

# Introduction to Quantum Computing

Johannes Kepler University Linz / Fall 2023

Prof. Dr. Richard Kueng, MSc ETH

Special thanks to **Kristina Kirova**, **Johannes Kofler**, **Alexander Ploier**, and **Jadwiga Wilkens** (alphabetical ordering) for carefully checking and revising these lecture notes.

Copyright ©2024. All rights reserved.

These lecture notes are composed using an adaptation of a template designed by Mathias Legrand, licensed under CC BY-NC-SA 3.0 (<http://creativecommons.org/licenses/by-nc-sa/3.0/>).

# Contents

<b>1</b>	<b>Motivation and outline</b> .....	<b>1</b>
1.1	Motivation: integer factorization	1
1.2	Quantum processing units (QPUs)	3
1.2.1	Non-technical analogy .....	3
1.2.2	Different types of quantum hardware .....	5
1.3	Tentative overview of topics	8
1.4	Open-source toolkits to play around with quantum circuits	8
1.5	Exam and grading process	9
<b>2</b>	<b>Single qubit circuits I</b> .....	<b>10</b>
2.1	Introduction	10
2.2	Gaining intuition	11
2.2.1	Overall layout of single-qubit quantum circuits .....	11
2.2.2	Classical options: identity and bit-flip gate .....	12
2.2.3	Quantum options: superposition and sign-flip .....	13
2.3	Rigorous formalism: matrix-vector multiplication	15
2.4	Application: the BB84 quantum key distribution	20
<b>3</b>	<b>Single qubit circuits II</b> .....	<b>25</b>
3.1	Motivation and outline	25
3.2	Excursion: complex numbers	26

<b>3.3</b>	<b>Ultimate limits of single-qubit logic</b>	<b>28</b>
3.3.1	Recapitulation	28
3.3.2	Clifford gates	30
3.3.3	Universal gate sets	34
<b>3.4</b>	<b>Pauli rotation gates</b>	<b>36</b>
<b>3.5</b>	<b>Application: restricted sum of parity computations</b>	<b>39</b>
<b>4</b>	<b>Two qubit circuits</b>	<b>43</b>
<b>4.1</b>	<b>Classical reversible operations on 2 bits</b>	<b>43</b>
4.1.1	Combining single-bit operations in parallel	44
4.1.2	The Kronecker product	46
<b>4.2</b>	<b>Quantum operations on 2 qubits</b>	<b>48</b>
4.2.1	Quantum gates on 2 qubits	48
4.2.2	Quantum states on 2 qubit	49
4.2.3	The CNOT gate	50
4.2.4	Examples: $CNOT_{1 \rightarrow 0}$ and a random number generator	51
<b>5</b>	<b>Bell states &amp; Superdense Coding</b>	<b>54</b>
<b>5.1</b>	<b>Motivation: The Bell state</b>	<b>54</b>
5.1.1	Stronger than classical correlations	55
<b>5.2</b>	<b>More Bell states</b>	<b>60</b>
<b>5.3</b>	<b>Bell measurement</b>	<b>60</b>
<b>5.4</b>	<b>Superdense Coding</b>	<b>61</b>
<b>5.5</b>	<b>Quantum Games: The Prisoner's Dilemma</b>	<b>63</b>
<b>6</b>	<b>Entanglement</b>	<b>67</b>
<b>6.1</b>	<b>Entanglement</b>	<b>67</b>
6.1.1	Rotated Bell states	68
<b>6.2</b>	<b>The CHSH game and Bell inequalities</b>	<b>70</b>
6.2.1	The CHSH game	70
6.2.2	Optimal classical strategies	71
6.2.3	Optimal quantum strategy	73
<b>6.3</b>	<b>CHSH rigidity and monogamy of entanglement</b>	<b>76</b>
<b>6.4</b>	<b>Bell inequalities and the violation of local realism</b>	<b>76</b>
<b>6.5</b>	<b>The E91 protocol for quantum key distribution</b>	<b>77</b>
<b>7</b>	<b>Quantum teleportation</b>	<b>80</b>
<b>7.1</b>	<b>Motivation</b>	<b>80</b>
<b>7.2</b>	<b>Background: marginal and conditional probabilities</b>	<b>82</b>
7.2.1	Marginal probabilities	82
7.2.2	Conditional probability distributions	83



7.2.3	Example 1: Bell state readout . . . . .	84
7.2.4	Example 2: Drawing straws . . . . .	86
7.3	<b>Quantum <math>T</math>-gate teleportation</b>	<b>87</b>
7.4	<b>Quantum state teleportation</b>	<b>91</b>
<b>8</b>	<b>General <math>n</math>-qubit architectures . . . . .</b>	<b>98</b>
8.1	<b>General <math>n</math>-qubit architectures</b>	<b>98</b>
8.2	<b>Classical description of <math>n</math>-qubit architectures</b>	<b>99</b>
8.2.1	State vector representation of general $n$ -qubit states . . . . .	99
8.2.2	Circuit matrix representation of general $n$ -qubit circuits . . . . .	101
8.2.3	Classical simulation of $n$ -qubit logic and readout . . . . .	104
8.3	<b>Implementing classical circuits with quantum logic</b>	<b>106</b>
8.3.1	Quantum realizations of elementary logical gates . . . . .	107
8.3.2	Quantum realization of entire Boolean circuits . . . . .	108
8.4	<b>Synopsis</b>	<b>111</b>
<b>9</b>	<b>Amplitude amplification circuits . . . . .</b>	<b>113</b>
9.1	<b>Motivation</b>	<b>113</b>
9.2	<b>Setup</b>	<b>113</b>
9.3	<b>Overall idea for a quadratic quantum advantage</b>	<b>116</b>
9.3.1	high-level vision . . . . .	116
9.4	<b>Concrete circuit construction</b>	<b>119</b>
9.4.1	Circuit 1: reflection about good solutions ('Grover oracle') . . . . .	119
9.4.2	Circuit 2: reflection about uniform superposition ('diffusion operator') . . . . .	121
9.4.3	Combination of the two circuit blocks . . . . .	121
9.5	<b>Full quantum search algorithm</b>	<b>122</b>
<b>10</b>	<b>Fourier-type transforms . . . . .</b>	<b>124</b>
10.1	<b>Fourier transform</b>	<b>124</b>
10.1.1	Motivation . . . . .	124
10.1.2	Discrete Fourier transform . . . . .	125
10.2	<b>Quantum Fourier transform</b>	<b>127</b>
10.2.1	Quantum implementation . . . . .	128
10.2.2	Fast (classical) Fourier transform . . . . .	130
10.3	<b>QFT as a subroutine - Quantum phase estimation</b>	<b>131</b>
<b>11</b>	<b>Shor's algorithm for integer factorization . . . . .</b>	<b>136</b>
11.1	<b>Motivation: hard instances of integer factorization</b>	<b>136</b>
11.2	<b>Reducing Integer Factorization to order finding</b>	<b>137</b>
11.2.1	The order finding problem . . . . .	137
11.2.2	Solving integer factorization via order finding . . . . .	139

<b>11.3</b>	<b>Efficiently solving order finding on a quantum computer</b>	<b>141</b>
11.3.1	Recapitulation: quantum phase estimation . . . . .	141
11.3.2	Identifying the order parameter in eigenvalues of a simple reversible circuit	143
11.3.3	Approximate eigenvalues of this simple reversible circuit via phase estimation	144
<b>11.4</b>	<b>Synopsis: implementation of Shor's algorithm</b>	<b>147</b>
<b>12</b>	<b>Learning from quantum experiments . . . . .</b>	<b>149</b>
<b>12.1</b>	<b>Motivation</b>	<b>149</b>
<b>12.2</b>	<b>Stylized learning challenge: data hiding</b>	<b>150</b>
12.2.1	Encoding strategy . . . . .	152
12.2.2	Conventional approach . . . . .	153
12.2.3	Quantum-enhanced approach . . . . .	155
<b>12.3</b>	<b>Demonstration on an actual quantum computer</b>	<b>157</b>
	<b>Bibliography . . . . .</b>	<b>164</b>

# 1. Motivation and outline

Date: October 4, 2023

## 1.1 Motivation: integer factorization

One of the core objectives in computer science is, well, to compute things. On a high level, this is typically achieved by developing algorithms that break down potentially complicated tasks into a sequence of simpler, standardized operations. These standardized operations can then be executed on (classical) hardware. A modern CPU, for instance, can execute billions of elementary logical and arithmetic operations in mere seconds, so we are blessed with substantial amounts of raw computing power.

Alas, raw computing power may not always be enough. There is a wealth of computing problems, where scalability issues prevent even supercomputers from going to really large problem sizes. One well-known problem of this kind is *integer factorization*: decompose a (typically large) number  $N \in \mathbb{N}$  comprised of  $n$  bits ( $n = \lfloor \log_2(N) \rfloor + 1$ ) into a product of prime numbers, i.e.

$$N = F_0 \times \cdots \times F_{m-1} \quad \text{with} \quad F_0, \dots, F_{m-1} \in \mathbb{N} \text{ prime.} \quad (\text{I.I})$$

The fundamental theorem of arithmetic states that every positive integer has a unique prime factorization (if we arrange the factors in non-decreasing order, i.e.  $F_{i-1} \leq F_i$ ). And it is relatively easy to check that the maximum number of factors  $m$  must obey  $m \leq \log_2(N) \approx n$ . So, there can never be too many factors. Also, and more remarkably, it is possible to check that each proposed factor  $F_i$  is actually a prime number. This is courtesy of the AKS algorithm which also scales polynomially in  $n$ . Together, these two insights ensure that it is always possible to efficiently check whether a proposed integer factorization (I.I) is valid. Here, efficiently means that the number of required

### Agenda:

- 1 motivation: integer factorization
- 2 quantum processing units (QPUs)
- 3 overview of topics
- 4 grading process

integer factorization of a  $n$ -bit number

operations scales (at most) polynomially in the representation size  $n$  (bit length) of  $N$ .

But, how can we actually find an integer factorization in the first place? The easiest algorithm is *trial division* which goes back to Fibonacci and is often taught in middle school: systematically test whether  $N$  is divisible by a smaller number. For instance,  $12 = 2 \times 6 = 2 \times 2 \times 3$  which is a valid integer factorization. This algorithm works well if there are a lot of small prime factors, because it is comparatively cheap to identify those. And subsequent divisions reduce the remaining problem size considerably. But, trial division can become extremely resource-intensive if this is not the case. The worst case occurs if  $N = F_0 \times F_1$ , where  $F_0 < F_1$  are unknown prime numbers of size  $\approx \sqrt{N}$ . In such a situation, we need very many trial divisions to find the first prime factor. Indeed, naively trying all numbers between 2 and  $F_0 \approx \sqrt{N}$  requires a total of  $\approx \sqrt{N}$  trial divisions. This number alone is exponentially large in the bit size  $n$  required to represent  $N$ :

$$\sqrt{N} = N^{1/2} = 2^{\log_2(N)/2} \approx 2^{n/2}.$$

It is possible to considerably improve the (worst-case) runtime of trial division by only considering numbers that prime to begin with (e.g. don't try 4, 6, 9, ... at all). But, this is not enough to overcome this general worst-case scaling. The number of required operations for every trial division variant known to date is still dominated by  $2^{n/2}$ . It should be noted that trial division is not the best known algorithm for integer factorization. But even the current state of the art – the general number field sieve – cannot factor a  $n$ -bit number with a number of basic operations that scales polynomially in  $n$ . This feature is actually exploited by widely employed cryptography schemes, most notably RSA public-key encryption.

To summarize: integer factorization is an example of a problem that is difficult to solve (best known algorithms scale exponentially in input size), but easy to verify (correctness of a proposed factorization (1.1) can be checked in only polynomially many steps). This is the trademark structure of a wide and important class of problems – the problem class **NP** which you might remember from your computational complexity lecture. Other example problems of this kind include 3-SAT, the traveling salesman problem, knapsack, minesweeper and many more. But, among **NP**-problems, integer factorization is special. Firstly, we have strong reasons to believe that it is not quite as difficult as other problems, like 3-SAT. Secondly, and more importantly for this course, we have omitted an important detail in our discussion of known factoring algorithms. We actually know an algorithm that is capable of factoring a  $n$ -bit number using a number of elementary operations that scales only polynomially in  $n$  – *Shor's algorithm*. The “only” caveat is that this algorithm cannot be executed on conventional hardware, but requires a different type of hardware that is more expressive – a quantum computer. We will discuss the precise workings of this breakthrough algorithm in this class, but also the underlying type of hardware.

cost of factoring algorithms scales super-polynomially in bit size  $n$

hardness of factoring is basis of RSA public-key encryption

Shor's algorithm can factor integers efficiently, but requires a quantum computer

## 1.2 Quantum processing units (QPUs)

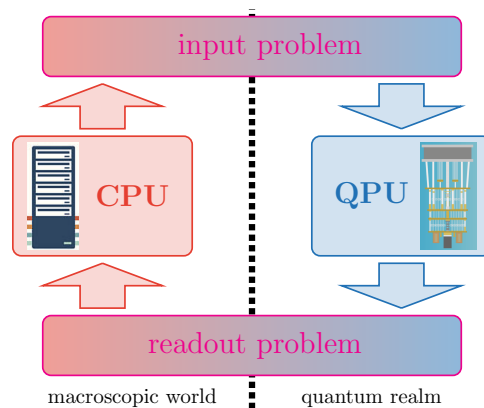
Shor's algorithm for efficient integer factorization posits an interesting conundrum for tried and tested computer science. For decades, we have divided computational problems into different classes of difficulty. This difficulty is measured by the number of operations (or runtime) that is required to solve these problems on any type of computing architecture. But, at least initially, we have only considered computing architectures that model the layout of a modern computer, e.g. Turing machines, or logical circuits. The existence of Shor's algorithm suggests that this may be too restrictive. It works on a different hardware proposal – a quantum computer – and can efficiently solve a problem that we believed to be hard (factoring). A hypothetical quantum computer is at least as powerful as any conventional type of hardware, but it can also natively do things that are impossible – or at least: very expensive – for modern computers. And unprecedented advances over the last decade have brought us closer to actually build and operate these machines.

### 1.2.1 Non-technical analogy

Quantum computers are not the next generation of supercomputers. Rather, they are an entirely new type of computing hardware based on the rules of quantum mechanics – the laws of nature that govern physical systems at microscopic scales (e.g. on the level of individual atoms). And, although well-understood, these rules are radically different from everyday experience. Understanding how a quantum computer actually works is therefore not that easy and we will do so in multiple steps. Although more successful than any other physical theory, quantum mechanics does not have the reputation of being either simple, or intuitive. Some quantum mechanical effects are responsible for the astonishing power of quantum computers, while other effects again limit their potential considerably. Balancing these blessings and curses against each other to still obtain a net gain can be surprisingly tricky. And, as a result, we actually do not know many problems for which quantum computers offer an unconditional (mathematically rigorous) advantage. But, we know some and are constantly looking for more.

In order to get a first intuition about quantum computers, a high-level comparison with conventional hardware can be helpful. The core of most current computing devices is a central processing unit (CPU). It can be tasked to carry out any possible set of instructions we throw at it, but is not necessarily good at computing specific things (a jack of all trades, master of none). This is where alternative processing units come in. One important example are graphical processing units (GPUs). They are designed to solve specialized mathematical operations, in this case large matrix matrix multiplications, much more efficiently than traditional CPUs. The original motivation for this setup is computer graphics, but GPUs are also well-suited for training neural networks and simulating macroscopic physical systems.

However, even GPUs struggle with the excessive number of mathematical



**Figure 1.1** Schematic illustration of a hybrid quantum-classical computer: A conventional Central Processing Unit (CPU) can outsource certain computational task to a Quantum Processing Unit (QPU). The resulting hybrid architecture combines the strengths of both hardware platforms, but also suffers from information-transmission bottlenecks (input problem and readout problem).

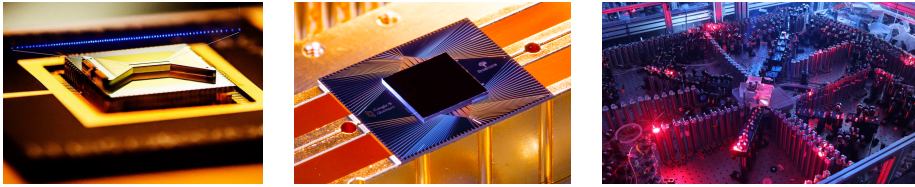
operations that would be required to accurately simulate physical and chemical processes beneath the nanoscale. Problems of this type occur naturally in material science (e.g. the search for high-temperature superconductors), pharmaceuticals and chemistry (e.g. ab initio drug design) and fundamental physics (e.g. probing exotic field theories). All these problems have one thing in common. They adhere to the rules of quantum mechanics. And this renders them extremely difficult to handle with classical (in the sense of macroscopic; not quantum mechanical) computations and hardware. Hence, it would be great if we had a different type of processing unit that is capable of handling these kind of problems. This is the conceptual origin of quantum computers that is often attributed to Richard Feynman:

*“Nature isn’t classical, dammit, and if you want to make a simulation of nature, you’d better make it quantum mechanical, and by golly it’s a wonderful problem, because it doesn’t look so easy.”*

Richard Feynman, 1981.

The term *Quantum Processing Unit* (QPUs) captures the intended purpose more accurately than the colloquially used term quantum computer. QPUs are not designed to supersede conventional computers (like CPUs or GPUs) as a whole, but are specialized processing units that can further augment computing power. The result is a *hybrid quantum-classical computer*, schematically illustrated in Figure 1.1. This combination produces a completely new and different type of computing architecture that comes with novel opportunities, but also novel challenges. We will discuss both throughout the course of this lecture.

quantum computers are special-purpose processing units (QPUs)



**Figure 1.2** Pictures of different types of quantum hardware: ion-trap computer (left), superconducting circuit architecture (center) and optical platform (right). Pictures are taken from [phys.org](https://phys.org), [qmunity.tech](https://qmunity.tech) and [phys.org](https://phys.org), respectively.

### 1.2.2 Different types of quantum hardware

Let us now briefly discuss the most promising ways to actually implement a quantum computing device. Note that a lot of quantum physics enters when it comes to the precise working of these devices. This is not the main focus of this lecture and we therefore content ourselves with a high-level overview of the three prevalent platforms. Photographs of each are collected in Fig. 1.2

#### Trapped ion quantum computing

Atoms are individual particles that are very small (radius about  $10^{-10}\text{m}$ ). They consist of a positively charged core and a hull of electrons that is negatively charged. It is possible to remove individual electrons to produce a positively charged particle – an *ion*. This charge interacts with external electric fields. And one can use these effects to trap ions at a specific location in 3D space (Paul trap, Nobel Prize 1989). This has allowed quantum pioneers to create entire chains of ions that are trapped in a 1D line, see Fig. 1.2 (left). *Qubits* – the fundamental ‘binary’ carriers of quantum information – are stored in electronic states of each ion, e.g.  $0 \leftrightarrow$ ground state,  $1 \leftrightarrow$ first excited state. External lasers are then used to flip individual qubits, while multi-qubit operations are achieved by coupling the state of the qubit in question with external motion states of the entire ion chain.

electronic states of ions carry quantum information

The result is a fully-functional quantum computing platform where information is stored in a collection of ions. Today, roughly 30 qubits can be implemented in this fashion. The clever way of executing multi-qubit operations ensures that this device has full connectivity. That is, we can let every qubit talk to every other qubit. One of the downsides of ion-trap platforms is that they are relatively slow and the motional states – which are essential for all-to-all interactions between the qubits – are difficult to initialize and can have rather brief lifetimes. Also, scaling up to (much) larger qubit numbers is challenging, because the underlying geometry – a chain of ions – is inherently one-dimensional. A two-dimensional lattice of ions could host much more qubits, but these assemblages are still in a rather early stage.

Finally, it is worthwhile to point out that ion trap quantum computing is (almost) an Austrian invention. The theoretical proposal is due to I. Cirac and P. Zoller (1995) who then both worked in Innsbruck. To this date, Innsbruck

remains a global player in ion trap quantum computing. The spin-off company **Alpine Quantum Technologies** is the first European company that actually sets out to sell these quantum computers. They might visit us at some point throughout this lecture.

### Superconducting quantum computing

In this platform type, quantum computing is implemented with superconducting electronic circuits. Qubits are stored in microscopical circuits that roughly resemble a resonant circuit which oscillates. The key difference is that they are comprised of superconducting material (no resistance) and contain a capacitor (like a resonant circuit), but also an additional nonlinearity – a superconducting tunnel junction (Nobel Prize in 1973). The state of such a qubit is determined by the number of electrons which reside on one side of the circuit compared to the other. These states can be flipped by microwave pulses sent to an antenna coupled to the qubit.

electronic circuits carry quantum information

This basic building block (circuit plus antenna) is in principle scalable using existing chip manufacturing techniques. Many such individual circuits can be arranged on a 2D plane which today can host up to 100 superconducting qubits. Interactions between these qubits can be achieved by coupling two superconducting qubits to an intermediate coupling circuit. These intermediate circuits must reside between the two qubits in question and must also be fabricated. This effectively limits interactions to nearest neighbors: every qubit can only talk to its immediate neighbors.

Today, superconducting platforms are the prevalent type of quantum hardware. Companies like IBM, Google, Rigetti and more have really pushed this technology over the last couple of years. Today's devices can host between 53 (Google Sycamore) and 127 (IBM Eagle) qubits. They also operate much quicker than existing ion trap devices. These desirable effects have led to the first justifiable demonstrations of a quantum advantage – i.e. a well-defined task where quantum computers outperform even the largest supercomputers to date by a substantial margin. We will discuss one such result towards the end of this course. However, superconducting quantum computers are not perfect. They have to operate at extremely low temperatures (300mK) to ensure that the underlying material is perfectly superconducting. This may, in fact, soon become a bottleneck for further scaling up to larger qubit sizes, because the additional electrical components produce heat that needs to be counteracted. Also, the stringent limitation to nearest-neighbor interactions can considerably slow down the actual realization of a given quantum circuit.

### Optical architectures

So far, we have mainly talked about quantum computing platforms. But these are only one aspect of the larger field of quantum technology. When it comes to communication and networks – think internet – light is a very promising carrier of information. Light can quickly and reliably cover large distances to convey information. And, it comes in discrete packages of 'light particles',

photons carry quantum information



called *photons*. These photons can be used to represent a qubit, e.g. via polarization: 0  $\leftrightarrow$  horizontal, 1  $\leftrightarrow$  vertical. These polarizations can be picked out and modified with linear optical elements – think mirrors – which effectively implements single-qubit transformations. What is more, optical devices don't require low temperatures and can be built and scaled-up relatively easily.

The big problem is that photons don't directly interact with each other. This makes it very difficult to execute multi-qubit operations – a prerequisite for creating interesting correlations between qubits and performing interesting computations. One way to overcome this issue is at the source: nanomaterials, like quantum dots, can be used to create multiple photons at once which do already exhibit powerful correlations (entanglement). This initial budget of quantum correlation can subsequently be consumed to execute powerful quantum computing procedures, like state teleportation which we will discuss in due time. This technology also forms the basis of the nascent quantum internet. Alain Aspect, John F. Clauser and Anton Zeilinger – a compatriot who was born in Ried – are pioneers of this type of quantum technology. Last year (04.10.2022), they received the Physics Nobel Prize for these groundbreaking contributions.

So far, we have emphasized the potential of photons to carry quantum bits over large distances. However, it is also possible to devise actual quantum computers based on photons. These, so-called, measurement-based computing architectures are remarkably different from conventional circuit architectures. As such, they go beyond the scope of this lecture.

### Synopsis

The above explanations are extremely crude and superficial, they sweep a lot of important and groundbreaking insights under the rug. But we hope that they convey a high-level message: it is actually possible to build quantum computing platforms and there are, in fact, several ways to do so. And each comes with their own advantages and disadvantages. It should also be noted that qubits and elementary operations (single qubit + two-qubit) are not enough by themselves. We also need a way to initialize the qubits (qubit initialization) and way to access the final result (qubit readout). We will discuss all these ingredients in the next lecture. It is also important to note that all these operations are challenging (we operate on the tiniest scales imaginable) and not perfect. Every operation is bound to incur a small error. And these errors can, and do, accumulate when we start combining many operations to build a larger circuit. Today, this severely hampers our ability to scale up quantum computing technology. There are, however, ways to overcome these issues. We will briefly cover this topic of *quantum error correction* towards the end of this course.

different ways to realize QPUs

### 1.3 Tentative overview of topics

Lectures will occur weekly on Wednesdays, 13:45–15:15. There will be 13 standard lectures in total. This is very little time to cover the vast area of quantum computing. As a result, we will make compromises and sub-select topics which we then discuss in depth. The core focus of this class will be *quantum circuits* and, by extension, *quantum algorithms*. These are arguably the ‘raison d’être’ for building quantum hardware in the first place. They also can be understood as an interesting generalization of digital circuits. To get everyone up to speed, we start small and gradually build up to more involved and potentially impactful algorithms described by quantum circuits. Here is a tentative list of 13 topics:

focus on quantum circuits & algorithms

- 1 **Motivation and outline**
- 2 **Single-qubit circuits 1:** gaining intuition, mathematical formalism and one application: quantum cryptography (BB84)
- 3 **Single-qubit circuits 2:** universal gate sets, approximation theorems (Solayev-Kitaev), reversibility and one application: compute the parity of a sum of rational numbers
- 4 **Two-qubit circuits 1:** basic building blocks, mathematical formalism and one application: superdense coding
- 5 **Two-qubit circuits 2:** entanglement, Bell inequalities and device-independent quantum cryptography (Ekert)
- 6 **guest lecture by Dr. Johannes Kofler:** the foundational implications of quantum effects
- 7 **Teleportation subroutines:** conditional gate applications,  $T$ -gate teleportation, quantum state teleportation,
- 8 **Many-qubit circuits:** relation between quantum and classical circuits
- 9 **Amplitude amplification** and quadratic speedup for 3-SAT (*Grover*)
- 10 **Quantum Fourier transforms:** Hadamard transformation, the Bernstein-Vazirani algorithm and the Quantum Fourier transform
- 11 **Shor’s algorithms** for discrete logarithms and integer factorization
- 12 **Basics of quantum error correction**
- 13 **(Machine) learning from quantum experiments**

### 1.4 Open-source toolkits to play around with quantum circuits

Recent years has also seen the emergence of open-access software that simulates quantum processing units on conventional hardware. . Examples include **qiskit** (by IBM), **Cirq** (by Google) and **pennylane** (by Xanadu). These are all great ways to get valuable intuition by playing around with small quantum computations (involving  $\lesssim 10$  qubits). We strongly encourage you to use (at least) one of these to play around and gather intuition in a playful fashion. Also, the internet is full with great (and freely available) courses that approach quantum computing from this angle.

open-access software emulates QPUs and provides ample opportunity to play around

## 1.5 Exam and grading process

The tentative exam date is **February 7th, 2024, 10:00 –12:00**. We offer a Moodle exam which you can attend either from home or in a designated lecture hall. We also propose to do an **open-book exam**, i.e. you can use lecture notes, personal summaries and even the internet. This, however, means that the individual questions will be more challenging: you will have to analyze and understand how concrete quantum circuits operate. Additional details and example questions will follow in due course.

open book exam via moodle

## 2. Single qubit circuits I

Date: October 11, 2023

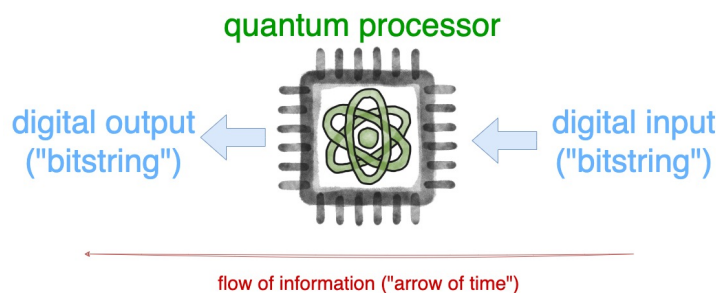
### 2.1 Introduction

Today, we take our first steps into the realm of quantum computation. A high-level schematic of a quantum processing unit (QPU) is displayed in Fig. 2.1. Note that a QPU is a *digital device*: it ‘eats’ bitstrings and ‘spits out’ bitstrings. This figure also highlights a seemingly unconventional choice of convention.

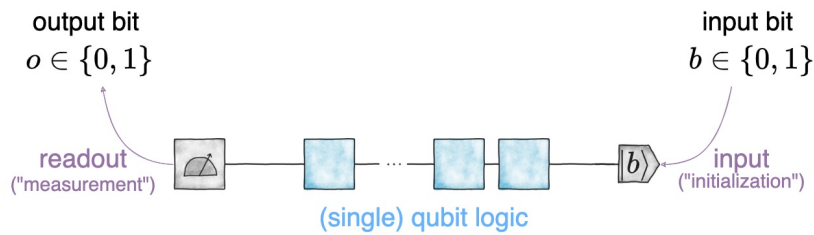
**Convention (reading circuit diagrams from right to left).** In this class, we read circuit diagrams from right to left. The red arrow in Fig. 2.1 visually illustrates this flow of information (“arrow of time”). The reason for this convention will become clear later on: it plays nicely with the mathematical formalism we use to capture quantum logic (matrix-vector multiplication).

#### Agenda:

- 1 introduction
- 2 gaining intuition
- 3 rigorous formalism: matrix-vector multiplication
- 4 application: quantum key distributions (BB84)



**Figure 2.1** Schematic illustration of a quantum processing unit (QPU): on a high level, a QPU maps bitstrings to bitstrings. Also, in this class we read circuit diagrams from right to left. The red arrow underscores this convention.



**Figure 2.2** Schematic illustration of a single-qubit processor (QPU): input (very right) and output (very left) of a single-qubit QPU are conventional bits. Inbetween, single qubit logic (blue) is used to process the input bit directly at the quantum level. Disruptive effects happen at the quantum-classical interface (purple arrows), in particular the readout stage.

For the remainder of this lecture – and most lectures – we adopt a hardware perspective and represent QPUs by quantum circuits. Today, we focus on circuits that affect a single quantum bit, called *qubit*. We will see that these circuits can execute well-known logical functionalities (e.g. negation), but other elementary gates don't have a classical counterpart at all. For illustrative purposes, we will heavily use the *Quantum Circuit Library* developed by Jadwiga Wilkens who is now a quantum PhD student at JKU [Wil23].

qubit = quantum bit

## 2.2 Gaining intuition

### 2.2.1 Overall layout of single-qubit quantum circuits

A QPU operates on two fundamentally different levels. Input and output do correspond to conventional bit strings. However, the logic in-between is executed on the microscopic level. There, genuine quantum effects become available and can be used to perform completely new types of (quantum) logic. Fig. 2.2 illustrates such a setting for a single qubit. There a single qubit is initialized with an input bit value  $b \in \{0, 1\}$  (right). Subsequently, a collection of single-qubit logical gates is applied (center). This is where the actual quantum computation happens. Once this is completed, we perform a readout step where the qubit is measured to produce a single output bit  $o \in \{0, 1\}$ . Throughout the course of today's lecture, we will explore the workings of such a hybrid quantum-classical architecture. We will discover that the quantum logic part is captured by a nice deterministic and even reversible formalism. The interfaces between quantum and classical realm are more disruptive, by comparison. Readout, in particular, can produce true randomness – something that is impossible for conventional (deterministic) hardware. Let us now start to discover the workings and interplay of these different constituents in a step-by-step fashion.

### 2.2.2 Classical options: identity and bit-flip gate

Hybrid quantum-classical architectures, like the one displayed in Fig. 2.2, are capable of executing basic logical functionalities. A simple, but often under-appreciated, logical gate is the *identity operation*, i.e. do nothing. This operation maps a bit to itself, i.e.

$$\mathbb{I}(b) = b \quad \text{for } b \in \{0, 1\}.$$

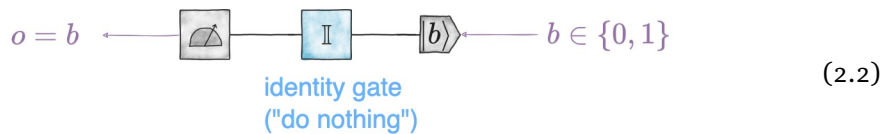
logical identity ( $\mathbb{I}$ )

Equivalently, we can fully capture this (trivial) action by the following truth table:

	0	1	
0	1	0	
1	0	1	

(identity truth table). (2.1)

A full pipeline with qubit initialization, identity gate (blue) and readout stage looks as follows:



As intended, the final output bit  $o$  is equal to the input bit  $b$  ( $o = b$ ). This showcases that we can use a single-qubit QPU to reliably store one bit of information within the quantum realm and recover it exactly at some later point in time.

Another important logical gate is the *negation*, or bit-flip operation:

bit-flip ( $X$ )

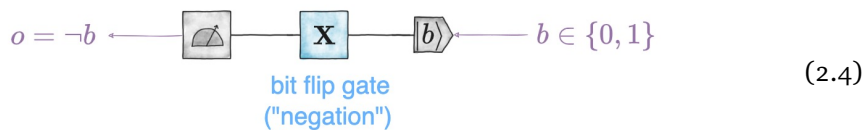
$$X(b) = \neg b \quad \text{for } b \in \{0, 1\},$$

where  $\neg 0 = 1$  and  $\neg 1 = 0$ . This important logical operation is covered by the following truth table:

	0	1	
0	1	0	
1	0	1	

(bit-flip truth table). (2.3)

A single-qubit QPU can also implement this functionality:



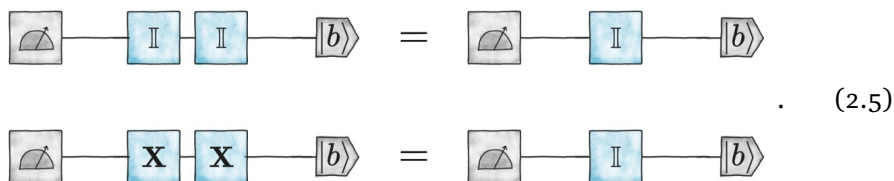
Together, Eq. (2.2) and Eq. (2.4) showcases that we can use a hybrid quantum-classical pipeline to implement the most important single-bit operations. These

operations, however, have an additional feature. Applying them twice has no effect on the (qu)bit in question:

$$\mathbb{I}(\mathbb{I}(b)) = \mathbb{I}(b) = b \quad \text{and} \quad \mathbf{X}(\mathbf{X}(b)) = \neg(\neg b) = b \quad \text{for } b \in \{0, 1\}.$$

This, in particular, means that identity and bit-flip are reversible logical operations. They do not erase any information about the input bit. In fact, their action can be readily undone by applying the same gate again. Note that not all conceivable single-bit functions have this feature. There are in total 4 Boolean functions that map a single bit onto a single bit. Identity and bit-flip are two of them. The other two correspond to re-setting the bit to one particular value. I.e.  $f(b) = 0$  (reset to 0) or  $f(b) = 1$  (reset to 1), for  $b \in \{0, 1\}$ . Clearly, these re-set operations are not reversible, because they completely erase the input bit. The quantum implementations of identity and bit-flip do adhere to *reversibility*. This is captured by the following streamlined circuit diagram equations:

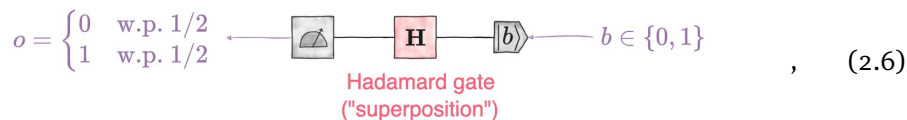
QPUs can execute *reversible* single-bit logic



### 2.2.3 Quantum options: superposition and sign-flip

We have just seen that a single-qubit QPU can reproduce basic logical functionalities, as long as they are reversible ( $\mathbb{I}$  and  $\mathbf{X}$ ). This is a good start that suggests that QPUs may be capable of executing conventional logical operations. But, by itself, this is not really spectacular (yet). Let us now discuss some genuinely quantum operations that don't have a conventional counterpart. First and foremost, there is the *Hadamard* or *superposition gate*:

Hadamard/superposition (**H**)



where 'w.p. 1/2' is short for 'with probability 1/2'. This classical-quantum-classical pipeline takes an arbitrary single-bit input  $b$  and produces a uniformly random output bit. We write

$$o \stackrel{\text{unif}}{\sim} \{0, 1\}$$

to denote that  $o = 0$  and  $o = 1$  happen with equal probability 1/2 each. This feature is a striking deviation from conventional logic which is fundamentally deterministic. The execution of a Hadamard gate uses an interesting quantum

effect, called *superposition*: a binary quantum system (qubit) can assume both bit values at the same time.

If we readout (measure) such a superposition of binary values, the outcome bit we obtain is truly random: both  $o = 0$  and  $o = 1$  occur with equal probability. This is the same situation as a fair coin flip. The Hadamard gate provides the means to observe true randomness by bringing a qubit into equal superposition. And, equally strikingly, we can use another Hadamard gate to exit superposition again. Much like identity and bit-flip, the Hadamard gate is also reversible. In fact, it is also its own inverse:

$$\begin{array}{c} \text{[Coin Icon]} \\ \text{---} \\ \text{[H]} \text{ [H]} \\ \text{---} \\ \text{[b]} \end{array} = \begin{array}{c} \text{[Coin Icon]} \\ \text{---} \\ \text{[I]} \\ \text{---} \\ \text{[b]} \end{array} . \quad (2.7)$$

We emphasize that this is not obvious. We will do such a calculation later on, or in an exercise. Together, Eq. (2.6) and Eq. (2.7) reveal a striking quantum phenomenon. The first equation showcases that the Hadamard gate can be used to generate uniformly random bits. In standard binary logic, randomization requires an external seed and cannot be undone without erasing the bit in question. Or, put differently: the only way to map a random bit  $r$  into a deterministic bit  $o$  is to erase and reset. This breaks any correlations with the original input bit. The Hadamard gate, however, is not like this at all. We can apply it twice to completely undo its effect and recover a perfect correlation between input bit  $b$  and output bit  $o$ .

The ‘truth table’ of the Hadamard gate reflects this, because it doesn’t adhere to the rules of conventional logic:

	0	1	
0	$1/\sqrt{2}$	$1/\sqrt{2}$	(Hadamard ‘truth table’).
1	$1/\sqrt{2}$	$-1/\sqrt{2}$	

The detailed numbers in this table should become clear later on. For now, we emphasize two things:

- (i) the magnitude of each entry is the same, that is 0 and 1 feature in equal measure within the superposition;
- (ii) the two rows (columns) are distinct. This means that information about the input qubit is actually preserved.

We conclude this section with another quantum logic gate, the *sign-flip gate*. The ‘truth table of the Hadamard gate’ already suggests that quantum logic can feature positive and negative numbers. The sign-flip is used to change the sign of the 1-contribution within a superposition:

sign-flip ( $Z$ )

	0	1	
0	1	0	(sign-flip ‘truth table’).
1	0	-1	



Interestingly, this sign-flip doesn't do anything if we apply it to quantum encodings of conventional logic. In particular,

$$\begin{array}{c}
 o = b \leftarrow \text{[Hadamard]} \text{---} \text{[Z]} \text{---} \text{[Measurement } |b\rangle] \leftarrow b \in \{0, 1\} \\
 \text{sign-flip gate} \\
 \text{(not identity (?))}
 \end{array}
 , \quad (2.8)$$

which looks identical to the action of the identity gate in Eq. (2.2). Also, much like the identity gate (and every other gate we've encountered so far), the sign-flip gate is also its own reverse:

$$\text{[Hadamard]} \text{---} \text{[Z]} \text{---} \text{[Z]} \text{---} \text{[Measurement } |b\rangle] = \text{[Hadamard]} \text{---} \text{[I]} \text{---} \text{[Measurement } |b\rangle] . \quad (2.9)$$

Importantly, the action of a sign gate becomes nontrivial if we apply it to superpositions of different bit values. This is achieved by combining  $Z$  with the Hadamard gate  $H$ . For instance,

$$\text{[Hadamard]} \text{---} \text{[H]} \text{---} \text{[Z]} \text{---} \text{[H]} \text{---} \text{[Measurement } |b\rangle] = \text{[Hadamard]} \text{---} \text{[X]} \text{---} \text{[Measurement } |b\rangle] , \quad (2.10)$$

which showcases that we can sandwich  $Z$  (sing-flip) between two Hadamard gates  $H$  (enter/leave superposition) to execute a logical bit-flip  $X$ . This is a very strong case for the nontrivial behavior of a sign-flip gate ( $Z \neq \mathbb{I}$ ).

### 2.3 Rigorous formalism: matrix-vector multiplication

We have now completed a first look at how quantum circuits work. We have seen that they can implement standard logic (e.g. identity and bit-flip), but also have new and seemingly mysterious features. The apparent randomness generation via Hadamard which can be completely undone via another Hadamard are a striking example of this kind. We now present a mathematical formalism that can be used to reproduce all these new quantum features. It equips the intuition we gained so far with a rigorous underpinning. This is essential when it comes to exploring the realm of quantum computing and is one of the most transformative developments within quantum science. Over the past 30 years or so, the field has moved away from complicated physical equations and gradually developed a succinct and finite dimensional alternative that nonetheless captures all interesting quantum effects. In fact, basic matrix-vector multiplication is enough to keep track of any QPU that is comprised of (finitely many) qubits. Today, we present these rules for the special case of a single qubit. The central object used to describe a single-qubit QPU is the current *state* of the qubit involved.

**Definition 2.1 (single-qubit state vector).** The *state* of a single qubit keeps track of its quantum logical level. At each point, it is given by a 2-dimensional vector

$$|\psi\rangle := \boldsymbol{\psi} = \begin{pmatrix} \psi_0 \\ \psi_1 \end{pmatrix} \in \mathbb{C}^2.$$

The individual coefficients can be complex-valued numbers, but must obey

$$\| |\psi\rangle \|^2 = \| \boldsymbol{\psi} \|^2 = |\psi_0|^2 + |\psi_1|^2 = 1 \quad (\text{state normalization}). \quad (2.11)$$

We use the somewhat strange notation  $|\psi\rangle$  to denote state vectors. This is called a ‘ket’ and features prominently in the quantum computing literature. The following example tells us how we can imprint a classical bit  $b \in \{0, 1\}$  into the initial state of a qubit.

**Example 2.2 (Qubit initialization).**

$$|0\rangle = \mathbf{e}_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{and} \quad |1\rangle = \mathbf{e}_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

We can interpret the first state vector  $|0\rangle$  as ‘everything is concentrated at 0 (first entry)’, while  $|1\rangle$  means that ‘everything is concentrated at 1. Moreover, both state vectors obey the normalization condition (2.11):

$$\begin{aligned} \| |0\rangle \|^2 &= \| \mathbf{e}_0 \|^2 = |1|^2 + |0|^2 = 1, \\ \| |1\rangle \|^2 &= \| \mathbf{e}_1 \|^2 = |0|^2 + |1|^2 = 1. \end{aligned}$$

■

As Definition 2.1 suggests, state vectors can be used to keep track of qubits throughout a sequence of quantum logical gates. This, however, necessitates a formalism to unambiguously characterize the action of quantum gates. And a way to imprint their action onto the current state vector of a qubit.

**Definition 2.3 (single-qubit gate action).** A single-qubit gate is fully described by a  $2 \times 2$  matrix

$$\mathbf{U} = \begin{pmatrix} U_{0,0} & U_{0,1} \\ U_{1,0} & U_{1,1} \end{pmatrix} \in \mathbb{C}^{2 \times 2}.$$

The action of gate  $\mathbf{U}$  on quantum state  $|\psi\rangle = \boldsymbol{\psi} \in \mathbb{C}^2$  is captured by matrix-vector multiplication :

$$|\psi_{\text{final}}\rangle = \boldsymbol{\psi}_{\text{final}} = \mathbf{U}\boldsymbol{\psi} = \begin{pmatrix} U_{0,0} & U_{0,1} \\ U_{1,0} & U_{1,1} \end{pmatrix} \begin{pmatrix} \psi_0 \\ \psi_1 \end{pmatrix} = \begin{pmatrix} U_{0,0}\psi_0 + U_{0,1}\psi_1 \\ U_{1,0}\psi_0 + U_{1,1}\psi_1 \end{pmatrix} \in \mathbb{C}^2.$$

What is more, each gate matrix  $\mathbf{U}$  must be *unitary* and, therefore, reversible:

$$\mathbf{U}^\dagger \mathbf{U} = \mathbf{U} \mathbf{U}^\dagger = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \text{where} \quad \begin{pmatrix} U_{0,0} & U_{0,1} \\ U_{1,0} & U_{1,1} \end{pmatrix}^\dagger = \begin{pmatrix} \bar{U}_{0,0} & \bar{U}_{1,0} \\ \bar{U}_{0,1} & \bar{U}_{1,1} \end{pmatrix} \quad (2.12)$$

denotes matrix adjugation (transposition plus taking complex conjugates<sup>1</sup>).

<sup>1</sup>Recall that the *complex conjugate* of a complex number  $z = a + ib$  is defined as  $\bar{z} = a - ib$ .

state of a qubit is a normalized 2D vector

single-qubit gates are unitary  $2 \times 2$  matrices (‘truth tables’)

gate action on qubit state = matrix-vector multiplication

The following instructive example showcases how this matrix-vector multiplication formalism allows us to recover classical logical operations.

**Example 2.4 (matrix representation of classical gates).** The matrix representations of the classical operations *identity* ( $\mathbb{I}$ ) and *bit-flip* ( $\mathbf{X}$ ) are

$$\mathbb{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

It is easy to check that both matrices are unitary matrices. What is more,

$$\begin{aligned} \mathbb{I}|0\rangle &= \mathbb{I}\mathbf{e}_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \mathbf{e}_0 = |0\rangle, \\ \mathbb{I}|1\rangle &= \mathbb{I}\mathbf{e}_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \mathbf{e}_1 = |1\rangle, \end{aligned}$$

which puts ‘do nothing’ into concrete formulas. Likewise

$$\begin{aligned} \mathbf{X}|0\rangle &= \mathbf{X}\mathbf{e}_0 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \mathbf{e}_1 = |1\rangle, \\ \mathbf{X}|1\rangle &= \mathbf{X}\mathbf{e}_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \mathbf{e}_0 = |0\rangle, \end{aligned}$$

puts formulas to the action of a bit-flip. It is not a coincidence that these matrix representations are in one-to-one correspondence to the truth tables of the corresponding logical functionalities. Matrix-vector multiplications is just another way of reading logical truth tables. ■

**Exercise 2.5 (matrix representation of the Hadamard gate).** The matrix representation of the Hadamard gate is given as

$$\mathbf{H} = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix} \in \mathbb{C}^{2 \times 2}.$$

- 1 Show that this matrix is unitary by verifying Eq. (2.12) for  $\mathbf{U} = \mathbf{H}$ .
- 2 The action of a Hadamard gate maps deterministic bit states  $|0\rangle$  and  $|1\rangle$  into uniform superposition states  $|+\rangle$  and  $|-\rangle$ . Use matrix-vector multiplication to verify the following state vector representations:

$$|+\rangle = \mathbf{H}|0\rangle = \mathbf{H}\mathbf{e}_0 = \frac{1}{\sqrt{2}} (\mathbf{e}_0 + \mathbf{e}_1) = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle), \quad (2.13)$$

$$|-\rangle = \mathbf{H}|1\rangle = \mathbf{H}\mathbf{e}_1 = \frac{1}{\sqrt{2}} (\mathbf{e}_0 - \mathbf{e}_1) = \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle). \quad (2.14)$$

Both expressions on the very right should be interpreted as ‘both 0 and 1 in equal measure’. Note, however that one combination has a ‘+’ inbetween, while the other one has a ‘-’. This sign difference ensures that both superpositions can be undone again.

- 3 Verify reversibility by showing  $\mathbf{H}^2 = \mathbb{I}$  via matrix-vector multiplication.

Definition 2.1 (state vector) and Definition 2.3 provide us with all the rules we need to keep track of a state vector throughout an arbitrary long sequence of single-qubit gates. The following proposition can be derived from these rules and allows for compressing multiple gate actions into a single matrix.

**Proposition 2.6 (sequential gate composition rule).** Let  $\mathbf{U}, \mathbf{V} \in \mathbb{C}^{2 \times 2}$  be matrix representations of two single-qubit gates. Then, the total action of sequentially applying both is captured by the *matrix-matrix product* of all gates involved:

sequential gate composition  
= matrix-matrix product

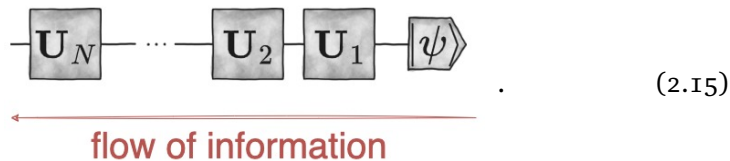
$$\begin{aligned} \mathbf{U}_{\text{tot}} = \mathbf{V} \times \mathbf{U} &= \begin{pmatrix} U_{0,0} & U_{0,1} \\ U_{1,0} & U_{1,1} \end{pmatrix} \times \begin{pmatrix} V_{0,0} & V_{0,1} \\ V_{1,0} & V_{1,1} \end{pmatrix} \\ &= \begin{pmatrix} U_{0,0}V_{0,0} + U_{0,1}V_{1,0} & U_{0,0}V_{0,1} + U_{0,1}V_{1,1} \\ U_{1,0}V_{0,0} + U_{1,1}V_{1,0} & U_{1,0}V_{0,1} + U_{1,1}V_{1,1} \end{pmatrix}. \end{aligned}$$

This composition rule with the matrix product straightforwardly extends to  $N$  sequential gate applications:  $\mathbf{U}_{\text{tot}} = \mathbf{U}_N \times \mathbf{U}_{N-1} \cdots \times \mathbf{U}_2 \times \mathbf{U}_1$ .

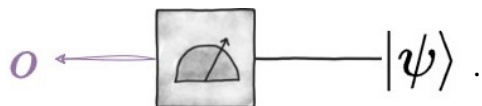
We leave the proof as an instructive exercise and instead want to draw attention to the way we write down and read large matrix-vector products. Suppose that we sequentially apply  $N$  single-qubit gates  $\mathbf{U}_1, \dots, \mathbf{U}_N$  to an arbitrary starting state (vector)  $|\psi\rangle = \boldsymbol{\psi} \in \mathbb{C}^2$ . Then, we can compute the final state vector as

$$|\psi_{\text{final}}\rangle = \boldsymbol{\psi}_{\text{final}} = \mathbf{U}_N \times \cdots \times \mathbf{U}_2 \times \mathbf{U}_1 \boldsymbol{\psi} \in \mathbb{C}^2.$$

In words: we start our matrix-vector multiplication on the very right and keep going. The convention to read circuit diagrams from left to right as well exactly resembles this ordering:



We now have all the pieces in place to initialize a qubit and keep track of its state throughout a sequence of arbitrary many single qubit gates. All that is missing now is a formula for executing the readout at the very end. That is, we need to assign meaning to the following operation:



**Definition 2.7 (single-qubit readout).** A single-qubit readout (measurement) operation always produces a valid bit value  $o \in \{0, 1\}$ . But it does so probabilistically.

The probability of obtaining outcome  $o \in \{0, 1\}$  depends on the underlying state vector  $|\psi\rangle = \boldsymbol{\psi} \in \mathbb{C}^2$ :

$$p_0 = \Pr_{|\psi\rangle} [o = 0] = |\langle 0|\psi\rangle|^2 = \left| \mathbf{e}_0^\dagger \boldsymbol{\psi} \right|^2 = |\psi_0|^2 \geq 0, \quad (2.16)$$

$$p_1 = \Pr_{|\psi\rangle} [o = 1] = |\langle 1|\psi\rangle|^2 = \left| \mathbf{e}_1^\dagger \boldsymbol{\psi} \right|^2 = |\psi_1|^2 \geq 0. \quad (2.17)$$

outcome probabilities = squared magnitudes of state vector entries

This should be read as: ‘the probability of obtaining outcome  $o = 0$  ( $o = 1$ ) when reading out a qubit in state  $|\psi\rangle$  is  $|\psi_0|^2$  ( $|\psi_1|^2$ ).

Here, we use another bit of quantum notation: we write  $\langle 0|$  ( $\langle 1|$ ) to denote the dual/adjoint state vector of  $|0\rangle = \mathbf{e}_0$  ( $|1\rangle = \mathbf{e}_1$ ):

$$\langle 0| = \mathbf{e}_0^\dagger = \begin{pmatrix} 1 & 0 \end{pmatrix} \quad \text{and} \quad \langle 1| = \mathbf{e}_1^\dagger = \begin{pmatrix} 0 & 1 \end{pmatrix}.$$

This is called a ‘bra’ and plays nicely with the ‘ket’. Indeed, combining a ‘bra’ (row vector) with a ‘ket’ (column vector) produces a number (bra-ket, i.e. inner product). For instance,

$$\langle 0|0\rangle = \mathbf{e}_0^\dagger \mathbf{e}_0 = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = 1 \times 1 + 0 \times 0 = 1,$$

$$\langle 0|1\rangle = \mathbf{e}_0^\dagger \mathbf{e}_1 = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = 1 \times 0 + 0 \times 1 = 0,$$

$$\langle 1|0\rangle = \mathbf{e}_1^\dagger \mathbf{e}_0 = \begin{pmatrix} 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = 0 \times 1 + 1 \times 0 = 0,$$

$$\langle 1|1\rangle = \mathbf{e}_1^\dagger \mathbf{e}_1 = \begin{pmatrix} 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = 0 \times 0 + 1 \times 1 = 1$$

and an extension to other ‘bra’ and ‘ket’ vectors is straightforward.

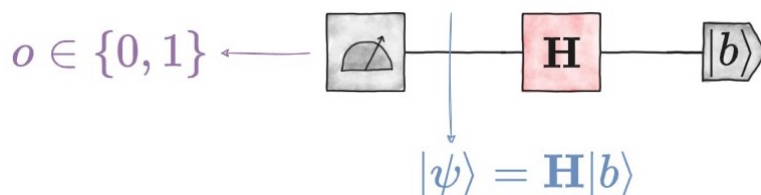
Note that normalization of the state vector (Eq. (2.11) in Definition 2.1) ensures that  $p_0$  and  $p_1$  defined in Eqs. (2.16), (2.17) define a valid binary probability distribution (think: coin toss):  $p_0, p_1 \geq 0$  and

$$p_0 + p_1 = |\psi_0|^2 + |\psi_1|^2 = \|\boldsymbol{\psi}\|^2 = \|\psi\rangle\|^2 = 1.$$

We now have everything in place to reproduce the actual workings of *all* example circuits so far. We’ll do one concrete example and leave the rest as a very instructive exercise.

**Example 2.8 (quantum random number generator).** Consider the quantum circuit from Eq. (2.6), i.e.:

quantum random number generator



Let us do the computation for  $b = 1$  (the case for  $b = 0$  is similar and we leave it as an instructive exercise). We start by using matrix-vector multiplication to compute the final state vector (blue):

$$|\psi\rangle = \begin{pmatrix} \psi_0 \\ \psi_1 \end{pmatrix} = \mathbf{H}|1\rangle = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}.$$

Hence,  $\psi_0 = 1/\sqrt{2}$ ,  $\psi_1 = -1/\sqrt{2}$  and perfect randomness generation follows from invoking the rule for single-qubit readout (Definition 2.7):

$$\begin{aligned} \Pr_{|\psi\rangle} [o = 0] &= |\langle 0|\psi\rangle|^2 = |\psi_0|^2 = \left|1/\sqrt{2}\right|^2 = 1/2, \\ \Pr_{|\psi\rangle} [o = 1] &= |\langle 1|\psi\rangle|^2 = |\psi_1|^2 = \left|-1/\sqrt{2}\right|^2 = 1/2. \end{aligned}$$

This is just the definition of a perfect random bit  $o \stackrel{\text{unif}}{\sim} \{0, 1\}$ . ■

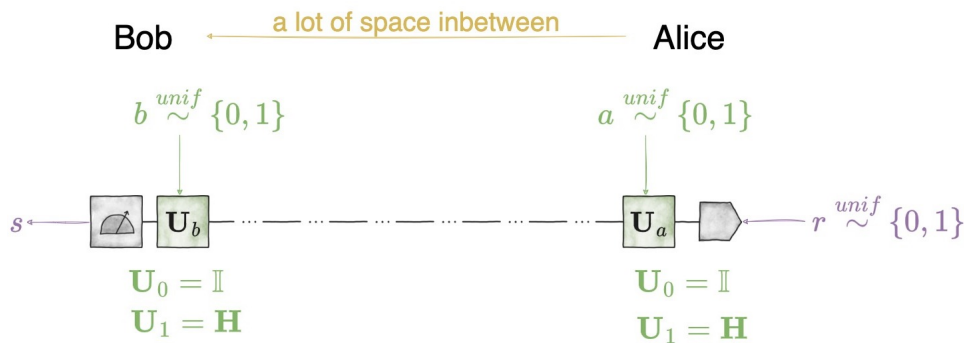
## 2.4 Application: the BB84 quantum key distribution

We now have gathered enough insights about single-qubit circuits to discuss our first quantum application: the BB84 protocol for *quantum key distributions (QKD)* [BB14]. The main goal of KD is to distribute a uniformly random seed among two parties – Alice and Bob – such that it remains private. I.e. information about the secret bit string is not available for any third party (Eve). The Q in QKD indicates that this is achieved by using quantum computing features. The twist is that the rules of quantum computing allow for detecting whether an eavesdropper might tamper with the connection between Alice and Bob. If this happens, then Alice and Bob can abort the protocol and throw away the random key generated so far. This does not protect private randomness, but allows for detecting an attack and aborting – the next best thing. This is a striking advantage over conventional key distribution protocols where ‘person in the middle’ attacks can not be detected on the fundamental level.

The setup for the BB84 key exchange protocol is depicted in Fig. 2.3. At the core is a classical identity channel, like the one presented in Eq. (2.2). Alice (right) uses private randomness to sample a random bit  $r$ , imprints this into a qubit and sends this qubit to Bob. Bob can perform the readout and perfectly recovers this bit. Here, it is also useful to think in terms of photons as carriers of quantum information: photons can cover a lot of distance in short time and are ideally suited for this type of information transmission protocol. But, a mere identity channel is not secure. In particular, it does not allow to detect actions of a potential eavesdropper in the middle. This is where the green quantum gate boxes come into play. They are a placeholder for one of two possible actions: (i) do nothing ( $\mathbb{I}$ ) or (ii) enter/leave superposition ( $\mathbf{H}$ ). Alice and Bob toss a private random coin ( $a, b \stackrel{\text{unif}}{\sim} \{0, 1\}$ ) to decide which action they apply. This gives rise to four potential quantum circuits that each occur

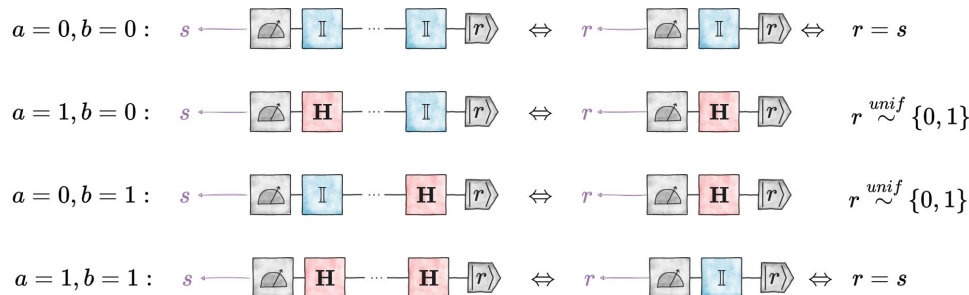
QKD = Quantum Key Distribution

use ‘quantum fragility’ to detect eavesdroppers



**Figure 2.3** Schematic illustration of the BB84-protocol: Alice and Bob use a single-qubit circuit to communicate a single bit of information (purple). Each player obfuscates this identity channel by either applying  $I$  (do nothing) or  $H$  (enter/leave superposition). If both players happen to apply the same gate, the transmission is perfect ( $s = r$ ). Otherwise, the output bit is completely random ( $s \stackrel{unif}{\sim} \{0,1\}$ ). This dichotomy is impossible to achieve classically and allows for sharing a key *and* detecting person in the middle attacks.

