Quantum Algorithms in Measurement-Based Quantum Computing

Dissertation zur Erlangung des
Doktorgrades der Naturwissenschaften (Dr. rer. nat.)

dem

Fachbereich Physik der Philipps-Universität Marburg

vorgelegt von

Maximilian Schwetz

aus

Erlangen

Marburg, 2024

Vom Fachbereich Physik der Philipps-Universität als Dissertation angenommen am:
29.08.2024

Erstgutachter:

Prof. Dr. Reinhard M. Noack
Arbeitsgruppe Vielteilchennumerik
Fachbereich Physik
Philipps Universität Marburg
Renthof 6, 35032 Marburg

Zweitgutachter:

Prof. Dr. Salvatore Manmana
Institut für Theoretische Physik
Georg-August-Universität Göttingen
Friedrich-Hund-Platz 1, 37077 Göttingen

Tag der mündlichen Prüfung: 06.09.2024

Hochschulkennziffer:

1180

Abstract

Measurement-based quantum computing (MBQC) is a formulation of quantum computing alternative to the canonical circuit model. While equivalent in terms of universal computational power, MBQC can be advantageous in experiments and can highlight the core mechanics of quantum algorithms. The Deutsch-Jozsa and Simon algorithms are two of the most prominent quantum algorithms. The Deutsch-Jozsa algorithm determines if a function is constant or balanced and is frequently used as introduction to the realm of quantum algorithms. The Simon algorithm finds the period of a bitwise function and was the first algorithm proven to be supreme to its classical counterpart. We formulate the two- and three-qubit Deutsch-Jozsa algorithm as well as the n-qubit Simon algorithm in the language of MBQC. We employ the framework of the ZX-calculus, a graphical tensor description of quantum states and operators, to translate the circuit representations of these algorithms to MBQC and to outline the necessary single-qubit projective measurements to implement their oracles. The resulting ZX-diagrams depict the resource cluster states of the oracles, and the measurement axes can be read directly off the diagrams. Formulating these established algorithms in the framework of MBQC should aid in understanding the core mechanics and can serve as a blueprint for experimental implementation.

German Abstract / Deutsche Zusammenfassung

Measurement-based quantum computing (MBQC/Messbasiertes Quantenrechnen) ist eine zu den ansonsten üblichen Quantenschaltkreisen alternative Darstellung des Quantenrechnens. Obwohl beide die gleiche Rechenleistung besitzen, kann MBQC in Experimenten Vorteile aufweisen und die innere Mechanik von Quantenalgorithmen verdeutlichen. Die Deutsch-Jozsa- und Simon-Algorithmen sind zwei der berühmtesten Quantenalgorithmen. Der Deutsch-Jozsa-Algorithmus stellt fest, ob eine Funktion konstant oder ausbalanciert ist, und wird häufig als Einführung in die Welt der Quantenalgorithmen verwendet. Der Simon-Algorithmus bestimmt die Periodizität einer bitweisen Funktion und war der erste Algorithmus, für den bewiesen wurde, dass er seinem klassisches Pendant überlegen ist. Wir formulieren den Deutsch-Jozsa Algorithmus für zwei und drei Qubits und den Simon Algorithmus für n Qubits in der Sprache von MBQC. Dabei verwenden wir den ZX-calculus, eine graphische Tensorschreibweise von Quantenzuständen und -operatoren, um die Schaltkreisdarstellungen in die Sprache von MBQC zu übersetzen und um die nötigen projektiven Ein-Qubit-Messungen für die Anwendung der Orakel darzustellen. Die entstandenen ZX-Diagramme stellen die benötigten Clusterzustände der Orakel dar. Die Messachsen lassen sich direkt aus den Diagrammen ablesen. Die Beschreibung dieser bekannten Algorithmen in der Sprache von MBQC soll dabei helfen, deren innere Mechanik besser zu verstehen. Des Weiteren können die Diagramme als Blaupause für eine experimentelle Umsetzung dienen.

Contents

0. Preface **8**

1. Introduction **12**
1.1. Measurement-based quantum computing and the ZX-calculus 12
1.2. Historical developement 14
1.3. Roadmap 16

2. Measurement-based quantum computing **18**
2.1. Short Introduction to quantum mechanics 18
2.1.1. Classical Computing 18
2.1.2. Quantum States 19
2.1.3. Quantum Gates 21
2.1.4. Quantum circuits 22
2.1.5. Deutsch-Josza algorithm 23
2.2. Manipulating information with measurements 24
2.2.1. Quantum teleportation 24
2.2.2. Cluster state 25
2.2.3. Measurements on cluster states 27
2.3. Measurement calculus 27
2.3.1. Introduction 27
2.3.2. Measurement patterns 28
2.3.3. Hadamard gate in the measurement calculus 30
2.3.4. Combinations of patterns 31
2.3.5. Standardization 32
2.3.6. Decoupling 33
2.3.7. Bridging nodes 34
2.3.8. Examples 35
2.3.9. Dependency theorems 38

3. ZX-calculus **40**
3.1. Rules of ZX-calculus 40
3.1.1. Nodes and edges 40
3.1.2. Rewriting rules 41
3.2. Translation of quantum circuits 43
3.3. ZX-calculus and MBQC 44
3.3.1. Measurements and cluster state as ZX-diagrams 45

3.3.2. Post-measurement corrections . . . 46
3.3.3. MBQC-protocol . . . 48
3.4. Advanced rules . . . 49
3.5. Related descriptions of quantum computing . . . 51
3.5.1. ZH- and other calculi . . . 52
3.5.2. Topological quantum computing . . . 53
3.5.3. Categorical quantum mechanics . . . 54

4. Algorithms in MBQC **56**
4.1. Deutsch-Jozsa algorithm . . . 56
4.1.1. Circuit formulation . . . 56
4.1.2. Two-qubit Deutsch-Jozsa . . . 58
4.1.3. Translation to ZX-calculus . . . 60
4.1.4. Algorithm for three qubits . . . 61
4.1.5. Translation to MBQC . . . 65
4.1.6. Algorithm on a rectangular lattice . . . 67
4.2. Simon algorithm . . . 68
4.2.1. Circuit formulation . . . 68
4.2.2. Oracle design . . . 71
4.2.3. Oracle generalization . . . 72
4.2.4. Factorization . . . 74
4.2.5. Translation to ZX-calculus . . . 75
4.2.6. n-qubit Simon algorithm . . . 78

5. Summary and Conclusion **80**
5.1. Deutsch-Jozsa algorithm . . . 80
5.2. Simon algorithm . . . 81
5.3. Conclusion . . . 82

Appendices **85**

A. Personal notes and acknowledgements **86**

B. Wissenschaftlicher Werdegang / academic resumé (German) **88**

0. Preface

Preceding the dissertation itself, we like to seize the opportunity to address a few words to the reader. We strongly encourage any reader to at least browse through this preface. Here, we will locate the topic of this thesis within the vast cosmos of theoretical physics and give a diagram to illustrate the cross-sections. Building on that, we will present the levels of difficulty that need to be surpassed to understand this thesis in its entirety. The learning path will also be presented in a diagram. After that, we provide a summary of the thesis in "simple" language. Although the thesis itself might not be easy to understand itself, even readers without education in physics shall at least gain a modest understanding about the topics at hand. Finally, we will briefly discuss the novelties that this thesis introduces.

To which field of physics does this thesis belong? In Fig. 1, we portray the subjects of physics that are covered in this thesis. The main part of the thesis is located in the cross-section of measurement-based quantum computing (MBQC) and the ZX-calculus. MBQC is a language to describe quantum computing, like the circuit model that is canonically used in textbooks. Quantum information and quantum information theory are the analoga to classical information theory and computing, but under the premise that information is stored in quantum objects (qubits). The ZX-calculus is a graphical, diagrammatical language to describe quantum states and operators as diagrams. There are other related calculi (mentioned in Sec. 3.5) and they all have their foundation in the description of monodial categories. To close the circle, category theory, as one of the most abstract frameworks within mathematics, can also be used to derive quantum mechanics without explicit representations that are canonically used in textbooks in order to ensure comprehensibility.

Is this topic difficult? This thesis was written with the intent that everyone with an education in physics from the level of a masters degree can comprehend at least the grand scheme of the thesis. For everyone without that degree of expertise in physics, there is a summary in simple language in the next paragraph. There are essentially four layers of the onion to the topic of this work. These important steps—that also roughly represent the authors learning path—are illustrated in Fig. 2. The first layer is a good understanding of quantum mechanics. The second layer is expertise in quantum computing and information theory. There is a very brief introduction to these topics in Sec. 2.1. This introduction is not extensive enough to build the foundations for the rest of the thesis, but may serve as a good recapitulation for any reader who feels a bit

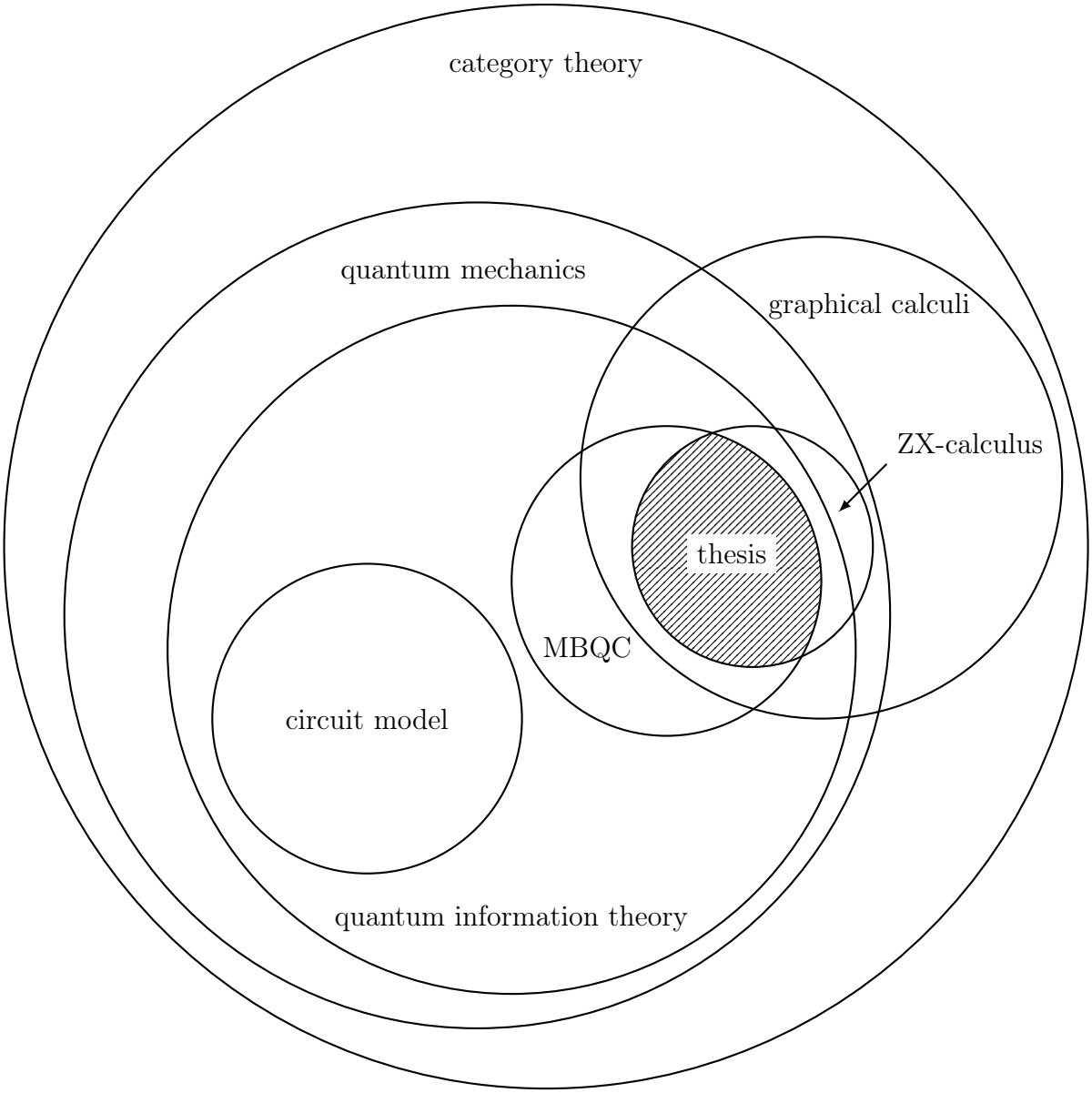


Figure 1.: Classification of the topic into the scheme of related subjects within physics and mathematics. The topic of the main part is hatched.

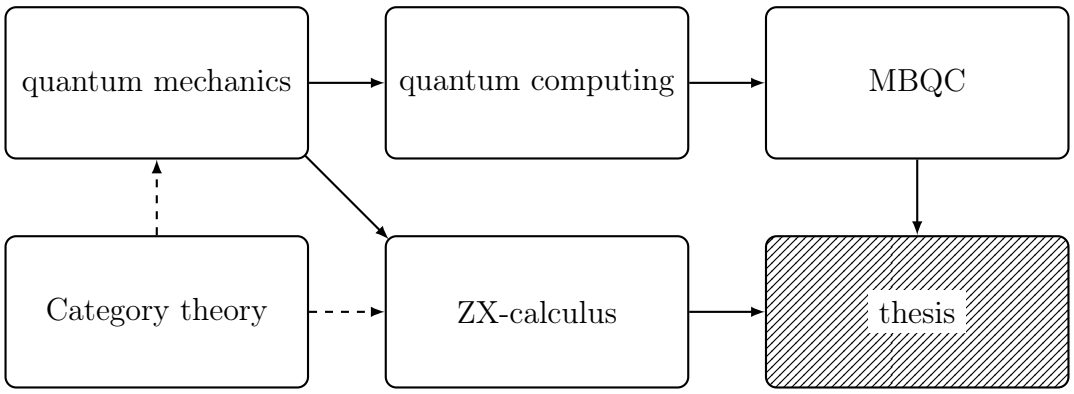


Figure 2.: Learning steps to the topic of this thesis. Dashed arrows represent a correlation that is typically not taught in courses and text books.

rusty in these subjects. It may also arouse interest for further reading in the reader with less expertise. The third layer of the onion is MBQC. We feel that our introduction to MBQC in Chap. 2, although limited to the purposes of this work, is self-contained and sufficiently comprehendible for any reader with a good understanding of (circuit-model) quantum computing. Obtaining a feeling for MBQC might be the biggest barrier because we are so familiar with the circuit model of quantum computing and therefore have to learn an entire new language. The fourth and final layer of the onion is the ZX-calculus as graphical language to describe MBQC and to formulate algorithms in terms of MBQC. This last layer may seem more intuitive since the graphical notation in terms of diagrams can serve as a pedagogically accessible theory. For chiefs who want to caramelize their onion, we state that there is a superordinate theory—category theory and categorical quantum mechanics—that can explain all the topics in this thesis from an ultimately fundamental point of view. Understanding of categorical quantum mechanics is definitely not necessary for understanding this thesis, but it is, nevertheless, mentioned as it forms the arch that connects "everything" related to quantum mechanics. Also, the author of this thesis aspires to delve deeper into this rabbit hole in the future. To come back to the question at the beginning of this paragraph: "difficult" is an inherently subjective word and should be used with care; however, we deem it fair to say that many who are not active in research in this very topic might categorize the topic as very abstract and convoluted, thus—if you will—difficult.

Can you explain the thesis in simple language? For everyone without major education in (theoretical) physics, we want to give a summary of the thesis in simple language. Everyday computers store information (for example images) on many small bits that are either 0 or 1. Bits in quantum computers (qubits) can also be 0 and 1, but, additionally, also a mixture of both. This difference leads to quantum computers solving some tasks faster. There are many programming languages to run a classical computer. Quantum computers can also be described with different languages. Usually, researchers use the circuit model, a language that is very similar to the languages on everyday computers.

Measurement-based quantum computing (MBQC) is a different language. MBQC can't do things better, only different. In MBQC, all the difficult steps that are "quantum" are done at the beginning and could be outsourced to a special quantum device. After that, there are only "easy" steps left that do not need quantum treatment. We write down MBQC in diagrams from the ZX-calculus. These diagrams can be used to illustrate all of the quantum world. In the main part of this thesis, we treat two quantum algorithms. They each solve a task faster than any everyday computer could. These tasks are rather theoretic and do, at this level, not solve any everyday problem. They could, however, be the basis for a real-life application of quantum computing in the future. The two algorithms are usually taught in the language of the circuit model. We translate the algorithms to the language of MBQC. These algorithms have not yet been formulated in this language. The results are diagrams that tell a scientist how to build an experiment to use these algorithms.

What is new in this thesis? The main parts of this thesis have already been published or submitted for peer review and publishing. In Ref. [1] we formulated a measurement-based version of the Deutsch-Jozsa algorithm using the ZX-calculus. While the description of MBQC with the ZX-calculus is not new, we have not found any rigorous translations in our background search. To our knowledge, our description of the three-qubit Deutsch-Jozsa algorithm in the language of MBQC has not been published before. A second publication is, at the time of submission of this thesis, in peer review, and is published as Ref. [2] on the preprint server. There, we develop a deterministic procedural algorithm to express any oracle of the Simon algorithm with CNOT- and X-gates. Subsequently, we formulated the two-qubit version of the Simon algorithm in the framework of MBQC explicitly and derived a systematic scheme to illustrate the resource state of the n-qubit version. We do not know of any of these results in literature. In both of the mentioned publication, the second author contributed through fruitful discussions, council and editorial work.

1. Introduction

Quantum computing will likely fundamentally change the landscape of computers in the medium-term future. When scientist achieve building quantum computers of sufficient size, many problems that require computing times of more than the lifetime of the universe on classical computers will be able to be solved in hours. The advantage is not simply that there is a speedup of existing processes, but that a variety of algorithms are sorted into completely different categories of complexity. Certain algorithms scale polynomially, that is, with a power of the system size on a quantum computer, whereas they would, for example, scale exponentially on a classical computer. Once achieved, this so-called ´´quantum supremacy" over classical computers will probably be of application not only in canonical computational algorithms, but will also extend to other fields in computer science such as cryptography, machine-learning, artificial intelligence, and medical research. Looking back on the development of the first computers in the middle of the 20th century and its subsequent domination of our civilisation, quantum computers might have the ability to once again revolutionize our work and our private lives.

Just as classical computing is not limited to silicon chips, quantum computers can be realized on a plethora of physical systems. Possible architectures range from trapped ions to topological qubits (quantum bits). In classical computation, programs can be written in a variety of programming languages, ranging from low-level to sophisticated, as they share the same power: being able to simulate any operation of a Turing machine. A Turing machine is a universal idealization of a classical computer; all modern (non-quantum) computers are mathematically equivalent to a Turing machine. Similarly, there are a variety of descriptions of quantum computing, all equivalent in their power of executing universal quantum computing. The canonical language of quantum computing is the well-known *circuit model* which represents the manipulation of Hilbert-space states in a gate-based visualization similar to classical logic gates. The circuit model is predominantly used in textbooks for the introduction of quantum computing because of its easy-to-understand composition of elementary elements of computation and because of its chronological representation of information flow from left to right.

1.1. Measurement-based quantum computing and the ZX-calculus

Measurement-based quantum computing (MBQC) is an alternate approach to quantum computing that is conceptually different than the canonically used circuit model. In

contrast to the circuit model, in which entangling multi-qubit gates effectuate the computing power of quantum mechanics, MBQC pre-encodes the entanglement in a highly entangled multi-qubit initial state that is prepared in advance [3]. Typically, this resource state is a cluster state, an eigenstate of the ZXZ-stabilizer that is the ground state of lattices with Ising interaction [4]. Use of other states, such as the two-dimensional AKLT state with spin greater than 3/2, are also possible, but the information processing protocol will then deviate from the canonical approach to MBQC [5, 6]. Single-qubit measurements on the entangled initial state are then sufficient to realize quantum computing in its full generality [3, 7, 8]. The MBQC model can lead to alternate experimental implementations of quantum algorithms that can, in some cases, be more flexible and efficient than circuit-based implementations. In addition, it is useful for highlighting and understanding certain fundamental aspects of quantum computing that are not equally prominent in a quantum- circuit-based representation.

From an experimental point of view, it is convenient that, in MBQC, the quantum-specific part of the process, the generation of entanglement, is carried out at the beginning. This preparation of the initial cluster state could be outsourced to a device whose sole purpose is to generate entanglement. The prepared state would then be handed over to a device that need only carry out one-qubit measurements. The latter are—in comparison with entangling multi-qubit gates in a circuit-based setup—easy to perform. The composite device would, in effect, function as a universal quantum computer. On a fundamental level, MBQC leads to a compelling alternate picture of quantum computing: one reason for the advantage of quantum computing over classical computing can, among other properties, be found in the necessity to generate a multi-qubit entangled state. Hence, the creation of such a state is generally difficult because it cannot be carried out by any classical device. The measurements carried out in the subsequent phase of MBQC can be interpreted as classical tools that utilize the previously created resource of quantum entanglement for calculational purposes. The pedagogical appeal of this framework is that the "quantumness" and the classical operations are clearly separated. This picture can potentially provide a better understanding of where the advantage over classical computing actually lies.

In other contexts, the circuit model might nevertheless be the more suitable framework to utilize. The set of tools for information processing in the circuit model is analogous to the logic-gate model of classical computing in which elementary logic gates are used as building blocks to compose a complex algorithm. The decomposition into a small set of fundamental elements as well as the resulting linear flow of information processing from left to right in a diagram are reminiscent of the canonical model of classical computing. In MBQC, the flow of information processing during a computation is much less evident due to the multi-dimensional architecture of a graph state. In addition, in the quantum circuit model, the actual effect of one gate on a set of qubits can be denoted much more compactly and understandably than its analog in MBQC, the effect of a single-qubit measurement on a highly entangled multi-partite state.

In MBQC, the key element for information processing is a sequence of measurements

on the qubits of a cluster state. The measurements are carried out along selected axes to realize a specific algorithm. Models such as the *measurement calculus* [9] or the *monad* description [10] have been formulated to try to encompass the complexity of MBQC. After gaining an understanding of MBQC through an introduction to the measurement calculus, we will transition to the graphical notation of the *ZX-calculus*, which is able to represent graph states as well as graph-like operators in a unified way [11]. This graphical calculus was introduced for carrying out derivations in multi-qubit quantum computation and information. On a computational level, the diagrams are tensor-network descriptions of both quantum states and quantum operators; in fact, the notation does not distinguish between the two. The ZX-calculus is also naturally suited to describe cluster states in arbitrary topologies and has a primitive notation to denote projective, single-qubit measurements. Therefore, it is a framework that is well-suited for describing MBQC in an intuitive manner.

The set of calculation rules that can be applied within the ZX-calculus are not only used for transforming ZX-diagrams into one another, but are also capable of translating an algorithm formulated in terms of quantum circuits to the language of MBQC [12]. The aforementioned rules are equations that tell the user how to transform ZX-diagrams. On a low level, they simply represent tensor equations. They range from single-node identities to large-scale node and edge manipulations in the graph of a ZX-diagram. One application for these rules is to simplify diagrams and thus to optimize quantum circuits [12]. Another application of the ZX-calculus is to translate quantum circuits into the language of ZX-diagrams.

1.2. Historical developement

The history of quantum computing reaches back almost half a century. In 1980, Paul Benioff proved that computing on quantum-mechanical particles is theoretically possible and that it can be at least as powerful as a Turing machine [13, 14]. Two years later Richard Feynman [15] realized that information processing and computing on qubits can be more powerful than classical computing in some cases. It was later found that some algorithmic problems that are categorized to be in the classical non-polynomial-time complexity class fall into a polynomial-time quantum complexity class [16]. This means that some computational problems, like integer factoring, experience an exponential speedup when transitioning from classical to quantum computers [17]. The prospect of quantum supremacy led to many quantum algorithms being developed for potential quantum computers in the late 20th century. These algorithms all have the distinct feature of solving problems with fundamentally fewer operations than classical algorithms would need.

Among these, the *Deutsch-Jozsa-Algorithm* was one of the first quantum algorithms to be proven to be computationally superior to functionally equivalent algorithms on classical computers; in fact, it was specifically designed to show the power of quantum computing. A one-qubit precursor to the algorithm was originally proposed by David

Deutsch [18]; however, this original variant is not deterministic. The algorithm was subsequently extended to treat n qubits, to be deterministic, and to be less demanding computationally by Deutsch and Jozsa [19] and by Cleve *et al.* [20]. In the present day, the Deutsch-Jozsa-Algorithm is typically used as a useful introduction to the theoretical superiority of quantum computers over classical computers.

While the Deutsch-Jozsa-Algorithm is typically the first one treated in introductions, the algorithm proposed by Daniel Simon in 1993 and published in 1994 was the first quantum algorithm that was proven to be exponentially faster than any classical computation on the same problem [21]. The algorithm solves a "promise problem" in which a "black-box" oracle executes a function with a property that is not known initially but is promised to lie within a given set of properties. The specific proposal by Simon was to find the period of a function, that is, the numerical distance between two inputs that trigger the same output. The algorithm was later generalized to solve a general *hidden subgroup* problem on arbitrary groups [22, 23]. Even though Simon did not fully realize the significance of his development immediately, the Simon algorithm subsequently inspired Peter Shor to apply a similar theoretical idea to the problems of factoring and of discrete logarithms [24]. Eventually, this lead to the development of Shor's famous algorithm for prime factorization, a lighthouse in the realm of quantum algorithms [17].

While the circuit model of quantum computing was utilized starting from the early stages of research in quantum computing due to its similarity to classical logic-gate models [25], alternative paradigms such as quantum annealing [26–30], adiabatic quantum computing [31], quantum walk [32, 33], and topological quantum computing [34] have since been formulated. One-way quantum computing on a cluster state by means of single-qubit projective measurements was first theorized by Robert Raussendorf and Hans Briegel at the beginning of this century [7]. Following their basic description of quantum computing on these highly entangled states, generalized entangled graph states, of which cluster states are one instance, have been incorporated into the picture of MBQC [35], and algorithms have been formulated and experimentally executed using the methods of MBQC [36].

In the search for alternative descriptions of quantum computing alternative to the circuit model, graphical calculi were introduced in the second decade of this century. Rooted in the formulation of categorical quantum mechanics (the description of quantum mechanics using category theory) [37–39], the ZX-calculus was proposed by Bob Coecke and Ross Duncan in 2011; it was already mentioned in their original paper that it can be utilized to describe MBQC in an intuitive manner [11]. Other graphical calculi such as the ZH-calculus [40] and the scalable ZX-calculus [41] have since been introduced, each having its own strength and use of application.

1.3. Roadmap

Our goal will be to systematically derive measurement-based formulations of the Deutsch-Josza and the Simon algorithms using the graphical framework of the ZX-calculus. We know of no such descriptions in the published literature.

We will start by introducing MBQC in Chap. 2. At the beginning of the chapter, there will be a very short recapitulation of the most important aspects of quantum computing and of the circuit model. After identifying, as an example, quantum teleportation as a manipulation of quantum information with projective measurements, we will introduce a formal calculus for MBQC, the *measurement calculus*. This calculus will serve as a first complete description of MBQC and will include defined resource states and a set of transformation rules to execute any quantum computation with projective measurements only. The measurement calculus is formulated in terms of text and symbols. As examples, we will represent basic quantum gates such as the Hadamard gate in the measurement-based language.

In Chap. 3, we will transition to the graphical ZX-calculus. We will introduce the basic ingredients of ZX-diagrams, green and red spiders with their respective legs as connectors. Building on this tensor description of quantum states and operators, we will describe the basic rules of transforming ZX-diagrams, which are typically used to reduce the complexity of diagrams. An important step will then be to realize that ZX-diagrams of a specific, restricted form can be understood to be blueprints for MBQC. After giving a basic example of MBQC using the ZX-calculus, we will introduce two advanced rules of the language based on methods from graph theory.

