Quantum information processing with Clifford quantum cellular automata

Von der Fakultät für Mathematik und Physik
der Gottfried Wilhelm Leibniz Universität Hannover

zur Erlangung des Grades
Doktor der Naturwissenschaften
Dr. rer. nat.

genehmigte Dissertation

von

Dipl. Phys. Johannes Gütschow
geboren am 23.12.1981, in Hamburg

2012

Referent: Prof. Dr. Reinhard F. Werner
Koreferent: Prof. Dr. Tobias J. Osborne
Tag der Promotion: 9. November 2012

Abstract

In this thesis, we study quantum information processing under constraints on the available operations and resources. The experimental difficulty in implementing quantum computers with many parallel qubits renders resource-efficient quantum computational schemes with a reduced set of operations a necessity for successful quantum information processing.

We focus on Clifford operations, especially Clifford quantum cellular automata (CQCAs), which are a building block of several quantum computational schemes. By loosening the translation invariance, we incorporate Clifford causal operations and convolutional stabilizer codes into the framework of CQCAs. Memory channels are utilized as a resource-efficient way to implement causal operations and convolutional encoders. Based on this this framework, we study the performance of quantum convolutional codes under resource constraints.

In our analysis of CQCAs we present a complete classification of one-dimensional CQCAs and their time evolution. We determine invariant states and prove that CQCAs generate entanglement at the maximal rate possible for translation-invariant operations. We furthermore show that the spacetime image of a broad class of linear Cellular automata, including CQCAs, exhibits a self similar structure.

It is proven that quasi-local causal operations can be implemented by forgetful memory channels, whereas finite depth causal operations correspond to strictly forgetful channels. We find the required memory dimension equals the index of the causal operation, enabling a resource efficient implementation. Furthermore, we introduce the use of Bratteli diagrams to analyze the memory dynamics. We prove the existence of causal inverses and bounds on their resource requirements. This gives us the means to construct finite depth inverses of memory channels, i.e. operations that recover the sent information with a delay independent of the transmission length.

A theory of Clifford memory channels is established, including a criterion for forgetfulness. We employ this theory to introduce a new channel-based approach to quantum convolutional codes which facilitates a better understanding of catastrophic errors and the construction of non-catastrophic and finite-depth encoders and decoders. We prove a Hamming bound for convolutional codes, finding that convolutional codes have the potential to outperform block codes under resource constraints.

Keywords: Quantum information, Cellular automata, Clifford operations

Zusammenfassung

In der vorliegenden Arbeit untersuchen wir Quanteninformationsverarbeitung mit eingeschränkten Ressourcen und Operationen. Da Quantenrechnungen mit vielen Qubits bisher kaum realisiert werden können, sind Ressourcen schonende, auf einem reduzierten Satz von Operationen basierende Konzepte für Quantencomputer notwendig.

Der Schwerpunkt dieser Arbeit liegt auf Clifford Operationen, insbesondere Clifford Quantenzellularautomaten (CQCAs), die als Baustein in vielen Konzepten für Quantenrechner verwendet werden. Durch die Lockerung der Translationsinvarianz integrieren wir kausale Clifford Operationen, Clifford Speicherkanäle und Stabilisator Faltungscodes in den CQCA Formalismus. Wir verwenden Speicherkanäle als Ressourcen schonende Implementierung von kausalen Operationen und Faltungscodes und untersuchen mit diesem Ansatz die Leistung von Quantenfaltungscodes unter eingeschränkten Ressourcen.

Unsere Analyse von CQCAs ergibt eine Klassifizierung der Zeitentwicklung eindimensionaler CQCAs. Wir bestimmen invariante Zustände und zeigen, dass CQCAs Verschränkung mit der für translationsinvariante Operationen maximalen Rate erzeugen. Weiterhin zeigen wir, dass die Zeitentwicklung von linearen Zellularautomaten eine selbstähnliche Struktur hat.

Wir zeigen, dass quasi-lokale kausale Operationen durch vergessliche Speicherkanäle implementiert werden, während kausale Operationen mit endlicher Ausbreitung strikt vergesslichen Speicherkanälen entsprechen. Die für die Implementierung nötige Speicherdimension entspricht dem Index der kausalen Operation. Dies ist die Grundlage für eine Ressourcen schonende Implementierung kausaler Operationen. Außerdem führen wir die Verwendung von Bratteli Diagrammen zur Analyse der Speicherdynamik ein. Wir beweisen die Existenz von kausalen Inversen und bestimmen Schranken an die benötigten Ressourcen. Diese Resultate nutzen wir für die Konstruktion von Inversen endlicher Tiefe, also Operationen, die einen Speicherkanal mit einer Verzögerung, die unabhängig von der Länge der Übertragung ist, invertieren.

Wir entwickeln eine Theorie der Clifford Speicherkanäle, die ein Kriterium für Vergesslichkeit beinhaltet. Mit Hilfe dieser Theorie führen wir einen neuen Formalismus für Quantenfaltungscodes ein, der ein besseres Verständnis nicht katastrophaler Codierer und ihrer Konstruktion ermöglicht. Wir beweisen eine Hamming-Schranke für Faltungscodes und zeigen, dass Faltungscodes das Potential besitzen, unter Ressourcenbeschränkung bessere eine Coderaten zu ermöglichen, als blockorientierte Codes.

Schlagworte: Quanteninformation, Zellularautomaten, Clifford Operationen

Contents

List of figures

Notation

Here, we will give a brief overview over the notation used in this thesis. While aiming to be consistent throughout the whole thesis, there are still bits of notation that are not unique, letters that are used multiple times for different objects, and objects that do not use the notation they should.

The different quantum system used for the description of quantum processes use calligraphic letters, i.e. $\mathscr{A}$, $\mathscr{B}$, etc. as labels. Usually $\mathscr{B}$ is the output system of a process and $\mathscr{A}$ the input system. Processes with memory use $\mathscr{M}$ as an additional memory system. If the systems are subdivided, the subsystems are denoted by subscript indices, for example $\mathscr{A}_1$.

Mathematically, the systems are described by their observable algebras. The observable algebra of a quantum systems is denoted in black letter font by the same letter used to denote the system, i.e. $\mathfrak{A}$, $\mathfrak{B}$, and $\mathfrak{M}$. Elements of the algebras (observables) are denoted by uppercase italic letters: A, B, C. Sometimes we also use lowercase italic letters, for example if an observable is decomposed.

Maps between quantum systems also use uppercase italic letters. Their arguments are put in square brackets. Maps on an infinite chain of systems, for example Quantum cellular automata, are usually called T while maps acting only on a finite number of systems, for example quantum channels, usually use S. We use this to easily distinguish between a global map and its decomposition into maps acting only on a finite number of systems.

In most cases we use Ξ with some subscript index to denote the phase space. For the dimension of the underlying field we use p, because we mostly deal with fields of prime characteristic. Phase space vectors are usually called ξ, η, and ζ, also with subscript indices. The map between phase space and the observable algebra, is denoted by $\mathbf{w}$. The phase space map describing a quantum operation T is denoted by the same letter, but lower case and bold face: $\boldsymbol{t}$. The phase function is called $\lambda(\bullet)$.

When dealing with translation invariant algebras or automorphisms we will often use the lattice translation, which we denote by τ. Translations by n sites are denoted by τ^n. For translations on the phase space we use $\hat{\tau}$ and $\hat{\tau}^n$.

For the Pauli matrices that we will often use, we employ two different notations. When dealing with coding questions we will usually use X, Y, and Z, while we will mainly use σ_i in the CQCA-chapter.

1 Introduction

Quantum information theory has come a long way from the first ideas of using quantum systems for computing purposes, simulation of quantum mechanical systems, and cryptography to its current state as an independent research field. Several theoretical models for universal quantum computers, the most prominent ones being the gate model [1], measurement based quantum computation [2], universal quantum cellular automata [3, 4], and quantum walks [5, 6], are implemented in experimental setups using trapped ions [7, 8], neutral atoms in optical lattices [9, 10], Josephson junctions [11], and many ideas more. Various algorithms exist which would allow a quantum computer to solve problems more efficiently than a classical computer. Amongst those are algorithms which have a polynomial or even superpolynomial speedup over all known classical algorithms for the problem. The most famous one is the Shor algorithm for prime factorization [12], allowing a quantum computer to decypher current cryptographic systems such as RSA. Other algorithms include database searching [13] and solving linear equations [14].[1]

Quantum cryptography enables provably secure generation of secret keys that can be used to encrypt classical messages. Several key generation schemes exist in theory and are implemented in experiment with diverse setups.

However big the improvements and achievements are, most quantum information technologies are still far from being commercially usable. Quantum cryptography is still a niche product and open questions regarding the security remain. Quantum computing has an even longer way to go: despite progress in the number of qubits that can be coherently controlled, experiments are still far from implementing computations that a classical computer could not do as well. To assess whether an experimental setup does what it was programmed to, the operation has to undergo a process tomography. The needed measurements and the classical post-processing scale exponentially in the number of qubits, rendering full tomography of multi-qubit processes impossible unless special symmetries are assumed [15]. Therefore, quantum computational schemes have to build upon a set of basic operations, acting only on a few qubits each, which are easy to verify, provably universal, and scalable.

Most schemes of quantum information processing need a way to correct errors to fight the noise inevitably introduced through interaction with the environment.

[1] A fairly complete list (at the time of writing) including references can be found under http://math.nist.gov/quantum/zoo.

Different schemes for error-correction exist, with convolutional codes among the most promising candidates for quantum communication. Despite a lot of evidence that convolutional codes have the potential to outperform block codes a rigorous analysis is still open.

It is therefore crucial for the success of quantum computation to find computational schemes which use operations that can be efficiently described and tested, while keeping the capability of solving problems a classical computer can not solve efficiently. These basic operations have to be chosen such that the number of qubits needed for basic computational tasks and therefore the required quantum memory is reduced to a minimum. The potential performance of convolutional codes has to be assessed rigorously to identify boundary conditions for the search for resource efficient error-correction.

In this thesis we combine both approaches. We use symmetries like translation-invariance and reductions to special sets of gates to reduce the dimension of the space of quantum operations we have to consider. A translation-invariant operation is such that it acts in the same way on each quantum system or group of neighboring quantum systems. In this restricted setting universal quantum computation is still possible if general input states are allowed [3, 4]. Furthermore, we will often focus our analysis on Clifford operations, which are used as building blocks in several quantum computational schemes and are a central element of quantum error-correction codes.

To reduce the resource requirements, we consider the local implementability of global operations and the minimal quantum memory requirements. In the case of error-correction codes we carry out a performance analysis under constraint spatial resources to assess which types of codes will be beneficial in a this setting.

Let us now briefly comment on our main results.

Main results

We develop a complete classification of one-dimensional CQCAs and their time evolution, distinguishing periodic CQCAs (Proposition 5.1.1), CQCAs with gliders (Proposition 5.1.9), and fractal CQCAs. We prove, that the time evolution and therefore the class of a CQCA is determined only by the trace of its matrix representation. For glider CQCAs we prove an equivalence theorem (Theorem 5.1.10) which enables us to carry over results obtained for a standard glider CQCA to all other CQCAs of the same equivalence class. This is used in the study of quasifree states in Section 5.2.4. We furthermore prove that CQCAs generate entanglement at a linear rate starting from stabilizer (Theorems 5.3.6 and 5.3.8) and quasifree (Section 5.3.2) states, saturating the bound for entanglement generation by translation-invariant operations (Theorem 3.2.3).

We prove that the spacetime image of all CQCAs is self-similar, with the class of fractal CQCAs as the only class which produces a spacetime image with non-integer

Hausdorff dimension—a fractal. In our proof we generalize our analysis to linear Cellular automata on Abelian groups, therefore integrating a multitude of results for special classes and examples into a common framework (Chapter 6).

Using a channel representation of reversible causal operations (Theorem 4.1.1), we prove that reversible, quasi-local causal operations can be implemented by a series of forgetful memory channels (Theorem 4.1.3). Finite depth causal operations, i.e. operations which transfer information only finitely far in one timestep, correspond to strictly forgetful memory channels (Corollary 4.1.4). For the latter we prove the existence of inverses up to a shift, which we call causal inverses (Theorem 4.5.1). We find the required memory dimension to equal the index of the causal operation (Theorem 4.2.1) enabling a resource efficient implementation of causal operations. Furthermore, we introduce the use of Bratteli diagrams to analyze the memory dynamics and derive a bound on the memory depth (Equation 4.27) of reversible strictly forgetful channels which we use to derive bounds on the needed memory for causal inverses (Section 4.5.2).

A theory of Clifford memory channels is established, including a classical description which gives rise to an easy to evaluate criterion for forgetfulness (Theorem 3.7.4). We show that reversible Clifford channels are either strictly forgetful or not forgetful at all (Corollary 3.7.5).

We introduce a new channel-based approach to quantum convolutional codes which enables a better understanding of catastrophic errors and the construction of non-catastrophic and finite-depth encoders and decoders. We present an example (Example 7.1.3) revealing a flaw in a commonly used definition of catastrophic errors and develop a corrected definition (Definition 7.1.4). Based on the decomposition of causal operations, we develop a new algorithm to determine the coding operations from the stabilizer generators. This algorithm is able to find non-catastrophic encoders in cases where the established algorithm fails and we prove conditions on the stabilizer generators which guarantee the existence of non-catastrophic coding operations. Furthermore, our results on causal inverses provide a way to construct finite-depth decoders.

Finally, we prove a Hamming bound for convolutional codes (Section 7.3), finding that convolutional codes have the potential to outperform block codes under resource constraints (Section 7.4).

Overview

We begin with an overview over our mathematical prerequisites in Chapter 2, where we first introduce C^*, approximately finite (AF), and quasi-local algebras, as well as Bratteli diagrams (Section 2.1). In the following, Weyl systems (Section 2.2), which are the foundation of the classical description of Clifford operations, are introduced. The chapter is concluded with a short introduction to stabilizer states (Section 2.3).

Chapter 3 introduces the basic concepts used in this thesis, but also contains some first results. We begin by introducing the neighborhood calculus [16] for operations on a chain (Section 3.1) and continue with a brief introduction to QCAs (Section 3.2), their index theory and entanglement generation. Causal operations are introduced in Section 3.3. Our notion of resource requirements is briefly elucidated in the following section. We move on to Quantum channels with memory (Section 3.5) to pave the ground for the theory of Clifford memory channels. Section 3.6 introduces the theory of Clifford channels, while Sections 3.7 and 3.6.3 incorporate our results on Clifford memory channels. Finally, quantum error-correction and convolutional codes are introduced (Section 3.9), where we also demonstrate shortcomings in the standard framework for quantum convolutional codes.

The following chapter (Chapter 4) contains our results on causal operations. The decomposition of causal operations into a series of memory channels is introduced in Section 4.1. In Section 4.2 we relate the memory requirements to the index and use the result to find a memory efficient decomposition (Section 4.3) including the Clifford case (Section 4.3.1). We continue with the introduction of Bratteli diagrams for the analysis of the memory dynamics in Section 4.4, again including the Clifford case. The chapter is concluded with the study of causal inverses (Section 4.5).

CQCAs are studied in Chapter 5. We begin with the general formalism and continue with the classification (Section 5.1). Section 5.2 is dedicated to time asymptotics including invariant states and stationary states. Entanglement generation is studied in Section 5.3. We add some short notes on finite systems (Section 5.4) and review some applications (Section 5.5). The chapter is concluded with circuit and channel implementations of causal CQCAs (Section 5.7) and a comment on the Bratteli diagrams of causal CQCAs (Section 5.6).

The class of fractal CQCAs is further analyzed in Chapter 6. We generalize the classical description of CQCAs to linear cellular automata over Abelian groups (Section 6.1). First, we present a heuristic explanation for the self similar structure in the spacetime image of an example CQCA (Section 6.3). In the following, we develop the general recursion scheme, prove that all linear CAs generate self similar spacetime images (Section 6.4), and show how to compute features of the spacetime image like the fractal dimension.

In the final chapter of this thesis, we use the formalism of Clifford causal operations and memory channels to introduce a channel-based approach to quantum convolutional stabilizer codes. We begin with a review of different definitions of catastrophic errors to motivate the definition used in this thesis (Section 7.1). In Section 7.2.1 we present an algorithm to determine encoders and finite depth decoders for convolutional stabilizer codes given the stabilizer generators and show under which conditions the encoder is non-catastrophic. We then derive the quantum convolutional Hamming bound (Section 7.3) and utilize it to compare the possible performance of block codes and convolutional codes (Section 7.4).

The thesis is concluded with an outlook and some open questions.

2 Mathematical prerequisites

In this chapter we will introduce the basic mathematical methods needed in the rest of the thesis. We will mainly describe quantum mechanics in terms of C^* observable algebras, which corresponds to the Heisenberg picture of quantum mechanics. Instead of evolving a state, described by a vector of the systems Hilbert space or a density matrix, we evolve observables backwards in time to describe the same physical process and the same measurement results. The observable algebra of the whole system can be described in terms of the observable algebras of the constituent systems via AF-algebras. We are usually interested only in observables that are localized on a finite part of the system. They can easily be described in a basis of the observable algebras of the constituent systems. States however have to be considered globally. Global entangled states can not be described as linear combinations of local states which would our analysis unnecessarily complex, as we are usually interested in system properties we can measure on a finite part of the system, i.e. with localized observables. Additionally, in the Hilbert space of infinitely many tensor factors the definition of a scalar product is problematic because it is not guaranteed that the scalar product defined by the product of the single site scalar products converges.

We will begin with a brief introduction of some needed basics on C^*-algebras and linear maps. Furthermore, AF-algebras and Bratteli diagrams will be introduced (Section 2.1). Throughout the thesis we will often use discrete Weyl systems which give us a way to describe certain quantum time evolutions (the Clifford operations) in an efficient way by a classical time evolution in phase space. Weyl systems will be introduced in Section 2.2. Finally we will introduce stabilizer states which are tightly connected to Weyl systems and Clifford operations (Section 2.3).

2.1 C^*-algebras

A C^*-*algebra* $\mathfrak{A}$ is a vector space over the complex numbers that is additionally equipped with a product "$\cdot$" and an adjoint operation "*". The product has to be associative and distributive but not necessarily commutative. The adjoint operation * has to satisfy

$$(A+B)^* = A^* + B^*, \tag{2.1}$$

$$(A\cdot B)^* = B^*\cdot A^*, \tag{2.2}$$

$$(\lambda A)^* = \bar{\lambda} A^*, \tag{2.3}$$
$$(A^*)^* = A \tag{2.4}$$

for every A, $B \in \mathfrak{A}$ and every $\lambda \in \mathbb{C}$. Furthermore every C^*-algebra is equipped with a *norm* $\|\bullet\|_\infty$ that fulfills

$$\|\lambda A\|_\infty = |\lambda| \|A\|_\infty, \tag{2.5}$$
$$\|A + B\|_\infty \leqslant \|A\|_\infty + \|B\|_\infty, \tag{2.6}$$
$$\|A \cdot B\|_\infty \leqslant \|A\|_\infty \cdot \|B\|_\infty, \tag{2.7}$$
$$\|A^* A\|_\infty = \|A\|_\infty^2, \tag{2.8}$$

for all A, $B \in \mathfrak{A}$ and all $\lambda \in \mathbb{C}$.

A C^*-algebra $\mathfrak{A}$ can have an *identity element* $\mathbb{1}_\mathfrak{A}$ with the property $\mathbb{1}_\mathfrak{A} A = A = A\mathbb{1}_\mathfrak{A}$ for all $A \in \mathfrak{A}$. If a C^*-algebra has no identity we can always embed it into a larger algebra with an identity. We will therefore always assume that the algebras we are working with have an identity element. Unless the algebra consists only of 0 we always have $\|\mathbb{1}_\mathfrak{A}\|_\infty = 1$.

If the product is commutative a C^*-algebra is called *Abelian*. The set of elements that commute with all elements of an algebra $\mathfrak{A}$ is called the *center* $\mathfrak{C}(\mathfrak{A})$ of the algebra:

$$\mathfrak{C}(\mathfrak{A}) := \{A | A \in \mathfrak{A}, [A, B] = 0 \; \forall B \in \mathfrak{A}\}. \tag{2.9}$$

The center of $\mathfrak{A}$ is an Abelian subalgebra of $\mathfrak{A}$.

Finitely dimensional C^*-algebras have an especially simple structure: they are always isomorphic to a direct sum of *full matrix algebras*, i.e. algebras of square matrices over the complex numbers.

$$\mathfrak{A} \cong \bigoplus_{P_i \in \min \mathfrak{A}} \mathfrak{A} P_i, \tag{2.10}$$

where "min $\mathfrak{A}$" is the set of non-zero self-adjoint minimal central projections[1] of $\mathfrak{A}$. Each $\mathfrak{A}P_i$ is isomorphic to a full matrix algebra $\mathcal{M}_{\dim(P_i)}(\mathbb{C})$. In the following we will often only write $\mathcal{M}_d$ for $\mathcal{M}_d(\mathbb{C})$. The set $\{d_i = \dim(P_i)\}_i$, is called the dimension vector of $\mathfrak{A}$. Two finite dimensional C^*-algebras are isomorphic if and only if their dimension vectors are equal up to permutations. For details see e.g. Section I.11 of [17] or Chapter III of Davidsons textbook [18].

C^*-algebras are closely connected to algebras of bounded operators on Hilbert spaces. On the one hand the algebra of bounded operators $\mathcal{B}(\mathcal{H})$ on a Hilbert space $\mathcal{H}$ is always a C^*-algebra, on the other hand by the Gelfand-Naimark theorem [19] an arbitrary C^*-algebra $\mathfrak{A}$ is isometrically isomorphic to $\mathcal{B}(\mathcal{H})$ for some Hilbert space $\mathcal{H}$.

[1] Central projections commute with all elements of $\mathfrak{A}$. A projection is called minimal if it can not be decomposed into a sum of two projections.

A *state* on a C^*-algebra is a linear functional $\omega : \mathfrak{A} \to \mathbb{C}$ which is *positive* meaning $\omega(A^*A) \geqslant 0 \; \forall A \in \mathfrak{A}$ and *normalized*, i.e. $\omega(\mathbb{1}_{\mathfrak{A}}) = 1$.

2.1.1 Linear maps

Let us introduce a few properties of linear maps that we will frequently use. A map $T : \mathfrak{B} \to \mathfrak{A}$ from one algebraic structure $\mathfrak{B}$ to another algebraic structure $\mathfrak{A}$ is called a *homomorphism* if it preserves the structure of the underlying algebraic structures $\mathfrak{B}$ and $\mathfrak{A}$ (ring, vector space, algebra, ...). In the case of linear maps the structure of addition is always preserved. Therefore every linear map on a vector space is a homomorphism. If the underlying structures are algebras we additionally need to preserve the multiplicative structure: $T[A \cdot B] = T[A] \cdot T[B]$. On a C^*-algebra the adjoint operation must also be preserved: $T[A^*] = T[A]^*$. There are several special cases of homomorphisms we will use in this thesis. An *isomorphism* is a bijective homomorphism and a *monomorphism* is an injective homomorphism. An *endomorphism* is a homomorphism from an object to itself; if it is also bijective it is called an *automorphism*. Sometimes we also speak of *morphisms* that are structure preserving maps in a generalized setting used in category theory.

We now restrict ourselves to maps between C^*-algebras. A map $T : \mathfrak{B} \to \mathfrak{A}$ is called *positive* if $T[B] \geqslant 0 \; \forall B \geqslant 0 \in \mathfrak{B}$. We say than an operator B is positive ($B \geqslant 0$) if B is hermitian and has a nonnegative spectrum. Thus positive maps map positive operators onto positive operators. In quantum mechanics this property is important, because density matrices describing states of the system are positive operators. They have to be mapped to other density matrices by the time evolution and therefore the time evolution map has to map positive operators to positive operators. Since physics should be describable in a local way it is important to know that we can leave systems on which a map acts trivially out of the analysis. E.g. when describing our experiment we do not want to have to include the whole lab and ultimately the universe into our description. Therefore we need to know that important properties of a map are not changed when we add extra systems on which the map acts trivially. In the case of positivity this is not guaranteed. A map that acts positive on a system itself might be no longer positive if we add an ancilla system and extend the map by the identity on this system. The customary example for this is the partial transposition map. To circumvent this problem the possibility of adding additional systems is taken into account in the following changed definition of positivity: A map $T : \mathfrak{B} \to \mathfrak{A}$ is called *completely positive* (CP) if $T \otimes \mathrm{id}_n : \mathfrak{B} \otimes \mathcal{M}_n \to \mathfrak{A} \otimes \mathcal{M}_n$ is positive for all n. Here id_n denotes the *identity map* on $\mathcal{M}_n$.

A map is called *trace preserving* if $\mathrm{tr}(T[B]) = \mathrm{tr}[B] \; \forall B \in \mathfrak{B}$. Finally, a map is called *unital* if it maps the identity element of the input algebra to the identity element of the output algebra: $T[\mathbb{1}_{\mathfrak{B}}] = \mathbb{1}_{\mathfrak{A}}$.

To determine if a map is completely positive we will often use the criterion that a

linear map $T: \mathfrak{B} \to \mathfrak{A}$ is completely positive if and only if

$$\sum_{i,j} a_i^* T[b_i^* b_j] a_j \geqslant 0 \quad \forall a_i \in \mathfrak{A}, \forall b_i \in \mathfrak{B}. \tag{2.11}$$

For a proof see Appendix 1.

To denote a time evolution in the Heisenberg picture we use the description of the map acting on the observable algebra. The T evolved system is then denoted by the state $\omega \circ T: \mathfrak{B} \to \mathbb{C}$, thus the observable is first transformed according to the Heisenberg representation of the channel and then fed into the original state of the system.

2.1.2 AF-algebras and Bratteli diagrams

Approximately finite dimensional (AF) algebras are inductive limits of series of finite dimensional C^*-algebras [20].

Definition 2.1.1. *A C^*-algebra $\mathfrak{A}$ is called approximately finite dimensional (AF) if there exists a sequence of finite dimensional C^*-algebras such that $\mathfrak{A}$ is the norm closure of $\bigcup_n \mathfrak{A}_n$. The $\mathfrak{A}_n$ are embedded into each other in the sense that $\mathfrak{A}_n$ is a subalgebra of $\mathfrak{A}_{n+1}$*

$$\mathfrak{A} = \overline{\bigcup_n \mathfrak{A}_n}. \tag{2.12}$$

Alternatively, an AF-algebra can be defined as the limit $n \to \infty$ of a series of finite dimensional C^*-algebras $\mathfrak{A}_n$ embedded into each other by a series of C^*-homomorphisms $\alpha_n: \mathfrak{A}_n \to \mathfrak{A}_{n+1}$:

$$\mathfrak{A} = \lim_{n\to\infty} \quad \mathfrak{A}_0 \xrightarrow{\alpha_0} \mathfrak{A}_1 \cdots \mathfrak{A}_n \xrightarrow{\alpha_n} \mathfrak{A}_{n+1}. \tag{2.13}$$

Bratteli originally required a unit element $\mathbb{1}$ for every AF-algebra. Then the α_n are unital C^*-homomorphisms and $\mathfrak{A}_0 = \mathbb{C}\mathbb{1}$. As we are always dealing with unital algebras we will only consider this case in this thesis.

The diagrams of the type of Equation (2.13) can be be extended to include the internal structure of the algebras. Each $\mathfrak{A}_n$ is finite dimensional and can be decomposed into a direct sum of full matrix algebras

$$\mathfrak{A}_n \cong \bigoplus_{P_{ni} \in \min \mathfrak{A}_n} \mathfrak{A}_n P_{ni} \cong \bigoplus_{ni} \mathcal{M}_{d_{ni}}(\mathbb{C}). \tag{2.14}$$

Finite dimensional C^*-algebras can be embedded into each other by homomorphisms in the following way.

Lemma 2.1.2 ([18]). *Let α be a unital C^*-homomorphism from a finite dimensional C^*-algebra $\mathfrak{A}_n \cong \bigoplus_{j=1}^{k} \mathcal{M}_{d_{nj}}$ into a second finite dimensional C^*-algebra $\mathfrak{A}_m \cong \bigoplus_{i=1}^{l} \mathcal{M}_{d_{mi}}$. Then the map α is defined up to unitary equivalence (in $\mathfrak{A}_m$) by an $l \times k$ matrix $A = \sum_{ij} a_{ij} e_{ij}$, $a_{ij} \in \mathbb{N}$ such that*

$$A \begin{pmatrix} d_{n1} \\ \vdots \\ d_{nk} \end{pmatrix} = \begin{pmatrix} d_{m1} \\ \vdots \\ d_{ml} \end{pmatrix}. \tag{2.15}$$

Furthermore

$$\sum_{j=1}^{k} a_{ij} d_{nj} = d_{mi}, \quad 1 \leqslant i \leqslant l. \tag{2.16}$$

We can now illustrate every unital C^*-homomorphism α by a graph using its matrix A in the following way: The summand dimensions d_{nk} and d_{ml} form the edges of the graph. Then for each pair (d_{nj}, d_{mi}) a_{ij} edges connecting d_{nj} with d_{mi} are added to complete the graph. Combining the graphs for all the homomorphisms α_n of an AF-algebra $\mathfrak{A}$ into one graph we obtain the *Bratteli diagram* of $\mathfrak{A}$. Bratteli diagrams were introduced in [20] to study isomorphisms between AF-algebras. We followed the formulation of Davidsons textbook [18].

Example 2.1.3. *Figure 2.1 shows an example of a Bratteli diagram. The algebras in the example are: $\mathfrak{A}_0 \cong \mathcal{M}_1$, $\mathfrak{A}_1 \cong \mathcal{M}_2 \oplus \mathcal{M}_1$, $\mathfrak{A}_2 \cong \mathcal{M}_3 \oplus \mathcal{M}_2$, $\mathfrak{A}_3 \cong \mathcal{M}_3 \oplus \mathcal{M}_5 \oplus \mathcal{M}_2$, $\mathfrak{A}_4 \cong \mathcal{M}_8 \oplus \mathcal{M}_6 \oplus \mathcal{M}_2 \oplus \mathcal{M}_2$. In the diagram one can see that Equation (2.16) is fulfilled. The dimension of each vertex equals the sum over all incoming edges from the layer above multiplied with the dimension of the vertex from which the edge originates.*

As an example for the embedding automorphisms consider $\alpha_1 : \mathcal{M}_2 \oplus \mathcal{M}_1 \to \mathcal{M}_3 \oplus \mathcal{M}_2$ with

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \oplus (b_{11}) \mapsto \begin{pmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ 0 & 0 & b_{11} \end{pmatrix} \oplus \begin{pmatrix} b_{11} & 0 \\ 0 & b_{11} \end{pmatrix}$$

A special class of AF-algebras are the *uniformly hyperfinite* (UHF) algebras. A UHF-algebra $\mathfrak{A}$ is an AF-algebra where all the $\mathfrak{A}_n$ are full matrix algebras. A unital embedding of $\mathcal{M}_m$ into $\mathcal{M}_n$ requires $m|n$. Therefore the $\mathfrak{A}_n = \mathcal{M}_{d_n}$ are given by a series of integers d_n with $d_n | d_{n+1}$. An important feature of UHF-algebras is that their center is always trivial, i.e. it only consists of multiples of the identity.

2.1.3 Quasi-local algebras

Even when dealing with infinite lattices of quantum systems we are usually interested in properties that can be observed locally and are thus described by localized

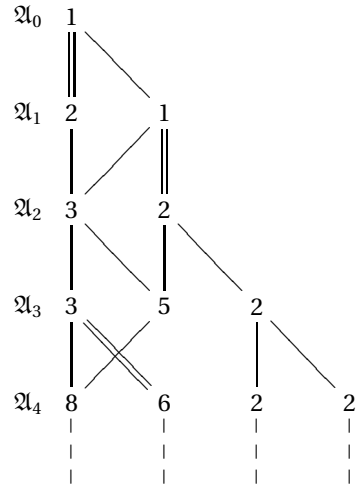


Figure 2.1: Example of a Bratteli diagram.

observables, i.e. observables that only differ from the identity on finitely many systems contained in a finite region of the lattice. The operations we consider, e.g. quantum cellular automata map localized observables onto localized observables. Therefore it is appropriate to describe the observable algebra of our system in terms of localized observables. We use the *quasi-local algebras* that are constructed from algebras on finite subsets of the lattice.

Let $\mathbb{Z}^s$ be a lattice[2] of quantum systems with observable algebras $\mathfrak{A}_i$. For every finite subset Λ of $\mathbb{Z}^s$ the observable algebra $\mathfrak{A}_\Lambda$ is of the form $\mathfrak{A}_\Lambda = \bigotimes_{i\in\Lambda}\mathfrak{A}_i$. From the set of all local algebras we construct the quasi-local algebra $\mathfrak{A}_{\mathbb{Z}^s}$ as the norm closure of the union of all local algebras $\mathfrak{A}_\Lambda$, $\Lambda \subset \mathbb{Z}^s$, $|\Lambda| < \infty$:

$$\mathfrak{A}_{\mathbb{Z}^s} = \overline{\bigcup_{\Lambda\subset\mathbb{Z}} \mathfrak{A}_\Lambda}. \tag{2.17}$$

For infinite subsets of $\mathbb{Z}^s$, e.g. $\Lambda = \mathbb{Z}^{s-1}$, we define the quasi-local algebra in an analogous way as the closure of the union of the algebras $\mathfrak{A}(\Upsilon)$ of all finite subsets Υ of Λ. We will often abbreviate the notation and write $\mathfrak{A} = \bigotimes_{i\in\mathbb{Z}^s}\mathfrak{A}_i$ to denote the quasi-local algebra. For algebras on subsets λ we use the notation $\mathfrak{A}_\Lambda$ as well as $\mathfrak{A}(\Lambda)$.

A useful property of quasi-local algebras is that all observables can be described by localized observables up an arbitrarily small error ε (in operator norm): Let $A \in$

[2] The regularity of the lattice is never used. Any graph would be sufficient.

$\mathfrak{A}$, then for all $\varepsilon > 0$ there exists an observable $A_\varepsilon \in \Lambda_\varepsilon$, $\Lambda_\varepsilon \subset \mathbb{Z}^s$ finite, such that $\|A - A_\varepsilon\|_\infty \leqslant \varepsilon$.

The localization of observables can also be described by the localization of their commutant, as we will show in the following lemma.

Lemma 2.1.4. *Let $A \in \mathfrak{A}$, $\mathfrak{A} = \bigotimes_{i\in\mathbb{Z}} \mathfrak{A}_i$ and let $\Lambda \subset \mathbb{Z}$. If and only if $[A, \mathfrak{A}(\Lambda)] = \{0\}$, A is localized on Λ^c, where Λ^c denotes the complement of Λ in $\mathbb{Z}$.*

Proof. A is an element of a quasi local algebra, so it can be approximated up to any $\varepsilon > 0$ by an element A_ε that is located on a finite region Λ_ε. Since Λ_ε is finite, we can decompose A_ε into $A_\varepsilon = \sum_i A_{\varepsilon,i} \otimes A^c_{\varepsilon,i}$, where $A_{\varepsilon,i} \in \mathfrak{A}(\Lambda_\varepsilon \cap \Lambda)$ and $A^c_{\varepsilon,i} \in \mathfrak{A}(\Lambda_\varepsilon \cap \Lambda^c)$:

$$\left\| A - \sum_i A_{\varepsilon,i} \otimes A^c_{\varepsilon,i} \right\| \leqslant \varepsilon. \tag{2.18}$$

Let $\mathscr{U}_\varepsilon$ be the group of unitary elements of $\mathfrak{A}(\Lambda_\varepsilon \cap \Lambda)$. By assumption $[A, U] = 0$ for all $U \in \mathscr{U}_\varepsilon$. We define the averaged element

$$\begin{aligned} A'_\varepsilon &:= \int_{\mathscr{U}_\varepsilon} \mathrm{d}U\, U^* A_\varepsilon U \\ &= \sum_i A_{\varepsilon,i} \otimes \int_{\mathscr{U}_\varepsilon} \mathrm{d}U U^* A^c_{\varepsilon,i} U \\ &= \sum_i A_{\varepsilon,i} \otimes a^c_{\varepsilon,i} \mathbb{1}, \end{aligned}$$

where we used the invariance of the Haar measure $\mathrm{d}U$ on $\mathscr{U}_\varepsilon$. Now

$$\begin{aligned} \|A - A'_\varepsilon\| &= \left\| \int_{\mathscr{U}_\varepsilon} \mathrm{d}U\, U^* (A - A_\varepsilon) U \right\| \\ &\leqslant \int_{\mathscr{A}_\varepsilon} \mathrm{d}U \|U^* (A - A_\varepsilon) U\| \leqslant \varepsilon. \end{aligned}$$

Therefore A can be approximated by finitely localized elements of $\mathfrak{A}(\Lambda^c)$ and is therefore an element of $\mathfrak{A}(\Lambda^c)$.

The other direction is trivially true. □

A state ω on a quasi-local algebra $\mathfrak{A}$ is a positive and normalized functional $\omega : \mathfrak{A} \to \mathbb{C}$ that is defined by a family of density matrices $\{\omega_\Lambda\}_{\Lambda\subset\mathbb{Z}^s}$ on the subalgebras $\mathfrak{A}_\Lambda$, Λ finite. For an observable $A \in \Lambda_1$ the expectation is $\omega(A) = \mathrm{tr}\,(\omega_{\Lambda_1} A)$. When extending the observable A from Λ_1 to $\Lambda_2 \supset \Lambda_1$ by tensoring with $\mathbb{1}_{\Lambda_2\setminus\Lambda_1}$ the expectation values have to be consistent. Therefore, we have to ensure that the density matrices satisfy $\mathrm{tr}_{\Lambda_2\setminus\Lambda_1}(\omega_{\Lambda_2}) = \omega_{\Lambda_1}$ for all Λ_1 and all Λ_2 whenever $\Lambda_2 \supset \Lambda_1$.

A quasi-local algebra $\mathfrak{A}$ is an AF-algebra if the individual algebras $\mathfrak{A}_i$ are finite dimensional. For general theory of quasi-local algebras see e.g. [21].

Proofs and more detailed information about C^*-algebras can be found in the textbooks by Bratteli and Robinson [21, 22], Takesaki [17, 23, 24] and Davidson [18] amongst others. A good source on maps on operator algebras is Paulsens textbook [25].

2.2 Weyl systems

Throughout this thesis we will often be concerned with finding ways to describe quantum systems and their evolution classically in an efficient way.[3] Of course this is not possible for general quantum dynamics. Therefore, we will study special models where such an efficient classical description exists: Clifford channels, stabilizer codes and Clifford quantum cellular automata (CQCAs). In all cases we will use the same method to map the quantum system and its evolution to a classical system with a classical evolution. The tools we use are Weyl systems, which connect operators on a finite dimensional Hilbert space to a commutative group called phase space by projective representations. We choose a basis of $\mathcal{B}(\mathcal{H})$, describe it classically and decompose all other operators with respect to this basis. For an efficient classical description of the evolution to exist, it has to stay within this basis. If a basis element is mapped to a superposition of basis elements we must keep track of the coefficients that will in general be exponentially many with progressing time of the evolution. The time-evolutions that map Weyl operators to Weyl operators (times a phase we can keep track of independently) are called Clifford operations.

Let us first introduce Weyl systems: Weyl operators are often referred to as *generalized Pauli matrices* because qubit Weyl operators are closely related to the Pauli matrices. We show this in the following example:

Example 2.2.1. *The Pauli matrices*

$$\sigma_1 = \sigma_x = X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \tag{2.19}$$

$$\sigma_2 = \sigma_y = Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \tag{2.20}$$

$$\sigma_3 = \sigma_z = Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \tag{2.21}$$

$$\sigma_0 = \mathbb{1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{2.22}$$

[3] with efficient we mean with polynomial complexity

form a basis of hermitian unitary operators that spans the algebra of complex 2×2*-matrices. They fulfill the relations*

$$\sigma_1^2 = \sigma_2^2 = \sigma_3^2 = \mathbb{1}, \tag{2.23}$$

$$\sigma_i \sigma_j = \varepsilon_{ijk} i \sigma_k, \quad i \neq j, \tag{2.24}$$

$$\sigma_i \sigma_j = -\sigma_j \sigma_i, \quad i \neq j. \tag{2.25}$$

We now identify each Pauli matrix with an element of the vector space $\mathbb{F}_2^2$*:*

$$\begin{array}{rcl} \mathrm{w}(0,0) & := & \mathbb{1}, \\ \mathrm{w}(1,0) & := & \sigma_1, \\ \mathrm{w}(0,1) & := & \sigma_3, \\ \mathrm{w}(1,1) & := & i\sigma_2. \end{array} \tag{2.26}$$

The operators $\mathrm{w}(q,p)$ *are called Weyl operators. The group operation translates from* $\mathbb{F}_2^2$ *to the Weyl operators as follows. Using (2.23-2.25) we have*

$$\begin{array}{rcl} \mathrm{w}(\xi+\eta) & = & \mathrm{w}(\xi_X+\eta_X, \xi_Z+\eta_Z) \\ & = & (-1)^{\xi_Z \eta_X} \mathrm{w}(\xi_X, \xi_Z)\mathrm{w}(\eta_X, \eta_Z) = (-1)^{\xi_Z \eta_X} \mathrm{w}(\xi)\mathrm{w}(\eta) \end{array} \tag{2.27}$$

for $\xi = (\xi_X, \xi_Z)$*,* $\eta = (\eta_X, \eta_Z) \in \mathbb{F}_2^2$*. The group operation is thus preserved up to a phase. We call such a construction a* projective representation*. As the Weyl operators form a basis of all complex* 2×2*-matrices, we can describe all observables in terms of weighted sums of Weyl operators.*

General Weyl systems can be defined by projective representations over even dimensional finite abelian groups [26, 27]. We use a physical motivation for our definition here. The systems we will consider are composed of finitely many qudits. Subequently, we will also allow systems of mixed dimensions, e.g. qubits and qutrits, and show that the dynamics of those systems essentially split into dynamics of the subsets of qudits of the same dimension. Furthermore, we will generalize our approach to infinitely many qudits. This approach gives us more structure in the group which will actually be a direct sum of even-dimensional vector spaces over finite fields.

Definition 2.2.2. *A projective representation of a group* $\mathcal{G}$ *is a function* w *from* $\mathcal{G}$ *to the unitary operators* $U(\mathcal{H})$ *on a Hilbert space* $\mathcal{H}$ *with the property*

$$\mathrm{w}(\xi \circ \eta) = f(\xi, \eta)\mathrm{w}(\xi)\mathrm{w}(\eta) \tag{2.28}$$

for all $\xi, \eta \in \mathcal{G}$ *and suitable phases* $|f(\xi,\eta)| = 1$*. Thus* w *preserves the group law up to a phase. The function* f *is called the* factor system*.*

We now come to the formal definition of Weyl systems over groups that are even dimensional vector spaces over finite fields. The generalization to direct sums of such vector spaces is straight forward and will be carried out in Section 2.2.2. We largely follow [28].

Definition 2.2.3 ([28]). *A Weyl system over a $\mathbb{F}$-vector space Ξ is a triple $(\mathrm{w}, f, \mathcal{H})$, consisting of a Hilbert space $\mathcal{H}$, a linear mapping $\mathrm{w}: \Xi \to \mathrm{U}(\mathcal{H})$ and a phase valued function $f: \Xi^2 \to \mathrm{U}(1)$, with*

$$\mathrm{w}(\xi+\eta) = f(\xi,\eta)\mathrm{w}(\xi)\mathrm{w}(\eta) \tag{2.29}$$

for all $\xi, \eta \in \Xi$. $\mathrm{U}(\mathcal{H})$ is the set of all unitary operators on $\mathcal{H}$.

For each Weyl system $(\mathrm{w}, f, \mathcal{H})$ we define the Weyl algebra $\mathfrak{A}(\mathrm{w}, f)$ as the closure of the set of all Weyl operators in the operator norm topology.

The Weyl algebra $\mathfrak{A}(\mathrm{w}, f)$ is a C^*-algebra. To every operator $\mathrm{w}(\xi)$ there is an adjoint operator $\mathrm{w}(\xi)^*$, with the *-mapping from $\mathscr{B}(\mathcal{H})$. It is given by

$$\mathrm{w}(\xi)^* = f(\xi,-\xi)\mathrm{w}(-\xi), \tag{2.30}$$

since

$$\begin{aligned} \mathrm{w}(\xi)\mathrm{w}(\xi)^* &= \mathbb{1} = f(\xi,-\xi)\mathrm{w}(\xi)\mathrm{w}(-\xi) \\ \Rightarrow \quad \mathrm{w}(\xi)^* &= f(\xi,-\xi)\mathrm{w}(-\xi). \end{aligned}$$

We now consider a system of n qudits, which are indexed by i, $i \in \Lambda$, $|\Lambda| = n$. As shown in Example 2.2.1 the single qubit phase space is $\mathbb{F}_2^2$. For a single qudit it is $\mathbb{F}_d^2$. The following construction only works for d prime, because we need the field structure of $\mathbb{F}_d$. For non-prime d the cyclic group $\mathbb{Z}_d$ is not a field. For the compositions of systems we use the tensor product of Hilbert spaces in the quantum case and the direct sum of phase spaces in the classical description. Thus the phase space for n qudits is given by $\widetilde{\Xi} = \mathbb{F}^{2n} = \mathbb{Z}_d^{2n} = \bigoplus_{i=1}^n \mathbb{Z}_d^2$, where d is prime.

The space of classical configurations is the vector space $\mathbb{F}^n$, which consist of all n-tuples $q^\Lambda = (q(i))^{i\in\Lambda}$. We can interpret the q^Λ as functions from Λ to $\mathbb{F}$. This allows us to write the phase space $\widetilde{\Xi} = \mathbb{F}^{2n}$ as $\Xi = \mathbb{F}^\Lambda \oplus \mathbb{F}^\Lambda$, because $\mathbb{F}^{2n}$ is isomorphic to $\mathbb{F}^\Lambda \oplus \mathbb{F}^\Lambda = \mathbb{F}^n \oplus \mathbb{F}^n$. The Hilbert space of the qudits Λ consist of all complex valued functions $\psi : \mathbb{F}^\Lambda \to \mathbb{C}$. The scalar product is defined via

$$\langle \psi_1, \psi_2 \rangle = \int \mathrm{d}q^\Lambda \overline{\psi_1(q^\Lambda)} \psi_2(q^\Lambda), \tag{2.31}$$

where

$$\int \mathrm{d}q^\Lambda \phi(q^\Lambda) := d^{-n} \sum_{q^\Lambda \in \mathbb{F}^\Lambda} \phi(q^\Lambda) \tag{2.32}$$

is the normalized sum. We denote this Hilbert space by $\mathscr{L}_2(\mathbb{F}^\Lambda)$. It can be decomposed into a product of one-qudit Hilbert spaces: $\mathscr{L}_2(\mathbb{F}^\Lambda) = \bigotimes_{i\in\Lambda} \mathscr{L}_2(\mathbb{F}^i) = \bigotimes_{i=1}^n \mathscr{L}_2(\mathbb{F})$. The one-qudit Hilbert spaces correspond to $\mathbb{C}^d$, therefore the whole Hilbert space is $\bigotimes_{i=1}^n \mathbb{C}^d$.

Now we associate a unitary operator, a Weyl operator, $\mathrm{w}(q^\Lambda, p^\Lambda)$ to every phase space vector $(q^\Lambda, p^\Lambda) \in \mathbb{F}^\Lambda \oplus \mathbb{F}^\Lambda$:

$$\mathrm{w}(q^\Lambda, p^\Lambda)\psi(a^\Lambda) := \chi(p^\Lambda, a^\Lambda)\psi(a^\Lambda - q^\Lambda) \tag{2.33}$$

The function χ assigns a phase to every pair $p^\Lambda, a^\Lambda \in \mathbb{F}^\Lambda$ of vectors using

$$\chi(p^\Lambda, a^\Lambda) := \varepsilon_d^{p^\Lambda \cdot a^\Lambda} \in \mathrm{U}(1), \tag{2.34}$$

where $\varepsilon_d = \exp(2\pi i d^{-1})$ is a dth root of identity and $p^\Lambda \cdot a^\Lambda = \sum_{i\in\Lambda} p_i^\Lambda a_i^\Lambda$ is the usual scalar product for vectors over finite fields, using addition and multiplication modulus d. A general Weyl operator consists of a part that acts as a discrete phase space translation and a part that adds a state dependent phase. In the qubit case this is exactly the action of X and Z. This is why $X = \mathrm{w}(1,0)$ and $Z = \mathrm{w}(0,1)$ are sometimes also used for Weyl operators on qudits.[4] The property $X^2 = Z^2 = \mathbb{1}$ is generalized to $X^d = Z^d = \mathbb{1}$ for qudits.

The operators defined above fulfill the relation (2.29), because for all $\psi \in \mathcal{H}$ and all $a^\Lambda \in \mathbb{F}^\Lambda$

$$\begin{aligned}
& \chi(p^\Lambda, s^\Lambda)\mathrm{w}(q^\Lambda, p^\Lambda)\mathrm{w}(s^\Lambda, r^\Lambda)\psi(a^\Lambda) \\
= \;& \chi(p^\Lambda, s^\Lambda)\mathrm{w}(q^\Lambda, p^\Lambda)\chi(r^\Lambda, a^\Lambda)\psi(a^\Lambda - s^\Lambda) \\
= \;& \chi(p^\Lambda, s^\Lambda)\chi(p^\Lambda, a^\Lambda - s^\Lambda)\chi(r^\Lambda, a^\Lambda)\psi(a^\Lambda - s^\Lambda - q^\Lambda) \\
= \;& \chi(p^\Lambda, a^\Lambda)\chi(r^\Lambda, a^\Lambda)\psi(a^\Lambda - s^\Lambda - q^\Lambda) \\
= \;& \chi(p^\Lambda + r^\Lambda, a^\Lambda)\psi(a^\Lambda - (q^\Lambda + s^\Lambda)) \\
= \;& \mathrm{w}(q^\Lambda + s^\Lambda, p^\Lambda + r^\Lambda)\psi(a^\Lambda)
\end{aligned}$$

holds. This provides us with a method to multiply Weyl operators by adding the corresponding phase space vectors

$$\chi(p^\Lambda, s^\Lambda)\mathrm{w}(q^\Lambda, p^\Lambda)\mathrm{w}(s^\Lambda, r^\Lambda) = \mathrm{w}(q^\Lambda + s^\Lambda, p^\Lambda + r^\Lambda), \tag{2.35}$$

in accordance with (2.29). To simplify further expressions we introduce the notation $\xi := (\xi_X, \xi_Z) = (q^\Lambda, p^\Lambda)$, $\eta := (\eta_X, \eta_Z) = (s^\Lambda, r^\Lambda)$ and $\varepsilon_d^{\beta(\xi,\eta)} := \chi(\xi_X, \eta_Z) = \chi(p^\Lambda, s^\Lambda)$, thus $\beta(p^\Lambda, q^\Lambda) = \sum_{i\in\Lambda} p_i^\Lambda q_i^\Lambda$. The triple

$$\left(\mathrm{w}, \varepsilon_d^\beta, \mathscr{L}_2(\mathbb{F}^\Lambda)\right) \tag{2.36}$$

forms a Weyl system. We refer to this as the Standard Weyl system. As we will see in the remainder of this section this triple preserves the locality structure. We therefore take this as our standard system throughout the thesis.

[4] The position of p and q in the phase space vectors is exchanged with respect to most of the literature. However, we continue our use of notation in order to maintain consistency with previously published sections of this thesis.

An interesting property of the Weyl operators are the commutation relations they satisfy. We use the Weyl relations (2.35) and the symplectic form

$$\sigma(\xi,\eta) = \beta(\xi,\eta) - \beta(\eta,\xi) \tag{2.37}$$

to determine the commutation relations via

$$\begin{aligned} \mathrm{w}(\xi)\mathrm{w}(\eta) &= \varepsilon_d^{-\beta(\xi,\eta)}\mathrm{w}(\xi+\eta) \\ &= (\varepsilon_d^{\beta(\xi,\eta)})^{-1}\varepsilon_d^{\beta(\eta,\xi)}\mathrm{w}(\eta)\mathrm{w}(\xi) \\ &= \varepsilon_d^{\sigma(\eta,\xi)}\mathrm{w}(\eta)\mathrm{w}(\xi) \end{aligned}$$

to be

$$\mathrm{w}(\xi)\mathrm{w}(\eta) = \varepsilon_d^{\sigma(\eta,\xi)}\mathrm{w}(\eta)\mathrm{w}(\xi). \tag{2.38}$$

Thus, two Weyl operators $\mathrm{w}(\xi)$, $\mathrm{w}(\eta)$ commute if and only if the symplectic form $\sigma(\xi,\eta)$ vanishes. They anti-commute if $\sigma(\xi,\eta) = d/2$ holds. Because d has to be prime, anti-commutation is only possible if $d = 2$. The symplectic form encodes the commutation relations. It will therefore play an important role when studying Clifford operations.

In total the Weyl system consists of d^{2n} operators. However, using the classical correspondence we only need $2n$ phase space vectors to describe all Weyl operators. The symplectic form allows us to compute the time evolution of all Weyl operators using only the time evolution of those operators which correspond to a basis of the phase space; of course only if the time evolution is of Clifford type. Every bounded operator in $\mathscr{B}(\mathcal{H})$ can be written as a sum of Weyl operators and therefore a standard Weyl system forms a basis of the algebra of all bounded operators $\mathcal{B}(\mathcal{H})$ on the Hilbert space $\mathcal{H} = \mathscr{L}_2(\mathbb{F}^\Lambda)$. Thus the standard Weyl system is irreducible. We denote the Weyl algebra of the standard Weyl system by

$$\mathfrak{A}(\Lambda) := \mathfrak{A}\left(\mathrm{w}, \varepsilon_d^\beta, \Lambda\right) = \mathrm{sp}\left\{\mathrm{w}(\xi) | \xi \in \Xi(\Lambda)\right\}. \tag{2.39}$$

The Weyl algebra $\mathfrak{A}(\Lambda)$ has a local structure. To every subset $\Upsilon \subset \Lambda$ we assign the local phase space $\Xi(\Upsilon) = \mathbb{F}^\Upsilon \oplus \mathbb{F}^\Upsilon$ and the local subalgebra $\mathfrak{A}(\Upsilon)$, which consists of all operators in $\mathfrak{A}(\Lambda)$ that are localized on Υ. By $\sup(\xi)$ we denote the support of ξ which is the set $\{x \in \Lambda | \xi(x) \neq 0\}$ where $\xi(x)$ is non-zero. The Weyl operators $\mathrm{w}(\xi)$ with $\sup(\xi) \subset \Upsilon$ again form a basis of $\mathfrak{A}(\Upsilon)$. Thus we can restrict any vector ξ with $\sup(\xi) \subset \Upsilon$ to $\Xi(\Upsilon)$ without any loss of information. On the other hand all vectors $\Xi(\Upsilon)$ can be extended to Ξ by setting $\xi(x) = 0$, $\forall x \in \Lambda \backslash \Upsilon$. In a similar way we extend operators from $\mathfrak{A}(\Upsilon)$ to operators on $\mathfrak{A}(\Lambda)$ by tensoring with identities:

$$A \in \mathfrak{A}(\Upsilon) \to \mathbb{1}^{\otimes \Lambda \backslash \Upsilon} \otimes A \in \mathfrak{A}(\Lambda) \tag{2.40}$$

The local algebras have the dimension $|(\mathbb{F}^\Upsilon)^2| = d^{2|\Upsilon|}$. They are are generated by $|\Upsilon|$ one-site algebras $\mathfrak{A}_j$, $j \in \Upsilon$. Each one-site algebra is isomorphic to the full matrix

algebra $\mathcal{M}_d(\mathbb{C})$ over the one-site Hilbert space $\mathbb{C}^d$ and thus has the dimension d^2. We build up the local algebra from the one-site algebras using the tensor product

$$\mathfrak{A}(\Upsilon) = \bigotimes_{j\in\Upsilon} \mathfrak{A}_j \cong \bigotimes_{j\in\Upsilon} \mathcal{M}_d(\mathbb{C}) = \mathcal{M}_d(\mathbb{C})^{\otimes\Upsilon}. \tag{2.41}$$

Thus a Weyl operator $\mathrm{w}(\xi)$ can be written as a tensor product of one-site Weyl operators

$$\mathrm{w}(\xi) = \bigotimes_{i} \mathrm{w}(\xi(i))^{(i)}. \tag{2.42}$$

The superscript (i) will often be omitted, since we are only interested in the relative position of the tensor factors with respect to each other, which can be represented in the following way:

Example 2.2.4. *For qubits the Weyl operators are (up to global phases) tensor products of Pauli matrices. Let $\xi(x) \in (\mathbb{F}^{\mathbb{Z}})^2$,*

$$\xi(x) = \left(\begin{array}{ccccccccc} x & = & \cdots & -2 & -1 & 0 & 1 & 2 & \cdots \\ \hline \xi_+ & = & \cdots & 0 & 1 & 1 & 0 & 0 & \cdots \\ \xi_- & = & \cdots & 0 & 0 & 1 & 1 & 0 & \cdots \end{array}\right).$$

The support of $\xi(x)$ is evidently the interval $[-1,1] \subset \mathbb{Z}$. Thus $\mathrm{w}(\xi(x))$ is localized on this interval and we only have to take tensor factors from this interval into account. On all other tensor factors, $\mathrm{w}(\xi(x))$ is just the identity. Then, using (2.42), we have

$$\begin{aligned} \mathrm{w}(\xi) &= \bigotimes_{x=-1}^{1} \mathrm{w}(\xi(x))^{(x)} \\ &= \mathrm{w}(1,0)^{(-1)} \otimes \mathrm{w}(1,1)^{(0)} \otimes \mathrm{w}(0,1)^{(1)} \\ &= \sigma_1^{(-1)} \otimes i\sigma_2^{(0)} \otimes \sigma_3^{(1)}. \end{aligned}$$

Another convenient property of standard Weyl systems is that if we are dealing with irreducible systems, all Weyl operators (excluding multiples of the identity) are tracefree.

Theorem 2.2.5. *For any irreducible standard Weyl system all Weyl operators except multiples of the identity are tracefree.*

Proof. We consider all possible decompositions of the Weyl operator $\mathrm{w}(\xi)$ into a product

$$\mathrm{w}(\xi) = \mathrm{w}(\eta)\mathrm{w}(\xi-\eta)\varepsilon_d^{\beta(\eta,\xi-\eta)} \tag{2.43}$$

for every phase space element η. Using the commutation relations together with the cyclic commutation of operators under the trace we have

$$\begin{aligned} & \operatorname{tr}(\mathrm{w}(\eta)\mathrm{w}(\xi-\eta)) &=& \operatorname{tr}(\mathrm{w}(\xi-\eta)\mathrm{w}(\eta))\varepsilon_d^{\sigma(\xi-\eta,\eta)} \\ \Leftrightarrow \quad & \sigma(\xi-\eta,\eta) &=& 0 \qquad \text{or} \\ & \operatorname{tr}(\mathrm{w}(\eta)\mathrm{w}(\xi-\eta)) &=& \varepsilon_d^{-\beta(\eta,\xi-\eta)}\operatorname{tr}(\mathrm{w}(\xi)) = 0 \end{aligned}$$

and therefore the condition

$$0 = \sigma(\xi-\eta,\eta) = \sigma(\xi,\eta) - \sigma(\eta,\eta) = \sigma(\xi,\eta) \quad \forall \eta \in \Xi \tag{2.44}$$

for all Weyl operators that are not traceless. Because the symplectic form is not degenerate (we have an irreducible system), Equation (2.44) can only be fulfilled by $\xi = 0$. Therefore, only multiples of the identity can have a non-zero trace. □

A more general definition of the Weyl algebra which works also in the case where the phase space Ξ does not correspond to a system of qudits is the following:

$$\mathfrak{A}(\Xi) = \operatorname{sp}\{\mathrm{w}(\xi) | \xi \in \Xi\}. \tag{2.45}$$

2.2.1 Commutant and center of a Weyl algebra

Let us first consider some questions regarding commutants of Weyl systems and their relations to symplectic complements and maximally isotropic subspaces. We begin with some definitions.

Definition 2.2.6. *The symplectic complement M^σ of a space $M \subset \Xi$ is defined by*

$$M^\sigma = \{\xi \in \Xi | \forall \eta \in M : \sigma(\xi,\eta) = 0\}. \tag{2.46}$$

The radical $\operatorname{rad} M$ *of M is defined by*

$$\operatorname{rad} M = \{\xi \in M | \sigma(\xi,\eta) = 0, \forall \eta \in M\} = M \cap M^\sigma. \tag{2.47}$$

Let $\dim M = k$ *and* $\operatorname{rank}\sigma_{|M} = 2l$*, then* $\dim \operatorname{rad} M = \dim M - \operatorname{rank}\sigma_{|M} = k - 2l$*. The symplectic space associated to M is defined by*

$$M^{\mathrm{red}} = M/\operatorname{rad} M. \tag{2.48}$$

Every space N that fulfills $M = \operatorname{rad} M \oplus N$ is symplectic with respect to $\sigma_{|N}$ and isomorphic to M^{red}. For details see e.g [29, 30].

Now let us prove the following:

Theorem 2.2.7. *Let Ξ be a symplectic space of dimension $2n$ and let M be a subspace of Ξ. Then the following holds.*

1. $\dim(M) + \dim(M^\sigma) = 2n = \dim\Xi$
2. $M^{\sigma\sigma} = M$
3. $\mathfrak{A}(M)' = \mathfrak{A}(M^\sigma)$, *where* $\mathfrak{A}(M)'$ *denotes the commutant of* $\mathfrak{A}(M)$ *in* $\mathfrak{A}(\Xi)$.
4. $\mathfrak{A}(M \cap N) = \mathfrak{A}(M) \cap \mathfrak{A}(N)$
5. $\mathfrak{A}(M) \cong \mathfrak{A}(\operatorname{rad} M) \otimes \mathfrak{A}(M^{\mathrm{red}})$

Proof.

1. This hold for all symmetric, alternating or hermitian bilinear forms iff they are non-degenerate (see e.g. [31], Chapter XV, Proposition 1.2.)

2. $M \subset M^{\sigma\sigma}$: Let $\xi \in M$ and $\eta \in M^\sigma$. Then $\sigma(\xi,\eta) = 0$. As η is arbitrary in M^σ $\xi \in M^{\sigma\sigma}$ and $M \subset M^{\sigma\sigma}$.

   $M^{\sigma\sigma} \subset M$: Let $\xi \in M^{\sigma\sigma}$. ξ has a decomposition in terms of M and its orthogonal space: $\xi = \xi_M + \xi_{M^\perp}$, $\xi_M \in M$ and $\xi_{M^\perp} \in M^\perp$. We already know that $M \subset M^{\sigma\sigma}$. Thus $\xi_M \in M^{\sigma\sigma}$ and $\xi_{M^\perp} \in M^{\perp\sigma\sigma}$. As we will show in the next paragraph $M^{\perp\sigma\sigma} \cap M^{\sigma\sigma} = \{0\}$ and therefore $\xi^\perp = 0$ and $\xi \in M$.

   First we need to prove $M^{\perp\sigma} \cap M^\sigma = \{0\}$: Let $\xi \in M^\sigma$ and $\xi \in M^{\perp\sigma}$. Then $\sigma(\xi,\eta) = 0$, $\forall \eta \in M$ and $\sigma(\xi,\zeta) = 0$, $\forall \zeta \in M^\perp$. M and $M^\perp$ span Ξ and thus, by linearity of σ, $\sigma(\xi,\eta) = 0$, $\forall \eta \in \Xi$ and therefore $\xi = 0$ as σ is non-degenerate. M^σ and $M^{\perp\sigma}$ again span the whole space Ξ, as shown in 1. $\dim(M^\sigma) + \dim(M^{\perp\sigma}) = \dim(\Xi) - \dim(M) + \dim(\Xi) - \dim(M^\perp) = \dim(\Xi)$ and $M^\sigma \cap M^{\perp\sigma} = \{0\}$. Using the exact same reasoning we can now show that $M^{\sigma\sigma} \cap M^{\perp\sigma\sigma} = \{0\}$.

3. The commutant of $\mathfrak{A}(M)$ is $\mathfrak{A}(M)' = \{A \in \mathfrak{A}(\Xi) | \forall B \in \mathfrak{A}(M) : [A,B] = 0\}$. It suffices to evaluate this for an algebraic basis of $\mathfrak{A}(M)$ so $B = \mathrm{w}(\eta_j)$, where η_j form a basis of M. A is arbitrary: $A = \sum_i a_i \mathrm{w}(\xi_i)$, where ξ_i are a basis of Ξ.

$$[A,B] = \sum_i a_i \mathrm{w}(\xi_i + \eta_j) \mathrm{e}^{-i\beta(\xi_i,\eta_j)} \left(1 - \mathrm{e}^{i\sigma(\xi_i,\eta_j)}\right)$$

   $\mathrm{w}(\xi_{i_1} + \eta_j) = \mathrm{w}(\xi_{i_2} + \eta_j)$ implies $\xi_{i_1} = \xi_{i_2}$, so all summands have to vanish individually and $\left(1 - \mathrm{e}^{i\sigma(\xi_i,\eta_j)}\right) = 0$, $\forall \eta_j \in M$. This is equivalent to $\sigma(\xi_i,\eta_j) = 0$, $\forall \eta_j \in M$ and thus holds iff $\xi_i \in M^\sigma$. Therefore $\mathfrak{A}(M)' = \mathfrak{A}(M^\sigma)$.

4. $\mathfrak{A}(M \cap N) \subset \mathfrak{A}(M) \cap \mathfrak{A}(N)$: Let $A \in \mathfrak{A}(M \cap N)$, then $A = \sum_i a_i \mathrm{w}(\xi_i)$, $\xi_i \in M \cap N$. Therefore $A \in \mathfrak{A}(M)$ and $A \in \mathfrak{A}(N)$; thus $A \in \mathfrak{A}(M) \cap \mathfrak{A}(N)$.

$\mathfrak{A}(M) \cap \mathfrak{A}(N) \subset \mathfrak{A}(M \cap N)$: $\mathfrak{A}(X)$ is defined as the span of all Weyl operators over X (see Equation 2.39). It is sufficient to check if all Weyl operators of $\mathfrak{A}(M) \cap \mathfrak{A}(N)$ are contained in $\mathfrak{A}(M \cap N)$. Let $A \in \mathfrak{A}(M) \cap \mathfrak{A}(N)$ and let A be a Weyl operator. Then we have $A = \mathrm{w}(\xi)$, $\xi \in M$ and $A = \mathrm{w}(\eta)$, $\eta \in N$. But as σ is non-degenerate $\mathrm{w}(\xi) = \mathrm{w}(\eta)$ implies $\xi = \eta$ and therefore $\xi \in M \cap N$ and $A \in \mathfrak{A}(M \cap N)$.

5. We have $M \cong \operatorname{rad} M \oplus M^{\mathrm{red}}$ and therefore $\mathfrak{A}(M) \cong \mathfrak{A}(\operatorname{rad} M) \otimes \mathfrak{A}(M^{\mathrm{red}})$.

□

It is now easy to determine the center $\mathfrak{C}(\mathfrak{A}(M))$ of a Weyl algebra $\mathfrak{A}(M)$ in terms of the underlying phase space:

$$\begin{aligned} \mathfrak{C}(\mathfrak{A}(M)) &= \mathfrak{A}(M) \cap \mathfrak{A}(M)' \\ &= \mathfrak{A}(M) \cap \mathfrak{A}(M^{\sigma}) \\ &= \mathfrak{A}(M \cap M^{\sigma}) \\ &= \mathfrak{A}(\operatorname{rad} M). \end{aligned}$$

The center $\mathfrak{C}(\mathfrak{A}(M))$ of the Weyl algebra $\mathfrak{A}(M)$ is the Weyl algebra $\mathfrak{A}(\operatorname{rad} M)$ of the radical of M.