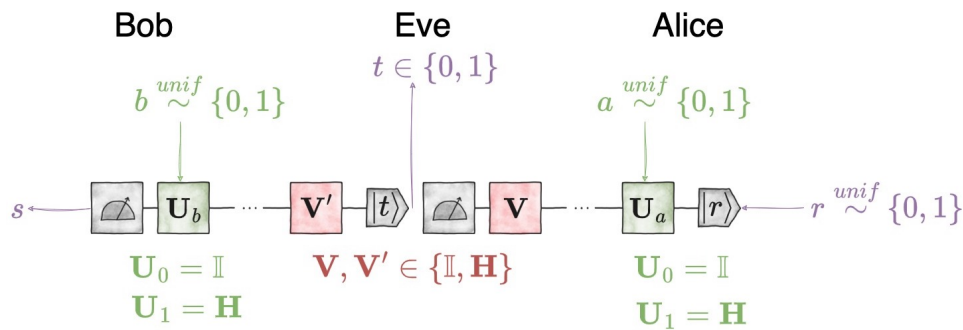
with probability 1/4:



Note that whenever  $a = b$ , the effective classical-quantum-classical channel transmits the original bit perfectly, i.e.  $s = r$ . If instead  $a \neq b$ , then a uniformly random bit is produced, i.e.  $r \stackrel{unif}{\sim} \{0,1\}$ . Such effective classical-quantum-classical channels are completely useless, because they erase all information about the initial bit. This is an interesting and nontrivial dichotomy: whenever Bob correctly guesses Alice's obfuscation, he can recover her bit perfectly. Otherwise, he effectively destroys Alice's (qu)bit and produces a random bit instead.

Why is this noteworthy? Well, this dichotomy also affects any potential eavesdropper who intercepts the qubit somewhere in the middle, see Fig. 2.4 for an illustration. Either, Eve guesses Alice's obfuscation correctly. Or, she destroys the underlying bit message and re-initializes the qubit to an uncorrelated, random bit value. And here is where things get interesting. If Alice uses private randomness to decide her obfuscation action, then Eve has no way to anticipate her move. She must make the wrong guess in about half the cases.

BB84-protocol allows for transmitting keys & detecting eavesdroppers



**Figure 2.4** Illustration of an eavesdropper attack on the BB84 protocol: a third party (Eve, red) intercepts the qubit in the middle. She performs qubit readout to obtain her own bit  $t$ , before re-initializing the qubit to send it on to Bob. In order to maximize her chances of learning  $r$  and to obfuscate her attack, she can also apply two quantum gates  $\mathbf{V}$  and  $\mathbf{V}'$  that resemble the gate actions of Alice and Bob.

Crucially, whenever she guesses wrong, she re-inserts a random bit value to Bob. And this can now be detected. After all, Alice and Bob expect to get perfectly correlated bits whenever they happen to execute the same obfuscation gate (which happens in about  $1/2$  of all cases):  $r = s$ . An interception of Eve in the middle must break this perfect correlation. And repeating the protocol sufficiently often (with new private randomness in each go) will allow Alice+Bob to detect eavesdropping by comparing some of their input/output values over a public channel. More precisely, they execute  $T \gg 1$  rounds of their protocol and, after completion, they broadcast their private random bitstrings  $\mathbf{a}, \mathbf{b} \in \{0, 1\}^T$ . This allows them to discard all uncorrelated bits and focus on the instances where perfect correlations should happen:  $a_t = b_t$  should imply  $s_t = r_t$ . This already shrinks the possible bits to roughly  $T/2$ . They then select certain instances within this remaining string to check whether  $s_t = r_t$  is actually true. This is again achieved via open communication over insecure channels. An instance where  $s_t \neq r_t$  rises suspicion. Assuming the hardware works perfectly as intended (which is a very strong assumption), this must be the signature of an eavesdropper!

This ability to detect actions of an eavesdropper are a core feature of quantum cryptography protocols. Here, we have only scratched the surface and discussed the key ideas behind one of the oldest and most basic protocols of this type. However, at this point, you have all the information you need to execute a proper analysis of the entire protocol – under the additional assumption that Eve intercepts in the middle via readout plus initialization and only uses quantum gates that mimic the possibilities of Alice and Bob.

**Exercise 2.9 (complete analysis of the BB84-protocol).** Perform a rigorous treatment of the BB84-protocol under the additional assumption that Eve's gates are  $\mathbf{V}, \mathbf{V}' \in \{\mathbb{I}, \mathbf{H}\}$ . This results in a total of  $2 \times 4 \times 2 = 16$  different incarnations



of the intercepted protocol – one for each choice of  $a$  (Alice), as well as  $V, V'$  (Eve) and  $b$  (Bob). How many of these incarnations lead to an undetectable and successful attack, i.e.  $s = r$ , as well as  $t = r$ ? How many of these incarnations leave traces of Eve's actions that can be detected by Alice and Bob? And how many leave Eve completely clueless, i.e.,  $t \stackrel{unif}{\sim} \{0, 1\}$ ?

Use your findings to argue that Eve does not stand a chance if Alice+Bob repeat the protocol many times and use private randomness to choose  $a$  and  $b$ .

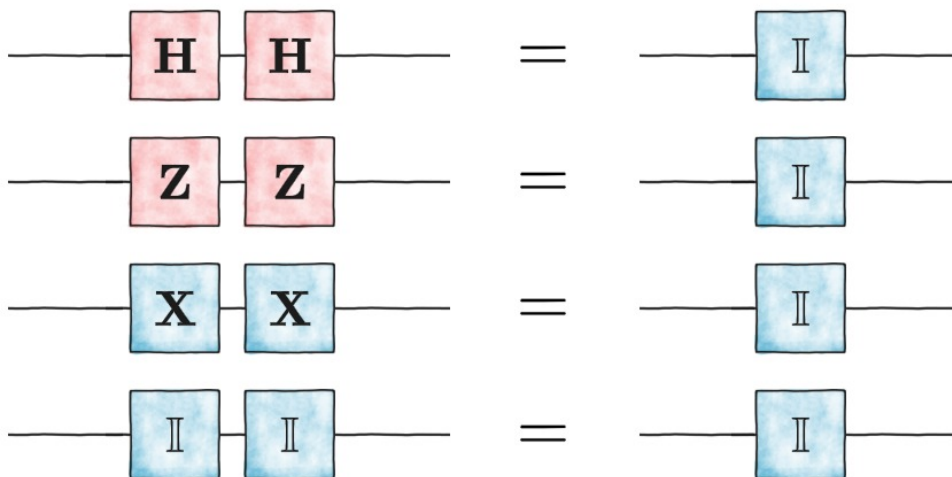
It turns out that the aforementioned assumptions on the type of attack can be removed completely. A proper security analysis for any type of quantum attack did take another 29 years to come up with [Tom+13]. Needless to say, these arguments go beyond the scope of this introductory lecture.

## Problems

**Problem 2.10 (random number generator).** Do the computation in Example 2.8 for input bit  $b = 0$ .

**Problem 2.11 (gate composition rule).** Prove Proposition 8.6.

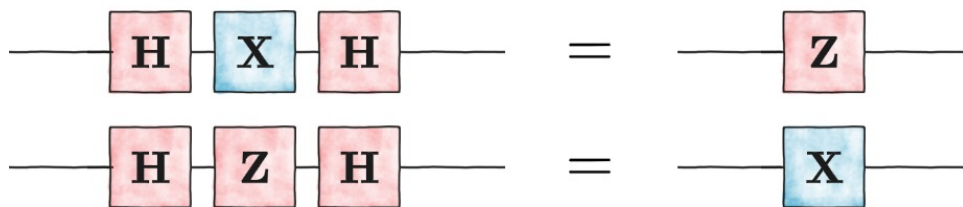
**Problem 2.12 (Reversibility of all gates introduced so far).** Use the sequential gate composition rule (Proposition 8.6) to rigorously prove the following circuit identities:



This, in particular, ensures that all these gates are reversible and to constitute their own inverses.

**Problem 2.13 (some useful circuit identities).** Use the sequential gate composition

rule (Proposition 8.6) to rigorously prove the following circuit identities:



Note that this implies that two Hadamard gates convert  $X$ -gate (bit-flip) into  $Z$ -gate (sign-flip) and vice versa. **Hint:** use the following matrix representations of all gates involved:

$$\mathbf{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \mathbf{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad \text{and} \quad \mathbf{H} = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix}.$$

**Problem 2.14 (matrix representation of the Hadamard gate, see Exercise 2.5).** The matrix representation of the Hadamard gate is given as

$$\mathbf{H} = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix} \in \mathbb{C}^{2 \times 2}.$$

- 1 Show that this matrix is unitary by verifying Eq. (2.12) for  $\mathbf{U} = \mathbf{H}$ .
- 2 The action of a Hadamard gate maps deterministic bit states  $|0\rangle$  and  $|1\rangle$  into uniform superposition states  $|+\rangle$  and  $|-\rangle$ . Use matrix-vector multiplication to verify the following state vector representations:

$$|+\rangle = \mathbf{H}|0\rangle = \mathbf{H}\mathbf{e}_0 = \frac{1}{\sqrt{2}}(\mathbf{e}_0 + \mathbf{e}_1) = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad (2.18)$$

$$|-\rangle = \mathbf{H}|1\rangle = \mathbf{H}\mathbf{e}_1 = \frac{1}{\sqrt{2}}(\mathbf{e}_0 - \mathbf{e}_1) = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \quad (2.19)$$

Both expressions on the very right should be interpreted as ‘both 0 and 1 in equal measure’. Note, however that one combination has a ‘+’ inbetween, while the other one has a ‘-’. This sign difference ensures that both superpositions can be undone again.

- 3 Verify reversibility by showing  $\mathbf{H}^2 = \mathbb{I}$  via matrix-vector multiplication.

**Problem 2.15 (complete analysis of the BB84-protocol, see Exercise 2.9).** Perform a rigorous treatment of the BB84-protocol under the additional assumption that Eve’s gates are  $\mathbf{V}, \mathbf{V}' \in \{\mathbb{I}, \mathbf{H}\}$ . This results in a total of  $2 \times 4 \times 2 = 16$  different incarnations of the intercepted protocol – one for each choice of  $a$  (Alice), as well as  $\mathbf{V}, \mathbf{V}'$  (Eve) and  $b$  (Bob). How many of these incarnations lead to a undetectable and successful attack, i.e.  $s = r$ , as well as  $t = r$ ? How many of these incarnations leave traces of Eve’s actions that can be detected by Alice and Bob? And how many leave Eve completely clueless, i.e.,  $t \stackrel{\text{unif}}{\sim} \{0, 1\}$ ? Use your findings to argue that Eve does not stand a chance if Alice+Bob repeat the protocol many times and use private randomness to choose  $a$  and  $b$ .

## 3. Single qubit circuits II

Date: October 18, 2023

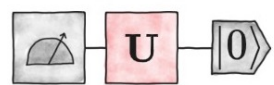
### 3.1 Motivation and outline

Last week (Lecture 1), we started to explore single-qubit quantum logic. We saw that quantum gates allow a native execution of completely new functionalities. Today, we continue along these lines and push single-qubit quantum logic to the ultimate limits of their capabilities. In a certain sense, quantum logic is 'infinitely more expressive' than conventional single-bit logic. As a teaser, we point out that the negation gate  $X$  is the only non-trivial reversible single-bit gate. But, because  $X^2 = \mathbb{I}$  (do nothing), the list of all reversible single-bit circuits is a very short one:


$$\text{---} \boxed{\mathbb{I}} \text{---}, \text{---} \boxed{X} \text{---} \quad (3.1)$$

are the only two options.

In quantum logic, the situation could not be more different. We will see that two (appropriately chosen) elementary quantum gates suffice to generate an *infinite amount of single-qubit circuits* that have distinct logical functionality. Every  $2 \times 2$  unitary matrix is valid in quantum logic:


$$\text{---} \boxed{U} \text{---} \text{---} \boxed{|0\rangle}, \quad \mathbf{U} = \begin{pmatrix} U_{0,0} & U_{0,1} \\ U_{1,0} & U_{1,1} \end{pmatrix}$$
$$\mathbf{U}^\dagger \mathbf{U} = \mathbf{U} \mathbf{U}^\dagger = \mathbb{I}$$

#### Agenda:

- 1 complex numbers
- 2 ultimate limits of single-qubit logic
- 3 list of prominent gates
- 4 application: restricted parity computations
- 5 bonus: Bloch sphere visualization

And, what is more, we can approximate it with only logarithmically many elementary quantum gates (in the desired approximation accuracy). This powerful circuit synthesis result is known as the Solovay-Kitaev theorem. En route to this surprising result, we will need to remember the basic structure and elementary properties of complex numbers. So, today is a good occasion to review them.

### 3.2 Excursion: complex numbers

The field of complex numbers  $\mathbb{C}$  is an extension of the real numbers  $\mathbb{R}$ . Arguably, the most characteristic feature of complex numbers is the *imaginary unit*  $i = \sqrt{-1}$ . It was first introduced to formally solve polynomial equations, most notably  $x^2 = -1$  (which cannot have a solution over real numbers only). Today, we know that the solutions of every polynomial equation can be expressed as complex numbers, i.e. combinations of purely real and imaginary numbers:

$$a + ib \quad \text{with } a, b \in \mathbb{R}.$$

complex numbers have  
real+imaginary parts

This deep result is known as the fundamental theorem of algebra. The collection of all such  $z$ 's forms the field of complex numbers  $\mathbb{C}$ . Much like normal numbers, we can *add* complex numbers using familiar rules:

$$z + z' = (a + ib) + (a' + ib') = (a + a') + i(b + b'),$$

addition & multiplication

i.e. we add real and imaginary parts of complex numbers separately. *Multiplication* takes a bit more work and we must also use the formal definition  $i^2 = -1$  to obtain

$$\begin{aligned} z \times z' &= (a + ib)(a' + ib) = a \times a' + ib \times a' + ia \times b' + i^2 b \times b' \\ &= (a \times a' - b \times b') + i(a \times b' + b \times a'). \end{aligned} \quad (3.2)$$

This is a bit cumbersome in the real+imaginary part representation. We will soon discuss another representation which makes multiplication much easier. For now, we point out that complex numbers come with a new operation called *complex conjugation*:

$$\bar{z} = \overline{(a + ib)} = a - ib.$$

complex conjugation

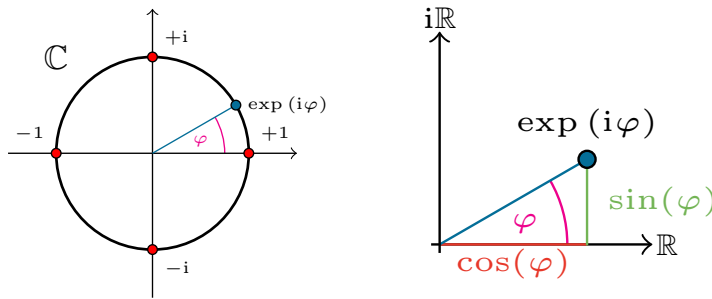
This operation flips the sign of the imaginary part. This operation is actually trivial for numbers that are real-valued to begin with:  $\bar{a} = a$  for all  $a \in \mathbb{R}$ . Complex conjugation allows us to define the *absolute value* of a complex number:

$$|z| = \sqrt{\bar{z} \times z} = \sqrt{(a - ib) \times (a + ib)} = \sqrt{a^2 + b^2}.$$

This looks like the (Euclidean) length of a 2-dimensional (real-valued) vector  $\mathbf{z} \in \mathbb{R}^2$ . Such an analogy between complex numbers  $z \in \mathbb{C}$  and 2D vectors  $\mathbf{z} \in \mathbb{R}^2$  is no coincidence. We can obtain a lot of geometric intuition by envisioning

complex plane

$$z = a + ib \quad \text{as} \quad \mathbf{z} = \begin{pmatrix} a \\ b \end{pmatrix}.$$



**Figure 3.1** Complex unit circle (left) and Euler's formula (right): (Left) It is instructive to view the field of complex numbers as a 2-dimensional plane (real+imaginary part). The unit circle within this plane contains complex numbers  $z \in \mathbb{C}$  with absolute value  $|z| = 1$ . These are called complex phases and can be parametrized by a single angle  $\varphi$ . (Right) Euler's theorem provides the justification for our notation of complex phases:  $\exp(i\varphi) = \cos(\varphi) + i \sin(\varphi)$ .

The  $x$ -coordinate tabulates the purely real part ( $\mathbb{R}$ ), while the  $y$ -coordinate tabulates the purely imaginary part ( $i\mathbb{R}$ ). General complex numbers have nontrivial coordinates in both and therefore live in a *complex plane*. Prominent real and imaginary numbers obey  $|z| = 1$ , e.g.  $|+1| = |-1| = 1$  and also  $|+i| = |-i| = 1$ . These four points therefore live on the unit circle within the complex plane. We refer to Fig. 3.1 (left) for a visual illustration.

Complex numbers  $z \in \mathbb{C}$  that obey  $|z| = \sqrt{\bar{z}z} = 1$  are called (*complex*) *phases*. These are points on the complex unit circle and can be conveniently represented by a single angle  $\varphi \in [0, 2\pi)$  in radians<sup>1</sup>:

$$z = \begin{pmatrix} \cos(\varphi) \\ \sin(\varphi) \end{pmatrix} \in \mathbb{R}^2 \quad \Leftrightarrow \quad z = \cos(\varphi) + i \sin(\varphi) \in \mathbb{C}.$$

Finally, we can use *Euler's theorem* to compactly express the right-hand representation as a single exponential:

$$\exp(i\varphi) = \cos(\varphi) + i \sin(\varphi). \quad (3.3)$$

The geometric intuition behind this celebrated result is displayed in Fig. 3.1 (right). We leave a rigorous proof as an instructive exercise at the end of this section.

**Example 3.1 ('showoff' formula).** The following mathematical formula combines  $i$ ,  $e$  and  $\pi$  in a perfectly correct fashion:

$$e^{i\pi} = \exp(i\pi) = -1.$$

It describes an angle representation of the 'east pole' ( $-1$ ) on the complex unit circle (recall that  $180^\circ \leftrightarrow \pi$ ). ■

<sup>1</sup>Here are the conversion rules for the four most important angles:  $0^\circ \leftrightarrow 0$ ,  $90^\circ \leftrightarrow \pi/2$ ,  $180^\circ \leftrightarrow \pi$ ,  $270^\circ \leftrightarrow 3\pi/4$  and  $360^\circ \leftrightarrow 2\pi$ .

complex phase

The last thing we need to know about complex phases is that they play nicely with each other when it comes to multiplication. Let  $\exp(i\varphi) = \cos(\varphi) + i \sin(\varphi)$  and  $\exp(i\varphi') = \cos(\varphi') + i \sin(\varphi')$  be two complex phases. We can use the multiplication rule (3.2) and trigonometric identities to compute

$$\begin{aligned} \exp(i\varphi) \times \exp(i\varphi') &= (\cos(\varphi) + i \sin(\varphi)) \times (\cos(\varphi') + i \sin(\varphi')) \\ &= (\cos(\varphi) \cos(\varphi') - \sin(\varphi) \sin(\varphi')) \\ &\quad + i (\cos(\varphi) \sin(\varphi') + \sin(\varphi) \cos(\varphi')) \\ &= \cos(\varphi + \varphi') + i \sin(\varphi + \varphi') \\ &= \exp(i(\varphi + \varphi')). \end{aligned}$$

This confirms a famous property of exponential functions in the realm of complex numbers: multiplication of exponentials is the same as adding the exponents. The concept of a complex phase and this multiplication rule will be important in our study of quantum circuits. It deserves a prominent display.

**Fact 3.2 (complex phase).** A complex number  $z \in \mathbb{C}$  is called a *complex phase* if  $|z| = \sqrt{\bar{z} \times z} = 1$ . It can be represented as  $z = \exp(i\varphi)$ , where  $\varphi \in [0, 2\pi]$  is a single angle. Multiplication of two complex phases is the same as adding the corresponding angles, i.e.

$$z \times z' = \exp(i\varphi) \times \exp(i\varphi') = \exp(i(\varphi + \varphi')).$$

multiplication rule for complex phases

Let us do a simple example that cycles through the four prominent points on the complex unit circle:

$$\begin{aligned} i &= \exp(i\pi/2), \\ i^2 &= i \times i = \exp(i\pi/2) \times \exp(i\pi/2) = \exp(i(\pi/2 + \pi/2)) = \exp(i\pi) = -1, \\ i^3 &= i \times i^2 = \exp(i\pi/2) \times \exp(i\pi) = \exp(i(\pi/2 + \pi)) = \exp(i3\pi/2) = -i, \\ i^4 &= i \times i^3 = \exp(i\pi/2) \times \exp(i3\pi/2) = \exp(i(\pi/2 + 3\pi/2)) = \exp(i2\pi) = 1. \end{aligned}$$

**Exercise 3.3 (Proof of Euler's theorem).** Prove Eq. (3.3) by using  $i^2 = -1$  and the following three Taylor series expansions:

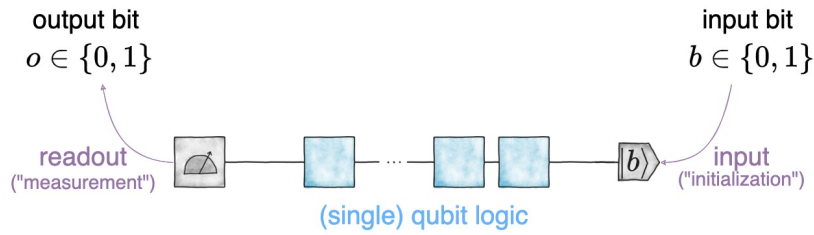
$$\exp(z) = \sum_{k=0}^{\infty} \frac{z^k}{k!}, \quad \cos(z) = \sum_{k=0}^{\infty} (-1)^k \frac{z^{2k}}{(2k)!}, \quad \sin(z) = \sum_{k=0}^{\infty} (-1)^k \frac{z^{2k+1}}{(2k+1)!}.$$

### 3.3 Ultimate limits of single-qubit logic

#### 3.3.1 Recapitulation

Recall that a single-qubit quantum processor maps bitstrings to bitstrings. An initial bit value  $b \in \{0, 1\}$  is used to initialize the *qubit state vector*

$$|\psi\rangle = |0\rangle = \mathbf{e}_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{or} \quad |\psi\rangle = |1\rangle = \mathbf{e}_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$



**Figure 3.2** Schematic illustration of a single-qubit processor from Lecture 1 (see also Fig. 2.2 there): a single-qubit processor takes a single bit  $b \in \{0, 1\}$  as input and produces a single-bit output  $o \in \{0, 1\}$ . The logic in-between is executed on the quantum level, where different logical operations become available. The readout stage is also special and can give rise to true randomness.

This qubit state vector is a 2-dimensional vector with complex entries that can be used to keep track of the quantum logic content when we apply quantum gates. The action of each single-qubit gate is described by a complex-valued matrix  $\mathbf{U} \in \mathbb{C}^{2 \times 2}$  that is also unitary, i.e.  $\mathbf{U}^\dagger \mathbf{U} = \mathbf{U} \mathbf{U}^\dagger = \mathbb{I}$ . The action of one gate matrix  $\mathbf{U}$  on state vector  $|\psi\rangle = \boldsymbol{\psi}$  can be computed with matrix-vector multiplication:

$$\boldsymbol{\psi}' = \mathbf{U} \boldsymbol{\psi} = \begin{pmatrix} U_{0,0} & U_{0,1} \\ U_{1,0} & U_{1,1} \end{pmatrix} \begin{pmatrix} \psi_0 \\ \psi_1 \end{pmatrix} = \begin{pmatrix} U_{0,0}\psi_0 + U_{0,1}\psi_1 \\ U_{1,0}\psi_0 + U_{1,1}\psi_1 \end{pmatrix}.$$

This rule readily extends to the application of  $T \gg 1$  different gates:  $|\psi'\rangle = \boldsymbol{\psi}' = \mathbf{U}_{T-1} \times \cdots \times \mathbf{U}_1 \times \mathbf{U}_0 \boldsymbol{\psi}$ , where  $\times$  denotes matrix-matrix multiplication. The last ingredient concerns the readout stage. If the final state is  $\boldsymbol{\psi}'$ , then the probability of obtaining outcome bit  $o = 0$  and  $o = 1$  is

$$p_0 = \Pr_{|\psi\rangle} [o = 0] = |\psi_0|^2 \quad \text{and} \\ p_1 = \Pr_{|\psi\rangle} [o = 1] = |\psi_1|^2.$$

Note that each formula features the absolute value of a complex number. And such absolute values are invariant under multiplication with any complex phase  $\exp(i\varphi)$ :

$$p_0 = |\psi_0|^2 = |\exp(i\varphi) \psi_0|^2 \quad \text{and} \quad p_1 = |\psi_1|^2 = |\exp(i\varphi) \psi_1|^2.$$

This has an important consequence. Our description of quantum logic in terms of matrix-vector multiplication carries redundant information. Indeed, the overall complex phase does not matter at all. This affects both state vectors and gate matrices.

**Example 3.4 (states).** The following four state vectors all describe a logical 0 bit ('everything is in zero'):

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad +i|0\rangle = \begin{pmatrix} i \\ 0 \end{pmatrix}, \quad -|0\rangle = \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \quad -i|0\rangle = \begin{pmatrix} -i \\ 0 \end{pmatrix}.$$

outcome probabilities don't depend on complex phases

■

**Example 3.5 (gates).** The following two unitary matrices describe the same gate action:

$$\begin{pmatrix} \exp(-i\pi/8) & 0 \\ 0 & \exp(i\pi/8) \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1 & 0 \\ 0 & \exp(i\pi/4) \end{pmatrix}.$$

■

**Non-example 3.6 (states).** The following two states are *not equivalent*

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \begin{pmatrix} +1/\sqrt{2} \\ +1/\sqrt{2} \end{pmatrix} \quad \text{and} \quad |i+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle) = \begin{pmatrix} 1/\sqrt{2} \\ i/\sqrt{2} \end{pmatrix},$$

because there is no way to write  $|i+\rangle$  as  $\exp(i\varphi)|+\rangle$ .

■

This *phase invariance* of both state vectors and gate actions is an important feature of quantum computing and also deserves a prominent display.

**Fact 3.7 (phase invariance of state vectors and gate matrices).** Two state vectors  $|\psi\rangle, |\psi'\rangle$  encode the same information content if  $|\psi'\rangle = \exp(i\varphi)|\psi\rangle$  for some  $\varphi \in [0, 2\pi)$ . Likewise, two gate matrices  $\mathbf{U}, \mathbf{U}'$  encode the same action if  $\mathbf{U}' = \exp(i\varphi')\mathbf{U}$  for some  $\varphi' \in [0, 2\pi)$ . If this is the case, we succinctly write

$$|\psi'\rangle = \boldsymbol{\psi}' \sim \boldsymbol{\psi} = |\psi\rangle \quad \text{and also} \quad \mathbf{U}' \sim \mathbf{U}.$$

phase invariance of state vectors & gate matrices

### 3.3.2 Clifford gates

In the introduction, we have already emphasized that there are only two non-trivial single-bit circuits:  $\mathbb{I}$  (do nothing) and  $\mathbf{X}$  (bit-flip), see Eq. (3.1). Let us now explore what happens in the quantum case. Let's start with the two most prominent single-qubit gates that are featured in Lecture 1:

$$\mathbf{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

The quantum circuit model allows us to combine these *elementary gates* to construct new quantum logical functionalities. Note that many gate combinations must be trivial because both  $\mathbf{X}$  and  $\mathbf{H}$  are reversible. In particular,

$$\mathbf{X}^2 = \mathbf{X} \times \mathbf{X} = \mathbb{I} \quad \text{and} \quad \mathbf{H}^2 = \mathbf{H} \times \mathbf{H} = \mathbb{I},$$

which tells us that long sequences of only  $\mathbf{X}$  or only  $\mathbf{H}$  don't accomplish anything. Indeed,  $\mathbf{X}^D = \mathbf{X}$  if  $D$  is odd and  $\mathbf{X}^D = \mathbb{I}$  else if  $D$  is even. Likewise,  $\mathbf{H}^D = \mathbf{H}$  if  $D$  is odd and  $\mathbf{H}^D = \mathbb{I}$  else. Truly new logical functionalities can only be achieved if we alternate  $\mathbf{X}$  and  $\mathbf{H}$  gates. For instance, we can create



the following ‘cousins’ of the Hadamard gate:

$$\begin{array}{l}
 \text{---} \boxed{\text{H}} \text{---} \\
 \text{---} \boxed{\text{H}} \boxed{\text{X}} \text{---} \\
 \text{---} \boxed{\text{X}} \boxed{\text{H}} \boxed{\text{X}} \text{---} \\
 \text{---} \boxed{\text{X}} \boxed{\text{H}} \text{---}
 \end{array}
 \quad
 \begin{array}{l}
 \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\
 \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \\
 \frac{1}{\sqrt{2}} \begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix} \\
 \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}
 \end{array}
 \quad . \quad (3.4)$$

Each of them features the minus sign in a different location. We can also sandwich the  $X$  gate between two  $H$ s to obtain the *sign-flip gate*:

sign-flip gate  $Z$

$$\text{---} \boxed{\text{H}} \boxed{\text{X}} \boxed{\text{H}} \text{---} \quad \mathbf{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} . \quad (3.5)$$

This matrix  $Z$  is also called the *Pauli-z* gate, while  $X$  is known as the *Pauli-x* gate. Finally, we can combine  $X$  and  $Z$  to get an action that is equivalent to the *Pauli-y* gate  $Y$ :

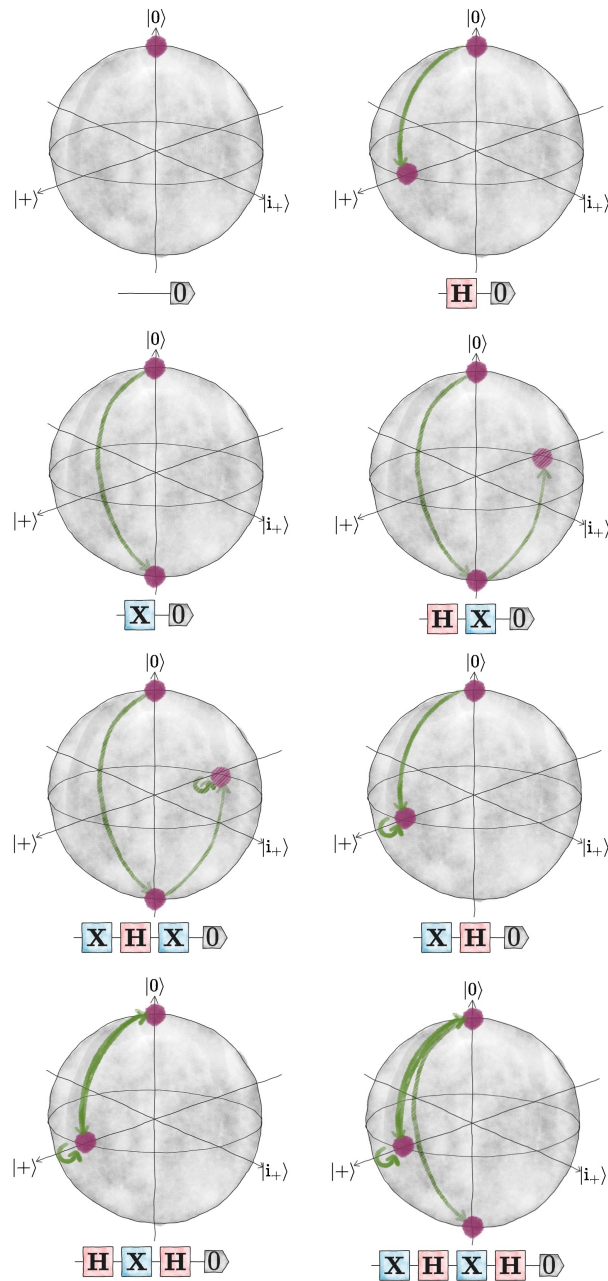
$$\mathbf{XZ} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} = (-i) \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \sim \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = \mathbf{Y}, \quad (3.6)$$

where we have used the gate invariance under global phases from Fact 3.7. This phase invariance of quantum gate actions, in fact, also implies that we are done. We cannot reach any new functionalities anymore. Also, note that Eq. (3.5) allows us to replace the elementary  $X$ -gate with an elementary  $Z$ -gate. After all, we can transform one into the other by investing two Hadamard gates.

**Lemma 3.8** The two elementary gates  $X$  (bit-flip) and  $H$  (Hadamard) generate a total of 8 functionally distinct quantum functionalities: one identity ( $I$ ), three Pauli gates ( $X, Y, Z$ ) and four Hadamard-type gates displayed Eq. (3.4).

We leave a proof of this technical statement as an instructive exercise that can be easily automated. and instead refer to Fig. 3.3 for a visualization of this statement. It uses a geometric analogy between single-qubit state vectors (with complex coefficients and phase invariance) and real-valued 3D-vectors that are confined to a sphere. In this *Bloch sphere representation*, antipodal points are

Bloch sphere representation of single-qubit state vectors



**Figure 3.3** All 8 functionally distinct quantum gates formed from only the gates  $X$  and  $H$  applied to the zero  $|0\rangle$  state visualized on a Bloch Sphere.

always orthogonal to each other. E.g.  $|0\rangle$  and  $|1\rangle$  from north and south pole of the sphere and also obey  $\langle 0|1\rangle = \mathbf{e}_0^\dagger \mathbf{e}_1 = 0$  (orthogonality).

**Exercise 3.9 (Proof of Lemma 3.8).** Verify the correctness of Lemma 3.8 by writing a piece of code that generates gate combinations of a certain length, computes their matrix representation via matrix-matrix multiplication and terminates once no new functionally distinct matrices can be achieved (i.e. all newly generated matrices are equivalent to existing ones up to a global phase).

Lemma 3.8 lists a total of 8 different logical functionalities that can be obtained from combining two elementary quantum gates:  $\mathbf{H}$  and  $\mathbf{Z}$  (or, equivalently:  $\mathbf{H}$  and  $\mathbf{X}$ ). Let us now see if we can push this number even further if we replace  $\mathbf{Z}$  with another quantum gate. The *phase gate*

phase gate

$$\mathbf{S} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$$

introduces complex numbers into our gate model. It is also closely related to the sign-flip gate. Indeed,

$$\mathbf{S}^2 = \mathbf{S} \times \mathbf{S} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & i^2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \mathbf{Z},$$

so it is instructive to think of  $\mathbf{S}$  as a ‘square root’ of the sign-flip gate  $\mathbf{Z}$ . Higher powers of  $\mathbf{S}$  lead to lower right matrix entries that continue to jump around the complex unit circle:

$$\mathbf{S}^3 = \begin{pmatrix} 1 & 0 \\ 0 & -i \end{pmatrix} \quad \text{and finally} \quad \mathbf{S}^4 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \mathbb{I}.$$

The phase gate is the first gate we encounter that is not its own reverse. If we want to undo the action of  $\mathbf{S}$ , we must apply  $\mathbf{S}^\dagger = \mathbf{S}^3$ . A single application of the phase gate inserts complex numbers into rows or columns of existing gate matrix descriptions. Whether it is rows or columns depends on the ordering of gates. For instance,

$$\begin{aligned} \mathbf{S} \times \mathbf{H} &= \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \times \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ i & -i \end{pmatrix}, \quad \text{while} \\ \mathbf{H} \times \mathbf{S} &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & i \\ 1 & -i \end{pmatrix}. \end{aligned}$$

We can now use a combination of  $\mathbf{S}$  and  $\mathbf{H}$  to generate the actual Pauli-Y matrix from Eq. (3.6):

$$\mathbf{Y} = \mathbf{S} \times \mathbf{H} \times \mathbf{Z} \times \mathbf{H} \times \mathbf{S}^3.$$

We leave a verification of this formula as an instructive exercise. By now, it should not be a surprise that replacing  $\mathbf{Z}$  with  $\mathbf{S}$  (its square root) allows us to build more stuff. However, the total number of different functionalities is still finite.

**Definition 3.10 (single-qubit Clifford gates).** The two gates  $H$  (Hadamard) and  $S$  (phase) generate a total of 24 different single-qubit gate functionalities. These are called (*single-qubit*) *Clifford gates*.

Table 3.1 provides a complete list of these 24 operations and a way of how to realize them using Pauli rotations – the topic of today’s final chapter.

### 3.3.3 Universal gate sets

We now have seen that two quantum gates already generate many different logical functionalities – many more than the two nontrivial single-bit circuits  $I$  and  $X$ . Single-qubit Clifford gates, for instance, comprise a total of 24 gates with different quantum logic. It is therefore natural to wonder if we can do even better. Let us see if we can apply the trick from another time. More precisely, we replace  $S$  with its ‘square root’ which is called the *T-gate*:

T-gate

$$T = \begin{pmatrix} 1 & 0 \\ 0 & \exp(i\pi/4) \end{pmatrix} \quad \text{such that} \quad T^2 = \begin{pmatrix} 1 & 0 \\ 0 & \exp(i\pi/2) \end{pmatrix} = S,$$

where we have used the rule for multiplying complex phases from Fact 3.2, as well as  $\exp(i\pi/2) = +i$  (see e.g. Fig. 3.1 (left)). Euler’s formula (3.3) also implies

$$\exp(i\pi/4) = \cos(\pi/4) + i \sin(\pi/4) = (1 + i) / \sqrt{2},$$

so this complex phase contains both a real-valued and an imaginary contribution. Due to  $S = T^2$ , replacing  $S$  by  $T$  can only increase the number of quantum functionalities that can be reached. The actual gain is astonishing.

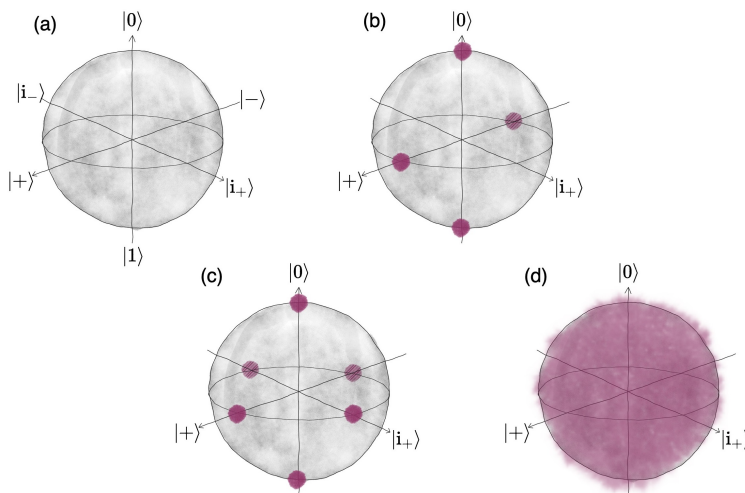
**Theorem 3.11 (universal gate set).** Together, the elementary Clifford gates  $H$ ,  $S$  and the T-gate  $T$  form a *universal gate set*: Every  $2 \times 2$  unitary matrix can be approximated to an arbitrary degree with sequences comprised of *only*  $H$  and  $T$  (we actually don’t need  $S$ , because  $S = T^2$ ).

Hadamard+T generate every  $2 \times 2$  unitary matrix

The following powerful statement stems from group theory and a proof would go beyond the scope of this lecture. Here, we instead emphasize the implications. There are infinitely many unitary  $2 \times 2$  matrices that are distinct from each other (even if we take into account phase invariance). Nonetheless, we can approximate each and every such matrix with sequential quantum circuits that only feature  $H$  (Hadamard) and  $T$  (T-gate). What is more, such universal gate sets are easy to find. The elementary Clifford gates  $\{H, S\}$  only need one additional non-Clifford gate  $U$  to become universal<sup>2</sup>. The precise nature of this additional third gate  $U$  does not matter at all!

Theorem 3.11 is interesting and tells us something about the ultimate possibilities of single-qubit logic. However it does not tell us anything about the cost associated with realizing this apparent potential. A natural cost parameter for quantum circuits is *circuit depth*: how many layers of either  $H$  or  $T$  are

<sup>2</sup>For technical reasons, we might actually need one additional gate  $U$ , as well as its inverse  $U^\dagger$ .



**Figure 3.4** (a) A Bloch sphere showing the labeling of the axis. (b) The single qubit quantum states reached with only combinations of  $X$  and  $H$ , see Lemma 3.8. (c) The single qubit quantum states reached with only Clifford gates, see Definition 3.10. (d) The single qubit quantum states reached with arbitrary long combinations of Clifford gates and the  $T$  gate as stated in Theorem 3.11.

required to a given target unitary  $U$  up to accuracy  $\epsilon \in (0, 1)$ ? This quantum synthesis problem has a surprisingly strong answer that is valid at a remarkable level of generality [Kit97; DNO5].

**Theorem 3.12 (efficient single-qubit synthesis (Solovay-Kitaev Theorem)).** Let  $\mathcal{G}$  be a universal gate set<sup>a</sup>, e.g.  $\mathcal{G} = \{H, T\}$  and let  $\epsilon$  be a desired approximation accuracy. Then, for every unitary  $2 \times 2$  matrix  $U$ , there exists a sequence of (at most)

$$D = \mathcal{O}(\log^c(1/\epsilon)) \quad (3.7)$$

that approximates the action of  $U$  up to accuracy  $\epsilon$ . Here,  $c \in [1, 3 + o(1)]$  is a constant that depends on the universal gate set in question.

<sup>a</sup>The original statement also requires that this gate set either contains inverses or can generate them in a constant number of steps.

Some elementary gate sets even achieve  $c = 1$ , in which case Eq. (4.12) is tight up to a constant factor. We emphasize that the required circuit depth only scales logarithmically in the desired target accuracy. Or, to put it differently: (after some burn-in period,) the approximation error of a quantum circuit approximation diminishes exponentially in the circuit depth one is willing to invest.

We have come a long way so far. Starting with  $H$  and  $X$  (or  $Z$ ), we found out that we can generate a total of 8 different quantum functionalities. This number went up to 24 after replacing  $X$  with the phase gate  $S$ . Moving from

efficient single-qubit circuit synthesis

$H$  and  $S$  to  $H$  and  $T$  – the ‘square root’ of  $S$  – had even more disruptive consequences. These two gates can generate every  $2 \times 2$  unitary matrix. We refer to Fig. 3.4 for an illustration of this journey.

### 3.4 Pauli rotation gates

Let us now move on and discuss a different approach towards quantum gates that is more analog in nature. This is also how modern quantum hardware executes quantum logic. The basic building blocks are the 3 *Pauli matrices*:

3 Pauli matrices

$$\mathbf{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \mathbf{Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \quad (3.8)$$

They describe important quantum logical operations, like bit-flip ( $\mathbf{X}$ ), sign-flip ( $\mathbf{Y}$ ) and a combination of both ( $\mathbf{Y} = i\mathbf{XZ} \sim \mathbf{XZ}$ ). However, Pauli matrices also feature prominently in the quantum physical formalism that is used to describe individual qubits<sup>3</sup>. They can also be used to generate qubit rotations along different axes. Let  $\theta \in [0, 2\pi]$  be an angle and define

Pauli rotations

$$\begin{aligned} \mathbf{R}_x(\theta) &= \exp(-i(\theta/2)\mathbf{X}) = \begin{pmatrix} \cos(\theta/2) & -i \sin(\theta/2) \\ -i \sin(\theta/2) & \cos(\theta/2) \end{pmatrix} \quad (\mathbf{X}\text{-rotation by } \theta), \\ \mathbf{R}_y(\theta) &= \exp(-i(\theta/2)\mathbf{Y}) = \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{pmatrix} \quad (\mathbf{Y}\text{-rotation by } \theta), \\ \mathbf{R}_z(\theta) &= \exp(-i(\theta/2)\mathbf{Z}) = \begin{pmatrix} \exp(-i\theta/2) & 0 \\ 0 & \exp(i\theta/2) \end{pmatrix} \quad (\mathbf{Z}\text{-rotation by } \theta). \end{aligned}$$

We leave the derivation of these matrix descriptions as an instructive exercise in matrix analysis. For now, we emphasize that one should really view these three operations as rotations. The following technical statement explains why.

**Lemma 3.13** All three Pauli rotations are  $2\pi$ -periodic (up to a global phase) and obey an angle addition rule. In formulas, we have for  $w = x, y, z$ ,

$$\mathbf{R}_w(2\pi) \propto \mathbf{R}_w(0) = \mathbb{1} \quad \text{and} \quad \mathbf{R}_w(\theta)\mathbf{R}_w(\theta') = \mathbf{R}_w(\theta + \theta'), \quad (3.9)$$

regardless of  $\theta, \theta' \in \mathbb{R}$ .

Again, we leave a rigorous derivation as an instructive exercise. Note that this angle addition rule is very similar to the phase multiplication rule from Fact 3.2. Lemma 3.13 ensures  $R_w(\theta) \propto R_w(\theta \bmod 2\pi)$  ( $2\pi$ -periodicity) and also completely specifies the reverse operation:

$$\mathbf{R}_w(\theta)^{-1} = \mathbf{R}_w(-\theta) \quad \text{because} \quad \mathbf{R}_w(\theta)\mathbf{R}_w(-\theta) = \mathbf{R}_w(\theta - \theta) = \mathbf{R}_w(0) = \mathbb{1}.$$

Now suppose that we execute multiple Pauli rotations around the same axis. Then, a similar argument highlights that the order of rotations does not matter:

$$\mathbf{R}_w(\theta)\mathbf{R}_w(\theta') = \mathbf{R}_w(\theta + \theta') = \mathbf{R}_w(\theta' + \theta) = \mathbf{R}_w(\theta')\mathbf{R}_w(\theta).$$

<sup>3</sup>They are related to the *spin* of a quantum particle across the different coordinate axes in 3-dimensional space.

This means that we can accumulate and/or permute different Pauli rotations of the same kind at will – an extremely useful feature. The following circuit visualization depicts a straightforward extension of this formula to  $D + 1$  different rotations:

rotations across same axis  
play nicely

$$\begin{array}{c} \text{---} \boxed{\mathbf{R}_w(\theta_D)} \text{---} \cdots \text{---} \boxed{\mathbf{R}_w(\theta_0)} \text{---} \\ \text{---} \boxed{\mathbf{R}_w(\theta_{\text{tot}})} \text{---} \end{array} = \quad , \quad (3.10)$$

$\theta_{\text{tot}} = \theta_0 + \cdots + \theta_D$

for  $w = x, y, z$  and  $\theta_0, \dots, \theta_D \in [0, 2\pi)$  arbitrary. Note, however, that this is only true for rotations along the same axis. Pauli rotations across different axes don't have this feature:

$$\mathbf{R}_w(\theta)\mathbf{R}_{w'}(\theta') \neq \mathbf{R}_{w'}(\theta')\mathbf{R}_w(\theta) \quad \text{whenever } w \neq w'$$

for almost all angles  $\theta, \theta' \in [0, 2\pi)$ . The following diagrammatic reformulation reminds us that order matters a lot in general:

$$\text{---} \boxed{\mathbf{R}_w(\theta)} \boxed{\mathbf{R}_{w'}(\theta')} \text{---} \not\equiv \text{---} \boxed{\mathbf{R}_{w'}(\theta')} \boxed{\mathbf{R}_w(\theta)} \text{---}$$

$w \neq w'$

We conclude this section by reproducing some of the gates we have seen today with Pauli rotations.

- 1 The *T-gate*  $\mathbf{T}$  is equivalent to a  $z$ -rotation with angle  $\theta = \pi/4$ :

$$\mathbf{R}_z(\pi/4) = \begin{pmatrix} \exp(-i\pi/8) & 0 \\ 0 & \exp(i\pi/8) \end{pmatrix} \sim \begin{pmatrix} 1 & 0 \\ 0 & \exp(i\pi/4) \end{pmatrix} = \mathbf{T}.$$

- 2 The *phase gate*  $\mathbf{S}$ -gate is equivalent to a  $z$ -rotation with angle  $\theta = \pi/2$ :

$$\mathbf{R}_z(\pi/2) = \begin{pmatrix} \exp(-i\pi/4) & 0 \\ 0 & \exp(i\pi/4) \end{pmatrix} \sim \begin{pmatrix} 1 & 0 \\ 0 & \exp(i\pi/2) \end{pmatrix} = \mathbf{S}.$$

This angle is exactly twice as large as the angle required for the T-gate.

- 3 The *sign gate*  $\mathbf{Z}$  is equivalent to a  $z$ -rotation with angle  $\theta = \pi$ :

$$\mathbf{R}_z(\pi) = \begin{pmatrix} \exp(-i\pi/2) & 0 \\ 0 & \exp(i\pi/2) \end{pmatrix} \sim \begin{pmatrix} 1 & 0 \\ 0 & \exp(i\pi) \end{pmatrix} = \mathbf{Z}.$$

This angle is exactly twice as large as the angle required for the phase gate and four times as large as the angle required for the T-gate.

- 4 The *Hadamard gate*  $\mathbf{H}$  is equivalent to a combination of an  $x$ -rotation and a  $y$ -rotation with angle  $\pi/2$  and  $\pi/4$  each. The following computation periodicity and  $\cos(-\pi/2) = 0$ ,  $\sin(-\pi/2) = -1$  and  $\cos(\pi/4) = \sin(\pi/4) = 1/\sqrt{2}$ :

No.	Rotation composition	Matrix
1	$\mathbb{I}$	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
2	$X$	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
3	$Y$	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$
4	$Z$	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
5	$R_z\left(\frac{\pi}{2}\right)$	$\frac{\sqrt{2}}{2} \begin{pmatrix} 1-i & 0 \\ 0 & 1+i \end{pmatrix}$
6	$R_z\left(-\frac{\pi}{2}\right)$	$\frac{\sqrt{2}}{2} \begin{pmatrix} 1+i & 0 \\ 0 & 1-i \end{pmatrix}$
7	$R_x\left(\frac{\pi}{2}\right)$	$\frac{\sqrt{2}}{2} \begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix}$
8	$R_x\left(-\frac{\pi}{2}\right)$	$\frac{\sqrt{2}}{2} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix}$
9	$R_y\left(\frac{\pi}{2}\right)$	$\frac{\sqrt{2}}{2} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$
10	$R_y\left(-\frac{\pi}{2}\right)$	$\frac{\sqrt{2}}{2} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$
11	$R_x\left(\frac{\pi}{2}\right) \times R_y\left(\frac{\pi}{2}\right)$	$\frac{1}{2} \begin{pmatrix} 1-i & -1-i \\ 1-i & 1+i \end{pmatrix}$
12	$R_x\left(\frac{\pi}{2}\right) \times R_y\left(-\frac{\pi}{2}\right)$	$\frac{1}{2} \begin{pmatrix} 1+i & 1-i \\ -1-i & 1-i \end{pmatrix}$
13	$R_x\left(-\frac{\pi}{2}\right) \times R_y\left(\frac{\pi}{2}\right)$	$\frac{1}{2} \begin{pmatrix} 1+i & -1+i \\ 1+i & 1-i \end{pmatrix}$
14	$R_x\left(-\frac{\pi}{2}\right) \times R_y\left(-\frac{\pi}{2}\right)$	$\frac{1}{2} \begin{pmatrix} 1-i & 1+i \\ -1+i & 1+i \end{pmatrix}$
15	$R_y\left(\frac{\pi}{2}\right) \times R_x\left(\frac{\pi}{2}\right)$	$\frac{1}{2} \begin{pmatrix} 1+i & -1-i \\ 1-i & 1-i \end{pmatrix}$
16	$R_y\left(-\frac{\pi}{2}\right) \times R_x\left(\frac{\pi}{2}\right)$	$\frac{1}{2} \begin{pmatrix} 1-i & 1-i \\ -1-i & 1+i \end{pmatrix}$
17	$R_y\left(\frac{\pi}{2}\right) \times R_x\left(-\frac{\pi}{2}\right)$	$\frac{1}{2} \begin{pmatrix} 1-i & -1+i \\ 1+i & 1+i \end{pmatrix}$
18	$R_y\left(-\frac{\pi}{2}\right) \times R_x\left(-\frac{\pi}{2}\right)$	$\frac{1}{2} \begin{pmatrix} 1+i & 1+i \\ -1+i & 1-i \end{pmatrix}$
19	$XR_y\left(\frac{\pi}{2}\right)$	$\frac{\sqrt{2}}{2} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
20	$XR_y\left(-\frac{\pi}{2}\right)$	$\frac{\sqrt{2}}{2} \begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix}$
21	$YR_x\left(\frac{\pi}{2}\right)$	$\frac{\sqrt{2}}{2} \begin{pmatrix} -1 & -i \\ i & 1 \end{pmatrix}$
22	$YR_x\left(-\frac{\pi}{2}\right)$	$\frac{\sqrt{2}}{2} \begin{pmatrix} 1 & -i \\ i & -1 \end{pmatrix}$
23	$R_x\left(\frac{\pi}{2}\right) \times R_y\left(\frac{\pi}{2}\right) \times R_x\left(\frac{\pi}{2}\right)$	$\frac{\sqrt{2}}{2} \begin{pmatrix} 0 & -1-i \\ 1-i & 0 \end{pmatrix}$
24	$R_x\left(-\frac{\pi}{2}\right) \times R_y\left(\frac{\pi}{2}\right) \times R_x\left(-\frac{\pi}{2}\right)$	$\frac{\sqrt{2}}{2} \begin{pmatrix} 0 & -1+i \\ 1+i & 0 \end{pmatrix}$

**Table 3.1** All single-qubit Clifford gates that can be constructed using only  $H$  (Hadamard) and  $S$  (phase). The first column also provides a decomposition into (at most) 3 Pauli rotations and/or Pauli gates.



$$\begin{aligned}
& \mathbf{R}_x(\pi) \times \mathbf{R}_y(\pi/2) \\
&= \begin{pmatrix} \cos(\pi/2) & -i \sin(\pi/2) \\ -i \sin(-\pi/2) & \cos(\pi/2) \end{pmatrix} \times \begin{pmatrix} \cos(\pi/4) & -\sin(\pi/4) \\ \sin(\pi/4) & \cos(\pi/4) \end{pmatrix} \\
&= (-i) \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \times \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} = (-i) \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \sim \mathbf{H}.
\end{aligned}$$

The final statement of this section is an immediate consequence of Theorem 3.11 and the fact that we can use Pauli rotations to represent  $\mathbf{T} \sim \mathbf{R}_z(\pi/4)$  and  $\mathbf{H} \sim \mathbf{R}_x(\pi)\mathbf{R}_z(\pi/2)$ .

**Corollary 3.14 (Single-qubit Pauli rotations are universal).** Sequential combinations of Pauli rotations  $\mathbf{R}_x(\theta_x), \mathbf{R}_y(\theta_y), \mathbf{R}_z(\theta_z)$  with variable angles  $\theta_x, \theta_y, \theta_z \in [0, 2\pi)$  can be used to reach every single-qubit functionality.

Pauli rotations are universal

This elementary gate set is even more powerful than the ones we've seen so far because we can choose continuous angles. As a result, only very few such continuous gates suffice to exactly implement a unitary functionality  $\mathbf{U}$ .

**Exercise 3.15 (formal proofs of the matrix analysis equations).**

- 1 Verify all three matrix representations of Pauli rotations by writing down the matrix exponential and performing simplifications that are similar to the proof of Euler's equation (Exercise 3.3). **Hint:** use  $\mathbf{X}^2 = \mathbf{Y}^2 = \mathbf{Z}^2 = \mathbb{I}$  to decompose each Taylor series into an even and odd part.
- 2 Verify the rotation properties in Eq. (3.9), e.g. by inserting concrete angles and combining matrix-matrix multiplication rules with trigonometric identities.

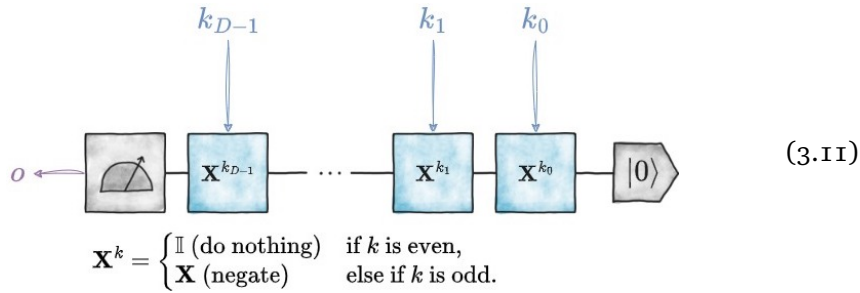
### 3.5 Application: restricted sum of parity computations

Let us now showcase how analog degrees of freedom can be exploited to solve certain problems that don't have an (obvious) digital solution. The following stylized challenge illustrates that this feature alone can already be quite empowering.

Envision a lecture hall with one instructor (me) and  $D \geq 2$  students (you). Once the game begins, the instructor hands out one integer number  $k_d \in \mathbb{Z}$  to each student. The students win the challenge if they can determine the parity (odd vs. even) of the sum of all integer  $k_{\text{tot}} = k_0 + \dots + k_{D-1}$ . But they must do so with very limited information transfer: the students can *only receive/transmit a single bit of information*. This limitation forces students to act sequentially and we can completely describe every possible strategy as a dynamic 1-bit circuit, where each student contributes one logical operation (that depends on the integer they have received).

The following classical strategy – visualized as a quantum circuit pipeline with only classical constituents – allows the students to always win this chal-

length:



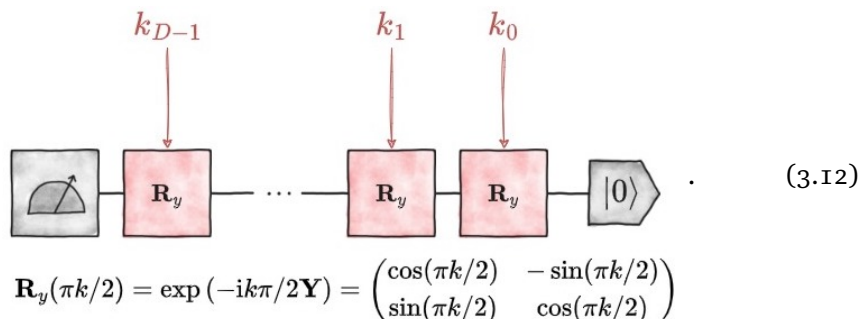
This circuit cleverly exploits a cute feature of the parity: the parity of a sum is the sum of the parities modulo 2. In this strategy, each student computes the parity of their number and, once she receives the bit, she either flips it (if their number is odd) or does nothing (if their number is even). This ensures that the final outcome bit is

$$o = \text{parity}(k_{D_1}) \oplus \cdots \oplus \text{parity}(k_0) = \text{parity}(k_0 + \cdots + k_{D-1}) = \text{parity}(k_{\text{tot}}).$$

In words: the outcome bit is 1 if and only if  $k_{\text{tot}}$  is an odd number. Otherwise, it evaluates to 0. Executing this strategy allows the students to win this restricted parity of sum computation with certainty.

So, let's increase the difficulty level. Instead of distributing integers  $k_0, \dots, k_{D-1} \in \mathbb{Z}$ , the instructor now distributes rational numbers  $k_0, \dots, k_{D-1} \in \mathbb{Q}$  (i.e. fractions) with the additional promise that the total sum  $k_{\text{tot}} = k_0 + \cdots + k_{D-1} \in \mathbb{Z}$  still adds up to an integer value. He then asks the students to come up with a strategy that again only involves a single bit of communication between them. This modification is quite nasty because the parity is not well-defined for general rational numbers. This prevents the students from re-using their winning strategy from before. In fact, we believe that it is impossible to come up with a 1-bit communication protocol that is still capable of winning this challenge with certainty.

