$, we write $\mathcal{P}^{(e)} = (\mathcal{H}^{(e)}, x \mapsto \mathcal{H}^{(e)}(x), \mathcal{K}^{(e)}, |w_0^{(e)}\rangle)$, and define the Hilbert space $\mathcal{H}$ and embedding $\mathcal{E} : \mathcal{H}_G \to \mathcal{H}$ as

$$\mathcal{H} = \bigoplus_{e \in E} \mathcal{H}^{(e)}, \qquad \text{and} \qquad \mathcal{E} : |e\rangle \mapsto |w_0^{(e)}\rangle. \tag{7.2.3}$$

As such, the embedding $\mathcal{E}$ defines an isometry between the flow space of the graph $\mathcal{H}_G$, and the subspace of $\mathcal{H}$ spanned by all the initial states of the individual span programs that make up the composition. Next, on high level, we relate the circulation space $\mathcal{C}_G$ to the input-independent subspace $\mathcal{K}$, and the minimum-energy $st$-flow in $G$ to the initial state $|w_0\rangle$, which has the elegant consequence that all positive witnesses in $\mathcal{P}$ can be interpreted as $st$-flows in $G$, and all negative witnesses in $\mathcal{P}$ can be interpreted as potential functions.

The following definitions and theorem make these statements more precise.

**7.2.5.** Definition (Graph composition of span programs). Let $G = (V, E)$ be an undirected graph, with strictly positive weights $(r_e)_{e \in E}$. Let $s, t \in V$ be such that $s \neq t$ and such that $s$ and $t$ are connected in $G$. We define

$$r'_e = \frac{r_e}{R_{s,t}(G, (r_e)_{e \in E})},$$

where the denominator is the effective resistance between $s$ and $t$ in $G$ weighted with $(r_e)_{e\in E}$, as defined in Definition 7.2.1.

Next, let $|f\rangle \in \mathcal{H}_G$ be the minimum-energy unit $st$-flow in $G$ with weights $(r'_e)_{e\in E}$. We associate a span program $\mathcal{P}^{(e)} = (\mathcal{H}^{(e)}, x \mapsto \mathcal{H}^{(e)}(x), \mathcal{K}^{(e)}, |w_0^{(e)}\rangle)$ on a fixed domain $\mathcal{D}$ to every edge $e \in E$. We recall the embedding $\mathcal{E} : \mathcal{H}_G \to \mathcal{H}$ from Equation (7.2.3), we let

$$\mathcal{H} = \bigoplus_{e\in E} \mathcal{H}^{(e)}, \qquad \mathcal{H}(x) = \bigoplus_{e\in E} \mathcal{H}^{(e)}(x),$$
$$\mathcal{K} = \bigoplus_{e\in E} \mathcal{K}^{(e)} \oplus \mathcal{E}(\mathcal{C}_G), \qquad |w_0\rangle = \mathcal{E}(|f\rangle),$$

and we let $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$. Then, we refer to $\mathcal{P}$ as the *graph composition* of $G$ with weights $(r_e)_{e\in E}$ and span programs $(\mathcal{P}^{(e)})_{e\in E}$. ◀

In order to check the well-definedness of the span program $\mathcal{P}$, we must check that $|w_0\rangle$ has unit norm, and is an element of $\mathcal{K}^\perp$. Since $\mathcal{E}$ is an isometry, we obtain that

$$\||w_0\rangle\|^2 = \|\mathcal{E}(|f\rangle)\|^2 = \||f\rangle\|^2 = R_{s,t}(G, (r'_e)_{e\in E}) = \frac{R_{s,t}(G, (r_e)_{e\in E})}{R_{s,t}(G, (r_e)_{e\in E})} = 1,$$

where we used that the size of $|f\rangle$ is the effective resistance by Definition 7.2.1, and that multiplying all resistances with some scalar, also multiplies the effective resistance by that same scalar. Thus, it remains to check that $|w_0\rangle \in \mathcal{K}^\perp$. To that end, we observe that $|w_0\rangle$ is a linear combination of the individual $|w_0^{(e)}\rangle$'s, and as such is orthogonal to all $\mathcal{K}^{(e)}$'s. Moreover, since $|f\rangle$ is a minimum-energy unit $st$-flow, we find by Lemma 7.2.2 that it is in $\mathcal{C}_G^\perp$, and thus $|w_0\rangle = \mathcal{E}(|f\rangle) \in \mathcal{E}(\mathcal{C}_G^\perp) \subseteq \mathcal{E}(\mathcal{C}_G)^\perp$. Thus, $|w_0\rangle$ is indeed in $\mathcal{K}^\perp$, proving the well-definedness of the span program $\mathcal{P}$ in Definition 7.2.5.

Now, we analyze the properties of the graph composition construction presented in Definition 7.2.5 more closely, to obtain the results similar in flavor to those presented in Theorems 7.1.4 and 7.1.6 for the AND- and OR-compositions, respectively. To that end, we first define the *availability graph* $G^+(x)$ and the *unavailability graph* $G^-(x)$ that derive from $G$ given a particular input $x \in \mathcal{D}$.

**7.2.6.** Definition (Availability and unavailability graphs). Let $G = (V, E)$ be an undirected graph. Let $\mathcal{P}$ be a graph composition of $G$, with strictly positive weights $(r_e)_{e\in E}$, and span programs $(\mathcal{P}^{(e)})_{e\in E}$ on a common domain $\mathcal{D}$. Let $x \in \mathcal{D}$.

1. We let
$$E^+(x) = \{x \in \mathcal{D} : x \text{ is positive in } \mathcal{P}^{(e)}\}.$$
Then, we define the *availability graph* $G^+(x) = (V, E^+(x))$, and we put the weights $(r_e^+)_{e\in E^+(x)}$ on its edges, where for all $e \in E^+(x)$,
$$r_e^+ = r'_e w_+(x, \mathcal{P}^{(e)}).$$

2. We let
$$E^-(x) = E \setminus E^+(x).$$
For all $v \in V$, we denote $S_v(x) \subseteq V$ to be the set of vertices in the same connected component as $v$ in $G^+(x)$, and we let $V^-(x) = \{S_v(x) : v \in V\}$, i.e., the set of all connected components in $G^+(x)$. Next, we define the *unavailability graph* $G^-(x) = (V^-(x), E^-(x))$, where we interpret the edge $e \in E^-(x)$ that connects $v$ to $w$ in $G$, as an edge connecting $S_v(x)$ to $S_w(x)$ in $G^-(x)$. We put the weights $(r_e^-)_{e\in E^-(x)}$ on its edges, where
$$r_e^- = \frac{r'_e}{w_-(x, \mathcal{P}^{(e)})}.$$
◀

Next, we characterize the positive and negative witnesses of a graph composition of span programs.

**7.2.7.** THEOREM (Properties of the graph composition of span programs). *Let $G = (V, E)$ be an undirected graph. Let $\mathcal{P}$ be a graph composition of $G$, with strictly positive weights $(r_e)_{e\in E}$, and span programs $(\mathcal{P}^{(e)})_{e\in E}$ on a common domain $\mathcal{D}$. Let $x \in \mathcal{D}$, and let $G^+(x) = (V, E^+(x))$ and $G^-(x) = (V^-(x), E^-(x))$ with weights $(r_e^+)_{e\in E^+(x)}$ and $(r_e^-)_{e\in E^-(x)}$ be defined as in Definition 7.2.6. Then,*

1. *$x$ is a positive input for $\mathcal{P}$ if and only if $s$ and $t$ are connected in $G^+(x)$ and if and only if $S_s(x) = S_t(x)$.*

2a. *If $x$ is a positive input for $\mathcal{P}$, then the positive witnesses for $x$ in $\mathcal{P}$ are the vectors $|w\rangle$ of the form*
$$|w\rangle = \sum_{\substack{e\in E\\ f'_e\neq 0}} f'_e\sqrt{r'_e}|w^{(e)}\rangle + \sum_{\substack{e\in E\\ f'_e=0}} |\overline{w}^{(e)}\rangle,$$
*where $\sum_{e\in E^+(x)} f'_e\sqrt{r_e^+}\,|e\rangle \in \mathcal{H}_{G^+(x)}$ is a unit st-flow in $G^+(x)$, $f'_e = 0$ when $e \in E^-(x)$, $|w^{(e)}\rangle$ is a positive witness for $x$ in $\mathcal{P}^{(e)}$, and finally $|\overline{w}^{(e)}\rangle \in \mathcal{K}^{(e)} \cap \mathcal{H}^{(e)}(x)$.*

2b. *If $x$ is a negative input for $\mathcal{P}$, then the negative witnesses for $x$ in $\mathcal{P}$ are the vectors $|w\rangle$ of the form*
$$|w\rangle = \sum_{\substack{e\in E\\ f'_e\neq 0}} f'_e\sqrt{r'_e}|w^{(e)}\rangle + \sum_{\substack{e\in E\\ f'_e=0}} |\overline{w}^{(e)}\rangle,$$
*where $|w^{(e)}\rangle$ is a negative witness for $x$ in $\mathcal{P}^{(e)}$, and $|\overline{w}^{(e)}\rangle \in (\mathcal{K}^{(e)})^\perp \cap \mathcal{H}^{(e)}(x)^\perp \cap \mathrm{Span}\{|w_0^{(e)}\rangle\}^\perp \cap \mathcal{H}^{(e)}$, $f'_e = 0$ whenever $e \in E^+(x)$, and $f'_e = f_e^-/w_-(x, \mathcal{P}^{(e)})$ whenever $e \in E^-(x)$, with $\sum_{e\in E^-(x)} f_e^-\sqrt{r_e^-}\,|e\rangle \in \mathcal{H}_{G^-(x)}$ a flow in $G^-(x)$ that derives from a potential function $U^- : V^-(x) \to \mathbb{C}$ using the weights $(r_e^-)_{e\in E^-(x)}$, satisfying $U^-_{S_s(x)} - U^-_{S_t(x)} = 1$.*

*3a. For all $x \in \mathcal{D}$,*

$$w_+(x, \mathcal{P}) = R_{s,t}(G^+(x), (r_e^+)_{e \in E^+(x)}).$$

*3b. For all $x \in \mathcal{D}$,*

$$w_-(x, \mathcal{P}) = R_{S_s(x), S_t(x)}(G^-(x), (r_e^-)_{e \in E^-(x)})^{-1}.$$

**Proof:**
We start by proving claims 2a and 3a. To that end, suppose that $x$ is a positive input and let $|w\rangle \in \mathcal{H}$. For all $e \in E$, we denote the projection of $|w\rangle$ onto the subspace $\mathcal{H}^{(e)}$ by $|\overline{w}^{(e)}\rangle$, and we let $f'_e = \langle \overline{w}^{(e)} | w_0^{(e)} \rangle / \sqrt{r'_e}$. If $f'_e \neq 0$, we write $|w^{(e)}\rangle = |\overline{w}^{(e)}\rangle / (\sqrt{r'_e} f'_e)$. We also write

$$|f'\rangle = \sum_{e \in E} f'_e \sqrt{r'_e}\, |e\rangle \in \mathcal{H}_G, \quad \text{and} \quad \left|f^+\right\rangle = \sum_{e \in E^+(x)} f'_e \sqrt{r_e^+}\, |e\rangle \in \mathcal{H}_{G^+(x)}.$$

Then, we have the sequence of equivalences

$$\begin{aligned}
&|w\rangle \text{ is a positive witness for } x \Leftrightarrow |w\rangle \in \mathcal{H}(x) \wedge |w\rangle - |w_0\rangle \in \mathcal{K} \\
&\quad \Leftrightarrow \left[\forall e \in E, |\overline{w}^{(e)}\rangle \in \mathcal{H}^{(e)}(x) \wedge |\overline{w}^{(e)}\rangle - f'_e \sqrt{r'_e} |w_0^{(e)}\rangle \in \mathcal{K}^{(e)}\right] \\
&\qquad \wedge \mathcal{E}(|f'\rangle) - \mathcal{E}(|f\rangle) \in \mathcal{E}(\mathcal{C}_G) \\
&\quad \Leftrightarrow \forall e \in E, \begin{cases} |w^{(e)}\rangle \in \mathcal{H}^{(e)}(x) \wedge |w^{(e)}\rangle - |w_0^{(e)}\rangle \in \mathcal{K}^{(e)}, & \text{if } f'_e \neq 0, \\ |\overline{w}^{(e)}\rangle \in \mathcal{K}^{(e)} \cap \mathcal{H}^{(e)}(x), & \text{if } f'_e = 0 \end{cases} \\
&\qquad \wedge |f'\rangle \in |f\rangle + \mathcal{C}_G \\
&\quad \Leftrightarrow |f'\rangle \in \mathcal{H}_G \text{ is a unit } st\text{-flow on } G \\
&\qquad \wedge \forall e \in E, \begin{cases} |w^{(e)}\rangle \text{ is a positive witness for } x \text{ in } \mathcal{P}^{(e)}, & \text{if } f'_e \neq 0, \\ |\overline{w}^{(e)}\rangle \in \mathcal{K}^{(e)} \cap \mathcal{H}^{(e)}(x), & \text{if } f'_e = 0 \end{cases} \\
&\quad \Leftrightarrow \left|f^+\right\rangle \in \mathcal{H}_{G^+(x)} \text{ is a unit } st\text{-flow on } G^+(x) \\
&\qquad \wedge \forall e \in E, \begin{cases} |w^{(e)}\rangle \text{ is a positive witness for } x \text{ in } \mathcal{P}^{(e)}, & \text{if } f'_e \neq 0, \\ |\overline{w}^{(e)}\rangle \in \mathcal{K}^{(e)} \cap \mathcal{H}^{(e)}(x), & \text{if } f'_e = 0. \end{cases}
\end{aligned}$$

This proves claim 2a. Using the notation above, the size of any positive witness $|w\rangle$ can be written as

$$\||w\rangle\|^2 = \sum_{\substack{e \in E \\ f'_e \neq 0}} |f'_e|^2 r'_e \left\||w^{(e)}\rangle\right\|^2 + \sum_{\substack{e \in E \\ f'_e = 0}} \left\||\overline{w}^{(e)}\rangle\right\|^2,$$

which is minimized when for all $e \in E$ where $f'_e = 0$, we choose $|\overline{w}^{(e)}\rangle = 0$, and for all $e \in E$ where $f'_e \neq 0$, we choose the minimal positive witness for $x$ in $\mathcal{P}^{(e)}$. Then, the witness size simplifies to

$$\||w\rangle\|^2 = \sum_{e \in E^+(x)} |f'_e|^2 r'_e w_+(x, \mathcal{P}^{(e)}) = \sum_{e \in E^+(x)} |f'_e|^2 r_e^+,$$

which is minimized when we take $|f^+\rangle$ to be the minimal unit $st$-flow in the graph $G^+(x)$, with weights $r_e^+$ for all $e \in E^+(x)$, in which case the above expression becomes the effective resistance between $s$ and $t$ in $G^+(x)$. This proves claim 3a.

Now, we turn to the negative case. To that end, let $|w\rangle \in \mathcal{H}$. Again, we let $|\overline{w}^{(e)}\rangle$ be the projection of $|w\rangle$ onto $\mathcal{H}^{(e)}$, and we write $f'_e = \langle \overline{w}^{(e)} | w_0^{(e)} \rangle$. If $f'_e \neq 0$, we write $|w^{(e)}\rangle = |\overline{w}^{(e)}\rangle / (\sqrt{r'_e} f'_e)$. We also let

$$|f'\rangle = \sum_{e \in E} f'_e \sqrt{r'_e}\, |e\rangle \in \mathcal{H}_G, \quad \text{and} \quad |f^-\rangle = \sum_{e \in E^-(x)} f_e^- \sqrt{r_e^-}\, |e\rangle \in \mathcal{H}_{G^-(x)},$$

with $f_e^- = f'_e w_-(x, \mathcal{P}^{(e)})$ for all $e \in E^-(x)$. Then, we have the sequence of equivalences

$$\begin{aligned}
&|w\rangle \text{ is a negative witness for } x \Leftrightarrow |w\rangle \in \mathcal{K}^\perp \cap \mathcal{H}(x)^\perp \wedge \langle w_0 | w \rangle = 1 \\
&\Leftrightarrow \forall e \in E, |\overline{w}^{(e)}\rangle \in (\mathcal{K}^{(e)})^\perp \cap \mathcal{H}^{(e)}(x)^\perp \cap \mathcal{H}^{(e)} \\
&\qquad \wedge \mathcal{E}(|f'\rangle) \in \mathcal{K}^\perp \wedge \mathcal{E}(|f'\rangle)^\dagger \mathcal{E}(|f\rangle) = 1 \\
&\Leftrightarrow \forall e \in E, \begin{cases} |w^{(e)}\rangle \in (\mathcal{K}^{(e)})^\perp \cap \mathcal{H}^{(e)}(x)^\perp \cap \mathcal{H}^{(e)}, & \text{if } f'_e \neq 0, \\ |\overline{w}^{(e)}\rangle \in (\mathcal{K}^{(e)})^\perp \cap \mathcal{H}^{(e)}(x)^\perp \cap \mathrm{Span}\{|w_0^{(e)}\rangle\}^\perp \cap \mathcal{H}^{(e)}, & \text{if } f'_e = 0 \end{cases} \\
&\qquad \wedge |f'\rangle \in \mathcal{C}_G^\perp \wedge \langle f' | f \rangle = 1 \\
&\Leftrightarrow \exists U : V \to \mathbb{C} \text{ s.t. } |f'\rangle \in \mathcal{H}_G \text{ is derived from } U \text{ with weights } (r'_e)_{e \in E} \\
&\qquad \wedge U_s - U_t = 1 \\
&\qquad \wedge \forall e \in E, \begin{cases} |w^{(e)}\rangle \text{ is a negative witness for } x \text{ in } \mathcal{P}^{(e)}, & \text{if } f'_e \neq 0, \\ |\overline{w}^{(e)}\rangle \in (\mathcal{K}^{(e)})^\perp \cap \mathcal{H}^{(e)}(x)^\perp \cap \mathrm{Span}\{|w_0^{(e)}\rangle\}^\perp \cap \mathcal{H}^{(e)}, & \text{if } f'_e = 0, \end{cases}
\end{aligned}$$

where in the last equivalence we used that every flow $|f\rangle \in \mathcal{C}_G^\perp \subseteq \mathcal{H}_G$ is derived from a potential function $U : V \to \mathbb{C}$, according to Lemma 7.2.4. Furthermore, in the final statement observe that if $f'_e \neq 0$, then $|w^{(e)}\rangle$ is a negative witness for $x$ in $\mathcal{P}^{(e)}$, and in particular $x$ is a negative input for $\mathcal{P}^{(e)}$, which implies that $e \in E^-(x)$, and thus $|f'\rangle$ only has support on $E^-(x)$. Therefore, the function $U : V \to \mathbb{C}$ must be constant on every connected component in $G^+(x)$, and thus we can define the function $U^- : V^-(x) \to \mathbb{C}$ such that $U_v = U^-_{S_v(x)}$. This is a potential function on $G^-(x)$, and as such using the weights $(r_e^-)_{e \in E^-(x)}$ generates a flow which for every $e \in E^-(x)$ connecting $v$ and $w$ in $G$ has value

$$\frac{U^-_{S_v(x)} - U^-_{S_w(x)}}{r_e^-} = \frac{U^-_{S_v(x)} - U^-_{S_w(x)}}{r'_e} \cdot w_-(x, \mathcal{P}^{(e)}) = f'_e w_-(x, \mathcal{P}^{(e)}) = f_e^-.$$

Thus, the existence of a potential function $U$ on $G$ that generates the flow $|f'\rangle$ with weights $(r'_e)_{e \in E}$, is equivalent to the existence of a potential function $U^-$ on $G^-(x)$ that generates the flow $|f^-\rangle$ with weights $(r_e^-)_{e \in E^-(x)}$. Hence, we can push

the series of equivalences one step further, and obtain that

$$
\begin{aligned}
&|w\rangle \text{ is a negative witness for } x \\
&\Leftrightarrow \exists U^- : V^-(x) \to \mathbb{C} \text{ s.t. } \left|f^-\right\rangle \in \mathcal{H}_{G^-(x)} \text{ is derived from } U^- \\
&\qquad \text{with weights } (r_e^-)_{e\in E^-(x)} \wedge U^-_{S_s(x)} - U^-_{S_t(x)} = 1 \\
&\qquad \wedge \forall e \in E, \begin{cases} |w^{(e)}\rangle \text{ is a negative witness for } x \text{ in } \mathcal{P}^{(e)}, & \text{if } f'_e \neq 0, \\ |\overline{w}^{(e)}\rangle \in (\mathcal{K}^{(e)})^\perp \cap \mathcal{H}^{(e)}(x)^\perp \cap \mathrm{Span}\{|w_0^{(e)}\rangle\}^\perp \cap \mathcal{H}^{(e)}, & \text{if } f'_e = 0. \end{cases}
\end{aligned}
$$

This proves claim 2b.

Moreover, using the notation introduced above, we can express the size of a negative witness $|w\rangle$ as before as

$$
\||w\rangle\|^2 = \sum_{\substack{e\in E \\ f'_e\neq 0}} |f'_e|^2 r'_e \left\||w^{(e)}\rangle\right\|^2 + \sum_{\substack{e\in E \\ f'_e=0}} \left\||\overline{w}^{(e)}\rangle\right\|^2 .
$$

Thus, to minimize this norm, we choose $|w^{(e)}\rangle$ to be the minimal negative witness whenever $f'_e \neq 0$, and we choose $|\overline{w}^{(e)}\rangle = 0$ whenever $f'_e = 0$. Then, we obtain

$$
\||w\rangle\|^2 = \sum_{e\in E^-(x)} |f'_e|^2 r'_e w_-(x, \mathcal{P}^{(e)}) = \sum_{e\in E^-(x)} |f_e^-|^2 \frac{r'_e}{w_-(x, \mathcal{P}^{(e)})} = \sum_{e\in E^-(x)} |f_e^-|^2 r_e^-,
$$

where for all $e \in E^-(x)$ with $e$ connecting $S_v(x)$ to $S_w(x)$ and $v, w \in V$, we have $f_e^- = (U^-_{S_v(x)} - U^-_{S_w(x)})/r_e^-$, and $U^-_{S_s(x)} - U^-_{S_t(x)} = 1$. In order to compute $w_-(x, \mathcal{P})$, we must choose $U^- : V^-(x) \to \mathbb{C}$ such that we minimize the above expression.

To that end, observe that $|f^-\rangle \in \mathcal{C}^\perp_{G^-(x)}$, and that $|f^-\rangle$ must have inner product 1 with any unit $S_s(x)S_t(x)$-flow in $G^-(x)$. Therefore, if we let $|f_{\min}\rangle$ be the minimum-energy $S_s(x)S_t(x)$-flow in $G^-(x)$, we find that we must choose $|f^-\rangle \in |f_{\min}\rangle / \| |f_{\min}\rangle \|^2 + (\mathrm{Span}\{|f_{\min}\rangle\}^\perp \cap \mathcal{C}^\perp_{G^-(x)})$. It follows that we minimize the norm of $|f^-\rangle$ if we choose it to be equal to $|f_{\min}\rangle / \| |f_{\min}\rangle \|^2$, from which we derive that the minimum negative witness size becomes

$$
\begin{aligned}
w_-(x, \mathcal{P}) = \left\|\left|f^-\right\rangle\right\|^2 &= \frac{\||f_{\min}\rangle\|^2}{\||f_{\min}\rangle\|^4} \\
&= \frac{1}{\||f_{\min}\rangle\|^2} = \frac{1}{R_{S_s(x),S_t(x)}\left(G^-(x), (r_e^-)_{e\in E^-(x)}\right)},
\end{aligned}
$$

where the last equality follows from the definition of the effective resistance, i.e., item 3 in Definition 7.2.5. This completes the proof of claim 3b.

Finally, claim 1 follows easily from claims 3a and 3b using Lemma 6.1.3, and thus the proof is complete. □

We can use the graph composition of span programs to recover the AND- and OR-composition constructions from the previous section, i.e., Definitions 7.1.3 and 7.1.5. For the AND-construction, using the notation from Definition 7.1.3, we can take $G = (V, E)$ to be the line graph of length $n$, with $s$ and $t$ being the nodes at both ends of the line. To each of the edges $e_1, \ldots, e_n \in E$, we associate a span program $\mathcal{P}^{(j)}$ and a weight $\alpha_j$, with $j \in [n]$. It is then an easy task to check that the graph composition of $G$ with weights $(\alpha_j)_{e_j \in E}$ and span programs $(\mathcal{P}^{(j)})_{e_j \in E}$ equals the AND-composition from Definition 7.1.3. We can recover the OR-composition from Definition 7.1.5 in a similar manner, but by using a graph with $n$ parallel edges instead. These graphs are respectively referred to as a series and parallel graph, and they visualized in Figure 7.2.2.

One of the nice properties of the graph composition construction is that it is recursive, i.e., the span programs used as inputs to graph composition, can themselves be constructed using the graph composition construction. Such a recursive definition can also be visually observed on the graph level, i.e., one can replace an edge $e$ connecting $v$ to $w$ by an $st$-connectivity graph, where $s$ and $t$ are identified with $v$ and $w$. This can be observed in Figure 7.2.3.

In doing so, it becomes apparent that recursive applications of the AND- and OR-construction allow for performing graph compositions with any series-parallel graph, i.e., a graph that can be constructed by starting with a single edge, and iteratively substituting edges by series or parallel graphs as shown in Figure 7.2.2. However, since not every graph is series-parallel, the graph composition construction introduced in this section is indeed more general than the AND- and OR-compositions introduced in the previous section.

### 7.2.3 Special case: planar graphs

In the special case where $G$ is a planar graph, and it remains planar when we add an edge between $s$ and $t$, then we can interpret the negative witnesses a bit more elegantly. The reason is that a planar graph admits a planar dual, and it turns out that we can interpret the negative witnesses as flows in this dual graph. In doing so, we recover the characterization from [JK17]. Moreover, we show that applying graph composition of span programs using the planar dual, is the same as taking the negation of a graph composition of span programs with the original graph.

We start with developing an intuitive idea of planar duality in the graph composition setting. To that end, suppose that we have an undirected graph $G = (V, E)$, and distinct source and sink nodes $s, t \in V$. Furthermore, suppose that $G$ remains planar if we add an edge between $s$ and $t$. Without loss of generality, we can always embed $G$ into the plane in such a way that $s$ and $t$ are on the outside of the graph[5]. Now, we can think of the nodes $s$ and $t$ as

[5]There exists a quite elegant proof of this fact – consider any planar embedding of the graph

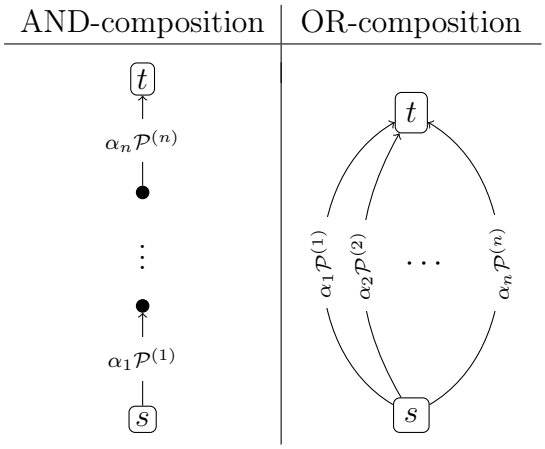



Figure 7.2.2: Graph compositions that recover the AND- and OR-composition results. The graph on the left- and right-hand sides are referred to as series and parallel graphs, respectively.