2.2.2 Weyl systems for qudits of different dimensions

When studying Clifford operations we do not want to restrict ourselves to systems that are composed of only one kind of subsystem. Instead we want to allow combinations of qudits with different dimensions d. Then the phase space is no longer a vector space Ξ over a field $\mathbb{F}$, but the direct sum of vector spaces Ξ_i over different finite fields $\mathbb{F}_i$. Ξ is still an even-dimensional additive abelian group. As addition is the only operation we need for the phase space representation of Weyl systems, we can still define a Weyl system over Ξ. We then have the Weyl relations

$$\mathrm{w}(\xi)\mathrm{w}(\eta) = \mathrm{e}^{-i\beta(\xi,\eta)}\mathrm{w}(\xi+\eta), \tag{2.49}$$

$$\mathrm{w}(\xi)\mathrm{w}(\eta) = \mathrm{e}^{i\sigma(\eta,\xi)}\mathrm{w}(\eta)\mathrm{w}(\xi) \quad \text{and} \tag{2.50}$$

$$\mathrm{w}^*(\xi) = \mathrm{e}^{-i\beta(\xi,\xi)}\mathrm{w}(-\xi). \tag{2.51}$$

β and σ are defined differing from (2.36) and (2.37) as

$$\beta(\xi,\eta) = 2\pi \sum_i \frac{\xi(i)_X \eta(i)_Z}{d(i)} \quad \text{and} \tag{2.52}$$

$$\sigma(\xi,\eta) = \beta(\xi,\eta) - \beta(\eta,\xi). \tag{2.53}$$

$d(x)$ is the internal dimension of site x. All other results are valid analogous to the case of only one kind of qudits.

In Section 2.2.4 we will show that systems of different qudits are actually of limited interest for Clifford dynamics, as the dynamics will essentially split into dynamics on systems with the same dimension.

2.2.3 Weyl systems for infinitely many qudits

When studying Clifford quantum cellular automata (CQCAs) we will be mostly concerned with infinite lattices of qudits ($\mathbb{Z}^s$). Cellular automata are local operations. They map localized observables to localized observables. Thus we can use the quasi-local algebra on the lattice of qudits that can be constructed from local Weyl algebras in the way introduced for general quasi-local algebras in Section 2.1.3.

2.2.4 Homomorphisms of phase spaces

Clifford time evolutions are described by linear maps on phase space. It is thus important to study the properties of these phase space homomorphisms. We will show in this section that general homomorphisms between two phase spaces that are both direct sums of vector spaces over finite fields of prime order split into directs sums of homomorphisms that only act on the parts of the phase spaces that have the same underlying finite field. In the quantum picture this means that the qubits on the input of a channel might only influence the qubits on the output system but e.g. not qutrits on the output part (except for a global phase). To prove this we need some results from finite group theory.

A *torsion abelian group* $\mathcal{G}$ is a group in which all elements g have *finite period*, that is $g^p = 0$ for some finite $p(g)$. This period is called the *order* of an element. Given a torsion abelian group $\mathcal{G}$ we denote by $\mathcal{G}(p)$ the subgroup of elements of order p, for any prime p. If $\mathcal{G}(p)$ is finite it is a *p-group*, a finite group in which all elements except 0 have a period that is a power of p. A fundamental theorem of group theory states that every torsion abelian group $\mathcal{G}$ is (isomorphic to) the direct sum of its subgroups $\mathcal{G}(p)$ for all primes p such that $\mathcal{G}(p) \neq 0$ (see e.g. [31], I.8).

If a group $\mathcal{G}$ is a direct sum of several groups $\mathcal{G}_i$ than each of these groups $\mathcal{G}_i$ is a homomorphic image of the whole group $\mathcal{G}$ under the homomorphism $\pi_i : \mathcal{G} \to \mathcal{G}_i$, $\pi_i(g) = \pi_i(g_1 \oplus g_2 \dots g_n) = \pi_i$. h_i is surely a homomorphism, because $\pi_i(g+l) = g_i + l_i = \pi_i(g) + \pi_i(l)$.

Theorem 2.2.8. *Let $\mathcal{G}$ and $\mathcal{L}$ be torsion abelian groups and let $h : \mathcal{G} \to \mathcal{L}$ be a homomorphism. Then h is the direct sum of homomorphisms $h_p : \mathcal{G}(p) \to \mathcal{L}(p)$ for all primes p which fulfill $\mathcal{G}(p) \neq 0$ and $\mathcal{L}(p) \neq 0$. Subgroups $\mathcal{G}(p) \neq 0$ with $\mathcal{L}(p) = 0$ are sent to 0 (they are always in the kernel of h). On subgroups $\mathcal{L}(p) \neq 0$ with $\mathcal{G}(p) = 0$ we extend the image of $\bigoplus_p h_p(g)$ by 0.*

Proof. The groups $\mathcal{G}$ and $\mathcal{L}$ can be decomposed into p-groups:

$$\mathcal{G} \cong \bigoplus_p \mathcal{G}(p) \qquad \mathcal{L} \cong \bigotimes_q \mathcal{L}(q).$$

An element of $g \in \mathcal{G}$ can be decomposed as $g \cong \bigoplus_p g_p$ with $g_p \in \mathcal{G}(p)$, and $l \in \mathcal{L}$ analogously.

We construct homomorphisms $h_{p,q} : \mathcal{G}(p) \to \mathcal{L}(q)$ by restricting the input of h to $\mathcal{G}(p)$ and projecting to $\mathcal{L}(q)$ using the homomorphism π_q and obtain $h_{p,q}(g_p) = \pi_q \circ h(g_p \oplus 0_{\mathcal{G}(r),\, r\neq p})$. This construction gives us $h(g_p \oplus 0_{\mathcal{G}(r),\, r\neq p}) = \bigoplus_q h_{p,q}(g_p)$ and $\mathrm{h}(g) = \sum_p h(g_p \oplus 0_{\mathcal{G}(r),\, r\neq p})$. We will now show that $h_{p,q}(\mathcal{G}(p)) = 0$ whenever $p \neq q$ and thus $h_{p,q}$ is the trivial homomorphism for $p \neq q$.

$h_{p,q}(\mathcal{G}(p))$ is a subgroup of $\mathcal{L}(q)$ (see e.g. [32], chapter I). $\mathcal{L}(q)$ is of order q^n for some n. By Lagranges' theorem all subgroups of $\mathcal{L}(q)$ are of order q^m with $m < n$. All subgroups are also q-groups. On the other hand all elements $g_p \neq 0$ of $\mathcal{G}(p)$ are of an order of a power of p bounded by p^x, where p^x is the order of the group $\mathcal{G}(p)$. The image of $h_{p,q}(g_p)$ has an order a that divides p^x: We have $p^y g_p = 0$, $y \leqslant x$ and $a\, h_{p,q}(g_p) = 0$. Because $h_{p,q}$ is a homomorphism we also have $p^y \cdot h_{p,q}(g_p) = 0$ and thus $p^y = ka$, $k \in \mathbb{N}$ and a is a divisor of p^x. So the order a of $h_{p,q}(g_p)$ has to divide both p^x and q^m and therefore $a = p^i = q^j$, $i \leqslant y$, $j \leqslant m$. As the logarithms of primes are rationally independent (follows from the unique-prime-factorization theorem) this can only be true if either $p = q$ or $a = 1$. So whenever $p \neq q$ $h_{p,q}$ is the trivial homomorphism that maps everything to 0.

The sum $h(g) = \sum_p h(g_p \oplus 0_{\mathcal{G}(r),\, r\neq p})$ simplifies to $h(g) = \bigoplus_p h_{p,p}(g_p)$ and the theorem follows. □

In the case of phase spaces the p-groups are always of the form $\mathbb{Z}_p^{2n}$. We will use the above result to investigate the structure of Clifford channels in Section 3.6.

2.2.5 The symplectically adjoint map

When studying channels between Weyl systems certain conditions on the symplectic forms of the systems have to be fulfilled. It is then sometimes necessary to shift the action of the homomorphism on the phase space from the argument of one symplectic form to the argument of the other symplectic form. We have just seen that homomorphisms between phase spaces split into homomorphisms between spaces of the same underlying field. We can therefore restrict our studies to the case of phase spaces that are actually vector spaces over finite fields and not direct sums of vector spaces.

Lemma 2.2.9. *Let $h : \Xi_2 \to \Xi_1$ be a homomorphism connecting the phase spaces Ξ_2 and Ξ_1 with $h(\Xi_2) \subset \Xi_1$. Furthermore let σ_1 and σ_2 be the standard symplectic forms*

on Ξ_1 and Ξ_2 respectively. Then the symplectically adjoint map h^σ with respect to σ_1 and σ_2 is uniquely defined by

$$\sigma_2(\xi, h^\sigma \eta) = \sigma_1(h\xi, \eta) \quad \forall \eta \in \Xi_1,\, \xi \in \Xi_2. \tag{2.54}$$

Proof.

existence: The map $\xi \mapsto f(\xi) = \sigma_1(h\xi, \eta)$ is linear. It has the form $f(\xi) = m \cdot \xi_X + n \cdot \xi_Z$ with m and n taken from Ξ_2 and determined by the images of basis vectors. Furthermore σ_2 in non-degenerate. Then there is a unique $\eta' \in \Xi_1$ with $\sigma_2(\xi, \eta') = f(\xi)$, $\forall \xi$, namely $\eta' = (-n, m)$. The map $\eta \mapsto h^\sigma(\eta) = \eta'$ is the symplectically adjoint map of h. As σ_1 and σ_2 are additive in the second argument, h^σ is additive, too.

Uniqueness: Assume there are two different symplectically adjoint maps h^σ and k^σ of h. Then we have

$$\sigma_2(\xi, h^\sigma \eta) = \sigma_1(h\xi, \eta) = \sigma_2(\xi, k^\sigma \eta)$$

and therefore

$$\begin{aligned} \sigma_2(\xi, h^\sigma \eta) - \sigma_2(\xi, k^\sigma \eta) &= 0 \\ \sigma_2(\xi, h^\sigma \eta - k^\sigma \eta) &= 0 \quad \forall \xi \in \Xi_2,\, \eta \in h(\Xi_2). \end{aligned}$$

But σ_2 is non-degenerate, thus this can only be fulfilled if $h^\sigma \eta = k^\sigma \eta$, $\forall \eta \in \Xi_1$. But then $h^\sigma = k^\sigma$ and so the symplectically adjoint map of h is uniquely defined.

□

The existence of a symplectically adjoint map for the general case follows immediately: as h decomposes into homomorphisms that only act on subsystems of the same dimension we can write $\sigma_1(h\xi, \eta) = \sum_p \sigma_{1,p}(h_p \xi_p, \eta_p)$. We can use the symplectically adjoint maps on the subsystems and use them to define the global symplectically adjoint map:

$$\sigma_2(\xi, h^\sigma \eta) = \sum_p \sigma_{2,p}(\xi_p, h_p^\sigma \eta_p). \tag{2.55}$$

2.2.6 Laurent polynomials

Phase space calculations can still be simplified using Laurent polynomials to describe the phase space vectors. We transform the binary strings to Laurent polynomials using

$$\hat{\xi}(u) = \sum_{x \in \mathbb{Z}} \xi(x) u^x. \tag{2.56}$$

We use these polynomials as abstract objects without ever evaluating them for some specific value of u. The transformation is illustrated with the following example:

$$\begin{pmatrix} \cdots & 0 & 1 & \underline{1} & 0 & 1 & 0 & \cdots \\ \cdots & 0 & 0 & \underline{1} & 1 & 0 & 0 & \cdots \end{pmatrix} \mapsto \begin{pmatrix} u^{-1}+1+u^2 \\ 1+u \end{pmatrix}.$$

We change the function w, introduced earlier, in a way that it maps vectors of Laurent polynomials to tensor products of Pauli matrices. By a slight abuse of notation we will also name it w.

2.3 Stabilizer states

A *stabilizer state* ω is a common eigenstate to a group of commuting operators $\mathcal{S} = \langle\{S_i\}\rangle$. This means that $\omega \circ S = \omega \ \forall S \in \mathcal{S}$. The group is generated by the set of generators $\mathbb{S} = \{S_i\}$ by multiplication. It therefore suffices to check the stabilizer condition $\omega \circ S = \omega$ for the generators S_i. For finitely many qubits this can be easily understood with the following example:

Example 2.3.1. *The stabilizer group $\mathcal{S} = \langle \sigma_1 \otimes \sigma_1, \sigma_3 \otimes \sigma_3 \rangle$ stabilizes the Bell state $\psi = 1/\sqrt{2}(|1,1\rangle + |0,0\rangle)$. We check this by applying the stabilizer generators to the state (normalization is omitted):*

$$\begin{aligned} (\sigma_1 \otimes \sigma_1)\psi &= (\sigma_1 \otimes \sigma_1)(|1\rangle \otimes |1\rangle) + (\sigma_1 \otimes \sigma_1)(|0\rangle \otimes |0\rangle) \\ &= |0,0\rangle + |1,1\rangle = \psi \\ (\sigma_3 \otimes \sigma_3)\psi &= (\sigma_3 \otimes \sigma_3)(|1\rangle \otimes |1\rangle) + (\sigma_3 \otimes \sigma_3)(|0\rangle \otimes |0\rangle) \\ &= (-|1\rangle) \otimes (-|1\rangle) + |0\rangle \otimes |0\rangle \\ &= (-1)^2 |1,1\rangle + |0,0\rangle = \psi \end{aligned}$$

We now turn our attention to translation-invariant stabilizer states. For infinitely many qubits we cannot explicitly write down the state, so we use an abstract definition. A translation-invariant stabilizer state is defined by a translation-invariant set of operators $\mathbb{S} = \{\mathrm{w}(\hat{\tau}^x \xi),\, x \in \mathbb{Z}\}$ where $\hat{\tau}^x$ is the lattice translation by x sites. In [33] it was proven that such a set defines a pure translation-invariant stabilizer state if and only if ξ is reflection invariant and the Laurent polynomials of ξ have no common (non-monomial) divisors, i.e. $\gcd(\xi_X, \xi_Z) = 1$. This imposes the following conditions on the structure of the polynomials:

1. ξ is of odd length, because reflection invariant Laurent polynomials of even length are always divisible by $(1+u)$. We will write $l = 2n+1$.
2. $\xi(0) \neq \binom{0}{0}$, thus the center element is not $\mathbb{1}$. Else ξ would have the divisor $(u^{-1}+u)$.

3. At least two different types of Pauli matrices (both different from the identity) have to occur (e.g. X and Y). Else $\xi_+ = 0$ or $\xi_- = 0$ or $\xi_+ = \xi_-$, each case implying common divisors.

For a stabilizer generator $S_i = \sigma_1^{i-1} \otimes \sigma_3^i \otimes \sigma_1^{i+1}$ the corresponding phase space vector is $\xi = \binom{101}{010}$.

Stabilizer states are connected to maximally isotropic subspaces of the phase space in the sense that a stabilizer state is fixed by a set of commuting Weyl operators. In the phase space description commutation relations are encoded in the symplectic form $\sigma(\xi,\eta)$. If $[\mathrm{w}(\xi),\mathrm{w}(\eta)] = 0$ we have $\sigma(\xi,\eta) = 0$. Thus abelian algebras of Weyl operators correspond to subspaces on which the symplectic form vanishes. Those subspaces are called *isotropic subspaces*. If an isotropic subspace $\mathcal{I}$ has the additional property that $\sigma(\eta,\xi) = 0$, $\forall \xi \in \mathcal{I}$ implies $\eta \in \mathcal{I}$, we call $\mathcal{I}$ *maximally isotropic* (all vectors for which σ vanishes on $\mathcal{I}$ are contained in $\mathcal{I}$). Maximally isotropic subspaces correspond to maximally abelian algebras. In [33] it was shown that the above condition on ξ (reflection invariance and $\gcd(\xi_X,\xi_Z) = 1$) is equivalent to the condition that $\mathcal{P}\xi$ is a maximally isotropic subspace. $\mathcal{P}\xi$ denotes the space generated by the products of ξ and all elements of $\mathcal{P}$. We have $\langle \mathrm{w}(\eta), \eta \in \mathcal{P}\xi\rangle = \langle \mathrm{w}(\hat{\tau}^x\xi), x \in \mathbb{Z}\rangle = \mathcal{S}$. $\mathcal{P}\xi$ is the phase space of the stabilizer group $\mathcal{S}$.

Finally let us introduce the notion of minimal stabilizer generators. The minimal stabilizer generators are those stabilizer operators that generate the stabilizer group (through multiplication) while having the smallest support of all such sets of operators. There are in general many such sets, so the choice is not unique. For example, the stabilizer generators $\sigma_3 \otimes \mathbb{1}$ and $\mathbb{1} \otimes \sigma_3$ would stabilize the product state $\psi = |\uparrow\rangle \otimes |\uparrow\rangle$. Obviously the combination $\sigma_3 \otimes \sigma_3$ and $\mathbb{1} \otimes \sigma_3$ would do the same, but with one generator with larger support. In this case the first set is minimal, while the second is not. In the case of translation-invariant pure stabilizer states we do not have this issue. The set of generators is always translation invariant and fulfills the conditions introduced above. Assume that a given generator of a translation invariant pure stabilizer state is not minimal. Then it is composed of at least two stabilizer generators that must also fulfill the above conditions. In particular, all of the generators have to be the same. This implies that the polynomials of the original non-minimal generator have common divisors. But this is not possible, because it is required that the polynomials are coprime. Thus the generators of translation invariant pure stabilizer states are always minimal.

3 Basic concepts

In this chapter we will introduce the concepts that form the foundation for the main results of this thesis—from the basic setting of operations on a chain of quantum systems to the theory of Clifford memory channels. We begin with the introduction of the neighborhood calculus for local operations on a chain of quantum systems. This will be our basis to introduce the algebraic formalism for quantum cellular automata That will be employed in Chapter 5 to study Clifford quantum cellular automata. Furthermore, we prove bounds on the entanglement generation of general QCAs (Section 3.2.1). We proceed in Section 3.3 with the introduction of causal operations to describe processes that transfer information only in one direction. Causal operations with a finite range of information transfer are QCAs, therefore we can use the QCA formalism for their description. Resource requirements of quantum operations, which will be a basis of the memory analysis for the implementation of causal operations by memory channels in Chapter 4 and the performance comparison of convolutional and block codes under resource constraints in Chapter 7, will be briefly introduced in Section 3.4.

In the following sections (3.5–3.8) quantum channels with memory and Clifford channels will be introduced. We will start with the concept of quantum memory channels and their description by completely positive maps. As a special class of quantum channels we will study Clifford channels which map tensor products of Pauli matrices to multiples of tensor products of Pauli matrices. They are classically simulable and we will introduce their description by symplectic matrices over finite fields. We will introduce an algorithm to complete partially defined reversible Clifford operations (Section 3.6.2) which we will later use to find encoders of quantum error-correction codes. An algorithm to decompose a Clifford operation into series quantum gates will be presented in Section 3.6.3. We will continue with Clifford channels with memory which can be used to implement the encoding operation of quantum convolutional codes. After establishing the formalism we prove a forgetfulness criterion and furthermore show that every forgetful reversible Clifford channel is strictly forgetful. Finally we will show how to reduce the resource requirements of Clifford operations using a decomposition into a causal structure of smaller operations.

The chapter will be concluded with two sections covering the basics of general quantum error-correction and convolutional codes. After introducing the concept of block oriented error-correction codes in Section 3.9.1 we will show that Clifford operations alone are not enough to correct arbitrary single qubit errors. In Sec-

tion 3.9.3 we will introduce stabilizer codes which use Clifford operations as far as possible and show how they can be described with Clifford channels. Convolutional codes generalize block codes to allow information transfer between the individual steps of the encoding (Section 3.10). This model is believed to have certain benefits over block codes which we briefly reflect in Section 3.10.1. Finally, we will introduce the standard formalism for convolutional stabilizer codes leading to a description in terms of matrices over the ring of Laurent polynomials. Using example codes we will furthermore comment on the shortcomings in this formalism in finding encoded Pauli matrices of finite length.

As this chapter is an introductory chapter a lot of the material is widely known in the field. The presentation of those results will often be based on existing literature. However, some results are unpublished up to now respectively published only in a diploma thesis [34] or internal reports [35, 36], while other results are from recent papers.

3.1 Neighborhood of local operations on a chain

We will often be concerned with locality properties of operations on a chain of quantum systems. We will express these locality properties in terms of *neighborhoods* of the operations. First let us define our understanding of a local operation.

Definition 3.1.1. *An operation $T:\ \mathfrak{B}(\mathbb{Z}) \to \mathfrak{A}(\mathbb{Z})$ from a chain of quantum system $\mathfrak{A}(\mathbb{Z}) = \bigotimes_{i\in\mathbb{Z}} \mathfrak{A}(\mathbb{Z})_i$ to another chain of quantum systems $\mathfrak{B} = \bigotimes_{i\in\mathbb{Z}} \mathfrak{B}_i$ is called local, if and only if $\left[T\left[A_{[m,n]}\right], B_{[k,l]}\right] = 0$ for any two observables $A_{[m,n]} \in \bigotimes_{i=m}^{n} \mathfrak{A}_i$ and $B_{[k,l]} \in \bigotimes_{j=k}^{l} \mathfrak{A}_j$ if $\min(|m-k|, |n-l|)$ is sufficiently large, i.e. the observables are localized sufficiently far apart. This only means that $T[A]$ can only have an influence on a finite region around A.*

The neighborhood of a local operation is defined as follows.

Definition 3.1.2. *Let T be a local operation from a chain of quantum systems $\mathfrak{A}(\mathbb{Z}) = \bigotimes_{i\in\mathbb{Z}} \mathfrak{A}_i$ to a chain of quantum systems $\mathfrak{B}(\mathbb{Z}) = \bigotimes_{j\in\mathbb{Z}} \mathfrak{B}_j$. Then the neigborhood $\mathcal{N}(T)$ of T is the set of pairs $(i,j) \in \mathbb{Z}\times\mathbb{Z}$ such that for some $A_i \in \mathfrak{A}_i$ and $B_j \in \mathfrak{B}_j$ we have $\left[A_i, T\left[B_j\right]\right] \neq 0$:*

$$(i,j) \in \mathcal{N}(T) \Leftrightarrow \exists A_i \in \mathfrak{A}_i, B_j \in \mathfrak{B}_j :\ \left[A_i, T\left[B_j\right]\right] \neq 0. \tag{3.1}$$

This means that observables localized on output cell j can have an influence on input cell i, or—in the Schrödinger picture sense—that input on cell i can have an influence on the output on cell j. Equivalently we can use

$$(i,j) \notin \mathcal{N}(T) \Leftrightarrow \left[\mathfrak{A}_i, T\left[\mathfrak{B}_j\right]\right] \subset \{0\}. \tag{3.2}$$

Figure 3.1 shows an example neighborhood.

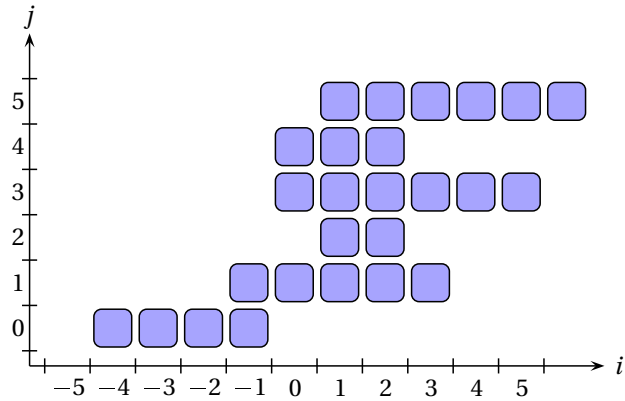


Figure 3.1: Neigborhood of a local operation on a chain of quantum systems; in each step the image of cell y is localized in those cells x marked by a blue box.

Using the above definition we can easily describe the neighborhood of compositions or inverses of operations. Let $\Lambda \subset \mathbb{Z}$ and let $\mathcal{N}_1, \mathcal{N}_2 \subset \mathbb{Z} \times \mathbb{Z}$ be neighborhood relations. We define the following operations:

$$\mathcal{N}_1 \circ \Lambda := \{i | \exists j \in \Lambda, (i,j) \in \mathcal{N}_1\}, \tag{3.3}$$

$$\mathcal{N}_1 \circ \mathcal{N}_2 := \{(i,k) | \exists j, (i,j) \in \mathcal{N}_1 \wedge (j,k) \in \mathcal{N}_2\}, \tag{3.4}$$

$$\mathcal{N}_1^{-1} := \{(j,i) | (i,j) \in \mathcal{N}_1\}. \tag{3.5}$$

To simplify notation we will write $\mathcal{N} \circ i = \mathcal{N} \circ \{i\} = \{j | (i,j) \in \mathcal{N}\}$.

Now we restrict ourselves to automorphisms. We require $\mathfrak{B}(\mathbb{Z}) = \mathfrak{A}(\mathbb{Z})$ and $\mathfrak{B}_i = \mathfrak{A}_i$. The following results for the neighborhoods of compositions of automorphisms will prove to be useful for our analysis of causal operations [16]:

Lemma 3.1.3. *Let T_1, T_2 be automorphisms of a quasi local algebra $\mathfrak{A}(\mathbb{Z})$. Then*

1. *$\mathcal{N}(T_1) \circ j$ is the smallest set $\lambda \subset \mathbb{Z}$ such that $T(\mathfrak{A}_j) \subset \mathfrak{A}(\Lambda)$,*
2. *$\mathcal{N}(T_1 \circ T_2) \subset \mathcal{N}(T_1) \circ \mathcal{N}(T_2)$,*
3. *$\mathcal{N}(T_1^{-1}) = \mathcal{N}(T_1)^{-1}$.*

Proof.

1. We have $i \notin \mathcal{N}(T) \circ j$ if and only if $(i,j) \notin \mathcal{N}(T)$ which is in turn equivalent to $\left[A_i, T[A_j]\right] = 0$ for all $A_i \in \mathfrak{A}_i$ and all $A_j \in \mathfrak{A}_j$. Now we can use Lemma 2.1.4

with $\Lambda = (\mathcal{N}(T) \circ j)^c$ and obtain $T[\mathfrak{A}_y] \subset \mathfrak{A}(\mathcal{N}(T) \circ j)$. Now we have to show minimality. Take an arbitrary set Λ' that fulfills $T[\mathfrak{A}_j] \subset \mathfrak{A}(\Lambda')$. Let $i \notin \Lambda'$. Then $[\mathfrak{A}_i, \mathfrak{A}(\Lambda')] = \{0\}$ and therefore $[\mathfrak{A}_i, \mathfrak{A}(\Lambda')] = \{0\}$. But this means that $i \notin \mathcal{N}(T) \circ j$ and therefore $\mathcal{N}(T) \circ j \subset \Lambda'$. Thus $\mathcal{N}(T) \circ j$ is the minimal set Λ such that $T(\mathfrak{A}_i) \subset \mathfrak{A}(\Lambda)$.

2. Let $A_i \in \mathfrak{A}_i$, then $T_2[A_i] \in \mathfrak{A}(\mathcal{N}(T_2) \circ i)$ and $T_2[A_i]$ can be approximated by sums of tensor products of elements of $\mathfrak{A}_j$, $j \in \mathcal{N}(T_2) \circ i$. Applying T_1 to the approximation gives us a sum of products localized in $\mathcal{N}(T_1) \circ j$, $j \in \mathcal{N}(T_2) \circ i$. Therefore $T_1 \circ T_2(A_i) \in \mathfrak{A}(\mathcal{N}(T_1) \circ \mathcal{N}(T_2) \circ i$ and thus $\mathcal{N}(T_1 \circ T_2) \subset \mathcal{N}(T_1) \circ \mathcal{N}(T_2)$. The inclusion can be strict which can be easily seen with the example $T_2 = T_1^{-1}$.

3. We have

$$\begin{aligned} (i,j) \notin \mathcal{N}(T^{-1}) &\Leftrightarrow \left[\mathfrak{A}_i, T^{-1}[\mathfrak{A}_j]\right] = \{0\} \\ &\Leftrightarrow \left[T[\mathfrak{A}_i], \mathfrak{A}_j\right] = \{0\} \\ &\Leftrightarrow (j,i) \notin \mathcal{N}(T) \\ &\Leftrightarrow (i,j) \notin \mathcal{N}(T)^{-1}, \end{aligned}$$

and therefore $\mathcal{N}(T^{-1}) = \mathcal{N}(T)^{-1}$. In the second step we used that T is an automorphism and in the last step we used the definition of $\mathcal{N}^{-1}$.

□

If $\mathfrak{A}(\mathbb{Z})$ is translation invariant there is a shift $\tau : \mathfrak{A}(\mathbb{Z}) \to \mathfrak{A}(\mathbb{Z})$, $\tau[\mathfrak{A}_i] = \mathfrak{A}_{i-1}$. Concatenating an operation T with a shift results in a shift of the neighborhood.

Lemma 3.1.4. *Let $T : \mathfrak{A}(\mathbb{Z}) \to \mathfrak{A}(\mathbb{Z})$ be a quasi-local automorphism, let $\mathfrak{A}(\mathbb{Z})$ be translation invariant, and let $n \in \mathbb{N}$. Then*

$$\mathcal{N}(\tau^n \circ T) = \{(i,j) | (i+n,j) \in \mathcal{N}(T)\}, \tag{3.6}$$

$$\mathcal{N}(T \circ \tau^n) = \{(i,j) | (i,j-n) \in \mathcal{N}(T)\}. \tag{3.7}$$

Proof.

$$\begin{aligned} \mathcal{N}(\tau^n \circ T) &= \left\{(i,j) | \, \exists A_i \in \mathfrak{A}_i, B_j \in \mathfrak{A}_j : \left[A_i, (\tau^n \circ T)[B_j]\right] \neq 0\right\} \\ &= \left\{(i,j) | \, \exists A_i \in \mathfrak{A}_i, B_j \in \mathfrak{A}_j : \left[\tau^{-n}[A_i], T[B_j]\right] \neq 0\right\} \\ &= \left\{(i,j) | \, \exists A_{i+n} \in \mathfrak{A}_{i+n}, B_j \in \mathfrak{A}_j : \left[A_{i+n}, T[B_j]\right] \neq 0\right\} \\ &= \{(i,j) | \, (i+n,j) \in \mathcal{N}(T)\} \end{aligned}$$

$$\begin{aligned}\mathcal{N}(T\circ\tau^n) &= \left\{(i,j)\middle|\, A_i\in\mathfrak{A}_i, B_j\in\mathfrak{A}_j : \left[A_i,(T\circ\tau^n)[B_j]\right]\neq 0\right\}\\ &= \left\{(i,j)\middle|\, \exists A_i\in\mathfrak{A}_i, B_{j-n}\in\mathfrak{A}_j : \left[A_i,T\left[B_{j-n}\right]\right]\neq 0\right\}\\ &= \{(i,j)|\,(i,j-n)\in\mathcal{N}(T)\}\end{aligned}$$

□

3.2 Quantum cellular automata

A reversible quantum cellular automaton (QCA) T is a reversible discrete time operation on a lattice of quantum systems. Usually translation-invariance of the quantum system and the operation is assumed additionally but this it not necessary. However we will always assume time-invariance, i.e., in every time step the operation will be the same. Furthermore QCAs are required to be local transformations, i.e. information can only spread finitely far in one time-step of the QCA.

Again we will use the transformation of observables rather than the transformation of states to describe the evolution of the system. In mathematical terms a reversible QCA is therefore just a local automorphism of the observable algebra: $T:\ \mathfrak{A}(\mathbb{Z})\to\mathfrak{A}(\mathbb{Z})$. QCAs have the property that they are uniquely determined by the images of single cite observables [37]. We call the local operation mapping observables from the algebra of the ith system to observables on the algebra of a neighborhood of the ith system T_i. As the QCA is a local automorphism so are the local operations T_i:

$$T_i\left[A_i\right] := T\left[A_i\right]\,\forall A_i\in\mathfrak{A}_i. \tag{3.8}$$

This can be expressed in the neighborhood calculus:

$$T_i:\mathfrak{A}_i\to\mathfrak{A}(\mathcal{N}(T)\circ i). \tag{3.9}$$

The time evolution of a QCA is illustrated in Figure 3.2.

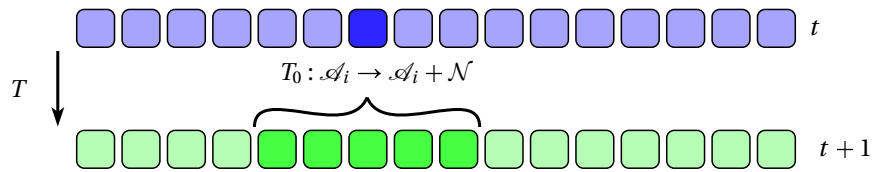


Figure 3.2: Time evolution of a QCA; the images of the observables on system $\mathscr{A}_i$ are contained in the set of observables of the systems $\mathscr{A}_i+\mathcal{N}$ in the next time step.

In the case of translation-invariant QCAs the situation is simplified even further. Translation-invariant operations commute with the lattice translation τ ($[T,\tau]=0$) and therefore $T\circ\tau[A]=\tau\circ T[A]$ for all $A\in\mathfrak{A}(\mathbb{Z})$. Then $T[A_i]=\tau^i\circ T\circ\tau^{-i}[A_i]$ and therefore $T_i[A_i]=\tau^i\circ T_0\circ\tau^{-i}[A_i]$ for all $i\in\mathbb{Z}$ and all $A_i\in\mathfrak{A}_i$. Thus we can use the same local transformation T_0 for all sites. Naturally, the neighborhood of a translation-invariant QCA is also translation-invariant:

$$\mathcal{N}(T)=\{(i,j)|\,i\geqslant j-n_-\wedge i\leqslant j+n_+\}\,. \tag{3.10}$$

This is depicted in Figure 3.3.

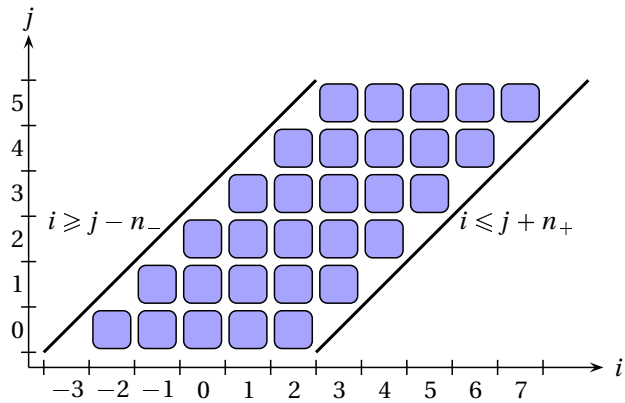


Figure 3.3: Neighborhood scheme of a translation-invariant QCA

The local operations T_i are endomorphisms from $\mathfrak{A}_i$ to $\mathfrak{A}(\mathcal{N}(T)\circ i)$. The global transformation T is uniquely determined by the local transformations, but not all combinations of local transformations produce a valid global transformation. To begin with, the global transformation has to be an automorphism. Thus it preserves the commutation relations of the observable algebra, i.e. $[T(A),T(B)]=T[A,B]$ holds for all $A,B\in\mathfrak{A}(\mathbb{Z})$. Especially the commutation of the single cell algebras of different cells has to be preserved:

$$\big[T_i[\mathfrak{A}_i],T_j[\mathfrak{A}_j]\big]=\{0\}\ \forall i,j\in\mathbb{Z},\,i\neq j. \tag{3.11}$$

Given that all T_i are endomorphisms this guarantees that T is an automorphism, if the index matches.

The index of a QCA characterizes the information flow. Two local operations can only be combined if the index coincides. This is easy to understand from the information flow point of view with the following example: let $\mathfrak{A}(\mathbb{Z})\bigotimes_{i\in\mathbb{Z}}\mathfrak{A}_i$ with

$\mathfrak{A}_i = \mathcal{M}_d$ be translation invariant and let T_i be the shift to the right (in the Heisenberg picture). Then a full single cell algebra $\mathcal{M}_d$ is shifted into cell $i+1$, which is also a full matrix algebra $\mathcal{M}_d$. Thus the content of cell $i+1$ has to be shifted to the right, too. For any QCA, be it translation-invariant or not, the index has to be a conserved quantity along the chain, which can be computed locally from any local rule. We will now present the mathematical definition of the index.

First let us group the systems in such a way, that the QCA T has only nearest neighbor interactions, i.e., $\mathcal{N}(T) = \{(i,j)|i \geqslant j-1 \wedge i \leqslant j+1\}$. Now we consider the image of two adjacent cells:

$$T\big[\mathfrak{A}_i \otimes \mathfrak{A}_{i+1}\big] \subset (\mathfrak{A}_{i-1} \otimes \mathfrak{A}_i) \otimes (\mathfrak{A}_{i+1} \otimes \mathfrak{A}_{i+2}).$$

To describe the embedding we use support algebras.

Definition 3.2.1. *Let $\mathfrak{B}_1$ and $\mathfrak{B}_2$ be finite dimensional C^*-algebras and let $\mathfrak{A} \subset \mathfrak{B}_1 \otimes \mathfrak{B}_2$ be a subalgebra. Then the support algebra $\mathbf{S}(\mathfrak{A}, \mathfrak{B}_1)$ of $\mathfrak{A}$ in $\mathfrak{B}_1$ is the smallest C^* algebra $\mathfrak{C}_1$ such that $\mathfrak{A} \subset \mathfrak{C}_1 \otimes \mathfrak{B}_2$.*

We define the algebras

$$\mathfrak{R}_{2i} = \mathbf{S}\left(T\big[\mathfrak{A}_{2i} \otimes \mathfrak{A}_{2i+1}\big], \mathfrak{A}_{2i-1} \otimes \mathfrak{A}_{2i}\right), \tag{3.12}$$

$$\mathfrak{R}_{2i+1} = \mathbf{S}\left(T\big[\mathfrak{A}_{2i} \otimes \mathfrak{A}_{2i+1}\big], \mathfrak{A}_{2i+1} \otimes \mathfrak{A}_{2i+2}\right), \tag{3.13}$$

where $\mathfrak{R}_{2i}$ describes flow of information to the right while $\mathfrak{R}_{2i+1}$ describes flow of information to the left. Figure 3.4 illustrates how the algebras are embedded into each other. The index is defined as

$$\operatorname{ind} T := \frac{r(2i)}{d(2i)} = \frac{d(2i+1)}{r(2i+1)}, \tag{3.14}$$

where $d(i) = \dim(\mathfrak{A}_i)$ and $r_i = \dim(\mathfrak{R}_i)$ are the dimensions of the algebras (e.g. $\dim(\mathcal{M}_d) = d$). The index is multiplicative in the sense that

$$\operatorname{ind}(T_1 \circ T_2) = \operatorname{ind} T_1 \operatorname{ind} T_2. \tag{3.15}$$

For proofs and further results on the index of QCAs see [38].

3.2.1 General bounds on the entanglement generation of QCAs

In this section we state general bounds on the evolution of the entanglement of a finite number of consecutive spins with the rest of the chain under the action of a localized automorphism (e.g. a QCA) as introduced in [39]. We will only consider the case of pure states on the whole chain. In this case the proper measure of entanglement is given by the von Neumann entropy

$$S = -\operatorname{tr} \rho_S \log_2 \rho_S,$$

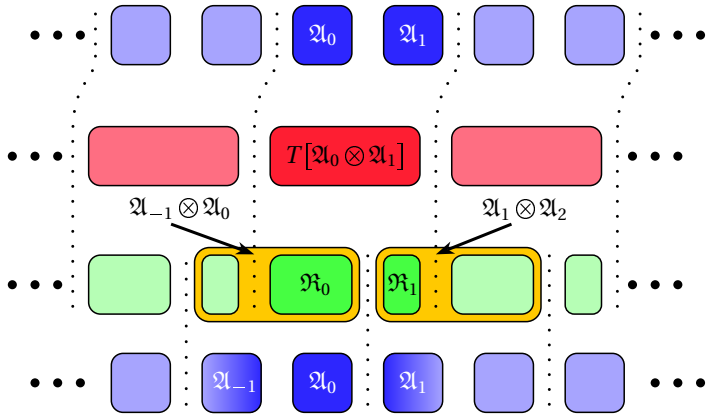


Figure 3.4: Flow of information and the embedding of the according algebras in a QCA

where ρ_S is the reduced density matrix of the finite segment of spins. In the case of mixed states the derived bounds still hold for the entropy generation, but the von Neumann entropy is not directly related to the entanglement.

The non-translation-invariant case

Theorem 3.2.2. *Consider the observable algebra of an infinite chain of d-level systems*

$$\mathfrak{A}_{\mathbb{Z}}^{(d)} := \bigotimes_{i=-\infty}^{+\infty} \mathfrak{A}_i^{(d)}, \quad \mathfrak{A}_i^{(d)} \cong \mathcal{M}_d,$$

and an automorphism $T : \mathfrak{A}_{\mathbb{Z}}^{(d)} \to \mathfrak{A}_{\mathbb{Z}}^{(d)}$. Let us introduce the notation: For $\Upsilon = [m_1, m_2]$, $\mathfrak{A}_\Upsilon^{(d)} := \bigotimes_{k\in\Upsilon} \mathfrak{A}_k^{(d)}$. Suppose that T satisfies the locality condition

$$T(\mathfrak{A}_\Lambda^{(d)}) \subset \mathfrak{A}_{\mathcal{N}\circ\Lambda}^{(d)},$$

where $\mathcal{N}$ is the neighborhood of T (extended in a way that $i \subset \mathcal{N} \circ i$) and $\Lambda = [k_1, k_2] \subset \mathcal{N} \circ \Lambda = [l_1, l_2]$ hold.

Let ω be a state on the spin-chain $\mathfrak{A}_{\mathbb{Z}}^{(d)}$, and let us define the T-evolved state as $\omega' := \omega \circ T$. The restrictions of these states to a subalgebra $\mathfrak{A}_\Upsilon^{(d)}$ will be denoted by ω_Υ and ω'_Υ, respectively. Then the following bounds hold for the von Neumann entropies of the restricted states:

$$S(\omega_\Lambda) - 2n \log_2 d \leqslant S(\omega'_\Lambda) \leqslant S(\omega_\Lambda) + 2n \log_2 d, \tag{3.16}$$

where $n = |(\mathcal{N}\circ\Lambda)\backslash\Lambda| = l_2 - l_1 - k_2 + k_1$. *Moreover, these bounds are sharp.*

Proof. Restricting the automorphism T to the subsubalgebra $\mathfrak{A}_\Lambda^{(d)}$, we obtain the monomorphism $T_\Lambda : \mathfrak{A}_\Lambda^{(d)} \to \mathfrak{A}_{\mathcal{N}\circ\Lambda}^{(d)}$. This monomorphism can be extended to an automorphism $\widetilde{T} : \mathfrak{A}_{\mathcal{N}\circ\Lambda}^{(d)} \to \mathfrak{A}_{\mathcal{N}\circ\Lambda}^{(d)}$.[1] Let us introduce the following state on $\mathfrak{A}_{\mathcal{N}\circ\Lambda}$:

$$\widetilde{\omega}'(\mathcal{N}\circ\Lambda) := \omega_{\mathcal{N}\circ\Lambda}\circ\widetilde{T}.$$

From this construction it immediately follows that $\widetilde{\omega}'(\mathcal{N}\circ\Lambda)_\Lambda = \omega'_\Lambda$. Moreover, since $\widetilde{\omega}'(\mathcal{N}\circ\Lambda)$ and $\omega_{\mathcal{N}\circ\Lambda}$ are connected by an automorphism their von Neumann entropies are equal: $S(\widetilde{\omega}'(\mathcal{N}\circ\Lambda)) = S(\omega_{\mathcal{N}\circ\Lambda})$.

We will prove the bounds (3.16) using the subadditivity [40] of the von Neumann entropy. The subchain $\mathfrak{A}_{\mathcal{N}\circ\Lambda}^{(d)}$ can be divided as $\mathfrak{A}_{\mathcal{N}\circ\Lambda}^{(d)} = \mathfrak{A}_\Lambda^{(d)} \otimes \mathfrak{A}_{rest}^{(d)}$, where $\mathfrak{A}_{rest}^{(d)}$ is isomorphic to the algebra of $d^n \times d^n$ matrices, hence the maximal entropy of a state defined on $\mathfrak{A}_{rest}^{(d)}$ is $n\log_2 d$. The triangle inequality and the subadditivity theorem give the following inequalities:

$$\begin{aligned} S(\omega_\Lambda) - n\log_2 d \leqslant \quad & S(\omega_{\mathcal{N}\circ\Lambda}) \quad \leqslant S(\omega_\Lambda) + n\log_2 d \\ S(\omega'_\Lambda) - n\log_2 d \leqslant \quad & S(\widetilde{\omega}'(\mathcal{N}\circ\Lambda)) \quad \leqslant S(\omega'_\Lambda) + n\log_2 d \end{aligned}$$

Now, using that $S(\omega_{\mathcal{N}\circ\Lambda}) = S(\widetilde{\omega}'(\mathcal{N}\circ\Lambda))$ we immediately obtain the bounds (3.16).

The sharpness of the inequalities can be shown using a state on the spin-chain where the sites at $2i$ are maximally entangled with the sites at $2i+1$. We consider the translation τ which shifts all one-cell algebras by one cell to the right as our time evolution. The extended neighborhood of τ is $\mathcal{N}(\tau) = \{(x,y)|x \leqslant y, x \geqslant y-1\}$. Then $k_2 - k_1 = 2$ and $l_2 - l_1 = 3$, and we get $n = l_2 - l_1 - k_2 + k_1 = 1$. Now, restricting this state to the subalgebra $\mathfrak{A}_{[2i,2i+3]}^{(d)}$ the entropy of the restriction is zero. However, the entropy of this restriction after the time evolution will be $2\log_2 d$, since the two sites at the border will be maximally entangled with sites outside the considered region. This example is illustrated in Figure 3.5.

In the above example the generated entanglement is destroyed in the next step, so the bound is only saturated for one time step. But a slightly more involved example shows, that the bound can be saturated for arbitrarily many timesteps:

Let again T be the translation automorphism on $\mathfrak{A}_\mathbb{Z}^{(d)}$, and let us consider the state which is defined as the direct product of maximally entangled states between the lattice site at i and the lattice site at $-i+1$ for all i. So in this state the lattice site at 1 is fully entangled with the lattice site at 0, the lattice site at 2 is fully entangled with the lattice site at -1, and so on. Now, we will consider subsystems of arbitrary length $k = k_2 - k_1$. If k is even, $k = 2j$, then the subsystem we consider

[1] This can be seen by noting that $\mathfrak{A}_{\mathcal{N}\circ\Lambda}^{(d)} = T(\mathfrak{A}_\Lambda^{(d)}) \otimes \mathfrak{B}$, where $\mathfrak{B} \cong \bigotimes_{k\in(\mathcal{N}\circ\Lambda)\backslash\Lambda} \mathfrak{A}_k^{(d)}$. If Q is an isomorphism between the latter two algebras, then one can define $\widetilde{T}_{\mathcal{N}\circ\Lambda}$ as $T_\Lambda \otimes Q$.

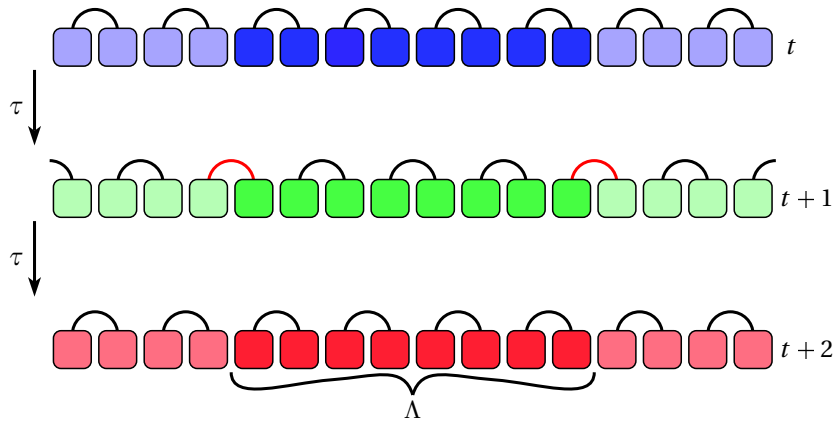


Figure 3.5: Entanglement generation of the shift acting on a state with maximally entangled nearest neighbors (depicted by the links); in the first step entanglement between the region Λ and the rest of the chain is created (red links). In the next step it is destroyed.

is the interval $[-j+1, j]$. Its original entropy is 0, and the entropy grows linearly during the time-evolution saturating our linear bound until it reaches the maximal entropy it can obtain, namely $k \log_2 d$. After this it stays constant. If k is odd, $k = 2j+1$, we consider the interval $[-j, j]$ as our subsystem. The original entropy of the subsystem is $\log_2 d$, and the entropy grows linearly and saturates our bound until it reaches $(2j+1)\log_2 d$, then it stays constant. The example is illustrated in Figure 3.6.

□

The translation-invariant case

In the previous subsection we showed that the bounds (3.16) on entanglement generation are sharp in the general case. However, considering translation-invariant states and translation invariant QCAs, i.e., local automorphisms that commute with the translations, we can sharpen these bounds further.

Theorem 3.2.3. *Consider the observable algebra of an infinite chain of d-level systems*

$$\mathfrak{A}_{\mathbb{Z}}^{(d)} := \bigotimes_{i=-\infty}^{+\infty} \mathfrak{A}_i^{(d)}, \quad \mathfrak{A}_i^{(d)} \cong M_d,$$

and a QCA T acting on $\mathfrak{A}_{\mathbb{Z}}^{(d)}$ having a neighborhood $\mathcal{N}$ of n additional cells, i.e. we

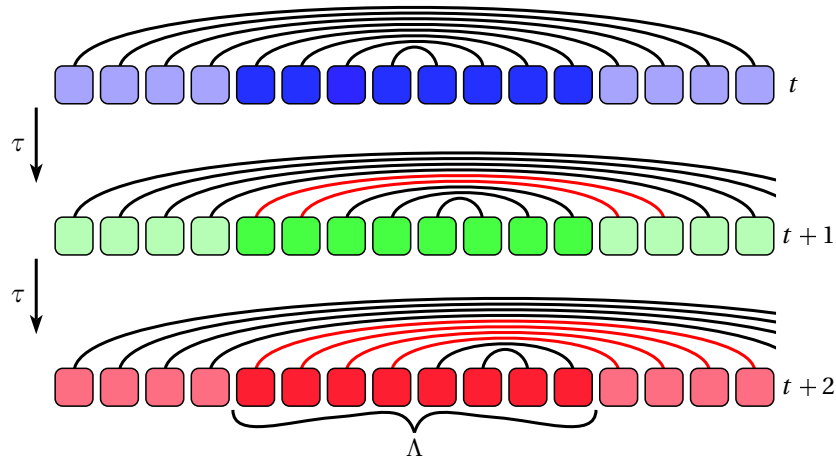


Figure 3.6: Entanglement generation of the shift acting on a state with maximally entagled pairs (depicted by the links). Each step creates two maximally entangled pairs between the region Λ and the rest of the chain (red links) until the entanglement reaches $|\Lambda|$ pairs.

have $|(\mathcal{N}\circ\Lambda)\backslash\Lambda| = n$*. Thus* T *is an automorphism satisfying*

$$T(\mathfrak{A}_i)^{(d)} \subset \mathfrak{A}^{(d)}_{\mathcal{N}\circ i} = \mathfrak{A}^{(d)}_{i-n_1} \otimes \mathfrak{A}^{(d)}_{i-n_1+1} \otimes \cdots \otimes \mathfrak{A}^{(d)}_{i+n_2-1} \otimes \mathfrak{A}^{(d)}_{i+n_2}, \quad T\circ\tau = \tau\circ T, \tag{3.17}$$

where $i \in \mathbb{Z}$ *and* n_1 *and* n_2 *are integers satisfying* $n_1 + n_2 = n \geqslant 0$*.*

Let ω *be a translation-invariant state on the spin-chain, and let us define the time-evolved state (at time* $t \in \mathbb{N}$*) as* $\omega^t := \omega \circ T^t$*. The von Neumann entropy* $S_L(t)$ *of the restriction of* ω^t *to* L *consecutive qubits can be bounded in the following way:*

$$S_L(0) - nt\log_2 d \leqslant S_L(t) \leqslant S_L(0) + nt\log_2 d. \tag{3.18}$$

Moreover, these bounds are sharp for $d = 2$*.*

Proof. Since the state ω is translation-invariant and T commutes with the translations, the "entropy production" is the same for the automorphisms T and $T\circ\tau^k$ ($k \in \mathbb{Z}$), hence we can assume without loss of generality that in Equation (3.17) we have $n_1, n_2 \geqslant 0$.

Let us denote the restriction of a state $\varphi : \mathfrak{A}^{(d)}_{\mathbb{Z}} \to \mathbb{C}$ to $\mathfrak{A}^{(d)}_{\Upsilon} = \bigotimes_{k=\in\Upsilon} \mathfrak{A}^{(d)}_k$ by φ_Υ. Consider the subalgebra $\mathfrak{A}^{(d)}_\Lambda$, $\Lambda = [0, L-1]$ of $\mathfrak{A}^{(d)}_{\mathbb{Z}}$, which corresponds to L qudits. Restricting the automorphism T to this subalgebra, we get a monomorphism T_L :

$\mathfrak{A}_\Lambda^{(d)} \to \mathfrak{A}_{\mathcal{N}\circ\Lambda}^{(d)}$. We will also consider the inverse automorphism T^{-1}, and restrict T^{-1} to a monomorphism $(T^{-1})_L : \mathfrak{A}_\Lambda^{(d)} \to \mathfrak{A}_{\mathcal{N}^{-1}\circ\Lambda}^{(d)}$.[2]. The monomorphism $(T^{-1})_L$ can be extended to an automorphism $\widetilde{T}_L^{-1} : \mathfrak{A}_{\mathcal{N}^{-1}\circ\Lambda}^{(d)} \to \mathfrak{A}_{\mathcal{N}^{-1}\circ\Lambda}^{(d)}$.

Let ω be a translation-invariant state on $\mathfrak{A}_\mathbb{Z}^{(d)}$, then $\omega^1 := \omega\circ T$ will be translation-invariant, too. Let us also define the following state on $\mathfrak{A}_{\mathcal{N}^{-1}\circ\Lambda}^{(d)}$

$$\widetilde{\omega}(\mathcal{N}^{-1}\circ\Lambda) := \omega^1_{\mathcal{N}^{-1}\circ\Lambda}\circ\widetilde{T}_L^{-1}.$$

The von Neumann entropy of $\widetilde{\omega}(\mathcal{N}^{-1}\circ\Lambda)$ and $\omega^1_{\mathcal{N}^{-1}\circ\Lambda}$ are the same (since they are connected by an automorphism), and it follows from the definition of $\widetilde{T}_L^{-1}$ that $\widetilde{\omega}(\mathcal{N}^{-1}\circ\Lambda)_\Lambda = \omega_\Lambda$.

Now, from the strong subadditivity of the von Neumann entropy it follows that for a translation-invariant state ω $S(\omega_\Upsilon) \geqslant S(\omega_\Lambda)$ if $|\Upsilon| \geqslant |\Lambda|$ [42], hence

$$S_L(1) = S(\omega^1_\Lambda) \leqslant S(\omega^1_{\mathcal{N}^{-1}\circ\Lambda}). \tag{3.19}$$

On the other hand, using the subadditivity of the entropy for the state $\widetilde{\omega}(\mathcal{N}^{-1}\circ\Lambda)$ (dividing the observable algebra of the subchain $[-n_2, n_1]$ as: $\mathfrak{A}_{\mathcal{N}^{-1}\circ\Lambda}^{(d)} = \mathfrak{A}_\Lambda^{(d)} \otimes \mathfrak{A}_{(\mathcal{N}^{-1}\circ\Lambda)\setminus\Lambda}^{(d)}$), we get:

$$S_L(\widetilde{\omega}(\mathcal{N}^{-1}\circ\Lambda)) \leqslant S_L(\widetilde{\omega}(\mathcal{N}^{-1}\circ\Lambda)_\Lambda) + (n_1+n_2)\log_2 d = S_L(0) + n\log_2 d \tag{3.20}$$

Combing the fact that $S(\omega^1_{\mathcal{N}^{-1}\circ\Lambda}) = S(\widetilde{\omega}(\mathcal{N}^{-1}\circ\Lambda))$ with the inequalities (3.19) and (3.20) we arrive at the $S(1) \leqslant S(0) + n\log_2 d$. By simple induction we obtain the desired upper bound:

$$S_L(t) \leqslant S_L(0) + nt\log_2 d.$$

The lower bound in (3.18) can simply be obtained by reversing the time arrow: suppose that for a QCA automorphism T this lower bound does not hold, this would mean that for the QCA T^{-1} the upper bound would not hold, which is a contradiction as we proved the upper bound just now.

The sharpness of the inequalities for $d = 2$ follows from the study of of Clifford QCAs acting on the "all spins up" state in Section 5.3.1. □

3.3 Causal Operations

Let us begin by defining what we will call a causal operation. We work with a chain of quantum systems $\mathscr{A}_i$, $i \in \mathbb{Z}$, each described by its observable algebra $\mathfrak{A}_i$. The

[2] The fact that the range of $\mathfrak{A}_\Lambda$ under the the action of T^{-1} is in $\mathfrak{A}_{\mathcal{N}^{-1}\circ\Lambda}$ was shown in [41], page 22.

whole chain is described by the quasi-local algebra $\mathfrak{A}_{\mathbb{Z}} = \bigotimes_{i\in\mathbb{Z}}\mathfrak{A}_i$. We think of the chain as being expanded in time, not necessarily in space. Systems with higher indices belong to later times. Causality then means that information can only travel forward in time—only to systems with larger indices. In the Heisenberg picture of course the direction is reversed and observables can only travel backward in time, to systems with smaller indices.

In the literature the term causal operation was also used in a different sense. According to [43] an operation T is causal if $T(\mathfrak{A}) \subset \mathfrak{A}$, i.e., there is no propagation of information at all. The operations we consider here are then called semi-causal.

Definition 3.3.1. *A quasi-local CPU map $T : \mathfrak{A}_{\mathbb{Z}} \to \mathfrak{A}_{\mathbb{Z}}$ is called causal if and only if for all $i \in \mathbb{Z}$ we have*

$$T\big[\mathfrak{A}_{\leqslant i}\big] \subset \mathfrak{A}_{\leqslant i}, \tag{3.21}$$

where $\mathfrak{A}_{\leqslant i} = \bigotimes_{j\in\mathbb{Z}}^{j\leqslant i}\mathfrak{A}_j$ is the observable algebra of the left half-chain and $\mathfrak{A}_{>i} = \bigotimes_{j\in\mathbb{Z}}^{j>i}\mathfrak{A}_j$ is the observable algebra of the right half-chain according to a cut between systems i and $i+1$.

Let us first show the equivalence to a commutational definition of causality:

Lemma 3.3.2. *Let $T : \mathfrak{A}_{\mathbb{Z}} \to \mathfrak{A}_{\mathbb{Z}}$ be a CPU map. Then T is causal if and only if $\big[T\big[\mathfrak{A}_{\leqslant i}\big], \mathfrak{A}_{>i}\big] = 0$, for all $i \in \mathbb{Z}$.*

Proof. First assume that T is causal. Therefore we have $T\big[\mathfrak{A}_{\leqslant i}\big] \subset \mathfrak{A}_{\leqslant i}$ for all $i \in \mathbb{Z}$ and by Lemma 2.1.4 $\big[T\big[\mathfrak{A}_{\leqslant i}\big], \mathfrak{A}_{>i}\big] = \{0\}$ for all $i \in \mathbb{Z}$.

Now assume $\big[T\big[\mathfrak{A}_{\leqslant i}\big], \mathfrak{A}_{>i}\big] = 0$, for all $i \in \mathbb{Z}$. Then again by Lemma 2.1.4 we have $T\big[\mathfrak{A}_{\leqslant i}\big] \subset \mathfrak{A}_{\leqslant i}$. □

So far we did not make any restrictions on how far information can travel. As long as the image of a localized observable $A \in \mathfrak{A}_\Lambda$, $\Lambda \subset (-\infty, i]$ can be approximated by finitely localized elements of the quasi-local algebra of the left half-chain up to any $\varepsilon > 0$ in operator norm the transformation is valid. Therefore input on system i can have an influence on all future systems. If the inputs on system i can only influence a finite number of outputs in the range $[i, i+\tau]$ we say that the process has finite depth. In the Heisenberg picture this is defined as follows:

Definition 3.3.3. *A causal operation T has finite depth τ if and only if $T\big[\mathfrak{A}_{[i,j]}\big] \subset \mathfrak{A}_{[i-\tau,j]}$.*

The neighborhood of a finite depth causal operation T is of the form

$$\mathcal{N}(T) \subset \{(i,j) | i \leqslant j \wedge i \geqslant j - \tau\}. \tag{3.22}$$

Thus the neighborhood of a causal process is restrained to a diagonal strip in the i, j plane as depicted in Figure 3.7. We will now determine the minimal strip containing

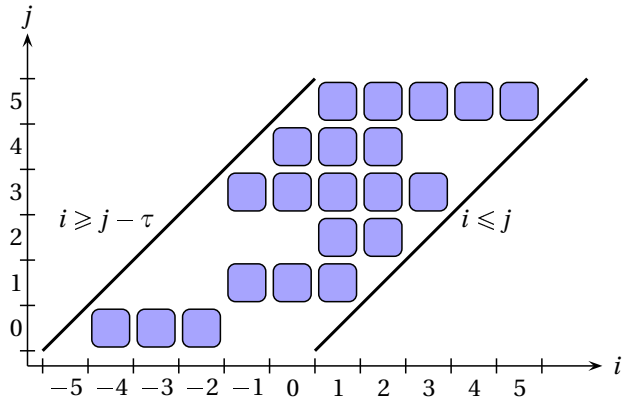


Figure 3.7: Neighborhood of a causal process with finite depth τ; the neighborhood can vary between the lattice sites, but global bounds on its size exist.

the neighborhood. The width of the minimal strip is given by the memory depth τ. Locally the memory depth can vary as the following example shows: Consider a chain of qubits and an operation using one additional qubit of memory. On all even sites the local operation is the identity on the system and the memory. On all odd sites system qubit and memory qubit are swapped such that the memory input goes to the output system and the input system to the memory output. This process is causal and implements the identity on even sites and a shift on the odd sites. The localization of the images of single site observables thus depend on the site. The neighborhood is shown in Figure 3.8.

To determine bounds on the memory requirement of a channel implementations of a causal process and its causal inverse we need to make the neighborhood bounds tight. On the left we make the bound tight by introducing a minimal memory depth $\tau_m = \min\{\tau | \forall t_1, t_2, T(\mathfrak{A}([t_1, t_2])) \subset \mathfrak{A}([t_1 - \tau, t_2])\}$. From now on we will only work with τ_m and, by a slight abuse of notation, call it the depth and denote it by τ

The right bound $i \leqslant j$ can also be tightened in some cases. Consider a causal process T. If we concatenate it with a shift τ_λ we shift the localization area to the left. The right bound $i \leqslant j$ is not tight any more, because there is no j for which $(j, j) \in \mathcal{N}(T)$. If a causal process contains no such shift the right bound is tight and we call the channel global-shift-free.

Definition 3.3.4. *A causal process T is called global-shift-free, if and only if there exists a $j \in \mathbb{Z}$, such that $(j, j) \in \mathcal{N}(T)$.*

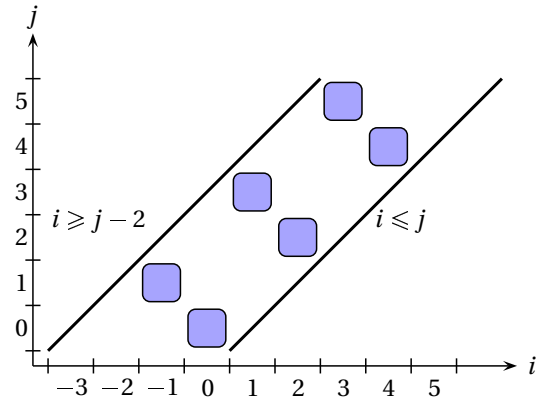


Figure 3.8: Neighborhood of the causal process defined by alternating swap and identity; This process is causal and has finite depth 2. All even sites remain unchanged, while all observables on odd sites are shifted by two sites.

This means that on at least one site j the image of $\mathfrak{A}_j$ has support on $\mathfrak{A}_j$. Locally the process can still contain shifts like in our example above.

If a process T is not global-shift-free it can be decomposed into a shift τ_λ and a global-shift-free process $\tilde{T}$: $T = \tilde{T}\tau_\lambda$.

Lemma 3.3.5. *Let T be a causal process which is not global-shift-free and has memory depth τ. Then there exists an integer $\lambda > 0$ such that $\tilde{T} = T\tau_{-\lambda}$ is causal and global-shift-free.*

Proof. T has a neighborhood of the form $\mathcal{N}(T) \subset \{(i,j)|i \leqslant j-\lambda, i \geqslant j-\tau\}$ for some $\lambda > 0$, because it is not global-shift-free and of depth τ. If several possible λ exists we choose the maximal one. Then there is a j with $(j-\lambda, j) \in \mathcal{N}(T)$. The neighborhood of $\tau_{-\lambda}$ is $\mathcal{N}(\tau_{-\lambda}) = \{(i,j)|i = j+\lambda\}$. Then

$$\begin{aligned}
\mathcal{N}(T\tau_{-\lambda}) &\subset \mathcal{N}(T) \circ \mathcal{N}(\tau_{-\lambda}) \\
&\subset \{(i,k)|\exists j, (i,j) \in \mathcal{N}(T) \wedge (j,k) \in \mathcal{N}(\tau_{-\lambda})\} \\
&= \{(i,k)|\exists j, i \leqslant j-\lambda \wedge i \geqslant j-\tau \wedge j = k+\lambda\} \\
&= \{(i,k)|i \leqslant k \wedge i \geqslant k+\lambda-\tau\}.
\end{aligned}$$

Thus $\hat{T}$ is causal and has memory depth $\hat{\tau} = \tau - \lambda$. By our choice of λ it is also global-shift-free. □

3.4 Resource requirements

In our analysis of causal operations and convolutional codes we will be concerned with the question of minimal memory implementations of causal operations and convolutional encoders and decoders that use minimal spatial resources, i.e. a minimal number of qubits. Here we will define what we mean when we speak of (spatial) resources and memory. Recently implementations of quantum computational tasks with minimal spatial resources have been studied by different groups. In [44, 45] minimal resource implementations of different quantum protocols were analyzed. The spacial resource considered is the logarithm of the needed ancilla space. If we think of qubits this would be the number of ancilla qubits, that is needed to carry out the protocol. In [46, 47, 48] minimal memory encoders for convolutional codes are studied. Here the considered resource is the memory needed to implement the encoder, which is defined to be number of qubits that is passed on from one step of the encoding to the next.

In Chapter 7 we will carry out a resource analysis to compare the performance of block codes to the performance of convolutional codes. For this task we need to compare the total spatial resource requirement of the encoders respectively decoders, the number of qubits a quantum computer needs to implement the encoding or decoding. In an earlier publication we called this the needed memory [49]. However, this lead to confusion with conflicting definitions of memory requirements like in [46, 47, 48]. In this thesis we will call the total spatial resources needed just resources and the term memory will denote the memory of a convolutional code or a quantum memory channel (in accordance with [46, 47, 48]). The resources needed for an operation are also equal to the support of this operation in terms of qubits. However, in a circuit we might need additional resources to store data qubits that are not affected by the operation. In this case the needed resources are determined by delaying preparation of input qubits as far as possible and shift all the measurements forward as far as possible. Then in at each point of time we can determine the needed memory by just counting parallel wires. See Figure 3.9.