Nonetheless, a clever group of students can overcome this sorry state of affairs by going quantum: they replace the single bit of shared (classical) information with a single qubit of shared quantum information. This generalization allows the students to implement a sequential single-qubit quantum circuit of the following form:



Instead of an individual bit, the students now pass an individual qubit between themselves. They can choose to apply a quantum gate and/or perform a readout and initialize a new qubit. A moment of thought reveals that only the very last student should perform an actual readout. And only the very first student should initialize their qubit. All students in the middle apply a quantum gate that depends on the fraction  $k_d \in \mathbb{Q}$  they received. The strategy depicted in Eq. (3.12) suggests to use a Pauli-y rotation across angle  $\theta_k = \pi k/2$ . The phase addition rule for Pauli rotations from Eq. (3.10) highlights why this strategy is a good idea:

$$\mathbf{R}_y(\pi k_{D-1}/2) \times \cdots \times \mathbf{R}_y(\pi k_0/2) = \mathbf{R}_y(\pi(k_{D-1} + \cdots + k_0)/2) = \mathbf{R}_y(\pi k_{\text{tot}}/2)$$

And, with the additional promise that  $k_{\text{tot}}$  is integer, we can conclude that the final state vector is

$$\begin{aligned} |\psi(k_{\text{tot}})\rangle &= \mathbf{R}_y(\pi k_{\text{tot}}/2)|0\rangle \\ &= \begin{pmatrix} \cos(\pi k_{\text{tot}}/2) & -\sin(\pi k_{\text{tot}}/2) \\ \sin(\pi k_{\text{tot}}/2) & \cos(\pi k_{\text{tot}}/2) \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} \cos(\pi k_{\text{tot}}/2) \\ -\sin(\pi k_{\text{tot}}/2) \end{pmatrix} \\ &= \begin{cases} \pm|0\rangle & \text{if } k_{\text{tot}} \text{ even,} \\ \pm|1\rangle & \text{if } k_{\text{tot}} \text{ odd.} \end{cases} \end{aligned}$$

The last simplification follows from the behavior of trigonometric functions: whenever  $k_{\text{tot}}$  is even,  $\sin(\pi k_{\text{tot}}/2) = 0$  and all of the information must concentrate in the ‘only 0’ branch. Conversely, if  $k_{\text{tot}}$  is odd, then  $\cos(\pi k_{\text{tot}}/2) = 0$  and all information must concentrate in the ‘only 1’ branch. There can be an additional sign factor, but this does not matter, because the readout stage is invariant under global phases.

This means that once the last person contributes their gate and starts the readout, the outcome bit  $o$  is perfectly correlated with the parity of the total sum:

- 1  $o = 0$  must happen with certainty if  $k_{\text{tot}} = k_0 + \cdots + k_{D-1}$  is an even number.
- 2  $o = 1$  must happen with certainty if  $k_{\text{tot}} = k_0 + \cdots + k_{D-1}$  is an odd number.

So, going quantum allows the students to always win a very constrained multi-player challenge for which no optimal classical strategy is known.

## Problems

**Problem 3.16 (Proof of Euler’s theorem, see Exercise 3.3).** Prove  $\exp(i\varphi) = \cos(\varphi) + i\sin(\varphi)$  by using  $i^2 = -1$  and the following three Taylor series expansions:

$$\exp(z) = \sum_{k=0}^{\infty} \frac{z^k}{k!}, \quad \cos(z) = \sum_{k=0}^{\infty} (-1)^k \frac{z^{2k}}{(2k)!}, \quad \sin(z) = \sum_{k=0}^{\infty} (-1)^k \frac{z^{2k+1}}{(2k+1)!}.$$

**Problem 3.17 (Finding all gates generated by  $H$  and  $Z$ , see Exercise 3.9).** Verify the correctness of Lemma 3.8 by writing a piece of code that generates gate combinations of certain length, computes their matrix representation via matrix-matrix multiplication and terminates once no new functionally distinct matrices can be achieved (i.e. all newly generated matrices are equivalent to existing ones up to a global phase).

**Problem 3.18 (formal proofs of matrix exponential identities, see Exercise 3.15).**

- 1 Verify all three matrix representations of Pauli rotations by writing down the matrix exponential and performing simplifications that are similar to the proof of Euler's equation (Exercise 3.3). **Hint:** use  $\mathbf{X}^2 = \mathbf{Y}^2 = \mathbf{Z}^2 = \mathbb{I}$  to decompose each Taylor series into an even and odd part.
- 2 Verify the rotation properties in Eq. (3.9), e.g. by inserting concrete angles and combining matrix-matrix multiplication rules with trigonometric identities.

## 4. Two qubit circuits

Date: 25 October 2023

Lecturers: Kristina Kirova & Jadwiga Wilkens

A 1-qubit quantum processing unit (QPU) is a great starting point for gaining a better understanding of the distinctions between classical and quantum computing by introducing the concept of superposition. However, to harness the full power of quantum phenomena in quantum computing, we require at least a 2-qubit QPU and the possible logical quantum operations.

In this lecture, we delve into the world of 2-qubit gate logic. We begin by exploring the classical bit level, first making sure that the gates are reversible, then examining concepts like the sequential and parallel application of logical gates with a truth table, which are subsequently expressed in mathematical terms. Following this, we transition to the realm of 2-qubit quantum gates, and we discuss their implications for universality in both the 2-qubit and multi-qubit scenarios making a promising point about why and how quantum computing can be more powerful than classical computing.

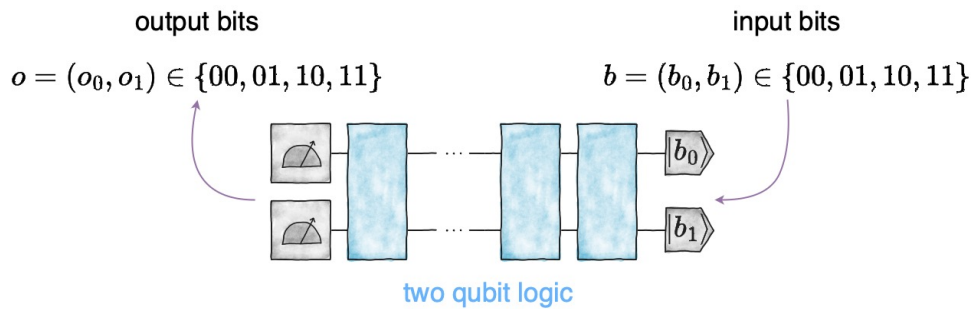
### Agenda:

- 1 motivation
- 2 classical operations and their truth tables
- 3 Kronecker product vs. matrix product
- 4 XOR and CNOT gate
- 5 universal 2-qubit gate set and its implications

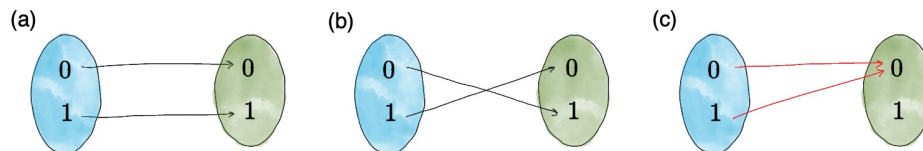
### 4.1 Classical reversible operations on 2 bits

As we saw in Lecture 1, there are two 1-bit reversible gate operations taking an input bit  $b \in \{0, 1\}$  to an output bit  $o \in \{0, 1\}$  in such a way that no information is lost. Namely the bit-flip operation  $X$  and the 'do nothing' operation  $I$ . In order to have a look at what reversibility means, Fig. 4.2 shows examples of different logical operations mapping an input bit to an output bit. Every input bit has to be mapped to *exactly one* output bit, else the mapping (/logical operation/gate) is not reversible.

Adding a second bit to this process, the general layout now looks like Fig. 4.1, input bit pairs  $b \in \{00, 01, 10, 11\}$  are manipulated and mapped onto



**Figure 4.1** Schematic illustration of a two-qubit processor: Input and output consist of 2 combined conventional bits giving 4 possible inputs and output. In between, depicted in blue, the 2-qubit logic operating at the quantum level.



**Figure 4.2** Illustration of 1-bit mappings. (a) A valid reversible mapping, the identity operation. (b) A valid reversible mapping, the bit-flip operation. (c) A not reversible mapping, for example, resetting the bit to 0.

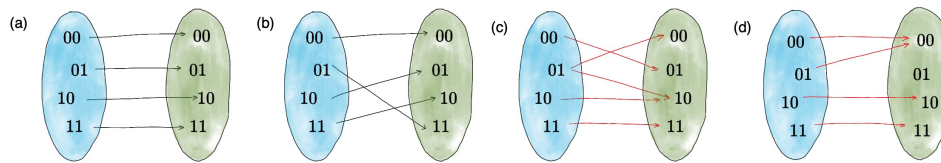
output bit pairs  $o \in \{00, 01, 10, 11\}$ . In the 1 qubit case, there are 2 distinct logical reversible operators where every bit from the input set is mapped to exactly one bit in the output set. How many reversible operations are there now for 2 bits? The same explanation that applies to 1-bit operations about what a 2-bit operation must follow to be reversible still stands. In Fig. 4.3 some illustrations are given for reversible and not reversible 2-bit gates. This argument, when followed through, boils down to a combinatorics problem; how many distinct ordering of 4 distinguishable objects (here the 4 combinations of 2 bits) are there? Or formulated differently; how many distinct permutations of these said 4 objects are there? The answer is  $4!$ , read as 4 factorial, which is  $4! = 1 \times 2 \times 3 \times 4 = 24$ .

There are 24 reversible 2 bit operations

#### 4.1.1 Combining single-bit operations in parallel

The analysis above just shows *how many* distinct reversible classical gates there are, not how their truth tables or sequence of gates look.

For that, let us look at some easy examples to get a grasp of how to build compositions of 2 (qu)bit gates. Since the bit-flip operator is already known, the easiest non-trivial example is flipping one bit while leaving one bit and doing nothing to it or vice versa. In Fig. 4.4 all four truth tables of combinations of



**Figure 4.3** Illustration of 2 bit mappings (a) - (c) reversible mappings. (d) In a non-reversible mapping, the 00 and 11 bits are both mapped to the 00 bits meaning that this mapping reverse is not clear or unambiguous.

	00	01	10	11
00	0	0	1	0
01	0	0	0	1
10	1	0	0	0
11	0	1	0	0

(a) Flipping the first bit

	00	01	10	11
00	0	0	0	1
01	0	0	1	0
10	0	1	0	0
11	1	0	0	0

(c) Flipping both bits

	00	01	10	11
00	0	1	0	0
01	1	0	0	0
10	0	0	0	1
11	0	0	1	0

(b) Flipping the second bit

	00	01	10	11
00	1	0	0	0
01	0	1	0	0
10	0	0	1	0
11	0	0	0	1

(d) Identity on both bits

**Figure 4.4** All four truth tables of combinations of bit-flip and identity operations on two bits corresponding to Fig. 4.5.

bit-flip and identity operations are depicted while in Fig. 4.5 the corresponding gate diagrams are shown.

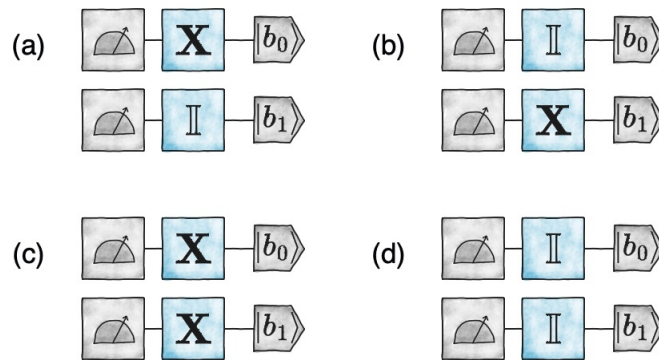
After visualizing the 2-qubit flip/identity in parallel combinations, hopefully, the following part will be clearer even though the mathematical concept of gates in parallel will be a bit tricky but crucial for the rest of the semester.

Applying two gates *in parallel* is written with a Kronecker product sign  $\otimes$  in between, (recall that applying two gates *after each other* is written with a  $\times$  in between). Putting Fig. 4.5 into the Kronecker product notation they read as

- (a)  $X \otimes I$  Flipping the first bit,
- (b)  $I \otimes X$  Flipping the second bit,
- (c)  $X \otimes X$  Flipping both bits,
- (d)  $I \otimes I$  Identity on both bits.

sequential gates  $\rightarrow$  matrix product  
parallel gates  $\rightarrow$  kronecker product

The positioning of gates or rather matrices around the Kronecker product  $\otimes$  is crucial since gates left of the  $\otimes$  are only acting on the first (qu)bit and gates written on the right are only acting on the second (qu)bit. Recall that the



**Figure 4.5** All possible combinations of identity and bit-flip single-bit gates in 2 qubit circuit. (a) flip the first bit and leave the second. (b) flip the second bit and leave the first. (c) flip both bits at the same time (parallel). (d) don't do anything to both bits.

ordering of gates when applied after each other also is important and, in general, it does not hold that the matrix product of two matrices  $\mathbf{A}$ ,  $\mathbf{B}$  is the same when changing their order:  $\mathbf{A} \times \mathbf{B} \neq \mathbf{B} \times \mathbf{A}$ . An example for the importance of ordering can be seen in Chapter 3 where the different combinations of  $\mathbf{H}$  and  $\mathbf{X}$  are shown, more precisely Equation 3.4. Looking at the matrix representation it very clearly shows that  $\mathbf{H} \times \mathbf{X} \neq \mathbf{X} \times \mathbf{H}$ .

The same holds for the Kronecker product, in general,  $\mathbf{A} \otimes \mathbf{B} \neq \mathbf{B} \otimes \mathbf{A}$ . This can already be seen in the truth table representation of bit-flip and identity combinations in Fig. 4.4:  $\mathbf{X} \otimes \mathbb{I} \neq \mathbf{X} \otimes \mathbb{I}$ .

Ordering of gates matters!

### 4.1.2 The Kronecker product

Before giving a more general definition of how to compute the Kronecker product, here is an example of how to actually calculate the Kronecker product of two matrices:

**Example 4.1** (Kronecker product of bit-flip and identity).

$$\begin{aligned} \mathbf{X} \otimes \mathbb{I} &= \begin{pmatrix} 0 & \mathbb{I} \\ \mathbb{I} & 0 \end{pmatrix} \otimes \mathbb{I} = \begin{pmatrix} 0\mathbb{I} & \mathbb{I}\mathbb{I} \\ \mathbb{I}\mathbb{I} & 0\mathbb{I} \end{pmatrix} \\ &= \begin{pmatrix} 0 & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & 0 & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}. \end{aligned}$$

The coloring in this example and in Fig. 4.4 denotes the entries of the gate acting on the first (qu)bit which are not zero. ■

**Definition 4.2** (Kronecker Product of two single-qubit gate actions). The parallel



action of two single-qubit gates, described by  $2 \times 2$  matrices

$$\mathbf{U} = \begin{pmatrix} U_{0,0} & U_{0,1} \\ U_{1,0} & U_{1,1} \end{pmatrix} \in \mathbb{C}^{2 \times 2} \text{ and } \mathbf{V} = \begin{pmatrix} V_{0,0} & V_{0,1} \\ V_{1,0} & V_{1,1} \end{pmatrix} \in \mathbb{C}^{2 \times 2},$$

is written as  $\mathbf{U} \otimes \mathbf{V}$  and captured by calculating the Kronecker product as

$$\begin{aligned} \mathbf{U} \otimes \mathbf{V} &= \begin{pmatrix} U_{0,0} \times \mathbf{V} & U_{0,1} \times \mathbf{V} \\ U_{1,0} \times \mathbf{V} & U_{1,1} \times \mathbf{V} \end{pmatrix} \\ &= \begin{pmatrix} U_{0,0}V_{0,0} & U_{0,0}V_{0,1} & U_{0,1}V_{0,0} & U_{0,1}V_{0,1} \\ U_{0,0}V_{1,0} & U_{0,0}V_{1,1} & U_{0,1}V_{1,0} & U_{0,1}V_{1,1} \\ U_{1,0}V_{0,0} & U_{1,0}V_{0,1} & U_{1,1}V_{0,0} & U_{1,1}V_{0,1} \\ U_{1,0}V_{1,0} & U_{1,0}V_{1,1} & U_{1,1}V_{1,0} & U_{1,1}V_{1,1} \end{pmatrix} \in \mathbb{C}^{4 \times 4}. \end{aligned}$$

Note that the resulting matrix has  $2 \times 2 = 4$  columns and  $2 \times 2 = 4$  rows. So, a parallel application of gates increases the size of the resulting truth table. In contrast, the matrix multiplication of gates which are applied sequentially (one after the other) does not increase the dimension of the truth table representation.

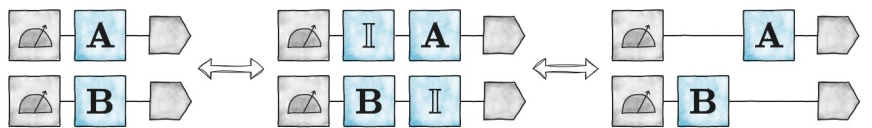
In order to get a bit more familiar with the Kronecker product, here some properties which are good to keep in mind for further calculations and a general understanding. For example taking the matrix product of parallel gates, which are denoted with capital letters  $\mathbf{A}$ ,  $\mathbf{B}$ ,  $\mathbf{C}$ , and  $\mathbf{D}$  boils down to:

$$(\mathbf{A} \otimes \mathbf{B}) \times (\mathbf{C} \otimes \mathbf{D}) = (\mathbf{A} \times \mathbf{C}) \otimes (\mathbf{B} \times \mathbf{D}). \quad (4.1)$$

This has a neat consequence to grasp what it means to apply two gates in parallel: When you use two gates at the same time, it's kind of like using one gate on the first bit and leaving the other bit alone. Then you use the second gate on the other bit, while keeping the first one unchanged with an identity gate. Putting this explanation into an equation gives

$$(\mathbf{A} \otimes \mathbb{I}) \times (\mathbb{I} \otimes \mathbf{B}) = (\mathbf{A} \times \mathbf{B}).$$

And putting this explanation into a figure gives



It is easy to count the total number of classical 2-bit truth tables (matrices) which arise from Kronecker products of reversible single-bit operations. In circuit language, this corresponds to the parallel application of single-bit gates and there are exactly  $2 \times 2 = 4$  of them (either  $\mathbf{X}$  or  $\mathbb{I}$  on each bit). This number (4) is much smaller than the total number of reversible two-bit circuits (24).

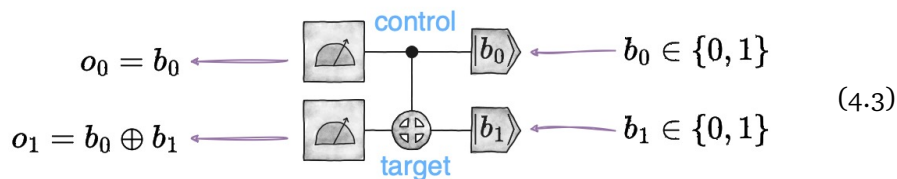
So, there must be circuits whose truth table cannot be expressed as a single Kronecker product.

One such gate is a reversible implementation of the XOR operation, called *controlled not* or *CNOT*.

It acts as follows: the first bit is not manipulated but the second bit is the direct sum  $\oplus$  of the first and the second bit, which is defined as only being 1 if exactly one of the two bits is 1, else 0,

	00	01	10	11	
00	1	0	0	0	(CNOT <sub>0→1</sub> 'truth table'). (4.2)
01	0	1	0	0	
10	0	0	0	1	
11	0	0	1	0	

This operation always requires 2 bits: a control bit and a target bit. Here the first bit denoted as 0 is the control bit and the second bit denoted as 1 is the target bit, written as  $0 \rightarrow 1$  in the caption of the CNOT. There is also the CNOT<sub>1→0</sub> which will be discussed in a later subsection.



Since information is flowing from the control bit to the target bit, this gate cannot be viewed as acting separately on the first and separately on the second bit. It always acts simultaneously on both bits and cannot be written in any other way if there are only two bits available.

## 4.2 Quantum operations on 2 qubits

Equipped with a new tool to express parallel operations, let us have a look at quantum gates and quantum states.

### 4.2.1 Quantum gates on 2 qubits

The generalization of the Kronecker product from classical gates to quantum gates is straightforward: As long as a gate that now acts in parallel with other gates has a matrix representation one can always calculate the Kronecker product of this composition. Since in the former lectures quantum gates like the Hadamard  $H$  or sign-flip  $Z$  were already introduced, letting them act on a qubit from a 2-qubit ensemble is straightforward.

**Example 4.3** (Kronecker product of  $H$  and  $Z$ ).

$$H \otimes Z = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & -1 & 0 & -1 \\ 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix} \quad (4.4)$$

■

### 4.2.2 Quantum states on 2 qubit

In order to understand how a 2-qubit quantum operation manipulates states, only a formal definition of how a 2-qubit quantum state is expressed in the state vector formalism is missing:

2-qubit state vectors have 4 entries

**Definition 4.4 (2-qubit quantum state).** The state of a 2-qubit system is characterized by a 4-dimensional vector

$$|\psi\rangle = \boldsymbol{\psi} = \begin{pmatrix} \psi_{00} \\ \psi_{01} \\ \psi_{10} \\ \psi_{11} \end{pmatrix} \in \mathbb{C}^4. \quad (4.5)$$

The individual coefficients, as in the single-bit state vector, can be complex-valued numbers and it must hold that

$$\| |\psi\rangle \|^2 = |\psi_{00}|^2 + |\psi_{01}|^2 + |\psi_{10}|^2 + |\psi_{11}|^2 = 1 \quad (\text{state normalization}). \quad (4.6)$$

Analogous to the introduction of a 1-qubit quantum state, the classical bit representation is standing out in terms of usage for extracting the probabilities of a quantum state. How to imprint two classical bits  $b \in \{00, 01, 10, 11\}$  into the initial state of two qubits is given in the following example:

**Example 4.5 (2-qubit initialization).**

$$|0\rangle \otimes |0\rangle = \mathbf{e}_{00} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad |0\rangle \otimes |1\rangle = \mathbf{e}_{01} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad (4.7)$$

$$|1\rangle \otimes |0\rangle = \mathbf{e}_{10} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad |1\rangle \otimes |1\rangle = \mathbf{e}_{11} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \quad (4.8)$$

■

To abbreviate writing the classical imprinting, it is very common in the literature to write  $|0\rangle \otimes |0\rangle$  as  $|00\rangle$  or  $|0\rangle \otimes |1\rangle$  as  $|01\rangle$  where the Kronecker product is omitted and the bits shifted in one ket.

Analogous to the already discussed gates, two single-qubit states can be paired together via the Kronecker product.

**Example 4.6 (2-qubit quantum state vector from two single-qubits).** Given two single-qubit quantum states  $|\psi_0\rangle = \begin{pmatrix} \psi_{0,0} \\ \psi_{0,1} \end{pmatrix} \in \mathbb{C}^{2 \times 2}$  and  $|\psi_1\rangle = \begin{pmatrix} \psi_{1,0} \\ \psi_{1,1} \end{pmatrix} \in \mathbb{C}^{2 \times 2}$  the resulting 2-qubit state can be expressed as

$$|\psi\rangle = |\psi_0\rangle \otimes |\psi_1\rangle \quad (4.9)$$

$$= \begin{pmatrix} \psi_{0,0} \\ \psi_{0,1} \end{pmatrix} \otimes \begin{pmatrix} \psi_{1,0} \\ \psi_{1,1} \end{pmatrix} = \begin{pmatrix} \psi_{0,0}\psi_{1,0} \\ \psi_{0,0}\psi_{1,1} \\ \psi_{0,1}\psi_{1,0} \\ \psi_{0,1}\psi_{1,1} \end{pmatrix} = \begin{pmatrix} \psi_{00} \\ \psi_{01} \\ \psi_{10} \\ \psi_{11} \end{pmatrix} \in \mathbb{C}^{4 \times 4} \quad (4.10)$$

■

Note that this is not a general statement in the sense that not every quantum state  $|\psi\rangle$  can be written in terms of two single-qubit states but every system composed of two single-qubit states can be written as above. If, for example, the circuit starts in a classical bit imprinted state and the gates applied do not transfer any information from one qubit to the other, the action of the gate ( $\mathbf{U} \otimes \mathbf{V}$ ) is fully captured by letting  $\mathbf{U}$  only act on the first qubit and  $\mathbf{V}$  only act on the second qubit,

$$(\mathbf{U} \otimes \mathbf{V})(\boldsymbol{\psi}_0 \otimes \boldsymbol{\psi}_1) = \mathbf{U}\boldsymbol{\psi}_0 \otimes \mathbf{V}\boldsymbol{\psi}_1. \quad (4.11)$$

But as discussed in the classical 2-bit gate section, there are gates whose action cannot be described as a single gate acting on the first qubit and another single gate acting on the second qubit. As in the classical case the most prominent candidate is the quantum CNOT gate whose matrix representation is indeed the same, see the truth table (4.2) and the diagram as written in Fig. 4.3. In quantum computing, this gate and its properties have huge implications in terms of computing strength in the quantum realm which will be discussed in more detail in the next lecture. Much like in classical circuit logic, these operations allow to *conditionally* modify qubits. This is essential to build more complicated functionalities out of simple elementary building blocks.

### 4.2.3 The CNOT gate

One of the most important features of the CNOT gate is that, paired with a universal 1-qubit gate set, it forms a *universal 2-qubit gate set*. Meaning that every 2-qubit unitary  $\mathbf{U} \in \mathbb{C}^{4 \times 4}$  with 16 entries can be, to arbitrary precision, approximated by only 3 gates: the Hadamard 1-qubit gate  $\mathbf{H}$ , the 1 qubit T-gate  $\mathbf{T}$ , and the CNOT gate  $\mathbf{CNOT}$  acting on two qubits.

universal 2-qubit gate set

**Theorem 4.7 (efficient 2-qubit synthesis).** Let  $\mathcal{G}$  be a universal 2-qubit gate set<sup>a</sup>, e.g.  $\mathcal{G} = \{H, T, CNOT\}$  and let  $\epsilon'$  be a desired approximation accuracy. Then, for every unitary  $4 \times 4$  matrix  $U$  containing  $m$  *CNOT* gates and arbitrarily many single-qubit gates, there exists a sequence of (at most)

$$D = \mathcal{O}(m \log^c(m/\epsilon')) \quad (4.12)$$

gates that approximates the action of  $U$  up to accuracy  $\epsilon'$ . Here,  $c \in [1, 3 + o(1)]$  is a constant that depends on the universal gate set in question.

<sup>a</sup>The original statement also requires that this gate set either contains inverses or can generate them in a constant number of steps.

efficient 2-qubit circuit synthesis

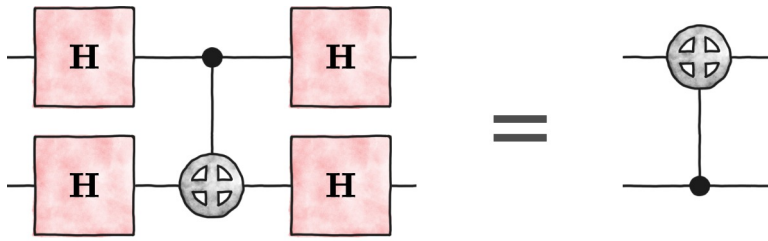
This theorem resembles the Theorem 3.12 of the universal 1-qubit gate set with the big difference that the number of one standing-out gate, the CNOT gate, plays a role in the depth of the approximating circuit. This should at first be surprising since this kind of special treatment of one gate did not happen in the 1-qubit case. To grasp a better understanding of this above stated theorem, we will look at the case where there is an arbitrary circuit with single-qubit unitaries and  $m$  CNOT gates, mixed randomly together.

Taking a closer look at the role of the CNOT gate, it becomes clear that any application of it is a somehow disruptive operation conditionally modifying one qubit state. If we would zoom into one wire, let's say the wire of the second qubit, all the single-qubit gate operations between two CNOT gates can be summarized to one unitary (by simply calculating the matrix product of them). This arbitrary unitary can now be approximated with  $\mathcal{O}(\log^c(1/\epsilon))$  many  $H$  and  $T$  gates to a precision of  $\epsilon$ . But a CNOT cannot be written as a unitary acting solely on the second qubit, meaning it can not just be pushed into a single-qubit unitary which is then approximated. The CNOT changes the qubit state conditionally to the first one, so after the CNOT the next single-qubit gates have to be approximated *again*, with an order of depth of  $\mathcal{O}(\log^c(1/\epsilon))$ . This little procedure continues until all single-qubit gates between the  $m$  CNOT gates are approximated to an  $\epsilon' = \epsilon m$  precision in total. Note that here we use two different precision variables, namely  $\epsilon$  for the single-qubit unitary precision and  $\epsilon'$  for the whole 2-qubit unitary precision.

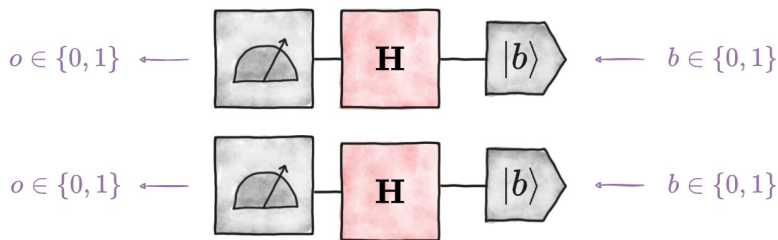
#### 4.2.4 Examples: CNOT<sub>1→0</sub> and a random number generator

We will now introduce a very useful (and perhaps somewhat surprising) circuit identity: flipping the CNOT gate with 4 Hadamard gates, fig.4.6. By "flipping" we mean that the control qubit becomes the target one.

**Example 4.8 (flipping the CNOT).** Let's show this by writing the two matrices explicitly, using the rules for concatenating quantum operations.



**Figure 4.6** The CNOT gate can be reversed with 4 Hadamard gates. This circuit identity shows that the target and control qubits can be interchanged by applying a Hadamard gate on every qubit before and after the CNOT gate.



**Figure 4.7** Random number generator with 2 qubits. This quantum circuit accepts two classical bits,  $b \in \{0, 1\}$  which initialize the two quantum states in either  $|0\rangle$  or  $|1\rangle$ . Applying a Hadamard to both qubits and then measuring results in two independent random bits.

$$(H \otimes H) CNOT (H \otimes H) \quad (4.13)$$

$$= \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \quad (4.14)$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad (4.15)$$

which is the truth table of an operation which flips the first bit conditioned on the second one.

This result implies that by acting locally on the two qubits we can change the flow of information. ■

**Example 4.9 (2-qubit quantum random number generator).** Let us now extend our example for a quantum number generator from the second lecture, Example 2.8, to 2 qubits. Consider the simple circuit, Fig. 4.7, where we initialize both qubits in the  $|1\rangle$  state (for  $b = 1$ ), apply a Hadamard to each, and measure.

The state just before the measurement is:

$$|\psi\rangle = \begin{pmatrix} \psi_{00} \\ \psi_{01} \\ \psi_{10} \\ \psi_{11} \end{pmatrix} = (\mathbf{H} \otimes \mathbf{H}) (|1\rangle \otimes |1\rangle) \quad (4.16)$$

$$= \left( \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix} \otimes \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix} \right) \left( \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right) \quad (4.17)$$

$$= \begin{pmatrix} 1/2 & 1/2 & 1/2 & 1/2 \\ 1/2 & -1/2 & 1/2 & -1/2 \\ 1/2 & 1/2 & -1/2 & -1/2 \\ 1/2 & -1/2 & -1/2 & 1/2 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1/2 \\ -1/2 \\ -1/2 \\ 1/2 \end{pmatrix}. \quad (4.18)$$

Now recall that the outcome probabilities are the squared magnitudes of the state vector entries, namely  $|\psi_0\rangle, |\psi_1\rangle, |\psi_2\rangle, |\psi_3\rangle$ . Analogous to the rule for single-qubit readout (Definition 2.7), we now compute:

every outcome is equally likely

$$\begin{aligned} \Pr_{|\psi\rangle} [o = 00] &= |\langle 00|\psi\rangle|^2 = |\psi_0|^2 = |1/2|^2 = 1/4, \\ \Pr_{|\psi\rangle} [o = 01] &= |\langle 01|\psi\rangle|^2 = |\psi_1|^2 = |-1/2|^2 = 1/4. \\ \Pr_{|\psi\rangle} [o = 10] &= |\langle 10|\psi\rangle|^2 = |\psi_2|^2 = |-1/2|^2 = 1/4. \\ \Pr_{|\psi\rangle} [o = 11] &= |\langle 11|\psi\rangle|^2 = |\psi_3|^2 = |1/2|^2 = 1/4. \end{aligned}$$

We see that all the possible 4 outcomes are equally likely. Let us now focus only on the first bit, without loss of generality. The probability of measuring an outcome 0 on it is given by

$$\Pr [o_1 = 0] = \sum_{i=0}^1 \Pr [o_1 o_2 = 0i] = 1/4 + 1/4 = 1/2.$$

This is just the definition of a perfect random bit  $o \stackrel{\text{unif}}{\sim} \{0, 1\}$  which we saw in Ex. 2.8. We now have a device which can generate 2 random bits simultaneously. It is important to note that these 2 bits are *independent* of each other. This is not always the case and in the next two lectures, we will see examples of creating correlations with quantum circuits which cannot be done classically. ■

## 5. Bell states & Superdense Coding

Date: 08 November 2023

Lecturers: Kristina Kirova & Jadwiga Wilkens

Today, we will start analyzing and exploiting the extraordinary correlations that become possible when working with quantum circuits. As we shall see, the (joint) quantum state of two qubits ( $n = 2$ ) can be correlated in ways that are inconceivable from a classical perspective. This has both important conceptual implications, as well as practical ones.

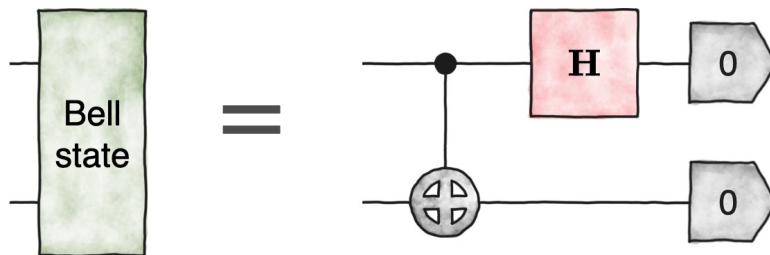
### Agenda:

- 1 Bell States
- 2 Equivalence Theorem
- 3 Superdense Coding
- 4 Quantum Games

### 5.1 Motivation: The Bell state

Our starting point is the output of the following, seemingly innocuous, quantum circuit, Fig. 5.1. It contains two qubits, one Hadamard (superposition) gate and a CNOT (classical gate).

We can use the unitary matrix framework to compute all amplitudes of the resulting quantum state:



**Figure 5.1** Circuit generating a Bell state. This quantum circuit initializes both qubits in 0, then applies Hadamard to the first qubit, followed by a CNOT.



$$\begin{aligned}
|\psi_{\text{Bell}}\rangle &= \mathbf{CNOT} (\mathbf{H} \otimes \mathbb{I})|00\rangle \\
&= \mathbf{CNOT} (\mathbf{H}|0\rangle \otimes |0\rangle) \\
&= \mathbf{CNOT} \left( \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \otimes |0\rangle \right) \\
&= \frac{1}{\sqrt{2}} \mathbf{CNOT}|00\rangle + \frac{1}{\sqrt{2}} \mathbf{CNOT}|10\rangle \\
&= \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle). \tag{5.1}
\end{aligned}$$

Bell state:  $\frac{1}{\sqrt{2}} (|00\rangle + |11\rangle)$

Let us now compute the probability of each readout outcome in the same manner as in ex.4.9:

$$\begin{aligned}
\Pr_{|\psi_{\text{Bell}}\rangle} [o = 00] &= |\langle 00|\psi\rangle|^2 = |\psi_0|^2 = \left|1/\sqrt{2}\right|^2 = 1/2, \\
\Pr_{|\psi_{\text{Bell}}\rangle} [o = 01] &= |\langle 01|\psi\rangle|^2 = |\psi_1|^2 = |0|^2 = 0. \\
\Pr_{|\psi_{\text{Bell}}\rangle} [o = 10] &= |\langle 10|\psi\rangle|^2 = |\psi_2|^2 = |0|^2 = 0. \\
\Pr_{|\psi_{\text{Bell}}\rangle} [o = 11] &= |\langle 11|\psi\rangle|^2 = |\psi_3|^2 = \left|1/\sqrt{2}\right|^2 = 1/2.
\end{aligned}$$

Out of the four possible amplitudes, only two are nonzero and equal to  $1/2$ . This looks a lot like the value of two random coins that are *perfectly correlated* with each other. Both coins always show the same value (0 or 1) with a probability of  $1/2$  each.

perfectly correlated outcomes

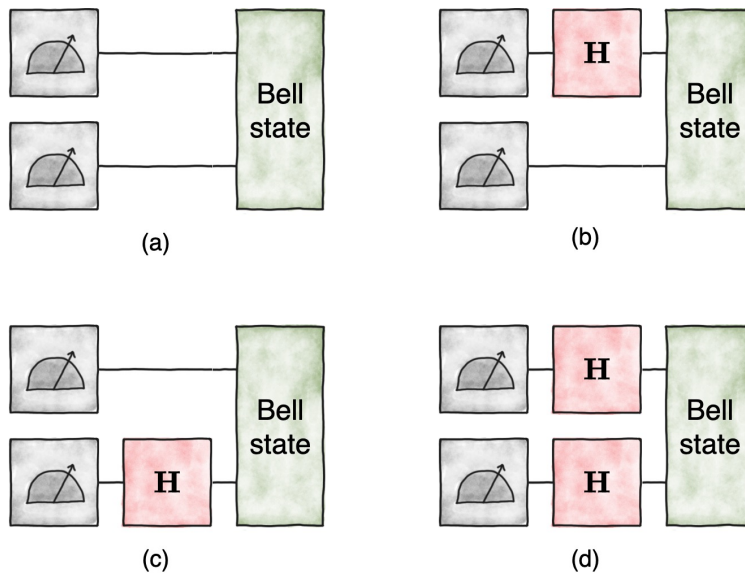
### 5.1.1 Stronger than classical correlations

The above result is nice, but does not look very revolutionary yet as one can also imitate such correlated randomness by first tossing a fair coin and then communicating the value to both readout locations. What is so special about this Bell state then? Let us now apply the Hadamard gate to one or both qubits, Fig. 5.2.

In the trivial case in which a Bell state is prepared and measured, Fig. 5.2(a), both outcomes are perfectly correlated, as we have been discussing so far.

For the quantum circuits shown in Fig. 5.2(b), where a Hadamard is applied to the first qubit, the final state is

$$\begin{aligned}
|\psi_{\text{final(b)}}\rangle &= (\mathbf{H} \otimes \mathbb{I})|\psi_{\text{Bell}}\rangle \\
&= (\mathbf{H} \otimes \mathbb{I}) \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle) \\
&= \frac{1}{\sqrt{2}} \left( \left( \frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) |0\rangle + \left( \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) |1\rangle \right) \\
&= \frac{1}{2} (|00\rangle + |10\rangle + |01\rangle - |11\rangle)
\end{aligned}$$



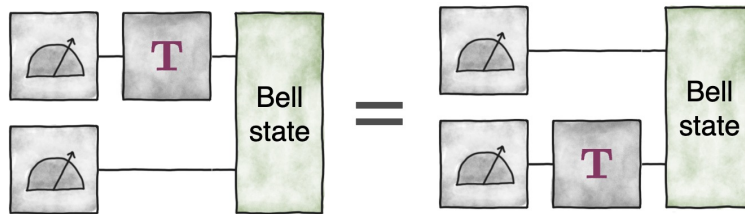
**Figure 5.2** Four simple circuits including the Bell state. (a) Bell state followed by a measurement. Outcomes are perfectly correlated. (b) A Hadamard gate applied to the first qubit destroys the correlation. (c) A Hadamard gate applied to the second qubit also destroys the correlation. (d) A Hadamard gate applied to both qubits preserves the correlation.

and for the circuit shown in Fig. 5.2(c) it is

$$\begin{aligned}
 |\psi_{\text{final}(c)}\rangle &= (\mathbb{I} \otimes \mathbf{H}) |\psi_{\text{Bell}}\rangle \\
 &= (\mathbb{I} \otimes \mathbf{H}) \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle) \\
 &= \frac{1}{\sqrt{2}} \left( |0\rangle \left( \frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) + |1\rangle \left( \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) \right) \\
 &= \frac{1}{2} (|00\rangle + |10\rangle + |01\rangle - |11\rangle).
 \end{aligned}$$

Firstly, the quantum states which both circuits produce are the same, which is somewhat surprising and we will discuss it in more detail below. Another thing to note is that the probability of measuring each of the 4 possible outcomes is the same, namely  $|1/2|^2 = 1/4$ . Recall that we obtained the same probability distribution in Ex. 4.9, the circuit generating random numbers with 2 qubits. It seems that by applying a single Hadamard we have destroyed the perfect correlation and have generated random bits instead.

In the case where a Hadamard gate is applied to both qubits, Fig. 5.2(d), is surprising and *without a classical analogue* as it preserves the perfect correlation between the measurement outcomes on both qubits. Let's compute the final state explicitly:



**Figure 5.3** Local equivalence of Bell state. Applying a T gate on either of the two qubits in a Bell state leads to the same final quantum state.

$$\begin{aligned}
 |\psi_{\text{final(d)}}\rangle &= (\mathbf{H} \otimes \mathbf{H}) |\psi_{\text{Bell}}\rangle \\
 &= (\mathbf{H} \otimes \mathbf{H}) \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle) \\
 &= \frac{1}{\sqrt{2}} \left( \left( \frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) \left( \frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) + \left( \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) \left( \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) \right) \\
 &= \frac{1}{2\sqrt{2}} (|00\rangle + |01\rangle + |10\rangle + |11\rangle + |00\rangle - |01\rangle - |10\rangle + |11\rangle) \\
 &= \frac{1}{2\sqrt{2}} (2|00\rangle + 2|11\rangle) \\
 &= \frac{|00\rangle + |11\rangle}{\sqrt{2}} \\
 &= |\psi_{\text{Bell}}\rangle
 \end{aligned}$$

We recover the original Bell state as if no Hadamard gates were ever applied. We have already seen that the Hadamard gate is its own inverse,  $\mathbf{H}\mathbf{H} = \mathbb{I}$ , but when applied consecutively to the *same* qubit. What is happening here? Is this because of the Hadamard gate or the specific properties of the Bell state?

What is more, there does not exist a classical device which behaves in the same way under such conditions. Namely, producing perfectly correlated bits, then producing random bits if a Hadamard gate is applied to either register and yet correlated ones if the Hadamard gate is applied to both. This is what we call *stronger than classical correlations* and we will see more examples of this in the next lecture.

No classical equivalence

**Example 5.1 (T gate on a Bell state).** Let us now pick another gate, say the T gate, and check if similar properties hold. Explicit computation of the final states produced by applying the  $\mathbf{T}$  gate on the first or on the second qubit of a Bell state, Fig. 5.3,

$$\begin{aligned}
|\psi_{\text{final}(1)}\rangle &= (\mathbf{T} \otimes \mathbb{I}) |\psi_{\text{Bell}}\rangle \\
&= (\mathbf{T} \otimes \mathbb{I}) \frac{1}{\sqrt{2}} (|00\rangle |11\rangle) \\
&= \frac{1}{\sqrt{2}} (|00\rangle + e^{i\pi/4} |11\rangle) \\
&= (\mathbb{I} \otimes \mathbf{T}) \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle) \\
&= (\mathbb{I} \otimes \mathbf{T}) |\psi_{\text{Bell}}\rangle
\end{aligned}$$

reveals that given a Bell state, regardless of which qubit we apply the gate to, we produce the same state. ■

To summarize, we just demonstrated that

$$(\mathbf{H} \otimes \mathbb{I}) |\psi_{\text{Bell}}\rangle = (\mathbb{I} \otimes \mathbf{H}) |\psi_{\text{Bell}}\rangle \quad (5.2)$$

$$(\mathbf{T} \otimes \mathbb{I}) |\psi_{\text{Bell}}\rangle = (\mathbb{I} \otimes \mathbf{T}) |\psi_{\text{Bell}}\rangle. \quad (5.3)$$

These are some of the properties that make the Bell State so special and worth dedicating a whole lecture to. The above result is a particular case of the following:

**Theorem 5.2** Let  $U$  be a single-qubit unitary matrix. Then,

$$(U \otimes \mathbb{I}) |\psi_{\text{Bell}}\rangle = (\mathbb{I} \otimes U^T) |\psi_{\text{Bell}}\rangle$$

where  $T$  denotes matrix transposition.

Gate Equivalence

$$(U \otimes \mathbb{I}) |\psi_{\text{Bell}}\rangle = (\mathbb{I} \otimes U^T) |\psi_{\text{Bell}}\rangle$$

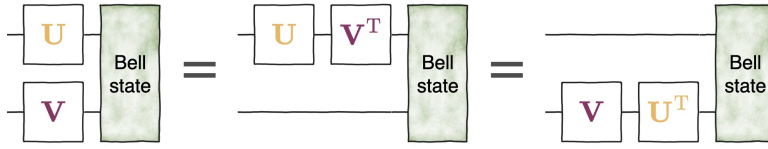
*Proof.* Recall Theorem 3.11, which states that  $\{\mathbf{H}, \mathbf{T}\}$  is a universal single qubit gate set. Hence, any arbitrary unitary gate  $U$  can be written as  $U = V_N \cdots V_1 V_0$  with  $V_k \in \{\mathbf{H}, \mathbf{T}\}$  for some  $N \geq 1$ . Let  $U$  act on the first qubit of the Bell state:

$$\begin{aligned}
U \otimes \mathbb{I} |\psi_{\text{Bell}}\rangle &= (V_N \cdots V_1 V_0) \otimes \mathbb{I} |\psi_{\text{Bell}}\rangle \\
&= (V_N \otimes \mathbb{I}) \cdots (V_1 \otimes \mathbb{I}) (V_0 \otimes \mathbb{I}) |\psi_{\text{Bell}}\rangle \\
&= (V_N \otimes \mathbb{I}) \cdots (V_1 \otimes \mathbb{I}) (\mathbb{I} \otimes V_0) |\psi_{\text{Bell}}\rangle \\
&= (\mathbb{I} \otimes V_0) (V_N \otimes \mathbb{I}) \cdots (V_1 \otimes \mathbb{I}) |\psi_{\text{Bell}}\rangle,
\end{aligned}$$

where we have used Eq. (5.2) or Eq. (5.3) to move the first gate  $V_0 \in \{\mathbf{H}, \mathbf{T}\}$  from the top to the bottom. The last line is a simple commutation operation.

Repeat this  $N - 1$  times to obtain

$$\begin{aligned}
U \otimes \mathbb{I} |\psi_{\text{Bell}}\rangle &= (\mathbb{I} \otimes V_0) (\mathbb{I} \otimes V_1) \cdots (\mathbb{I} \otimes V_N) |\psi_{\text{Bell}}\rangle \\
&= \mathbb{I} \otimes (V_0 V_1 \cdots V_N) |\psi_{\text{Bell}}\rangle.
\end{aligned} \quad (5.4)$$



**Figure 5.4** Visualization of Corollary 5.3: The following three circuits are all equivalent in the sense that they produce the same final quantum state. There is no way of distinguishing these circuits.

The sequence of gates  $(V_0 V_1 \cdots V_N)$  looks like the gate decomposition of  $\mathbf{U}$ , but in reverse order. This reverse ordering can be expressed via the *transpose* operation. Indeed,  $V_k^T = V_k$ , because  $H^T = H$  and  $T^T = T$ . In turn,

$$\begin{aligned} \mathbf{U}^T &= (V_N \cdots V_1 V_0)^T \\ &= V_0^T V_1^T \cdots V_N^T \\ &= V_0 V_1 \cdots V_N \end{aligned}$$

Inserting this back into Eq. (5.4) we conclude the proof:

$$\mathbf{U} \otimes \mathbb{I} |\psi_{\text{Bell}}\rangle = \mathbb{I} \otimes \mathbf{U}^T |\psi_{\text{Bell}}\rangle.$$

■

Here is an immediate consequence of Theorem 7.1 that justifies the particular result of circuit Fig. 5.2 (d) where applying Hadamard gates to both qubits did not change the final state.

**Corollary 5.3** Let  $\mathbf{U}, \mathbf{V}$  be arbitrary single-qubit gates. Then,

$$\mathbf{U} \otimes \mathbf{V} |\psi_{\text{Bell}}\rangle = (\mathbf{U} \times \mathbf{V}^T) \otimes \mathbb{I} |\psi_{\text{Bell}}\rangle = \mathbb{I} \otimes (\mathbf{V} \times \mathbf{U}^T) |\psi_{\text{Bell}}\rangle.$$

Ultimately, the three circuits shown in Fig. 5.4 all produce the same quantum state. This also explains why applying Hadamard gates to both qubits of the Bell state did nothing to the state, Fig. 5.2(d). It also looks like we have found a way to “teleport” gates from one of the qubits in the Bell state to the other. This is a very powerful property which we will use intensively in the next two lectures.

*Proof.* Both claims follow from decomposing  $\mathbf{U} \otimes \mathbf{V}$  either as  $\mathbf{U} \otimes \mathbf{V} = (\mathbf{U} \otimes \mathbb{I}) \times (\mathbb{I} \otimes \mathbf{V})$  or  $(\mathbb{I} \otimes \mathbf{V}) \times (\mathbf{U} \otimes \mathbb{I})$  and subsequently applying Theorem 7.1. For instance,

$$\begin{aligned} \mathbf{U} \otimes \mathbf{V} |\psi_{\text{Bell}}\rangle &= (\mathbb{I} \otimes \mathbf{V}) \times (\mathbf{U} \otimes \mathbb{I}) |\psi_{\text{Bell}}\rangle \\ &= (\mathbb{I} \otimes \mathbf{V}) \times (\mathbb{I} \otimes \mathbf{U}^T) |\psi_{\text{Bell}}\rangle \\ &= \mathbb{I} \otimes (\mathbf{V} \times \mathbf{U}^T) |\psi_{\text{Bell}}\rangle, \end{aligned}$$

and the other reformulation can be deduced in a similar fashion. ■

## 5.2 More Bell states

So far we have been analysing the simple circuit which produces the Bell state starting with both qubits initialized in the 0 state. What happens if we were to choose different inputs? There are  $2^2 = 4$  different choices, one for each input string of size  $n = 2$ , as depicted in Fig. 5.5.

The 4 Bell states are then:

$$|\psi_{\text{Bell}}(0, 0)\rangle = \mathbf{CNOT} (\mathbf{H} \otimes \mathbb{I})|0, 0\rangle = \frac{1}{\sqrt{2}} (|0, 0\rangle + |1, 1\rangle), \quad (5.5)$$

$$|\psi_{\text{Bell}}(0, 1)\rangle = \mathbf{CNOT} (\mathbf{H} \otimes \mathbb{I})|0, 1\rangle = \frac{1}{\sqrt{2}} (|0, 1\rangle + |1, 0\rangle), \quad (5.6)$$

$$|\psi_{\text{Bell}}(1, 0)\rangle = \mathbf{CNOT} (\mathbf{H} \otimes \mathbb{I})|1, 0\rangle = \frac{1}{\sqrt{2}} (|0, 0\rangle - |1, 1\rangle), \quad (5.7)$$

$$|\psi_{\text{Bell}}(1, 1)\rangle = \mathbf{CNOT} (\mathbf{H} \otimes \mathbb{I})|1, 1\rangle = \frac{1}{\sqrt{2}} (|0, 1\rangle - |1, 0\rangle). \quad (5.8)$$

These quantum states do look different but have similar features overall. They all describe perfect correlation or anti-correlation between the two qubits involved. In fact, we can generate all of them from the original Bell state. To this end, we recall the following elementary single-qubit operations:

All 4 Bell states are locally convertible

$$\mathbf{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \text{ i.e. } \begin{cases} \mathbf{X}|0\rangle = |1\rangle, \\ \mathbf{X}|1\rangle = |0\rangle, \end{cases} \quad (\text{bit flip}),$$

$$\mathbf{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \text{ i.e. } \begin{cases} \mathbf{Z}|0\rangle = |0\rangle, \\ \mathbf{Z}|1\rangle = -|1\rangle, \end{cases} \quad (\text{sign flip}).$$

We can define a conditional application of these single-qubit gates that depends on an external input bit  $b \in \{0, 1\}$ :

$$\mathbf{X}^0 = \mathbb{I}, \mathbf{X}^1 = \mathbf{X} \quad \text{and} \quad \mathbf{Z}^0 = \mathbb{I}, \mathbf{Z}^1 = \mathbf{Z}.$$

In words: if  $b = 0$ , we do nothing (identity operation). Else if  $b = 1$ , we apply the bit (sign) flip gate.

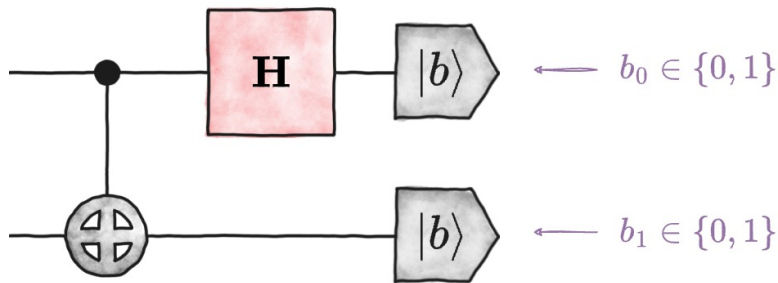
**Proposition 5.4** Let  $b_0, b_1 \in \{0, 1\}$  be two possible input bits. Then,

$$|\psi_{\text{Bell}}(b_0, b_1)\rangle = (\mathbf{Z}^{b_0} \mathbf{X}^{b_1} \otimes \mathbb{I}) |\psi_{\text{Bell}}(0, 0)\rangle.$$

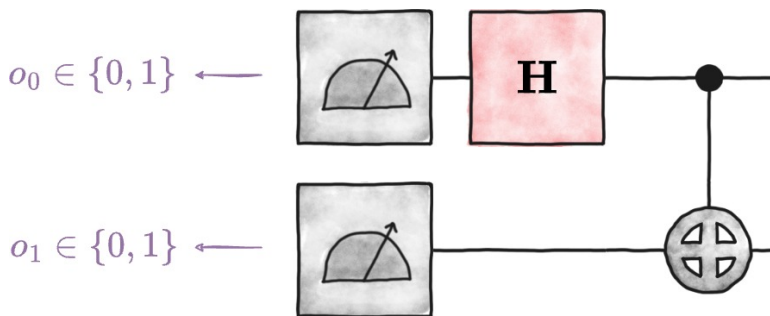
**Exercise 5.5** Prove Proposition 5.4, e.g. by directly computing all advertised 2-qubit states and verifying equality.

## 5.3 Bell measurement

A Bell measurement is a joint measurement performed on two qubits, which determines which of the four Bell states the two qubits are in. It consists of the



**Figure 5.5** Circuit generating all the four Bell states. Depending on the input bits,  $b$ , a different state is generated.



**Figure 5.6** Bell measurement. Applying a *CNOT* and a *H* just before the measurement is equivalent to measuring in the Bell basis.

same operations preparing the Bell state but in reverse order: first applying a CNOT gate to both qubits and then a Hadamard gate on the first qubit, Fig. 5.6. We can think of the CNOT gate un-entangling the two previously entangled qubits. This allows the information to be converted from quantum information to a measurement of classical information.

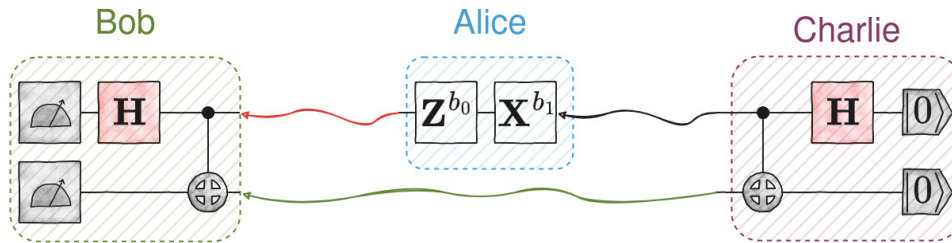
Each of the 4 possible outcomes denotes one of the Bell states, as defined in Eqs. 5.8.

Bell measurement is the reverse of Bell state preparation

## 5.4 Superdense Coding

This protocol called *superdense coding* is often used in literature to showcase how the Bell State could be used in a real-world application, keeping in mind that for simplicity there are only two qubits involved to make calculations understandable. The naming will become clearer after stating the protocol.

The setting is depicted in Fig. 5.7 and is the following: Alice wants to send Bob a message which is encoded in *two* classical bits  $b \in (00, 01, 10, 11)$ , with her only sending *one* qubit to Bob. For that to work, they have to share a strongly correlated qubit pair which is prepared and sent by a neutral third party, here called Charlie. As one can see in the diagram in Fig. 5.7 the prepared



**Figure 5.7** Scheme of the superdense coding protocol. Charlie prepares Bell State and sends one part to Alice and the other part to Bob. Alice performs a quantum operation on her qubit depending on which 2 bits  $(b_0 b_1) \in (00, 01, 10, 11)$  she wants to send to Bob. Then she sends her qubit to Bob which applies a  $\text{CNOT}_{0 \rightarrow 1}$  gate and a Hadamard gate on the first qubit. A measurement performed by Bob in the computational basis will reveal which 2 bits Alice sent him.

qubit pair is the first bell basis state as stated in Eq. (5.1). Alice then applies quantum operations to her qubit to encode the two classical bits she wants to send to Bob. She has four different options to encode her message and for Bob to decode the send qubit correctly:

resulting state	applied gates	message
$\frac{1}{\sqrt{2}} ( 00\rangle +  11\rangle)$		00
$\frac{1}{\sqrt{2}} ( 10\rangle +  01\rangle)$		01
$\frac{1}{\sqrt{2}} ( 00\rangle -  11\rangle)$		10
$\frac{1}{\sqrt{2}} ( 01\rangle -  10\rangle)$		11

(5.9)

After Alice applies one of the four operations, she sends her qubit to Bob. Bob, already holding the other part of the Bell state qubit pair or receiving it at the same time performs a measurement in the first bell basis by applying a  $\text{CNOT}_{0 \rightarrow 1}$  gate where Alice's qubit is the control qubit and his qubit the target qubit and after that a Hadamard gate on Alice's qubit before measuring and recording the classical bit string.

This protocol takes advantage of Proposition 5.4, the fact that Alice's action on her part of the Bell state is *equivalent* to changing the qubit initialization of Bob, see Fig. 5.8. Since the CNOT and Hadarmard gate are reversible, Alice's actions directly influence the result of the readout operations of Bob.



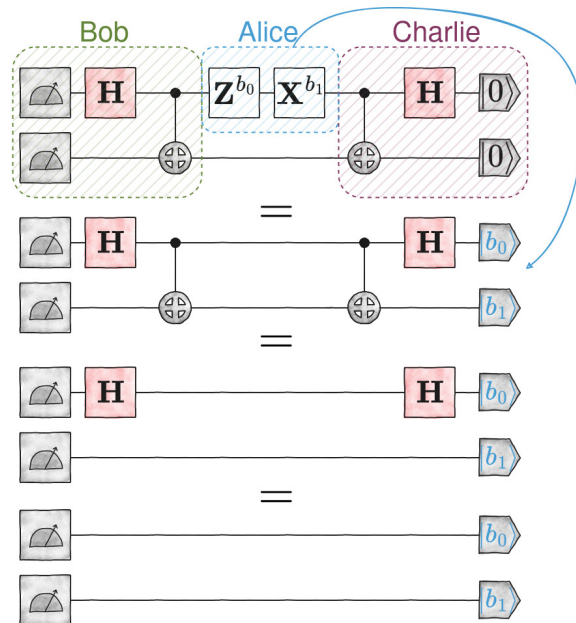


Figure 5.8 Diagrams sketching the equivalences in the superdense coding protocol.