Using the framework MBQC and the ZX-calculus, we will translate algorithms to MBQC in Chap. 4. We will start by introducing the non-canonical version of the Deutsch-Jozsa algorithm in the circuit model, that is, the version of the algorithm that does not require auxiliary qubits. After translating the two-qubit algorithm to MBQC using the ZX-calculus, we will discuss a general method for formulating the effect of the oracle of the Deutsch-Jozsa algorithm. This formalism will then be utilized to derive a MBQC-version of the three-qubit Deutsch-Josza algorithm in the language of ZX-diagrams. We will observe that the algorithm gains a new degree of complexity for more than two qubits. The result will be a special geometric graph cluster state and a recipe for choosing a set of measurements to execute the Deutsch-Jozsa algorithm for any instance of the oracle. Finally, we will also formulate a version of the algorithm that is based on a rectangular cluster-state lattice, which might be advantageous for experimental work.

For the Simon algorithm, we will start by introducing the canonical circuit-based formulation of the algorithm. We will then formulate a novel deterministic algorithm to translate any periodic function to an oracle circuit of the Simon algorithm that consists of CNOT- and X-gates only. This algorithm will find its application when we subsequently translate the two-qubit Simon algorithm to the framework of MBQC explicitly. The result will be a ZX-diagram that incorporates both the form of the required cluster state as well as a recipe for single-qubit measurements that realizes the algorithm in MBQC.

Building on that, we will exhibit the topology of the graph state that is required to implement the n-qubit version. We will give explicit diagrams that implement the 2-, 3- and 4-qubit versions and will explain how the algorithm gains complexity in the sense of the increase in the number of cross sections within the required graph state pictured in two dimensions.

Finally, in Chap. 5, we will discuss the key aspects of this work and give an outlook for potential future work.

2. Measurement-based quantum computing

The main goal of this chapter is to introduce the concept of *measurement-based quantum computing (MBQC)*. We will first repeat some of the basic properties of classical and quantum information theory. This repetition will serve as a reminder of the key mathematical aspects to the reader as well as a clarification of the nomenclature that is used in this work. We will also introduce the well-known Deutsch-Jozsa quantum algorithm as an example. This is one of the two algorithms that will be the center of this work and will be discussed in Chap. 4. After covering the canonical circuit description of quantum computing, we will transition to a measurement-based description and introduce the *measurement calculus*, a framework that can formalize the steps executed in MBQC. Having understood the measurement calculus, it will be easier to comprehend the treatment of MBQC within a graphical language, which we will portray in Chap. 3 and Chap. 4.

2.1. Short Introduction to quantum mechanics

We will start with a very brief recapitulation of quantum mechanics. In doing this, we will first describe classical computing in a few sentences and then compare that to quantum computing. After recalling how information is processed on quantum objects, we will give a very compact introduction to the canonical circuit model of quantum computing, including quantum states and gates. Finally, we will give the Deutsch-Jozsa algorithm as a typical example of a quantum algorithm, an algorithm that will be translated to MBQC in Chap. 4.

2.1.1. Classical Computing

In classical dynamics, properties of an object can typically be described on a continuous scale. For example, a bicycle rider can ride at a speed of 8 km/h or at a speed of 30 km/h. Much larger speeds, such as 1000 km/h, while not realistic from the perspective of physiology, are not prohibited by physics as long as we do not reach the speed of light. Every speed between the mentioned values is also possible. The rider could ride at a speed of 8.782 km/h.

In classical computing, information is coded as bits. These are distinguishable classical states that can be either 0 or 1. Larger systems can store multiple bits and, as such,

more information. For example, the bitwise representation of the decimal number 9 is given by

$$9_{10} = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 1001_2 \tag{2.1.1}$$

and requires at least four bits to store. Sequences of bits can also be used to express larger pieces of information such as text, photos or videos.

Information on bits can be altered by logic gates. An example of a logic gate is the NOT- or X-gate, which flips the information on one bit from 0 to 1 or from 1 to 0. There are many more logic gates, but it is important to state that it is sufficient to have access to one specific two-bit gate in order to simulate any other logic gate. It is therefore called "functionally complete" [42]. This universal classical gate is the NAND-gate (*not AND*)

$$\mathrm{NAND}(b_1, b_2) = \begin{cases} 0, & b_1 = b_2 = 1 \\ 1, & \text{else} \end{cases} \tag{2.1.2}$$

which returns 0 (false) if and only if both of the two input bits is in the state 1.

Solving problems that are written in terms of bits requires algorithms, for example, an algorithm for factoring a natural number. Depending on the size n of the problem (an example would be the n digits of a natural number), the algorithm will require N steps. Usually, the complexity of an algorithm is expressed in terms of a bounding scaling, where $O(2n^2)$, for example, means that of the order of $2n^2$ steps are required to solve the given problem. Some problems can be solved in fewer steps with quantum computing.

2.1.2. Quantum States

In contrast to classical mechanics, which describes macroscopic scales, quantities on microscopic scales typically must be quantized. For example, the angular momentum of a small object such as an electron bound in an atom can only come in multiples of some small fundamental value. That is, any value between those discrete values is prohibited by the nature of quantum mechanics. Consequently, incrementing the value of a quantum mechanical quantity is only possible in discrete quanta, in contrast to the bicycle, which could change its speed by any small amount.

The state of a property that is described by quantum mechanics is represented by a vector in a *Hilbert space*. The latter is a complex inner-product space that is made into a complete metric space by a distance function that arises from the inner product [43]. In this work, we use the *bra-ket* notation; a vector v in a Hilbert space is defined by a so-called *ket* $|v\rangle$.

To give an example, consider a two-level system, which can either be in the state 0 or 1. From an information-theoretical standpoint, this can be considered a *bit*. Thus, in quantum mechanics, we call it a *qubit* (quantum bit). We denote a general state of the system as

$$|\psi\rangle = \alpha\,|0\rangle\ + \beta\,|1\rangle\ , \tag{2.1.3}$$

where α and β are complex amplitudes describing "how much" of the system is in the respective basis state $|0\rangle$ or $|1\rangle$. The amplitudes are normalized as $|\alpha|^2 + |\beta|^2 = 1$.

The dual elements to kets are *bras*. A bra $\langle\alpha|$ is defined as the hermitian conjugate to a ket, i.e.,

$$\langle\alpha| = |\alpha\rangle^\dagger \ , \tag{2.1.4}$$

and $\langle\alpha|\ |\alpha\rangle = \langle\alpha|\alpha\rangle = 1$.

As expressed in Eq. (2.1.3), a quantum state can be in a superposition of basis states. This is in contrast to classical systems that are always in a distinct (classical) state. The amplitudes of a quantum state also determine the probability of a measurement outcome. If we measure a quantum state, we can only detect it to be in one of the basis states of the measurement. For example, in Eq. (2.1.3), we can only ever detect (e.g., see, feel, smell, measure) $|0\rangle$ or $|1\rangle$ if we measure in this computational basis. The probability of measuring $|0\rangle$ is given by $|\alpha|^2$, the probablity of measuring $|1\rangle$ by $|\beta|^2$.

Larger quantum states are constructed via the tensor product $\otimes$. The tensor product describes all possible combinations of basis states of multiple smaller systems, each equipped with an amplitude corresponding to the product of amplitudes of the basis states in the original system. Put simply, we effectively glue together states (or operators). As an example, consider two of the quantum bits (or *qubits*) of Eq. (2.1.3). The *product state* of the two qubits is formed as

$$\begin{aligned}
|\phi\rangle &= |\psi_1\rangle \otimes |\psi_2\rangle \\
&= \big(\alpha_1 |0_1\rangle + \beta_1 |1_1\rangle\big) \otimes \big(\alpha_2 |0_2\rangle + \beta_2 |1_2\rangle\big) \\
&= \alpha_1\alpha_2 |0_1\rangle \otimes |0_2\rangle + \alpha_1\beta_2 |0_1\rangle \otimes |1_2\rangle + \beta_1\alpha_2 |1_1\rangle \otimes |0_2\rangle + \beta_1\beta_2 |1_1\rangle \otimes |1_2\rangle \\
&\equiv \alpha_1\alpha_2 |0_1 0_2\rangle + \alpha_1\beta_2 |0_1 1_2\rangle + \beta_1\alpha_2 |1_1 0_2\rangle + \beta_1\beta_2 |1_1 1_2\rangle \\
&\equiv \alpha_1\alpha_2 |00\rangle + \alpha_1\beta_2 |01\rangle + \beta_1\alpha_2 |10\rangle + \beta_1\beta_2 |11\rangle \ ,
\end{aligned} \tag{2.1.5}$$

where we have denoted the tensor products of basis states without the subscripts as it is evident by their order.

A unique property that multi-partite quantum states can have is *entanglement*. They can be in an intertwined multi-qubit state in which there is no clear description of the state of each of the qubits alone. As an example, consider the two-qubit state

$$\frac{1}{\sqrt{2}}\big(|00\rangle + |11\rangle\big) \ . \tag{2.1.6}$$

The state is formed so that the two qubits are always in the same state, but, since it is a superposition of the both-zero state and the both-one state, a descripion of each qubit's state individually is impossible. Such states have no equivalent in classical mechanics.

A quantum state cannot directly be observed. A physical system can be measured and render an outcome. The outcome of the measurement is determined by chance with a probability given by the squared amplitudes of the possible outcomes. Once a system

is measured, it collapses exactly to the state that corresponds to the outcome. This projection onto one of the eigenstates of the projective measurement operator is called a *projective measurement*.

In applications, qubits can be realized using a wide range of physical systems. A historical example would be a system consisting of two energy levels in an atom and the excitation of the atom could be caused by a photon. In quantum computing, the dominant technologies at the current time are superconducting qubits [44], optical qubits (polarization) [45], and topological qubits [34].

2.1.3. Quantum Gates

Just as gates in classical computing manipulate the information stored in classical bits, quantum gates alter the information stored in qubits. Quantum gates on qubits are represented by special unitary matrices. Such a gate $U \in SU(2)$ satisfies $U^\dagger U = \mathbb{1}$ and $\det(U) = 1$. Its key properties are that it preserves normalization and that it is reversible. The Lie-Group $\mathfrak{su}(2)$ that generates $SU(2)$ has a representation

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \tag{2.1.7}$$

which is called the *Pauli basis*. Thus, the Pauli matrices can generate any special unitary matrix U via the operation

$$U = \exp\Big(\mathrm{i}\left(aX + bY + cZ\right)\Big), \qquad (a, b, c \in \mathbb{R}) \tag{2.1.8}$$

and, as such, any change in the information content of a single qubit.

The *Clifford group* normalizes the Pauli-group consisting of X, Y, Z, and $\mathbb{1}$ [46]. The underlying set consists of the Hadamard gate

$$H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \tag{2.1.9}$$

the phase gate

$$S = \begin{pmatrix} 1 & 0 \\ 0 & \mathrm{i} \end{pmatrix}, \tag{2.1.10}$$

and the 2-qubit controlled-X (or controlled-NOT) gate

$$\mathrm{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{2.1.11}$$

The Clifford set alone, accompanied by single-qubit rotations generated by the Pauli set,

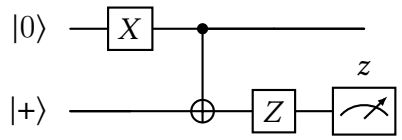


Figure 2.1.: Example of a quantum circuit with initial states, gates and final measurement.

has computational power equal to that of a *Turing machine*. A Turing machine can be efficiently simulated by a classical computer [46]. When augmented by the phase-shift gate

$$T = \sqrt{S} = \begin{pmatrix} 1 & 0 \\ 0 & \mathrm{e}^{\mathrm{i}\,\pi/4} \end{pmatrix} , \tag{2.1.12}$$

or, in fact, any non-Clifford gate, the combined set is *universal* for quantum computing. This means that any computation in the realm of quantum information theory can be performed by a combination of gates from this set, which is sometimes called the *Clifford+T* set [46].

2.1.4. Quantum circuits

Quantum circuits are the quantum analog to classical computational circuits. They describe the action of a network of gates on an initial state. After the information is processed, some part of the information is read out. An example circuit is depicted in Fig. 2.1. The informational flow is read from left to right. The kets on the left describe the initial state of the qubits. In the case of Fig. 2.1 it is

$$|\psi_0\rangle = |0\rangle\,|+\rangle = \frac{1}{\sqrt{2}}\big(\,|00\rangle + |01\rangle\,\big) . \tag{2.1.13}$$

Typically the initial state is a product state, which has no entanglement.

From left to right, the state of the qubits is altered by quantum gates. These are the logic gates of quantum computing and are, as stated above, special unitary operations. Gates are represented by rectangles. The X-gate in Fig. 2.1 flips the basis states of the first qubit. As an example of a two-qubit gate, Fig. 2.1 also contains a CNOT-gate which was introduced in Eq. (2.1.11). It is depicted by the connector between the two lines and the $\oplus$ symbol on the target qubit, describing the controlled addition modulo 2.

Any computation needs to be concluded by a readout; otherwise, the manipulation of the information would not be detectable. This is even more the case in quantum mechanics than in classical computation because a quantum state can only collapse if a measurement is carried out. Measurements are depicted in the circuit language as boxed scales. As an example, in Fig. 2.1, the last element of the second row indicates that the second qubit is eventually measured in the computational basis $\{|0\rangle, |1\rangle\}$. The results of the measurements directly imply the results of the algorithm.

	Variants			
	(i)	(ii)	(iii)	(iv)
$f(0)$	0	1	0	1
$f(1)$	0	1	1	0

Table 2.1.: Output variants of a one-bit boolean function. Variants (i) and (ii) are constant, while variants (iii) and (iv) are balanced.

2.1.5. Deutsch-Josza algorithm

As an example of how quantum computing works, we describe here the *Deutsch-Jozsa-Algorithm*, which was one of the first quantum algorithms to be proven to be computationally superior to functionally equivalent algorithms on classical computers; in fact, it was specifically designed to show the power of quantum computing. As one of the main parts of this thesis, we will translate the quantum circuit description of this algorithm to a measurement-based description in Chap. 4.

A one-qubit precursor to the algorithm was originally proposed by D. Deutsch [18]; however, this original variant is not deterministic. The algorithm was subsequently extended to treat n qubits, to be deterministic, and to be less demanding computationally by Deutsch and Jozsa [19] and by Cleve *et al.* [20]. In the present day, the Deutsch-Jozsa-Algorithm is typically used as a useful introduction to *quantum advantage*, the theoretical superiority of quantum computers over classical computers.

The aim of the algorithm is to determine if a function is *constant* or *balanced*. We take a function $f : Z_2^n \ni \boldsymbol{\sigma} \to f(\boldsymbol{\sigma}) \in Z_2$ that maps a binary representation $\boldsymbol{\sigma}$ of length n to one bit $f(\boldsymbol{\sigma})$. We call $f(\boldsymbol{\sigma})$ *constant* if the outcome is the same for all choices of $\boldsymbol{\sigma}$, i.e., $\forall \boldsymbol{\sigma} \in Z_2^n : f(\boldsymbol{\sigma}) = 0$ or $\forall \boldsymbol{\sigma} \in Z_2^n : f(\boldsymbol{\sigma}) = 1$. We call the function *balanced* if exactly one-half of the outcomes are 0 and one-half are 1, i.e., $|\{\boldsymbol{\sigma} \in Z_2^n : f(\boldsymbol{\sigma}) = 0\}| = |\{\boldsymbol{\sigma} \in Z_2^n : f(\boldsymbol{\sigma}) = 1\}|$. It is important to emphasize that the algorithm is based on the premise that f can only be constant or balanced and nothing else. The possible variants of f are tabulated in Table 2.1.

It is easy to show that a quantum computer can solve this problem with less work than a classical computer. Suppose we have an oracle that returns the bit $f(\boldsymbol{\sigma})$ for any $\boldsymbol{\sigma}$ that we feed into it. A classical computer requires $2^{n-1} + 1$ queries of the oracle to solve the problem in the worst case, in which f is balanced, but the first $2^n/2 = 2^{n-1}$ queries all yield the same output. On a quantum device, one can determine the character of the function f with just one call to the oracle, i.e., at constant cost.

The canonical way of formulating the Deutsch-Jozsa algorithm makes use of an auxiliary qubit. Here, however, we will work with a variant of the algorithm that uses no auxiliary qubit, but only the query qubits. In Chap. 4, we will profit from the reduced number of qubits as we translate the algorithm to measurement-based quantum computing.

The single-qubit-input variant of the one-bit Deutsch-Jozsa algorithm is depicted in

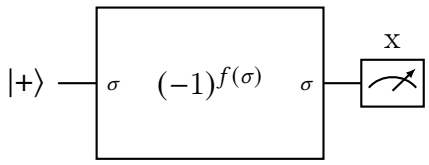


Figure 2.2.: Deutsch-Jozsa algorithm for one qubit. The oracle adds a phase of -1 to each basis element $|\sigma\rangle$ when $f(\sigma) = 1$. Measurement of $|+\rangle$ reveals that f is constant, $|-1\rangle$ that f is balanced.

Fig. 2.2. (All quantum circuits were typeset with the help of the *Quantikz*-package [47].) We prepare one working qubit in the state $|+\rangle = |0\rangle + |1\rangle$ (omitting normalization here and throughout this work). We then apply an oracle operation to the working qubit, where the effect of the oracle depends on the character of the instance of the function f under investigation. If $f(\sigma)$ is 1, then the oracle adds a phase of -1 to the projection onto the basis state $|\sigma\rangle$. That is, it transforms the input state as

$$|+\rangle = |0\rangle + |1\rangle \xrightarrow{\text{oracle}} (-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle \ . \tag{2.1.14}$$

The oracle can be realized by the identity operation if f is constant and by a Pauli-Z-gate if f is balanced.

After the application of the oracle, the only remaining step is to measure the working qubit in the x-basis. If the measurement yields the outcome corresponding to the basis state $|+\rangle$, then f is constant. For the other outcome, f is balanced. In the one-qubit case, it can easily be checked that the oracle of Eq. (2.1.14) will transform $|+\rangle$ to $\pm|+\rangle$ if and only if $f(0) = f(1)$. Note that the overall phase of ± 1 does not have physical relevance and cannot be measured. If, on the other hand, $f(0) \neq f(1)$, the oracle will transform $|+\rangle$ to $\pm|-\rangle$. Hence, a measurement in the x-basis will determine the character of f.

2.2. Manipulating information with measurements

In the circuit model of quantum computing, quantum states of information are manipulated by quantum gates. It is sometimes overlooked that measurements can also be used to manipulate the information that is stored on qubits. We will now show that measurements can not only serve as an operation for implementing the readout at the end of an algorithm, but also as a means of manipulating the information itself.

2.2.1. Quantum teleportation

As an example for a measurement-based manipulation of quantum information, we will describe a scheme for teleporting one qubit's worth of information from one physical qubit to another. The quantum circuit of the teleportation is depicted in Fig. 2.3.

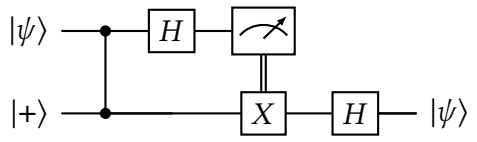


Figure 2.3.: Teleportation in the circuit model of quantum computing [48]. The state $|\psi\rangle$ is transported from the upper to the lower qubit by a combination of quantum gates. The two most important parts of the algorithm are the application of the controlled-phase gate and the measurement of the first qubit along the z-axis, followed by an X-gate only if the outcome of the measurement is 1. Loosely drawn after Ref. [8].

The first qubit is in the state $|\psi\rangle$ that is to be teleported. The second qubit is prepared in the superposition state $|+\rangle = |0\rangle + |1\rangle$ (normalization omitted). A controlled-phase gate (CPHASE = $|00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 10| - |11\rangle\langle 11|$) is applied to the two qubits. The controlled-phase gate has the property of only negating the amplitude of the sub-Hilbert-space in which both qubits are in the $|1\rangle$ state.

The first qubit is then rotated by a Hadamard-gate $H = |0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|$ and, finally, measured along the z-axis. The Hadamard gate is a basis change from the computational basis (eigenstates of the Pauli-z-matrix) to and from the x-basis $\{|+\rangle, |-\rangle\}$ (eigenstates of the Pauli-x-matrix). It is therefore equivalent to replace the H-gate and the z-measurement by an x-measurement, collapsing the state of the first qubit to either $|+\rangle$ or $|-\rangle$.

The measurement is the control switch of the X-gate on the second qubit. If the outcome of the measurement is 1, corresponding to the collapsed state $|1\rangle$, the second qubit is altered by the X-gate ($X = |0\rangle\langle 1| + |1\rangle\langle 0|$). If the outcome is 0, it is left as it is. After applying another Hadamard gate to the second qubit, it is easy to check that it is now exactly in the state $|\psi\rangle$. Thus, the state $|\psi\rangle$ was teleported from qubit 1 to qubit 2 using a measurement.

We note that the the X- and H-gates on the second qubit are actually only single-qubit post measurement corrections. Single-qubit gates are, in general, easy to execute and can efficiently be simulated on a classical computer. The elements of the teleportation that are unique to quantum computing are thus the entangling gate CPHASE—which leads to an entangled two-qubit state—and the measurement that collapses a multi-partite state.

2.2.2. Cluster state

We have learned in the last subsection that an entangled state is a prerequisite for the execution of teleportation through measurement. There are a variety of entangled states that make universal quantum computing possible through measurement [49, 50]. They share the property that they need to have "enough" entanglement. However, they also cannot possess "too much" entanglement [51]. The most famous of these suitable states is the so-called *cluster state*, a multi-partite entangled graph state that is an eigenstate

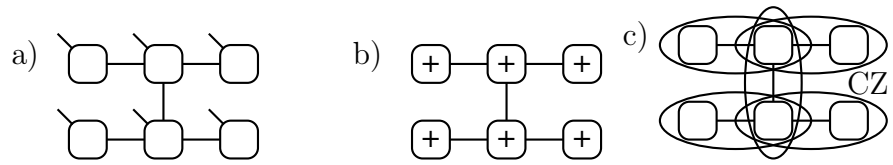


Figure 2.4.: Initialization of a cluster state. a) Graph configuration of the qubits. Protruding edges symbolize degrees of freedom. b) Prepare all qubits in the $|+\rangle = |0\rangle + |1\rangle$ state (normalization omitted). b) Apply controlled phase gates (CZ) to all pairs of qubits that are adjacent inside the graph.

of the ZXZ-stabilizer [52].

A cluster state on a network of qubits can be realized in two steps, also shown in Fig. 2.4 [8]. First, we initialize all qubits to be in the $|+\rangle = |0\rangle + |1\rangle$ state (normalization omitted). This may be achieved by applying Hadamard gates to all qubits in an all-$|0\rangle$ state. Second, apply controlled phase gates

$$\mathrm{CZ} = \mathrm{CPHASE} = |00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 10| - |11\rangle\langle 11| \tag{2.2.1}$$

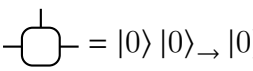
to all pairs of qubits that are neighbors in the network. The latter means that the two physical qubits are connected through a bond (or edge, in the language of graphs). Via this procedure, we obtain a state with the same amplitude for each basis vector of the multipartite Hilbert space. For every pair of adjacent qubits inside the graph, the amplitude of the basis states in which both of them are in the state 1 is negated once. On a chain of three qubits, for example, the (unnormalized) cluster state reads

$$|\mathrm{Cluster}_{3\text{-chain}}\rangle = |000\rangle + |001\rangle + |010\rangle - |011\rangle + |100\rangle + |101\rangle - |110\rangle + |111\rangle \ . \tag{2.2.2}$$

We can also represent the cluster state in a fashion that better resembles the information flow when we start computing with measurements. We write each qubit inside the graph as as a tensor

$$A^k = |k\rangle\,|k\rangle_{\rightarrow}\,|k\rangle_{\uparrow}\,\langle +|_{\leftarrow}\,Z^k\,\langle +|_{\downarrow}\,Z^k\,, \tag{2.2.3}$$

where the arrows describe the direction of the connection in the graph—also called virtual bond in the 2D tensor network. The superscripts on the right side of Eq. (2.2.3) imply a power of 0 or 1. A ket with no index represents a physical bond, that is, the degrees of freedom of the qubit where $k \in \{0, 1\}$. If a virtual bond does not exist for a specific tensor, then the corresponding ket or bra is also missing in Eq. (2.2.3). The bottom center tensor in the example cluster state in Fig. 2.4 can be written as


$$= |0\rangle\,|0\rangle_{\rightarrow}\,|0\rangle_{\uparrow}\,\langle +|_{\leftarrow} + |1\rangle\,|1\rangle_{\rightarrow}\,|1\rangle_{\uparrow}\,\langle -|_{\leftarrow} \ . \tag{2.2.4}$$

2.2.3. Measurements on cluster states

In the upcoming sections and chapters, single-qubit measurements inside cluster states will take on a prominent role. Having defined single-qubit tensors in Eq. (2.2.3), we can also define a measurement of the degrees of freedom of such an individual qubit. The projective measurement operator

$$M_\alpha = |\alpha\rangle \langle\alpha| \tag{2.2.5}$$

corresponding to output α acts on the physical degree of freedom of the qubit tensor. Hence, on the cluster state tensor in Eq. (2.2.3), the measurement yields the new state

$$M_\alpha A^k = \langle\alpha|k\rangle |k\rangle_\rightarrow |k\rangle_\uparrow \langle+|_\leftarrow Z^k \langle+|_\downarrow Z^k \,. \tag{2.2.6}$$

As an example, say we measure the qubit in Eq. (2.2.4) in the x-basis and the outcome is 1, corresponding to the projection $M = |-\rangle \langle-| = |0\rangle \langle 0| - |0\rangle \langle 1| - |1\rangle \langle 0| + |1\rangle \langle 1|$. The resulting tensor for the single qubit is then

$$MA = |-\rangle \Big(|0\rangle_\rightarrow |0\rangle_\uparrow \langle+|_\leftarrow - |1\rangle_\rightarrow |1\rangle_\uparrow \langle-|_\leftarrow \Big) \,. \tag{2.2.7}$$

As is obvious from Eq. (2.2.7), the degree of freedom has collapsed to $|-\rangle$. The remaining tensor structure, i.e., the entanglement with the rest of the graph, which is encorporated in the term in parentheses, has also been altered. It is thus evident that a projective single-qubit measurement on a cluster state alters the information that is stored in the state as entanglement. We will show in the upcoming sections that projective single-qubit measurements are sufficient for executing universal quantum computations on cluster states.

2.3. Measurement calculus

2.3.1. Introduction

We have learned in the last section that it is possible to process information solely using measurements. Building on that, we will introduce a calculus in this section that demonstrates the workings of computing with cluster states. The *measurement calculus*—similarly to classical calculus—has its own irreducible objects and calculation laws. These encompass all possible calculations using only projective single-qubit measurements on cluster states. They form a set that is universal for quantum computing.

Since there is an unlimited number of combinations and variants of measurements and operations on a multipartite system, it would be convenient to have a uniform description of these actions. Danos *et al.* have proposed a calculus that is specific to the possibilities and needs of measurement-based quantum computing (MBQC) [9]. Similar to a programming language, they provide a distinct set of possible elementary gates as well as the resources for executing computations.