3.5 Quantum channels with memory

Every quantum operation from system $\mathscr{A}$ to system $\mathscr{B}$ can be described in the formalism of quantum channels. Preparations and measurements, as well as unitary on non-unitary time evolutions are all special types of quantum channels. In the following we will assume familiarity with the basics of quantum channels.

However, we will introduce a special kind of quantum channels, the quantum memory channels which were first introduced in [50] and systematically studied in [51]. Quantum memory channels are used to describe processes which introduce correlations into a chain of quantum systems. Thus the transformation applied to

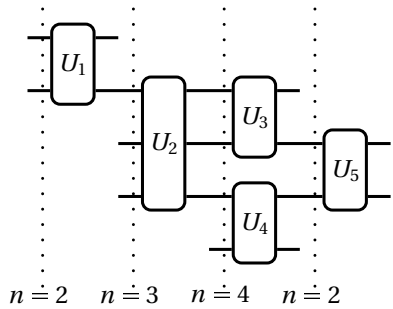


Figure 3.9: The resource requirements in a quantum circuit can be determined by just counting parallel wires at each point in time if preparation and measurements are shifted as far to the end respectively the beginning of the circuit as possible.

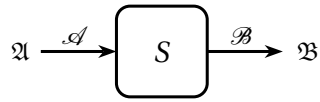


Figure 3.10: Quantum channel S implementing a quantum transformation from system $\mathscr{A}$ to system $\mathscr{B}$; the operator algebras on input and output system are denoted by $\mathfrak{A} = \mathcal{B}(\mathcal{H}_{\mathscr{A}})$ and $\mathfrak{B} = \mathcal{B}(\mathcal{H}_{\mathscr{B}})$.

an input can depend on past inputs and information from past inputs can influence future outputs or even be completely delayed to future outputs. An example for such a process is a stream of particles passing through an optical cavity. Each particle will interact with the field in the cavity. After the interaction the state of the cavity will depend on the state of the particle before the interaction and vice versa. Thus, when the next particle interacts with the cavity, its state after the interaction depends on the state of the cavity before the interaction and therefore might also depend on the state of the last particle before the interaction and possibly all particles that passed the cavity before.

To model the storage of information from earlier uses of the channel we add an additional memory input and output system. The memory output of one use of the memory channel is fed into the next use of the channel as memory input. Therefore, the memory system has to be of the same kind on input and output. In our example of an optical cavity the cavity is the memory system while the atom passing through it are both the input and output systems. A schematic of such a memory channel and its concatenation is presented in Figure 3.11.

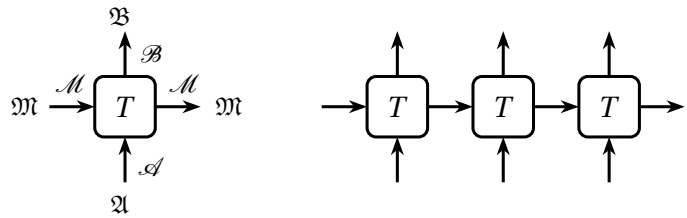


Figure 3.11: A quantum memory channel and its concatenation; the memory system $\mathscr{M}$ is passed on from one channel use to the next.

We can also approach memory channels from a different perspective. We consider an operation that acts on an infinite chain of quantum systems in such a way that information only travels in one direction. Furthermore we require the operation to be translation-invariant. Obviously this is a causal operation. In [51] it was shown how such a translation-invariant causal operation can be decomposed and represented as a quantum memory channel. This is depicted in Figure 3.12. In Section 4.1 we will present a generalization of this construction to non-translation-invariant causal operations.

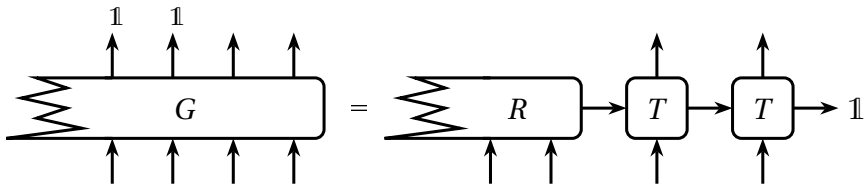


Figure 3.12: The structure theorem for causal operations shows how a translation-invariant causal operation G can be decomposed into an initialization R and the repeated use of a memory channel S.

Mathematically we can describe quantum channels either in the Schrödinger or in the Heisenberg picture. In the first case the channel is a completely positive and trace preserving (CPTP) linear map transforming input density matrices to output density matrices. In the second case a quantum channel is represented by completely positive and unital (CPU) linear map that transforms operators acting on the output system to operators acting on the input system. The operators on both input and output system form C^*-algebras. As argued in Section 2.1 we will mainly use the Heisenberg picture to avoid the problems arising from infinite tensor products of Hilbert spaces.

In the case of memory channels the algebras on the input respectively output are tensor products of the algebras of the data input algebra $\mathfrak{A} = \mathcal{B}(\mathcal{H}_{\mathscr{A}})$ and the memory input algebra $\mathfrak{M} = \mathcal{B}(\mathcal{H}_{\mathscr{M}})$ respectively the data output algebra $\mathfrak{B} = \mathcal{B}(\mathcal{H}_{\mathscr{B}})$ and the memory output algebra $\mathfrak{M} = \mathcal{B}(\mathcal{H}_{\mathscr{M}})$. This leads us to the following definition:

Definition 3.5.1. *[[51]] A quantum channel with memory is a completely positive and unital map $S : \mathcal{B}(\mathcal{H}_{\mathscr{B}}) \otimes \mathcal{B}(\mathcal{H}_{\mathscr{M}}) \mapsto \mathcal{B}(\mathcal{H}_{\mathscr{M}}) \otimes \mathcal{B}(\mathcal{H}_{\mathscr{A}})$, where $\mathcal{H}_{\mathscr{A}}$ is the input Hilbert space, $\mathcal{H}_{\mathscr{B}}$ the output Hilbert space, and $\mathcal{H}_{\mathscr{M}}$ the memory Hilbert space.*

In the following we will often only use the notation $\mathfrak{A}$ instead of $\mathcal{B}(\mathcal{H}_{\mathscr{A}})$. On first sight, swapping the position of the memory algebra from output to input might seem odd. However, it makes a lot of sense when we consider several consecutive uses of the channel. By $S_n : \mathfrak{B}^{\otimes n} \otimes \mathfrak{M} \mapsto \mathfrak{M} \otimes \mathfrak{A}^{\otimes n}$ we denote the n-fold concatenation of S:

$$S_n = \left(S \otimes \mathrm{id}_{\mathfrak{A}}^{\otimes n-1}\right) \circ \ldots \circ \left(\mathrm{id}_{\mathfrak{B}}^{\otimes n-2} \otimes S \otimes \mathrm{id}_{\mathfrak{A}}\right) \circ \left(\mathrm{id}_{\mathfrak{B}}^{\otimes n-1} \otimes S\right). \tag{3.23}$$

Because we swap the memory system with input respectively output system we can use the original S in the concatenation. If the memory system was at the beginning or the end of the chain we would need to introduce a different version of S for every step of the concatenation. Furthermore these would not be localized and the construction would cause problems if we do not know the number of channel concatenations a priori, which is a reasonable setting, e.g. when we think of the transmission of a message of unknown length (like a quantum phone call).

We will now introduce an important property of memory channels, the forgetfulness: a channel is forgetful if the initial state of the memory has no effect on the outputs in the limit of infinitely many uses of the channel. In the Heisenberg picture this corresponds to the situation that every observable localized on the memory output converges to the identity on the memory input in the limit of infinitely many channel uses. Formally this is defined as follows:

Definition 3.5.2 ([51])**.** *Let $S : \mathfrak{B} \otimes \mathfrak{M} \mapsto \mathfrak{M} \otimes \mathfrak{A}$ be a quantum memory channel, S_n its n-fold concatenation, and let $\hat{S} : \mathfrak{M} \mapsto \mathfrak{M} \otimes \mathfrak{A}$ be the concatenated channel in which all outputs (except the memory output) are ignored: $\hat{S}_n[M] := S_n[\mathbb{1}_{\mathfrak{B}}^{\otimes n} \otimes M]$ for all $M \in \mathfrak{M}$. Then S is forgetful if and only if there exists a sequence of quantum channels[3] $\tilde{S}_n : \mathfrak{M} \to \mathfrak{A}^{\otimes n}$ such that*

$$\lim_{n\to\infty} \left\| \hat{S}_n - \mathbb{1}_{\mathfrak{M}} \otimes \tilde{S}_n \right\|_{cb} = 0, \tag{3.24}$$

where $\mathbb{1}_{\mathfrak{M}}$ denotes the channel that returns the identity on $\mathfrak{M}$ whatever the input was. The norm used is the so called completely bounded norm *or cb-norm which is defined via $\|S\|_{cb} = \sup_n \|S \otimes id_n\|_{\infty}$. It has the nice properties that $\|S_1 \otimes S_2\|_{cb} = \|S_1\|_{cb} \cdot \|S_2\|_{cb}$ and $\|S\|_{cb} = 1$ for every channel S.*

[3] actually a sequence of linear maps is sufficient

If Equation (3.24) holds for finite n, the channel is called strictly forgetful. The idea of $\hat{S}_n$ is illustrated in Figure 3.13. Figure 3.14 illustrates Equation (3.24).

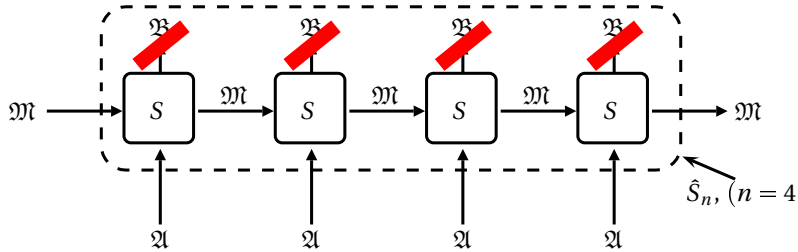


Figure 3.13: The channel $\hat{S}_n$ is derived from S_n by ignoring all outputs except for the memory output. In the Heisenberg picture this corresponds to setting the observables on all output systems to $\mathbb{1}$.

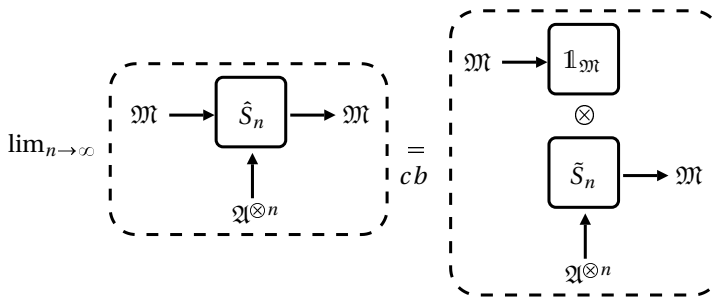


Figure 3.14: A memory channel is called forgetful if and only if in the limit of infinitely many uses the influence of the initial memory on the outputs vanishes. In the Heisenberg picture this means that in the limit of infinitely many channel uses we can find a sequence of channels $\tilde{S}_n$ such that $\mathbb{1}_{\mathfrak{M}} \otimes \tilde{S}_n$ is cb-norm close to $\hat{S}_n$.

If the memory algebra is finite we can use the ∞-norm instead of the cb-norm in Equation (3.24) which greatly simplifies the task to determine if a channel is forgetful or not [51].

For strictly forgetful channels we define the memory depth as

$$\tau = \min\left\{ n \,\middle|\, \left\| \hat{S}_n - \mathbb{1}_{\mathfrak{M}} \otimes \tilde{S}_n \right\|_{cb} = 0 \right\} - 1. \tag{3.25}$$

The memory depth is therefore a bound on the number of channel uses that information can stay in the channels memory and have an influence on the outputs.

Each memory channel defines a causal process. Strictly forgetful memory channels define strictly forgetful causal processes which have the same memory depth as the channel. We will further investigate the relation between causal processes and memory channels in Chapter 4.

3.6 Clifford channels

Clifford channels are so called covariant channels. A covariant channel fulfills an intertwining property with respect to transformations by projective representations of a group $\mathcal{G}$ and is defined in the following way:

Definition 3.6.1. *A channel $S\colon \mathfrak{B} \to \mathfrak{A}$ is called covariant with respect to two representations $U_1 \in \mathfrak{A}$ and $U_2 \in \mathfrak{B}$ of a group $\mathcal{G}$, if*

$$S\big[U_2(g)\,B\,U_2(g)^*\big] = U_1(g)S\big[B\big]U_1(g)^* \tag{3.26}$$

holds for all $B \in \mathfrak{B}$ and $g \in \mathcal{G}$.

Clifford channels are covariant with respect to Weyl systems. U_1 and U_2 are projective representations of an abelian group $\mathcal{G}$ of phase space vectors. In the literature they are defined as follows:

Definition 3.6.2 ([52])**.** *A channel $S:\ \mathcal{B}(\mathcal{H}_2) \to \mathcal{B}(\mathcal{H}_1)$ is a Clifford channel, if it is covariant with respect to two Weyl systems $(\mathrm{w}_1, f_1, \mathcal{H}_1)$ and $(\mathrm{w}_2, f_2, \mathcal{H}_2)$ over an abelian group $\mathcal{G}$.*

We will now alter this definition to better suit our needs. We will always use systems that are composed of qudits of possibly different dimensions. As a basis of the operator algebra we usually take local Weyl operators, i.e. Weyl operators localized on a single qudit each. A standard Weyl system (see Section 2.2) gives us a phase space description of such a basis. Of course, given the input and output systems the standard Weyl systems are fixed and there is no more freedom to change the representation of the phase space group Ξ_i if we use the standard Weyl system. However we can use a subset of the standard Weyl system as the projective representation used in the covariance condition (3.26). The standard Weyl system uses a phase space Ξ and assigns an operator $\mathrm{w}(\xi)$ to each $\xi \in \Xi$. We can now use a subgroup $\widetilde{\mathcal{G}}$ of Ξ and the operators defined by the standard Weyl system. We merely have to require the existence of a homomorphism $h:\ \Xi \to \widetilde{\mathcal{G}}$. Even if $\widetilde{\mathcal{G}} = \Xi$ we still have some freedom, because we can choose the homomorphism h. If we use this construction on the input and the output side of the channel we end up with two standard Weyl systems $(\mathrm{w}_i, \mathrm{e}^{-i\beta_i}, \mathcal{H}_i)$ over the input and output systems and two homomorphisms $h_i:\ \mathcal{G} \to \mathcal{G}_i \subset \Xi_i$ which induce projective representations of $\mathcal{G}$ on the input and output systems. Let us make this formal:

Lemma 3.6.3. *Let $\mathcal{G}$ and $\widetilde{\mathcal{G}}$ be abelian groups, $h\colon \mathcal{G} \to \widetilde{\mathcal{G}}$ a homomorphism connecting them and $\tilde{\mathrm{w}}\colon \widetilde{\mathcal{G}} \to \mathcal{B}(\mathcal{H})$ a projective representation of $\widetilde{\mathcal{G}}$ on $\mathcal{H}$ with the factor system $\widetilde{f}$. Then $\mathrm{w} = \tilde{\mathrm{w}} \circ h\colon \mathcal{G} \to U(\mathcal{H})$ is a projective representation of $\mathcal{G}$ on $\mathcal{H}$ with factor system f, $f(\xi,\eta) = \widetilde{f}(h\xi, h\eta)$.*

Proof. We just have to show that w fulfills the condition 2.28 for projective representations:

$$\begin{aligned}
\mathrm{w}(\xi+\eta) &= \tilde{\mathrm{w}}(h(\xi+\eta)) \\
&= \tilde{\mathrm{w}}(h\xi + h\eta) \\
&= \widetilde{f}(h\xi, h\eta)\tilde{\mathrm{w}}(h\xi)\tilde{\mathrm{w}}(h\eta) \\
&= \underbrace{\widetilde{f}(h\xi, h\eta)}_{f(\xi,\eta)}\mathrm{w}(\xi)\mathrm{w}(\eta).
\end{aligned}$$

□

With this in mind we make the following preliminary definition:

Definition 3.6.4 (preliminary)**.** *The map $S\colon \mathcal{B}(\mathcal{H}_2) \to \mathcal{B}(\mathcal{H}_1)$ is a Clifford channel if it is covariant with respect to two projective representations $\mathrm{w}_1(\xi_1) \in \mathcal{B}(\mathcal{H}_1)$, $\xi_1 \in h_1\mathcal{G} \subset \Xi_1$ and $\mathrm{w}_2(\xi_2) \in \mathcal{B}(\mathcal{H}_2)$, $\xi_2 \in h_2\mathcal{G} \subset \Xi_2$ of a group $\mathcal{G}$, which is also a two-dimensional vector space over a finite field. $(\mathrm{w}_i, e^{i\beta_i m}, \mathcal{H}_i)$ are the standard Weyl systems of Ξ_i over $\mathcal{H}_i$, where $\Xi_i = \bigoplus_x \mathbb{F}_{d_x}^2$ and $\mathcal{H}_i = \bigotimes_x \mathbb{C}^{d_x}$.*

We will however make an additional restriction. As in [52] we always require the input system to be irreducible. Only in this case all Clifford channels map Weyl operators to multiples of Weyl operators. If both input and output system are allowed to be reducible this is in general not the case. An easy example is the case where the h_i both map everything to 0. These surely are homomorphisms, but the representations we obtain only contain the identity, so any channel is covariant with respect to these representations. Furthermore we will show in the following that all channels obeying Definition 3.6.4 and mapping Weyl operators onto Weyl operators also obey the simpler Definition 3.6.5.

Our interest in Clifford channels is motivated by the search for quantum evolutions that allow an efficient classical description. Such a description is possible, if the evolution does not change the basis. This motivates the study of channels that map Weyl operators to scalar multiples of Weyl operators.

Let $S\colon \mathcal{B}(\mathcal{H}_2) \to \mathcal{B}(\mathcal{H}_1)$ be a Clifford channel with respect to $\mathrm{w}_1(\xi_1)$, $\forall \xi_1 \in \Xi_1 = h_1\mathcal{G}$ and $\mathrm{w}_2(\xi_2)$, $\forall \xi_2 \in \Xi_2 \subset h_2\mathcal{G}$. We chose $\Xi_1 = h_1\mathcal{G}$ to make the input system irreducible. Thus we can write every $\xi_1 \in \Xi_1$ as $h_1\xi$ with $\xi \in \mathcal{G}$. Using the covariance

condition (3.26) we can determine the commutation relations of $S[\mathrm{w}_2(\eta_2)]$ with arbitrary Weyl operators $\mathrm{w}_1(\xi_1)$.

$$\begin{aligned}\mathrm{w}_1(\xi_1)S[\mathrm{w}_2(\eta_2)] &= \mathrm{w}_1(h_1\xi)S[\mathrm{w}_2(\eta_2)]\mathrm{w}_1(h_1\xi)^*\mathrm{w}_1(h_1\xi)\\ &= S[\mathrm{w}_2(h_2\xi)\mathrm{w}_2(\eta_2)\mathrm{w}_2(h_2\xi)^*]\mathrm{w}_1(h_1\xi)\\ &= \mathrm{e}^{i\sigma_2(\eta_2,h_2\xi)}S[\mathrm{w}_2(\eta_2)]\mathrm{w}_1(\xi_1)\end{aligned}$$

for all $\eta_2 \in \Xi_2$ and all $\xi_1 \in \Xi_1$. $S[\mathrm{w}_2(\eta)]$ can be expressed as a sum of Weyl operators $\sum_i c_i \mathrm{w}_1(\eta_1^i)$ on $\mathcal{H}_1$ with $\mathrm{w}_1(\eta_1^i) \neq \mathrm{w}_1(\eta_1^j)$ for $i \neq j$. Again we determine the commutation relations

$$\mathrm{w}_1(\xi_1)\sum_i c_i \mathrm{w}_1(\eta_1^i) = \left(\sum_i c_i \mathrm{e}^{i\sigma_1(\eta_1^i,\xi_1)}\mathrm{w}_1(\eta_1^i)\right)\mathrm{w}_1(\xi_1).$$

The commutators are

$$\begin{aligned}[S[\mathrm{w}_2(\eta_2)],\mathrm{w}_1(\xi_1)] &= \left(1-\mathrm{e}^{i\sigma_2(\eta_2,h_2\xi)}\right)S[\mathrm{w}_2(\eta_2)]\mathrm{w}_1(\xi_1),\\ \left[\sum_i c_i\mathrm{w}_1(\eta_1^i),\mathrm{w}_1(\xi_1)\right] &= \left(\sum_i c_i\left(1-\mathrm{e}^{i\sigma_1(\eta_1^i,\xi_1)}\right)\mathrm{w}_1(\eta_1^i)\right)\mathrm{w}_1(\xi_1).\end{aligned}$$

For the sum to commute in the same way as $S[\mathrm{w}_2(\eta_2)]$ the following must hold:

$$\sum_i c_i\left(1-\mathrm{e}^{i\sigma_1(\eta_1^i,\xi_1)}\right)\mathrm{w}_1(\eta_1^i) = \left(\sum_i c_i\mathrm{w}_1(\eta_1^i)\right)\left(1-\mathrm{e}^{i\sigma_2(\eta,h_2\xi)}\right). \tag{3.27}$$

All the Weyl operators are linearly independent, so we can treat this equation as a set of equations, and either

$$c_i = 0 \quad \text{or} \quad \sigma_1(\eta_1^i,h_1\xi) = \sigma_2(\eta_2,h_2\xi)\ \forall\eta_2\in\Xi_2\ \forall\xi\in\mathcal{G}$$

has to hold for each i. σ_1 is non-degenerate, because it belongs to a standard Weyl system, so $\sigma_1(\eta_1^i,\xi_1) = \sigma_1(\eta_1^j,\xi_1)\ \forall\xi_1$ implies $\eta_1^i = \eta_1^j$ which is excluded by construction. Therefore only one c_i can be non-zero and $S[\mathrm{w}_2(\eta_2)] = c(\eta_2)\mathrm{w}_1(\eta_1(\eta_2))$ with some complex factor $c(\eta_2)$. The dependence of η_1 on η_2 is yet to be determined. But at this point it is already proven that Clifford channels with irreducible input systems map Weyl operators to multiples of Weyl operators.

The commutation relations can still give us more information about the possible transformations. We know that $\sigma_1(\eta_1(\eta_2),h_1\xi) = \sigma_2(\eta_2,h_2\xi)$, $\forall\xi\in\mathcal{G}$ and $\forall\eta_2\in\Xi_2$. The maps σ_1,σ_2, h_1 and h_2 are all linear in all arguments, which makes $\eta_1(\eta_2)$ a linear map. We write $\eta_1(\eta_2) = \mathbf{s}\eta_2$ or simply $\eta_1 = \mathbf{s}\eta_2$ where $\mathbf{s}$ is the classical description of S up to the coefficients $c(\eta_2)$, which we will call $\lambda(\eta_2)$ in the future. A Clifford

channel mapping from Weyl system $(\mathrm{w}_2, \mathrm{e}^{i\beta_2}, \mathcal{H}_2)$ to Weyl system $(\mathrm{w}_1, \mathrm{e}^{i\beta_1}, \mathcal{H}_1)$ that is covariant with respect to the representations $\mathrm{w}_1 \circ h_1$ and $\mathrm{w}_2 \circ h_2$ of a group $\mathcal{G}$, where w_1 is irreducible, has the form

$$S[\mathrm{w}_2(\eta_2)] = \lambda(\eta_2)\mathrm{w}_1(\mathbf{s}\eta_2) \tag{3.28}$$

We will now further simplify the setting by showing that given a channel of the form (3.28) it is covariant with respect to the representations $\mathrm{w}_1 \circ \mathbb{1}$ and $\mathrm{w}_2 \circ \mathbf{s}^\sigma$ of the group Ξ_1. $\mathbf{s}^\sigma$ is the symplectically adjoint map of $\mathbf{s}$ as introduced in Lemma 2.2.9. Let us check if this channel is Weyl covariant: let $B \in \mathcal{B}(\mathcal{H}_2)$, $B = \sum_i c_i \mathrm{w}_2(\eta_2^i)$. It follows that

$$\begin{aligned}
& S[\mathrm{w}_2(h_2\xi)B\mathrm{w}_2(h_2\xi)^*] \\
= & \sum_i c_i \mathrm{e}^{i\sigma_2(\eta_2^i, h_2\xi)} S[\mathrm{w}_2(\eta_2^i)] \\
= & \sum_i c_i \mathrm{e}^{i\sigma_2(\eta_2^i, h_2\xi)} \lambda(\eta_2^i)\mathrm{w}_1(\mathbf{s}\eta_2^i)\mathrm{w}_1(h_1\xi)\mathrm{w}_1(h_1\xi)^* \\
= & \sum_i c_i \mathrm{e}^{i\sigma_2(\eta_2^i, h_2\xi)} \mathrm{e}^{i\sigma_1(h_1\xi, \mathbf{s}\eta_2^i)} \lambda(\eta_2^i)\mathrm{w}_1(h_1\xi)\mathrm{w}_1(\mathbf{s}\eta_2^i)\mathrm{w}_1(h_1\xi)^* \\
= & \sum_i c_i \mathrm{e}^{i\left(\sigma_2(\eta_2^i, h_2\xi) - \sigma_1(\mathbf{s}\eta_2^i, h_1\xi)\right)} \mathrm{w}_1(h_1\xi) S[\mathrm{w}_2(\eta_2^i)] \mathrm{w}_1(h_1\xi)^*
\end{aligned}$$

S is Weyl covariant if and only if there exist $\mathcal{G}$, h_1, h_2 with $\mathrm{w}_1 \circ h_1 : \mathcal{G} \to U(\mathcal{H})$ irreducible such that

$$\sigma_2(\eta_2^i, h_2\xi) = \sigma_1(\mathbf{s}\eta_2^i, h_1\xi) \; \forall \eta_2^i \in \Xi_2 \; \forall \xi \in \mathcal{G}. \tag{3.29}$$

It turns out that we can always make a simple choice of $\mathcal{G}$. It can be chosen to equal Ξ_1 and $h_1 = \mathbb{1}$. We obtain

$$\sigma_2(\eta_2, h_2\xi) = \sigma_1(\mathbf{s}\eta_2, \xi) = \sigma_2(\eta, \mathbf{s}^\sigma\xi),$$

where we used the existence and uniqueness of the symplectically adjoint map $\mathbf{s}^\sigma$ as proven in Section 2.2.5. The map h_2 is uniquely determined to be

$$h_2 = \mathbf{s}^\sigma. \tag{3.30}$$

Now we can simplify Definition 3.6.4:

Definition 3.6.5. *$S : \mathcal{B}(\mathcal{H}_2) \to \mathcal{B}(\mathcal{H}_1)$ is a Clifford channel, if it is covariant with respect to two projective representations $\mathrm{w}_1(\xi_1) \in \mathcal{B}(\mathcal{H}_1)$, $\xi_1 \in \Xi_1 = \mathcal{G}$ and $\mathrm{w}_2(\xi_2) \in U(\mathcal{H}_2)$, $\xi_2 \in h\mathcal{G}$ of a group $\mathcal{G}$, which is also a two-dimensional vector space over a finite field. The w_i are the standard Weyl systems of Ξ_i on $\mathcal{H}_i$, where $\Xi_i = \bigoplus_j \mathbb{F}_{d_j}^2$ and $\mathcal{H}_i = \bigotimes_j \mathbb{C}^{d_j}$.*

The above can be combined in the following theorem:

Theorem 3.6.6 (Generalized version of [53])**.** *A channel $S:\ \mathcal{B}(\mathcal{H}_2) \to \mathcal{B}(\mathcal{H}_1)$ is a Clifford channel if and only if there exists a linear map $\mathbf{s}:\ \Xi_2 \to \Xi_1$ and a function $\lambda : \Xi_2 \to \mathbb{C}$ such that*

$$S[\mathrm{w}_2(\eta_2)] = \lambda(\eta_2)\mathrm{w}_1(\mathbf{s}\eta_2) \tag{3.31}$$

for all $\eta_2 \in \Xi_2$. The systems $(\mathrm{w}_1, e^{i\beta_1}, \mathcal{H}_1)$ and $(\mathrm{w}_2, e^{i\beta_2}, \mathcal{H}_2)$ are standard Weyl systems.[4]

Now that we have established the equivalence of Weyl covariant channels with irreducible input systems to channels of the form (3.28) we will only use the latter form, because it is useful for a classical description of the channels.

However, so far we have not investigated which conditions $\mathbf{s}$ and λ have to fulfill in order to make S a valid quantum channel—a completely positive and unital map. We will examine this in the next theorem.

Theorem 3.6.7. *Let $(\mathrm{w}_1, e^{i\beta_1}, \mathcal{H}_1)$ be the standard Weyl system over the phase space Ξ_1 and let $(\mathrm{w}_2, e^{i\beta_2}, \mathcal{H}_2)$ be the standard Weyl system over Ξ_2. Then a linear map $S:\ \mathcal{B}(\mathcal{H}_2) \to \mathcal{B}(\mathcal{H}_1)$ is a Clifford channel if and only if*

$$S[\mathrm{w}_2(\eta)] = \lambda(\eta)\mathrm{w}_1(\mathbf{s}\eta)\ \forall \eta \in \Xi_2 \tag{3.32}$$

holds for a linear map $\mathbf{s}$ and $\lambda : \Xi_2 \to \mathbb{C}$ with $\lambda(0) = 1$. Furthermore, the matrix

$$M_{ij} = \lambda(\eta_i - \eta_j)\frac{e^{i\beta_2(\eta_i - \eta_j, \eta_j)}}{e^{i\beta_1(\mathbf{s}\eta_i - \mathbf{s}\eta_j, \mathbf{s}\eta_j)}} \tag{3.33}$$

has to be positive definite for arbitrary η_i.

Proof. In Theorem 3.6.6 we already proved that every channel of the given form is Weyl covariant with respect to $(\mathrm{w}_1, \mathrm{e}^{i\beta_1}, \mathcal{H}_1)$ and the projective representation $\mathrm{w}_2 \circ \mathbf{s}^\sigma$ of Ξ_1 on $\mathcal{H}_2$. What remains to show is that the map is actually completely positive and unital. Proving unitality is straight forward, because

$$S[\mathbb{1}] = S[\mathrm{w}_2(0)] = \lambda(0)\mathbb{1} \stackrel{!}{=} \mathbb{1}$$

implies $\lambda(0) = 1$. To prove the second condition we use the Condition (2.11) from Section 2.1 which states that a channel is completely positive if and only if

$$\sum_{i,j} a_i^* S[b_i^* b_j] a_j \geqslant 0$$

[4] In the original theorem $\mathbf{s} = \mathrm{id}$ is assumed.

holds for all $a_i \in \mathcal{B}(\mathcal{H}_1)$ and all $b_i \in \mathcal{B}(\mathcal{H}_2)$. We decompose b_i into Weyl operators, $b_i = \sum_k c_k \mathrm{w}_2(\xi_k)^*$, and obtain $\sum_{i,j} \hat{a}_i^* S[\mathrm{w}_2(\xi_i)\mathrm{w}_2(\xi_j)^*]\hat{a}_j \geqslant 0$. Furthermore we decompose $\hat{a}_i$ into $\hat{a}_i = \mathrm{w}_1(\xi_i)\tilde{a}_i$. Thus (2.11) becomes

$$\begin{aligned}
0 &\leqslant \sum_{i,j} \tilde{a}_i^* \mathrm{w}_1(\mathbf{s}\eta_i)^* S[\mathrm{w}_2(\eta_i)\mathrm{w}_2(\eta_j)^*]\mathrm{w}_1(\mathbf{s}\eta_j)\tilde{a}_j \\
&= \sum_{i,j} \tilde{a}_i^* \mathrm{w}_1(\mathbf{s}\eta_i)^* S[\mathrm{w}_2(\eta_i-\eta_j)\mathrm{e}^{-i\beta_2(\eta_i,-\eta_j)}\mathrm{e}^{-i\beta_2(\eta_j,\eta_j)}]\mathrm{w}_1(\mathbf{s}\eta_j)\tilde{a}_j \\
&= \sum_{i,j} \tilde{a}_i^* \mathrm{w}_1(\mathbf{s}\eta_i)^* \lambda(\eta_i-\eta_j)\mathrm{w}_1(\mathbf{s}\eta_i - \mathbf{s}\eta_j) \\
&\quad \mathrm{e}^{-i\beta_2(\eta_i,-\eta_j)}\mathrm{e}^{-i\beta_2(\eta_j,\eta_j)}\mathrm{w}_1(\mathbf{s}\eta_j)\tilde{a}_j \\
&= \sum_{i,j} \lambda(\eta_i-\eta_j)\mathrm{e}^{-i\beta_2(\eta_i,-\eta_j)}\mathrm{e}^{-i\beta_2(\eta_j,\eta_j)}\mathrm{e}^{i\beta_1(\mathbf{s}\eta_i,-\mathbf{s}\eta_j)}\mathrm{e}^{i\beta_1(\mathbf{s}\eta_j,\mathbf{s}\eta_j)} \\
&\quad \tilde{a}_i^* \underbrace{\mathrm{w}_1(\mathbf{s}\eta_i)^*\mathrm{w}_1(\mathbf{s}\eta_i)}_{=\mathbb{1}} \underbrace{\mathrm{w}_1(\mathbf{s}\eta_j)^*\mathrm{w}_1(\mathbf{s}\eta_j)}_{=\mathbb{1}} \tilde{a}_j \\
&= \sum_{i,j} \tilde{a}_i^* \tilde{a}_j \lambda(\eta_i-\eta_j)\frac{\mathrm{e}^{i\beta_2(\eta_i-\eta_j,\eta_j)}}{\mathrm{e}^{i\beta_1(\mathbf{s}\eta_i-\mathbf{s}\eta_j,\mathbf{s}\eta_j)}}.
\end{aligned}$$

This is equivalent to the matrix

$$M_{ij} = \lambda(\eta_i-\eta_j)\frac{e^{i\beta_2(\eta_i-\eta_j,\eta_j)}}{e^{i\beta_1(\mathbf{s}\eta_i-\mathbf{s}\eta_j,\mathbf{s}\eta_j)}} \tag{3.34}$$

being positive definite, because a matrix M is positive definite if and only if it fulfills $a^*Ma > 0$ for all a. □

Remark 3.6.8. *For some combinations of Ξ_1 and Ξ_2 the only possible homomorphism is $h \equiv 0$. This follows directly from Theorem 2.2.8. Thereby for some combinations of input and output systems, e.g. qudits of different prime dimensions d_1 and d_2, the only possible Clifford channel is the completely depolarizing channel.*

Clifford channels are also often denoted as Weyl-covariant channels. They have been studied for some time e.g. in [28, 52, 54, 53] amongst others. A special class of Clifford channels, the channels where all the Kraus operators are Weyl operators, have been studied under the name Pauli-diagonal channels in [55, 56, 57].

In the following our aim is to draw conclusions about valid functions $\lambda(\xi)$ from Equation (3.33) which is in general difficult. Therefore we will consider several different special cases.

3.6.1 Special types of Clifford channels

We will now restrict the free parameters of our Clifford channel model to describe different types of channels. For each type we will evaluate Condition (3.33).

Mixed systems

On mixed systems (systems that consist of qudits with different dimensions) the homomorphism $\mathbf{s}$ of a Clifford channel S decomposes into parts that only act on qudits of the same dimension. For phase space vectors $\Xi_{2,p}$ and $\Xi_{2,q}$, $p \neq q$ from different dimensional subsystems the condition (3.33) simplifies to

$$M_{ij} = \lambda(\eta_i - \eta_j)\frac{\mathrm{e}^{-i\beta_2(\eta_j,\eta_j)}}{\mathrm{e}^{-i\beta_1(\mathbf{s}\eta_j,\mathbf{s}\eta_j)}} \; \forall \eta_i \in \Xi_{2,p} \; \forall \eta_j \in \Xi_{2,q},\; p \neq q. \tag{3.35}$$

However, the factors λ might still depend on the whole phase space, which allows information transfer. In general a Clifford channel over mixed systems can not be decomposed into channels on the subsystems of the same dimension despite the phase space homomorphism being decomposable. The interaction between the subsystems is limited to the factor λ.

Homomorphisms

An algebraic homomorphism $S:\ \mathfrak{B} \to \mathfrak{A}$ fulfills $S[ab] = S[a]S[b]$ and $S[a+b] = S[a] + S[b]$ $\forall a, b \in \mathfrak{B}$. If S is a quantum channel it is a linear map and the second condition is fulfilled. With this in mind the first condition can be evaluated on a basis of $\mathfrak{B}$. Furthermore every channel that fulfills $S[ab] = S[a]S[b]$ on $\mathfrak{B}$ is also a $*$-homomorphism and $S[a^*] = S[a]^*$ (see e.g. [25], Theorem 3.18(iii)).

Let $S:\ \mathcal{B}(\mathcal{H}_2) \to \mathcal{B}(\mathcal{H}_1)$ be a homomorphic Clifford channel between the Weyl systems $(\mathrm{w}_1, \mathrm{e}^{i\beta_1}, \mathcal{H}_1)$ and $(\mathrm{w}_2, \mathrm{e}^{i\beta_2}, \mathcal{H}_2)$. S is unital and

$$S[\mathbb{1}] = S[\mathrm{w}_2(\eta)^* \mathrm{w}_2(\eta)] = S[\mathrm{w}_2(\eta)]^* S[\mathrm{w}_2(\eta)] = \overline{\lambda(\eta)}\lambda(\eta)\mathbb{1} \stackrel{!}{=} \mathbb{1}. \tag{3.36}$$

Therefore, we have

$$\overline{\lambda(\eta)} = \lambda(\eta)^{-1} \tag{3.37}$$

and finally obtain $|\lambda(\eta)| = 1$.

To determine further conditions on λ we use

$$\begin{aligned} S[\mathrm{w}_2(\xi)]S[\mathrm{w}_2(\eta)] &= \lambda(\xi)\lambda(\eta)\mathrm{w}_1(\mathbf{s}\xi)\mathrm{w}_1(\mathbf{s}\eta) \\ &= \tfrac{\lambda(\xi)\lambda(\eta)}{\lambda(\xi+\eta)} S[\mathrm{w}_2(\xi+\eta)]\mathrm{e}^{-i\beta_1(\mathbf{s}\xi,\mathbf{s}\eta)} \\ &= S[\mathrm{w}_2(\xi)\mathrm{w}_2(\eta)]\tfrac{\lambda(\xi)\lambda(\eta)}{\lambda(\xi+\eta)}\mathrm{e}^{i\beta_2(\xi,\eta)}\,\mathrm{e}^{-i\beta_1(\mathbf{s}\xi,\mathbf{s}\eta)} \end{aligned}$$

and obtain

$$\lambda(\xi+\eta) = \lambda(\xi)\lambda(\eta)\frac{\mathrm{e}^{i\beta_2(\xi,\eta)}}{\mathrm{e}^{i\beta_1(\mathbf{s}\xi,\mathbf{s}\eta)}}. \tag{3.38}$$

In group theory this is known as the coboundary condition [27]. Furthermore, a homomorphic channel has to preserve the *-operation. Together with

$$\begin{aligned} & S[\mathrm{w}_2(\xi)^*] &= S[\mathrm{w}_2(\xi)]^* \\ \Leftrightarrow \quad & \lambda(-\xi)\mathrm{w}_1(-\mathbf{s}\xi)\mathrm{e}^{i\beta_2(-\xi,\xi)} &= \bar{\lambda}(\xi)\mathrm{e}^{i\beta(-\mathbf{s}\xi,\mathbf{s}\xi)}\mathrm{w}_1(-\mathbf{s}\xi) \end{aligned}$$

this leads us to the condition

$$\lambda(-\xi) = \overline{\lambda(\xi)} \frac{e^{i\beta_2(\xi,\xi)}}{e^{i\beta_1(\mathbf{s}\xi,\mathbf{s}\xi)}}. \tag{3.39}$$

This condition introduces no further restrictions, since we can derive it from the coboundary condition (3.38) using $\lambda(0) = 1$ and (3.37). This coincides with the fact that any completely positive homomorphism is also a *-homomorphism.

A homomorphic channel also preserves the commutation relations between operators. We can use this to gain information about $\mathbf{s}$. On the one hand we have

$$\begin{aligned} S[\mathrm{w}_2(\xi)]S[\mathrm{w}_2(\eta)] &= \lambda(\xi)\lambda(\eta)\mathrm{w}_1(\mathbf{s}\xi)\mathrm{w}_1(\mathbf{s}\eta) \\ &= \lambda(\xi)\lambda(\eta)\mathrm{w}_1(\mathbf{s}\eta)\mathrm{w}_1(\mathbf{s}\xi)e^{i\sigma_1(\mathbf{s}\eta,\mathbf{s}\xi)} \\ &= S[\mathrm{w}_2(\eta)]S[\mathrm{w}_2(\xi)]e^{i\sigma_1(\mathbf{s}\eta,\mathbf{s}\xi)}. \end{aligned} \tag{3.40}$$

But on the other hand

$$\begin{aligned} S[\mathrm{w}_2(\xi)]S[\mathrm{w}_2(\eta)] &= S[\mathrm{w}_2(\xi)\mathrm{w}_2(\eta)] \\ &= S[\mathrm{w}_2(\eta)\mathrm{w}_2(\xi)]e^{i\sigma_2(\eta,\xi)} \\ &= S[\mathrm{w}_2(\eta)]S[\mathrm{w}_2(\xi)]e^{i\sigma_2(\eta,\xi)} \end{aligned} \tag{3.41}$$

also holds. From (3.40) and (3.41) we deduce

$$\sigma_1(\mathbf{s}\eta,\mathbf{s}\xi) = \sigma_2(\eta,\xi)\ \forall \xi,\eta \in \Xi_2. \tag{3.42}$$

This condition can also be derived from the coboundary condition (3.38). This is not surprising since the preservation of commutation relations is a consequence of S being a homomorphism, which in turn lead us to the coboundary condition.

Now assume that $\mathbf{s}\xi = \mathbf{s}\eta$ for some $\xi,\eta \in \Xi_2$. Then $\sigma_1(\mathbf{s}\xi - \mathbf{s}\eta, \mathbf{s}\zeta) = 0$ and $\sigma_2(\xi - \eta,\zeta) = 0\ \forall \zeta \in \Xi_2$. But that means that $\xi = \eta$ because σ_2 is non-degenerate. Thus $\mathbf{s}$ is injective and Ξ_1 contains an isomorphic image of Ξ_2. Restricting Ξ_1 to $\mathbf{s}\Xi_2$ we obtain a reversible channel (see next section).

Let us combine our findings into a theorem and prove that these conditions are not only necessary but also sufficient:

Theorem 3.6.9. *Let $(\mathrm{w}_1, e^{i\beta_1}, \mathcal{H}_1)$ and $(\mathrm{w}_2, e^{i\beta_2}, \mathcal{H}_2)$ be standard Weyl systems over the phase spaces Ξ_1 and Ξ_2, where Ξ_1 contains a homomorphic image of Ξ_2. It follows that $S\colon \mathcal{B}(\mathcal{H}_2) \to \mathcal{B}(\mathcal{H}_1)$, $S[\mathrm{w}_2(\xi)] = \lambda(\xi)\mathrm{w}_1(\mathbf{s}\xi)$ with a linear map $\mathbf{s}\colon \Xi_2 \to \Xi_1$ and a function $\lambda\colon \Xi_2 \to \mathbb{C}$ is a homomorphic Clifford channel if and only if the following holds $\forall \xi,\eta \in \Xi_2$:*

1. $\lambda(0) = 1$,
2. $|\lambda(\xi)| = 1 \quad \Leftrightarrow \quad \overline{\lambda(\xi)} = \lambda(\xi)^{-1}$,
3. $\lambda(\xi+\eta) = \lambda(\xi)\lambda(\eta)\frac{e^{i\beta_2(\xi,\eta)}}{e^{i\beta_1(\mathbf{s}\xi,\mathbf{s}\eta)}}$ *(the coboundary condition),*

4. $\mathbf{s}$ *is injective,*

5. $\sigma_1(\mathbf{s}\xi,\mathbf{s}\eta)=\sigma_2(\xi,\eta)$.

Proof. We already proved that conditions 1-5 are necessary. To show that they are also sufficient we use the following approach: from Theorem 3.6.7 we know that (a) $\lambda(0)=1$ has to hold for S to be unital and (b) the matrix

$$M_{ij}=\lambda(\eta_i-\eta_j)\frac{\mathrm{e}^{i\beta_2(\eta_i-\eta_j,\eta_j)}}{\mathrm{e}^{i\beta_1(\mathbf{s}\eta_i-\mathbf{s}\eta_j,\mathbf{s}\eta_j)}}$$

has to be positive definite for S to be completely positive. S is required to be a homomorphism thus (c) $S[ab]=S[a]S[b]$ has to hold.

(a) is a direct requirement, so nothing has to be proven. To prove (b) we need an additional relation on λ which we obtain from condition 2 using $\eta=-\xi$:

$$\lambda(-\xi)=\lambda(\xi)^{-1}\frac{\mathrm{e}^{i\beta_2(\xi,\xi)}}{\mathrm{e}^{\xi\beta_1(\mathbf{s}\xi,\mathbf{s}\xi)}}.$$

Now using this and condition 2 we have

$$\begin{aligned}
M_{ij} &= \lambda(\eta_i-\eta_j)\frac{\mathrm{e}^{i\beta_2(\eta_i-\eta_j,\eta_j)}}{\mathrm{e}^{i\beta_1(\mathbf{s}\eta_i-\mathbf{s}\eta_j,\mathbf{s}\eta_j)}}\\
&= \lambda(\eta_i)\lambda(-\eta_j)\frac{\mathrm{e}^{-i\beta_2(\eta_i,\eta_j)}}{\mathrm{e}^{-i\beta_1(\mathbf{s}\eta_i,\mathbf{s}\eta_j)}}\cdot\frac{\mathrm{e}^{i\beta_2(\eta_i-\eta_j,\eta_j)}}{\mathrm{e}^{i\beta_1(\mathbf{s}\eta_i-\mathbf{s}\eta_j,\mathbf{s}\eta_j)}}\\
&= \lambda(\eta_i)\lambda(-\eta_j)\frac{\mathrm{e}^{-i\beta_2(\eta_j,\eta_j)}}{\mathrm{e}^{-i\beta_1(\eta_j\eta_j)}}\\
&= \frac{\lambda(\eta_i)}{\lambda(\eta_j)}.
\end{aligned}$$

We can write the matrix M as a dyadic product $M=A^*A$ with $A=(\overline{\lambda(\eta_1)},\ldots,\overline{\lambda(\eta_n)})$. Thus it is positive definite and S completely positive. Now all that is left to show is that S is a homomorphism. As S is linear it is sufficient to show that it is a homomorphism on Weyl operators:

$$\begin{aligned}
S[\mathrm{w}_2(\xi)\mathrm{w}_2(\eta)] &= S[\mathrm{w}_2(\xi+\eta)]\mathrm{e}^{-i\beta_2(\xi,\eta)}\\
&= \lambda(\xi+\eta)\mathrm{w}_1(\mathbf{s}\xi+\mathbf{s}\eta)\mathrm{e}^{-i\beta_2(\xi,\eta)}\\
&= \underbrace{\lambda(\xi+\eta)\frac{1}{\lambda(\xi)\lambda(\eta)}\frac{\mathrm{e}^{-i\beta_2(\xi,\eta)}}{\mathrm{e}^{-i\beta_1(\mathbf{s}\xi,\mathbf{s}\eta)}}}_{=1}S[\mathrm{w}_2(\xi)]S[\mathrm{w}_2(\eta)].
\end{aligned}$$

Evaluating the coboundary condition with exchanged roles of ξ and η we have

$$\begin{aligned} & & \lambda(\xi+\eta) &= \lambda(\eta+\xi) \\ \Leftrightarrow \quad & \lambda(\xi)\lambda(\eta)\frac{\mathrm{e}^{i\beta_2(\xi,\eta)}}{\mathrm{e}^{i\beta_1(\mathbf{s}\xi,\mathbf{s}\eta)}} &= \lambda(\xi)\lambda(\eta)\frac{\mathrm{e}^{-i\beta_2(\xi,\eta)}}{\mathrm{e}^{-i\beta_1(\mathbf{s}\xi,\mathbf{s}\eta)}} \\ \Leftrightarrow \quad & \mathrm{e}^{i\beta_2(\xi,\eta)-i\beta_2(\eta,\xi)} &= \mathrm{e}^{i\beta_1(\mathbf{s}\xi,\mathbf{s}\eta)-i\beta_1(\mathbf{s}\eta,\mathbf{s}\xi)} \\ \Leftrightarrow \quad & \sigma_2(\xi,\eta) &= \sigma_1(\mathbf{s}\xi,\mathbf{s}\eta) \quad \forall \xi,\eta\in\Xi_2. \end{aligned}$$

As shown above this can only hold if $\mathbf{s}$ is injective. This ends our proof. □

Homomorphic channels on mixed systems have an especially simple structure.

Lemma 3.6.10. *Let $S:\ \mathcal{B}(\mathcal{H}_2)\to\mathcal{B}(\mathcal{H}_1)$ be a homomorphic Clifford channel on mixed Weyl systems $(\mathrm{w}_1, e^{i\beta_1}, \mathcal{H}_1)$ and $(\mathrm{w}_2, e^{i\beta_2}, \mathcal{H}_2)$ over the phase spaces*

$$\Xi_1 = \bigoplus_p \mathbb{Z}_{d_p}^{2n_p} \quad \text{and} \quad \Xi_2 \bigoplus_q \mathbb{Z}_{d_q}^{2n_q},$$

where Ξ_1 contains a homomorphic image of Ξ_2 Then S can be decomposed into a tensor product of channels S_p acting on the subsystems of qudits with dimension p:

$$\begin{aligned} S[\mathrm{w}_2(\eta)] &= \bigotimes_p S_p[\mathrm{w}_{2,p}(\eta_p)],\ \forall\eta\in\Xi_2 \\ &= \bigotimes_p \lambda_p(\eta_p)\mathrm{w}_{1,p}(\mathbf{s}_p\eta_p),\ \forall\eta\in\Xi_2, \end{aligned} \tag{3.43}$$

where η_p is the restriction of η to the subspace of the phase space over $\mathbb{Z}_p$ and similar for $\mathrm{w}_{i,p}$.

Proof. From Theorem 3.6.9 we know that S has to fulfill a set of conditions to be a homomorphic Clifford channel. To prove our lemma we only use the condition

$$\lambda(\xi+\eta)=\lambda(\xi)\lambda(\eta)\mathrm{e}^{i\beta_2(\xi,\eta)}\mathrm{e}^{-i\beta_1(\mathbf{s}\xi,\mathbf{s}\eta)}\ \forall\xi,\eta\in\Xi_2$$

Now let $\xi\in\Xi_{2,p}$ and $\eta\in\Xi_{2,q}$, $p\neq q$. Then $\mathrm{e}^{i\beta_2(\xi,\eta)}=1$ and because of the decomposition of homomorphisms (Theorem 2.2.8) also $\mathrm{e}^{i\beta_1(\mathbf{s}\xi,\mathbf{s}\eta)}=1$. Therefore

$$\lambda(\xi+\eta)=\lambda(\xi)\lambda(\eta)\ \forall\xi\in\Xi_{2,p},\eta\in\Xi_{2,q},\ p\neq q.$$

Together with the homomorphism decomposition of $\mathbf{s}\eta$ and the local structure of w_1 this immediately gives (3.43) and therefore the theorem. □

Reversible channels

Reversible channels are a special case of homomorphic channels. For S to be reversible it has to be an automorphism. For a Clifford channel this means that $\mathbf{s}\Xi_2 = \Xi_1$ and $\mathbf{s}$ is an isomorphism. Therefore the two systems are the same up to a reordering of subsystems. Everything we know from the homomorphic case holds also in the reversible case. The condition on the symplectic forms now holds for all phase space vectors in both Ξ_2 and Ξ_1, as $\mathbf{s}\Xi_2 = \Xi_1$:

$$\sigma_1(\mathbf{s}\eta, \mathbf{s}\xi) = \sigma_2(\eta, \xi) \quad \forall\, \xi, \eta \in \Xi_2. \tag{3.44}$$

Therefore $\mathbf{s}$ preserves the symplectic form and thus is a symplectic transformation. $\mathbf{s}$ is an isomorphism[5], which makes $\mathbf{s}^\sigma$ a symplectic transformation too.

In the reversible case a theorem analogous to Theorem 3.6.9 holds. The only change is that now the two phase spaces are isomorphic and $\mathbf{s}\Xi_2 = \Xi_1$. Similarly a factorization lemma in the sense of Lemma 3.6.10 holds. All the proofs work in exactly the same way so we do not state them here.

Example: the depolarizing channel

The depolarizing channel transmits an arbitrary quantum state perfectly with probability $1-p$ and replaces it with the maximally mixed state with probability $0 \leqslant p \leqslant 1$. On the level of density matrices ρ the transformation reads

$$\rho \mapsto \tilde{\rho} = (1-p)\rho + p\frac{\mathbb{1}}{d}. \tag{3.45}$$

To find the corresponding transformation in the Heisenberg picture we start with

$$\begin{aligned}
\mathrm{tr}\,(\tilde{\rho}A) &= \mathrm{tr}\left(\left((1-p)\rho + p\frac{\mathbb{1}}{d}\right)A\right) \\
&= (1-p)\mathrm{tr}\,(\rho A) + p\frac{1}{d}\mathrm{tr}\,(A) \\
&= (1-p)\mathrm{tr}\,(\rho A) + p\frac{1}{d}\mathrm{tr}\,(A)\underbrace{\mathrm{tr}\,(\rho)}_{=1} \\
&= \mathrm{tr}\left(\rho\left((1-p)A + p\frac{\mathbb{1}}{d}\mathrm{tr}\,(A)\right)\right)
\end{aligned}$$

and find

$$\tilde{A} = (1-p)A + p\frac{\mathbb{1}}{d}\mathrm{tr}\,(A). \tag{3.46}$$

[5] Actually, symplectic transformations between vector spaces of the same dimension are always isomorphisms [29].

The input and output systems are the same, and so are the Weyl systems on input and output. All Weyl operators except the identity are traceless (see Theorem 2.2.5). This gives us a simplified transformation

$$\widetilde{\mathrm{w}(\xi)} = \begin{cases} (1-p)\mathrm{w}(\xi) & \text{for} \quad \xi \neq 0 \\ \mathrm{w}(\xi) & \text{for} \quad \xi = 0 \,(\mathrm{w}(0) = \mathbb{1}) \end{cases} . \tag{3.47}$$

In the Clifford description this corresponds to $\mathbf{s} = \mathrm{id}$. Because $S[\mathrm{w}(\xi)] = \lambda(\xi)\mathrm{w}(\xi)$ we have

$$\lambda(\xi) = \begin{cases} (1-p) & \text{for} \quad \xi \neq 0 \\ 1 & \text{for} \quad \xi = 0 \,(\mathrm{w}(0) = \mathbb{1}) \end{cases} . \tag{3.48}$$

In the case of the completely depolarizing channel ($p = 1$) we have $\lambda = 0$ for all $\xi \neq 0$.

3.6.2 Completion of partly defined Clifford channels

In this section we will present an algorithm to find a completion of a partly defined homomorphic Clifford channel to a unitary Clifford channel. This completion is useful if we have the stabilizers of a stabilizer quantum error-correction code given and want to find a complete encoding transformation.

We will only consider the phase space transformation, because finding fitting complex phases is always possible. The partly defined homomorphic Clifford channel is described by a linear mapping from one subspace of the phase space to another subspace while preserving the symplectic form. This map has to be completed to a symplectic map acting on the whole phase space, which then describes a unitary completion of the partly defined channel.

Let Ξ be the n-qubit phase space with the standard symplectic form σ and let Δ_1 and Δ_2 be subspaces of Ξ of the same dimension. Furthermore let $\mathbf{r} : \Delta_2 \to \Delta_1$ be a linear map such that $\sigma(\mathbf{r}\eta_i, \mathbf{r}\eta_j) = \sigma(\eta_i, \eta_j)$ for all $\eta_i, \eta_j \in \Delta_2$.

The existence of a symplectic completion is a direct consequence of Witt's theorem which we present here as in [30], Corollary 2.2.8. In [30] the underlying field is assumed to have characteristic zero, however the result also holds for characteristic p.

Lemma 3.6.11. *Let (Ξ_1, σ_1) and (Ξ_2, σ_2) be two symplectic vector spaces of the same dimension $2n$, and let $\Delta_i \subset \Xi_i$ be two subspaces of the same dimension k. Let $\mathbf{r} : \Delta_1 \to \Delta_2$ be a linear isomorphism such that*

$$\sigma_2(\mathbf{r}\xi, \mathbf{r}\eta) = \sigma_1(\xi, \eta) \quad \forall \xi, \eta \in \Delta_1. \tag{3.49}$$

Then $\mathbf{r}$ can be extended to a symplectic isomorphism $\mathbf{s} : \Xi_1 \to \Xi_2$.

In our case $\Xi_2 = \Xi_1$, $\sigma_2 = \sigma_1$.

The transformation is given is by a set of rules $B_i \mapsto A_i$, $i \in \Lambda$ which fulfills the following properties:

- A_i and B_i are elements of the same algebra $\mathfrak{A}(\Xi)$, where Ξ is the n-qubit phase space. In phase space the rules are represented by $\eta_i \mapsto \xi_i$ with $\xi_i \in \Delta_1$ and $\eta_i \in \Delta_2$.
- The transformation preserves the commutation relations, therefore we have $\left[A_i, A_j\right] = \left[B_i, B_j\right]$ and in phase space $\sigma(\xi_i, \xi_j) = \sigma(\eta_i, \eta_j)$.

We will use the set of transformation rules

$$\begin{aligned} XII &\mapsto IXI, \\ ZII &\mapsto ZZZ, \\ IZI &\mapsto IXX, \end{aligned}$$

as an example througout this section. In phase space we have

$$\begin{aligned} (1,0,0,0,0,0) &\mapsto (0,0,1,0,0,0), \\ (0,1,0,0,0,0) &\mapsto (0,1,0,1,0,1), \\ (0,0,0,1,0,0) &\mapsto (0,0,1,0,1,0). \end{aligned}$$

The following algorithm completes the set of rules to describe the transformation rules of a Clifford unitary. The existence of elements we pick during the algorithm is always guaranteed by Lemma 3.6.11.

- Determine the numbers p_g of given pairs of rules and s_g of given single rules as well as the number of unknown pairs and unknown single rules. A pair consists of two rules $\eta_i \mapsto \xi_i$ and $\eta_j \mapsto \xi_j$ with $\sigma(\eta_i, \eta_j) = 1$ while a single rule fulfills $\sigma(\eta_i, \eta_j) = 0$ for all j. The numbers are given by the commutation matrix C, $c_{ij} = \sigma(\eta_i, \eta_j)$. Each pair (η_i, η_j) gives $c_{ij} = c_{ji} = 1$. As the members of pairs commute with all other operators the rows i and j as well as the columns i and j are zero in all other positions. Therefore the rank of C equals twice the number of pairs: $p_g = \mathrm{rank}(C)/2$. The dimension of C equals the total number of rules. Therefore $s_g = \dim(C) - \mathrm{rank}(C)$.
- The number of needed single rules equals the number of given single rules, because they have to be completed to pairs: $s_n = s_g$, while the number of needed pairs is determined by the dimension, the number of given pairs, and the number of given single rules. We need n pairs in total, one for each qubit. p_g pairs are given from the beginning, another s_g pairs are created from the s_g given single rules and the new single rules: $p_n = n - p_g - s_g$. In the example we have

$$C = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{3.50}$$

  and thus $p_g = 1$, $s_g = 1$, $s_n = 1$, and $p_n = 1$.

- Reorder the rules such that rules 1 through $2p_g$ belong to pairs while rules $2p_g+1$ through $2p_g+s_g$ are single rules.

- If $p_g > 0$, remove all pairs, because they do not have to be changed. To ensure that all new rules commute with the pairs, compute the commutant of the pairs on input and output system. In phase space this corresponds to the symplectic complement. The commutant on the input side is defined by the phase space $\Xi^{in}_{c_{p_g}} = \left\{\zeta \in \Xi | \sigma(\zeta,\xi_i) = 0 \; \forall \xi_i, 1 \leqslant i \leqslant 2p_g\right\}$. On the output the commutant is given by $\Xi^{out}_{c_{p_g}} = \left\{\zeta \in \Xi | \sigma(\zeta,\eta_i) = 0 \; \forall \eta_i, 1 \leqslant i \leqslant 2p_g\right\}$. If $p_g = 0$, $\Xi^{in}_0 = \Xi^{out}_0 = \Xi$

- If $s_g > 0$, do the following for each single rule $\eta_i \mapsto \xi_i$, $2p_g+1 \leqslant i \leqslant 2p_g+s_g$:
  - A pairing rule for rule i, $\eta_i \mapsto \xi_i$, has to anticommute with rule i and commute with all other single rules j, $i < j \leqslant 2p_g+s_g$. First compute the symplectic complement $\Xi^{in}_{c_{i-1},t}$ of ξ_j, $i < j \leqslant 2p_g+s_g$ in $\Xi^{in}_{c_{i-1}}$ and the symplectic complement $\Xi^{out}_{c_{i-1},t}$ of η_j,$i < j \leqslant 2p_g+s_g$ in $\Xi^{out}_{c_{i-1}}$. Take elements $\xi_t \in \Xi^{in}_{c_{i-1},t}$ and $\eta_t \in \Xi^{out}_{c_{i-1},t}$ with $\sigma(\xi_i,\xi_t) = 1$ and $\sigma(\eta_i,\eta_t) = 1$. This gives a new rule $\eta_t \mapsto \xi_t$ which forms a pair with rule i. Assign the index $t = i + s_g$.
  - Calculate the symplectic complement $\Xi^{in}_{c_i}$ of $\left\{\xi_i, \xi_{i+s_g}\right\}$ in $\Xi^{in}_{c_{i-1}}$ and the symplectic complement $\Xi^{\text{out}}_{c_i}$ of $\left\{\eta_i, \eta_{i+s_g}\right\}$ in $\Xi^{out}_{c_{i-1}}$.
  - If $i < 2p_g+s_g$, increase i by one and repeat the process, else go to the next step.

  In the example the rule $(0,0,1,0,0,0) \mapsto (0,0,0,0,0,1)$ respectively $IXI \mapsto IIZ$ is added to form the second pair.

- If $p_g + s_g < n$, additional pairs have to be found. Repeat the following for $p_g+s_g+1 \leqslant i \leqslant n$.
  - Take elements $\xi_{2i-1} \in \Xi^{\text{in}}_{c_{i-1}}$ and $\eta_{2i-1} \in \Xi^{\text{out}}_{c_{i-1}}$.
  - Take ξ_{2i} from $\Xi^{\text{in}}_{c_{i-1}}$ with $\sigma(\xi_{2i-1},\xi_{2i}) = 1$ and η_{2i} analogously.
  - $\eta_{2i-1} \mapsto \xi 2i-1$ and $\eta_{2i} \mapsto \xi 2i$ form a new pair of rules.
  - Calculate the symplectic complement $\Xi^{\text{in}}_{c_i}$ of $\{\xi_{2i-1},\xi_{2i}\}$ in $\Xi^{\text{in}}_{c_{i-1}}$ and analogously for the output side.
  - If $i < n$, increase i by one and repeat, else go to the next step.

  The example transformation can be completed by the rules $(0,0,0,0,1,0) \mapsto (0,1,0,0,0,0)$ respectively $IIX \mapsto ZII$ and $(0,0,0,0,0,1) \mapsto (1,0,1,0,0,0)$ respectively $IIZ \mapsto XXI$.

- Finally we transform the rules such that the output side is in the standard basis. The input sides of the rules then are the columns of the transformation matrix. The example transformation is

$$\mathbf{s} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

In practice the output side (or sometimes the input side) of the rules would usually be given in the standard basis. This eases the computation of the commutant and finding pairing elements for existing rules as well as new pairs, because in the standard basis it is immediately clear, which elements commute with a given set of elements and which do not commute.

3.6.3 Decomposition of Clifford channels

Here we present an algorithm to decompose a Clifford channel into a circuit made up of CNOT and local gates. Algorithms of this kind have been know for quite some time for circuits of qubits and qudits [58, 59, 60, 61]. Here we will briefly explain such an algorithm in a form that we need for the minimal memory decomposition of Clifford operations in Section 3.8. Our algorithm will only cover the case of qubits.

We have the following operations that are a basis of all Clifford operations

Hadamard The Hadamard gate

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{3.51}$$

implements the transformation $X \mapsto Z, Z \mapsto X$ on the Pauli matrices. In phase space it is represented by

$$\boldsymbol{h} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{3.52}$$

Phase The phase gate

$$P = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \tag{3.53}$$

implements the transformation $X \mapsto Y, Z \mapsto Z$. In phase space it is represented by

$$\boldsymbol{p} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}. \tag{3.54}$$

CNOT The CNOT gate is the only two qubit gate that is required to complete the basis. Here we use qubit one as source and qubit two as the target. Reversing this order works analogously. The CNOT is represented by the following matrix (in the standard basis)

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{3.55}$$

The action of the CNOT on Pauli operators is

$$\begin{aligned} XI &\mapsto XX, \\ ZI &\mapsto ZI, \\ IX &\mapsto IX, \\ IZ &\mapsto ZZ. \end{aligned}$$

In the phase space representation the CNOT has the following matrix

$$\boldsymbol{cnot} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3.56}$$

CZ The the controlled-Z (CZ) gate is not used in the decomposition algorithm. However, it plays an important rule in the circuits of CQCAs (see Section 5.7). It is represented by the unitary

$$\mathrm{C}Z = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}. \tag{3.57}$$

The action of the CZ on Pauli operators is

$$\begin{aligned} XI &\mapsto XZ, \\ ZI &\mapsto ZI, \\ IX &\mapsto ZX, \\ IZ &\mapsto IZ. \end{aligned}$$

In phase space the CZ is represented by the matrix

$$\mathbf{cz} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}. \tag{3.58}$$

We want to use these operations to apply a modified Gaussian algorithm (that respects and preserves symplecticity) to perform the decomposition. Thus we will operate on the circuit matrix by multiplication with the matrices of basic operations from the left to add and swap rows. For circuits on more than two qubits we tensor the gates with the identity operation on all qubits that are not affected by the gate. In phase space this corresponds to a direct sum with the identity on the additional systems. We will extend the notation of the operations to denote which qubits they are working on. For the phase gate in phase space we use $\boldsymbol{p}(i)$ to denote that it is acting on qubit i, where we start counting at 0. The Hadamard gate will have an analogous notation. The CNOT gate will be denoted by $\boldsymbol{cnot}(i,j)$ where i is the source qubit and j is the target.

In the classical description the rows of the phase space matrix $\boldsymbol{s}$ of a Clifford circuit S correspond to Pauli matrices on the individual qubits. Row $2i$ corresponds to X_i and we will denote it both by $\boldsymbol{s}_{2i}$ and x_i. Row $2i+1$ corresponds to Z_i and we will denote it both by $\boldsymbol{s}_{2i+1}$ and z_i.