## 5.5 Quantum Games: The Prisoner's Dilemma

Alice and Bob robbed a bank and stashed their loot in their office. Unfortunately, somebody rats them out and the police find the stash. Although not sure if both of them together are the culprits or just one of them, they arrest both of them before they have the chance to talk to each other and interrogate them separately. Alice and Bob face two options during the interrogation: Either they confess and get a reduced sentence or they lie and claim that it was only the other one giving them the chance to deflect the accusation and come free. There is just one catch: if both of them accuse the other one, both of them will face prison time. There are four different outcomes concerning their prison time: 1) Both confess, then both get 1 year in prison, 2) Alice deflects and Bob confesses, Alice comes free and Bob faces 5 years in prison, 3) Bob deflects the accusations and Alice confesses, Bob comes free and Alice faces 5 years in prison, or 4) both of them deflect and accuse each other and both of them get each 3 years in prison. This is visualized in the following table, using tuples of prison time where the first entry refers to Alice's prison time and the second entry to Bob's prison time:

		Alice		
		confess	deflect	
Bob	confess	(1,1)	(0,5)	.
	deflect	(5,0)	(3,3)	

(5.10)

The government is technologically highly advanced and in their internal prison-time-sentence system, they encode the decision into bits; confessing corresponds to bit 0 and accusing the other to bit 1. The procedure is the following: The computer starts in the 00 state and the alleged perpetrators can choose if they want to confess or deflect by applying logical gate operations: confession **C** is represented by the identity operation  $\mathbb{I}$  while deflecting **D** is represented by the bit flip operation **X**,

$$|\psi_{\text{final}}\rangle = \begin{array}{c} \boxed{\mathbb{I}/\mathbf{X}} \text{---} |0\rangle \\ \boxed{\mathbb{I}/\mathbf{X}} \text{---} |0\rangle \end{array} = \begin{array}{c} \boxed{\mathbf{C}/\mathbf{D}} \text{---} |0\rangle \\ \boxed{\mathbf{C}/\mathbf{D}} \text{---} |0\rangle \end{array} . \quad (5.11)$$

The time in prison will be calculated using the readout bits, and for the judicial system to be able to read it off faster our usual bit string states are just labeled with the prison times for Alice and Bobs in the subscription:

$$\begin{aligned} |\psi_{11}\rangle &= |00\rangle \Rightarrow 1 \text{ year for Alice, 1 year for Bob,} \\ |\psi_{05}\rangle &= |10\rangle \Rightarrow 0 \text{ years for Alice, 5 years for Bob,} \\ |\psi_{50}\rangle &= |01\rangle \Rightarrow 5 \text{ years for Alice, 1 year for Bob,} \\ |\psi_{33}\rangle &= |11\rangle \Rightarrow 3 \text{ years for Alice, 3 years for Bob.} \end{aligned}$$

All four combinations of confessing and deflecting from the table 5.10 can be put into equations with input bit string 00 on which logical gates are applied and the final state dictates the prison time,

$$\begin{aligned} (\mathbf{C} \otimes \mathbf{C})|\psi_{11}\rangle &= \mathbf{C}|0\rangle \otimes \mathbf{C}|0\rangle = \mathbb{I}|0\rangle \otimes \mathbb{I}|0\rangle = |00\rangle = |\psi_{11}\rangle, \\ (\mathbf{C} \otimes \mathbf{D})|\psi_{11}\rangle &= \mathbf{C}|0\rangle \otimes \mathbf{D}|0\rangle = \mathbb{I}|0\rangle \otimes \mathbf{X}|0\rangle = |01\rangle = |\psi_{50}\rangle, \\ (\mathbf{D} \otimes \mathbf{C})|\psi_{11}\rangle &= \mathbf{D}|0\rangle \otimes \mathbf{C}|0\rangle = \mathbf{X}|0\rangle \otimes \mathbb{I}|0\rangle = |10\rangle = |\psi_{05}\rangle, \\ (\mathbf{D} \otimes \mathbf{D})|\psi_{11}\rangle &= \mathbf{D}|0\rangle \otimes \mathbf{D}|0\rangle = \mathbf{X}|0\rangle \otimes \mathbf{X}|0\rangle = |11\rangle = |\psi_{33}\rangle. \end{aligned}$$

One can quickly see that both parties confessing would overall minimize the global prison time to 2 years in total, but when looking from Alices' (or Bobs') perspective confessing also can lead to 5 years in prison. While deflecting can lead to no prison time at all or 3 years in prison, the obvious choice here for the individual is to always deflect, although that tactic leads to an overall prison time of 6 years. Since both parties will have the same thought process, both will deflect hence maximizing the global prison time but seemingly minimizing the individual prison time, that is why this problem is called the prisoners' dilemma.

But recently the government went along with the quantum hype and, without really knowing how to use it, bought a quantum computer to replace

its old prison-time-sentence system. And since Bell states are the new hot stuff, of course, there will be Bell states involved.

The changes are the following: first, the matrix representation of deflecting changes to

$$\mathbf{D} = -i\mathbf{Y} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix},$$

which action can also be represented as  $\mathbf{D}|0\rangle = |1\rangle$  and  $\mathbf{D}|1\rangle = -|0\rangle$ .

Secondly, the states encoding prison time look a bit like Bell states now:

$$\begin{aligned} |\psi_{11}\rangle &= \frac{1}{\sqrt{2}} (|00\rangle + i|11\rangle), & |\psi_{05}\rangle &= \frac{1}{\sqrt{2}} (|10\rangle - i|01\rangle), \\ |\psi_{50}\rangle &= \frac{1}{\sqrt{2}} (|01\rangle - i|10\rangle), & |\psi_{33}\rangle &= \frac{1}{\sqrt{2}} (i|00\rangle + |11\rangle). \end{aligned}$$

Nothing changes if Alice and Bob play by the book and either confess or deflect, the government tested that and we can just calculate the prison time states as above and verify it by ourselves:

$$\begin{aligned} (\mathbf{C} \otimes \mathbf{C})|\psi_{11}\rangle &= \frac{1}{\sqrt{2}} [(\mathbf{C} \otimes \mathbf{C})|00\rangle + i(\mathbf{C} \otimes \mathbf{C})|11\rangle] = \frac{1}{\sqrt{2}} [|00\rangle + i|11\rangle] = |\psi_{11}\rangle, \\ (\mathbf{C} \otimes \mathbf{D})|\psi_{11}\rangle &= \frac{1}{\sqrt{2}} [(\mathbf{C} \otimes \mathbf{D})|00\rangle + i(\mathbf{C} \otimes \mathbf{D})|11\rangle] = \frac{1}{\sqrt{2}} [|01\rangle - i|10\rangle] = |\psi_{50}\rangle, \\ (\mathbf{D} \otimes \mathbf{C})|\psi_{11}\rangle &= \frac{1}{\sqrt{2}} [(\mathbf{D} \otimes \mathbf{C})|00\rangle + i(\mathbf{D} \otimes \mathbf{C})|11\rangle] = \frac{1}{\sqrt{2}} [|10\rangle - i|01\rangle] = |\psi_{05}\rangle, \\ (\mathbf{D} \otimes \mathbf{D})|\psi_{11}\rangle &= \frac{1}{\sqrt{2}} [(\mathbf{D} \otimes \mathbf{D})|00\rangle + i(\mathbf{D} \otimes \mathbf{D})|11\rangle] = \frac{1}{\sqrt{2}} [|11\rangle + i|00\rangle] = |\psi_{33}\rangle. \end{aligned}$$

Alice, always up to date on what the quantum world is up to right now, recognizes the quantum character of the new device while being interrogated, and realizes her chance: she doesn't apply the confess nor the deflect gate but another gate which she calls the miracle move:

$$\begin{aligned} \mathbf{M} &= \frac{1}{\sqrt{2}} \begin{pmatrix} i & -1 \\ 1 & -i \end{pmatrix}, \quad \mathbf{M} = \mathbf{SHYSZ}, \\ \mathbf{M}|0\rangle &= \frac{1}{\sqrt{2}} (i|0\rangle + |1\rangle), \quad \mathbf{M}|1\rangle = \frac{1}{\sqrt{2}} (-|0\rangle - i|1\rangle). \end{aligned}$$

By doing so the resulting quantum state is a superposition of different prison time states, and interestingly it doesn't matter what Bob chooses (who is not as up-to-date and just applies the confess or deflect gate). To see that, the application has to be written out as done above, and the resulting bit strings

have to be regrouped to see which superposition this miracle move creates.

$$\begin{aligned}
 (\mathbf{M} \otimes \mathbf{C})|\psi_{11}\rangle &= \frac{1}{\sqrt{2}} [(\mathbf{M} \otimes \mathbf{C})|00\rangle + i(\mathbf{M} \otimes \mathbf{C})|11\rangle], \\
 &= \frac{1}{\sqrt{2}} [\mathbf{M}|0\rangle \otimes \mathbf{C}|0\rangle + i\mathbf{M}|1\rangle \otimes \mathbf{C}|1\rangle], \\
 &= \frac{1}{\sqrt{2}} \left[ \frac{1}{\sqrt{2}}(i|0\rangle + |1\rangle) \otimes |0\rangle + \frac{1}{\sqrt{2}}i(-|0\rangle - i|1\rangle) \otimes |1\rangle \right], \\
 &= \frac{1}{2} [i|00\rangle + |10\rangle - i|01\rangle + |11\rangle], \\
 &= \frac{1}{\sqrt{2}} [|\psi_{33}\rangle + |\psi_{05}\rangle].
 \end{aligned}$$

$$\begin{aligned}
 (\mathbf{M} \otimes \mathbf{D})|\psi_{11}\rangle &= \frac{1}{\sqrt{2}} [(\mathbf{M} \otimes \mathbf{D})|00\rangle + i(\mathbf{M} \otimes \mathbf{D})|11\rangle], \\
 &= \frac{1}{\sqrt{2}} [\mathbf{M}|0\rangle \otimes \mathbf{D}|0\rangle + i\mathbf{M}|1\rangle \otimes \mathbf{D}|1\rangle], \\
 &= \frac{1}{\sqrt{2}} \left[ \frac{1}{\sqrt{2}}(i|0\rangle + |1\rangle) \otimes |0\rangle + \frac{1}{\sqrt{2}}i(-|0\rangle - i|1\rangle) \otimes -|1\rangle \right], \\
 &= \frac{1}{2} [i|00\rangle + |10\rangle + i|01\rangle - |11\rangle], \\
 &= \frac{1}{\sqrt{2}} [|\psi_{33}\rangle - |\psi_{05}\rangle].
 \end{aligned}$$

Alice cheated the system, it doesn't matter what Bob does, Alice might walk free even though Bob deflected and gets 5 years in prison. Worst case, Alice faces 3 years as she would when she deflects. And Bob either faces 3 years or 5 years in prison, which doesn't seem fair because this clearly seems like a rigged game now.

There are more consequences following the introduction of quantum moves into the prisoners' dilemma, in [EWL99] the concept is explained in a more rigorous mathematical formalism. The analysis is being taken even further by parameterizing the quantumness of a move while looking at average prison times for Alice and Bob.

# 6. Entanglement

Date: 15 November 2023

Lecturer: Johannes Kofler

## 6.1 Entanglement

Entanglement is the phenomenon when two or more quantum systems are correlated in such a (non-classical) way that even a perfect and complete description of all individual systems does not fully specify their joint state. And vice versa, knowing everything about their joint state, does not imply maximal knowledge about the individual constituents. When two or more systems are in an entangled state, they – in some sense – cannot be thought of as individual systems anymore, even if they are separated in space. This is, in fact, what Erwin Schrödinger called the “essence of quantum physics”.

In the bipartite case, i.e., for two quantum systems  $A$  and  $B$ , product (or separable) states have the form

$$|\psi\rangle_{AB} = |\varphi\rangle_A \otimes |\varphi\rangle_B. \quad (6.1)$$

**Definition 6.1 (entanglement, bipartite case).** A pure bipartite quantum state  $|\psi\rangle_{AB}$  is *entangled* if and only if it is not a product state (i.e. not separable). This means that it is impossible to write  $|\psi\rangle_{AB}$  in the form of Eq. (6.1).

The Bell states are, of course, not of this form, i.e. there is no way to write a Bell state such that it factorizes into an individual state  $|\varphi\rangle_A$  for Alice and an individual state  $|\varphi\rangle_B$  for Bob. There also exist measures to quantify the amount of entanglement in a quantum state, and the Bell states are indeed maximally entangled.

As we will see later, entangled states can give rise to correlations whose “strength” cannot be achieved by any classical process. Moreover, entanglement

### Agenda:

- 1 entanglement
- 2 the CHSH game
- 3 quantum key distributions (QKD) revisited: the E91 protocol

entangled quantum states are not separable

Bell states are maximally entangled

is a necessary resource for many quantum information technologies such as quantum computing and entanglement-based quantum cryptography.

### 6.1.1 Rotated Bell states

Corollary 5.3, see also Fig. 5.4, highlights that the Bell state somehow correlates both qubit wires in a nontrivial way. In fact, it is impossible to discern whether a certain unitary (think: gate, circuit) has been applied to the first qubit or the second one. Both qubits (wires) are linked. These circuit reformulations, however, do feature a transpose. And this can be different from the (complex-valued) adjoint. For real-valued matrices, however, transposition and adjunction coincide and both denote the inverse of the continuous gate. Rotation gates are one rich family of real-valued unitary transformations. Parametrized by a single angle, they correspond to

$2 \times 2$  rotation matrix

$$\mathbf{R}(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \quad \text{with } \theta \in [0, 2\pi).$$

Note that we use polarization (or spin) angles here, which are half as large as angles on the Bloch sphere. So, we do not need all the factors  $1/2$  as in Lecture 3. Rotation matrices are comparatively intuitive and obey very nice composition and inversion rules.

**Fact 6.2** Let  $\mathbf{R}(\theta_A), \mathbf{R}(\theta_B)$  be two rotation matrices. Then,

$$\begin{aligned} \mathbf{R}(\theta_A)\mathbf{R}(\theta_B) &= \mathbf{R}(\theta_A + \theta_B) && \text{(composition),} \\ \mathbf{R}(-\theta_A) &= \mathbf{R}(\theta_A)^T = \mathbf{R}(\theta_A)^{-1} && \text{(transposition/inversion).} \end{aligned}$$

■

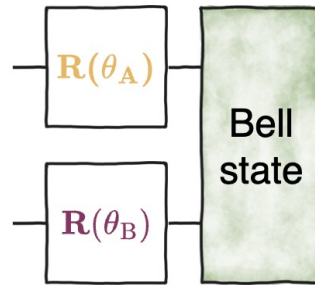
**Exercise 6.3** Verify the composition rule of Fact 6.2 by computing the matrix product and using trigonometric identities. Then, conclude the transposition/inversion rule directly from the composition rule.

It is interesting to consider Bell states, where each qubit is rotated by a different angle. From now on we use subscript  $A$  to denote the first qubit and subscript  $B$  to denote the second qubit. This notation convention will become clear later on. For  $\theta_A, \theta_B$ , we define

$$|\psi_{\text{Bell}}(\theta_A, \theta_B)\rangle = \mathbf{R}(\theta_A) \otimes \mathbf{R}(\theta_B) |\psi_{\text{Bell}}\rangle. \quad (6.2)$$

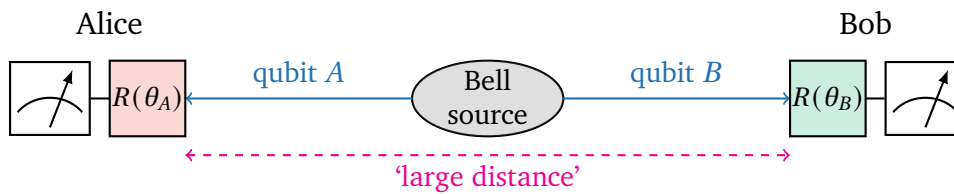
This state can be created by applying two independent rotation gates to each qubit, Fig. 6.1.

Access to a quantum computer is one way to prepare such rotated Bell states. But it is not the only one, and far from the most interesting case. All we need to create such a state is a source that produces two qubits whose joint quantum state is described by  $|\psi_{\text{Bell}}\rangle$ . This can, for instance, be achieved by nanomaterials – so-called quantum dots – that create a pair of photons whose



**Figure 6.1** Preparing the state  $|\psi_{\text{Bell}}(\theta_A, \theta_B)\rangle$  by rotating each qubit at an angle  $\theta_A$  and  $\theta_B$  respectively.

polarization degree is maximally entangled.<sup>1</sup> These photons can then travel (with the speed of light) to distant locations. There, quantum aficionados – whom we call Alice and Bob – can ‘catch’ the photons and use polarization filters to implement each rotation. In this context, the following visualization is more appropriate:



This visualization depicts a scenario that is equivalent to the circuit in Fig. 6.1. But, the underlying geometry is very different. Since their creation, the two qubits have travelled in opposite directions. The recipients, Alice and Bob, are very far away from each other, apply rotations independently and perform single-qubit measurements. Nonetheless, the measurement outcomes they can obtain remain very correlated. And, what is more severe, rotations performed by Alice appear to affect Bob’s side and vice versa. This alternative viewpoint isolates the strangeness of the following, mathematically valid, statement.

**Lemma 6.4** For any  $\theta_A, \theta_B \in [0, 2\pi)$ , the outcome probabilities of measuring the rotated Bell state in Eq. (6.2) always obey

$$\Pr_{|\psi_{\text{Bell}}(\theta_A, \theta_B)\rangle} [o = 00] = \Pr_{|\psi_{\text{Bell}}(\theta_A, \theta_B)\rangle} [o = 11] = \frac{1}{2} \cos^2(\theta_A - \theta_B),$$

$$\Pr_{|\psi_{\text{Bell}}(\theta_A, \theta_B)\rangle} [o = 01] = \Pr_{|\psi_{\text{Bell}}(\theta_A, \theta_B)\rangle} [o = 10] = \frac{1}{2} \sin^2(\theta_A - \theta_B).$$

*Proof.* Let us combine Corollary 5.3, Theorem 7.1 and the defining properties

<sup>1</sup>Armando Rastelli from the physics department at JKU is, in fact, one of the world-leading experts in fabricating such entangling photon sources.

Bell states preserve correlations under joint rotations

of rotation matrices (Fact 6.2) to obtain

$$\begin{aligned} |\psi_{\text{Bell}}(\theta_A, \theta_B)\rangle &= \mathbf{R}(\theta_A) \otimes \mathbf{R}(\theta_B) |\psi_{\text{Bell}}\rangle = \mathbb{1} \otimes \mathbf{R}(\theta_B) \mathbf{R}(\theta_A)^T |\psi_{\text{Bell}}\rangle \\ &= \mathbb{1} \otimes \mathbf{R}(\theta_B - \theta_A) |\psi_{\text{Bell}}\rangle \\ &= \frac{\cos(\theta_B - \theta_A)}{\sqrt{2}} |0, 0\rangle + \frac{\sin(\theta_B - \theta_A)}{\sqrt{2}} |0, 1\rangle \\ &\quad - \frac{\sin(\theta_B - \theta_A)}{\sqrt{2}} |1, 0\rangle + \frac{\cos(\theta_B - \theta_A)}{\sqrt{2}} |1, 1\rangle. \end{aligned}$$

We can now square these amplitudes to obtain the probabilities of the 2-bit outcomes in question. ■

The following two extreme cases are noteworthy:

- 1  $\theta_A = \theta_B$  (same rotation on both qubits): in this case  $\theta_A - \theta_B = 0$  and the trigonometric relations  $\cos^2(0) = 1$ ,  $\sin^2(0) = 0$  ensure

$$\Pr_{|\psi_{\text{Bell}}(\theta_A, \theta_B)\rangle} [o = 00] = \Pr_{|\psi_{\text{Bell}}(\theta_A, \theta_B)\rangle} [o = 11] = \frac{1}{2},$$

while the outcomes 0, 1 and 1, 0 can never occur. This always produces perfectly correlated outcome bits for both qubits.

- 2  $\theta_A = \theta_B \pm \pi/2$  (shifted angle): in this case  $\theta_A - \theta_B = \pm\pi/2$  and the trigonometric relations  $\cos^2(\theta/2) = 0$ ,  $\sin^2(\theta/2) = 1$  ensure

$$\Pr_{|\psi_{\text{Bell}}(\theta_A, \theta_B)\rangle} [o = 01] = \Pr_{|\psi_{\text{Bell}}(\theta_A, \theta_B)\rangle} [o = 10] = \frac{1}{2},$$

while the outcomes 0, 0 and 1, 1 can never occur. This produces perfectly anticorrelated outcome bits for both qubits.

Lemma 6.4 interpolates between those extreme cases in a smooth fashion. If  $\theta_A$  and  $\theta_B$  are close, we are still likely to get very correlated output bits. If instead  $\theta_A - \theta_B$  is close to  $\pi/2$ , we are likely to obtain very anti-correlated output bits instead. Changing the relative difference of both angles allows us to interpolate between perfect correlation and perfect anticorrelation.

## 6.2 The CHSH game and Bell inequalities

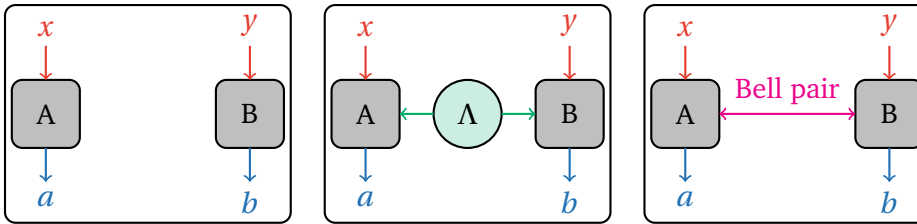
### 6.2.1 The CHSH game

The CHSH game is a modern view on a seminal thought experiment by John Clauser (Nobel Prize 2022), Michael Horne, Abner Shimony and Richard Holt from 1969. It has been intended to test the fundamental limits of (any) classical explanation for quantum mechanical effects that involve the Bell state. This is a clever sharpening of revolutionary observations by John Bell in 1964 [Bel64].

The CHSH game involves two players, A (for Alice) and B (for Bob), as well as a referee. We refer to Fig. 6.2 for a visualization. Throughout the duration of the game, A and B must not communicate with each other. Each of them receives a uniformly random bit from a quizzmaster and are tasked to commit to another single bit as output:

the CHSH game





**Figure 6.2** Three variants of the CHSH game: Two players, A (for Alice) and B (for Bob), play as partners in the following setting. A quizmaster (red) provides each with a uniformly random input  $x \in \{0, 1\}$  for A and  $y \in \{0, 1\}$  for Bob. They then have to output one bit (blue) each. They win the game if  $a \oplus b = x \wedge y$ . The interesting twist is that A and B cannot talk to each other once the game has started. There are three potential scenarios that meet these overall desiderata: (Left): A and B perform purely deterministic strategies (grey boxes that implement a single-bit function). (Center:) similar setting, but A and B share some joint random seed  $\Lambda$  (green) before the game starts. This allows them to potentially hedge bets and gamble. (Right:) In the quantum variant of the CHSH game, A and B share a Bell state (magenta) which they can rotate and measure after the game has started.

- A receives  $x \in \{0, 1\}$  (uniformly random) and outputs  $a \in \{0, 1\}$ ,
- B receives  $y \in \{0, 1\}$  (uniformly random) and outputs  $b \in \{0, 1\}$ .

The players win if the two output bits obey  $a \oplus b = x \wedge y$ , so the optimal strategy depends on both input bits. More precisely:

- 1  $(x, y) = (0, 0)$  implies that they win if  $(a, b) = (0, 0)$  or  $(a, b) = (1, 1)$  (perfect correlation),
- 2  $(x, y) = (0, 1)$  implies that they win if  $(a, b) = (0, 0)$  or  $(a, b) = (1, 1)$  (perfect correlation),
- 3  $(x, y) = (1, 0)$  implies that they win if  $(a, b) = (0, 0)$  or  $(a, b) = (1, 1)$  (perfect correlation),
- 4  $(x, y) = (1, 1)$  implies that they win if  $(a, b) = (0, 1)$  or  $(a, b) = (1, 0)$  (perfect anti-correlation),

This looks like an easy and somewhat boring game. But, remember that Alice and Bob cannot talk to each other! And while three game settings suggest an easy winning strategy, the fourth setting (perfect anti-correlation) asks for a completely orthogonal strategy. And, since A and B only have access to one input bit, how should they prepare for this situation?

### 6.2.2 Optimal classical strategies

The dilemma from above turns out to be impossible to overcome with traditional means. This is the content of the following statement that basically underscores that the naive strategy – Alice and Bob both always output 0 (1) – is optimal among all deterministic strategies.

**Proposition 6.5** The best deterministic classical strategy for the CHSH game wins with probability  $3/4 = 0.75$ .

*Proof.* Both players, Alice and Bob, receive a (uniformly random) bit and are tasked to produce a single output bit. There are only four different single-bit functions that make sense in this context:

$$\begin{array}{ll} f_0(0) = 0, f_0(1) = 0 & \text{(constant, always 0),} \\ f_1(0) = 0, f_1(1) = 1 & \text{(balanced, identity),} \\ f_2(0) = 1, f_2(1) = 0 & \text{(balanced, bit-flip),} \\ f_3(0) = 1, f_3(1) = 1 & \text{(constant, always 1).} \end{array}$$

Recall that there are 4 possible inputs for the CHSH game. Three of them  $((0, 0), (0, 1), (1, 0))$  ask for perfectly correlated output bits  $((0, 0)$  or  $(1, 1))$  while only one input  $(1, 1)$  requires perfectly anti-correlated output bits  $((0, 1)$  or  $(1, 0))$ . Importantly, Alice (Bob) doesn't have access to the full input string, they only see the first (second) bit. So, there is no way to prepare a strategy to handle the fourth scenario. Instead, it seems better to always provide correlated outputs, e.g. by jointly agreeing to always output 0 (Alice and Bob always apply  $f_0$  to their input) or, equivalently, by jointly agreeing to always output 1 (Alice and Bob both apply  $f_3$  to their input bits). Such a perfectly correlated strategy succeeds in 3 of the 4 possible scenarios. Since all four scenarios occur with equal probability, the overall probability of success is  $3/4 = 0.75$ , as advertised.

An exhaustive search over all possible combinations of  $4^2 = 16$  function combinations underscores that these strategies are indeed as good as it gets. ■

The above proof reveals a dilemma that arises when Alice and Bob want to come up with a very good strategy for the CHSH game. Each of them only receives one-half of the input string, and they must not (cannot) communicate while the game is going on. This partial information is not enough for them to discern whether the interesting special case (input  $((1, 1)$  which asks for anti-correlated bits) has actually happened. And so, it does not seem to make sense to pay attention to this special case at all. Stubbornly outputting 0, regardless of the actual input, looks like a highly competitive strategy.

An interesting question is now whether there are *randomized* strategies that allow Alice and Bob to do better than that. A randomized strategy could look as follows: before the game starts, Alice and Bob are allowed to meet, conspire and exchange information. They could use this opportunity to share a random seed  $\Lambda$  that allows them to hedge bets and 'gamble' later on in the game. This shared random seed should really be viewed as a mathematical abstraction of a joint strategy that may involve additional information, as well as multiple strategies.

But, to not interfere with the defining rule of the CHSH game, we insist that the shared random seed is (statistically) independent from the random bits  $x$  and  $y$  the referee is going to use once the game actually starts in

earnest. This assumption is crucial and actually sufficient to prove the following generalization of Proposition 6.5.

**Theorem 6.6 (Bell inequality for CHSH).** Any classical strategy conceivable – even those that use (arbitrary amounts of) shared randomness between both players – can win the CHSH game with a probability of at most  $3/4 = 0.75$ . Hence, the Bell inequality for the CHSH game, limiting all classical strategies, reads:  $p_{\text{succ}}^{\text{class}} \leq \frac{3}{4}$ .

best classical CHSH strategy wins with probability  $\leq 0.75$

To be more precise, this upper bound on the optimal classical strategy hinges on two explicit assumptions:

- separated players ('locality'): Alice receives input bit  $x$  and outputs  $a \in \{0, 1\}$ ; she has zero information about either  $y$  or  $b$  from Bob's side. Analogously, Bob's outcome  $b$  does not depend on  $x$  or  $a$ .
- uncorrelated randomness ('free will'): the referee samples both input bits  $x, y$  uniformly at random; importantly, these random numbers are completely uncorrelated from the shared random seed  $\Lambda$  that Alice and Bob use to power their strategies.

There is a third implicit assumption, that is sometimes called 'realism'. It states that it is possible to assign probabilities to all input/output tuples arising from potential strategies, irrespective of whether they occur in the actual game or not.

*Proof sketch of Theorem 6.6.* The overall idea is that shared randomness cannot improve over the best deterministic strategy. A shared random seed allows the players to switch between different deterministic strategies – depending on the value of the random seed. The resulting probability of success then becomes a weighted sum over winning probabilities of individual deterministic strategies:

$$p_{\text{succ}} = \sum_k \Pr[\text{success} | \text{strategy } k] p(k) \quad \text{with} \quad p_k \geq 0, \sum_k p(k) = 1.$$

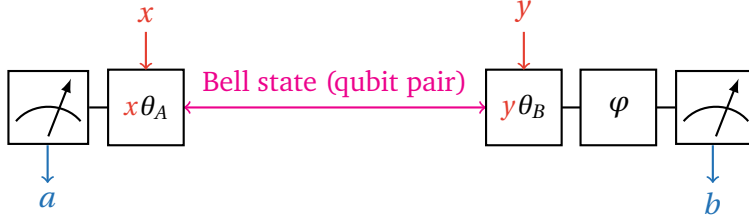
But, we already know from Proposition 6.5 that  $\Pr[\text{success} | \text{strategy } k] \leq 3/4$  for all  $k$ . It is impossible to overcome this threshold with probabilistic averaging.

Note, however, that this strategy only works if the random seed ( $p(k)$  in our case) is statistically independent of the game settings  $(x, y)$ . Otherwise, the expression would not factorize nicely and the argument becomes void. This is why uncorrelated randomness is an important assumption behind Theorem 6.6. ■

### 6.2.3 Optimal quantum strategy

We have now prepared the stage for a very astonishing observation: a quantum generalization of the CHSH – which does not violate any game rules – allows the players to win with a success probability that is quite a bit larger than  $3/4 = 0.75$ . To achieve such a noteworthy improvement, Alice and Bob share a

Bell state between them before the game starts, see Fig. 6.2 (right). Once the game begins, each player uses the input bit to perform a certain rotation on their half of the Bell state and measure their qubit:



At first sight, this setup looks a bit asymmetric. Bob features an additional rotation gate  $R(\varphi)$  while Alice doesn't. Due to the invariants of a rotated Bell state, this is in fact the most general setup conceivable. Consider now the scenario where Alice receives input bit  $x$  and Bob receives input bit  $y$ . After applying rotations, they produce a rotated Bell state that depends on  $x$  and  $y$ . We succinctly write

$$|\psi(x, y)\rangle := |\psi_{\text{Bell}}(x\theta_A, y\theta_B + \varphi)\rangle = \mathbf{R}(x\theta_A) \otimes \mathbf{R}(y\theta_B + \varphi) |\psi_{\text{Bell}}\rangle.$$

Note that since this state depends on the rotation angles  $\theta_A, \theta_B, \varphi \in [0, 2\pi)$ , Lemma 6.4 conveniently provides us with the associated outcome probabilities for perfectly correlated and anti-correlated measurement outcome bits:

$$\begin{aligned} \Pr_{|\psi(x,y)\rangle} [o = 00] &= \Pr_{|\psi(x,y)\rangle} [o = 11] = \frac{1}{2} \cos^2(-x\theta_A + y\theta_B + \varphi), \\ \Pr_{|\psi(x,y)\rangle} [o = 01] &= \Pr_{|\psi(x,y)\rangle} [o = 10] = \frac{1}{2} \sin^2(-x\theta_A + y\theta_B + \varphi). \end{aligned}$$

We can now use these probabilities to analyze the success probability in each CHSH game setting as a function of the rotation angles involved:

- 1  $(x, y) = (0, 0)$  which asks for  $0 = x \wedge y = a \oplus b$ . In words, Alice and Bob win the game if their output bits are perfectly correlated, that is they should output either  $(a, b) = (0, 0)$  or  $(a, b) = (1, 1)$ . The probability of success is

$$\begin{aligned} p_{\text{succ}}(0, 0) &= \Pr_{|\psi(0,0)\rangle} [o = 00] + \Pr_{|\psi(0,0)\rangle} [o = 11] \\ &= \frac{1}{2} \cos^2(-0\theta_A + 0\theta_B \varphi) + \frac{1}{2} \cos^2(-0\theta_A + 0\theta_B + \varphi) \\ &= \cos^2(\varphi). \end{aligned}$$

- 2  $(x, y) = (0, 1)$  which asks for  $0 = x \wedge y = a \oplus b$ . Again, Alice and Bob win if their output bits are perfectly correlated. The probability of success is

$$\begin{aligned} p_{\text{succ}}(0, 1) &= \Pr_{|\psi(0,1)\rangle} [o = 00] + \Pr_{|\psi(0,1)\rangle} [o = 11] \\ &= \frac{1}{2} \cos^2(-0\theta_A + 1\theta_B \varphi) + \frac{1}{2} \cos^2(-0\theta_A + 1\theta_B + \varphi) \\ &= \cos^2(\theta_B + \varphi). \end{aligned}$$

- 3  $(x, y) = (1, 0)$  which asks for  $0 = x \wedge y = a \oplus b$ . For the last time, Alice and Bob win if their output bits are perfectly correlated. The probability of success is

$$\begin{aligned} p_{\text{succ}}(1, 0) &= \Pr_{|\psi(1,0)\rangle} [o = 00] + \Pr_{|\psi(1,0)\rangle} [o = 11] \\ &= \frac{1}{2} \cos^2(-1\theta_A + 0\theta_B + \varphi) + \frac{1}{2} \cos^2(-1\theta_A + 0\theta_B + \varphi) \\ &= \cos^2(-\theta_A + \varphi). \end{aligned}$$

- 4  $(x, y) = (1, 1)$  which asks for  $1 = x \wedge y = a \oplus b$ . In this last scenario, Alice and Bob win if their output bits are perfectly anti-correlated. The probability of success is

$$\begin{aligned} p_{\text{succ}}(1, 1) &= \Pr_{|\psi(1,1)\rangle} [o = 01] + \Pr_{|\psi(1,1)\rangle} [o = 10] \\ &= \frac{1}{2} \sin^2(-1\theta_A + 1\theta_B + \varphi) + \frac{1}{2} \sin^2(-1\theta_A + 1\theta_B + \varphi) \\ &= \sin^2(\theta_A + \theta_B + \varphi). \end{aligned}$$

We can now take into account the distribution of inputs  $(x, y)$  to combine all these four probabilities into a single success probability. By assumption, all possible 2-bit inputs occur with equal probability  $1/2^2 = 1/4$  (uniform distribution). So, the overall probability of success becomes

$$p_{\text{succ}} = \frac{1}{4} [\cos^2(\varphi) + \cos^2(\theta_B + \varphi) + \cos^2(-\theta_A + \varphi) + \sin^2(-\theta_A + \theta_B + \varphi)].$$

We are now in a position to optimize the quantum strategy by choosing  $\theta_A, \theta_B, \varphi$  to make this success probability as large as possible. The cosine function becomes large if all the angles are (relatively) close to zero and is symmetric, i.e.  $\cos(-\alpha) = \cos(+\alpha)$ . Here is one choice that yields the same success probability for each of the four challenges:

$$\theta_A = -\pi/4, \theta_B = +\pi/4, \varphi = -\pi/8.$$

It achieves

$$p_{\text{succ}} = \cos^2(\pi/8) = \frac{1}{4} (2 + \sqrt{2}) \gtrsim 0.85.$$

Although not perfect, this success probability is considerably larger than the best classical success probability ( $3/4 = 0.75$ ) conceivable. In fact, it can be shown that this quantum success probability is optimal. Even by using (arbitrary) quantum resources and (arbitrary) quantum circuits one cannot beat this threshold. This is worth a prominent display.

**Theorem 6.7 (optimal quantum strategy for the CHSH game).** For the CHSH game (2 players), there is an optimal quantum strategy that (only) uses 2-qubit Bell states and two single-qubit rotations for each player. This

best quantum CHSH strategy wins with probability  $> 0.85$

strategy achieves a success probability of  $p_{\text{succ}} = (2 + \sqrt{2})/4 \gtrsim 0.85$ .

### 6.3 CHSH rigidity and monogamy of entanglement

In this section, we briefly discuss the implications of actually observing a success probability that is (close to)  $p_{\text{succ}} = (2 + \sqrt{2})/4$ . It turns out that such a high probability of success is only possible if Alice and Bob share a (state close to the) maximally entangled Bell state.

**Fact 6.8 (CHSH rigidity and monogamy, informal).** Suppose that Alice and Bob play the CHSH game, but do not necessarily know (or trust) whether they actually both measure one half of a Bell state. Then, they can use the success probability they achieve to test their assumptions. In fact, an optimal success probability of  $(2 + \sqrt{2})/4$  is only achievable if they indeed share a (possibly rotated) Bell state. Moreover, this even implies that their two-qubit state is completely uncorrelated (think: private) from any external players. ■

This fact subsumes two strong observations: (i) playing a CHSH game and winning with optimal success probability essentially certifies that the underlying protocol works as intended. There is no additional need for benchmarking and/or bug fixing. This remarkable feature is also called ‘self-testing’.

(ii) the quantum correlations within a two-qubit Bell state are maximally strong. They are, in fact, so strong that it is impossible to still couple this quantum system to another one. This feature is known as *monogamy of entanglement*.

*monogamy of entanglement:* if two quantum systems are maximally entangled they cannot be entangled with a third one

### 6.4 Bell inequalities and the violation of local realism

We should note that the importance of the violation of Bell’s inequalities goes far beyond winning a game with larger-than-classical success probability. In fact, what John Bell did [Bel64], was a rather historic achievement: He made it possible to experimentally test a hitherto almost metaphysical problem, which some of the greatest minds in the history of physics could not solve, namely whether or not the physical world may eventually allow for a classical description after all. I.e. can there exist an underlying reality “beneath” the quantum state, described by “hidden variables” (similar to the random seed  $\Lambda$  introduced earlier) not yet known in quantum theory? This classical worldview is called “realism”. Alone, it does not seem to be testable. But Bell combined it with two more very plausible assumptions, denoting the resulting worldview of “local realism”.

Local realism encompasses all classical theories about the physical universe which obey the three assumptions of realism (physical properties are defined by hidden variables and exist independent of and prior to measurement), locality (no influence can propagate faster than the speed of light), and freedom of choice (measurement settings can be chosen independently of the hidden variables).

The worldview of local realism implies Bell inequalities. Over many decades, Bell inequalities have been violated in laboratories all around the world. The 2022 Nobel Prize in Physics was awarded to some of the most significant of these experiments.

## 6.5 The E91 protocol for quantum key distribution

We now have seen two compelling features of the two-qubit Bell state. On the one hand, it allows two agents, Alice and Bob, to obtain perfectly correlated output bits that are still random  $((0, 0)$  and  $(1, 1)$  with probability  $1/2$  each). And, on the other hand, the agents can play a CHSH game to self-test and certify the underlying setup. Arthur Ekert was the first to realize the potential impact of a combination of these two effects [Eke91]. He devised a protocol, which first appeared in 1991, that uses shared entanglement to establish a private random key between two distant parties. This key can then be used in a one-time-pad protocol, which is an information-theoretic secure encryption technique, i.e. it remains safe even if an adversary has infinite computing power.

The key idea is to distribute many Bell states among Alice and Bob. Each of them uses private randomness to measure halves of each state in one of several designated basis. Each measurement provides Alice with a private bit that is strongly correlated – via entanglement in the original Bell state – with the private bit that Bob has obtained in the same round.

Once this randomized (Bell) measurement stage is completed, Alice and Bob exchange their basis choices over a public channel, e.g. the internet or a telephone. By itself, the choice of basis setting does not reveal any information about the obtained measurement outcomes. They then identify instances, where they happened to measure in the same basis, to identify perfectly correlated output bits. These then form the basis of their private key. But, before jumping to premature conclusions, they also use a considerable amount of their measurement data to imitate a CHSH game and compute their (approximate) success probability. Note that, in this step, it is necessary to also communicate measurement outcomes with each other. So, the bits they use for CHSH testing cannot be used as a private key anymore. Nonetheless, this CHSH game imitation is essential for the protocol and equips it with the uncanny ability to detect ‘wiretap’ or ‘man in the middle’ attacks.

If they achieve a close-to-optimal CHSH success probability, the rigidity of the CHSH game and monotony of entanglement (Fact 6.8) ensure that their protocol must have worked as intended. In particular, the shared correlations between their Bell states are private in the sense that they cannot be correlated to any external quantum system that is beyond their control. The latter is very general and includes a potential eavesdropper (‘man in the middle’), even if this agent is very powerful and has access to arbitrary amounts of (quantum) computing resources. As soon as such a hypothetical eavesdropper tampers with the shared Bell states, they must interfere with the perfect entanglement between Alice and Bob (monogamy of entanglement). And such an attack

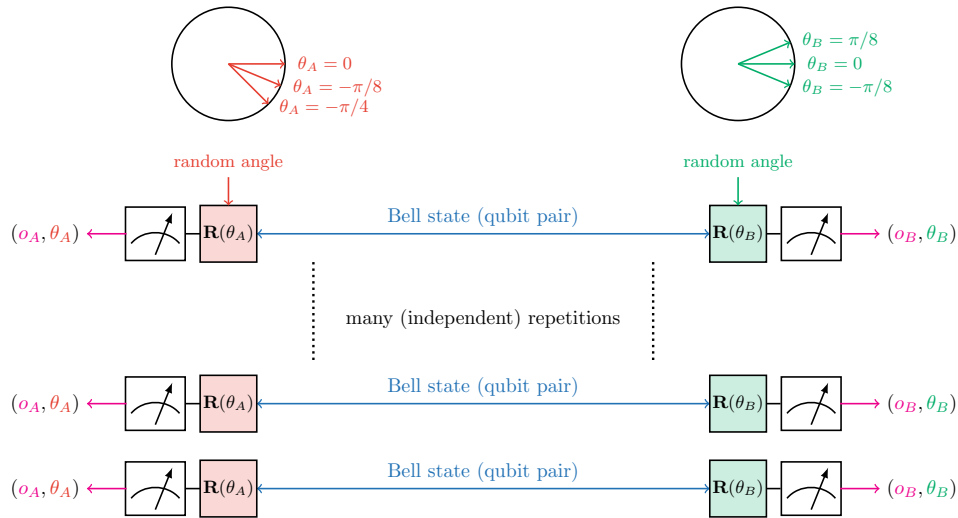
shared entanglement enables detecting ‘wiretap’ or ‘man in the middle’ attacks

would necessarily manifest itself in a (much) smaller success probability for the CHSH game. Alice and Bob, however, can check for exactly that and abort their protocol if their imitation of the CHSH game is not as successful as it ought to be.

The actual  $E_{91}$  protocol uses 3 different rotations for each participant (Alice and Bob) that are depicted in Fig. 6.3. A quick look at them reveals that two of each are perfectly aligned with each other. These are well-suited to establish shared randomness (via perfect correlation of Bell outcome measurements). On the other hand, two of each, are also perfectly suited to play the winning strategy for the CHSH game. A full and secure execution of the  $E_{91}$  protocol requires access to many shared Bell pairs so that we can generate sufficiently long shared private keys and also have enough statistics to approximately determine the CHSH success probability. We encourage you to have a closer look by yourself and implement a variant of the  $E_{91}$  protocol using, for instance, QISKIT to simulate the generation and subsequent measurement of Bell states. Such a small coding project would also allow you to discover for yourself, how powerful the CHSH game is at detecting potential attacks on the Bell state. A brutal man-in-the-middle-attack could, for instance, involve an additional third qubit which the eavesdropper swaps into the circuit to funnel out one-half of the entangled state. The CHSH test, however, would detect such a clumsy attack almost immediately.

**Exercise 6.9** Go through our high-level description of the  $E_{91}$  protocol and make it precise. Which pairs of basis rotations allow for extracting a pair of perfectly correlated, uniformly random bits? Which pairs of basis rotations instead allow for playing a CHSH game? **Optional:** use QISKIT to simulate the  $E_{91}$  protocol in the presence of a malicious eavesdropper who tampers with the pristine Bell-state preparation circuit. Show that the (estimated) CHSH success probability really drops below  $(2 + \sqrt{2})/4$  in any such scenario.





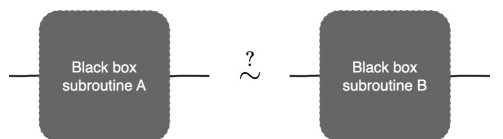
**Figure 6.3** Illustration of the *E91* protocol: Two players – Alice on the left and Bob on the right – share many perceived Bell pairs (blue) among. For each Bell state, they perform independent rotations that are selected uniformly from three options (red circle for Alice, green circle for Bob) and follow it up with a computational basis measurement (magenta). Two of the three angles are the same on each side. The entire process gives each party a list of measurement settings and corresponding results. They then share their measurement settings over a public channel. If the same angle was measured – this occurs in 2 of the 9 setting combinations – then the perfect correlations of Bell state measurements provide them with one shared random bit. In 4 out of 9 cases, their measurement setting combination belongs to the CHSH game. Then, they also communicate their measurement outcomes as they are required for the calculation of the success probability. If the estimated success probability is (close to) optimal, they can be sure that the entire protocol has worked as intended. In particular, no eavesdropping whatsoever can have taken place. If the success probability is not close to optimal, something fishy must have happened. Alice and Bob then abort the protocol, because their shared key may not be correct and/or secure.

# 7. Quantum teleportation

Date: 22 November 2023

## 7.1 Motivation

Today, we prepare the stage for scaling up our quantum architectures to many qubits. An important prerequisite to doing so is the ability to recognize sub-routines and reason about them. In particular, we want to know whether two quantum subroutines are equivalent or not. In pictures,



The following rigorous statement fully resolves this question for the case of a single qubit<sup>1</sup>.

**Theorem 7.1 (equivalence of single-qubit functionalities).** Two single-qubit functionalities (e.g. sub-circuits)  $A, B$  are equivalent if they *always* lead to the same readout probabilities. That is, for all input states  $|\psi\rangle$  and all single-qubit unitaries  $U$ , we must have

$$\Pr_{UA|\psi\rangle} [o = s] = \Pr_{UB|\psi\rangle} [o = s] \quad \text{for } s \in \{0, 1\},$$

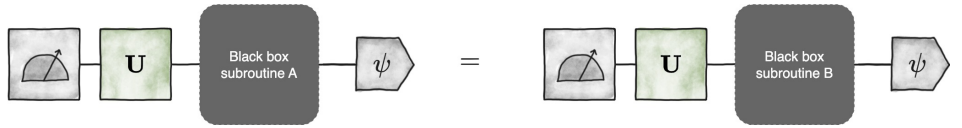
The requirements put forth by this statement are best visualized in another

<sup>1</sup>A generalization to  $n \geq 1$  qubits is relatively straightforward, but would go beyond the scope of this lecture.

### Agenda:

- 1 motivation
- 2 marginal & conditional probabilities
- 3  $T$ -gate teleportation
- 4 state teleportation

picture:



for every input state  $|\psi\rangle$  and every subsequent unitary gate  $U$ . This characterization of equivalence is intuitive: two (black box) subroutines are equivalent if and only if it is impossible to detect any functional difference between the two.

It should not come as a surprise that quantum circuits can ‘hide’ functional differences better than conventional circuits. The following example highlights that different input states and different unitaries can both be necessary to detect them.

**Example 7.2 (checking different input states  $|\psi\rangle$  and unitaries  $U$  matters for Theorem 7.1).** Consider  $A = I$  (do nothing) and  $B = Z$  (sign flip). Then,

$$B|0\rangle = |0\rangle = A|0\rangle \quad \text{and} \quad B|1\rangle = -|1\rangle \sim |1\rangle = A|1\rangle,$$

but the two gates are clearly not equivalent. To see this, set  $|\psi\rangle = |+\rangle = H|0\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$  (and recall  $|-\rangle = H|1\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$ ). Then,

$$\begin{aligned} B|\psi\rangle &= B|+\rangle = B(|0\rangle + |1\rangle)/\sqrt{2} = (|0\rangle - |1\rangle)/\sqrt{2} = |-\rangle = H|1\rangle, \\ A|\psi\rangle &= A|+\rangle = |+\rangle = H|0\rangle. \end{aligned}$$

Both states describe equal superpositions between 0 and 1 and produce equivalent readout probabilities:

$$\begin{aligned} \Pr_{B|\psi\rangle} [o_0 = 0] &= \Pr_{|+\rangle} [o_0 = 0] = |\langle 0|+\rangle|^2 = \frac{1}{2}, \\ \Pr_{A|\psi\rangle} [o_0 = 0] &= \Pr_{|-\rangle} [o_0 = 0] = |\langle 0|-\rangle|^2 = \frac{1}{2}. \end{aligned}$$

Nonetheless, the actual states are very different from each other. Applying one subsequent Hadamard gate  $U = H$  reveals this difference:  $UB|\psi\rangle = H|-\rangle = H \times H|1\rangle = |1\rangle$ , while  $UA|\psi\rangle = H|+\rangle = H \times H|0\rangle = |0\rangle$ . This ensures

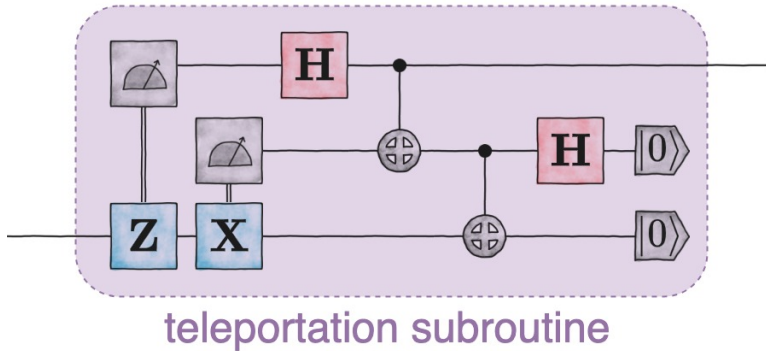
$$\begin{aligned} \Pr_{UB|\psi\rangle} [o_0 = 0] &= \Pr_{|1\rangle} [o_0 = 0] = 0, \\ \Pr_{UA|\psi\rangle} [o_0 = 0] &= \Pr_{|0\rangle} [o_0 = 1] = 1. \end{aligned}$$

These readout probabilities are as different as they get: one always produces 0 and one always produces 1. ■

This example showcases that we may need the ability to choose different input states *and* different subsequent unitaries to unravel a functional difference between quantum sub-routines. Theorem 7.1 then follows from carefully arguing that this is enough to unravel *all* functional differences between single-qubit circuits. We leave this analysis as an instructive exercise.

**Exercise 7.3 (Proof of Theorem 7.1).** Provide a proof of Theorem 7.1 for the special case where both  $A$  and  $B$  are single-qubit gates.

At this point, it is worthwhile to emphasize that Theorem 7.1 also applies to more general quantum subroutines. The subroutines we analyze today, for instance, involve more than one qubit. A readout is performed on these auxiliary qubits. And, depending on the outcome obtained ( $o_k = 0$  or  $o_k = 1$ ), we perform different quantum gates on the remaining qubit wire. For instance,



where double lines indicate the conditional application of a quantum gate (if readout is 1, we apply the gate; otherwise we do nothing).

In order to properly analyze such quantum (sub-)routines, we need a framework that allows us to reason about such conditional operations, i.e. (gate) actions that depend on a random outcome we have observed earlier.

## 7.2 Background: marginal and conditional probabilities

### 7.2.1 Marginal probabilities

For simplicity and concreteness, we will focus on probability distributions that address *binary* outcomes/events. This will be enough to reason about quantum readout procedures, because they only ever produce bit values.

**Definition 7.4 (marginal probability distributions (two binary events)).** Consider a joint probability distribution  $\Pr [o_0 = s, o_1 = t]$  over two binary events  $s, t \in \{0, 1\}$ . Then, the *marginal probability for the first event* ( $o_0$ ) is

marginal probabilities

$$\Pr [o_0 = s] = \sum_{t=0}^1 \Pr [o_0 = s, o_t = t] = \Pr [o_0 = s, o_1 = 0] + \Pr [o_0 = s, o_1 = 1],$$

while the *marginal probability for the second* ( $o_1$ ) event is

$$\Pr [o_1 = t] = \sum_{s=0}^1 \Pr [o_0 = s, o_t = t] = \Pr [o_0 = 0, o_1 = t] + \Pr [o_0 = 1, o_1 = t].$$

Note that this definition readily extends to a joint probability distribution over more than two events. For three binary events ( $o_0 = s, o_1 = t, o_2 = u$ ) we

get

$$\begin{aligned}\Pr[o_0 = s] &= \sum_{t,u=0}^1 \Pr[o_0 = s, o_1 = t, o_2 = u], \\ \Pr[o_1 = t] &= \sum_{s,u=0}^1 \Pr[o_0 = s, o_1 = t, o_2 = u], \\ \Pr[o_2 = u] &= \sum_{s,t=0}^1 \Pr[o_0 = s, o_1 = t, o_2 = u].\end{aligned}$$

## 7.2.2 Conditional probability distributions

**Definition 7.5 (conditional probability distributions (two binary events)).** Consider a joint probability distribution  $\Pr[o_0 = s, o_1 = t]$  over two binary events  $s, t \in \{0, 1\}$ . Then, the two *conditional probabilities for the first event* ( $o_0$ ) are

conditional probabilities

$$\begin{aligned}\Pr[o_0 = s | o_1 = 0] &= \frac{\Pr[o_0 = s, o_1 = 0]}{\Pr[o_1 = 0]} \quad \text{for } s = 0, 1, \\ \Pr[o_0 = s | o_1 = 1] &= \frac{\Pr[o_0 = s, o_1 = 1]}{\Pr[o_1 = 1]} \quad \text{for } s = 0, 1.\end{aligned}$$

Likewise, the two *conditional probabilities for the second event* ( $o_1$ ) are

$$\begin{aligned}\Pr[o_1 = t | o_0 = 0] &= \frac{\Pr[o_0 = 0, o_1 = t]}{\Pr[o_0 = 0]} \quad \text{for } t = 0, 1, \\ \Pr[o_1 = t | o_0 = 1] &= \frac{\Pr[o_0 = 1, o_1 = t]}{\Pr[o_0 = 1]} \quad \text{for } t = 0, 1.\end{aligned}$$

(Care must be taken when the denominator approaches zero. This would mean that we condition on an event that can (almost) never happen).

Note that there are two conditional probability distributions for each event: one that assumes  $o_{0/1} = 0$  and one that assumes  $o_{0/1} = 1$ . Each of them is a valid probability distribution in its own right. Non-negativity follows directly from the construction. Normalization, on the other hand, follows from the fact that the relevant marginal distribution features in the denominator. For instance,

$$\sum_{s=0}^1 \Pr[o_0 = s | o_1 = 0] = \frac{\sum_{s=0}^1 \Pr[o_0 = s, o_1 = 0]}{\Pr[o_1 = 0]} = \frac{\Pr[o_1 = 0]}{\Pr[o_1 = 0]} = 1$$

and we obtain the same result for all other conditional probability distributions.

Similar to marginal probability distributions, Definition 7.5 also readily extends to more than two binary variables. However, the number of different possibilities grows very quickly! For three binary events ( $o_0 = s, o_1 = t, o_2 = u$ ), we can construct

$$\begin{aligned}\Pr[o_0 = s | o_1 = t, o_2 = u] &= \frac{\Pr[o_0 = s, o_1 = t, o_2 = u]}{\Pr[o_1 = t, o_2 = u]} \quad \text{for } s = 0, 1, \\ \Pr[o_1 = t | o_0 = s, o_2 = u] &= \frac{\Pr[o_0 = s, o_1 = t, o_2 = u]}{\Pr[o_0 = s, o_2 = u]} \quad \text{for } t = 0, 1, \\ \Pr[o_2 = u | o_0 = s, o_1 = t] &= \frac{\Pr[o_0 = s, o_1 = t, o_2 = u]}{\Pr[o_0 = s, o_1 = t]} \quad \text{for } u = 0, 1,\end{aligned}$$

(condition on two events), but also

$$\Pr [o_0 = s, o_1 = t | o_2 = u] = \frac{\Pr [o_0 = s, o_1 = t, o_2 = u]}{\Pr [o_2 = u]} \quad \text{for } s, t = 0, 1,$$

$$\Pr [o_0 = s, o_2 = u | o_1 = t] = \frac{\Pr [o_0 = s, o_1 = t, o_2 = u]}{\Pr [o_1 = t]} \quad \text{for } s, u = 0, 1,$$

$$\Pr [o_1 = t, o_2 = u | o_0 = s] = \frac{\Pr [o_0 = s, o_1 = t, o_2 = u]}{\Pr [o_0 = s]} \quad \text{for } t, u = 0, 1.$$

(condition on a single event).

**Exercise 7.6 (Bayes' theorem).** Prove the following statement known as *Bayes' theorem*:

Bayes' theorem

$$\Pr [o_1 = b | o_0 = a] = \frac{\Pr [o_0 = a | o_1 = b] \Pr [o_1 = b]}{\Pr [o_0 = a]}.$$

**Context:** Bayes' theorem highlights that the direction of correlations can be inverted. As such, it plays a pivotal role in statistics.

**Exercise 7.7 (Perfect correlations go both ways).** Suppose that we have a joint distribution of two binary variables that obey

$$\Pr [o_1 = t | o_0 = s] = \begin{cases} 1 & \text{if } s = t, \\ 0 & \text{else if } s \neq t. \end{cases} \quad (7.1)$$

In words: the value of  $o_0$  completely determines the value of  $o_1$  (perfect correlation). Use Bayes' rule to show that this also implies

$$\Pr [o_0 = s | o_1 = t] = \begin{cases} 1 & \text{if } s = t, \\ 0 & \text{else if } s \neq t. \end{cases}$$

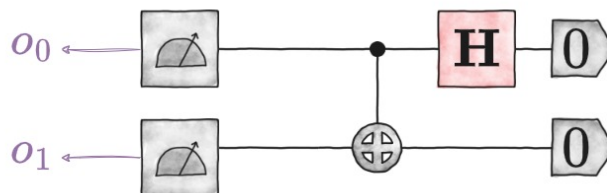
Is the converse direction also true? That is, does

$$\Pr [o_1 = t | o_0 = s] = \Pr [o_0 = s | o_1 = t]$$

necessarily imply perfect correlations in the sense of Eq. (7.1)?

### 7.2.3 Example 1: Bell state readout

Recall a basic Bell state preparation circuit, followed by reading out the two qubits involved:



We already know from Lecture 3 and Lecture 5 that this state (vector) corresponds to

$$|\psi_{\text{Bell}}\rangle = \mathbf{CNOT} (\mathbf{H} \otimes \mathbb{I}) |00\rangle = \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle).$$

And, for readout values  $o_0 = s$  and  $o_1 = t$  with  $s, t = 0, 1$ , we obtain

$$\Pr_{|\psi_{\text{Bell}}\rangle} [o_0 = s, o_1 = t] = \begin{cases} 1/2 & \text{if } s = t, \\ 0 & \text{else if } s \neq t. \end{cases} \quad (7.2)$$

Access to this joint probability distribution over a pair of events, allows us to compute both marginal and conditional probabilities. Let us start with the *marginal probabilities*. For the first outcome  $o_0$ , we obtain

$$\Pr_{|\psi_{\text{Bell}}\rangle} [o_0 = 0] = \Pr_{|\psi_{\text{Bell}}\rangle} [o_0 = 0, o_1 = 0] + \Pr_{|\psi_{\text{Bell}}\rangle} [o_0 = 0, o_1 = 1] = \frac{1}{2} + 0 = \frac{1}{2},$$

$$\Pr_{|\psi_{\text{Bell}}\rangle} [o_0 = 1] = \Pr_{|\psi_{\text{Bell}}\rangle} [o_0 = 1, o_1 = 0] + \Pr_{|\psi_{\text{Bell}}\rangle} [o_0 = 1, o_1 = 1] = 0 + \frac{1}{2} = \frac{1}{2}.$$

(Alternatively, we could have also deduced the second line from the first one:  $\Pr_{|\psi_{\text{Bell}}\rangle} [o_0 = 1] = 1 - \Pr_{|\psi_{\text{Bell}}\rangle} [o_0 = 0] = 1 - 1/2 = 1/2$ .) An almost identical computation reveals the same marginal probabilities for the second outcome  $o_1$ :

$$\Pr_{|\psi_{\text{Bell}}\rangle} [o_1 = 0] = \Pr_{|\psi_{\text{Bell}}\rangle} [o_0 = 0, o_1 = 0] + \Pr_{|\psi_{\text{Bell}}\rangle} [o_0 = 1, o_1 = 0] = \frac{1}{2} + 0 = \frac{1}{2},$$

$$\Pr_{|\psi_{\text{Bell}}\rangle} [o_1 = 1] = \Pr_{|\psi_{\text{Bell}}\rangle} [o_0 = 0, o_1 = 1] + \Pr_{|\psi_{\text{Bell}}\rangle} [o_0 = 1, o_1 = 1] = 0 + \frac{1}{2} = \frac{1}{2}.$$

We can put everything together into a single formula that succinctly captures the marginal probabilities of a Bell state readout procedure:

$$\Pr_{|\psi_{\text{Bell}}\rangle} [o_0 = s] = \Pr_{|\psi_{\text{Bell}}\rangle} [o_1 = s] = \frac{1}{2} \quad \text{for } s = 0, 1. \quad (7.3)$$

This display highlights two things: (i) the two marginal probabilities are identical and (ii) each marginal probability is equivalent to a uniformly random bit (think: coin toss).