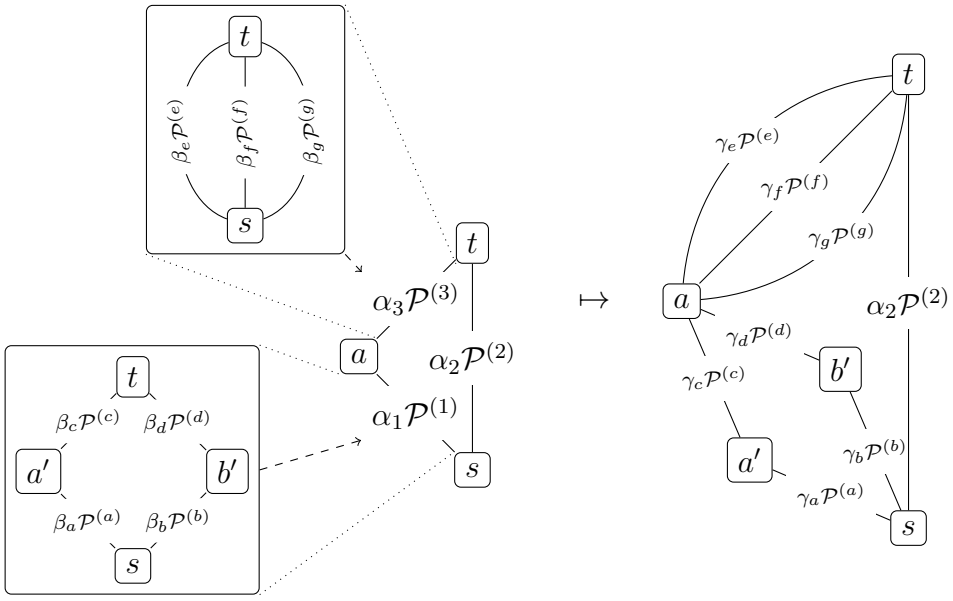



Figure 7.2.3: Visualization of recursive span program composition in terms of graphs. If the weights $\beta_a, \ldots, \beta_g$ are chosen such that the smaller graphs are properly normalized, i.e., such that their effective resistances are 1, then the resulting weights $\gamma_a, \ldots, \gamma_g$ just become the product of the weight on both edges, i.e., $\gamma_a = \alpha_1\beta_a$, and similarly for the other weights.

being on two opposite banks of a river, and the vertices as islands in this river,

with the added edge $\{s, t\}$ in the complex plane $\mathbb{C}$. Now take some point on the edge $\{s, t\}$ and compose the embedding with any Möbius transformation that has a pole at this location. Since this is a continuous operation, the resulting embedding of the graph is still planar. Moreover, $s$ and $t$ are connected through $\infty$, and as such are located on the outside of the graph.

connected by dams which represent the edges of the graph. An example is shown in Figure 7.2.4.

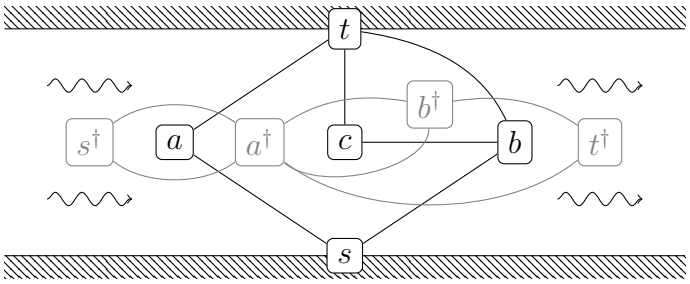



Figure 7.2.4: The planar dual of an *st*-connectivity graph. One can think of $s$ and $t$ as being located on two opposite banks of a river. The faces of its planar embedding form the set of nodes in the dual graph. Every edge in the original graph has a corresponding dual edge in the dual graph that crosses it.

Next, let's say that we consider a restricted graph $G^+(x)$, where only a subset of the edges of $G$ are present. Let $G^\dagger$ be the planar dual graph (to be formally defined shortly, in Definition 7.2.8), and we similarly consider a restriction of the dual graph $(G^\dagger)^+(x)$ by only selecting the edges that are dual to those *not* present in $G^+(x)$. Then, with the visualization in Figure 7.2.4 in mind, it becomes apparent that we are in exactly one of two cases, i.e., either there exists a path vertically going from $s$ to $t$ in $G^+(x)$ (i.e., we can cross the river along the dams, like in Figure 7.2.4), or there is a path horizontally going from $s^\dagger$ to $t^\dagger$ in $(G^\dagger)^+(x)$ (i.e., a boat can pass through the river).

Now, recall from the previous subsection that a positive input in the graph composition construction is an input such that there exists a path from $s$ to $t$ in the restricted graph $G^+(x)$. Thus, for a negative input, there exists a path from $s^\dagger$ to $t^\dagger$ in the restricted dual graph $(G^\dagger)^+(x)$. Hence, if instead of performing a graph composition $\mathcal{P}$ with graph $G$ and span programs $(\mathcal{P}^{(e)})_{e\in E}$, we perform a graph composition with the dual graph $G^\dagger$, and the negated span programs $(\neg\mathcal{P}^{(e)})_{e\in E}$, we end up computing the negation of the function that is computed by $\mathcal{P}$.

It turns out that this analogy can be pushed further by also taking the reciprocals of the weights, i.e., by setting the weights on the dual edges to be $1/r_e$. We show in Theorem 7.2.10 that with this choice of weights, we end up exactly constructing $\neg\mathcal{P}$.

First, we formally define the planar dual.

**7.2.8.** Definition (Planar dual of a graph). Let $G = (V, E)$ be an undirected planar graph with distinct source and sink nodes $s$ and $t$ that remains planar when we add an edge between $s$ and $t$. Let $F$ be the set of faces of a planar embedding of $G$ with the added edge between $s$ and $t$. For every edge $e \in E$, let

$e^\dagger$ be an edge connecting the two faces adjacent to $e$, and let the default direction of $e^\dagger$ be connecting the left face to the right face, when we traverse $e \in E$ in its default direction. Let $E^\dagger = \{e^\dagger : e \in E\}$ be the set of dual edges. The *planar dual graph* is $G^\dagger = (F, E^\dagger)$. Furthermore, let $s^\dagger$ and $t^\dagger$ be the faces that are adjacent to the edge between $s$ and $t$. ◀

It turns out that we can very elegantly characterize the circulation space and the minimum-energy unit $s^\dagger t^\dagger$-flow of the dual graph. This is the objective of the following lemma.

**7.2.9.** LEMMA (Properties of the flow space of the dual graph). *Let $G = (V, E)$ be an undirected planar graph with distinct source and sink nodes $s, t \in V$, that remains planar when we add an edge between $s$ and $t$. Let $(r_e)_{e\in E}$ be strictly positive weights on the edges. Let $G^\dagger = (F, E^\dagger)$ be the planar dual, with distinct source and sink nodes $s^\dagger, t^\dagger \in F$. For all $e \in E$, let $r_{e^\dagger} = 1/r_e$, and identify $|e\rangle$ in $\mathcal{H}_G$ and $|e^\dagger\rangle$ in $H_{G^\dagger}$, implying $\mathcal{H}_G = \mathcal{H}_{G^\dagger}$. Let $|f\rangle$ be the minimum-energy unit $st$-flow in $G$ with weights $(r_e)_{e\in E}$. Then,*

$$\mathcal{H}_G = \mathcal{C}_G \oplus \mathrm{Span}\{|f\rangle\} \oplus C_{G^\dagger} = \mathcal{H}_{G^\dagger}. \tag{7.2.4}$$

*Moreover, if $|f^\dagger\rangle$ is the minimum-energy $s^\dagger t^\dagger$-flow in $G^\dagger$ with weights $(r_{e^\dagger})_{e^\dagger\in E^\dagger}$, then $|f\rangle = |f^\dagger\rangle$.*

**Proof:**
The main thing to prove here is that any $st$-flow in the primal graph is orthogonal to any circulation in the dual graph. To that end, let $|f\rangle$ be an $st$-flow in $G$, and let $v \in V$ be an internal vertex (i.e., $v \notin \{s, t\}$). Consider the faces that are adjacent to this vertex, and let $e_1^\dagger, \ldots, e_n^\dagger \in E^\dagger$, with directions $m_1, \ldots, m_n \in \{-1, 1\}$ be the cycle in the dual graph encircling this vertex in the clockwise direction. Then, for all $j \in [n]$, $m_j = 1$ if and only if $e_j$, i.e., the primal edge of $e_j^\dagger$, is outgoing from $v$. Thus, if we send a flow of 1 along this cycle, we obtain a circulation in $G^\dagger$

$$|f^\dagger\rangle = \sum_{j=1}^{n} m_j \sqrt{r_{e_j^\dagger}} |e_j^\dagger\rangle \in \mathcal{C}_{G^\dagger}.$$

Moreover, $\mathcal{C}_{G^\dagger}$ is spanned by such circulations, and thus it remains to show that $|f\rangle$ is orthogonal to $|f^\dagger\rangle$. To that end, observe that

$$\langle f^\dagger | f\rangle = \sum_{j=1}^{n} m_j \sqrt{r_{e_j^\dagger}} f_{e_j} \sqrt{r_{e_j}} = \sum_{j=1}^{n} m_j f_{e_j} = \sum_{e\in N^+(v)} f_e - \sum_{e\in N^-(v)} f_e = 0.$$

Thus, indeed, any $st$-flow in $G$ is orthogonal to $\mathcal{C}_{G^\dagger}$. Finally, using claim 2 from Lemma 7.2.2, we indeed obtain Equation (7.2.4), and from symmetry it follows that $|f\rangle = |f^\dagger\rangle$. This completes the proof. □

We now proceed with analyzing the properties of graph compositions of span programs using the planar dual graph rather than the original one. The following theorem shows that if one negates the original span programs, and takes the reciprocal of the weights, then one essentially constructs the negation of the original graph composition span program.

**7.2.10.** THEOREM (Span program negation and the planar dual).
*Let $G = (V, E)$ be an undirected planar graph with distinct source and sink nodes $s, t \in V$ and strictly positive weights $(r_e)_{e\in E}$, that remains planar when we add an edge between $s$ and $t$. Let $G^\dagger = (F, E^\dagger)$ be its planar dual with source and sink nodes $s^\dagger, t^\dagger \in F$, and for all edges $e \in E$, let $r_{e^\dagger} = 1/r_e$. Furthermore, for all $e \in E$, let $\mathcal{P}^{(e)}$ be a span program on some fixed domain $\mathcal{D}$, and let $\mathcal{P}$ be the graph composition of $G$ with edge weights $(r_e)_{e\in E}$ and span programs $(\mathcal{P}^{(e)})_{e\in E}$. Let $\mathcal{P}^\dagger$ be the graph composition of $G^\dagger$ with edge weights $(r_{e^\dagger})_{e^\dagger\in E^\dagger}$ and span programs $(\neg\mathcal{P}^{(e)})_{e^\dagger\in E^\dagger}$. Then, $\mathcal{P}^\dagger = \neg\mathcal{P}$.*

**Proof:**
Write $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ and $\mathcal{P}^\dagger = (\mathcal{H}^\dagger, x \mapsto \mathcal{H}^\dagger(x), \mathcal{K}^\dagger, |w_0^\dagger\rangle)$. Similarly, for all $e \in E$, we write $\mathcal{P}^{(e)} = (\mathcal{H}^{(e)}, x \mapsto \mathcal{H}^{(e)}(x), \mathcal{K}^{(e)}, |w_0^{(e)}\rangle)$, and $\neg\mathcal{P}^{(e)} = (\mathcal{H}^{(e^\dagger)}, x \mapsto \mathcal{H}^{(e^\dagger)}(x), \mathcal{K}^{(e^\dagger)}, |w_0^{e^\dagger}\rangle)$. To show that $\mathcal{P}^\dagger$ is indeed the negation of $\mathcal{P}$, we must check that their four components are related as in the definition of span program negation, i.e., Definition 7.1.1.

We easily observe that $\mathcal{H}^\dagger = \mathcal{H}$, since

$$\mathcal{H}^\dagger = \bigoplus_{e^\dagger\in E^\dagger} \mathcal{H}^{(e^\dagger)} = \bigoplus_{e\in E} \mathcal{H}^{(e)} = \mathcal{H}.$$

Similarly, for any $x \in \mathcal{D}$, we have

$$\mathcal{H}^\dagger(x) = \bigoplus_{e^\dagger\in E^\dagger} \mathcal{H}^{(e^\dagger)}(x) = \bigoplus_{e\in E} \left(\mathcal{H}^{(e)}(x)^\perp \cap \mathcal{H}^{(e)}\right) = \mathcal{H}(x)^\perp.$$

Since the minimum-energy unit flows from $s$ to $t$ and $s^\dagger$ to $t^\dagger$ in the primal and dual graphs, $|f\rangle$ and $|f^\dagger\rangle$, respectively, are the same by virtue of Lemma 7.2.9, we obtain that $|w_0\rangle = |w_0^\dagger\rangle$. Thus, it remains to check that $\mathcal{K}^\dagger = (\mathcal{K} \oplus \mathrm{Span}\{|w_0\rangle\})^\perp$. To that end, observe that for every $e \in E$, we have $\mathcal{K}^{(e^\dagger)} = \mathcal{H}^{(e)} \cap (\mathcal{K}^{(e)} \oplus$

$\mathrm{Span}\{|w_0^{(e)}\rangle\})^\perp$. Thus, we have

$$\begin{aligned}
\mathcal{K}^\dagger &= \bigoplus_{e^\dagger \in E^\dagger} \mathcal{K}^{(e^\dagger)} \oplus \mathcal{E}(\mathcal{C}_{G^\dagger}) \\
&= \left[\bigoplus_{e\in E} \mathcal{H}^{(e)} \cap \left(\mathcal{K}^{(e)} \oplus \mathrm{Span}\{|w_0^{(e)}\rangle\}\right)^\perp\right] \oplus \mathcal{E}(\mathcal{H}_G \cap (\mathcal{C}_G \oplus \mathrm{Span}\{|f\rangle\})^\perp) \\
&= \left[\bigoplus_{e\in E} \left(\mathcal{H}^{(e)} \cap (\mathcal{K}^{(e)})^\perp\right) \cap \bigoplus_{e\in E} \left(\mathcal{H}^{(e)} \cap (\mathrm{Span}\{|w_0^{(e)}\rangle\})^\perp\right)\right] \\
&\quad \oplus \left[\mathcal{E}(\mathcal{H}_G) \cap \mathcal{E}(\mathcal{C}_G)^\perp \cap \mathrm{Span}\{|w_0\rangle\}^\perp\right] \\
&= \left(\left[\bigoplus_{e\in E} \mathcal{K}^{(e)}\right]^\perp \cap \left[\bigoplus_{e\in E} \mathrm{Span}\{|w_0^{(e)}\rangle\}\right]^\perp\right) \\
&\quad \oplus \left(\mathcal{E}(\mathcal{H}_G) \cap \mathcal{E}(\mathcal{C}_G)^\perp \cap \mathrm{Span}\{|w_0\rangle\}^\perp\right) \\
&= \left[\bigoplus_{e\in E} \mathcal{K}^{(e)} \oplus \mathcal{E}(\mathcal{H}_G)\right]^\perp \oplus \left(\mathcal{E}(\mathcal{H}_G) \cap \mathcal{E}(\mathcal{C}_G)^\perp \cap \mathrm{Span}\{|w_0\rangle\}^\perp\right) \\
&= \left[\bigoplus_{e\in E} \mathcal{K}^{(e)} \oplus \mathcal{E}(\mathcal{C}_G)\right]^\perp \cap \mathrm{Span}\{|w_0\rangle\}^\perp = \mathcal{K}^\perp \cap \mathrm{Span}\{|w_0\rangle\}^\perp.
\end{aligned}$$

This completes the proof. □

Thus, we have found a way to interpret the negation of a span program $\mathcal{P}$ formed by taking a graph composition with a planar graph, i.e., we build the negation $\mathcal{P}^\dagger$ by taking the graph composition using the dual graph $G^\dagger$, and using the negations of the span programs that are associated to the edges of the original graph, and the reciprocals of the edge weights. We can now use the properties proved in Theorem 7.1.2 to interpret the witnesses of $\mathcal{P}$ in $\mathcal{P}^\dagger$ and vice versa. In particular, we remark here that the negative witnesses in $\mathcal{P}$ can be interpreted as positive witnesses in $\mathcal{P}^\dagger$, which by Theorem 7.2.7 we know to be unit $s^\dagger t^\dagger$-flows in the availability graph of $G^\dagger$.

### 7.2.4 Graph composition examples

We continue this section with a showcase of the techniques developed thus far. To that end, we devise an optimal span program that computes the threshold function. This is the objective of the following example.

**7.2.11.** Example (Span program for the threshold function). Let $k \in [n]$ and $J$

be a finite set of size $n$, i.e., $|J| = n$. Define $f_k^{(J)} : \{0,1\}^J \to \{0,1\}$, as

$$f_k^J(x) = \begin{cases} 1, & \text{if } |x| \geq k, \\ 0, & \text{otherwise.} \end{cases}$$

For every $j \in J$, we define the identity span program (recall Example 6.1.4) that takes a bit string $x \in \{0,1\}^J$ and evaluates the bit $x_j$. The resulting span program is $\mathcal{P}^{(j)} = (\mathrm{Span}\{|*\rangle\}, x \mapsto \mathcal{H}_j(x), \{0\}, |*\rangle)$ on $\{0,1\}^J$, where

$$\mathcal{H}_j(x) = \begin{cases} \mathrm{Span}\{|*\rangle\}, & \text{if } x_j = 1, \\ \{0\}, & \text{otherwise.} \end{cases}$$

Then, for all $j \in J$, $\mathcal{P}^{(j)}$ computes $x_j$, and $W_+(\mathcal{P}^{(j)}) = W_-(\mathcal{P}^{(j)}) = 1$. In a slight abuse of notation, we will write $x_j$ for the span program $\mathcal{P}^{(j)}$.

Now, we inductively define span programs $\mathrm{Th}_J^k$ that compute the threshold function on $J$ with threshold $k$, by

$$\mathrm{Th}_J^1 = \bigvee_{j \in J} x_j, \qquad \text{and} \qquad \mathrm{Th}_J^{k+1} = \bigvee_{j \in J} ((n-k)x_j \wedge k\,\mathrm{Th}_{J\setminus\{j\}}^k).$$

We refer to $\mathrm{Th}_J^k$ as the *threshold span program on $J$ with threshold $k$*. For all $x \in \{0,1\}^J$, we have

$$w_+(x, \mathrm{Th}_J^k) = \begin{cases} \frac{n-k+1}{|x|-k+1}, & \text{if } |x| \geq k, \\ \infty, & \text{otherwise,} \end{cases} \qquad w_-(x, \mathrm{Th}_J^k) = \begin{cases} \infty, & \text{if } |x| \geq k, \\ \frac{k}{k-|x|}, & \text{otherwise.} \end{cases} \tag{7.2.5}$$

Thus, $\mathrm{Th}_J^k$ indeed computes $f_J^k$, and we have

$$W_+(\mathrm{Th}_J^k) = n-k+1, \quad W_-(\mathrm{Th}_J^k) = k, \quad \text{and} \quad C(\mathrm{Th}_J^k) = \sqrt{k(n-k+1)}.$$

◁

**Proof of the witness sizes in Example 7.2.11:**
We give a proof by induction to $k$ of the formulae for the witness sizes, as stated in Equation (7.2.5). If $k = 1$, then the expressions for the positive and negative witnesses reduce to those of the OR-span program, as introduced in Example 6.1.6. Thus, this provides the basis for induction.

Now, suppose that the expressions for the positive and negative witness sizes hold for all integers up to $k \in \mathbb{N}$. We then show that they also hold for $k+1$. To

that end, suppose that $x \in \{0,1\}^J$ such that $|x| \geq k+1$. Then,

$$\begin{aligned}
w_+(x, \mathrm{Th}_J^{k+1}) &= w_+\left(x, \bigvee_{j\in J}((n-k)x_j \wedge k\,\mathrm{Th}_{J\setminus\{j\}}^k)\right) \\
&= \left[\frac{1}{n}\sum_{j\in J}\frac{1}{w_+(x,(n-k)x_j \wedge k\,\mathrm{Th}_{J\setminus\{j\}}^k)}\right]^{-1} \\
&= \left[\frac{1}{n}\sum_{j\in J}\frac{n-k+k}{(n-k)w_+(x,x_j)+kw_+(x,\mathrm{Th}_{J\setminus\{j\}}^k)}\right]^{-1} \\
&= \left[\sum_{\substack{j\in J\\ x_j=1}}\frac{1}{(n-k)+k\frac{n-k}{|x|-k}}\right]^{-1} = \left[\frac{|x|(|x|-k)}{(n-k)(|x|-k)+k(n-k)}\right]^{-1} \\
&= \frac{n-k}{|x|-k},
\end{aligned}$$

and so we find that the formula for the positive witness size also holds for $k+1$. Similarly, suppose that $|x| \leq k$. Then,

$$\begin{aligned}
w_-(x, \mathrm{Th}_J^{k+1}) &= w_-\left(x, \bigvee_{j\in J}((n-k)x_j \wedge k\,\mathrm{Th}_{J\setminus\{j\}}^k)\right) \\
&= \frac{1}{n}\sum_{j\in J} w_-\left(x,(n-k)x_j \wedge k\,\mathrm{Th}_{J\setminus\{j\}}^k\right) = \frac{1}{n}\sum_{j\in J}\frac{n-k+k}{\frac{n-k}{w_-(x,x_j)}+\frac{k}{w_-(x,\mathrm{Th}_{J\setminus\{j\}}^k)}} \\
&= \sum_{\substack{j\in J\\ x_j=0}}\frac{1}{n-k+k\frac{k-|x|}{k}} + \sum_{\substack{j\in J\\ x_j=1}}\frac{1}{k\frac{k-|x|+1}{k}} = \frac{n-|x|}{n-|x|} + \frac{|x|}{k-|x|+1} = \frac{k+1}{k+1-|x|}.
\end{aligned}$$

Thus, the expression for the negative witness size also holds for $k+1$. This completes the proof of the induction step, and hence we conclude the validity of Equation (7.2.5). The expressions for the witness complexities and the span program complexity follow immediately. This completes the proof. □

Note that the complexity we obtain, i.e., $\sqrt{k(n-k+1)}$, matches the optimal value of the adversary bound for the threshold function on $n$ bits with threshold $k$, as proved in [Bel14, Proposition 3.32][6]. Furthermore, since we have defined the threshold span programs through AND- and OR-compositions, we can also view the entire construction as a graph composition of a series-parallel graph. We have drawn this graph in the case where $n = 4$ and $k = 3$ in Figure 7.2.5.

[6]It already follows from combining the results in [BBC+01; BHM+02; Rei11] that the adversary bound for the threshold function on $n$ bits with threshold $k$ is $\Theta(\sqrt{k(n-k+1)})$, and [Bel14] settles the exact constant.

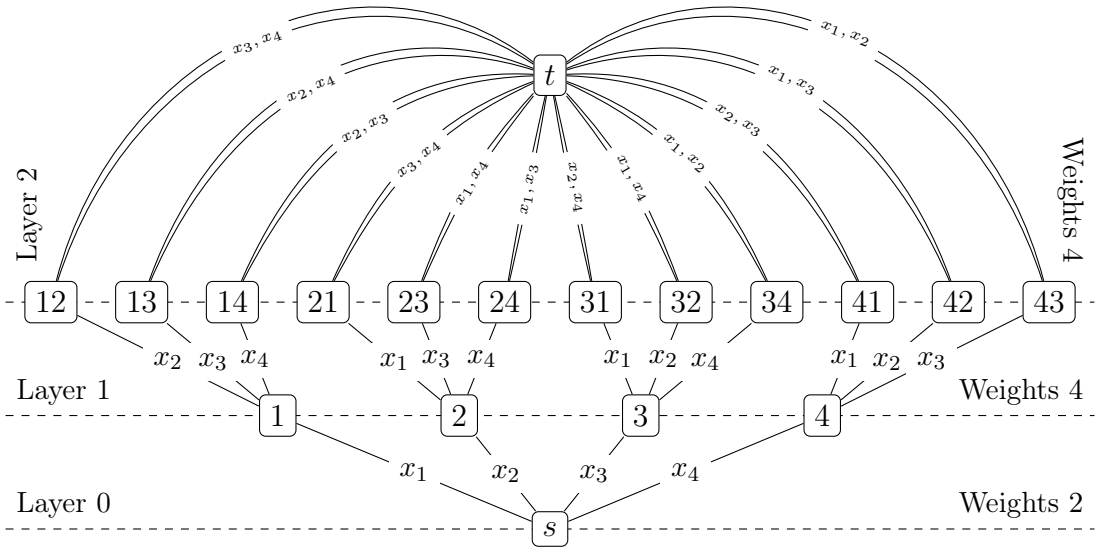



Figure 7.2.5: Illustration of the graph composition construction of a span program that evaluates the threshold function on $n = 4$ bits with threshold $k = 3$. The weights on all the edges in Layers 0, 1, 2 are 2, 4, 4, respectively.

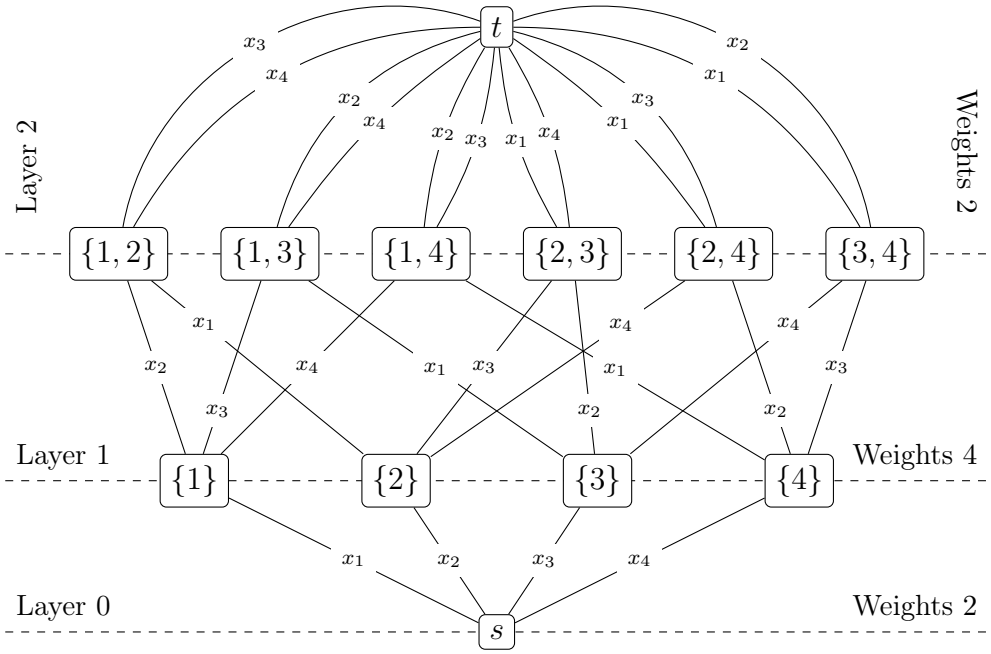



Figure 7.2.6: Illustration of the alternative graph composition construction of a span program that evaluates the threshold function on $n = 4$ bits, with threshold $k = 3$. The weights on the edges in Layers 0, 1, 2 are 2, 4, 2, respectively.

Recall that the number of dimensions in the Hilbert space of a span program formed by the graph composition construction is the sum of the dimensions of the Hilbert spaces in the individual span programs. In Example 7.2.11, each of the individual span programs are one-dimensional, which implies that the number of dimensions in the Hilbert space of $\mathrm{Th}_n^k$ is the number of edges in the graph. In the construction above, one can easily calculate that the number of edges is $n!/(n-k)!$, which scales *super-exponentially* in $n$ for large values of $k$. Thus, even though we have constructed a span program with optimal span program complexity, the dimension of the Hilbert space might still be very large. Since the number of qubits required to implement the span program algorithm, Algorithm 6.1.18, is at least the logarithm of the dimension of the Hilbert space $\mathcal{H}$, we ultimately obtain that the number of qubits scales super-linearly in $n$ for large values of $k$ as well. Note that the approximate counting algorithm in [BHM+02] achieves the optimal complexity, with a number of qubits logarithmic in $n$, so the algorithm constructed here is very suboptimal in terms of its space requirements.

One way to reduce the number of required qubits is by finding a graph composition with fewer edges that still computes the threshold function with optimal span program complexity. It turns out that we can indeed construct such a graph, but for that purpose we have to step out of the planar setting. This is the objective of the following example.

**7.2.12.** Example (Alternative span program for the threshold function).
Let $k \in [n]$, and let $f_n^k : \{0,1\}^n \to \{0,1\}$ be defined as

$$f_n^k(x) = \begin{cases} 1, & \text{if } |x| \geq k, \\ 0, & \text{otherwise.} \end{cases}$$

For every bit of the input, we define the identity span program (recall Example 6.1.4) that evaluates said bit, i.e., for every $j \in [n]$, we define the span program $\mathcal{P}^{(j)} = (\mathrm{Span}\{|*\rangle\}, x \mapsto \mathcal{H}_j(x), \{0\}, |*\rangle)$ on $\{0,1\}^n$, where

$$\mathcal{H}_j(x) = \begin{cases} \mathrm{Span}\{|*\rangle\}, & \text{if } x_j = 1, \\ \{0\}, & \text{otherwise.} \end{cases}$$

Then $\mathcal{P}^{(j)}$ computes $x_j$, and thus we use $x_j$ as shorthand notation for $\mathcal{P}^{(j)}$.

Now, we describe a graph $G = (V, E)$ with $k+1$ layers (see Figure 7.2.6 for an example). In the 0th layer, we have node $s$, and in the $k$th layer, we have node $t$. In the $j$th layer, with $j \in [k-1]$, we have nodes labeled by $S \subseteq [n]$, with $|S| = j$.