Acting from the left the operations have the following effects on a Clifford circuit matrix $\boldsymbol{s}_{i,j}$. The

Hadamard: $\boldsymbol{h}(i)$ swaps the $2i$th and $(2i+1)$th row. (swaps x_i and z_i)

Phase: $\boldsymbol{p}(i)$ adds the $(2i+1)$th row to the $2i$th row. (adds z_i to x_i)

CNOT: $\boldsymbol{cnot}(i,j)$ adds row $2i$ to row $2j$ and row $2j+1$ to $2i+1$. (adds x_i to x_j and z_j to x_j)

We cannot perform a standard Gaussian algorithm with these basic operations, because the Gaussian algorithm does not preserve symplecticity of the matrix. One can easily understand this, when thinking of the action on Pauli operators. If we add row $2i$ to row $2j$ we multiply the image of X_j by the image of X_i. But the image of X_j, $\bar{X}_j$ has to commute with the image of Z_i. That is no longer the case when we replace $\bar{X}_j$ by $\bar{X}_j\bar{X}_i$, because $\bar{X}_i$ anticommutes with $\bar{Z}_i$. Thus the operation does not preserve symplecticity. We can preserve the commutation relations and therefore the symplecticity by also replacing $\bar{Z}_i$ by $\bar{Z}_i\bar{Z}_j$. Thus if adding row $2i$ to row $2j$ we also have to add row $2j+1$ to row $2i+1$. This is exactly what a $\boldsymbol{cnot}(i,j)$ does. However, these limitations to the set of admissible row operations does not pose any problem. The starting point is always a symplectic matrix which stays symplectic during the whole process. The reduced degrees of freedom allow for a Gaussian algorithm with a reduced set of operations. This set is shown in Table 3.1.

The algorithm to decompose the Clifford operation into a circuit of basic operations works as follows:

1. If $\boldsymbol{s}_{0,0} = 0$ then there has to be a $\boldsymbol{s}_{i,0} \neq 0$. If i even, add $x_{i/2}$ to x_0, else add $z_{(i-1)/2}$ to x_0. If $\boldsymbol{s}_{00} = 1$ do nothing in this step.

Matrix operation	Implementation	side effects
add x_i to x_j $(i \neq j)$	$\boldsymbol{cnot}(i,j)$	adds z_j to z_i
add x_i to z_j $(i \neq j)$	$\boldsymbol{h}(j)\boldsymbol{cnot}(i,j)\boldsymbol{h}(j)$	adds x_j to z_i
add x_i to z_i	$\boldsymbol{h}(i)\boldsymbol{p}(i)\boldsymbol{h}(i)$	none
add z_i to x_j $(i \neq j)$	$\boldsymbol{h}(j)\boldsymbol{cnot}(j,i)\boldsymbol{h}(j)$	adds z_i to x_j
add z_i to x_i	$\boldsymbol{p}(i)$	none
swap z_i and x_i	$\boldsymbol{h}(i)$	none
swap x_i and x_j	$\boldsymbol{cnot}(i,j)\boldsymbol{cnot}(j,i)\boldsymbol{cnot}(i,j)$	swaps z_i and z_j

Table 3.1: Basic row operations we use

2. After step 1 we have a matrix with a 1 in the upper left corner. Now add row 0 (x_0) to row $2i$ (x_i) respectively $2i+1$ (z_i) whenever $\boldsymbol{s}_{2i,0} = 1$ respectively $\boldsymbol{s}_{2i+1,0} = 1$. As row 1 ($z_0$) will be altered by this process, it will be dealt with after all the other rows have been processed. Now we have a matrix that looks like

$$\begin{pmatrix} 1 & ? & \dots \\ 0 & ? & \dots \\ \vdots & \vdots & \ddots \end{pmatrix}.$$

3. Repeat with the second row. We have $s_{1,1} = 1$ because of symplecticity[6]. Now add row 1 to row $2i$ respectively $2i+1$ whenever $s_{2i,1} = 1$ respectively $s_{2i+1,1} = 1$. $s_{0,0}$ remains unchanged because all $s_{i,0} = 0$ for $i \neq 0$. Finally add row 1 to row 0 if $s_{0,1} = 1$. Now our matrix has the form

$$\begin{pmatrix} 1 & 0 & ? & \dots \\ 0 & 1 & ? & \dots \\ 0 & 0 & ? & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

Because the matrix is symplectic not only the first two columns are those of

[6] Column 0 of $\boldsymbol{s}$ corresponds to the image of X_0 under the Clifford circuit. It is $\bar{X}_0 = X \otimes \mathbb{1} \cdots$. It has to anticommute with the image of Z_0. Therefore $\bar{Z}_0 = Z/Y \otimes ? \cdots$ and $\boldsymbol{s}_{1,1} = 1$

the identity, but also the first two rows.[7] Thus we have

$$\begin{pmatrix} 1 & 0 & 0 & \dots \\ 0 & 1 & 0 & \dots \\ 0 & 0 & ? & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix},$$

and therefore the direct sum of the identity and a $2(n-1)\times 2(n-1)$ symplectic matrix.

4. Now repeat steps 1 to three with the smaller matrix until the whole matrix is the identity.

All the operations we used in the process correspond to gates. All these gates together act as S^{-1}. If we invert the gatestring and the phase gates, the gates act as S. We do not have to invert the other gates, because CNOT and Hadamard are their own inverses.

The whole process uses $\mathcal{O}(n^2)$ row operations that are each made up by a constant number ($\leqslant 3$) of gates. Thus we need at most of the order of $\mathcal{O}(n^2)$ elementary gates for the circuit. We did not consider the symplectic properties of the circuit matrix in the complexity analysis. The restrictions on the rows and columns decrease the number of needed gates, but we did not investigate if this changes the order of the number of used gates. However, there exists an algorithm which decomposes a general (reversible) Clifford operation into $\mathcal{O}(n^2/\log(n))$ elementary gates [60].

3.7 Clifford channels with memory

We will now utilize the formalism of Clifford channels to study Clifford channels with memory. We use the definition of memory channels introduced in Section 3.5 and adjust it to Clifford operations.

Definition 3.7.1. *A Clifford channel with memory* $S : \mathfrak{B}\otimes\mathfrak{M} \to \mathfrak{M}\otimes\mathfrak{A}$ *is a Clifford channel*

$$S[\mathrm{w}_2(\eta)] = \lambda(\eta)\mathrm{w}_1(\mathbf{s}\eta)\ \forall \eta \in \Xi_2,$$

in which $\Xi_1 = \Xi_{\mathscr{M}} \oplus \Xi_{\mathscr{A}}$ *and* $\Xi_2 = \Xi_{\mathscr{B}} \oplus \Xi_{\mathscr{M}}$.

[7] At this point the images of X_0 and Z_0 are already fixed. The images of all other Pauli matrices have to commute with both of them. To commute with $\bar{X}_0 = X\otimes\mathbb{1}\cdots$ their first tensor factor has to be either X or $\mathbb{1}$. To commute with $\bar{Z}_0 = Z\otimes\mathbb{1}\cdots$ it has to be either Z or $\mathbb{1}$. So the only possibility is $\mathbb{1}$ and the first two rows of the transformed $\boldsymbol{s}$ are those of the identity.

The Weyl operators are composed of Weyl operators on the memory system and the input respectively output system via

$$\begin{aligned} \mathrm{w}_1(\xi) &= \mathrm{w}_{\mathscr{M}}(\xi_{\mathscr{M}}) \otimes \mathrm{w}_{\mathscr{A}}(\xi_{\mathscr{A}}) \; \forall \xi_{\mathscr{M}} \in \Xi_{\mathscr{M}}, \xi_{\mathscr{A}} \in \Xi_{\mathscr{A}}, \\ \mathrm{w}_2(\eta) &= \mathrm{w}_{\mathscr{B}}(\eta_{\mathscr{B}}) \otimes \mathrm{w}_{\mathscr{M}}(\eta_{\mathscr{M}}) \; \forall \eta_{\mathscr{B}} \in \Xi_{\mathscr{B}}, \eta_{\mathscr{M}} \in \Xi_{\mathscr{M}}. \end{aligned}$$

By extending the phase space vectors to the whole input respectively output space using

$$\begin{aligned} \tilde{\xi}_{\mathscr{M}} &= \xi_{\mathscr{M}} \oplus 0_{\mathscr{A}}, \\ \tilde{\xi}_{\mathscr{A}} &= 0_{\mathscr{M}} \oplus \xi_{\mathscr{A}}, \\ \tilde{\eta}_{\mathscr{B}} &= \eta_{\mathscr{B}} \oplus 0_{\mathscr{M}}, \\ \tilde{\eta}_{\mathscr{M}} &= 0_{\mathscr{B}} \oplus \eta_{\mathscr{M}} \end{aligned}$$

we obtain

$$\begin{aligned} \mathrm{w}_1(\xi) &= \mathrm{w}_1(\tilde{\xi}_{\mathscr{M}} + \tilde{\xi}_{\mathscr{A}}) = \mathrm{w}_1(\tilde{\xi}_{\mathscr{M}})\mathrm{w}_1(\tilde{\xi}_{\mathscr{A}}) \\ &= \mathrm{w}_{\mathscr{M}}(\xi_{\mathscr{M}}) \otimes \mathbb{1}_{\mathscr{A}} \cdot \mathbb{1}_{\mathscr{M}} \otimes \mathrm{w}_{\mathscr{A}}(\xi_{\mathscr{A}}), \qquad (3.59) \\ \mathrm{w}_2(\eta) &= \mathrm{w}_2(\tilde{\eta}_{\mathscr{B}} + \tilde{\eta}_{\mathscr{M}}) = \mathrm{w}_2(\tilde{\eta}_{\mathscr{B}})\mathrm{w}_2(\tilde{\eta}_{\mathscr{M}}) \\ &= \mathrm{w}_{\mathscr{B}}(\eta_{\mathscr{B}}) \otimes \mathbb{1}_{\mathscr{M}} \cdot \mathbb{1}_{\mathscr{B}} \otimes \mathrm{w}_{\mathscr{M}}(\eta_{\mathscr{M}}). \qquad (3.60) \end{aligned}$$

We observe that the Weyl operators can be decomposed into tensor products of Weyl operators which are localized solely on the memory respectively the input our output system. In our notation phase space vectors with a subscript letter for a system denote the restriction of the phase space vector to this system. An additional tilde denotes that we extended the phase space vectors from one subsystem to the whole system by adding zeros on the other subsystems. Then we have

$$S[\mathrm{w}_{\mathscr{B}}(\eta_{\mathscr{B}}) \otimes \mathrm{w}_{\mathscr{M}}(\eta_{\mathscr{M}})] = \lambda(\eta_{\mathscr{B}} \oplus \eta_{\mathscr{M}})\mathrm{w}_{\mathscr{M}}((\mathbf{s}\eta)_{\mathscr{M}}) \otimes \mathrm{w}_{\mathscr{A}}((\mathbf{s}\eta)_{\mathscr{A}}). \qquad (3.61)$$

Lemma 3.7.2. *The concatenation of two Clifford memory channels S_1 and S_2 with*

$$S_i : \mathfrak{B}_i \otimes \mathfrak{M} \to \mathfrak{M} \otimes \mathfrak{A}_i$$

and

$$\begin{aligned} S_1[\mathrm{w}_{\mathscr{B}_1}(\eta_{\mathscr{B}_1}) \otimes \mathrm{w}_{\mathscr{M}}(\eta_{\mathscr{M}})] &= \lambda_1(\eta_{\mathscr{B}_1} \oplus \eta_{\mathscr{M}})\mathrm{w}_{\mathscr{M}}((\mathbf{s}_1\eta)_{\mathscr{M}}) \otimes \mathrm{w}_{\mathscr{A}_1}((\mathbf{s}_1\eta)_{\mathscr{A}_1}) \\ S_2[\mathrm{w}_{\mathscr{B}_2}(\eta_{\mathscr{B}_2}) \otimes \mathrm{w}_{\mathscr{M}}(\eta_{\mathscr{M}})] &= \lambda_2(\eta_{\mathscr{B}_2} \oplus \eta_{\mathscr{M}})\mathrm{w}_{\mathscr{M}}((\mathbf{s}_2\eta)_{\mathscr{M}}) \otimes \mathrm{w}_{\mathscr{A}_2}((\mathbf{s}_2\eta)_{\mathscr{A}_2}) \end{aligned}$$

is again a Clifford channel with memory $S = S_1 \circ S_2$ and

$$\begin{aligned} S[\mathrm{w}_3(\eta)] = {} & \lambda_2(\eta_{\mathscr{B}_2} \oplus \eta_{\mathscr{M}})\lambda_1(\eta_{\mathscr{B}_1} \oplus (\tilde{\mathbf{s}}_2\eta)_{\mathscr{M}}) \cdot \\ & \mathrm{w}_1((\tilde{\mathbf{s}}_1\tilde{\mathbf{s}}_2\eta)_{\mathscr{M}} \oplus (\tilde{\mathbf{s}}_1\tilde{\mathbf{s}}_2\eta)_{\mathscr{A}_1} \oplus (\tilde{\mathbf{s}}_2\eta)_{\mathscr{A}_2}) \end{aligned} \qquad (3.62)$$

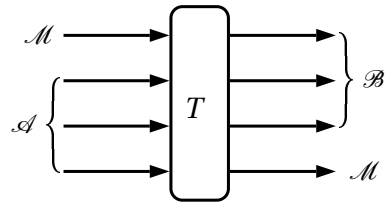


Figure 3.15: Scheme of a Clifford channel with three qudits input, three qudits output and one qudit memory

holds. Where, to simplify notation, we denote by $\tilde{\mathbf{s}}_1$ *and* $\tilde{\mathbf{s}}_2$ *the extension of* $\mathbf{s}_1$ *and* $\mathbf{s}_2$ *by the identity to the whole phase space (e.g.* $\tilde{\mathbf{s}}_2 : \Xi_{\mathscr{B}_2} \oplus \Xi_{\mathscr{M}} \oplus \Xi_{\mathscr{A}_1} \to \Xi_{\mathscr{M}} \oplus \Xi_{\mathscr{A}_2} \oplus \Xi_{\mathscr{A}_1}$*).*

Proof. Let $S : \mathfrak{B}_1 \otimes \mathfrak{B}_2 \otimes \mathfrak{M} \to \mathfrak{M} \otimes \mathfrak{A}_1 \otimes \mathfrak{A}_2$ be the concatenation of the channels. Then we have

$$\begin{aligned}
& S[\mathrm{w}_3(\eta)] \\
= \;& S[\mathrm{w}_{\mathscr{B}_1}(\eta_{\mathscr{B}_1}) \otimes \mathrm{w}_{\mathscr{B}_2}(\eta_{\mathscr{B}_2}) \otimes \mathrm{w}_{\mathscr{M}}(\eta_{\mathscr{M}})] \\
= \;& (S_1 \otimes \mathrm{id}_{\mathscr{A}_2})[\mathrm{w}_{\mathscr{B}_1}(\eta_{\mathscr{B}_1}) \otimes S_2[\mathrm{w}_{\mathscr{B}_2}(\eta_{\mathscr{B}_2}) \otimes \mathrm{w}_{\mathscr{M}}(\eta_{\mathscr{M}})]] \\
= \;& (S_1 \otimes \mathrm{id}_{\mathscr{A}_2})[\mathrm{w}_{\mathscr{B}_1}(\eta_{\mathscr{B}_1}) \otimes \lambda_2(\eta_{\mathscr{B}_2} \oplus \eta_{\mathscr{M}}) \mathrm{w}_{\mathscr{M}}((\tilde{\mathbf{s}}_2\eta)_{\mathscr{M}}) \otimes \mathrm{w}_{\mathscr{A}_2}((\tilde{\mathbf{s}}_2\eta)_{\mathscr{A}_2})] \\
= \;& \lambda_2(\eta_{\mathscr{B}_2} \oplus \eta_{\mathscr{M}}) S_1[\mathrm{w}_{\mathscr{B}_1}(\eta_{\mathscr{B}_1}) \otimes \mathrm{w}_{\mathscr{M}}((\tilde{\mathbf{s}}_2\eta)_{\mathscr{M}})] \otimes \mathrm{w}_{\mathscr{A}_2}((\tilde{\mathbf{s}}_2\eta)_{\mathscr{A}_2}) \\
= \;& \lambda_2(\eta_{\mathscr{B}_2} \oplus \eta_{\mathscr{M}}) \lambda_1(\eta_{\mathscr{B}_1} \oplus \tilde{\mathbf{s}}_2\eta_{\mathscr{M}}) \\
& \cdot \mathrm{w}_{\mathscr{M}}((\tilde{\mathbf{s}}_1\tilde{\mathbf{s}}_2\eta)_{\mathscr{M}}) \otimes \mathrm{w}_{\mathscr{A}_1}((\tilde{\mathbf{s}}_1\tilde{\mathbf{s}}_2\eta)_{\mathscr{A}_1}) \otimes \mathrm{w}_{\mathscr{A}_2}((\tilde{\mathbf{s}}_2\eta)_{\mathscr{A}_2}) \\
= \;& \lambda_2(\eta_{\mathscr{B}_2} \oplus \eta_{\mathscr{M}}) \lambda_1(\eta_{\mathscr{B}_1} \oplus \tilde{\mathbf{s}}_2\eta_{\mathscr{M}}) \mathrm{w}_1((\tilde{\mathbf{s}}_1\tilde{\mathbf{s}}_2\eta)_{\mathscr{M}} \oplus (\tilde{\mathbf{s}}_1\tilde{\mathbf{s}}_2\eta)_{\mathscr{A}_1} \oplus (\tilde{\mathbf{s}}_2\eta)_{\mathscr{A}_2})
\end{aligned}$$

and thus the proposition. □

Remark 3.7.3. *In this way we can create arbitrary concatenations of channels:*

$$\begin{aligned}
S[\mathrm{w}_{n+1}(\eta)] \quad = \quad & \lambda_n(\eta_{\mathscr{B}_n} \oplus \eta_{\mathscr{M}}) \textstyle\prod_{i=n-1}^{1} \lambda_i(\eta_{\mathscr{B}_i} \oplus (\tilde{\mathbf{s}}_{i+1} \cdots \tilde{\mathbf{s}}_n\eta)_{\mathscr{M}}) \\
& \cdot \mathrm{w}_1(\tilde{\mathbf{s}}_1 \cdots \tilde{\mathbf{s}}_n\eta)_{\mathscr{M}} \oplus (\textstyle\bigoplus_{i=1}^{n} (\tilde{\mathbf{s}}_{n-i+1} \cdots \tilde{\mathbf{s}}_n\eta)_{\mathscr{A}_i})).
\end{aligned} \tag{3.63}$$

The only restriction is that all the channels use the same memory system.

As in the study of general memory channels we are particularly interested in the forgetfulness of Clifford channels with memory. Using the definition of forgetfulness 3.5.2 introduced in Section 3.5 we will develop a criterion for the forgetfulness of Clifford memory channels.

Theorem 3.7.4. *A Clifford memory channel* $S : \mathfrak{B} \otimes \mathfrak{M} \to \mathfrak{M} \otimes \mathfrak{A}$*, defined by* $\mathbf{s}$ *and* λ *via* $S[\mathrm{w}_2(\xi)] = \lambda(\xi)\mathrm{w}_1(\mathbf{s}\xi)$ *is forgetful if and only if for all* $\xi \in \Xi_2$ *at least one of the two following conditions is fulfilled:*

1. *The restriction* $\hat{\mathbf{s}}_{\mathscr{M}}$ *of* $\mathbf{s}$ *to the memory fulfills*

$$\lim_{n\to\infty}(\hat{\mathbf{s}}_{\mathscr{M}}(\xi_{\mathscr{M}}))^n = 0. \tag{3.64}$$

2.

$$\lim_{n\to\infty}\prod_{i=1}^{n}\hat{\lambda}((\hat{\mathbf{s}}_{\mathscr{M}})^{i-1}\xi_{\mathscr{M}}) = 0, \tag{3.65}$$

where $\hat{\lambda}(\eta_{\mathscr{M}}) = \lambda(0_{\mathscr{B}}\oplus\eta_{\mathscr{M}})$.

Proof. We will prove forgetfulness for Weyl operators, which implies general forgetfulness as they form a basis of the observable algebra.

By the definition of forgetfulness a memory channel is forgetful if and only if there exists series of channels (or linear maps) $\tilde{S}_n : \mathfrak{M}\to\mathfrak{A}^{\otimes n}$ such that

$$\lim_{n\to\infty}\left\|\hat{S}_n - \mathbb{1}_{\mathfrak{M}}\otimes\tilde{S}_n\right\|_{\infty} = 0 \tag{3.66}$$

holds. In the above expression we used the property that $\mathfrak{M}$ is finite to substitute the cb-Norm with the ∞-norm in Equation (3.24). The channel $\hat{S}_n$ is constructed from S_n by choosing the observables on all output systems as the identity. In the phase space picture this means $\eta_{\mathscr{B}_i} = 0\ \forall i$.

We will now develop a simplified form of $\mathbf{s}_i$ for this case. In general we have $\tilde{\mathbf{s}}_i$

$$\tilde{\mathbf{s}}_i : \bigoplus_{j=1}^{i-1}\Xi_{\mathscr{B}_j}\oplus\Xi_{\mathscr{B}_i}\oplus\Xi_{\mathscr{M}}\oplus\left(\bigoplus_{j=i+1}^{n}\Xi_{\mathscr{A}_j}\right)\to\bigoplus_{j=1}^{i-1}\Xi_{\mathscr{B}_j}\oplus\Xi_{\mathscr{M}}\oplus\Xi_{\mathscr{A}_i}\oplus\left(\bigoplus_{j=i+1}^{n}\Xi_{\mathscr{A}_j}\right)$$

and

$$\begin{aligned}&\tilde{\mathbf{s}}_i\left(\bigoplus_{j=1}^{i-1}\xi_{\mathscr{B}_j}\oplus\xi_{\mathscr{B}_i}\oplus\xi_{\mathscr{M}}\oplus\left(\bigoplus_{j=i+1}^{n}\xi_{\mathscr{A}_j}\right)\right)\\ =\ &\bigoplus_{j=1}^{i-1}\xi_{\mathscr{B}_j}\oplus\mathbf{s}_i(\xi_{\mathscr{B}_i}\oplus\xi_{\mathscr{M}})\oplus\left(\bigoplus_{j=i+1}^{n}\xi_{\mathscr{A}_j}\right),\end{aligned}$$

where

$$\begin{aligned}\mathbf{s}_i(\xi_{\mathscr{B}_i}\oplus\xi_{\mathscr{M}}) &= (\mathbf{s}_i(\xi_{\mathscr{B}_i}\oplus\xi_{\mathscr{M}}))_{\mathscr{M}}\oplus(\mathbf{s}_i(\xi_{\mathscr{B}_i}\oplus\xi_{\mathscr{M}}))_{\mathscr{A}_i}\\ &= \mathbf{s}_{i,\mathscr{M}}(\xi_{\mathscr{B}_i}\oplus\xi_{\mathscr{M}})\oplus\mathbf{s}_{i,\mathscr{A}}(\xi_{\mathscr{B}_i}\oplus\xi_{\mathscr{M}}).\end{aligned} \tag{3.67}$$

Since all output systems $\mathscr{B}_i$ are the same and all input systems $\mathscr{A}_i$ are also equal $\mathbf{s}_i$

is independent of i. From the concatenation of $\tilde{\mathbf{s}}_i$ and $\tilde{\mathbf{s}}_{i+1}$ we obtain

$$\begin{aligned}
&\tilde{\mathbf{s}}_i\left(\tilde{\mathbf{s}}_{i+1}\left(\bigoplus_{j=1}^{i}\xi_{\mathscr{B}_j}\oplus\xi_{\mathscr{B}_{i+1}}\oplus\xi_{\mathscr{M}}\oplus\left(\bigoplus_{j=i+2}^{n}\xi_{\mathscr{A}_j}\right)\right)\right)\\
=\;&\tilde{\mathbf{s}}_i\left(\bigoplus_{j=1}^{i-1}\xi_{\mathscr{B}_j}\oplus\xi_{\mathscr{B}_i}\right.\\
&\left.\oplus\mathbf{s}_{\mathscr{M}}(\xi_{\mathscr{B}_{i+1}}\oplus\xi_{\mathscr{M}})\oplus\mathbf{s}_{\mathscr{A}}(\xi_{\mathscr{B}_{i+1}}\oplus\xi_{\mathscr{M}})\oplus\left(\bigoplus_{j=i+2}^{n}\xi_{\mathscr{A}_j}\right)\right)\\
=\;&\bigoplus_{j=1}^{i-1}\xi_{\mathscr{B}_j}\oplus\mathbf{s}_{\mathscr{M}}(\xi_{\mathscr{B}_i}\oplus\mathbf{s}_{\mathscr{M}}(\xi_{\mathscr{B}_{i+1}}\oplus\xi_{\mathscr{M}}))\oplus\mathbf{s}_{\mathscr{A}}(\xi_{\mathscr{B}_i}\oplus\mathbf{s}_{\mathscr{M}}(\xi_{\mathscr{B}_{i+1}}\oplus\xi_{\mathscr{M}}))\\
&\oplus\mathbf{s}_{\mathscr{A}}(\xi_{\mathscr{B}_{i+1}}\oplus\xi_{\mathscr{M}})\oplus\left(\bigoplus_{j=i+2}^{n}\xi_{\mathscr{A}_j}\right).
\end{aligned}$$

Along with

$$\hat{\mathbf{s}}(\xi_{\mathscr{M}})=\hat{\mathbf{s}}_{\mathscr{M}}(\xi_{\mathscr{M}})\oplus\hat{\mathbf{s}}_{\mathscr{A}}(\xi_{\mathscr{M}})=\mathbf{s}_{\mathscr{M}}(0_{\mathscr{B}}\oplus\xi_{\mathscr{M}})\oplus\mathbf{s}_{\mathscr{A}}(0_{\mathscr{B}}\oplus\xi_{\mathscr{M}}) \tag{3.68}$$

we can rewrite the concatenation in the case $\xi_{\mathscr{B}_i}=0_{\mathscr{B}_i}\;\forall i$ as follows:

$$\begin{aligned}
&\tilde{\mathbf{s}}_i\left(\tilde{h}_{i+1}^{\sigma}\left(\bigoplus_{j=1}^{i+1}0_{\mathscr{B}_j}\oplus\xi_{\mathscr{M}}\oplus\left(\bigoplus_{j=i+2}^{n}\xi_{\mathscr{A}_j}\right)\right)\right)\\
=\;&\bigoplus_{j=1}^{i-1}0_{\mathscr{B}_j}\oplus\hat{\mathbf{s}}_{\mathscr{M}}(\hat{\mathbf{s}}_{\mathscr{M}}(\xi_{\mathscr{M}}))\oplus\hat{\mathbf{s}}_{\mathscr{A}}(\hat{\mathbf{s}}_{\mathscr{M}}(\xi_{\mathscr{M}}))\oplus\hat{\mathbf{s}}_{\mathscr{A}}(\xi_{\mathscr{M}})\oplus\left(\bigoplus_{j=i+1}^{n}\xi_{\mathscr{A}_j}\right).
\end{aligned}$$

The concatenation of all $\tilde{\mathbf{s}}_i$ reads

$$\tilde{\mathbf{s}}_1\cdots\tilde{\mathbf{s}}_n\left(\bigoplus_{i=1}^{n}0_{\mathscr{B}_i}\oplus\xi_{\mathscr{M}}\right)=(\hat{\mathbf{s}}_{\mathscr{M}})^n\xi_{\mathscr{M}}\oplus\left(\bigoplus_{i=1}^{n}\hat{\mathbf{s}}_{\mathscr{A}}(\hat{\mathbf{s}}_{\mathscr{M}})^{n-i}\xi_{\mathscr{M}}\right). \tag{3.69}$$

By combining this with Equation 3.63 we obtain

$$\begin{aligned}
\hat{S}_n[\mathrm{w}_{\mathscr{M}}(\xi_{\mathscr{M}})] &= S_n\left[\textstyle\bigotimes_{i=1}^{n}0_{\mathscr{B}_i}\otimes\mathrm{w}_{\mathscr{M}}(\xi_{\mathscr{M}})\right]\\
&= \textstyle\prod_{i=1}^{n}\lambda(0_{\mathscr{B}_i}\oplus(\hat{\mathbf{s}}_{\mathscr{M}})^{i-1}\xi_{\mathscr{M}})\mathrm{w}_1((\hat{\mathbf{s}}_{\mathscr{M}})^n\xi_{\mathscr{M}}\\
&\quad\oplus(\textstyle\bigoplus_{i=1}^{n}\hat{\mathbf{s}}_{\mathscr{A}}(\hat{\mathbf{s}}_{\mathscr{M}})^{n-i}\xi_{\mathscr{M}}))\\
&= \textstyle\prod_{i=1}^{n}\hat{\lambda}((\hat{\mathbf{s}}_{\mathscr{M}})^{i-1}\xi_{\mathscr{M}})\mathrm{w}_{\mathscr{M}}((\hat{\mathbf{s}}_{\mathscr{M}})^n\xi_{\mathscr{M}})\\
&\quad\otimes(\textstyle\bigotimes_{i=1}^{n}\mathrm{w}_{\mathscr{A}_i}(\hat{\mathbf{s}}_{\mathscr{A}}(\hat{\mathbf{s}}_{\mathscr{M}})^{n-i}\xi_{\mathscr{M}})).
\end{aligned} \tag{3.70}$$

Thus on Weyl operators the channel $\hat{S}$ acts like the tensor product

$$\hat{S}_n[\mathrm{w}_{\mathscr{M}}(\xi_{\mathscr{M}})]=\hat{S}_n[\mathrm{w}_{\mathscr{M}}(\xi_{\mathscr{M}})]_{\mathscr{M}}\otimes\hat{S}_n[\mathrm{w}_{\mathscr{M}}(\xi_{\mathscr{M}})]_{\mathscr{A}} \tag{3.71}$$

where

$$\hat{S}_n[\mathrm{w}_{\mathscr{M}}(\xi_{\mathscr{M}})]_{\mathscr{A}} = \prod_{i=1}^{n} \hat{\lambda}((\hat{\mathbf{s}}_{\mathscr{M}})^{i-1}\xi_{\mathscr{M}}) \bigotimes_{i=1}^{n} \mathrm{w}_{\mathscr{A}_i}(\hat{\mathbf{s}}_{\mathscr{A}}(\hat{\mathbf{s}}_{\mathscr{M}})^{n-i}\xi_{\mathscr{M}}), \quad (3.72)$$

$$\hat{S}_n[\mathrm{w}_{\mathscr{M}}(\xi_{\mathscr{M}})]_{\mathscr{M}} = \mathrm{w}_{\mathscr{M}}((\hat{\mathbf{s}}_{\mathscr{M}})^{n}\xi_{\mathscr{M}}). \quad (3.73)$$

The next step is to investigate the forgetfulness of the channel. We will prove forgetfulness by first constructing a condition that is sufficient for forgetfulness. In the second step we will show that it is also necessary.

The condition (3.66) is satisfied if $\lim_{n\to\infty} \hat{S}_n = \lim_{n\to\infty} \mathbb{1}_{\mathfrak{M}} \otimes \tilde{S}_n$ holds. We can check this by evaluating both sides on a basis of the observable algebra. Naturally, we choose the Weyl operators. Thus

$$\lim_{n\to\infty} \hat{S}_n[\mathrm{w}_{\mathscr{M}}(\xi_{\mathscr{M}})] = \lim_{n\to\infty} \mathbb{1}_{\mathfrak{M}} \otimes \tilde{S}_n[\mathrm{w}_{\mathscr{M}}(\xi_{\mathscr{M}})] \quad \forall \xi_{\mathscr{M}} \in \Xi_{\mathscr{M}}$$

must hold. We use the splitting of $\hat{S}_n[\mathrm{w}_{\mathscr{M}}(\xi_{\mathscr{M}})]$ introduced in (3.71) to (3.73). To fulfill this for every $\xi_{\mathscr{M}} \in \Xi_{\mathscr{M}}$ one of the following has to hold:

1. $$\lim_{n\to\infty} \hat{S}_n[\mathrm{w}_{\mathscr{M}}(\xi_{\mathscr{M}})]_{\mathscr{M}} = \mathrm{w}_{\mathscr{M}}((\hat{\mathbf{s}}_{\mathscr{M}})^{n}\xi_{\mathscr{M}}) = \mathbb{1}_{\mathfrak{M}} \quad (3.74)$$

   and

   $$\lim_{n\to\infty} \tilde{S}_n[\mathrm{w}_{\mathscr{M}}(\xi_{\mathscr{M}})] = \lim_{n\to\infty} \hat{S}_n[\mathrm{w}_{\mathscr{M}}(\xi_{\mathscr{M}})]_{\mathscr{A}}. \quad (3.75)$$

2. $$\lim_{n\to\infty} \hat{S}_n[\mathrm{w}_{\mathscr{M}}(\xi_{\mathscr{M}})] = 0 \quad (3.76)$$

   and

   $$\lim_{n\to\infty} \mathbb{1}_{\mathfrak{M}} \otimes \tilde{S}_n[\mathrm{w}_{\mathscr{M}}(\xi_{\mathscr{M}})] = 0. \quad (3.77)$$

For (1.) we must choose $\tilde{S}_n[\mathrm{w}_{\mathscr{M}}(\xi_{\mathscr{M}})] = \hat{S}_n[\mathrm{w}_{\mathscr{M}}(\xi_{\mathscr{M}})]_{\mathscr{M}}$ and therefore (3.75) is fulfilled. Equation (3.74) translates into

$$\lim_{n\to\infty} (\hat{\mathbf{s}}_{\mathscr{M}})^{n}\xi_{\mathscr{M}} = 0 \quad (3.78)$$

on phase space. So $(\hat{\mathbf{s}}_{\mathscr{M}})^n$ must map any phase space vector $\xi_{\mathscr{M}}$ to zero in the limit $n \to \infty$.

In the case (2.) the actual series we choose for $\tilde{S}_n$ is irrelevant as long as it goes to zero for $n \to \infty$ to fulfill (3.77). From (3.76) we have

$$\lim_{n\to\infty} \mathrm{w}_{\mathscr{M}}((\hat{\mathbf{s}}_{\mathscr{M}})^{n}\xi_{\mathscr{M}}) \otimes \prod_{i=1}^{n} \hat{\lambda}((\hat{\mathbf{s}}_{\mathscr{M}})^{i-1}\xi_{\mathscr{M}}) \bigotimes_{i=1}^{n} \mathrm{w}_{\mathscr{A}_i}(\hat{\mathbf{s}}_{\mathscr{A}}(\hat{\mathbf{s}}_{\mathscr{M}})^{n-i}\xi_{\mathscr{M}}) = 0.$$

Weyl operators are always nonzero, so the only term that can actually vanish is the product of phases and

$$\prod_{i=1}^{n} \hat{\lambda}((\hat{\mathbf{s}}_{\mathscr{M}})^{i-1}\xi_{\mathscr{M}}) = 0. \tag{3.79}$$

If either (3.78) or (3.79) are fulfilled for every $\xi_{\mathscr{M}} \in \Xi_{\mathscr{M}}$ a channel is forgetful for Weyl operators. A general memory observable is of the form $A = \sum_i c_i \mathrm{w}_{\mathscr{M}}(\eta_i)$. By linearity we know that $\lim_{n\to\infty} \hat{S}_n[A_{\mathscr{M}}] - \mathbb{1}_{\mathfrak{M}} \otimes \tilde{S}_n[A_{\mathscr{M}}] \quad \forall A_{\mathscr{M}} \in \mathfrak{M}$ and therefore (3.66) holds. Now we will show that this is not only sufficient but also necessary.

We started from $\lim_{n\to\infty} \hat{S}_n = \lim_{n\to\infty} \mathbb{1}_{\mathfrak{M}} \otimes \tilde{S}_n$ to find a sufficient condition for forgetfulness. Let us now assume that there is a $\xi_{\mathscr{M}}$ such that

$$\lim_{n\to\infty} \hat{S}_n[\xi_{\mathscr{M}}] - \mathbb{1}_{\mathfrak{M}} \otimes \tilde{S}_n[\xi_{\mathscr{M}}] \neq 0$$

for any possible series $\tilde{S}_n$. This implies that

$$\lim_{n\to\infty} \left\| \hat{S}_n[\xi_{\mathscr{M}}] - \mathbb{1}_{\mathfrak{M}} \otimes \tilde{S}_n[\xi_{\mathscr{M}}] \right\|_{\infty} > 0,$$

because $\|x\|_{\infty} = 0$ implies $x = 0$ and the channel would not be forgetful. □

Corollary 3.7.5. *Every forgetful reversible Clifford channel S with n qudits memory is strictly forgetful. The memory depth of the channel is $2n$ or smaller. S is strictly forgetful if and only if $\hat{\mathbf{s}}_{\mathscr{M}}$ is represented by a nilpotent matrix.*

Proof. As we are dealing with a reversible channel we have $|\lambda| = 1$. So a reversible Clifford channel can only be forgetful if $\lim_{n\to\infty} \mathrm{w}_{\mathscr{M}}((\hat{\mathbf{s}}_{\mathscr{M}})^n \xi_{\mathscr{M}}) = \mathbb{1}_{\mathfrak{M}}$ holds. $\hat{\mathbf{s}}_{\mathscr{M}}$ is represented by a $2n \times 2n$-matrix A over the field $\mathbb{Z}_d$. This matrix has $2n$ components that can take values from $\mathbb{Z}_d$. Thus there are d^{4n^2} different $2n \times 2n$-matrices over $\mathbb{Z}_d$. The series $A \mapsto (A^i)_{i\in\mathbb{N}}$ is a series of such matrices, therefore it can only take d^{4n^2} different values. Hence it is either periodic or identical to the zero matrix from some $k < d^{4n^2}$ onwards. In the latter case A is nilpotent. If $A \mapsto (A^i)_{i\in\mathbb{N}}$ is periodic, the channel is not forgetful. However, if $(A^i)_{i\in\mathbb{N}} = 0$ for $i \geqslant k$ the channel is strictly forgetful. Reversible Clifford channels that are forgetful but not strictly forgetful are thus not realizable with finite memory.

Now we determine the memory depth. Every nilpotent matrix can be transformed into an upper triangular matrix with zero diagonal. The product of two such matrices is again an upper triangular matrix. Now also the first upper secondary diagonal is zero. Multiplication with another matrix of the first type lets the next secondary diagonal vanish. Thus the powers of a nilpotent $2n \times 2n$-matrix are zero at the latest starting from the $2n$-th power. This proves the second proposition. □

In general the memory system can be made up by qudits of different dimensions. Then the memory to memory transformation has the form of a direct sum of transformations acting on subsystems of a certain dimension. Since they do not interact the memory depth is governed by the depths of the individual transformations on the subsystems. The upper bound is $2m$, where m is the largest number of qudits in the subsystem that contains the most qudits.

Remark 3.7.6. *Two extremal cases occur if either the Condition (3.64) or Condition (3.65) is fulfilled for all $\xi_{\mathcal{M}}$. As shown in Corollary 3.7.5, tn the first case the matrix which represents $\hat{\mathbf{s}}_{\mathcal{M}}$ is nilpotent. In the second case we have*

$$\lim_{n\to\infty}\prod_{i=1}^{n}\hat{\lambda}((\hat{\mathbf{s}}_{\mathcal{M}})^{i-1}\xi_{\mathcal{M}})=0\;\forall\xi_{\mathcal{M}}. \tag{3.80}$$

The matrix representation $\hat{h}^{\sigma}_{\mathcal{M}}$ has been shown in Corollary 3.7.5 to be either periodic or nilpotent. Therefore we only have to multiply finitely many $\hat{\lambda}$ to check for forgetfulness. So Equation (3.80) becomes

$$|\prod_{i=1}^{d^{4n^2}}\hat{\lambda}((\hat{\mathbf{s}}_{\mathcal{M}})^{i-1}\xi_{\mathcal{M}})|<1\;\forall\xi_{\mathcal{M}}, \tag{3.81}$$

where n is the number of memory subsystems and d their internal dimension. This result also holds for single $\xi_{\mathcal{M}}$ and can thus be used for channels that do not fulfill Condition 3.65 for all $\xi_{\mathcal{M}}$.

Mixed systems

In Section 3.6.1 we showed that for Clifford channels with mixed input and/or output systems the interaction between the different-dimensional subsystems can be mediated only by the factor λ. Reversible channels factor completely so there is no interaction at all. For memory channels that means that in the reversible case there is no interaction between memory and input-output if the memory system consists of qudits of dimensions that do not occur in the input and output systems. In this case the memory can be removed from the description of the channel and the result is a memoryless channel.

3.8 Minimal resource decomposition of Clifford circuits and channels

Given a Clifford unitary S our goal is to determine the resources needed to implement it and, if possible, reduce the resource usage by permutation of input and

output qubits. The resource requirements are defined as shown in Figure 3.16; resource requirements of an operation are governed by the size of its support as introduced in Section 3.4. We split the operation up into smaller operations which have the structure of a memory system. The decomposition is thus a special case of the decomposition of causal channels introduced in Section 4.1. We define the resource requirements of this system as the maximum of the resource requirements of the individual operations. In the following will present an algorithm to determine a minimal resource decomposition of an arbitrary Clifford unitary.

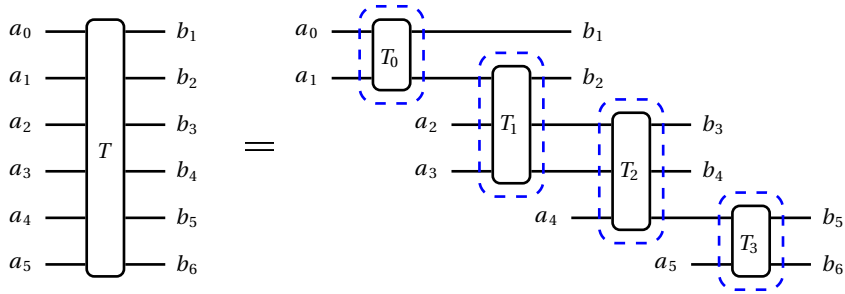


Figure 3.16: Decomposition of a quantum operation into a concatenation of several operations. The resource requirements of the decomposition is defined as the maximum of the support of the individual operations.

The symplectic matrix representing a causal circuit has the form shown in Figure 3.16: on the diagonal we have overlapping square blocks of size $2m_i$, where m_i is the memory usage of the ith operation in the circuit. The lower left part of the matrix is zero, the upper right part can be nonzero. In the following we will show that this condition is also sufficient and present an algorithm to determine the decomposition.

The whole transformation is represented by a $2n \times 2n$ symplectic matrix $\mathbf{s}$ over $\mathbb{F}_2$. The matrices of the individual operations $\mathbf{s}_i$ are of size $2m_i \times 2m_i$. To concatenate the operations we embed all of them into $2n \times 2n$ symplectic matrices by the direct sum $\tilde{\mathbf{s}}_i = \mathbb{1}^{\oplus d_i} \oplus \mathbf{s}_i \oplus \mathbb{1}^{n-m_i-d_i}$

Theorem 3.8.1. *A symplectic $2n \times 2n$ matrix S admits a decomposition into a convolutional product of smaller matrices if and only if it is of the form shown in Figure 3.17.*

Furthermore the spatial resources needed for the circuit is

$$\operatorname{res}(S) = \max(m_i) \tag{3.82}$$

where the m_i can be computed via $m_i = n - h_i - \sum_{j=1}^{j<i} d_j$.

Proof.

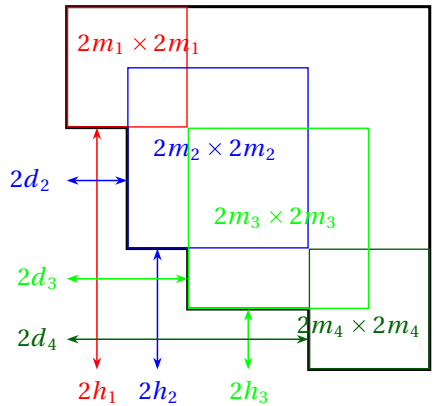


Figure 3.17: Matrix of the Clifford operation that shows what kind of decomposition is possible. The lower left part is zero, the rest can be nonzero

1. Assume the operation S has a decomposition into a causal chain of operations S_i. To study the form of the matrix $\mathbf{s}_i$ of S_i we study the matrices $\tilde{\mathbf{s}}_i$. The concatenation of the S_i corresponds to multiplication of the matrices $\tilde{\mathbf{s}}_i$.

   To prove that the product of the $\mathbf{s}_i$ is of the form depicted in Figure 3.17 we consider the multiplication of two matrices A and B that are both block diagonal matrices with only one non-identity block. This is exactly the form the $\mathbf{s}_i$ and their products have. These matrices can be decomposed into the sum of a matrix that is non-zero only on the non-identity block and an identity matrix: $A_{ij} = \widetilde{A}_{ij} + \delta_{ij}$ where $\widetilde{A}_{ij} \neq 0$ only if $i < 2m_1$ and $j < 2m_1$; $B_{ij} = \widetilde{B}_{ij} + \delta_{ij}$ where $\widetilde{B}_{ij} \neq 0$ only if $2d_2 \leqslant i < 2(m_2 + d_2)$ and $2d_2 \leqslant j < 2(m_2 + d_2)$. Multiplying A and B we get

$$\begin{aligned}(AB)_{ij} &= \sum_k \left(\widetilde{A}_{ik} + \delta_{ik}\right)\left(\widetilde{B}_{kj} + \delta_{kj}\right) \\ &= \sum_k \left(\widetilde{A}_{ik}\widetilde{B}_{kj} + \widetilde{A}_{ik}\delta_{kj} + \delta_{ik}\widetilde{B}_{kj} + \delta_{ik}\delta_{kj}\right) \\ &= \sum_k \left(\widetilde{A}_{ik}\widetilde{B}_{kj}\right) + \widetilde{A}_{ij} + \widetilde{B}_{ij} + \delta_{ij}.\end{aligned}$$

   The matrix $\widehat{AB}_{ij} = \sum_k \left(\widetilde{A}_{ik}\widetilde{B}_{kj}\right)$ is nonzero only if $i < 2m$ and $2d_2 \leqslant j < 2(d_2 + m_2)$. The first $2d_2$ columns of the matrix AB are those of A. This means that a patch of zeros that might exist in A in the first $2d_2$ columns is unchanged by the multiplication with B. Thus the product of the $\mathbf{s}_i$ will result in a matrix of the form in Figure 3.17.

2. Assume the matrix has this structure. Then use the following algorithm:

   - First we use CNOT and local gates (from the left) and the algorithm from Section 3.6.3 to transform the first $2d_2$ columns into the first $2d_2$ columns of the identity. We only need the first $2m_1$ rows to do that, so the resulting operation has support on the first m_1 qubits. As shown in Section 3.6.3 the matrix is symplectic and stays symplectic during the process the first $2d_2$ rows are also the first $2d_2$ rows of the identity.
   - We now have a matrix that is the direct sum of a $2d_2 \times 2d_2$ Identity and a $2(n-d_2) \times 2(n-d_2)$ symplectic matrix. We repeat step one for this symplectic submatrix and so forth until the resulting matrix is a $2m_{max} \times 2m_{max}$ symplectic matrix.

   Like in Section 3.6.3 we obtain a gate string which we invert to get the decomposition of our Clifford operation. But this time the gatestrings of each decomposition step have a limited support. In the ith step of the decomposition all the gates together form a Clifford operation S_i with a support on m_i qubits. More precisely it has its support on qubits d_i (we count from 0) to $d_i + m_i - 1$. Thus the resulting circuit is of the causal form shown in Figure 3.17.

□

In some cases it is possible to further reduce the resource requirements by a reordering of input and output qubits.

Remark 3.8.2. *At first sight it might seem that this decomposition gives us a way to determine the minimal memory needed to encode a convolutional code. However this is not the case, because in the first step of the encoding we have to add the initial memory inputs. Thus to decompose the circuit to encode a convolutional code we already need the circuit including the memory qubits.*

3.8.1 Reordering of input and output to reduce resource requirements

Sometimes the resource usage of a circuit can be reduced by a reordering of input and output qubits. If for example the last input qubit of a circuit with n qubits affects the first output qubit the resource requirement will be n qubits. But if there is another output qubit that is not affected by the last input qubit its position can be exchanged with the first output and the memory requirement reduced. Our algorithm does not provably find the optimal solution in all cases yet, but it does reduce the resource requirements.

The matrix $\mathbf{s}$ describes the circuit S in the Heisenberg picture, it maps phase space descriptions of output observables to phase space descriptions of input observables. A reordering of output qubits therefore corresponds to a reordering of rows while a reordering of inputs corresponds to reordering of columns. Only if the whole block $\mathbf{s}_{x,y}$, $x \in \{2i, 2i+1\}$, $y \in \{2j, 2j+1\}$ is 0 there is no interaction between output qubit i and input qubit j. The blocks themselves can not be changed, only their positions can. To make this simpler, we change the matrix description of the circuit. The $2n \times 2n$-matrix will be replaced by a $n \times n$-matrix whose elements are integers from 0 to 15 to enumerate the 2^4 possible 2×2 blocks in the matrix $\mathbf{s}$. First we swap the columns of the matrix in such a way that the numbers of zeros decreases to the right and get the matrix $\hat{\mathbf{s}}$. Now we create a list to keep track of the zeros in the columns. The list contains all integers z_i such that there is a column that has exactly z_i zeros. They are decreasing from large z_i to small z_i. The second entry n_i of each list element is the number of columns that have z_i or more zeros. This list gives us a lower bound on the resources needed to implement the circuit: $m_i \geqslant n - z_i - n_{i+1}$ and therefore $m \geqslant \max\{n - z_i - n_{i+1}\}$. However only if the zeros are arranged in the right way we will meet this lower bound. In the following we will expose an algorithm to find a low memory implementation of the circuit. It is not known if this implementation will be the optimal one with respect to resource requirements.

Starting from the matrix $\hat{\mathbf{s}}_0$ we swap rows to arrange the zeros in the first n_1 columns at the bottom of the matrix. Now we take the submatrix where the first n_1 rows and columns are omitted and swap the zero-rich columns to the left, getting a matrix $\hat{\mathbf{s}}_1$. We could again create a list and get a new bound on the resources, but instead we will just determine z_2 and n_2 which may have changed from the original z_2 and m_2 as there might be zeros in the parts of the original matrix we just cut off. (We start counting in the list from 2 now and in general for $\hat{\mathbf{s}}_i$ we start from $i+1$). With the matrix $\hat{\mathbf{s}}_1$ we continue in the same fashion as for $\hat{\mathbf{s}}_0$ and so forth until we arrive at $z_{\max} = 0$.

If we use this optimization on memory channels we have to leave the memory unchanged. The first m input qubits and the last m output qubits can not be swapped with other qubits, because this would change the information that is passed on to the next use of the channel and thus change the channel. In the algorithm this means that the first m columns and the last m rows of the matrix can not be swapped in the optimization process.

3.9 Error-correction

In this section we will present a brief introduction to quantum error-correction. After the introduction of some basic notions, we will introduce the basic notions of block based error-correction and block stabilizer codes. We will then present a

no-go theorem for Clifford only error-correction, proving that an error-correction scheme that only uses Clifford operations for encoding, error-correction and encoding inverse can not correct arbitrary single qubit errors. Finally we will introduce convolutional stabilizer codes and show shortcomings in the existing formalism.

When we speak about error-correction in this thesis, we think of a quantum communication process where information is transmitted through a noisy transmission channel. We can describe errors either by operators acting on the states that we transmit or by channels transforming density matrices or observables. To protect the information we actually want to send, we encode it into a larger system with special symmetries which define the space of states we use in our encoded transmission—the codespace. We then send the larger system and use the symmetries to try to detect and correct errors. Informally speaking, this works because the state that was effected by errors will probably not have the symmetries of our codespace. We can then find the nearest (in terms of occurrence of the smallest errors) state in the codespace and assume that this state has been sent. The theory of quantum error-correction is the search for good codespaces in terms of high rate (the ratio of actual information we sent over the size of the encoded information), error-correction capabilities (the errors that we can detect and correct) and the complexity of the encoding and decoding (error-correction) operations.

3.9.1 Block coding

Let us begin with a short introduction to block coding. The notation, definitions and presentation of the theorems in this section will follow [28]. In the setting we will consider in this section (for an illustration see Figure 3.18) we have an input system $\mathscr{A}$, a system $\mathscr{B}$ for the transmission over the erroneous channel and an output system that is of the same type as the input system. We use $\mathfrak{A} = \mathcal{B}(\mathcal{H}_{\mathscr{A}})$ and $\mathfrak{B} = \mathcal{B}(\mathcal{H}_{\mathscr{B}})$ to denote the operator algebras on each system. Input and transmission systems are sets of k and n qudits respectively. Often, errors are assumed to

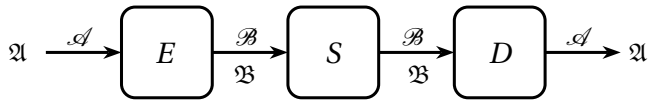


Figure 3.18: General setting for quantum error-correction; to protect the information on system $\mathscr{A}$ from errors during transmission it is encoded into a larger system $\mathscr{B}$ using the transformation E. $\mathscr{B}$ is transmitted via the noisy channel S. Possible errors are corrected using the decoding operation D which also inverts the encoding to reconstruct the input data.

be uncorrelated between different qudits. In this case the transmission channel decomposes into a tensor product of singe-qudit channels that are usually assumed to be the same for all qudits: $S = \tilde{S}^{\otimes n}$. See also Figure 3.19.

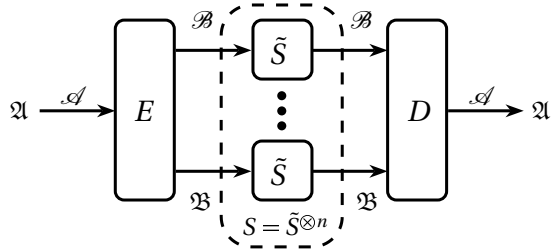


Figure 3.19: Error-correction in the case of uncorrelated errors; the transmission channel S decomposes into a tensor product of independent single-qudit channels $\tilde{S}$.

In Order to define which sets of errors are correctable we need to define what we mean by saying an error is localized on a given set of qubits.

Definition 3.9.1. *An operator A is localized on a subsystem $\mathscr{E}$ of $\mathscr{B}$ if $A = \tilde{A} \otimes \mathbb{1}_{\mathscr{B}\setminus\mathscr{E}}$ where $\tilde{A}$ is an element of $\mathfrak{A}(\mathscr{E})$. We call a channel S localized on a subsystem $\mathscr{E}$ if its Kraus decompositions only contain operators localized on $\mathscr{E}$. The set of operators localized on subsystems of at most e qudits is denoted by $\mathcal{E}_e$.*

Definition 3.9.2. *A (n,k,e) quantum error-correction code (qecc) is a pair (E,D) of channels with encoding channel $E\colon \mathfrak{B} \to \mathfrak{A}$ and decoding channel $D\colon \mathfrak{A} \to \mathfrak{B}$ such that $E \circ S \circ D = id$ for all $S\colon \mathfrak{B} \to \mathfrak{B}$ that contain only Kraus operators localized on (at most) e tensor factors. $\mathscr{A}$ is a system of k qudits and $\mathscr{B}$ a system of n qudits. $r = k/n$ is called the rate of the code. We will sometimes also use the notation (n,k,d), where d is the distance of the code. d and e are connected by $e = \lfloor (d-1)/2 \rfloor$.*

In the case of non-degenerate codes the Knill-Laflamme Theorem [62] gives a criterion for the error-correction capabilities of a given encoder E.

Theorem 3.9.3 (Knill-Laflamme)**.** *Let $\mathfrak{A}$ be the operator algebra of a system of k qudits and $\mathfrak{B}$ the operator algebra of a system of n qudits. Furthermore let $V : \mathcal{H}_{\mathscr{A}} \to \mathcal{H}_{\mathscr{B}}$ be an isometry that implements the encoding channel $E : \mathfrak{B} \to \mathfrak{A}$, $E(A) = V^*AV$. Then there exists a decoding channel $D : \mathfrak{A} \to \mathfrak{B}$ such that (E,D) is an (n,k,e) qecc if and only if*

$$E(F) \in \mathbb{C}\mathbb{1}_{\mathfrak{A}} \quad \forall F \in \mathcal{E}_{2e} \tag{3.83}$$

We show below that the condition (3.83) is equivalent to the often used condition

$$\left\langle \phi \right| V^* F_i^* F_j V \left| \psi \right\rangle = c(F_i, F_j) \left\langle \phi \right| \left. \psi \right\rangle \quad \forall F_{i,j} \in \mathcal{E}_e, \tag{3.84}$$

where $c(F_i, F_j)$ is independent of ψ and ϕ. To see that this condition is sufficient for error-correction, we consider an orthonormal basis ψ_i of $\mathcal{H}_{\mathscr{A}}$. The code space is spanned by the set $\{V\psi_i\}$ of encoded basis vectors. The error-correction condition (3.84) means that an error operator F has to preserve orthogonality in the code space to be perfectly detectable. For a set of errors to be correctable any product of two errors of the set has to be detectable or leave the code-space invariant. To correct a set of errors any combination of two different errors has to leave orthogonal encoded states orthogonal so they can be perfectly distinguished and the error can be corrected with probability one:

$$\left\langle F_k V\psi_i \right| \left. F_l V\psi_j \right\rangle = \delta_{ij} c(F_k, F_l) \quad \forall \psi_{i,j}, \forall F_{k,l} \in \mathcal{E}_e. \tag{3.85}$$

Condition (3.84) follows by linearity. On the other hand the orthonormal case (3.85) is a special case of (3.84) so (3.85) $\Leftrightarrow$ (3.84). Using this error-correction condition it is easy to see that the correctable errors form a linear space.

Lemma 3.9.4. *The space of correctable errors $\mathcal{E}_e$ is a linear subspace of $\mathfrak{B}$.*

Proof. Let $F_k \in \mathcal{E}_e$. Then $\sum_k a_k F_k \in \mathcal{E}_e$ because

$$\begin{aligned} \left\langle \sum_k a_k F_k V\psi_i \right| \left. \sum_l b_l F_l V\psi_j \right\rangle &= \sum_k a_k \sum_l b_l \left\langle F_k V\psi_i \right| \left. F_l V\psi_j \right\rangle \\ &= \underbrace{\sum_k a_k \sum_l b_l c(F_k, F_l)}_{\tilde{c}(\sum_k a_k F_k, \sum_l b_l F_l)} \delta_{ij}. \end{aligned}$$

□

Now, we want to show the equivalence of the conditions for error-correction derived above: (3.85) $\Leftrightarrow$ (3.83).

$\Rightarrow$

$$\begin{aligned} & \left\langle F_k V\psi_i \right| \left. F_l V\psi_j \right\rangle &=& \delta_{ij} c(F_k, F_l) & \forall \psi_{i,j}, \forall F_{k,l} \in \mathcal{E}_e \\ \Leftrightarrow\ & \left\langle \psi_i \right| V^* F_k^* F_l V \left| \psi_j \right\rangle &=& \delta_{ij} c(F_k, F_l) & \forall \psi_{i,j}, \forall F_{k,l} \in \mathcal{E}_e \\ \Leftrightarrow\ & E(F_k^* F_l)_{ij} &=& \delta_{ij} c(F_k, F_l) & \forall \psi_{i,j}, \forall F_{k,l} \in \mathcal{E}_e \\ \Leftrightarrow\ & E(F)_{ij} &=& \delta_{ij} c(F) & \forall i, j, \forall F \in \mathcal{E}_{2e} \\ \Leftrightarrow\ & E(F) &=& \mathbb{1} c(F) & \forall F \in \mathcal{E}_{2e} \end{aligned}$$

The transition from a product of two e-site errors to a single $2e$-site error works, because $\mathcal{E}$ is a linear set (Lemma 3.9.4) and products of e-site errors form a basis of $2e$-site errors.

⇐ Now we assume that (3.83) is satisfied. Then it holds for all $F \in \mathcal{E}_{2e}$ which fulfill $F = F_k^* F_l$ with $F_k, F_l \in \mathcal{E}_e$. Thus we have $E(F_k^* F_l) = \mathbb{1} c(F_k F_l)$ which we have just shown to be equivalent to $\langle F_k V\psi_i |\ F_l V\psi_j \rangle = \delta_{ij} c(F_k, F_l)$.

In the following, we will use the fact that we only have to check the conditions for a basis of the set of correctable errors and simplify the conditions for the case of stabilizer codes. Stabilizer codes are defined by the stabilizer generators, a set of commuting operators that fixes the code space in the sense that the code space consists of all states that are common eigenstates to all stabilizer generators. Stabilizer codes are therefore closely connected to the stabilizer states introduced in Section 2.3. Usually the stabilizer generators are Pauli products. In this case stabilizer codes can be encoded by Clifford channels (see e.g. [52]). Thus it is natural to chose Weyl operators as the basis of the set of errors and reformulate the error-correction conditions in the phase space picture.

In the phase space description, Condition (3.83) reads

$$E(\mathrm{w}(f)) = \lambda(f)\mathrm{w}(\mathbf{e}f) \stackrel{!}{=} c(f)\mathbb{1}_{\mathscr{A}}, \tag{3.86}$$

where $\mathrm{w}(f) \in \mathcal{E}_{2e}$ is a Weyl operator and $f \in \mathbb{F}_d^{2n}$ is an element of the phase space associated to the system $\mathscr{B}$. The set of phase space vectors of correctable errors is not a subspace of the phase space. The Weyl operator of the sum of two phase space vectors corresponds to the product of the individual vectors' operators, thus it is in general not correctable. To satisfy Condition 3.86 for all $f \in \Xi_{2e}^{\mathcal{E}}$, for each f one of the following has to hold:

1. $f \in \ker(e)$,
2. $\lambda(f) = 0$.

Now it is important to note that (a) $\mathbf{e}$ is a linear transformation and (b) $f \in \Xi_{2e}^{\mathcal{E}}$ form a basis of $\Sigma^{\mathscr{B}}$ whenever $e > 0$. This means that if some f fulfill the first condition, their whole linear span lies in the kernel of e. In the algebra $\mathfrak{B}$ this corresponds to the products of $F = \mathrm{w}(f)$. Because $\mathcal{E}_e$ is a linear space the whole subalgebra generated by $\mathrm{w}(f)$ would fulfill (3.86). However, correctable errors have to be mapped to the identity on the input system. Thus, the codespace can not have an overlap with the space of correctable errors, because this overlap would be projected onto the identity. In the extremal case that all f fulfill the first condition, the whole $\mathfrak{B}$ would be mapped to $\mathbb{1}$ by E and the codespace would be of dimension zero. Thus the second condition plays an important role.

To study $\mathbf{e}$ we extend E to a unitary transformation $\tilde{E}$ by adding a system $\mathscr{J}$ of $n-k$ ancilla qudits on the input side (see Section 3.6.2). The corresponding phase space transformation $\tilde{\mathbf{e}}$ is a $2n \times 2n$ symplectic transformation and Condition (3.86) transforms to

$$\mathrm{tr}_{\mathscr{J}} \tilde{E}(\mathrm{w}(f)) = c(f)\mathbb{1}_{\mathscr{A}}. \tag{3.87}$$

The partial trace is easy to carry out as Clifford channels map Weyl operators to multiples of Weyl operators that always have a tensor product form and

$$\begin{aligned}
\mathrm{tr}_{\mathscr{A}}\tilde{E}(\mathrm{w}(f)) &= \mathrm{tr}\,(\tilde{E}(\mathrm{w}(f))_{\mathscr{A}})\tilde{E}(\mathrm{w}(f))_{\mathscr{A}} \\
&= \mathrm{tr}\,(\tilde{E}(\mathrm{w}(f))_{\mathscr{A}})\tilde{\lambda}(f)\mathrm{w}(\tilde{\mathbf{e}}f)_{\mathscr{A}} \\
&= \underbrace{\mathrm{tr}\,(\tilde{E}(\mathrm{w}(f))_{\mathscr{A}})\tilde{\lambda}(f)}_{\lambda(f)}\mathrm{w}(\mathbf{e}f) \\
&\overset{!}{=} c(f)\mathbb{1}_{\mathscr{A}}.
\end{aligned}$$

We obtain the following conditions:

1. $\mathrm{w}(\mathbf{e}f) = \mathrm{w}(\tilde{\mathbf{e}}f)_{\mathscr{A}} = \mathbb{1}_{\mathscr{A}}$,

2. $\lambda(f) = \tilde{\lambda}(f)\mathrm{tr}\,(\tilde{E}(\mathrm{w}(f))_{\mathscr{A}}) = 0$.

For each f one of these conditions has to hold. $\tilde{\lambda}(f) \neq 0$, because $\tilde{E}$ is a unitary transformation. Therefore the second condition is equivalent to $\mathrm{tr}\,(\tilde{E}(\mathrm{w}(f))_{\mathscr{A}}) = 0$

We can now see that for general errors the first condition can never be true without the second condition holding at the same time. The only Weyl operator that is not tracefree is $\mathbb{1}$. So, for $\mathrm{tr}\,(\tilde{E}(\mathrm{w}(f))_{\mathscr{A}}) \neq 0$ we need $\tilde{E}(\mathrm{w}(f))_{\mathscr{A}} = \mathbb{1}_{\mathscr{A}}$. But for 1. to hold we also need $E(\mathrm{w}(f)) = \mathbb{1}_{\mathscr{A}}$. Therefore $\tilde{E}(\mathrm{w}(f)) = \mathbb{1}$ and thus $\tilde{\mathbf{e}}f = 0$. As $\tilde{E}$ is reversible this implies $\mathrm{w}(f) = \mathbb{1}_{\mathscr{B}}$ and thus $f = 0$. The first condition can only hold in the case of $F = \mathbb{1}$ where $E(F) = \mathbb{1}$ is given by unitality anyway. In the following we will therefore only consider the second condition.

3.9.2 No-go theorem for Clifford only error-correction

Stabilizer codes use Clifford channels for the encoding operation E. It would be convenient to also use Clifford channels for the decoding operation D. Undoing the transformation E is of course no problem with a Clifford channel. However, it turns out that the error-correction process can not be implemented as a Clifford channel if the code is capable of correcting arbitrary single qubit errors (or to say it more precisely, errors that form an algebraic basis of $\mathfrak{B}$).

Following Definition 3.9.2 we describe a Clifford only error-correction code as a pair of Clifford channels (E, D) such that $E \circ S \circ D = \mathbb{1}$ for all channels S whose Kraus operators belong to the set of correctable errors. All channels can be expressed in terms of Weyl operators. While this is straightforward for E and D, S needs a bit of work: not all operators in the set of correctable errors are Weyl operators. So a channel $S : S[A] = \sum_i K_i^* A K_i$ has to be rewritten with the K_i decomposed into Weyl

operators:

$$\begin{aligned} S[A] &= \sum_i K_i^* A K_i \\ &= \sum_i \left(\sum_k c_{i,k} \mathrm{w}(\xi_{i,k}) \right)^* A \left(\sum_l c_{i,l} \mathrm{w}(\xi_{i,l}) \right) \\ &= \sum_i \sum_k \sum_l c_{i,k}^* c_{i,l} \mathrm{w}(\xi_k)^* A \mathrm{w}(\xi_l) \\ &= \sum_{k,l} \tilde{c}_{k,l} \mathrm{w}(\xi_k)^* A \mathrm{w}(\xi_l), \end{aligned}$$

where the $\mathrm{w}(\xi_k)$ form a basis of the error space. It is important to note that the set of correctable error operators forms a linear space, while the ξ_k do not form a linear subspace of the phase space. Including the linear span of the ξ_k would make the error space an algebra and require the code to also correct all products of correctable errors. This is in general and specifically for arbitrary t-qubit errors not the case. In total we have

$$E[\mathrm{w}_2(\xi_2)] = \lambda_e(\xi_2) \mathrm{w}_1(\mathbf{e}\xi_2), \tag{3.88}$$

$$D[\mathrm{w}_1(\xi_1)] = \lambda_d(\xi_1) \mathrm{w}_2(\mathbf{d}\xi_1), \tag{3.89}$$

$$S[\mathrm{w}_2(\xi_2)] = \sum_{k,l} c_{k,l} \mathrm{w}_2(\xi_{2,k})^* \mathrm{w}_2(\xi_2) \mathrm{w}_2(\xi_{2,l}). \tag{3.90}$$

The channel S can be arbitrarily chosen from the set of channels of the form (3.90) where all $\xi_{2,k}$ are taken from the set of correctable phase space vectors. Firstly the channel $S = \mathrm{id}$ introducing no errors has to be in the set of correctable channels:

$$(E \circ \mathrm{id} \circ D)[\mathrm{w}_1(\xi_1)] = \lambda_d(\xi_1) \lambda_e(\mathbf{d}\xi_1) \mathrm{w}_1(\mathbf{e}\mathbf{d}\xi_1) \stackrel{!}{=} \mathrm{w}_1(\xi_1),$$

and therefore $\mathbf{e}\mathbf{d} = \mathbb{1}$ and $\lambda_e(\mathbf{d}\xi_1) = 1/\lambda_d \xi_1$, $\forall \xi_1$.