Let us now move on to determining the *conditional probabilities*. We start with conditioning on  $o_1 = 0$ :

$$\Pr_{|\psi_{\text{Bell}}\rangle} [o_0 = 0 | o_1 = 0] = \frac{\Pr_{|\psi_{\text{Bell}}\rangle} [o_0 = 0, o_1 = 0]}{\Pr [o_1 = 0]} = \frac{1/2}{1/2} = 1,$$

where we have inserted Eq. (7.2) for the numerator and Eq. (7.3) for the denominator. In a similar fashion, we obtain

$$\Pr_{|\psi_{\text{Bell}}\rangle} [o_0 = 1 | o_1 = 0] = \frac{\Pr_{|\psi_{\text{Bell}}\rangle} [o_0 = 1, o_1 = 0]}{\Pr [o_1 = 0]} = \frac{0}{1/2} = 0.$$

This confirms that these two conditional probabilities indeed form a valid probability distribution. Both probabilities are non-negative and add up to one.

Conditioning on  $o_1 = 1$  instead, tells a similar story, but with reversed roles:

$$\begin{aligned}\Pr_{|\psi_{\text{Bell}}\rangle} [o_0 = 0 | o_1 = 1] &= \frac{\Pr_{|\psi_{\text{Bell}}\rangle} [o_0 = 0, o_1 = 1]}{\Pr [o_1 = 1]} = \frac{0}{1/2} = 0, \\ \Pr_{|\psi_{\text{Bell}}\rangle} [o_0 = 1 | o_1 = 1] &= \frac{\Pr_{|\psi_{\text{Bell}}\rangle} [o_0 = 1, o_1 = 1]}{\Pr [o_1 = 1]} = \frac{1/2}{1/2} = 1.\end{aligned}$$

This is again a valid probability distribution over a single binary outcome. In fact, we can combine both into a single formula:

$$\Pr_{|\psi_{\text{Bell}}\rangle} [o_0 = s | o_1 = t] = \begin{cases} 1 & \text{if } s = t, \\ 0 & \text{else if } s \neq t. \end{cases} \quad (7.4)$$

This display highlights another feature of Bell state readout: the first outcome bit is perfectly correlated with the second outcome bit:  $o_0 = o_1$ . It should not come as a surprise that this perfect correlation persists if we exchange the roles of the two outcome bits. We leave this as an instructive exercise.

**Exercise 7.8 (Bell readout probabilities conditioned on the first outcome bit  $o_0$ ).** Use Eq. (7.2) and Eq. (7.3) to derive the following conditional probability distributions:

$$\Pr_{|\psi_{\text{Bell}}\rangle} [o_1 = t | o_0 = s] = \begin{cases} 1 & \text{if } t = s, \\ 0 & \text{else if } t \neq s. \end{cases}$$

### 7.2.4 Example 2: Drawing straws

You and four of your friends have just had dinner and now have to decide who does the dishes. To make the decision fair all of you agree to decide by drawing straws. This means there are 5 straws of which 1 is considerably shorter than the other 4 straws, they are mixed in a hat or beanie and one person at a time is drawing one straw from the hat without looking and not putting it back after drawing it. The person drawing the shortest straw loses and has to do the dishes and the other ones all win.

This can be described as a joint probability distribution over 5 binary events  $o_0, o_1, o_2, o_3, o_4 \in \{0, 1\}$ —one for each straw in the hat, aka player of the game. We say that player  $k$  loses if they draw the short straw,  $o_k = 1$  and  $o_k = 0$  otherwise. The associated probability distribution then becomes

$$\Pr [o_0 = s, o_1 = t, o_2 = u, o_3 = v, o_4 = w] = \begin{cases} 1/5 & \text{if } s + t + u + v + w = 1, \\ 0 & \text{else.} \end{cases}$$

for  $s, t, u, v, w \in \{0, 1\}$ . Access to this distribution allows us to compute the marginal probabilities for  $o_0$  only:

$$\begin{aligned}\Pr [o_0 = 1] &= \sum_{t,u,v,w=0}^1 \Pr [o_0 = 1, o_1 = t, o_2 = u, o_3 = v, o_4 = w] \\ &= 1/5 + 0 + \dots + 0 = \frac{1}{5}.\end{aligned}$$



This readily allows us to conclude that  $\Pr [o_0 = 0] = 1 - \Pr [o_0 = 1] = 1 - 1/5 = 4/5$ . A computation of the other four marginal probabilities looks very similar and produces exactly the same results (why?). We can therefore succinctly write

$$\Pr [o_k = 1] = 1/5 \quad \text{and} \quad \Pr [o_k = 0] = 4/5 \quad \text{for all } k = 0, 1, 2, 3, 4.$$

This ensures that the drawing of the straws is fair: each player has the same odds of winning/losing.

However, all five players draw from the same hat. This introduces dependencies between the individual scores of the players involved. Conditional probabilities are the proper way to reason about these effects which become most pronounced if we look at the score of the last player's conditional on the score of all players before them. For instance,

$$\begin{aligned} \Pr [o_4 = 1 | o_3 = 0, o_2 = 0, o_1 = 0, o_0 = 0] &= \frac{\Pr [o_0 = 0, o_1 = 0, o_2 = 0, o_3 = 0, o_4 = 1]}{\Pr [o_0 = 0, o_1 = 0, o_2 = 0, o_3 = 0]} \\ &= \frac{\Pr [o_0 = 0, o_1 = 0, o_2 = 0, o_3 = 0, o_4 = 1]}{\Pr [o_0 = 0, o_1 = 0, o_2 = 0, o_3 = 0, o_4 = 0] + \Pr [o_0 = 0, o_1 = 0, o_2 = 0, o_3 = 0, o_4 = 1]} \\ &= \frac{1/5}{0 + 1/5} = 1. \end{aligned}$$

And, in a similar fashion, we can conclude

$$\Pr [o_4 = 0 | o_3 = 0, o_2 = 0, o_1 = 0, o_0 = 0] = 0.$$

Note that these conditional probabilities are actually deterministic: player five is guaranteed to lose if players one, two, three, and four all win. A converse of this observation is also true. Suppose, for concreteness, that player three loses, i.e.  $o_2 = 1$ . Then, player five has no chance of also losing. In formulas,

$$\begin{aligned} \Pr [o_4 = 1 | o_3 = 0, o_2 = 1, o_1 = 0, o_0 = 0] &= 0, \\ \Pr [o_4 = 0 | o_3 = 0, o_2 = 1, o_1 = 0, o_0 = 0] &= 1 \end{aligned}$$

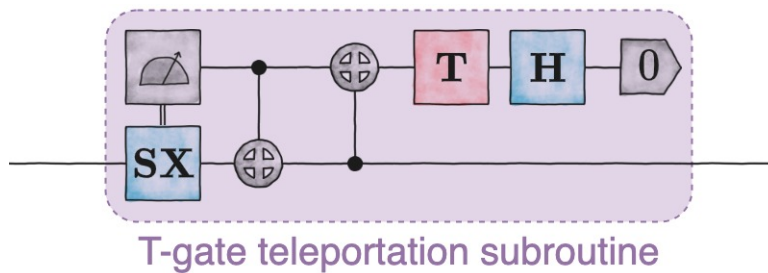
and we leave the actual derivation as a quick exercise. By now it should not come as a surprise that the same is true if any of the other players already lost. We can succinctly summarize our insights in the following display

$$\Pr \left[ o_4 = 1 \mid \sum_{k=0}^3 o_k = 0 \right] = 1 \quad \text{while} \quad \Pr \left[ o_4 = 1 \mid \sum_{k=0}^3 o_k = 1 \right] = 0.$$

In words: player five must lose ( $o_4 = 1$ ) if everybody else wins before and they must win ( $o_4 = 0$ ) if somebody else has already lost.

### 7.3 Quantum $T$ -gate teleportation

The concept of conditional probabilities is vital to analyze quantum subroutines that combine unitary (quantum) gates with partial qubit readout and conditional operations. Fig. 7.1 displays one such subroutine that plays a very prominent



**Figure 7.1** *T-gate teleportation*: This quantum subroutine effectively acts on a single qubit (lower line). Apart from a single  $T$ -gate (red), it only features Clifford operations ( $S$ ,  $X$  and  $CNOT$ ) and the readout of the top qubit. Interestingly this subroutine is equivalent to applying a  $T$ -gate on the second qubit, i.e.  $|\psi_{\text{out}}\rangle = T|\psi_{\text{in}}\rangle$ .

role in fault-tolerant quantum computation (we will talk about quantum error correction and fault tolerance in a future lecture). It also highlights that quantum computation is really different from conventional hardware design [GC99].

The circuit in Fig. 7.1 contains notation we haven't seen before. In particular, a double line emanates from the readout symbol and enters a quantum gate box. This depicts a *conditional gate application* that depends on the readout bit  $o_0$  we observe:

- (i) if  $o_0 = 0$ , we do nothing (i.e. apply the identity  $\mathbb{1}$  to the remaining qubit),
- (ii) else if  $o_0 = 1$ , we apply the gate  $SX$  to the remaining qubit.

The subroutine depicted in Fig. (7.1) takes a single qubit as input and also outputs a single qubit. It, therefore, corresponds to an effective single-qubit operation that we can execute if we have a quantum computer with (at least) two qubits. The main result of this section highlights that this effective operation is equivalent to one we already know.

**Theorem 7.9 (T-gate teleportation).** The effective single-qubit subroutine displayed in Fig. (7.1) is equivalent to applying a single-qubit  $T$ -gate:  $|\psi\rangle \mapsto T|\psi\rangle$  for every input  $|\psi\rangle$ .

Before moving on to a step-by-step analysis, it is worthwhile to suggestively rewrite part of the  $T$ -gate teleportation circuit. To this end, we define

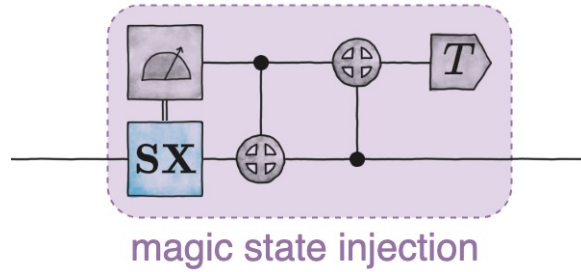
$$|T\rangle = TH|0\rangle = \frac{1}{\sqrt{2}} \left( |0\rangle + e^{i\pi/4}|1\rangle \right),$$

which is known as a *magic state*. Theorem 7.9 then tells us that we can use one such magic state to effectively apply a single  $T$  gate to another (arbitrary)

conditional gate application

magic state:  $|T\rangle = TH|0\rangle$

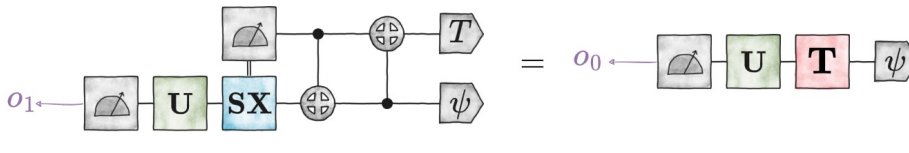
qubit. In pictures,



and this version of the protocol is known as *magic state injection*. It only features Clifford gates ( $S, H, CNOT$  and  $X = HS^2H$ ), as well as one single-qubit readout operation. Magic state injection does, however, convert access to one magic state  $|T\rangle$  in an effective application of a  $T$ -gate which is *not* a Clifford gate. This trick will become important once we discuss quantum error correction and fault-tolerant quantum computation.

magic state injection

Let us now move on to actually prove Theorem 7.9. Theorem 7.1 tells us that it is enough to show



for an *arbitrary* single-qubit unitary  $U$  and an *arbitrary* single-qubit state vector  $|\psi\rangle$ . The readout probabilities for the r.h.s. are now simply

$$\Pr_{UT|\psi} [o_0 = t] = |\langle t|UT|\psi\rangle|^2 \quad \text{for } t = 0, 1. \tag{7.5}$$

The other side requires quite a bit more work. But, we have all the necessary prerequisites to analyze it as well. Let us start with tracking the two-qubit state vector throughout the quantum circuit. We can write  $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$  with  $\alpha, \beta \in \mathbb{C}$  and  $|\alpha|^2 + |\beta|^2 = 1$  and obtain the following effective starting state:

$$\begin{aligned} |\varphi_0\rangle &= |T\rangle \otimes |\psi\rangle = \frac{1}{\sqrt{2}} \left( |0\rangle + e^{i\pi/4}|1\rangle \right) \otimes (\alpha|0\rangle + \beta|1\rangle) \\ &= \frac{\alpha}{\sqrt{2}}|00\rangle + \frac{\beta}{\sqrt{2}}|01\rangle + \frac{\alpha e^{i\pi/4}}{\sqrt{2}}|10\rangle + \frac{\beta e^{i\pi/4}}{\sqrt{2}}|11\rangle. \end{aligned}$$

Next, we apply two CNOT-gates with different control qubits. On a bit logic level, they act as

$$CNOT_{2 \rightarrow 1}|s, t\rangle = |s \oplus t, t\rangle \quad \text{and} \quad CNOT_{1 \rightarrow 2}|s, t\rangle = |s, t \oplus s\rangle$$

for  $s, t = 0, 1$  and we obtain

$$\begin{aligned} |\varphi_1\rangle &= CNOT_{2 \rightarrow 1}|\varphi_0\rangle = \frac{\alpha}{\sqrt{2}}|00\rangle + \frac{\beta}{\sqrt{2}}|11\rangle + \frac{\alpha e^{i\pi/4}}{\sqrt{2}}|10\rangle + \frac{\beta e^{i\pi/4}}{\sqrt{2}}|01\rangle \quad \text{and} \\ |\varphi_2\rangle &= CNOT_{1 \rightarrow 2}|\varphi_1\rangle = \frac{\alpha}{\sqrt{2}}|00\rangle + \frac{\beta}{\sqrt{2}}|10\rangle + \frac{\alpha e^{i\pi/4}}{\sqrt{2}}|11\rangle + \frac{\beta e^{i\pi/4}}{\sqrt{2}}|01\rangle. \end{aligned}$$

We can use elementary (state) vector and Kronecker operations to rewrite this state suggestively as

$$\begin{aligned}
|\varphi_2\rangle &= \frac{1}{\sqrt{2}}|0\rangle \otimes (\alpha|0\rangle + e^{i\pi/4}\beta|1\rangle) + \frac{e^{-i\pi/4}}{\sqrt{2}}|1\rangle \otimes (e^{i\pi/4}\beta|0\rangle + e^{i\pi/2}\alpha|1\rangle) \\
&= \frac{1}{\sqrt{2}}|0\rangle \otimes (\mathbf{T}(\alpha|0\rangle + \beta|1\rangle)) + \frac{e^{-i\pi/4}}{\sqrt{2}}|1\rangle \otimes (\mathbf{S}^\dagger(e^{i\pi/4}\beta|0\rangle + \alpha|1\rangle)) \\
&= \frac{1}{\sqrt{2}}|0\rangle \otimes (\mathbf{T}|\psi\rangle) + \frac{e^{-i\pi/4}}{\sqrt{2}}|1\rangle \otimes (\mathbf{S}^\dagger\mathbf{X}(\alpha|0\rangle + e^{i\pi/4}\beta|1\rangle)) \\
&= \frac{1}{\sqrt{2}}|0\rangle \otimes (\mathbf{T}|\psi\rangle) + \frac{e^{i\pi/4}}{\sqrt{2}}|1\rangle \otimes (\mathbf{S}^\dagger\mathbf{X}\mathbf{T}|\psi\rangle). \tag{7.6}
\end{aligned}$$

This reformulation of the state  $|\varphi_2\rangle$  already tells us quite a bit about the quantum logical configuration just before reading out the first qubit. It is a superposition of two distinct contributions: one for each classical readout value associated with the first qubit. If the readout value is  $o_0 = 0$ , we don't do anything to the remaining qubit and obtain

$$\begin{aligned}
\Pr_{(\mathbb{1}\otimes\mathbf{U})|\varphi_{\text{final}}\rangle} [o_0 = 0, o_1 = t] &= \Pr_{(\mathbb{1}\otimes\mathbf{U})|\varphi_2\rangle} [o_0 = 0, o_1 = t] \\
&= \left| \langle 0t | \left( \frac{1}{\sqrt{2}}|0\rangle \otimes (\mathbf{U}\mathbf{T}|\psi\rangle) + \frac{e^{-i\pi/4}}{\sqrt{2}}|1\rangle \otimes (\mathbf{U}\mathbf{S}^\dagger\mathbf{X}\mathbf{T}|\psi\rangle) \right) \right|^2 \\
&= \frac{1}{2} |\langle t | \mathbf{U}\mathbf{T} | \psi \rangle|^2.
\end{aligned}$$

Else if  $o_0 = 1$ , we do apply  $\mathbf{X}\mathbf{S}$  to the second qubit and obtain

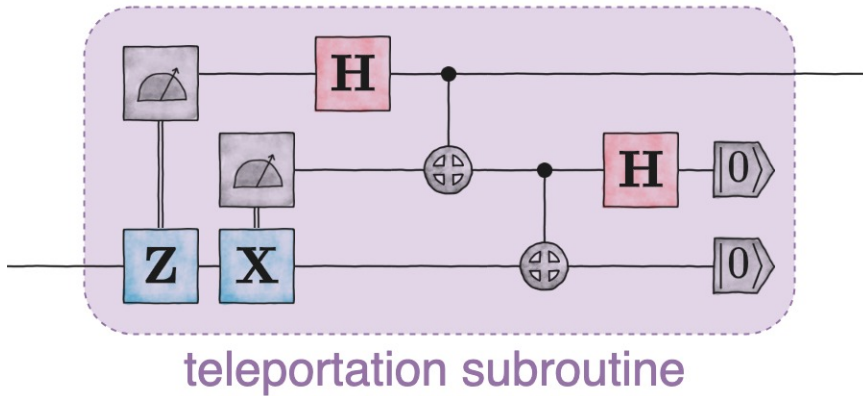
$$\begin{aligned}
\Pr_{(\mathbb{1}\otimes\mathbf{U})|\varphi_{\text{final}}\rangle} [o_0 = 1, o_1 = t] &= \Pr_{(\mathbb{1}\otimes\mathbf{U})\times(\mathbb{1}\otimes(\mathbf{X}\mathbf{S}))|\varphi_2\rangle} [o_0 = 1, o_1 = t] \\
&= \left| \langle 1t | \left( \frac{1}{\sqrt{2}}|0\rangle \otimes (\mathbf{U}\mathbf{X}\mathbf{S}\mathbf{T}|\psi\rangle) + \frac{e^{-i\pi/4}}{\sqrt{2}}|1\rangle \otimes (\mathbf{U}\mathbf{X}\mathbf{S}\mathbf{S}^\dagger\mathbf{X}\mathbf{T}|\psi\rangle) \right) \right|^2 \\
&= \frac{1}{2} |\langle t | \mathbf{U}\mathbf{X}\mathbf{S}\mathbf{S}^\dagger\mathbf{X}\mathbf{T} | \psi \rangle|^2 = \frac{1}{2} |\langle t | \mathbf{U}\mathbf{T} | \psi \rangle|^2.
\end{aligned}$$

These two computations nail down all four joint probabilities for  $s, t = 0, 1$ . Marginalization then implies

$$\begin{aligned}
\Pr_{(\mathbb{1}\otimes\mathbf{U})|\varphi_{\text{final}}\rangle} [o_0 = 0] &= \frac{1}{2} |\langle 0 | \mathbf{U}\mathbf{T} | \psi \rangle|^2 + \frac{1}{2} |\langle 1 | \mathbf{U}\mathbf{T} | \psi \rangle|^2 = \frac{1}{2}, \\
\Pr_{(\mathbb{1}\otimes\mathbf{U})|\varphi_{\text{final}}\rangle} [o_0 = 1] &= \frac{1}{2} |\langle 0 | \mathbf{U}\mathbf{T} | \psi \rangle|^2 + \frac{1}{2} |\langle 1 | \mathbf{U}\mathbf{T} | \psi \rangle|^2 = \frac{1}{2},
\end{aligned}$$

and the conditional outcome probabilities become

$$\begin{aligned}
\Pr_{(\mathbb{1}\otimes\mathbf{U})|\varphi_{\text{final}}\rangle} [o_1 = t | o_0 = 0] &= \frac{(1/2) |\langle t | \mathbf{U}\mathbf{T} | \psi \rangle|^2}{1/2} = |\langle t | \mathbf{U}\mathbf{T} | \psi \rangle|^2, \\
\Pr_{(\mathbb{1}\otimes\mathbf{U})|\varphi_{\text{final}}\rangle} [o_1 = t | o_0 = 1] &= \frac{(1/2) |\langle t | \mathbf{U}\mathbf{T} | \psi \rangle|^2}{1/2} = |\langle t | \mathbf{U}\mathbf{T} | \psi \rangle|^2,
\end{aligned}$$



**Figure 7.2** *Quantum teleportation subroutine:* An arbitrary quantum state enters this subroutine on the top right (first qubit wire). Once the protocol is completed, the *same* state leaves the lower left corner (third qubit wire), i.e.  $|\psi_{\text{out}}\rangle = |\psi_{\text{in}}\rangle$ . This is remarkable because there seems to be no quantum logical connection between the first and the last qubit.

In words: each conditional readout probability is equivalent to reading out the single-qubit state  $U^T|\psi\rangle$  instead. Also, this is valid for *any* input state  $|\psi\rangle$  and *any* subsequent unitary  $U$ . This allows us to conclude that Theorem 7.9 is valid as stated.

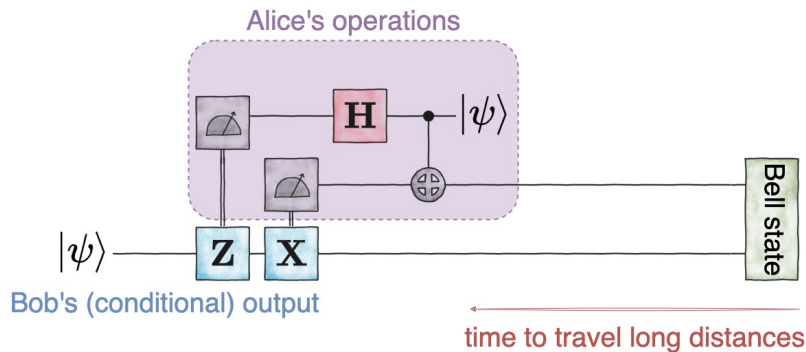
### 7.4 Quantum state teleportation

We are now ready for the main topic of today’s lecture: *quantum state teleportation*. The protocol dates back to 1993 [Ben+93], but we present it in a more modern framework – as another quantum circuit black box that has one incoming qubit wire and one outgoing qubit wire, see Fig. 7.2.

**Theorem 7.10 (correctness of quantum teleportation).** The teleportation subroutine in Fig. 7.2 acts like an effective one-qubit operation. Every input state on the top right gets exactly transferred to the bottom left, i.e.  $|\psi_{\text{out}}\rangle = |\psi_{\text{in}}\rangle$ .

quantum state teleportation

We provide a justification of the name ‘teleportation’ protocol in fig. 7.3. Let us now move on to provide a proof sketch for Theorem 7.10. Similar to  $T$ -gate teleportation, we build our arguments on the (sufficient) conditions on subroutine equivalence from Theorem 7.1. Concretely, we fix an arbitrary single-qubit state  $|\psi\rangle$ , an arbitrary single qubit unitary  $U$  and set out to show



justification of the term  
'teleportation'

**Figure 7.3** Interpretation as a quantum state 'teleportation' protocol: This interpretation stems from the observation that part of this quantum circuit can be prepared in advance. It is easy to recognize a Bell state preparation circuit at the beginning of qubits 2 and 3. Inserting it provides the following suggestive reformulation where we have artificially elongated the wires of qubit 2 and qubit 3. This highlights that the actual generation of the Bell state between qubit 2 and 3 can actually lie in the past, i.e. it occurred a long time before the actual state  $|\psi\rangle$  enters the picture. This extra time can, in principle, be spent on moving the two parts of the Bell state (qubit 2 and qubit 3) very far away from each other. The state teleportation subroutine then uses existing entanglement (Bell state) between two very distant locations to perfectly transmit a single-qubit state  $|\psi\rangle$  from one location (Alice's side, aka qubits 1 and 2) to a completely different location (Bob's side, aka qubit 3).

Note, however, that this protocol only works as intended if Alice communicates her readout values to Bob and Bob uses them to apply conditional quantum gates. If this is not the case, the protocol produces complete garbage. This subtle feature reconciles state teleportation with Einstein's postulate that no 'information'  $\psi$  can move faster than the speed of light.

The equality sign here indicates that the outcome probabilities associated with  $o_2 \in \{0, 1\}$  (left) and  $o \in \{0, 1\}$  (right) must be identical. We have also already streamlined this display a bit by incorporating the two-qubit Bell state. This readily allows us to write down the actual 3-qubit starting state. For  $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$  (with  $\alpha, \beta \in \mathbb{C}$  and  $|\alpha|^2 + |\beta|^2 = 1$ ), we obtain

$$\begin{aligned} |\varphi_0\rangle &= |\psi\rangle \otimes |\psi_{\text{Bell}}\rangle = (\alpha|0\rangle + \beta|1\rangle) \otimes \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\ &= \frac{\alpha}{\sqrt{2}}|000\rangle + \frac{\alpha}{\sqrt{2}}|011\rangle + \frac{\beta}{\sqrt{2}}|100\rangle + \frac{\beta}{\sqrt{2}}|111\rangle \end{aligned}$$

The CNOT between qubits 1 and 2 turns this state into

$$\begin{aligned} |\varphi_1\rangle &= (\mathbf{CNOT} \otimes \mathbb{I})|\varphi_0\rangle \\ &= \frac{\alpha}{\sqrt{2}}(\mathbf{CNOT}_{1 \rightarrow 2} \otimes \mathbb{I})|000\rangle + \frac{\alpha}{\sqrt{2}}(\mathbf{CNOT}_{1 \rightarrow 2} \otimes \mathbb{I})|011\rangle \\ &\quad + \frac{\beta}{\sqrt{2}}(\mathbf{CNOT}_{1 \rightarrow 2} \otimes \mathbb{I})|100\rangle + \frac{\beta}{\sqrt{2}}(\mathbf{CNOT}_{1 \rightarrow 2} \otimes \mathbb{I})|111\rangle \\ &= \frac{\alpha}{\sqrt{2}}|000\rangle + \frac{\alpha}{\sqrt{2}}|011\rangle + \frac{\beta}{\sqrt{2}}|110\rangle + \frac{\beta}{\sqrt{2}}|101\rangle \end{aligned}$$

and a Hadamard gate on the first qubit produces

$$\begin{aligned} |\varphi_2\rangle &= (\mathbf{H} \otimes \mathbb{I} \otimes \mathbb{I})|\varphi_1\rangle \\ &= \frac{\alpha}{\sqrt{2}}|+00\rangle + \frac{\alpha}{\sqrt{2}}|+11\rangle + \frac{\beta}{\sqrt{2}}|-10\rangle + \frac{\beta}{\sqrt{2}}|-01\rangle \\ &= \frac{\alpha}{2}|000\rangle + \frac{\alpha}{2}|100\rangle + \frac{\alpha}{2}|011\rangle + \frac{\alpha}{2}|111\rangle \\ &\quad + \frac{\beta}{2}|010\rangle - \frac{\beta}{2}|110\rangle + \frac{\beta}{2}|001\rangle - \frac{\beta}{2}|101\rangle. \end{aligned}$$

Written as is, this final 3-qubit state looks rather complicated. However, an interesting structure reveals itself if we start grouping the amplitudes in terms of the possible outcome bits for qubit 1 ( $o_0$ ) and qubit 2 ( $o_1$ ):

$$\begin{aligned} |\varphi_2\rangle &= \left(\frac{1}{2}|00\rangle\right) \otimes (\alpha|0\rangle + \beta|1\rangle) + \left(\frac{1}{2}|01\rangle\right) \otimes (\alpha|1\rangle + \beta|0\rangle) \\ &\quad + \left(\frac{1}{2}|10\rangle\right) \otimes (\alpha|0\rangle - \beta|1\rangle) + \left(\frac{1}{2}|11\rangle\right) \otimes (\alpha|1\rangle - \beta|0\rangle) \\ &= \left(\frac{1}{2}|00\rangle\right) \otimes (|\psi\rangle) + \left(\frac{1}{2}|0, 1\rangle\right) \otimes (\mathbf{X}|\psi\rangle) \\ &\quad + \left(\frac{1}{2}|10\rangle\right) \otimes (\mathbf{Z}|\psi\rangle) + \left(\frac{1}{2}|11\rangle\right) \otimes (\mathbf{XZ}|\psi\rangle), \end{aligned}$$

where we have used  $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ ,  $\mathbf{X}|\psi\rangle = \alpha|1\rangle + \beta|0\rangle$ ,  $\mathbf{Z}|\psi\rangle = \alpha|0\rangle - \beta|1\rangle$  and  $\mathbf{XZ}|\psi\rangle = \alpha|1\rangle - \beta|0\rangle$ . This regrouping tells an interesting story that comes in four parts:

- (i) If  $o_0 = 0, o_1 = 0$ , the third qubit must be in the state  $|\psi\rangle$ . Conditioned on these two readout outcomes, the protocol works perfectly and exactly transmits  $|\psi\rangle$  from the first qubit wire to the third one.
- (ii) Else if  $o_0 = 0, o_1 = 1$ , the third qubit must be in the state  $X|\psi\rangle$ . This is not a perfect state transmission from qubit one to qubit three, but the superfluous bit flip  $X$  can be undone. Conditioned on  $o_1 = 1$ , we apply an additional  $X$ -gate to qubit three to also recover  $X \times X|\psi\rangle = |\psi\rangle$  perfectly.
- (iii) Else if  $o_0 = 1, o_1 = 0$ , the third qubit must be in the state  $Z|\psi\rangle$ . This is not a perfect state transmission from qubit one to qubit three, but the superfluous sign flip  $Z$  can be undone. Conditioned on  $o_0 = 1$ , we apply an additional  $Z$ -gate to qubit three to also recover  $Z \times Z|\psi\rangle = |\psi\rangle$  perfectly.
- (iv) Else if  $o_0 = 1, o_1 = 1$ , the third qubit must be in the state  $XZ|\psi\rangle$ . This is essentially a combination of cases (ii) and (iii). Conditional application of both  $Z$  (because  $o_0 = 1$ ) and  $X$  (because  $o_1 = 1$ ), however, recovers the state perfectly. Provided that we apply these gates in the correct order. Doing  $X$  first (further right) and  $Z$  second (further left) ensures that the resulting state is  $Z \times X \times X \times Z|\psi\rangle = Z \times Z|\psi\rangle = |\psi\rangle$ .

Understanding these four cases is enough to complete a rigorous proof of Theorem 7.10. We have just shown that the conditional applications of  $X$  (associated with  $o_0 = 1$ ) and  $Z$  (associated with  $o_1 = 1$ ) produce a teleportation output that is always exactly equal to the teleportation input state  $|\psi\rangle$ . The final unitary  $U$  in Eq. (7.7) turns this state into  $U|\psi\rangle$  just before the final readout. This readout procedure is therefore equivalent to the right-hand side (simply prepare  $U|\psi\rangle$  and perform the readout).

Note, however, that our analysis above is merely a proof sketch and not yet a complete proof. Turning it into one requires a proper treatment via conditional readout probabilities that is similar to Section 7.3. We leave it as an instructive exercise that may be very relevant for the written exam.

**Exercise 7.11 (complete proof of Theorem 7.10).** See Prob. 7.16 for a detailed outline.



## Problems

**Problem 7.12 (Proof of Theorem 7.1).** Consider two single-qubit gate matrices  $\mathbf{A}$ ,  $\mathbf{B}$  (unitaries). Show that they must be equivalent (i.e.  $\mathbf{B} = e^{i\varphi}\mathbf{A}$  for some  $\varphi \in [0, 2\pi)$ ) if the following equality is true for all input states  $|\psi\rangle$  and all subsequent unitary gates  $\mathbf{U}$ :

$$\Pr_{\mathbf{UB}|\psi\rangle} [o = s] = \Pr_{\mathbf{UA}|\psi\rangle} [o = s] \quad \text{for } s = 0, 1.$$

**Challenging bonus question:** is it really necessary to consider all possible input states, as well as all possible unitaries?

**Problem 7.13 (Bayes' theorem).** Prove the following statement known as *Bayes' theorem*:

$$\Pr [o_1 = b | o_0 = a] = \frac{\Pr [o_0 = a | o_1 = b] \Pr [o_1 = b]}{\Pr [o_0 = a]}.$$

**Context:** Bayes' theorem highlights that the direction of correlations can be inverted. As such, it plays a pivotal role in statistics.

**Problem 7.14 (Drawing straws revisited).** Recall the drawing straws scenario, Ex. 7.2.4. What happens if everyone were to put back in the hat their straw after their turn? What would the probability of winning or losing be? Would it change after every turn? Justify your findings with the mathematical formulas developed in this lecture.

**Problem 7.15 (Perfect correlations go both ways).** Suppose that we have a joint distribution of two binary variables that obey

$$\Pr [o_1 = t | o_0 = s] = \begin{cases} 1 & \text{if } s = t, \\ 0 & \text{else if } s \neq t. \end{cases} \quad (7.8)$$

In words: the value of  $o_0$  completely determines the value of  $o_1$  (perfect correlation). Use Bayes' rule to show that this also implies

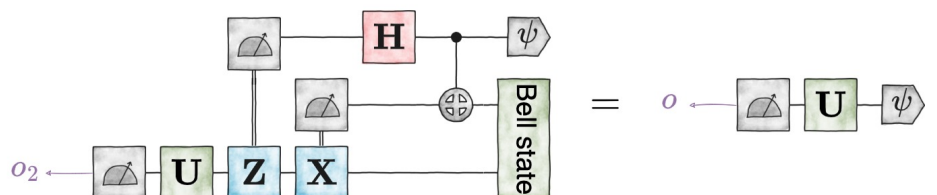
$$\Pr [o_0 = s | o_1 = t] = \begin{cases} 1 & \text{if } s = t, \\ 0 & \text{else if } s \neq t. \end{cases}$$

Is the converse direction also true? That is, does

$$\Pr [o_1 = t | o_0 = s] = \Pr [o_0 = s | o_1 = t]$$

necessarily imply perfect correlations in the sense of Eq. (7.8)?

**Problem 7.16 (proof of correctness for quantum state teleportation).** Consider the following two quantum circuits



where  $\mathbf{U}$  is an arbitrary single-qubit gate and  $|\psi\rangle$  is an arbitrary single-qubit input state (vector).

- 1 Write down the readout probabilities of the left-hand-circuit, i.e.

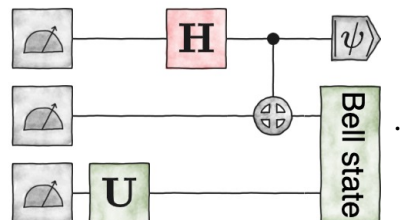
$$\Pr_{\mathbf{U}|\psi\rangle} [o = u] \quad \text{for } u = 0, 1.$$

- 2 Compute all joint readout probabilities of the final right-hand side state:  $\Pr_{|\varphi_{\text{final}}\rangle} [o_0 = s, o_1 = t, o_2 = u]$  for  $s, t, u = 0, 1$ .
- 3 Use your result from 2 to derive the marginal probabilities for readout bits one and two, i.e.  $\Pr_{|\varphi_{\text{final}}\rangle} [o_0 = s, o_1 = t]$  for  $s, t = 0, 1$ .
- 4 Combine your results from 2 and 3 to compute *all* conditional probabilities  $\Pr_{|\varphi_{\text{final}}\rangle} [o_2 = u | o_0 = s, o_1 = t]$ . Conclude that

$$\Pr_{|\varphi_{\text{final}}\rangle} [o_2 = u | o_0 = s, o_1 = t] = \Pr_{\mathbf{U}|\psi\rangle} [o = u] \quad \text{for } u = 0, 1,$$

regardless of the bit values for  $o_0$  and  $o_1$ . In other words: the state teleportation always operates as intended!

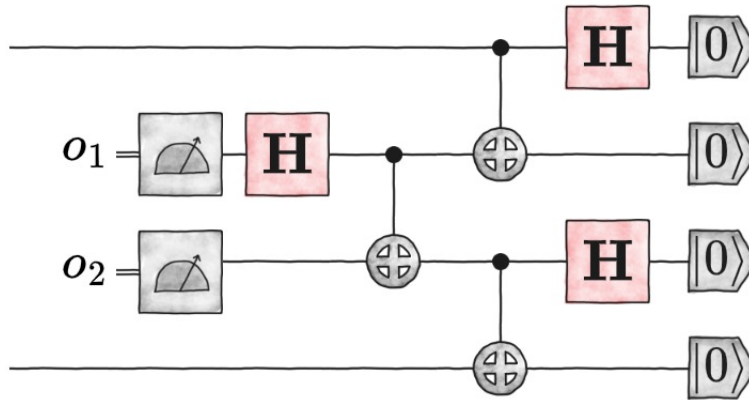
- 5 Suppose that Bob becomes impatient and performs a readout on his qubit before receiving the readout values of Alice (and before performing the conditional corrections). Then, the full teleportation protocol is cut short and effectively becomes



Compute the marginal probability distribution for Bob's readout of the third qubit:  $\Pr_{|\varphi_{\text{final}}\rangle} [o_2 = u]$  for  $u = 0, 1$ . Argue that this readout probability distribution does not contain *any* information about Alice's input state  $|\psi\rangle$  whatsoever.

**Context:** this observation resolves an apparent conflict between quantum state teleportation and the widespread belief that 'nothing' can propagate faster than the speed of light. According to the rules of quantum computing, the teleportation of  $|\psi\rangle$  from qubit one to qubit three happens instantly – regardless of the distance spanned by the initial Bell state. However, the readout values on Alice's side ( $o_0$  and  $o_1$ ) do affect the teleported state in a very particular fashion. If not properly undone, this 'washes out' all information about  $|\psi\rangle$ . In other words: the teleported state is useless for Bob until he receives Alice's readout values for  $o_0$  and  $o_1$ . This information, however, is classical and can only travel at the speed of light.

**Problem 7.17 (quantum repeaters).** Consider the following quantum circuit that involves 4 qubits and two partial measurements on qubit 2 and 3:



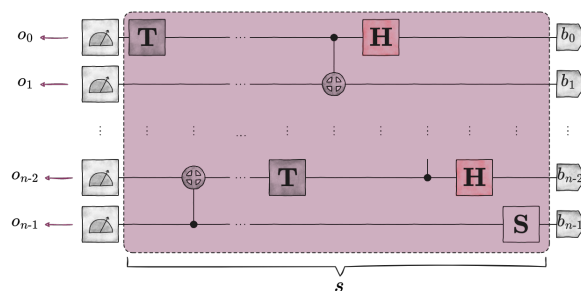
- 1 Compute the two-qubit output state  $\rho_{\text{out}}(o_0, o_0)$  for the special case where  $o_0, o_1 = 0$ . Can you recognize it?
- 2 What is the probability of obtaining  $o_0 = o_1 = 0$  when performing the partial measurement?
- 3 Argue that this circuit actually encompasses a quantum repeater for spreading entanglement across larger distances. But, the way we have set it up is probabilistic. The entanglement exchange protocol only works with a certain success probability (which one?).
- 4 **Optional:** do a full analysis that applies to all possible measurement outcomes  $o_0, o_1 \in \{0, 1\}$ . Can you correct the protocol (using  $o_0$  and  $o_1$ ) such that it is guaranteed to work in a deterministic fashion?

## 8. General $n$ -qubit architectures

Date: 29 November 2023

### 8.1 General $n$ -qubit architectures

Today, we make a substantial jump: we transition from few-qubit architectures (1, 2 or 3 qubits) to large-scale architectures that contain  $n \gg 1$  qubits. Conceptually, we are well-prepared for this increase in complexity. We already know all the relevant concepts, like qubit initialization at the beginning and qubit readout at the very end. The quantum circuits in between are also combinations of elementary 1- and 2-qubit gates that we already know. We



**Figure 8.1** A general  $n$ -qubit architecture has  $n$  input bits  $b_0, \dots, b_{n-1}$ , a central block of quantum logic and a final readout stage that recovers  $n$  bits  $o_0, \dots, o_{n-1}$ . the central block is solely comprised of elementary quantum gates, e.g. **H**, **T**, **S** and **CNOT**. The number  $s$  of elementary quantum gates is called the *size* of the quantum circuit.

#### Agenda:

- 1 general  $n$ -qubit circuits
- 2 strong & weak (classical) simulation
- 3 implementing classical circuits on quantum hardware
- 4 synopsis

$n$ -qubit architecture:  $n$  input qubits,  $n$  readout bits and a combination of elementary quantum gates inbetween

refer to Fig. 8.1 for a visualization<sup>1</sup>.

We are going to explore the fundamental possibilities of such large quantum architectures. We will see that simulation on classical hardware is possible, but does come with an exponential overhead (in  $n$ ). Conversely, we can use a hypothetical quantum architecture to execute *any* classical Boolean circuit with only linear overhead. To paraphrase: quantum architectures are never much worse than conventional hardware. But, conversely, building them might unlock exponential improvements in terms of (conventional) running time and/or circuit size. This window of opportunity is exploited by seminal quantum circuit constructions, like the ones by Shor for integer factorization and discrete logarithm. This, however, will be the topic of a future lecture.

## 8.2 Classical description of $n$ -qubit architectures

We have already introduced quite a bit of formalism that allows us to reason about quantum circuit architectures.

### 8.2.1 State vector representation of general $n$ -qubit states

Recall that a single qubit wire contains two complex-valued degrees of freedom. We can capture both of them with a  $2^1$ -dimensional state vector of amplitudes:

$$|\psi\rangle = \sum_{b_0=0}^1 \psi_{b_0} |b_0\rangle = \boldsymbol{\psi} = \begin{pmatrix} \psi_0 \\ \psi_1 \end{pmatrix} \in \mathbb{C}^2 = \mathbb{C}^{2^1}, \quad (8.1)$$

where we must also meet the normalization condition  $\|\boldsymbol{\psi}\|^2 = \langle\psi|\psi\rangle = \sum_{b_0=0}^1 |\psi_{b_0}|^2 = |\psi_0|^2 + |\psi_1|^2 = 1$ . The two most basic examples encode a single logical 0 and a single logical 1, respectively:

$$|0\rangle = \mathbf{e}_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{and} \quad |1\rangle = \mathbf{e}_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \quad (8.2)$$

We can use these basic building blocks to construct state vectors of more complex bit configurations. The case  $n = 2$ , for instance, featured prominently in Lecture 4. There are in total  $4 = 2^2$  bit strings of length  $n = 2$ . And we can use the Kronecker product to construct all of them from the basic state vectors in Eq. (8.2):

$$\begin{aligned} |00\rangle &= |0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = (1 \ 0 \ 0 \ 0)^T = \mathbf{e}_0 \in \mathbb{C}^4 = \mathbb{C}^{2^2}, \\ |01\rangle &= |0\rangle \otimes |1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = (0 \ 1 \ 0 \ 0)^T = \mathbf{e}_1 \in \mathbb{C}^4 = \mathbb{C}^{2^2}, \\ |10\rangle &= |1\rangle \otimes |0\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = (0 \ 0 \ 1 \ 0)^T = \mathbf{e}_2 \in \mathbb{C}^4 = \mathbb{C}^{2^2}, \\ |11\rangle &= |1\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = (0 \ 0 \ 0 \ 1)^T = \mathbf{e}_3 \in \mathbb{C}^4 = \mathbb{C}^{2^2}, \end{aligned}$$

<sup>1</sup>A  $n$ -qubit architecture can also feature partial readout and conditional gate applications (see Lecture 7). We disregard this option here for the sake of keeping things simple.

where we have written down the final expression as a row vector (transposition) to save a bit of paper space. Note that this identification between bitstrings (left) and standard basis vectors (right) is even nicer than one might expect: the 2-bit string on the left corresponds to a bit encoding  $\lfloor l \rfloor$  of the standard basis vector index. For  $l$  between 0 and  $3 = 2^2 - 1$ ,

$$\mathbf{e}_l = |\lfloor l \rfloor\rangle \quad \text{or equivalently} \quad |b_0 b_1\rangle = \mathbf{e}_{b_0+2 \times b_1}. \quad (8.3)$$

A general 2-qubit state vector can always be decomposed into a superposition over all these  $2^2$  bit configurations:

$$|\psi\rangle = \sum_{b_0, b_1=0}^1 \psi_{b_0 b_1} |b_0 b_1\rangle = \boldsymbol{\psi} = \begin{pmatrix} \psi_{00} \\ \psi_{01} \\ \psi_{10} \\ \psi_{11} \end{pmatrix} \in \mathbb{C}^4 = \mathbb{C}^{2^2}. \quad (8.4)$$

This state vector must obey the following normalization condition:

$$\|\boldsymbol{\psi}\|^2 = \langle \boldsymbol{\psi} | \boldsymbol{\psi} \rangle = \sum_{b_0, b_1=0}^1 |\psi_{b_0 b_1}|^2 = 1.$$

Comparing Eq. (8.4) with Eq. (8.1) already provides us with a blueprint on how to scale up these vector representations further. State vectors of  $n$ -bit strings can be constructed by forming  $n$ -fold Kronecker products of the basic bit configurations (8.2) involved. Each Kronecker product doubles the number of dimensions involved. So, we end up with state vectors that live in a  $2^n$ -dimensional complex space  $\mathbb{C}^{2^n}$ :

$n$ -qubit bitstring configurations

$$\begin{aligned} |0 \cdots 00\rangle &= |0\rangle \otimes \cdots \otimes |0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \cdots \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ &= (1 \ 0 \ \cdots \ 0 \ 0)^T = \mathbf{e}_0 \in \mathbb{C}^{2^n}, \\ |0 \cdots 01\rangle &= |0\rangle \otimes \cdots \otimes |0\rangle \otimes |1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \cdots \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ &= (0 \ 1 \ 0 \ \cdots \ 0)^T = \mathbf{e}_1 \in \mathbb{C}^{2^n}, \\ &\vdots \\ |01 \cdots 1\rangle &= |0\rangle \otimes |1\rangle \otimes \cdots \otimes |1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \cdots \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ &= (0 \ \cdots \ 0 \ 1 \ 0) = \mathbf{e}_{2^{n-2}} \in \mathbb{C}^{2^n}, \\ |11 \cdots 1\rangle &= |1\rangle \otimes |1\rangle \otimes \cdots \otimes |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \cdots \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ &= (0 \ 0 \ \cdots \ 0 \ 1) = \mathbf{e}_{2^n-1} \in \mathbb{C}^{2^n}. \end{aligned}$$

More succinctly, we obtain the following generalization of Eq. (8.3) to  $n$ -bit strings and numbers  $l$  between 0 and  $2^n - 1$ :

$$\mathbf{e}_l = |\lfloor l \rfloor\rangle \quad \text{or equivalently} \quad |b_0 \cdots b_{n-1}\rangle = \mathbf{e}_{b_0 \times 2^{n-1} + b_1 \times 2^{n-2} + \cdots + b_{n-1}}. \quad (8.5)$$

A general  $n$ -qubit state vector can form a superposition of all these  $2^n$   $n$ -bit configurations:

$$|\psi\rangle = \sum_{b_0, \dots, b_{n-1}=0}^1 \psi_{b_0 \dots b_{n-1}} |b_0 \dots b_{n-1}\rangle = \begin{pmatrix} \psi_{0 \dots 00} \\ \psi_{0 \dots 01} \\ \vdots \\ \psi_{01 \dots 1} \\ \psi_{11 \dots 1} \end{pmatrix} \in \mathbb{C}^{2^n}. \quad (8.6)$$

$n$ -qubit state vector has  $2^n$  (complex-valued) amplitudes

Each amplitude  $\psi_{b_0 \dots b_{n-1}}$  can be a complex number, but together they must obey the following normalization condition:

$$\|\boldsymbol{\psi}\|^2 = \langle \boldsymbol{\psi} | \boldsymbol{\psi} \rangle = \sum_{b_0, \dots, b_{n-1}=0}^1 |\psi_{b_0 \dots b_{n-1}}|^2 = 1.$$

Note that this sum ranges over all  $2^n$  different amplitudes that feature in Eq. (8.6).

**Exercise 8.1** The following representation of a general state puts more emphasis on the exponential amount of different superpositions that are allowed:

$$\boldsymbol{\psi} = \sum_{l=0}^{2^n-1} \psi_l \mathbf{e}_l \in \mathbb{C}^{2^n} \quad \text{or} \quad |\psi\rangle = \sum_{l=0}^{2^n-1} \psi_l |l\rangle.$$

Show that both formulas are equivalent to Eq. (8.6).

### 8.2.2 Circuit matrix representation of general $n$ -qubit circuits

By now, we are already acquainted with quantum gate matrices. They are reversible extensions of reversible binary logic. Prominent single-qubit gate matrices are

$$\mathbf{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad \text{and} \quad \mathbf{S} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix},$$

as well as the  $T$ -gate:

$$\mathbf{T} = \begin{pmatrix} 1 & 0 \\ 0 & \exp(i\pi/4) \end{pmatrix}.$$

In addition, we have also seen two important 2-qubit gates that allow us to do conditional quantum logic. The two possible CNOT gates are,

$$\mathbf{CNOT}_{1 \rightarrow 2} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}}_{|0\rangle\langle 0|} \otimes \mathbb{I} + \underbrace{\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}}_{|1\rangle\langle 1|} \otimes \mathbf{X},$$

$$\mathbf{CNOT}_{2 \rightarrow 1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} = \mathbb{I} \otimes \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}}_{|0\rangle\langle 0|} + \mathbf{X} \otimes \underbrace{\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}}_{|1\rangle\langle 1|},$$

where the subscript denotes the ‘flow of information’ from control to target.

Much like in conventional Boolean circuitry, we can now take these elementary gates and combine them to construct nontrivial functionalities on  $n$  qubits. Such a quantum circuit has to map a  $2^n$ -dimensional state vector  $|\psi_{in}\rangle$  ( $n$ -qubit input state) to another  $2^n$ -dimensional state vector  $|\psi_{out}\rangle$  ( $n$ -qubit output state). This action is described by a  $2^n \times 2^n$  circuit matrix  $\mathbf{U}$ :

$$|\psi_{out}\rangle = \mathbf{U}|\psi_{in}\rangle \quad \text{for all } |\psi_{in}\rangle \in \mathbb{C}^{2^n}. \quad (8.7)$$

$n$ -qubit circuit fully described by a  $2^n \times 2^n$  circuit matrix

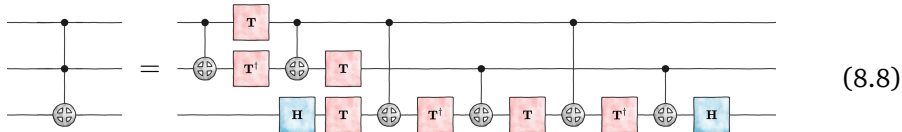
The matrix  $\mathbf{U}$  depends on the elementary gates involved, as well as their location within the circuit. Similar to the 1- and 2-qubit case, we can use products to construct this final matrix out of matrix representations of the individual constituents:

- 1 Parallel gate applications use the Kronecker product ‘ $\otimes$ ’ of the individual gate matrices involved (including  $\mathbb{I} \in \mathbb{C}^{2 \times 2}$  for qubit wires where nothing happens). For  $n$  qubits, this always produces a single  $2^n \times 2^n$  matrix for each gate layer
- 2 Combining sequential gate layers uses the matrix product ‘ $\times$ ’ of  $2^n \times 2^n$  gate layer matrices.

Kronecker (parallel) and matrix (sequential) products turn elementary gate matrices into full  $2^n \times 2^n$  circuit matrix

This general construction is best explained by means of an example.

**Example 8.2 (3-qubit Toffoli gate).** Consider the following combination of  $\mathbf{H}$ ,  $\mathbf{T}$ ,  $\mathbf{T}^\dagger = \mathbf{T}^7 = \text{diag}(0, \exp(-i\pi/4))$  and  $\mathbf{CNOT}$  that act on three qubit wires:



The r.h.s displays a sequential combination of 12 gate layers. We can use the Kronecker product to compute a  $2^3 \times 2^3$  matrix representation for each of them:

$$\begin{aligned} \mathbf{C}_0 &= \mathbb{I} \otimes \mathbb{I} \otimes \mathbf{H}, & \mathbf{C}_1 &= \mathbb{I} \otimes \mathbf{CNOT}_{1 \rightarrow 2}, & \mathbf{C}_2 &= \mathbb{I} \otimes \mathbb{I} \otimes \mathbf{T}^\dagger, \\ \mathbf{C}_3 &= |0\rangle\langle 0| \otimes \mathbb{I} \otimes \mathbb{I} + |1\rangle\langle 1| \otimes \mathbb{I} \otimes \mathbf{X} \quad (\text{why?}), \\ \mathbf{C}_4 &= \mathbb{I} \otimes \mathbb{I} \otimes \mathbf{T}, & \mathbf{C}_5 &= \mathbb{I} \otimes \mathbf{CNOT}_{1 \rightarrow 2}, & \mathbf{C}_6 &= \mathbb{I} \otimes \mathbb{I} \otimes \mathbf{T}^\dagger, \\ \mathbf{C}_7 &= \mathbf{C}_3, & \mathbf{C}_8 &= \mathbb{I} \otimes \mathbf{T} \otimes \mathbf{T}, & \mathbf{C}_9 &= \mathbf{CNOT}_{1 \rightarrow 2} \otimes \mathbf{H}, \\ \mathbf{C}_{10} &= \mathbf{T} \otimes \mathbf{T}^\dagger \otimes \mathbb{I} \quad \text{and} & \mathbf{C}_{11} &= \mathbf{CNOT}_{1 \rightarrow 2} \otimes \mathbb{I}. \end{aligned}$$

The final gate matrix then corresponds to the (ordered) matrix product of the 12 matrices that describe the individual layers:

$$\mathbf{U} = \mathbf{C}_{11} \times \mathbf{C}_{10} \times \mathbf{C}_9 \times \mathbf{C}_8 \times \mathbf{C}_7 \times \mathbf{C}_6 \times \mathbf{C}_5 \times \mathbf{C}_4 \times \mathbf{C}_3 \times \mathbf{C}_2 \times \mathbf{C}_1 \times \mathbf{C}_0. \quad (8.9)$$



Proper execution of this construction produces the following simple expression for the final unitary matrix:

$$CCNOT = \mathbf{U} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}. \quad (8.10)$$

3-qubit Toffoli gate (two-fold controlled-NOT)

This is the truth table of a two-fold controlled bitflip: the third truth value is flipped if the first two bits are equal to 1. Otherwise, nothing happens. The short-hand notation on the lhs of Eq. (8.8) succinctly captures this functionality.

■

**Exercise 8.3** Derive the matrix representation (8.10) yourself by completing the argument from the example: (i) form all 12 layer matrices  $\mathbf{C}_0, \dots, \mathbf{C}_{11}$  using a Kronecker product construction, (ii) combine these 12 layers sequentially by computing the matrix product in Eq. (8.9). **Hint:** don't do this by hand. Instead, write a piece of code that does it for you. Computers are good at this type of linear algebra.

Kronecker and matrix products have another nice feature: they play nicely with unitary matrices<sup>2</sup>.

**Lemma 8.4 (Kronecker and matrix product respect unitary structure).** The Kronecker product of two (or more) unitary matrices is again a unitary matrix. Likewise, the matrix product of two (or more) unitary matrices is again a unitary matrix.

The proof readily follows from the following appealing features of Kronecker and matrix products:  $(\mathbf{A} \otimes \mathbf{B})^\dagger = \mathbf{A}^\dagger \otimes \mathbf{B}^\dagger$ ,  $(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD})$  and, finally,  $(\mathbf{U} \times \mathbf{V})^\dagger = \mathbf{V}^\dagger \times \mathbf{U}^\dagger$ . We leave it as an instructive exercise in (multi-)linear algebra.

**Exercise 8.5** Prove Lemma 8.4.

The following proposition is now an immediate consequence of Lemma 8.4 and the way we construct matrix representations of general  $n$ -qubit circuits.

**Proposition 8.6 (circuit matrix).** The functionality of every  $n$ -qubit circuit is completely described by a  $2^n \times 2^n$  circuit matrix  $\mathbf{U}$ . This matrix is unitary and Kronecker + matrix products allow us to construct it from the circuit diagram.

every  $n$ -qubit circuit is fully captured by a unitary  $2^n \times 2^n$  matrix

This statement tells us that every  $n$ -qubit circuit is fully characterized by a  $2^n \times 2^n$  unitary matrix  $\mathbf{U}$ . Remarkably, the converse is also true:

<sup>2</sup>Recall that a  $D \times D$  matrix  $\mathbf{U}$  is unitary if  $\mathbf{U}^\dagger \mathbf{U} = \mathbf{U} \mathbf{U}^\dagger = \mathbb{I}$ , where  $\dagger$  denotes adjungation (transposition and complex conjugation) and  $\mathbb{I}$  is the  $D \times D$  identity matrix with ones on the main diagonal and zeroes everywhere else.

**Theorem 8.7 (Solovay-Kitaev ( $n$ -qubit case)).** Every unitary  $2^n \times 2^n$  matrix  $U$  can be approximated to arbitrary precision by a  $n$ -qubit quantum circuit that is solely comprised of  $H$ ,  $S$ ,  $CNOT$  (Clifford) and  $T$ -gates.

every  $2^n \times 2^n$  unitary matrix admits a  $n$ -qubit circuit approx. (Solovay-Kitaev)

This is the quantum generalization of a seminal result in logical circuitry: Every logical function  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$  can be represented as a logical circuit that is comprised solely of  $\neg$ ,  $\vee$  and  $\wedge$ . We don't have the time, nor the necessary background, to prove this powerful result. We emphasize instead, that we now start to feel the sheer size of quantum state space. A  $2^n \times 2^n$  unitary has exactly  $(2^n)^2 = 4^n$  real-valued degrees of freedom (why?). And, we need a comparable number of quantum gates to address all of them simultaneously. In other words: almost all possible  $n$ -qubit unitaries need exponentially many quantum gates (in  $n$ ) to approximate. This is a straightforward quantum generalization of Shannon's famous result that almost all  $n$ -bit logical functions require exponentially many logical operations ( $\neg$ ,  $\wedge$ ,  $\vee$ ) to realize.