Next, we connect $s$ to all nodes in the first layer, and label the edge connecting $s$ to $\{j\}$ by $x_j$. Similarly, from every vertex in the $(k-1)$th layer labeled by $S$, we let $n-k+1$ edges go to $t$, respectively labeled by $x_j$ for all $j \in [n] \setminus S$. Finally, with $j \in [k-2]$, there is an edge between a pair of nodes $S_j$ and $S_{j+1}$ in layers $j$ and $j+1$, precisely when we can write $S_{j+1} = S_j \cup \{j'\}$ for some $j' \in [n]$, and if this is the case the edge is labeled by $x_{j'}$.

It remains to put weights on the edges. To that end, if an edge connects layer $j$ to $j+1$, for $j \in [k-1]_0$, the corresponding weight is

$$r_j = \frac{(k-1)!(n-k+1)}{(k-j-1)!j!}.$$

Strong numerical evidence shows that the graph composition of graph $G$ with weights $(r_e)_{e\in E}$ and span programs $(\mathcal{P}^{(e)})_{e\in E}$ computes the function $f_n^k$, and the witness sizes and complexities are the same as in Example 7.2.11. ◁

We provide an illustration of this alternative construction for the threshold span program for $n = 4$ and $k = 3$ in Figure 7.2.6. We have gathered strong numerical evidence that this alternative construction achieves the same witness sizes and complexities as in Example 7.2.11. The analysis appears to be rather tricky, though, so we leave that for future work.

The number of edges in the new construction can be evaluated to be

$$\sum_{\ell=0}^{k} \binom{n}{\ell}(n-\ell) = n\sum_{\ell=0}^{k} \binom{n-1}{\ell} \leq n2^{n-1}.$$

Thus, the number of dimensions in the Hilbert space of the threshold span program constructed in Example 7.2.12 scales only exponentially in $n$, rather than super-exponentially as was the case in Example 7.2.11. This implies that the number of qubits required to implement this algorithm is linear in $n$, which is better than the construction in Example 7.2.11, but still severely suboptimal.

Finally, we observe that we can use the span program for the threshold function as a black box to build a span program for the exact-weight function. This is the objective of the following example.

**7.2.13.** Example (Span program for the exact-weight function).
Let $k \in [n]$ and let $J$ be a finite set of size $n$, i.e., $|J| = n$. We define the exact-weight function $f_J^k : \{0,1\}^J \to \{0,1\}$ as

$$f_J^k(x) = \begin{cases} 1, & \text{if } |x| = k, \\ 0, & \text{otherwise.} \end{cases}$$

We define the span program that computes the exact-weight function by

$$\mathrm{EW}_J^k = k\,\mathrm{Th}_J^k \wedge (n-k)(\neg\,\mathrm{Th}_J^{k+1}).$$

Then, for all $x \in \{0,1\}^J$, we have

$$w_+(x, \mathrm{EW}_J^k) = \begin{cases} \frac{n+2k(n-k)}{n}, & \text{if } |x| = k, \\ \infty, & \text{otherwise,} \end{cases} \qquad w_-(x, \mathrm{EW}_J^k) = \begin{cases} \infty, & \text{if } |x| = k \\ \frac{n}{||x|-k|}, & \text{otherwise.} \end{cases}$$

Thus, $\mathrm{EW}_J^k$ indeed computes $f_J^k$ and the resulting complexities are

$$W_+(\mathrm{EW}_J^k) = \frac{n + 2k(n-k)}{n}, \qquad W_-(\mathrm{EW}_J^k) = n,$$

and

$$C(\mathrm{EW}_J^k) = \sqrt{n + 2k(n-k)}.$$

◁

**Proof:**
We check the formulae for the witness sizes directly. Suppose that $x \in \{0,1\}^J$ with $|x| = k$. Then,

$$\begin{aligned} w_+(x, \mathrm{EW}_J^k) &= w_+(x, k\,\mathrm{Th}_J^k \wedge (n-k)(\neg\,\mathrm{Th}_J^{k+1})) \\ &= \frac{k w_+(x, \mathrm{Th}_J^k) + (n-k) w_+(x, \neg\,\mathrm{Th}_J^{k+1})}{k+n-k} \\ &= \frac{k w_+(x, \mathrm{Th}_J^k) + (n-k) w_-(x, \mathrm{Th}_J^{k+1})}{n} \\ &= \frac{k(n-k+1) + (n-k)(k+1)}{n} = \frac{n + 2k(n-k)}{n}, \end{aligned}$$

and similarly, if $|x| \neq k$, then,

$$\begin{aligned} w_-(x, \mathrm{EW}_J^k) &= w_-(x, k\,\mathrm{Th}_J^k \wedge (n-k)(\neg\,\mathrm{Th}_J^{k+1})) \\ &= \left[\frac{1}{k+n-k}\left[\frac{k}{w_-(x, \mathrm{Th}_J^k)} + \frac{n-k}{w_-(x, \neg\,\mathrm{Th}_J^{k+1})}\right]\right]^{-1} \\ &= n\left[\frac{k}{w_-(x, \mathrm{Th}_J^k)} + \frac{n-k}{w_+(x, \mathrm{Th}_J^{k+1})}\right]^{-1}. \end{aligned}$$

If $|x| < k$, then we can further rewrite this to

$$w_-(x, \mathrm{EW}_J^k) = n\left[k\frac{k-|x|}{k}\right]^{-1} = \frac{n}{k-|x|} = \frac{n}{||x|-k|},$$

and similarly if $|x| > k$, then we obtain

$$w_-(x, \mathrm{EW}_J^k) = n\left[(n-k)\frac{|x|-k}{n-k}\right]^{-1} = \frac{n}{|x|-k} = \frac{n}{||x|-k|}.$$

The witness complexities and the span program complexity then follow immediately. This completes the proof. □

The remarkable thing about the above construction is that numerical evidence strongly suggests that it is optimal, i.e., it seems that $\sqrt{n + 2k(n-k)}$ is indeed

the value of the adversary bound for the exact-weight function on $n$ bits with weight $k$. If true, then this relation is a novelty. The easiest way to prove it would be to come up with primal adversary bound solutions whose objective values match the complexity of the span program constructed here. We leave this for future work.

The obvious next step is to generalize the above construction for the exact-weight function to the more general class of interval functions, i.e., functions that output 1 whenever the Hamming weight $|x|$ of the input is in some interval $a \leq |x| \leq b$. Indeed, one can construct a span program of the form $\alpha \operatorname{Th}_J^a \wedge \beta(\neg \operatorname{Th}_J^{b+1})$, but it turns out that even by optimizing the choice of the weights $\alpha$ and $\beta$, one cannot always obtain a span program whose complexity equals the adversary bound for the corresponding interval function.

The smallest non-trivial example is the function $f : \{0,1\}^5 \to \{0,1\}$, where $f(x) = 1$ if and only if $2 \leq |x| \leq 3$. The best span program complexity that can be achieved with the above construction appears to be $\sqrt{12}$, but we have numerically generated a solution to the reformulated dual adversary bound that attains an objective value of approximately $\sqrt{11.8892}$. At this point, we have no credible closed-form formula for the interval function that predicts this value 11.8892, nor do we have any graph composition construction for this interval function that attains this complexity. Thus, this function is a very natural next step in obtaining a better understanding of optimal solutions to the reformulated dual adversary bound. It would be very interesting to come up with graph constructions that achieve optimality for all symmetric functions.

Finally, a very nice question for future research is whether we can use the construction for the threshold and exact-weight functions in the more general setting where the inputs are functions themselves again. That is, let $f_1, \ldots, f_n$ be boolean functions with disjoint support, and suppose that we would like to know if $k$ or more of these functions evaluate to 1 on some given tuple of inputs $x^{(1)}, \ldots, x^{(n)}$. Can we characterize the adversary value of this decision problem in terms of the adversary values of the individual functions $\mathrm{ADV}^{\pm}(f^{(j)})$, to obtain a result similar in flavor to Corollary 7.1.14? This problem is still wide open, and provides a very nice follow-up question from this work.

## 7.3 Quantum algorithms from classical decision trees

As a concrete demonstration of the applicability of the graph composition of span programs, we showcase how it can be used to take any classical algorithm that solves a decision problem, and turn it into a quantum one. The results achieved in this section are heavily based on [CMP22], even though there are slight differences in the approach used to arrive at them.

### 7.3.1 Introduction

Observe that any deterministic classical query algorithm computing a function $f : \mathcal{D} \to R$ with $\mathcal{D} \subseteq [\ell]^n$ and $R$ any set, can be visualized as a decision tree. The root node of the decision tree is the start of the algorithm, and every internal node, i.e., non-leaf node, is labeled by an index $j \in [n]$. The outgoing edges from this node are labeled by elements from $[\ell]$. The leaves are labeled by elements from the range $R$.

The classical deterministic algorithm represented by the decision tree traverses it by starting at the root node, querying the input at the labeled element, and then traverses the outgoing edge labeled by the outcome of the query. Once a leaf is reached, the label of the corresponding leaf is the output of the function. See Figure 7.3.1 for an example.

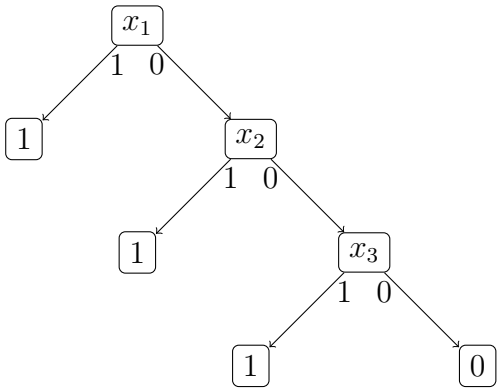



Figure 7.3.1: Illustration of a decision tree that computes the OR-function on 3 bits, i.e., $x_1 \vee x_2 \vee x_3$. The classical algorithm starts at the root node, where it queries the first bit, $x_1$. Then, it traverses the edge labeled by the outcome of this query, and repeats this process. Once it reaches a leaf, it outputs the label of the leaf.

It is clear that the classical deterministic algorithm based on a decision tree $T$ performs $\text{depth}(T)$ queries in the worst case, where $\text{depth}(T)$ is the length of the longest path from the root node to a leaf. The high-level question that we consider in this section is whether one can use the description of the decision tree to construct a quantum algorithm that computes the same function with bounded error, whilst making a number of queries that is significantly smaller than $\text{depth}(T)$.

There have been several recent works that formulate an answer to this question. In [LL16], Lin and Lin showed how a decision tree computing a function $f : \mathcal{D} \to R$, with $\mathcal{D} \subseteq \{0,1\}^n$ and $R$ any set, can be turned into a quantum query algorithm that computes $f$ with bounded error. The construction requires a "guessing scheme," i.e., at every internal node in the decision tree, one arbitrary outgoing edge is labeled the "guessed edge" and its associated outcome is referred

to as the "guessed outcome". The quantum algorithm they construct requires $\mathcal{O}(\sqrt{G \cdot \text{depth}(T)})$ queries to the input in the worst-case, where $G$ is the maximum number of non-guessed edges in a path from root to leaf in the decision tree. Since $G \leq \text{depth}(T)$, their quantum algorithm is always at least as efficient as its classical counterpart.

Next, in [BT20], Beigi and Taghavi reprove the result from [LL16] in the more general setting where the decision tree computes the function $f : \mathcal{D} \to R$, with $\mathcal{D} \subseteq [\ell]^n$, $\ell \in \mathbb{N}$ and $R$ any set. They arrive at this result through carefully constructing a feasible solution to the dual adversary bound, and showing that the objective value is at most $\mathcal{O}(\sqrt{G \cdot \text{depth}(T)})$, where $G$ is as before. Subsequently, in [BTT22], Beigi, Taghavi and Tajdini showed that the resulting algorithm can also be implemented time-efficiently, and in [Tag22], Taghavi used these techniques to construct a query-optimal quantum algorithm for the oracle identification problem, simplifying an earlier construction by Kothari [Kot14].

In this section, we restrict ourselves to the setting where $f : \mathcal{D} \to \{0,1\}$ with $\mathcal{D} \subseteq \{0,1\}^n$, and show that in this setting, the result in [BT20] can be phrased as a graph composition construction. We also argue that the optimal guessing scheme, as required in the constructions showcased in [LL16; BT20], bears a direct connection to the decision tree rank, as introduced by [EH89]. We subsequently improve over the construction presented in [BT20] by providing a better weight assignment on the edges of the graph. We also prove that this assignment of the weights is optimal, and derive a bound on the number of queries performed by the resulting algorithm in terms of the size of the decision tree. The collective state-of-the-art results, i.e., the combined results from [BT20; CMP22], are summarized in Theorem 7.3.8.

### 7.3.2 Decision trees and its properties

Intuitively speaking, the size of the decision tree tells us something about the complexity of the algorithm it represents. However, there are several ways in which the size of the decision tree can be measured. We restrict ourselves to decision trees that solve a decision problem with binary input, i.e., that compute a boolean function $f : \mathcal{D} \to \{0,1\}$ with $\mathcal{D} \subseteq \{0,1\}^n$. We formalize several of these size measures in the following definition.

**7.3.1.** Definition (Decision tree). Let $T = (V, E)$ be a tree, with root node $r \in V$. We refer to every vertex in $V$ that is not a leaf of the tree as an *internal node*, and we assume that all internal nodes have exactly two outgoing edges. We label every leaf $\ell$ with an output bit $b_\ell$, every internal node $v$ with an index $j_v \in [n]$, and for every internal node we label its outgoing edges by 0 and 1. The graph $T$ with all its labels is referred to as a *decision tree*.

1. The *decision tree depth* is defined to be the maximum length of a path from the root node to a leaf. We denote it by $\text{depth}(T)$.

2. The *decision tree size* is defined to be the number of internal nodes. We denote it by $\mathrm{size}(T)$.
3. The *decision tree rank* is defined to be the largest number of edge layers of a full binary tree that can be obtained by contracting vertices of the tree. We denote it by $\mathrm{rank}(T)$.
4. A *guessing scheme* is a set of edges, that for every internal node contains exactly one of its outgoing edges. We refer to these edges as the *guessed edges*. The *guessing number* of a particular scheme is the maximum number of non-guessed edges in any path going from the root node to a leaf. The *guessing complexity* is the minimum guessing number across all guessing schemes. ◀

In [LL16, page 4], Lin and Lin ask the question whether the guessing number[7] as defined above has some interpretation as a combinatorial measure studied in classical decision-tree complexity. Here, we answer that question in the affirmative, and proceed by proving that the guessing complexity of a decision tree, i.e., the guessing number of the optimal guessing scheme, is equal to its rank. The decision tree rank was indeed introduced earlier, in [EH89], in the context of learning theory.

We start by proving the following recursive characterization of the decision tree rank.

**7.3.2.** Lemma. *Let $T = (V, E)$ be a decision tree, $v \in V$ be a node of the decision tree, and let $T_v$ be the decision tree rooted at $v$, i.e., containing all the nodes and vertices that are either $v$ itself or below $v$. If $v$ is a leaf, then* $\mathrm{rank}(v) = 0$. *On the other hand, if $v$ is an internal node, we let its children be $v_L$ and $v_R$. Then, we have*

$$\mathrm{rank}(T_v) = \begin{cases} \max\{\mathrm{rank}(T_{v_L}), \mathrm{rank}(T_{v_R})\}, & \text{if } \mathrm{rank}(T_{v_L}) \neq \mathrm{rank}(T_{v_R}), \\ \mathrm{rank}(T_{v_L}) + 1, & \text{if } \mathrm{rank}(T_{v_L}) = \mathrm{rank}(T_{v_R}). \end{cases}$$

**Proof:**
If $v$ is a leaf, then $T_v$ is a single node, as such contains no edges and hence the rank is 0. Thus, it remains to consider the case where $v$ is an internal node. In that case, we observe that

$$\max\{\mathrm{rank}(T_{v_L}), \mathrm{rank}(T_{v_R})\} \leq \mathrm{rank}(T_v) \leq \max\{\mathrm{rank}(T_{v_L}), \mathrm{rank}(T_{v_R})\} + 1.$$

The left inequality follows from the fact that if we can find a full binary tree with $r$ edge layers below $v_L$, then it is by definition also below $v$, and similarly for $v_R$. The right inequality follows from the fact that if we can find a full binary tree with $r$ layers below $v$, then at least $r - 1$ of those layers will be below $v_L$ or $v_R$ (or both).

[7] The guessing number is $G$ in their notation.

Thus, it remains to show that $\text{rank}(T_v) > \max\{\text{rank}(T_{v_L}), \text{rank}(T_{v_R})\}$ if and only if $\text{rank}(T_{v_L}) = \text{rank}(T_{v_R})$. To that end, observe that if it so happens that $\text{rank}(T_v) > \max\{\text{rank}(T_{v_L}), \text{rank}(T_{v_R})\}$, then the maximum binary tree in $T$ contains $v$, which means that it contains two binary trees with $\text{rank}(T_v)-1$ edge layers below $v_L$ and $v_R$. Thus, indeed, we have $\text{rank}(T_{v_L}) = \text{rank}(T_v) - 1 = \text{rank}(T_{v_R})$. On the other hand, if $\text{rank}(T_{v_L}) = \text{rank}(T_{v_R})$, then the two binary trees below $v_L$ and $v_R$ combine to form a binary tree of size $\text{rank}(T_{v_L}) + 1$ below $v$. This completes the proof. □

The above characterization allows us to very succinctly define an optimal guessing scheme.

**7.3.3.** THEOREM (Guessing complexity and decision tree rank). *Let $T = (V, E)$ be a decision tree. Then the decision tree rank of $T$ and the guessing complexity of $T$ are equal.*

**Proof:**
We can directly observe that the guessing complexity of $T$ is at least the decision tree rank of $T$. Indeed, let $\text{rank}(T) = r$, and by the definition of rank we can obtain a full binary tree with $r$ edge layers by contracting nodes within $T$. Any guessing scheme of $T$ naturally induces a guessing scheme on this full binary tree with smaller or equal guessing number, and in a full binary tree there is always at least a path going from root to leaf that traverses exclusively non-guessed edges. Thus, the guessing complexity is at least $r$.

For the other inequality, we define a particular guessing scheme, and inductively show that its guessing number is exactly the rank of $T$. To that end, let $v$ be an internal node, with children $v_L$ and $v_R$. Suppose that $\text{rank}(T_{v_L}) \neq \text{rank}(T_{v_R})$, and without loss of generality assume that $\text{rank}(T_{v_L}) > \text{rank}(T_{v_R})$. Then, we label the edge going from $v$ to $v_L$ as the guessed edge. On the other hand, if $\text{rank}(T_{v_L}) = \text{rank}(T_{v_R})$, then we choose the guessed edge outgoing from $v$ arbitrarily.

We now prove that the guessing number of this particular guessing scheme is equal to the rank of $T$. The leaves provide the basis for induction, since the rank and guessing complexity of a decision tree containing only a single node are both trivially 0. Next, let $v$ be an internal node with children $v_L$ and $v_R$, and observe that the guessing scheme for $T$ naturally induces a guessing scheme on $T_v$, $T_{v_L}$ and $T_{v_R}$. Suppose that the guessing numbers of $T_{v_L}$ and $T_{v_R}$ are equal to their respective ranks. Then, if $\text{rank}(T_{v_L}) > \text{rank}(T_{v_R})$, we obtain that the guessing number of $T_v$ is the maximum of $\text{rank}(T_{v_L})$ and $\text{rank}(T_{v_R}) + 1$, which is simply $\text{rank}(T_{v_L})$. On the other hand, if $\text{rank}(T_{v_L}) = \text{rank}(T_{v_R}) = r$, then the guessing number of $T_v$ is $r+1$. By Lemma 7.3.2, we obtain that in both cases, the guessing number on $T_v$ is equal to $\text{rank}(T_v)$, completing the proof. □

### 7.3.3 Graph composition of a decision tree

Next, we show how a decision tree can be turned into a quantum algorithm using the graph composition of span programs. The core idea we employ next is to transform the decision tree $T = (V, E)$ into an $st$-connectivity graph $G = (V', E')$ with distinct source and sink nodes $s, t \in V'$. To that end, we let $G$ be the graph that is derived from $T$ by pruning all the leaves labeled 0 and contracting all the leaves labeled 1 into a single node labeled $t$. We let $s$ be the root of $T$. If an edge $e$ in the decision tree $T$ has label $b$ and is outgoing from a node $v$, then it is associated with the identity span program $\mathcal{P}^{(e)}$ that computes $x_{j_v}$ if $b = 1$, and $\neg x_{j_v}$ if $b = 0$, where $j_v$ is the element of the input that is being queried at node $v$. See Figure 7.3.2 for the resulting graph $G$ constructed from the decision tree $T$ showcased in Figure 7.3.1.

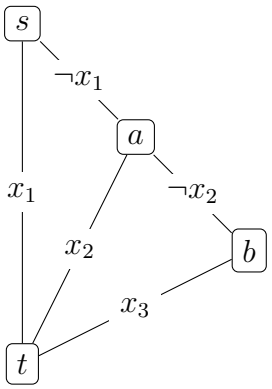



Figure 7.3.2: Illustration of the resulting graph $G$ constructed from the decision tree $T$ for the OR-function on three bits in Figure 7.3.1.

Now, we let $\mathcal{P}$ be the graph composition of $G$ with span programs $(\mathcal{P}^{(e)})_{e \in E'}$ and some strictly positive weights $(r_e)_{e \in E'}$ to be fixed later. We easily observe that it evaluates the same function $f : \{0,1\}^n \to \{0,1\}$ as the classical deterministic algorithm that traverses the decision tree $T$. Thus, we have indeed constructed a span program from a classical algorithm, which can be turned into a bounded-error quantum query algorithm using Algorithm 6.1.18.

In order to analyze the number of queries performed by this quantum algorithm, or equivalently the complexity of the composed span program, it remains to choose the weights in this composition. To that end, suppose that we choose the weights $(r_e)_{e \in E'}$. As in Definition 7.2.5, we will refer to the normalized weights by

$$r'_e = \frac{r_e}{R_{s,t}(G, (r_e)_{e \in E'})}.$$

We can now evaluate the positive and negative witness sizes according to Theorem 7.2.7. For a positive input $x \in \{0,1\}^n$, the classical algorithm traverses a unique path from the root node to a 1-labeled leaf in the decision tree $T$, and

as such there will be a unique path $P_x \subseteq E'$ that connects $s$ to $t$. The positive witness size, therefore, is the sum of the normalized weights along this path, i.e.,

$$w_+(x, \mathcal{P}) = \sum_{e \in P_x} r'_e. \tag{7.3.1}$$

On the other hand, suppose that we have a negative input $x \in \{0,1\}^n$. Since the graph $G$ is planar, we can think of a negative witness as a flow from $s^\dagger$ to $t^\dagger$ in the dual graph $G^\dagger$. To exhibit such a flow, consider the path $P_x$ in $G$ traversed by the classical algorithm, and let $V_x \subseteq V$ be the set of nodes that it visits. Next, let $\overline{P}_x$ be the set of edges that are outgoing from $V_x$, but are not in $P_x$. Then, we observe that the dual edges $\{e^\dagger : e \in \overline{P}_x\}$ form a path from $s^\dagger$ to $t^\dagger$ in the dual graph. See also the visualization in Figure 7.3.3. As such, we find that

$$w_-(x, \mathcal{P}) \leq \sum_{e \in \overline{P}_x} \frac{1}{r'_e}. \tag{7.3.2}$$

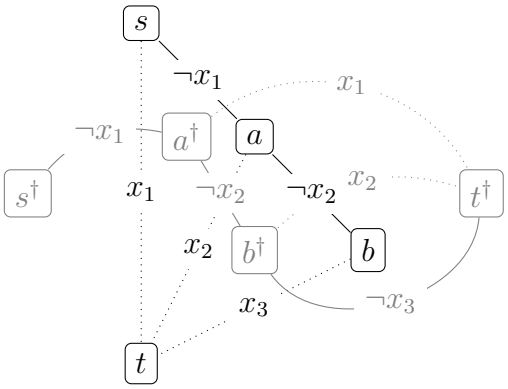



Figure 7.3.3: Visualization of a negative witness in the span program compiled from a decision tree. For each internal vertex visited by the classical algorithm, take the alternative edges that were not traversed by it. The duals of these edges forms an $s^\dagger t^\dagger$-path in the dual graph.

It is important to remark that this is truly an *upper bound* on the negative witness size, since we have only exhibited one possible negative witness and we have no guarantee that it is the optimal one. Indeed, one can construct cases where the inequality in Equation (7.3.2) is not tight.

We can interpret these witnesses directly in terms of the decision tree $T$. To that end, suppose that we assign a strictly positive weight $r_e$ to every edge in $T$. These weights naturally induce normalized weights $(r'_e)_{e \in E'}$ on the derived graph $G$, and we refer to the graph composition $\mathcal{P}$ of $G$ with weights $(r'_e)_{e \in E'}$ as the *span program compiled from $T$ with edge weights* $(r_e)_{e \in E}$. We let $\mathrm{Path}(T)$ be the set of paths from the root to a leaf in $T$, and for every path $P \in \mathrm{Path}(T)$, we

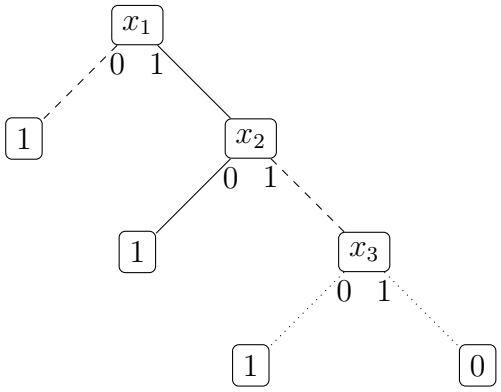



Figure 7.3.4: Visualization of the path $P$ and its complement $\overline{P}$. If the solid lines represent the path $P$, then its complement $\overline{P}$ is the set of dashed edges, i.e., the edges that are outgoing from the nodes traversed by $P$, but not traversed by $P$ itself. The dotted edges are neither part of $P$ nor $\overline{P}$ in this example.

let $\overline{P}$ be the set of edges that are outgoing from a node traversed by $P$, but *not* traversed by $P$ itself. See Figure 7.3.4 for an example.

Now, we observe that

$$W_+(\mathcal{P}) \leq \max_{P \in \mathrm{Path}(T)} \sum_{e \in P} r'_e, \qquad \text{and} \qquad W_-(\mathcal{P}) \leq \max_{P \in \mathrm{Path}(T)} \sum_{e \in \overline{P}} \frac{1}{r'_e},$$

since for every positive input $x$, the path $P_x$ in $G$ is also a path $P$ in $T$, and for every negative input $x$, the edge set $\overline{P}_x$ in $G$ is a subset of a path complement $\overline{P}$ in $T$. It follows that

$$C(\mathcal{P})^2 \leq \max_{P \in \mathrm{Path}(T)} \sum_{e \in P} r_e \cdot \max_{P \in \mathrm{Path}(T)} \sum_{e \in \overline{P}} \frac{1}{r_e}. \tag{7.3.3}$$

Note that we dropped the normalization factor $R_{s,t}(G, (r_e)_{e \in E'})$ from the $r_e$'s in the above equation, since they cancel when we multiply the positive and the negative witness sizes together.

Interestingly, the right-hand side of Equation (7.3.3) reduces to the upper bound on the span program complexity given in [BT20], for a particular assignment of the weights $(r_e)_{e \in E}$. In [CMP22], we achieve the same expression as the right-hand side of Equation (7.3.3), and we show how it is a generalization of [BT20]. We also show in [CMP22] how to choose the weights $(r_e)_{e \in E}$ so that the right-hand side of Equation (7.3.3) is minimized, which we proceed to showcase in the next subsection.

### 7.3.4 Optimal weight assignment

The task that remains is to choose the weights $(r_e)_{e \in E}$ in such a way that the right-hand side of Equation (7.3.3) is minimized. To that end, we give a recursive

definition of these weights. For all $v \in V$, let $T_v = (V_v, E_v)$ be the decision tree obtained from starting at node $v$ in $T$, rather than at the root node. Let $C_v^2$ be the optimal value of the right-hand side of Equation (7.3.3) when applied to $T_v$, i.e.,

$$C_v^2 = \min_{(r_e)_{e\in E_v}} \left[ \max_{P\in \mathrm{Path}(T_v)} \sum_{e\in P} r_e \cdot \max_{P\in \mathrm{Path}(T_v)} \sum_{e\in \overline{P}} \frac{1}{r_e} \right].$$

It follows that $C_v = 0$ whenever $v$ is a leaf, because there are no paths in the graph $T_v$.