We are interested in codes that are able to correct arbitrary t-qubit errors. Therefore, the set of correctable phase space vectors has to contain all phase space vectors that are localized on up to t sites. We will now pick an easy example of such a channel and show that the combination of Clifford encoder and Clifford decoder (including error-correction) can not correct arbitrary single qubit errors. Consider the channel $S\colon S[A] = \mathrm{w}_2(\eta_2)^* A \mathrm{w}_2(\eta_2)$ with η_2 in the set of correctable phase space vectors. On Weyl operators S acts as a multiplication with a phase:

$$\begin{aligned} S[\mathrm{w}_2(\xi_2)] &= \mathrm{w}_2(\eta_2)^* \mathrm{w}_2(\xi_2) \mathrm{w}_2(\eta_2) \\ &= \mathrm{e}^{i\sigma(\eta_2, \xi_2)} \mathrm{w}_2(\xi_2). \end{aligned}$$

The whole encoding and decoding process is

$$\begin{aligned}(E\circ S\circ D)[\mathrm{w}_1(\xi_1)] &= E[\lambda_d(\xi_1)\mathrm{e}^{i\sigma(\eta_2,\mathbf{d}\xi_1)}\mathrm{w}_2(\mathbf{d}\xi_1)]\\ &= \lambda_d(\xi_1)\lambda_e(\mathbf{d}\xi_1)\mathrm{e}^{i\sigma(\eta_2,\mathbf{d}\xi_1)}\mathrm{w}_1(\mathbf{ed}\xi_1)\\ &= \mathrm{e}^{i\sigma(\eta_2,\mathbf{d}\xi_1)}\mathrm{w}_1(\xi_1).\end{aligned}$$

To correct the errors $\mathrm{e}^{i\sigma(\eta_2,\mathbf{d}\xi_1)}=1$ for all ξ_1. If we want (E,D) to be able to correct arbitrary single qubit errors η_2 can be any phase space vector that has support only on a single cell. But σ is the symplectic form of a standard Weyl system and thus non-degenerate. So $\sigma_2(\eta_2,\mathbf{d}\xi_1)=0$ for all single cell phase space vectors η_2 implying $\mathbf{d}\xi_1=0$ which is a contradiction to ξ_1 being arbitrary and $\mathbf{d}$ reversible. Actually the single cell phase space vectors η_2 form a basis of the whole phase space Ξ_2. If there was a Clifford-only code that corrected all single cell errors it would correct all errors. This would correspond to the case $\mathbf{d}\xi_1=0$, $\forall\xi_1$. A code with this property would encode zero qubits and is totally useless.

3.9.3 Block stabilizer codes

An especially well studied class of block codes for quantum error-correction are the stabilizer codes. In a stabilizer code the codespace is defined as the common eigenspace to the eigenvalue one of a group of commuting operators, the stabilizer. To encode k logical qubits into n physical qubits we need a codespace of dimension 2^k. On $\mathbb{C}^{2^k}$ there are k commuting pairs $\bar{X}$, $\bar{Z}$ of Pauli X, Z operators. These operators map elements of the code space to elements of the code space. Let E be the encoding channel and ρ the density matrix of a state on the logical qubits. Then $E\rho E^*$ is an element of the codespace. Now $\mathrm{tr}\,(\bar{X}E\rho E^*)=\mathrm{tr}\,(E^*\bar{X}E\rho)$ and the $\bar{X},\bar{Z}$ can then be seen as encoded images of the Pauli matrices on the logical input qubits. The elements of the stabilizer have to commute with all encoded Pauli matrices, because $\bar{X}S\psi=\bar{X}\psi=\phi=S\phi=S\bar{X}\psi$ for all ψ in the codespace and all stabilizer operators S.

A stabilizer code is usually defined by a set of $n-k$ Pauli products that generate the stabilizer group $\mathcal{S}$ by multiplication. For an n-qubit block code the Pauli products are of length n. In phase space they are described by an $n-k$ dimensional isotropic subspace of the n-qubit phase space $\Xi_n=\mathbb{Z}_2^{2n}$. For the encoding operation we assume that ancilla qubits in the state $|0\rangle$ are inserted into the stream of data qubits. In each block $n-k$ ancilla qubits are inserted. $|0\rangle$ is stabilized by Z, therefore we have unencoded stabilizers Z_i, $1\leqslant i\leqslant n-k$ which correspond to the stabilizer generators S_i acting on the encoded qubits. As $n-k$ qubits on the input side are ancilla qubits in a fixed state we have k of the n qubits left for data in each block. Again we consider operators acting on the input qubits. Using the Pauli matrices we can construct every operator on the data qubits. The Pauli matrices on qubits $n-k+1$ to n obviously commute with the stabilizers Z_i, $1\leqslant i\leqslant n-k$ on the

ancilla qubits. Thus we can define a reversible operation that maps the unencoded stabilizers to the encoded stabilizers and the Pauli matrices on the data qubits onto the encoded Pauli matrices $\bar{X}_i$ and $\bar{Z}_i$. Using Lemma 3.6.11 we can complete this partially defined transformation. Now the obtained transformation maps operators on the input system to operators on the output system. To determine the encoding channel in the Heisenberg picture we have to invert the obtained transformation. We can do that by inversion of the matrix of the phase space representation.

As an example let us consider the well-known five-qubit code, the shortest block code than can correct arbitrary single-qubit errors [63]. Its stabilizer is given by the generators

$$\begin{aligned} S_1 &= XZZXI, \\ S_2 &= IXZZX, \\ S_3 &= XIXZZ, \\ S_4 &= ZXIXZ. \end{aligned}$$

The encoded Pauli matrices are

$$\begin{aligned} \bar{X} &= XXXXX, \\ \bar{Z} &= ZZZZZ. \end{aligned}$$

Let us assume that the data qubit is the last qubit in each block. Then we have the following transformation:

$$\begin{aligned} Z_1 &\mapsto XZZXI \\ Z_2 &\mapsto IXZZX \\ Z_3 &\mapsto XIXZZ \\ Z_4 &\mapsto ZXIXZ \\ X_5 &\mapsto XXXXX \\ Z_5 &\mapsto ZZZZZ. \end{aligned}$$

This set of rules can be completed by four rules for X_i, $1 \leqslant i \leqslant 4$. The inverse of the resulting transformation is the encoding unitary.

We will now briefly describe how errors are corrected. To check if a transmitted block of qubits contains errors all the stabilizer operators are measured. If the state ψ we receive is a codeword, all measurements of stabilizer generators S_i will give us the result $s_i = 1$ by definition. Now imagine an error F occurred and instead of the codeword ψ we receive the state $F\psi$. We assume that F is a Pauli product. Let ξ_i be the phase space vector of $S_i = \mathrm{w}(\xi_i)$ and let ζ be the phase space vector of the error $F = \mathrm{w}(\zeta)$. Then we have

$$s_i = \mathrm{tr}\,(S_i F \rho_\psi F^*) = \mathrm{e}^{i\sigma(\xi_i,\zeta)}\,\mathrm{tr}\,(S_i \rho_\psi F^* F) = \mathrm{e}^{i\sigma(\xi_i,\zeta)}. \tag{3.91}$$

	$\mathbb{1}$	X_1	Z_1	Y_1	X_2	Z_2	Y_2	X_3	Z_3	Y_3	X_4	Z_4	Y_4	X_5	Z_5	Y_5
S_1	0	0	1	1	1	0	1	1	0	1	0	1	1	0	0	0
S_2	0	0	0	0	0	1	1	1	0	1	1	0	1	0	1	1
S_3	0	0	1	1	0	0	0	0	1	1	1	0	1	1	0	1
S_4	0	1	0	1	0	1	1	0	0	0	0	1	1	1	0	1

Table 3.2: Error syndromes for the 5-qubit code

Thus the measurement result tells us if the error commutes or anticommutes with a given stabilizer. If the error is not a Pauli product, the measurements will project the error onto a Pauli product and we obtain the measurement results according to the projected error. We can use these measurement results to tell which error occurred. Hence correctable error has to have a unique signature in the measurement, called the error syndrome. For the 5 qubit code we obtain the error-syndromes displayed in Table 3.2.

If two errors $F_1 = \mathrm{w}(\zeta_1)$ and $F_2 = \mathrm{w}(\zeta_2)$ have the same error syndrome we have $\sigma(\zeta_1, S_i) = \sigma(\zeta_2, S_i)$ and therefore $\sigma(\zeta_1 + \zeta_2, S_i) = 0$ for all stabilizer generators S_i. Thus $F_1 F_2$ commutes with all stabilizer operators; it is a member of the normalizer[8] $\mathcal{N}(\mathcal{S})$ of $\mathcal{S}$ in $\mathbb{Z}_2^{2n}$. Now there are two possibilities for $F_1 F_2$. It can be a stabilizer itself, then it is no problem that we can not tell F_1 and F_2 apart because F_1 and F_2 have the same action on the code space and we can correct them with the same operation. But if $F_1 F_2 \in \mathcal{N}(\mathcal{S}) - \mathcal{S}$ the errors need different correction strategies. As we can not tell them apart, they can not be corrected. To construct a Pauli product with l non-identity factors from two Pauli products, the sum of their non-identity elements has to be at least l. Thus, if the minimum of the support of the elements of $\mathcal{N}(\mathcal{S}) - \mathcal{S}$ is d, all errors with support $t \leqslant (d-1)/2$ can be corrected. $d = \min\{\sup(N) | N \in \mathcal{N}(\mathcal{S}) - \mathcal{S}\}$ is called the distance of the code. If the minimal support of elements of $\mathcal{S}$ is smaller than d the code is called degenerate, else it is called non-degenerate.

In the case of non-degenerate codes the distance can be determined using only the stabilizer. By definition $\min\{\sup(S) | S \in \mathcal{S}\} = d$. Another way to define the distance using the stabilizer is to say that the distance is the minimal number of factors in which any two elements of the stabilizer differ. These two notions are equivalent, because the product of two stabilizer operators which differ on l factors will be an operator which is non-identity on l factors. On the other hand, an operator with l non-identity factors differs from the identity, which is always contained in the stabilizer, on l factors.

Of course, in general, a code of distance d can correct more errors, than the arbitrary errors on up to $(d-1)/2$ qubits. To be precise it can correct all errors from

[8] The normalizer of a subset $\mathcal{S}$ of a group $\mathcal{G}$ is the set of all elements $g \in \mathcal{G}$ such that $sg = gs$ for all $s \in \mathcal{S}$.

the set F_i if for all i, j $F_iF_j \in \mathcal{S} \cup (\mathcal{G} - \mathcal{N}(\mathcal{S}))$. However we are usually interested in the error-correction capabilities on channels with unknown errors and we are therefore interested in arbitrary errors and do not consider the additional error-correction capabilities. The distance of the example 5-qubit code is 3, therefore it can correct arbitrary single cubit errors.

For a more detailed introduction to stabilizer codes see e.g. [64].

3.10 Convolutional quantum error-correcting codes

We now add memory to the encoding and decoding channels to describe convolutional codes. We have an encoding channel $E: \mathfrak{B} \otimes \mathfrak{M}_E \to \mathfrak{M}_E \otimes \mathfrak{A}$, a decoding channel $D: \mathfrak{A} \otimes \mathfrak{M}_D \to \mathfrak{M}_D \otimes \mathfrak{B}$ and an unchanged transmission channel $S: \mathfrak{B} \to \mathfrak{B}$. Now we define a quantum convolutional code.

Definition 3.10.1. *$A(n,k,e,m_E,\tau)$ quantum convolutional error-correcting code is a pair (E,D) with encoding channel $E: \mathfrak{B} \otimes \mathfrak{M}_E \to \mathfrak{M}_E \otimes \mathfrak{A}$ and decoding channel $D: \mathfrak{A} \otimes \mathfrak{M}_D \to \mathfrak{M}_D \otimes \mathfrak{B}$ such that $E_\tau \circ S^{\otimes \tau} \circ D_\tau = id$ for all S localized on at most $e \cdot \tau$ elements. $\mathscr{A}$, $\mathscr{B}$, $\mathscr{M}_E$ and $\mathscr{M}_D$ are systems of k, n, m_E and m_D qudits respectively. τ is the memory depth of the encoding channel.*

Unfortunately this straight forward definition has a major drawback. The transmission channel S is the same in every block and—more importantly—the transmission channels of different blocks are uncorrelated. Therefore we can only describe integer numbers of errors per block. As we will show later the strength of convolutional codes lies exactly in the cases where the number of errors per block is not an integer. In this situation the convolutional structure can adapt to the errors while codes with independent blocks have to be able to correct the next largest integer number of errors. We change our definition and include memory channels as transmission channels.

Definition 3.10.2. *$A(n,k,e,m_E,\tau)$ quantum convolutional error-correcting code is a pair (E,D) with encoding channel $E: \mathfrak{B} \otimes \mathfrak{M}_E \to \mathfrak{M}_E \otimes \mathfrak{A}$ and decoding channel $D: \mathfrak{A} \otimes \mathfrak{M}_D \to \mathfrak{M}_D \otimes \mathfrak{B}$ such that $E_{\tau+1} \circ S_{\tau+1} \circ D_{\tau+1} = id$ for all S with memory depth smaller than τ localized on at most $e \cdot n(\tau+1)$ elements, where τ is the memory depth of the encoding channel The systems $\mathscr{A}$, $\mathscr{B}$, $\mathscr{M}_E$ and $\mathscr{M}_D$ are comprised of k, n, m_E and m_D qudits respectively. The number of correctable errors per block e can be fractional.*

Sometimes we will also use the notation (n,k,d,m_E,τ) for convolutional stabilizer codes. Here the d is the distance of the code as introduced for block coding. Using the distance only makes sense in a restricted setting, where we can guarantee minimal distances between individual errors. We will go into details on this in Section 7.3.

Convolutional codes where first introduced in 1998 by H. F. Chau in [65]. In the following years convolutional stabilizer codes were introduced, studied, and improved by different researchers, e.g. in [66, 67, 68, 69]. Let us first discuss the advantages of convolutional codes.

3.10.1 Reasons for using convolutional codes

Convolutional codes are widely used in the processing of classical information, because they offer a better ratio of performance to complexity than block codes. The performance is measured by the rate and the error-correction properties. The complexity is measured by the complexity of the error-detection and correction algorithm. In the quantum case [70] claims that convolutional codes have the potential to outperform block codes in the following senses:

Rate In general quantum convolutional codes need fewer physical qubits to protect the same number of logical qubits against errors than comparable block codes.

Complexity of the error-correction algorithm The classical algorithms needed to determine which error occurred from the measured error syndrome scale better with larger blocks than for block codes.

Performance Quantum convolutional codes have a better relation of performance to complexity than block codes.

In the paper [70] the authors also present examples that confirm these claims. The same paper also presents the so called tail-biting codes, which are block codes obtained by the application of convolutional codes to blocks of qubits with "periodic boundary conditions". The properties of these block codes are similar to the properties of convolutional codes. In Chapter 7 we will investigate these claims. Our focus will lie on the resource requirements of the codes, respectively the achievable code rate given a spatial resource bound and a error rate that has to be correctable. We will not consider the decoding complexity.

3.10.2 Convolutional stabilizer codes

It is possible to extend our notion of stabilizer codes to codes that allow for an overlap between the individual blocks of stabilizer generators. Their encoding operations can be described as quantum memory channels in the following way: some of the output qubits of the nth encoding step will be used as inputs in the $n+1$th step of the encoding. These qubits are the memory system of the memory channel describing the encoding. The stabilizer is usually assumed to be translation invariant by shifts of multiples of n qubits, thus every step is described by the same channel.

In Chapter 7 we will show how convolutional stabilizer codes can be described in an efficient way by Clifford memory channels.

Definition and formalism

Here we present the most common formulation of convolutional codes. We roughly follow [71] while occasionally altering definitions and notations to suit our needs. Just like a block stabilizer code, a convolutional stabilizer coded is defined by the generators of its stabilizer group. The stabilizer generators of a convolutional code also exhibit some kind of block structure, however, with overlapping blocks. In block codes the stabilizer generators (and thus the stabilizer group) of every block are the same. Considering only one block we can deduce all the properties of the code. The convolutional codes we consider are translation invariant. Their stabilizer generators are arranged in blocks of $n-k$ generators which are translates by multiples of n qubits of a single set of generators. We come to the following definition:

Definition 3.10.3 (based on [71])**.** *A (n,k,e,m,τ) convolutional stabilizer code is given by the stabilizer group $\mathcal{S}$*

$$\mathcal{S} = \mathrm{sp}\left\{S_{j,i} = I^{\otimes j\cdot n} \otimes S_{0,i},\, 1 \leqslant i \leqslant n-k,\, 0 \leqslant j\right\}, \tag{3.92}$$

where $S_{0,i} \in \mathcal{G}_{n(\tau+1)}$ and all $S_{i,j}$ are independent of each other. All generators of $\mathcal{S}$ have to commute. The memory depth τ of the code is defined as

$$\tau = \min_{l\in\mathbb{N}} \left\{ l \geqslant \mathrm{length}(\mathrm{sup}(S_{j,i}))\right\} - 1,$$

where $\mathrm{sup}(S_{j,i})$ is the support of $S_{j,i}$. Furthermore, the integer m tells us that there exists a convolutional encoder using m qubits of memory.

Of course strictly speaking all the generators need to act on the same space. We could achieve this by padding all of the $S_{j,i}$ with identities I from the right until they have the same length. If we think of the stabilizers as part of the quasi-local algebra it is obvious that the padding can be omitted. For some set of operators $\{A_i\}$ we denote by $\mathrm{sp}\{A_i\}$ the span of A_i, i.e. the set of operators that can be generated by multiplication of A_i. $\mathrm{sp}\{A_i\}$ is the multiplicative group generated by A_i.

A code of this form encodes k logical qubits per n physical qubits. Every "block" itself has an output of $n(\tau+1)$ qubits. The overlap of the blocks of stabilizer generators can be illustrated using the matrix S of all generators depicted in Figure 3.20. Every row of the matrix represents one generator $S_{j,i}$, every column represents one output qubit. The width of the rectangles represents the maximal possible support of the stabilizer generators, i.e. the positions where the operators may differ from the identity. Every box contains the same set of operators. Due to the translation invariance with respect to shifts by multiples of n, the resulting stabilizer generators

htb

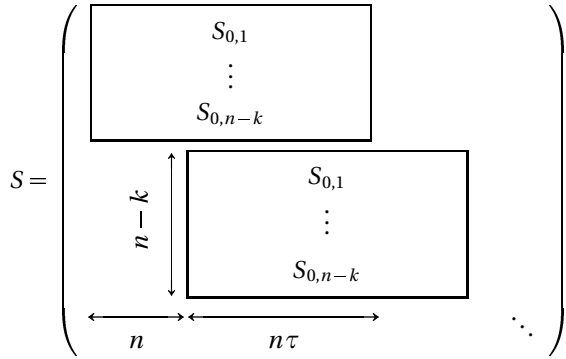


Figure 3.20: Structure of the stabilizer generators

are shifted copies of one set of operators $S_{0,i}$. Because the generators of different blocks overlap by $n\tau$ qubits we can not split the code space into blocks but have to consider the space stabilized by all stabilizer generators simultaneously. When $\tau = 0$ we obtain a block code. To use the code in a real world application the matrix S would have to be expanded by a bounded number of operators that do not fit into this pattern. These operators are needed for the initialization and the termination of the transmission. We neglect them here, because in the limit of long transmission they do not play a role for the rate and error-correction capabilities of the code.

We now introduce two example codes that we will use during our studies of convolutional codes. They were both introduced in the paper [70]. The first code uses a block length of $n = 3$ and encodes one qubit per block, therefore it is of rate $1/3$. The stabilizer generators have a length of two blocks and are defined by

$$\begin{array}{rcl} S_{j,1} & = & I^{\otimes(3j)} XXXXZY \\ S_{j,2} & = & I^{\otimes(3j)} ZZZZYX. \end{array} \tag{3.93}$$

The code can be found in Table IV of [70]. It can correct one error in a block given that the two adjacent blocks are free of errors. Therefore it is an $(3,1,3,m,1)$ code. We will later present a method to specify m given the stabilizer generators.

The second code is code 3 from Table IV of [70]. It also has a block length of $n = 3$ and is a rate $1/3$ code. It has distance $d = 5$ and can therefore correct any error pattern with at most two errors in the support of each stabilizer generator. The stabilizer generators are given by

$$\begin{array}{rcl} S_{j,1} & = & I^{\otimes(3j)} XXXIXZIXYXYZ \\ S_{j,2} & = & I^{\otimes(3j)} ZZZIZYIZXZXY. \end{array} \tag{3.94}$$

This code is a $(3,1,3,m,5)$ convolutional stabilizer code.

We now use the n-qubit-shift translation-invariance of the set of stabilizer generators to introduce a notation similar to the Laurent polynomial notation introduced in Section 2.2.6. Let us first define the shift operator

$$D[A] = \tau^n A. \tag{3.95}$$

We can now describe the generators by

$$S_{j,i} = D^j[S_{0,i}],\ 1 \leqslant i \leqslant n-k, 0 \leqslant j. \tag{3.96}$$

This enables us to only consider the first $n-k$ generators. All other generators are obtained by repeated application of D. The shift can be generalized using polynomials in D. We define the action of a polynomial $P(D) = \sum_j \alpha_j D^j$, $P(D) \in \mathcal{P}(D)$ on $A \in \mathcal{G}$ by

$$P(D)[A] = \prod_j \alpha_j D^j[A]. \tag{3.97}$$

This definition relies critically on the fact that all copies of A that are shifted by D^j commute, i.e. $[D^i[A], D^j[A]] = 0\ \forall i, j \in \mathbb{N}$. If this were not the case the product would not be well defined. If, during some calculation, we end up with a result that contains negative powers of D, we apply D^{-j}, where j is the lowest exponent occurring.

We will now shift to a phase space description of the elements of $\mathcal{G}$. To avoid matrices with tuples as coefficients we split the matrix S in two parts by replacing the vectors (ξ_X, ξ_Z) by $(\xi_X|\xi_Z)$.

Example 3.10.4.

$$\left(\begin{array}{ccccc} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{array}\right) \mapsto \left(\begin{array}{ccccc|ccccc} 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \end{array}\right) \tag{3.98}$$

To further simplify our notation we adapt the "algebraic Fourier transformation" used in 2.2.6 to translation invariance by n qubits. Using

$$\hat{\xi}_X(j) = \sum_i \xi_X(i \cdot n + j) D^i, 0 < j \leqslant n \tag{3.99}$$

we map the components of a phase space vector to tuples with n coefficients which are Laurent polynomials in D over $\mathbb{F}_2$ and thus elements of $\mathcal{P}$. We can also write $\hat{\xi} = \sum_i \hat{\xi}^{(i)} D^i$, where $\hat{\xi}^{(i)}$ are n-dimensional vectors with coefficients from $\mathbb{F}_2$. In the following we will work with the transformed phase space vectors exclusively and omit the "$\hat{}$".

For the example code 1 we obtain the following matrix:

$$\hat{S}_1 = \left(\begin{array}{ccc|ccc} 1+D & 1 & 1+D & 0 & D & D \\ 0 & D & D & 1+D & 1+D & 0 \end{array}\right). \tag{3.100}$$

Code 2 has the representation

$$\widehat{S}_2 = \left(\begin{array}{ccc|ccc} 1+D^3 & 1+D+D^2+D^3 & 1+D^2 & 0 & D^3 & D+D^2+D^3 \\ 0 & D^3 & D+D^2+D^3 & 1+D^3 & 1+D+D^2 & 1+D+D^3 \end{array} \right). \tag{3.101}$$

All the basic stabilizer generators and all their translates by multiples of n qubits have to commute pairwise. In the polynomial notation it is easy to check this. Two Pauli operators $A \widehat{=} (\xi_X, \xi_Z)$ and $B \widehat{=} (\eta_X, \eta_Z)$ commute, if and only if the symplectic form vanishes for their phase space vectors ($\xi_X \eta_Z + \eta_X \xi_Z = 0$). Therefore

$$AB = BA \Leftrightarrow \sum_i \xi_X^{(i)} \eta_Z^{(i)} - \xi_Z^{(i)} \eta_X^{(i)} = 0, \tag{3.102}$$

where $\xi_X^{(i)}$ is the ith block of ξ_X and $\xi_X^{(i)} \eta_Z^{(i)}$ is the scalar product. In the polynomial notation we can substitute the sum by the multiplication of the polynomials:

$$\xi_X(D)\eta_Z(1/D) - \xi_Z(D)\eta_X(1/D). \tag{3.103}$$

However to obtain Equation (3.102) we only need the summands which are products from coefficients of the same block i. Thus (3.102) is the coefficient of D^0 of Equation (3.103). Analyzing Equation (3.103) further we see that

$$\begin{aligned} (3.103) &= \sum_i \xi_X^{(i)} D^i \sum_j \eta_Z^{(j)} D^{-j} - \sum_i \xi_Z^{(i)} D^i \sum_j \eta_X^{(j)} D^{-j} \\ &= \sum_{i,j} \left(\xi_X^{(i)} \eta_Z^{(j)} - \xi_Z^{(i)} \eta_X^{(j)} \right) D^{i-j} \\ &= \sum_{l=-\tau+1}^{\tau-1} D^l \sum_{i=0}^{\tau-1} \xi_X^{(i)} \eta_Z^{(i-l)} - \xi_Z^{(i)} \eta_X^{(i-l)} \\ &= \sum_l D^l \sigma(\xi, D^l \eta) \end{aligned}$$

where $l = i - j$. Thus (3.103) actually computes the commutation relation of all shifted versions of ξ and η. Therefore we have the generalized commutator

$$\forall r, s,\ D^r[A] D^s[P] = D^s[P] D^r[A] \Leftrightarrow \xi_X(D)\eta_Z(1/D) - \xi_Z(D)\eta_X(1/D) = 0. \tag{3.104}$$

It can be used to directly check if an operator commutes with another operator and all its translates by multiple n-qubit shifts. This is often needed for convolutional codes, e.g. when checking if a set of commuting operators is a valid set of stabilizer generators or when determining encoded Pauli matrices for a given stabilizer.

Encoded Pauli operators

To determine the whole encoding operation we need to find the images of the Pauli matrices on the input data qubits. The operators have to be in $N(\mathcal{S})\backslash\mathcal{S}$ which means hat they commute with the stabilizer group while they are not contained in the stabilizer group themselves. Furthermore, they have to come in k pairs of anti-commuting operators which commute with all their translates (conditions (3.10) - (3.13) in [64]).

Together with the generalized commutator (3.104) the commutation with the stabilizer group suffices to determine the encoded Pauli operators. We now introduce an easy algorithm to directly compute possible encoded Pauli matrices that was first presented in [71]. However, this algorithm makes a special ansatz for the solution of the equation derived from the generalized commutator. This ansatz fails in many cases to find solutions for encoded Pauli operators of finite length. This will be demonstrated with our example codes.

The first step of the algorithm is to use Gaussian elimination to bring S in the standard form

$$S_{\text{std}} = \left(\begin{array}{ccc|ccc} \overbrace{A(D)}^{r} & \overbrace{B(D)}^{n-k-r} & \overbrace{C(D)}^{k} & \overbrace{E(D)}^{r} & \overbrace{F(D)}^{n-k-r} & \overbrace{G(D)}^{k} \\ 0 & 0 & 0 & J(D) & K(D) & L(D) \end{array}\right) \begin{matrix} \} r \\ \} n-k-r \end{matrix} . \tag{3.105}$$

The matrices $A(D)$ and $K(D)$ are diagonal and have full rank. The rank of $A(D)$ is also the rank of the X part of M and denoted by r. Using the ansatz,

$$\bar{X} = (U_1(D), U_2(D), U_3(D) | V_1(D), V_2(D), V_3(D)), \tag{3.106}$$

$$\bar{Z} = (U_1'(D), U_2'(D), U_3'(D) | V_1'(D), V_2'(D), V_3'(D)), \tag{3.107}$$

one can find the encoded operators. $V_2(D)$ can be taken to be 0, because $K(D)$ has full rank and the encoded Pauli operators can be multiplied by any stabilizer yielding another valid encoded Pauli operator. The same reasoning holds for $U_1(D)$ and $A(D)$. The encoded Pauli operators have to commute with all their translates which can be checked by Equation (3.104). Using the matrices U_i and V_i the commutator can be checked for all operators at once and

$$U_3(D)V_3(1/D) + U_3(1/D)V_3(D) = 0 \tag{3.108}$$

has to hold. In [71] the solution $U_3(D) = 0$ is used. As we will see in the following this restriction in the space of possible encoded X operators leads to problems

finding finite length encoded Z operators. For $\bar{X}$ we get the solution

$$\begin{aligned} U_1(D) &= 0, && (3.109)\\ U_2(D) &= L^T(1/D)K^{-1}(1/d)\Lambda(D), && (3.110)\\ U_3(D) &= \Lambda(D)\cdot I, && (3.111)\\ V_1(D) &= (U_2(D)F^T(1/D)+\Lambda(D)G^T(1/D))A^{-1}(1/D), && (3.112)\\ V_2(D) &= 0, && (3.113)\\ V_3(D) &= 0. && (3.114) \end{aligned}$$

The encoded Z operators $\bar{Z}$ fulfill

$$\begin{aligned} U_1'(D) &= 0, && (3.115)\\ U_2'(D) &= 0, && (3.116)\\ U_3'(D) &= 0, && (3.117)\\ V_1'(D) &= C^T(1/D)A^{-1}(1/D)/\Lambda(1/D), && (3.118)\\ V_2'(D) &= 0, && (3.119)\\ V_3'(D) &= I/\Lambda(1/D). && (3.120) \end{aligned}$$

The polynomial $\Lambda(D)$ can be chosen freely. However, the encoded Pauli matrices should be finite to ensure that decoding with a constant delay is possible (see Section 4.5). This corresponds to the condition that all the Laurent series involved are finite. Therefore, the choice of $\Lambda(D)$ is important. We call the suitable polynomial $\Lambda(D)$, the conditioning polynomial.

Definition 3.10.5 ([71])**.** *The conditioning polynomial $\Lambda(D)$ of a quantum convolutional code is the non-zero polynomial of minimal degree such that (3.109)-(3.114) only contain finite Laurent series.*

The conditioning polynomial always exists. Therefore, we can always find a set of encoded X operators that have finite support and respect the invariance by n qubit shifts. Because of (3.120) this only holds for the encoded Z operators, when Λ is a monomial. Thus, given encoded X operators we can not always find Z operators that respect the translations and have finite support. Let us illustrate this with our example code 1. The standard form of the stabilizer matrix is

$$S_{1,\mathrm{std}} = \left(\begin{array}{ccc|ccc} D+D^2 & 0 & D^2 & 1+D & 1+D+D^2 & D^2 \\ 0 & D^2 & D^2 & D+D^2 & D+D^2 & 0 \end{array}\right).$$

Because we have $r=2$ the matrices $B(D)$, $F(D)$, $J(D)$, $K(D)$ and $L(D)$ are all non-

existent. For the remaining matrices we have

$$A(D) = \begin{pmatrix} D+D^2 & 0 \\ 0 & D^2 \end{pmatrix}, \quad C(D) = \begin{pmatrix} D^2 \\ D^2 \end{pmatrix},$$
$$E(D) = \begin{pmatrix} 1+D & 1+D+D^2 \\ D+D^2 & D+D^2 \end{pmatrix}, \quad G(D) = \begin{pmatrix} D^2 \\ 0 \end{pmatrix}.$$

Using Equations (3.109)-(3.114) we obtain

$$U_1(D) = (0,0), \quad U_3(D) = \Lambda(D), \quad V_1(D) = (1,0), \quad V_3(D) = 0,$$

with $\Lambda(D) = D+1$ and $U_2(D)$ and $V_2(D)$ non-existent. The encoded X operator is $\bar{X} = Z\,I\,X\,I\,I\,X$. However, the conditioning polynomial Λ is not a monomial and therefore by Equation (3.120) $V_3'(D)$ will be an infinite Laurent series and the encoded Z operators will be of infinite length. Therefore, this method does not give us an encoder that allows decoding with a finite delay. In Chapter 7 we will show, that such a decoder exists and explicitly construct one.

For the second example we obtain

$$S_{2,\text{std}} = \left(\begin{array}{ccc|c} D^3+D^6 & 0 & D+D^4 & \\ 0 & D^6 & D^4+D^5+D^6 & \\ \hline 1+D+D^2+D^4+D^5+D^6 & 1+D^2+D^3+D^5+D^6 & 1+D+D^4 & \\ D^3+D^6 & D^3+D^4+D^5 & D^3+D^4+D^6 & \end{array}\right).$$

Again we have $r = 2$ and $B(D)$, $F(D)$, $J(D)$, $K(D)$ and $L(D)$ are all non-existent. Using Equations (3.109)-(3.114) we obtain

$$U_1(D) = (0,0), \quad U_3(D) = 1+D^3,$$
$$V_1(D) = (D^2+D^5+D^6, 1+D^2+D^5+D^6), \quad V_3(D) = 0,$$

with $\Lambda(D) = D+1$ and $U_2(D)$ and $V_2(D)$ non-existent. The encoded X operator is $\bar{X} = I\,Z\,X\,I\,I\,I\,Z\,Z\,I\,I\,I\,X\,I\,I\,I\,Z\,Z\,I\,Z\,Z\,I$. Again the encoded Z operator will have infinite length, because Λ is not a monomial. Thus the algorithm presented in [71] fails to find a non-catastrophic encoder for both example codes.

4 Causal operations and quantum channels with memory

Parts of the material in this chapter will be published in:
Johannes Gütschow, Tomáš Š. Rybár, and Reinhard F. Werner
Memory requirements for general reversible quantum stream processors

In this chapter we will consider the causal operations, causal inverses and their connection to memory channels. First we will study how to implement a globally defined causal operation in terms of a memory channel. This enables us to implement an operation that creates long range correlations on a chain of Quantum systems $\mathscr{A}_i$, $i \in \mathbb{Z}$ with a channel acting only on one of the systems $\mathscr{A}_i$ and an additional memory system $\mathscr{M}$ at a time.

For simplicity, and because in this case we can deduce more, we will only consider the case of operations which are quasi-local automorphisms. The decomposition we will introduce in Section 4.1 is similar to the structure theorem introduced in [51], which we stated in Section 3.5. As we will only consider reversible channels the structure theorem will be generalized and strengthened for this case in the sense that we will consider also non-translation-invariant operations and determine the size of the needed memory. The structure theorem of [51] itself can however be easily generalized to non-translation-invariant systems. A schematic of our decomposition is shown in Figure 4.1.

In this chapter we will prove that quasi-local reversible causal operations correspond to reversible forgetful memory channels, while finite depth reversible causal operations correspond to strictly forgetful reversible channels. The memory required to implement a finite depth causal operation is proven to equal the index of the operation (Section 4.2). Using this we will introduce a minimal memory implementation of a finite depth causal operation and present a method to obtain the channel transformation matrix in both the general and the Clifford case (Section 4.3).

Furthermore, we introduce the use of Bratteli diagrams to study the memory dynamics of causal processes and memory channels (Section 4.4). This formalism will be used to determine bounds on the depth of a finite depth causal operation

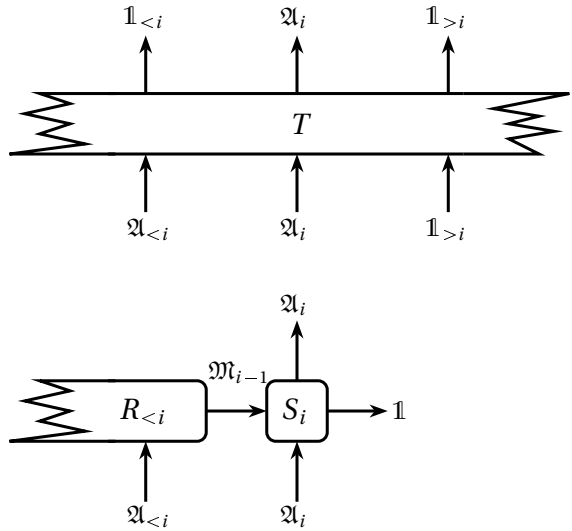


Figure 4.1: Decomposition of a causal operation into a series of memory channels; The example shows a decomposition on the level of single cell inputs. For larger inputs S_i will be replaced by a concatenation of memory channels.

given the needed memory. This will be used in Section 4.5 to derive bounds on the memory requirements of causal inverses.

When dealing with memory channels in quantum communication it is important to know if the effect of the channel can be reversed without waiting until the end of the transmission. If we for example use convolutional codes, we want to begin with the decoding after a finite number of qubits have been received, independent of the total length of the transmission. Of course every reversible memory channel can be inverted by just inverting the unitary transformation which implements the channel. However, the result will be an anti-causal channel as shown in Figure 4.2. To make this channel causal we have to delay the operation that recovers the first inputs until the end of the whole process. Thus the inputs will be recovered in reverse order and the recovery time depends on the length of the transmission. If we think of the channel as being the encoder of a convolutional error correcting code and the channel inverse as being the decoder it is easy to see, why this concept is practically unusable. On Not only can we start recovering the input not until after the end of the transformation, we also have to store the encoded qubits for a time proportional to the length of the transmission and need a quantum memory of the size of the transmitted message. As quantum memory is a costly resource,

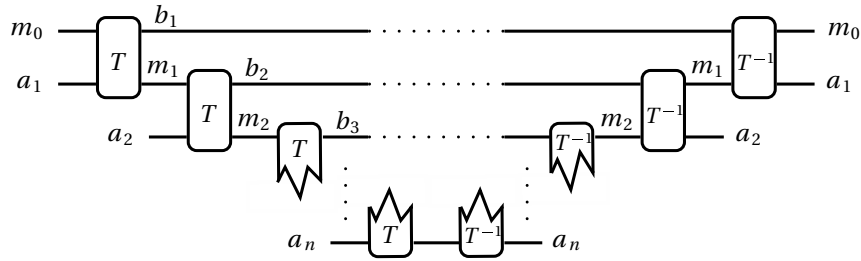


Figure 4.2: A memory channel inverse created by inverting the channel unitary; this construction creates an anti-causal operation. Making it causal introduces a delay in the recovery of the inputs that is proportional to the length of the transmission.

even more costly if it is supposed to store the information reliably for a long time, we need a scheme that reduces the memory to a minimal amount. Therefore we want to construct an inverse to a memory channel or causal operation that allows us to begin to recover the data that has been sent during the transmission with a constant delay between sending and recovering, independent of the length of the transmission. Such an operation will itself be causal. This approach is illustrated in Figure 4.3.

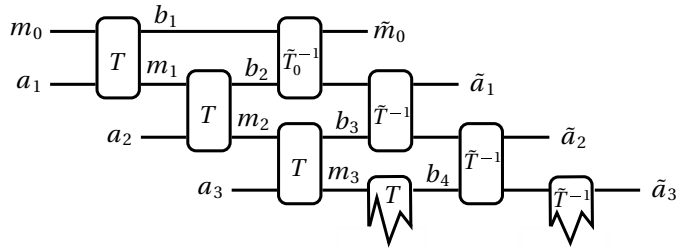


Figure 4.3: A causal inverse of a memory channel; the recovery of the sent information can begin after a delay independent of the transmission length. (Example for memory depth 2)

Therefore, it is of interest to know under which circumstances a causal operation T has a causal inverse, i.e. a causal operation T' which fulfills $T \circ T' = \tau^n$. A causal operation that has a causal inverse is called causally reversible. The shift τ^n introduces a delay between encoding and decoding of information. This delay should of course be minimal. In Section 4.5 we will study causally reversible operations and show how to construct causal inverses for a given causal operation or memory

channel. Furthermore, we will investigate the memory needed for a causal inverse and give upper and lower bounds on the needed memory given partial knowledge about the causal process.

4.1 Representing causal operations with memory channels

In this section, we will prove how a causal operation can be decomposed into a series of memory channels as introduced in Figure 4.1. We will work with an implementation of the channel S_i on the chain of quantum systems $\mathscr{A}_i$, $i \in \mathbb{Z}$ which uses the chain algebra $\mathfrak{A}_\mathbb{Z}$ also as the memory algebra. We denote this representation by a superscript c to make clear when we use it. It does not have the minimal memory properties we want to achieve using channels to implement causal operations but it is useful to prove the decomposition.

Furthermore, we will show that quasi-local causal operations correspond to forgetful memory channels, while finite depth causal operations correspond to strictly forgetful channels.

Let us start with the general case of the decomposition:

Theorem 4.1.1. *Let $T : \mathfrak{A}_\mathbb{Z} \to \mathfrak{A}_\mathbb{Z}$ be a quasi-local causal automorphism, $\mathfrak{A}_\mathbb{Z} = \bigotimes_{i=-\infty}^{\infty} \mathfrak{A}_i$ and $\mathfrak{A}_i = \mathcal{M}_{d_i}(\mathbb{C})$. It follows that T can be decomposed into a sequence of memory channels S_i in the sense that for any finitely localized observable $A \in \mathfrak{A}_{[j,i]} \subset \mathfrak{A}$*

$$T[A] = (R_{<j} \otimes id_{[j,i]} \circ S_j \otimes id_{[j+1,i]} \circ \cdots \circ id_{[j,i-1]} \otimes S_i)[A \otimes \mathbb{1}_{\mathfrak{M}_i}] \tag{4.1}$$

holds. In this Theorem we use an implementation of S_i on the chain algebra $\mathfrak{A}_\mathbb{Z}$ instead of additional memory algebra. To denote that we use this representation we call the channel S_i^c. The channels S_i^c are defined by

$$S_i^c : \mathfrak{A}_i \otimes \mathfrak{M}_i \to \mathfrak{M}_{i-1} \otimes \mathfrak{A}_i, \quad S_i^c[A_i \otimes M_i] = T[A_i]M_i \tag{4.2}$$

for all $A_i \in \mathfrak{A}_i$ and all $M_i \in \mathfrak{M}_i$, where

$$\mathfrak{M}_i = \mathbf{S}(T[\mathfrak{A}_{>i}], \mathfrak{A}_{\leqslant i}). \tag{4.3}$$

The initialization operation $R_{<i}^c$ is defined by

$$R_i^c : \mathfrak{M}_{i-1} \to \mathfrak{A}_{<i}, \quad R_i^c[M_{i-1}] = M_{i-1}. \tag{4.4}$$

Furthermore, all $\mathfrak{M}_i$ have trivial center.

Proof. Let us first define an additional set of memory algebras by commutation relations:

$$\mathfrak{N}_i = \{A \in \mathfrak{A}_{\leqslant i} | [A, T[\mathfrak{A}_{\leqslant i}]] = \{0\}\}. \tag{4.5}$$

By definition $\mathfrak{M}_i$ is included in $\mathfrak{N}_i$: we have $\mathfrak{M}_i = \mathbf{S}(T[\mathfrak{A}_{>i}], \mathfrak{A}\leqslant i)$, thus $\mathfrak{M}_i \subset \mathfrak{A}_{\leqslant i}$ and $[\mathfrak{M}_i, T[\mathfrak{A}_{\leqslant i}]] = \{0\}$, implying $\mathfrak{M}_i \subset \mathfrak{N}_i$. We define a channel

$$\tilde{S}_i^c : \mathfrak{A}_i \otimes \mathfrak{N}_i \to \mathfrak{N}_{i-1} \otimes \mathfrak{A}_i, \quad \tilde{S}_i^c[A_i \otimes N_i] = T[A_i]N_i \tag{4.6}$$

using the $\mathfrak{N}_i$ instead of $\mathfrak{M}_i$ in the same way as S_i^c.

Now we have all to start with the proof. Let us first show that the S_i^c are actually memory channels and fulfill the claims we made.

1. The multiplications $T[A_i]M_i$ and $T[A_i]N_i$ are well defined, because by definition of $\mathfrak{N}_i$, $[\mathfrak{N}_i, T[\mathfrak{A}_i]] = \{0\}$ and $\mathfrak{M}_i \subset \mathfrak{N}_i$.

2. S_i^c and $\tilde{S}_i^c$ map into $\mathfrak{A}_{\leqslant i}$: $T[\mathfrak{A}_i] \subset \mathfrak{A}_{\leqslant i}$ by causality of T. Furthermore, $M_i \in \mathfrak{M}_i \subset \mathfrak{A}_{\leqslant i}$ by definition and therefore $S_i^c : \mathfrak{A}_i \otimes \mathfrak{M}_i \to \mathfrak{A}_{\leqslant i}$. The same argument holds for $\tilde{S}_i^c$.

3. Now we can prove that S_i^c maps into $\mathfrak{M}_{i-1} \otimes \mathfrak{A}_i$: We have

$$\begin{aligned} T[\mathfrak{A}_i] &\subset \mathbf{S}(T[\mathfrak{A}_i], \mathfrak{A}_{<i}) \otimes \mathfrak{A}_i \\ &\subset \mathbf{S}(T[\mathfrak{A}_{\leqslant i}], \mathfrak{A}_{<i}) \otimes \mathfrak{A}_i \\ &= \mathfrak{M}_{i-1} \otimes \mathfrak{A}_i \end{aligned}$$

   and

$$\begin{aligned} \mathfrak{M}_i &\subset \mathbf{S}(T[\mathfrak{A}_{\leqslant i}], \mathfrak{A}_{<i}) \otimes \mathfrak{A}_i \\ &= \mathfrak{M}_{i-1} \otimes \mathfrak{A}_i, \end{aligned}$$

   thus $T[A_i]M_i \in \mathfrak{M}_{i-1} \otimes \mathfrak{A}_i$ for all $A_i \in \mathfrak{A}_i$ and all $M_i \in \mathfrak{M}_i$ and S_i maps into $\mathfrak{M}_{i-1} \otimes \mathfrak{A}_i$.

4. To prove that $\tilde{S}_i^c$ maps into $\mathfrak{N}_{i-1} \otimes \mathfrak{A}_i$ we first need to show that the commutator $\left[\tilde{S}_i^c[A_i \otimes N_i], T[\mathfrak{A}_{\leqslant i-1}]\right] = \{0\}$ for all $A_i \in \mathfrak{A}_i$ and all $N_i \in \mathfrak{N}_i$. As $\tilde{S}_i^c[A_i \otimes N_i] = T[A_i]N_i$ it suffices to show commutation for $T[A_i]$ and N_i individually. Now $[T[A_i], T[\mathfrak{A}_{\leqslant i-1}]] = \{0\}$ because T is a homomorphism. On the other hand $[N_i, T[\mathfrak{A}_{\leqslant i-1}]] \subset [N_i, T[\mathfrak{A}_{\leqslant i}]] = \{0\}$ by definition of $\mathfrak{N}_i$. Now we have all we need to prove that $\tilde{S}_i^c$ maps into $\mathfrak{N}_{i-1} \otimes \mathfrak{A}_i$. We know that $\tilde{S}_i^c$ maps into $\mathfrak{A}_{\leqslant i-1} \otimes \mathfrak{A}_i$ and that $\left[S_i^c[A_i \otimes N_i], T[\mathfrak{A}_{\leqslant i-1}]\right] = \{0\}$. Now we take an arbitrary element Z of the image of $\tilde{S}_i^c$. Z thus has the above properties. $\mathfrak{A}_i$ is finite dimensional, therefore we can decompose Z as $Z = \sum_{i,j} z_{ij} \otimes e_{ij}$, where e_{ij} are an orthogonal basis of $\mathfrak{A}_i$ (the matrix units). It follows that

$$\begin{aligned} & [\sum_{i,j} z_{ij} \otimes e_{ij}, T[\mathfrak{A}_{\leqslant i-1}]] = 0 \\ \Leftrightarrow \quad & \left[z_{ij}, T[\mathfrak{A}_{\leqslant i-1}]\right] = 0 \ \forall i, j \\ \Leftrightarrow \quad & Z \in \mathfrak{N}_{i-1} \otimes \mathfrak{A}_i. \end{aligned}$$

5. The maps S_i^c and $\tilde{S}_i^c$ are quantum channels:

$$\begin{aligned} S_i^c[A_1 \otimes M_1 \cdot A_2 \otimes M_2] &= T[A_1 A_2] M_1 M_2 \\ &= T[A_1] M_1 T[A_2] M_2 \\ &= S_i^c[A_1 \otimes M_1] S_i^c[A_2 \otimes M_2], \end{aligned}$$

where we used that $[T[\mathfrak{A}_{\leqslant i}], \mathfrak{M}_1] = \{0\}$. Furthermore, we have

$$S_i^c[A \otimes M]^* = T[A]^* M^* = T[A^*] M^* = S_i^c[A^* \otimes M^*].$$

Therefore, S_i^c is a C^*-homomorphism and therefore completely positive. S_i^c is furthermore obviously unital and thus a quantum channel. An analogous argument holds for $\tilde{S}_i^c$.

6. Now we show that the construction is consistent with respect to different block sizes. The decomposition does not use any information on the size of the blocks and the associated algebras. Thus it should not matter if we use for example $\mathfrak{B}_i = \mathfrak{A}_i \otimes \mathfrak{A}_{i+1}$ for a step of the decomposition and get the channel $S^c_{[i,i+1]} : \mathfrak{A}_i \otimes \mathfrak{A}_{i+1} \otimes \mathfrak{M}_{i+1} \to \mathfrak{M}_{i-1} \otimes \mathfrak{A}_i \otimes \mathfrak{A}_{i+1}$, or if we do two individual steps with $\mathfrak{A}_i$ and $\mathfrak{A}_{i+1}$ and concatenate the obtained channels S_i^c and S_{i+1}^c as shown in Figure 4.4. Following the definition for S_i^c, the channel $S^c_{[i,i+1]}$ is

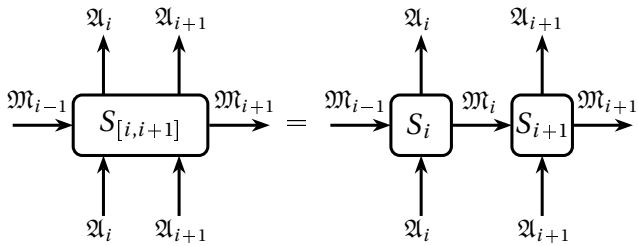


Figure 4.4: The decomposition of causal operations is consistent for different block sizes. Concatenating the channels of a fine grained decomposition one can construct the channels of a coarse grained decomposition. Proving this for the concatenation of two channels is sufficient; the rest follows by induction.

defined by

$$S^c_{[i,i+1]}[A_i \otimes A_{i+1} \otimes M_{i+1}] = T[A_i \otimes A_{i+1}] M_{i+1}. \tag{4.7}$$

This is well defined, as it is exactly the same definition as for S_i^c which does not consider the internal structure of $\mathfrak{A}_i$ at all. The memory algebras $\mathfrak{M}_i$ are

also the same in both cases by definition. The concatenation of S_i^c and S_{i+1}^c gives us

$$\begin{aligned} & \left(S_i^c \otimes \mathrm{id}_{i+1}\right) \circ \left(\mathrm{id}_i \otimes S_{i+1}^c\right) \left[A_i \otimes A_{i+1} \otimes M_{i+1}\right] \\ = \quad & \left(S_i^c \otimes \mathrm{id}_{i+1}\right) \left[\mathrm{id}_i[A_i] \otimes S_{i+1}[A_{i+1} \otimes M_{i+1}]\right] \\ = \quad & \left(S_i^c \otimes \mathrm{id}_{i+1}\right) \left[A_i \otimes \underbrace{T[A_{i+1}]M_{i+1}}_{=Z \in \mathfrak{M}_i \otimes \mathfrak{A}_{i+1}}\right]. \end{aligned} \tag{4.8}$$

The algebra $\mathfrak{A}_{i+1}$ is finite dimensional and we can decompose Z into $Z = \sum_{i,j} z_{ij} \otimes e_{ij}$. Thus we have

$$\begin{aligned} \left(S_i^c \otimes \mathrm{id}_{i=1}\right) \left[A_i \otimes T[A_{i+1}]M_{i+1}\right] &= \sum_{i,j} \underbrace{T[A_i]z_{ij}}_{\in \mathfrak{M}_{i-1} \otimes \mathfrak{A}_i} \otimes e_{ij} \\ &= T[A_i] \cdot \underbrace{\sum_{i,j} z_{ij} \otimes e_{ij}}_{\in \mathfrak{M}_{i-1} \otimes \mathfrak{A}_i \otimes \mathfrak{A}_{i+1}} \\ &= T[A_i] \cdot \left(T[A_{i+1}]M_{i+1}\right) \\ &= T[A_i \otimes A_{i+1}]M_{i+1} \end{aligned}$$

and therefore the concatenation equals the decomposition for a block size of two systems. As we did not make any assumptions on the size of the subsystems this arguments works by induction for any finite number of subsystems.

It is now easy to prove that the concatenation of S_i^c actually implements T. First we use the concatenation property to replace $S_j^c \circ \cdots \circ S_i^c$ by $S_{[j,i]}^c$. Then

$$R_{<j}^c \otimes \mathrm{id}_{[j,i]} \circ S_{[j,i]}^c[A \otimes \mathbb{1}_{\mathfrak{M}_i}] = R_{<j}^c \otimes \mathrm{id}_{[j,i]}\left[T[A]\mathbb{1}_{\mathfrak{M}_i}\right] = T[A]$$

for all $A \in \mathfrak{A}_{[j,i]}$. An analogous argument holds for $\tilde{S}_i^c$.

All that is left is to show that $\mathfrak{M}_i$ and $\mathfrak{N}_i$ have trivial center. Let us begin with $\mathfrak{N}_i$: we take an arbitrary $Z \in \mathfrak{N}_i \cap \mathfrak{N}_i'$ and show that it commutes with $T[\mathfrak{A}_j]\ \forall j$.

$\boldsymbol{j \leqslant i}$ $\left[Z, T[\mathfrak{A}_j]\right] = \{0\}$, because $Z \in \mathfrak{N}_i$.

$\boldsymbol{j > i}$ $T[A_j] = (R_{\leqslant i}^c \otimes \mathrm{id}_{[i+1,j]} \circ \tilde{S}_{[i+1,j]}^c)[\mathbb{1}_{[i+1,j-1]} \otimes A_j \otimes \mathbb{1}_{\mathfrak{N}_i}] \in \mathfrak{N}_i \otimes \mathfrak{A}_{[i+1,j]}$ for all $A_j \in \mathfrak{A}_j$. Therefore, $\left[Z, T[\mathfrak{A}_j]\right] = \{0\}$ because $Z \in \mathfrak{N}_i'$.

All j are covered, so $[Z, \mathfrak{A}] = [Z, T[\mathfrak{A}]] = \{0\}$ because T is an automorphism. Thus Z is in the center of $\mathfrak{A}$. The chain algebra $\mathfrak{A}$ has trivial center, therefore $Z \in \mathbb{C}\mathbb{1}$ and $\mathfrak{N}_i$ has trivial center. The memory algebra $\mathfrak{M}_i$ is a subalgebra of $\mathfrak{N}_i$, therefore the same argument holds for $\mathfrak{M}_i$ and the center of $\mathfrak{M}_i$ is also trivial. □

If we know that T has finite depth τ the situation is simplified further. The decomposition does not need an initialization operation R any more (see Figure 4.5), and the two memory definitions coincide.

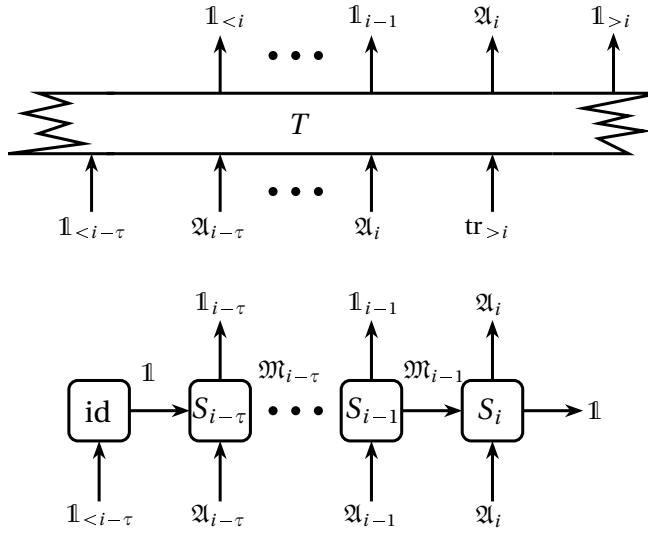


Figure 4.5: Decomposition of a causal operation with finite depth τ; the id operation at the beginning of the chain only maps the identity of the representation of the channel memory algebra onto the identity of the chain algebra.

Lemma 4.1.2. *Let $T : \mathfrak{A}_{\mathbb{Z}} \to \mathfrak{A}_{\mathbb{Z}}$ be a causal operation with finite depth τ and let $\mathfrak{A}_{\mathbb{Z}} = \bigotimes_{i=-\infty}^{\infty} \mathcal{M}_{d_i}$. It follows that T can be decomposed into a sequence of memory channels S_i in the sense that for any finitely localized observable $A \in \mathfrak{A}_{[j,i]} \subset \mathfrak{A}_{\mathbb{Z}}$ we have*

$$T[A] = (id_{\mathrm{init}} \otimes id_{[j-\tau,i]} \circ S_{j-\tau} \otimes id_{[j-\tau+1,i]} \circ \cdots \circ id_{[j-\tau,i-1]} \otimes S_i)[A \otimes \mathbb{1}_{\mathfrak{M}_i}]. \quad (4.9)$$

Again we use the implementations S_i^c of the channels S_i on the chain as in Theorem 4.1.1. The channel id_{init} maps the identity of the representation of $\mathfrak{M}_{j-\tau-1}$ used by the channel on the identity of $\mathfrak{A}_{\mathbb{Z}}$. Furthermore, all $\mathfrak{N}_i = \mathfrak{M}_i \cong \mathcal{M}_m$ are isomorphic to the same full matrix algebra.

Proof. We use the results from Theorem 4.1.1 as a starting point. It is clear from the definition of finite depth that $\mathfrak{M}_i$ is contained in $\mathfrak{A}_{[i-\tau,i]}$:

$$\mathfrak{M}_i = \mathbf{S}(T[\mathfrak{A}_{>i}], \mathfrak{A}_{\leqslant i}) \subset \mathbf{S}(\mathfrak{A}_{>i-\tau}, \mathfrak{A}_{\leqslant i}) \subset \mathfrak{A}_{[i-\tau,i]}.$$

To prove the decomposition formula we start from (4.1) and decompose the map $R_{<j}: \mathfrak{M}_{j-1} \to \mathfrak{A}_{<j}$ as

$$\begin{aligned} R_{<j}[M_{j-1}] &= (\mathrm{id}_{\mathrm{init}} \otimes \mathrm{id}_{[j-\tau,j-1]} \circ \hat{S}_{j-\tau} \otimes \mathrm{id}_{[j-\tau+1,j-1]} \circ \cdots \\ &\quad \cdots \circ \hat{S}_{j-1})[M_{j-1}], \end{aligned}$$

where $\hat{S}_i$ is the channel where the outputs are ignored: $\hat{S}_i[M_i] := S_i[\mathbb{1}_{\mathfrak{A}_i} \otimes M_i]$. The memory algebra $\mathfrak{M}_{j-1}$ is localized in $[j-\tau, j-1]$, therefore the output of the decomposition after $S_{j-\tau}$ has been applied is always $\mathbb{1}_{\mathfrak{M}_{j-\tau-1}}$. The operation $\mathrm{id}_{\mathrm{init}}$ just maps this to the identity of $\mathfrak{A}$.[1]

To show that $\mathfrak{N}_i = \mathfrak{M}_i$ let us first note that $\mathfrak{N}_i \subset \mathfrak{A}_{[i-\tau,i]}$: by definition $\mathfrak{N}_i = \{A \in \mathfrak{A}_{\leqslant i} | \, [A, T[\mathfrak{A}_{\leqslant i}]] = \{0\}\}$. As T is an automorphism, $T^{-1}[A] \in \mathfrak{A}_{>i}$. The finite depth of T then implies $A \in \mathfrak{A}_{[i-\tau,i]}$ and therefore $A \in \mathbf{S}(T[\mathfrak{A}_{>i}], \mathfrak{A}_{\leqslant i}) = \mathfrak{M}_i$ and $\mathfrak{N}_i = \mathfrak{M}_i$.

Let us now prove that all $\mathfrak{M}_i$ are isomorphic to full matrix algebras $\mathcal{M}_{d_i}(\mathbb{C})$. By definition T is of finite depth τ, therefore $\mathfrak{M}_i \subset \mathfrak{A}_{[i-\tau,i]} = \mathfrak{A}_{i-\tau} \otimes \cdots \otimes \mathfrak{A}_i$ and $\mathfrak{M}_i$ finite dimensional. Finite dimensional C^*-algebras are isomorphic to direct sums of full matrix algebras (see Section 2.1), where the number of summands equals the dimension of the center. Thus a finite dimensional C^*-algebra with one-dimensional center is isomorphic to a full matrix algebra. As $\mathfrak{M}_i$ has trivial center it is isomorphic to a full matrix algebra.

Now let us assume that $\dim \mathfrak{M}_i > \dim \mathfrak{M}_{i-1}$. In this case the image of $S_i^c : \mathfrak{A}_i \otimes \mathfrak{M}_i \to \mathfrak{M}_{i-1} \otimes \mathfrak{A}_i$ has a lower dimension than the preimage and S_i^c must have a non-trivial kernel. Let us assume that $A \in \mathfrak{A}_i \otimes \mathfrak{M}_i$ is in the kernel of S_i^c: $S_i^c[A] = \mathbb{1}$. We then have $[S_i^c[A], S_i^c[\mathfrak{A}_i \otimes \mathfrak{M}_i]] = \{0\}$. However, S_i^c is a homomorphism, so $[A, \mathfrak{A}_i \otimes \mathfrak{M}_i] = \{0\}$ and A is central in $\mathfrak{A}_i \otimes \mathfrak{M}_i$. $\mathfrak{A}_i$ and $\mathfrak{M}_i$ are both full matrix algebras so A central implies $A = \mathbb{1}$. Therefore the kernel of S_i^c is trivial and $\dim \mathfrak{M}_i \leqslant \dim \mathfrak{M}_{i-1}$.

On the other hand we can assume that $\dim \mathfrak{M}_i < \dim \mathfrak{M}_{i-1}$. We have just shown that S_i^c is an injective homomorphism—a monomorphism. Therefore, the map $S_i^c : \mathfrak{A}_i \otimes \mathfrak{M}_i \to S_i^c[\mathfrak{A}_i \otimes \mathfrak{M}_i]$ is also onto and thus an isomorphism. We obtain $S_i^c[\mathfrak{A}_i \otimes \mathfrak{M}_i] \cong \mathfrak{A}_i \otimes \mathfrak{M}_i$ and therefore a full matrix algebra of dimension $d_i m_i$. By assumption we have $S_i^c[\mathfrak{A}_i \otimes \mathfrak{M}_i] \subsetneq \mathfrak{M}_{i-1} \otimes \mathfrak{A}_i$ which is also a full matrix algebra of dimension $d_i m_{i-1}$. Therefore, $\mathfrak{M}_{i-1} \otimes \mathfrak{A}_i \cong \mathfrak{M}_i \otimes \mathfrak{A}_i \otimes \mathfrak{B}_i$ and $\mathfrak{M}_{i-1} \cong \mathfrak{M}_i \otimes \mathfrak{B}_i$ for some full matrix algebra $\mathfrak{B}_i \neq \mathbb{C}\mathbb{1}$. As T has finite depth $\mathfrak{B}_i \subset \mathfrak{A}_{[i-\tau,i]}$. Now we take an arbitrary $B \in \mathfrak{B}_i$ and investigate its commutation properties. From $\mathfrak{B}_i \subset \mathfrak{M}_{i-1}$ we deduce $[B, T[\mathfrak{A}_{\leqslant i}]] = \{0\}$. Furthermore, $T[\mathfrak{A}_{>i}] \subset \mathbf{S}(T[\mathfrak{A}_{>i}], \mathfrak{A}_{\leqslant i}) \otimes \mathfrak{A}_{>i}$. The commutator $[B, \mathfrak{A}_{>i}] = \{0\}$ because $B \in \mathfrak{A}_{[i-\tau,i]}$. Finally $[B, \mathbf{S}(T[\mathfrak{A}_{>i}], \mathfrak{A}_{\leqslant i})] = [B, \mathfrak{M}_i] = [B, S_i^c[\mathbb{1} \otimes \mathfrak{M}_i]]$. But $B \notin S_i^c[\mathbb{1} \otimes \mathfrak{M}_i]$ because by assumption B is not in the image of S_i^c. $\mathfrak{B}_i$ and $\mathfrak{M}_i$ are full matrix algebras, so $[B, \mathfrak{M}_i] = \{0\}$. Putting

[1] This is only necessary in the case that the channel uses another representation of $\mathfrak{M}_i$, which makes sense as the representation of $\mathfrak{M}_i$ on $\mathfrak{A}_{\mathbb{Z}}$ is reducible and therefore uses more resources then necessary.

it all together we have $[\mathfrak{B}_i, \mathfrak{A}] = [\mathfrak{B}_i, T[\mathfrak{A}]] = \{0\}$ and therefore $\mathfrak{B} = \mathbb{C}\mathbb{1}$. Therefore $\dim \mathfrak{M}_i \geqslant \dim \mathfrak{M}_{i-1}$ and thus $\mathfrak{M}_i \cong \mathfrak{M}_{i-1} \cong \mathfrak{M} = \mathcal{M}_m$. The required memory dimension is m and the same for all channels S_i^c.

Finally we can note that $\mathfrak{M}_i$ is embedded in $\mathfrak{A}_{[i-\tau,i]}$ and therefore m divides the cell dimension $\prod_{j=i-\tau}^{i} d_j$. □

It is also of interest to investigate the properties of the memory channels that correspond to translation invariant causal operations, especially their forgetfulness. It turns out that a memory channel corresponds to a causal operation if and only if it is forgetful. Furthermore, it corresponds to a finite depth causal operation if and only if it is strictly forgetful. Here the memory channels will not be implemented on the chain, but have an additional memory system as in the usual definition of memory channels. Therefore, the initialization operation $R_{<i}$ now actually does something non-trivial. We first treat the forgetful case and then move on to the special case of strictly forgetful channels. In the theorems we say that the channels have a forgetful representation. This comes from the fact that several memory channels can implement the same transformation on the chain of systems. In particular, a channel can have additional memory dimensions which do not interact with the other systems at all. Such a channel would not be forgetful (unless it is irreversible on the memory), but could still implement a causal operation. To circumvent this problem we speak of channels which have a (strictly) forgetful representation instead of (strictly) forgetful channels. In the proof we use the construction of a memory channel from the causal operation which will be (strictly) forgetful by definition.