### 8.2.3 Classical simulation of $n$ -qubit logic and readout

We now have all the ingredients in place to present a classical strategy that fully simulates the execution of a quantum circuit on a  $n$ -qubit architecture, like the one presented in Fig. 8.1. The key idea is to use  $2^n$ -dimensional state vectors to keep track of the quantum logic at each step of the quantum circuit. We start this procedure by forming a state vector representation of the  $n$ -qubit initialization:

$$|b_0 \cdots b_{n-1}\rangle = \mathbf{e}_{b_0 + 2 \times b_1 + \cdots + 2^{n-1} \times b_{n-1}} \in \mathbb{C}^{2^n}. \quad (8.11)$$

Now, let  $s$  be the *size* of the quantum circuit, i.e. the total number of nontrivial gates. Then, we can re-express the circuit as a sequential combination of  $s$  layers, where each layer contains exactly one nontrivial gate:

$$U = C_{s-1} \times \cdots \times C_1.$$

The individual layer matrices must be Kronecker products of exactly one nontrivial gate matrix with only identity matrices ( $\mathbb{I}$ ). For a single-qubit gate  $V$  at qubit wire  $a \in \{0, \dots, n-1\}$ , we get

$$C_k = \underbrace{\mathbb{I} \otimes \cdots \otimes \mathbb{I}}_{(a-1) \text{ times}} \otimes V \otimes \underbrace{\mathbb{I} \otimes \cdots \otimes \mathbb{I}}_{(n-a) \text{ times}} \quad (8.12)$$

Else if we have a CNOT-gate with control at qubit wire  $a \in \{0, \dots, n-1\}$  and target  $b \in \{0, \dots, n-1\}$ , there are two options: if  $a < b$  (control before target), we get

$$C_k = \underbrace{\mathbb{I} \otimes \cdots \otimes \mathbb{I}}_{a \text{ times}} \otimes |0\rangle\langle 0| \otimes \underbrace{\mathbb{I} \otimes \cdots \otimes \mathbb{I}}_{(n-a-1) \text{ times}} \\ + \underbrace{\mathbb{I} \otimes \cdots \otimes \mathbb{I}}_{s \text{ times}} \otimes |1\rangle\langle 1| \otimes \underbrace{\mathbb{I} \otimes \cdots \otimes \mathbb{I}}_{(b-a-1) \text{ times}} \otimes X \otimes \underbrace{\mathbb{I} \otimes \cdots \otimes \mathbb{I}}_{(n-b-1) \text{ times}}. \quad (8.13)$$

And if  $a > b$  (target before control), we get

$$\begin{aligned} \mathbf{C}_k = & \underbrace{\mathbb{1} \otimes \cdots \otimes \mathbb{1}}_{a \text{ times}} \otimes |0\rangle\langle 0| \otimes \underbrace{\mathbb{1} \otimes \cdots \otimes \mathbb{1}}_{(n-a-1) \text{ times}} \\ & + \underbrace{\mathbb{1} \otimes \cdots \otimes \mathbb{1}}_{b \text{ times}} \otimes \mathbf{X} \otimes \underbrace{\mathbb{1} \otimes \cdots \otimes \mathbb{1}}_{(a-b-1) \text{ times}} \otimes |1\rangle\langle 1| \otimes \underbrace{\mathbb{1} \otimes \cdots \otimes \mathbb{1}}_{(n-a-1) \text{ times}}. \end{aligned} \quad (8.14)$$

Importantly, every such layer matrix is sparse by construction.

**Fact 8.8** The matrices  $\mathbf{C}_k$  that arise from Eqs. (8.12), (8.13) and (8.14) are all extremely sparse: each row/column contains at most 2 nonzero entries. ■

This helps a lot with performance when it comes to matrix-vector multiplication. And, as it happens, this is precisely the operation we need to update our quantum logical configurations. We start with Eq. (8.11) and compute the final state vector  $\mathbf{U}|b_0 \cdots b_{n-1}\rangle$  via a sequence of  $s$  sparse matrix-vector multiplications:

$$\begin{aligned} |\psi_0\rangle &= |b_0 \cdots b_{n-1}\rangle \in \mathbb{C}^{2^n}, \\ |\psi_{k+1}\rangle &= \mathbf{C}_k |\psi_k\rangle \quad \text{for } k = 0, \dots, s-1, \\ |\psi\rangle &= |\psi_{s+1}\rangle = \mathbf{C}_s |\psi_s\rangle = \cdots = \mathbf{C}_s \times \cdots \times \mathbf{C}_0 |\psi_0\rangle = \mathbf{U} |b_0 \cdots b_{n-1}\rangle. \end{aligned}$$

This computation of the final state vector  $|\psi\rangle \in \mathbb{C}^{2^n}$  involves a total of  $s$  matrix-vector multiplications in  $2^n$  dimensions. Fortunately, each matrix  $\mathbf{C}_k$  involved is extremely sparse, see Fact 8.8. Sparse matrix-vector multiplication routines execute each update using only (order) sparsity  $\times$  dimension  $= 2 \times 2^n$  arithmetic operations. This produces a total resource cost of (at most)  $2s \times 2^n$  arithmetic operations, where  $s$  is the size of the circuit involved.

**Theorem 8.9 (strong simulation of a  $n$ -qubit circuit).** Let  $\mathbf{U}$  be a  $n$ -qubit quantum circuit comprised of  $s$  elementary gates (e.g.  $\mathbf{H}$ ,  $\mathbf{S}$ ,  $\mathbf{CNOT}$  and  $\mathbf{T}$ ). Given an input configuration  $|b_0 \cdots b_{n-1}\rangle$  with  $b_0, \dots, b_{n-1} \in \{0, 1\}^n$ , we can compute the final state vector  $|\psi\rangle = \mathbf{U}|b_0 \cdots b_{n-1}\rangle \in \mathbb{C}^{2^n}$  using a sequence of  $s$  sparse matrix-vector multiplications. The total cost is (at most)  $2s \times 2^n$ .

This process is called a *strong simulation* of the underlying  $n$ -qubit logic. After all, it does provide us with a complete classical description of all  $2^n$  amplitudes that feature in the final  $n$ -qubit configuration. In a sense, this description is even more fine-grained than the inner workings of an actual  $n$ -qubit processor. The latter, after all, does not access amplitudes directly. Instead, we can only sample  $n$ -bit strings  $o_0 \cdots o_{n-1} \in \{0, 1\}^n$  according to a probability distribution that involves the squared amplitudes:

$$\Pr_{\mathbf{U}|b_0 \cdots b_{n-1}\rangle} [o_0 \cdots o_{n-1}] = \Pr_{|\psi\rangle} [o_0 \cdots o_{n-1}] = |\psi_{o_0 \cdots o_{n-1}}|^2. \quad (8.15)$$

One repetition of a  $n$ -qubit quantum computation produces exactly one  $n$ -bit string  $o_0 \cdots o_{n-1}$ . And we may need very, very many repetitions of this process

strong simulation = keep track of  $2^n$ -dim. state vector

to get even a rough idea about the  $2^n$  different (squared) amplitudes involved. The process of being able to sample  $n$ -bit strings from the correct probability distribution (8.15) is called a *weak simulation protocol*. Such protocols actually implement the functionality of a  $n$ -qubit architecture from the beginning (input  $n$ -bit string) to the end (outcome  $n$ -bit string).

Access to the full  $2^n$ -dimensional state vector is enough to sample outcome strings according to the accompanying probability distribution (8.15). Note that this step involves (pseudo-)randomness. The resource cost involved is linear in the length of the state vector and therefore exponential in the number of qubits.

**Proposition 8.10 (strong simulation implies weak simulation).** Assuming access to uniformly random bits and a state vector  $|\psi\rangle \in \mathbb{C}^{2^n}$ , we can sample  $n$ -bit strings according to  $\Pr_{|\psi\rangle}[o_0 \cdots o_{n-1}] = |\psi_{o_0 \cdots o_{n-1}}|^2$  with order  $2^n$  arithmetic operations.

We leave a proof of this statement as an instructive exercise (there are multiple ways to construct a weak simulator). The following consequence now immediately follows from combining Theorem 8.9 with Proposition 8.10.

**Corollary 8.11 (exponential overhead when moving from quantum to classical hardware).** Suppose that we wish to execute a circuit with  $s$  elementary gates on a  $n$ -qubit architecture. Then, we can simulate the entire pipeline on classical hardware. The overhead in resource cost is, however, exponential in  $n$ : we need roughly  $2(s + \text{const}) \times 2^n$  arithmetic operations (and access to random bits).

**Exercise 8.12 (does weak simulation imply strong simulation? (challenging)).** Suppose you have access to a weak simulator for an unknown quantum state vector  $|\psi\rangle$ . How often do you need to run it and produce an outcome bitstring  $o_0 \cdots o_{n-1}$  until you get a good idea about all  $2^n$  amplitudes  $\psi_{o_0 \cdots o_{n-1}}$  involved? Is this task possible at all?

weak simulation = sample from correct readout distribution

strong (& weak) simulation doable, but with exponential overhead (in  $n$ )

### 8.3 Implementing classical circuits with quantum logic

Before, we have asked ourselves whether we can use classical hardware to simulate access to a (hypothetical)  $n$ -qubit architecture. Let us now ask the reverse question: can we use a  $n'$ -qubit architecture to simulate a (hypothetical)  $n$ -bit Boolean circuit?

Recall that a Boolean circuit has  $n$  input bits and  $m$  output bits. In between, it can execute elementary logical gates like NOT ( $\neg$ ), AND ( $\wedge$ ) and OR ( $\vee$ ). The *size*  $s$  of a circuit counts the total number of logical gates. We refer to standard textbooks and lecture notes for detail, e.g. [Kue22].

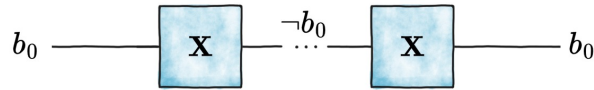
Note that these gates are not native for a quantum architecture. Let us now show how to reformulate them as sequences of native quantum gates (like *CNOT*, *H* and *T*).

### 8.3.1 Quantum realizations of elementary logical gates

#### Reversible negation ( $\neg$ )

The negation operation is special, because it is reversible. The bitflip gate  $X$  implements it natively on single qubits:

reversible NOT: 1 bitflip  $X$



At face value, there is no overhead when implementing this logical functionality on a quantum chip. We may, however, have to revert it at a later point if we need access to the original bit value at a later point within the circuit execution. This reversion costs one additional  $X$ -gate and is displayed at the left of the above diagram.

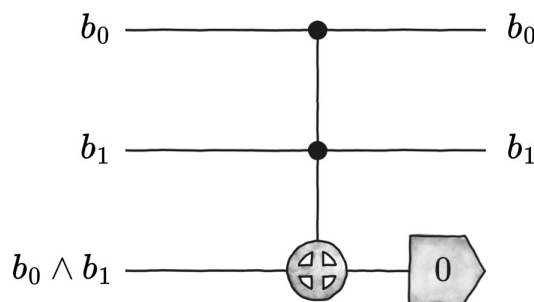
#### Reversible AND ( $\wedge$ )

Let us now move on to realizing the logical AND operation ( $\wedge$ ):

$$b_0 \wedge b_1 = \begin{cases} 0 & \text{if } b_0 = 0 \text{ or } b_1 = 0, \\ 1 & \text{else } (b_0 = 1 \text{ and } b_1 = 1). \end{cases}$$

This operation is not reversible. If we receive 1, we know that both  $b_0$  and  $b_1$  must have been 1. But, if we receive 0 instead, we don't know which of the three input configurations produced it. We can, however, implement  $\wedge$  in a reversible fashion if we allow for an additional (qu)-bit wire. The following 3-qubit circuit uses a Toffoli gate (two-fold controlled-NOT) to do the job:

reversible AND: 3 qubits, 1 Toffoli gate



Correctness of the implementation follows directly from the truth table of the Toffoli gate (8.10): the third bit is flipped from 0 to 1 if and only if  $b_0 = b_1 = 1$ . Otherwise, it stays in 0.

Note that the Toffoli gate is typically not a native gate on a quantum computer. We can, however, use Example 8.2 to decompose  $CCNOT$  into a combination of 2 Hadamards, 25  $T$  gates and 6  $CNOT$ s. This produces the following overhead in terms of elementary quantum gates.

**Lemma 8.13 (reversible implementation of AND ( $\wedge$ )).** We can realize one two-bit AND-gate using one additional qubit (initialized in 0), 6 *CNOT* gates and 27 single-qubit gates (i.e. 33 elementary gates in total).

**Exercise 8.14 (impossibility of implementing  $\wedge$  with two qubits).** Argue that it is impossible to realize a logical AND operation with only two qubits.

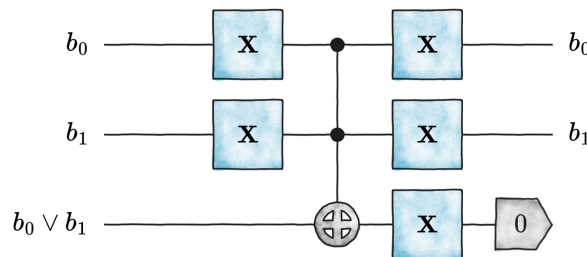
- 1 Why is it impossible to find a 2-qubit unitary  $U \in \mathbb{C}^{4 \times 4}$  such that  $U|b_0 b_1\rangle = |\psi(b_0, b_1)\rangle \otimes |b_0 \wedge b_1\rangle$  for all  $b_0, b_1 \in \{0, 1\}$ ?
- 2 Does this situation change if we allow for a partial readout of the first qubit combined with a conditional unitary on the remaining qubit?

### Reversible OR ( $\vee$ )

Finally, we need to realize the logical OR operation ( $\vee$ ):

$$b_0 \vee b_1 = \begin{cases} 0 & \text{if } b_0 = 0 \text{ and } b_1 = 0, \\ 1 & \text{else } (b_0 = 1 \text{ or } b_1 = 1). \end{cases}$$

Again, this operation is not reversible. If we receive 0, we know that both  $b_0$  and  $b_1$  must have been 0. But, if we receive 1 instead, we don't know which of the three possible input configurations produces it. Similar to logical AND ( $\wedge$ ), we can still implement  $\vee$  in a reversible fashion if we allow for an additional (qu)-bit wire. The following 3-qubit circuit again uses a single Toffoli gate:



reversible OR: 3 qubits, 1 Toffoli gate, 5 bitflips

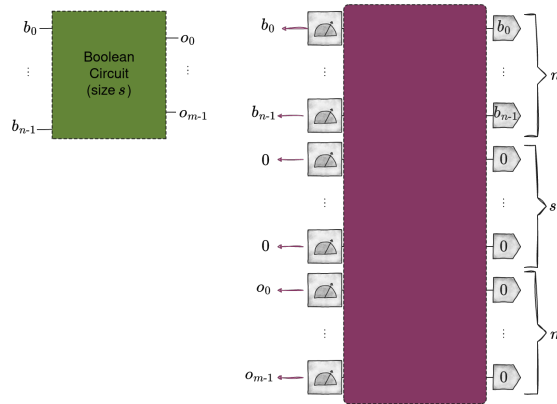
This realization draws inspiration from our AND-implementation. The third qubit wire is initialized to the truth value 1. The only way to change this is to trigger a two-fold controlled-NOT gate. Additional negations on the first two-qubit wires (which are undone later on) ensure that the Toffoli gate fires if and only if both inputs are 0. This produces an effective logical OR gate.

Similar to before, we can further decompose the Toffoli gate into elementary quantum gates (see Example 8.2). Doing so produces the following resource count.

**Lemma 8.15 (reversible implementation of OR ( $\vee$ )).** We can realize one two-bit OR-gate using one additional qubit (initialized in 0), 6 *CNOT* gates and 32 single-qubit gates (i.e. 38 elementary gates in total).

### 8.3.2 Quantum realization of entire Boolean circuits

The reversible execution of a single AND or OR gate requires one additional qubit wire and a total of (at most) 38 elementary quantum gates (see Lemma 8.13



**Figure 8.2** Implementation of Boolean circuits on quantum hardware: every Boolean circuit with  $n$  input bits,  $m$  output bits and  $s$  elementary logical gates (green, lhs) can be converted into a functionally equivalent quantum circuit (purple, rhs). This quantum circuit is larger and longer, but the increase in cost is (at most) linear in circuit size  $\max\{n, m, s\}$ .

and Lemma 8.15). Putting everything together produces an extra overhead that takes into account these extra costs for implementing AND and OR gates.

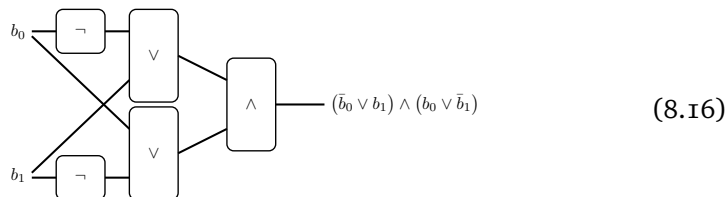
**Theorem 8.16 (realizing a classical circuit on a quantum architecture).** Consider a Boolean circuit with  $n$  input bits,  $m$  output bits and circuit size  $s$ . Then, we can (perfectly and deterministically) simulate this circuit with a quantum architecture that features  $n + m + s$  qubits and requires (at most)  $38 \times s$  elementary quantum gates ( $H, T, CNOT, X$ ).

For most circuits, circuit size  $s$  is the dominant cost parameter. Theorem 8.16 then highlights that the overheads scale (at most) linearly in this dominant cost factor: nr. of qubits =  $O(s)$  and nr. of quantum gates =  $O(s)$ .

**Corollary 8.17 (linear overhead when moving from classical to quantum).** Every classical circuit can also be executed on a quantum architecture. The overhead is (at most) linear in the original circuit size.

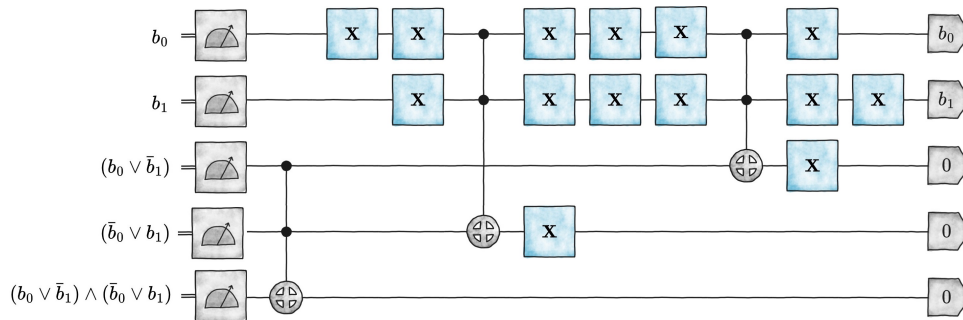
linear overhead permits executing classical circuits on quantum hardware

Let us first illustrate the conversion behind Theorem 8.16 by means of a concrete example. Consider the following logical function with  $n = 2$  input bits and  $m = 1$  output bit:

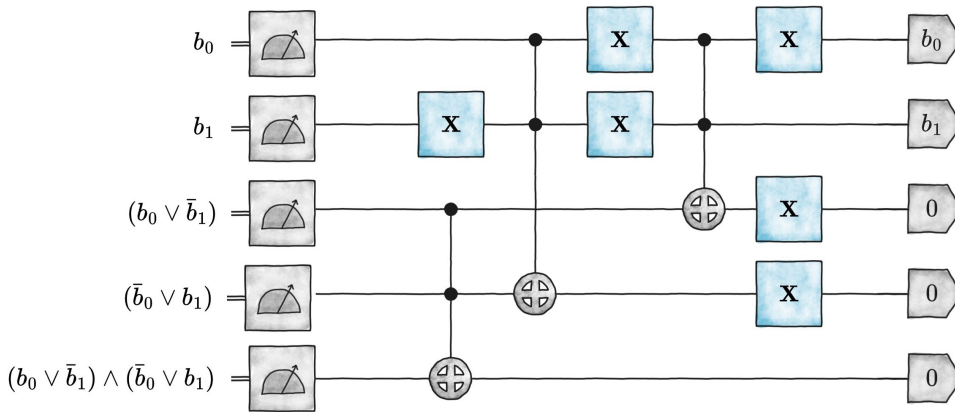


This circuit goes from left to right and contains  $s = 5$  Boolean gates (two  $\neg$ , two  $\vee$  and one  $\wedge$ ). Here is a quantum implementation of the same functionality

which we read from right to left instead:



Each readout is guaranteed to recover the advertised in question with certainty. There are no superpositions whatsoever and the functionality is perfectly deterministic. Already a single look at this circuit reveals that there is a lot of potential for improvement. The  $X$ -gates, in particular, are their own inverses (i.e.  $X \times X = \mathbb{I}$ ) and many of them cancel. Here is a more streamlined representation of this quantum circuit:



In order to compile it into elementary quantum gates, we can use the decomposition of the Toffoli gate (two-fold controlled-NOT gate) into  $H, T, T^\dagger$  and  $CNOT$  from Example 8.2. Doing so produces an actual quantum circuit that acts on

$$n' = 5 = 2 + 3$$

qubit wires. The first two wires carry the logical inputs  $b_0$  and  $b_1$ . Qubits 3, 4 and 5 compute intermediate logical values. Incidentally, the final logical value is also the output of the circuit. There are  $\tilde{s} = 3$  non-reversible operations ( $\wedge$  and  $\vee$ ). Using Lemma 8.13 and Lemma 8.15, we can invest one additional qubit wire and a total of 33 ( $\wedge$ ) and 38 ( $\vee$ ) elementary quantum gates to implement these functionalities in a reversible fashion. Logical negation ( $\neg$ ) is easier by comparison. We can achieve it by applying  $X$  and (potentially) reverting this bit flip again at a later stage to recover the original truth value back. To summarize:



- 1 For each non-reversible operation ( $\wedge, \vee$ ), we need one additional qubit wire and (at most) 38 elementary quantum gates.
- 2 For each output bit, we (may) need an additional qubit wire that is initialized to 0. A single **CNOT** allows us to copy the relevant truth variable into this wire.

It should not come as a surprise that this construction is general. Carrying it out for a general Boolean circuit with  $n$  input bits,  $m$  output bits and  $s$  logical operations, produces the resource overhead displayed in Theorem 8.16.

**Exercise 8.18 (Proof of Theorem 8.16).** Generalize the construction above to general Boolean circuits with  $n$  input bits,  $m$  output bits and a total of  $s$  nontrivial Boolean gates ( $\neg, \wedge, \vee$ ).

**Exercise 8.19 (more direct quantum execution of logical equality).** Can you find a more direct and less wasteful implementation of the logical functionality behind Eq. (8.16)? **Hint:** look up parity check circuits.

## 8.4 Synopsis

Today, we started to compare classical and quantum (hardware) architectures directly with each other. In particular, we have seen that quantum hardware is never much worse than classical hardware. Theorem 8.16 states that every classical circuit can also be executed on a quantum architecture with (at most) linear overhead. In contrast, the transition from quantum to classical hardware looks much more daunting. Theorem 8.9 states that it is possible to simulate quantum architectures with classical software (and hardware, by extension). But, the overhead in cost is substantial: our sparse matrix-vector subroutine could only guarantee a runtime of  $2^n \times \text{circuit-size}$  for a general  $n$ -qubit circuit. This exponential overhead grows quickly. For  $n = 10$ , we obtain  $2^n = 1024$  – which is still manageable. But already  $n = 100$  produces  $2^{100} \approx 1.26 \times 10^{30}$  – this overhead quickly exhausts even the most powerful supercomputers.

At this point, we should point out that sparse matrix-vector multiplication is only one approach to simulate quantum architectures on classical hardware. This approach is also called *array-based simulation* because it involves large arrays (matrices and vectors). Other approaches include *tensor network-based simulations*, *stabilizer-based simulations*, simulation based on *decision diagrams* and many more. These all have their own strengths and weaknesses. But, ultimately, each and every classical simulator developed to date starts to struggle with an exponential cost increase (in the number of qubits  $n$ ).

**Exercise 8.20 (consequences of efficient classical simulation of quantum architectures).** Suppose that it was possible to simulate a general  $n$ -qubit architecture with only polynomial overhead (in  $n$ ). I.e. every quantum circuit with  $s$  gates can be simulated with  $\text{poly}(n) \times s$  arithmetic operations. Shor's algorithm is such a quantum circuit: it factorizes a  $n$  bit integer by repeatedly executing a quantum circuit of size  $s = O(n^3)$ . Use this piece of information to conclude that efficient

$$\begin{array}{ccc} \text{classical} & \xrightarrow{\text{linear}} & \text{quantum} \\ \text{quantum} & \xrightarrow{\text{exponential}} & \text{classical} \end{array}$$

classical simulation of quantum circuits would imply a polynomial-runtime algorithm (in bit size  $n$ ) for integer factorization. What would this mean for the RSA public key encryption protocol?

## 9. Amplitude amplification circuits

Date: 13 December 2023

### 9.1 Motivation

By now, we have all necessary pieces in place to start talking about actual quantum algorithms. Note that the term quantum algorithm can be a bit misleading. Actually, these are quantum circuits designed to achieve certain computational tasks. Today we discuss *amplitude amplification* circuits. The well-known *Grover 'algorithm'* is a particularly prominent example of this kind. It uses quantum effects to find satisfying assignments of a Boolean function faster than a conventional brute-force search ever could. Today, we analyze this circuit and show that this quantum speedup is *quadratic* in nature. Our analysis will also highlight the advantages of a clean mathematical formalism. The matrix-vector multiplication framework will allow us to draw rigorous conclusions without explicitly having to know the underlying logical functionalities. This is a key feature of quantum algorithm design and analysis. (After all, we don't have the hardware (yet) to run these algorithms.)

#### Agenda:

- 1 setup
- 2 overall idea
- 3 circuit design
- 4 pros and cons

### 9.2 Setup

Consider a  $n$ -bit Boolean function

$$f : \{0, 1\}^n \rightarrow \{0, 1\}.$$

Our task is to find

$$\mathbf{b} = b_0 \cdots b_{n-1} \in \{0, 1\}^n \quad \text{such that} \quad f(\mathbf{b}) = 1.$$

Let us start by collecting important cost parameters. The first one is the cost of computing  $f(\mathbf{b})$  for a given input. To ease our transition into the quantum

task: find  $\mathbf{b} \in \{0, 1\}^n$  s.t.  
 $f(\mathbf{b}) = 1$  ('positive answer')

realm, we don't consider the function  $f$  directly, but a Boolean circuit  $C_f$  that implements it. The *circuit size*

$$\text{size}(f) = \# \text{ of elementary logical gates in } C_f$$

measures the cost of implementing this circuit. It is also in one-to-one correspondence with the runtime required to compute  $f$  on a digital computer (or a Turing machine). Another important indicator for the difficulty of this problem is the *ratio of positive answers*:

ratio of positive answers  $r(f)$

$$r(f) = \frac{\# \text{ of inputs such that } f(\mathbf{b}) = 1}{\text{total number of inputs } \mathbf{b} \in \{0, 1\}^n} = \frac{1}{2^n} \sum_{\mathbf{b} \in \{0, 1\}^n} f(\mathbf{b}) \in [0, 1].$$

Together, circuit size and ratio of positive answers bound the expected runtime of a simple randomized solution strategy:

---

**Algorithm 9.1** randomized search for positive answers

---

```

1 while SUCCESS = 0 do
2   sample  $\mathbf{b} = b_0 \dots b_{n-1} \stackrel{\text{unif}}{\sim} \{0, 1\}^n$ 
3   compute  $f(\mathbf{b}) \in \{0, 1\}$ 
4   if  $f(\mathbf{b}) = 1$  then
5     set SUCCESS = 1 and output  $\mathbf{b}$ 

```

---

Instead of discussing this classical strategy, let us reformulate it in terms of a  $(n + \text{size}(f) + 1)$ -qubit architecture. We can use Hadamard gates to create uniform superpositions of all input strings  $\mathbf{b} \in \{0, 1\}^n$  and use a reversible implementation of  $C_f$  to subsequently compute the superposition of all  $f(\mathbf{b})$ s. Fig. 9.1 provides a visualization of such a quantum circuit. How often do we need to execute this circuit (or equivalently: test random input bit strings) until we find a positive answer?

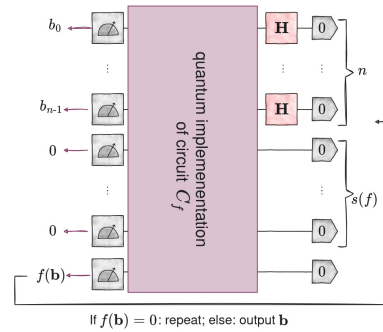
**Proposition 9.1** Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  be a Boolean function with ratio of positive answers  $r(f)$ . Then with probability at least 95%, a total of  $3/r(f)$  repetitions of the quantum circuit displayed in Fig. 9.1 provides us with (at least) one bitstring  $\mathbf{b} \in \{0, 1\}^n$  that obeys  $f(\mathbf{b}) = 1$ .

randomized search for positive answers requires  $3/r(t)$  attempts

Before providing a proof, we emphasize that a number of (approximately)  $1/r(f)$  random input choices is also necessary. It is extremely unlikely to get lucky and sample a positive answer earlier.

*Proof of Proposition 9.1.* The proof follows from computing the state vector of the  $(n + s + 1)$  qubits from beginning to end. In the beginning, we have  $|\varphi_0\rangle = |0 \dots 0\rangle = (|0\rangle)^{\otimes(n+s+1)}$ . Applying Hadamards to the first  $n$  qubits produces a (partial) superposition

$$\begin{aligned} |\varphi_1\rangle &= \mathbf{H}^{\otimes n} \otimes \mathbb{I}^{\otimes(s+1)} |0\rangle^{\otimes(n+s+1)} = (\mathbf{H}|0\rangle)^{\otimes n} \otimes |0\rangle^{s+1} \\ &= \frac{1}{\sqrt{2^n}} \sum_{b_0, \dots, b_{n-1}=0}^1 |b_0 \dots b_{n-1}\rangle \otimes |0\rangle^{\otimes(s+1)} = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{b} \in \{0, 1\}^n} |\mathbf{b}0^s0\rangle. \end{aligned}$$



**Figure 9.1** Quantum implementation of a randomized search for positive answers: this quantum circuit implements Algorithm 9.1. The purple quantum circuit evaluates a Boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  in a reversible fashion. This requires (at most)  $\text{size}(f)$  auxiliary qubits to implement  $\wedge$  and  $\vee$  and a final qubit onto which the result  $f(\mathbf{b})$  is imprinted on:  $\mathbf{C}_f |b_0 \dots b_{n-1} 0^s 0\rangle = |b_0 \dots b_{n-1} 0^s f(\mathbf{b})\rangle$ .

It is now time to apply the purple circuit block from Fig. 9.1:

$$|\varphi_2\rangle = \mathbf{C}_f |\varphi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{b} \in \{0,1\}^n} |\mathbf{b} 0^s f(\mathbf{b})\rangle.$$

This is a uniform superposition over  $2^n$  different bitstrings. Note furthermore that for each  $\mathbf{b} \in \{0, 1\}^n$ , the accompanying function value  $f(\mathbf{b})$  now features at the last qubit location. The readout stage collapses this superposition and selects one of the  $2^n$  contributions uniformly at random. The first  $n$  bit values tell us the (randomly selected) input string  $\mathbf{b} = b_0 \dots b_{n-1}$ , while the final readout value tells us  $f(\mathbf{b})$ .

The probability with which we randomly select a bit string  $\mathbf{b}$  that obeys  $f(\mathbf{b}) = 1$  is in one-to-one correspondence with the ratio of positive answers:

$$\Pr_{\mathbf{b} \sim_{\text{unif}} \{0,1\}^n} [f(\mathbf{b}) = 1] = r(f) \text{ and } \Pr_{\mathbf{b} \sim_{\text{unif}} \{0,1\}^n} [f(\mathbf{b}) = 0] = 1 - r(f).$$

How many trials do we need until we are guaranteed to sample (at least) one bit string that obeys  $f(\mathbf{b}) = 1$ ? To answer this question, let  $T$  denote the total number of times we evaluate the circuit (i.e. the total number of random inputs we try). Then, the probability of *never* sampling a positive answer is

$$\begin{aligned} \Pr [\text{all } T \text{ trials fail}] &= (1 - r(f))^T = \exp(\log(1 - r) \times T) \\ &\leq \exp(-rT), \end{aligned}$$

because  $\log(1 - x) \leq -x$  for all  $x \in [0, 1]$ . If we set  $T = 3/r(f)$ , we obtain a failure probability of at most  $\exp(-3) < 0.05$ . In other words: we get a positive answer with probability  $> 1 - 0.05 = 0.95$ . ■

Many important search problems fall into this broad category. Let us provide two examples.

**Example 9.2 (SAT).** Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  be a Boolean formula in CNF. Then, the task of finding a positive answer  $f(\mathbf{b}) = 1$  also solves the famous satisfiability problem. CNF formulas also have efficient circuit implementations, i.e.  $\text{size}(f) = \text{poly}(n)$ . So, evaluating  $f$  (or constructing the circuit  $C_f$ ) is not the main bottleneck. What makes this problem hard is that we may have to check exponentially many inputs: the ratio of positive answers  $r(f)$  can be as small as  $r(f) = 1/2^n$ . ■

search for positive answers  
covers satisfiability (SAT)

**Example 9.3 (unstructured data base search).** We can also interpret the Boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  as a label function in an unstructured database of bit strings. Viewed from this angle, the task of finding  $\mathbf{b} \in \{0, 1\}^n$  with  $f(\mathbf{b}) = 1$  boils down to finding a database entry with the label ‘yes’. Problems of this kind often occur as subroutines in more involved algorithms. ■

unstructured data base search  
also covered

### 9.3 Overall idea for a quadratic quantum advantage

#### 9.3.1 high-level vision

We have just seen that the ratio of positive answers  $r(f)$  plays an important role when it comes to looking for positive answers. A standard execution of randomized search requires  $T \approx 3/r(f)$  invocations of the function/circuit in question. Each such evaluation comes with a circuit size  $\text{size}(f)$ . Hence, the total cost is

$$T \times \text{size}(f) = 3(\text{size}(f) + n)/r(f) \quad (9.1)$$

scales *linearly* in the inverse ratio of positive answers. Note furthermore that the ratio of positive answers can very well be exponentially small ( $r(f) = 1/2^n$  if there is exactly one positive answer). In these cases, the total cost (9.1) also explodes exponentially.

total classical cost is  
 $\approx \text{size}(f)/r(f)$

Let us now present a high-level idea that uses quantum circuits to achieve a *quadratic improvement* in total cost. The resulting quantum circuit  $\mathbf{G}$  will have (quantum) circuit size

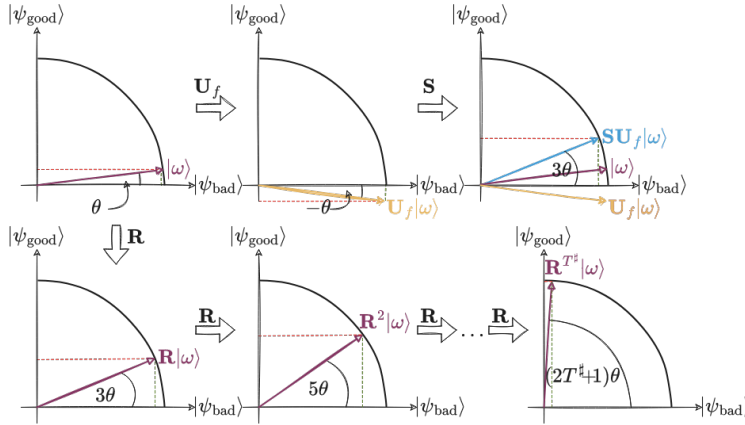
$$\text{size}(\mathbf{G}) = O\left((\text{size}(f) + n) / \sqrt{r(f)}\right). \quad (9.2)$$

The first contribution is comparable to the cost of executing the circuit  $C_f$  once. The second contribution is where things get interesting:  $1/\sqrt{r(f)} = \sqrt{1/r(f)}$  is *quadratically* smaller than the  $1/r(f)$ -term that dominates the classical cost (9.1). How can we hope to achieve such a quadratic improvement? There are two main ideas that we shall now cover.

total quantum cost is  
 $\approx \text{size}(f)/\sqrt{r(f)}$

#### (i) re-interpret the uniform superposition in a problem-related fashion:

Fix the Boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  and let  $t_+(f)$  be the number of inputs such that  $f(\mathbf{b}) = 1$ . Likewise, let  $t_-(f) = 2^n - t_+(f)$  be the number of inputs such that  $f(\mathbf{b}) = 0$ . Then, we can rewrite the uniform superposition



**Figure 9.2** *Geometric intuition behind amplitude amplification:* view the uniform superposition  $|\omega\rangle$  as a combination of the superposition of bad input strings  $|\psi_{\text{bad}}\rangle$  ( $f(\mathbf{b}) = 0$ ) and the superposition of good input strings  $|\psi_{\text{good}}\rangle$  ( $f(\mathbf{b}) = 1$ ). The amplitude of  $|\psi_{\text{good}}\rangle$  is proportional to  $\theta \approx \sqrt{r(f)} \ll 1$  (top right). A reflection about  $|\psi_{\text{good}}\rangle$  (top center) followed by a reflection about  $|\omega\rangle$  (top right) implement a rotation  $\mathbf{R}$  that amplifies this amplitude (bottom right). A sequential application of many rotations amplifies this good amplitude to approximately everything (bottom left).

over all  $n$ -bit strings as

$$\begin{aligned}
 |\omega\rangle &= (\mathbf{H}|0\rangle)^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{b} \in \{0,1\}^n} |\mathbf{b}\rangle \\
 &= \frac{1}{\sqrt{2^n}} \sum_{\mathbf{b}: f(\mathbf{b})=0} |\mathbf{b}\rangle + \frac{1}{\sqrt{2^n}} \sum_{\mathbf{b}: f(\mathbf{b})=1} |\mathbf{b}\rangle \\
 &= \sqrt{\frac{t_-(f)}{2^n}} \frac{1}{\sqrt{t_-(f)}} \sum_{\mathbf{b}: f(\mathbf{b})=0} |\mathbf{b}\rangle + \sqrt{\frac{t_+(f)}{2^n}} \frac{1}{\sqrt{t_+(f)}} \sum_{\mathbf{b}: f(\mathbf{b})=1} |\mathbf{b}\rangle \\
 &= \sqrt{\frac{2^n - t_+(f)}{2^n}} |\psi_{\text{bad}}\rangle + \sqrt{\frac{t_+(f)}{2^n}} |\psi_{\text{good}}\rangle \\
 &= \sqrt{1 - r(f)} |\psi_{\text{bad}}\rangle + \sqrt{r(f)} |\psi_{\text{good}}\rangle. \tag{9.3}
 \end{aligned}$$

This tells us that the uniform superposition is actually a linear combination of two quantum states: the superposition of ‘bad inputs’  $|\psi_{\text{bad}}\rangle$  and the superposition of ‘good inputs’  $|\psi_{\text{good}}\rangle$ . The amplitude in front of the superposition of good inputs scales like the square root of the ratio of positive answers. This is a small number, but not quite as small as  $r(f)$  itself. Amplitude amplification ultimately leverages this quadratic discrepancy. To make our life easier later on, we replace the amplitudes in Eq. (9.3) with trigonometric functions. Let  $\theta \in [0, 2\pi)$  be the angle that obeys

$$\sqrt{r(f)} = \sin(\theta), \quad \text{such that} \quad |\omega\rangle = \cos(\theta) |\psi_{\text{bad}}\rangle + \sin(\theta) |\psi_{\text{good}}\rangle. \tag{9.4}$$

We refer to Fig. 9.2 (top left) for a visual illustration. The problem is that for  $\theta \ll 1$  (which happens if  $r(f) \ll 1$ ),  $\cos(\theta) \approx 1$  and  $\sin(\theta) \approx 0$ . This is also why our first quantum circuit needs so many trials to get a positive answer. However, it is also true that the uniform superposition  $|\omega\rangle = |\mathbf{H}^{\otimes n}|0 \dots 0\rangle$  does also contain a bit of the good answers:  $\sin(\theta) > 0$  whenever  $r(f) > 0$ . We refer to Fig. 9.2 (left) for a visual illustration. The second quantum idea is designed to increase the amplitude of  $|\psi_{\text{good}}\rangle$  at the cost of diminishing the amplitude of  $|\psi_{\text{bad}}\rangle$ .

uniform superposition  $|\omega\rangle$  contains both bad and good bitstrings

**(ii) amplify the amplitude belonging to the superposition of 'good' bitstrings**

Recall that quantum circuits act as unitary matrices on state vectors. What is more, we can view Eq. (9.4) as an effective single-qubit state vector with only two amplitudes:

$$|\omega\rangle = \cos(\theta)|\psi_{\text{bad}}\rangle + \sin(\theta)|\psi_{\text{good}}\rangle = \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} \quad \text{with} \quad \langle \psi_{\text{bad}} | \psi_{\text{good}} \rangle = 0.$$

Next, suppose that we are somehow able to implement the following rotation gate on this effective qubit:

$$\mathbf{R} = \begin{pmatrix} \cos(2\theta) & -\sin(2\theta) \\ \sin(2\theta) & \cos(2\theta) \end{pmatrix}. \quad (9.5)$$

We emphasize that this is not really a single-qubit gate, but a full-fledged  $n$ -qubit circuit (perhaps even larger). We demand, however, that it transforms the two special states  $|\psi_{\text{bad}}\rangle, |\psi_{\text{good}}\rangle$  in exactly this fashion. In particular,

$$\begin{aligned} \mathbf{R}|\omega\rangle &= \begin{pmatrix} \cos(2\theta) & -\sin(2\theta) \\ \sin(2\theta) & \cos(2\theta) \end{pmatrix} \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} \\ &= \begin{pmatrix} \cos(2\theta)\cos(\theta) - \sin(2\theta)\sin(\theta) \\ \sin(2\theta)\cos(\theta) + \cos(2\theta)\sin(\theta) \end{pmatrix} \\ &= \begin{pmatrix} \cos((2+1)\theta) \\ \sin((2+1)\theta) \end{pmatrix}, \end{aligned}$$

because  $\sin(\alpha+\beta) = \sin(\alpha)\cos(\beta) + \cos(\alpha)\sin(\beta)$  and  $\cos(\alpha+\beta) = \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta)$ . More generally, we obtain for  $T \geq 2$

iteratively increase amplitude of good bitstrings within  $|\omega\rangle$

$$\mathbf{R}^{\times T}|\omega\rangle = \mathbf{R} \times \dots \times \mathbf{R}|\omega\rangle = \begin{pmatrix} \cos((2T+1)\theta) \\ \sin((2T+1)\theta) \end{pmatrix}.$$

In other words:  $T$  sequential applications of  $\mathbf{R}$  increase the angle  $\theta$  approximately  $T$ -fold. And a larger angle also makes the sin-contribution larger at the cost of the cos-contribution. The extreme case occurs at angle  $\pi/2$  where  $\sin(\pi/2) = 1$  and  $\cos(\pi/2) = 0$ . That is, if we choose

$$T_{\#} \geq 2 \quad \text{such that} \quad (2T_{\#} + 1)\theta \approx \pi/2, \quad (9.6)$$



the resulting quantum state will (almost) only feature good contributions:

$$\begin{aligned} \mathbf{R}^{\times T_{\#}} \mathbf{H}^{\otimes n} |0 \dots 0\rangle &= \cos((2T_{\#} + 1)\theta) |\psi_{\text{bad}}\rangle + \sin((2T_{\#} + 1)\theta) |\psi_{\text{good}}\rangle \\ &\approx \cos(\pi/2) |\psi_{\text{bad}}\rangle + \sin(\pi/2) |\psi_{\text{good}}\rangle = 0 \times |\psi_{\text{bad}}\rangle + 1 \times |\psi_{\text{good}}\rangle \\ &= |\psi_{\text{good}}\rangle = \frac{1}{\sqrt{r(f)}} \sum_{\mathbf{b}: f(\mathbf{b})=1} |\mathbf{b}\rangle. \end{aligned}$$

Note that the final expression only features ‘good’ bit strings  $\mathbf{b} \in \{0, 1\}^n$  such that  $f(\mathbf{b}) = 1$ . Any one of them solves our problem. And performing the readout will provide us with precisely one of them. Finally, note that  $\theta$  is in one-to-one correspondence with  $1/\sqrt{r(f)}$ , courtesy of Eq. (9.4) and the fact that  $\sin(\theta) \approx \theta$  for  $\theta \ll 1$  (in fact  $\sin(\theta) \leq \theta$  for all  $\theta \in [0, 2\pi)$ ). The following rigorous proposition is an immediate consequence of our analysis.

**Proposition 9.4 (amplitude amplification, high-level).** Fix a Boolean function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  with ratio of positive events  $r(f)$  and set  $\theta \in [0, 2\pi)$  such that  $\sin(\theta) = \sqrt{r(f)}$ . Suppose that it is possible to generate a  $n$ -qubit circuit  $\mathbf{R}$  that acts as Eq. (9.5) on  $|\psi_{\text{bad}}\rangle$  and  $|\psi_{\text{good}}\rangle$ . Then,  $T_{\#} \leq \pi/(4\sqrt{r(f)})$  sequential applications of  $\mathbf{R}$  (approximately) turn the uniform superposition  $|\omega\rangle = \mathbf{H}^{\otimes n} |0 \dots 0\rangle$  into a superposition  $|\psi_{\text{good}}\rangle$  of only positive answers.

Note that the term approximately is actually necessary in Proposition 9.4. The optimal choice of  $T_{\#}$  in Eq. (9.6) requires us to approximate the possibly irrational fraction  $\pi/(2\theta)$  with an integer  $2T_{\#} + 1$ . This necessarily introduces rounding errors. Fortunately, these rounding errors are small and  $\sin$  is a smoothly varying function. In particular,  $2T_{\#} + 1 \approx \pi/(2\theta)$  is enough to ensure  $\sin((2T_{\#} + 1)\theta) \approx \sin(\pi/2) = 1$  via a continuity argument.

## 9.4 Concrete circuit construction

Proposition 9.4 provides us with a clear strategy on how to look for positive answers with the help of a quantum architecture. What is still missing is a concrete realization of the (effective) rotation matrix  $\mathbf{R}$  in Eq. (9.5). The following concrete construction dates back to Grover’s work from 1996. It combines two big quantum circuits that each act as a geometric reflection (think: mirrors). Both are illustrated in Fig. 9.3 and deserve a bit of extra attention.

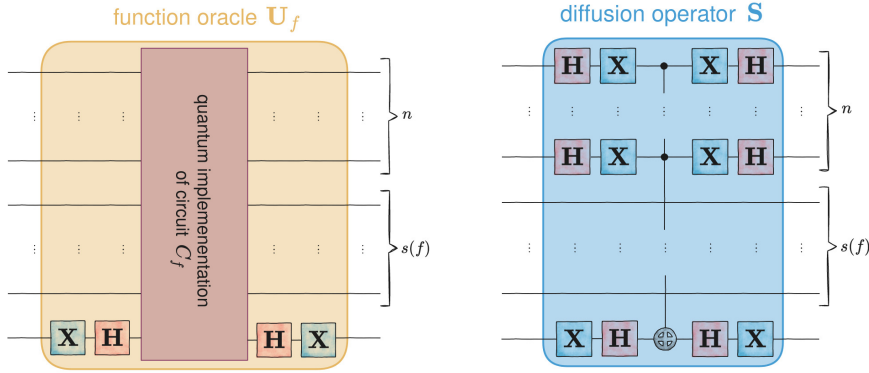
### 9.4.1 Circuit 1: reflection about good solutions (‘Grover oracle’)

Let us first take a closer look at the lhs circuit in Fig. 9.3, the so-called function oracle.

**Lemma 9.5 (action of function oracle).** The first circuit displayed in Fig. 9.3 acts on the first  $n$  qubits as

$$\mathbf{U}_f = \sum_{\mathbf{b} \in \{0,1\}^n} (-1)^{f(\mathbf{b})} |\mathbf{b}\rangle\langle\mathbf{b}|. \quad (9.7)$$

the function oracle acts as a reflection about ‘good bitstrings’ ( $\mathbf{b}$  s.t.  $f(\mathbf{b}) = 1$ )



**Figure 9.3** Quantum circuit blocks that achieve amplitude amplification:

(left): the function oracle  $\mathbf{U}_f$  uses a reversible implementation of  $C_f$  to change the sign of good bitstring answers:  $\mathbf{U}_f|\mathbf{b}\rangle = (-1)^{f(\mathbf{b})}|\mathbf{b}\rangle$  for  $\mathbf{b} \in \{0, 1\}^n$ .

(right): the diffusion operator acts as a reflection about the uniform superposition:  $\mathbf{S}|\omega\rangle = -|\omega\rangle$  and  $\mathbf{S}|\nu\rangle = +|\nu\rangle$  whenever  $\langle\omega|\nu\rangle = 0$  (orthogonality).

In words: This unitary matrix reflects ‘good input states’ and leaves ‘bad input states’.

*Proof.* We can without loss restrict our attention to the first  $n$  qubits plus the last one. The part in the middle is only required to reversibly implement logical  $\wedge, \vee$ . It starts in 0 and ends in 0. The remaining circuit prepares the last qubit in the state  $|-\rangle = \mathbf{H}\mathbf{X}|0\rangle = \mathbf{H}|1\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$ . We then XOR the value of  $f(\mathbf{b})$  to it. For  $\mathbf{b} \in \{0, 1\}^n$ , we obtain

$$\begin{aligned} \mathbf{C}_f \times (\mathbb{1}^{\otimes n} \otimes (\mathbf{H}\mathbf{X})) |\mathbf{b}0\rangle &= |\mathbf{b}\rangle \otimes (|0 \oplus f(\mathbf{b})\rangle - |1 \oplus f(\mathbf{b})\rangle) / \sqrt{2} \\ &= (-1)^{f(\mathbf{b})} |\mathbf{b}\rangle \otimes |-\rangle. \end{aligned}$$

The final  $\mathbf{X}$  and  $\mathbf{H}$  convert the  $|-\rangle$ -state back into the  $|0\rangle$ -state we start with:

$$\mathbf{U}_f|\mathbf{b}0\rangle = (-1)^{f(\mathbf{b})}|\mathbf{b}0\rangle \quad \text{for } \mathbf{b} \in \{0, 1\}^n.$$

Eq. (9.7) subsumes all these actions on different bit strings into a single display. ■

We can use Lemma 9.5 to infer the action of  $\mathbf{U}_f$  on the effective qubit spanned by  $|\psi_{\text{b}}\rangle = |\psi_{\text{bad}}\rangle$  and  $|\psi_{\text{g}}\rangle = |\psi_{\text{good}}\rangle$ . Linearity ensures

$$\begin{aligned} \mathbf{U}_f|\psi_{\text{b}}\rangle &= \sqrt{1-r(f)} \sum_{\mathbf{b}:f(\mathbf{b})=0} \mathbf{U}_f|\mathbf{b}\rangle = \sqrt{1-r(f)} \sum_{\mathbf{b}:f(\mathbf{b})=0} (-1)^{f(\mathbf{b})}|\mathbf{b}\rangle \\ &= \sqrt{1-r(f)} \sum_{\mathbf{b}:f(\mathbf{b})=0} |\mathbf{b}\rangle = |\psi_{\text{b}}\rangle \end{aligned}$$

and, likewise

$$\begin{aligned} \mathbf{U}_f|\psi_{\text{g}}\rangle &= \sqrt{r(f)} \sum_{\mathbf{b}:f(\mathbf{b})=1} \mathbf{U}_f|\mathbf{b}\rangle = \sqrt{r(f)} \sum_{\mathbf{b}:f(\mathbf{b})=1} (-1)^{f(\mathbf{b})}|\mathbf{b}\rangle \\ &= -\sqrt{r(f)} \sum_{\mathbf{b}:f(\mathbf{b})=1} |\mathbf{b}\rangle = -|\psi_{\text{g}}\rangle. \end{aligned}$$

Putting everything together produces

$$\mathbf{U}_f^{\text{eff}} = |\psi_b\rangle\langle\psi_b| - |\psi_g\rangle\langle\psi_g| = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \quad (9.8)$$

### 9.4.2 Circuit 2: reflection about uniform superposition ('diffusion operator')

Let us now take a closer look at the rhs circuit in Fig. 9.3, the so-called diffusion operator. At the heart of this diffusion operator is a  $n$ -fold controlled-NOT gate – a straightforward generalization of the Toffoli gate to  $(n + 1)$  qubits.

**Lemma 9.6** The second circuit displayed in Fig. 9.3 acts on the first  $n$  qubits as

$$\mathbf{S} = \mathbb{I}^{\otimes n} - 2|\omega\rangle\langle\omega|. \quad (9.9)$$

In words: this is a reflection about the state  $|\omega\rangle$ , i.e.  $\mathbf{S}|\omega\rangle = -|\omega\rangle$  and  $\mathbf{S}|\nu\rangle = +|\nu\rangle$  whenever  $\langle\omega|\nu\rangle = 0$  (orthogonality).

**Exercise 9.7 (Proof of Lemma 9.6).** Show that the circuit depicted in Fig. 9.3 is indeed described by Eq. (9.9).

We can use  $|\omega\rangle = \cos(\theta)|\psi_{\text{bad}}\rangle + \sin(\theta)|\psi_{\text{good}}\rangle$  (Eq. (9.4)) to infer the action of  $\mathbf{S}$  on the effective qubit spanned by  $|\psi_{\text{bad}}\rangle$  and  $|\psi_{\text{good}}\rangle$ :

$$\begin{aligned} \mathbf{S}^{\text{eff}} &= (|\psi_b\rangle\langle\psi_b| + |\psi_g\rangle\langle\psi_g|) - 2(\cos(\theta)|\psi_b\rangle + \sin(\theta)|\psi_g\rangle)(\cos(\theta)\langle\psi_b| + \sin(\theta)\langle\psi_g|) \\ &= (1 - 2\cos^2(\theta))|\psi_b\rangle\langle\psi_b| - \sin(\theta)\cos(\theta)|\psi_b\rangle\langle\psi_g| \\ &\quad - \sin(\theta)\cos(\theta)|\psi_g\rangle\langle\psi_b| + (1 - 2\sin^2(\theta))|\psi_g\rangle\langle\psi_g| \\ &= \begin{pmatrix} 1 - 2\cos^2(\theta) & -\sin(\theta)\cos(\theta) \\ -\sin(\theta)\cos(\theta) & 1 - 2\sin^2(\theta) \end{pmatrix} = - \begin{pmatrix} \cos(2\theta) & \sin(2\theta) \\ \sin(2\theta) & -\cos(2\theta) \end{pmatrix}. \end{aligned} \quad (9.10)$$

Here, we have used some additional trigonometric identities<sup>1</sup>. Although it looks similar, this is not a rotation. The determinant, in particular, is  $-\cos^2(\theta) - \sin^2(\theta) = -1$  which is indicative of a reflection (think mirror image).

### 9.4.3 Combination of the two circuit blocks

We now have effective single-qubit actions of both the function oracle (Eq. (9.8)) and the diffusion operator (Eq. (9.10)). A combination of the two yields the following effective gate matrix:

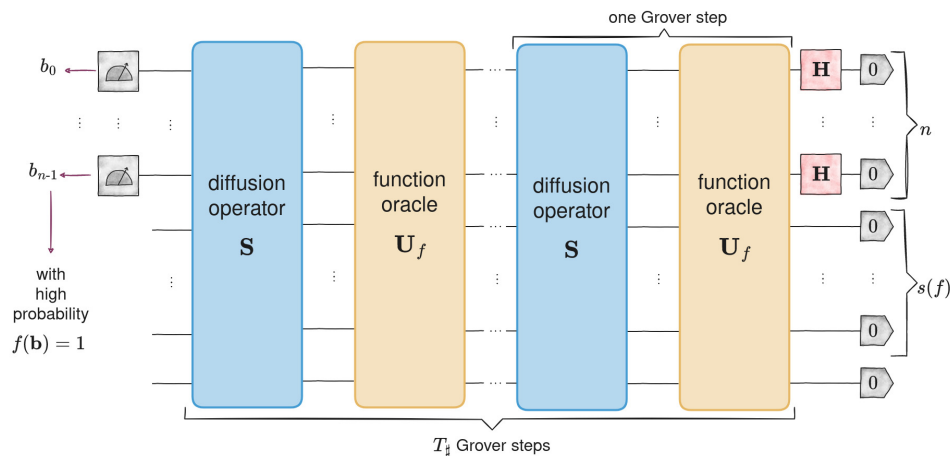
$$\begin{aligned} \mathbf{R} &= \mathbf{S}^{\text{eff}} \times \mathbf{U}_f^{\text{eff}} = - \begin{pmatrix} \cos(2\theta) & \sin(2\theta) \\ \sin(2\theta) & -\cos(2\theta) \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \\ &= - \begin{pmatrix} \cos(2\theta) & -\sin(2\theta) \\ \sin(2\theta) & \cos(2\theta) \end{pmatrix}. \end{aligned} \quad (9.11)$$

This is now a proper rotation matrix, at least up to a global sign ( $-1$ ) that does not matter for quantum logic. Note furthermore that this effective single-qubit functionality has exactly the form we need for Proposition 9.4.

<sup>1</sup>In particular:  $\sin(\theta)\cos(\theta) = \sin(2\theta)/2$ ,  $1 - 2\sin^2(\theta) = \cos(2\theta)$  and  $1 - 2\cos^2(\theta) = 2\sin^2(\theta) - 1 = -\cos(2\theta)$  (because  $\cos^2(\theta) + \sin^2(\theta) = 1$ ).

the diffusion operator acts as a reflection about  $|\omega\rangle$  (uniform superposition)

a combination of diffusion and function oracles acts as an effective rotation



**Figure 9.4** Full amplitude amplification circuit: generate a uniform superposition over all  $n$ -bit strings and then sequentially apply  $T_{\#}$  combinations of diffusion operator and function oracle. A proper choice of  $T_{\#} \approx \pi/(4r(f))$  amplifies the amplitudes of ‘good bit strings’ ( $\mathbf{b}$  s.t.  $f(\mathbf{b}) = 1$ ) while essentially erasing the amplitudes of ‘bad bit strings’ ( $\mathbf{b}$  s.t.  $f(\mathbf{b}) = 0$ ). The total circuit size is  $O(T_{\#}(n + \text{size}(f)))$ .

## 9.5 Full quantum search algorithm

We now have all the pieces in place for a fully-fledged quantum search algorithm. At the heart of it are two building blocks:

- (i) the function oracle which is essentially a reversible implementation of the function  $f$  (Fig. 9.3, left) and
- (ii) the diffusion operator with a  $n$ -fold controlled NOT-gate at its center (Fig. 9.3, right).

In contrast to our quantum implementation of random search in Fig. 9.1, we now group many of these elementary blocks into a single quantum circuit. A total of  $T_{\#} \approx \pi(2\theta) \gtrsim \pi/(4\sqrt{r(f)})$  to be precise. The last missing ingredient is an initial preparation of the uniform superposition  $|\omega\rangle = \mathbf{H}^{\otimes n}|0\dots 0\rangle$  on the first  $n$  qubits. We refer to Fig. 9.4 for a visual illustration. Proposition 9.4 and the explicit construction of the required rotation matrix in Eq. (9.11) readily imply the following theoretical underpinning.

**Theorem 9.8 (amplitude amplification circuit).** Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  be a Boolean function with ratio of positive answers  $r(f)$  and circuit size  $\text{size}(f)$ . Set  $T_{\#} \approx \pi/(4\sqrt{r(f)})$ . Then, the circuit displayed in Fig. 9.4 almost produces a uniform superposition of all bitstrings  $\mathbf{b}$  such that  $f(\mathbf{b}) = 1$ . A readout of the first  $n$  qubits produces one of them with a very high probability.

amplitude amplification achieves quadratic speedup

The total size of this circuit is  $O(\text{size}(f)/\sqrt{r(f)})$  (assuming  $\text{size}(f) \geq n$ ). For sufficiently large input sizes  $n \gg 1$ , this becomes much cheaper than the

total cost of  $3 \times \text{size}(f)/r(f)$  required to execute a random search protocol (Algorithm 9.1). This *quadratic discrepancy* becomes particularly pronounced if the ratio of positive answers  $r(f)$  is very small ( $r(f) \approx 1/2^n$ ).

Next, we point out that the quantifier ‘almost’ in Theorem 9.8 takes into account rounding issues when choosing the optimal number of iterations  $T_{\#} \approx \pi/(4\sqrt{r(f)})$ . We can handle this by repeating the circuit multiple times until we get an output  $\mathbf{b}$  that fulfils  $f(\mathbf{b}) = 1$  (evaluating the function for a single input is very cheap by comparison). And continuity arguments ensure that the probability of getting a good string is very close to  $\sin(\pi/2) = 1$ .