Now, let $v \in V$ be an internal node. It has two children, $v_L$ and $v_R$, connected to $v$ by edges $e_L$ and $e_R$. We claim that

$$C_v = \frac{C_{v_L} + C_{v_R} + \sqrt{(C_{v_L} - C_{v_R})^2 + 4}}{2},$$

and that the optimal weights for the edges $e_L$ and $e_R$ are

$$r_{e_L} = \frac{C_{v_R} - C_{v_L} + \sqrt{(C_{v_L} - C_{v_R})^2 + 4}}{2} \qquad \text{and} \tag{7.3.4a}$$

$$r_{e_R} = \frac{C_{v_L} - C_{v_R} + \sqrt{(C_{v_L} - C_{v_R})^2 + 4}}{2}. \tag{7.3.4b}$$

We prove these claims in the following sequence of lemmas.

**7.3.4.** Lemma. *Let $v \in V$ be an internal node in $T$ and let $v_L$ and $v_R$ be its children. Then,*

$$C_v \leq \frac{C_{v_L} + C_{v_R} + \sqrt{(C_{v_L} - C_{v_R})^2 + 4}}{2}.$$

**Proof:**
Let $(r_e)_{e\in E_{v_L}}$ and $(r_e)_{e\in E_{v_R}}$ be optimal edge weights in $T_{v_L} = (V_{v_L}, E_{v_L})$ and $T_{v_R} = (V_{v_R}, E_{v_R})$, i.e., assignments of $(r_e)_{e\in E_{v_L}}$ and $(r_e)_{e\in E_{v_R}}$ that minimize the right-hand side of Equation (7.3.3) applied to the decision trees $T_{v_L}$ and $T_{v_R}$, respectively. Then, we define

$$C_{v_L}^+ = \max_{P\in \mathrm{Path}(T_{v_L})} \sum_{e\in P} r_e, \qquad C_{v_L}^- = \max_{P\in \mathrm{Path}(T_{v_L})} \sum_{e\in \overline{P}} \frac{1}{r_e},$$

$$C_{v_R}^+ = \max_{P\in \mathrm{Path}(T_{v_R})} \sum_{e\in P} r_e, \qquad C_{v_R}^- = \max_{P\in \mathrm{Path}(T_{v_R})} \sum_{e\in \overline{P}} \frac{1}{r_e}.$$

Next, we define

$$\forall e \in E_{v_L}, \quad r'_e = \sqrt{\frac{C_{v_L}^-}{C_{v_L}^+}} r_e, \qquad \text{and} \qquad \forall e \in E_{v_R}, \quad r'_e = \sqrt{\frac{C_{v_R}^-}{C_{v_R}^+}} r_e,$$

and we let

$$(C^+_{v_L})' = \max_{P\in\mathrm{Path}(T_{v_L})} \sum_{e\in P} r'_e, \qquad (C^-_{v_L})' = \max_{P\in\mathrm{Path}(T_{v_L})} \sum_{e\in \overline{P}} \frac{1}{r'_e},$$
$$(C^+_{v_R})' = \max_{P\in\mathrm{Path}(T_{v_R})} \sum_{e\in P} r'_e, \qquad (C^-_{v_R})' = \max_{P\in\mathrm{Path}(T_{v_R})} \sum_{e\in \overline{P}} \frac{1}{r'_e},$$

to obtain that

$$(C^+_{v_L})' = \sqrt{C^+_{v_L}C^-_{v_L}} = C_{v_L} = \sqrt{C^+_{v_L}C^-_{v_L}} = (C^-_{v_L})', \qquad \text{and}$$
$$(C^+_{v_R})' = \sqrt{C^+_{v_R}C^-_{v_R}} = C_{v_R} = \sqrt{C^+_{v_R}C^-_{v_R}} = (C^-_{v_R})',$$

where we used that the $r_e$'s were the optimal choice to begin with, and hence $C^+_{v_L}C^-_{v_L} = C^2_{v_L}$, and similarly for $v_R$. Now, we let $r_{e_L}$ and $r_{e_R}$ as in Equation (7.3.4), and we obtain that

$$\begin{aligned} C^+_v &:= \max_{P\in\mathrm{Path}(T_v)} \sum_{e\in P} r'_e = \max\left\{r_{e_L} + (C^+_{v_L})', r_{e_R} + (C^+_{v_R})'\right\} \\ &= \max\left\{r_{e_L} + C_{v_L}, r_{e_R} + C_{v_R}\right\}, \end{aligned}$$

and similarly,

$$\begin{aligned} C^-_v &:= \max_{P\in\mathrm{Path}(T_v)} \sum_{e\in \overline{P}} \frac{1}{r'_e} = \max\left\{\frac{1}{r_{e_R}} + (C^-_{v_L})', \frac{1}{r_{e_L}} + (C^-_{v_R})'\right\} \\ &= \max\left\{\frac{1}{r_{e_R}} + C_{v_L}, \frac{1}{r_{e_L}} + C_{v_R}\right\}. \end{aligned}$$

Thus, we obtain that

$$\begin{aligned} C_v &\le \sqrt{C^+_v C^-_v} \le \max\{C^+_v, C^-_v\} \\ &= \max\left\{r_{e_L} + C_{v_L}, r_{e_R} + C_{v_R}, \frac{1}{r_{e_R}} + C_{v_L}, \frac{1}{r_{e_L}} + C_{v_R}\right\}. \end{aligned}$$

It follows easily that the first two expressions inside the maximum evaluate to the desired expression. To show that the last two expressions in the maximum also simplify to the same expression, it remains to show that $r_{e_L}r_{e_R} = 1$. To that end, we use the formula $(a+b)(a-b) = a^2 - b^2$ with $a = \sqrt{(C_{v_L} - C_{v_R})^2 + 4}$ and $b = C_{v_R} - C_{v_L}$, to deduce that

$$\begin{aligned} 4r_{e_L}r_{e_R} &= (C_{v_R} - C_{v_L} + \sqrt{(C_{v_L} - C_{v_R})^2 + 4})(C_{v_L} - C_{v_R} + \sqrt{(C_{v_L} - C_{v_R})^2 + 4}) \\ &= (C_{v_L} - C_{v_R})^2 + 4 - (C_{v_R} - C_{v_L})^2 = 4. \end{aligned}$$

This completes the proof. □

Now, we proceed with the opposite inequality.

**7.3.5.** LEMMA. *Let $v \in V$ be an internal node in $T$ and let $v_L$ and $v_R$ be its children. Then,*

$$C_v \geq \frac{C_{v_L} + C_{v_R} + \sqrt{(C_{v_L} - C_{v_R})^2 + 4}}{2}.$$

**Proof:**
Let $(r_e)_{e \in E_v}$ be any weight assignment to the edges in $T_v$. Then, we define

$$C^+_{v_L} = \max_{P \in \mathrm{Path}(T_{v_L})} \sum_{e \in P} r_e, \qquad C^-_{v_L} = \max_{P \in \mathrm{Path}(T_{v_L})} \sum_{e \in \overline{P}} \frac{1}{r_e},$$

$$C^+_{v_R} = \max_{P \in \mathrm{Path}(T_{v_R})} \sum_{e \in P} r_e, \qquad C^-_{v_R} = \max_{P \in \mathrm{Path}(T_{v_R})} \sum_{e \in \overline{P}} \frac{1}{r_e},$$

$$C^+_{v} = \max_{P \in \mathrm{Path}(T_{v})} \sum_{e \in P} r_e, \qquad C^-_{v} = \max_{P \in \mathrm{Path}(T_{v})} \sum_{e \in \overline{P}} \frac{1}{r_e}.$$

We let $e_L$ and $e_R$ be the edges connecting $v$ with $v_L$ and $v_R$, respectively. Then,

$$C^+_v = \max\{r_{e_L} + C^+_{v_L}, r_{e_R} + C^+_{v_R}\}, \quad \text{and} \quad C^-_v = \max\left\{\frac{1}{r_{e_R}} + C^-_{v_L}, \frac{1}{r_{e_L}} + C^-_{v_R}\right\}.$$

We let $\gamma = \sqrt{C^+_v / C^-_v}$. Then, we have

$$\sqrt{C^+_v C^-_v} = \frac{C^+_v}{\gamma} = \gamma C^-_v,$$

and hence we also find that

$$\sqrt{C^+_v C^-_v} = \frac{1}{2}\left[\frac{C^+_v}{\gamma} + \gamma C^-_v\right].$$

Next, let $\delta \in [0,1]$, and we observe that for for all real $a$ and $b$, we have $\max\{a,b\} \geq \delta a + (1-\delta)b$. Thus,

$$\begin{aligned}
&\sqrt{C^+_v C^-_v} \\
&\quad = \frac{1}{2}\left[\max\left\{\frac{r_{e_L}}{\gamma} + \frac{C^+_{v_L}}{\gamma}, \frac{r_{e_R}}{\gamma} + \frac{C^+_{v_R}}{\gamma}\right\} + \max\left\{\frac{\gamma}{r_{e_R}} + \gamma C^-_{v_L}, \frac{\gamma}{r_{e_L}} + \gamma C^-_{v_R}\right\}\right] \\
&\quad \geq \frac{1}{2}\left[\delta\left(\frac{r_{e_L}}{\gamma} + \frac{C^+_{v_L}}{\gamma}\right) + (1-\delta)\left(\frac{r_{e_R}}{\gamma} + \frac{C^+_{v_R}}{\gamma}\right)\right. \\
&\qquad \left. + \delta\left(\frac{\gamma}{r_{e_R}} + \gamma C^-_{v_L}\right) + (1-\delta)\left(\frac{\gamma}{r_{e_L}} + \gamma C^-_{v_R}\right)\right] \\
&\quad = \frac{1}{2}\left[\delta\left(\frac{C^+_{v_L}}{\gamma} + \gamma C^-_{v_L}\right) + (1-\delta)\left(\frac{C^+_{v_R}}{\gamma} + \gamma C^-_{v_R}\right)\right. \\
&\qquad \left. + \left(\delta\frac{r_{e_L}}{\gamma} + (1-\delta)\frac{\gamma}{r_{e_L}}\right) + \left(\delta\frac{\gamma}{r_{e_R}} + (1-\delta)\frac{r_{e_R}}{\gamma}\right)\right] \\
&\quad \geq \delta\sqrt{C^+_{v_L} C^-_{v_L}} + (1-\delta)\sqrt{C^+_{v_R} C^-_{v_R}} + 2\sqrt{\delta(1-\delta)}.
\end{aligned}$$

where we used that for any $a, b \geq 0$, $(a+b)/2 \geq \sqrt{ab}$. Now, we observe that $\sqrt{C^+_{v_L} C^-_{v_L}} \geq C_{v_L}$, and similarly for the right part. Thus, we obtain that

$$\sqrt{C^+_v C^-_v} \geq \delta C_{v_L} + (1-\delta) C_{v_R} + 2\sqrt{\delta(1-\delta)}.$$

We now write $\Delta = C_{v_L} - C_{v_R}$, and we define

$$\delta = \frac{1}{2} + \frac{\Delta}{2\sqrt{\Delta^2+4}}.$$

Then indeed $\delta \in [0,1]$, and

$$\begin{aligned}
\sqrt{C^+_v C^-_v} &\geq \frac{C_{v_L} + C_{v_R}}{2} + \frac{\Delta^2}{2\sqrt{\Delta^2+4}} + 2\sqrt{\frac{1}{4} - \frac{\Delta^2}{4(\Delta^2+4)}} \\
&= \frac{C_{v_L} + C_{v_R}}{2} + \frac{\Delta^2}{2\sqrt{\Delta^2+4}} + \sqrt{\frac{4}{\Delta^2+4}} \\
&= \frac{C_{v_L} + C_{v_R}}{2} + \frac{\Delta^2+4}{2\sqrt{\Delta^2+4}} = \frac{C_{v_L} + C_{v_R} + \sqrt{\Delta^2+4}}{2}.
\end{aligned}$$

Since this holds for every weight assignment $(r_e)_{v \in E_v}$, it also holds for the optimal one in particular. But for the optimal weight assignment we have that $C_v = \sqrt{C^+_v C^-_v}$ by definition, and hence we obtain the inequality from the lemma statement. This completes the proof. □

We have now found a recursive characterization of the optimal upper bound on the complexity of a span program compiled from a decision tree. We summarize this recursive result in the corollary below.

**7.3.6.** Corollary. *Let $T = (V, E)$ be a decision tree, and let $v \in V$. If $v$ is a leaf, then $C_v = 0$. If $v \in V$ is an internal node of $T$ with children $v_L$ and $v_R$, then*

$$C_v = \frac{C_{v_L} + C_{v_R} + \sqrt{(C_{v_L} - C_{v_R})^2 + 4}}{2}.$$

This gives us a recursive way to compute the complexity of the span program $\mathcal{P}$ that is compiled from the decision tree $T$ with the optimal weight assignment. We can use this recursive computation to prove an upper bound on the complexity in terms of the size of the decision tree. This is the objective of the following lemma.

**7.3.7.** Lemma. *Let $\mathcal{P}$ be the span program compiled from the decision tree $T$ with the optimal weight assignment. Then $C(\mathcal{P}) \leq \sqrt{2\,\mathrm{size}(T)}$.*

**Proof:**
We give a proof by induction. If $T$ is a single node, then it computes a constant function, and hence indeed $C(\mathcal{P}) = 0 = \sqrt{2 \cdot 0} = \sqrt{2 \cdot \text{size}(T)}$. This provides the basis for induction.

Next, let $v \in V$ be an internal node of $T$, and let $v_L$ and $v_R$ be its children. Let $n_L = \text{size}(T_{v_L})$ and $n_R = \text{size}(T_{v_R})$ be the number of internal nodes in the trees rooted at $v_L$ and $v_R$, respectively, and suppose that $C_{v_L} \leq \sqrt{2n_L}$, and $C_{v_R} \leq \sqrt{2n_R}$. Then, using Corollary 7.3.6 and the formula $(a+b)^2 \leq 2(a^2+b^2)$ for all $a, b \geq 0$, we obtain

$$\begin{aligned} C_v^2 &\leq \frac{2(C_{v_L} + C_{v_R})^2 + 2(C_{v_L} - C_{v_R})^2 + 8}{4} = C_{v_L}^2 + C_{v_R}^2 + 2 \\ &\leq 2(n_L + n_R + 1) = 2\,\text{size}(T_v). \end{aligned}$$

This provides the induction step, and thus the proof is complete. □

Thus, we have proved that if we compile a span program $\mathcal{P}$ from a decision tree $T$ with the optimal weight assignment, then the span program complexity $C(\mathcal{P})$ is at most the square root of twice the number of internal nodes in the decision tree. We can combine this observation with the result from [LL16, Theorem 8], to obtain the following theorem.

**7.3.8.** THEOREM. *Let $T$ be a decision tree that computes a boolean function $f : \{0,1\}^n \to \{0,1\}$. Then,*

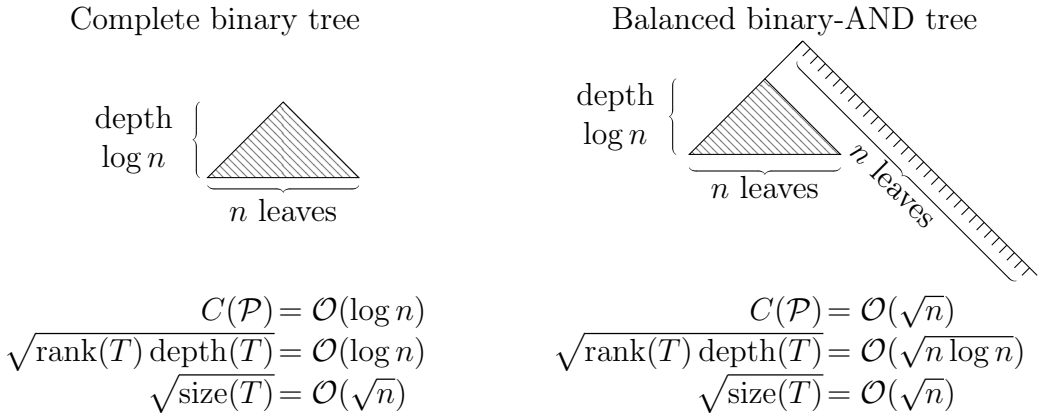
1. $D(f) \leq \text{depth}(T)$.
2. $Q(f) = \mathcal{O}(\min\{\sqrt{\text{rank}(T)\,\text{depth}(T)}, \sqrt{\text{size}(T)}\})$.

**Proof:**
Claim 1 is trivial. The first part of the minimum in claim 2 is proved in [LL16, Theorem 8], and reproduced in [BT20, Section 3][8], and the second part is proved in Lemma 7.3.7. This completes the proof. □

## 7.3.5 Discussion

From the statement of Theorem 7.3.8, there are a few follow-up questions that naturally come to mind. First, one might wonder whether both terms in the minimum that appears in claim 2 are necessary, or whether one is stronger than the other. In Figure 7.3.5, we present two different examples of decision trees, one in which $\text{depth}(T)\,\text{rank}(T) \ll \text{size}(T)$, and one where the opposite is true. Thus, we conclude that the two parts of the minimum in claim 2 are in general incomparable.

[8]The results in [LL16] and [BT20] are phrased using the guessing complexity, rather than the decision tree rank. However, since we proved in Theorem 7.3.3 that these two are equal, we phrase the result here in terms of decision tree rank.




$$C(\mathcal{P}) = \mathcal{O}(\log n) \qquad C(\mathcal{P}) = \mathcal{O}(\sqrt{n})$$
$$\sqrt{\mathrm{rank}(T)\,\mathrm{depth}(T)} = \mathcal{O}(\log n) \qquad \sqrt{\mathrm{rank}(T)\,\mathrm{depth}(T)} = \mathcal{O}(\sqrt{n \log n})$$
$$\sqrt{\mathrm{size}(T)} = \mathcal{O}(\sqrt{n}) \qquad \sqrt{\mathrm{size}(T)} = \mathcal{O}(\sqrt{n})$$

Figure 7.3.5: Examples showing separations between the two bounds derived in claim 2 of Theorem 7.3.8. The shaded regions represent complete binary trees. In the left example $\mathrm{rank}(T)\,\mathrm{depth}(T) \ll \mathrm{size}(T)$, and in the right example $\mathrm{size}(T) \ll \mathrm{rank}(T)\,\mathrm{depth}(T)$.

Another question that comes to mind is if the results presented here can be generalized to randomized classical query algorithms, i.e., can we also take a randomized classical query algorithm with bounded error and turn it into a bounded-error quantum query algorithm? The answer is yes – since in the classical randomized setting, one can without loss of generality assume that a sufficient number of random bits are generated before the execution of the algorithm starts, we can think of a randomized classical algorithm with failure probability $\delta_1 > 0$ as a family of decision trees, where the specific instance is chosen according to the random bits being generated before the execution of the algorithm. If one now takes such a randomly generated decision tree and turns it into a quantum query algorithm with bounded-error $\delta_2 > 0$, then this whole procedure is a quantum query algorithm with failure probability at most $\delta_1 + \delta_2$. Standard failure probability suppression techniques can ensure that $\delta_1$ and $\delta_2$ are small enough such that their sum is at most $1/3$. More details are supplied in [BT20; CMP22].

There are many interesting directions of future research to pursue from this result. For instance, before arriving at Equation (7.3.3), we merely used an upper bound on the negative witness size. It would be interesting to figure out if we can more tightly characterize the negative witness size, in terms of quantities that are derived directly on the decision tree itself.

Finally, the result in [BT20] also holds in the setting where the input and output of the function is arbitrary, rather than boolean. It would be interesting to figure out whether we can interpret the constructions in these more general settings as *st*-connectivity graph compositions as well.

# Chapter 8

# Approximate span programs

In the previous chapter, we provided a construction to turn *classical* query algorithms into span programs, which subsequently can be turned into *quantum* query algorithms with bounded error. We also showed that this black-box approach is capable of achieving quantum speed-ups in terms of the number of queries the algorithm performs.

A natural analogue of this question is whether it is possible to turn a quantum query algorithm into a span program as well. The difficulty here is that the most natural class of quantum algorithms to consider consists of those that have bounded error, i.e., that succeed with a probability that is lower bounded by 2/3. It turns out that the ability to handle such imperfections requires a modification of the span program framework. To that end, we introduce the concept of approximate span programs.

The presentation in this chapter is mostly based on [IJ19], but it has some subtle but crucial differences. We first introduce the concept of approximate span programs in Section 8.1. After that, we give an algorithm to evaluate such approximate span programs, in Section 8.2, which attains a polynomial improvement over the algorithm from [IJ19]. Finally, we discuss how approximate span programs are equivalent to quantum query algorithms with bounded error, in Section 8.3.

## 8.1 Definition and basic properties

Recall from Figure 6.1.2 that we label an input $x \in \mathcal{D}$ as a positive input if and only if there is no component of $|w_0\rangle$ in $\mathcal{K}^\perp \cap \mathcal{H}(x)^\perp$. This is a very rigid cut-off – pictorially, even if $|w_0\rangle$ is hovering a tiny bit above the ground plane in Figure 6.1.2, we dub it a negative input (with a very high negative witness size).

The idea of approximate span programs is to move the decision rule for inputs being positive a little bit in favor of those inputs for which $|w_0\rangle$ is very close to the ground plane. We formalize this idea in the following definition.

**8.1.1.** Definition (Approximate span program).
Let $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ be a span program on a domain $\mathcal{D}$. Let $\delta \in (0, 1]$ be an approximation parameter. We say that $\mathcal{P}$ $\delta$-approximates the function $f_\delta : \mathcal{D} \to \{0, 1\}$, defined as

$$f_\delta(x) = \begin{cases} 0, & \text{if } \left\| \Pi_{\mathcal{K}^\perp \cap \mathcal{H}(x)^\perp} |w_0\rangle \right\|^2 \geq \delta, \\ 1, & \text{otherwise.} \end{cases}$$

For every $x \in \mathcal{D}$, we say that $x$ is a $\delta$-approximate positive input whenever $f_\delta(x) = 1$, and we say that it is a $\delta$-approximate negative input whenever $f_\delta(x) = 0$.

For every $x \in \mathcal{D}$, we define the approximate positive witness size as

$$\widetilde{w}_+(x, \mathcal{P}) = \min\{ \||w\rangle\|^2 : |w\rangle \in \mathcal{H}(x), |w\rangle - \Pi_{\mathcal{K}+\mathcal{H}(x)} |w_0\rangle \in \mathcal{K} \}, \tag{8.1.1}$$

and we say that the vectors $|w\rangle$ that satisfy the constraints of the set on the right-hand side are approximate positive witnesses for $x$ in $\mathcal{P}$.

Finally, we define the quantities

$$\widetilde{W}_-(\mathcal{P}, \delta) = \max_{x \in f_\delta^{-1}(0)} w_-(x, \mathcal{P}), \qquad \widetilde{W}_+(\mathcal{P}, \delta) = \max_{x \in f_\delta^{-1}(1)} \widetilde{w}_+(x, \mathcal{P}),$$

and

$$\lambda(\mathcal{P}, \delta) = \max_{x \in f_\delta^{-1}(1)} \frac{1}{\delta w_-(x, \mathcal{P})}. \qquad \blacktriangleleft$$

In Figure 8.1.1, we develop some intuition for the objects we just defined. An input is dubbed positive if the squared distance between $|w_0\rangle$ and the ground plane, i.e., the size of the vertical component of $|w_0\rangle$ is smaller than $\delta$. We also define approximate positive witnesses to be vectors $|w\rangle$ that are in the ground plane, and that can be reached from the projection of $|w_0\rangle$ onto the ground plane, i.e., $|w_0'\rangle = \Pi_{\mathcal{K}+\mathcal{H}(x)} |w_0\rangle$, by adding an element from $\mathcal{K}$.

Even though the definition of approximate span programs we employ here for the most part resembles the definition of *negatively approximating span programs* in [IJ19], there is a slight difference in the definition of the approximate positive witnesses. In our notation, the definition of the approximate positive witness size used in [IJ19] is

$$\overline{w}_+(x, \mathcal{P}) = \min\{ \||w\rangle\|^2 : |w\rangle - |w_0\rangle \in \mathcal{K} \wedge |w\rangle - \Pi_{\mathcal{K}^\perp \cap \mathcal{H}(x)^\perp} |w_0\rangle \in \mathcal{H}(x) \}.$$

The reason that we use a different definition, i.e., Equation (8.1.1) instead, is because it features a more convenient operational interpretation, which we derive later in Lemma 8.1.3. From Figure 8.1.1, we observe directly that $\widetilde{w}_+(x, \mathcal{P}) \leq \overline{w}_+(x, \mathcal{P})$, since the approximate positive witnesses according to Definition 8.1.1 are projections onto the ground plane of those according to [IJ19].

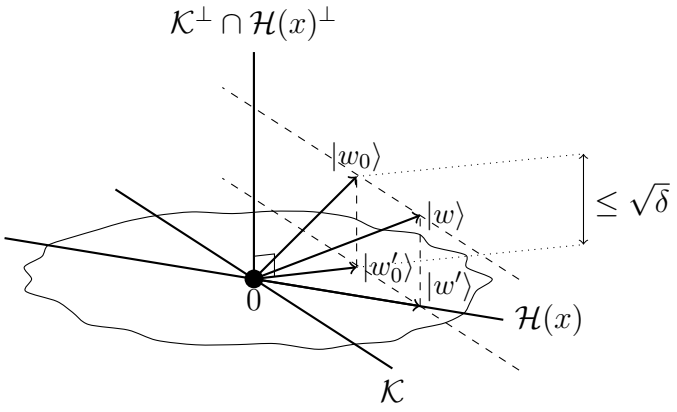



Figure 8.1.1: Illustration of a positive input in an approximate span program. $|w_0'\rangle = \Pi_{\mathcal{K}+\mathcal{H}(x)} |w_0\rangle$ is the projection of $|w_0\rangle$ onto the ground plane. The vector $|w\rangle$ is an approximate positive witness according to the definition in [IJ19, Definition 2.4]. The vector $|w'\rangle$ is an approximate positive witness according to the definition we use in this document, i.e., Definition 8.1.1.

Another thing that is important to understand is the role of $\lambda(\mathcal{P}, \delta)$. To that end, suppose that we have a span program $\mathcal{P}$ with approximation parameter $\delta \in (0, 1]$. If $x$ is a $\delta$-approximate negative input for $\mathcal{P}$, then we immediately find from Definition 8.1.1 that $1/w_-(x, \mathcal{P}) = \left\| \Pi_{\mathcal{K}^\perp \cap \mathcal{H}(x)^\perp} |w_0\rangle \right\|^2 \geq \delta$. On the other hand, if $x$ is a $\delta$-approximate positive input for $\mathcal{P}$, then we find that $w_-(x, \mathcal{P}) \geq 1/(\delta\lambda(\mathcal{P}, \delta))$, or equivalently $1/w_-(x, \mathcal{P}) \leq \delta\lambda(\mathcal{P}, \delta)$. Thus, we observe that $\lambda(\mathcal{P}, \delta) \in [0, 1)$ defines a "gap" between positive and negative inputs, and this gap vanishes in the limit where $\lambda(\mathcal{P}, \delta) \uparrow 1$.

As in the exact case in Theorem 6.1.13, we can develop an operational interpretation of the approximate positive witness size. This is the objective of the next lemma.