Theorem 4.1.3. *Let $S_i : \mathfrak{A}_i \otimes \mathfrak{M} \to \mathfrak{M} \otimes \mathfrak{A}_i$ be a quantum memory channel. It follows that $T : \mathfrak{A}_{\mathbb{Z}} \to \mathfrak{A}_{\mathbb{Z}}$, $T[A_{[i,j]}] = \lim_{n\to\infty} R_{<i-n} \otimes id_{[i-n,j]} \circ S_{[i-n,j]}[A_{[i,j]}]$ is a quasi-local causal process if and only if S_i has a forgetful representation.*

Proof. Let us first assume S_i is forgetful. Then by [51], Proposition 5, for every $M \in \mathfrak{M}$ and for all $\varepsilon > 0$ there exists an $N \in \mathbb{N}$ and an $A_{[i-N,i]} \in \mathfrak{A}_{[i-N,i]}$ such that

$$\left\| \hat{S}_{[i-N,i]}[M] - \mathbb{1}_{\mathfrak{M}} \otimes A_{[i-N]} \right\|_{\infty} \leqslant \varepsilon \left\| M \right\|_{\infty}. \tag{4.10}$$

To show that T is quasi-local we need to show that the images of localized observables can be approximated by localized observables up to any $\varepsilon > 0$: for all $A \in \mathfrak{A}$, A finitely localized, and every $\varepsilon > 0$ there exists a finite set $\Lambda \subset \mathbb{Z}$ and an observable $A_\varepsilon \in \mathfrak{A}(\Lambda)$ such that $\left\| T[A] - A_\varepsilon \right\|_{\infty} \leqslant \varepsilon$. As Λ is finite it is contained in some interval $[i-N, j]$.

For a localized observable A, $A \in \mathfrak{A}_{[i,j]}$, $T[A]$ can be expressed in terms of S:

$$\begin{aligned} T[A] &= (R_{<i-N} \otimes \mathrm{id}_{[i-N,j]}) \circ (\hat{S}_{[i-N,i-1]} \otimes \mathrm{id}_{[i,j]}) \circ S_{[i,j]}[A] \\ &= (R_{<i-N} \otimes \mathrm{id}_{[i-N,j]}) \circ (\hat{S}_{[i-N,i-1]} \otimes \mathrm{id}_{[i,j]})[\sum_k M_k \otimes A_k],\ A_k \in \mathfrak{A}_{[i,j]} \\ &= \sum_k (R_{<i-N} \otimes \mathrm{id}_{[i-N,i-1]}) \circ \hat{S}_{[i-N,i-1]}[M_k] \otimes A_k. \end{aligned}$$

Now let us construct an approximation A_ε of $T[A]$:

$$\begin{aligned} & \|T[A] - A_\varepsilon\|_\infty \\ = & \left\| \sum_k (R_{<i-N} \otimes \mathrm{id}_{[i-N,i-1]}) \circ \hat{S}_{[i-N,i-1]}[M_k] \otimes A_k - \sum_k \tilde{A}_{\varepsilon,k} \otimes A_k \right\|_\infty \\ \leqslant & \sum_k \left\| (R_{<i-N} \otimes \mathrm{id}_{[i-N,i-1]}) \circ \hat{S}_{[i-N,i-1]}[M_k] - \tilde{A}_{\varepsilon,k} \right\|_\infty \cdot \|A_k\|_\infty \\ \leqslant & \sum_k \left\| \hat{S}_{[i-N,i-1]}[M_k] - \mathbb{1}_{\mathfrak{M}} \otimes A_{[i-N,i-1],k} \right\|_\infty \cdot \|A_k\|_\infty . \end{aligned}$$

Every summand can be made smaller than any $\varepsilon_k > 0$ by Equation (4.10) and all A_k are bounded. Thus the sum is finite and can be made smaller than any $\varepsilon = \sum_k \varepsilon_k \|A_k\|_\infty$. Therefore, $T[A]$ can be approximated by finitely localized observables and hence T is quasi-local.

Now let us assume that T is quasi-local. We have to show that Equation (4.10) holds. For every $M \in \mathfrak{M}$ we have

$$\begin{aligned} & \left\| \hat{S}_{[i-N,i]}[M] - \mathbb{1}_{\mathfrak{M}} \otimes A_{[i-N,i]} \right\|_\infty \\ = & \left\| (R_{<i-N} \otimes \mathrm{id}_{[i-N,i]}) \circ \hat{S}_{[i-N,i]}[M] - \mathrm{id}_{\mathrm{init}}[\mathbb{1}_{\mathfrak{M}}] \otimes A_{[i-N,i]} \right\|_\infty \\ = & \left\| T[R_{<i}[M]] - A_{[i-N,i]} \right\|_\infty \leqslant \varepsilon_M, \end{aligned}$$

because T is quasi-local. The $R_{<i}$ in $T[R_{<i}[M]]$ is needed to bring the M from the channels memory algebra to the memory algebra on the chain. Now we write $\varepsilon_M = \tilde{\varepsilon}_M \|M\|_\infty$, take $\varepsilon = \sup_M \tilde{\varepsilon}_M$ and obtain Equation (4.10). □

Unsurprisingly, if the channel is strictly forgetful, the resulting causal operation has finite memory depth.

Corollary 4.1.4. *Given S_i and T as in Theorem 4.1.3. It follows that T is finite depth if and only if S_i has a strictly forgetful representation. Furthermore the memory depth τ of T equals the memory depth of S_i.*

Proof. First let us assume that S_i is strictly forgetful with depth τ. We have already proven that in this case T is quasi-local. What is left to show is that $T[A_{[i,j]}] \subset$

$\mathfrak{A}_{[i-\tau,j]}$ for all $A_{[i,j]} \in \mathfrak{A}_{[i,j]}$. Let $A_{[i,j]} \in \mathfrak{A}_{[i,j]}$. The the decomposition (4.9) has already been shown in Lemma 4.1.2:

$$T[A_{[i,j]}] = (\mathrm{id}_{\mathrm{init}} \otimes \mathrm{id}_{[i-\tau,j]}) \circ (\hat{S}_{[i-\tau,i-1]} \otimes \mathrm{id}_{[i,j]}) \circ S_{[i,j]}[A_{[i,j]}] \in \mathfrak{A}_{[i-\tau,j]}$$

and therefore T is of depth τ.

On the other hand let T be of depth τ. It follows that $T[M] = \mathrm{id}_{\mathrm{init}} \circ \hat{S}_{[i-\tau,i]}[M] \in \mathfrak{A}_{[i-\tau,i]}$ and thus $\hat{S}_{[i-\tau,i]}[M] \in \mathfrak{M} \otimes \mathfrak{A}_{[i-\tau,i]}$. Now take $\tilde{S}_{[i-n,i]} = \mathrm{tr}_{\mathfrak{M}}(\hat{S}_{[i-\tau,i]}[M])$ for all $n \geqslant \tau$. We have $\left\| \hat{S}_{[i-n,i]}[M] - \mathbb{1}_{\mathfrak{M}} \otimes \tilde{S}_{[i-n,i]} \right\|_{\infty} = 0$ for all $n \geqslant \tau$ and therefore S_i is strictly forgetful with depth τ. □

Remark 4.1.5. *In the literature, the theory of memory channels assumes translation-invariance, because it describes the situation where the same transmission channel is used several times. However, it also makes sense to study if a series of compatible memory channels is forgetful—compatible meaning that the memory dimension of all channels is the same. Our proofs do not assume translation invariance at any point. To generalize the results to non-translation-invariant systems one only has to extend the notion of forgetfulness to this setting and check if all the theorems of [51] also hold in this extended setting.*

4.2 Memory requirements and the index

Now we will investigate the minimal memory needed to implement a causal operation by a memory channel to pave the ground for a resource efficient implementation if the causal operations. A finite depth causal operation T is obviously a QCA. We will now show that the memory requirement of the channel representation S_i of T equals the index of T. As introduced in Section 3.2 the index is defined for nearest neighbor QCAs using support algebras that describe the information flow. For a general nearest neighbor QCA we have

$$\begin{aligned} \mathfrak{R}_{2i} &= \mathbf{S}(T[\mathfrak{A}_{2i} \otimes \mathfrak{A}_{2i+1}], \mathfrak{A}_{2i-1} \otimes \mathfrak{A}_{2i}), \\ \mathfrak{R}_{2i+1} &= \mathbf{S}(T[\mathfrak{A}_{2i} \otimes \mathfrak{A}_{2i+1}], \mathfrak{A}_{2i+1} \otimes \mathfrak{A}_{2i+2}). \end{aligned}$$

The index is defined as

$$\mathrm{ind}\, T := \frac{r_{2i}}{d_{2i}} = \frac{d_{2i+1}}{r_{2i+1}},$$

where d_i is the dimension of $\mathfrak{A}_i = \mathcal{M}_{d_i}$ and r_i is the dimension of $\mathfrak{R}_i = \mathcal{M}_{r_i}$.

To bring the causal process T into nearest neighbor form we have to group cells. By grouping together τ cells we obtain larger cell algebras

$$\mathfrak{B}_i = \bigotimes_{j=\tau i}^{\tau i+\tau-1} \mathfrak{A}_j = \mathfrak{A}_{[\tau i,(\tau+1)i-1]}.$$

With these larger algebras we have $T[B_i] \in T[\mathfrak{A}_{[\tau i,(\tau+1)i-1]}] \subset \mathfrak{A}_{[\tau i-\tau,(\tau+1)i-1]} = \mathfrak{B}_{i-1} \otimes \mathfrak{B}_i$ for all $B_i \in \mathfrak{B}_i$. Thus the grouping makes T a nearest neighbor QCA. We obtain

$$\begin{aligned} \mathfrak{R}_{2i} &= \mathbf{S}(T[\mathfrak{B}_{2i} \otimes \mathfrak{B}_{2i+1}], \mathfrak{B}_{2i-1} \otimes \mathfrak{B}_{2i}), \\ \mathfrak{R}_{2i+1} &= \mathbf{S}(T[\mathfrak{B}_{2i} \otimes \mathfrak{B}_{2i+1}], \mathfrak{B}_{2i+1} \otimes \mathfrak{B}_{2i+2}) \\ &= \mathbf{S}(T[\mathfrak{B}_{2i+1}], \mathfrak{B}_{2i+1}). \end{aligned}$$

Obviously, $\mathfrak{R}_{2i}$ and $\mathfrak{R}_{2i+1}$ commute. Following [38] we can furthermore state that $\mathfrak{R}_{2i+1}$ and $\mathfrak{R}_{2i+2}$ also commute, because T is an automorphism. Of course two algebras $\mathfrak{R}_i$ and $\mathfrak{R}_j$ which are further apart also commute. Therefore, all $\mathfrak{R}_i$ commute pairwise. By definition $\mathfrak{R}_{\mathbb{Z}} = \bigotimes_i \mathfrak{R}_i$ contains $T[\mathfrak{A}_{\mathbb{Z}}]$ and as T is an automorphism $\mathfrak{A}_{\mathbb{Z}} \subset \mathfrak{R}_{\mathbb{Z}}$. On the other hand, by definition $\mathfrak{R}_{\mathbb{Z}} \subset \mathfrak{A}_{\mathbb{Z}}$ and thus $\mathfrak{R}_{\mathbb{Z}} = \mathfrak{A}_{\mathbb{Z}}$. We can conclude that all $\mathfrak{R}_i$ each have a trivial center, because every element in the center of an $\mathfrak{R}_i$ is also in the center of $\mathfrak{R}_{\mathbb{Z}}$ which is trivial. As the $\mathfrak{R}_i$ are finite-dimensional, they are full matrix algebras $\mathcal{M}_{r_i}$.

By definition we have $T[\mathfrak{B}_{2i} \otimes \mathfrak{B}_{2i+1}] \subset \mathfrak{R}_{2i} \otimes \mathfrak{R}_{2i+1}$. Because $\mathfrak{R} = \mathfrak{A}$ we actually have equality. We have

$$\mathfrak{R}_{2i} \otimes \mathfrak{R}_{2i+1} = T[\mathfrak{B}_{2i} \otimes \mathfrak{B}_{2i+1}] \subset \mathfrak{M}_{2i-1} \otimes \mathfrak{B}_{2i} \otimes \mathfrak{B}_{2i+1}$$

and $\mathfrak{R}_{2i} \subset \mathfrak{B}_{2i-1} \otimes \mathfrak{B}_{2i}$ which leads to $\mathfrak{R}_{2i} \subset \mathfrak{M}_{2i-1} \otimes \mathfrak{B}_{2i}$. On the other hand $\mathfrak{M}_{2i-1} \otimes \mathfrak{B}_{2i}$ commutes with all $\mathfrak{R}_i$ except $\mathfrak{R}_{2i}$: we only need to check commutation with $\mathfrak{R}_{2i-1}$, because the support of all other $\mathfrak{R}_i$ lies on other cells and therefore commute by definition. For $\mathfrak{R}_{2i-1}$ we have

$$\mathfrak{R}_{2i-1} = \mathbf{S}(T[\mathfrak{B}_{2i-1}], \mathfrak{B}_{2i-1})$$

which commutes with $\mathfrak{M}_{2i-1} \otimes \mathfrak{B}_{2i}$. Therefore, $\mathfrak{M}_{2i-1} \otimes \mathfrak{B}_{2i} \subset \mathfrak{R}_{2i}$ and

$$\mathfrak{R}_{2i} = \mathfrak{B}_{2i-1} \otimes \mathfrak{B}_{2i}. \tag{4.11}$$

Now we can compute the index:

$$\operatorname{ind} T = \frac{r_{2i}}{d_{2i}} = \frac{m_{2i-1} d_{2i}}{d_{2i}} = m_{2i-1}, \tag{4.12}$$

where m_{2i-1} is the dimension of the memory algebra $\mathfrak{M}_{2i-1} = \mathcal{M}_{m_{2i-1}}$. As the memory is translation invariant (and so is the index) we have $\operatorname{ind} T = m$. Now let us summarize our findings in a theorem.

Theorem 4.2.1. *Let $T : \mathfrak{A}_{\mathbb{Z}} \to \mathfrak{A}_{\mathbb{Z}}$ be a causal automorphism with finite depth. It follows that the dimension of the memory algebra $\mathfrak{M} = \mathcal{M}_m$ needed to implement T by a series of memory channels $S_i : \mathfrak{A}_i \otimes \mathfrak{M} \to \mathfrak{M} \otimes \mathfrak{A}_{i-1}$ equals the index of T:*

$$m = \operatorname{ind} T. \tag{4.13}$$

Furthermore, ind T *is an integer.*

Proof. The first part was already proven above. As m is an integer ind T has to be an integer. □

4.3 A memory efficient representation

So far we ignored the question how to determine a representation of S_i that acts only on $\mathfrak{A}_i$ and an additional memory algebra $\mathfrak{M}$ that is not embedded into the whole chain algebra $\mathfrak{A}_\mathbb{Z}$. The representation S_i^c on the chain is useful for the proofs, but to implement the channel in this way we would need global access to the chain algebra $\mathfrak{A}_\mathbb{Z}$ in case of general causal operations by forgetful channels and access to $\tau+1$ subsystems at a time for the implementation of finite depth causal operations by strictly forgetful channels. With this representation we need a lot of memory to store all those systems at the same time—something we wanted to avoid by building the memory channel representation in the first place. To implement S_i as a memory channel in a resource efficient way the memory algebra has to be the observable algebra of an additional system $\mathscr{M}$ that is passed on from step to step. In each step the channel S_i only needs access to the system $\mathscr{A}_i$ and the memory system $\mathscr{M}$. We will now present a way to derive such a representation in the case of finite depth causal operations T.

The memory $\mathfrak{M}_i$ at position i is defined as the support algebra $\mathbf{S}(T[\mathfrak{A}_{>i}], \mathfrak{A}_{\leqslant i})$. T is of depth τ thus $\mathfrak{M}_i \subset \mathfrak{A}_{[i-\tau,i]}$ and $\mathfrak{M}_i \cong \mathcal{M}_m$. The full matrix algebra $\mathcal{M}_m$ has a basis of m^2 Weyl operators $\mathrm{w}(\xi_j)$, $1 \leqslant j \leqslant m^2$. Now we pick a basis $M_{i,j}$ of $\mathfrak{M}_i$ and a basis $M_{i-1,j}$ of $\mathfrak{M}_{i-1}$ which fulfill the same internal relations (multiplication, addition) as $\mathrm{w}(\xi_j)$. Furthermore, we also pick a basis $\mathrm{w}(\eta_{i,j})$, $m^2 < j \leqslant m^2 + d_i^2$ of $\mathfrak{A}_i$. Now we have all that is needed to determine $S_i : \mathfrak{A}_i \otimes \mathfrak{M} \to \mathfrak{M} \otimes \mathfrak{A}_i$: to fix $S_i[\mathbb{1}_\mathfrak{A} \otimes \mathrm{w}(\xi_j)]$ we use $S_i^c[M_{i,j}] = M_{i,j} = \sum_k M_{i-1,k} \otimes A_{j,k} = \sum_{k,l} a_{i,j,k,l} M_{i-1,k} \otimes \mathrm{w}(\eta_{i,l})$ and obtain

$$S_i[\mathbb{1}_{\mathfrak{A}_i} \otimes \mathrm{w}(\xi_j)] = \sum_{k,l} a_{i,j,k,l} \mathrm{w}(\xi_k) \otimes \mathrm{w}(\eta_{i,l}) = \sum_x a_{i,j,x} \mathrm{w}(\zeta_{i,x}),$$

where $\zeta_{i,x} = \xi_k \oplus \eta_{i,l}$ and x covers all combinations of k and l. Analogously, we derive

$$S_i[\mathrm{w}(\eta_{i,j}) \otimes \mathbb{1}_\mathfrak{M}] = \sum_x a_{i,j,x} \mathrm{w}(\zeta_{i,x}).$$

Finally, we obtain

$$\begin{aligned} S_i[\mathrm{w}(\zeta_{i,k})] &= S_i[\mathrm{w}(\eta_{i,j} \oplus \xi_l)] \\ &= S_i[\mathrm{w}(\eta_{i,j} \otimes \mathbb{1}_{\mathfrak{M}})] S_i[\mathbb{1}_{\mathfrak{A}_i} \otimes \mathrm{w}(\xi_l)] \\ &= \left(\sum_x a_{i,j,x} \mathrm{w}(\zeta_{i,x}) \right) \left(\sum_y a_{i,l,y} \mathrm{w}(\zeta_{i,y}) \right) \\ &= \sum_{x,y} a_{i,j,x} a_{i,l,y} \mathrm{w}(\zeta_{i,x} + \zeta_{i,y}) \mathrm{e}^{-i\beta(\zeta_{i,x}, \zeta_{i,y})} \\ &= \sum_p s_{i,kp} \mathrm{w}(\zeta_{i,p}), \end{aligned}$$

where

$$s_{i,kp} = \sum_{x,y} a_{i,j,x} a_{i,l,y} \mathrm{e}^{i\beta(\zeta_{i,x}, \zeta_{i,y})} \quad \text{with } x, y \text{ s.t. } \zeta_{i,x} + \zeta_{i,y} = \zeta_{i,p}. \tag{4.14}$$

Thus the channel S_i is defined by a $(d_i^2 m^2) \times (d_i^2 m^2)$-matrix with entries $s_{i,kp}$ such that

$$S_i[\mathrm{w}(\zeta_{i,k})] = \sum_p s_{i,kp} \mathrm{w}(\zeta_{i,p}), \tag{4.15}$$

with the coefficients $s_{i,kp}$ defined in Equation (4.14).

4.3.1 Clifford causal operations

For Clifford causal operations it is especially easy to derive a channel implementation. Pauli products are mapped to Pauli products so it is easy to divide them up to input and memory system. To further simplify the situation, we restrict ourselves to qubits and causal operations which are translation-invariant with respect to shifts by n qubits. Possible global phases are omitted.

Now let us describe an algorithm to derive a channel representation of a given causal reversible finite depth Clifford operation.[2]

- The algorithm takes as input a list of $2n$ transformation rules $\xi_i \mapsto \eta_i$, $\xi_i \in \mathbb{Z}_2^{2n}$, $\eta_i \in \mathbb{Z}_2^{2n(\tau+1)}$. These rules completely describe the causal operation T, because they are the complete images of a cell (of n qubits) and the operation is assumed to be translation invariant with respect to shifts by n qubits. We assume ξ_i to be a standard basis of $\mathbb{Z}_2^{2n}$ in standard ordering: $\xi_1 = XI\cdots$, $\xi_2 = ZI\cdots$, etc. If we are given a set of rules that is not in this form, we can always transform it to this form by adding and swapping rules. Therefore, it is sufficient to provide the algorithm with the set of η_i in the right order.

[2] If the operation was not finite depth it would correspond to a non-forgetful channel and not be a quasi-local operation. Therefore all quasi-local Clifford causal operations have finite-depth.

- The algorithm returns the phase space matrix $\mathbf{s}$ of a channel S that implements T. The ordering of systems is in accordance with the usual convention for memory channels: $S : \mathfrak{A}_i \otimes \mathfrak{M} \to \mathfrak{M} \otimes \mathfrak{A}_{i-1}$.
- The algorithm uses the following steps
  - Generate all shifted images: shift η_i to the right by j blocks (jn qubits) for $0 \leqslant j \leqslant \tau$. Everything that is shifted out at the right side will just be cut, while the left side will be padded with I, respectively $(0,0)$ in the phase space picture. E.g. $XYZ \mapsto IXY$ respectively $(1,0,1,1,0,1) \mapsto (0,0,1,0,1,1)$. We obtain an extended set η_i, $1 \leqslant i \leqslant 2n(\tau+1)$.
  - Generate the phase space representation of $\mathfrak{M}$ by cutting every element η_i $1 \leqslant i \leqslant 2n(\tau+1)$ by n qubits on the right. We call the result $\tilde{\eta}_i$, $1 \leqslant i \leqslant 2n(\tau+1)$. Each $\tilde{\eta}_i$ is of length of at most $n\tau$ qubits. Actually, the last $2n$ phase space vectors $\tilde{\eta}_i$, $2n\tau+1 \leqslant i \leqslant 2n(\tau+1)$ will always be 0, because the according η_i have only one non-zero block.
  - Now find a standard basis ζ_j, $1 \leqslant j \leqslant 2m$ for $\tilde{\eta}_i$. A standard basis is a basis such that the commutation relations are the same as the ones of $(1,0,\ldots)$, $(0,1,0,\ldots)$, etc. Such a basis always exists because the memory algebra is always isomorphic to a full matrix algebra (Lemma 4.1.2). The number of basis vectors $2m$ shows, that the channel needs m qubits of memory. It is in general not necessary to use a standard basis, but then the resulting phase space transformation would not be symplectic with respect to the standard symplectic form. Transforming an arbitrary basis into a standard basis can be carried out by the methods explained in the following chapter in Section 5.3.1 to derive commuting pairs of anticommuting operators from cut stabilizer operators.
  - Decompose the $\tilde{\eta}_i$ in terms of the basis ζ_j and obtain η_i^m, $1 \leqslant i \leqslant 2n\tau$, each being a phase space vector of length $2m$ (m qubits).
  - Determine the output-to-input transformation $\mathbf{s}_{o-i}$: take the n rightmost qubits of η_i, $1 \leqslant i \leqslant 2n$ and use them as columns of a matrix. This $2n \times 2n$-matrix is the output-to-input matrix $\mathbf{s}_{o-i}$.
  - Determine the output-to-memory transformation $\mathbf{s}_{o-m}$: use the first $2n$ η_i^m as columns of a matrix. This $2m \times 2n$-matrix is the output-to-memory transformation $\mathbf{s}_{o-m}$.
  - Determine the memory-to-input transformation $\mathbf{s}_{m-i}$: take the rightmost n qubits of the memory basis ζ_j, $1 \leqslant j \leqslant 2m$ and use them as the columns of a matrix. This $2n \times 2m$-matrix is the memory-to-input matrix $\mathbf{s}_{m-i}$.
  - Determine the memory-to-memory transformation $\mathbf{s}_{m-m}$: take ζ_j, $1 \leqslant j \leqslant 2m$ and shift them to the right by one block (n qubits). Decompose

them in terms of ζ_i and use the resulting vectors as columns of a matrix. This $2m \times 2m$-matrix is the memory-to-memory matrix $\mathbf{s}_{m-m}$.

– Compose the four matrices to obtain $\mathbf{s}$:

$$\mathbf{s} = \left(\begin{array}{c|c} \mathbf{s}_{o-m} & \mathbf{s}_{m-m} \\ \hline \mathbf{s}_{o-i} & \mathbf{s}_{m-i} \end{array}\right) \tag{4.16}$$

In both obtaining the memory-to-input and the memory-to-memory transformation we used the translation-invariance of the causal operation T.

4.4 Bratteli diagrams and the depth of strictly forgetful channels

In this section we will introduce the use of Bratteli diagrams in the analysis of quantum memory channels and causal operations. We will consider only strictly forgetful and reversible memory channels and therefore translation invariant finite-depth causal operations (see Corollary 4.1.4). Throughout this section, the notation of memory channels will be used. To analyze a causal operation we will use its memory channel representation as derived in Lemma 4.1.2 and Section 4.3. Let $S: \mathfrak{A} \otimes \mathfrak{M} \to \mathfrak{M} \otimes \mathfrak{A}$ be a reversible quantum memory channel, i.e. a CPU C^*-homomorphism. Let $\hat{S}$; $\mathfrak{M} \to \mathfrak{M} \otimes \mathfrak{A}$, $\hat{S}[M] = S[\mathbb{1}_{\mathfrak{A}} \otimes M]$ be the usual restriction to memory outputs with ignored data outputs. $\hat{S}$ is obviously also a unital C^*-homomorphism (and therefore CPU). Furthermore, let $\hat{S}_n: \mathfrak{M} \to \mathfrak{M} \otimes \mathfrak{A}^n$ be its n-fold concatenation as introduced in Section 3.5.

Our goal is to study how the memory algebra $\mathfrak{M}$ evolves under repeated use of $\hat{S}$. We assume strict forgetfulness in τ steps, therefore, by the forgetfulness condition (3.24), we know that $S_\tau[\mathfrak{M}] \subset \mathbb{1}_{\mathfrak{M}} \otimes \mathfrak{A}^{\otimes\tau}$. τ is assumed to be minimal with this property, i.e. S is not forgetful for $\delta < \tau$ steps. We generalize this to subalgebras of $\mathfrak{M}$ that are mapped to the identity of $\mathfrak{M}$ after n concatenations of $\hat{S}$. We define

$$\mathfrak{Z}_n = \left\{ M \in \mathfrak{M} | \hat{S}_n[M] \in \mathfrak{Z}_0 \otimes \mathfrak{A}^{\otimes n} \right\}, \tag{4.17}$$

where $\mathfrak{Z}_0 = \mathbb{C}\mathbb{1}_{\mathfrak{M}}$. $\mathfrak{Z}_n$ is a subalgebra of $\mathfrak{M}$ because $\hat{S}$ is a C^*-homomorphism. By assumption of strict forgetfulness we have $\mathfrak{Z}_\tau = \mathfrak{M}$. We will now use this set of subalgebras of $\mathfrak{M}$ to derive an upper bound on the memory depth of S given $\mathfrak{M}$. First let us show that $\mathfrak{Z}_n \subset \mathfrak{Z}_{n+1}$. Let $M \in \mathfrak{Z}_n$ be arbitrary, then we have

$$\begin{aligned} \hat{S}_{n+1}[M] &= \left(\hat{S} \otimes \mathrm{id}_{\mathfrak{A}^{\otimes n}} \circ \hat{S}_n\right)[M] \\ &= \hat{S} \otimes \mathrm{id}_{\mathfrak{A}^{\otimes n}}[\mathbb{1}_{\mathfrak{M}} \otimes A_n] \\ &= \mathbb{1}_{\mathfrak{M}} \otimes \mathbb{1}_{\mathfrak{A}} \otimes A_n \in \mathfrak{Z}_0 \otimes \mathfrak{A}^{\otimes n+1}, \end{aligned}$$

where $A_n \in \mathfrak{A}^{\otimes n}$. Therefore $M \in \mathfrak{Z}_n$ implies $M \in \mathfrak{Z}_{n+1}$ and thus $\mathfrak{Z}_n \subset \mathfrak{Z}_{n+1}$. Assume that one of the inclusions is not strict, i.e. $\mathfrak{Z}_{k-1} = \mathfrak{Z}_k$ for some $k \leqslant \tau$. We can write the action of $\hat{S}_\tau$ on $\mathfrak{M}$ as

$$\hat{S}_\tau[\mathfrak{M}] = \left(\hat{S}_k \otimes \mathrm{id}_{\mathfrak{A}}^{\otimes(\tau-k)}\right)[\hat{S}_{\tau-k}[\mathfrak{M}]] \subset \mathbb{1}_{\mathfrak{M}} \otimes \mathfrak{A}^{\otimes\tau}. \tag{4.18}$$

Now we use the assumption $\mathfrak{Z}_{k-1} = \mathfrak{Z}_k$ and obtain

$$\hat{S}_{\tau-k}[\mathfrak{M}] \subset \mathfrak{Z}_k \otimes \mathfrak{A}^{\otimes(\tau-k)} = \mathfrak{Z}_{k-1} \otimes \mathfrak{A}^{\otimes(\tau-k)}.$$

By definition of $\mathfrak{Z}_{k-1}$ its elements are mapped into $\mathfrak{Z}_0 \otimes \mathfrak{A}^{\otimes(k-1)}$ by $\hat{S}_{k-1}$ and therefore

$$\begin{aligned} \mathbb{1}_{\mathfrak{M}} \otimes \mathfrak{A}^{\otimes(\tau-1)} &\supset \left(\hat{S}_{k-1} \otimes \mathrm{id}_{\mathfrak{A}}^{\otimes(\tau-k)}\right)[\hat{S}_{\tau-k}[\mathfrak{M}]] \\ &= \hat{S}_{\tau-1}[\mathfrak{M}]. \end{aligned}$$

This implies that S is forgetful in $\tau - 1$ steps which is a contradiction to τ minimal. Thus all inclusions have to be strict and

$$\mathbb{C}\mathbb{1}_{\mathfrak{M}} = \mathfrak{Z}_0 \subsetneq \mathfrak{Z}_1 \subsetneq \cdots \subsetneq \mathfrak{Z}_{\tau-1} \subsetneq \mathfrak{Z}_\tau = \mathfrak{M}. \tag{4.19}$$

To determine the maximal depth τ in dependence of the memory algebra $\mathfrak{M}$ we have to study which strict inclusions are possible and what is the maximal number of successive inclusions given the memory algebra $\mathfrak{M}$. The embedding is a unital C^*-homomorphism so Lemma 2.1.2 applies. The chain of successive embeddings starting from $\mathbb{C}\mathbb{1}$ and ending at $\mathfrak{M}$ can be visualized by a finite Bratteli diagram. Searching for the maximal depth is the same as determining the Bratteli diagram of $\mathfrak{M}$ with the maximal number of layers.

First let us consider a single step of the embedding. Given are two finite dimensional C^*-algebras $\mathfrak{A}_n = \bigoplus_{j=1}^{k_n} \mathcal{M}_{d_{n,j}}$ and $\mathfrak{A}_{n+1} = \bigoplus_{i=1}^{k_{n+1}} \mathcal{M}_{d_{n+1,i}}$ with $\mathfrak{A}_n \subset \mathfrak{A}_{n+1}$. Lemma 2.1.2 states that every unital C^*-homomorphism from $\mathfrak{A}_n$ to $\mathfrak{A}_{n+1}$ is given up to unitary equivalence by a matrix A which encodes the embedding of the summands of $\mathfrak{A}_n$ into the summands of $\mathfrak{A}_{n+1}$. The matrix has to fulfill

$$\sum_{j=1}^{k_n} a_{ij} d_{n,j} = d_{n+1,i}. \tag{4.20}$$

This means that the sum of the dimensions of all the summands of $\mathfrak{A}_n$ that are embedded into a specific summand of $\mathfrak{A}_{n+1}$ has to equal the dimension of that summand. On the other hand every summand of $\mathfrak{A}_n$ has to be embedded into $\mathfrak{A}_{n+1}$. this leads to

$$\sum_{i=1}^{k_{n+1}} a_{ij} \geqslant 1. \tag{4.21}$$

Equations (4.20) and (4.21) together tell us that the maximal summand dimension of $\mathfrak{A}_n$ has to be lesser or equal than the maximal summand dimension of $\mathfrak{A}_{n+1}$:

$$\max_j(d_{n,j}) \leqslant \max_i(d_{n+1,i}). \tag{4.22}$$

Furthermore we have the bound

$$\sum_{j=1}^{k_n} d_{n,j} \leqslant \sum_{i=1}^{k_{n+1}} d_{n+1,i}. \tag{4.23}$$

Now let us investigate what implications this has for the strict embedding of $\mathfrak{A}_n$ into $\mathfrak{A}_{n+1}$ ($\mathfrak{A}_n \subsetneq \mathfrak{A}_{n+1}$). In the end, we want to determine the maximal depth τ of a memory channel given its memory algebra $\mathfrak{M}$. This is equivalent to finding the maximal number of steps to embed $\mathbb{C}\mathbb{1}$ into $\mathfrak{M}$ (see also Equation 4.19). To do this we identify a quantity that is non-increasing when moving down in the chain of embeddings. Together with a bound on the number of possible steps in which this quantity stays constant we can determine a bound on the number of total embedding steps. The quantity we use is the sum of all summand dimensions $d_{n,i}$:

$$d(n) = \sum_{i=1}^{k_n} d_{n,i}. \tag{4.24}$$

Furthermore let

$$\Delta d(n) = d(n+1) - d(n) = \sum_{j=1}^{k_n} \left(\left(\sum_{i=1}^{k_{n+1}} a_{ij} \right) - 1 \right) d_{n,j} \tag{4.25}$$

be the increase in d in embedding step n. We divide the possible embedding into three groups, according to the number of summands in the algebras $\mathfrak{A}_n$ and $\mathfrak{A}_{n+1}$.

$k_n = k_{n+1} = k$: Here we distinguish two cases.

- In the first case no summand of $\mathfrak{A}_n$ is embedded into more than one summand of $\mathfrak{A}_{n+1}$ and therefore $\sum_{i=1}^{k} a_{ij} = 1$ for all j. Therefore, only k elements a_{ij} can be one, the rest is zero. Each column of A contains one non-zero element. We also have $\sum_{j=1}^{k} a_{ij} d_{n,j} = d_{n+1,i}$ (4.20) which shows that each row of A has to contain at least one non-zero entry. But this implies that A is a permutation matrix and thus $\delta(\mathfrak{A}_n) = \pi\delta(\mathfrak{A}_{n+1})$ and therefore $\mathfrak{A}_n \cong \mathfrak{A}_{n+1}$, which is excluded because we only allow strict inclusions.[3]
- If however there is a j such that $\sum_{i=1}^{k} a_{ij} > 1$ then $\Delta d(n) > 0$.

[3] By $\delta(\mathfrak{A}_n)$ we denote the vector of dimensions $d_{n,i}$.

$k_n < k_{n+1}$: In this case A has more rows than columns. By (4.20) every row has to contain at least one non-zero entry and at least k_{n+1} of the a_{ij} are non-zero. Therefore, at least one column has to contain more than one non-zero entry and there exists a j such that $\sum_{i=1}^{k_{n+1}} a_{ij} > 1$ and again $\Delta d(n) > 0$. Furthermore, $\Delta d(n) \geqslant k_{n+1} - k_n$ because there are $k_{n+1} - k_n$ non-zero a_{ij} more than columns which each contribute $d_{n,j} \geqslant 1$ to $\Delta d(n)$.

$k_n > k_{n+1}$: Again there are two possibilities:

- If there exists a j such that $\sum_{i=1}^{k_{n+1}} a_{ij} > 1$, then $\Delta d(n) > 0$.
- If there is no such j, $\Delta d(n)$ can be zero. There are more columns than rows, so it is possible to have only one non-zero entry in each column. If additionally all non-zero entries are one we have $\sum_{i=1}^{k_{n+1}} a_{ij} = 1$ and $\Delta d(n) = 0$.

Using these results we can obtain a bound on the number of inclusion steps given two algebras $\mathfrak{A}_\tau$ and $\mathfrak{A}_0 \subsetneq \mathfrak{A}_\tau$. Let us first assume that $\mathfrak{A}_\tau$ is a full matrix algebra, $\mathfrak{A}_\tau = \mathcal{M}_d$, and $\mathfrak{A}_0 = \mathbb{C}\mathbb{1}$. We start the decomposition from $\mathfrak{A}_\tau$. In each step $d(n)$ can stay constant or decrease (with decreasing n). The number of summands k_n can decrease, increase or stay constant. However, it is bounded by $d(n)$ in every step. Furthermore, decreasing k_n by l will decrease $d(n)$ by at least l. k_n is bounded by $d(n) \leqslant d(\tau)$, so there are at most $d(\tau) - 1$ steps that leave $d(n)$ unchanged. Decreasing k_n does not allow for more steps with unchanged $d(n)$, because $d(n)$ that is bounding k_n will decrease with it, not allowing more room for k_n to increase than we had from the beginning. k_n can be increased by 1 by splitting a summand $\mathcal{M}_{d_{n+1,i}}$ into $\mathcal{M}_1 \oplus \mathcal{M}_{d_{n+1,i}-1}$. So the bound of $d(\tau) - 1$ steps can be saturated. Starting from $\mathcal{M}_d$ we get

$$\mathfrak{A}_{\tau - d(\tau)+1} = \bigoplus_{i=1}^{d(\tau)} \mathcal{M}_1. \tag{4.26}$$

The first steps are also depicted in Figure 4.6.

To decrease $d(n)$ by only 1 per step we need two one-dimensional summands in $\mathfrak{A}_{n+1}$ to embed $\mathcal{M}_1$ in $\mathcal{M}_1 \oplus \mathcal{M}_1$. If we start from (4.26) we have optimal conditions and by joining two one-dimensional summand in each step we reach $\mathfrak{A}_0 = \mathcal{M}_1$ in $d(\tau) - 1$ steps. Thus we have a total of $2d(\tau) - 2$ inclusion steps and $\tau = 2d - 2$ for $\mathfrak{A}_\tau = \mathcal{M}_d$. The Bratteli diagram of the inclusion steps is shown Figure 4.7

If $\mathfrak{A}_\tau = \bigoplus_{i=1}^{k_\tau} \mathcal{M}_{d_{\tau,i}}$ is not a full matrix algebra, we can use the same inclusion structure on the summands of $\mathfrak{A}_\tau$. During the embedding of one summand into $\mathcal{M}_1$ the others remain unchanged and after $n = \sum_{i=1}^{k_\tau} 2d_{\tau,i} - 2$ steps we reach $\mathfrak{A}_{\tau-n} = \bigoplus_{i=1}^{k_\tau} \mathcal{M}_1$. Another $k_\tau - 1$ steps of embedding leave us with $\mathfrak{A}_0 = \mathcal{M}_1$. Thus the total number of embedding steps is $\tau = \sum_{i=1}^{k_\tau} (2d_{\tau,i} - 1) - 1$.

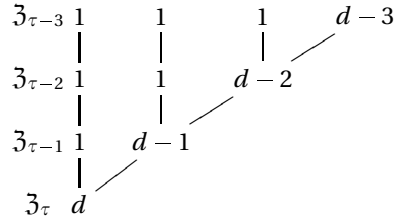


Figure 4.6: First steps of the embedding of $\bigoplus_{i=1}^{d(\tau)} \mathcal{M}_1$ into $\mathcal{M}_d$

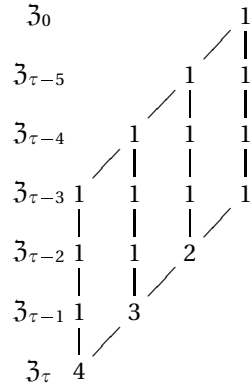


Figure 4.7: Maximal depth embedding of $\mathcal{M}_1$ into $\mathcal{M}_4$

As proven in Lemma 4.1.2 and Corollary 4.1.4 the memory algebra of a strictly forgetful memory channel is always a full matrix algebra $\mathfrak{M} = \mathcal{M}_m$. The maximal depth is

$$\tau = 2(m-1). \tag{4.27}$$

Remark 4.4.1. *However it is an open question if we can construct a memory channel from any given Bratteli diagram. If this is not the case, the bound might actually be stricter.*

The bound proven here only holds in the case of reversible channels. For general memory channels a recent result shows that the maximal memory depth is $\tau = m^2 - 1$ [72].

4.4.1 The Clifford case

In the case of Clifford channels we can use the phase space representation to calculate the decomposition and all the minimal central projections and the according algebras. Here we will only consider the case of systems composed of qubits. Let us first investigate how to determine the decomposition of a subalgebra $\mathfrak{A}(\Xi_i) = \mathfrak{Z}_i$ of $\mathfrak{A}(\Xi) = \mathfrak{M}$ given the phase spaces Ξ and $\Xi_i \subset \Xi$. Let $d_i = \dim \Xi_i$ be the dimension of Ξ_i. To find the decomposition of $\mathfrak{A}(\Xi_i)$ into a direct sum of full matrix algebras we need the minimal central projections of $\mathfrak{A}(\Xi_i)$. They are elements of the center $\mathfrak{C}(\mathfrak{A}(\Xi_i))$ of $\mathfrak{A}(\Xi_i)$ which is the Weyl algebra spanned by the maximally isotropic subspace (the radical) of Ξ_i: $\mathfrak{C}(\mathfrak{A}(\Xi_i)) = \mathfrak{A}(\mathrm{rad}\Xi_i)$.

The algebra $\mathfrak{A}(\Xi_i)$ is isomorphic to the tensor product of the algebra of the radical $\mathrm{rad}\Xi_i = \Xi_i \cap \Xi_i^\sigma$ of Ξ_i and the symplectic subspace associated with Ξ_i.[4] We therefore have

$$\mathfrak{A}(\Xi_i) \cong \mathfrak{A}(\mathrm{rad}\Xi_i) \otimes \mathfrak{A}(\Xi_i/\mathrm{rad}\Xi_i) = \mathfrak{A}(\mathrm{rad}\Xi_i) \otimes \mathfrak{A}(\Xi_i^{\mathrm{red}}). \tag{4.28}$$

The first factor is the center of $\mathfrak{A}(\Xi_i)$ and thus an abelian algebra of dimension 2^{d_c}. The second factor is a full matrix algebra of dimension 2^{d_s}, because $\Xi_i/\mathrm{rad}\Xi_i$ is a symplectic space of symplectic dimension $d_s = (d_i - d_c)/2$ with $d_c = \dim(\mathrm{rad}\Xi_i)$. Therefore, we have the decomposition

$$\mathfrak{A}(\Xi_i) \cong \bigoplus_{j=1}^{2^{d_c}} \mathcal{M}_{2^{d_s}}. \tag{4.29}$$

This gives the Bratteli diagram of Weyl algebras an especially simple structure. Every layer is completely defined by two integers: the dimension of the radical of Ξ_i, d_c, and the dimension of Ξ_i, d_i, because $d_s = (d_i - d_c)/2$. In each step of the embedding (starting from $\mathfrak{M}$) d_i has to decrease by at least one. We are interested in maximal depth embeddings, so we will only consider a decrease by one. In this case d_c can either decrease or increase by one, because $d_i - d_c$ has to either stay constant or decrease by two in each step. The resulting algebras are always of the form of Equation (4.29), consisting of 2^{d_c} summands, each a full matrix algebra of dimension 2^{d_s}. Furthermore, we immediately obtain a bound on the depth of Clifford memory channels; starting from a memory algebra of n qubits the embedding will reach $\mathfrak{Z}_0 = \mathbb{C}\mathbb{1}$ in at most $2n$ steps. This result has already been obtained in Corollary 3.7.5.

To determine the edges of the Bratteli diagram we need the minimal central projections. If two minimal central projections of adjacent layers have an overlap there is an edge between the nodes corresponding to the algebras associated to the projections. We are going to label the minimal central projections by their stabilizer set. Let ξ_i be an orthogonal basis of $\mathrm{rad}\Xi_i$, then the operators $S_i = \mathrm{e}^{-\frac{i}{2}\beta(\xi_i,\xi_i)}\mathrm{w}(\xi_i)$

[4] For the basics on symplectic spaces and Weyl systems see Section 2.2.1.

form a hermitian and unitary basis of $\mathfrak{C}(\mathfrak{A}(\Xi_i))$:

$$S_i^* = \mathrm{e}^{\frac{i}{2}\beta(\xi_i,\xi_i)}\mathrm{e}^{-i\beta(\xi_i,\xi_i)}\mathrm{w}(-\xi_i) = \mathrm{e}^{-\frac{i}{2}\beta(\xi_i,\xi_i)}\mathrm{w}(\xi_i) = S_i$$
$$S_iS_i^* = \mathrm{e}^{-i\beta(\xi_i,\xi_i)}\mathrm{w}(\xi_i)\mathrm{w}(\xi_i) = \mathrm{e}^{-i\beta(\xi_i,\xi_i)}\mathrm{e}^{-i\beta(\xi_i,\xi_i)}\mathrm{w}(0) = \mathbb{1}.$$

Here we used the fact that the underlying field is $\mathbb{Z}_2$. Therefore, the results of this section are only valid for qubits.

Each S_i has an eigenspace $\mathcal{H}_i$ with a basis $\psi_{i,j}$ to eigenvalue 1: $S_i\psi_{i,j} = \psi_{i,j}$. Now let P be a projector. If $PS_i = -P$, then $P\psi_{i,j} = PS_i\psi_{i,j} = -P\psi_{i,j}$ and thus $P\psi_{i,j} = 0$. We are now going to create projectors $P(n)$ with $P(n)S_i = (-1)^{n(\xi_i)}P(n)$. The map $n(\xi)$ is an element of the dual space of Ξ_i and defined by $n(\xi) = n(\sum_i s_i\xi_i) = \sum_i n_i s_i$, where n_i and s_i are elements of $\mathbb{Z}_2$. There are 2^{d_c} different choices for the n_i and therefore 2^{d_c} different projectors for a given isotropic space $\mathrm{rad}\Xi_i$. The multiplication of $P(n)$ with arbitrary Weyl operators $\mathrm{w}(\xi) = \mathrm{w}(\sum_i s_i\xi_i)$ is defined by the multiplication with the stabilizers $S_i = \mathrm{e}^{-\frac{i}{2}\beta(\xi_i,\xi_i)}\mathrm{w}(\xi_i)$:

$$\begin{aligned}
P(n)\mathrm{w}(\xi) &= P(n)\mathrm{w}(\sum_i s_i\xi_i)\\
&= P(n)\prod_i \mathrm{e}^{s_i\frac{i}{2}\beta(\xi_i,\xi_i)}S_i^{s_i}\\
&= \left(\prod_i \mathrm{e}^{s_i\frac{i}{2}\beta(\xi_i,\xi_i)}\right)\prod_i(-1)^{n_is_i\xi_i}P(n)\\
&= \left(\prod_i \mathrm{e}^{s_i\frac{i}{2}\beta(\xi_i,\xi_i)}\right)(-1)^{n(\xi)}P(n).
\end{aligned}$$

Now we are going to build an operator with these properties. As we are in the center of $\mathfrak{A}(\Xi_i)$ all operators commute and we can build up the whole operator by factors $P_i(n)$ for all the basis elements S_i. Each S_i squares to $\mathbb{1}$ as it is unitary and hermitian. So it makes sense to use the ansatz $P_i(n) = \lambda_1(n,i)\mathbb{1} + \lambda_2(n,i)S_i$: we obtain

$$P_i(n)S_i = \lambda_1(n,i)S_i + \lambda_2(n,i)\mathbb{1} \stackrel{!}{=} (-1)^{n_i}\left(\lambda_1(n,i)\mathbb{1} + \lambda_2(n,i)S_i\right).$$

We can choose $\lambda_1(n,i) = 1/2$ and obtain $\lambda_2(n,i) = (-1)^{n_i}1/2$. Thus $P_i(n) = \frac{1}{2}\left(\mathbb{1} + (-1)^{n_i}S_i\right)$ and

$$P(n) = \prod_i \frac{1}{2}\left(\mathbb{1} + (-1)^{n_i}S_i\right). \tag{4.30}$$

As $P_i(n)^2 = P_i(n)$ it is easy to check that $P(n)^2 = P(n)$. It is also easy to see that the projectors are actually minimal: each basis element S_i is projected onto either $P(n)$ or $-P(n)$ by $P(n)$. A general element of $\mathfrak{C}(\mathfrak{A}(\Xi_i)) = \mathfrak{A}(\mathrm{rad}\Xi_i)$ is therefore projected onto a multiple of $P(n)$. Thus every $P(n)$ projects onto a one-dimensional

subspace. Is is important to note that all the projectors $P(n)$ are actually different and orthogonal. The product of $P(n)$ and $P(m)$, $n \neq m$ is

$$\begin{aligned} P(n)P(m) &= \prod_i \frac{1}{2}\left(\mathbb{1} + (-1)^{n_i} S_i\right) \prod_j \frac{1}{2}\left(\mathbb{1} + (-1)^{m_j} S_j\right) \\ &= \prod_i \frac{1}{4}\left(\mathbb{1} + (-1)^{n_i} S_i + (-1)^{m_i} S_i + (-1)^{n_i+m_i}\mathbb{1}\right) \\ &= \prod_{i|n_i=m_i} \frac{1}{2}\left(\mathbb{1} + (-1)^{n_i} S_i\right) \prod_{i|n_i\neq m_i} \frac{1}{4} \underbrace{\left((-1)^{n_i} S_i + (-1)^{n_i+1} S_i\right)}_{=0} \\ &= 0, \end{aligned}$$

because for $n \neq m$ there is at least one i with $n_i \neq m_i$.

To derive the Bratteli diagram of a reversible finite depth Clifford channel or Clifford causal operation we will always use the channel. If we are given a causal operation T we will first determine the channel S that implements T using the algorithm described in Section 4.3.1 and derive the Bratteli diagram from the channel.

We start from the channels phase space matrix $\mathbf{s}$ which acts on the space Ξ. Its restriction to memory outputs $\hat{\mathbf{s}}$ maps from $\Xi_{\mathscr{M}}$ to Ξ with $\hat{\mathbf{s}}\xi_{\mathscr{M}} = \mathbf{s}_{\mathscr{M}}\xi_{\mathscr{M}} \oplus \mathbf{s}_{\mathscr{M}-\mathscr{A}}\xi_{\mathscr{M}}$. It is therefore sufficient to look at the memory to memory transformation $\mathbf{s}_{\mathscr{M}}$ which acts on the memory phase space $\Xi_{\mathscr{M}} = \mathbb{Z}_2^{2m}$. The corresponding algebra is $\mathfrak{M} = \mathfrak{A}(\Xi_{\mathscr{M}})$. The subalgebras $\mathfrak{Z}_n$ are defined via subspaces of the memory phase space: we have $\mathfrak{Z}_n = \mathfrak{A}(\Xi_n)$, where $\Xi_n = \left\{\xi \in \Xi_{\mathscr{M}} | \mathbf{s}^n_{\mathscr{M}}\xi = 0\right\}$. The spaces Ξ_n can be easily constructed from the memory phase space using the transformation $\mathbf{s}_{\mathscr{M}}$. Let $\xi \in \Xi_n$, then

$$\xi \in \Xi_n \Leftrightarrow \mathbf{s}^n_{\mathscr{M}}\xi = 0 \Leftrightarrow \mathbf{s}^{n-1}_{\mathscr{M}}\mathbf{s}_{\mathscr{M}}\xi = 0 \Leftrightarrow \mathbf{s}_{\mathscr{M}}\xi \in \Xi_{n-1}, \tag{4.31}$$

and consequently $\Xi_{n-1} = \mathbf{s}_{\mathscr{M}}\Xi_n$. Given $\Xi_\tau = \Xi_{\mathscr{M}}$ we can compute all Ξ_n via $\Xi_{\tau-n} = \mathbf{s}^n_{\mathscr{M}}\Xi_{\mathscr{M}}$.

Let $\xi_{\mathscr{M},i}$ be a basis of $\Xi_{\mathscr{M}}$, then $\xi_{n,i} = \mathbf{s}^{\tau-n}_{\mathscr{M}}\xi$ is a (overcomplete) basis of Ξ_n. The dimension of d_n Ξ_n equals the rank of the basis $\xi_{n,i}$. If we only want to compute the layers of the Bratteli diagram we only need the dimension $d_{n,c}$ of the radical $\mathrm{rad}\Xi_n$. We can compute it using the commutativity matrix $c_{n,ij} = \sigma(\xi_{n,i}, \xi_{n,j})$.[5] By definition σ vanishes on $\mathrm{rad}\Xi_n$ while on Ξ_n^{red} $c_{n,ij}$ has full rank because Ξ_n^{red} is a symplectic space of dimension $d_{n,s}$. Thus $d_{n,s} = \mathrm{rank}(c_{n,ij})/2$ and $d_{n,c} = d_n - 2d_{n,s} = \mathrm{rank}(\xi_{n,i}) - \mathrm{rank}(c_{n,ij})$. Using (4.29) we obtain

$$\mathfrak{Z}_n \cong \bigoplus_{k=1}^{2^{d_{n,c}}} \mathcal{M}_{2^{d_{n,s}}}. \tag{4.32}$$

[5] For more information on the commutativity matrix see Section 7.2.3.

To determine the full diagram we have to explicitly calculate $\mathrm{rad}\,\Xi_n$ and use (4.30) to calculate the projectors. The edges are determined by the overlap between the projections associated to the nodes.

Remark 4.4.2. *It is also easy to directly derive the diagram from the causal operation. Given the images of the right half chain, we take those that are also supported on the left half chain. This is a finite set, because T has finite depth. The support of those images on the left halfchain forms the memory algebra. To get the subalgebras $\mathfrak{Z}_i$ we take those images, whose support is of length $\leqslant i$ on the left halfchain. Then we can calculate the the Bratteli layers from the decomposition of $\mathfrak{Z}_i$ and the edges from the overlap of the minimal central projections of adjacent layers.*

We conclude this subsection with an illustrative example.

Example 4.4.3. *The Bratteli diagram of the memory algebra can be derived directly from the memory-to-memory transformation of a Clifford memory channel. Let us consider the channel S which, reduced to memory output observables, implements the following transformation $\hat{S}$*

$$\begin{aligned} XII &\mapsto IZZZX \\ ZII &\mapsto XIIXX \\ IXI &\mapsto IXZZZ \\ IZI &\mapsto XXYXI \\ IIX &\mapsto ZIYYI \\ IIZ &\mapsto IXIXX, \end{aligned}$$

where the blue parts of the images are located on the input systems, while the black parts are located on the memory. The blue parts will not be used to derive the Bratteli diagram. The memory-to-memory transformation has the phase space representation

$$\mathbf{s}_{\mathscr{M}} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

The diagram is derived starting from the standard basis of the 3-qubit phase space. It has a trivial symplectic complement, thus there is only one minimal central projection in the algebra and we have a full matrix algebra as expected. For the next step of the diagram we determine the new phase space by acting with $\mathbf{s}_{\mathscr{M}}$ on the basis of the first step. The matrix $\mathbf{s}_{\mathscr{M}}$ does not have full rank, so in the second step the phase space has a non-trivial symplectic complement. We have two minimal central projections

and the algebra $\mathfrak{Z}_5$ decomposes into two summands: $\mathfrak{Z}_5 = \mathcal{M}_4 \oplus \mathcal{M}_4$. For the further steps we obtain the spaces accordingly and arrive at the Bratteli diagram shown in Figure 4.8.

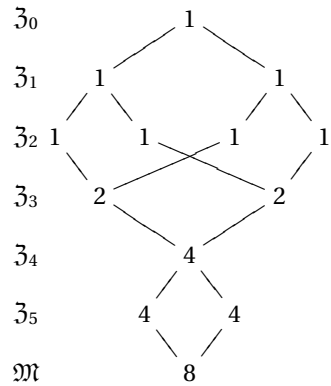


Figure 4.8: Bratteli diagram of the example Clifford memory channel

4.5 Causal inverses

In the following we will determine the conditions for causal reversibility and investigate the memory requirements of channel implementations of causal inverses. The definition of the causal inverse uses the lattice translation τ which only makes sense if the lattice is translation invariant. Therefore, we always assume the system to be translation invariant. However, we make no assumption on the size of the systems $\mathcal{A}_i$, thus they can be composite systems leading to a system that is only translation invariant by shifts by n subsystems. The operation itself does not have to be translation invariant.

Theorem 4.5.1. *Let $T : \mathfrak{A}_{\mathbb{Z}} \to \mathfrak{A}_{\mathbb{Z}}$ be a causal operation and let $\tau \in \mathbb{N}$. Then the following statements are equivalent:*

1. *T is a C*-automorphism with depth τ.*
2. *T is injective and there is a causal operation T' with $TT' = \tau^{\tau}$.*
3. *T is surjective and there is a causal operation T'' with $T''T = \tau^{\tau}$.*

A causal operation T with these properties is called causally reversible. T' and T'' are uniquely determined by T and related by $T''\tau^{\tau} = \tau^{\tau}T'$.

Proof.

1. ⇒ 2. We know that T is an automorphism so it is injective. Furthermore T^{-1} exists and we can define $T' = T^{-1}\tau^\tau$, therefore $TT' = \tau^\tau$. Now we use the causality and finite depth of T to prove causality of T'. The neighborhood of T is $\mathcal{N}(T) = \{(i,j)|i \leqslant j \wedge i \geqslant j - \tau\}$. It follows that

$$\begin{aligned}\mathcal{N}(T') &\subset \mathcal{N}(T^{-1}) \circ \mathcal{N}(\tau^\tau) \\ &= \left\{(i,k)|\exists j, (i,j) \in \mathcal{N}(T)^{-1} \wedge (j,k) \in \mathcal{N}(\tau^\tau)\right\} \\ &= \{(i,k)|\exists j, j \leqslant i \wedge j \geqslant i - \tau \wedge j = k - \tau\} \\ &= \{(i,k)|i \leqslant k \wedge i \geqslant k - \tau\}\end{aligned}$$

and T' is causal.

2. ⇒ 3. Assume T is not surjective. It follows that there exists an element $A \in \mathfrak{A}_\mathbb{Z}$ such that there is no element $B \in \mathfrak{A}_\mathbb{Z}$ with $T[A] = B$. This implies $TT'[\mathfrak{A}_\mathbb{Z}] \subsetneq \mathfrak{A}_\mathbb{Z}$ which is a contradiction to $TT' = \tau^\tau$. Thus T is bijective and therefore a unique two sided inverse T^{-1} exists. Then $T' = T^{-1}\tau^\tau$ and $T'' = \tau^\tau T^{-1}$. We thus have $T''\tau^\tau = \tau^\tau T'$. Using this relation and the causality of T' we now prove causality of T'':

$$\begin{aligned}\mathcal{N}(T'') &\subset \mathcal{N}(\tau^\tau) \circ \mathcal{N}(T') \circ \mathcal{N}(\tau^{-\tau}) \\ &= \{(i,k)|\exists j, (i,j) \in \mathcal{N}(\tau^\tau) \wedge (j,k) \in \mathcal{N}(T')\} \circ \mathcal{N}(\tau^{-\tau}) \\ &= \{(i,k)|i \leqslant k - \tau \wedge i \geqslant k - 2\tau\} \circ \mathcal{N}(\tau^{-\tau}) \\ &= \{(i,l)|\exists l, i \leqslant k - \tau \wedge i \geqslant k - 2\tau \wedge k = l + \tau\} \\ &= \{(i,l)|i \leqslant l \wedge i \geqslant l - \tau\},\end{aligned}$$

and thus T'' causal.

3. ⇒ 1. First we show that T is an automorphism. Assume T is not injective, then there exist elements $A, B \in \mathfrak{A}_\mathbb{Z}$ such that $T[A] = T[B]$. It follows that $\tau^\tau[A] = (T'' \circ T)[A] = (T'' \circ T)[B] = \tau^\tau[B]$. As the lattice translation τ is an automorphism we have $A = B$ and T is injective. Therefore, T is invertible and the only thing left to show is that it preserves the structure, i.e. $T[AB] = T[A]T[B]$. Every completely positive map (on C^* algebras) fulfills $T[X^*X] \geqslant T[X]^*T[X]$ and $T[X^*] = T[X]^*$.[6] We have

$$\begin{aligned}X^*X &= T[T^{-1}[X^*X]] \\ &\geqslant T[T^{-1}[X]^*T^{-1}[X]] \\ &\geqslant T[T^{-1}[X]]^*T[T^{-1}[X]] \\ &= X^*X.\end{aligned}$$

[6] See [25] Propositions 3.3 and 2.12

Both inequalities have to hold tight and thus we obtain

$$T[T^{-1}[X]^* T^{-1}[X]] = T[T^{-1}[X]]^* T[T^{-1}[X]].$$

As T is invertible it is bijective and so is T^{-1}. To each $Y \in \mathfrak{A}_{\mathbb{Z}}$ there is an $X \in \mathfrak{A}_{\mathbb{Z}}$ such that $Y = T^{-1}[X]$ and we have $T[Y^* Y] = T[Y]^* T[Y] \; \forall Y \in \mathfrak{A}_{\mathbb{Z}}$. Now we evaluate this for $Y = A + B$ and $Y = A + iB$ which gives us

$$\begin{aligned} T[A^* B] + T[B^* A] &= T[A]^* T[B] + T[B]^* T[A], \\ T[A^* B] - T[B^* A] &= T[A]^* T[B] - T[B]^* T[A] \end{aligned}$$

and by combination of both

$$T[A^* B] = T[A]^* T[B] \; \forall A, B \in \mathfrak{A}_{\mathbb{Z}}.$$

Thus T is an automorphism.

To prove that T has finite depth τ we use the neighborhood calculus. As T is bijective, a unique two-sided inverse T^{-1} exists and $T^{-1} = \tau^{\tau} T''$. By Lemma 3.1.4 $\mathcal{N}(T^{-1}) = \{(i,j) | \, (i-\tau, j) \in \mathcal{N}(T'')\}$ and

$$\begin{aligned} \mathcal{N}(T) &= \{(i,j) | \, (j-\tau, i) \in \mathcal{N}(T'')\} \\ &\subset \{(i,j) | \, i \geqslant j - \tau\}. \end{aligned}$$

By assumption T is causal and therefore $\mathcal{N}(T) \subset \{(i,j) | \, i \leqslant j \wedge i \geqslant j - \tau\}$. Thus T has finite depth τ. □

The minimal shift τ^{τ} that is achievable is determined by the depth τ of the operation T. Larger shifts would of course be possible, but usually one is interested in the minimal delay between encoding and decoding and therefore the minimal shift.

4.5.1 Construction of the causal inverse

This section will answer two questions: How do we construct a causal inverse of a finite depth causal operation and how can we use the notion of causal operations and causal inverses to construct on-line inverses for memory channels.

The answer to the first question is basically given in Theorem 4.5.1: the right causal inverse T' of a causal process T with depth τ is given by $T^{-1}\tau^{\tau}$. As we want to implement the inversion after the operation we are only interested in the right inverse. However, for the left inverse everything works analogously. This definition of the right causal inverse uses the inverse T^{-1} of T, which we do not know yet how to compute. Fortunately, the decomposition of T as a series of memory channels gives us a direct way to determine T^{-1}. Using the decomposition introduced

in Lemma 4.1.2 and Section 4.3 we derive a memory channel representation S_i of T. We then invert all the unitaries S_i as well as their ordering and construct the inverse T^{-1} using Lemma 4.1.2 and Corollary 4.1.4 (with the difference that T^{-1} will be anti-causal instead of causal and the concatenation of the channels S_i^{-1} will be a memory channel transmitting information backwards in time, an anti-causal memory channel).[7] Now we apply the shift τ^τ and obtain T'.

To determine an on-line inverse of a quantum memory channel, e.g. an on-line decoder for a quantum convolutional code, we use a similar construction. In this case the starting point is a strictly forgetful memory channel S with memory depth τ. We first take the inverse S^{-1} and transform it into an anti-causal operation T^{-1}, which we shift by τ to obtain a causal operation T'. Now we can decompose T' into a memory channel S' which is a causal inverse of S. The memory needed for S' will be discussed in the next section.

4.5.2 Memory requirements of the causal inverses

In this section we investigate bounds on the memory needed for the channel implementation of the (right) inverse causal process T' of a strictly forgetful causal process T given only partial knowledge of T. We will consider two cases: τ is known and only the memory dimension $m(T) = \mathrm{ind}(T)$ is known. As shown in Section 4.1 we can assume $\mathfrak{M} = \mathcal{M}_m$ to be a full matrix algebra.

Lemma 4.5.2. *Let T be a causal process with memory depth τ, then the left inverse T' exists and*

$$m(T') = \frac{d^\tau}{m(T)}, \tag{4.33}$$

where $d = \dim(\mathfrak{A}_i)$ is the cell dimension.

Proof. By Theorem 4.5.1 the left inverse T' exists. By (3.15) and Theorem 4.2.1 the minimal memory dimensions fulfill $m(T)m(T') = d^\tau$. The result follows immediately. □

If only the memory dimension of the causal process T is known, we have to distinguish two cases; the corresponding automorphism T contains a global shift or it is free of global shifts, as introduced in Definition 3.3.4.[8]

If T is allowed to contain shifts, there is no lower bound on the memory needed by the inverse T'. T could be a shift itself leading to $T' = \mathrm{id}$ which needs memory

[7] The decomposition of anti-causal operations into an anti-causal memory channel works exactly as in the causal case, just with a reversed time direction. The same is true for building an anti-causal operation out of an anti-causal memory channel.

[8] If the process is not translation-invariant it can contain shifts locally (See Figure 3.8).

dimension 1 corresponding to no memory at all. However, there is an upper bound on the required memory. Given $m(T)$, τ is bounded by $\tau \leqslant 2(m(T)-1)$ (4.27) and

$$m(T') = \frac{d^{\tau}}{m(T)} \leqslant \frac{d^{2(m(T)-1)}}{m(T)}. \tag{4.34}$$

In Figure 4.9 this is illustrated for qubits.

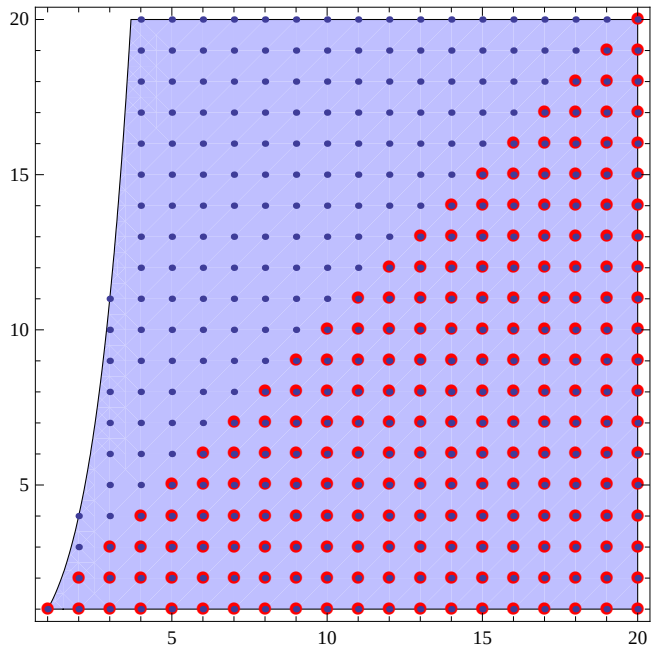


Figure 4.9: Possible combinations of memory requirements for T and T' when T is allowed to contain arbitrary shifts; the cell dimension is 2, the unit is the number of qubits. The light blue area is the area allowed by Equation (4.34). The blue dots are the possible combinations in the general case and the red dots are the possible combinations in the Clifford case

For the case without global shifts we can obtain a lower bound, because we can determine the depth of the causal inverse T'.