The measurement calculus encompasses everything from preparing the system to reading out the results. Preparation is carried out by setting a subset of all qubits to the input state. Additional auxiliary qubits needed for the calculation are uniformly set to a suitably defined initial state. The calculation is then performed using an elementary set of actions that include entangling pairs of qubits, local measurements on single qubits, and rotating qubits as corrections. After a set of elementary actions is carried out in a suitable order, an output state is created on a subset of the qubits. This subset can also overlap with the subset of input qubits. The collection of input, calculation, and output spaces as well as the calculation commands is called a *pattern*. Patterns represent unitary operations on quantum states.

As stated in the beginning of this section, there are an unlimited number of operations that can be realized using MBQC. It is convenient to standardize the patterns in order to compare and concatenate them. Danos *et al.* give several rewrite rules that let us transform patterns into a standardized form [9]. The convention is to have patterns in the *NEMC-form*. This form first prepares all qubits (N), then entangles pairs (E), executes measurements (M) and finally corrects remaining qubits. This leaves us with distinct sets of different operations, thus making it possible to prepare a resource state in advance and to subsequently calculate on it using local projective measurements. It also generates distinct building blocks for common quantum operations and algorithms in terms of MBQC.

The standardization also has the effect that it shows how to reduce computational complexity to a minimum through commutation of operators. In MBQC, the axes of measurements are, in general, dependent on the outcomes of earlier measurements. This means the algorithm has several layer, each depending on earlier layers. In general, the dependency schema can be arbitrary complex. However, by transforming a pattern to NEMC-form, one also obtains a form with minimal computational complexity. That is, no pattern that realizes the same unitary has a lower complexity. This is also useful for determining the computational depth of certain actions.

2.3.2. Measurement patterns

We will start by introducing one-dimensional patterns. Such patterns are capable of altering one qubit worth of information and can be realized on a string of pairwise connected qubits. The two-dimensional case works in similar fashion in terms of semantics. However, patterns of more than one dimension can realize quantum computations for arbitrarily large informational systems.

We first define the *space* of a pattern. It consists of an input, a calculation and an output space. The *input space* $\mathcal{I}$ is prepared to be in the initial state of the computation. In one dimension, this is only one qubit because we can only calculate with one virtual qubit. In general, the initial state can consist of any number of qubits. The output space $\mathcal{O}$ is the set of qubits in which the result of the calculation is stored. The same considerations regarding its size as for the input space apply. The input and output

spaces can overlap or can even be identical. The computation space C is the set of qubits on which the actual calculations are performed. They are manipulated using a certain set of operations that we define in the next paragraph, namely, measurements and corrections. For a non-trivial calculation, C has to overlap with both I and O. The n qubits in the total space $C = I \cup C \cup O$ are numbered from 1 to n. An example would be the total space

$$C = I \cup C \cup O = \{1\} \cup \{1, 2, 3\} \cup \{3\} \tag{2.3.1}$$

consisting of an input at 1, computations at 1, 2, and 3 as well as an output at 3.

The available commands are the preparation of individual qubits (N), the entangling of pairs of qubits (E), single-qubit projective measurements (M), and local corrections (C). Combinations of elements from this set can be used to realize universal quantum computing.

Preparation The preparation command N_i sets qubit i to the state $|+\rangle = |0\rangle + |1\rangle$ (normalization omitted). Later, we will just assume that all qubits other than the input qubits have been prepared by N and will omit the explicit preparation.

Entangling The entangling operation E_{ij} acts as an entangling two-qubit CPHASE (CZ) gate operating on qubits i and j, that is,

$$E_{ij} = |0_i 0_j\rangle\langle 0_i 0_j| + |0_i 1_j\rangle\langle 0_i 1_j| + |1_i 0_j\rangle\langle 1_i 0_j| - |1_i 1_j\rangle\langle 1_i 1_j| \ . \tag{2.3.2}$$

Measurement A measurement M_i^α is a local projective measurement of qubit i along the axis α in the x-y-plane. That means that M projects the qubit i onto one of the states

$$|+_\alpha\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle + e^{i\alpha}|1\rangle\big) \qquad \text{or} \qquad |-_\alpha\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle - e^{i\alpha}|1\rangle\big) \ . \tag{2.3.3}$$

The probability of each outcome is determined as is usual in quantum mechanics. The probabilities of each outcome do not affect the computational power of the pattern. The outcome of a measurement is denoted as $s_i \in \mathbb{Z}_2$, where $s_i = 0/1$ represents the projection onto $|+/-_\alpha\rangle$, respectively. The measurement outcome is relevant for the further procedure along the pattern because measurement axes and corrections can depend on earlier outcomes. Since the outcomes transport classical information (either a 0 or a 1), they are called *signals*. A general dependent measurement is denoted as ${}^t[M_i^\alpha]^s$, where t and s are signals, i.e., outcomes of one or of many earlier measurements. The signals change the axis of measurement as

$${}^t[M_i^\alpha]^s = M_i^{(-1)^s\alpha + t\pi} \ , \tag{2.3.4}$$

that is, t rotates the axis of measurement by 180°, and s conjugates the projective states. When one of the signals is 0, we omit the subscript. Note that

$$[M_i^\alpha]^1 = M_i^{-\alpha} = X_i M_i^\alpha X_i \tag{2.3.5}$$

$$^1[M_i^\alpha] \ = M_i^{\alpha+\pi} = Z_i M_i^\alpha Z_i \ , \tag{2.3.6}$$

where X_i and Z_i are the Pauli matrices on qubit i. These identities are the reason why Danos *et al.* refer to s and t as X- and Z-actions. Following this nomenclature, we write the special cases of $\alpha = 0$ and $\alpha = \pi$ as $M_i^0 \equiv M_i^x$ and $M_i^\pi \equiv M_i^y$, respectively.

Corrections The last operations of the measurement calculus are local corrections C. They are one-qubit gates of the Pauli group, and their application will, in general, depend on signals. For the one-way quantum computing model proposed by Raussendorf and Briegel [7], it is enough to allow the corrections to be either X- or Z-gates. For example, we write

$$X_i^s = \begin{cases} |0\rangle\langle 1| + |1\rangle\langle 0| & (s = 1) \\ \mathbb{1} & (s = 0) \end{cases} \tag{2.3.7}$$

for an X-correction on qubit i if signal s is carrying a single bit of information.

2.3.3. Hadamard gate in the measurement calculus

As a first full example of a pattern, we consider the spaces $\mathcal{I} = \{1\}$, $\mathcal{O} = \{2\}$ and $C = \{1, 2\}$. The command sequence is

$$X_2^{s_1} M_1^x E_{12} N_2 \ , \tag{2.3.8}$$

and we will show that this pattern implements the Hadamard gate. We read this sequence from right to left and start with the preparation N_2 of qubit 2 in the $|+_2\rangle$ state. The first qubit is initialized to the input state $|\psi_1\rangle = \alpha|0_1\rangle + \beta|1_1\rangle$. The total initial state then reads

$$\begin{aligned} |\psi_1\rangle \otimes |+_2\rangle \ &= \big(\alpha|0_1\rangle + \beta|1_1\rangle\big) \otimes \big(|0_1\rangle + |1_1\rangle\big) \\ &= \alpha|00\rangle + \alpha|01\rangle + \beta|10\rangle + \beta|11\rangle \ , \end{aligned} \tag{2.3.9}$$

where we have omitted normalization and have abbreviated $|k_1\rangle \otimes |l_2\rangle \equiv |kl\rangle$. Entangling the qubits by E_{ij} is the next step in bringing the pattern into NEMC-form. Applying the CPHASE-gate to both pairs, $\{1, 2\}$ and $\{2, 3\}$, flips the phase of the basis state $|11\rangle$. The resulting entangled state reads

$$\alpha|00\rangle + \alpha|01\rangle + \beta|10\rangle - \beta|11\rangle = \alpha\big(|{+}{+}\rangle + |{-}{+}\rangle\big) + \beta\big(|{+}{-}\rangle - |{-}{-}\rangle\big) \ , \tag{2.3.10}$$

where we have transformed the state to the x-basis $(+/-)$ for easier handling in the measurement phase. Measuring qubit 1 along the x-axis will yield the outcome $s_1 \in \{0, 1\}$

and the resulting projected state

$$(1-s_1)\big(\alpha\,|{+}{+}\rangle + \beta\,|{+}{-}\rangle\big) + s_1\big(\alpha\,|{-}{+}\rangle - \beta\,|{-}{-}\rangle\big)\,. \tag{2.3.11}$$

This state is a product state, so we can investigate qubit 2 in isolation from now on. The correction $X_2^{s_1}$ is applied only if $s_1 = 1$, so the state of the output qubit is

$$\begin{aligned} &(1-s_1)\big(\alpha\,|{+}\rangle + \beta\,|{-}\rangle\big) + s_1 X_2\big(\alpha\,|{+}\rangle - \beta\,|{-}\rangle\big) \\ &= (1-s_1)\big(\alpha\,|{+}\rangle + \beta\,|{-}\rangle\big) + s_1\big(\alpha\,|{+}\rangle + \beta\,|{-}\rangle\big) \\ &= \alpha\,|{+}\rangle + \beta\,|{-}\rangle = H\,|\psi\rangle\,, \end{aligned} \tag{2.3.12}$$

meaning that we have actually applied the Hadamard-gate

$$H \equiv \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.3.13}$$

to the state $|\psi\rangle$ while transporting it from qubit 1 to qubit 2.

2.3.4. Combinations of patterns

Patterns can also be combined. Recall that $\mathcal{I}$ is the input space, $\mathcal{C}$ is the computation space, and $\mathcal{O}$ is the output space. If, for two patterns $\mathcal{C}_1 \cap \mathcal{C}_2 = \mathcal{O}_1 = \mathcal{I}_2$, we can compose them by concatenating the commands and

$$\mathcal{C} \equiv \mathcal{C}_1 \cup \mathcal{C}_2\,, \quad \mathcal{I} = \mathcal{I}_1\,, \quad \mathcal{O} = \mathcal{O}_2\,, \tag{2.3.14}$$

where the union is clearly not disjoint because the output space of the first pattern coincides with the input space of the second pattern. Parallel computation is possible when $\mathcal{C}_1 \cap \mathcal{C}_2 = \emptyset$. We then also concatenate commands and call the resulting pattern the tensor product of the two patterns

$$\mathcal{C} \equiv \mathcal{C}_1 \cup \mathcal{C}_2\,, \quad \mathcal{I} = \mathcal{I}_1 \cup \mathcal{I}_2\,, \quad \mathcal{O} = \mathcal{O}_1 \cup \mathcal{O}_2\,. \tag{2.3.15}$$

Here the unions are disjoint. We obtain two patterns running in parallel, as the information flow of the two patterns does not intersect in the tensor product. Concatenating commands can be done chronologically or logically, meaning that we either order commands with respect to their order in the original pattern or group them together pattern-wise. In the tensor product, the commands of the two initial patterns commute and can thus be ordered in any way. On the other hand, we need to be careful when reordering commands after concatenating patterns because the commands do not commute in general. Details are given in the next subsection.

2.3.5. Standardization

In the previous subsections, we have learned that it is advantageous to sort the commands N, E, M, and C into NEMC-order. That is, we prepare (N) the qubits first, then entangle (E) pairs, measure (M), and finally correct (C) qubit information. When this ordering is used, the power of MBQC is most comprehensible. Single-qubit preparation and two-qubit entangling are the first two stages of the command sequence. As these operations are part of the Clifford group, they can be efficiently simulated on a classical computer [53] (Gottesman-Knill-theorem). Since the corrections at the end of the command sequence are also in the Clifford group, the relevant quantum computing (beyond the classical scope) must indeed happen during measurements.

A crucial part is still missing. What if we have a pattern that is not in NEMC-form? Such a pattern could, for example, be the result of a concatenation of two patterns (see previous subsection). Can we transform it so that we end up with a new pattern that is in the desired form? An algorithm for such a transformation will be presented in the following.

Rewriting a pattern in NEMC-form will require a set of commutation relations as the commands do not commute in general. First, to simplify the formalism, we assume that all qubits except for the input space are always initialized to be in the $|+\rangle$-state. The input space is always set to be the quantum state that is fed into the algorithm that the pattern represents. Thus, taking the preparation N as given, we are left with the task of transforming a pattern to an EMC-form. The following rewrite rules allow us to bring any sequence of commands into EMC-form [9]:

$$E_{ij}X_i^s = X_i^s Z_j^s E_{ij} \,, \tag{2.3.16}$$

$$E_{ij}X_j^s = X_j^s Z_i^s E_{ij} \,, \tag{2.3.17}$$

$$E_{ij}Z_{i/j}^s = Z_{i/j}^s E_{ij} \,, \tag{2.3.18}$$

$${}^t[M_i^\alpha]^s X_i^r = {}^t[M_i^\alpha]^{s+r} \,, \tag{2.3.19}$$

$${}^t[M_i^\alpha]^s Z_i^r = {}^{r+t}[M_i^\alpha]^s \,. \tag{2.3.20}$$

The rewrite rules can easily be proven by writing them out as matrices. Commands with all other combinations of site indices as well as with M and with E commute. Given a command sequence, we can now bring it into EMC-form. For example, we can first commute all entangling operations (E) to the right (beginning) and then commute all corrections (C) to the left end to end up in the order EMC.

As an example, we concatenate two Hadamard patterns, Eq. (2.3.8), as in Eq. (2.3.14). The combined pattern has input $\{1\}$, output $\{3\}$, and computation space $\{1, 2, 3\}$. The

combined command sequence (omitting the N's) reads

$$\begin{aligned} & X_3^{s_2} M_2^x E_{23} X_2^{s_1} M_1^x E_{12} \\ =& X_3^{s_2} M_2^x X_2^{s_1} Z_3^{s_1} M_1^x E_{23} E_{12} \\ =& X_3^{s_2} [M_2^x]^{s_1} Z_3^{s_1} M_1^x E_{23} E_{12} \\ =& X_3^{s_2} Z_3^{s_1} M_2^x M_1^x E_{23} E_{12} \,, \end{aligned} \tag{2.3.21}$$

where we have used Eq. (2.3.16) in the first step, Eq. (2.3.19) in the second step, and the fact that $[M_i^x]^y = M_i^x$ in the third step. Since we already know that we have multiplied two Hadamard gates H and that $H^2 = \mathbb{1}$, we conclude that we have implemented the identity acting over three physical qubits, teleporting the initial single-qubit state from qubit 1 to qubit 3. We are already familiar with teleportation from the circuit model depicted in Fig. 2.3. If we concatenate two instances of the circuit shown there, the result is a circuit that is equivalent to Eq. (2.3.21).

It is important to note that the standardization procedure described in the last paragraphs is deterministic only if we allow for some flexibility in the result. The single qubit corrections (C) commute if they act on different qubits. The same is true for E and M. Thus, we terminate the standardisation algorithm if it is in EMC-form, leaving the freedom of commutation of the commands.

2.3.6. Decoupling

Although it is not necessary from a purely theoretical point of view, we will introduce another rewrite rule additional to the ones of Danos *et al.* that allows us to decouple a pattern. This operation is, in some sense, the inverse of the entangling operation (E_{ij}). It will be important in experimental setups in which we are able to prepare a larger resource lattice but in which we only need a particular subspace for calculation. So far, we have only covered measurements in the x-y-plane. A z-measurement has the aforementioned decoupling action. This is why it is not used in the standard measurement calculus.

Consider a pattern of three qubits along a line. The middle qubit 2 is prepared using the standard operation N_2 and is thus in the $|+\rangle_2$ state. The states of the outer two qubits are irrelevant for the demonstration of decoupling. We set them to be unknown states $|\psi\rangle_1$ and $|\phi\rangle_3$. The line of qubits will then be entangled by $E_{12}E_{23}$. A measurement M_2^z of the center qubit 2 in the z-basis with outcome s_2 results in the unentangled three-qubit product state

$$Z_1^{s_2} |\psi\rangle_1 |s_2\rangle_2 Z_3^{s_2} |\phi\rangle_3 \,, \tag{2.3.22}$$

which, apart from local corrections $Z_i^{s_2}$, is the initial state of qubits 1 and 3 before entangling. Qubit 2, naturally, is in the projected state after measurement. This example is portrayed in Fig. 2.5.

Analogously to the canonical rewrite rules of measurement calculus, we want to formulate an additional rewrite rule for decoupling an entangled resource state. We saw

Figure 2.5.: Decoupling of an entangled resource state via a z-measurement. The dotted bonds on the right side visualize that the entanglement between the qubits has been annihilated.

in the last paragraph that a z-measurement annihilates the entanglement. To be precise, all entanglement bonds to neighboring qubits inside the graph are annihilated, even in two or more dimensions. Hence, we introduce the decoupling rewrite rule

$$M_i^z \prod_{j \in N(i)} E_{ij} \qquad = \qquad \prod_{j \in N(i)} Z_j^{s_i} \,, \tag{2.3.23}$$

where the neighborhood of qubit i inside the graph is denoted by $N(i)$. The latter incorporates all qubits that are connected to i via one edge. We have thus erased the entangling operations E_{ij} and are left with local corrections Z_j on the neighboring qubits. In othe words, the graph edges around qubit i have been cut.

2.3.7. Bridging nodes

After having shown in the previous subsections that the effects of x- and z-measurements are to push information through a graph with the Hadamard matrix and decoupling, respectively, we will now discuss the last remaining orthogonal basis direction of measurement. A y-measurement leads to a *bridging* of the qubit. That is, the neighbors of said qubit will gain interconnection, and the qubit will be removed from the graph. Analogously to the last subsection, we can introduce the general bridging rewrite rule

$$M_i^y \prod_{j \in N(i)} E_{ij} \qquad = \qquad \prod_{j,k \in N(i)} P_j(s_i) E_{jk} \,, \tag{2.3.24}$$

where the single-qubit phase gates on the neighbours are defined as

$$P(s_k) = |0\rangle\langle 0| + (-1)^{s_k}\,\mathrm{i}\,|1\rangle\langle 1| = \begin{pmatrix} 1 & 0 \\ 0 & (-1)^{s_k}\,\mathrm{i} \end{pmatrix} . \tag{2.3.25}$$

Coming back to the previous simple example, which has three qubits entangled by $E_{12}E_{23}$, a y-measurement will result in a state in which qubits 1 and 3 are entangled by the E_{13} action plus some local corrections. This example is illustrated in Fig. 2.6. The middle qubit is in the projected state after measurement, but it is disentangled from the rest of the network. Therefore, we call it “bridged”.

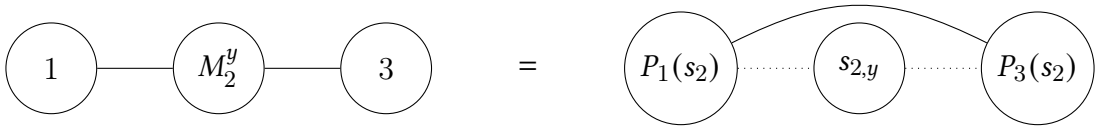


Figure 2.6.: Bridging of one qubit in an entangled resource state via a y-measurement. The dotted bonds on the right side visualize that the entanglement between the qubits has been destroyed. The state of the center qubit after measurement is the s_2-eigenstate of the Pauli-y-matrix, where $|0_{2,y}\rangle = |0\rangle + \mathrm{i}\,|1\rangle$ and $|1_{2,y}\rangle = \mathrm{i}\,|0\rangle + |1\rangle$ (normalization omitted).

2.3.8. Examples

Building on the rewrite rules of the last subsections, we will now introduce a number of patterns for specific tasks in quantum computing. Previously, we have already covered the examples for teleportation Eq. (2.3.21) and the Hadamard gate, Eq. (2.3.8).

x-Rotation We generalize the command sequence for the Hadamard gate to

$$\mathcal{J}(\alpha) \equiv X_2^{s_1} M_1^{-\alpha} E_{12} \,. \tag{2.3.26}$$

It is clear that $\mathcal{J}(0)$ is the Hadamard sequence and thus embodies the quantum gate with the same name. We can use Eq. (2.3.26) to implement a rotation around the x-axis by an angle α. In terms of matrices, we can write an x-rotation as

$$R_x(\alpha) \equiv \mathrm{e}^{\mathrm{i}\,\alpha X} = \mathrm{e}^{\mathrm{i}\,\alpha HZH} = H\,\mathrm{e}^{\mathrm{i}\,\alpha Z}\,H = \begin{pmatrix} \cos\alpha & \mathrm{i}\sin\alpha \\ \mathrm{i}\sin\alpha & \cos\alpha \end{pmatrix} = \cos\alpha\,\mathbb{1} + \mathrm{i}\sin\alpha\,X \,, \tag{2.3.27}$$

where X, Z and H are the Pauli-X, Pauli-Z and Hadamard matrices, respectively. We let Eq. (2.3.26) be the command sequence of a pattern on a two-qubit cluster state in which qubit 1 is in an initial state $|\psi\rangle = a\,|0\rangle + b\,|1\rangle$ and qubit 2 is in the standard initial state $|+\rangle = |0\rangle + |1\rangle$. This gives

$$\begin{aligned}
\mathcal{J}(\alpha)\,|\psi\rangle\,|+\rangle &= X_2^{s_1} M_1^{-\alpha} E_{12} \Big(a\big(\,|00\rangle + |01\rangle\,\big) + b\big(\,|10\rangle + |11\rangle\,\big) \Big) \\
&= X_2^{s_1} M_1^{-\alpha} \Big(a\big(\,|00\rangle + |01\rangle\,\big) + b\big(\,|10\rangle - |11\rangle\,\big) \Big) \\
&= X_2^{s_1} \Big(|0_1\rangle + (-1)^{s_1}\,\mathrm{e}^{-\mathrm{i}\,\alpha}\,|1_1\rangle \Big) \Big(\langle 0_1| + (-1)^{s_1}\,\mathrm{e}^{\mathrm{i}\,\alpha}\,\langle 1_1| \Big) \\
&\qquad \cdot \Big(a\big(\,|00\rangle + |01\rangle\,\big) + b\big(\,|10\rangle - |11\rangle\,\big) \Big) \\
&= X_2^{s_1} 2^{-3/2} \Big(\big(a + (-1)^{s_1}\,\mathrm{e}^{\mathrm{i}\,\alpha}\,b\big)\,|00\rangle + \big(a - (-1)^{s_1}\,\mathrm{e}^{\mathrm{i}\,\alpha}\,b\big)\,|01\rangle \\
&\qquad + \big((-1)^{s_1}\,\mathrm{e}^{-\mathrm{i}\,\alpha}\,a + b\big)\,|10\rangle + \big((-1)^{s_1}\,\mathrm{e}^{-\mathrm{i}\,\alpha}\,a - b\big)\,|11\rangle \Big) \\
&= \Big(|0\rangle + (-1)^{s_1}\,\mathrm{e}^{-\mathrm{i}\,\alpha}\,|1\rangle \Big) \Big(\big(a + \mathrm{e}^{\mathrm{i}\,\alpha}\,b\big)\,|0\rangle + \big(a - \mathrm{e}^{\mathrm{i}\,\alpha}\,b\big)\,|1\rangle \Big) \,.
\end{aligned} \tag{2.3.28}$$

In the last line, we observe that qubit 2 is now in the state $J(\alpha)\,|\psi\rangle$, where

$$J(\alpha) = \begin{pmatrix} 1 & \mathrm{e}^{\mathrm{i}\,\alpha} \\ 1 & -\mathrm{e}^{\mathrm{i}\,\alpha} \end{pmatrix} . \tag{2.3.29}$$

The command sequence (2.3.26) thus realizes the unitary matrix (2.3.29) up to normalization while transporting the state from one qubit to another. It is easy to check that $J(\alpha)H = R_x(\alpha)$ up to an irrelevant overall phase. We already know that $\mathcal{J}(0)$ realizes the Hadamard gate and conclude that the pattern $\mathcal{R}(\alpha) \equiv \mathcal{J}(\alpha)\mathcal{J}(0)$ executes an x-rotation by the angle α. The corresponding combined command sequence is

$$\begin{aligned} & X_3^{s_2} M_2^{-\alpha} E_{23} X_2^{s_1} M_1^x E_{12} \\ = & X_3^{1+s_1+s_2} Z_3^{s_1} M_2^{\alpha} M_1^x E_{23} E_{12} \,, \end{aligned} \tag{2.3.30}$$

where we have used the standardization scheme and the fact that $X_3^{s_2}[M_2^{-\alpha}]^{s_1} = X_3^{1+s_1+s_2} M_2^{\alpha}$ because we flip the measurement outcomes by the superscript s_1.

z-Rotation For a rotation around the z-axis, we calculate that $R_z(\alpha) = HJ(\alpha)$, just as in the last example [9]. If we follow the same steps as above, we obtain the command sequence

$$X_3^{s_2} Z_3^{1+s_1} M_2^x M_1^{\alpha} E_{23} E_{12} \tag{2.3.31}$$

on the same spaces, $\mathcal{I} = \{1\}, \mathcal{O} = \{3\}, C = \{1,2,3\}$, as for the x-rotation.

General Rotation A general one-qubit rotation can be described in terms of *Euler angles* as $R_x(\alpha)R_z(\beta)R_x(\gamma)$ [54]. Thus, we can concatenate the x- and z-rotations from the previous paragraphs to obtain a seven-qubit pattern. However, Danos *et al.* give a sequence that implements an arbitrary rotation on just five qubits in terms of the $\mathcal{J}(\alpha)$ patterns [55]:

$$\mathcal{R}(\alpha,\beta,\gamma) = \mathcal{J}(0)\mathcal{J}(-\alpha)\mathcal{J}(-\beta)\mathcal{J}(-\gamma) \,. \tag{2.3.32}$$

After some calculation, one obtains the standardized form of this command sequence,

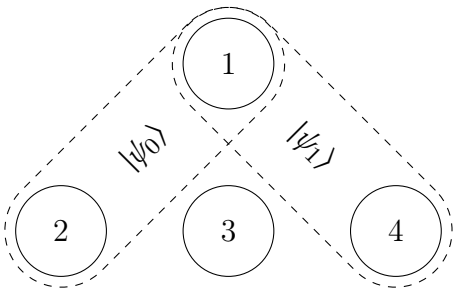
$$X_5^{s_2+s_4} Z_5^{s_1+s_3} M_4^x [M_3^{\alpha}]^{s_2} [M_2^{\beta}]^{s_1} M_1^{\gamma} E_{12345} \,, \tag{2.3.33}$$

where $E_{12345} \equiv E_{12}E_{23}\ldots E_{45}$.

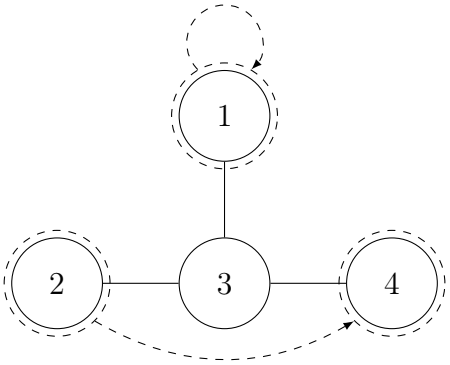
CNOT For the first two-qubit gate, we recall that $CNOT = (\mathbb{1} \otimes H)CZ(\mathbb{1} \otimes H)$. We have already derived all elements of this sequence, so we find a pattern with $\mathcal{I} = \{1,2\}$, $C = \{1,2,3,4\}$, $\mathcal{O} = \{1,4\}$ that realizes the CNOT gate. The command sequence reads

$$H_{3\to 4} E_{13} H_{2\to 3} = X_4^{s_3} M_3^x E_{34} E_{13} X_3^{s_2} M_2^x E_{23} = X_4^{s_3} Z_4^{s_2} Z_1^{s_2} M_3^x M_2^x E_{13} E_{23} E_{34} \,. \tag{2.3.34}$$

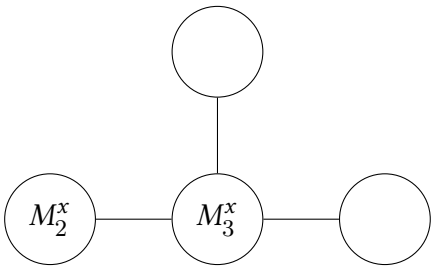
The pattern for the CNOT gate is illustrated in more detail in Fig. 2.7.