**8.1.2.** Lemma. *Let $\mathcal{P}$ be a span program on $\mathcal{D}$. Let $x \in \mathcal{D}$, and let $\Phi$ be the ideal phase variable for $x$ in $\mathcal{P}$. Then, the corresponding approximate positive witness size is*

$$\widetilde{w}_+(x, \mathcal{P}) = \mathbb{P}\left[\Phi \neq 0\right] \mathbb{E}\left[ \frac{1}{\sin^2(\pi\Phi)} \middle| \Phi \neq 0 \right].$$

**Proof:**
The proof presented here is very similar to the proof of Theorem 6.1.13. Let $R_0, \ldots, R_k$ and $\theta_0, \ldots, \theta_k$ be as in Jordan's lemma, i.e., Lemma 6.1.11. We decompose $|w_0\rangle$ into the subspaces $R_j$, i.e., for all $j \in [k]_0$, we write $|w_0^{(j)}\rangle = \Pi_{R_j} |w_0\rangle$. Now, we observe that $(\mathcal{K} + \mathcal{H}(x))^\perp = \mathcal{K}^\perp \cap \mathcal{H}(x)^\perp = \mathcal{K}^\perp \cap R_0$, and thus

$$\Pi_{\mathcal{K}+\mathcal{H}(x)} |w_0\rangle = |w_0\rangle - |w_0^{(0)}\rangle = \sum_{j=1}^{k} |w_0^{(j)}\rangle.$$

Hence, the approximate positive witnesses are exactly the vectors $|w\rangle \in \mathcal{H}$ of the form

$$|w\rangle = |w^{(0)}\rangle + \sum_{j=1}^{k} |w^{(j)}\rangle,$$

where $|w^{(0)}\rangle \in \mathcal{K} \cap \mathcal{H}(x)$, and for all $j \in [k]$ we have that $|w^{(j)}\rangle$ is the unique vector in $R_j$ such that $\langle w_0^{(j)}|w^{(j)}\rangle = 1$. The size of the corresponding vector $|w\rangle$ is expressed as

$$\||w\rangle\|^2 = \||w^{(0)}\rangle\|^2 + \sum_{j=1}^{k} \||w^{(j)}\rangle\|^2,$$

which is minimized when we choose $|w^{(0)}\rangle = 0$. In that case, using elementary geometry and the visualization in Figure 6.1.4, the witness size can be rewritten to

$$\begin{aligned}\widetilde{w}_+(x,\mathcal{P}) = \sum_{j=1}^{k} \||w^{(j)}\rangle\|^2 = \sum_{j=1}^{k} \frac{\||w_0^{(j)}\rangle\|^2}{\sin^2(\theta_j/2)} = \sum_{\phi\in\mathrm{supp}(\Phi)\setminus\{0\}} \frac{\mathbb{P}[\Phi=\phi]}{\sin^2(\pi\phi)} \\ = \mathbb{P}\left[\Phi \neq 0\right] \mathbb{E}\left[\frac{1}{\sin^2(\pi\Phi)}\middle|\Phi \neq 0\right].\end{aligned}$$

This completes the proof. □

Similar to Theorem 6.1.14, the above result can be recast into a finite-precision result on the probability of obtaining 0 after a run of phase estimation with a finite number of bits of precision, as is shown in the next lemma.

**8.1.3.** Lemma. *Let $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ be a span program on $\mathcal{D}$, and let $x \in \mathcal{D}$. Suppose we run phase estimation with $k \in \mathbb{N}$ bits of precision, on the initial state $|w_0\rangle$, with span program unitary $U(x,\mathcal{P})$, and denote the output by $\Phi_k$. Then,*

$$\begin{aligned}\mathbb{P}\left[\Phi=0\right] \leq \mathbb{P}\left[\Phi_k=0\right] \leq \mathbb{P}\left[\Phi=0\right] + \mathbb{P}\left[\Phi\neq 0\right]\cdot\frac{1}{2^{2k}}\mathbb{E}\left[\frac{1}{\sin^2(\pi\Phi)}\middle|\Phi\neq 0\right] \\ = \mathbb{P}\left[\Phi=0\right] + \frac{\widetilde{w}_+(x,\mathcal{P})}{2^{2k}}.\end{aligned}$$

**Proof:**
We follow the proof of Theorem 6.1.14. The left-most inequality is already proved there, so it remains to prove the upper bound on $\mathbb{P}[\Phi_k = 0]$. To that end, recall from Equation (6.1.4) that we can write

$$\mathbb{P}[\Phi_k = 0] = \mathbb{E}\left[\frac{\sin^2(\pi 2^k \Phi)}{2^{2k}\sin^2(\pi\Phi)}\right],$$

where we take limits if the expression is ill-defined due to the denominator being equal to 0. Indeed, if $\Phi = 0$, then the expression within the expectation evaluates to 1, and thus we obtain

$$\mathbb{P}[\Phi_k = 0] = \mathbb{P}[\Phi = 0] + \mathbb{P}[\Phi \neq 0]\mathbb{E}\left[\frac{\sin^2(\pi 2^k \Phi)}{2^{2k}\sin^2(\pi\Phi)}\middle|\Phi \neq 0\right].$$

Finally, we use that $\sin^2(\pi 2^k\Phi) \leq 1$, and plug in the formula for $\widetilde{w}_+(x,\mathcal{P})$ from Lemma 8.1.2. This completes the proof. □

## 8.2 Approximate span program algorithm

With the observations from the previous subsection in mind, we would like to develop a quantum algorithm that computes the function $f_\delta$ that is $\delta$-approximated by a span program $\mathcal{P}$. Such an algorithm is constructed in [IJ19], and we restate the relevant theorem here.

**8.2.1.** Theorem ([IJ19, Corollary 3.7]: Approximate span program algorithm). *Let $\mathcal{P}$ be a span program and $\delta \in (0,1]$. Let $f_\delta : \mathcal{D} \to \{0,1\}$ be the function it $\delta$-approximates. Then, there exists a quantum algorithm that computes $f_\delta$, which depends only on the input $x \in \mathcal{D}$ through calls to the span program operator $U(x,\mathcal{P})$, and performs this operation a number of times that scales as*

$$\mathcal{O}\left(\frac{\sqrt{\widetilde{W}_+(\mathcal{P},\delta)\widetilde{W}_-(\mathcal{P},\delta)}}{(1-\lambda(\mathcal{P},\delta))^{3/2}}\log\frac{1}{1-\lambda(\mathcal{P},\delta)}\right).$$

Note that the above big-$\mathcal{O}$-notation holds in the limit where $\lambda \uparrow 1$. That is, even though the above expression evaluates to 0 whenever $\lambda(\mathcal{P},\delta) = 0$, that does not mean that the query complexity is 0 in that case.

The polynomial dependence on $1-\lambda(\mathcal{P},\delta)$ is a bit mysterious, that is, it is not intuitively clear where the exponent $3/2$ comes from, and whether it is necessary in the first place. In this section, we show that the power of $3/2$ can indeed be improved to 1, at the expense of some additional polylogarithmic factors in $\widetilde{W}_+(\mathcal{P},\delta)$, $\widetilde{W}_-(\mathcal{P},\delta)$ and $1/(1-\lambda(\mathcal{P},\delta))$. Thus, we improve over the algorithm from [IJ19] by a factor of $1/\sqrt{1-\lambda(\mathcal{P},\delta)}$.

One of the core elements of the approximate span program algorithm is that we will be performing several runs of phase estimation consecutively on the same state, i.e., without measuring and repreparing the state in between. To accommodate the analysis of this algorithmic technique, we define some notation for this first, in the following definition.

**8.2.2.** Definition (Multiple runs of phase estimation). Let $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ be a span program. Let $n \in \mathbb{N}$, and let $\mathbf{k} = (k_1, \ldots, k_n) \in (\mathbb{N} \cup \{\infty\})^n$ be a precision vector. Now, consider the following algorithm.

1. Prepare a register $\mathcal{A}$ in the state $|w_0\rangle$.
2. For $\ell = 1, \ldots, n$,
   (a) If $k_\ell = \infty$, then measure the state in $\mathcal{A}$ in the eigenbasis of $U(x, \mathcal{P})$. Denote the measured phase by $\Phi_\ell$ and output it.
   (b) Otherwise, add a register of $k_\ell$ auxiliary bits, initialized in state $|0\rangle^{\otimes k_\ell}$.
   (c) Run phase estimation, i.e., Algorithm 2.4.3, with these $k_\ell$ bits of precision, on register $\mathcal{A}$ and with span program unitary $U(x, \mathcal{P})$. Denote the outcome by $\Phi_\ell$.

We refer to the joint random variable $(\Phi_1, \ldots, \Phi_\ell)$ as the *outcomes of multiple runs of phase estimation with precision vector* $\mathbf{k}$. We write the state that is left in register $\mathcal{A}$ after post-selecting on all runs of phase estimation giving outcome 0, by $|w_0^{(\mathbf{k})}\rangle$. ◀

One of the crucial properties we will be using is that the order in which we perform these runs of phase estimation is not important, i.e., that all of these runs commute. The following lemma makes this claim precise.

**8.2.3.** Lemma. *Let $\mathcal{P}$ be a span program, and let $\mathbf{k} \in (\mathbb{N} \cup \{\infty\})^n$ be a precision vector. Let $\pi \in S_n$ be a permutation, and let $\mathbf{k}' = \mathbf{k} \circ \pi$ be the precision vector formed by permuting the entries of $\mathbf{k}$ according to $\pi$. Let $(\Phi_1, \ldots, \Phi_n)$ and $(\Phi'_1, \ldots, \Phi'_n)$ be the outcomes of multiple runs of phase estimation with precision vectors $\mathbf{k}$ and $\mathbf{k}'$, respectively. Then, $(\Phi_1, \ldots, \Phi_n) = (\Phi'_{\pi(1)}, \ldots, \Phi'_{\pi(n)})$. As such, for any event $A$, we have that $\mathbb{P}[(\Phi_1, \ldots, \Phi_n) \in A]$ is independent of the order in which we perform the individual runs of phase estimation.*

**Proof:**
What we need to prove here is that two runs of phase estimation commute. If $k$ and $k'$ are both finite, we easily observe on the circuit level that the runs of phase estimation with $k$ and $k'$ bits of precision commute. Similarly, we find that a controlled application of the unitary $U(x, \mathcal{P})$ leaves its eigenspaces invariant, and as such a run with infinite precision also commutes with a finite precision run on the circuit level. Two infinite-precision runs also trivially commute since they are the same operations. This completes the proof. □

Thus, by virtue of the previous lemma, in the remainder of this section we will simply claim that we perform multiple consecutive runs of phase estimation, without specifying the order in which we run them.

The core idea of the approximate span program algorithm is to run phase estimation with slowly increasing levels of precision. In particular, we let $\mathbf{k} =$

$(1, \ldots, k)$, and we let $(\Phi_1, \ldots, \Phi_k)$ be the outcome of multiple runs of phase estimation with precision vector $\mathbf{k}$. Then, we estimate the quantity

$$\mathbb{P}\left[\bigcap_{\ell=1}^{k} \Phi_\ell = 0\right] = \prod_{\ell=1}^{k} \mathbb{P}\left[\Phi_\ell = 0 \middle| \bigcap_{\ell'=1}^{\ell-1} \Phi_{\ell'} = 0\right], \tag{8.2.1}$$

where equality holds because of the definition of conditional probability. Each of the factors on the right-hand side can be obtained by preparing the state $|w_0^{(1,\ldots,\ell-1)}\rangle \in \mathcal{H}$, then running phase estimation with $\ell$ bits of precision on this initial state, and finally running amplitude estimation on the output being 0.

The crucial observation is that we can use the approximate positive witness size to lower bound the factors in this product individually a priori, which we present in the following lemma. This lower bound ultimately allows us to favorably bound the number of iterations required in the amplitude estimation step.

**8.2.4.** Lemma. *Let $\mathcal{P}$ be a span program, and let $k \in \mathbb{N}$. Let $\mathbf{k} = (1, 2, \ldots, k, \infty)$ be a precision vector, and let $(\Phi_1, \ldots, \Phi_k, \Phi)$ be the outcomes of multiple runs of phase estimation with precision vector $\mathbf{k}$. Then,*

$$\mathbb{P}\left[\Phi_k = 0 \middle| \bigcap_{\ell=1}^{k-1} \Phi_\ell = 0\right] \geq \frac{\mathbb{P}\left[\Phi = 0\right]}{\mathbb{P}\left[\bigcap_{\ell=1}^{k-1} \Phi_\ell = 0\right]} \geq 1 - \frac{1}{\frac{2^{2(k-1)}\mathbb{P}[\Phi=0]}{\widetilde{w}_+(x,\mathcal{P})} + 1}.$$

**Proof:**
Using the definition of conditional probabilities, we obtain that

$$\mathbb{P}\left[\Phi_k = 0 \middle| \bigcap_{\ell=1}^{k-1} \Phi_\ell = 0\right] = \frac{\mathbb{P}\left[\bigcap_{\ell=1}^{k} \Phi_\ell = 0\right]}{\mathbb{P}\left[\bigcap_{\ell=1}^{k-1} \Phi_\ell = 0\right]}. \tag{8.2.2}$$

For the numerator, we observe using Bayes's rule that

$$\mathbb{P}\left[\bigcap_{\ell=1}^{k} \Phi_\ell = 0\right] \geq \mathbb{P}\left[\bigcap_{\ell=1}^{k} \Phi_\ell = 0 \middle| \Phi = 0\right] \mathbb{P}[\Phi = 0] = \mathbb{P}[\Phi = 0],$$

where the first factor is 1 since if we first run phase estimation with infinite precision and post-select on the outcome being 0, then all subsequent finite-precision runs of phase estimation will output 0 with certainty. Thus, plugging this back into Equation (8.2.2), we obtain that

$$\mathbb{P}\left[\Phi_k = 0 \middle| \bigcap_{\ell=1}^{k-1} \Phi_\ell = 0\right] \geq \frac{\mathbb{P}\left[\Phi = 0\right]}{\mathbb{P}\left[\bigcap_{\ell=1}^{k-1} \Phi_\ell = 0\right]}.$$

This proves the first inequality.

Next, by standard probability theory, we find that $\mathbb{P}[\bigcap_{\ell=1}^{k-1} \Phi_\ell = 0] \leq \mathbb{P}[\Phi_{k-1} = 0]$, and thus

$$\frac{\mathbb{P}[\Phi = 0]}{\mathbb{P}\left[\bigcap_{\ell=1}^{k-1} \Phi_\ell = 0\right]} \geq \frac{\mathbb{P}[\Phi = 0]}{\mathbb{P}[\Phi_{k-1} = 0]} \geq \frac{\mathbb{P}[\Phi = 0]}{\mathbb{P}[\Phi = 0] + \frac{\widetilde{w}_+(x,\mathcal{P})}{2^{2(k-1)}}}.$$

where we used the operational interpretation of the approximate witness size, i.e., Lemma 8.1.3, in the last inequality. Rewriting the right-hand side yields

$$\frac{\mathbb{P}[\Phi = 0]}{\mathbb{P}\left[\bigcap_{\ell=1}^{k-1} \Phi_\ell = 0\right]} \geq 1 - \frac{\frac{\widetilde{w}_+(x,\mathcal{P})}{2^{2(k-1)}}}{\mathbb{P}[\Phi = 0] + \frac{\widetilde{w}_+(x,\mathcal{P})}{2^{2(k-1)}}} = 1 - \frac{1}{\frac{2^{2(k-1)}\mathbb{P}[\Phi=0]}{\widetilde{w}_+(x,\mathcal{P})} + 1}.$$

This completes the proof. □

Now, we turn to describing the approximate span program algorithm in detail. Before giving the final result in Algorithm 8.2.9, we present a few subroutines, starting with one that prepares the state $|w_0^{(1,\ldots,\ell)}\rangle$, in Algorithm 8.2.5.

---

**Algorithm 8.2.5:** Construction of the post-selected initial state.

---

**Input:**

1: $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$: a span program.
2: $\ell \in \mathbb{N}$: the maximum number of bits of precision used in phase estimation.
3: $\delta > 0$: a failure probability parameter.
4: $C_{|w_0\rangle}$: a quantum circuit acting on $\mathcal{H}$ that implements $|0\rangle \mapsto |w_0\rangle$.
5: $U$: a quantum circuit acting on $\mathcal{H}$ that implements $U(x, \mathcal{P})$.

**Derived objects:**

1: For all $j \in [\ell]$, a register $D_j$ containing $j$ qubits.
2: For all $j \in [\ell]$, $PE_j$: the phase estimation circuit with $j$ bits of precision, acting on $D_j \otimes \mathcal{H}$.
3: The circuit $R_0$ acting on $\bigotimes_{j=1}^{\ell} D_j$ that reflects through the state that is $|0\rangle$ on all registers $D_j$.
4: The circuit $C = \prod_{j=1}^{\ell} PE_j \cdot C_{|w_0\rangle}$ acting on $\bigotimes_{j=1}^{\ell} D_j \otimes \mathcal{H}$.

**Assumption:** $|(\langle w_0^{(1,\ldots,\ell)}| \otimes \langle 0|^{\otimes \ell}) C |0\rangle| \geq 1/2$.

**Output:** A state $|\psi'\rangle \in \bigotimes_{j=1}^{\ell} D_j \otimes \mathcal{H}$ such that

$$\left\| |\psi'\rangle - |0\rangle^{\otimes \ell} \otimes |w_0^{(1,\ldots,\ell)}\rangle \right\| \leq \delta.$$

**Queries:**

1: Number of calls to $U(x, \mathcal{P})$: $\mathcal{O}(2^k \log(1/\delta))$.
2: Number of calls to $C_{|w_0\rangle}$: $\mathcal{O}(\log(1/\delta))$.

**Procedure:** Constr-init-state$(\ell, \delta, C_{|w_0\rangle})$.

1: Run Fixed-point-ampl$(1/2, \delta, R_0, C)$.

---

**Proof of the properties of Algorithm 8.2.5:**
The claims about the number of calls follow directly from the properties of the fixed-point amplitude amplification algorithm, Algorithm 2.4.5. Thus, it remains to prove that the resulting state $|\psi'\rangle$ is indeed $\delta$-close to $|0\rangle^{\otimes \ell} \otimes |w_0^{(1,\ldots,\ell)}\rangle$.

To that end, observe that the circuit $C$ performs $\ell$ consecutive runs of phase estimation on the initial state $|w_0\rangle$, with precision vector $(1, \ldots, \ell)$. By definition, if we post-select on the measurement outcome of all these runs of phase estimation resulting in 0, then the resulting state in $\mathcal{H}$ is the state $|w_0^{(1,\ldots,\ell)}\rangle$. Thus, we can write

$$C\,|0\rangle = a\,|0\rangle^{\otimes \ell} \otimes |w_0^{(1,\ldots,\ell)}\rangle + |\perp\rangle\,,$$

where $|\perp\rangle$ does not have any overlap with the subspace $\mathcal{A} = \mathrm{Span}\{|0\rangle^{\otimes \ell}\} \otimes \mathcal{H}$, and we know that $|a| \geq 1/2$ by assumption. Moreover, observe that $R_0$ reflects through this subspace $\mathcal{A}$, and hence by the properties of the fixed-point amplitude amplification algorithm, Algorithm 2.4.5, we end up preparing a state $|\psi'\rangle$ such that

$$\left\| |\psi'\rangle - |0\rangle^{\otimes \ell} \otimes |w_0^{(1,\ldots,\ell)}\rangle \right\| \leq \delta.$$

This completes the proof. □

For technical reasons, we assume that the overlap between $C\,|0\rangle$ and $|w_0^{(1,\ldots,\ell)}\rangle$ is at least $1/2$. If we let $(\Phi_1, \ldots, \Phi_\ell)$ be the outcome of multiple runs of phase estimation with precision vector $(1, \ldots, \ell)$, then we can rewrite this assumption as

$$\mathbb{P}\left[\bigcap_{\ell'=1}^{\ell} \Phi_{\ell'} = 0\right] \geq \frac{1}{4}.$$

Whenever we call the above subroutine, we will have to check that this assumption is indeed satisfied.

Next, we show how we can compute approximations to the individual factors in the product in Equation (8.2.1). To that end, we introduce the zero-phase overlap estimation routine, in Algorithm 8.2.6.

**Algorithm 8.2.6:** Zero-phase overlap estimation

**Input:**
1: $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$: a span program.
2: $\ell \in \mathbb{N}$: the maximum number of bits of precision in phase estimation.
3: $M > 0$: a precision parameter.
4: $\delta > 0$: a failure probability tolerance parameter.
5: $C_{|w_0\rangle}$: a circuit acting on $\mathcal{H}$ that implements $|0\rangle \mapsto |w_0\rangle$.
6: $U$: a routine acting on $\mathcal{H}$ that implements $U(x, \mathcal{P})$.
**Derived objects:**
1: $N = \lceil 18 \ln(2/\delta) \rceil$.

2: $(\Phi_1, \ldots, \Phi_\ell)$ is the outcome of multiple runs of phase estimation with $U(x, \mathcal{P})$ on $|w_0\rangle$, with precision vector $(1, \ldots, \ell)$.
3: For all $j \in [\ell]$, let $D_j$ be a register of $j$ qubits.
4: Let $C' = \text{CONSTR-INIT-STATE}(\ell - 1, 1/(9M), C_{|w_0\rangle})$ be a circuit acting on $\bigotimes_{j=1}^{\ell-1} D_j \otimes \mathcal{H}$ that approximately prepares $|w_0^{(1,\ldots,\ell)}\rangle$.
5: Let $C = PE_\ell \cdot C'$ be a circuit acting on $\bigotimes_{j=1}^{\ell} D_j \otimes \mathcal{H}$ that performs phase estimation with $U(x, \mathcal{P})$ on $|w_0^{(1,\ldots,\ell)}\rangle$ with $\ell$ bits of precision.
6: Let $R_0$ be the circuit that reflects through the all-zeros state of $D_\ell$.

**Assumption:**

$$\mathbb{P}\left[\bigcap_{\ell'=1}^{\ell-1} \Phi_{\ell'} = 0\right] \geq \frac{1}{4}.$$

**Output:** $p_0^{(\ell)} \in [0, 1]$ such that $|p_0^{(\ell)} - p| \leq 2\pi\sqrt{p(1-p)}/M + \pi^2/M^2$, where

$$p = \mathbb{P}\left[\Phi_\ell = 0 \middle| \bigcap_{\ell'=1}^{\ell-1} \Phi_{\ell'} = 0\right].$$

**Success probability:** Lower bounded by $1 - \delta$.
**Queries:**
1: Number of calls to $U$: $\mathcal{O}(2^\ell M \log(M) \log(1/\delta))$.
2: Number of calls to $C_{|w_0\rangle}$: $\mathcal{O}(M \log(M) \log(1/\delta))$.
**Procedure:** ZERO-PHASE-EST$(\mathcal{P}, \ell, M, \delta, C_{|w_0\rangle}, U)$
1: Output the median of $N$ independent runs of AMP-EST$(M, C, R_0)$.

---

**Proof of the properties of Algorithm 8.2.6:**
The claims about the number of queries follow directly from the properties of the amplitude estimation algorithm, Algorithm 2.4.4, and the initial-state construction algorithm, Algorithm 8.2.5. Thus, it remains to prove the lower bound on the success probability.

To that end, we first observe that in every run of amplitude estimation, we make $M$ calls to the routine that constructs the initial state $|w_0^{(1,\ldots,\ell)}\rangle$. Since we run each of these preparation routines with norm precision $1/(9M)$, the total norm error that accumulates due to this part is at most $1/9$. Thus, a single run of amplitude estimation succeeds with probability $p' \geq 8/\pi^2 - 1/9 > 2/3$, in which case it outputs a number $\widetilde{p}$ such that

$$|\widetilde{p} - p| \leq \frac{2\pi\sqrt{p(1-p)}}{M} + \frac{\pi^2}{M^2}.$$

Now, for all $j \in [N]$, let $X_j$ be the random variable that is 1 if the above inequality is satisfied on the $j$th iteration, and 0 otherwise. Then, we obtain by Hoeffding's

inequality that

$$\begin{aligned}\mathbb{P}\left[\sum_{j=1}^{N} X_j \leq \frac{N}{2}\right] &\leq \mathbb{P}\left[\left|\sum_{j=1}^{N} X_j - Np'\right| \geq \left|\frac{N}{2} - Np'\right|\right] \leq 2\exp\left(-\frac{2|\frac{N}{2} - Np'|^2}{N}\right) \\ &\leq 2\exp\left(-2N\left|\frac{1}{2} - \frac{2}{3}\right|^2\right) = 2\exp\left(-\frac{N}{18}\right) \\ &\leq 2\exp\left(-\frac{18\ln(2/\delta)}{18}\right) = \delta.\end{aligned}$$

This completes the proof. □

Next, we introduce an algorithm that given a span program $\mathcal{P}$ computes the function that it 1/2-approximates, i.e., whose approximation parameter is $\delta = 1/2$ as defined in Definition 8.1.1. This results in Algorithm 8.2.7.

---

**Algorithm 8.2.7:** Renormalized approximate span program algorithm

---

**Input:**

1: $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$: a span program.
2: $\overline{W}_+$: an upper bound on $\widetilde{W}_+(\mathcal{P}, 1/2)$.
3: $\overline{\lambda} \in (0,1)$: an upper bound on $\lambda(\mathcal{P}, 1/2)$.
4: $C_{|w_0\rangle}$: a quantum circuit acting on $\mathcal{H}$ that implements $|0\rangle \mapsto |w_0\rangle$.
5: $U$: a quantum circuit acting on $\mathcal{H}$ that implements $U(x, \mathcal{P})$.

**Derived objects:**

1: $k = \lceil \frac{1}{2}\log\left(\frac{4\overline{W}_+}{1-\overline{\lambda}}\right)\rceil + 1$.
2: For all $\ell \in [k]$, $M^{(\ell)} = \max\{\lceil \frac{32\pi k\sqrt{\overline{W}_+}}{(1-\overline{\lambda})2^{\ell-2}}\rceil, \lceil \frac{4\pi\sqrt{k}}{\sqrt{1-\overline{\lambda}}}\rceil\}$.

**Output:** 1 if $x$ is a positive 1/2-approximate input, 0 otherwise.
**Success probability:** Lower bounded by 2/3.
**Queries:** Number of calls to $U$ and $C_{|w_0\rangle}$: $\widetilde{\mathcal{O}}(\sqrt{\overline{W}_+}/(1-\overline{\lambda}))$.
**Procedure:** Renorm-Approx-Span$(\mathcal{P}, \overline{W}_+, \overline{\lambda}, C_{|w_0\rangle}, U)$

1: For $\ell = 1, \ldots, k$,
   1. Run $p_0^{(\ell)} =$ Zero-phase-est$(\mathcal{P}, \ell - 1, M^{(\ell)}, 1/(3k), C_{|w_0\rangle}, U)$.
   2. If $\prod_{\ell'=1}^{\ell} p_0^{(\ell')} < \frac{1}{2} - \frac{1-\overline{\lambda}}{8}$, stop and output 1.
   3. If $\prod_{\ell'=1}^{\ell} p_0^{(\ell')} > \frac{\overline{\lambda}}{2} + \frac{\overline{W}_+}{2^{2\ell}} + \frac{1-\overline{\lambda}}{8}$, stop and output 0.

---

**Proof of the properties of Algorithm 8.2.7:**
We start by proving that the algorithm always outputs either 1 or 0. To that end, observe that in the $k$th iteration, we have

$$\frac{\overline{\lambda}}{2} + \frac{\overline{W}_+}{2^{2k}} + \frac{1-\overline{\lambda}}{8} < \frac{\overline{\lambda}}{2} + \frac{1-\overline{\lambda}}{4} + \frac{1-\overline{\lambda}}{8} = \frac{1}{2} - \frac{1-\overline{\lambda}}{8},$$

and hence we are guaranteed to obtain either output 1 or 0 in that iteration.

Next, we check the claims on the number of queries. Summing the cost expressions from Algorithm 8.2.6 yields that the number of calls to $C_{|w_0\rangle}$ and $U$ scale as

$$\mathcal{O}\left(\sum_{\ell=1}^{k} M^{(\ell)} \log(M^{(\ell)}) \log(k)\right), \quad \text{and} \quad \mathcal{O}\left(\sum_{\ell=1}^{k} 2^{\ell} M^{(\ell)} \log(M^{(\ell)}) \log(k)\right),$$

respectively. Since the first part of the maximum in the definition of $M^{(\ell)}$ dominates the second part in all parameters, we obtain that

$$M^{(\ell)} = \mathcal{O}\left(\frac{\sqrt{\overline{W}_+}}{(1-\overline{\lambda})2^{\ell}}\right), \qquad \text{and} \qquad k = \mathcal{O}\left(\log\left(\frac{\overline{W}_+}{1-\overline{\lambda}}\right)\right),$$

and thus the number of calls to $C_{|w_0\rangle}$ and $U$ scales as

$$\widetilde{\mathcal{O}}\left(\frac{\sqrt{\overline{W}_+}}{1-\overline{\lambda}}\right),$$

where the tilde hides factors that are polylogarithmic in $\overline{W}_+$ and $1/(1-\overline{\lambda})$. This completes the proof of the claimed number of queries.

Thus, it remains to prove the lower bound on the success probability. To that end, we first make the assumption that all calls to the zero-phase estimation algorithm succeed. Since we know from Algorithm 8.2.6 that each call fails with probability at most $1/(3k)$, the total probability that this assumption is not satisfied is $1/3$. Hence, under this assumption, it remains to prove that the algorithm always succeeds.

To that end, we prove several properties of the state of the algorithm using induction over the iterations. The induction hypothesis is that in the $\ell$th iteration, with $\ell \in [k]$, the following two claims hold:

1. The assumption of the zero-phase overlap estimation algorithm is satisfied.
2. We have
$$\left|\prod_{\ell'=1}^{\ell-1} p_0^{(\ell')} - \mathbb{P}\left[\bigcap_{\ell'=1}^{\ell-1} \Phi_{\ell'} = 0\right]\right| \leq \frac{1-\overline{\lambda}}{8} \cdot \frac{\ell-1}{k}. \tag{8.2.3}$$

Indeed, these two claims trivially hold in the first iteration. The assumption for the zero-phase overlap estimation algorithm becomes $1 \geq 1/4$, and in the second claim both the left- and right-hand side are 0.