Lemma 4.5.3. *Let T be global-shift-free, causal and of memory depth τ. Then T' is causal, global-shift-free and of depth τ.*

Proof. We know $\mathcal{N}(T) \subset \{(i,j)|i \leqslant j \wedge i \geqslant j-\tau\}$ where both inequalities are tight for at least one j each (the index j can be different for the inequalities). From Lemma 3.1.3 we get $\mathcal{N}(T^{-1}) = \{(i,j)|i \leqslant j+\tau \wedge i \geqslant j\}$ where again both inequalities are tight for at least one j each. T^{-1} is anti-causal and global-shift-free. From Theorem 4.5.1 we know that $T' = T^{-1}\tau^{\tau}$ and

$$\begin{aligned}\mathcal{N}(T') &\subset \mathcal{N}(T^{-1}) \circ \mathcal{N}(\tau^{\tau}) \\ &\subset \{(i,k)|\exists j, i \leqslant j+\tau \wedge i \geqslant j \wedge k = j+\tau\} \\ &= \{i \leqslant k \wedge i \geqslant k-\tau\}.\end{aligned}$$

Again, the inequalities are tight for at least one k each and thus T' is causal, global-shift-free, and of depth τ. □

So the memory depth of T' is the same as the depth of T and we can formulate a bound analogous to (4.34) and get

$$m(T) \leqslant \frac{d^{2(m(T')-1)}}{m(T')} \tag{4.35}$$

which gives us an implicit lower bound on $m(T')$ given $m(T)$. The result is shown in Figure 4.10 for qubits.

If we know the shift contained in T, we can also derive upper and lower bounds. The decomposition of T into $T = \tilde{T}\tau^{\lambda}$ (Lemma 3.3.5) together with (4.33) gives us $\tau \leqslant 2(m(T)/d^{\lambda}-1)+\lambda$ and thus

$$m(T') \leqslant \frac{1}{m(T)} d^{2(\frac{m(T)}{d^{\lambda}}-1)+\lambda}, \tag{4.36}$$

which contains (4.34) as a special case. To obtain a lower bound we consider the neighborhood of T' and obtain $T' = T^{-1}\tau^{\tau} = \tau^{-\lambda}\hat{T}^{-1}\tau^{\tau}$, where we used Theorem 4.5.1 and again Lemma 3.3.5. The neighborhood is given via

$$\begin{aligned}\mathcal{N}(T') &\subset \mathcal{N}(\tau^{-\lambda}) \circ \mathcal{N}(\hat{T})^{-1} \circ \mathcal{N}(\tau^{\tau}) \\ &= \{(i,k)|\exists j, j = i-\lambda \wedge k \leqslant j \wedge k \geqslant j-\tau+\lambda\} \circ \mathcal{N}(\tau^{\tau}) \\ &= \{(i,k)|k \leqslant x-\lambda \wedge k \geqslant j-\tau\} \circ \mathcal{N}(\tau^{\tau}) \\ &= \{(i,j)|\exists k, k \leqslant i-\lambda \wedge k \geqslant i-\tau \wedge k = j-\tau\} \\ &= \{(i,j)|i \leqslant j-\lambda \wedge i \geqslant j-\tau+\lambda\}.\end{aligned}$$

T' is causal, global-shift-free, and of depth $\tau' = \tau - \lambda$. Using 4.33 we get the implicit lower bound

$$m(T)d^{-\lambda} \leqslant \frac{d^{2(m(T')-1)}}{m(T')}. \tag{4.37}$$

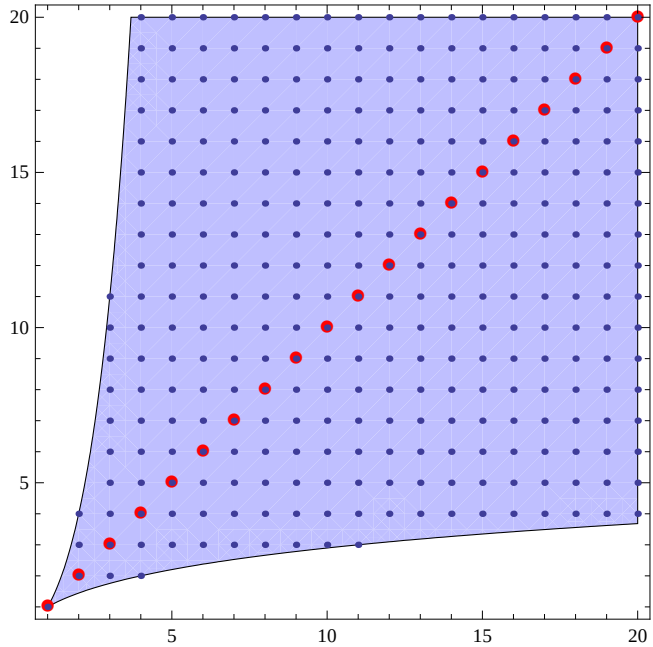


Figure 4.10: Possible combinations of memory requirements for T and T' when T is global-shift-free; the cell dimension is 2, the unit is the number of qubits. The light blue area is the area allowed by Equations (4.34) and (4.35). The blue dots are the possible combinations in the general case and the red dots are the possible combinations in the Clifford case

Now let us assume that $\mathfrak{A}_i = (\mathcal{M}_p)^{\otimes k}$, where p is a prime; the systems each consist of k qudits of dimension p. It follows that $d = p^k$, $m(T) = \mathrm{ind}(T) = p^t$, and $m(T') = \mathrm{ind}(T') = p^r$ (see Lemma 4.1.2). In this case the upper bound is

$$r \leqslant 2k(p^{(t-k\lambda)} - 1) + k\lambda - t \tag{4.38}$$

which is derived taking the logarithm to basis p on both sides of the general bound. It is easy to see, that the upper bound is shifted to higher values of $m(T)$ by λ, where k acts as a factor in this shift. The implicit lower bound is

$$t \leqslant 2k(p^r - 1) + k\lambda - r. \tag{4.39}$$

Again λ introduces a shift on the bound towards higher values of $m(T)$. A shift in T does not change the shape of the region of allowed memory combinations but

shifts it along the $m(T)$ axis. The superposition of all the shifted regions gives the region for unknown shift shown in Figure 4.9.

In the above analysis we used a very general bound on the memory depth τ. For special classes of channels this bound is stricter. For a reversible Clifford channel T with n of qudits memory the memory depth is $\tau \leqslant 2\log_p(m(T)) = 2n$, where p is the dimension of the qudits (see Corollary 3.7.5). Analogous to the general case we use the index formula to determine the memory depth when the shift is known and obtain $\tau \leqslant 2\log_p(m(T)/p^{k\lambda}) + \lambda$. Using the notation of the prime-power case the upper bound is

$$r \leqslant (2k-1)(t-k\lambda). \tag{4.40}$$

Similarly we derive the lower bound

$$r \geqslant \frac{t-k\lambda}{2k-1}. \tag{4.41}$$

The region of possible memory combinations for Clifford channels is smaller than the general region. Its boundaries are given by linear relations in contrast to the exponential relations of the general region. An especially interesting case is $m(T)$ is prime and $\lambda = 0$. Then $m(T') = m(T)$; the inverse always needs the same amount of memory as the channel. In Figure 4.11 the dependence of the possible combinations on k is shown.

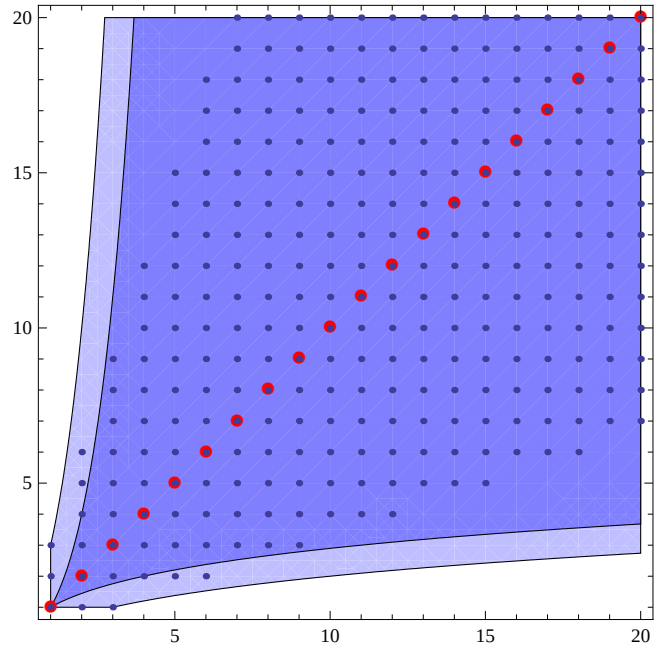


Figure 4.11: Possible combinations of memory requirements for T and T' for 1 and 2 qubits of memory. The larger areas are for 2 qubits of memory. The filled areas are for the general case, the dots are for the Clifford case, where the blue dots are for two qubits of memory and the red dots for one qubit.

5 Clifford quantum cellular automata

Parts of the material in this chapter have been published in:
Johannes Gütschow, Sonja Uphoff, Reinhard F. Werner, and Zoltán Zimborás
Time asymptotics and entanglement generation of Clifford quantum cellular automata
Journal of Mathematical Physics, 51(1), 2010, [arXiv:0906.3195]
Johannes Gütschow
Entanglement generation of Clifford quantum cellular automata
Applied Physics B, 98(4), 2009, [arXiv:1001.1062]

In this chapter we will study Clifford Quantum Cellular Automata (CQCAs) and their time evolution and entanglement generation properties. CQCAs are a special class of Quantum Cellular Automata first described in [33]. As the name indicates, they are QCAs that use the Clifford group operations. They can be defined for arbitrary lattice dimensions and prime cell dimensions. CQCAs are defined as follows:

Definition 5.0.4. *A Clifford Quantum Cellular Automaton T is an automorphism of the quasi-local observable algebra of the infinite spin chain that maps tensor products of Pauli matrices to multiples of tensor products of Pauli matrices and commutes with the lattice translation τ.*

This chapter is organized as follows: First, we will briefly introduce a classical description of CQCAs. In the following part, Section 5.1, we will introduce the three classes of CQCAs: periodic CQCAs, CQCAs with gliders and fractal CQCAs. the latter class produced spacetime images with a fractal structure and will be further investigated in Chapter 6. The focus will be on the glider class, where we will prove equivalence relations for glider CQCAs. In Section 5.2 we will deal with states which are invariant under CQCA action and consider the convergence of states to invariant states. Using these results we will study the entanglement generation of CQCAs starting from stabilizer states and quasi-free states. In Section 5.5 we will give a short review of applications of Clifford quantum cellular automata in quantum computational schemes. In the final Section 5.7 we will show how CQCAs can be implemented by quantum circuits and quantum memory channels.

Our goal is to find a classical description of the CQCAs. It is well known that Clifford operations can be simulated efficiently by a classical computer. Therefore, it

is not surprising that an efficient classical description of CQCAs exists. This description was introduced in [33]. We will only give a short overview of the topic, for proofs and details we refer to the existing literature.

Let us start with a short outline of the construction we use. Weyl operators (tensor products of (generalized) Pauli matrices) are a basis of the (quasi-local) observable algebra. Every CQCA T maps Weyl operators to multiples of Weyl operators. The factor can only be a complex phase, that can be fixed uniquely by the phase of single cell Weyl operators. We can thus describe the action of the CQCA T on Pauli matrices by a classical cellular automaton $\boldsymbol{t}$ acting on their labels "1,2,3" for $\sigma_1, \sigma_2, \sigma_3$. We could keep track of the phase separately, but for our analysis this is unnecessary. As introduced in Section 2.2 we use Weyl systems to map (generalized) Pauli matrices and products thereof to a classical space called phase space. Before we continue with the mathematical definition, we want to illustrate the classical description by a simple example:

Example 5.0.5. *We define our example CQCA on the algebra $\mathfrak{A}(\mathbb{Z})$ of observables on a spin chain[1] by the rule*

$$T_i : \mathfrak{A}_i \to \mathfrak{A}_{\mathcal{N}+i},$$

$$\begin{aligned} T[\sigma_1^i] &= T_0[\sigma_1^i] = \sigma_3^i, \\ T[\sigma_3^i] &= T_0[\sigma_3^i] = \sigma_3^{i-1} \otimes \sigma_1^i \otimes \sigma_3^{i+1}. \end{aligned}$$

The image of σ_2 follows from the product of the images of σ_1 and σ_3, since we require T_0 to be a homomorphism (see Section 3.2):

$$T[\sigma_2^i] = T_0[\sigma_2^i] = -\sigma_3^{i-1} \otimes \sigma_2^i \otimes \sigma_3^{i+1}.$$

The global transformation T has to be an automorphism to be a CQCA. As we already defined the image of σ_2 to be the product of σ_1 and σ_3 we only have to check if the commutation relations are preserved to verify that T is an automorphism:

$$\begin{aligned} [T[\sigma_1^i], T[\sigma_1^j]] &= [\sigma_3^i, \sigma_3^j] = 0, \\ [T[\sigma_3^i], T[\sigma_3^j]] &= [\sigma_3^{i-1} \otimes \sigma_1^i \otimes \sigma_3^{i+1}, \sigma_3^{j-1} \otimes \sigma_1^j \otimes \sigma_3^{j+1}] = 0, \end{aligned}$$

and

$$\begin{aligned} [T[\sigma_3^i], T[\sigma_1^j]] &= [\sigma_3^{i-1} \otimes \sigma_1^i \otimes \sigma_3^{i+1}, \sigma_3^j] = 0 \quad i \neq j, \\ \{T[\sigma_3^i], T[\sigma_1^j]\} &= \{\sigma_3^{i-1} \otimes \sigma_1^i \otimes \sigma_3^{i+1}, \sigma_3^j\} = 0 \quad i = j\,. \end{aligned}$$

The translates of Pauli matrices and their tensor products form a basis of the observable algebra so we can extend T to the whole algebra using its definition on

[1] The algebra is the UHF (Uniformly HyperFinite) algebra $\mathfrak{A}(\mathbb{Z}) := \bigotimes_{k=-\infty}^{+\infty} \mathcal{M}_2$.

single site Pauli matrices. This automaton will be used extensively in the following parts of the section, so we give it the name G_s*. By neglecting a global phase, we can also think of the CQCA* T_G *as a classical automaton acting on the labels* $(1 \mathrel{\widehat{=}} \sigma_1, 2 \mathrel{\widehat{=}} \sigma_2, 3 \mathrel{\widehat{=}} \sigma_3, 0 \mathrel{\widehat{=}} \sigma_0 = \mathbb{1})$ *of the Pauli matrices. We define the operation* $\odot$ *to have the following properties to resemble to the multiplication of Pauli matrices:* $i \odot i = 0$, $i \odot j = k$ *for* $i, j, k = 1,2,3$ *with* $i \neq j \neq k$ *and* $0 \odot i = i \odot 0 = i$ *for* $i = 0, \ldots, 3$.

Now we can illustrate the evolution of the observable $\sigma_3^{-1} \otimes \sigma_2^0 \otimes \sigma_1^1$ *as follows (the underlined labels are situated at the origin):*

$$\mathbf{g}_s(3\underline{2}1) = \mathbf{g}_s(3\underline{0}0) \cdot \mathbf{g}_s(0\underline{2}0) \cdot \mathbf{g}_s(0\underline{0}1) = \begin{array}{cccccc} & 3 & 1 & \underline{3} & & \\ \odot & & 3 & \underline{-2} & 3 & \\ \odot & & & & 3 & \\ \hline = & 3 & -i2 & \underline{i1} & 0 & \end{array} = (32\underline{1}).$$

We notice that the observable moves on the lattice under the action of the CQCA G_s *but is not changed otherwise. We call observables with this property* gliders*. Their existence can be observed easily, when we consider the space-time images of one-cell observables. Single site observables* σ_1 *and* σ_3 *generate "checkerboards" of* σ_1 *and* σ_3 *matrices. As* σ_1 *is mapped to* σ_3 *in the first step, the* σ_1*-checkerboard is the same as the* σ_3*-checkerboard shifted one step in time. If we additionally shift one of them in the space direction by one cell, the checkerboards are exactly the same up to two diagonals and thus cancel out as shown in Figure 5.1. We thus produced a very simple observable* $\sigma_1 \otimes \sigma_3$ *on which the automaton acts as a translation—a basic glider.*

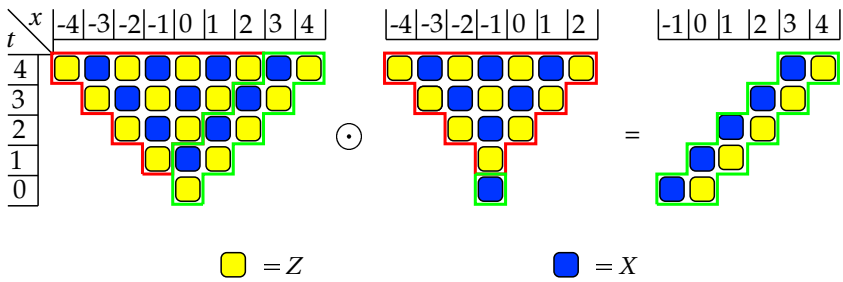


Figure 5.1: Glider of the example CQCA (5.1); the blue boxes represent σ_1, the yellow ones σ_3.

Another interesting property, as shown in [33], of this automaton is the fact that it maps the "all spins up" product state to a cluster-state, which is a one-dimensional version of the two-dimensional resource-state for the "One Way Quantum Computing" scheme by Raussendorf and Briegel [2]. It is also the basic ingredient of a scheme

of "quantum computation via translation-invariant operations on a chain of qubits" by Raussendorf [73]. In a similar way, the update rule G_s (but with σ_1 and σ_3 exchanged) has appeared as the time-evolution of spin chains implemented by a Hamiltonian that is subjected to periodic quenches [74, 75]. It has even been realized experimentally in an NMR-System [76]. More information on the quantum computing schemes that use CQCAs can be found in Section 5.4.

Now we introduce the phase space description for general CQCAs. It is based on the theory of Weyl systems which relate generalized Pauli matrices to a classical vector space we call phase space. This is introduced in Section 2.2. Here we will only give a short example to show how tensor products of Pauli matrices are described by phase space vectors ξ using a mapping $\mathrm{w}(\xi)$:

$$\cdots \mathbb{1} \otimes \sigma_1^{-1} \otimes \sigma_2^0 \otimes \sigma_3^1 \otimes \sigma_1^2 \otimes \mathbb{1} \cdots = \mathrm{w}\begin{pmatrix} \cdots & 0 & 1 & \underline{1} & 0 & 1 & 0 & \cdots \\ \cdots & 0 & 0 & \underline{1} & 1 & 0 & 0 & \cdots \end{pmatrix}.$$

The automaton G_s can now be described by a a classical CA $\mathbf{g}_s$ which can be represented by a 2×2-matrix with polynomial entries which we will also denote by $\mathbf{g}_s$. We first illustrate this with our example CQCA G_s.

In phase space our CQCA-rule reads

$$\begin{aligned} \mathbf{g}_s \begin{pmatrix} 1 \\ 0 \end{pmatrix} &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \\ \mathbf{g}_s \begin{pmatrix} 0 \\ 1 \end{pmatrix} &= \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}, \\ \mathbf{g}_s \begin{pmatrix} 1 \\ 1 \end{pmatrix} &= \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}. \end{aligned}$$

Now we transform the binary strings to Laurent polynomials by indicating the position by multiplication with a variable u and adding all terms from the different positions to a Laurent polynomial:

$$\begin{aligned} \mathbf{g}_s \begin{pmatrix} 1 \\ 0 \end{pmatrix} &= \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \\ \mathbf{g}_s \begin{pmatrix} 0 \\ 1 \end{pmatrix} &= \begin{pmatrix} 1 \\ u^{-1} + u \end{pmatrix}, \\ \mathbf{g}_s \begin{pmatrix} 1 \\ 1 \end{pmatrix} &= \begin{pmatrix} 1 \\ u^{-1} + 1 + u \end{pmatrix}. \end{aligned}$$

We arrange the images of $\binom{1}{0}$ and $\binom{0}{1}$ in a 2×2-matrix

$$\mathbf{g}_s = \begin{pmatrix} 0 & 1 \\ 1 & u^{-1} + u \end{pmatrix}. \tag{5.1}$$

The image under G_s *of an arbitrary tensor product of Pauli matrices is now determined by the multiplication of the corresponding vector of polynomials by the matrix representation* $\mathbf{g}_s$ *of* G_s*. We will later argue that this works for all CQCA.*

Now we come back to the mathematical definition of CQCAs: as a reminder we will briefly repeat the properties of Weyl systems introduced in Section 2.2. The Weyl operators fulfill the relation

$$\mathrm{w}(\xi+\eta) = e^{\frac{2\pi i}{p}(\eta_X \xi_Z)} \mathrm{w}(\xi)\mathrm{w}(\eta)$$

and therefore the commutation relation

$$\mathrm{w}(\xi)\mathrm{w}(\eta) = e^{\frac{2\pi i}{p}(\xi_X \eta_Z - \xi_Z \eta_X)} \mathrm{w}(\eta)\mathrm{w}(\xi).$$

In the following we will use $\varepsilon_p = \exp(2\pi i/p)$ to simplify the notation. In both cases terms of the type $\xi_X \eta_Z$ are scalar products where the addition is carried out modulo 2. The arguments of the Weyl operators are elements of a vector space over the finite field $\mathbb{Z}_p$, the phase space, and thus commute. Of course the corresponding Weyl operators do not necessarily commute. Their commutation relations are encoded in the symplectic form $\sigma(\xi,\eta) = \beta(\xi,\eta) - \beta(\eta,\xi) = \xi_X \eta_Z - \xi_Z \eta_X \in \mathbb{Z}_p$.

To further simplify our classical description, we transform the binary strings to Laurent polynomials as introduced in Section 2.2.6. Our example observable then looks like

$$\begin{pmatrix} \cdots & 0 & 1 & \underline{1} & 0 & 1 & 0 & \cdots \\ \cdots & 0 & 0 & \underline{1} & 1 & 0 & 0 & \cdots \end{pmatrix} \mapsto \begin{pmatrix} u^{-1}+1+u^2 \\ 1+u \end{pmatrix}.$$

The function w now maps vectors of Laurent polynomials to tensor products of Pauli matrices.

Now we describe CQCAs in this picture. A CQCA T maps every Pauli matrix to a tensor product of those. Because it is an automorphism and determined by the local transformation T_0 (see Section 3.2) we can calculate the image of any observable from the image of the Pauli matrices it can be decomposed into. Of course in general the images of neighboring Pauli matrices overlap and have to be multiplied. Because we required them to commute this multiplication is uniquely defined. In the phase space picture with binary strings this multiplication becomes an addition. The CQCAs action is thus a convolution[2] $\star$ of the observables vector of binary strings $(\xi_X^t, \xi_Z^t)^{\mathrm{T}}$ with the automaton's single cell images $(\boldsymbol{t}_{X\to X}, \boldsymbol{t}_{X\to Z})^{\mathrm{T}}$ and $(\boldsymbol{t}_{Z\to X}, \boldsymbol{t}_{Z\to Z})^{\mathrm{T}}$:

$$\begin{aligned} \begin{pmatrix} \xi_X^{t+1}(x) \\ \xi_Z^{t+1}(x) \end{pmatrix} &= \left(\begin{pmatrix} \boldsymbol{t}_{X\to X} & \boldsymbol{t}_{Z\to X} \\ \boldsymbol{t}_{X\to Z} & \boldsymbol{t}_{Z\to Z} \end{pmatrix} \bigstar \begin{pmatrix} \xi_X^t \\ \xi_Z^t \end{pmatrix} \right)(x) \\ &= \begin{pmatrix} (\boldsymbol{t}_{X\to X} \star \xi_X^t)(x) + (\boldsymbol{t}_{Z\to X} \star \xi_Z^t)(x) \\ (\boldsymbol{t}_{X\to Z} \star \xi_X^t)(x) + (\boldsymbol{t}_{Z\to Z} \star \xi_Z^t)(x) \end{pmatrix}. \end{aligned}$$

[2] $\xi \star \eta = \sum_y \xi(-y)\tau^y \eta$

The transformation (2.56) to the Laurent polynomials has the nice property of turning convolutions into multiplications:

$$\begin{aligned}
\widehat{\xi \star \eta} &= \sum_y \widehat{\xi(-y)\tau^y \eta} \\
&= \sum_x \sum_y \xi(-y)\eta(x+y)u^x \\
&= \sum_{k=x+y} \sum_{l=-y} \xi(l)\eta(k)u^{k+l} \\
&= \sum_l \xi(l)u^l \sum_k \eta(k)u^k \\
&= \hat{\xi} \cdot \hat{\eta}.
\end{aligned}$$

Because of this property we refer to (2.56) as an algebraic Fourier transform. We obtain

$$\begin{aligned}
\begin{pmatrix} \hat{\xi}_X^{t+1}(u) \\ \hat{\xi}_Z^{t+1}(u) \end{pmatrix} &= \begin{pmatrix} \hat{\boldsymbol{t}}_{X\to X}(u) & \hat{\boldsymbol{t}}_{Z\to X}(u) \\ \hat{\boldsymbol{t}}_{X\to Z}(u) & \hat{\boldsymbol{t}}_{Z\to Z}(u) \end{pmatrix} \cdot \begin{pmatrix} \hat{\xi}_X^t(u) \\ \hat{\xi}_Z^t(u) \end{pmatrix} \\
&= \begin{pmatrix} \hat{\boldsymbol{t}}_{X\to X}(u) \cdot \hat{\xi}_X^t(u) + \hat{\boldsymbol{t}}_{Z\to X}(u) \cdot \hat{\xi}_Z^t(u) \\ \hat{\boldsymbol{t}}_{X\to Z}(u) \cdot \hat{\xi}_X^t(u) + \hat{\boldsymbol{t}}_{Z\to Z}(u) \cdot \hat{\xi}_Z^t(u) \end{pmatrix}.
\end{aligned}$$

In the following we will omit the hat "$\hat{}$" and the variable u to simplify notation. We then have

$$\xi^{t+1} = \boldsymbol{t}\xi^t = \begin{pmatrix} \boldsymbol{t}_{11} & \boldsymbol{t}_{12} \\ \boldsymbol{t}_{21} & \boldsymbol{t}_{22} \end{pmatrix} \begin{pmatrix} \xi_X^t \\ \xi_Z^t \end{pmatrix}. \tag{5.2}$$

As an automorphism the CQCA leaves the commutation relations invariant. A representation of the CQCA on phase space therefore has to leave the symplectic form σ invariant.[3] Such a translation-invariant symplectic map is called *symplectic cellular automaton* (SCA). We can find a SCA and an appropriate phase function $\lambda(\xi)$ for every CQCA.

Proposition 5.0.6 ([33])**.** *Let T be a CQCA. Then we can write*

$$T[\mathrm{w}(\xi)] = \lambda(\xi)\mathrm{w}(\mathbf{t}\xi) \tag{5.3}$$

with a symplectic cellular automaton $\mathbf{t}$ *and a translation invariant phase function* $\lambda(\xi)$ *that fulfills*

$$\lambda(\xi+\eta) = \lambda(\xi)\lambda(\eta)\boldsymbol{\varepsilon}_d^{\beta(\xi,\eta)-\beta(\mathbf{t}\xi,\mathbf{t}\eta)}$$

as well as $|\lambda(\xi)| = 1\ \forall \xi$*. Furthermore,* $\lambda(\xi)$ *is uniquely determined for all* ξ *by the choice of* $\lambda(\xi)$ *on one site.*

[3] See Section 2.2 for a definition of σ.

In the following analysis of CQCAs we neglect the global phase and consider the symplectic cellular automata only. As we can always find appropriate phase functions all results for SCAs translate to the world of CQCAs directly. We have already seen above that there exists a very convenient representation of CQCAs as 2×2-matrices with polynomial entries. After one further definition we will cite the according theorem.

Definition 5.0.7. *$\mathcal{P}$ is the ring of Laurent polynomials over $\mathbb{Z}_p$. $\mathcal{R}$ is the subring of $\mathcal{P}$, which consists of all polynomials, which are reflection invariant with center $u = 0$.*

Theorem 5.0.8 ([33])**.** *Every CQCA T is represented up to a phase by a unique 2×2-matrix $\mathbf{t}$ with entries from $\mathcal{P}$. Such a matrix represents a CQCA if and only if*

- $\det(\mathbf{t}) = u^{2a}$, $a \in \mathbb{Z}$;
- *all entries are symmetric polynomials centered around the same (but arbitrary) lattice point a;*
- *the entries of both column vectors, which are the pictures of $(1,0)$ and $(0,1)$, are coprime.*[4]

For proofs of Proposition 5.0.6 and Theorem 5.0.8 we refer to [33].

We can further simplify these statements by only considering automata centered around 0. The lattice translation τ is a SCA that by definition commutes with all other SCAs. Its matrix is

$$\tau = \begin{pmatrix} u^{-1} & 0 \\ 0 & u^{-1} \end{pmatrix}. \tag{5.4}$$

The determinant is given by $\det(\tau) = u^{-2}$. Therefore, every SCA with determinant u^{-2a} can be written as the product of the ath power of the lattice translation and an automaton centered around 0 which has determinant one.[5] We call these automata *centered symplectic cellular automata* (CSCAs) and in the following sections we point our focus to those. Centered automata are described by 2×2 matrices over $\mathcal{R}$.

CSCAs and CQCAs each form a group. This group is generated by a countably infinite set of basis automata. The CSCA form the group $\Gamma = \mathrm{SL}(2, \mathcal{R})$, which is the group of all 2×2-matrices with determinant 1 over the ring $\mathcal{R}$ of centered reflection invariant Laurent polynomials with binary coefficients. The group Γ_0 is the group of local automata. Their generators are

$$P = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \quad \text{and} \quad H = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}.$$

[4] Two Laurent polynomials are called coprime if the only common divisors are monomials.

[5] The lattice translation τ shifts states to the right, so observables are shifted to the left and thus we have negative powers of u.

Additionally we have the shear transformations

$$S_n := \begin{pmatrix} 1 & 0 \\ u^n + u^{-n} & 1 \end{pmatrix}, n \in \mathbb{N}$$

which complete the set of generators of Γ. For proofs see [33].

5.1 Classification

In the following we will only consider the evolution of single cell observables and gliders. Any observable can be represented by products and sums of single cell observables, because Weyl operators form a basis of the observable algebra and can be composed of tensor products of single cell Weyl operators themselves. This product structure is invariant under the action of the automaton, because the automaton is an automorphism of the observable algebra. This means that when we discuss the time evolution of a CQCA T, we will consider the action of powers $\boldsymbol{t}^n$ of the matrix of the CSCA on phase space vectors which only contain constants in the polynomials. For example the image of $\sigma_1 = \mathrm{w}(1,0)$ after n time steps of T is given by the first column vector of $\boldsymbol{t}^n$ (and a global phase). The matrix $\boldsymbol{t}$ does not always have an eigenvalue in $\mathcal{R}$, because it is a matrix over a ring without multiplicative inverses for all elements. But for some of the automata eigenvalues do exist. These automata are the glider-automata, because, as seen in Example 5.0.5, on a special set of observables (the gliders) they act as lattice translations. Even the minimal gliders are multi cell observables, so we study them independently of the single cell observables.

We will prove that if the trace is a polynomial consisting of only two summands, i.e. it is of the form $\mathrm{tr}\,\boldsymbol{t} = \pm(u^{-n} + u^n)$, two eigenvectors exist and the automaton has gliders. If this does not hold, we can distinguish two cases. The trace can be either a constant, or an arbitrary symmetric polynomial. In the first case the automaton is periodic, in the second case it generates a time evolution which has fractal properties. In fact, as we will prove in Chapter 6, all CSCAs have a self similar structure. But this structure is trivial for periodic and glider automata, so we only call the remaining CSCAs fractal. In the glider case we will prove equivalence theorems for automata with speed one gliders. We will also give an example to show that this equivalence does not hold for CSCAs with gliders that move more than one step.

5.1.1 Periodic automata

CSCAs with matrices that have a trace independent of u show periodic behavior.

Proposition 5.1.1. *A CSCA* $\mathbf{t}$ *is periodic if and only if* $\mathrm{tr}\,\mathbf{t} = c$ *for* $c \in \mathbb{Z}_p$.

Proof. Let us first assume $\operatorname{tr} \boldsymbol{t} = c$, $c \in \mathbb{Z}$. By the Cayley-Hamilton theorem we get $\boldsymbol{t}^2 = \boldsymbol{t} \cdot \operatorname{tr} \boldsymbol{t} - \mathbb{1} = c\boldsymbol{t} - \mathbb{1}$. Decompositions of higher powers of $\boldsymbol{t}$ can be obtained by multiplying the decomposition of $\boldsymbol{t}^2$ with $\boldsymbol{t}$ on both sides and using the $\boldsymbol{t}^2$ decomposition to get down to $\boldsymbol{t}^n = \tilde{c}\boldsymbol{t} + \hat{c}\mathbb{1}$. The coefficients $\tilde{c}$ and $\hat{c}$ are sums of powers of c and thus elements of the underlying field $\mathbb{Z}_p$. Therefore, there are only finitely many different decompositions $\boldsymbol{t}^n = \tilde{c}\boldsymbol{t} + \hat{c}\mathbb{1}$ and $\boldsymbol{t}$ is periodic.

If on the other hand $\operatorname{tr} \boldsymbol{t} = a$ is a non-constant polynomial, $\boldsymbol{t}$ can not be periodic. $\operatorname{tr} \boldsymbol{t}^n = \tilde{a}a + 2\hat{a}$ and $\tilde{a}$ contains a term a^n. This is the highest order term appearing and thus it can not be canceled out by another term in $\tilde{a}$ or $\hat{a}$. Thus $\operatorname{tr} \boldsymbol{t}^n$ has the highest order term a^{n+1} and as there are no non-monomial elements in $\mathcal{P}$ that have inverses, we have $\operatorname{dg}(\operatorname{tr} \boldsymbol{t}^n) = n\operatorname{dg}(\operatorname{tr} \boldsymbol{t})$. Thus $\operatorname{dg}(\operatorname{tr} \boldsymbol{t}^n) = \operatorname{dg}(\operatorname{tr} \boldsymbol{t})$ if only if $\operatorname{dg}(\operatorname{tr} \boldsymbol{t}) = 0$ and thus $\operatorname{tr} \boldsymbol{t} = c$, $c \in \mathbb{Z}_p$. □

Proposition 5.1.2. *If the underlying field is $\mathbb{Z}_2$, a CSCA* $\mathbf{t}$ *is periodic with period $c+2$ if* $\operatorname{tr} \mathbf{t} = c$ *for $c \in \mathbb{Z}_2$.*

Proof. Again, using the Cayley-Hamilton theorem we get $\boldsymbol{t}^2 = \boldsymbol{t} \cdot \operatorname{tr} \boldsymbol{t} + \mathbb{1} = c\boldsymbol{t} + \mathbb{1}$. Now, if $c = 0$, we have $\boldsymbol{t}^2 = \mathbb{1}$. If $c = 1$, we have $\boldsymbol{t}^3 = \boldsymbol{t}^2 + \boldsymbol{t} = 2\boldsymbol{t} + \mathbb{1} = \mathbb{1}$. □

Proposition 5.1.3. *Let* $\mathbf{t}$ *be a CSCA, $\xi \in \mathcal{P}^2$ a non-zero phase space vector and the underlying field $\mathbb{Z}_2$. If $\mathbf{t}\xi = \xi$ holds, then* $\mathbf{t}$ *is periodic with period two.*

Proof. We use $\boldsymbol{t}^2 = \boldsymbol{t} \cdot \operatorname{tr} \boldsymbol{t} + \mathbb{1}$ and $\boldsymbol{t}\xi = \xi$:

$$\begin{aligned} & \boldsymbol{t}^2\xi &&= \xi \\ \Leftrightarrow \quad & (\boldsymbol{t} \cdot \operatorname{tr} \boldsymbol{t})\xi + \mathbb{1}\xi &&= \xi \\ \Leftrightarrow \quad & \xi(\operatorname{tr} \boldsymbol{t}) &&= 0 \\ \Leftrightarrow \quad & \operatorname{tr} \boldsymbol{t} &&= 0 \end{aligned}$$

Thus $\boldsymbol{t}$ is of period two by Proposition 5.1.2. □

5.1.2 Automata with gliders

Let us first define our notion of a glider.

Definition 5.1.4. *A glider is an observable on which the CQCA T acts as a lattice translation. In the Laurent polynomial picture a translation is a multiplication by u^n, $n \in \mathbb{Z}$. We will refer to an observable, which is mapped to a translated version of itself by T^2 but not by T as a semi-glider. In the Laurent polynomial picture this corresponds to a multiplication by $-u^n$ in every step. The case $n = 0$ is excluded, because then there is no propagation.*

We have already seen this behavior in Example 5.0.5. Now we determine the conditions a CQCA has to fulfill to have gliders. In general, we can not diagonalize the matrices of the corresponding CSCA, because all our calculations are over a finite field and the entries are polynomials in u. The polynomials are usually not invertible, so the equations which occur in the diagonalization cannot be solved mechanically. Furthermore, the diagonal matrix would not correspond to a CSCA as u^n and u^{-n} are not centered. Hence we take a different approach. First, let us introduce some terms: we call a glider $\xi = (\xi_X, \xi_Z)$ a *minimal glider* if and only if its two entries in phase space ξ_X, ξ_Z have no common non-invertible divisor[6]. The *wedge-product* of two phase space vectors shall be defined as $\xi \wedge \eta = \xi_X \eta_Z - \eta_X \xi_Z$. We define the *involution* of a polynomial $p \in \mathcal{P}$ as the substitution of u by u^{-1} and denote it by $\bar{p}$. For $p \in \mathcal{R}$ we have $\bar{p} = p$. Finally we will also need the following proposition:

Proposition 5.1.5. *In the ring $\mathcal{P}$ of Laurent polynomials over the finite field $\mathbb{Z}_p$, the only invertible elements are monomials.*

Proof. The inverse element $\eta = \sum_{j\in\Upsilon} b_j u^j$ of a polynomial $\xi = \sum_{i\in\Lambda} a_i u^i$ has to fulfill

$$\left(\sum_{j\in\Upsilon} b_j u^j\right)\left(\sum_{i\in\Lambda} a_i u^i\right) = \sum_{i,j} a_i b_j u^{i+j} = 1, \tag{5.5}$$

where Λ and Υ are the localization areas of the polynomials. The localization areas are each simply connected and thus also contain sites where the coefficients are zero, if and only if there are sites with nonzero coefficients corresponding to higher and lower powers (both are necessary). Thus the coefficients of the largest ($i_{\max}$, $j_{\max}$) and the lowest ($i_{\min}$, $j_{\min}$) occurring indices are nonzero. In the product all the coefficients for indices $i + j \neq 0$ have to vanish. As the coefficient corresponding to $i_{\min} + j_{\min}$ is nonzero by construction, $i_{\min} + j_{\min} = 0$ holds. The same is true for the maximal indices. Therefore, we have

$$\begin{aligned} i_{\min} &= -j_{\min} \\ i_{\max} &= -j_{\max}. \end{aligned}$$

To fulfill both equations we need $i_{\min} = i_{\max}$ and $j_{\min} = j_{\max}$. Thus ξ and η are monomials. □

Now we have all we need to prove the following theorems:

Proposition 5.1.6. *Given a CSCA* **t**, *a non-zero phase space vector ξ, and a non-identity monomial λ with* $\mathbf{t}\xi = \lambda\xi$, *the following is true:*

[6] The phase space vector of a minimal glider is maximal with respect to the notation introduced in [33].

1. *$\bar{\xi}$ fulfills $\mathbf{t}\bar{\xi} = \bar{\lambda}\bar{\xi}$; thus it is a glider with the same speed but different direction as ξ.*

2. *$\mathbf{t}$ is uniquely given by*

$$\mathbf{t}_{11} = \frac{\lambda\xi_X\bar{\xi}_Z - \bar{\lambda}\bar{\xi}_X\xi_Z}{\xi \wedge \bar{\xi}}, \tag{5.6}$$

$$\mathbf{t}_{12} = \frac{\left(\bar{\lambda} - \lambda\right)\xi_X\bar{\xi}_X}{\xi \wedge \bar{\xi}}, \tag{5.7}$$

$$\mathbf{t}_{21} = \frac{\left(\lambda - \bar{\lambda}\right)\xi_Z\bar{\xi}_Z}{\xi \wedge \bar{\xi}}, \tag{5.8}$$

$$\mathbf{t}_{22} = \frac{\bar{\lambda}\xi_X\bar{\xi}_Z - \lambda\bar{\xi}_X\xi_Z}{\xi \wedge \bar{\xi}}. \tag{5.9}$$

3. *$tr(\mathbf{t}) = \bar{\lambda} + \lambda$*

4. *All gliders are multiples of*

$$\xi = \begin{pmatrix}\xi_X\\ \xi_Z\end{pmatrix} = \begin{pmatrix}\frac{-\mathbf{t}_{12}}{\gcd(\lambda-\mathbf{t}_{11},\mathbf{t}_{12})}\\ \frac{\lambda-\mathbf{t}_{11}}{\gcd(\lambda-\mathbf{t}_{11},\mathbf{t}_{12})}\end{pmatrix}, \tag{5.10}$$

or

$$\bar{\xi} = \begin{pmatrix}\bar{\xi}_X\\ \bar{\xi}_Z\end{pmatrix} = \begin{pmatrix}\frac{-\mathbf{t}_{12}}{\gcd(\lambda-\mathbf{t}_{11},\mathbf{t}_{12})}\\ \frac{\bar{\lambda}-\mathbf{t}_{11}}{\gcd(\lambda-\mathbf{t}_{11},\mathbf{t}_{12})}\end{pmatrix}. \tag{5.11}$$

Proof.

1. We use $\boldsymbol{t}\xi = \lambda\xi$ and take the involution on both sides. $\boldsymbol{t}$ consists of palindromes, thus $\bar{\boldsymbol{t}} = \boldsymbol{t}$ and we have $\boldsymbol{t}\bar{\xi} = \bar{\lambda}\bar{\xi}$.

2. We write $\boldsymbol{t}\xi = \lambda\xi$ and $\boldsymbol{t}\bar{\xi} = \bar{\lambda}\bar{\xi}$ component wise yielding the four equations

$$\begin{array}{lrcl}
(\mathrm{I}) & \boldsymbol{t}_{11}\xi_X + \boldsymbol{t}_{12}\xi_Z &=& \lambda\xi_X,\\
(\mathrm{II}) & \boldsymbol{t}_{21}\xi_X + \boldsymbol{t}_{22}\xi_Z &=& \lambda\xi_Z,\\
(\bar{\mathrm{I}}) & \boldsymbol{t}_{11}\bar{\xi}_X + \boldsymbol{t}_{12}\bar{\xi}_Z &=& \bar{\lambda}\bar{\xi}_X,\\
(\bar{\mathrm{II}}) & \boldsymbol{t}_{21}\bar{\xi}_X + \boldsymbol{t}_{22}\bar{\xi}_Z &=& \bar{\lambda}\bar{\xi}_Z.
\end{array}$$

   Combining them in the right way we have

$$\boldsymbol{t}_{11}\left(\xi \wedge \bar{\xi}\right) = \lambda\xi_X\bar{\xi}_Z - \bar{\lambda}\bar{\xi}_X\xi_Z, \tag{5.12}$$

$$\boldsymbol{t}_{12}\left(\xi \wedge \bar{\xi}\right) = \left(\bar{\lambda} - \lambda\right)\xi_X\bar{\xi}_X, \tag{5.13}$$

$$\boldsymbol{t}_{21}\left(\xi \wedge \bar{\xi}\right) = \left(\lambda - \bar{\lambda}\right)\xi_Z\bar{\xi}_Z, \tag{5.14}$$

$$\boldsymbol{t}_{22}\left(\xi \wedge \bar{\xi}\right) = \bar{\lambda}\xi_X\bar{\xi}_Z - \lambda\bar{\xi}_X\xi_Z. \tag{5.15}$$

By assumption, $\boldsymbol{t}$ exists and is a CSCA, so the division by $\xi \wedge \bar{\xi}$ gives a polynomial result and we have Equations (5.6) to (5.9). In Proposition 5.1.8 we will show which conditions ξ has to fulfill in order for the division to be valid. This in turn guarantees the existence of a CSCA $\boldsymbol{t}_\xi$, with a glider ξ.

3.

$$\begin{aligned} \operatorname{tr}\boldsymbol{t} &= \boldsymbol{t}_{11} + \boldsymbol{t}_{22} = \frac{\lambda\xi_X\bar{\xi}_Z - \bar{\lambda}\bar{\xi}_X\xi_Z + \bar{\lambda}\xi_X\bar{\xi}_Z - \lambda\bar{\xi}_X\xi_Z}{\xi \wedge \bar{\xi}} \\ &= \frac{\left(\lambda + \bar{\lambda}\right)\left(\xi \wedge \bar{\xi}\right)}{\xi \wedge \bar{\xi}} = \lambda + \bar{\lambda} \end{aligned}$$

4. We now use (I) and (II) together with $\det\boldsymbol{t} = 1$ and $\operatorname{tr}\boldsymbol{t} = \lambda + \bar{\lambda}$ to derive the a solution for ξ. We obtain

$$\xi_X\left(\lambda - \boldsymbol{t}_{11}\right) = \xi_Z\boldsymbol{t}_{12}.$$

This equation for ξ_X and ξ_Z has still one free parameter. One particular solution for the equation is $\xi_X^{(part)} = \boldsymbol{t}_{12}$, $\xi_Z^{(part)} = \lambda - \boldsymbol{t}_{11}$. To obtain the minimal glider we have to divide these components of the particular solution by their greatest common divisor, and thus obtain (5.10). For (5.11) we do the same with $(\bar{\mathrm{I}})$ and $(\bar{\mathrm{II}})$. An arbitrary glider can be written as the minimal glider defined by either (5.10) or (5.11) multiplied by a Laurent polynomial in u.

□

Remark 5.1.7. *We could extend our definition of gliders to combinations of CSCAs and observables with polynomial eigenvalues λ. These observables would be mapped to products of translates of themselves. We can show that this extension would not yield any new gliders: a CSCA $\mathbf{t}$ has to fulfill* $\det\mathbf{t} = 1$. *With (I) to $(\bar{I}I)$ we have*

$$\begin{aligned} \det\mathbf{t}\cdot\left(\xi\wedge\bar{\xi}\right)^2 &= \left(\mathbf{t}_{11}\mathbf{t}_{22} + \mathbf{t}_{12}\mathbf{t}_{21}\right)\cdot\left(\xi\wedge\bar{\xi}\right)^2 \\ &= \left(\lambda\xi_X\bar{\xi}_Z + \bar{\lambda}\bar{\xi}_X\xi_Z\right)\cdot\left(\bar{\lambda}\xi_X\bar{\xi}_Z + \lambda\bar{\xi}_X\xi_Z\right) \\ &\quad - \left(\left(\bar{\lambda}+\lambda\right)\xi_X\bar{\xi}_X\right)\cdot\left(\left(\lambda+\bar{\lambda}\right)\xi_Z\bar{\xi}_Z\right) \\ &= \lambda\bar{\lambda}\left(\xi_X^2\bar{\xi}_Z^2 + 2\xi_X\xi_Z\bar{\xi}_X\bar{\xi}_Z + \bar{\xi}_X^2\xi_Z^2\right) \\ &= \lambda\bar{\lambda}\left(\xi\wedge\bar{\xi}\right)^2 \\ \Leftrightarrow\quad \det\mathbf{t} &= \lambda\bar{\lambda} \stackrel{!}{=} 1 \\ \Leftrightarrow\quad \lambda &= \pm u^{\pm n} \text{ or } \lambda \notin \mathcal{P}. \end{aligned}$$

The only possible solutions are $\lambda = u^{\pm n}$, $n \in \mathbb{N}$, because by Proposition 5.1.5 in $\mathcal{P}$ monomial are the only elements with inverses.

Proposition 5.1.8. *A minimal $\xi \in \mathcal{P}^2$ is a glider for a CSCA* $\mathbf{t}$ *with eigenvalue $\pm u^{\pm n}$, $n \in \mathbb{Z}\backslash 0$ if and only if $\xi \wedge \bar{\xi}$ is a divisor of $u^n - u^{-n}$ and $\xi \wedge \bar{\xi} \neq 0$.*

Proof. If ξ is a glider for $\boldsymbol{t}$ with eigenvalue u^n then ξ is a glider for $-\boldsymbol{t}$ with eigenvalue $-u^n$. Furthermore, $\bar{\xi}$ is a glider for $\boldsymbol{t}$ with eigenvalue u^{-n}. Thus it suffices to prove the case u^n as all other cases follow.

First, let us assume that ξ is minimal and that $\xi \wedge \bar{\xi} \neq 0$ is a divisor of $u^n - u^{-n}$. For ξ to be a glider with eigenvalue u^n, $\boldsymbol{t}\xi = u^n\xi$ has to hold. We use Equations (5.12) through (5.15) to construct $\boldsymbol{t}$. We have to show that (a) $\boldsymbol{t}\xi = u^n\xi$, (b) $\det \boldsymbol{t} = 1$, and (c) all the divisions in Equations (5.12) to (5.15) yield elements of $\mathcal{P}$.

Conditions (a) and (b) can be shown by a direct computation. Condition (c) obviously holds for Equations (5.13) and (5.14). For the other two equations we use a simple trick:

$$\begin{aligned} \boldsymbol{t}_{11}\left(\xi \wedge \bar{\xi}\right) &= u^n\xi_X\bar{\xi}_Z - u^{-n}\bar{\xi}_X\xi_Z \\ &= u^n\xi_X\bar{\xi}_Z - u^{-n}\bar{\xi}_X\xi_Z + \underbrace{\left(u^{-n}\xi_X\bar{\xi}_Z - u^{-n}\xi_X\bar{\xi}_Z\right)}_{=0} \\ &= (u^n - u^{-n})\xi_X\bar{\xi}_Z + u^{-n}\left(\xi \wedge \bar{\xi}\right). \end{aligned}$$

It is now apparent that $\xi \wedge \bar{\xi}$ also divides the right hand side of (5.12) if it is a divisor of $u^n - u^{-n}$. For (5.15) an analogous argument holds.

Now let us show the converse. We assume that a CSCA $\boldsymbol{t}$ with $\boldsymbol{t}\xi = u^n\xi$ exists. The matrix $\boldsymbol{t}$ has the form proven in Proposition 5.1.6. The Laurent polynomials form an Euclidean ring, which implies that the extended Euclidean algorithm is applicable [77]. Since ξ is minimal, the greatest common divisor of ξ_X and ξ_Z is 1, and so, according to the extended Euclidean algorithm, we can chose η_X and η_- such that

$$\xi_X\eta_- - \xi_-\eta_+ = \gcd(\xi_X, \xi_Z) = 1.$$

Then we have

$$\begin{aligned} & \left(u^n - u^{-n}\right) \\ = \; & \left(u^n - u^{-n}\right) \cdot \underbrace{\left(\xi_X\eta_Z - \xi_Z\eta_X\right)}_{=1,\,\text{as}\,\xi\,\text{min.}} \cdot \underbrace{\left(\bar{\xi}_X\bar{\eta}_Z - \bar{\xi}_Z\bar{\eta}_X\right)}_{=1,\,\text{as}\,\bar{\xi}\,\text{min.}} \\ = \; & \left(u^n - u^{-n}\right) \cdot \left(\xi_X\bar{\xi}_X\eta_Z\bar{\eta}_Z - \xi_Z\bar{\xi}_X\eta_X\bar{\eta}_Z - \xi_X\bar{\xi}_Z\eta_Z\bar{\eta}_X + \xi_Z\bar{\xi}_Z\eta_X\bar{\eta}_X\right) \\ = \; & \left(u^n - u^{-n}\right)\xi_X\bar{\xi}_X\eta_Z\bar{\eta}_Z + \left(u^n - u^{-n}\right) \cdot \xi_Z\bar{\xi}_Z\eta_X\bar{\eta}_X \\ & -u^{-n}\xi_Z\bar{\xi}_X\eta_X\bar{\eta}_Z + u^{-n}\xi_Z\bar{\xi}_X\eta_X\bar{\eta}_Z \underbrace{+u^{-n}\xi_X\bar{\xi}_Z\eta_X\bar{\eta}_Z - u^{-n}\xi_X\bar{\xi}_Z\eta_X\bar{\eta}_Z}_{=0} \\ & +u^{-n}\xi_X\bar{\xi}_Z\eta_Z\bar{\eta}_X - u^n\xi_X\bar{\xi}_Z\eta_Z\bar{\eta}_X \underbrace{+u^{-n}\bar{\xi}_X\xi_Z\eta_Z\bar{\eta}_X - u^{-n}\bar{\xi}_X\xi_Z\eta_Z\bar{\eta}_X}_{=0} \end{aligned}$$

$$
\begin{aligned}
= \quad & \underbrace{\left(u^n - u^{-n}\right)\xi_X\bar{\xi}_X}_{=-\boldsymbol{t}_{12}\left(\xi\wedge\bar{\xi}\right)}\eta_Z\bar{\eta}_Z + \underbrace{\left(u^n - u^{-n}\right)\xi_Z\bar{\xi}_Z}_{=\boldsymbol{t}_{21}\left(\xi\wedge\bar{\xi}\right)}\eta_X\bar{\eta}_X \\
& + \underbrace{\left(u^{-n}\xi_X\bar{\xi}_Z - u^n\xi_Z\bar{\xi}_X\right)}_{=\boldsymbol{t}_{22}\left(\xi\wedge\bar{\xi}\right)}\eta_X\bar{\eta}_Z + \underbrace{\left(u^{-n}\xi_Z\bar{\xi}_X - u^n\xi_X\bar{\xi}_Z\right)}_{=\boldsymbol{t}_{11}\left(\xi\wedge\bar{\xi}\right)}\eta_Z\bar{\eta}_X \\
& + u^{-n}\eta_X\bar{\eta}_Z\left(\xi\wedge\bar{\xi}\right) + u^{-n}\eta_Z\bar{\eta}_X\left(\xi\wedge\bar{\xi}\right)
\end{aligned}
$$

which implies that $\xi\wedge\bar{\xi}$ is nonzero and divides $u^n - u^{-n}$. □

We have shown in Proposition 5.1.6 that $\operatorname{tr}\boldsymbol{t} = \pm(u^{-n} + u^n)$ is a necessary condition for $\boldsymbol{t}$ to have (semi)gliders. The following proposition shows that this condition is also sufficient.

Proposition 5.1.9. *A CSCA possesses gliders with eigenvalues $\lambda_+ = \pm u^n$ and $\lambda_- = \bar{\lambda}_+ = \pm u^{-n}$ if and only if* $\operatorname{tr}\mathbf{t} = \pm(u^{-n} + u^n)$.

Proof. The "only if" part was already shown in Proposition 5.1.6.

We now assume that $\operatorname{tr}\boldsymbol{t} = u^{-n} + u^n$ and use this to evaluate the characteristic polynomial of $\boldsymbol{t}$. We have

$$\det(\boldsymbol{t} - \lambda\mathbb{1}) = (\boldsymbol{t}_{11} - \lambda)\cdot(\boldsymbol{t}_{22} - \lambda) - \boldsymbol{t}_{12}\boldsymbol{t}_{21} \stackrel{!}{=} 0.$$

Using $\det\boldsymbol{t} = 1$ and $\operatorname{tr}\boldsymbol{t} = u^{-n} + u^n$ we have

$$\lambda^2 - \lambda\cdot(u^{-n} + u^n) + 1 = 0,$$

which is solved by $\lambda_\pm = u^{\pm n}$, $n \in \mathbb{N}$. Thus the CSCA possesses gliders. The case $\operatorname{tr}\boldsymbol{t} = -(u^{-n} + u^n)$ works analogously. □

Now, after we have found the conditions for the existence of gliders, we want to investigate if we can find equivalence classes to connect glider CSCAs. If a set of glider CSCAs can be shown to be equivalent we only need to study one representative. We will make use of this equivalence to show how invariant quasifree states can be constructed for a huge set of glider automata by only studying the prototype G.

Consider an arbitrary CSCA $\boldsymbol{t}$ with (semi-)gliders and a second CSCA $\boldsymbol{b}$. If we transform $\boldsymbol{t}$ by conjugating with $\boldsymbol{b}$ we have $\widetilde{\boldsymbol{t}} = \boldsymbol{b}^{-1}\boldsymbol{t}\boldsymbol{b}$ which, using

$$\operatorname{tr}\widetilde{\boldsymbol{t}} = \operatorname{tr}(\boldsymbol{b}^{-1}\boldsymbol{t}\boldsymbol{b}) = \operatorname{tr}(\mathbb{1}\boldsymbol{t}) = \operatorname{tr}\boldsymbol{t},$$

has the same trace as $\boldsymbol{t}$ and thus is a (semi-)glider automorphism too. What is maybe more surprising is that the converse is also true for gliders of propagation speed one and the same wedge product: any CSCA $\boldsymbol{t}$ with one-step gliders ξ and $\xi\wedge\bar{\xi} = u^{-1} - u$ is equivalent to the standard-glider CSCA $\mathbf{g}_s$ (5.1) by the equivalence relation $\boldsymbol{t} = \boldsymbol{b}^{-1}\mathbf{g}_s\boldsymbol{b}$ for a CSCA $\boldsymbol{b}$. CSCAs $\boldsymbol{t}$ with gliders ξ with $\xi\wedge\bar{\xi} = u - u^{-1}$ are equivalent to the standard CSCA with the glider $\eta = \left(\begin{smallmatrix}1\\-u\end{smallmatrix}\right)$.

Theorem 5.1.10. *Let $\xi = (\xi_X, \xi_Z) \in \mathcal{P}^2$ be minimal and $\xi \wedge \bar{\xi} \neq 0$. Then the following three statements are equivalent:*

1. *There is a CSCA* $\mathbf{t}$ *with* $\mathbf{t}\xi = u\,\xi$.
2. *There is a CSCA* $\mathbf{b}$ *with* $\mathbf{b}\xi = \binom{1}{\pm u}$.
3. $\xi \wedge \bar{\xi} = \pm(u^{-1} - u)$.

Proof.

3 $\Leftrightarrow$ 1: This has already been shown in Proposition 5.1.8, because $\pm(u - u^{-1})$ is the only antisymmetric divisor of $u - u^{-1}$ and $\xi \wedge \bar{\xi}$ is always antisymmetric.[7]

1 $\Rightarrow$ 2: We assume that 1 (and therefore also 3) holds and analyze the conditions this imposes on $\boldsymbol{b}$: We start constructing $\boldsymbol{b}$ using $\boldsymbol{b}\xi = \binom{1}{\pm u}$, $\boldsymbol{b} = \bar{\boldsymbol{b}}$, and $\boldsymbol{b}\bar{\xi} = \binom{1}{\pm u^{-1}}$ and obtain

$$\begin{array}{lrcl}
(\mathrm{I}) & \boldsymbol{b}_{11}\xi_X + \boldsymbol{b}_{12}\xi_Z &=& 1, \\
(\mathrm{II}) & \boldsymbol{b}_{21}\xi_X + \boldsymbol{b}_{22}\xi_Z &=& \pm u, \\
(\bar{\mathrm{I}}) & \boldsymbol{b}_{11}\bar{\xi}_X + \boldsymbol{b}_{12}\bar{\xi}_Z &=& 1, \\
(\bar{\mathrm{II}}) & \boldsymbol{b}_{21}\bar{\xi}_X + \boldsymbol{b}_{22}\bar{\xi}_Z &=& \pm u^{-1}.
\end{array}$$

Using combinations of these equations we can derive conditions on $\boldsymbol{b}$. For $\boldsymbol{b}_{11}$ we have

$$\begin{array}{rl}
& (\mathrm{I}) \cdot \bar{\xi}_Z - (\bar{\mathrm{I}}) \cdot \xi_Z \\
\Leftrightarrow & \boldsymbol{b}_{11}\xi_X\bar{\xi}_Z + \boldsymbol{b}_{12}\xi_Z\bar{\xi}_Z - \boldsymbol{b}_{11}\bar{\xi}_X\xi_Z - \boldsymbol{b}_{12}\bar{\xi}_Z\xi_Z = \bar{\xi}_Z - \xi_Z \\
\Leftrightarrow & \boldsymbol{b}_{11}\left(\xi \wedge \bar{\xi}\right) = \bar{\xi}_Z - \xi_Z. \qquad (5.16)
\end{array}$$

Similarly one can derive

$$\begin{aligned}
\boldsymbol{b}_{12}\left(\xi \wedge \bar{\xi}\right) &= \xi_X - \bar{\xi}_X, && (5.17)\\
\boldsymbol{b}_{21}\left(\xi \wedge \bar{\xi}\right) &= \pm\left(u\bar{\xi}_Z - u^{-1}\xi_Z\right), && (5.18)\\
\boldsymbol{b}_{22}\left(\xi \wedge \bar{\xi}\right) &= \pm\left(u^{-1}\xi_X - u\bar{\xi}_X\right). && (5.19)
\end{aligned}$$

First we need to show that the matrix $\boldsymbol{b}$ is actually a CSCA, i.e. that (a) all the right sides of the equations (5.16) to (5.19) can be divided by $\xi \wedge \bar{\xi}$ and (b) $\det \boldsymbol{b} = 1$. All $\boldsymbol{b}_{ij}$ have to be reflection invariant. That is guaranteed because

[7] By ξ antisymmetric we mean $\xi(u^n) = -\xi(u^{-n})$. For qubits symmetric and anti-symmetric are the same.

we constructed $\boldsymbol{b}$ in a symmetric way. To show (a) we rewrite the right hand sides of (5.18) and (5.19):

$$\begin{aligned}\boldsymbol{b}_{21}\left(\xi \wedge \bar{\xi}\right) &= \pm\left(\overline{u^{-1}\xi_Z} - u^{-1}\xi_Z\right),\\ \boldsymbol{b}_{22}\left(\xi \wedge \bar{\xi}\right) &= \pm\left(u^{-1}\xi_X - \overline{u^{-1}\xi_X}\right).\end{aligned}$$

Now all the right hand sides are of the form $\eta - \bar{\eta}$ and therefore antisymmetric. This implies that they can be divided by $u - u^{-1}$ which proves (a).

(b) can be shown by direct computation:

$$\begin{aligned}\det \boldsymbol{b} &= \frac{1}{\left(\xi \wedge \bar{\xi}\right)^2} \cdot \left(\pm\left(\bar{\xi}_Z - \xi_Z\right) \cdot \left(u^{-1}\xi_X - u\bar{\xi}_X\right)\right.\\ &\quad \left. \mp\left(u\bar{\xi}_Z - u^{-1}\xi_Z\right) \cdot \left(\xi_X - \bar{\xi}_X\right)\right)\\ &= \frac{\pm 1}{\left(\xi \wedge \bar{\xi}\right)^2} \cdot \left(u^{-1} - u\right) \cdot \left(\xi \wedge \bar{\xi}\right)\\ &= \frac{\pm\left(u^{-1} - u\right)}{\xi \wedge \bar{\xi}} \stackrel{!}{=} 1.\end{aligned}$$

Now we see that only gliders with the same wedge product can be mapped onto each other. Gliders ξ with $\xi \wedge \bar{\xi} = u - u^{-1}$ can be mapped onto $\binom{1}{-u}$ and gliders with $\xi \wedge \bar{\xi} = u^{-1} - u$ can be mapped onto $\binom{1}{u}$.

This step only works for $\xi \wedge \bar{\xi} = \pm(u^{-1} + u)$ which corresponds to one step gliders. Later on, we will consider gliders with higher propagation speed and give counterexamples to similar notions of equivalence for their automata.

2 ⇒ 3: A direct computation shows that

$$\xi \wedge \bar{\xi} = \underbrace{\det \boldsymbol{b}}_{=1} \cdot (\xi \wedge \bar{\xi}) = \boldsymbol{b}\xi \wedge \boldsymbol{b}\bar{\xi} = \tbinom{1}{\pm u} \wedge \tbinom{1}{\pm u^{-1}} = \pm\left(u^{-1} - u\right).$$

This completes our proof.

□

Remark 5.1.11. *For semigliders with eigenvalue $-u$ an analogous theorem holds. We only ommitted the case $-u$ to avoid confusion with the $\pm$ of the wedge product.*

We now show that for automata with higher propagation speed the gliders are not equivalent in the above sense. It is apparent from Proposition 5.1.8 that for a fixed $n > 1$ there are gliders with wedge products that differ by more than a sign. These can not be transformed into each other, because the wedge product is invariant

under transformation with CSCAs (see Theorem 5.1.10, part 2). Another way to see that there are different types of n-step glider automata, is the fact that we always have automata which are powers of one-step automata and also automata whose roots are not CSCAs. These can not be transformed into each other. But even automata for gliders with the same wedge product can not always be connected by a third CSCA. To show this, it is sufficient to find two phase space vectors ξ and η with $\xi \wedge \bar{\xi} = \eta \wedge \bar{\eta}$ and $\xi \wedge \bar{\xi}$ dividing $(u^{-n} - u^n)$ for some n which can not be transformed into each other by a CSCA. We choose the vectors $\xi = \binom{1}{u+u^2}$, $\eta = \binom{1+u}{u^2}$ and $\mathbb{Z}_2$ as the underlying field. Their wedge product $\xi \wedge \bar{\xi} = \eta \wedge \bar{\eta} = u^{-2} + u^{-1} + u + u^2$ divides $(u^{-3} + u^3)$. It is a valid wedge product for a 3-step glider. If an automaton $\boldsymbol{b}$ with $\boldsymbol{b}\eta = \xi$ existed, it would have to fulfill the equations

$$\begin{aligned} \boldsymbol{b}_{11}\eta_X + \boldsymbol{b}_{12}\eta_Z &= \xi_X = 1 \\ \boldsymbol{b}_{11}\bar{\eta}_X + \boldsymbol{b}_{12}\bar{\eta}_Z &= \bar{\xi}_X = 1. \end{aligned}$$

From these we obtain

$$\begin{aligned} \boldsymbol{b}_{11}(\eta \wedge \bar{\eta}) &= \bar{\eta}_Z + \eta_Z \\ \Leftrightarrow \boldsymbol{b}_{11}(u^{-2} + u^{-1} + u + u^2) &= u^{-2} + u^2 \end{aligned}$$

which can not be solved by any $\boldsymbol{b}_{11} \in \mathcal{R}$.

5.1.3 Fractal automata

All CSCAs that are neither periodic nor have gliders are called fractal. Fractal means that the graph of the spacetime evolution of one cell observables is self similar in the limit of infinitely many timesteps. We will prove the self similarity of all CSCA time evolutions in Chapter 6. As the self similarities for glider and periodic CSCAs are trivial we only call the remaining CQCAs fractal. Most but not all fractal CSCA actually produce a spacetime image of non-integer fractal dimension, while periodic and glider CSCA have spacetime images of integer dimension. Here we only give an example to illustrate the self similarity. The evolution of the automaton

$$\mathbf{f} = \begin{pmatrix} 0 & 1 \\ 1 & u^{-1} + 1 + u \end{pmatrix} \tag{5.20}$$

is shown in Figure 5.2.

Nevertheless, we will state one short lemma that we will need later on to prove the convergence of product states.