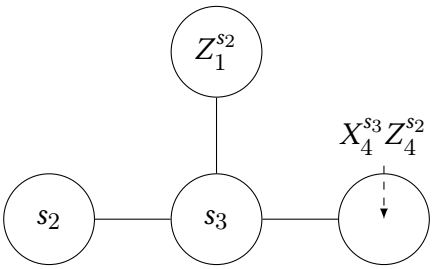
(a) Physical qubits 1 and 2 are prepared in the initial state $|\psi_0\rangle$, 3 and 4 in the $|+\rangle$ state. At the end of the algorithm, the two-qubit state $|\psi_1\rangle$ = CNOT $|\psi_0\rangle$ can be found on physical qubits 1 and 4.



(b) Resource configuration and information flow. Straight lines denote entangling with the CPHASE gate. The logical qubit 1 stays on physical qubit 1, and logical qubit 2 is transported from physical qubit 2 to physical qubit 4 during the algorithm.



(c) Measurements in chronological order : M_2^x, M_3^x.



(d) Outcomes s_2 and s_3 of the measurements and respective corrections. Corrections on qubit 4 are read from right to left.

Figure 2.7.: Visualization of the CNOT gate in the framework of measurement calculus. The input is defined by qubits 1 and 2 while the output is written onto qubits 1 and 4.

Controlled-U The last example that we will cover here is the application of a controlled-U gate. The latter is a general unitary gate U that is applied to the target qubit if and only if the control qubit is in the 1 state. It was shown in Refs. [55, 56] that a unitary operation can be written in terms of general rotations as

$$U' = \mathrm{e}^{\mathrm{i}\,\alpha} R_z(\beta) R_x(\gamma) R_z(\delta) = \mathrm{e}^{\mathrm{i}\,\alpha}\, \mathrm{e}^{-\mathrm{i}(\beta+\gamma+\delta)/2}\, J(0) J(\beta) J(\gamma) J(\delta)\,, \tag{2.3.35}$$

where we have used the J-rotations from Eq. (2.3.29) on the right side of the equation. If we assume that the unitary has a decomposition $U = \mathrm{e}^{\mathrm{i}\,\alpha} J(0)J(\beta)J(\gamma)J(\delta)$, the command sequence of the controlled operation can be written in terms of J-operations as

$$CU_{12} \;= J_1^0 J_1^\alpha J_2^2 J_2^{\beta+\pi} J_2^{-\gamma/2} J_2^{-\pi/2} J_2^0 CZ_{12} J_2^{\pi/2} J_2^{\gamma/2} J_2^{-(\pi+\beta+\delta)/2} J_2^0 CZ_{12} J_2^{-(\pi+\beta-\delta)/2}\,, \tag{2.3.36}$$

where $J_i^\tau \equiv J_i(\tau)$, and the subscripts describe logical qubits. We translate this command sequence to a command sequence acting on physical qubits of a cluster state using the identities of the measurement calculus that are explained in the previous subsections. We derive the sequence

$$\begin{aligned} Z_k^{s_i+s_g+s_e+s_c+s_a} X_k^{s_j+s_h+s_f+s_d+s_b} X_C^{s_B} Z_C^{s_A+s_e+s_c} \\ \cdot M_B^0 M_A^{-\alpha} M_j^0 [M_i^{-\beta-\pi}]^{s_h+s_f+s_d+s_b} [M_h^{\gamma/2}]^{s_g+s_e+s_c+s_a} [M_g^{\pi/2}]^{s_f+s_d+s_b} \\ \cdot M_f^0 [M_e^{-\pi/2}]^{s_d+s_b} [M_d^{-\gamma/2}]^{s_c+s_a} [M_c^{(\pi+\beta+\delta)/2}]^{s_b} M_b^0 M_a^{(\pi+\beta-\delta)} \\ \cdot E_{BC} E_{AB} E_{jk} E_{ij} E_{hi} E_{gh} E_{fg} E_{Af} E_{ef} E_{de} E_{cd} E_{bc} E_{ab} E_{Ab}\,. \end{aligned} \tag{2.3.37}$$

Here $\mathcal{I} = \mathcal{O} = \{C, k\}$ denote the control and target qubits, respectively, which are identical for input and output. A graph of the pattern is shown in Fig. 2.8. The labels of the nodes in the figure are coherent with Eq. (2.3.37). It is evident from the graph that the algorithmic complexity of the pattern is 7, i.e., the pattern can be executed in just seven consecutive rounds of measurement. For example, the measurements on the qubits j, A, B, a, b, f can be executed in parallel because they do not depend on each other. In contrast, b, c, d, e, h, i, k have to be measured one at a time because they all depend on the respective outcomes measured beforehand.

2.3.9. Dependency theorems

In the last paragraph, we have learned about dependency in MBQC patterns. As a matter of fact, we can make more sophisticated statements about the power of measurement patterns when we analyse the type of measurements and the dependencies.

In most of the examples above, we have only used x- and y-measurements. However, it is necessary to have access to more measurement axes than x and y to implement universal quantum computing. If we only have x- and y-measurements, then the combined actions of all patterns form a representation of the *Clifford group* [9]. The latter is the stabilizer of the *Pauli group*, i.e., the Pauli matrices. If all gates of a quantum computation

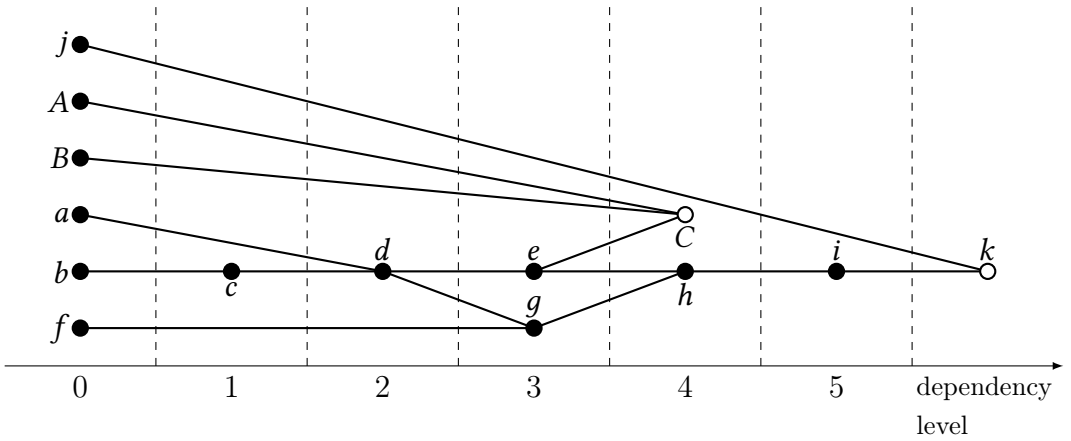


Figure 2.8.: The dependency graph for the controlled-U pattern in Eq. (2.3.37), redrawn from [9]. Output qubits are denoted by unfilled nodes. The target qubit k has the maximal dependency length as it is dependent on a chain of six other qubits. Thus, the controlled-U realisation has a computational depth of 7.

belong to the Clifford group, then, according to the *Gottesman-Knill theorem*, it can be simulated efficiently on a classical computer [53]. In consequence, we need to allow for more than just x- and y-measurements in order to achieve a quantum advantage over classical information theory.

In the last subsection, we argued that patterns can have a varying amount of inter-dependency. If there is no dependency in a pattern, i.e., all measurements could, in principle, be executed in parallel, then it also belongs to the Clifford group and can thus be simulated efficiently on a classical computer [9]. Therefore, dependent measurements and, potentially, corrections, i.e., signal forwarding, are necessary elements of a pattern for universal, non-classical, quantum computing.

3. ZX-calculus

In Chap. 2 we introduced a calculus that formalizes measurement-based quantum computing (MBQC). Although it is useful for proving the universality of quantum computation on cluster states, there is a newer, graphical, calculus that makes an even more intuitive treatment of MBQC possible. In the following, we will introduce the *ZX-calculus*, which can elucidate much more of the inner workings of MBQC. We will also rediscover a number of rules from the measurement calculus from another perspective. Nevertheless, our introduction to the ZX-calculus will be self-contained; knowledge from the last chapter, while recommended, is not necessarily required.

We will limit the introduction of the ZX-calculus to aspects that are relevant for the upcoming description of algorithms within the framework of MBQC. General introductions to the ZX-calculus can be found in Refs. [11, 12]. A more thorough and illustrative, pedagogical book on diagrammatic quantum languages and information processing is Ref. [57].

3.1. Rules of ZX-calculus

The ZX-calculus describes both quantum states and quantum operations as graphs. The nodes of the graph represent physical qubits. The edges represent connections between qubits, i.e., *entanglement*. If edges protrude from a graph, that is, if they do not connect two qubits, then they represent degrees of freedom. Such a degree of freedom could, for example, be the spin of an electron. The degrees of freedom of the graph (protruding edges) are the only possible way to gain access to the information stored in the graph state. This access will be realized by (projective) measurements.

3.1.1. Nodes and edges

The graph-based language of the ZX-calculus contains two types of nodes: Z-nodes, colored green here (or lightly shaded for the visually impaired reader) and X-nodes, colored red (darkly shaded) here. They are, on a tensor level, defined as

$$= |0\ldots0\rangle\langle 0\ldots0| + \mathrm{e}^{\mathrm{i}\alpha}\,|1\ldots1\rangle\langle 1\ldots1| \tag{3.1.1}$$

and

$$= |+\cdots+\rangle\langle +\cdots+| + \mathrm{e}^{\mathrm{i}\alpha}\,|-\cdots-\rangle\langle -\cdots-|\ , \tag{3.1.2}$$

where $|\pm\rangle = |0\rangle \pm |1\rangle$. As mentioned above, we neglect normalization factors. The number inside the node represents the phase of the qubit. For Z-nodes, the phase determines the angle of the axis of quantization in the x-y-plane, measured relative to the the x-axis. For X-nodes, the angle determines the angle in the z-y-plane, relative to the z-axis. If the phase of a node is zero, we omit the phase angle in the node. Incoming and outgoing legs represent degrees of freedom, i.e., tensor indices. We note that all ZX-diagrams were typeset with the *zx-calculus* package [58].

Examples of nodes include single, isolated qubits in the basis-states $|0\rangle$ for the computational z-basis or the basis state $|+\rangle$ from the x-basis. They are represented by the ZX-diagrams

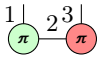
 $= |+\rangle + |-\rangle = |0\rangle$, (3.1.3)

 $= |0\rangle + |1\rangle = |+\rangle$, (3.1.4)

respectively.

Connecting nodes via edges creates graphs that represent either quantum states or quantum operators. The graph language does not distinguish between the two. It either should be clear from context or it must be defined weather a degree of freedom is a ket or a bra. Most times, it is simply not necessary to differentiate if a degree of freedom corresponds to a ket or a bra because the two are dual to one another anyway. If degrees of freedom are contracted, the notation defines that one is a ket and the other is a bra.

We have already given examples of single-qubit states in Eq. (3.1.3) and Eq. (3.1.4). A more complex example, which requires more than one node, is the completely anti-symmetric Bell state $|01\rangle - |10\rangle$. In the framework of the ZX-calculus, it can be written as

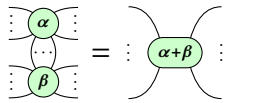
 $= \big(|0\rangle_1 \langle 0|_2 - |1\rangle_1 \langle 1|_2\big)\big(|+\rangle_2 |+\rangle_3 - |-\rangle_2 |-\rangle_3\big)$
$= |0\rangle_1 |+\rangle_3 - |0\rangle_1 |-\rangle_3 - |1\rangle_1 |+\rangle_3 - |1\rangle_1 |-\rangle_3 = |0\rangle_1 |1\rangle_3 - |1\rangle_1 |0\rangle_3$, (3.1.5)

as can be checked by hand. Here we have omitted overall prefactors, including the normalization, for brevity and have added labels to the degrees of freedom in order to assign them to be bras and kets. This assignment is somewhat arbitrary, e.g., our identification of protruding edges 1 and 3 as kets could well have been changed to bras or a mixture of both; this degree of flexibility is inherent in the graph description of the ZX-calculus, as mentioned above.

3.1.2. Rewriting rules

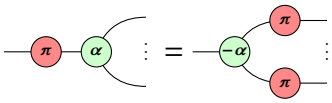
Calculations are carried out in the ZX-calculus by applying rules to transform diagrams. These rules are the basic operations of the ZX-calculus, just like the rewriting rules of the measurement calculus in Eqs. (2.3.16)-(2.3.20)—or arithmetic operations such as

addition or multiplication for classical calculus. Rules can, for example, be derived by expressing a diagram in tensor notation and transforming, typically simplifying, the tensor contractions. The comprehensive list of basic rules, redrawn from Ref. [59], is as follows:

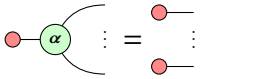


(3.1.6)

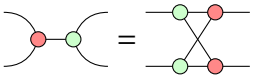
(3.1.7)



(3.1.8)



(3.1.9)



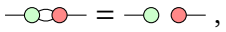
(3.1.10)

Note that rules (3.1.6)-(3.1.10) are also valid when Z-nodes and X-nodes are interchanged in the diagrams. We also note that the equations generally only hold up to an overall complex factor. As it is impossible to access the overall phase of a quantum state, we neglect this detail and simply speak of equality. A framework for a variant of the ZX-calculus that covers the phase exactly can be found in Ref. [60].

Rule (3.1.6) can easily be proven using that $\langle\alpha|\beta\rangle = \delta_{\alpha,\beta}$ for any orthonormal basis $\{|\alpha\rangle, |\beta\rangle\}$. It states that neighboring tensors within the same basis (here the Z-basis) can be merged because they have the same entangling properties (3.1.1) as a single tensor. In other words, any neighboring tensors of the same basis type do not affect the entanglement of the neighborhood with its surroundings.
Rule (3.1.7) directly follows from $|0\rangle\langle 0| + |1\rangle\langle 1| = \mathbb{1}$.
Rule (3.1.9) can also easily be proven within the bra-ket representation. It states that a terminal node in the opposing basis will decouple a graph at the connected node. In other words, opposing bases will reduce the entanglement inside the graph. We will see in the next paragraph that this can also be interpreted as a measurement in the opposing basis, which destroys entanglement in a tensor state.

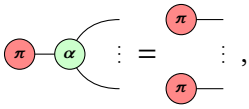
We can also make the following statement about the entanglement generated when creating a ZX-diagram. Connecting nodes of a particular kind (Z- or X-nodes) to nodes of the same kind will add no extra entanglement to the rest of the graph state, as is encoded by rule (3.1.6). On the other hand, adding X- nodes adjacently to Z-nodes and vice versa will, in general, change the amount of entanglement in the graph state because their connecting degrees of freedom are chosen in different bases.

Two additional rules, the *antipode* rule,



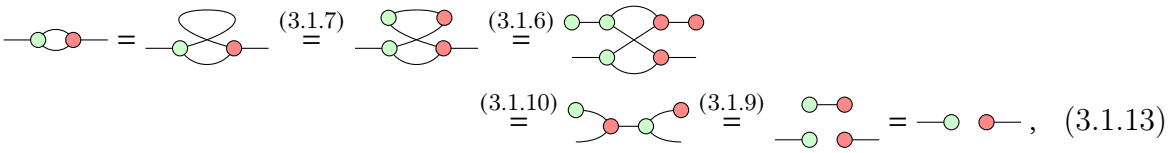
(3.1.11)

and the π-*decoupling* rule,



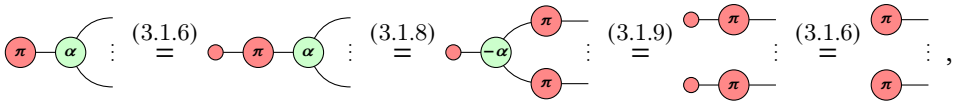
(3.1.12)

can be proven directly from the basic rules [12]. For the antipode rule (3.1.11), we calculate



(3.1.13)

where we have indicated the rule utilized for each step. In the last step, we have omitted the overall factor, which just represents normalization. For the π-decoupling rule (3.1.12), we calculate



(3.1.14)

where we have again indicated the rules used.

3.2. Translation of quantum circuits

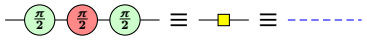
The ZX-calculus is universal for quantum operations. That is, any linear map can be represented by a ZX-diagram. The domain and the image of the map are both represented by protruding legs or, in other words, degrees of freedom of the composite tensor.

Since the ZX-calculus is universal for quantum operations, there has to be a surjective map from quantum circuits to ZX-diagrams. An easy way to show the universality of ZX-diagrams is to translate a universal set of quantum gates into ZX-diagrams. The set $\{\text{CNOT}, H, Z_\alpha\}$, consisting of the controlled NOT gate, the Hadamard gate and a general z-rotation, is a universal set for quantum computing. The CNOT-gate can be translated into a ZX-diagram using

$$
\begin{aligned}
&= \Big(|0\rangle_2 \langle 0|_1 \langle 0|_* + |1\rangle_2 \langle 1|_1 \langle 1|_* \Big) \Big(|+\rangle_4 \langle +|_3 |+\rangle_* + |-\rangle_4 \langle -|_3 |-\rangle_* \Big) \\
&= |0_2{+}_4\rangle \langle 0_1{+}_3| + |0_2{-}_4\rangle \langle 0_1{-}_3| + |1_2{+}_4\rangle \langle 1_1{+}_3| - |1_2{-}_4\rangle \langle 1_1{-}_3| \\
&= |00\rangle \langle 00| + |01\rangle \langle 01| + |10\rangle \langle 11| + |11\rangle \langle 10| \\
&= \text{CNOT} \, .
\end{aligned}
\tag{3.2.1}
$$

We have indicated the degrees of freedom (legs) of each tensor in the diagram and have ported the labels to the bra-ket notation for readability. In the last step, we have dropped the indices because it is clear from the context that indices 1 and 3 correspond to the

input and indices 2 and 4 to the output of the map if we read the diagram from left to right. The two single-qubit gates of the universal set can be translated as



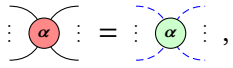
$$= \big(|0\rangle\langle 0| + \mathrm{i}\,|1\rangle\langle 1|\big)\big(|+\rangle\langle +| + \mathrm{i}\,|-\rangle\langle -|\big)\big(|0\rangle\langle 0| + \mathrm{i}\,|1\rangle\langle 1|\big) \tag{3.2.2}$$

$$= |0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1| = H$$

$$= |0\rangle\langle 0| + \mathrm{e}^{\mathrm{i}\,\alpha}\,|1\rangle\langle 1| = Z_\alpha\,. \tag{3.2.3}$$

In Eq. (3.2.2), we have indicated that the diagram for the Hadamard gate is often abbreviated as a single yellow square node when the action of the Hadamard gate is to be highlighted. In other cases, when many connections with Hadamard gates are present in a diagram, it is simply drawn as a blue and dashed edge. The latter will also mostly be the case when we represent MBQC with ZX-diagrams.

The Hadamard gate, taken as a basis transformation from the computational z-basis to the eigenbasis of the X-Pauli-matrix, gives rise to another rule,



(3.2.4)

which states that the Hadamard gate, applied to all legs of a node, swaps the color of the node, i.e., swaps X- and Z-nodes. The identity (3.2.4) can be proven in the bra-ket notation, since

$$H^{\otimes n}\,|0\ldots0\rangle\langle 0\ldots0|\,H^{\otimes n} + \mathrm{e}^{\mathrm{i}\alpha}H^{\otimes n}\,|1\ldots1\rangle\langle 1\ldots1|\,H^{\otimes n}$$

$$= |+\cdots+\rangle\langle +\cdots +| + \mathrm{e}^{\mathrm{i}\alpha}\,|-\cdots-\rangle\langle -\cdots -|\,, \tag{3.2.5}$$

where we have implicitly used that $H^\dagger = H$. Similarly to rules (3.1.6)-(3.1.10), rule (3.2.4) is also valid when colors are swapped, since $HH = H^2 = \mathbb{1}$.

Hadamard edges (blue and dashed) between nodes of the same kind will also generate more entanglement than bare lines. As can be seen in Eq. (3.2.2), a Hadamard edge consists of a combination of X- and Z- nodes and will thus generate entanglement for either kind of neighboring nodes of the of same color. On the other hand, adding Hadamard edges between nodes of the same color will not add entanglement because the resulting graph will not have additional edges between red and green nodes (after applying rule (3.1.6)). These properties can also be derived canonically within the Hilbert-space model by recognizing that a Hadamard gate is a basis transformation from the Z- to the X-basis and vice versa.

3.3. ZX-calculus and MBQC

In Chap. 2, MBQC and the Measurement Calculus, a framework that allows for a systematic treatment of patterns for MBQC. We will show in this section that the

ZX-calculus provides an equally powerful, if not more comprehensible, description of MBQC. Measurements can be identified as single nodes in a ZX-diagram. A cluster state as resource for MBQC consist of only Z-nodes and Hadamard edges in the ZX-calculus. With those prerequisites, it is possible to create an easy-to-read, visual protocol for measurement-based calculations.

3.3.1. Measurements and cluster state as ZX-diagrams

A projective quantum measurement is an orthogonal projection of a quantum state to a sub-Hilbert-space. The probability of a measurement outcome—which determines the projection space—is given by the amplitude squared that corresponds to said sub-Hilbert-space. In this work, we will neglect that a measurement can, in general, yield multiple possible outcomes. Instead, we will assume that we will always measure the first outcome in the single-qubit measurements that we will perform in MBQC. In the next subsection, we will elucidate why this presupposition does not destroy the universality of MBQC.

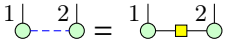
Measurements can be represented as single nodes in the ZX-calculus. Note that a projection or a collapse of a state can be represented by a single bra in the bra-ket notation. A collapse of a qubit state into the sub-space spanned by $|\alpha\rangle$ is denoted as the projector $|\alpha\rangle\langle\alpha|$, applied to the state. If we omit the information about the sub-space after measurement, we can also represent the collapse by the bra $\langle\alpha|$. In the ZX-calculus, single-qubit bras (as well as kets) are simply denoted by one node with one leg. A projection to the computational basis space belonging to outcome 0, for example, would be represented by

(3.3.1)

A measurement in the xy-plane with angle α would be denoted by

(3.3.2)

In MBQC, the measurements are executed on a multi-qubit state with special entanglement properties. Typically, this is a cluster state as introduced in Chap. 2. A cluster state can be represented in the ZX-calculus as a diagram that consists of only three elements: Z-nodes without phase, Hadamard-inner-edges, and protruding regular legs [59]. As an example, we investigate the most simple non-trivial diagram, consisting of two connected Z-nodes with one degree of freedom each. We obtain



$$
\begin{aligned}
&= \big(|0\rangle_1\langle 0|_* + |1\rangle_1\langle 1|_*\big) H_* \big(|0\rangle_* |0\rangle_2 + |1\rangle_* |1\rangle_2\big) \\
&= \big(|0\rangle_1\langle 0|_* + |1\rangle_1\langle 1|_*\big)\big(|+\rangle_* |0\rangle_2 + |-\rangle_* |1\rangle_2\big) \\
&= |0_1 0_2\rangle + |0_1 1_2\rangle + |1_1 0_2\rangle - |1_1 1_2\rangle \\
&= \mathrm{CZ}\,|+_1 +_2\rangle
\end{aligned}
\tag{3.3.3}
$$

and observe that this is exactly the definition of a cluster state that we introduced in Chap. 2. Eq. (3.3.3) also holds for larger constellations of nodes, each representing a

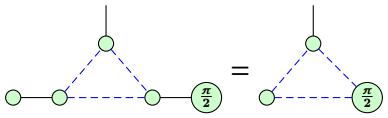


Figure 3.1.: Example of an MBQC-protocol. The measurement operators for the lower two qubits of the cluster state are contracted into the node. It means that those qubits must be measured in the x-basis (Z-nodes) with angles 0 and $\pi/2$, respectively.

qubit, as long as two nodes are connected by at most one Hadamard edge.

A cluster state only has Z-nodes, and measurements are single nodes attached to these Z-nodes. Since we can merge adjacent nodes of the same color, we can contract Z-measurements straight into the Z-nodes of the cluster state. An example is shown in Fig. 3.1. There, the lower two qubits of the cluster state would, in the expanded notation, have projective measurement operators attached to them—a Z-node with phase 0 and phase $\pi/2$, respectively, for the left and right qubits. Instead, the measurement nodes are contracted into the respective qubits according to rule (3.1.6). The upper qubit retains its degree of freedom. Thus, the full diagram represents a single-qubit state which is the result of a measurement-based calculation. It is left as an illustrative exercise to the reader that this state is actually $|0\rangle - \mathrm{i}\,|1\rangle =$ —$\left(-\frac{\pi}{2}\right)$—.
Any node inside the cluster state that has no measurement operators or protruding legs attached to it can be interpreted as including an intrinsic measurement prompt. This reduces the complexity of the notation of MBQC within the ZX-calculus even further.

3.3.2. Post-measurement corrections

When covering MBQC in the framework of the ZX-calculus, it is, as mentioned above, usual to assume that the outcomes of measurements are always 0 (or +, respectively). Of course, in an experimental setup, this is not justified. We show here that this simplified assumption does not destroy the power of MBQC. A detailed discussion, along with an extensive example, can be found in Ref. [59].

If a measurement outcome other than 0 were to occur, then the "error" has to be propagated through the graph, and it possibly affects future measurements. This propagation of the measurement error is done through a classical channel. The operator, who decides on the measurement axes and angles, needs to detect the outcome and, if it is not the desired outcome, alter some of the measurement angles for the measurements that are still to come.

How undesired outcomes affect future measurements can be deduced visually within the framework of the ZX-calculus. An example of this error propagation is shown in Fig. 3.2. The cluster state under investigation consists of four qubits and has five inter-connections, and it is depicted in Fig. 3.2a) and Fig. 3.2b) with abbreviated and explicit notations of

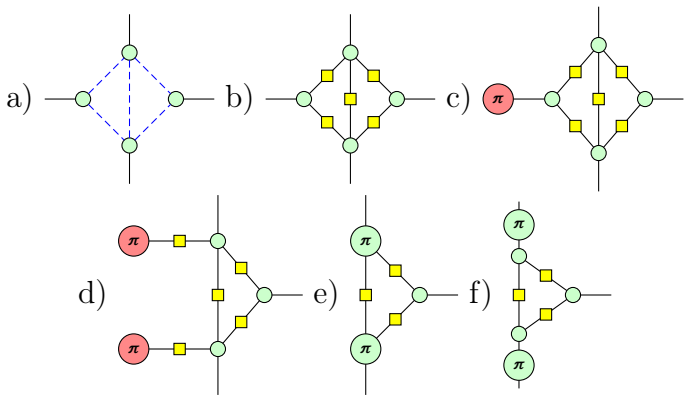


Figure 3.2.: Propagation of measurement outcomes other than 0. a) Underlying cluster state with simplified Hadamard-edges. b) Visualization of the cluster state with explicit Hadamard edges. c) A measurement in the computational basis with undesired outcome 1 is performed on the left qubit. d) Rule (3.1.12) decouples the graph at the measured qubit. e) Rules (3.2.4) and (3.1.6) contract the phases into the upper and lower qubits. f) The additional phases on the upper and lower qubits can be interpreted as corrections of the measurement direction on the respective qubits.