Now, we suppose that the induction hypothesis holds for some $\ell - 1$, with $\ell \geq 2$. We proceed by proving the following four claims:

1. We have
$$\left|\prod_{\ell'=1}^{\ell} p_0^{(\ell')} - \mathbb{P}\left[\bigcap_{\ell'=1}^{\ell} \Phi_{\ell'} = 0\right]\right| \leq \frac{1-\overline{\lambda}}{8} \cdot \frac{\ell}{k}.$$

2. If we output 1, then the input to the span program is 1/2-approximately positive.
3. If we output 0, then the input to the span program is 1/2-approximately negative.
4. If we don't output anything, then the assumption for the call to the zero-phase overlap estimation algorithm in iteration $\ell + 1$ is satisfied.

Claims 1 and 4 provide the induction step. Claims 2 and 3 prove that the algorithm always produces the correct output.

We start by proving claim 1. To that end, observe that since the previous iteration did not terminate, we have

$$\frac{1}{2} - \frac{1-\overline{\lambda}}{8} \leq \prod_{\ell'=1}^{\ell-1} p_0^{(\ell')} \leq \frac{\overline{\lambda}}{2} + \frac{\overline{W}_+}{2^{2\ell}} + \frac{1-\overline{\lambda}}{8}.$$

Now, suppose that we have a 1/2-approximate positive input. By Lemma 8.1.3 and the induction hypothesis, Equation (8.2.3), we have

$$\begin{aligned}\mathbb{P}\left[\Phi = 0\right] + \frac{\widetilde{w}_+(x,\mathcal{P})}{2^{2(\ell-1)}} \geq \mathbb{P}\left[\Phi_\ell = 0\right] \geq \mathbb{P}\left[\bigcap_{\ell'=1}^{\ell-1} \Phi_{\ell'} = 0\right] &\geq \prod_{\ell'=1}^{\ell-1} p_0^{(\ell')} - \frac{1-\overline{\lambda}}{8} \\ &\geq \frac{1}{2} - \frac{1-\overline{\lambda}}{4},\end{aligned}$$

and so, using Lemma 8.2.4 and the previous equation, we find that

$$\begin{aligned}\mathbb{P}\left[\Phi_\ell = 0 \middle| \bigcap_{\ell'=1}^{\ell-1} \Phi_{\ell'} = 0\right] &\geq 1 - \frac{1}{\frac{2^{2(\ell-1)}\mathbb{P}[\Phi=0]}{\widetilde{w}_+(x,\mathcal{P})} + 1} \geq 1 - \frac{1}{\frac{2^{2(\ell-1)}}{\widetilde{w}_+(x,\mathcal{P})} \cdot \left(\frac{1}{2} - \frac{1-\overline{\lambda}}{4}\right)} \\ &\geq 1 - \frac{\overline{W}_+}{2^{2(\ell-1)} \cdot \frac{1}{4}} = 1 - \frac{\overline{W}_+}{2^{2\ell-4}}.\end{aligned}$$

On the other hand, suppose that we are in a 1/2-approximate negative input. Then, again employing Lemma 8.2.4 and the induction hypothesis, Equation (8.2.3), yields

$$\begin{aligned}\mathbb{P}\left[\Phi_\ell = 0 \middle| \bigcap_{\ell'=1}^{\ell-1} \Phi_{\ell'} = 0\right] &\geq \frac{\mathbb{P}\left[\Phi = 0\right]}{\mathbb{P}\left[\bigcap_{\ell'=1}^{\ell-1} \Phi_{\ell'} = 0\right]} \geq \frac{\frac{1}{2}}{\prod_{\ell'=1}^{\ell-1} p_0^{(\ell')} + \frac{1-\overline{\lambda}}{8}} \\ &\geq \frac{\frac{1}{2}}{\frac{\overline{\lambda}}{2} + \frac{\overline{W}_+}{2^{2(\ell-1)}} + \frac{1-\overline{\lambda}}{4}} \geq \frac{1}{1 + \frac{\overline{W}_+}{2^{2\ell-3}}} \geq 1 - \frac{\overline{W}_+}{2^{2\ell-3}},\end{aligned}$$

where we used that $1/(1+x) \geq 1 - x$ for all $x \geq 0$. Thus, we have now proved that in both the positive and the negative case, we have

$$p := \mathbb{P}\left[\Phi_\ell = 0 \middle| \bigcap_{\ell'=1}^{\ell-1} \Phi_{\ell'} = 0\right] \geq 1 - \frac{\overline{W}_+}{2^{2\ell-4}}.$$

Hence, the zero-phase overlap estimation algorithm outputs an estimate $p_0^{(\ell)}$ that satisfies

$$|p_0^{(\ell)} - p| \leq \frac{2\pi\sqrt{p(1-p)}}{M^{(\ell)}} + \frac{\pi^2}{(M^{(\ell)})^2} \leq \frac{2\pi\sqrt{\overline{W}_+}}{2^{\ell-2}M^{(\ell)}} + \frac{\pi^2}{(M^{(\ell)})^2}$$
$$\leq \frac{1-\overline{\lambda}}{16k} + \frac{1-\overline{\lambda}}{16k} = \frac{1-\overline{\lambda}}{8k}.$$

Thus,

$$\left|\prod_{\ell'=1}^{\ell} p_0^{(\ell')} - \mathbb{P}\left[\bigcap_{\ell'=1}^{\ell} \Phi_{\ell'} = 0\right]\right| = \left|p_0^{(\ell)} \prod_{\ell'=1}^{\ell-1} p_0^{(\ell')} - p \cdot \mathbb{P}\left[\bigcap_{\ell'=1}^{\ell-1} \Phi_{\ell'} = 0\right]\right|$$
$$\leq \left|p_0^{(\ell)} \prod_{\ell'=1}^{\ell-1} p_0^{(\ell')} - p_0^{(\ell)} \cdot \mathbb{P}\left[\bigcap_{\ell'=1}^{\ell-1} \Phi_{\ell'} = 0\right]\right| + \left|\left(p_0^{(\ell)} - p\right) \cdot \mathbb{P}\left[\bigcap_{\ell'=1}^{\ell-1} \Phi_{\ell'} = 0\right]\right|$$
$$\leq \left|\prod_{\ell'=1}^{\ell-1} p_0^{(\ell')} - \mathbb{P}\left[\bigcap_{\ell'=1}^{\ell-1} \Phi_{\ell'} = 0\right]\right| + \left|p_0^{(\ell)} - p\right|$$
$$\leq \frac{1-\overline{\lambda}}{8} \cdot \frac{\ell-1}{k} + \frac{1-\overline{\lambda}}{8k} = \frac{1-\overline{\lambda}}{8} \cdot \frac{\ell}{k}.$$

This proves claim 1.

Next, if we output 1, then we have

$$\mathbb{P}\left[\Phi = 0\right] \leq \mathbb{P}\left[\bigcap_{\ell'=1}^{\ell} \Phi_{\ell'} = 0\right] \leq \prod_{\ell'=1}^{\ell} p_0^{(\ell')} + \frac{1-\overline{\lambda}}{8} < \frac{1}{2},$$

and hence we indeed have a 1/2-approximate positive input. This proves claim 2.

Similarly, if we output 0, then we find using Lemma 8.1.3 and Equation (8.2.3) that

$$\mathbb{P}[\Phi = 0] + \frac{\widetilde{w}_+(x,\mathcal{P})}{2^{2\ell}} \geq \mathbb{P}\left[\Phi_\ell = 0\right] \geq \mathbb{P}\left[\bigcap_{\ell'=1}^{\ell} \Phi_{\ell'} = 0\right] \geq \prod_{\ell'=1}^{\ell} p_0^{(\ell')} - \frac{1-\overline{\lambda}}{8}$$
$$> \frac{\overline{\lambda}}{2} + \frac{\overline{W}_+}{2^{2\ell}},$$

and thus

$$\frac{1}{\delta w_-(x,\mathcal{P})} = 2\mathbb{P}[\Phi = 0] > \overline{\lambda} \geq \lambda(\mathcal{P}, 1/2), \quad \text{or} \quad \widetilde{w}_+(x,\mathcal{P}) > \overline{W}_+ \geq \widetilde{W}_+(\mathcal{P}, 1/2).$$

Directly from the definition of approximate span programs, i.e., Definition 8.1.1, we observe that both imply $x$ is a 1/2-approximate negative input. This proves claim 3.

Finally, if we don't output anything on this iteration, then

$$\mathbb{P}\left[\bigcap_{\ell'=1}^{\ell} \Phi_{\ell'} = 0\right] \geq \prod_{\ell'=1}^{\ell} p_0^{(\ell)} - \frac{1-\overline{\lambda}}{8} \geq \frac{1}{2} - \frac{1-\overline{\lambda}}{4} \geq \frac{1}{4}.$$

Thus, the assumption for the zero-phase overlap estimation routine is satisfied. This proves claim 4, and thus the proof is complete. □

Now that we have presented the approximate span program algorithm in the special case where the span program $\mathcal{P}$ is 1/2-approximating the function that we wish to compute, we show how this result can also be used in the more general setting where we want to compute the function that is $\delta$-approximated by $\mathcal{P}$. Similar to the exact case, we employ span program renormalization, as defined in Definition 6.1.16. To that end, we investigate the implications of the renormalization construction in the approximate span program context, in the following theorem.

**8.2.8.** Theorem (Approximate span program renormalization).
*Let $\mathcal{P}$ be a span program, $\delta \in (0,1]$ and $\beta > 0$. Next, let $\mathcal{P}'$ be $\mathcal{P}$ renormalized with parameter $\beta$. Then,*

1. *$\mathcal{P}$ $\delta$-approximates $f : \mathcal{D} \to \{0,1\}$ $\Leftrightarrow$ $\mathcal{P}'$ $\delta'$-approximates $f$, where*

$$\delta' = \frac{\beta^2 + 1}{\beta^2/\delta + 1}.$$

2. *We have the relations*

$$\widetilde{W}_-(\mathcal{P}', \delta') = \frac{\beta^2 \widetilde{W}_-(\mathcal{P}, \delta) + 1}{\beta^2 + 1}, \qquad \widetilde{W}_+(\mathcal{P}', \delta') = \left(1 + \frac{1}{\beta^2}\right) \widetilde{W}_+(\mathcal{P}, \delta),$$

*and*

$$\lambda(\mathcal{P}', \delta') = \frac{\beta^2 + \delta}{\frac{\beta^2}{\lambda(\mathcal{P}, \delta)} + \delta}.$$

**Proof:**
We denote $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$, and $\mathcal{P}' = (\mathcal{H}', x \mapsto \mathcal{H}'(x), \mathcal{K}', |w_0'\rangle)$. Using the characterization of the witness sizes from Theorem 6.1.17, we obtain

$$\begin{aligned}
&\left\|\Pi_{(\mathcal{K}')^\perp \cap \mathcal{H}'(x)^\perp} |w_0'\rangle\right\|^2 \geq \delta' \quad \Leftrightarrow \quad w_-(x, \mathcal{P}') \leq \frac{1}{\delta'} \\
&\Leftrightarrow \quad \frac{\beta^2 w_-(x, \mathcal{P}) + 1}{\beta^2 + 1} \leq \frac{\beta^2/\delta + 1}{\beta^2 + 1} \\
&\Leftrightarrow \quad w_-(x, \mathcal{P}) \leq \frac{1}{\delta} \quad \Leftrightarrow \quad \left\|\Pi_{\mathcal{K}^\perp \cap \mathcal{H}(x)^\perp} |w_0\rangle\right\|^2 \geq \delta.
\end{aligned}$$

This proves claim 1. For the second claim, the formulae for $\widetilde{W}_-(\mathcal{P}', \delta')$ and $\widetilde{W}_+(\mathcal{P}', \delta')$ follow directly from Theorem 6.1.17, and thus it remains to check the formula for $\lambda(\mathcal{P}', \delta')$. To that end, observe that

$$\begin{aligned}\lambda(\mathcal{P}', \delta') &= \max_{x \in f^{-1}(1)} \frac{1}{\delta' w_-(x, \mathcal{P}')} = \max_{x \in f^{-1}(1)} \frac{\beta^2/\delta + 1}{\beta^2 + 1} \cdot \frac{\beta^2 + 1}{\beta^2 w_-(x, \mathcal{P}) + 1} \\ &= \max_{x \in f^{-1}(1)} \frac{\beta^2 + \delta}{\beta^2 \delta w_-(x, \mathcal{P}) + \delta} = \frac{\beta^2 + \delta}{\frac{\beta^2}{\lambda(\mathcal{P}, \delta)} + \delta}.\end{aligned}$$

This completes the proof. □

Finally, we can now give the most general version of the approximate span program algorithm, where we only require that $\delta < 1/2$. The final result is Algorithm 8.2.9.

---

**Algorithm 8.2.9:** Approximate span program algorithm

---

**Input:**

1: $\mathcal{P}$: a span program.
2: $\delta \in (0, 1/2)$: an approximation parameter.
3: $\overline{W}_+$: an upper bound of $\widetilde{W}_+(\mathcal{P}, \delta)$.
4: $\overline{\lambda}$: an upper bound of $\lambda(\mathcal{P}, \delta)$.
5: $C_{|w_0\rangle}$: a circuit acting on $\mathcal{H}$ that implements $|0\rangle \mapsto |w_0\rangle$.
6: $U$: a circuit acting on $\mathcal{H}$ that implements $U(x, \mathcal{P})$.

**Derived objects:**

1: $\beta = \sqrt{1/(1/\delta - 2)}$.
2: $\mathcal{P}' = \mathcal{P}$ renormalized with constant $\beta$.
3: $\overline{W}'_+ = \left(\frac{1}{\delta} - 1\right) \overline{W}_+$.
4: $\overline{\lambda}' = \frac{1+\overline{\lambda}}{2}$.

**Output:** 1 if $x$ is a $\delta$-approximate positive input for $\mathcal{P}$, 0 otherwise.

**Success probability:** Lower bounded by 2/3.

**Queries:** Number of calls to $C_{|w_0\rangle}$ and $U$: $\widetilde{\mathcal{O}}(\sqrt{\overline{W}_+/\delta}/(1 - \overline{\lambda}))$.

**Procedure:** Approx-Span$(\mathcal{P}, \delta, \overline{W}_+, \overline{\lambda}, C_{|w_0\rangle}, U)$:

1: Run Renorm-Approx-Span$(\mathcal{P}', \overline{W}'_+, \overline{\lambda}', C_{|w'_0\rangle}, U')$, where $C_{|w'_0\rangle}$ and $U'$ are constructed exactly like in Algorithm 6.1.18.

---

**Proof of the properties of Algorithm 8.2.9:**
Let $f_\delta : \mathcal{D} \to \{0, 1\}$ be the function that $\mathcal{P}$ $\delta$-approximates. Using the first property of Theorem 8.2.8, we obtain that $\mathcal{P}'$ $\delta'$-approximates $f_\delta$ with

$$\delta' = \frac{\beta^2 + 1}{\beta^2/\delta + 1} = \frac{\frac{1}{\frac{1}{\delta} - 2} + 1}{\frac{1}{\delta} \cdot \frac{1}{\frac{1}{\delta} - 2} + 1} = \frac{1 + \frac{1}{\delta} - 2}{\frac{1}{\delta} + \frac{1}{\delta} - 2} = \frac{1}{2}.$$

Thus, $\mathcal{P}'$ indeed 1/2-approximates the function we want to compute. Again using Theorem 8.2.8, we observe that

$$\widetilde{W}_+(\mathcal{P}', \delta') = \left(1 + \frac{1}{\beta^2}\right) \widetilde{W}_+(\mathcal{P}, \delta) \leq \left(\frac{1}{\delta} - 1\right) \overline{W}_+ = \overline{W}'_+.$$

We also observe using the formula for $\lambda(\mathcal{P}', \delta')$ in Theorem 8.2.8 that

$$\begin{aligned}\lambda(\mathcal{P}', \delta') &= \frac{\beta^2 + \delta}{\frac{\beta^2}{\lambda(\mathcal{P},\delta)} + \delta} \leq \frac{\frac{1}{\frac{1}{\delta}-2} + \delta}{\frac{1}{\left(\frac{1}{\delta}-2\right)\overline{\lambda}} + \delta} = \frac{1 + 1 - 2\delta}{\frac{1}{\overline{\lambda}} + 1 - 2\delta} = 1 - \frac{\frac{1}{\overline{\lambda}} - 1}{\frac{1}{\overline{\lambda}} + 1 - 2\delta} \\ &\leq 1 - \frac{1 - \overline{\lambda}}{1 + \overline{\lambda}} \leq 1 - \frac{1 - \overline{\lambda}}{2} = \frac{1 + \overline{\lambda}}{2} = \overline{\lambda}'.\end{aligned}$$

Next, we observe that $C'_{|w_0\rangle}$ and $U'$ indeed implement the initial state construction routine, and the new span program unitary $U(x, \mathcal{P}')$, respectively. Thus, we can indeed use Algorithm 8.2.7 to compute $f_\delta$, and the cost requirements follow directly. This completes the proof. □

Finally, if $\mathcal{P}$ $\delta$-approximate a particular function $f$, then it also $1/\widetilde{W}_-(\mathcal{P}, \delta)$-approximates the same function. Thus, we can always choose $\delta' = 1/\widetilde{W}_-(\mathcal{P}, \delta)$, and plug $\delta'$ into the Algorithm 8.2.9. We prove this in the following corollary.

**8.2.10.** Corollary. *Let $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ be a span program and $\delta \in (0, 1/2)$ be an approximation parameter. Let $f_\delta$ be the function that $\mathcal{P}$ $\delta$-approximates. Then, there exists a quantum algorithm that evaluates $f_\delta$ with probability at least 2/3, using*

$$\widetilde{O}\left(\frac{\sqrt{\widetilde{W}_+(\mathcal{P}, \delta)\widetilde{W}_-(\mathcal{P}, \delta)}}{1 - \lambda(\mathcal{P}, \delta)}\right)$$

*calls to a circuit that prepares the state $|w_0\rangle$, and a circuit that implements the span program unitary $U(x, \mathcal{P})$.*

**Proof:**
We take $\delta' = 1/\widetilde{W}_-(\mathcal{P}, \delta)$, $\overline{W}_+ = \widetilde{W}_+(\mathcal{P}, \delta)$, and $\overline{\lambda} = \lambda(\mathcal{P}, \delta)$, and run Approx-span($\mathcal{P}$, $\delta'$, $\overline{W}_+$, $\overline{\lambda}$, $C_{|w_0\rangle}$, $U$). Indeed, we find that if $x$ is a $\delta$-approximate negative input, then

$$\left\|\Pi_{\mathcal{K}^\perp \cap \mathcal{H}(x)^\perp} |w_0\rangle\right\|^2 = \frac{1}{w_-(x, \mathcal{P})} \geq \frac{1}{\widetilde{W}_-(\mathcal{P}, \delta)} = \delta',$$

and thus it is also a $\delta'$-approximate negative input. Moreover, there exists an input $x$ for which the above inequality attains equality, and thus we find that

$\delta' \geq \delta$. This also implies that if $x$ is a $\delta$-approximate positive input for $\mathcal{P}$, it also is a $\delta'$-approximate positive input, and so $f_\delta = f_{\delta'}$. We immediately find that

$$\widetilde{W}_+(\mathcal{P}, \delta') = \widetilde{W}_+(\mathcal{P}, \delta) = \overline{W}_+,$$

and thus $\overline{W}_+$ is indeed an upper bound for $\widetilde{W}_+(\mathcal{P}, \delta')$. Finally, it follows that

$$\lambda(\mathcal{P}, \delta') = \max_{x \in f_{\delta'}^{-1}(1)} \frac{1}{\delta' w_-(x, \mathcal{P})} \leq \max_{x \in f_\delta^{-1}(1)} \frac{1}{\delta w_-(x, \mathcal{P})} = \lambda(\mathcal{P}, \delta) = \overline{\lambda},$$

and thus $\overline{\lambda}$ is indeed an upper bound for $\lambda(\mathcal{P}, \delta')$. Thus, all the assumptions of Algorithm 8.2.9 are satisfied, and the resulting complexity statement follows. This completes the proof. □

Comparing the statement of Corollary 8.2.10 with Theorem 8.2.1 from earlier work reveals that we indeed speed up the algorithm presented in [IJ19, Corollary 3.7] by a factor of $\sqrt{1 - \lambda(\mathcal{P}, \delta)}$.

We remark here that the assumption that $\delta \in (0, 1/2)$ is not really necessary. One can also give a similar renormalization construction when $\delta \in (1/2, 1]$, simply by taking an OR-composition of $\mathcal{P}$ with a trivial span program that has only positive inputs. In this way, one can turn any span program $\mathcal{P}$ that $\delta$-approximates $f$ into one that $1/2$-approximates $f$, and similarly use Algorithm 8.2.7.

We end this subsection with an example of an approximate span program that computes the threshold function.

**8.2.11.** Example. Let $x \in \{0,1\}^n$, and define

$$\mathcal{H} = \mathrm{Span}\{|j\rangle : j \in [n]\}, \qquad \mathcal{H}(x) = \mathrm{Span}\{|j\rangle : x_j = 1\},$$
$$\mathcal{K} = \{0\}, \qquad |w_0\rangle = \frac{1}{\sqrt{n}} \sum_{j=1}^{n} |j\rangle .$$

Let $\mathcal{P} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$ be a span program on $\{0,1\}^n$. For all $k \in [n]$, let $\delta = 1 - (k-1)/n$. Then, $\mathcal{P}$ $\delta$-approximates the threshold function $f_n^k : \{0,1\}^n \to \{0,1\}$ on $n$ bits with threshold $k$, i.e.,

$$f_n^k(x) = \begin{cases} 1, & \text{if } |x| \geq k, \\ 0, & \text{otherwise.} \end{cases}$$

For all $x \in \{0,1\}^n$, we have $w_-(x, \mathcal{P}) = n/(n - |x|)$, and $\widetilde{w}_+(x, \mathcal{P}) = |x|/n$. Thus,

$$\widetilde{W}_-(\mathcal{P}, \delta) = \frac{n}{n-k+1}, \quad \widetilde{W}_+(\mathcal{P}, \delta) = \frac{k}{n}, \quad \text{and} \quad \lambda(\mathcal{P}, \delta) = 1 - \frac{1}{n-k+1}.$$

Thus, the approximate span program algorithm evaluates the threshold function with a number of queries to $U(x, \mathcal{P})$ that satisfies

$$\widetilde{\mathcal{O}}\left(\frac{\sqrt{\frac{n}{n-k+1}\cdot\frac{k}{n}}}{1-\left(1-\frac{1}{n-k+1}\right)}\right) = \widetilde{\mathcal{O}}\left(\sqrt{k(n-k+1)}\right).$$

Moreover, we can implement the span program unitary $U(x, \mathcal{P})$ with a single phase query to the bit string $x$, which means that the above expression is an upper bound for the query complexity of the threshold function on $n$ bits with threshold $k$. ◁

It is remarkable that we recover the optimal query complexity for the threshold function with the approximate span program framework, up to polylogarithmic factors. In particular, this implies that the approximate span program algorithm we derived in this subsection is essentially optimal in the parameters $\widetilde{W}_+(\mathcal{P}, \delta)$, $\widetilde{W}_-(\mathcal{P}, \delta)$ and $1/(1-\lambda(\mathcal{P}, \delta))$, i.e., optimal up to polylogarithmic factors, since if we found an algorithm that had a better polynomial dependence on these parameters, this would also improve the query complexity of the algorithm for threshold function sketched above, which we know to be impossible.

Consequently, it is an interesting direction for further research to investigate if the polylogarithmic overhead in the approximate span program algorithm can be removed. If this is the case, then the above approximate span program would indeed exactly recover the query complexity of the threshold function. However, there are many places in the construction presented here where polylogarithmic factors arise, and it is not at all clear how these can be overcome. It might be required to develop some fundamentally new algorithmic techniques to fully resolve this question, and we leave that for future work.

The improved algorithm for evaluating approximate span programs has a couple of interesting implications. Most notably, it directly improves the *witness-size estimation* algorithm [IJ19, Theorem 3.8], which estimates the witness size of a particular input $x \in \mathcal{D}$ to a span program $\mathcal{P}$. We improve the dependence on the precision from $\mathcal{O}(1/\varepsilon^{3/2})$ to $\mathcal{O}(1/\varepsilon)$. This algorithm is used as a subroutine in some subsequent works, all of which are consequently improved as well. Examples include algorithms that estimate graph resistance [JK17, Lemma 25], graph capacitance [JJK+18, Corollary 20], and circuit rank [DKW19, Theorem 8].

## 8.3 Equivalence with quantum query algorithms

We briefly mention here that it is possible to take a quantum query algorithm that computes a boolean function $f : \mathcal{D} \to \{0, 1\}$, where $\mathcal{D} \subseteq \{0, 1\}^n$, with bounded error, turn it into an approximate span program, and then turn it back into a quantum algorithm that computes $f$ with bounded error again. Moreover, this

can be done without significant overhead in the number of queries and the space requirements [Jef20]. Thus, if we are searching for a query-efficient or space-efficient quantum algorithm to compute a given boolean function $f : \mathcal{D} \to \{0,1\}$, we can without loss of generality instead look for an approximate span program that computes this function.

In this section, we investigate whether we can obtain a similar result with regards to the number of *gates* used in the quantum algorithm. That is, we wonder whether it is possible to take a quantum algorithm computing a boolean function $f : \mathcal{D} \to \{0,1\}$, where $\mathcal{D} \subseteq \{0,1\}^n$, with bounded error, turn it into an approximate span program, and then turn it back into a quantum algorithm, without incurring significant overhead in the number of gates. This section is largely based on [CJO+20] and its main purpose is to explain how the results presented in [CJO+20] can be read in the notation and context in which we introduced approximate span programs in this thesis.

We make several assumptions on the quantum algorithm that we start with, and we refer to the algorithms that satisfy these assumptions as *clean algorithms*.

**8.3.1.** Definition (Clean quantum algorithm). Let $\mathcal{A}$ be a quantum query algorithm that does not perform any intermediate measurements, and acts on $\mathbb{C}^{[n]\times W} = \mathbb{C}^{[n]\times W'\times\{0,1\}}$ with the last register being the answer register. Suppose that the number of gates used by $\mathcal{A}$ is $T$, the number of queries it performs is $Q$, and the initial state has $|0\rangle$ in the answer register, so it can be expressed as $|\Psi_0\rangle = |\psi_0\rangle|0\rangle$ for some $|\psi_0\rangle \in \mathbb{C}^{[n]\times W'}$. Define the final accepting state as $|\Psi_T\rangle := |\psi_0\rangle|1\rangle$. $\mathcal{A}$ is a *clean quantum algorithm* if it satisfies the following properties:

1. *Consistency:* For all inputs $x \in \{0,1\}^n$,

$$|\langle\Psi_T|\Psi_T(x)\rangle|^2 = p_1(x), \qquad \text{and} \qquad |\langle\Psi_T|(I\otimes X)|\Psi_T(x)\rangle|^2 = p_0(x),$$

   where $p_b(x) = \|(I\otimes|b\rangle\langle b|)|\Psi_T(x)\rangle\|^2$ is the probability that $\mathcal{A}$ outputs $b$ on input $x$, $|\Psi_T(x)\rangle$ is the final state of algorithm $\mathcal{A}$ on input $x$, and $X$ denotes the Pauli matrix implementing the logical NOT.
2. *Commutation:* $(I \otimes X)$ commutes with every unitary $U_t$ of the algorithm, where $X$ acts on the answer register.
3. *Query-uniformity:* Two consecutive queries are not more than $\lfloor 3T/Q \rfloor$ time steps apart, and the first and last queries are separated by at most $\lfloor 3T/Q \rfloor$ time steps from the start and the finish of the algorithm, respectively. ◀

In [CJO+20, Section 2.1], we prove that the above assumptions on the form of a quantum algorithm are not really restrictive, as we show that one can take any quantum algorithm $\mathcal{A}$ and turn it into a clean quantum algorithm $\mathcal{A}'$ that computes the same function with success probability that is at least the square

of the success probability of $\mathcal{A}$, uses only one extra qubit, and has a constant multiplicative overhead in the number of queries and gates.

Next, we formalize a way to access the individual operations that a clean quantum algorithm $\mathcal{A}$ performs on any given time step. This results in the definition of the *algorithm access oracles*, as provided below.