A more serious issue is the fact that the number of repetitions  $T_{\#}$  depends on the ratio of positive answers  $r(f)$ . And this ratio may not be known in advance! To make matters worse, it is absolutely possible to overshoot amplitude amplification. If we get  $T_{\#}$  wrong by a factor of 2, for instance, we effectively end with a superposition of bad answers:

$$(\mathbf{S} \times \mathbf{U}_f)^{2T_{\#}} |\omega\rangle \approx \cos(2(\pi/2)) |\psi_{\text{bad}}\rangle + \sin(2(\pi/2)) |\psi_{\text{good}}\rangle = -|\psi_{\text{bad}}\rangle.$$

This periodicity is an unavoidable feature when constructing amplitude amplification the way we have done it here (which dates back to Grover in the 90s). However, much more recently, researchers have developed an alternative way to construct amplitude amplification circuits that don’t have this problem. This construction is based on a technique called quantum singular value transform. Discussing it would go beyond the scope of this introductory lecture.

amplitude amplification can overshoot

# 10. Fourier-type transforms

Date: 10 January 2024

Lecturer: Alexander Ploier

## 10.1 Fourier transform

### 10.1.1 Motivation

The Fourier transform is one of the most powerful tools in signal processing, data analysis and applied math. But, its utility extends to quantum computing as well. Shor's algorithms, for instance, use a (quantum) Fourier transform as an important subroutine. This is more than enough justification to look at this transformation and its quantum realization. For starters, let us discuss what a Fourier transform does and why (classical) people care. From that point on, we can derive the quantum Fourier transform. So what exactly is the Fourier transform anyway? Here is an informal definition from audio processing:

**Example 10.1 (Fourier transform in audio processing).** The Fourier transform takes a signal in the time domain, e.g. an audio signal, and transforms it into its different components in the frequency domain:

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)e^{-2\pi i x \xi} dx.$$

Imagine we have a smoothie in front of us, any kind of smoothie, and we want to find out what it is made of. Of course, we can drink it, and see if we can guess all the ingredients, but that is not very scientific, is it? A better way to handle this would be to pour the smoothie through a series of filters. Each and every one of those filters has exactly one ingredient until there is nothing left of our smoothie. Of course, we do want additional restrictions for our filters.

### Agenda:

- 1 Fourier transform
- 2 Discrete Fourier transform
- 3 Quantum Fourier transform
- 4 Quantum phase estimation

Fourier transform

- Filters must be independent of each other. (E.g. Banana filter only captures the amount of bananas in the smoothie.)
- Filters must be complete. (E.g. We won't get the real recipe if we do not check for something.)
- Ingredients must be combineable. (E.g. Recombining the ingredients in any order has to lead back to the same smoothie)

The Fourier transform finds the *recipe* for a signal, in the same way filtering a smoothie will give us the recipe to make a new one.

For example, earthquake vibrations can be separated into vibrations of different frequencies and amplitudes, which allows buildings to be designed to withstand the strongest ones.

### 10.1.2 Discrete Fourier transform

The traditional Fourier transform is an analogue operation that is defined for analogue signals (functions). There is, however, also a discrete variant thereof. This transformation is characterized by the (discrete) dimension  $D$  and features prominently in modern (digital) audio+image processing tasks. There, the dimension  $D$  is typically finite, but very large.

**Example 10.2 (Discrete Fourier transform in audio processing).** Imagine we have a messed-up audio signal, with some annoying frequencies. We want to filter those out to get a clear signal. We have a 10 second audio clip with  $4.4 \times 10^5$  data points. To transform this signal into the frequency domain, to get rid of those unwanted ones, we apply the straight-forward **discrete Fourier transform**. It takes  $D^2$  operations to transform, with  $4.4 \times 10^5$  data points, this yields around  $10^{11}$  operations i.e. a hundred billion operations. Which is quite a lot. ■

Let us now take a look at the discrete Fourier transform on a smaller scale, i.e. two data points or  $D = 2$ . Let us denote the associated discrete Fourier transform by  $DFT_2$ . To construct it, we use the following important fact from algebra (fundamental theorem of algebra).

**Fact 10.3 ( $D$ -th roots of unity).** For  $D = 1, 2, 3, \dots$ , there are exactly  $D$  solutions to the polynomial equation  $x^D = 1$ . These are complex phases and take the following form:  $\exp((k \times 2\pi i)/D)$  with  $k = 0, \dots, D - 1$ . ■

$D$ -th roots of unity

If  $D = 2$ , the equation is  $x^2 = 1$  and we know that there are two solutions:  $x = +1$  and  $x = -1$ . We can also write them using the language of complex phases. Define  $\omega = e^{2\pi i/D}$ . For  $D = 2$  we have

$$\omega = e^{2\pi i/2} = -1,$$

$$\omega^2 = e^{2\pi i} = +1,$$

$$\omega^3 = e^{3\pi i} = -1,$$

$$\vdots$$

For  $D = 3$ , we get three solutions, namely:

$$\begin{aligned} \omega &= e^{2\pi i/3} = -0.5 + 0.866i \\ \omega^2 &= (e^{2\pi i/3})^2 = -0.5 - 0.866i \\ \omega^3 &= (e^{2\pi i/3})^3 = 1 \\ \omega^4 &= -0.5 + 0.866i \\ &\vdots \end{aligned}$$

This construction extends to general  $D$ . At the heart is the  $D$ -th root of unity  $\omega_D = \exp(2\pi i/D) \in \mathbb{C}$  which is one solution to  $x^D = 1$ . The other solutions can be obtained by taking powers of this complex amplitude:  $\omega_D^k$  obeys  $x^D = 1$  for  $k = 0, \dots, D - 1$ . Powers of  $D$ -th roots of unity are the main building blocks of the discrete Fourier transform.

**Theorem 10.4 (Discrete Fourier Transform (DFT) matrix in  $D$  dimensions).** Fix a dimension  $D \geq 2$ . Then, the discrete Fourier transform is defined by the action of the following  $D \times D$  matrix:

discrete Fourier transform

$$\mathbf{DFT}_D = \frac{1}{\sqrt{D}} \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_D & \omega_D^2 & \omega_D^3 & \cdots & \omega_D^{D-1} \\ 1 & \omega_D^2 & \omega_D^4 & \omega_D^6 & \cdots & \omega_D^{2D-2} \\ 1 & \omega_D^3 & \omega_D^6 & \omega_D^9 & \cdots & \omega_D^{3D-3} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_D^{D-1} & \omega_D^{2D-2} & \omega_D^{3D-3} & \cdots & \omega_D^{(D-1)(D-1)} \end{pmatrix}.$$

Let us illustrate this construction by means of a couple of examples. For  $D = 2$ , the DFT matrix becomes

$$\mathbf{DFT}_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & \omega \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

As you can see,  $\mathbf{DFT}_2$  is equal to the matrix representation of the single-qubit Hadamard gate. For  $D = 3$ , we instead obtain

$$\mathbf{DFT}_3 = \frac{1}{\sqrt{3}} \begin{pmatrix} 1 & 1 & 1 \\ 1 & -0.5 + 0.866i & -0.5 - 0.866i \\ 1 & -0.5 - 0.866i & -0.5 + 0.866i \end{pmatrix}$$

**Example 10.5 (computing discrete Fourier transforms).** For  $D = 3$ , the discrete Fourier transform acts on vectors of dimension 3 via matrix-vector multiplication. The vector  $\mathbf{e}_1 = (1, 0, 0)^T$ , for instance, gets mapped to

$$\hat{\mathbf{e}}_1 = \mathbf{DFT}_3 \mathbf{e}_1 = \frac{1}{\sqrt{3}} \begin{pmatrix} 1 & 1 & 1 \\ 1 & -0.5 + 0.866i & -0.5 - 0.866i \\ 1 & -0.5 - 0.866i & -0.5 + 0.866i \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{3}} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

■

**Exercise 10.6** Find the Fourier transform for  $D = 4$ .



## 10.2 Quantum Fourier transform

For the quantum Fourier transform, we set  $D = 2^n$ . Then we can implement the  $DFT$  algorithm on a  $n$ -qubit circuit as a  $2^n \times 2^n$  unitary matrix. But we can do better than that. We are going to derive an  $n$ -qubit circuit which yields an exponential speed-up over this straight-forward approach. We show that by being clever and looking really hard at the matrix we can reduce the number of layers in our quantum circuit to  $n^2$ . Before we can do that, we first have to show that our general  $DFT_D$  matrix is, in fact, a unitary matrix.

**Proposition 10.7** The matrix  $DFT_D$  is unitary.

$DFT_D$  is unitary

*Proof.* It is known that an operator is unitary if its columns are orthonormal. Denote the  $k$ -th column vector of  $DFT_D$  as follows:

$$\mathbf{f}_k = \frac{1}{\sqrt{D}} \begin{pmatrix} 1 \\ \omega^{k \times 1} \\ \vdots \\ \omega^{k \times (D-1)} \end{pmatrix} \in \mathbb{C}^D$$

This leads to the following result for the inner product:

$$\langle \mathbf{f}_k, \mathbf{f}_l \rangle = \mathbf{f}_k^\dagger \mathbf{f}_l = \frac{1}{D} \sum_{n=0}^{D-1} \omega^{nk} \overline{\omega^{nl}} = \frac{1}{D} \sum_{n=0}^{D-1} (\omega^{k-l})^n$$

From here it is easy to see that if  $k = l$ , the result is 1. If that is not the case, we notice that  $\frac{1}{D} \sum_{n=0}^{D-1} (\omega^{k-l})^n$  is a geometric series, therefore we expand the sum yielding:

$$\frac{1}{D} \sum_{n=0}^{D-1} (\omega^{k-l})^n = \frac{1}{D} \frac{\omega^{D(k-l)} - 1}{\omega^{k-l} - 1} = 0,$$

with  $\omega^{D(k-l)} - 1 = 0$  because  $\omega$  is a  $D$ -th root of unity (and therefore  $\omega^{D(k-l)} = (\omega^D)^{k-l} = (+1)^{k-l} = +1$ ). ■

Now that we have established that  $DFT_D$  is unitary we can think about how to implement this as a combination of simple unitary quantum gates. Before doing that we want to shown an easy quantum circuit example and calculate the Fourier transform of one of our two basis states, when only having one qubit available.

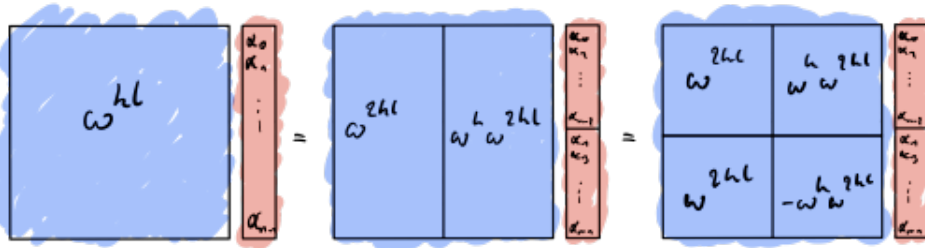
**Example 10.8 (computing quantum Fourier transform).** Let us now calculate the (quantum) Fourier transform of  $|0\rangle = (1 \ 0)^T$ . We have already shown that  $DFT_1 = \mathbf{H}$ . Therefore,

$$|\widehat{0}\rangle = \mathbf{H}|0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = |+\rangle. \quad \blacksquare$$

Due to  $\mathbf{DFT}_D$  being unitary, we have an exponentially large matrix, which can be used in straight-forward calculations, but we can do better than that. Looking really hard at our  $\mathbf{DFT}_D$  matrix one can see that we have certain symmetries that we can use to our advantage. The first step is to separate our matrix into even and odd columns. Similarly, we separate the vector that is to be transformed as well. All we did was basically re-organize the basis vectors. Furthermore, due to  $\omega^{\frac{D}{2}} = -1$  we have

$$\omega^{k+\frac{D}{2}} = -\omega^k.$$

Now if  $\omega$  is the primitive  $D$ -th root of unity, then  $\omega^2$  is the primitive  $\frac{D}{2}$ -th root of unity. Therefore, the matrices whose  $k, l$ -th entry is  $\omega^{2kl}$  are just  $\mathbf{DFT}_{D/2}$ . Now we can write  $\mathbf{DFT}_D$  as four smaller matrices  $\mathbf{DFT}_{D/2}$ . Turns out the calculation then goes from one application of  $\mathbf{DFT}_D$  to two applications of  $\mathbf{DFT}_{D/2}$ . Which can also turn into a total of four applications of  $\mathbf{DFT}_{D/4}$ . As long as  $D = 2^n$  for some  $n$ , we can break this down into  $N$  calculations of  $\mathbf{DFT}_1$ .



**Figure 10.1** Illustration on how symmetries can be used to break up a large DFT matrix-vector multiplications into smaller chunks.

### 10.2.1 Quantum implementation

Let us now move on to the main part of today's lecture. The action of the DFT can be implemented by a remarkably short and elegant quantum circuit. This realization of the DFT only works in  $2^n$  dimensions ( $n$  qubits) and is known as the *quantum Fourier transform* (QFT).

**Theorem 10.9 (quantum Fourier transform).** We can implement the functionality of a  $2^n$  discrete Fourier transform on  $n$ -qubits with  $n^2$  one- and two-qubit gates.

quantum Fourier transform

Now the question arises, how does one implement that in a quantum algorithm? We need to separate the odd and even terms, with the latter being the one where the least significant bit is 0. Therefore, we can apply the  $\mathbf{DFT}_{D/2}$  to both the even and odd parts together. We do this by applying it to the  $m - 1$  most significant bits, recombining these two sets appropriately is done by applying the Hadamard gate to the least significant bit.

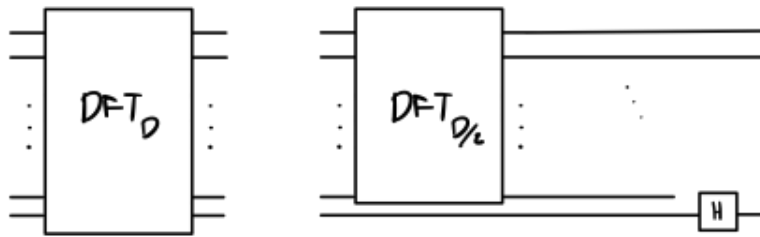


Figure 10.2  $QFT_{D/2}$  and a Hadamard gate

The next step is to apply the phase multiplication. We use the controlled phase shift, where the least significant bit is our control qubit, to multiply only the odd terms by the shift without doing anything to the even terms. The phase associated with each shift is equal to  $\omega^{2^n}$ .

constructing  $QFT$  circuit with one- and two-qubit gates

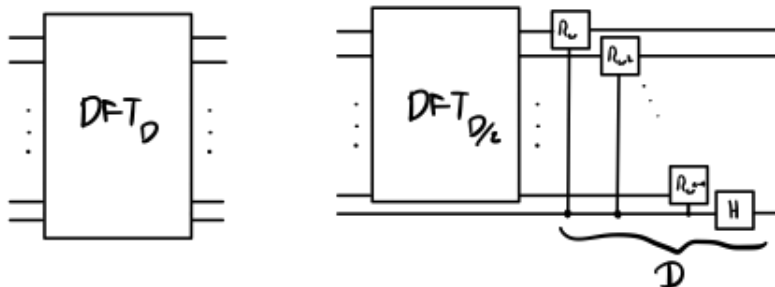


Figure 10.3  $DFT_D$  is reduced to  $DFT_{D/2}$  and  $D$  additional gates

**Example 10.10 (construct  $QFT$  circuit for three qubits).** Let us now construct the quantum Fourier transformation for three qubits together. We turn  $DFT_3$  into  $DFT_2$ , a Hadamard gate and two controlled phase rotations. A controlled phase rotation with angle  $\theta$  is a 2-qubit gate whose action is characterized by the following  $4 \times 4$  matrix:

$$CR_{\theta} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\theta} \end{pmatrix}$$

Then we turn  $DFT_2$  into  $DFT_1 = H$  and into 1 additional controlled rotation.

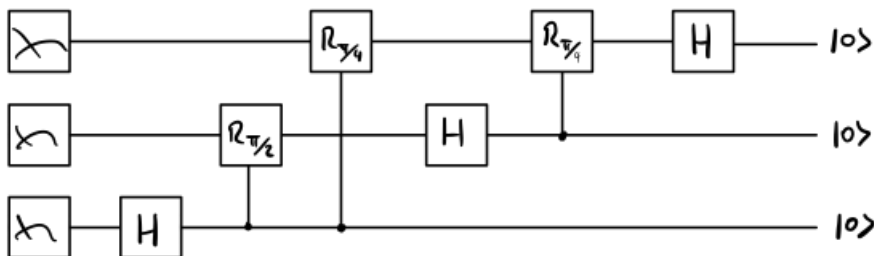


Figure 10.4 Quantum circuit for the DFT with 3 qubits.

In general the  $n$  qubit quantum Fourier transform can be implemented in the fashion shown in Fig. 10.5 The inverse of the quantum Fourier transform is

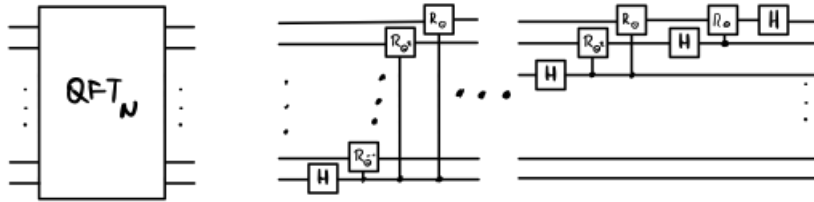


Figure 10.5  $n$ -qubit circuit with only one- and two-qubit gates

then constructed in the usual fashion. The order of the gates, from right to left, is inverted. Furthermore, the controlled phase rotations are replaced by their inverse,  $R_{\theta^k}^\dagger$ . Hadamard gates are not replaced due to them being their own inverse.

$QFT$  and  $QFT^{-1}$

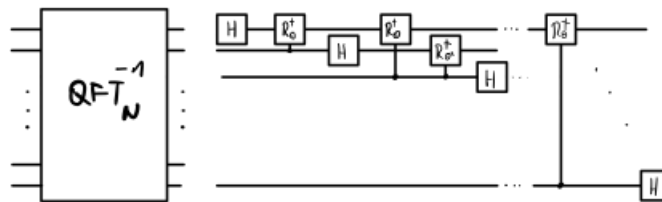


Figure 10.6  $n$ -qubit circuit of the inverse quantum Fourier transform

$$CR_{\theta^k}^\dagger = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{-i\theta} \end{pmatrix}$$

**Exercise 10.11** Construct the four-qubit quantum circuit that implements  $QFT_4$  with only one- and two-qubit gates.

### 10.2.2 Fast (classical) Fourier transform

We have just constructed an  $n$ -qubit circuit that implements the functionality of a  $2^n$ -dimensional Fourier transform. Our circuit consists either of a single layer comprised of single-qubit Hadamard gates or controlled phase gates, with the least significant qubit being the controlling one. The circuit contains  $n(n + 1)/2$  layers. Each layer contains a single Hadamard gate or a controlled phase gate. This reformulation is desirable when we attempt to simulate this circuit on classical hardware. Indeed,

from  $\mathcal{O}(D^2) \rightarrow \mathcal{O}(D \log_2^2(D))$

$$W = \underbrace{\left( \mathbb{I}^{\otimes(n-1)} \otimes H \right) \times \left( \mathbb{I}^{\otimes(n-2)} \otimes CR_x \right) \times \dots \times \left( H \otimes \mathbb{I}^{\otimes(n-1)} \right)}_{n(n+1)/2 \text{ matrix products}}$$

and each matrix on the RHS is sparse. This observation allows us to decompose the matrix-vector multiplication  $Wx$  with  $x \in \mathbb{C}^{2^n}$  into a sequence of  $n(n+1)/2$

sparse matrix-vector multiplications:

$$\mathbf{y}_0 = \mathbf{x}, \mathbf{y}_1 = \left(\mathbf{H} \otimes \mathbb{I}^{\otimes(n-1)}\right) \mathbf{y}_0, \dots, \mathbf{y}_{n-1} = \left(\mathbb{I}^{\otimes(n-1)} \otimes \mathbf{H}\right) \mathbf{y}_{n-2}.$$

Each matrix-vector multiplication only costs about  $2 \times 2^n$  resources, because the matrix involved is extremely sparse. This produces a total cost of  $n^2 \times 2^n$  operations to compute  $\mathbf{W}\mathbf{x}$  – a result that looks more impressive when we set  $D = 2^n$  and  $n^2 = \log_2^2(D)$ .

**Theorem 10.12 (Fast Fourier Transform).** In dimension  $D = 2^n$ , the fast Fourier transform admits a fast matrix-vector multiplication. For any  $\mathbf{x} \in \mathbb{C}^D$ , we can compute  $\mathbf{W}_D \mathbf{x}$  with (order)  $D \log_2^2(D)$  operations.

fast Fourier transform

It is even possible to further improve the cost to order  $D \log_2(D)$  operations only. Note, however, that (order)  $D \log_2^2(D)$  is already *much* better than the naive cost of  $D^2$  associated with naive matrix-vector multiplication. This was achieved by using the sparsity. This might not be obvious by just looking at the matrix displayed in Theorem 10.4. Breaking this down into basic quantum gates revealed that we can rewrite this matrix as a product of  $n(n+1)/2$  sparse matrices. Fast transformations, like the FFT, are the backbone of signal processing as we know it.

**Example 10.13 (discrete and quantum Fourier transform in audio processing).** Let us consider that we have an audio signal (44 kHz) with  $4.4 \times 10^5$  data points. Using the  $\mathbf{DFT}_D$  for the transformation would yield us around  $10^{11}$  operations. Our  $\mathbf{QFT}_D$  ansatz would yield  $10^7$  operations with the best-case scenario being  $10^6$  operations. Were we to calculate one operation a second the difference would be going from 3169 years to 116 days. ■

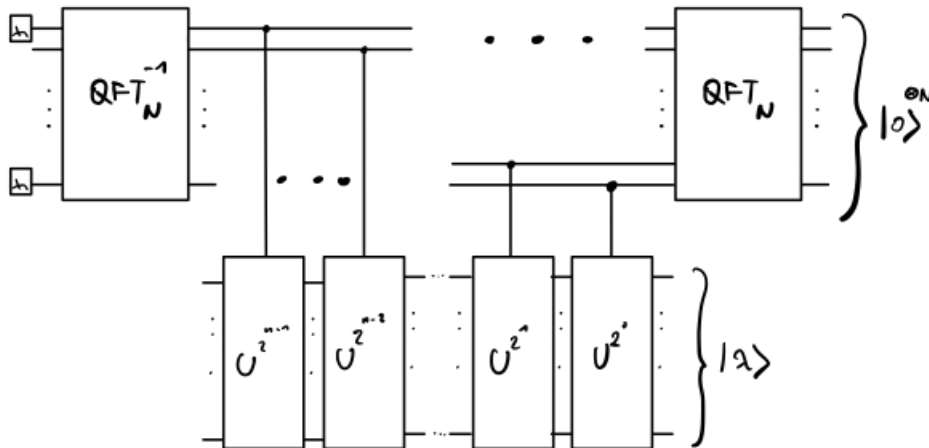
### 10.3 QFT as a subroutine - Quantum phase estimation

Now that we have established a quantum circuit with only one- and two-qubit gates for the quantum Fourier transform, let us take a look at the phase estimation problem, where the QFT is used as a subroutine. There, the task is to approximate a certain eigenvalue<sup>1</sup>  $\exp(i2\pi\theta)$  with  $\theta \in [0, 2\pi)$ . In order to achieve this, we will need a quantum circuit that implements the unitary  $\mathbf{U}$  and a quantum state  $|\lambda\rangle$  that obeys

$$\mathbf{U}|\lambda\rangle = e^{2\pi i\theta} |\lambda\rangle. \quad (10.1)$$

In the terminology of linear algebra, the state vector representation of  $|\lambda\rangle$  forms an eigenvector of  $\mathbf{U}$  with eigenvalue  $\exp(i2\pi\theta)$ .

<sup>1</sup>Eigenvalues of unitary matrices can be complex valued, but they must have modulus one, (because  $\mathbf{U}^\dagger \mathbf{U} = \mathbb{I}$ )



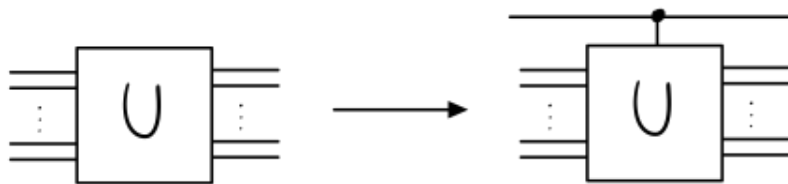
**Figure 10.7** Quantum circuit for the quantum phase estimation: this circuit acts on  $(N + n)$  qubits, features two quantum Fourier transforms, as well as  $N$  conditional applications of the unitary  $U$ .

**Theorem 10.14 (quantum phase estimation).** Suppose that we can implement a  $n$ -qubit unitary  $U$  and can also generate a quantum state  $|\lambda\rangle$  that obeys Eq. (10.1). Then, we can embed these into a larger  $(n + N)$ -qubit circuit that approximates the associated eigenvalue  $\exp(i2\pi\theta)$  up to  $N$  digits. This circuit is displayed in Fig. 11.1 and prominently features the QFT.

quantum phase estimation

Quantum phase estimation is a very useful meta-strategy in quantum computing. Let us now analyze step by step what's going on.

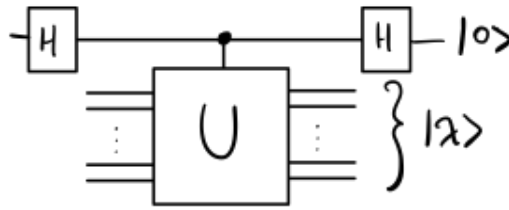
**Example 10.15 (basic quantum phase estimation example).** Let us build some intuition on what it is that we are trying to achieve with the quantum phase estimation algorithm. As already mentioned in the previous theorem we have access to a unitary matrix  $U$ . We can use this quantum circuit to create a controlled- $U$  operation. We can do this by adding a control qubit and then



**Figure 10.8** Creating a controlled operation from a quantum circuit.

replacing every gate with a controlled version of that gate. Now let us consider the circuit shown in Fig. 10.9 The initial state of the circuit is given by

$$|\pi_0\rangle = |\lambda\rangle|0\rangle.$$



**Figure 10.9** One qubit phase estimation.

The Hadamard gate on the zero state turns this into

$$|\pi_1\rangle = |\lambda\rangle|+\rangle = \frac{1}{\sqrt{2}}|\lambda\rangle|0\rangle + \frac{1}{\sqrt{2}}|\lambda\rangle|1\rangle.$$

Now the controlled operation of our unitary matrix is applied to that, but only if the control qubit is one, yielding

$$|\pi_2\rangle = \frac{1}{\sqrt{2}}|\lambda\rangle|0\rangle + \frac{1}{\sqrt{2}}|\mathbf{U}(\lambda)\rangle|1\rangle.$$

Now we can use the previous theorem, assuming that  $|\lambda\rangle$  is an eigenvector of the unitary matrix, we can rewrite this in the following form

$$|\pi_2\rangle = \frac{1}{\sqrt{2}}|\lambda\rangle|0\rangle + \frac{1}{\sqrt{2}}e^{2\pi i\theta}|\lambda\rangle|1\rangle = |\lambda\rangle \otimes \left( \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}e^{2\pi i\theta}|1\rangle \right).$$

Finally the second Hadamard gate is applied which results in the state

$$|\pi_3\rangle = |\lambda\rangle \otimes \left( \frac{1 + e^{2\pi i\theta}}{2}|0\rangle + \frac{1 - e^{2\pi i\theta}}{2}|1\rangle \right).$$

Measuring this now yields the outcomes for 0 and 1 with the following probabilities

$$p_0 = \left| \frac{1 + e^{2\pi i\theta}}{2} \right|^2 = \cos^2 \pi\theta$$

$$p_1 = \left| \frac{1 - e^{2\pi i\theta}}{2} \right|^2 = \sin^2 \pi\theta$$

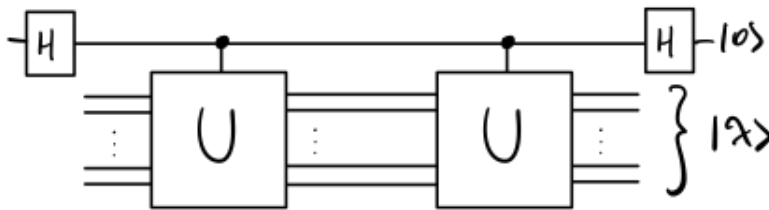
The following table shows results for some  $\theta$

$\theta$	$\cos^2 \pi\theta$	$\sin^2 \pi\theta$
0.0000	1.0000	0.0000
0.1250	0.8536	0.1464
0.2500	0.5000	0.5000
0.5000	0.0000	1.0000

These two probabilities always sum up to one. Only in the case  $\theta = 0$  or  $\theta = 0.5$  we know for a fact what the result of the circuit is without an error. If we were to write  $\theta$  in binary notation and round it off to one bit after the binary point we would have a number like  $0.a$  with  $a$  being either 0 if  $\theta = 0$  or it being 1 if  $\theta = 0.5$ . If it is any other number, there is room for error, but the closer we get to either 0 or 0.5 the smaller it gets. ■

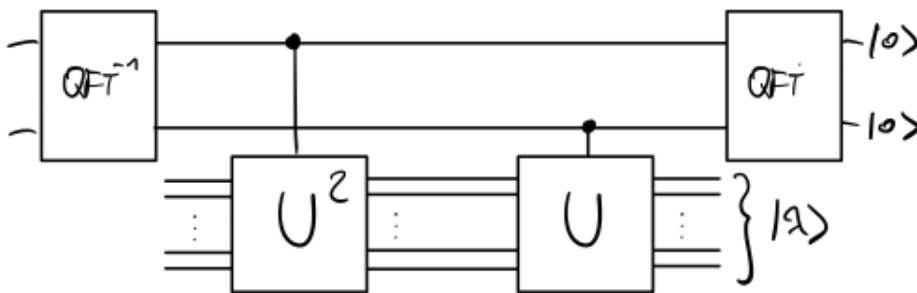
Now the question arises, how can we learn more about  $\theta$ ?

*Proof. (Sketch)* The circuit above approximates  $\theta$  to a single bit of accuracy, which is fine in some cases, but in others it is not. For example Shor's algorithm needs a lot more than that. The first idea one might have is to double the phase, meaning to replace our controlled- $U$  operation with two copies of that operation. We are effectively performing the same calculation just replacing



**Figure 10.10** One qubit phase estimation with two copies of the controlled operation.

$\theta$  with  $2\theta$ . If now we can represent  $\theta = 0.a_1a_2a_3\dots$  then  $2\theta = a_1.a_2a_3\dots$ . And because  $\theta = 1$  is equal to  $\theta = 0$  on the unit circle, we see that  $a_1$  has no influence on our probabilities. We are now getting a guess for the second bit after the binary point. Now we can combine the estimate for the first and second bit to a new two-qubit quantum circuit which has the structure shown in Fig. 10.11



**Figure 10.11** Two qubit phase estimation.

Repeating the aforementioned steps again. We double the number of controlled- $U$  operations, doubling the angle  $2\theta$  in the process to  $4\theta$ , we can get a guess for the third bit after the binary point. Repeating this step more often yields us



better and closer results, but the more bits we want to guess, the more qubits we have to add to our final quantum circuit and the bigger our  $QFT$  block has to be. ■

## Problems

**Problem 10.16** Find the Fourier transform for  $D = 4$ .

**Problem 10.17** Construct the four-qubit quantum circuit that implements  $QFT_4$  with only one- and two-qubit gates.

**Problem 10.18** Use quantum phase estimation to approximately count the ratio of positive answers in a Boolean search problem. Note that this closes a loophole we encountered when discussing amplitude amplification in the previous lecture. **Hint:** the unitary  $U$  is closely related to one amplitude amplification block. The state  $|\lambda\rangle$  is closely related the uniform superposition  $|\omega\rangle$ .

# 11. Shor's algorithm for integer factorization

Date: 17 January 2024

## 11.1 Motivation: hard instances of integer factorization

Today, we finally have all the pieces in place to properly discuss the most prominent quantum algorithm: *Shor's algorithm* for integer factorization from 1994. *Integer factorization* is the problem of decomposing a (large) integer  $N$  into a product of smaller numbers that are all prime. The first algorithms to solve integer factorization date back to Fibonacci in the 1202. Methods like trial division work well for products of many small primes. But, they become really expensive if  $N$  is the product of two distinct prime numbers:

$$N = p \times q \quad \text{where } p, q \text{ prime.} \quad (11.1)$$

In this worst case, trial division may require up to  $\sqrt{N}$  different attempts at division – a number that scales exponentially in the bit length  $n = \lfloor \log_2(N) \rfloor + 1$  of  $N$ . Even the best algorithms known to date scale like

$$O\left(\exp\left(c \times (n \log^2(n))^{1/3}\right)\right),$$

which is still exponential in  $n^{1/3}$ . The apparent hardness of such integer factorization problems is not only a curse, but also a blessing. Hardness of factoring forms the basis of several cryptographic primitives, most notably RSA.

The main result of today is Shor's algorithm for integer factorization from 1994. This is really a hybrid classical-quantum approach, where the bulk is fully classical. It does, however, use a  $(3n + 1) = O(n)$ -qubit architecture to (exponentially) speed up one crucial subroutine. This is achieved by cleverly employing quantum phase estimation – the main topic of the last lecture.

### Agenda:

- 1 motivation: hard integer factorization instances
- 2 (classical) reduction of integer factorization to order finding
- 3 (quantum) algorithm for order finding
- 4 synopsis: Shor's algorithm

best classical factorization scaling is exponential in  $n^{1/3}$

The remainder of today's lecture discusses different aspects of this algorithm. In Sec. 11.2 we use modular arithmetic to reduce the (classically hard) problem of integer factorization to another (classically hard) problem that looks very different: finding the order of an exponential modulo  $N$ . Subsequently, we adapt quantum phase estimation to solve this order finding problem much faster than the best-known classical procedure. This will require  $(3n + 1)$  qubits, a total of  $O(n^3)$  elementary one- and two-qubit gates and (possibly) a few rounds of repetitions to guarantee that we read out the correct solution by measuring the first  $(2n + 1)$  qubits. This is the content of Sec. 11.3. Finally, Sec. 11.4 combines both parts and states Shor's algorithm in its full glory.

## 11.2 Reducing Integer Factorization to order finding

Let us now present a fully classical analysis that reduces integer factorization to another problem called order finding.

### 11.2.1 The order finding problem

Let us first introduce a couple of concepts from arithmetic. For two positive integers  $a, N$ , the *greatest common divisor* (*gcd*) is the largest number that divides both  $a$  and  $N$ . We denote it by writing  $\text{gcd}(a, N)$ . For example,  $\text{gcd}(8, 12) = 4$  and  $\text{gcd}(11, 27) = 1$ . Note that we can compute gcd's efficiently on a classical computer. Euclid's algorithm, for instance, runs in time  $O(n^2)$ , where  $n$  is the bit length of  $\max\{a, N\}$ .

greatest common  
divisor (gcd)

Let us now move on to review the basics of *modular arithmetic* (think: binary math, but extended to  $N$ -ary number systems). For a positive integer  $N$ , we define

$$\mathbb{Z}_N = \{0, 1, 2, \dots, N - 1\}.$$

For example  $\mathbb{Z}_2 = \{0, 1\}$  (binary alphabet),  $\mathbb{Z}_3 = \{0, 1, 2\}$  (ternary alphabet) and  $\mathbb{Z}_{16} = \{0, \dots, 15\}$  (integer representation of the hexadecimal alphabet). Every  $\mathbb{Z}_N$  is a set of integers which we can endow with arithmetic operations modulo  $N$ :

arithmetic modulo  $N$

$$x + y \bmod N \quad \text{and} \quad x \times y \bmod N.$$

For instance, the following modular relations should be very familiar for computer scientists:

$$\begin{aligned} 1 \times 1 \bmod 2 &= 1 = 1 \bmod 2, \\ 2 \times 1 \bmod 2 &= 1 + 1 \bmod 2 = 0 \bmod 2, \\ 3 \times 1 \bmod 2 &= 1 + 1 + 1 \bmod 2 = 1 \bmod 2, \\ &\vdots \end{aligned}$$

**Example 11.1** (first 10 digits modulo  $N$  for  $N = 3, 4, 5$ ). The first ten digits ( $\mathbb{Z}_{10} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ ) assume the following values in different modular

arithmetic frameworks:

$$N = 3 : \mathbb{Z}_{10} \mapsto \{0, 1, 2, 0, 1, 2, 0, 1, 2, 0\},$$

$$N = 4 : \mathbb{Z}_{10} \mapsto \{0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2\},$$

$$N = 5 : \mathbb{Z}_{10} \mapsto \{0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0\}.$$

Notice the repeating patterns that occur if we represent 10 numbers in arithmetic modulo  $N < 10$ . This *periodicity* is a general feature of modular arithmetic that will become crucial later on. ■

The set  $\mathbb{Z}_N$  from Eq. (II.2.1) is guaranteed to contain elements  $a \in \mathbb{Z}_N$  that don't share a gcd with  $N$ . We accumulate all of them and define

$$\mathbb{Z}_N^* = \{a \in \mathbb{Z}_N : \gcd(a, N) = 1\}.$$

These numbers are special, because each  $a \in \mathbb{Z}_N^*$  and  $N$  are coprime. For instance,

$$\mathbb{Z}_{15}^* = \{1, 2, 4, 7, 8, 11, 13\}, \quad (\text{II.2})$$

$$\mathbb{Z}_{35}^* = \{1, 2, 3, 4, 6, 8, 9, 11, 12, 13, 16, 17, 18, 19, 22, \\ 23, 24, 26, 27, 29, 31, 32, 33, 34\}. \quad (\text{II.3})$$

We are now ready to present a key fact from modular arithmetic on which we build our integer factorization algorithm.

**Fact 11.2 (order of  $a$  in  $\mathbb{Z}_N^*$ ).** Fix  $N \in \mathbb{N}$ . Then, for every  $a \in \mathbb{Z}_N^*$  there must exist a positive integer  $r$  such that  $a^r = 1 \pmod N$ . The smallest such  $r$  is called the *order of  $a$  in  $\mathbb{Z}_N^*$* . ■

order finding problem

For this fact to hold, it is important that  $a \in \mathbb{Z}_N^*$  to begin with. If  $\gcd(a, N) \neq 1$ , there can be no order to begin with (why?). Note that the order  $r$  of  $a$  in  $\mathbb{Z}_N^*$  is closely related to periodicities, like the ones we saw above. More precisely, it is the period of the function  $f_{a,N}(x) = a^x \pmod N$ . Indeed,

$$f_{a,N}(x+r) = a^{x+r} \pmod N = a^x \times a^r \pmod N = a^x \times 1 \pmod N = f_{a,N}(x).$$

Let us familiarize ourselves with this concept by executing two concrete example calculations.

**Example 11.3 (order of  $a = 13$  in  $\mathbb{Z}_{15}^*$ ).** Let us consider the set  $\mathbb{Z}_{15}^*$  from Eq. (II.2) and choose  $a = 13$ . We can then evaluate the order by trial search over candidate exponents  $r = 1, 2, 3, \dots$ :

$$13^1 = 13 \pmod{15}, \quad 13^2 = 169 = 4 \pmod{15}, \\ 13^3 = 2197 = 7 \pmod{15}, \quad 13^4 = 28561 \pmod{15}.$$

So, the order of  $a = 13$  in  $\mathbb{Z}_{15}^*$  is  $r = 4$ . ■

**Example 11.4 (order of  $a = 2$  in  $\mathbb{Z}_{35}$ ).** Let us consider the set  $\mathbb{Z}_{15}^*$  from Eq. (11.3) and choose  $a = 2$ . We can then evaluate the order by trial search over the candidate exponent. Since  $2^1, 2^2, 2^3, 2^4, 2^5 < 35 = N$ , we can start our search at  $r = 6$ :

$$\begin{aligned} 2^6 &= 64 \bmod 35 = 29 \bmod 35, & 2^7 &= 128 \bmod 35 = 23 \bmod 35, \\ 2^8 &= 256 \bmod 35 = 11 \bmod 35, & 2^9 &= 512 \bmod 35 = 22 \bmod 35, \\ 2^{10} &= 1024 \bmod 35 = 9 \bmod 35, & 2^{11} &= 2048 \bmod 35 = 18 \bmod 35, \\ 2^{12} &= 4096 \bmod 35 = 1 \bmod 35. \end{aligned}$$

This allows us to conclude that the order of  $a = 2$  in  $\mathbb{Z}_{35}^*$  is  $r = 12$ . We can also see that the cost of trial search over possible exponents can grow very quickly once  $N$  becomes somewhat large. ■

These examples highlight that order finding is possible. But the number of trial exponentiations can grow quickly once  $N$  becomes (somewhat) large. This increase in cost seems to be a fundamental problem. To date, no efficient classical algorithm is known that can determine the order of  $a$  in  $\mathbb{Z}_N^*$  with polynomial resources in  $n = \lfloor \log_2(N) \rfloor + 1$ . We will, however, develop an efficient quantum algorithm in Sec. 11.3 below.

### 11.2.2 Solving integer factorization via order finding

Let us now present a (classical) protocol that solves the (worst case of) integer factorization, see Eq. (11.1), via the order finding problem defined in Fact 11.2. Formally speaking, this is a proper reduction: every step in our procedure will be (very) efficient – with the sole exception of order finding. We refer to Algorithm 11.1 for details.

**Theorem 11.5 (integer factorization via order finding).** Suppose that  $N = p \times q$  is an odd product of two distinct primes. Then, Algorithm 11.1 is guaranteed to terminate and output (at least) one factor:  $f_0 \in \{p, q\}$  or  $f_1 \in \{p, q\}$ .

(classical) reduction from integer factorization to order finding

This result implies that order finding is at least as difficult as (our special case of) integer factorization. Typically, such reduction arguments are used to argue that a problem is difficult. Here, we will do the opposite. We will develop an efficient algorithm for order finding in order to argue that integer factorization becomes cheap if we had access to a quantum computer.

*Proof of Theorem 11.5.* The input  $N$  allows us to initialize arithmetic modulo  $N$  and sample  $a \in \mathbb{Z}_N$  uniformly. Although very unlikely, it can happen that  $a$  shares a nontrivial greatest common divisor with  $N$ . In this case,  $N = p \times q$  and  $p, q$  prime demands that this gcd is either  $p$  or  $q$ . In other words: we have found a factor by chance!

Let us now focus on the much more likely case, where  $\gcd(a, N) = 1$ . In this case, the order of  $a$  in  $\mathbb{Z}_N^*$  is well-defined. We can identify the smallest  $r$  such that  $a^r = 1 \bmod N$  (e.g. via brute force search). Two situations can

**Algorithm 11.1** *Integer Factorization via order finding*


---

**Input:**  $N \in \mathbb{N}$  ▷ integer to be factorized  
**Output:** two positive integers  $f_0, f_1 \in \mathbb{N}$  ▷ pair of candidate factors

```

1 while
2   dochoose  $a \in \mathbb{Z}_N = \{0, 1, \dots, N\}$  randomly
3   if  $\gcd(a, N) > 1$  then ▷ we found a factor by chance
4     success, set  $f_0 = \gcd(a, N)$ ,  $f_1 = 1$  and break while loop
5   else if  $f = \gcd(a, N) = 1$  then
6     identify smallest  $r$  s.t.  $a^r = 1 \pmod N$  ▷ find order of  $a$  in  $\mathbb{Z}_N^*$ 
7     if  $a$  is odd then
8       failure, move to step 2
9     else if  $a$  is even then
10      set  $x = a^{r/2} \pmod N$  and define  $(x + 1), (x - 1)$ 
11      if  $(x \pm 1) = 0 \pmod N$  then
12        failure, move to step 2
13      else if  $(x \pm 1) \neq 0 \pmod N$  then
14        success, compute  $f_0 = \gcd(x + 1, N)$  and  $f_1 = \gcd(x - 1, N)$ 
        and break while loop
15 output  $f_0$  and  $f_1$ 

```

---

arise:  $r$  is odd or  $r$  is even. If  $r$  is odd, the attempt fails and we re-try with a different  $a \in \mathbb{Z}_N$ . Else if  $r$  is even, then  $r/2$  is a positive integer and we can define  $x = a^{r/2} \pmod N$ . The modular version of  $(a + b)(a - b) = a^2 - b^2$  then ensures

$$(x + 1) \times (x - 1) \pmod N = a^r - 1 \pmod N = 0 \pmod N,$$

because  $a^r = 1 \pmod N$ . Returning to ordinary arithmetic, this is equivalent to

$$(x + 1) \times (x - 1) = kN \quad \text{for some } k \in \mathbb{N}. \quad (11.4)$$

This starts to look very close to a factorization. However, this display becomes completely trivial if either  $(x + 1)$  or  $(x - 1)$  is itself equal to a multiple of  $N$ . The last if-condition in line 11 of Alg. 11.1 checks for precisely this possibility. If  $(x \pm 1) = 0 \pmod N$ , then we have learned nothing new and restart. Else if  $(x \pm 1) \neq 0 \pmod N$ , Eq. (11.4) must contain nontrivial information about the factors. Using  $N = pq$ , we rewrite it as  $(x + 1)(x - 1) = kpq$  for some  $k \in \mathbb{N}$  and  $(x \pm 1) \neq k'pq$ . This is only possible if either  $\gcd(x + 1, N) \in \{p, q\}$  or  $\gcd(x - 1, N) \in \{p, q\}$  (or both). ■

So far, Algorithm 11.1 looks rather abstract and difficult to grasp. It is therefore instructive to execute it in small-scale examples.

**Example 11.6 (Factoring  $N = 15$  with Algorithm 11.1).** Suppose that we start with  $a = 13$  which obeys  $\gcd(13, 15) = 1$  and therefore doesn't yet provide any information about the factors of  $N = 15$ . To change this, we use trial

exponentiation to find the order of  $a = 13$  in  $\mathbb{Z}_{15}^*$ . We already did that in Example 11.3:  $r = 4$ . This number is even and we can use it to define  $x = a^{r/2} \bmod 15 = 13^2 \bmod 15 = 4$  and, in turn,  $(x + 1) = 5$ ,  $(x - 1) = 3$ . Neither of these numbers is a multiple of 15, so the last condition is also met and we are guaranteed to learn something about the factors of 15. Indeed,

$$\gcd(x + 1, N) = \gcd(5, 15) = 5 \quad \text{and} \quad \gcd(x - 1, N) = \gcd(3, 15) = 3.$$

Remarkably, we have learned both factors of  $N = 3 \times 5$  in one go. This is even better than the promise from Theorem 11.5. ■

**Example 11.7 (Factoring  $N = 35$  with Algorithm 11.1).** Here, we will also discuss choices of  $a$  for which the algorithm fails. If this happens, we start over with a new choice of  $a \in \mathbb{Z}_{35}^*$ :

Try  $a = 11$  and use brute force to identify the order of 11 in  $\mathbb{Z}_{35}^*$ :  $r = 3$  which is not even and we have to start over.

Try  $a = 2$  and use brute force to identify the order of 2 in  $\mathbb{Z}_{35}^*$ :  $r = 12$  which is even and allows us to define  $x = a^{r/2} \bmod 35 = 2^6 \bmod 35 = 29$ . Fortunately, neither  $(x + 1) = 30$  nor  $(x - 1) = 28$  are themselves multiples of  $N$ . So, computing their gcds with  $N$  reveals at least one factor (and, by chance, we in fact get both of them again):

$$\gcd(x + 1, N) = \gcd(30, 35) = 5 \quad \text{and} \quad \gcd(x - 1, N) = \gcd(28, 35) = 7.$$

■

### 11.3 Efficiently solving order finding on a quantum computer

Section 11.2 provided an alternative approach for (hard instances of) integer factorization. This method is very different from well-known factoring methods, like trial division. It isolates most of the hardness in a single sub-task: find the order of  $a$  in  $\mathbb{Z}_N^*$  (where  $\gcd(a, N) = 1$ ):

$$\text{find } r \in \mathbb{Z}_N \quad \text{such that} \quad a^r = 1 \bmod N.$$

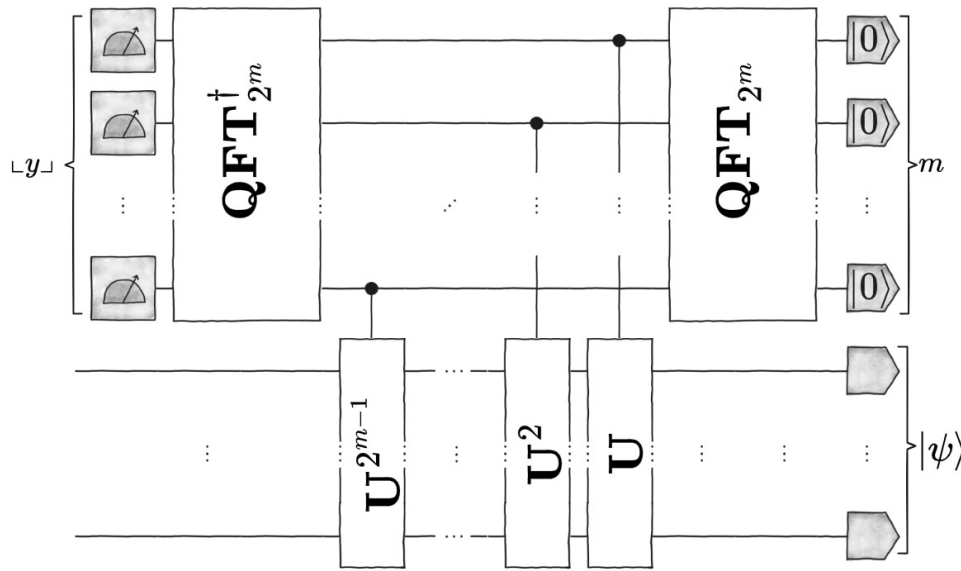
We don't know any classical algorithm for this order finding problem that scales polynomially in the bit length of  $N$ . Trial and error, for instance, may take us (exponentially in  $n$ ) many attempts to find the order. However, unlike the original factoring problem, order finding is much more amenable to a genuine quantum solution. In fact, we can use phase estimation for it. This was Shor's seminal insight in 1994.

#### 11.3.1 Recapitulation: quantum phase estimation

Quantum phase estimation is a versatile quantum circuit architecture that exploits the (very, very fast) quantum Fourier transform. At the heart are a  $n$ -qubit unitary circuit  $\mathbf{U}$  and a  $n$ -qubit state (vector)  $|\psi\rangle$  that obey

$$\mathbf{U}|\psi\rangle = \exp(2\pi i\theta) |\psi\rangle. \tag{11.5}$$

quantum phase estimation for eigenvalue+eigenvector pair of  $n$ -qubit unitary  $\mathbf{U}$



**Figure 11.1** Quantum phase estimation circuit: Let  $\mathbf{U}$  be a  $n$ -qubit unitary and let  $|\psi\rangle$  be a state vector representation of an eigenvector  $\exp(2\pi i\theta)$  of  $\mathbf{U}$ . Then, for  $m \geq 1$ , the advertised  $(m+n)$ -qubit circuit approximates the first  $m$  bits of the (eigen-)phase  $\theta \in (0, 2\pi)$ . More precisely, the  $n$ -bit readout  $\lfloor y \rfloor$  is a bit encoding of  $y \approx 2^m \times \theta$ .

In mathematical terms,  $\mathbf{U}$  reflects a *unitary matrix*,  $|\psi\rangle$  reflects an *eigenvector* and the associated *eigenvalue* is  $\exp(2\pi i\theta)$ . Our task is to approximate the *phase*  $\theta$  up to  $m$  bits of accuracy.

There is a quantum solution to phase estimation. It requires that we can create the state  $|\psi\rangle$  and know a quantum circuit realization for  $\mathbf{U}$ . The associated circuit uses  $(m+n)$  qubits and is displayed in Fig. 11.1. It combines a  $m$ -qubit quantum Fourier transform (QFT) with  $m$  controlled applications of powers of  $\mathbf{U}$  on the lower batch of  $n$  qubits. Finally, we apply a reverse QFT to the first  $m$  qubits and perform a readout. This produces a bit string  $\lfloor y \rfloor \in \{0, 1\}^m$  which we interpret as bit encoding of  $0 \leq y \leq 2^m - 1$ . Remarkably, the fraction  $y/2^m$  is likely to be an approximation of  $\theta$  up to accuracy  $1/2^{m+1}$ .

**Fact 11.8 (performance guarantee for quantum phase estimation).** Suppose that  $\mathbf{U}$  and  $|\psi\rangle$  obey Eq. (11.5) with phase  $\theta$  and choose  $m \in \mathbb{N}$ . Then, with probability at least 40%, the  $(m+n)$ -qubit circuit displayed in Fig. 11.1 produces an outcome  $\lfloor y \rfloor \in \{0, 1\}^m$  which can be viewed as bit encoding of  $y \in \{0, 2^m - 1\}$  that obeys

$$\left| \frac{y}{2^m} - \theta \right| \leq \frac{1}{2^{m+1}}. \quad (11.6)$$

■



Note that this fact says nothing about the cost of executing the phase estimation circuit. This depends on the way we construct  $|\psi\rangle$  and – which is much more severe – how we implement the controlled application of  $\mathbf{U}^{2^k} = \mathbf{U} \times \cdots \times \mathbf{U}$  for  $k = 1, \dots, 2^m - 1$ . This gate count quickly explodes if one is not careful. Fortunately, for Shor's algorithm, a trick will allow us to deal with this issue nicely.

### 11.3.2 Identifying the order parameter in eigenvalues of a simple reversible circuit

In order to build a bridge between (classical) order finding and quantum phase estimation, we need to identify qubit encodings of  $\mathbb{Z}_N$ , the 'right' unitary circuit  $\mathbf{U}$  and a promising eivenvector  $|\psi\rangle$ .

A qubit encoding of  $\mathbb{Z}_N$  is easy to accomplish. Set  $n = \lfloor \log_2(N) \rfloor + 1$  and identify each  $z \in \mathbb{Z}_N$  with its bit encoding  $\lfloor z \rfloor \in \{0, 1\}^n$ . In a second step, we associate such bit encodings with deterministic  $n$ -bit input state vectors:

$n$ -qubit encoding of  $\mathbb{Z}_N$   
( $n = O(\log(N))$ )

$$|z\rangle_N := |\lfloor (z \bmod N) \rfloor\rangle \quad \text{for } z \in \mathbb{Z}_N \text{ and } \lfloor z \rfloor \in \{0, 1\}^n.$$

With a slight abuse of notation, we denote an entire bit string by the number  $z \in \mathbb{Z}_N$  it encodes. Note, moreover, that we can also encode modular arithmetic in these state vector labels. In particular,

$$\begin{aligned} |z + z'\rangle_N &= |(z + z' \bmod N)\rangle_N = |\lfloor (z + z' \bmod N) \rfloor\rangle, \\ |z \times z'\rangle_N &= |(z \times z' \bmod N)\rangle_N = |\lfloor (z \times z' \bmod N) \rfloor\rangle. \end{aligned}$$

This (qu-)bit encoding allows us to represent modular addition and multiplication as classical reversible circuits. For  $a \in \mathbb{Z}_N^*$  fixed, we define

$$\mathbf{A}_a |z\rangle_N = |z + a\rangle_N \quad (\text{addition by } a \text{ modulo } N), \quad (\text{II.7})$$

$$\mathbf{M}_a |z\rangle_N = |a \times z\rangle_N \quad (\text{multiplication by } a \text{ modulo } N). \quad (\text{II.8})$$

for all possible  $z \in \mathbb{Z}_N$ . Modulo some technical fineprint<sup>1</sup>, these operations map  $n$ -bit encodings of  $z \in \mathbb{Z}_N$  to  $n$ -bit encodings of  $z' \in \mathbb{Z}_N$  in a reversible fashion. The reverse of  $\mathbf{A}_a$  is simply  $\mathbf{A}_{(N-a)}$ :

$$\mathbf{A}_{(N-a)} \mathbf{A}_a |z\rangle_N = \mathbf{A}_{(N-a)} |z + a\rangle_N = |(z + a) + (N - a)\rangle_N = |z + N\rangle_N = |z\rangle_N.$$

The reverse operation of  $\mathbf{M}_a$  is much more interesting by comparison. It is  $\mathbf{M}_{a^{r-1}}$ , where  $r$  is the order of  $a \in \mathbb{Z}_N^*$ :

$$\mathbf{M}_{a^{r-1}} \mathbf{M}_a |z\rangle_N = \mathbf{M}_{a^{r-1}} |a \times z\rangle_N = |a^r \times z\rangle_N = |1 \times z\rangle_N = |z\rangle_N.$$

These displays ensure that both  $\mathbf{A}_a$  and  $\mathbf{M}_a$  are classical reversible circuits (and therefore also unitary). What is more, they can be implemented with only  $O(n^2)$  elementary quantum gates.

<sup>1</sup>Whenever  $N \neq 2^n - 1$ , there are  $n$ -bit strings  $y \in \{0, 1\}^n$  that do not describe bit encodings of any  $z \in \mathbb{Z}_N$ . To complete the formal definition of  $\mathbf{A}_a$  and  $\mathbf{M}_a$ , we let both of them act trivially on such input configurations:  $\mathbf{A}_a |y\rangle = |y\rangle$  and  $\mathbf{M}_a |y\rangle = |y\rangle$ .

**Exercise 11.9 (quantum circuit implementation of reversible addition and reversible multiplication).** Sketch how you would construct a quantum implementation of the permutation matrices  $\mathbf{A}_a$  (addition by  $a$  modulo  $N$ ) and  $\mathbf{M}_a$  (multiplication by  $a$  modulo  $N$ ) on  $n = \lfloor \log_2(N) \rfloor + 1$  qubits. Show that the number of elementary gates required does not exceed  $O(n^2)$ .

efficient quantum circuits for addition & multiplication by  $a$  modulo  $N$  ( $O(n^2)$  gates)

We leave the proof as an instructive exercise. Instead, let us emphasize that the multiplication operator  $\mathbf{M}_a$  looks to be connected to the order  $r$  of  $a$  in  $\mathbb{Z}_N^*$  in a nontrivial fashion that we might be able to exploit. In order to make this correspondence precise, we must take a look at the eigenvalues and eigenvectors of  $\mathbf{M}_a$ .

**Proposition 11.10 (eigenvalues+eigenvectors of the modular multiplication operator).** For  $N$ ,  $a \in \mathbb{Z}_N^*$  with  $\gcd(a, N) = 1$ , let  $r$  be the order of  $a$  in  $\mathbb{Z}_N^*$  and set  $\omega_r = \exp(2\pi i/r)$  ( $r$ -th root of unity). Then, the modular multiplication circuit  $\mathbf{M}_a$  defined in Eq. (11.8) has the following  $r$  eigenvalue+eigenvector pairs:

(many) eigenvalues of modular multiplication circuit by  $a$  isolate order  $r$

$$\lambda_j = \omega_r^j = \exp(2\pi i(j/r)) \quad \text{and} \quad |\psi_j\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \omega_r^{-j \times k} |a^k\rangle_N$$

for  $j = 0, \dots, r-1$ .

*Proof sketch.* Let us start with an explicit calculation for the special case  $j = 0$ :

$$\begin{aligned} \mathbf{M}_a |\psi_0\rangle_N &= \frac{1}{\sqrt{r}} (\mathbf{M}_a |1\rangle_N + \mathbf{M}_a |a\rangle_N + \dots + \mathbf{M}_a |a^{r-2}\rangle_N + \mathbf{M}_a |a^{r-1}\rangle_N) \\ &= \frac{1}{\sqrt{r}} (|a\rangle_N + |a^2\rangle_N + \dots + |a^{r-1}\rangle_N + |a^r\rangle_N) \\ &= \frac{1}{\sqrt{r}} (|a\rangle_N + |a^2\rangle_N + \dots + |a^{r-1}\rangle_N + |1\rangle_N) \\ &= (+1) |\psi_0\rangle = \omega_r^0 |\psi_0\rangle_N. \end{aligned}$$

Here, we have used  $|a^r\rangle_N = |1\rangle_N$  which follows from the fact that  $r$  is the order of  $a$  in  $\mathbb{Z}_N^*$  ( $a^r = 1 \pmod{N}$ ). In short,  $\mathbf{M}_a |\psi_0\rangle = \omega_r^0 |\psi_0\rangle$  which is the defining property of an eigenvector  $|\psi_0\rangle$  with eigenvalue  $\omega_r^0$ . Verifying the remaining  $(r-1)$  eigenvalue equations with  $j = 1, \dots, r-1$  can be achieved in a similar fashion (hint: use  $\omega_r^r = 1$ ). We leave it as an exercise. ■

**Exercise 11.11 (eigenvalues and eigenvectors of modular addition operator).** Fix  $N \in \mathbb{N}$ ,  $a \in \mathbb{Z}_N^*$  and let  $r$  be the order of  $a$  in  $\mathbb{Z}_N^*$ . Identify at least  $r$  eigenvector-eigenvalue pairs of the modular addition operator  $\mathbf{A}_a$  defined in Eq. (11.7). **Hint:** take inspiration from our analysis of  $\mathbf{M}_a$  and/or Fourier series with finite periodicity.