Hadamard edges, respectively. The left qubit is measured first within the protocol of MBQC. The measurement basis is the computational basis, denoted by a projection onto $|0\rangle$ or $|1\rangle$, represented by a (red) X-node, either with phase 0 or π, in the ZX-calculus. Usually, the outcome 0 is assumed. Here we assume the outcome 1, which is represented by the measurement projector π— in the ZX-calculus, which is applied to the left qubit, as shown in Fig. 3.2c).
To see the effect of this measurement, we apply rule (3.1.12) and rule (3.2.4), obtaining the diagram in Fig. 3.2f). The Z-nodes with a phase π attached to the upper and lower qubits are interpreted as an alteration of the measurements on those qubits. If these qubits were to be measured in the x-basis (Z-nodes), the outcomes would need to be swapped. To check if the swapping would indeed not occur for the desired outcome in the measurement, we apply rule (3.1.9) after replacing the phase π in Fig. 3.2c) with zero. Propagating this through the diagram will eradicate the phase-π nodes in Fig. 3.2f).

Undesired measurement outcomes can always be propagated through the graph to alter future measurements. Eventually, there must be a final measurement or several final measurements that extract some information from the state. This would be the result of the calculation. At this point, a plethora of corrective phases would be propagated to the final measurements from all previous measurements. The outcome or outcomes of the final measurement have to be corrected according to these phases in order to obtain the correct result for the computation. Since this is always possible, we restrict the treatment of algorithms within the framework of MBQC to measurements with outcomes 0 or + to drastically reduce the complexity of the measurement protocols.

3.3.3. MBQC-protocol

We have now introduced all necessary ingredients for the description of MBQC within the framework of the ZX-calculus. In the following, we will summarize the general protocol for MBQC with ZX-diagrams. Using the example in Fig. 3.1, we will describe the prerequisites, the calculations, and the readout of a measurement-based calculation.

The prerequisite for MBQC is a cluster state. In terms of the ZX-calculus, a cluster state consists of only Z-nodes with zero phase that are connected via Hadamard edges (blue and dashed). This state is the resource that contains all the entanglement (enough and not too much [51]) that is required for universal quantum computing. In the example in Fig. 3.1, we have cluster state consisting of three qubits that are circularly inter-connected (triangle).

The calculation is carried out by applying projective single-qubit measurements. These measurements can be represented by single nodes that are attached to the protruding legs of the cluster state diagram. Contracting the Z-measurement nodes into the cluster state nodes, as noted in Sec. 3.3.1, leads to an even compacter notation. All Z-nodes without protruding legs then count as measured qubits, and all other Z-nodes (with protruding legs) of the cluster state represent qubits that have not yet been measured. The protruding degrees of freedom designate that the diagram represents a quantum state or quantum operator that can still be manipulated or read out with further measurements. In Fig. 3.1, the cluster state of three qubits is already measured on the bottom two qubits, as these nodes do not have protruding legs. The left qubit is measured along the axis with a phase angle 0 in the x-y-plane (Z-nodes), that is, in the Pauli-x-eigenbasis. The right qubit is measured in the same plane with a phase angle $\pi/2$, that is, in the Pauli-y eigenbasis. Both measurements are assumed to have yielded the first outcome, corresponding to the projectors $\langle + | = \langle 0 | + \langle 1 |$ and $\langle 0 | + \mathrm{i} \langle 1 |$, respectively.

A computation, be it a complex algorithm or a simple calculation, needs to have an output. Since quantum states can, generally, not be accessed directly, we need to extract the information through measurements. One possibility for reading out the result is that there are still protruding edges in the diagram after the algorithm finishes, as depicted in Fig. 3.1. The algorithm would then have to be designed in such a way that a measurement of those degrees of freedom will give a distinct string of information that can help solve the underlying problem the algorithm intends to solve. For example, since the diagram in Fig. 3.1 represents the state $|0\rangle - \mathrm{i}\,|1\rangle$, a measurement in the y-basis would yield the outcome 1 with certainty and could give the answer to a yes/no question.
Another possibility for extracting information is to measure all qubits during the execution of an algorithm. The algorithm could then be designed so that all but one combinations of measurement angles would collapse the state. Then, the algorithm would give an answer to a yes/no question in the sense that the state either collapses or gives a (non-random) result. We will discover that such a mechanism is valid for the Deutsch-Josza-algorithm that we will cover in Chap. 4.

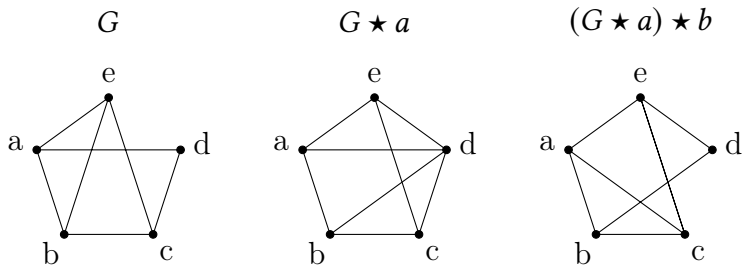


Figure 3.3.: Local complementation of an example graph. Graph G is first complemented around a to yield $G \star a$. The latter is complemented around b to yield $(G \star a) \star b$.

3.4. Advanced rules

In the previous sections, we have introduced a set of rules that can be used to transform ZX-diagrams. These rules are the basic arithmetic operations of the ZX-calculus. They were proven by writing them down in the bra-ket notation as they only center around one node or edge. In this section, we will introduce larger-scale identities for ZX-diagrams that incorporate transformations of whole sections of a ZX-graph. For this, we need to include some basic operations of graph theory. Thorough introductions in the context of graph-theoretic quantum computing (ZX-calculus) can be found in Refs. [12, 61] and we use their notation. Local complementation of graphs was introduced in Ref. [62].

A ZX-diagram has an underlying mathematical graph. The vertices of the graph are the qubits (Z-nodes), and its edges are the bonds of the cluster state (edges in the ZX-diagram). Two important operations on graphs are *local complementation* and *pivoting*. The former adds and removes edges around a node in a defined pattern. The latter is the term for changing the interconnectivity in the neighborhood of two nodes connected by one edge. We will use both of these methods in corresponding advanced rules in the ZX-calculus.

If G is a mathematical graph, and u is a vertex in G, we call $G \star u$ the *local complementation* of G around u. It contains the same vertices as G but has edges between all neighbours v and w of u if and only if w and u are not connected in G. All remaining edges are the same as in G. An example for local complementation is shown in Fig. 3.3 on a graph with five vertices.

The other graph operation that we will cover here is focused on two nodes. When u and v are two vertices in a graph G that are connected through an edge, we introduce the *pivot* of G with respect to the edge uv. It is defined as $G \wedge uv = G \star u \star v \star u$. Pivoting is better understood when we regard distinct subsets of the neighborhoods of u and v inside G. The following reasoning is depicted in Fig. 3.4. Divide the union of the neighborhoods of u and v into the exclusive neighborhoods of u and v, called U and V, and the shared neighborhood UV. Then, pivoting with respect to uv means that we pairwisely complement the edges connecting U, V, and UV. On top of that, we exchange

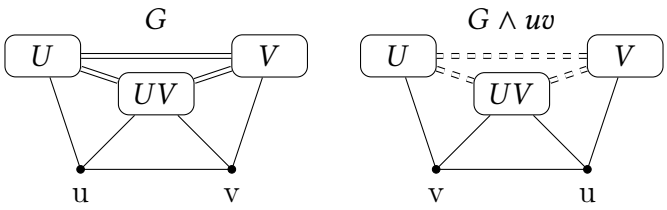


Figure 3.4.: Pivoting a graph with respect to an edge uv. The edges connecting the exclusive neighborhoods of u and v, U and V, and their shared neighborhood UV, are pairwisely complemented. Also, the nodes u and v are exchanged.

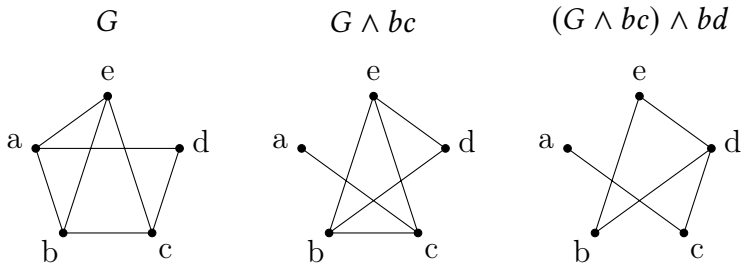


Figure 3.5.: Pivoting example graph. Graph G is first pivoted with respect to the edge bc to yield $G \wedge bc$. The latter is pivoted around cd to yield $(G \wedge bc) \wedge bd$. Note that, instead of exchanging the pivoting anchors, we redistribute the edges accordingly.

u and v. In Fig. 3.4, the distinct neighborhoods are depicted as rounded rectangles, and the sets of edges in between as double lines. Complementation is denoted by dashed double lines. An example of pivoting is shown in Fig. 3.5, reusing the example graph from Fig. 3.3.

The importance of the aforementioned graph operations is that we can use them in the ZX-calculus to define advanced rules that exceed the standard rules in complexity. It can be shown that the required properties for quantum information processing on multipartite states are conserved when applying the following rules.

In a ZX-diagram that is in MBQC-form, attaching a $-\pi/2$-X-node to node u and $\pi/2$-Z-nodes to its neighbors is equivalent to a local complementation of the graph around u. This rule is depicted in Fig. 3.6 and was proven using the standard rules of the ZX-calculus in Ref. [63].

The most complex ZX-rule that we introduce here involves graph pivoting. It was shown in Ref. [64] that we can achieve the pivoting of the graph of an MBQC-ready ZX-diagram with respect to two qubits by applying Hadamard nodes to those qubits and π-Z-nodes to the cross section of the neighborhoods of the qubits. The rule is depicted in Fig. 3.7. In the first equality, we have indicated the exchange of the two pivots by the crossing protruding legs on the top. In the last equality, we have resolved this crossing by actually exchanging the two top nodes.

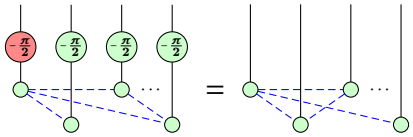


Figure 3.6.: Complementing rule of ZX-calculus. An X-node with phase $-\pi/2$, attached to a node complements the neighborhood of said node and adds Z-nodes with phase $-\pi/2$ to the neighbors.

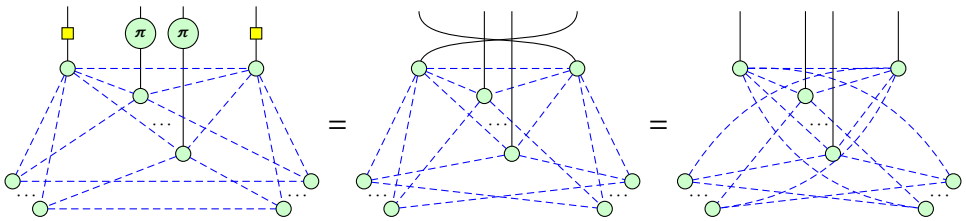


Figure 3.7.: Pivoting rule of ZX-calculus. The pivot of a ZX-diagram in MBQC-form can be achieved by modifying the pivots by Hadamard- and the shared neighborhood by Z-nodes with phase π. The first two diagrams are redrawn after Ref. [12].

From Fig. 3.6 and Fig. 3.7 follow two graph simplifications that were proven in Ref. [12]. The first one deletes a node with a phase that is a multiple of $\pi/2$ from a graph. It makes use of graph complementation and is depicted in Fig. 3.8. The second one deletes two adjacent nodes with a phase that is a multiple of π and utilizes graph pivoting. This identity is depicted in Fig. 3.9. These simplification will have a prominent role in Chap. 4 when we simplify graphs after we have carried out single-qubit measurements, hence reducing the size of the cluster-state graph.

3.5. Related descriptions of quantum computing

In the previous sections, we have explained everything about the ZX-calculus that we require to treat algorithms in terms of MBQC, which we will do in Chap. 4. Before we delve into the analysis of algorithms, we will take a short excursion. We will answer the question if the ZX-calculus has some unique properties that make it the most suitable for treating quantum mechanics in a graph-like language. We will discover that there are many more formalisms for graphical treatment of quantum states and operations. Some

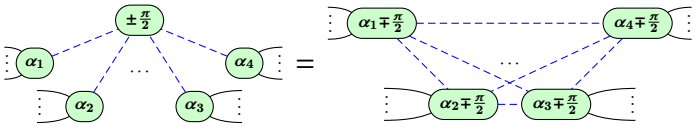


Figure 3.8.: A node with phase that is a multiple of $\pi/2$ is erased while using complementation of the neighborhood. The image is redrawn after Ref. [12].

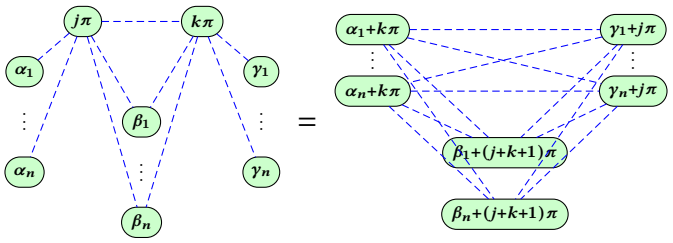


Figure 3.9.: Method of erasing two nodes with a phase that is a multiple of π. The resulting graph is the result after pivoting with respect to these two nodes. The image is redrawn after Ref. [12].

are closely related to the ZX-calculus, some are more abstract and can be viewed as abstractions and generalizations of calculi like the ZX-calculus. Of course, all formalisms covered here are equivalent in their computational power, that is, they are able to represent universal quantum computing.

3.5.1. ZH- and other calculi

The ZX-calculus is only one of many graphical calculi for graph states. Indeed, it was the first one to be introduced [11]. The equivalent power of all such calculi in terms of quantum computing has its foundation in *categorical quantum computing* (see Sec. 3.5.3), of which the calculi are only specific representations. We will briefly describe the ZH- and the scalable ZX-calculus (SZX-calculus) here.

The nodes in the ZX-calculus are built around two complementary bases, where Z- and X-nodes correspond to the eigenbases of the x- and the z-Pauli-matrices, respectively. Another calculus that is based around two different bases is the ZH-calculus [40]. The first type of nodes in the ZH-calculus are the same Z-nodes as in the ZX-calculus. In contrast to the ZX-calculus, there are no X-nodes in ZH-calclulus, but instead n-ary generalizations of the Hadamard gate, represented by nodes in the shape of a yellow box. These H-boxes are defined as

$$n \,\vdots\, \boxed{\alpha} \,\vdots\, m = \sum \alpha^{i_1 i_2 \ldots i_n j_1 j_2 \ldots j_m} \left|i_1 i_2 \ldots i_n\right\rangle \left\langle j_1 j_2 \ldots j_m\right| \tag{3.5.1}$$

for a node with n incoming and m outgoing legs. It is evident that Eq. (3.5.1) represents a tensor of which all entries but one are 1. The remaining entry is the phase α. The node $\boxed{-1}$ represents the Hadamard gate and, as such, the yellow box edge from the ZX-calculus. It is easy to show that the generators from the ZX-calculus can be built by the generators of the ZH-calculus. The Z-nodes are the same, and the X-node can be created by attaching H-boxes with a phase of -1 to all protruding edges of Z-nodes.
While the ZX-calculus is well-suited for describing graph states, the ZH-calculus has its advantages in treating hypergraph states, an extension to graph states that are also relevant for MBQC [65], as well as quantum search algorithms [66] and various types of proofs of non-locality [67].

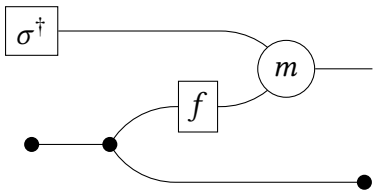


Figure 3.10.: Topological graph of the Deutsch-Jozsa algorithm, redrawn after [68].

The ZX-calculus can be understood as an elementary, low-level, programming language for quantum information, analogous to assembly language for classical computers. It can handle at most one qubit in one vertex of the graph and the amount of entanglement of two qubits within one edge. A scalable ZX-calculus (SZX-calculus) was introduced in Ref. [41] to incorporate many degrees of freedom into one edge of the graph and many qubits into one node of the graph. As a key element they provide the *dividers* and *gatherers*

 , (3.5.2)

where the thick lines represent many ZX-edges, out of whom one is separated (thin line). They are used to switch between the handling of single qubits and registers of qubits. With the feature of covering registers of qubits, the SZX-calculus is well-suited to describe larger calculations and algorithms.

3.5.2. Topological quantum computing

Another graphical calculus is the topological analysis of quantum algorithms [68–70]. It uses categorical topological symbols to represent the elements of quantum algorithms: initial states, readouts, and entanglement operations in diagrams. An example is depicted in Fig. 3.10, which shows the topological description of the Deutsch-Jozsa algorithm that will also feature in Chap. 4. An advantage of the topological graph description is that the inner workings of an algorithm can appear much more clearly than in the canonical circuit model, in which the "quantumness" sometimes appears hidden underneath the Turing-like model. In Fig. 3.10, the initial states of the algorithm are prepared in the left two nodes, the main calculation (oracle) is executed in the f-node, and the readout of the result is measured at the only protruding leg of the diagram. The snake-like connection of the nodes visualizes entanglement as the resource of the quantum computation at hand. Topological quantum computing is mainly used to prove the correctness and investigate the inner workings of quantum algorithms. A good introduction can be found in the original paper [68].

3.5.3. Categorical quantum mechanics

Topological quantum computing as well as the calculi introduced here, including the ZX-calculus, are simply representations of categorical quantum mechanics. In standard textbooks on quantum mechanics, the physics is usually explained using a defined set of rules. For example, quantum states are unit vectors in a Hilbert space, and measurements are Hermitian operators. However, one can ask the questions of why is it a Hilbert space, what do the non-normalized vectors represent, and why do we work with complex numbers when measurements can only ever yield real results. Categorical quantum mechanics treats the field not by using operators and rules, but instead by using (monodial) categories to explore the reasons why canonical quantum mechanics works in the first place. In a slightly poetical manner, categorical quantum mechanics is "[...] showcasting the forest rather than the trees" [39, p.3, l.3f].

A category incorporates two structures: *objects* and *morphisms* [38]. These terms describe only abstract algebraic structures and are not restricted to members from physics or mathematics. In category theory, the contents or inner structures of the objects step into the background and mathematical insights are extracted solely from the relationships between objects and between categories (*functors*). The property that the relations are more important than the objects themselves, like, for example, in group theory, is rare in mathematics. A prominent category for the description of quantum mechanics is **Hilb**, the category of Hilbert-spaces (objects) and bounded linear maps (morphisms), from which the canonical description of quantum mechanics can be derived [39]. The most important categories for quantum mechanics are *monoidal categories*, which allow processes (morphisms) to evolve in parallel as well as sequentially. Looking at it from the point of view of Hilbert-space quantum mechanics, this is a clear hint a a link to tensor products of states or operators.

Category theory also has its own graphical notation, which visualizes morphisms as the key element of relations between objects (or categories). An example is displayed in Fig. 3.11. Lines represent identity morphisms. Objects could be denoted as letters at the ends of lines, but are omitted if not necessary. Recall that the objects only play a secondary roll. Morphisms from the identity to any image are drawn as triangles. The trapezoidal node is a morphism that is asymmetric under the dagger operation of monoidal dagger categories. Actually, Fig. 3.11 depicts the Deutsch-Jozsa algorithm in the graphical notation and framework of dagger compact closed categories. Note the apparent similarity to Fig. 3.10, which is expected because topological quantum computing is only one subform of categorical quantum mechanics. For a thorough introduction to categorical quantum mechanics, the interested reader is referred to Ref.[39]. Further derivations and connections to graphical calculi like the ZX-calculus can be found in Refs. [11, 37, 38, 71].

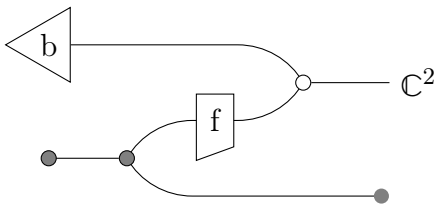


Figure 3.11.: Categorical description of the Deutsch-Jozsa algorithm, interpreted as the graphical notation of the morphism within the category **FHilb** of finite dimensional Hilbert spaces and bounded linear maps. Redrawn (and rotated) from [39].

4. Algorithms in MBQC

Using the knowledge of measurement-based quantum computing (MBQC) and its representation in the graphical language of the ZX-calculus from Chap. 2 and Chap. 3, we can now treat quantum algorithms. Since the Deutsch-Jozsa-algorithm serves as a canonical introduction to algorithms that have quantum superiority, we will also cover it first. After showcasing the power of MBQC and the ZX-calculus by deriving a three-qubit MBQC version of the Deutsch-Jozsa algorithm, we will turn to the Simon algorithm, for which we will derive a method to systematically build any oracle from a distinct set of gates. Building on this method, we will explicitly formulate the two-qubit Simon algorithm within MBQC and present the structure of the graph that is required for the n-qubit Simon algorithm.

Large parts of the first section (Deutsch-Jozsa algorithm) in this chapter have previously been published in Ref. [1]. Large parts of the second section (Simon algorithm) have, at the time of submission of this thesis, been submitted for peer review and publishing. The submission is published on the preprint server as Ref. [2]. In both cases, the contributions of the second author included fruitful discussions and editorial work.

4.1. Deutsch-Jozsa algorithm

For the Deutsch-Jozsa algorithm, we will first introduce the theory behind the algorithm, then formulate the two-qubit version within the circuit model and translate it to MBQC using the ZX-calculus. Subsequently, we will treat the three-qubit version using the same procedure, utilizing a scheme for the general description of its oracles within the circuit model. The three-qubit Deutsch-Jozsa algorithm within MBQC can be implemented on an eleven-qubit graph state. As an alternate picture, we will also introduce a graph state on a rectangular lattice that can carry out the three-qubit algorithm using projective measurements only.

4.1.1. Circuit formulation

We have already introduced the one-qubit version of the Deutsch-Jozsa algorithm in Sec. 2.1.5, but repeat it here for the purpose of readability. Since this subsection is a repetition, it is possible to skip straight to the next subsection. The aim of the Deutsch-Jozsa algorithm is to determine if a function is *constant* or *balanced* [19]. We take a function $f : Z_2^n \ni \boldsymbol{\sigma} \rightarrow f(\boldsymbol{\sigma}) \in Z_2$ that maps a binary representation $\boldsymbol{\sigma}$

	Variants			
	(i)	(ii)	(iii)	(iv)
$f(0)$	0	1	0	1
$f(1)$	0	1	1	0

Table 4.1.: Output variants of a one-bit boolean function. Variants (i) and (ii) are constant, variants (iii) and (iv) are balanced. This table is a duplicate of Table 2.1 and is repeated here for the purpose of readability.

of length n to one bit $f(\boldsymbol{\sigma})$. We call $f(\boldsymbol{\sigma})$ *constant* if the outcome is the same for all choices of $\boldsymbol{\sigma}$, i.e., $\forall \boldsymbol{\sigma} \in Z_2^n : f(\boldsymbol{\sigma}) = 0$ or $\forall \boldsymbol{\sigma} \in Z_2^n : f(\boldsymbol{\sigma}) = 1$. We call the function *balanced* if exactly one-half of the outcomes are 0 and one-half are 1, i.e., $|\{\boldsymbol{\sigma} \in Z_2^n : f(\boldsymbol{\sigma}) = 0\}| = |\{\boldsymbol{\sigma} \in Z_2^n : f(\boldsymbol{\sigma}) = 1\}|$. It is important to emphasize that the algorithm is based on the premise that f can only be constant or balanced and nothing else. The possible variants of f in the case of one bit ($n = 1$) are tabulated in Table 4.1.

It is easy to show that a quantum computer can solve this problem with less work than a classical computer. Suppose we have an oracle that returns the bit $f(\boldsymbol{\sigma})$ for any $\boldsymbol{\sigma}$ that we feed into it. A classical computer requires $2^{n-1} + 1$ queries of the oracle to solve the problem in the worst case, in which f is balanced, but the first $2^n/2 = 2^{n-1}$ queries all yield the same output. On a quantum device, one can determine the character of the function f with just one call to the oracle, i.e., at constant cost.

The canonical way of formulating the Deutsch-Jozsa algorithm makes use of an auxiliary qubit. Here, however, we will work with a variant of the algorithm that uses no auxiliary qubit, but only the query qubits. In the following, we will profit from the reduced number of qubits as we translate the algorithm to measurement-based quantum computing.

The single-qubit-input variant of the one-bit Deutsch-Jozsa algorithm is shown in Fig. 4.1. (All quantum circuits were typeset with the help of the *Quantikz*-package [47].) We prepare one working qubit in the state $|+\rangle = |0\rangle + |1\rangle$. We then apply an oracle operation to the working qubit, where the effect of the oracle depends on the character of the instance of the function f under investigation. If $f(\sigma)$ is 1, then the oracle adds a phase of -1 to the projection onto the basis state $|\sigma\rangle$. That is, it transforms the input state as

$$|+\rangle = |0\rangle + |1\rangle \xrightarrow{\text{oracle}} (-1)^{f(0)} |0\rangle + (-1)^{f(1)} |1\rangle \,. \tag{4.1.1}$$

The oracle can be realized by the identity operation if f is constant and by a Pauli-Z-gate if f is balanced.

After the application of the oracle, the only remaining step is to measure the working qubit in the x-basis. If the measurement yields the outcome corresponding to the basis state $|+\rangle$, then f is constant. For the other outcome, f is balanced. In the one-qubit case, it can easily be checked that the oracle of Eq. (4.1.1) will transform $|+\rangle$ to $\pm|+\rangle$ if and only if $f(0) = f(1)$. Note that the overall phase of ± 1 does not have physical relevance and cannot be measured. If, on the other hand, $f(0) \neq f(1)$, the oracle will transform

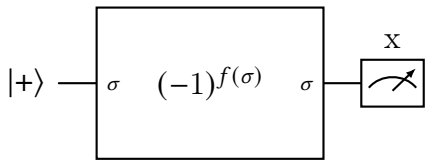


Figure 4.1.: Deutsch-Jozsa algorithm for one qubit. The oracle adds a phase of -1 to each basis element $|\sigma\rangle$ when $f(\sigma) = 1$. Measurement of $|+\rangle$ reveals that f is constant, $|-1\rangle$ that f is balanced. This figure was already introduced in **??**, but it is repeated here for the purpose of readability.

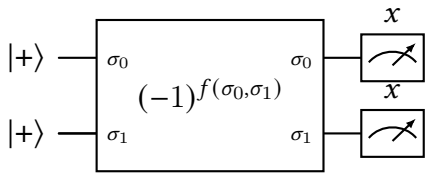


Figure 4.2.: Two-qubit Deutsch-Jozsa algorithm. The variants of the oracle are depicted in Fig. 4.3.

$|+\rangle$ to $\pm|-\rangle$. Hence, a measurement in the x-basis will determine the character of f.

4.1.2. Two-qubit Deutsch-Jozsa

The Deutsch-Jozsa algorithm for two or more qubits is a relatively straightforward extension of the one-qubit case. Here we will treat the two-qubit case explicitly, taking it as an example to illustrate how to use ZX-calculus to systemically convert the circuit-model implementation to a measurement-based implementation. A description of the original measurement-based implementation of the two-qubit Deutsch-Jozsa algorithm can be found in Ref. [72]. We remark that Ref. [72] uses the variant of the algorithm that utilizes an auxiliary qubit. This variant requires entangling operations. In the following, we will show, in a visually evident way using ZX-diagrams, that no entanglement is necessary if one utilizes the formulation of the Deutsch-Jozsa algorithm with no auxiliary qubit.