**8.3.2.** Definition (Algorithm access oracles). Let $\mathcal{A}$ be a clean quantum algorithm acting on Hilbert space $\mathcal{H}$, with $T$ time steps. Let $\mathcal{S} \subseteq [T]$ be the set of time steps on which $\mathcal{A}$ performs a query to the input, and for all $t \in [T] \setminus \mathcal{S}$, let $U_t$ be the unitary on $\mathcal{H}$ that $\mathcal{A}$ performs at time step $t$. Next, we define three operations:

1. Let $O_{\mathcal{A}}$ be a circuit that acts on $\mathbb{C}^{[T]} \otimes \mathcal{H}$, and implements the operation
$$O_{\mathcal{A}} : |t\rangle |\psi\rangle \mapsto |t\rangle U_t |\psi\rangle .$$
We refer to this operation as the *algorithm oracle*.
2. Let $O_{\mathcal{S}}$ be a circuit that acts on $\mathbb{C}^{[T]}$ and implements the operation
$$O_{\mathcal{S}} : |t\rangle \mapsto \begin{cases} -|t\rangle , & \text{if } t \in \mathcal{S}, \\ |t\rangle , & \text{otherwise.} \end{cases}$$
We refer to this operation as the *query time step oracle*.
3. Let $O_x$ be the oracle acting on $\mathcal{H}$ that is performed by $\mathcal{A}$ on each of the time steps $t \in \mathcal{S}$. We refer to this operation as the *input oracle*. ◀

Now, we define how one can take a clean quantum algorithm $\mathcal{A}$, and turn it into a span program.

**8.3.3.** Definition (Span program compiled from a clean quantum algorithm). Let $\mathcal{A}$ be a clean quantum algorithm acting on the state space $\mathbb{C}^{[n]\times W}$, computing a boolean function $f : \mathcal{D} \to \{0,1\}$ with $\mathcal{D} \subseteq \{0,1\}^n$ and with failure probability at most $\varepsilon > 0$, with $T$ time steps and $Q$ queries performed at the time steps $\mathcal{S} = \{q_1, \ldots, q_Q\} \subseteq [T]$. For notational convenience, let $q_0 = 0$ and $q_{Q+1} = T+1$. For all $\ell \in [Q+1]$, let $B_\ell \subseteq [T+1]$ be the $\ell$th block of contiguous non-query time steps. Let
$$a = \sqrt{\frac{\varepsilon}{2Q+1}}, \qquad \text{and} \qquad M = \max_{\ell \in [Q+1]} \sqrt{|B_\ell|}.$$

Now, define the spaces

$$\forall i \in [n], b \in \{0,1\}, \quad \mathcal{H}_{i,b} = \operatorname{Span}\{|t,b,i,j\rangle : t+1 \in \mathcal{S}, j \in W\},$$
$$\mathcal{H}_{\text{true}} = \operatorname{Span}\{|t,0,i,j\rangle : t+1 \in [T+1] \setminus \mathcal{S}, i \in [n], j \in W\}.$$

Then, for any input $x \in \{0,1\}^n$, let

$$\mathcal{H}(x) = \left(\bigoplus_{i=1}^{n} \mathcal{H}_{i,x_i}\right) \oplus \mathcal{H}_{\text{true}}, \qquad \text{and} \qquad \mathcal{H} = \left(\bigoplus_{\substack{i=1 \\ b\in\{0,1\}}}^{n} \mathcal{H}_{i,b}\right) \oplus \mathcal{H}_{\text{true}}.$$

Additionally, let $\mathcal{Z} = [n] \times W$. For $\ell \in \{2, \ldots, Q\}$, we define the linear map $\Phi_\ell$ from $\mathbb{C}^{\mathcal{Z}}$ to $\mathcal{H}$ as

$$\begin{aligned}\Phi_\ell |\psi\rangle = |q_{\ell-1} - 1\rangle \frac{|-\rangle}{\sqrt{2}} |\psi\rangle + |q_\ell - 1\rangle \frac{|+\rangle}{\sqrt{2}} U_{q_\ell - 1} \cdots U_{q_{\ell-1}+1} |\psi\rangle \\ + \frac{1}{M} \sum_{t=q_{\ell-1}}^{q_\ell - 2} |t\rangle |0\rangle U_t \cdots U_{q_{\ell-1}+1} |\psi\rangle .\end{aligned}$$

Similarly, we define the linear operator $\Phi_{Q+1}$ from $\mathbb{C}^{\mathcal{Z}}$ to $\mathcal{H}$ as

$$\begin{aligned}\Phi_{Q+1} |\psi\rangle = |q_Q - 1\rangle \frac{|-\rangle}{\sqrt{2}} + \frac{1}{M} \sum_{t=q_Q}^{T-1} |t\rangle |0\rangle U_t \cdots U_{q_Q+1} |\psi\rangle \\ + \frac{1}{a} |T\rangle |0\rangle U_T \cdots U_{q_Q+1} |\psi\rangle .\end{aligned}$$

Then, let

$$\mathcal{K} = \bigoplus_{\ell=2}^{Q+1} \Phi_\ell(\mathbb{C}^{\mathcal{Z}}).$$

Finally, let

$$\begin{aligned}|w_0\rangle = \frac{1}{MN} \sum_{t=0}^{q_1-2} |t\rangle |0\rangle U_t \cdots U_1 |\Psi_0\rangle + |q_1 - 1\rangle \frac{|+\rangle}{\sqrt{2}} U_{q_1-1} \cdots U_1 |\Psi_0\rangle \\ + \frac{1}{N(Ca^2+1)} \left[ |q_Q - 1\rangle \frac{|-\rangle}{\sqrt{2}} U_{q_Q+1}^\dagger \cdots U_T^\dagger |\Psi_T\rangle \right. \\ \left. + \frac{1}{M} \sum_{t=q_Q}^{T-1} |t\rangle |0\rangle U_{t+1}^\dagger \cdots U_T^\dagger |\Psi_T\rangle \right] - \frac{Ca}{N(Ca^2+1)} |T\rangle |0\rangle |\Psi_T\rangle ,\end{aligned}$$

where

$$C = \frac{T - q_Q}{M^2} + \frac{1}{2}, \qquad \text{and} \qquad N = \sqrt{\frac{q_1 - 1}{M^2} + \frac{1}{2} + \frac{C}{Ca^2+1}}.$$

Then, let $\mathcal{P}_{\mathcal{A}} = (\mathcal{H}, x \mapsto \mathcal{H}(x), \mathcal{K}, |w_0\rangle)$. We refer to $\mathcal{P}_{\mathcal{A}}$ as *the span program compiled from* $\mathcal{A}$. ◀

Finally, we state the core result from [CJO+20].

**8.3.4.** THEOREM (Implementation of the span program algorithm of $\mathcal{P}_{\mathcal{A}}$). *Let $\mathcal{A}$ be a clean quantum algorithm as defined in Definition 8.3.1, and let $O_{\mathcal{A}}$, $O_{\mathcal{S}}$ and $O_x$ be as in Definition 8.3.2. Next, let $\mathcal{P}_{\mathcal{A}}$ be the span program compiled from $\mathcal{A}$, as defined in Definition 8.3.3. Suppose that $\mathcal{A}$ performs $T$ time steps, $Q$ queries, acts on $k$ qubits, and computes a boolean function $f : \mathcal{D} \to \{0,1\}$, with $\mathcal{D} \subseteq \{0,1\}^n$ and with failure probability at most $\varepsilon < 1/10$. Let $\delta = 1/(3(2S+1))$. Then,*

1. *$\mathcal{P}_{\mathcal{A}}$ is well-defined.*
2. *$\neg\mathcal{P}_{\mathcal{A}}$ $\delta$-approximates $\neg f$.*
3. *$\widetilde{W}_-(\neg\mathcal{P}_{\mathcal{A}}, \delta) = \mathcal{O}(Q)$.*
4. *$\widetilde{W}_+(\neg\mathcal{P}_{\mathcal{A}}, \delta) = \mathcal{O}(Q)$.*
5. *$\lambda(\neg\mathcal{P}_{\mathcal{A}}, \delta) \leq 5\varepsilon$.*

*Moreover, we can implement the approximate span program algorithm from [IJ19], i.e., Theorem 8.2.1, applied to $\neg\mathcal{P}_{\mathcal{A}}$ with*

6. *$\mathcal{O}(Q\log(Q))$ calls to $O_x$.*
7. *$\mathcal{O}(T\log(Q))$ calls to $O_{\mathcal{A}}$ and $O_{\mathcal{S}}$.*
8. *$\mathcal{O}(T\,\mathrm{polylog}(T))$ additional gates.*
9. *$\mathcal{O}(\mathrm{polylog}(T) + k^{o(1)})$ auxiliary qubits.*

*If we additionally require that the error probability of $\mathcal{A}$ is $o(1/Q^2)$, then the $\log(Q)$ factors and the $k^{o(1)}$ term can be removed. Similarly, if we assume that $T = k^{1+\Omega(1)}$, we can also remove the $k^{o(1)}$ term.*

**Proof sketch:**
For claim 1, we need to check that $|w_0\rangle$ is a unit vector. This follows from [CJO+20, Lemma 19].

For claim 2, we must first check that the span program $\mathcal{P}_{\mathcal{A}}$ is actually the same as the one constructed in [CJO+20]. This follows from [CJO+20, Definition 14, Lemma 18 and Lemma 19].

Next, [CJO+20, Lemma 17] states that $\mathcal{P}_{\mathcal{A}}$ *positively* $5\varepsilon$*-approximates* $f$, where the concept of positive approximation is defined in [CJO+20, Definition 6], or equivalently in [IJ19]. From the discussion in [IJ19], we additionally observe that negating a span program turns positive approximation into negative approximation. We also observe that the definition of negative approximation in [IJ19] coincides with the concept of $\delta$-approximation in Definition 8.1.1, if we choose $\delta$ equal to the reciprocal of the maximum negative witness size amongst the negative inputs. Thus, indeed, we find that $\neg\mathcal{P}_{\mathcal{A}}$ $\delta$-approximates $\neg f$, where we must choose

$$\delta = \max_{x\in(\neg f)^{-1}(0)} 1/w_-(x, \neg\mathcal{P}_{\mathcal{A}}) = \max_{x\in f^{-1}(1)} 1/w_+(x, \mathcal{P}_{\mathcal{A}}) = 1/(3(2Q+1)),$$

where the final equality is [CJO+20, Lemma 16].

Claims 3, 4 and 5 follow from [CJO+20, Lemma 16 and Lemma 17], and claims 6–9 follow from [CJO+20, Theorem 20]. □

The full proof of Theorem 8.3.4 is extremely tedious, not so insightful, and would easily stretch $\sim 100$ pages. Therefore, we omit it in this thesis, and refer the interested reader to [CJO+20].

Interestingly, subsequent insights have revealed that the construction in Definition 8.3.3 can be significantly improved, and in particular some back-of-the-envelope calculations indicated that with the improved construction, the $k^{o(1)}$ term and the factors $\log(Q)$ can be dropped in Theorem 8.3.4. One could try and formalize these improvements to obtain a cleaner statement of Theorem 8.3.4.

A downside of the above construction is that it uses an approximate span program rather than an exact one, and as such lacks the potential to be plugged into the nice composition properties proved in Chapter 7. This limits the applicability of the result outlined in this section.

To illustrate the above argument, consider the following possible application of the above construction. Suppose that we have $n$ quantum algorithms computing boolean functions $f^{(1)}, \ldots, f^{(n)}$ on a common domain, and we wanted to compute some logical formula of the function outcomes, e.g., $f^{(1)} \wedge \cdots \wedge f^{(n)}$. One possible way to achieve this is to take the quantum algorithms, turn them into approximate span programs with the construction in Theorem 8.3.4, compose them using a logical composition construction, and then turn the composed span program back into a quantum algorithm.

The issue is, however, that if one tries to port the logical composition constructions, e.g., Theorems 7.1.4 and 7.1.6, to the approximate setting, then the approximation factors $\lambda(\mathcal{P}, \delta)$ add. Hence, using Theorem 8.3.4 will cause the resulting quantum algorithm to have an approximation factor $\lambda(\mathcal{P}, \delta)$ that is the sum of the failure probabilities of the individual quantum algorithms. Thus, the composition result using approximate span programs requires logarithmic overhead in boosting the success probability before performing composition, and consequently we don't achieve anything over performing the composition directly on the level of the algorithms itself.

This is a negative observation regarding the use of *approximate* span programs for performing compositions. Hence, it underlines the fact that quantum algorithms computing boolean functions with bounded error don't compose well, and supports the argument that *exact* span programs and dual adversary bound solutions computing are in general more flexible and portable building blocks in quantum algorithm design.

# Chapter 9

# Discussion

In this part, we saw a self-contained introduction into the theory of span programs. We end this part with some closing remarks on the subject.

The first thing to note is that constructing span programs is not an easy task in general. Theoretically, through the connection with the dual adversary bound, we should always be able to generate an optimal span program computing $f : \mathcal{D} \to \{0, 1\}$, with $\mathcal{D} \subseteq \{0, 1\}^n$ by solving the dual adversary bound for $f$, and converting the resulting solution into a span program. However, the number of degrees of freedom in the resulting SDP grows quadratically in the number of elements in the domain, which itself is typically exponential in $n$. In practice, then, this approach is typically not very useful whenever the domain size is larger than approximately 100 inputs.

Secondly, the composition constructions from Sections 7.1 and 7.2 make span programs a lot more well-behaved compared to the bounded-error quantum query algorithms they generate. After all, as we argued at the end of Section 7.1.1, we can use the exact notion of span programs to merge many small solutions of computational problems into bigger ones, whilst avoiding having to mitigate errors that occur in the intermediate stages of the computation. As such, span programs are much more flexible objects than bounded-error quantum algorithms.

We note that this ease of composition, which makes the theory of exact span programs very appealing, is lost when we switch to approximate span programs. Intuitively speaking, since approximate span programs are inherently a little bit imprecise, they too suffer from the necessity to mitigate these imperfections when one tries to compose them. This also means that using the black-box conversion from bounded-error quantum algorithms into approximate span programs, as briefly hinted at in Section 8.3, comes with inherent limitations, and does not enable the full potential of span programs on any computational problem for which we have an efficient quantum algorithm available.

In the long run, we can expect that quantum algorithms that solve real-world problems will have many different components, and stitching together all these

building blocks efficiently could potentially result in much more efficient quantum algorithms for problems of interest. Recent papers on resource estimation for specific quantum algorithms provide very suitable test cases for this proposition. As such, it would be very interesting to see if span program composition results can be used to improve the results obtained in these works. We explicitly mention a recent work on Louvain's algorithm for community detection as an exemplary instance [CFN+22].

As a final note, we mention that one appealing future direction of research is an attempt to find span program constructions that compute all symmetric functions with optimal complexity. In this chapter we briefly mentioned how we can compute functions like threshold, exact-weight and parity, but there are still many symmetric functions that we don't know optimal span program constructions for. In an effort to categorize the current progress, we present Table 9.1. It is a very nice direction of future research to try and fill this table with more rows, and gradually understand better how we can construct and optimally compose span programs that compute symmetric functions.

| Function | Positive inputs | Optimal adversary value | Note |
|---|---|---|---|
| AND | $\|x\| = n$ | $\sqrt{n}$ | |
| OR | $\|x\| \neq 0$ | $\sqrt{n}$ | |
| Parity | $\|x\|$ odd | $n$ | |
| Threshold-$k$ | $\|x\| \geq k$ | $\sqrt{k(n-k+1)}$ | $(0 \leq k \leq n)$ |
| Exact-weight-$k$ | $\|x\| = k$ | $\sqrt{n+2k(n-k)}$* | $(0 \leq k \leq n)$ |
| Interval-1-$k$ | $1 \leq \|x\| \leq k$ | $\sqrt{n/k+(k+1)(n-k)}$* | $(0 < k < n)$ |

Table 9.1: Current progress on categorizing optimal span program constructions for symmetric functions $f : \{0,1\}^n \to \{0,1\}$. The second column defines the particular function $f$, i.e., $f(x) = 1$ if and only if the expression in the second column is true. The expressions labeled by * are not formally proved yet – we have merely gathered strong numerical evidence.

# Bibliography


[Aar20] Scott Aaronson. "Shadow Tomography of Quantum States". In: *SIAM Journal on Computing* 49.5 (2020). arXiv:1711.01053.

[ABP19] Srinivasan Arunachalam, Jop Briët, and Carlos Palazuelos. "Quantum Query Algorithms Are Completely Bounded Forms". In: *SIAM Journal on Computing* 48.3 (2019), pp. 903–925. Appeared earlier in the Proceedings of the 9th Innovations in Theoretical Computer Science Conference, (ITCS 2018), arXiv:1711.07285.

[AdFD+03] Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. "An Introduction to MCMC for Machine Learning". In: *Machine Learning* 50.1-2 (2003), pp. 5–43.

[AHN+21] Srinivasan Arunachalam, Vojtěch Havlíček, Giacomo Nannicini, Kristan Temme, and Pawel Wocjan. "Simpler (Classical) and Faster (Quantum) Algorithms for Gibbs Partition Functions". In: *IEEE International Conference on Quantum Computing and Engineering, (QCE 2021)*. 2021, pp. 112–122. arXiv:2009.11270.

[Amb02] Andris Ambainis. "Quantum Lower Bounds by Quantum Arguments". In: *Journal of Computer and System Sciences* 64.4 (2002), pp. 750–767. Appeared earlier in the Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing (STOC 2000), arXiv:quant-ph/0002066.

[Amb06] Andris Ambainis. "Polynomial degree vs. quantum query complexity". In: *Journal of Computer System Sciences* 72.2 (2006), pp. 220–238. Appeared earlier in the Proceedings of the 44th Symposium on Foundations of Computer Science (FOCS 2003), arXiv:quant-ph/0305028.

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. "The Space Complexity of Approximating the Frequency Moments". In: *Journal of Computer and Systems Sciences* 58.1 (1999), pp. 137–147.

[Āri15] Agnis Āriņš. "Span-Program-Based Quantum Algorithms for Graph Bipartiteness and Connectivity". In: *Proceedings of the 10th International Doctoral Workshop on Mathematical and Engineering Methods in Computer Science, (MEMICS 2015)*. 2015, pp. 35–41. arXiv:1510.07825.

[AS04] Scott Aaronson and Yaoyun Shi. "Quantum lower bounds for the collision and the element distinctness problems". In: *Journal of the ACM* 51.4 (2004), pp. 595–605. Appeared earlier in the Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS 2002), arXiv:quant-ph/0112086.

[ASS21] Atithi Acharya, Siddhartha Saha, and Anirvan M. Sengupta. *Informationally complete POVM-based shadow tomography*. 2021. arXiv:2105.05992.

[AW99] Daniel S. Abrams and Colin P. Williams. *Fast quantum algorithms for numerical integrals and stochastic processes*. 1999. arXiv:quant-ph/9908083.

[BBB+97] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh V. Vazirani. "Strengths and Weaknesses of Quantum Computing". In: *SIAM Journal on Computing* 26.5 (1997), pp. 1510–1523. arXiv:quant-ph/9701001.

[BBC+01] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. "Quantum lower bounds by polynomials". In: *Journal of the ACM* 48.4 (2001), pp. 778–797. Appeared earlier in the Proceedings of the 39th Annual Symposium on Foundations of Computer Science, (FOCS 1998), arXiv:quant-ph/9802049.

[BBH+98] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. "Tight bounds on quantum searching". In: *Fortschritte der Physik: Progress of Physics* 46.4-5 (1998), pp. 493–505.

[BDG+11] Gilles Brassard, Frédéric Dupuis, Sébastien Gambs, and Alain Tapp. *An optimal quantum algorithm to approximate the mean and its application for approximating the median of a set of points over an arbitrary distance*. 2011. arXiv:1106.4267.

[Bel12] Aleksandrs Belovs. "Learning-Graph-Based Quantum Algorithm for k-Distinctness". In: *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science, (FOCS 2012)*. 2012, pp. 207–216. arXiv:1205.1534.

[Bel14] Aleksandrs Belovs. "Applications of the Adversary Method in Quantum Query Algorithms". University of Latvia, Riga, Latvia, 2014. arXiv:1402.3858.

[Bel15] Aleksandrs Belovs. *Variations on quantum adversary*. 2015. arXiv: 1504.06943.

[Ben80] Paul Benioff. "The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines". In: *Journal of statistical physics* 22.5 (1980), pp. 563–591.

[Bes86] Julian Besag. "On the statistical analysis of dirty pictures". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 48.3 (1986), pp. 259–279.

[BH10] Kurt Binder and Dieter W. Heermann. *Monte Carlo Simulation in Statistical Physics*. Springer Berlin Heidelberg, 2010.

[BHM+02] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. "Quantum amplitude amplification and estimation". In: *Contemporary Mathematics* 305 (2002), pp. 53–74. arXiv : quant - ph/0005055.

[Blu67] Manuel Blum. "A Machine-Independent Theory of the Complexity of Recursive Functions". In: *Journal of the ACM* 14.2 (1967), pp. 322–336.

[BNR+07] Harry Buhrman, Ilan Newman, Hein Röhrig, and Ronald de Wolf. "Robust Polynomials and Quantum Algorithms". In: *Theory of Computing Systems* 40.4 (2007), pp. 379–395. arXiv : quant - ph/0309220.

[BR12] Aleksandrs Belovs and Ben W. Reichardt. "Span Programs and Quantum Algorithms for st-Connectivity and Claw Detection". In: *Proceedings of the 20th Annual European Symposium on Algorithms (ESA 2012)*. 2012, pp. 193–204. arXiv:1203.2603.

[BSS14] Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava. "Twice-Ramanujan Sparsifiers". In: *SIAM Review* 56.2 (2014), pp. 315–334. arXiv:0808.0163.

[BSV+08] Ivona Bezáková, Daniel Stefankovic, Vijay V. Vazirani, and Eric Vigoda. "Accelerating Simulated Annealing for the Permanent and Combinatorial Counting Problems". In: *SIAM Journal on Computing* 37.5 (2008), pp. 1429–1454.

[BT19] Salman Beigi and Leila Taghavi. "Span program for non-binary functions". In: *Quantum Information & Computing* 19.9&10 (2019), pp. 760–792. arXiv:1805.02714.

[BT20] Salman Beigi and Leila Taghavi. "Quantum Speedup Based on Classical Decision Trees". In: *Quantum* 4 (2020), p. 241. arXiv: 1905.13095.

[BTT22] Salman Beigi, Leila Taghavi, and Artin Tajdini. "Time-and Query-optimal Quantum Algorithms Based on Decision Trees". In: *ACM Transactions on Quantum Computing* 3.4 (2022), pp. 1–31. arXiv: 2105.08309.

[Cat04] Olivier Catoni. *Statistical learning theory and stochastic optimization: Ecole d'Eté de Probabilités de Saint-Flour, XXXI-2001*. Vol. 1851. Springer Science & Business Media, 2004.

[CBG21] Arjan Cornelissen, Johannes Bausch, and András Gilyén. *Scalable Benchmarks for Gate-Based Quantum Computers*. 2021. arXiv : 2104.10698.

[CCH+19] Shouvanik Chakrabarti, Andrew M. Childs, Shih-Han Hung, Tongyang Li, Chunhao Wang, and Xiaodi Wu. *Quantum algorithm for estimating volumes of convex bodies*. 2019. arXiv:1908.03903.

[CF02] Ronald Cramer and Serge Fehr. "Optimal Black-Box Secret Sharing over Arbitrary Abelian Groups". In: *22nd Annual International Cryptology Conference (CRYPTO 2002)*. Vol. 2442. Lecture Notes in Computer Science. Springer, 2002, pp. 272–287.

[CFN+22] Chris Cade, Marten Folkertsma, Ido Niesen, and Jordi Weggemans. *Quantum Algorithms for Community Detection and their Empirical Run-times*. 2022. arXiv:2203.06208.

[CH22] Arjan Cornelissen and Yassine Hamoudi. *A Sublinear-Time Quantum Algorithm for Approximating Partition Functions*. 2022. Accepted to: Symposium on Discrete Algorithms (SODA 2023) and the 26th Conference on Quantum Information Processing (QIP 2023). arXiv:2207.08643.

[Chi22] Andrew Childs. *Lecture Notes on Quantum Algorithms*. 2022. http: //www.cs.umd.edu/~amchilds/qa/qa.pdf.

[CHJ22] Arjan Cornelissen, Yassine Hamoudi, and Sofiène Jerbi. "Near-optimal Quantum algorithms for multivariate mean estimation". In: *54th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2022)*. ACM, 2022, pp. 33–43. Presented at: 25th Annual Conference on Quantum Information Processing (QIP 2022). arXiv:2111.09787.

[CHL+22] Sitan Chen, Brice Huang, Jerry Li, Allen Liu, and Mark Sellke. *Tight Bounds for State Tomography with Incoherent Measurements*. 2022. arXiv.2206.05265.

[CJ21] Arjan Cornelissen and Sofiene Jerbi. *Quantum algorithms for multivariate Monte Carlo estimation*. 2021. arXiv:2107.03410.

[CJO+20] Arjan Cornelissen, Stacey Jeffery, Māris Ozols, and Alvaro Piedrafita. "Span Programs and Quantum Time Complexity". In: *Proceedings of the 45th International Symposium on Mathematical Foundations of Computer Science, (MFCS 2020)*. Vol. 170. 2020, 26:1–26:14. arXiv:2005.01323.

[CMB18] Chris Cade, Ashley Montanaro, and Aleksandrs Belovs. "Time and space efficient quantum algorithms for detecting cycles and testing bipartiteness". In: *Quantum Information & Computation* 18.1&2 (2018), pp. 18–50. arXiv:1610.00581.

[CMO+21] Arjan Cornelissen, Nikhil S. Mande, Maris Ozols, and Ronald de Wolf. *Exact quantum query complexity of computing Hamming weight modulo powers of two and three*. 2021. arXiv:2112.14682.

[CMP22] Arjan Cornelissen, Nikhil S. Mande, and Subhasree Patro. *Improved Quantum Query Upper Bounds Based on Classical Decision Trees*. 2022. Presented at: 17th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2022). Accepted to: 42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2022). arXiv:2203.02968.

[Coo71] Stephen A. Cook. "The Complexity of Theorem-Proving Procedures". In: *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, (STOC 1971)*. ACM, 1971, pp. 151–158.

[Cor19] Arjan Cornelissen. *Quantum gradient estimation of Gevrey functions*. 2019. arXiv:1909.13528.

[DF88] Martin E. Dyer and Alan M. Frieze. "On the Complexity of Computing the Volume of a Polyhedron". In: *SIAM Journal on Computing* 17.5 (1988), pp. 967–974.

[DF91] Martin Dyer and Alan Frieze. "Computing the volume of convex bodies: a case where randomness provably helps". In: *Probabilistic combinatorics and its applications* 44.123-170 (1991), pp. 0754–68052.

[DFK91] Martin E. Dyer, Alan M. Frieze, and Ravi Kannan. "A Random Polynomial Time Algorithm for Approximating the Volume of Convex Bodies". In: *Journal of the ACM* 38.1 (1991), pp. 1–17.

[DKW19] Kai DeLorenzo, Shelby Kimmel, and R. Teal Witter. "Applications of the Quantum Algorithm for st-Connectivity". In: *14th Conference on the Theory of Quantum Computation, Communication and Cryptography, (TQC 2019)*. Vol. 135. 2019, 6:1–6:14. arXiv:1904.05995.

[dWol22] Ronald de Wolf. *Quantum Computing: Lecture Notes*. 2022. arXiv:1907.09415.

[DY00] Adrian Dragulescu and Victor M Yakovenko. "Statistical mechanics of money". In: *The European Physical Journal B-Condensed Matter and Complex Systems* 17.4 (2000), pp. 723–729. arXiv:cond-mat/0001432.

[EH89] Andrzej Ehrenfeucht and David Haussler. "Learning Decision Trees from Random Examples". In: *Information and Computation* 82.3 (1989), pp. 231–246.

[Fey82] Richard P Feynman. "Simulating Physics with Computers". In: *International Journal of Theoretical Physics* 21.6/7 (1982).

[FG06] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006.

[FGG+99] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. "Bound on the number of functions that can be distinguished with k quantum queries". In: *Physical Review A* 60.6 (1999), p. 4331. arXiv:quant-ph/9901012.

[FV17] Sacha Friedli and Yvan Velenik. *Statistical mechanics of lattice systems: a concrete mathematical introduction*. Cambridge University Press, 2017.

[GAW19] András Gilyén, Srinivasan Arunachalam, and Nathan Wiebe. "Optimizing quantum optimization algorithms via faster quantum gradient computation". In: *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA 2019)*. 2019. arXiv:1711.00465.

[Geo11] Hans-Otto Georgii. "Gibbs measures and phase transitions". In: *Gibbs Measures and Phase Transitions*. de Gruyter, 2011.

[GG84] Stuart Geman and Donald Geman. "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6.6 (1984), pp. 721–741.

[GL20] András Gilyén and Tongyang Li. "Distributional Property Testing in a Quantum World". In: *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*. Vol. 151. LIPIcs. 2020, 25:1–25:19. arXiv:1902.00814.