### 11.3.3 Approximate eigenvalues of this simple reversible circuit via phase estimation

Proposition 11.10 is noteworthy for the following reason:  $\mathbf{M}_a$  admits a simple reversible circuit (see Lemma 11.9) and the eigenvalue associated with

eigenvector  $|\psi_j\rangle$  is a complex phase that encodes the period we are looking for:

$$\mathbf{M}_a |\psi_j\rangle_N = \exp(2\pi i (j/r)) |\psi_j\rangle_N = \exp(2\pi i \theta_j) |\psi_j\rangle \quad \text{with} \quad \theta_j = j/r.$$

Quantum Phase estimation now looks to be almost tailor-made for this task! But with quantum phase estimation, we always have to be careful that the requirements for approximation accuracy are not too large. Otherwise, the required number  $m$  of readout qubits can explode and the entire procedure becomes infeasible. The following fact provides good news: the number of additional qubits  $m$  can be chosen to linear in the original qubit size  $n$ . This is as good as it gets.

**Fact 11.12 (required accuracy for quantum phase estimation).** Suppose that we can run quantum phase estimation with  $m$  control qubits for  $\mathbf{M}_a$  (unitary matrix) and  $|\psi_j\rangle$  (eigenvector with phase  $\theta_j = j/r$ ), where  $j = 0, \dots, r-1$ :

quantum phase estimation with  $m = O(n)$  qubits suffices to read out order  $r$

- **worst case:**  $j = 0$ ,  $\theta_0 = 0/r = 0$  and we learn nothing about the order whatsoever;
- **best case:**  $j = 1$ ,  $\theta_1 = 1/r$  and we can approximate  $r$  by the closest integer to  $(y/2^m)^{-1} = 2^m/y$ .
- **general case:**  $\theta_j = j/r$  is an actual fraction. We can use a classical algorithm – called *continued fractions* – to efficiently find the closest fraction  $u/v$  in lowest terms satisfying  $u, v \in \mathbb{Z}_N$  (and  $v \neq 0$ ).

Unless  $j = 0$ , setting  $m = \lfloor \log_2(2N^2) \rfloor = 2\lfloor \log_2(N) \rfloor + 1 \leq 2n + 1$  is enough to determine  $j/r$  as fraction with probability (at least) 40%. ■

We refer to standard textbooks for a proof of this fact and a review of the continued fraction algorithm which also allows to recover the order  $r$  itself. Note also that this fact only guarantees a successful approximation with a certain probability of success, namely greater than 40%. So, it can happen that we get unlucky with our quantum phase estimation approximation  $\lfloor y \rfloor$ . This issue can be offset by repeating the entire procedure multiple times and classically checking the observed outcomes  $\lfloor y_k \rfloor$  for consistency (is it really a fraction? is it really minimal? and does the period  $r$  feature?)

Here is another noteworthy feature that renders the quantum phase estimation circuit more efficient than one might naively assume

**Lemma 11.13 (efficient circuit implementation of  $\mathbf{M}_a^{2^k}$ ).** For each  $k = 0, \dots, m$ , we can implement the  $2^k$ -th power  $\mathbf{M}_a^{2^k}$  of  $\mathbf{M}_a$  as  $\mathbf{M}_{a^{2^k}}$ . The latter is another multiplication matrix that can be implemented with only  $O(n^2)$  elementary quantum gates.

*Proof.* The following equation is valid for all  $n$ -bit strings  $z \in \{0, 1\}^n$

$$\mathbf{M}_a^{2^k} |z\rangle_N = \underbrace{\mathbf{M}_a \times \dots \times \mathbf{M}_a}_{2^k \text{ times}} |z\rangle_N = |a^{2^k} z\rangle_N = \mathbf{M}_{a^{2^k}} |z\rangle_N.$$

The technical fineprint<sup>2</sup> from above ensures that this suffices to conclude that the two matrices  $M_a^{2^k}$  and  $M_{a^{2^k}}$  must be identical overall. ■

Lemma 11.13 ensures that we can implement each controlled operation in quantum phase estimation (central part of Fig. 11.1) with only  $O(n^2)$  gates. Since there are  $m = (2n + 1)$  such controlled circuits, this produces a total cost of  $O(n^3)$  gates for the central block. The cost of the two quantum Fourier transforms is only  $O(m^2) = O(n^2)$ , so we obtain a total gate count of  $O(n^3)$  to execute the quantum phase estimation circuit. The only thing missing now, is the preparation of an eigenvector  $|\psi_j\rangle$  (ideally with  $j \neq 0$ ). Explicitly constructing one  $|\psi_j\rangle$  in a deterministic fashion is a daunting task. We can, however, easily prepare the uniform superposition over all possible eigenvectors  $|\psi_j\rangle$ . This is the content of the following lemma.

**Lemma 11.14** Let  $|\psi_j\rangle_N$  be the eigenvectors of  $M_a$  defined in Proposition 11.10. Then, we have

$$\frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} |\psi_j\rangle_N = |1\rangle_N = |0 \cdots 01\rangle.$$

In words: the uniform superposition of all  $n$ -qubit eigenvectors  $|\psi_j\rangle$  is the computational basis state  $|1\rangle_N = |0 \cdots 01\rangle \in \{0, 1\}^n$ .

*Proof.* This follows from an identity that we saw in the last lecture. Summing over all (powers) of an  $r$ -th root of unity produces 0 (amplitudes cancel):

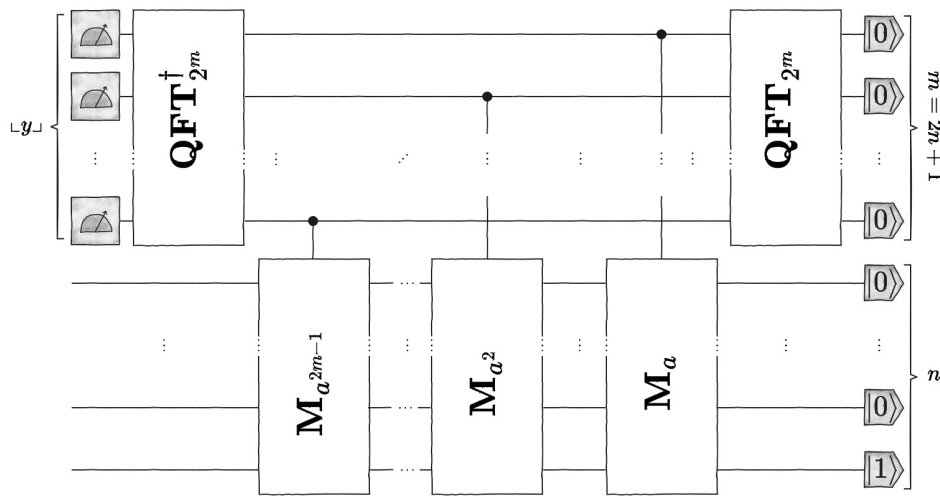
$$\begin{aligned} \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} |\psi_j\rangle_N &= \left( \frac{1}{r} \sum_{j=0}^{r-1} \omega_r^0 \right) |1\rangle_N + \left( \frac{1}{r} \sum_{j=0}^{r-1} \omega_r^{-j} \right) |a\rangle_N \\ &\quad + \cdots + \left( \frac{1}{r} \sum_{j=0}^{r-1} \omega_r^{-j(r-1)} \right) |a^{r-1}\rangle_N \\ &= 1 \times |1\rangle_N + 0 \times |a\rangle_N + \cdots + 0 \times |a^{r-1}\rangle_N \\ &= |1\rangle_N. \end{aligned}$$

efficient implementation of phase estimation circuit ( $O(n^3)$  gates)

efficient preparation of (a superposition of) eigenvectors

This is the last ingredient that we needed to implement order finding on a quantum computer: we can simply initialize the second  $n$ -qubit register in  $|0 \cdots 01\rangle = |1\rangle_N$  and view this deterministic bit string as a uniform superposition of all  $r$  eigenvectors  $|\psi_j\rangle$ . Linearity of quantum mechanics (matrix-vector multiplication) then ensures that the remaining  $(n + m)$ -qubit circuit produces a superposition of all  $r$  phase estimation protocols – one for each eigenvector. Reading out the first  $m$  qubits then collapses this superposition into one particular branch, i.e. one particular  $j = 0, \dots, r - 1$ . Out of these  $r$  possible trajectories, only  $j = 0$  is completely useless. So, with probability  $(r - 1)/r \geq 1/2$ , we are in a position to extract useful information about  $r$ . This, however, is contingent on the quantum phase estimation subroutine

<sup>2</sup>This is a continuation of the previous footnote. We let every modular multiplication circuit act trivially on input configurations  $y$  that don't encode elements  $z \in \mathbb{Z}_N$ :  $M_a|y\rangle = |y\rangle = M_{a^{2^k}}|y\rangle$ .



**Figure 11.2** *quantum part of Shor's algorithm*: This adaptation of quantum phase estimation is designed to solve the order finding problem for  $a$  in  $\mathbb{Z}_N$ . The circuit requires  $3n + 1$  qubits, where  $n = \lfloor \log_2(N) \rfloor + 1$ , and initializes them in the bit string  $0 \cdots 01$ . Subsequently, we apply a Quantum Fourier Transform on the first  $m = (2n + 1)$  gates and follow it up with  $m$  controlled applications of different arithmetic multiplication circuits  $M_{a^j}$ . Finally, we apply an inverse Quantum Fourier Transform on the first  $(2m + 1)$  qubits and read them out to obtain a bit string representation of  $y \in \{0, 2^m - 1\}$  that is likely to encode the order  $r$  of  $a$  in  $\mathbb{Z}^n$ :  $y/2^m \approx j/r$  for some  $j = 0, \dots, r - 1$  with 'reasonable' probability.

succeeding. According to Theorem 11.8, this happens with probability at least 40%. Total odds of about  $(r - 1)/r \times 0.4 \geq 0.2$  are not bad at all. A constant number of repetitions should suffice to learn  $j/r$  and, by extension, the period  $r$ . This realization deserves a prominent display and even its own synopsis section.

#### 11.4 Synopsis: implementation of Shor's algorithm

**Theorem 11.15 (quantum order finding).** Fix  $N \in \mathbb{N}$ , set  $n = \lfloor \log_2(N) \rfloor + 1$  and let  $a \in \mathbb{Z}_N$  be such that  $\gcd(a, N) = 1$ . Then, the quantum circuit displayed in Fig. 11.2 is comprised of  $O(n^3)$  elementary one- and two-qubit gates and identifies the smallest integer  $r$  such that  $a^r = 1 \pmod N$  with a constant probability of success.

To paraphrase: this quantum circuit solves the order finding problem with only  $O(n^3)$  quantum resources!

order finding via quantum phase estimation works with gate count  $O(n^3)$

Note that this circuit has a lot of structure and can be divided into three qualitatively different blocks: a quantum Fourier transform (one part of the qubits) followed by a classical reversible circuit followed by the inverse quantum

Fourier transform:

$$\mathbf{U} = (\text{QFT}_{2^{2n+1}} \otimes \mathbb{I}^{\otimes n}) \mathbf{R} (\text{QFT}_{2^{2n+1}} \otimes \mathbb{I}^{\otimes n})^\dagger.$$

What is more, we apply this circuit to a very simple starting configuration:  $|0 \dots 01\rangle \in \{0, 1\}^{3n+1}$ . This is remarkable, because

$$(\text{QFT}_{2^{2n+1}} \otimes \mathbb{I}^{\otimes n}) |\mathbf{b}\rangle, \quad \mathbf{R}|\mathbf{b}\rangle \quad \text{and} \quad (\text{QFT}_{2^{2n+1}} \otimes \mathbb{I}^{\otimes n})^\dagger |\mathbf{b}\rangle$$

would be easy to compute classically for every  $|\mathbf{b}\rangle \in \{0, 1\}^{3n+1}$ . A sequential combination of these three quantum subroutines, however, would yield an exponential improvement over the best known approach for order finding. This exponential speedup becomes much more relevant if we use it as a (quantum) subroutine in Algorithm 11.1. There, all other computational steps can be executed on a classical computer in cubic runtime  $O(n^3)$ . Hence, the following corollary is an immediate consequence of Theorem 11.15 and Theorem 11.5 (reformulate factoring as an order finding problem).

**Corollary 11.16 (efficient hybrid quantum-classical algorithm for Integer Factorization (Shor, 1994)).** Let  $N = p \times q$ , with  $p, q$  prime and set  $n = \lfloor \log_2(N) \rfloor + 1$ . Then, we can determine one factor ( $p$  or  $q$ ) by repeating Algorithm 11.1 sufficiently often. The order finding step, in particular, is outsourced to a  $(3n + 1)$ -qubit architecture which executes a circuit of size  $O(n^3)$ . This also bounds the overall cost of all remaining classical computing steps.

hybrid quantum-classical algorithm solves integer factorization at  $O(n^3)$  cost

Contrast this  $O(n^3)$  scaling with the best known fully classical factorization strategy for  $N = p \times q$  with  $p, q$  prime that scales exponentially in  $n^{1/3}$ .

## Problems

**Problem 11.17 (quantum circuit implementation of reversible addition and reversible multiplication).** Sketch how you would construct a quantum implementation of the permutation matrices  $\mathbf{A}_a$  (addition by  $a$  modulo  $N$ ) and  $\mathbf{M}_a$  (multiplication by  $a$  modulo  $N$ ) on  $n = \lfloor \log_2(N) \rfloor + 1$  qubits. Show that the number of elementary gates required does not exceed  $O(n^2)$ .

**Problem 11.18 (eigenvalues and eigenvectors of modular addition operator).** Fix  $N \in \mathbb{N}$ ,  $a \in \mathbb{Z}_N^*$  and let  $t$  the smallest integer that obeys  $t \times a = 1 \pmod{N}$ . Identify at least  $t$  eigenvector-eigenvalue pairs of the modular addition operator  $\mathbf{A}_a$  defined in Eq. (11.7). textbfHint: take inspiration from our analysis of  $\mathbf{M}_a$  and/or Fourier series with finite periodicity.

## 12. Learning from quantum experiments

Date: 24 January 2024

### 12.1 Motivation

Broadly speaking, the main promise and *raison d'être* of quantum computers is that they may have the potential to solve certain problems more efficiently than traditional processing units. Exponential improvements in resource cost are therefore the ultimate objective.

And we do know a couple of computational problems, where fully functional quantum computers can make a substantial difference. Shor's quantum-classical approach to integer factorization comes into mind here, as does amplitude amplification. These approaches isolate quantum circuit size (and classical runtime) as the main cost parameter. And they focus on computational tasks that seem to be hard for existing computing platforms. This, however, means that the proposed quantum solutions must also reflect some of this complexity. And this demand goes way beyond the capabilities of today's nascent quantum computers. These are not perfect and each applied gate is prone to errors. This, unfortunately, restricts us from executing quantum circuits with very short depth. And although the number of qubits  $n$  is growing, the polynomial scaling of famous quantum circuits – like the  $O(n^3)$ -size of Shor's algorithm – grows even faster. So, the more qubits we use, the larger these circuits must become. For now, this unavoidable scaling prevents us from executing Shor's algorithm on existing quantum hardware. If  $n = 51$ , then we simply can't run execute order  $53^3 = 1.5 \times 10^5$  elementary quantum gates in a sufficiently reliable fashion.

But, at the same time, the qubit sizes of existing quantum computing platforms do become respectable. The Google sycamore chip, for instance,

#### Agenda:

- 1 motivation
- 2 stylized learning challenge: data hiding
- 3 two approaches: conventional and quantum-enhanced
- 4 execution on real quantum computer

current quantum hardware too noisy to execute traditional quantum algorithms

boasts  $n = 53$  qubits. So, there should be ‘something’ new and unexpected that we can actually do with them. (Machine) learning can serve as a guiding motivation in this regard. There, the broad goal is not to solve a computationally hard problem, but to learn something that is initially hidden from us. And such learning processes come with their own reasonable cost parameters, in particular *training data size*. This is not a computational cost parameter, but a statistical one. How much information do we need in order to distil underlying principles? And this statistical nature plays nicely with quantum computing architectures which also produce outcome bit strings that are statistical in nature.

switch cost from *runtime* (algorithms) to *training data size* (machine learning)

In fact, this correspondence goes even deeper. Because in quantum computing, we can manipulate the way we generate data by adjusting the quantum circuit we use prior to measurements. And it is reasonable to expect that some quantum circuit executions produce more valuable outcome data than others. This effect is also well-known in machine learning: ‘good data’ lets you learn quickly while ‘bad data’ may require a lot more training effort.

And today, we shall use an ML-inspired view on learning problems based on quantum (computing) experiments to identify exponential quantum advantages.

## 12.2 Stylized learning challenge: data hiding

Let us approach the broad topic of *learning from the quantum world* by means of a concrete example. This example is very stylized, but intended to illustrate the underlying principles and possibilities. The key idea is based on data hiding and involves two players. Player 1 possesses a private string  $\mathbf{w} \in \{x, y, z\}^n$  of length  $n$  and Player 2 wants to learn that string. Player 1 encodes the message/string into a quantum state  $|\hat{\psi}(\mathbf{w})\rangle$  where the encoding process itself is known to both players. She can now be asked to send a copy of that quantum state to Player 2, but every copy is costing Player 2, let’s say, 1000USD. Player 2 has to learn the message *as fast as possible*, meaning with as few copies as possible (because they are expensive). She can do whatever she wants with that state. Apply more gates, leave it lying around for a year, measure the qubits to gather classical data, you name it you get it. The only burden she is carrying is that she should request as few copies as possible from Player 1 and that the ultimate goal is to learn the hidden string. And it turns out that the strategy of how to learn the message influences heavily how many copies are needed to learn the message with certainty. Since this stylized scenario is designed such that in the full quantum realm the learning process is exponentially faster (by exploiting entanglement), there are a few steps involved to ensure that speed-up. For example that the message consists out of three symbols  $\{x, y, z\}$ , and not as usual out of bits  $\{0, 1\}$  is necessary for the quantum approach to excel. The preparation of the quantum state which is send and the encoding process itself also makes sure that the learning process is not ‘too easy’.

stylized data hiding challenge

Let us get a bit more concrete and jump into the details of preparing the



initial quantum state  $|\hat{\psi}_0\rangle$  (where then the message is imprinted on).

Again, Player 1 possesses a private ternary string  $\mathbf{w} \in \{x, y, z\}^n$  and imprints it into a  $n$ -qubit state vector. This state vector is a randomly assigned bit string  $\hat{b}_0, \dots, \hat{b}_{n-1} \stackrel{\text{unif}}{\sim} \{0, 1\}$  with the additional constraint that the parity of the total sum is even:

$$|\hat{\psi}_0\rangle = |\hat{b}_0, \hat{b}_1, \dots, \hat{b}_{n-1}\rangle \text{ with } \hat{b}_k \stackrel{\text{iid}}{\sim} \{0, 1\} \text{ and } \hat{b}_0 \oplus \dots \oplus \hat{b}_{n-1} = 0. \quad (\text{I2.1})$$

We use a 'hat' to denote the fact that this input state is randomly generated. The even parity of sum constraint can, for instance, be enforced by setting  $\hat{b}_{n-1} = \hat{b}_0 \oplus \dots \oplus \hat{b}_{n-2}$ .

randomly generated qubit initialization  $|\hat{\psi}_0\rangle$

**Example 12.1** ( $|\hat{\psi}_0\rangle$  for  $n = 1$  and  $n = 2$  qubits). For a single qubit, the sum of parity constraint is completely binding. The result is

$$(n = 1) : \quad |\hat{\psi}_0\rangle = |0\rangle \quad \text{with certainty.}$$

For  $n = 2$ , there are two bit strings with even parity. The state vector assumes either with equal probability:

$$(n = 2) : \quad |\hat{\psi}_0\rangle = \begin{cases} |0, 0\rangle & \text{with prob. } 1/2, \\ |1, 1\rangle & \text{with prob. } 1/2. \end{cases}$$

■

The randomization of initial quantum states used to imprint the same message ensure the exponential growth of needed copies when not choosing the optimal quantum strategy.

The encoding strategy  $\mathbf{w} \mapsto |\hat{\psi}(\mathbf{w})\rangle = \mathbf{U}(\mathbf{w})|\hat{\psi}_0\rangle$  is known to both players and will be explained in the next section. At this point it is worthwhile to emphasize that Player 2 will have to readout the qubits involved in order to get any actionable advice at all. And measurements destroy the underlying quantum state (collapse of the wave function). So Player 2 must be prepared to invest several 1000USD to learn anything at all.

secret  $n$ -trit string is imprinted on  $|\hat{\psi}_0\rangle$

The overarching questions now are:

- 1 How does the number of state copies required (i.e. the money spent), scale with the number of qubits  $n$  and, by extension, with the size of the hidden string? This is our *toy model for training data size*. It is reasonable to expect that this cost will grow. But, how does it grow (polynomial vs. exponential)?
- 2 Does the way the state copies are processed have an impact on this cost parameter? After all, different ways of accessing this quantum state vector may lead to readout bits that carry more or less information about the underlying secret. This is our way of varying the *quality of training data*.

Today, we construct a variant of this data-hiding game, where discrepancies are as pronounced as possible. Any quantum-classical learning approach conceivable

that only addresses individual copies of  $|\hat{\psi}(\mathbf{w})\rangle$  must scale exponentially with qubit size  $n$ . We will call this the *conventional approach* and an exponential scaling is probably not too surprising. After all, there are  $3^n$  possibilities for the hidden string  $\mathbf{w} \in \{x, y, z\}^n$ . What is surprising, is that we also offer an alternative that is (exponentially) more efficient. It is possible to construct a quantum-enhanced readout protocol that processes pairs of  $|\hat{\psi}(\mathbf{w})\rangle$  in a genuine quantum fashion. We call this the *quantum-enhanced approach*. It is cheap (short circuits) and uncovers the hidden string after only a constant number of iterations. In turn, the cost associated with both secret learning approaches deviates exponentially in the size  $n$  of the task:

$$T_{\text{conv}}(n) = 2^{\Omega(n)} \quad \text{while} \quad T_{\text{qe}}(n) = \mathcal{O}(n).$$

The first statement highlights that the cost for the conventional approach scales (at least) exponentially in the number of qubits  $n$ . The second statement, in stark contrast, states that the cost for the quantum-enhanced approach scales (at most) linearly in the number of qubits  $n$ .

exponential discrepancy in cost to learn hidden secret

### 12.2.1 Encoding strategy

We are now ready to explain the high-level rules of our learning challenge that result in the provably exponential cost discrepancy displayed in Eq. (12.2). To this end, we must first specify the encoding procedure

$$\{x, y, z\}^n \ni \mathbf{w} \mapsto |\hat{\psi}(\mathbf{w})\rangle = \mathbf{U}(\mathbf{w})|\hat{\psi}_0\rangle \in (\mathbb{C}^2)^{\otimes n}.$$

The reason why this string involves trits instead of bits is based on the elementary building blocks of our encoding. It uses the three most prominent single-qubit gates, namely *identity*, *Hadamard* and *phase* gates:

$$\mathbb{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \mathbf{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad \mathbf{S} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}.$$

We will use these gates to imprint the secret key onto (probabilistic mixtures of)  $n$ -qubit computational basis states. For  $w \in \{x, y, z\}$ , we set

$$\mathbf{V}_z = \mathbb{I}, \quad \mathbf{V}_x = \mathbf{H} \quad \text{and} \quad \mathbf{V}_y = \mathbf{S} \times \mathbf{H}. \quad (12.2)$$

The labels  $x, y, z$  respect an intimate connection between these three unitaries and the three Pauli matrices.

**Exercise 12.2 (connection between Eq. (12.2) and Pauli matrices).** The three non-trivial Pauli matrices are defined as

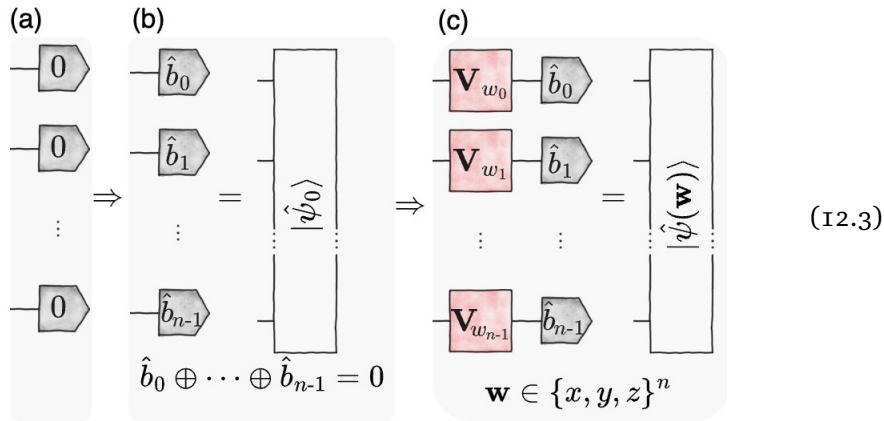
$$\mathbf{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \mathbf{Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \mathbf{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Verify the following relation between these Pauli matrices and the unitary transformations in Eq. (12.2):

$$\mathbf{X} = \mathbf{V}_x^\dagger \times \mathbf{Z} \times \mathbf{V}_x, \quad \mathbf{Y} = \mathbf{V}_y^\dagger \times \mathbf{Z} \times \mathbf{V}_y, \quad \mathbf{Z} = \mathbf{V}_z^\dagger \times \mathbf{Z} \times \mathbf{V}_z.$$

These three unitaries allow us to imprint exactly one trit  $w \in \{x, y, z\}$  on a single qubit computational basis state. This encoding strategy readily extends to  $n$ -trit strings  $\mathbf{w} = (w_0, \dots, w_{n-1}) \in \{x, y, z\}^n$  and  $n$  qubits:

concrete encoding strategy



Here,  $|\hat{\psi}_0\rangle = |\hat{b}_0, \dots, \hat{b}_{n-2}, \hat{b}_0 \oplus \dots \oplus \hat{b}_{n-2}\rangle$  is the randomly constructed input string from Eq. (I2.1). With the encoding strategy at hand, we can introduce the two different approaches on how to access this quantum state.

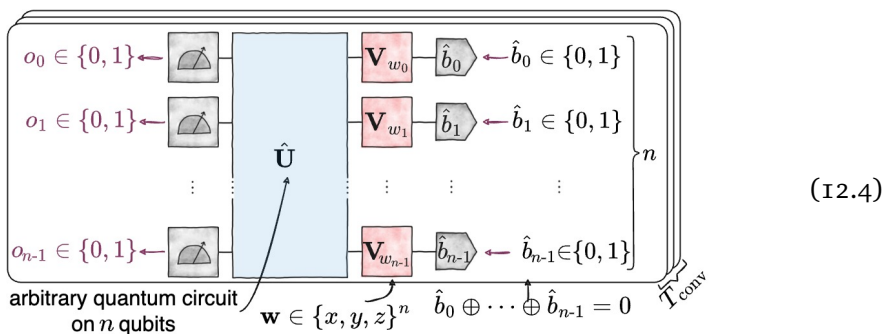
### 12.2.2 Conventional approach

Our first approach is inspired by the way people have performed experiments for centuries now. This is why we dub it the conventional approach. The key idea consists of three steps:

- (i) acquire a probe (buy the state  $|\hat{\psi}(\mathbf{w})\rangle$ ),
- (ii) do something with the given probe (apply a unitary circuit),
- (iii) perform a  $n$ -qubit readout to observe the underlying behavior.

These three basic steps are then repeated many times to get enough data in order to draw credible solutions. Here is an illustration of such an approach for our hidden-data challenge:

conventional approach (sequential): modify single states, readout, repeat.



In this illustration, we always start with buying a single copy of  $|\hat{\psi}(\mathbf{w})\rangle$ . We then apply a unitary circuit (blue) and perform a  $n$ -qubit readout to get a

classical bit string (magenta). This conversion is probabilistic and destructive. Hence, a single run of this procedure will not provide us with enough statistical data. This is why we repeat it  $T_{\text{conv}}$  times. Note, however, that the type of experiment – i.e. the choice of unitary circuit – can, and should, change between repetitions. This allows us, for instance, to sequentially check different aspects of the underlying state  $|\hat{\psi}(\mathbf{w})\rangle$  one at a time. But more sophisticated methods can also come into play. For instance, we could use powerful learning algorithms in order to execute optimised scheduling procedures to make the most of the data we generate. Machine learning comes to mind here. What is more, we don't impose any constraint on the computational cost associated with individual unitary transformations. Every conceivable quantum circuit (even if it is arbitrarily large) is fair game.

Although brief and high-level, these arguments should highlight that the conventional readout approach from Eq. (12.4) is extremely general. It virtually encompasses *every* conceivable readout strategy that uses quantum measurements in a sequential fashion to learn something about a quantum system. Given this level of generality – and the absence of any computational restrictions on both the quantum computation *and* the conventional data-processing – the following rigorous result might be surprising.

**Theorem 12.3 (Lower bound on any conventional strategy [HKP21]).** Consider the data hiding strategy  $\{x, y, z\}^n \ni \mathbf{w} \mapsto |\hat{\psi}(\mathbf{w})\rangle$  from Eq. (12.3) for  $n$  qubits. Then, *any* conventional (data) readout procedure illustrated in Eq. (12.4) requires

$$T_{\text{conv}}(n) = 2^{\Omega(n)}$$

repetitions to recover  $\mathbf{w} \in \{x, y, z\}^n$  with probability of success  $> 50\%$ .

The big- $\Omega$  notation highlights that the cost grows (at least) exponentially in  $n$ . This is a very general statement and the proof is not easy. Technically speaking it requires an additional randomization over one sign factor for each qubit and employs sophisticated techniques from statistical learning theory, probability theory and quantum computing. Such a level of sophistication is necessary to handle arbitrary quantum circuits and arbitrary classical data processing, including any machine learning model.

The underlying idea, however, is rather simple and boils down to our data hiding strategy (12.3). Although it looks simple, it does imprint the secret trit string into a  $n$ -qubit system  $|\hat{\psi}(\mathbf{w})\rangle$  that must be interpreted as a probability distribution over  $2^n/2$  different quantum state vectors

$$\mathbf{U}(\mathbf{w})|b_0, \dots, b_{n-2}, b_0 \oplus \dots \oplus b_{n-1}\rangle = \mathbf{U}(w_1)|b_0\rangle \otimes \dots \otimes \mathbf{U}(w_{n-1})|b_0 \oplus \dots \oplus b_{n-2}\rangle.$$

with  $b_0, \dots, b_{n-2} \in \{0, 1\}^{n-1}$ . So there are a lot of degrees of freedom available to maliciously hide even  $3^n$  different trit strings. What is more, the encoding strategy is based on very specific single-qubit unitaries that are as different from each other as possible. This ensures that we actually occupy radically different corners of this huge space of possibilities. And this makes it

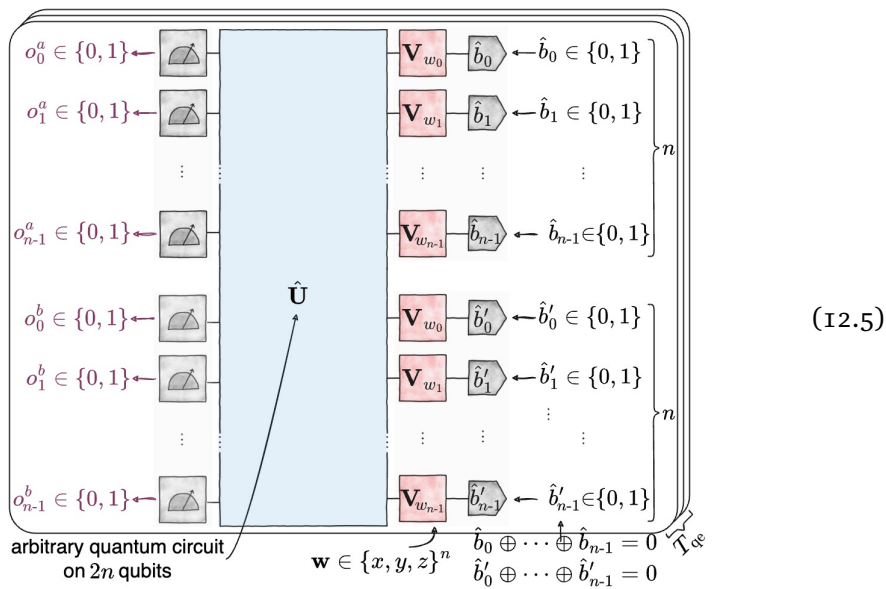
extremely difficult to do anything (substantially) smarter than iteratively asking binary questions: is  $\mathbf{w} = \mathbf{v}$ , where  $\mathbf{v} \in \{x, y, z\}^n$  is our current best guess. And since there are  $3^n$  possible guesses, we are forced to ask this question exponentially often (in  $n$ ). We will sketch a single-qubit caricature of this effect in Annex 1 below.

### 12.2.3 Quantum-enhanced approach

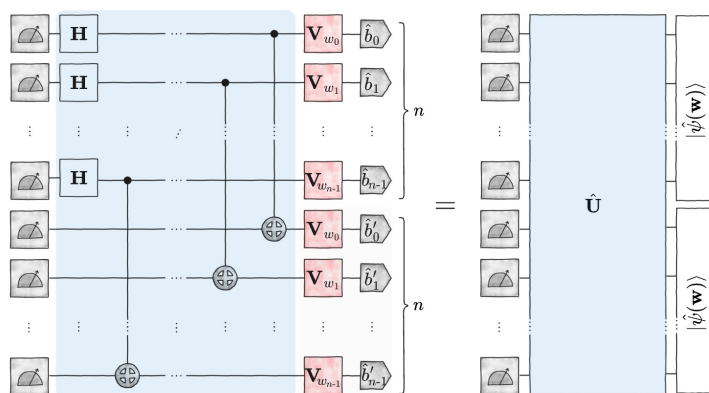
In the previous section, we have outlined very powerful arguments that seem to suggest that the data hiding procedure from Eq. (12.3) is very secure. In our data hiding game, this means that Player 2 is forced to buy exponentially many copies of  $|\hat{\psi}(\mathbf{w})\rangle$  in order to learn a secret string  $\mathbf{w} \in \{x, y, z\}^n$ . This binds despite the fact that the underlying data extraction model looks very general and powerful. However, the conventional approach from the previous section is still deeply rooted in a classical way of thinking. And if one is willing to accept the quantum computing paradigm, it is not the only way to approach this challenge.

If we assume access to a (sufficiently large) quantum computer, we can envision processing multiple copies of the unknown state  $|\hat{\psi}(\mathbf{w})\rangle$  at the same time and within the same quantum circuit. The easiest setup for such a *quantum parallel* data processing routine involves two state copies and looks as follows:

quantum-enhanced approach (parallel): modify state pairs, readout, repeat.



It is easy to see that this framework is at least as general as the conventional approach. After all, we can always choose to divide up the general  $2n$ -qubit unitary circuit into two parallel (and uncorrelated)  $n$ -qubit unitary circuits. Doing so effectively reduces one run of this protocol into two independent runs of the conventional protocol. But from this new point of view, such a separation starts to look a bit wasteful. What if we instead use this additional expressiveness to insert correlations (think: CNOTs) and superpositions (think:



**Figure 12.1** *Quantum enhanced learning protocol*: this  $2n$ -qubit circuit processes two copies of  $|\hat{\psi}(\mathbf{w})\rangle$  in parallel. The circuit executes a total of  $n$  Bell basis measurements that each connect the  $k$ -th qubit of the first copy with the  $k$ -th qubit of the second copy. This readout stage requires exactly  $n$  single qubit gates ( $\mathbb{I}, \mathbf{H}, \mathbf{S} \times \mathbf{H}$ ) and exactly  $n$  two-qubit CNOT gates. Each data hiding state is also comparatively cheap to create: we need two random  $n$ -bit initializations and (at most)  $2n$  single qubit gates ( $\mathbb{I}, \mathbf{H}, \mathbf{S} \times \mathbf{H}$ ). Viewed as one circuit from beginning to end, this demonstration requires  $2n$  qubits, (at most)  $3n$  single-qubit gates and exactly  $n$  CNOT gates. This is doable even on noisy, intermediate-scale quantum hardware.

Hadamards) between qubits from the first copy of  $|\hat{\psi}(\mathbf{w})\rangle$  (top) and the second copy of  $|\hat{\psi}(\mathbf{w})\rangle$  (bottom)? Thinking further along these lines highlights that this new, *quantum-enhanced approach* is capable of doing something that the conventional approach cannot fully mimic (at least not with an exponential overhead in repetitions): creating entanglement between qubits from each copy directly at the quantum level. The following constructive result highlights that such quantum-enhanced readout protocols become a game changer for our data hiding challenge:

**Theorem 12.4 (Upper bound for a fixed quantum-enhanced strategy [HKP21]).**

Consider the data hiding strategy  $\{x, y, z\}^n \ni \mathbf{w} \mapsto |\hat{\psi}(\mathbf{w})\rangle$  from Eq. (12.3) for  $n$  qubits. Then, there is a simple quantum-enhanced procedure that allows for uncovering the hidden string  $\mathbf{w}$  after already

$$T_{\text{qe}} = \mathcal{O}(n)$$

repetitions. The quantum part of this procedure is very cheap and illustrated in Fig. 12.1. A total  $T_{\text{qe}}$  repetitions produce enough statistical data to check  $\mathbf{v} \stackrel{?}{=} \mathbf{w}$  for every candidate  $\mathbf{v} \in \{x, y, z\}^n$  in linear time.

In contrast to Theorem 12.3, this result is constructive. We have one concrete solution strategy and need to show that it actually works (efficiently).

In turn, the actual proof is also much simpler by comparison. In fact, a thorough analysis of the single-qubit case ( $n = 1$ ) already conveys much of the main ideas. We will provide such an analysis in Annex 2 below. The key idea is to use Bell-type measurements on both single-qubit states to unravel information about all possible encoding unitaries  $\mathbf{U}(w)$  at once. This general idea extends to  $n$ -qubit systems, because we have fine-tuned the (randomized) initial state  $|\hat{\psi}_0\rangle$  from Eq. (12.1) in precisely the right way to make it work.

To summarize: Our data imprinting strategy is designed to be hard for conventional readout procedures, but also contains a deliberate loophole: it is very easy to crack with a simple quantum-enhanced procedure. This is deliberate: after all, we are looking for stylized example challenges that admit a quantum advantage. And the discrepancy between Theorem 12.3 (exponential lower bound on any conventional strategy) and Theorem 12.4 (linear upper bound for one quantum-enhanced strategy) achieves just that. At this point, it is worthwhile to re-emphasize that this exponential discrepancy does not (necessarily) manifest itself in algorithm runtime. Instead, it targets the number of repetitions that are required to learn the underlying secret. This is an appropriate and very general model for ‘training data size’ in machine learning.

### 12.3 Demonstration on an actual quantum computer

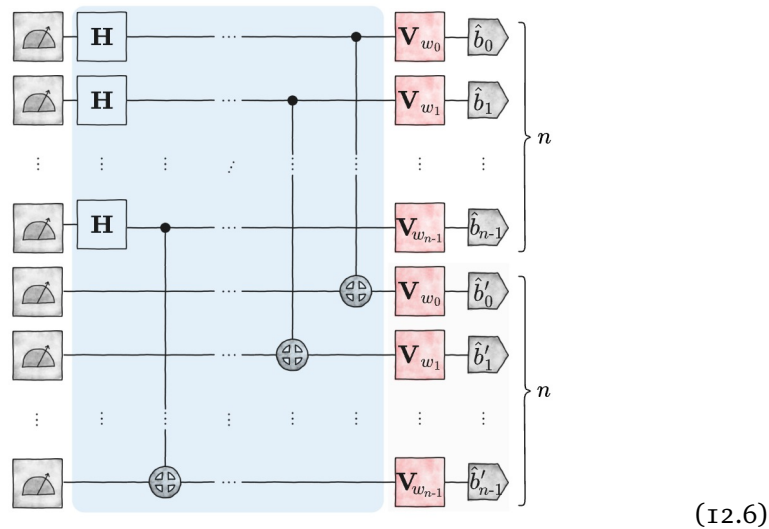
In the previous section, we have laid out a new approach towards quantum advantage. We have presented a (stylized) learning challenge where we first imprint a trit string  $\mathbf{w} \in \{x, y, z\}^n$  onto a mixed  $n$ -qubit quantum state and subsequently analyze how many experiments it takes to acquire enough (training) data in order to confidently recover this now hidden string. And we have seen that the type of data acquisition makes a huge difference: any conventional data acquisition procedure necessarily requires exponentially many repetitions (in  $n$ ) while a quantum-enhanced approach gets by with only a linear number of repetitions. And, what is more, all steps in this protocol are relatively simple, even for large qubit sizes  $n$ .

Let us first see how expensive it is to create  $|\hat{\psi}(\mathbf{w})\rangle$  for a fixed (and arbitrary) trit string  $\mathbf{w} \in \{x, y, z\}^n$ . To this end, we must first prepare  $|\psi_0\rangle$  which is a (classical) probabilistic average over all possible  $2^n/2$  input states  $|b_0, \dots, b_{n-1}\rangle$  with even sum of parity. We can effectively generate such a state by sampling one even parity initialization  $|\hat{b}_0, \dots, \hat{b}_{n-1}\rangle$  uniformly at random whenever we prepare  $|\hat{\psi}_0\rangle$  (the probabilistic average over all possible even parity initialization will then create the mixed state  $|\psi_0\rangle$ ). To create such a string, we simply sample the first  $(n - 1)$  bits uniformly at random and adjust the last bit to ensure even (sum of) parity. Subsequently, we need to imprint the string  $\mathbf{w}$  by applying  $\mathbf{V}_{w_0} \otimes \dots \otimes \mathbf{V}_{w_{n-1}}$  to our random initialization  $|\hat{b}_0, \dots, \hat{b}_{n-1}\rangle$  (with even parity). But this is also cheap, because we apply one single qubit unitary to each qubit.

Next, we need to have a look at the different readout protocols. The



conventional readout protocol is a bit tricky, because the model is very general. Moreover, this is also not the main point. Rigorous math tells us that any such approach must be bad. So, let us instead focus on the quantum-enhanced protocol. There, we must first prepare two copies of  $|\hat{\psi}(\mathbf{w})\rangle$  in parallel. This is not difficult, but requires  $2n$  available (and working) qubits. If we have a quantum computer with, say, 53 qubits, then we can only play the data hiding game up to  $n = \lfloor 53/2 \rfloor = 26$  qubits. The subsequent entangling procedure involves a total of  $n$  parallel CNOT gates followed by  $n$  parallel Hadamard gates. And, finally, we need to perform measurements on all  $2n$  qubits involved. This is again a routine task for any working quantum computer. So, in summary, a full execution of our data hiding game with quantum-enhanced readout boils down to the following depth-4 circuit which involves  $4n$  gates:

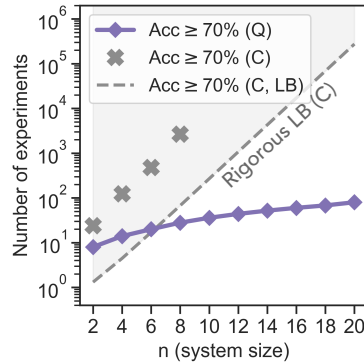


(all CNOTs affect different qubits and can therefore be stacked into a depth-1 layer). This looks doable even on existing quantum computers that are noisy (which limits available circuit depth) and comparatively small (which limits the maximum  $n$  we can go to). And, in fact, it has been done. Google – who operates one of the largest and most reliable quantum computers to date – teamed up with researchers from Caltech and JKU (yours truly) to actually run a slight modification of our data hiding game [Hua+22]. The results are depicted in Figure 12.2

For technical reasons<sup>1</sup> we could only scale up to  $n = 20$  (for one  $\rho(\mathbf{w})$ ) i.e.  $2n = 40$  qubits in total. But this is enough to actually witness a substantial discrepancy between conventional strategies and quantum-enhanced strategies:  $2^{20} \approx 1.04 \times 10^6$  is much larger than  $n = 20$ . The plot in Fig. 12.2 shows just that. The dashed solid line is the lower bound from Theorem 12.3. Any conventional readout strategy must be north-west of this exponential growth line. The green dots depict one such strategy which essentially involves randomly guessing the correct imprinting unitaries in a hardware-friendly and

<sup>1</sup>One qubit is broken and connectivity becomes an issue.





**Figure 12.2** Empirical performance of conventional (gray) and quantum-enhanced (purple) learning on the 53-qubit Google Sycamore chip [Hua + 22]: The task is to recover a hidden trit string  $\mathbf{w} \in \{x, y, z\}^n$  that is imprinted onto a randomized  $n$ -qubit state  $|\hat{\psi}(\mathbf{w})\rangle$ . This log plot displays the number of quantum executions/training data size as a function of qubit number  $n$ . The dashed line illustrates the fundamental lower bound from Theorem 12.3.

data hiding case study on 53-qubit processor (Google)

automated fashion. The purple dots are where things get interesting. These depict the performance of the quantum-enhanced readout protocol and it is way south-west of the dashed line. This is strong empirical support for the theoretic assertion from Theorem 12.4 and one of the first large-scale demonstrations of a quantum advantage on real quantum hardware.

### Annex 1: single-qubit analysis of a conventional approach

For  $n = 1$  qubit, our initial state collapses to a simple (deterministic) initialization:

$$|\hat{\psi}_0\rangle = |0\rangle.$$

Now, let us implement the encoding strategy. It is based on mapping a single trit  $w \in \{x, y, z\}$  onto this initial state by applying a single-qubit unitary that depends on  $w$ . According to our encoding strategy, we have  $\mathbf{V}_x = \mathbf{H}$  (Hadamard),  $\mathbf{V}_y = \mathbf{S} \times \mathbf{H}$  (Hadamard+phase) and  $\mathbf{V}_z = \mathbb{I}$  (do nothing). It is easy to determine the resulting states:

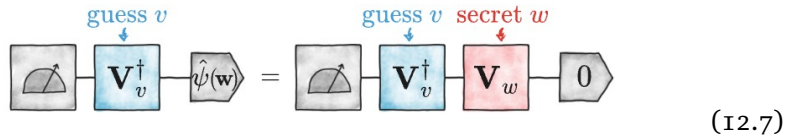
$$|\hat{\psi}(w)\rangle = \mathbf{V}|0\rangle = \begin{cases} \mathbf{H}|0\rangle = |+\rangle & \text{if } w = x, \\ \mathbf{S} \times \mathbf{H}|0\rangle = |i+\rangle & \text{else if } w = y, \\ \mathbb{I}|0\rangle = |0\rangle & \text{else if } w = z. \end{cases}$$

Note that both

$$|+\rangle = \mathbf{H}|0\rangle = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \quad \text{and} \quad |i+\rangle = \mathbf{S} \times \mathbf{H}|0\rangle = \frac{1}{\sqrt{2}} (|0\rangle + i|1\rangle).$$

are perfect superpositions between  $|0\rangle$  and  $|1\rangle$ , but the pre-factor differs.

The task is now to devise a quantum circuit that can identify the single trit  $w \in \{x, y, z\}$  hidden within  $|\hat{\psi}(w)\rangle$ . We will not do the fully general case advertised in Sec. 12.2, but instead, consider a special case. Since the encoding strategy is known to us, we can try to undo it. Here is a guessing circuit that attempts to achieve this goal:



Here, we have already suggestively included the encoding procedure on the right hand side. A quick look at this single-qubit circuit architecture already tells us that two things can happen:

- 1  $v = w$  (correct guess): in this case, the two unitaries in Eq. (12.7) cancel out and we end up simply measuring the zero state  $|0\rangle$ . More formally,

$$\Pr_{V_v^\dagger|\hat{\psi}(w)\rangle}[0] = \left| \langle 0 | V_v^\dagger V_w | 0 \rangle \right|^2 = 1 = |\langle 0 | 0 \rangle|^2 = 1.$$

(Here we have stuck to the density matrix formalism, because it is this one that will generalize to  $n$  qubits).

- 2  $v \neq w$  (incorrect guess): in this case, the two unitaries  $V_w$  and  $V_v^\dagger$  cannot cancel, but their combination produces a state that is always in perfect superposition between  $|0\rangle$  and  $|1\rangle$ . More formally,

$$\Pr_{V_v^\dagger|\hat{\psi}(w)\rangle}[0] = \left| \langle 0 | V_v^\dagger V_w | 0 \rangle \right|^2 = \frac{1}{2} \quad \text{for all } v, w \in \{x, y, z\} \text{ with } v \neq w.$$

We leave a derivation as an instructive exercise.

**Exercise 12.5** Verify the following general equation for  $v, w \in \{x, y, z\}$  by directly computing all  $3 \times 3 = 9$  state vector amplitudes:

$$\Pr_{V_v^\dagger V_w | 0} [0] = \left| \langle 0 | V_v^\dagger \times V_w | 0 \rangle \right|^2 = \begin{cases} 1 & \text{if } v = w, \\ 1/2 & \text{else if } v \neq w. \end{cases}$$

It is worthwhile to emphasize that the two possible outcome distributions are radically different. One (correct guess) produces a deterministic outcome of always 0, while the other (incorrect guess) produces a uniform distribution, where 0 and 1 both occur with probability 1/2. It is very easy to distinguish these two situations by re-running the quantum circuit a few times. If we ever observe a 1, we know that our guess must have been incorrect. And 1 occurs with probability 1/2, so we don't have to wait very long. Already 4 quantum circuit runs (coin tosses) are enough.

But this is also where the issues arise. Re-running the test circuit from Eq. (12.7) only ever allows us to check whether our current guess  $v$  equals the

hidden trit  $w$ . If this is not the case, i.e.  $v \neq w$ , we do not get any actionable advice on what string to try next. After all, every possible combination of  $v$  (guess) and  $w$  with  $v \neq w$  produces the same (uniform) outcome distribution.

So, what does that mean for us if we want to recover a hidden data trit  $w \in \{x, y, z\}$  that is encoded into  $|\hat{\psi}(w)\rangle$ ? The strategy displayed above only really leaves us with one option: making random guesses  $v$  and using repetitions of the corresponding quantum circuit to check whether  $v = w$ . For  $n = 1$  (a single trit of hidden data) this is not too costly (yet). There are only  $3^1 = 3$  possibilities and we obtain

$$T_{\text{conv}}(1) \gtrsim 4 \times 3^1, \quad (12.8)$$

where 4 takes into account the (expected) number of circuit evaluations required to distinguish a deterministic distribution (correct guess) from the uniform one (incorrect guess).

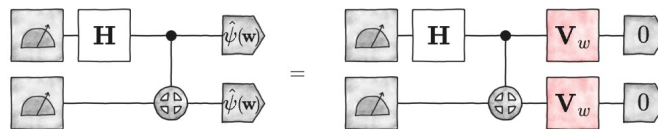
However, the form of Eq. (12.8) suggests an exponential dependence on the number of qubits  $n = 1$ . Indeed, as  $n$  increases, the total number of secret trit strings  $\mathbf{w} \in \{x, y, z\}^n$  grows as  $3^n$ . And, if randomly guessing the correct string is essentially our only chance, we will feel this exponential growth in the number of options.

**Exercise 12.6** Extend this analysis to  $n = 2$  and  $n = 3$  qubits. Show that the number of ‘guess circuits’ you need will grow indeed proportionally to  $3^2$  and  $3^3$ , respectively. This is a strong indicator of exponential growth in the number of qubits. **Hint:** the random initialization of  $|\hat{\psi}_0\rangle$  plays an important role here.

We emphasize that such a generalization is *not* enough to deduce Theorem 12.3. After all, we are analyzing only one particular readout strategy (guess the correct string by unravelling) and not all of them. It does, however, convey the main gist: our secret imprinting is designed in a way that makes it very difficult to extract ‘global information’ about the secret string  $\mathbf{w} \in \{x, y, z\}^n$ .

## Annex 2: single-qubit analysis of the quantum-enhanced approach

For  $n = 1$ , our quantum-enhanced circuit strategy boils down to the following circuit on  $2 \times 1 = 2$  qubits:



There are three different circuit configurations — one for each  $w \in \{x, y, z\}$  — that want to be analyzed. We learn everything about the measurement outcome if we compute the final 2-qubit quantum state

$$|\varphi(w)\rangle = (\mathbf{H} \otimes \mathbb{1})\mathbf{CNOT}(\mathbf{V}_w \otimes \mathbf{V}_w)|0, 0\rangle.$$

1  $w = x$ , i.e.  $V_w = V_x = H$ :

$$\begin{aligned} |\varphi(x)\rangle &= (H \otimes \mathbb{I}) \mathbf{CNOT} (H \otimes H) |0, 0\rangle \\ &= (H \otimes \mathbb{I}) \mathbf{CNOT} |+, +\rangle \\ &= (H \otimes \mathbb{I}) |+, +\rangle = |0, +\rangle \\ &= \frac{1}{\sqrt{2}} (|0, 0\rangle + |0, 1\rangle) \end{aligned}$$

2  $w = y$ , i.e.  $V_w = V_y = SH$ :

$$\begin{aligned} |\varphi(y)\rangle &= (H \otimes \mathbb{I}) \mathbf{CNOT} (SH \otimes SH) |0, 0\rangle \\ &= (H \otimes \mathbb{I}) \mathbf{CNOT} |i+, i+\rangle \\ &= (H \otimes \mathbb{I}) \frac{1}{2} (|0, 0\rangle + i|0, 1\rangle + i|1, 1\rangle - |1, 0\rangle) \\ &= \frac{1}{\sqrt{2}} (|1, 0\rangle + i|0, 1\rangle). \end{aligned}$$

3  $w = z$ , i.e.  $V_w = V_z = \mathbb{I}$ :

$$\begin{aligned} |\varphi(z)\rangle &= (H \otimes \mathbb{I}) \mathbf{CNOT} (\mathbb{I} \otimes \mathbb{I}) |0, 0\rangle \\ &= (H \otimes \mathbb{I}) \mathbf{CNOT} |0, 0\rangle \\ &= (H \otimes \mathbb{I}) |0, 0\rangle = |+, 0\rangle \\ &= \frac{1}{\sqrt{2}} (|0, 0\rangle + |1, 0\rangle). \end{aligned}$$

So, in summary, the three different choices for  $w$  lead to different outcome probability distributions for the two readout bits. If we collect the probabilities for obtaining 00, 01, 10 and 11 in a 4-dimensional vector, we obtain

$$\mathbf{p}_x = \frac{1}{2} \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{p}_y = \frac{1}{2} \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}, \quad \mathbf{p}_z = \frac{1}{2} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad \begin{matrix} \Pr_{|\varphi(w)\rangle} [00] \\ \Pr_{|\varphi(w)\rangle} [01] \\ \Pr_{|\varphi(w)\rangle} [10] \\ \Pr_{|\varphi(w)\rangle} [11] \end{matrix} \cdot \quad (\text{I2.9})$$

Note that these three possible outcome distributions are very different from each other. A (small) constant number of quantum circuit repetitions (coin tosses) allow us to do a 2-step identification procedure.

Suppose, for the sake of illustration, that the underlying secret is  $w = x$ . Then,  $\mathbf{p}_x$  tells us that we obtain outcome 00 with probability 1/2. And we expect to see this outcome after only 2 repetitions (stopping time for coin tossing). As soon as we observe 00 once, we can exclude  $w = y$  for the hidden string ( $\mathbf{p}_y$  has 0 weight on 00) and are left with two options:  $w = x$  or  $w = z$ . We then invest a couple of extra repetitions (coin tosses) to look for outcome 01. As soon as we observe this outcome, we can also rule out  $\mathbf{p}_z$  and can be sure that the underlying hidden trit is  $w = x$ . We expect that 2 additional

repetitions are enough. The search procedure for other trits is analogous and gets by with equally few runs of the quantum-enhanced readout procedure.

In summary, we can conclude that the quantum-enhanced readout protocol gets by with

$$T_{\text{qe}}(1) = \text{const} \quad \text{where} \quad \text{const} \approx 4$$

repetitions only. What is more, this procedure is fixed and *not* conditional on a guess by us. The fixed Bell-type measurement allows us to correctly uncover the secret trit  $w$  after very few repetitions of the same quantum circuit. These are all excellent signs for a benign generalization to  $n \geq 1$  qubits.

## Bibliography

- [Bel64] J. Bell. “On The Einstein Podolsky Rosen Paradox”. In: *Physics* 1.3 (1964), pages 195–200. DOI: <https://doi.org/10.1103/PhysicsPhysiqueFizika.1.195>.
- [BB14] C. H. Bennett and G. Brassard. “Quantum cryptography: Public key distribution and coin tossing”. In: *Theor. Comput. Sci.* 560 (2014), pages 7–11. DOI: [10.1016/j.tcs.2014.05.025](https://doi.org/10.1016/j.tcs.2014.05.025). URL: <https://doi.org/10.1016/j.tcs.2014.05.025>.
- [Ben+93] C. H. Bennett et al. “Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels”. In: *Phys. Rev. Lett.* 70 (13 1993), pages 1895–1899. DOI: [10.1103/PhysRevLett.70.1895](https://doi.org/10.1103/PhysRevLett.70.1895). URL: <https://link.aps.org/doi/10.1103/PhysRevLett.70.1895>.
- [DN05] C. M. Dawson and M. A. Nielsen. *The Solovay-Kitaev algorithm*. 2005. arXiv: [quant-ph/0505030](https://arxiv.org/abs/quant-ph/0505030) [quant-ph].
- [EWL99] J. Eisert, M. Wilkens, and M. Lewenstein. “Quantum Games and Quantum Strategies”. In: *Physical Review Letters* 83.15 (1999), pages 3077–3080. DOI: [10.1103/physrevlett.83.3077](https://doi.org/10.1103/physrevlett.83.3077). URL: <https://doi.org/10.1103%2Fphysrevlett.83.3077>.
- [Eke91] A. Ekert. “Quantum Cryptography Based on Bell’s Theorem”. In: *Phys. Rev. Lett.* 67.6 (1991), pages 661–663. DOI: <https://doi.org/10.1103/PhysRevLett.67.661>.
- [GC99] D. Gottesman and I. L. Chuang. “Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations”. In: *Nature* 402.6760 (1999), pages 390–393. DOI: [10.1038/46503](https://doi.org/10.1038/46503). URL: <https://doi.org/10.1038/46503>.

- [HKP21] H.-Y. Huang, R. Kueng, and J. Preskill. “Information-Theoretic Bounds on Quantum Advantage in Machine Learning”. In: *Phys. Rev. Lett.* 126 (19 2021), page 190505. DOI: [10.1103/PhysRevLett.126.190505](https://doi.org/10.1103/PhysRevLett.126.190505). URL: <https://link.aps.org/doi/10.1103/PhysRevLett.126.190505>.
- [Hua+22] H.-Y. Huang et al. “Quantum advantage in learning from experiments”. In: *Science* 376.6598 (2022), pages 1182–1186. DOI: [10.1126/science.abn7293](https://doi.org/10.1126/science.abn7293). eprint: <https://www.science.org/doi/pdf/10.1126/science.abn7293>. URL: <https://www.science.org/doi/abs/10.1126/science.abn7293>.
- [Kit97] A. Y. Kitaev. “Quantum computations: algorithms and error correction”. In: *Russian Mathematical Surveys* 52.6 (1997), page 1191. DOI: [10.1070/RM1997v052n06ABEH002155](https://doi.org/10.1070/RM1997v052n06ABEH002155). URL: <https://dx.doi.org/10.1070/RM1997v052n06ABEH002155>.
- [Kue22] R. Kueng. *Introduction to Computational Complexity* (lecture notes). JKU Linz, Austria, 2022. URL: <https://iic.jku.at/files/eda/kueng-complexity.pdf>.
- [Tom+13] M. Tomamichel et al. “A monogamy-of-entanglement game with applications to device-independent quantum cryptography”. In: *New J. Phys.* 15 (2013), pages 103002, 24. ISSN: 1367-2630. DOI: [10.1088/1367-2630/15/10/103002](https://doi.org/10.1088/1367-2630/15/10/103002). URL: <https://doi.org/10.1088/1367-2630/15/10/103002>.
- [Wil23] J. Wilkens. *Quantum Circuit Library*. 2023. URL: <https://github.com/wilkensJ/drawio-library>.