For the two-bit case, the function $f(\sigma_0, \sigma_1)$ has eight possible realizations, each of which must be either constant or balanced; they are listed in Table 4.2. The quantum algorithm depicted in Fig. 4.2 in schematic circuit representation, is used to determine which character f has. The two working qubits are prepared in the $|+\rangle$-state. The oracle then adds a phase of -1 to all computational basis states $|\sigma_0, \sigma_1\rangle$ ($\sigma_{0/1} \in \{0, 1\}$) if and only if $f(\sigma_0, \sigma_1) = 1$. A measurement of both qubits in the x-basis reads out the character of f: if $|{+}{+}\rangle$ is measured, then f is constant; if any other state is measured, then f is balanced.

Within a circuit model without auxiliary qubits, the eight variants of the oracle can be implemented as shown in Fig. 4.3. Circuits for the variant with an auxiliary qubit can be found, for example, in Ref. [72]. The oracle will apply one of these circuits to the

	Variants							
	(i)	(ii)	(iii)	(iv)	(v)	(vi)	(vii)	$(viii)$
$f(0,0)$	0	1	0	0	0	1	1	1
$f(0,1)$	0	1	0	1	1	0	0	1
$f(1,0)$	0	1	1	0	1	0	1	0
$f(1,1)$	0	1	1	1	0	1	0	0
α_0	0	0	0	0	π	π	0	π
α_1	0	0	π	0	0	π	0	π
α_2	0	0	0	π	π	π	π	0
α_3	0	0	0	0	0	π	π	0

Table 4.2.: Outputs of a two-bit boolean function that is either *constant* or *balanced*. Variants (i) and (ii) are constant, while variants (iii) through $(viii)$ are balanced. The last four rows show the control angles for the measurement-based version of the algorithm, which is introduced in diagram (4.1.2).

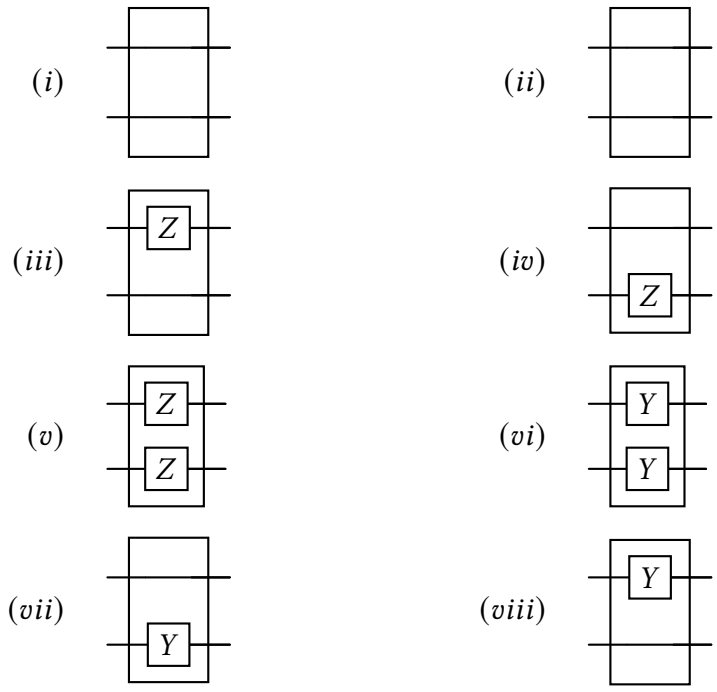
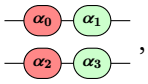


Figure 4.3.: Circuit-model implementations of all variants of the oracle for the Deutsch-Jozsa algorithm on two input qubits as tabulated in Table 4.2. Here X and Y are Pauli gates. Overall phases, which do not alter the outcome of measurements, are ignored.

two working qubits. Note that information about the character of f is hidden to us; it is contained in the choice of the circuit used by the oracle. We know only that the circuit is restricted to one of the eight variants depicted in Fig. 4.3.

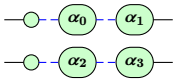
4.1.3. Translation to ZX-calculus

For the circuits in Fig. 4.3, we need to translate the Pauli Z and Y gates. It is evident that $Z =$ and $Y \propto XZ =$. Using these translations, we can write a general instance of the two-bit oracle as



, (4.1.2)

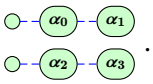
where the angles α_i are chosen by the oracle according to Table 4.2 to represent the different combinations of Y and Z. Using the Hadamard abbreviation (3.2.2), we can translate diagram (4.1.2) to the measurement-based quantum-computing description



. (4.1.3)

Note that we have added an extra identity = — to the left of each row so that the Hadamard edge does not protrude.

Recall that the Deutsch-Jozsa algorithm starts with the working qubits in the $|+\rangle$ state and ends with a measurement $\langle +|$ of the same state. In the ZX-calculus, both are represented by . By adding this to diagram (4.1.3) and using rule (3.1.6) in the trivial way, = , we obtain the diagram for the full algorithm:



. (4.1.4)

According to our description in Sec. 3.3.1, diagram (4.1.4) can be interpreted as two linear cluster states, each with three qubits as a resource state. The qubits of the upper linear cluster state are measured along axes 0, α_0, and α_1, respectively. The lower chain can be interpreted analogously. Unfavorable measurement outcomes, that is, outcomes other than 0, would need to be propagated through the procedure and might affect other measurement angles, as stated above. If one of the measurements yields no result, the state has collapsed. In that case, the tested function f can only be balanced. Conversely, if the state does not collapse, f is constant.

Notice that the measurement-based description exposes the structure of the two-bit case much more clearly than the circuit description. As was shown in Ref. [73], the resource that leads to the quantum supremacy of the Deutsch-Jozsa over classical algorithms is not entanglement but rather the ability of the (quantum) oracle to apply a function to a state that is an arbitrary superposition of basis states. In ZX-diagram (4.1.4), it is evident that there are two strings of information processing (along the two linear cluster

states) that do not interact with one another, i.e., have no entanglement. Nevertheless, the Deutsch-Jozsa algorithm requires only one call to the oracle, in contrast to the two calls of the function required by the classical algorithm. Thus, the ZX-diagram (4.1.4) is in accord with and elucidates the statement of Ref. [73].

4.1.4. Algorithm for three qubits

While, in the two-qubit case, all oracles could be realized using only single-qubit unitary gates, the same simple procedure does not apply for more than two qubits. In general, one needs generalized controlled gates in higher dimensions with arbitrary control conditions, of which the *Toffoli gate* is a simple example. We remark that the publication [1] leading to this thesis was partly inspired by the statement in Ref. [72] that it is possible to implement a three- or multi-qubit Deutsch-Jozsa-algorithm using combination of simple CNOT gates between working qubits and an auxiliary qubit. However, as was shown in Refs. [74] and [75], this is actually not the case—one needs a more comprehensive approach to manage the complexity of three or more qubits.

As in the one- and two-qubit cases, we can write out the possible functions f that are either constant or balanced. For any number of input bits, there are two possible constant functions, all ones or all zeros. The number of balanced functions, however, grows super-exponentially with the number of n of input bits. In particular, there are a total of

$$N_{\text{balanced}} = \begin{pmatrix} 2^n \\ 2^n/2 \end{pmatrix} = \begin{pmatrix} 2^n \\ 2^{n-1} \end{pmatrix} = \frac{(2^n)!}{\left((2^{n-1})!\right)^2} \tag{4.1.5}$$

balanced binary functions on n bits, which is the number of ways of assigning $2^n/2$ ones to the 2^n possible inputs of the function f. For $n = 3$, this amounts to 2 constant functions and 70 balanced functions.

A subset of the possible functions are tabulated in Table 4.3. If we examine, for example, variant (3), which is

$$f(0,0,0), \ldots, f(1,1,1) = 0,\ 0,\ 0,\ 0,\ 1,\ 1,\ 1,\ 1\,, \tag{4.1.6}$$

we see that a quantum oracle for this function is easily implemented by adding the desired phase of -1 if and only if the first qubit is in state 1. If we consider variant (4) of f from Table 4.3, however, we find that the implementation is not that straightforward. The quantum oracle for the balanced function

$$f(0,0,0), \ldots, f(1,1,1) = 0,\ 0,\ 0,\ 1,\ 0,\ 1,\ 1,\ 1 \tag{4.1.7}$$

can no longer be implemented using only single-qubit phases. The reason for this is that adding a phase of -1 to the basis vectors $|011\rangle$, $|101\rangle$, $|110\rangle$, and $|111\rangle$ only is an *entangling operation*. That is, this operation has the effect of transforming a product state into an entangled state and vice versa. In contrast, the operation in the previous example

	Variants							
	(1)	(2)	(3)	(4)	(5)	⋯	(71)	(72)
$f(0,0,0)$	0	1	0	0	0	⋯	1	1
$f(0,0,1)$	0	1	0	0	0	⋯	1	1
$f(0,1,0)$	0	1	0	0	0	⋯	1	1
$f(0,1,1)$	0	1	0	1	1	⋯	0	1
$f(1,0,0)$	0	1	1	0	1	⋯	1	0
$f(1,0,1)$	0	1	1	1	0	⋯	0	0
$f(1,1,0)$	0	1	1	1	1	⋯	0	0
$f(1,1,1)$	0	1	1	1	1	⋯	0	0

Table 4.3.: Constant or balanced outputs of a three-bit boolean function. Variants (1) and (2) are constant, variants (3) through (72) are balanced.

is not an entangling operation because it can be implemented using a one-qubit phase manipulation only. Entangling oracles are a characteristic that categorically separates the case of three or more input qubits from the case of one or two in the Deutsch-Jozsa algorithm [74].

In order to develop a systematic approach to adding the desired relative phases in dependence on the control condition, we closely follow the approach of Refs. [75] and [76]. The relative phase added by the oracle can be expressed as the unitary mapping

$$U\,|\boldsymbol{\sigma}\rangle = \mathrm{e}^{\mathrm{i}\theta(\boldsymbol{\sigma})}\,|\boldsymbol{\sigma}\rangle\ , \tag{4.1.8}$$

where $|\boldsymbol{\sigma}\rangle = |\sigma_0\sigma_1\ldots\rangle$ is a basis vector of the multidimensional Hilbert space, e.g., $|010\rangle$, and $\theta(\boldsymbol{\sigma}) = \pi f(\boldsymbol{\sigma})$ is the applied phase, with $\theta \in \{0,\pi\}$. We can write θ as

$$\theta(\boldsymbol{\sigma}) = \pi\, f(\boldsymbol{\sigma}) = \sum_{\mathbf{x}:f(\mathbf{x})=1} \pi\, \delta_{\boldsymbol{\sigma},\mathbf{x}}\ , \tag{4.1.9}$$

where $\delta_{\boldsymbol{\sigma},\mathbf{x}}$ is 1 if and only if $\boldsymbol{\sigma} = \mathbf{x}$. This means we can write $f(\boldsymbol{\sigma})$ as sum of *Kronecker deltas*. Since each part of the sum in Eq. (4.1.9) yields one if and only if the exact control condition of f is matched and zero otherwise, we can rewrite the nonzero Kronecker deltas as

$$\delta_{\boldsymbol{\sigma},\mathbf{x}} = \prod_{i=0}^{n-1} \Big|\sigma_i - x_i\Big|\ . \tag{4.1.10}$$

As an example, we again take function variant (4) from Table 4.3. Since the function (4) yields one for the inputs $\mathbf{y} \in \{(011),(101),(110),(111)\}$ and zero otherwise, we can

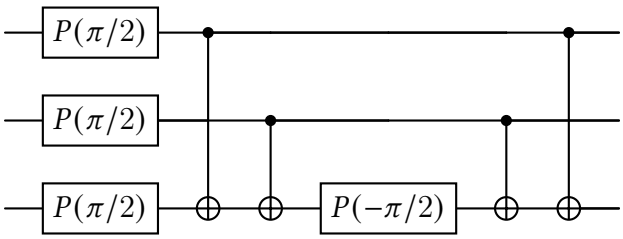


Figure 4.4.: Example of an oracle circuit for the balanced function f (variant (4) in Table 4.3), where $f(000), f(001), \ldots, f(111) = 0, 0, 0, 1, 0, 1, 1, 1$.

write the non-zero deltas as

$$\delta_{(011),\mathbf{x}} = (1 - x_0)x_1x_2 = x_1x_2 - x_0x_1x_2 \tag{4.1.11}$$
$$\delta_{(101),\mathbf{x}} = x_0(1 - x_1)x_2 = x_0x_2 - x_0x_1x_2 \tag{4.1.12}$$
$$\delta_{(110),\mathbf{x}} = x_0x_1(1 - x_2) = x_0x_1 - x_0x_1x_2 \tag{4.1.13}$$
$$\delta_{(111),\mathbf{x}} = x_0x_1x_2 \tag{4.1.14}$$

Adding these up and using the fact that $x_ix_j = \frac{1}{2}(x_i + x_j - x_i \oplus x_j)$ for $x_{i/j} \in \{0, 1\}$ yields

$$\begin{aligned} \theta(\mathbf{x}) &= \pi \left(x_0x_1 + x_0x_2 + x_1x_2 - 2x_0x_1x_2 \right) \\ &= \frac{\pi}{2} \Big(x_0 + x_1 + x_2 - x_0 \oplus x_1 \oplus x_2 \Big) \, . \end{aligned} \tag{4.1.15}$$

The first three phases in Eq. (4.1.15) involve non-entangling operations on the work qubits. Namely, this can be implemented by applying a phase of $\pi/2$, i.e., the one-qubit gate

$$P(\pi/2) = \begin{pmatrix} 1 & 0 \\ 0 & \mathrm{e}^{\mathrm{i}\pi/2} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & \mathrm{i} \end{pmatrix} \, , \tag{4.1.16}$$

to all three qubits. The third term represents a phase of $-\pi/2$ if and only if there is an odd number of ones in the basis vector. Speaking in the language of the *Clifford set*, this can be implemented as the gate sequence

$$\mathrm{CNOT}_{01}\mathrm{CNOT}_{12}\, P_2(-\pi/2)\, \mathrm{CNOT}_{12}\mathrm{CNOT}_{01} \, , \tag{4.1.17}$$

where CNOT_{ij} represents a CNOT gate between qubits i and j, and $P_2(-\pi/2)$ represents the gate $P(-\pi/2)$ [replacing $\pi/2$ with $-\pi/2$ in Eq. (4.1.16)] on qubit 2. Putting these gates together, the oracle for function (4) from Table 4.3 can be implemented as the circuit depicted in Fig. 4.4.

If we code the output combination of the balanced function f in binary representation, i.e., $b_7b_6 \ldots b_1b_0$, we can express which variant we have as a single decimal number. For example, variant (4) in Table 4.3 can be written as $00010111_2 = 23_{10}$. Each bit b_i, $i \in [0, 7]$ then corresponds to one of the eight input basis states. Similarly to the

$P(\pi/2)$ gate, Eq. (4.1.16), we can express the general phase function as

$$\begin{aligned}
&\theta[b_7,\ldots,b_0](\mathbf{x}) \\
&\quad= \frac{\pi}{2}\Big(2b_7 + \Big(-b_6 - b_5 + b_2 + b_1 + \frac{1}{2}b_0\Big)\,x_1 \\
&\qquad\quad + \Big(-b_6 + b_4 - b_3 + b_1 + \frac{1}{2}b_0\Big)\,x_2 \\
&\qquad\quad + \Big(-b_5 + b_4 - b_3 + b_2 + \frac{1}{2}b_0\Big)\,x_3 \\
&\qquad\quad - \Big(b_7 - b_5 - b_3 + b_1 + \frac{1}{2}b_0\Big)\,x_1 \oplus x_2 \\
&\qquad\quad - \Big(b_7 - b_6 - b_3 + b_2 + \frac{1}{2}b_0\Big)\,x_1 \oplus x_3 \\
&\qquad\quad - \Big(b_7 - b_6 - b_5 + b_4 + \frac{1}{2}b_0\Big)\,x_2 \oplus x_3 \\
&\qquad\quad + \frac{1}{2}b_0\,x_1 \oplus x_2 \oplus x_3\Big)\,.
\end{aligned} \tag{4.1.18}$$

This phase logic can be straighforwardly implemented as the circuit depicted in Fig. 4.5, where the one-qubit phase gates can be derived from Eq. (4.1.18) as

$$P_1 = P\Big(\frac{\pi}{2}\Big(-b_6 - b_5 + b_2 + b_1 + \frac{1}{2}b_0\Big)\Big) \tag{4.1.19}$$

$$P_2 = P\Big(\frac{\pi}{2}\Big(-b_6 + b_4 - b_3 + b_1 + \frac{1}{2}b_0\Big)\Big) \tag{4.1.20}$$

$$P_3 = P\Big(\frac{\pi}{2}\Big(-b_5 + b_4 - b_3 + b_2 + \frac{1}{2}b_0\Big)\Big) \tag{4.1.21}$$

$$P_{12} = P\Big(-\frac{\pi}{2}\Big(b_7 - b_5 - b_3 + b_1 + \frac{1}{2}b_0\Big)\Big) \tag{4.1.22}$$

$$P_{13} = P\Big(-\frac{\pi}{2}\Big(b_7 - b_6 - b_3 + b_2 + \frac{1}{2}b_0\Big)\Big) \tag{4.1.23}$$

$$P_{23} = P\Big(-\frac{\pi}{2}\Big(b_7 - b_6 - b_5 + b_4 + \frac{1}{2}b_0\Big)\Big) \tag{4.1.24}$$

$$P_{123} = P\Big(\frac{\pi}{4}b_0\Big)\,. \tag{4.1.25}$$

Note that we have neglected the term $\pi\, b_7$ from Eq. (4.1.18) because it leads solely to an overall phase that has no physical significance. Note also that such a circuit for the general four-qubit Deutsch-Jozsa oracle has been derived in Ref. [75].

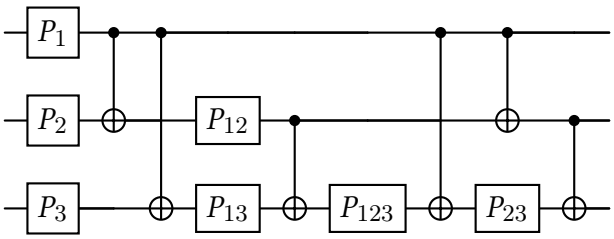
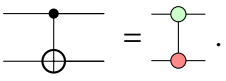


Figure 4.5.: General oracle circuit of the three-qubit Deutsch-Jozsa algorithm. The one-qubit phase gates P_α are determined by the realization of the oracle function $f(x_0, x_1, x_2)$ and can be constructed according to Eqs. (4.1.19)–(4.1.25).

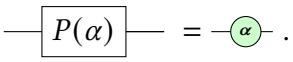
4.1.5. Translation to MBQC

Now that we have derived the general oracle circuit for the three-qubit Deutsch-Jozsa algorithm, Fig. 4.5, we want to translate this circuit into a ZX-diagram. As proven in Eq. (3.2.1), we can write a CNOT operation on two qubits as



(4.1.26)

Furthermore, each single-qubit gate in Fig. 4.5 can be translated as



(4.1.27)

Applying identities (4.1.26) and (4.1.27), Fig. 4.5 translates to the ZX-diagram depicted in Fig. 4.6. A general description of the translation process as well as additional applications are given in Ref. [11]. This raw translation of a circuit to a ZX-diagram is, however, not yet in a form suitable for MBQC, which can consist of only Z-nodes and Hadamard edges, as discussed in Sec. 3.3.1.

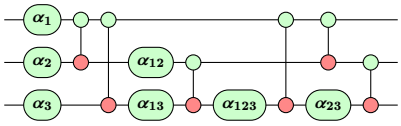


Figure 4.6.: Raw translation of the circuit in Fig. 4.5 to a ZX-diagram.

In order to obtain a measurement-based description of the three-qubit Deutsch-Jozsa algorithm, we must simplify and rewrite the graph in Fig. 4.6 so that it fulfills the requirements for MBQC. First, we recall that the algorithm starts with the qubits being initialized to be in the $|+\rangle$ state. It ends with the measurement of the "all $+$"-state $\langle + \cdots + |$. Therefore, we must add $-\!\circ$ nodes to all incoming and outgoing legs of the diagram of Fig. 4.6 to obtain a description of the full algorithm, not just the oracle.

This leaves us with the closed ZX-diagram given in Fig. 4.7. Since the diagram has no

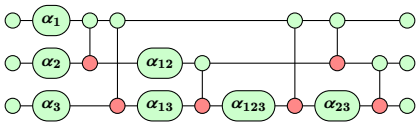


Figure 4.7.: Closed form of the raw translation with initial states and final measurements.

protruding legs, it represents a scalar. Thus, it immediately determines the result of all measurements.

In the ZX-diagram of Fig. 4.7, we apply the color-swapping rule (3.2.4) to all X-nodes (red) and the contraction rule (3.1.6) to all adjacent Z-nodes (green) that are connected through regular edges (solid), arriving at the diagram given in Fig. 4.8 We notice that

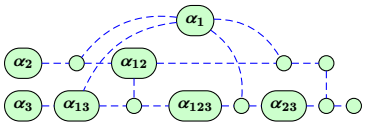
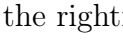


Figure 4.8.: ZX-diagram of the Deutsch-Jozsa algorithm with all nodes changed to Z-nodes and all adjacent Z-nodes contracted.

we can remove the rightmost node on the middle row in this diagram because  ($H^2 = I$). Furthermore, we again use the color-swap rule (3.2.4) to change the "tail" on the rightmost end of the third row to an X-node (red). This yields the diagram in Fig. 4.9. We now apply rule (3.1.9),

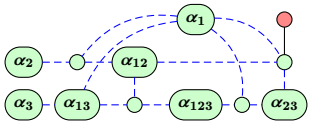
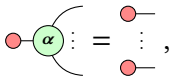
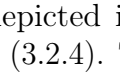


Figure 4.9.: ZX-diagram of the Deutsch-Jozsa algorithm after eradicating all combinations of nodes that represent the identity.



(4.1.28)

which states that an X-measurement (X-node/red node) on a Z-node (green) will decouple and erase the latter inside the graph, to the rightmost node in the middle row, obtaining the diagram depicted in Fig. 4.10. Finally, we notice that  because of the color-swap rule (3.2.4). This means that, using the contraction rule (3.1.6), we can merge the X-node (red) into its neighbor. After some rearrangement, we obtain the final result, Fig. 4.11.

The diagram of Fig. 4.11 can be interpreted in the language of MBQC as follows: Prepare a cluster state with the geometry of the diagram, where nodes are qubits and

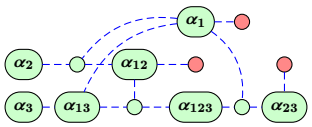


Figure 4.10.: Partially decoupled ZX-diagram of the Deutsch-Jozsa algorithm.

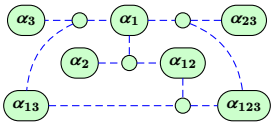


Figure 4.11.: Full ZX-diagram of the three-qubit Deutsch-Jozsa algorithm in measurement form.

edges depict entanglement. Then measure all qubits in their respective bases at the given angles. If no angle is given, measure in the x-basis. The angles of measurement to implement a given realization of the oracle are given by Eqs. (4.1.19)–(4.1.25). For unfavorable measurement outcomes, that is, measurements that do not yield the first outcome, corrective measurement angles are propagated through the diagram, as explained in Sec. 3.3.2. Since we have derived a graph with a measurement pattern whose outputs are all in the $|+\rangle$-state, the following statement holds: If any of the measurements yield no outcome, meaning that the state has collapsed beforehand, then the tested function is balanced. If we obtain an outcome for all measurements, then the tested function is constant.

4.1.6. Algorithm on a rectangular lattice

For experimental implementations, it can be convenient to embed the aforementioned pattern into a rectangular lattice cluster state because such lattice states are more straightforward to treat than cluster states corresponding to arbitrary graphs, i.e., do not require special attention to the geometry of the resource state. By adding a few more qubits, we can extract the pattern of our algorithm from a rectangular lattice. We can use the decoupling rule (3.1.9) to decouple qubits from a cluster state and thus effectively remove them. The complementing rule of ZX-calculus, depicted in Fig. 3.8, states that a measurement at an angle of $\pm\pi/2$ in the XY-plane leads to the removal of that qubit along with complementing the subgraph consisting of all qubits in the neighborhood of the qubit. Given these two rules, it is straightforward to show that the diagram of Fig. 4.12 is equivalent to that of Fig. 4.11. The derivation of this diagram follows the same rules and principles as above, but this time it is the aim to apply the rules in the opposite direction to enlarge the graph in the right way to make it a rectangular lattice. We state explicitly that we do not claim to have proven this lattice to be the smallest possible lattice cluster state that can contain the three-qubit Deutsch-Jozsa algorithm. Instead, our formulation should be regarded as a proof of concept.

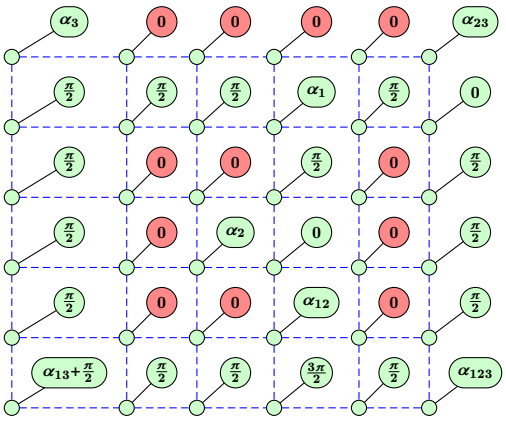


Figure 4.12.: Lattice version of the cluster-state Deutsch-Jozsa algorithm. The pattern in Fig. 4.11 is embedded into this lattice. Spare qubits are removed or bridged with X-measurements or $\pi/2$-XY-measurements.

4.2. Simon algorithm

While the Deutsch-Jozsa algorithm is canonically used as an introduction to quantum superior algorithms, it was not the first one to be proven to be advantageous over classical algorithms. This role is occupied by the algorithm proposed by Paul Simon in 1993 [21]. In this section, we will recapitulate the quantum-circuit formulation of the Simon algorithm and analyze its oracle. Subsequently, we will develop a procedural algorithm to express any possible instance of the oracle of the Simon algorithm as quantum circuit. Then, for the two-qubit algorithm, we will combine these circuits and translate the result to MBQC using the ZX-calculus, just as we did for the Deutsch-Jozsa algorithm in the previous section. Finally, we will derive the form of the graph states for the n-qubit algorithm.

4.2.1. Circuit formulation

Simon's algorithm searches to determine the period of a function. That is, it determines if a function outputs the same value for pairs of inputs that are a defined distance apart. The Simon algorithm is only one instance of the underlying general *hidden subgroup problem* [77, 78]. However, in our work, we will restrict ourselves to the Simon algorithm as the most prominent example.