[Gla03] Paul Glasserman. *Monte Carlo Methods in Financial Engineering*. Springer New York, 2003.

[GLF+10] David Gross, Yi-Kai Liu, Steven T Flammia, Stephen Becker, and Jens Eisert. "Quantum state tomography via compressed sensing". In: *Physical review letters* 105.15 (2010), p. 150401. arXiv:0909.3304.

[Gro96] Lov K. Grover. "A Fast Quantum Mechanical Algorithm for Database Search". In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*. Ed. by Gary L. Miller. ACM, 1996, pp. 212–219. arXiv:quant-ph/9605043.

[Gro98] Lov K. Grover. "A Framework for Fast Quantum Mechanical Algorithms". In: *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing (STOC 1998)*. ACM, 1998, pp. 53–62. arXiv:quant-ph/9711043.

[GSL+19] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. "Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC 2019)*. 2019, pp. 193–204. arXiv:1806.01838.

[Ham21] Yassine Hamoudi. "Quantum Sub-Gaussian Mean Estimator". In: *29th Annual European Symposium on Algorithms, (ESA 2021)*. Vol. 204. LIPIcs. 2021, 50:1–50:17. arXiv:2108.12172.

[Hei02] Stefan Heinrich. "Quantum Summation with an Application to Integration". In: *Journal of Complexity* 18.1 (2002), pp. 1–50. arXiv:quant-ph/0105116.

[Hei04] Stefan Heinrich. "On the Power of Quantum Algorithms for Vector Valued Mean Computation". In: *Monte Carlo Methods Applications* 10.3-4 (2004), pp. 297–310.

[Hei27] Werner Heisenberg. "Über den anschaulichen Inhalt der quantentheoretischen Kinematik und Mechanik". In: *Zeitschrift für Physik* 43.3 (1927), pp. 172–198.

[HHJ+17] Jeongwan Haah, Aram W. Harrow, Zhengfeng Ji, Xiaodi Wu, and Nengkun Yu. "Sample-Optimal Tomography of Quantum States". In: *IEEE Transactions on Information Theory* 63.9 (2017), pp. 5628–5641. arXiv:1508.01797.

[HHL09] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum Algorithm for Linear Systems of Equations". In: *Physical Review Letters* 103 (15 2009), p. 150502. arXiv:0811.3171.

[HKP20] Hsin-Yuan Huang, Richard Kueng, and John Preskill. "Predicting many properties of a quantum system from very few measurements". In: *Nature Physics* 16 (2020), pp. 1050–1057. arXiv:2002.08953.

[HLŠ07] Peter Høyer, Troy Lee, and Robert Špalek. "Negative weights make adversaries stronger". In: *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC 2007)*. 2007, pp. 526–535. arXiv:quant-ph/0611054.

[HLY+22] Hong-Ye Hu, Ryan LaRose, Yi-Zhuang You, Eleanor Rieffel, and Zhihui Wang. *Logical shadow tomography: Efficient estimation of error-mitigated observables*. 2022. arXiv:2203.07263.

[HM19] Yassine Hamoudi and Frédéric Magniez. "Quantum Chebyshev's Inequality and Applications". In: *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*. Vol. 132. 2019, 69:1–69:16. arXiv:1807.06456.

[Hop20] Samuel B. Hopkins. "Mean estimation with sub-Gaussian rates in polynomial time". In: *The Annals of Statistics* 48.2 (2020), pp. 1193–1213.

[Hub15] Mark Huber. "Approximation algorithms for the normalizing constant of Gibbs distributions". In: *The Annals of Applied Probability* 25.2 (2015), pp. 974–985. arXiv:1206.2689.

[HW20] Aram W. Harrow and Annie Y. Wei. "Adaptive Quantum Simulated Annealing for Bayesian Inference and Estimating Partition Functions". In: *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, (SODA 2020)*. 2020, pp. 193–212. arXiv:1907.09965.

[HWM+21] William J Huggins, Kianna Wan, Jarrod McClean, Thomas E O'Brien, Nathan Wiebe, and Ryan Babbush. "Nearly Optimal Quantum Algorithm for Estimating Multiple Expectation Values". In: (2021). arXiv:2111.09283.

[IJ19] Tsuyoshi Ito and Stacey Jeffery. "Approximate Span Programs". In: *Algorithmica* 81.6 (2019), pp. 2158–2195. arXiv:1507.00432.

[Jef14] Stacey Jeffery. "Frameworks for Quantum Algorithms". PhD thesis. University of Waterloo, Ontario, Canada, 2014.

[Jef20] Stacey Jeffery. "Span Programs and Quantum Space Complexity". In: *Proceedings of the 11th Innovations in Theoretical Computer Science Conference, (ITCS 2020)*. Vol. 151. 2020, 4:1–4:37. arXiv:1908.04232.

[Jer95] Mark Jerrum. "A Very Simple Algorithm for Estimating the Number of k-Colorings of a Low-Degree Graph". In: *Random Struct. Algorithms* 7.2 (1995), pp. 157–166.

[JJK+18] Michael Jarret, Stacey Jeffery, Shelby Kimmel, and Alvaro Piedrafita. "Quantum Algorithms for Connectivity and Related Problems". In: *Proceedings of the 26th Annual European Symposium on Algorithms, (ESA 2018)*. 2018, pp. 1–13. arXiv:1804.10591.

[JK17] Stacey Jeffery and Shelby Kimmel. "Quantum Algorithms for Graph Connectivity and Formula Evaluation". In: *Quantum* 1 (2017), p. 26. arXiv:1704.00765.

[Jor05] Stephen P Jordan. "Fast quantum algorithm for numerical gradient estimation". In: *Physical review letters* 95.5 (2005), p. 050501. arXiv:quant-ph/0405146.

[Jor75] Camille Jordan. "Essai sur la géom'etrie à $n$ dimensions". In: *Bulletin de la Société Mathématique de France* 3 (1875), pp. 103–174.

[JS89] Mark Jerrum and Alistair Sinclair. "Approximating the permanent". In: *SIAM journal on computing* 18.6 (1989), pp. 1149–1178.

[JS93] Mark Jerrum and Alistair Sinclair. "Polynomial-Time Approximation Algorithms for the Ising Model". In: *SIAM Journal on Computing* 22.5 (1993), pp. 1087–1116.

[JS96] Mark Jerrum and Alistair Sinclair. "The Markov chain Monte Carlo method: an approach to approximate counting and integration". In: *Approximation Algorithms for NP-hard problems, PWS Publishing* (1996).

[JSV04] Mark Jerrum, Alistair Sinclair, and Eric Vigoda. "A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries". In: *Journal of the ACM* 51.4 (2004), pp. 671–697.

[JVV86] Mark Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. "Random Generation of Combinatorial Structures from a Uniform Distribution". In: *Theoretical Computer Science* 43 (1986), pp. 169–188.

[Kit96] Alexei Y. Kitaev. "Quantum measurements and the Abelian Stabilizer Problem". In: *Electron. Colloquium Comput. Complex.* TR96-003 (1996). arXiv:quant-ph/9511026.

[Kit97] Alexei Y. Kitaev. "Quantum computations: algorithms and error correction". In: *Russian Mathematical Surveys* 52.6 (1997), p. 1191.

[KO22] Robin Kothari and Ryan O'Donnell. *Mean estimation when you have the source code; or, quantum Monte Carlo methods*. 2022. arXiv.2208.07544.

[Kol18] Vladimir Kolmogorov. "A Faster Approximation Algorithm for the Gibbs Partition Function". In: *Conference On Learning Theory, (COLT 2018)*. Vol. 75. Proceedings of Machine Learning Research. PMLR, 2018, pp. 228–249. arXiv:1608.04223.

[Kot14] Robin Kothari. "An optimal quantum algorithm for the oracle identification problem". In: *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014)*. Vol. 25. 2014, pp. 482–493. arXiv:1311.7685.

[KW93] Mauricio Karchmer and Avi Wigderson. "On Span Programs". In: *Proceedings of the Eigth Annual Structure in Complexity Theory Conference*. 1993, pp. 102–111.

[LdW21] Noah Linden and Ronald de Wolf. *Average-Case Verification of the Quantum Fourier Transform Enables Worst-Case Phase Estimation*. 2021. arXiv:2109.10215.

[Lev73] Леонид Анатольевич Левин (Leonid Anatolievich Levin). "Универсальные задачи перебора (Universal sequential search problems)". In: *Проблемы передачи информации (Problemy peredachi informatsii)* 9.3 (1973), pp. 115–116.

[LL16] Cedric Yen-Yu Lin and Han-Hsuan Lin. "Upper Bounds on Quantum Query Complexity Inspired by the Elitzur–Vaidman Bomb Tester". In: *Theory of Computing* 12.1 (2016), pp. 1–35. arXiv:1410.0932.

[Llo96] Seth Lloyd. "Universal quantum simulators". In: *Science* 273.5278 (1996), pp. 1073–1078.

[LM19] Gábor Lugosi and Shahar Mendelson. "Mean Estimation and Regression Under Heavy-Tailed Distributions: A Survey". In: *Foundations of Computational Mathematics* 19.5 (2019), pp. 1145–1190. arXiv:1906.04280.

[LMR+11] Troy Lee, Rajat Mittal, Ben W. Reichardt, Robert Špalek, and Mario Szegedy. "Quantum Query Complexity of State Conversion". In: *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science, (FOCS 2011)*. 2011, pp. 344–353. arXiv:1011.3020.

[LV06] László Lovász and Santosh S. Vempala. "Simulated annealing in convex bodies and an $O^*(n^4)$ volume algorithm". In: *Journal of Computer System Sciences* 72.2 (2006), pp. 392–417.

[Man80] Юрий Манин (Yuri Manin). *Вычислимое и невычислимое (Computable and Noncomputable)*. Советское радио (Sovietskoe Radio), 1980.

[MNR+11] Frédéric Magniez, Ashwin Nayak, Jérémie Roland, and Miklos Santha. "Search via Quantum Walk". In: *SIAM Journal on Computing* 40.1 (2011), pp. 142–164. arXiv:quant-ph/0608026.

[Mon15] Ashley Montanaro. "Quantum speedup of Monte Carlo methods". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 471.2181 (2015), p. 20150301. arXiv:1504.06987.

[MS13] Elchanan Mossel and Allan Sly. "Exact thresholds for Ising–Gibbs samplers on general graphs". In: *The Annals of Probability* 41.1 (2013), pp. 294–328. arXiv:0903.2906.

[MW05] Chris Marriott and John Watrous. "Quantum Arthur-Merlin games". In: *Computional Complexity* 14.2 (2005), pp. 122–152. arXiv:cs/0506068.

[NC00] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[NNP04] Ventzislav Nikov, Svetla Nikova, and Bart Preneel. "On the Size of Monotone Span Programs". In: *4th International Conference in Security in Communication Networks (SCN 2004)*. Vol. 3352. Lecture Notes in Computer Science. Springer, 2004, pp. 249–262.

[NY83] Arkadij Semenovič Nemirovskij and David Borisovich Yudin. "Problem complexity and method efficiency in optimization". In: (1983).

[ORR13] Maris Ozols, Martin Roetteler, and Jérémie Roland. "Quantum rejection sampling". In: *ACM Transactions on Computation Theory* 5.3 (2013), 11:1–11:33. arXiv:1103.2774.

[OW16] Ryan O'Donnell and John Wright. "Efficient quantum tomography". In: *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, (STOC 2016)*. ACM, 2016, pp. 899–912. arXiv:1508.01907.

[Rei09] Ben W. Reichardt. "Span Programs and Quantum Query Complexity: The General Adversary Bound Is Nearly Tight for Every Boolean Function". In: *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2009)*. 2009, pp. 544–551. arXiv:0904.2759.

[Rei10] Ben Reichardt. "Span programs and quantum query algorithms". In: *Electronic Colloquium on Computational Complexity* TR10-110 (2010).

[Rei11] Ben W. Reichardt. "Reflections for quantum query algorithms". In: *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2011)*. 2011, pp. 560–569. arXiv:1005.1601.

[RŠ12] Ben W. Reichardt and Robert Špalek. "Span-Program-Based Quantum Algorithm for Evaluating Formulas". In: *Theory of Computing* 8.1 (2012), pp. 291–319. arXiv:0710.2630.

[RV10] Mark Rudelson and Roman Vershynin. "Non-asymptotic theory of random matrices: extreme singular values". In: *Proceedings of the International Congress of Mathematicians 2010 (ICM 2010) (In 4 Volumes) Vol. I: Plenary Lectures and Ceremonies Vols. II–IV: Invited Lectures*. 2010, pp. 1576–1602. arXiv:1003.2990.

[RV13] Mark Rudelson and Roman Vershynin. "Hanson-wright inequality and sub-gaussian concentration". In: *Electronic Communications in Probability* 18 (2013), pp. 1–9. https://arxiv.org/abs/1306.2872.

[Sho97] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Journal on Computing* 26.5 (1997), pp. 1484–1509. arXiv:quant-ph/9508027.

[Sin82] Ya Sinai. "Theory of phase transitions: rigorous results". In: *International series in natural philosophy* 108 (1982).

[SVV09] Daniel Stefankovic, Santosh S. Vempala, and Eric Vigoda. "Adaptive simulated annealing: A near-optimal connection between sampling and counting". In: *Journal of the ACM* 56.3 (2009), 18:1–18:36. arXiv:cs/0612058.

[Sze04] Mario Szegedy. "Quantum Speed-Up of Markov Chain Based Algorithms". In: *Proceedings of the 45th Symposium on Foundations of Computer Science (FOCS 2004)*. 2004, pp. 32–41.

[Tag22] Leila Taghavi. "Simplified quantum algorithm for the oracle identification problem". In: *Quantum Machine Intelligence* 4.2 (2022), pp. 1–7. arXiv:2109.03902.

[Ter99] Barbara Terhal. "Quantum Algorithms and Quantum Entanglement". University of Amsterdam, 1999.

[TOV+11] Kristan Temme, Tobias J Osborne, Karl G Vollbrecht, David Poulin, and Frank Verstraete. "Quantum metropolis sampling". In: *Nature* 471.7336 (2011), pp. 87–90. arXiv:0911.3635.

[Tro15] Joel A. Tropp. "An Introduction to Matrix Concentration Inequalities". In: *Foundations and Trends in Machine Learning* 8.1-2 (2015), pp. 1–230. arXiv:1501.01571.

[Tur37] Alan M. Turing. "On computable numbers, with an application to the Entscheidungsproblem". In: *Proc. London Math. Soc.* s2-42.1 (1937), pp. 230–265.

[vACG+22] Joran van Apeldoorn, Arjan Cornelissen, András Gilyén, and Giacomo Nannicini. *Quantum tomography using state-preparation unitaries*. 2022. Accepted to: Symposium on Discrete Algorithms (SODA 2023) and the 26th Conference on Quantum Information Processing (QIP 2023). `arXiv:2207.08800`.

[Val79] Leslie G. Valiant. "The Complexity of Computing the Permanent". In: *Theoretical Computational Science* 8 (1979), pp. 189–201.

[vApe21] Joran van Apeldoorn. "Quantum Probability Oracles & Multidimensional Amplitude Estimation". In: *16th Conference on the Theory of Quantum Computation, Communication and Cryptography, (TQC 2021)*. Vol. 197. LIPIcs. 2021, 9:1–9:11.

[VC72] J.P. Valleau and D.N. Card. "Monte Carlo estimation of the free energy by multistage sampling". In: *The Journal of Chemical Physics* 57.12 (1972), pp. 5457–5462.

[WA08] Pawel Wocjan and Anura Abeyesinghe. "Speedup via quantum sampling". In: *Physical Review A* 78.4 (2008), p. 042336. `arXiv:0804.4259`.

[WCN+09] Pawel Wocjan, Chen-Fu Chiang, Daniel Nagaj, and Anura Abeyesinghe. "Quantum algorithm for approximating partition functions". In: *Physical Review A* 80.2 (2009), p. 022340.

[YLC14] Theodore J. Yoder, Guang Hao Low, and Isaac L. Chuang. "Fixed-point quantum search with an optimal number of queries". In: *Physical review letters* 113.21 (2014), p. 210501. `arXiv:1409.3305`.

[Yue22] Henry Yuen. *An improved sample complexity lower bound for quantum state tomography*. 2022. `arXiv:2206.11185`.

# Abstract

Over the past half century, the advent of the computer has increased our ability to perform computations tremendously. Consequently, we can now solve computational problems much more efficiently than ever before, the effects of which have revolutionized many aspects of society, ranging from how government keeps track of tax records, to how we use routing software to navigate to our destination.

However, over the last decade it has become increasingly apparent that the development of conventional computational hardware is reaching its physical limits. In an attempt to overcome this barrier, new research areas have focused on rethinking the very physical principles based on which we perform computations. Quantum computing has emerged among the most promising of these new areas, and as such has experienced an explosive increase of interest over the past twenty years.

Quantum computing loosely speaking encapsulates the idea of performing computations based on the principles of quantum mechanics, and as such describes a computational model that is fundamentally different from its conventional counterpart. The central question we address in this thesis is how powerful this new computational model is, i.e., we take computational problems and assess how efficiently they can be solved on a quantum computer. Oftentimes, we compare the obtained results to those in the conventional setting, to quantify the computational advantage that quantum computers provide us with on a theoretical level.

The thesis is divided in two parts. In Part I, we develop quantum algorithms that solve estimation problems. Specifically, we investigate three problems separately, namely mean estimation, state tomography, and partition function estimation. For the first two we generalize our findings to the multivariate setting, and we augment our results with matching lower bound proofs, providing a precise characterization of the problems' quantum computational complexity. Surprisingly, we conclude that for the mean estimation problem quantum computers provide a computational advantage over conventional ones, if and only if

the number of quantum samples used exceeds the dimension of the estimated quantity.

In Part II, we investigate the span program formalism, which is a framework for designing quantum algorithms. We provide a self-contained overview of the formalism with emphasis on visual and intuitive interpretations of the relevant objects. We elaborate on how the framework relates to the adversary bound, and simplify the exposition over previous works. Moreover, we arrive at three new results. First, we show how this framework can be used to turn classical decision trees into quantum algorithms. Second, we improve the best-known algorithm that evaluates approximate span programs. Finally, we show that quantum algorithms of a particular type correspond to span programs that preserve time-efficiency.

# Samenvatting

Gedurende de afgelopen halve eeuw heeft de opkomst van de computer ons vermogen om berekeningen uit te voeren flink vergroot. Daardoor kunnen we nu computationele problemen op een veel efficiëntere manier oplossen dan ooit tevoren. De effecten hiervan hebben de maatschappij in vele opzichten revolutionair veranderd, van de manier waarop de overheid de belastingen bijhoudt, tot de manier waarop we naar onze bestemming navigeren aan toe.

Echter, gedurende het afgelopen decennium is het steeds duidelijker geworden dat de ontwikkeling van conventionele rekenhardware tegen zijn natuurkundige limitaties aanloopt. In een poging deze barrière te overwinnen hebben verschillende onderzoeksgebieden zich gefocust op het heroverwegen van de natuurkundige principes, gebaseerd waarop wij onze berekeningen uitvoeren. Quantum computing is daarbij als één van de meest veelbelovende van deze gebieden uit de bus gekomen, en heeft daardoor gedurende de afgelopen twintig jaar een explosieve toename in interesse meegemaakt.

Quantum computing omvat kortgezegd het idee om berekeningen uit te voeren op basis van de principes van kwantummechanica, en dus beschrijft het een rekenmodel dat fundamenteel anders is dan zijn conventionele tegenhanger. De centrale vraag die we in dit proefschrift behandelen is hoe krachtig dit nieuwe rekenmodel is, d.w.z., we beschouwen computationele problemen en bepalen hoe efficiënt zij door een kwantumcomputer opgelost kunnen worden. Vaak vergelijken we daarbij onze resultaten met die uit de conventionele context, om zodoende in te schatten hoe groot het computationele voordeel is dat kwantumcomputers op theoretisch niveau kunnen bieden.

Dit proefschrift is opgedeeld in twee delen. In Deel I ontwikkelen we kwantumalgoritmen die schattingsproblemen oplossen. Specifiek onderzoeken we drie aparte problemen, namelijk gemiddeldeschatting (Engels: *mean estimation*), toestandstomografie (Engels: *state tomography*), en partitiefunctieschatting (Engels: *partition function estimation*). Voor de eerste twee problemen generaliseren we onze bevindingen naar de multivariate context, en we voegen bewijzen van bij-

passende ondergrensen aan onze resultaten toe, waarmee we een preciese karakterisatie van de computationele complexiteit in de kwantumcontext bereiken. Verrassend genoeg concluderen we dat voor het gemiddeldebepalingsprobleem, kwantumcomputers een computationeel voordeel bieden dan en slechts dan als het aantal kwantumrealisaties dat we gebruiken groter is dan de dimensie van het te schatten object.

In Deel II onderzoeken we het opspanningsprogrammaformalisme (Engels: *span program formalism*), wat een raamwerk voor het ontwikkelen van kwantumalgoritmen is. We geven een op zichzelf staand overzicht van het formalisme, waarbij we de nadruk leggen op visuele en intuïtieve interpretaties van de relevante objecten. We laten zien hoe het raamwerk zich tot de tegenstandergrens (Engels: *adversary bound*) verhoudt, en versimpelen de uiteenzetting t.o.v. eerdere werken. Bovendien komen we uit op drie nieuwe resultaten. Ten eerste laten we zien hoe het raamwerk gebruikt kan worden om klassieke beslisbomen (Engels: *decision trees*) in kwantumalgoritmen om te zetten. Ten tweede verbeteren we het best bekende algoritme dat benaderingsopspanningsprogramma's (Engels: *approximate span programs*) evalueert. Tot slot laten we zien dat kwantumalgoritmen van een bepaald type corresponderen met opspanningsprogramma's (Engels: *span programs*) die de tijdsefficiëntie behouden.

*Titles in the ILLC Dissertation Series:*

ILLC DS-2018-02: **Hugo Huurdeman**
*Supporting the Complex Dynamics of the Information Seeking Process*

ILLC DS-2018-03: **Corina Koolen**
*Reading beyond the female: The relationship between perception of author gender and literary quality*

ILLC DS-2018-04: **Jelle Bruineberg**
*Anticipating Affordances: Intentionality in self-organizing brain-body-environment systems*

ILLC DS-2018-05: **Joachim Daiber**
*Typologically Robust Statistical Machine Translation: Understanding and Exploiting Differences and Similarities Between Languages in Machine Translation*

ILLC DS-2018-06: **Thomas Brochhagen**
*Signaling under Uncertainty*

ILLC DS-2018-07: **Julian Schlöder**
*Assertion and Rejection*

ILLC DS-2018-08: **Srinivasan Arunachalam**
*Quantum Algorithms and Learning Theory*

ILLC DS-2018-09: **Hugo de Holanda Cunha Nobrega**
*Games for functions: Baire classes, Weihrauch degrees, transfinite computations, and ranks*

ILLC DS-2018-10: **Chenwei Shi**
*Reason to Believe*

ILLC DS-2018-11: **Malvin Gattinger**
*New Directions in Model Checking Dynamic Epistemic Logic*

ILLC DS-2018-12: **Julia Ilin**
*Filtration Revisited: Lattices of Stable Non-Classical Logics*

ILLC DS-2018-13: **Jeroen Zuiddam**
*Algebraic complexity, asymptotic spectra and entanglement polytopes*

ILLC DS-2019-01: **Carlos Vaquero**
*What Makes A Performer Unique? Idiosyncrasies and commonalities in expressive music performance*

ILLC DS-2019-02: **Jort Bergfeld**
*Quantum logics for expressing and proving the correctness of quantum programs*

ILLC DS-2019-03: **András Gilyén**
*Quantum Singular Value Transformation & Its Algorithmic Applications*

ILLC DS-2019-04: **Lorenzo Galeotti**
*The theory of the generalised real numbers and other topics in logic*

ILLC DS-2019-05: **Nadine Theiler**
*Taking a unified perspective: Resolutions and highlighting in the semantics of attitudes and particles*

ILLC DS-2019-06: **Peter T.S. van der Gulik**
*Considerations in Evolutionary Biochemistry*

ILLC DS-2019-07: **Frederik Möllerström Lauridsen**
*Cuts and Completions: Algebraic aspects of structural proof theory*

ILLC DS-2020-01: **Mostafa Dehghani**
*Learning with Imperfect Supervision for Language Understanding*

ILLC DS-2020-02: **Koen Groenland**
*Quantum protocols for few-qubit devices*

ILLC DS-2020-03: **Jouke Witteveen**
*Parameterized Analysis of Complexity*

ILLC DS-2020-04: **Joran van Apeldoorn**
*A Quantum View on Convex Optimization*

ILLC DS-2020-05: **Tom Bannink**
*Quantum and stochastic processes*

ILLC DS-2020-06: **Dieuwke Hupkes**
*Hierarchy and interpretability in neural models of language processing*

ILLC DS-2020-07: **Ana Lucia Vargas Sandoval**
*On the Path to the Truth: Logical & Computational Aspects of Learning*

ILLC DS-2020-08: **Philip Schulz**
*Latent Variable Models for Machine Translation and How to Learn Them*

ILLC DS-2020-09: **Jasmijn Bastings**
*A Tale of Two Sequences: Interpretable and Linguistically-Informed Deep Learning for Natural Language Processing*

ILLC DS-2020-10: **Arnold Kochari**
*Perceiving and communicating magnitudes: Behavioral and electrophysiological studies*

ILLC DS-2020-11: **Marco Del Tredici**
*Linguistic Variation in Online Communities: A Computational Perspective*

ILLC DS-2020-12: **Bastiaan van der Weij**
*Experienced listeners: Modeling the influence of long-term musical exposure on rhythm perception*

ILLC DS-2020-13: **Thom van Gessel**
*Questions in Context*

ILLC DS-2020-14: **Gianluca Grilletti**
*Questions & Quantification: A study of first order inquisitive logic*

ILLC DS-2020-15: **Tom Schoonen**
*Tales of Similarity and Imagination. A modest epistemology of possibility*

ILLC DS-2020-16: **Ilaria Canavotto**
*Where Responsibility Takes You: Logics of Agency, Counterfactuals and Norms*

ILLC DS-2020-17: **Francesca Zaffora Blando**
*Patterns and Probabilities: A Study in Algorithmic Randomness and Computable Learning*

ILLC DS-2021-01: **Yfke Dulek**
*Delegated and Distributed Quantum Computation*

ILLC DS-2021-02: **Elbert J. Booij**
*The Things Before Us: On What it Is to Be an Object*

ILLC DS-2021-03: **Seyyed Hadi Hashemi**
*Modeling Users Interacting with Smart Devices*

ILLC DS-2021-04: **Sophie Arnoult**
*Adjunction in Hierarchical Phrase-Based Translation*

ILLC DS-2021-05: **Cian Guilfoyle Chartier**
*A Pragmatic Defense of Logical Pluralism*

ILLC DS-2021-06: **Zoi Terzopoulou**
*Collective Decisions with Incomplete Individual Opinions*

ILLC DS-2021-07: **Anthia Solaki**
*Logical Models for Bounded Reasoners*

ILLC DS-2021-08: **Michael Sejr Schlichtkrull**
*Incorporating Structure into Neural Models for Language Processing*

ILLC DS-2021-09: **Taichi Uemura**
*Abstract and Concrete Type Theories*

ILLC DS-2021-10: **Levin Hornischer**
*Dynamical Systems via Domains: Toward a Unified Foundation of Symbolic and Non-symbolic Computation*

ILLC DS-2021-11: **Sirin Botan**
*Strategyproof Social Choice for Restricted Domains*

ILLC DS-2021-12: **Michael Cohen**
*Dynamic Introspection*

ILLC DS-2021-13: **Dazhu Li**
*Formal Threads in the Social Fabric: Studies in the Logical Dynamics of Multi-Agent Interaction*

ILLC DS-2022-01: **Anna Bellomo**
*Sums, Numbers and Infinity: Collections in Bolzano's Mathematics and Philosophy*

ILLC DS-2022-02: **Jan Czajkowski**
*Post-Quantum Security of Hash Functions*

ILLC DS-2022-03: **Sonia Ramotowska**
*Quantifying quantifier representations: Experimental studies, computational modeling, and individual differences*

ILLC DS-2022-04: **Ruben Brokkelkamp**
*How Close Does It Get?: From Near-Optimal Network Algorithms to Suboptimal Equilibrium Outcomes*

ILLC DS-2022-05: **Lwenn Bussière-Carae**
*No means No! Speech Acts in Conflict*

ILLC DS-2023-01: **Subhasree Patro**
*Quantum Fine-Grained Complexity*

ILLC DS-2023-02: **Arjan Cornelissen**
*Quantum multivariate estimation and span program algorithms*