A function f is said to have a period s if all elements of its domain that are separated by s yield the same image under f. Consider a function $f : \mathbb{S}^n \to \mathbb{S}^n$, where $\mathbb{S} = \{0, 1\}$, which maps from n-bit binary numbers to n-bit binary numbers. Its period s is defined as

$$\exists!\, s \in \mathbb{S}^n : \ \forall\, a, b \in \mathbb{S}^n,\ a \neq b : \ f(a) = f(b) \ \Leftrightarrow \ a = b \oplus s\,, \tag{4.2.1}$$

where $\oplus$ is bit-wise addition modulo 2. From now on, we will use the term *periodic*

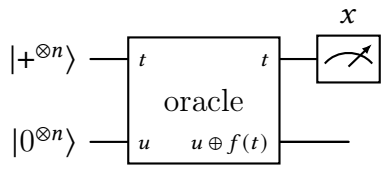


Figure 4.13.: Simon algorithm in the quantum circuit framework.

if a function fulfills the aforementioned definition of periodicity modulo 2 and neglect differing definitions of the word in other contexts. As long as $s \neq 0$, the function is *two-to-one*. In the special case $s = 0$, the function is *bijective*. Given such a function, the aim of the Simon algorithm is to determine the period s of f. We assume that the function is only accessible as a black box (oracle), which can act on a (quantum) input, but that no information about the internal properties of the black box is available.

The Simon algorithm works by applying the function f to an n-qubit quantum state that is a superposition of all possible inputs. The output of f is stored on n auxiliary qubits, and each pair of input basis states that are s apart are "marked" by the same basis state as on the auxiliaries. This scheme generates entanglement between the working and the auxiliary qubits, entanglement that groups together pairs of basis states that are a period s apart. A quantum-interference-based filter is subsequently used to filter out a bit string t that fulfills the linear equation $s \oplus t = 0$. After the procedure is repeated sufficiently many times to obtain enough linearly independent instances of t, a classical solver for linear equation systems can be used to solve for the period s.

We start with n working qubits, where n is equal to the dimension of the function domain and image, each prepared to be in the state $|+\rangle \simeq |0\rangle + |1\rangle$. In addition to the working qubits, we will need n auxiliary qubits, initially prepared to be in the state $|0\rangle$. The full Simon algorithm, including the initial states and the measurement of the final state, is depicted schematically in Fig. 4.13.

We assume that the oracle applies the function f to a register of n working qubits, putting the result in the register of n auxiliary qubits. In doing so, we assume that the oracle takes into account the output value of the function f and maps a general basis state of the full system (working and auxiliary register) $|t\rangle|x\rangle$ as $|t\rangle|x\rangle \rightarrow |t\rangle|x \oplus f(t)\rangle$. Since all qubits of the auxiliary register are prepared in the $|0\rangle$-state, the effect of the oracle is to map $|t\rangle|0^n\rangle \rightarrow |t\rangle|f(t)\rangle$, and we end up with the image under f of the working register on the auxiliary register. Thus, after the application of the oracle, the system is in the entangled state

$$|\psi\rangle = \sum_{t \in \{0,1\}^n} |t\rangle|f(t)\rangle \ . \tag{4.2.2}$$

In effect, basis states $|t\rangle$ and $|t'\rangle$ for which $t' = t \oplus s$ in the working register are "marked" by the same basis state of the auxiliary register, $|f(t)\rangle = |f(t')\rangle$.

The final step of the quantum algorithm as depicted in Fig. 4.13 is the measurement of the qubits of the working register in their respective x-bases; the possible measurement outcomes will lie in $\{+,-\}^n$. However, instead of carrying out the measurements in the x-basis, we could apply an n-qubit Hadamard gate $H^{\oplus n}$ to the working register then carry out the measurements in the computational z-basis. After the Hadamard gate, the complete state, Eq. (4.2.2), transforms to

$$\begin{aligned} |\psi'\rangle &= \sum_{r\in\{0,1\}^n} \sum_{t\in\{0,1\}^n} (-1)^{r\cdot t} |r\rangle |f(t)\rangle \\ &= \sum_{r\in\{0,1\}^n} \sum_{b\in \mathrm{Im}(f)} \sum_{\substack{a \\ f(a)=b}} (-1)^{r\cdot a} |b\rangle \ . \end{aligned} \tag{4.2.3}$$

If $s = 0^n$, f is a *one-to-one* function, and $|\psi'\rangle$ simplifies to

$$|\psi'_{s=0}\rangle = \sum_{r\in\{0,1\}^n} \sum_{t\in\{0,1\}^n} (-1)^{r\cdot t} |r\rangle \ . \tag{4.2.4}$$

If we now perform a measurement on the working qubits—which are in state (4.2.4)—in the computational basis, all measurement outcomes have the same probability. This also holds for the original case in which the n-qubit Hadamard gate is not present, and the measurement is carried out in the x-basis. As $s = 0^n$ in this case, all outcomes m satisfy $s \oplus m = 0$.

For the case of a nonvanishing period $s \neq 0^n$, f is a *two-to-one* function, and Eq. (4.2.3) can be rewritten as

$$|\psi'\rangle = \sum_{r\in\{0,1\}^n} \sum_{b\in \mathrm{Im}(f)} \left[(-1)^{r\cdot a_1} + (-1)^{r\cdot a_2}\right] |r\rangle |b\rangle \ , \tag{4.2.5}$$

where $f(a_1) = f(a_2) = b$. Using that $a_2 = a_1 \oplus s$, we obtain

$$\begin{aligned} |\psi'\rangle &= \sum_{r\in\{0,1\}^n} \sum_{b\in \mathrm{Im}(f)} \left[(-1)^{r\cdot a_1} + (-1)^{r\cdot a_1 \oplus s}\right] |r\rangle |b\rangle \\ &= \sum_{r\in\{0,1\}^n} \sum_{b\in \mathrm{Im}(f)} \left\{(-1)^{r\cdot a_1}\left[1 + (-1)^{r\cdot s}\right]\right\} |r\rangle |b\rangle \ . \end{aligned} \tag{4.2.6}$$

The term in square brackets in the last line of Eq. (4.2.6) will vanish if and only if $r \cdot s = 1$ and will be non-zero only when $r \cdot s = 0$. Thus, measuring the working qubits in the computational basis will generate an outcome m where m satisfies $m \cdot s = 0$. This result also holds for the circuit without the n-qubit Hadamard gate as long as the measurement is carried out in the x-basis.

After a single measurement of the working qubits, we obtain one binary number m that satisfies $m \cdot s = 0$, but not s itself. In order to determine s, the quantum part of the algorithm must be executed repeatedly until $n - 1$ linearly independent outcomes m are

obtained. The resulting set of $n-1$ linear equations can be efficiently solved using a classical algorithm to obtain s, provided that $s \neq 0^n$. If $s \neq 0^n$, the result is indeterminate, as any m will satisfy $m \cdot s = 0$. To make sure that $s \neq 0^n$, we can simply apply f to two different inputs a_1 and a_2 that satisfy $a_2 = a_1 \oplus s$, where s is the solution of the set of linear equations. If $f(a_1) = f(a_2)$, we know that f is a *two-to-one* function for which we have obtained the correct period s. If, on the other hand, $f(a_1) \neq f(a_2)$, the function f must be *one-to-one* with $s = 0^n$.

4.2.2. Oracle design

Until now, we have conceptualized the oracle as a black box that applies a particular instance f of a class of possible oracle functions to an arbitrary input state consisting of n working qubits and n auxiliary qubits. Mathemetically, the operation that is carried out is to add the image under f of the state of the working qubits to the state of the auxiliary qubits as has already been visualized in Fig. 4.13. For practical application, however, the oracle must have a physical implementation that depends on and implements a particular specific instance f. We now consider how to carry out such an implementation in a quantum circuit picture with the aim of subsequently reformulating the implementation in MBQC.

Since the function of the oracle is to add bits (modulo 2) to the auxiliary qubits conditionally depending on the state of the working qubits, it is natural to use CNOT gates to construct the circuit implementing f. A CNOT gate will, for a two-qubit basis state, add a bit (modulo 2) to the state of the second qubit if and only if the first qubit is in the state $|1\rangle$. Mathematically, the gate can be written as

$$\mathrm{CNOT} = |00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 11| + |11\rangle\langle 10| \ . \tag{4.2.7}$$

Since the oracle can include information flow from all working to all auxiliary qubits, a circuit formulation should also be able to include any CNOT gate that has one of the working qubits as its control and one of the auxiliary qubits as its target.

Normally, the CNOT gate is only activated when the first (control) qubit is 1. However, here it is convenient to be able to activate the CNOT using a 0 as the control condition. We can do this by adding an X-gate to the auxiliary qubit. Since the function of an X-gate is to swap each bit in the qubit basis, following a CNOT with an X-gate implements a control gate in which the bit is flipped in the second qubit's basis if and only if the basis state of the first qubit is $|0\rangle$. As there is freedom to displace the X-gates to the right in the quantum circuit, we group all X-gates as the last possible operation of the oracle.

Note that the X-gates described above are actually not required for the algorithm. As is evident in Fig. 4.13, adding single-qubit gates to the auxiliary register does not affect the outcome of the measurement on the working register. Indeed, the essential ingredient for finding the period is the entanglement between working and auxiliary qubits. Nevertheless, a case may be made for retaining the X-gates within our theoretical

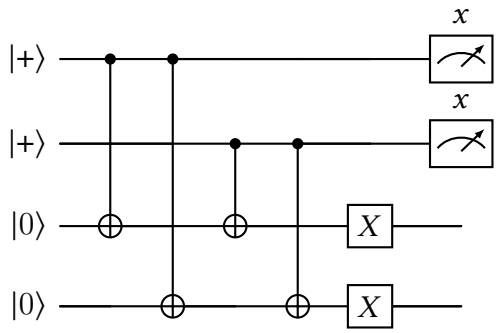


Figure 4.14.: One particular instance of the 2-qubit oracle for Simon's algorithm along with the the initial states and measurement operations. The oracle instance shown in the figure is the one with the maximum number of gates. Other instances of f imply that some of the shown gates are "switched off".

treatment. The primary reason is that we need to check that the s is correct in the classical post-quantum processing of the algorithm (e.g., in order to determine if $s = 0$). For a general oracle, the value of f is available in the auxiliary qubits and we can check any value by sampling. While a reduced oracle without X-gates will still be able to tell us if the images of two samples under f are the same, it will not yield the true value of the image under f. In this paper, we retain the X-gates, implementing the most general version of the oracle.

As mentioned in the previous paragraphs, any function f with any period s can be implemented using a distinct set of CNOT gates, each of which uses a particular working qubit as the control qubit and acts on a particular auxiliary qubit and additional X-gates on appropriately selected auxiliary qubits. A specific set of such gates for one particular (arbitrarily chosen) instance f of the oracle function is depicted in Fig. 4.14. Indeed, Fig. 4.14 contains the maximum number of possible gates for oracles in the Simon algorithm. Other instances of f would imply fewer gates. That is, some of the gates in Fig. 4.14 would be present and some absent. Each unique combination from this maximum set of possible gates realizes one particular instance of f.

4.2.3. Oracle generalization

In the previous subsection, we have shown that the oracle for any n-bit periodic function f can be realized as a network consisting of CNOT gates between n working and n auxiliary qubits and X gates acting on the auxiliary qubits. In order to reformulate the network for such an oracle in MBQC, we must first gain a formal understanding of what type of function f corresponds to what combination of CNOTs and X-gates. In the following, we formulate a deterministic algorithm to construct a quantum network implementing the oracle for a given periodic function f.

A general n-qubit function f has 2^n possible outputs ($|\mathrm{Im}\, f| = 2^n$). Each output $\boldsymbol{\omega_\sigma}$ is

the image of a basis state $\boldsymbol{\sigma}$ of the domain. For example, $\boldsymbol{\sigma} = 010$ might be a basis state and $\boldsymbol{\omega_\sigma} = 110$ its corresponding image, so $f(010) = 110$. We define the *characteristic* C_f of the function f as the output string of all computational domain basis states

$$C_f = \prod_{\boldsymbol{\sigma}\in\mathbb{Z}_2^n} \text{“}\omega_{\boldsymbol{\sigma}}\text{”} = \omega_{0\cdots00}\,\omega_{0\cdots01}\cdots\omega_{1\cdots11}\,, \tag{4.2.8}$$

where the double quotes “...” indicate that the meaning of the product is to concatenate the binary numbers. For n qubits, the characteristic of a function is a string of length $n\,2^n$ bits. For example, the two-to-one function g that is defined by

$$g(00) = 10\,, \quad g(01) = 01\,, \quad g(10) = 01\,, \quad g(11) = 10 \tag{4.2.9}$$

would be represented by the characteristic

$$C_g = \underbrace{10}_{g(00)}\ \underbrace{01}_{g(01)}\ \underbrace{01}_{g(10)}\ \underbrace{10}_{g(11)}\,. \tag{4.2.10}$$

Any classical mapping f with period modulo 2 between input and output qubits can be implemented as a quantum network consisting of CNOT gates acting on particular pairs of input and output qubits, along with X-gates on the output qubits. We will encode a configuration of these gates as a concatenated binary string of length $n2^n$, as we have done for the characteristic of the function, see Eq. (4.2.8). We define the characteristic of a CNOT gate between input qubit i and output qubit j to have ones at the bit positions at which the corresponding basis vector is changed. Mathematically, we define the characteristic of a CNOT gate as

$$\left[C_{\mathrm{CNOT},j,k}\right]_m = \begin{cases} 1\,, & m = (2^n - lj - 1)n + k\,,\ \forall l \in \mathbb{N},\ 0 \le l < n \\ 0\,, & \text{otherwise} \end{cases}\,, \tag{4.2.11}$$

where $(j,k \in \mathbb{N},\ 1 \le j,k \le n)$. In words, we divide the characteristic into 2^n sets of n bits, with each set corresponding to one input basis state. That is, the first n bits are the image of the first basis state, the subsequent n bits are the image of the second basis state, and so on. A CNOT gate will generate nonzero bits only in the n-bit sets that correspond to basis states for which the tested control bit is 1. Each bit of these n-bit sets corresponds to one of the n output bits, so the first bit of the characteristic is the first output bit of the first input basis state. Within such a set, the bit corresponding to the target qubit is 1, and all others are 0.

As an example, we consider a system with two domain and two image bits. This directly implies that the characteristic of both the function and the gates have a length of $2 \cdot 2^2 = 8$ bits. Furthermore, we take the CNOT gate to act between qubit 1 of the domain and qubit 2 of the image, as depicted in the circuit of Fig. 4.15. The characteristic of

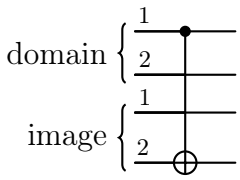


Figure 4.15.: Circuit for the CNOT gate with control bit 1 and target bit 2 for a 2-bit system.

this gate, according to Eq. (4.2.11), is

$$\left[C_{\mathrm{CNOT},1,2}\right]_m = \begin{cases} 1\,, & m \in \{5,7\} \\ 0\,, & \text{otherwise} \end{cases} \quad \Rightarrow \quad C_{\mathrm{CNOT},1,2} = 00000101\,. \tag{4.2.12}$$

This characteristic has the following content: The control qubit is the first qubit. Hence, the third and fourth set of $n = 2$ bits, corresponding to the basis vectors 10 and 11, are not all zero. Within these sets, the second bit is set to 1 because the target qubit of the CNOT gate is the second of the image qubits.

Analogously to the CNOT gate of Eq. (4.2.11), we can characterize the X-gate on image qubit k as

$$\left[C_{X,k}\right]_m = \begin{cases} 1\,, & m = ln + k\,,\ \forall l \in \mathbb{N},\ 0 \le l < 2^n \\ 0\,, & \text{otherwise.} \end{cases} \tag{4.2.13}$$

Put in words, this means that the k-th bit of each set of n bits is set to 1, corresponding to the k-th image qubit for each domain qubit. For example, an X-gate on the second of two image bits would read $C_{X,2} = 01010101$.

4.2.4. Factorization

We want to show that an oracle composed of only CNOT- and X-gates can indeed realize any instance f of a periodic (modulo 2) bitwise function. For that, we outline a deterministic procedure to implement the oracle for every possible function f. To do this, we will add the characteristics of the possible gates to the characteristic of f until the resulting string no longer contains nonzero bits. The steps required will tell us what combination of gates is needed to realize f.

The function f is represented by its characteristic C_f. We denote the set of gates that can realize the oracle for any periodic function by $S = \{\mathrm{CNOT}_{jk}, X_k\}$, where $1 \le j, k \le n$. As an example, the set S for two qubits is displayed in Table 4.4. We need to determine which subset of S will implement a particular realization of f. This is a factorization of C_f within the set of characteristics $C[S] = \{C(s)\,;\ s \in S\}$ of the set of gates S.

This factorization can be carried out in a deterministic, algorithmic way. Notice, for

gate	CNOT_{11}	CNOT_{12}	CNOT_{21}	CNOT_{22}	X_1	X_2
char.	00001010	00000101	00100010	00010001	10101010	01010101
angle	α	β	γ	δ	ϕ	η

Table 4.4.: The set $S = \{\mathrm{CNOT}_{jk}, X_k\}$ $(1 \leq j,k \leq n)$, along its characterizations, for two qubits.

example, that in Table 4.4, the first nonzero bit of each element CNOT_{jk} is in a different position. The only operation that will change the first nonzero bit is an X-gate acting on the first image qubit. Hence, adding the characteristic of said gate to the characteristic C_f with a leading 1 (modulo 2) will flip the first bit to zero. Note, however, that it will also change other bits. Thus, we need to step through the bits of C_f from left to right. Each time a nonzero bit appears, we add the characteristic of the gate with corresponding first nonzero character. After the application of, at most, $n2^{n-1}$ gates, the bit string will consist of only zeros. The sequence of gates required to do this uniquely determines the circuit that realizes our chosen f.

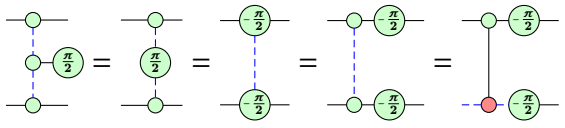
Example: Consider the two-qubit function f with $f(00), f(01), f(10), f(11) = 10, 11, 11, 10$. As is easily seen, f has a period $s = 11$ and its characteristic is $C_f = 10111110$. The characteristics of the possible gates can be read off from Table 4.4. Following the procedure outlined above, we obtain

$$10111110 \xrightarrow{X_1} 00010100 \xrightarrow{\mathrm{CNOT}_{22}} 00000101 \xrightarrow{\mathrm{CNOT}_{12}} 00000000\,. \tag{4.2.14}$$

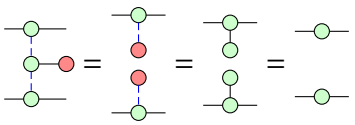
Thus, the oracle for this particular realization of the two-bit periodic function f is implemented by the gate sequence CNOT_{12}, CNOT_{22}, X_1.

4.2.5. Translation to ZX-calculus

Having derived a formal algorithm to describe any of the oracle instances as circuits, we can now start to translate the oracles to MBQC using the ZX-calculus. Since a general implementation of the oracle requires the ability to switch either CNOT or X-gates on or off, we utilize an adaptive description that can control whether or not these gates appear in the oracle. In particular, we use the adaptive version of a CNOT gate,



, (4.2.15)



. (4.2.16)

These identities can be straightforwardly derived from Eq. (3.1.9) and Fig. 3.8. Eq. (4.2.15) tells us that the CNOT gate introduced in Eq. (4.2.11) is equivalent to a ZX-diagram with a phase-zero Z-node that has both a terminal $\pi/2$-Z-node attached to it as well as two further Z-nodes, that are connected to it via Hadamard edges (up to phase corrections). These required correcting phases take on the form of two $\pi/2$-Z-nodes and two Hadamard nodes and need to be included when this ZX-network fragment is embedded in a larger ZX-diagram. Note that the protruding $\pi/2$-node in the leftmost part of Eq. (4.2.15) represents the bra $\langle 0| + \mathrm{i}\langle 1|$ and thus acts as a measurement of the unit-eigenvalue eigenstate of the Y-Pauli matrix.

Eq. (4.2.16) shows that inserting a zero-phase X-node rather than a $\pi/2$-Z-node into Eq. (4.2.15) disconnects the two Z-nodes on the top and on the bottom of the diagram. Thus, one can control whether or not a CNOT gate between two nodes is present by choosing whether there is a $\pi/2$-Z-node or a phase-less X-node.

In Sec. 3.3.1, we introduced the prerequisites for characterizing MBQC with the ZX-calculus and described several identities that are required to translate the Simon algorithm to a measurement-based form. Here we will first develop a procedure for the two-qubit algorithm and then explain how it can be extended to the n-qubit case.

Fig. 4.14 depicts a schematic circuit for the two-qubit Simon algorithm that includes all gates that could occur in a specific realization of the oracle: CNOT-gates are present between all pairs of input and auxiliary qubits and X-gates on all auxiliary qubits. Some (or all) of these gates may not be present in a circuit realizing a particular instance of the oracle. Thus, an arbitrary oracle can be constructed by switching on or off particular gates. A general circuit for which each gate can be switched on or switched off can be realized in the ZX-calculus using the adaptive description of the CNOT gate, Eqs. (4.2.15) and (4.2.16), along with the rules in Eqs. (3.1.6)-(3.1.12).

A raw translation of the two-qubit circuit is depicted in Fig. 4.16. Note that we have identified the initial states of the qubits as $|+\rangle =$ and $|0\rangle =$  . The adaptive CNOT gates are controlled by a node that we depict as

$$\alpha = \begin{cases} \quad , & \alpha = 0 \\ \quad , & \alpha = \frac{\pi}{2} \end{cases} . \tag{4.2.17}$$

This node is adaptive in the sense that it is chosen to be either an X-measurement or a Y-measurment depending on the particular realization of the function that the oracle represents. Which of the two permitted distinct phases, $\alpha = 0$ or $\alpha = \pi/2$, is chosen determines the color of the node. As expressed in Eq. (4.2.15), the adaptive CNOT gate also requires some extra correcting phases to be applied to the two qubits adjacent to the one being measured. Those are included into the raw translation by

$$\alpha_c = \begin{cases} 0 & \alpha = 0 \\ \frac{\pi}{2} & \alpha = \frac{\pi}{2} \end{cases} , \tag{4.2.18}$$

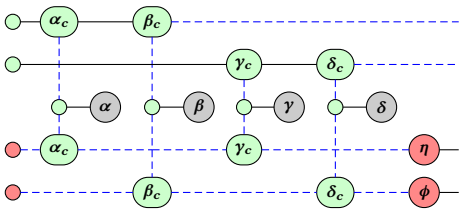


Figure 4.16.: Raw translation of the Simon algorithm into a ZX-diagram.

where α is the same value as in Eq. (4.2.17). The same scheme applies to all adaptive angles. The nodes in Fig. 4.16 with angles η and ϕ represent the X-gates; the phase of η or ϕ is π if an X-gate is present and 0 if not.

We apply the color-swap rule (3.2.4) and the contraction rule (3.1.6) to obtain an initial, simplified version of the oracle, which is depicted in Fig. 4.17.

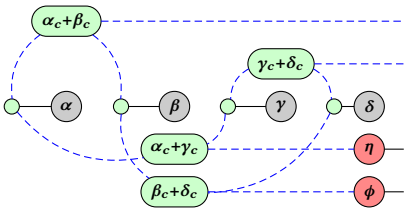


Figure 4.17.: Oracle of the Simon algorithm after applying the color-swap and the contraction rule to the raw translation of Fig. 4.16.

First, notice that the two protruding edges on the bottom of Fig. 4.16 represent the auxiliary qubits. Since we no longer utilize the information encoded in the auxiliary qubit after applying the oracle, we can omit the protruding legs to the right of the red nodes. More formally, we add a projective measurement —● in the x-basis to annihilate the protruding legs. Using the contraction rule (3.1.6) again, we can contract the newly added nodes into the neighboring Z-nodes. On the two top-most protruding legs, we then apply the identity —○— so that they become regular, rather than Hadamard, legs. We thereby obtain the diagram given in Fig. 4.18.

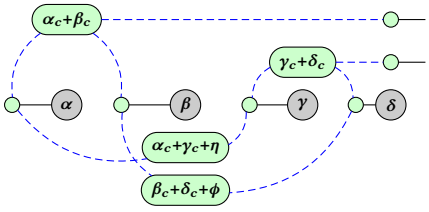


Figure 4.18.: Intermediate result of the simplification process of the Simon oracle.

A straighforward reordering of the graph yields the final simplified diagram, Fig. 4.19, which implements the two-qubit Simon algorithm in the framework of MBQC. As is

required for MBQC, it can be seen that all qubits of the many-body-system are Z-nodes. Measurements on these qubits are always carried out in the y-z-plane, at an angle that depends on the corrective factors. The four remaining qubits are to be measured in the adaptive manner described above. They are depicted as grey nodes, which either represent a $\pi/2$-z-measurement or a zero-phase x-measurement.

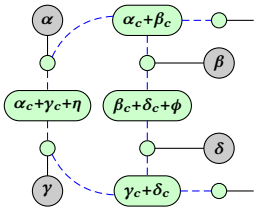


Figure 4.19.: Measurement-based form of the two-qubit Simon algorithm.

To clarify the meaning of the MBQC protocol in terms of the ZX-calculus, we give here a short explanation of how to interpret Fig. 4.19. First, a cluster state, consisting of qubits ordered in a graph as the green nodes connected by blue and dashed lines, is created. Then, all qubits of the cluster state, except for the two green nodes to the right, are measured according to the angles depicted in the figure. If outcomes other than 0 occur during a measurement, corrective factors must be propagated through the graph; subsequent measurement angles and/or the final result might have to be adapted. Eventually, the two qubits on the right are measured in the x-basis. For an oracle representing a periodic function, the bit string m that is measured satisfies the condition $m \cdot s = 0$, where s is the period. Finally, it is necessary to repeat the computation sufficiently often to obtain n linearly independent equations for s (where $n = 2$ here), and these equations must be solved classically, just as in the quantum-circuit formulation of the algorithm.

4.2.6. n-qubit Simon algorithm

In Sec. 4.2.5, we derived the MBQC formulation of the two-qubit Simon algorithm. Here we will expand this formulation to treat n qubits. The translation scheme and the following measurement procedure is essentially the same as in the two-qubit case. In the following, we will construct the graphs of the cluster states that are required to formulate the n-qubit Simon algorithm in terms of MBQC.

Above, we have learned that a general instance of the oracle of the Simon algorithm can be constructed using a network consisting of CNOT gates between arbitrary unique pairs of working and auxiliary qubits. In total, there are n^2 possible CNOT gates, each of which can be realized by an adaptive CNOT gate in the ZX-language. In the ZX-scheme, each pair of working and auxiliary qubits can be pulled together to form one node in the ZX-diagram, so that we obtain n^2 adaptive CNOT gates connecting these nodes. In the

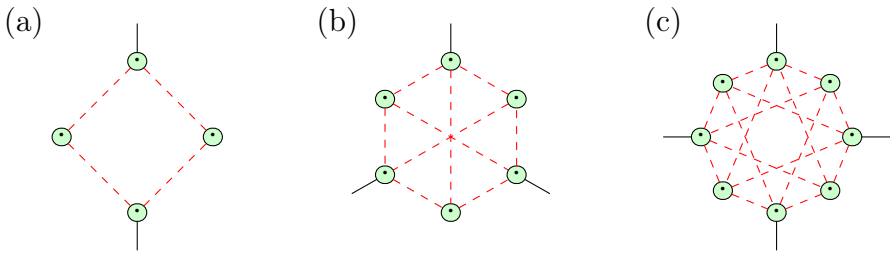
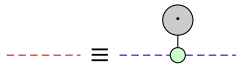


Figure 4.20.: Topology of the cluster state required to perform the a) two-, b) three- and c) four-qubit Simon-algorithm in MBQC.

two-qubit case, depicted in Fig. 4.19, we have four qubits, connected by $2^2 = 4$ adaptive CNOT gates. Each node is connected to exactly n other gates.

In order to better display the topology of the graphs of n-qubit, measurement-based oracles, we introduce a compact notation for an adaptive CNOT gate. We define red and dashed lines as



(4.2.19)

where the dot in the grey node represents the measurement defined in Eq. (4.2.17) in which the choice of measurement angle determines the presence or absence of a CNOT gate. Since we do not specify the measurement angle explicitly in this notation, we also omit the corrective angles by writing

(4.2.20)

where the corrective angles to be applied to the qubits neighboring the measured node are specified by Eq. (4.2.18).

The form of the graph for the cluster state of the n-qubit version of the Simon algorithm is determined by the number n of working and auxiliary qubits as well as their connections through adaptive CNOT gates. For visualization, we arrange all qubits on a ring, alternating between auxiliary and working qubits. We then connect each working qubit with all auxiliary qubits via red and dashed lines, which represent adaptive CNOT gates. The working qubits are distinguished by protruding legs that are used for the final measurement. Thus, the n-qubit algorithm is represented by a ring-like graph consisting of $2n$ nodes, n^2 edges, and n protruding legs.

As an example, recall the pattern of the oracle for the two-qubit algorithm, Fig. 4.19. We place the four qubits (2 working and 2 auxiliary) on a ring, forming a square. Connecting working to auxiliary qubits and introducing protruding legs leads to the schematic diagram depicted in Fig. 4.20a). The scheme is easily extended to three qubits, Fig. 4.20b), and four qubits, Fig. 4.20c), as well as to an arbitrary number of qubits by continuing the scheme straightforwardly.

5. Summary and Conclusion

We have translated the oracles of both the Deutsch-Jozsa and the Simon algorithm to the language of MBQC, reformulating the quantum-circuit descriptions as ZX-diagrams and then bringing the ZX-diagrams into a form that represents a measurement-based algorithm on a cluster state. The results are patterns that describe both the topology of the underlying cluster state as well as a sequence of single-qubit projective measurements that must be carried out in order to implement a particular realization of the oracle.

The three-qubit Deutsch-Joza algorithm is implemented on an 11-qubit cluster state on which single-qubit projective measurements suffice to determine if the underlying function is constant or balanced. In addition, we have derived an alternate, but equivalent, ZX-diagram that has a rectangular form that could potentially be useful for experimental implementations.

In reformulating the Simon algorithm, an essential intermediate step is the construction of a deterministic algorithm to represent any given periodic function as a quantum circuit consisting only of CNOT and X gates. Utilizing this algorithm, we have reformulated the general two-qubit oracle of the Simon algorithm in the language of MBQC explicitly. For the n-qubit version of the algorithm, we have introduced a compact notation for the adaptive measurement nodes and have depicted the cluster states for the three- and four-qubit algorithm as examples. We have found that the complexity of the cluster state increases with the number of qubits in the sense of an increasing number of cross-sections when drawing the graph in two dimensions.

5.1. Deutsch-Jozsa algorithm

For the two-qubit Deutsch-Josza algorithm, we have found a comprehensible illustration for the hypothesis of Ref. [73] that the origin of the quantum supremacy in the Deutsch-Jozsa algorithm is the ability of the quantum oracle to operate on a superposition state, i.e., to utilize quantum parallelism, rather than in any use of entanglement. This illustrates that the MBQC formulation of an algorithm, coupled with its representation in terms of ZX-calculus, can elucidate the underlying structure of a quantum computation and clarify the extent and origin of the quantum advantage. As stated in Sec. 3.5, there are other routes to such a clarification, for example, via the topological description of quantum algorithms [68]. Both the topological description and the ZX-calculus itself can be viewed as being part of the more fundamental and general framework of categorical quantum mechanics. For further information, we refer the reader who is interested in

most fundamental treatment of quantum mechanics to Refs. [79] and [80]. An MBQC implementation of the two-qubit version of the Deutsch-Josza algorithm has already been formulated in Ref. [72]. However, the authors did not include a derivation of their MBQC formulation.

We remark that the complexity of the two-qubit case is low enough so that educated guessing is sufficient to find a working and presumably optimal MBQC implementation; in our opinion, this no longer holds for the three-qubit case. Ref. [72] also includes a generalization to n qubits that is based on the assumption that the two- and more-qubit versions of the algorithm have the same structural complexity. However, it was shown in Ref. [74] that this is not the case, and our work, based on a different, graphical approach, also contradicts this simplified generalization. The eleven-qubit cluster state that we have provided shows significantly more interconnectivity than the star-like structure proposed in Ref. [72].

For experimental motivation, we have introduced a version of the three-qubit Deutsch-Jozsa algorithm that is based on a rectangular lattice cluster state. The state is derived by applying the transformation rules of the ZX-calculus in the opposite direction compared to the translations of the oracles to MBQC. By carefully adding (formally redundant) nodes, we have obtained a 6×6-qubit rectangular lattice cluster state. The original 11-qubit cluster state can be extracted from this lattice by a set of decoupling z-measurements on a subset of the qubits.

5.2. Simon algorithm

We have constructed an MBQC-suited ZX-network for the oracle of the two-qubit Simon algorithm explicitly; the resulting cluster state consists of eight qubits ordered in the shape of a square. Furthermore, we have described how to construct cluster states for the general n-qubit algorithm and have given explicit forms for such states in the three- and four-qubit cases.

The deterministic algorithm to transform any periodic function into a circuit representation using CNOT and X gates that we have developed in Secs. 4.2.3-4.2.4 is an essential milestone on the road to reformulating the Simon algorithm in MBQC. We know of no such constructive quantum-circuit-based algorithm in the literature. Algorithms to construct an oracle for the Simon algorithm, such as that formulated in Ref. [81], do exist. However, in our understanding, they are all based on the assumption that a specific fixed period s is known in advance. In our opinion, assuming a given period in order to construct an oracle is somewhat circular because the whole purpose of the calculation is to determine said unknown period. In our algorithm, such an assumption would strongly limit the allowed inputs and outputs of the function. It can be argued that such a restriction also, essentially, contains knowledge of the period. Finding the period of a function with no prior knowledge is, in our opinion, the core of the motivation behind the Simon algorithm. We remark that guessing the period and subsequently

constructing the algorithm around it will become exponentially difficult as the function becomes larger. We appreciate that the question of how to design oracles for oracular quantum algorithms and, in particular, the issue of how much presupposed knowledge about the function or oracle is available, is still a subject of discussion and, at this point in time, is a somewhat philosophical issue.

Specific quantum circuits for the oracle of the two-qubit Simon algorithm have been published in Ref. [82], albeit without specifying a rigorous algorithm to construct them. We remark that, just as for the two-qubit Deutsch-Jozsa algorithm, the two-qubit case is of sufficiently low complexity that a working, and presumably optimal, MBQC implementation can be found relatively easily by educated guessing; in our opinion, this is no longer possible for systems for more qubits. Our approach of deterministically constructing implementations of a general oracle for a given algorithm is in line with other published work such as Ref. [75], in which arbitrary oracle circuits are constructed for the four-qubit Deutsch-Jozsa algorithm.

The topology of the cluster states that we have constructed to implement a general n-qubit Simon algorithm gives information about the complexity of the algorithm. We have shown that our graph for the general algorithm consists of $2n$ nodes connected by n^2 edges. Each of these generalized edges consists of one measurement node and two basic cluster-state edges. Thus, as expected, the quantum resources required for the algorithm grow only polynomially with the size of the problem. On the other hand, only the two-qubit Simon algorithm can be realized as a planar (two-dimensional) cluster state without cross-sections. For all larger variants, the graphs do have two-dimensional cross sections and thus require a higher number of dimensions for their implementation.

This topological complexity is especially relevant for experimental implementation. A simplified version of the two-bit Simon algorithm has already been experimentally implemented by the authors of Ref. [82]. For larger systems, it will be a challenge to implement the cluster states that we have derived. Large quantum states on arbitrary graphs are especially difficult to produce and control, both in solid-state and in photonic systems. We have shown that the number of two-dimensional cross sections increases as the number of bits in the oracle function increase. The increased interconnectivity between qubits will also make the creation of the associated cluster states less tractable.

5.3. Conclusion

We hope that the measurement-based formulations of two prominent quantum algorithms that we have provided in this thesis can serve as a blueprint for experimental implementation. We also hope that the work in this thesis inspires researchers to formulate other algorithms in the language of MBQC. While the translation from quantum circuit to ZX-representation is well-known, and the conversion to MBQC form is relatively straightforward, we know of no explicit reformulations of quantum algorithms using this method in the previous literature. In any case, subsequent simplification of the

ZX-diagram is less straightforward, at least if an optimal pattern that minimizes the size of the cluster state is the goal.

In this work, we have shown that the MBQC graphs for the oracles are equivalent to their counterparts in the framework of circuit quantum computing. However, we have neither argued nor proven that these are the cluster states that require the smallest number of qubits and/or the smallest interconnectivity. This raises a question of optimal minimization that cannot be answered through the framework of the ZX-calculus alone because the latter is only a descriptive language. Nevertheless, the rules of ZX-calculus are a good tool to visualize a path to a minimal solution, as is discussed in Ref. [12]. Thus, interesting subjects for future work would be to investigate if the patterns derived here do, in fact, yield a minimal cluster state and, in addition, to formulate schemes for finding such patterns in general and for proving that they are, in fact, minimal.

While we have specifically treated the Deutsch-Jozsa and Simon algorithms here, the methods we have outlined are relatively general and can be applied to other quantum circuits, i.e., to other quantum algorithms. Thus, our treatment here is intended to illustratrate the scheme using basic, but still nontrivial, instances of algorithms. An obvious extension would be to translate the Deutsch-Jozsa algorithms with more than three qubits; a general scheme for the n-qubit case would be of particular interest.

Historically, Simon's formulation of the Simon algorithm led to Shor's development of his famous prime-factorization algorithm [17, 21, 24]. This chronology can be taken as inspiration for the development of the MBQC-variant of the Shor algorithm. Since the underlying structure of the hidden subgroup problem in the Simon algorithm is also an integral part of the Shor algorithm, it seems only natural to apply the principles of this work to the Shor algorithm. The methods developed here might additionally be useful for reformulating other algorithms, information protocols, error correction codes, etc. in a MBQC picture.

We hope that our work on algorithms in the framework of MBQC can increase understanding and motivate future research in fields besides the circuit model. An ongoing discussion about alternative pictures can broaden diversity in the interpretation of quantum computing and potentially help in the development of larger experimental devices. After all, it must be the aim to implement quantum devices large enough to solve problems faster than any classical computer ever could. Then, our everyday life might be revolutionized again.

Appendices

A. Personal notes and acknowledgements

First and foremost, I would like to sincerely thank my supervisor Prof. Dr. R. Noack for guiding me through my time as a PhD student. I look back to many fruitful discussions and advice that has helped me grow as a theoretical physicist and reach the milestone of my first publications. I want to highlight the extensive work he put into editorial corrections on my texts that immensely helped me elevate my English scientific writing skills to the level that can be witnessed in this thesis. Prof. Noack also chose to leave me with full freedom in the choice of topics that I liked to work on. I always experienced a very respectful and a calm working atmosphere with no more pressure than the one I put on myself. I also thank him for the friendly relationship and pleasant conversations besides physics throughout the almost five years that I was part of the work group. I encourage everyone to apply for a thesis in the work group *Vielteilchenphysik* of Prof. Noack at the *University of Marburg*.

As an orientation for future students, I would like to give a few insights to my working hours. It is not meant as advice or ideal in any context. Respecting the working conditions for many PhD students in Germany at current time, I only want to give my own scenario as one out of many possible scenarios. During my time as PhD student, I was employed on public service funding for 75% of a full-time job. The salary amounts to roughly 2000 Euros per month after taxes. Formally, the 75% corresponds to 30 hours of work per week. However, in university in Germany, this has typically no significance because much more work is either expected or done voluntarily by students. I have logged my work hours on a daily basis. In the first one and a half years, I have decided to keep the workload at around 40 hours per week. After being a little bit uncontent with my own productivity at the end of that time, I aligned my working hours more to the level of 30 hours per week. This reduction did not reduce my overall productivity at all. It has to be stated that besides preparing and holding exercise classes for students, my supervisor left me with no additional work other than my own research. During the full 4 years and 10 months of my employment, I have amassed a total overtime of about 135 working days compared to a 30 hour working week. I explicitly state here that any overtime was done voluntarily, without any pressure, and without compensation. Compared to a 40 hour working week, this amasses to a deficit of about 75 hours. I also want to state that I have always taken my full holiday leave that amounts to 30 days per year in public service in Germany. In my personal opinion, success and productivity cannot directly be measured in working hours. I suggest that quality of work is measured by the output,

not by the input. Being motivated to put in more work should, of course, be encouraged and rewarded, but only to the point that general efficiency or sustainability over the working life is not affected. The ratio of efficiency to recovery time is, naturally, very different between individual human beings and this fact should, in my opinion, always be kept in mind when is feeling the need to judge someones work.

Finally, I would like to wish good luck to any plagiarism hunters in the future. Naturally, I have listed all literature that I have used during the creation of this thesis. Citation is done by the standards that hold in the field of physics at the time of submission. If standards change in the next decades, I am looking forward to discuss both citation and the content of this work again, under the background of future standards. I am happy that both society and scientific society has come a long way and that it is still evolving and I am definitely willing to judge my own work again when consensus changes.

B. Wissenschaftlicher Werdegang / academic resumé (German)

Bachelor-Studium Eidgenössische Technische Hochschule (ETH) Zürich, 2011-2015, abgeschlossen mit dem Titel Physics ETH-Bachelor of Science, Gesamtnote 4,55 (entspricht 1,45 nach deutscher Benotung)

Master-Studium Universität des Saarlandes, Saarbrücken, 2015-2018, und Philipps-Universität Marburg, 2018 - 2020, abgeschlossen mit der Gesamtnote 1,8

Promotion Promotionsstudent & wissenschaftlicher Mitarbeiter, 2020-2024

Veröffentlichungen [1] (veröffentlicht) und [2] (im Prozess der Veröffentlichung, eingereicht, veröffentlicht auf dem Preprint-Server)

References

[1] M. Schwetz and R. M. Noack, "Three-qubit Deutsch–Jozsa in measurement-based quantum computing", International Journal of Quantum Information, 2350046 (2024).

[2] M. Schwetz and R. M. Noack, *Simon algorithm in measurement-based quantum computing*, 2024.

[3] T.-C. Wei, "Measurement-Based Quantum Computation", Mar. 2021.

[4] H. J. Briegel and R. Raussendorf, "Persistent Entanglement in Arrays of Interacting Particles", Phys. Rev. Lett. **86**, 910–913 (2001).

[5] T.-C. Wei, I. Affleck, and R. Raussendorf, "Two-dimensional Affleck-Kennedy-Lieb-Tasaki state on the honeycomb lattice is a universal resource for quantum computation", Phys. Rev. A **86**, 032328 (2012).

[6] A. Miyake, "Quantum computational capability of a 2D valence bond solid phase", Annals of Physics **326**, July 2011 Special Issue, 1656–1671 (2011).

[7] R. Raussendorf and H. J. Briegel, "A One-Way Quantum Computer", Phys. Rev. Lett. **86**, 5188–5191 (2001).

[8] M. A. Nielsen, "Cluster-state quantum computation", Reports on Mathematical Physics **57**, 147–161 (2006).

[9] V. Danos, E. Kashefi, and P. Panangaden, "The Measurement Calculus", Journal of the Association for Computing Machinery **54**, 8–es (2007).

[10] L. Voufo, "Quantum Circuits: From a Network to a One-Way Model", Electronic Notes in Theoretical Computer Science **270**, Proceedings of the Joint 5th International Workshop on Quantum Physics and Logic and 4th Workshop on Developments in Computational Models (QPL/DCM 2008), 191–210 (2011).

[11] B. Coecke and R. Duncan, "Interacting quantum observables: categorical algebra and diagrammatics", New Journal of Physics **13**, 043016 (2011).

[12] R. Duncan, A. Kissinger, S. Perdrix, and J. van de Wetering, "Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus", Quantum **4**, 279 (2020).

[13] P. Benioff, "The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines", Journal of Statistical Physics **22**, 563–591 (1980).

[14] M. Schwetz, "Framework for Universal NMR Quantum Computing Using Heisenberg Spin Interaction", PhD thesis (Philipps-Universität Marburg, 2020).

[15] R. P. Feynman, "Simulating physics with computers", International Journal of Theoretical Physics **21**, 467–488 (1982).

[16] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, 2010).

[17] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", SIAM Journal on Computing **26**, 1484–1509 (1997).

[18] D. Deutsch and R. Penrose, "Quantum theory, the Church-Turing principle and the universal quantum computer", Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences **400**, 97–117 (1985).

[19] D. Deutsch and R. Jozsa, "Rapid solution of problems by quantum computation", Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences **439**, 553–558 (1992).

[20] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca, "Quantum algorithms revisited", Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences **454**, 339–354 (1998).

[21] D. Simon, "On the power of quantum computation", in Proceedings 35th Annual Symposium on Foundations of Computer Science (1994), pp. 116–123.

[22] G. Alagic, C. Moore, and A. Russell, "Quantum algorithms for Simon's problem over general groups", in ACM-SIAM Symposium on Discrete Algorithms (2006).

[23] G. Alagic, C. Moore, and A. Russell, "Quantum Algorithms for Simon's Problem over Nonabelian Groups", ACM Trans. Algorithms **6**, `10.1145/1644015.1644034` (2010).

[24] G. Salton, D. Simon, and C. Lin, *Exploring Simon's Algorithm with Daniel Simon.*

[25] R. P. Feynman, "Quantum mechanical computers", Foundations of Physics **16**, 507–531 (1986).

[26] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse Ising model", Phys. Rev. E **58**, 5355–5363 (1998).

[27] P. Ray, B. K. Chakrabarti, and A. Chakrabarti, "Sherrington-Kirkpatrick model in a transverse field: Absence of replica symmetry breaking due to quantum fluctuations", Phys. Rev. B **39**, 11828–11832 (1989).

[28] D. de Falco, B. Apolloni, and N. Cesa-Bianchi, "A numerical implementation of quantum annealing", in (July 1988).

[29] B. Apolloni, C. Carvalho, and D. de Falco, "Quantum stochastic optimization", Stochastic Processes and their Applications **33**, 233–244 (1989).

[30] A. Finnila, M. Gomez, C. Sebenik, C. Stenson, and J. Doll, "Quantum annealing: A new method for minimizing multidimensional functions", Chemical Physics Letters **219**, 343–348 (1994).

[31] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, *Quantum Computation by Adiabatic Evolution*, 2000.

[32] A. M. Childs, R. Cleve, E. Deotto, E. Farhi, S. Gutmann, and D. A. Spielman, "Exponential algorithmic speedup by a quantum walk", in Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing, STOC '03 (2003), pp. 59–68.

[33] A. M. Childs, L. J. Schulman, and U. V. Vazirani, "Quantum Algorithms for Hidden Nonlinear Structures", in 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07) (2007), pp. 395–404.

[34] V. Lahtinen and J. K. Pachos, "A Short Introduction to Topological Quantum Computation", SciPost Phys. **3**, 021 (2017).

[35] M. Hein, W. Dür, J. Eisert, R. Raussendorf, M. Nest, and H. Briegel, "Entanglement in Graph States and its Applications", **162**, `10.3254/1-58603-660-2-115` (2006).

[36] P. Walther, K. J. Resch, T. Rudolph, E. Schenck, H. Weinfurter, V. Vedral, M. Aspelmeyer, and A. Zeilinger, "Experimental one-way quantum computing", Nature **434**, 169–176 (2005).

[37] S. Abramsky and B. Coecke, "A categorical semantics of quantum protocols", in Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, 2004. (2004), pp. 415–425.

[38] S. ABRAMSKY and R. DUNCAN, "A categorical quantum logic", Mathematical Structures in Computer Science **16**, 469–489 (2006).

[39] S. Abramsky and B. Coecke, "Categorical Quantum Mechanics", in *Handbook of Quantum Logic and Quantum Structures*, edited by K. Engesser, D. M. Gabbay, and D. Lehmann (Elsevier, Amsterdam, 2009), pp. 261–323.

[40] M. Backens and A. Kissinger, "ZH: A Complete Graphical Calculus for Quantum Computations Involving Classical Non-linearity", Electronic Proceedings in Theoretical Computer Science **287**, 23–42 (2019).

[41] T. Carette, D. Horsman, and S. Perdrix, "SZX-Calculus: Scalable Graphical Quantum Reasoning", in 44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019), Vol. 138, edited by P. Rossmanith, P. Heggernes, and J.-P. Katoen, Leibniz International Proceedings in Informatics (LIPIcs) (2019), 55:1–55:15.

[42] M. Mano and C. Kime, *Logic and Computer Design Fundamentals Pearson New International Edition* (Pearson Deutschland, 2013).

[43] N. Bourbaki, *Topological Vector Spaces* (Springer Berlin Heidelberg, 2003).

[44] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell, et al., "Quantum supremacy using a programmable superconducting processor", Nature **574**, 505–510 (2019).

[45] P. Kok, W. J. Munro, K. Nemoto, T. C. Ralph, J. P. Dowling, and G. J. Milburn, "Linear optical quantum computing with photonic qubits", Rev. Mod. Phys. **79**, 135–174 (2007).

[46] G. Nebe, E. M. Rains, and N. J. A. Sloane, *The invariants of the Clifford groups*, 2000.

[47] A. Kay, "Quantikz", 2019.

[48] X. Zhou, D. W. Leung, and I. L. Chuang, "Methodology for quantum logic gate construction", Physical Review A **62**, 10.1103/physreva.62.052316 (2000).

[49] M. Van den Nest, A. Miyake, W. Dür, and H. J. Briegel, "Universal Resources for Measurement-Based Quantum Computation", Phys. Rev. Lett. **97**, 150504 (2006).

[50] M. Van den Nest, W. Dür, G. Vidal, and H. J. Briegel, "Classical simulation versus universality in measurement-based quantum computation", Phys. Rev. A **75**, 012337 (2007).

[51] D. Gross, S. T. Flammia, and J. Eisert, "Most Quantum States Are Too Entangled To Be Useful As Computational Resources", Phys. Rev. Lett. **102**, 190501 (2009).

[52] I. B. Djordjevic, *Quantum Information Processing, Quantum Computing, and Quantum Error Correction* (Academic Press, 2021).

[53] D. Gottesman, "The Heisenberg Representation of Quantum Computers", 10.48550/ARXIV.QUANT-PH/9807006 (1998).

[54] L. Euler, "Formulae generales pro translatione quacunque corporum rigidorum", Novi Commentarii academiae scientiarum Petropolitanae **20**, 189–207 (1776).

[55] V. Danos, E. Kashefi, and P. Panangaden, "Parsimonious and robust realizations of unitary maps in the one-way model", Phys. Rev. A **72**, 064301 (2005).

[56] P. Jorrand and S. Perdrix, "Unifying quantum computation with projective measurements only and one-way quantum computation", in Quantum Informatics 2004, Vol. 5833, edited by Y. I. Ozhigov (International Society for Optics and Photonics, 2005), pp. 44 –51.

[57] B. Coecke and A. Kissinger, *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning* (Cambridge University Press, 2017).

[58] L. Colisson, *zx-calculus – A library to typeset ZX Calculus diagrams*, [available on CTAN].

[59] M. Backens, H. Miller-Bakewell, G. de Felice, L. Lobski, and J. van de Wetering, "There and back again: A circuit extraction tale", Quantum **5**, 421 (2021).

[60] E. Jeandel, S. Perdrix, and R. Vilmart, "A Complete Axiomatisation of the ZX-Calculus for Clifford+T Quantum Mechanics", in Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '18 (2018), pp. 559–568.

[61] M. Mhalla and S. Perdrix, "Graph States, Pivot Minor, and Universality of (X, Z)-Measurements", International Journal of Unconventional Computing **9** (2012).

[62] A. Kotzig, "Eulerian lines in finite 4-valent graphs and their transformations", in Colloquium on Graph Theory Tihany 1966 (1968), pp. 219–230.

[63] R. Duncan and S. Perdrix, "Graph States and the Necessity of Euler Decomposition", in Mathematical Theory and Computational Practice, edited by K. Ambos-Spies, B. Löwe, and W. Merkle (2009), pp. 167–177.

[64] R. Duncan and S. Perdrix, "Pivoting makes the ZX-calculus complete for real stabilizers", in Proceedings of the 10th International Workshop on Quantum Physics and Logic, Castelldefels (Barcelona), Spain, 17th to 19th July 2013, Vol. 171, edited by B. Coecke and M. Hoban, Electronic Proceedings in Theoretical Computer Science (2014), pp. 50–62.

[65] J. Miller and A. Miyake, "Hierarchy of universal entanglement in 2D measurement-based quantum computation", npj Quantum Information **2**, 16036 (2016).

[66] M Rossi, D Bruß, and C Macchiavello, "Hypergraph states in Grover's quantum search algorithm", Physica Scripta **2014**, 014036 (2014).

[67] M. Gachechiladze, C. Budroni, and O. Gühne, "Extreme Violation of Local Realism in Quantum Hypergraph States", Phys. Rev. Lett. **116**, 070401 (2016).

[68] J. Vicary, "Topological Structure of Quantum Algorithms", in 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science (2013), pp. 93–102.

[69] M. Wilson and G. Chiribella, "A Diagrammatic Approach to Information Transmission in Generalised Switches", Electronic Proceedings in Theoretical Computer Science **340**, 333–348 (2021).

[70] S. Breiner, C. Miller, and N. Ross, "Graphical Methods in Device-Independent Quantum Cryptography", Quantum **3**, `10.22331/q-2019-05-27-146` (2017).

[71] P. Selinger, "Dagger Compact Closed Categories and Completely Positive Maps: (Extended Abstract)", Electronic Notes in Theoretical Computer Science **170**, Proceedings of the 3rd International Workshop on Quantum Programming Languages (QPL 2005), 139–163 (2007).

[72] M. S. Tame and M. S. Kim, "Scalable method for demonstrating the Deutsch-Jozsa and Bernstein-Vazirani algorithms using cluster states", Phys. Rev. A **82**, 030305 (2010).

[73] N. Johansson and J.-Å. Larsson, "Efficient classical simulation of the Deutsch–Jozsa and Simon's algorithms", Quantum Information Processing **16**, 233 (2017).

[74] J. Siewert and R. Fazio, "Quantum Algorithms for Josephson Networks", Phys. Rev. Lett. **87**, 257905 (2001).

[75] N. Schuch and J. Siewert, "Implementation of the Four-Bit Deutsch–Jozsa Algorithm with Josephson Charge Qubits", physica status solidi (b) **233** (2002).

[76] D. Collins, K. W. Kim, and W. C. Holton, "Deutsch-Jozsa algorithm as a test of quantum computation", Physical Review A **58**, R1633–R1636 (1998).

[77] M. Ettinger and P. Høyer, "On Quantum Algorithms for Noncommutative Hidden Subgroups", Advances in Applied Mathematics **25**, 239–251 (2000).

[78] M. Ettinger, P. Hoyer, and E. Knill, "Hidden Subgroup States are Almost Orthogonal", (1999).

[79] B. Coecke and A. Kissinger, *Categorical Quantum Mechanics I: Causal Quantum Processes*, Nov. 2017.

[80] B. Coecke and A. Kissinger, "Categorical quantum mechanics II: Classical-quantum interaction", International Journal of Quantum Information **14**, 1640020 (2016).

[81] V. authors, *Qiskit Textbook*, 2023.

[82] M. S. Tame, B. A. Bell, C. Di Franco, W. J. Wadsworth, and J. G. Rarity, "Experimental Realization of a One-Way Quantum Computer Algorithm Solving Simon's Problem", Phys. Rev. Lett. **113**, 200501 (2014).