Lemma 5.1.12. *If a CSCA* $\mathbf{t}$ *is fractal,* $\mathbf{t}^n$ *for* $n \in \mathbb{N}$ *is also fractal.*

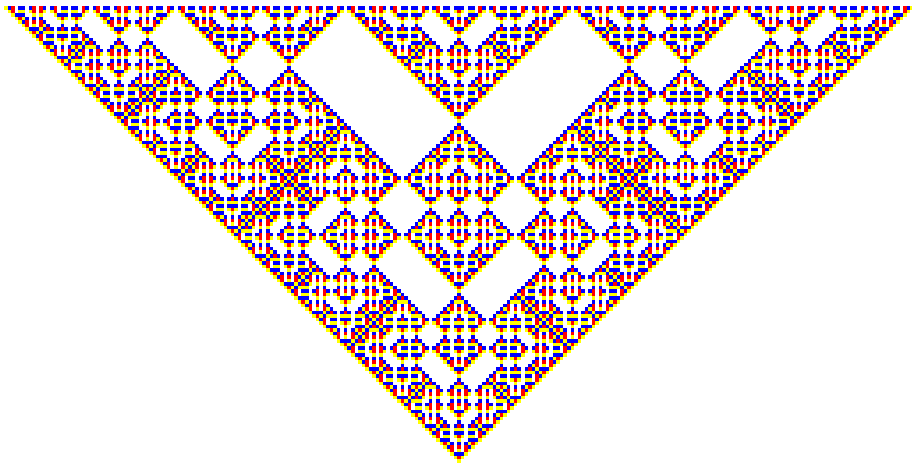


Figure 5.2: Time evolution of a fractal CSCA (5.20); time increases upwards. The different colors illustrate the different Pauli matrices.

Proof. We prove that $\boldsymbol{t}^n$ is fractal by showing that it can be neither periodic nor have gliders. Obviously, no power of $\boldsymbol{t}$ can be periodic if $\boldsymbol{t}$ is not periodic, so only the glider case remains. If $\boldsymbol{t}^n$ has minimal gliders ξ and $\bar{\xi}$, then by

$$\boldsymbol{t}^n(\boldsymbol{t}\xi) = \boldsymbol{t}\boldsymbol{t}^n\xi = \lambda(\boldsymbol{t}\xi)$$

$\boldsymbol{t}\xi$ is also a glider for $\boldsymbol{t}^n$ with the same eigenvalue λ and thus a multiple of ξ. Hence $\boldsymbol{t}\xi = \tilde{\lambda}\xi$ holds for a monomial[8] $\tilde{\lambda}$. If $\tilde{\lambda} = 1$ the automaton would be periodic by Proposition 5.1.3 which has already been ruled out. For $\tilde{\lambda} = u^{\pm n}$, which is the only other possibility, $\boldsymbol{t}$ has gliders. Thus for $\boldsymbol{t}^n$ to have gliders $\boldsymbol{t}$ has to have gliders. This is a contradiction to the assumption that $\boldsymbol{t}$ is fractal. So any power of a fractal CSCA is always fractal. □

5.2 Time asymptotics

In this section, we will consider different types of states on a spin chain and their evolution under CQCA action. Our focus lies on the search for invariant states and the convergence of other states towards invariant states. We consider special types of states: product states, stabilizer states and quasifree states. Because CQCAs act in a translation-invariant manner, it is natural to look at translation-invariant states.

[8] By remark 5.1.7 the only possible eigenvalues are monomials.

We will therefore only consider this class of states. Furthermore, a lot of the methods used here only work for qubits. Therefore, we will only consider qubits and the underlying finite field is $\mathbb{Z}_2$ throughout this section.

The only state that we know to be invariant for all CQCAs (and all QCAs) is the tracial state, which vanishes on all finite Pauli products except the identity and is defined as the limit of states that have the density operator $\frac{1}{2}\mathbb{1}$ for each tensor factor. We strongly suspect this state to be the only invariant state for fractal CQCAs, but have no complete proof yet.

5.2.1 Invariant states for periodic CQCAs

It is clear that all states are periodic under the action of periodic CQCAs. Therefore, a state is either invariant or does not converge at all. Finding invariant states for periodic automata is in general very easy. For example, if a CQCA T satisfies $T^p = \mathbb{1}$ for some finite $p \in \mathbb{N}^+$, then all states of the type $\frac{1}{p}\sum_{n=1}^{p} \omega \circ T^n$ are invariant. Moreover, any invariant state is of this form, because for ω invariant $\frac{1}{p}\sum_{n=1}^{p} \omega \circ T^n = \omega$. Finding pure invariant states is more complicated. In Section 5.2.3 we show, that for some period-two CQCAs pure invariant stabilizer states exist. For CQCAs without propagation (local CQCAs) there also exist invariant product states. Namely, if one Pauli matrix is left invariant by such a CQCA, then the state that gives expectation value one on this matrix and vanishes on the others is left invariant by this CQCA. A state with the same expectation value for all Pauli matrices is always periodic under the action of a non-propagating CQCA.

5.2.2 Invariance and convergence of product states for non-periodic automata

In this section we will consider states that are translation-invariant product states with respect to single cell systems. They are of the form $\omega(\mathrm{w}(\xi)) = \prod_i \Phi(\mathrm{w}(\xi_i))$, where Φ is a state on a single cell.[9]

Proposition 5.2.1. *For a glider CQCA T there exist no translation-invariant product states that are T-invariant except the tracial state. All other translation-invariant product states weakly converge to T-invariant states*

$$\omega_\infty = \lim_{n\to\infty} \omega \circ T^n \tag{5.21}$$

with the following properties:

- $\omega_\infty(\mathrm{w}(\xi)) = 1$ *if* $\mathrm{w}(\xi) = \mathbb{1}$,

[9] This definition extends to a definition of the state on all observables by linearity.

- $|\omega_{\mathscr{C}}(\mathrm{w}(\xi))| < 1$ *if* $\mathrm{w}(\xi)$ *is a product of gliders,*
- $\omega_{\mathscr{C}}(\mathrm{w}(\xi)) = 0$ *otherwise.*

If $\Phi(\sigma_j) = 1$ *for some* j*, then* $\omega_{\mathscr{C}}$ *is the tracial state.*

Proof. First we prove the non-existence of invariant product states. A state is invariant if $\omega \circ T = \omega$, i.e. $\omega(TA) = \omega(A)\,\forall A \in \mathfrak{A}_{\mathbb{Z}}$ is fulfilled. We require the automaton to be non-periodic which implies a finite (non-zero) propagation, so at least two[10] of the Pauli matrices are mapped to tensor products of at least three Pauli matrices[11]. Thus for these two Pauli matrices $\mathrm{w}(\xi)$

$$\begin{aligned}\Phi(\mathrm{w}(\xi)) &= \omega(\mathrm{w}(\xi)) \\ &= \omega(T\mathrm{w}(\xi)) \\ &= \prod_{i\in\mathcal{N}} \Phi(\mathrm{w}((\boldsymbol{t}\xi)_i))\end{aligned}$$

has to hold. For all Pauli matrices σ_i, $|\Phi(\sigma_i)| \leqslant 1$. If $|\Phi(\sigma_i)| = 1$, then $\Phi(\sigma_j) = 0\,\forall j \neq i$.[12] Now lets assume that $\Phi(\sigma_i) \neq 0$ for some i. The image of a single cell observable is either another local observable or has to include at least two different types of Pauli matrices (different from the identity, e.g. σ_1 and σ_2). Else $\xi_X = 0$ or $\xi_Z = 0$ or $\xi_X = \xi_Z$, each case implying common divisors.

Let us consider the case that there exists a Pauli matrix which is not mapped to a tensor product. It can not be mapped to itself, because then by Proposition 5.1.3 the automaton would be periodic. So it has to be mapped to another Pauli matrix which has to expand in the next step. Thus we only need to consider the case of expanding Pauli matrices. The image has to consist of more than one kind of Pauli matrix, so $|\Phi(\sigma_i)| = 1$ is ruled out as an invariant state. Moreover, the image can not contain the original Pauli matrix even once, because $\Phi(\sigma_{i_k}) = \Phi(T\sigma_{i_k}) = (\Pi_{j\in\mathcal{N}\setminus k}\Phi(\sigma_{i_j}))\Phi(\sigma_{i_k})$ implies $|\Phi(\sigma_{i_j})| = 1\,\forall i_j$ which is already ruled out. So no Pauli matrix may occur in its own image, particularly not in the central position. If we only consider those central positions, we get a local automaton. It is an easy calculation to show that all of these automata, which map no Pauli matrix onto itself, have trace one. So the trace of our CQCA contains a constant which is a contradiction to the condition that it has gliders. Therefore, $\Phi(\sigma_i) = 0$ for $i = 1,2,3$ and the only invariant state is the one with the density matrix $\rho_\Phi = \frac{1}{2}\mathbb{1}$, i.e., the tracial state.

[10] The image of the third Pauli matrix is always determined by the product of the other two.

[11] The image has to be a tensor product of at least three Pauli matrices, because an identity in the middle is not allowed.

[12] The eigenstates of the three Pauli matrices lie on the three axes of the Bloch sphere. If ρ is an eigenstate to one of the Pauli matrices the two eigenstates of each of the other Pauli matrices have equal overlap with ρ. But every Pauli matrix has one eigenvector to eigenvalue 1 and one to -1. Thus the expectation value is zero.

In the following we consider the convergence properties of product states. Obviously, it suffices to establish the convergence of Pauli products. First, suppose that some $|\Phi(\sigma_j)| = 1$. It follows, according to Lemma 2.1, that for all Pauli products different from $\mathbb{1}$, and all times except at most one, the evolved product will contain a Pauli matrix different from σ_j, and hence will have zero expectation in $\omega \circ T^n$. Hence ω_∞ is the tracial state.

To treat the remaining cases, we assume from now on that $|\Phi(\sigma_j)| \leqslant \lambda < 1$ for all three j. Hence if some Pauli product has k factors different from $\mathbb{1}$, its expectation in ω is at most λ^k. Let $k(t)$ be the number of non-identity factors in the t^{th} iterate of some Pauli product. If $k(t)$ diverges, we have nothing to prove. So we may assume from now on that there is a constant $k_{max} < \infty$ such that $k(t) < k_{max}$ infinitely often. We focus on the subsequence with $k(t) < k_{max}$.

If the overall length of the Pauli product (largest degree − smallest degree) remains finite, we must have a glider for some power of T, since we have assumed the absence of periodic finite configurations. As argued in the proof of Lemma 5.1.12, this is also a glider for T. In fact, for any product of glider elements, the left going and the right going gliders will eventually be separated, and from that point onwards the ω-expectation will not change anymore. Hence, the limit does exist and will be some number of modulus < 1.

Therefore, we need only consider the case that k_{max} is finite, but the positions of non-identity Pauli factors get more and more spread out. This is only possible if some Pauli products near the edges of the given product have a similar property: no sub-product that gets widely separated from the rest is allowed to develop configurations with unbounded k_{max}, because then the overall bound could not hold. Thus we again find bounded configurations with smaller k_{max}, which may again be smaller gliders, or split up even further. By downwards induction we obtain a complete decomposition into gliders for any configuration with non-divergent k_{max}. This completes our proof. □

Proposition 5.2.2. *Under the action of fractal CQCAs all product states converge to the tracial state.*

Proof. To show convergence for the fractal case, we need the results from Section 5.1.3 and Appendix 2. They state that a tensor product with only one kind of Pauli matrix can occur only once in the history of a fractal CQCA (Lemma 2.1) and that the number of non-identity tensor factors grows unbounded (Lemma 2.2). With the arguments used in the proof of Proposition 5.2.1 this means that for fractal CQCAs any given product state converges to the state that gives a zero expectation value for any non-trivial tensor product of Pauli operators, i.e. to the tracial state. □

5.2.3 Invariance and convergence of stabilizer states

In this section we consider pure translation-invariant stabilizer states that were introduced in Section 2.3. CQCAs and translation-invariant stabilizer states are tightly connected: a stabilizer state $\omega_{\mathrm{w}(\xi)}$ with stabilizer $\mathcal{S} = \langle \mathrm{w}(\hat{\tau}^i \xi),\, i \in \mathbb{Z} \rangle$ has to fulfill the condition that the polynomials of ξ are coprime. This must also hold for the column vectors of a CQCA matrix. This indicates a close connection between CQCAs and translation-invariant stabilizer states. Indeed, CQCAs map pure translation-invariant stabilizer states to pure translation-invariant stabilizer states. Furthermore, any translation-invariant stabilizer state can be generated by a single step of a CQCA from the "all spins up" state, which is a stabilizer product state with stabilizer generators $\mathbb{S} = \{\mathrm{w}(\tau_i(0,1)) = Z_i,\, i \in \mathbb{Z}\}$ [33].

Invariance

Let us first study which CQCAs leave pure translation-invariant stabilizer states invariant.

Proposition 5.2.3. *Only periodic CQCAs can leave pure translation-invariant stabilizer states invariant. For each stabilizer state $\omega_{\mathrm{w}(\xi)}$ with stabilizer group $\mathcal{S} = \langle \mathrm{w}(\hat{\tau}^i \xi),\, i \in \mathbb{Z} \rangle$ the CQCAs that leave the state invariant are periodic and form a group. The corresponding CSCAs are*

$$\mathbf{t}_\xi(a) = \begin{pmatrix} 1 + a\xi_X\xi_Z & a\xi_X^2 \\ a\xi_Z^2 & 1 + a\xi_X\xi_Z \end{pmatrix}, \tag{5.22}$$

for a reflection invariant and centered Laurent polynomial a.

Proof. The invariance condition for a stabilizer state $\omega_{\mathrm{w}(\xi)}$ with stabilizer group $\mathcal{S} = \langle \mathrm{w}(\hat{\tau}^i \xi),\, i \in \mathbb{Z} \rangle$ is

$$\boldsymbol{t}\xi = \lambda\xi,\, \lambda \in \mathcal{P},$$

where λ is an invertible element of $\mathcal{P}$, because $\boldsymbol{t}\xi$ must not have common divisors. As we are in characteristic two and deal with centered automata, only $\lambda = 1$ is possible.

In [33] it was proved that for every translation-invariant stabilizer state ω with stabilizer $\mathcal{S} = \langle \mathrm{w}(\hat{\tau}^i \xi),\, i \in \mathbb{Z} \rangle$ there exists a CQCA B, which maps $\mathrm{w}(0,1)$ to $\mathrm{w}(\xi)$. Thus the condition becomes

$$\boldsymbol{t}\mathbf{b}\binom{0}{1} = \lambda\mathbf{b}\binom{0}{1}.$$

If we know automata that leave the "all spins up" state invariant, we can construct

automata for arbitrary translation-invariant pure stabilizer states via

$$\begin{aligned} & \boldsymbol{t}\binom{0}{1} = \lambda\binom{0}{1} \\ \Leftrightarrow \quad & \underbrace{\mathbf{b}\boldsymbol{t}\mathbf{b}^{-1}}_{\boldsymbol{t}_\xi}\ \underbrace{\mathbf{b}\binom{0}{1}}_{\xi} = \lambda\mathbf{b}\binom{0}{1} \\ \Leftrightarrow \quad & \boldsymbol{t}_\xi\xi = \lambda\xi. \end{aligned} \tag{5.23}$$

The only type of CQCAs that leave the "all spins up" state invariant are the shear transformations, which are represented by matrices

$$\boldsymbol{t} = \begin{pmatrix} 1 & 0 \\ a & 1 \end{pmatrix},$$

where a is some reflection invariant and centered Laurent polynomial. These CQCAs all have $\operatorname{tr}\boldsymbol{t} = 0$ and therefore period two. For a general pure translation-invariant stabilizer state we can determine the CQCAs that leave it invariant from Equation 5.23. They are represented by the CSCAs with matrices

$$\boldsymbol{t}_\xi(a) = \begin{pmatrix} 1 + a\xi_X\xi_Z & a\xi_X^2 \\ a\xi_Z^2 & 1 + a\xi_X\xi_Z \end{pmatrix}.$$

The product of two such CQCAs also leaves the state invariant with $\boldsymbol{t}_\xi(a)\boldsymbol{t}_\xi(b) = \boldsymbol{t}_\xi(a+b)$. Each period 2 CQCA is its own inverse, so the inverse is also included in this set. We therefore have a group of period two CQCAs for each translation-invariant pure stabilizer state that leave this state invariant. □

Convergence

Theorem 5.2.4. *Consider an arbitrary translation-invariant stabilizer state $\omega_{\mathrm{w}(\xi)}$ and the respective stabilizer $\mathcal{S} = \langle \mathrm{w}(\hat{\tau}^x\xi), x \in \mathbb{Z}\rangle$. It follows that $\omega_{\mathrm{w}(\xi)}$ converges for every glider or fractal CQCA T.*

Proof. There exists a CQCA B satisfying $B[\mathrm{w}(\xi)] = \sigma_3$. We rewrite our state in terms of the σ_3-state

$$\omega_{\mathrm{w}(\xi)} = \omega_{\sigma_3} \circ B^{-1} \tag{5.24}$$

and use

$$\begin{aligned} & \omega_{\mathrm{w}(\xi)}(T^n[\textstyle\bigotimes \sigma_i]) \\ = \quad & \omega_{\sigma_3}(B^{-1}T^n[\textstyle\bigotimes \sigma_i]) \\ = \quad & \omega_{\sigma_3}((B^{-1}TB)^n B^{-1}[\textstyle\bigotimes \sigma_i]). \end{aligned}$$

$B^{-1}TB$ is also a CQCA of the same type (same trace) as T and $B^{-1}[\bigotimes \sigma_i]$ is a tensor product of Pauli matrices. ω_{σ_3} is a product state and thus converges according to Propositions 5.2.1 and 5.2.2 for glider and fractal CQCAs. □

5.2.4 Stationary quasifree states and convergence of quasifree states for one-step glider CQCAs

In the previous sections we discussed stationary states and convergence of states under general CQCA actions. In this section we consider the particular glider CQCA G which is represented by the CSCA

$$\mathbf{g} = \begin{pmatrix} 1 & u^{-1}+u \\ 1 & u^{-1}+1+u \end{pmatrix}. \tag{5.25}$$

For this CQCA we can obtain new types of stationary states and new convergence results by employing the Araki-Jordan-Wigner transformation. Using the Theorem for glider equivalence 5.1.10 we can construct invariant states for all automata with gliders that move one step in space every timestep. According to this theorem for any speed-one glider CQCA B, there is a CQCA A such that $B = AGA^{-1}$, and if ω is a G-invariant state, then $\omega \circ A^{-1}$ will be B-invariant.[13]

Araki-Jordan-Wigner transformation

The Jordan-Wigner transformation is a way to map a finite spin-chain algebra $\mathcal{M}_2^{\otimes N}$ to the algebra of a finite fermion chain. This method has been extensively used in solid state physics [78, 79, 80]. However, the method cannot be carried over directly to two-sided infinite chains. One has to introduce an additional infinite "tail-element" for the transformation to work. This extended transformation was introduced by Araki in his study of the two-sided infinite XY-chain [81], and it is sometimes referred to as the Araki-Jordan-Wigner construction.

The C*-algebra describing a two-sided infinite fermion chain is $\mathcal{F}$=CAR($\ell^2(\mathbb{Z})$), i.e. it is the C*-algebra generated by $\mathbb{1}$ and the annihilation and creation operators $\{c_x\}_{x\in\mathbb{Z}}$ and $\{c_x^*\}_{x\in\mathbb{Z}}$, satisfying the canonical anticommutation relations:

$$c_x\, c_y^* + c_y^* c_x = \delta_{x,y}\mathbb{1}, \qquad c_x\, c_y + c_y\, c_x = 0.$$

The translation automorphism $\tau_{\mathcal{F}}$ on this algebra is defined by $\tau_{\mathcal{F}}(c_x) = c_{x+1}$. $\mathcal{F}$ is isomorphic to the observable algebra $\mathfrak{A}_{\mathbb{Z}}$ of the spin chain, but there exists no isomorphism $\iota : \mathfrak{A}_{\mathbb{Z}} \to \mathcal{F}$ that satisfies the property $\iota \circ \tau = \tau_{\mathcal{F}} \circ \iota$[14]. This intertwining property would be needed to derive the translation invariance of a state $\omega \circ \iota$ on $\mathfrak{A}_{\mathbb{Z}}$ from that of ω on $\mathcal{F}$. This problem can be circumvented by the Araki-Jordan-Wigner construction.

The ordinary Jordan-Wigner isomorphism between the N-site spin-chain algebra $M_2^{\otimes N}$ (generated by the finite number of translated Pauli matrices $\{\sigma_1^x, \sigma_2^x, \sigma_3^x\}$

[13] As we deal with qubits here, the field underlying the phase space is $\mathbb{Z}_2$ and there is only one type of speed-one gliders because the signs in the wedge product do not play any role.

[14] Both the CAR algebra and the spin-chain algebra are UHF-algebras which are inductive limits of M_{2^k}. For the spin chain this is immediately clear, for the CAR see [18].

where $x \in \{0,1,\ldots,N-1\}$), and the N-site fermion-chain algebra (generated by $\{c_x,\, c_x^*\}$ where $x \in \{0,1,\ldots,N-1\}$), is given by

$$\begin{aligned}
\iota_{JW}^N(\sigma_1^x) &:= \prod_{y=0}^{x-1}(2c_y\,c_y^* - \mathbb{1})(c_x^* + c_x),\\
\iota_{JW}^N(\sigma_2^x) &:= \prod_{y=0}^{x-1}(2c_y\,c_y^* - \mathbb{1})i(c_x^* - c_x),\\
\iota_{JW}^N(\sigma_3^x) &:= 2c_x\,c_x^* - \mathbb{1}.
\end{aligned}$$

However, as we have mentioned, the Jordan-Wigner transformation can not be generalized to be a translation-intertwining isomorphism between the two-sided infinite spin and fermion chains. In an informal way, one could say that an element of the form "$\prod_{y=-\infty}^{-1}(2c_y\,c_y^* - \mathbb{1})$" would be needed in the definition of a Jordan-Wigner transformation for the two-sided infinite chain. However, $\mathcal{F}$ does not contain such an element. The basic idea of the Araki-Jordan-Wigner construction [81] is to extend the algebra $\mathcal{F}$ with such an infinite tail-element. More concretely, one defines the C*-algebra $\widetilde{\mathcal{F}}$ to be the extension of $\mathcal{F}$ by a self-adjoint unitary element U satisfying:[15]

$$Uc_xU = \begin{cases} c_x & \text{if } x \geqslant 0 \\ -c_x & \text{if } x < 0 \end{cases}.$$

Clearly, every element of $\widetilde{\mathcal{F}}$ can be uniquely written in the form $a+Ub$ with $a,b \in \mathcal{F}$, i.e. $\widetilde{\mathcal{F}} = \mathcal{F} + U\mathcal{F}$. The translation automorphism $\tau_{\mathcal{F}}$ can be extended to $\widetilde{\mathcal{F}}$ through the formula $\widetilde{\tau}_{\mathcal{F}}(a+Ub) := \tau_{\mathcal{F}}(a) + U(2c_0c_0^* - \mathbb{1})\tau_{\mathcal{F}}(b)$. Let us define the elements

$$A_x := \begin{cases} \prod_{y=0}^{x-1}(2c_y\,c_y^* - \mathbb{1}) & \text{if } x > 0 \\ \mathbb{1} & \text{if } x = 0\,, \\ \prod_{y=x}^{-1}(2c_y\,c_y^* - \mathbb{1}) & \text{if } x < 0 \end{cases}$$

and let us introduce

$$\widehat{\sigma_1}^x := UA_x(c_x^* + c_x), \quad \widehat{\sigma_2}^x := UA_x\,i(c_x^* - c_x), \quad \widehat{\sigma_3}^x := (2c_x\,c_x^* - \mathbb{1}). \tag{5.26}$$

Let us denote by $\widehat{\mathcal{F}}$ the C*-subalgebra of $\widetilde{\mathcal{F}}$ that is generated by the elements of $\{\widehat{\sigma_1}^x, \widehat{\sigma_2}^x, \widehat{\sigma_3}^x \,|\, x \in \mathbb{Z}\}$. A direct computation shows that the elements defined in (5.26) having the same spatial index x satisfy the Pauli-relations, while any two of these elements having two different spatial indices commute. Hence $\widehat{\mathcal{F}}$ and $\mathfrak{A}_{\mathbb{Z}}$ are isomorphic, and an isomorphism is given by the map $\Pi : \widehat{\mathcal{F}} \to \mathfrak{A}$, defined as

$$\Pi(\widehat{\sigma_1}^x) = \sigma_1^x, \quad \Pi(\widehat{\sigma_2}^x) = \sigma_2^x, \quad \Pi(\widehat{\sigma_3}^x) = \sigma_3^x \;\forall x \in \mathbb{Z}.$$

[15] In the literature the symbol T is used almost exclusively for denoting this unitary element. However, we chose to denote it by U to avoid confusion with the QCAs we denote by T here.

Moreover, if we denote by $\hat{\tau}_{\mathcal{F}}$ the restriction of $\tilde{\tau}_{\mathcal{F}}$ to $\hat{\mathcal{F}}$ then

$$\tau \circ \Pi = \Pi \circ \hat{\tau}_{\mathcal{F}},$$

i.e. Π intertwines the translations of the two algebras.

Let ω be a translation-invariant state on the fermion-chain algebra $\mathcal{F}$. By defining $\tilde{\omega}(a+Ub) := \omega(a)$ we get a translation-invariant state on $\tilde{\mathcal{F}}$.[16] Restricting this state to $\hat{\mathcal{F}}$ we get a $\hat{\tau}_{\mathcal{F}}$-invariant state $\hat{\omega}$, and $\omega^{JW} = \hat{\omega} \circ \Pi^{-1}$ will be a translation-invariant state on the quantum spin-chain. In this way we can transfer translation-invariant states from the fermion-chain to the spin-chain.

Any CQCA automorphism T can naturally be transferred to an automorphism on $\hat{\mathcal{F}}$ commuting with $\hat{\tau}_{\mathcal{F}}$ by the definition $\hat{T} := \Pi \circ T \circ \Pi^{-1}$. In the case of the glider CQCA G (5.25) we can do even more. The transferred automorphism $\hat{G}$ (characterized by $\hat{G}[\widehat{\sigma_1}^x] = \widehat{\sigma_2}^x$, $\hat{G}[\widehat{\sigma_2}^x] = \widehat{\sigma_2}^{x-1}\widehat{\sigma_1}^x\widehat{\sigma_2}^{x+1}$) can also be extended to an automorphism $\tilde{G} : \tilde{\mathcal{F}} \to \tilde{\mathcal{F}}$ in a translation-invariant way ($\tilde{G} \circ \tilde{\tau}_{\mathcal{F}} = \tilde{\tau}_{\mathcal{F}} \circ \tilde{G}$) with the following definition:[17]

$$\tilde{G}[U] = U(c_{-1} + c_{-1}^*)(c_0 - c_0^*).$$

This definition is motivated by the association of U with a left-infinite string of $\widehat{\sigma_3}$. We have

$$G[\bigotimes_{-\infty}^{-1} \sigma_3] = -i \bigotimes_{-\infty}^{-2} \sigma_3 \otimes \sigma_1^{-1} \otimes \sigma_2^0.$$

The right hand side corresponds to

$$-iUA_{-1}UA_{-1}(c_{-1}^* + c_{-1})UA_0 i(c_0^* - c_0) = U(c_{-1} + c_{-1}^*)(c_0 - c_0^*).$$

The existence of a translation-invariant extension allows us to find stationary states of the glider CQCA by the Araki-Jordan-Wigner method.

The creation and annihilation operators c_x^* and c_x can be decomposed into sums of $\widehat{\sigma_1}^x$ and $\widehat{\sigma_2}^x$:

$$\begin{aligned} c_x &= \widehat{\sigma_1}^x + i\widehat{\sigma_2}^x, \\ c_x^* &= \widehat{\sigma_1}^x - i\widehat{\sigma_2}^x. \end{aligned}$$

Their images under $\tilde{G}$ can be calculated using the action of $\tilde{G}$ on $\widehat{\sigma_1}^x$ and $\widehat{\sigma_2}^x$. Now we restrict the $\tilde{G}$ automorphism to the fermion-chain subalgebra $\mathcal{F} \subset \tilde{\mathcal{F}}$ and obtain the automorphism $G_{\mathcal{F}}$, which acts in the following way:

$$G_{\mathcal{F}}(c_x) = \frac{1}{2}(c_{x-1}^* + c_{x-1} + c_{x+1}^* - c_{x+1}).$$

[16] It is clear that by this definition $\tilde{\omega}$ will be a normalized functional on $\tilde{F}$. The positivity of $\tilde{\omega}$ follows from $\tilde{\omega}((a+Ub)^*(a+Ub)) = \omega(a^*a + b^*b) \geqslant 0$.

[17] We only have to define the image of U under $\tilde{G}$, since any element $\tilde{f} \in \tilde{\mathcal{F}}$ can be uniquely written as a linear combination $\tilde{f} = \hat{f}_1 + U\hat{f}_2$, where $\hat{f}_1, \hat{f}_2 \in \hat{\mathcal{F}}$.

The automorphism $G_{\mathcal{F}} : \mathcal{F} \to \mathcal{F}$ takes an especially simple form in terms of majorana operators. These operators are defined as

$$m_{2x} := i(c_x - c_x^*), \quad m_{2x+1} := c_x + c_x^*, \tag{5.27}$$

for any $x \in \mathbb{Z}$, and they generate $\mathcal{F}$. The action of $G_{\mathcal{F}}$ on these operators is

$$G_{\mathcal{F}}(m_{2x}) = m_{2x-2}, \quad G_{\mathcal{F}}(m_{2x+1}) = m_{2x+3}.$$

Thus the majorana operators act as gliders for the transformed glider CQCA $G_{\mathcal{F}}$.

Clearly, if we find a state ω on $\mathcal{F}$ that is both $\tau_{\mathcal{F}}$- and $G_{\mathcal{F}}$-invariant, then the Araki-Jordan-Wigner transformed state ω^{JW} will be a τ- and G-invariant state on the quantum spin-chain. In the next section, we will briefly recall the definition of quasifree states on fermion-chains and then determine the translation- and $G_{\mathcal{F}}$-invariant quasifree states. In this way, using the Araki-Jordan-Wigner construction, we can obtain a whole class of translation- and G-invariant states on the spin-chain.

Stationary quasifree states

A state $\omega : \mathcal{F} \to \mathbb{C}$ is called *quasifree* if it vanishes on odd monomials of majorana operators

$$\omega(m_{x_1} \dots m_{x_{2n-1}}) = 0,$$

while on even monomials of majorana operators it factorizes in the following form:

$$\omega(m_{x_1} \dots m_{x_{2n}}) = \sum_{\pi} \operatorname{sgn}(\pi) \prod_{l=1}^{n} \omega(m_{x_{\pi(2l-1)}} m_{x_{\pi(2l)}}),$$

where the sum runs over all pairings of the set $\{1, 2, \dots, 2n\}$, i.e. over all the permutations π of the $2n$ elements that satisfy $\pi(2l-1) < \pi(2l)$ and $\pi(2l-1) < \pi(2l+1)$. Hence, if we assume that $x_i \neq x_j$ when $i \neq j$, then $\omega(m_{x_1} \dots m_{x_2 n})$ is simply the *Pfaffian* of the $2n \times 2n$ antisymmetric matrix $A_{i,j} := \omega(m_{x_i} m_{x_j})$.

An automorphism $\alpha : \mathcal{F} \to \mathcal{F}$ that maps any majorana operator onto a linear combination of majorana operators is called a *Bogoliubov automorphism* [82]. For any quasifree state ω, the Bogoliubov transformed state $\omega' := \omega \circ \alpha$ will again be quasifree.

A quasifree state ω is translation-invariant, i.e. $\omega \circ \tau_{\mathcal{F}} = \omega$, if and only if we have $\omega(m_x m_y) = \omega(m_{x+2} m_{y+2})$ for all $x, y \in \mathbb{Z}$. Translation-invariant quasifree states are characterized by a majorana two-point matrices that are a 2×2-block Toeplitz[18]

[18] A block Toeplitz matrix is a matrix which is composed of blocks which are the same on each diagonal.

matrices where the block on position (x,y) is of the form (for a proof, see e.g. [83])

$$\begin{bmatrix} \omega(m_{2x}m_{2y}) & \omega(m_{2x}m_{2y+1}) \\ \omega(m_{2x+1}m_{2y}) & \omega(m_{2x+1}m_{2y+1}) \end{bmatrix} = \frac{1}{2\pi}\int_{-\pi}^{\pi} \begin{bmatrix} q_{1,1}^{(\omega)}(p) & q_{1,2}^{(\omega)}(p) \\ q_{2,1}^{(\omega)}(p) & q_{2,2}^{(\omega)}(p) \end{bmatrix} e^{-ip(x-y)}\mathrm{d}p, \tag{5.28}$$

where $q_{i,j}^{(\omega)}$ are $L^{\infty}([-\pi,\pi])$ functions, and the matrix function

$$Q^{(\omega)}(p) = \begin{bmatrix} q_{1,1}^{(\omega)}(p) & q_{1,2}^{(\omega)}(p) \\ q_{2,1}^{(\omega)}(p) & q_{2,2}^{(\omega)}(p) \end{bmatrix} \in L^{\infty}_{2\times 2}([-\pi,\pi])$$

satisfies

$$(Q^{(\omega)}(p))^{\dagger} = Q^{(\omega)}(p), \quad Q^{(\omega)}(-p) = 2\cdot\mathbb{1} - (Q^{(\omega)}(p))^{T}, \quad 0 \leqslant Q^{(\omega)}(p) \leqslant 2\cdot\mathbb{1} \tag{5.29}$$

almost everywhere[19] ($(Q^{(\omega)}(p))^T$ denotes the transpose of $Q^{(\omega)}(p)$). A translation-invariant quasifree state ω is pure if and only if for almost every p the eigenvalues of $Q^{(\omega)}(p)$ are either 0 or 2. $Q^{(\omega)}(p)$ is called the *symbol* of the majorana two-point matrix $M^{\omega}_{x,y} = \omega(m_x m_y)$.

Now we are ready to characterize the translation-invariant quasifree states that are stationary with respect to the time-evolution $G_{\mathcal{F}}$.

Proposition 5.2.5. *A translation-invariant quasifree state ω is invariant under the $G_{\mathcal{F}}$ automorphism if and only if the symbol of its majorana two-point matrix has the following form:*

$$Q^{(\omega)}(p) = \begin{bmatrix} q_{1,1}^{(\omega)}(p) & 0 \\ 0 & q_{2,2}^{(\omega)}(p) \end{bmatrix},$$

where $q_{1,1}^{(\omega)}$ and $q_{2,2}^{(\omega)}$ are real $L^{\infty}([-\pi,\pi])$ functions that take values between 0 and 2 (almost everywhere).

Proof. The $G_{\mathcal{F}}$ automorphism acts on the majorana fermions as

$$G_{\mathcal{F}}(m_{2x}) = m_{2x-2}, \quad G_{\mathcal{F}}(m_{2x+1}) = m_{2x+3},$$

hence it is a Bogoliubov automorphism. Moreover, $G_{\mathcal{F}}$ commutes with the translations. Thus $\omega' = \omega \circ G_{\mathcal{F}}$ will again be a translation-invariant quasifree state, and ω' is equal to ω if and only if its majorana two-point matrix is the same as that of ω.

[19] The term almost everywhere means everywhere except for a set of measure zero.

The majorana two-point function of ω' is

$$\begin{aligned}
&\begin{bmatrix} \omega'(m_{2x}m_{2y}) & \omega'(m_{2x}m_{2y+1}) \\ \omega'(m_{2x+1}m_{2y}) & \omega'(m_{2x+1}m_{2y+1}) \end{bmatrix} \\
= \quad &\begin{bmatrix} \omega(G_{\mathcal{F}}(m_{2x}m_{2y})) & \omega(G_{\mathcal{F}}(m_{2x}m_{2y+1})) \\ \omega(G_{\mathcal{F}}(m_{2x+1}m_{2y})) & \omega(G_{\mathcal{F}}(m_{2x+1}m_{2y+1})) \end{bmatrix} \\
= \quad &\begin{bmatrix} \omega(m_{2x-2}m_{2y-2}) & \omega(m_{2x-2}m_{2y+3}) \\ \omega(m_{2x+3}m_{2y-2}) & \omega(m_{2x+3}m_{2y+3}) \end{bmatrix} \\
= \quad &\begin{bmatrix} \omega(m_{2x}m_{2y}) & \omega(m_{2x-2}m_{2y+3}) \\ \omega(m_{2x+3}m_{2y-2}) & \omega(m_{2x+1}m_{2y+1}) \end{bmatrix},
\end{aligned}$$

where we have used that ω is $\tau_{\mathcal{F}}$-invariant, and that $\tau_{\mathcal{F}}(m_x) = m_{x+2}$, which follows from the definition of the majorana operators (5.27). Comparing the majorana two-point functions, one can see that $\omega' = \omega$ if and only if $\omega(m_{2x}m_{2y+1}) = \omega(m_{2x-2}m_{2y+3})$ for any $x, y \in \mathbb{Z}$. From form (5.28) of the majorana two-point functions of translation-invariant quasifree states we know that there exists a function $q_{1,2}^{(\omega)} \in L^{\infty}([-\pi,\pi])$ such that

$$\omega(m_{2x}m_{2y+1}) = \frac{1}{2\pi}\int_{-\pi}^{\pi} q_{1,2}^{(\omega)}(p)e^{-ip(x-y)}\mathrm{d}p.$$

The condition $\omega(m_{2x}m_{2y+1}) = \omega(m_{2x-2}m_{2y+3})$, which should hold for all $x, y \in \mathbb{Z}$, means that

$$\begin{aligned}
&\frac{1}{2\pi}\int_{-\pi}^{\pi} q_{1,2}^{(\omega)}(p)e^{-ip(x-y)}\mathrm{d}p - \frac{1}{2\pi}\int_{-\pi}^{\pi} q_{1,2}^{(\omega)}(p)e^{-ip((x-1)-(y+1))}\mathrm{d}p \\
= \quad &\frac{1}{2\pi}\int_{-\pi}^{\pi} q_{1,2}^{(\omega)}(p)(1-e^{2ip})e^{-ip(x-y)}\mathrm{d}p = 0
\end{aligned}$$

must be satisfied. From the inversion theorem for Fourier transformations it follows that a $L^{\infty}([-\pi,\pi])$ function for which the Fourier transformation vanishes must be zero almost everywhere, hence $q_{1,2}^{(\omega)}(p)\cdot(1-e^{2ip}) = 0$, from which one concludes that $q_{1,2}^{(\omega)}(p) = 0$ (See e.g. [84]). This means that for a $\tau_{\mathcal{F}}$- and $G_{\mathcal{F}}$-invariant state ω the symbol of the two-point majorana matrix has to be of the form

$$\begin{bmatrix} q_{1,1}^{(\omega)}(p) & 0 \\ 0 & q_{2,2}^{(\omega)}(p) \end{bmatrix},$$

where $q_{1,1}^{(\omega)}$ and $q_{2,2}^{(\omega)}$ are $L^\infty([-\pi,\pi])$ functions, and according to (5.29) they also have to satisfy the inequalities $0 \leqslant q_{1,1}^{(\omega)}(p) \leqslant 2$ and $0 \leqslant q_{2,2}^{(\omega)}(p) \leqslant 2$ almost everywhere. Thus we have arrived at our proposition. □

Using the Araki-Jordan-Wigner transformation we can transfer such a $\tau_{\mathcal{F}}$- and $G_{\mathcal{F}}$-invariant quasifree state ω on $\mathcal{F}$ to a state ω^{JW} on the spin chain which is τ-invariant and stationary with respect to the glider CQCA G.

Convergence of quasifree states

In this section we will show that under the repeated action of the automorphism $G_{\mathcal{F}}$, a translation-invariant quasifree state will converge to one of the $G_{\mathcal{F}}$-invariant states specified in Proposition 5.2.5.

Proposition 5.2.6. *Let $\omega : \mathcal{F} \to \mathbb{C}$ be a translation-invariant quasifree state with a majorana two-point matrix that belongs to the symbol*

$$Q^{(\omega)}(p) = \begin{bmatrix} q_{1,1}^{(\omega)}(p) & q_{1,2}^{(\omega)}(p) \\ q_{2,1}^{(\omega)}(p) & q_{2,2}^{(\omega)}(p) \end{bmatrix},$$

where $Q^{(\omega)} \in L^\infty_{2\times 2}([-\pi,\pi])$ satisfies the relations (5.29). The n time-step evolved state is denoted by $\omega_n := \omega \circ G_{\mathcal{F}}^n$. The pointwise limit $\omega_\infty(A) := \lim_{n\to\infty} \omega_n(A)$ exists for all $A \in \mathcal{F}$, and the function $\omega_\infty : \mathcal{F} \to \mathbb{C}$ defined in this way will be a translation-invariant quasifree state with a majorana two-point matrix that belongs to the symbol

$$Q^{(\omega_\infty)}(p) = \begin{bmatrix} q_{1,1}^{(\omega)}(p) & 0 \\ 0 & q_{2,2}^{(\omega)}(p) \end{bmatrix}.$$

Proof. In the proof of Proposition 5.2.5 we already showed that if ω is a translation-invariant quasifree state, then $\omega_2 = \omega \circ G_{\mathcal{F}}$ will also be such a state. By induction it follows that ω_n is also a translation-invariant quasifree state for any $n \in \mathbb{N}^+$. Hence for an arbitrary odd monomial of majorana operators $\omega_n(m_{x_1} m_{x_2} \cdots m_{x_{2N+1}}) = 0$. Thus the limit $n \to \infty$ defined by

$$\omega_\infty(m_{x_1} m_{x_2} \cdots m_{x_{2N+1}}) := \lim_{n\to\infty} \omega_n(m_{x_1} m_{x_2} \cdots m_{x_{2N+1}})$$

exists and is zero.

Next, we prove the pointwise convergence of the majorana two-point matrix. The

majorana two-point matrix of ω_n is

$$\begin{aligned}
&\begin{bmatrix} \omega_n(m_{2x}m_{2y}) & \omega_n(m_{2x}m_{2y+1}) \\ \omega_n(m_{2x+1}m_{2y}) & \omega_n(m_{2x+1}m_{2y+1}) \end{bmatrix} \\
= &\begin{bmatrix} \omega(G_{\mathcal{F}}^n(m_{2x}m_{2y})) & \omega(G_{\mathcal{F}}^n(m_{2x}m_{2y+1})) \\ \omega(G_{\mathcal{F}}^n(m_{2x+1}m_{2y})) & \omega(G_{\mathcal{F}}^n(m_{2x+1}m_{2y+1})) \end{bmatrix} \\
= &\begin{bmatrix} \omega(m_{2x-2n}m_{2y-2n}) & \omega(m_{2x-2n}m_{2y+1+2n}) \\ \omega(m_{2x+1+2n}m_{2y-2n}) & \omega(m_{2x+1+2n}m_{2y+1+2n}) \end{bmatrix} \\
= &\begin{bmatrix} \omega(m_{2x}m_{2y}) & \omega(m_{2x-2n}m_{2y+1+2n}) \\ \omega(m_{2x+1+2n}m_{2y-2n}) & \omega(m_{2x+1}m_{2y+1}) \end{bmatrix}.
\end{aligned}$$

This means that the symbol $Q^{(\omega_n)}$ is given by

$$Q^{(\omega_n)}(p) = \begin{bmatrix} q_{1,1}^{(\omega)}(p) & q_{1,2}^{(\omega)}(p)e^{2inp} \\ q_{2,1}^{(\omega)}(p)e^{-2inp} & q_{2,2}^{(\omega)}(p) \end{bmatrix}.$$

The limits of elements of the two-point majorana matrix are:

$$\begin{aligned}
&\begin{bmatrix} \omega_\infty(m_{2x}m_{2y}) & \omega_\infty(m_{2x}m_{2y+1}) \\ \omega_\infty(m_{2x+1}m_{2y}) & \omega_\infty(m_{2x+1}m_{2y+1}) \end{bmatrix} \\
= &\begin{bmatrix} \lim_{n\to\infty}\omega_n(m_{2x}m_{2y}) & \lim_{n\to\infty}\omega_n(m_{2x}m_{2y+1}) \\ \lim_{n\to\infty}\omega_n(m_{2x+1}m_{2y}) & \lim_{n\to\infty}\omega_n(m_{2x+1}m_{2y+1}) \end{bmatrix} \\
= &\frac{1}{2\pi}\begin{bmatrix} \lim_{n\to\infty}\int_{-\pi}^{\pi} q_{1,1}^{(\omega)}(p)e^{-i(x-y)p}\mathrm{d}p & \lim_{n\to\infty}\int_{-\pi}^{\pi} q_{1,2}^{(\omega)}(p)e^{-i(x-y-2n)p}\mathrm{d}p \\ \lim_{n\to\infty}\int_{-\pi}^{\pi} q_{2,1}^{(\omega)}(p)e^{-i(x-y+2n)p}\mathrm{d}p & \lim_{n\to\infty}\int_{-\pi}^{\pi} q_{2,2}^{(\omega)}(p)e^{-i(x-y)p}\mathrm{d}p \end{bmatrix} \\
= &\frac{1}{2\pi}\int_{-\pi}^{\pi}\begin{bmatrix} q_{1,1}^{(\omega)}(p) & 0 \\ 0 & q_{2,2}^{(\omega)}(p) \end{bmatrix} e^{-ip(x-y)}\mathrm{d}p, \qquad (5.30)
\end{aligned}$$

where we have used the Riemann-Lebesgue lemma, which states that for any integrable function f defined on the interval $[a,b]$ (see e.g. Theorem IX.7 of[85]):

$$\lim_{z\to\pm\infty}\int_a^b f(x)e^{izx}dx = 0.$$

For an arbitrary even monomial of majorana operators the convergence can be

proven by:

$$\begin{aligned}
\omega_\infty(m_{x_1}m_{x_2}\cdots m_{x_{2N+1}}) &= \lim_{n\to\infty}\omega_n(m_{x_1}m_{x_2}\cdots m_{x_{2N+1}}) \\
&= \lim_{n\to\infty}\sum_\pi \mathrm{sgn}(\pi)\prod_{l=1}^{N}\omega_n(m_{x_{\pi(2l-1)}}m_{x_{\pi(2l)}}) \\
&= \sum_\pi \mathrm{sgn}(\pi)\prod_{l=1}^{N}\Big[\lim_{n\to\infty}\omega_n(m_{x_{\pi(2l-1)}}m_{x_{\pi(2l)}})\Big] \\
&= \sum_\pi \mathrm{sgn}(\pi)\prod_{l=1}^{N}\omega_\infty(m_{x_{\pi(2l-1)}}m_{x_{\pi(2l)}}). \qquad (5.31)
\end{aligned}$$

Since the finite linear combinations of majorana operators form a norm-dense subset in $\mathcal{F}$[20] the limit $\lim_{n\to\infty}\omega_n(f)$ must exist for any $f\in\mathcal{F}$ due to the uniform boundedness of the states in the sequence, and hence we have obtained with this pointwise limit a linear functional $\omega_\infty:\mathcal{F}\to\mathbb{R}$. Moreover, ω_∞ is uniquely determined by the values it takes on monomials of majorana operators, and since the equations (5.30),(5.31) are satisfied ω_∞ can only be the quasifree state for which the symbol of the majorana two-point function is given by (5.30). □

It is worth mentioning that although a pure quasifree state will, of course, stay pure for any time t, taking the weak, i.e. pointwise, limit of the states when $t\to\infty$ one obtains a mixed state (if the original state was not G-invariant).[21] In quantum many body physics such relaxation from pure to mixed states has been studied in a quite different setting, namely how certain time averages of pure states that evolve under Hamiltonian dynamics can be described by mixed states [86].

Finally, we emphasize that the results of this section using the AJW transformation can generally not be transferred to CQCAs other than the glider CQCA G, because most CQCAs do not have the form of Bogoliubov transformations on the CAR-Algebra. Automata with neighborhoods larger than nearest neighbors can map creation and annihilation operators to products of these on the CAR-algebra and are thus in general not Bogoliubov transformations. (An obvious exception are powers of G.) All automata that do not leave at least one Pauli matrix locally invariant do not allow for a tail element that is left invariant under the transformed CQCA. These automata are characterized by a constant in their trace polynomial. In conclusion, only nearest neighbor automata without a constant on the trace, i.e. some glider and period two automata and their powers, can be transferred to Bogoliubov transformations on the CAR-algebra. For more details see [87].

[20] $\mathcal{F}$ is a UHF algebra and the majorana operators are a basis of all finitely localized observables in $\mathcal{F}$ and are therefore norm-dense in $\mathcal{F}$.

[21] The state will appear to be a mixed state when measuring any localized observable. For finite t this is only true for observables with a localization area smaller than $\mathcal{N}t$, where $\mathcal{N}$ is the neighbourhood of the CQCA.

5.3 Entanglement generation

In this section we will investigate the entanglement generation properties of CQCAs. We consider CQCAs acting on stabilizer and quasifree states. We find that the entanglement generation is linear in time, similar to the case of Hamiltonian time evolution [88]. Entanglement dynamics of special QCAs in finite chains have also studied in [89]. Here we are interested in genral results for asymptotic entanglement generation for CQCAs on infinite chains of qubits.

5.3.1 Entanglement generation starting from translation-invariant stabilizer states

In this section we will consider the entanglement generation of CQCAs acting on translation-invariant pure stabilizer states. We will first calculate the bipartite entanglement in a general translation-invariant pure stabilizer state. Using this result we will present a proof of asymptotically linear growth of entanglement for non-periodic CQCAs.

We will start by repeating some basics on stabilizer states from the introduction in Section 2.3. For every translation-invariant stabilizer state ω with stabilizer generators $\mathbb{S} = \{\mathrm{w}(\hat{\tau}^x\xi), x \in \mathbb{Z}\}$ and stabilizer group $\mathcal{S} = \langle \mathrm{w}(\hat{\tau}^x\xi), x \in \mathbb{Z}\rangle$ there exists a CQCA T that maps $\mathrm{w}(0,1)$ to $\mathrm{w}(\xi)$. Each $\mathcal{S} = \langle \mathrm{w}(\hat{\tau}^x\xi), x \in \mathbb{Z}\rangle$ defines a unique translation-invariant stabilizer state if and only if ξ is reflection invariant and $\gcd(\xi_X, \xi_Z) = 1$ [33]. The image of a one-site Pauli matrix under the action of a CQCA B is always of this form. The study of the entanglement generation of CQCAs acting on initially unentangled stabilizer product states is thus equivalent to the study of the entanglement properties of translation-invariant stabilizer states.

There are several results on the entanglement entropy for stabilizer states in the literature, the most general example would be the formalism developed in [90]. One case considered is a bipartite split of the state $\omega_{\mathscr{A}\mathscr{B}}$ with respect to the subsystems $\mathscr{A}$ and $\mathscr{B}$. By $(\mathcal{P}\xi)_{\mathscr{A}}$ etc. we denote the phase spaces of $\mathcal{S}_{\mathscr{A}}$ etc. In the following we will refer to the whole stabilizer group $\mathcal{S} = \langle \mathrm{w}(\hat{\tau}^x\xi), x \in \mathbb{Z}\rangle$ simply as "stabilizer". The set of stabilizers $\mathcal{S}$ then splits up into three sets. $\mathcal{S}_{\mathscr{A}}$ and $\mathcal{S}_{\mathscr{B}}$ are the local stabilizers, which act non-trivially only on part $\mathscr{A}$ respectively $\mathscr{B}$. The third set $\mathcal{S}_{\mathscr{A}\mathscr{B}}$ accounts for correlations between the subsystems. It is defined as follows:

Definition 5.3.1. *The correlation subgroup $\mathcal{S}_{\mathscr{A}\mathscr{B}}$ for a bipartite stabilizer state is generated by all stabilizer generators that have support on both parts of the system.*

$\mathcal{S}_{\mathscr{A}}$ and $\mathcal{S}_{\mathscr{B}}$ together form the so called local subgroup. The correlation subgroup $\mathcal{S}_{\mathscr{A}\mathscr{B}}$ can be brought into a form where it consists of pairs of stabilizers whose projections on $\mathscr{A}$ (and $\mathscr{B}$) anticommute, but commute with all elements of other pairs

and the local subgroup. The entanglement or von Neumann entropy of such a stabilizer state is

$$E(\omega_{\mathscr{A}\mathscr{B}}) = \frac{1}{2}|\mathcal{S}_{\mathscr{A}\mathscr{B}}|, \tag{5.32}$$

if $\omega_{\mathscr{A}\mathscr{B}}$ is a pure state, where $|\mathcal{S}_{\mathscr{A}\mathscr{B}}|$ is the size of the minimal generating set of $\mathcal{S}_{\mathscr{A}\mathscr{B}}$.

Unfortunately, in [90] only finitely many qubits are considered. The proof of (5.32) relies heavily on this fact. However, there is a different approach to the bipartite entanglement in stabilizer states which we use to extend this result to infinitely many qubits.

In our approach we make use of the phase space description of stabilizer states introduced in [33] and explained in Section 2.3.

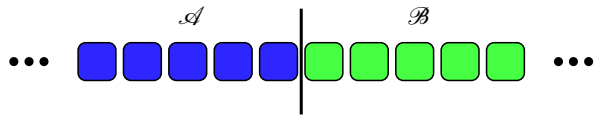


Figure 5.3: The spin chain is cut into two half-chains $\mathscr{A}$ and $\mathscr{B}$; we study the entanglement between these half-chains.

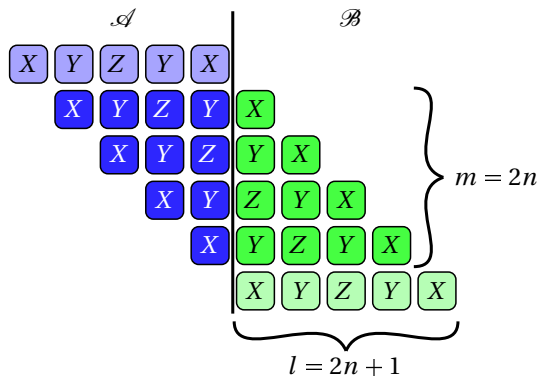


Figure 5.4: Cut stabilizer generators: it is apparent that for operators of length $l = 2n+1$, $2n$ of them have support on both $\mathscr{A}$ and $\mathscr{B}$.

Theorem 5.3.2. *The number of maximally entangled qubit pairs in a pure translation invariant stabilizer state stabilized by $\mathcal{S} = \langle \mathrm{w}(\hat{\tau}^x \xi), x \in \mathbb{Z} \rangle$ on a bipartite spin-chain is the number of pairs $\eta^i, \zeta^i \in (\mathcal{P}\xi)_{\mathscr{A}\mathscr{B}}$ with $\sigma_{\mathscr{A}}(\eta^i, \zeta^j) = \delta_{ij}$, $\sigma(\eta^i_{\mathscr{A}}, \eta^j_{\mathscr{A}}) = \sigma(\zeta^i, \zeta^j) = 0$, $\sigma(\eta^i_{\mathscr{A}}, \mu) = \sigma(\eta^i, \mu) = 0 \, \forall \mu \in (\mathcal{P}\xi)_{\mathscr{A}}$, and $\sigma(\zeta^i_{\mathscr{A}}, \mu) = \sigma(\zeta^i, \mu) =$*

$0\,\forall\mu\in(\mathcal{P}\xi)_{\mathcal{A}}$, *where* $\eta_{\mathcal{A}}$ *and* $\zeta_{\mathcal{A}}$ *denote the restrictions of the phase space vectors to subsystem* $\mathcal{A}$ *completed with* 0 *on* $\mathcal{B}$ *so we can use the the symplectic form* σ *of the whole chain.*

Proof. If we restrict the stabilizer to subsystem $\mathcal{A}$ (or $\mathcal{B}$) it is in general not translation invariant any more. Therefore, the corresponding subspace is not maximally isotropic. The restricted state is not a pure translation invariant stabilizer state. However the uncut stabilizer operators in $\mathcal{S}_{\mathcal{A}}$ stabilize a subspace of the statespace. The elements of the correlation subgroup $\mathcal{S}_{\mathcal{A}\mathcal{B}}$ that is generated by the cut stabilizer generators map this subspace onto itself because they commute with the elements of $\mathcal{S}_{\mathcal{A}}$. From the theory of stabilizer quantum error correction [64] we know that a pair of operators leaving a stabilized subspace invariant can be used to encode a logical qubit if the operators fulfill the same commutation relations as σ_1 and σ_3. As the restrictions of the elements of $\mathcal{S}_{\mathcal{A}\mathcal{B}}$ to one of the subsystems do not have to commute, such pairs of operators can exist. In the phase space description this means that we have to find $\eta,\zeta\in(\mathcal{P}\xi)_{\mathcal{A}\mathcal{B}}$ with $\sigma(\eta_{\mathcal{A}},\zeta_{\mathcal{A}})=1$, $\sigma(\eta_{\mathcal{A}},\mu_{\mathcal{A}})=\sigma(\eta,\mu)=0\,\forall\mu\in(\mathcal{P}\xi)_{\mathcal{A}}$, and $\sigma(\zeta_{\mathcal{A}},\mu_{\mathcal{A}})=\sigma(\zeta,\mu)=0\,\forall\mu\in(\mathcal{P}\xi)_{\mathcal{A}}$. Several such pairs encode several qubits. Of course, the operators from different pairs have to commute. Thus, the qubits are encoded through pairs of operators $(\bar\sigma_1^i,\bar\sigma_3^i)=(\mathrm{w}(\eta^i),\mathrm{w}(\zeta^i))$ whose phase space vectors fulfill $\sigma_{\mathcal{A}}(\eta^i,\zeta^j)=\delta_{ij}$, $\sigma(\eta^i_{\mathcal{A}},\eta^j_{\mathcal{A}})=\sigma(\zeta^i,\zeta^j)=0$, $\sigma(\eta^i_{\mathcal{A}},\mu)=\sigma(\eta^i,\mu)=0\,\forall\mu\in(\mathcal{P}\xi)_{\mathcal{A}}$, and $\sigma(\zeta^i_{\mathcal{A}},\mu)=\sigma(\zeta^i,\mu)=0\,\forall\mu\in(\mathcal{P}\xi)_{\mathcal{A}}$. As we have $\sigma(\eta_{\mathcal{A}},\zeta_{\mathcal{A}})+\sigma(\eta_{\mathcal{B}},\zeta_{\mathcal{B}})=\sigma(\eta_{\mathcal{A}}+\eta_{\mathcal{B}},\zeta_{\mathcal{A}}+\zeta_{\mathcal{B}})=\sigma(\eta,\zeta)=0\,\forall\eta,\zeta\in\mathcal{P}\xi$ we know that $\sigma(\eta_{\mathcal{B}},\zeta_{\mathcal{B}})=\sigma(\eta_{\mathcal{A}},\zeta_{\mathcal{A}})$, thus the restrictions of our operators to system $\mathcal{A}$ and system $\mathcal{B}$ fulfill the same commutation relations. We therefore have pairs of operators of the form $(\bar\sigma_1^A\otimes\bar\sigma_1^{\mathcal{B}},\bar\sigma_3^{\mathcal{A}}\otimes\bar\sigma_3^{\mathcal{B}})$. Each such pair encodes a Bell pair as seen in Example 2.3.1. Thus the number of maximally entangled qubit pairs is the number of such pairs of operators. □

We now show that the number of qubit pairs is $\frac{1}{2}|\mathcal{S}_{\mathcal{A}\mathcal{B}}|$. As mentioned in Section 2.3, only Weyl operators $\mathrm{w}(\xi)$ fulfilling certain conditions can span the stabilizer $\mathcal{S}=\langle\mathrm{w}(\hat\tau^x\xi),x\in\mathbb{Z}\rangle$ of a pure state. On the level of tensor products of Pauli matrices the above conditions have three important consequences that stem from the requirement for ξ_X and ξ_Z to have no common divisors and to be reflection invariant:

1. The length of the product has to be odd, because palindromes of even length are always divisible by $(1+u)$. We will write $l=2n+1$.
2. The central element of the product can not be the identity. Else ξ has the divisor $(u^{-1}+u)$.
3. At least two different types of elements (both different from the identity) have to occur (e.g. σ_1 and σ_2). Else $\xi_X=0$ or $\xi_Z=0$ or $\xi_X=\xi_Z$, each case implying common divisors.

If we make a bipartite cut[22] in our system, $2n$ stabilizers will be affected. All other operators are localized on one side of the cut, only those with localization on both sides are elements of $\mathcal{S}_{\mathscr{A}\mathscr{B}}$. If we could find k pairs of anticommuting operators in the projections of $\mathcal{S}_{\mathscr{A}\mathscr{B}}$ on the right (or left) half-chain there would be k pairs of maximally entangled qubits. In fact we can always find $k = n$ such pairs and thus $k = \frac{1}{2}|\mathcal{S}_{\mathscr{A}\mathscr{B}}|$.

Definition 5.3.3. *The bipartite entanglement $E(\omega_\xi)$ of a translation-invariant stabilizer state ω_ξ is the number of maximally entangled logical qubit pairs with respect to any bipartite cut. The bipartite entanglement equals the entanglement entropy, because the entanglement entropy of each Bell pair is* 1.

Theorem 5.3.4. *A pure translation-invariant stabilizer state ω_ξ of stabilizer generator length $2n+1$ entangles n logical qubit pairs maximally with respect to any bipartite cut.*

$$E(\omega_\xi) = \frac{\text{length}(\xi) - 1}{2}. \tag{5.33}$$

Proof. We use the criterion of Theorem 5.3.2 and explicitly construct pairs $\mathrm{w}(\xi_i)$, $\mathrm{w}(\eta_i)$ using methods from stabilizer codes for quantum error correction (see Section 3.10.2). As stated above only stabilizer generators localized on both sides of the cut are elements of the correlation group $\mathcal{S}_{\mathscr{A}\mathscr{B}}$. The projections of all other stabilizer generators are just the stabilizer generators themselves that trivially commute with all other stabilizer generators and their projections on $\mathscr{A}$ respectively $\mathscr{B}$. We now use the phase space representation of the Pauli products and build the following $2n \times 4n$-matrix from the cut translates of the stabilizer generators:

$$\left(\begin{array}{cccc|cccc} \xi_X^{-n} & \xi_X^{-n+1} & \cdots & \xi_X^{n-1} & \xi_Z^{-n} & \xi_Z^{-n+1} & \cdots & \xi_Z^{n-1} \\ 0 & \xi_X^{-n} & \cdots & \xi_X^{n-2} & 0 & \xi_Z^{-n} & \cdots & \xi_Z^{n-2} \\ \vdots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \xi_X^{-n} & 0 & \cdots & 0 & \xi_Z^{-n} \end{array}\right).$$

Let us assume that the outermost element of the stabilizer generators is a σ_1[23]. Then $\xi_X^{-n} = 1$ and $\xi_Z^{-n} = 0$. From Section 5.3.1 we know that at least one $\xi_Z^i \neq 0$. Let the i-th diagonal of the right part be the first non-zero one. We also use the

[22] Due to the translation invariance all possible cuts are equivalent.
[23] The other cases work equivalently.

reflection invariance of ξ to replace ξ^{-j} by ξ^{j} and obtain the following matrix:

$$\left(\begin{array}{ccccc|ccccccccc}
1 & \xi_X^{n-1} & \xi_X^{n-2} & \cdots & \xi_X^{n-1} & \cdots & 0 & \xi_Z^{n-i} & \xi_Z^{n-i-1} & \cdots & \xi_Z^{n-i} & 0 & \cdots & 0 & \\
0 & 1 & \xi_X^{n-1} & \cdots & \xi_X^{n-2} & & & & & & & \ddots & & & \\
\vdots & & \ddots & & \vdots & & & & & & 0 & \xi_Z^{n-i} & \xi_Z^{n-i-1} & \cdots & \xi_Z^{n-i} \\
 & & & & & \cdots & \vdots & & \ddots & & & & \ddots & \vdots & \\
 & & & & & & & & & & & & & 0 & \xi_Z^{n-i} \\
 & & & & & & & & & & & & & & 0 \\
 & & & & & & & & & & & & & & \vdots \\
0 & \cdots & 0 & 1 & \xi_X^{n-1} & & & & & & & & & & \\
0 & \cdots & 0 & 0 & 1 & \cdots & 0 & & \cdots & & & & & 0 &
\end{array}\right).$$

Now we can perform the Gaussian algorithm on the matrix to obtain an identity matrix in the left part. Since the rows are shifted copies of the first row, all operations will also be applied in a shifted copy. If we were to add the third row to the first, we would also add the fourth to the second and so forth. We only add rows to rows above, because the lower left part of the matrix is already zero. We therefore have

$$\left(\begin{array}{ccccc|ccccccc}
1 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & \zeta^{-n+i} & & \cdots & \zeta^{n-1} \\
0 & 1 & 0 & \cdots & 0 & & & & & & \ddots & \vdots \\
 & & & & & & & & & & 0 & \zeta^{-n+i} \\
\vdots & & \ddots & & \vdots & \vdots & & & \ddots & & & 0 \\
 & & & & & & & & & & & \vdots \\
0 & \cdots & 0 & 1 & 0 & & & & & & & \\
0 & \cdots & 0 & 0 & 1 & 0 & & & & \cdots & & 0
\end{array}\right).$$

The i-th row of the right part of the matrix remains unchanged, so $\zeta^{-n+i} = \xi_Z^{n-i} = 1$. We therefore obtain operators of the form

$$\tilde{s}_k = \mathrm{w}(\tilde{\xi}_k) = \sigma_1^{-n+k} \otimes \left(\bigotimes_{j=-n+k+1}^{-n+k+i-1} \mathbb{1}^j\right) \otimes \sigma_3^{-n+k+i} \otimes \left(\bigotimes_{j=-n+k+i+1}^{n-1} \sigma_{?_j}^j\right),$$

where the $\sigma_{?_j}$ can only be σ_3 or $\sigma_0 = \mathbb{1}$, $1 \leqslant k \leqslant 2n$, and identities on the left side are omitted. We can easily see that $\{\tilde{s}_k, \tilde{s}_{k+i-1}\} = 0$. As $i \leqslant n$ we can always find n pairs of anticommuting operators. We call these pairs $(s_{2i-1}, s_{2i})_{i=1\ldots n}$. Unfortunately these pairs do not necessarily commute with other pairs. But through multiplication of operators we can find new pairs that fulfill the necessary commutation relations. To show this we create a (symmetric) matrix $c_{i,j} = \sigma(\xi_i, \xi_j)$ of

commutation relations.

$$C = \begin{array}{c|ccccc} & s_1 & s_2 & s_3 & s_4 & \cdots \\ \hline s_1 & 0 & 1 & ? & ? & \cdots \\ s_2 & 1 & 0 & ? & ? & \cdots \\ s_3 & ? & ? & 0 & 1 & \cdots \\ s_4 & ? & ? & 1 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{array} .$$

A "1" stands for anticommutation, a "0" for commutation. In the end we want all operators from different pairs to commute, so all positions denoted by question marks should contain a zero entry. If we multiply operators the corresponding rows and columns are added, because multiplication of operators corresponds to addition of their phase space vectors and σ is linear. Through these operations we can bring the commutation matrix to the form

$$\widehat{C} = \begin{array}{c|ccccc} & s_1 & s_2 & s_3 & s_4 & \cdots \\ \hline s_1 & 0 & 1 & 0 & 0 & \cdots \\ s_2 & 1 & 0 & 0 & 0 & \cdots \\ s_3 & 0 & 0 & 0 & 1 & \cdots \\ s_4 & 0 & 0 & 1 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{array} .$$

To show that this is possible, we will consider a prototype of such an operation. Given the matrix

$$C = \begin{array}{c|ccccc} & s_1 & s_2 & s_3 & s_4 & \cdots \\ \hline s_1 & 0 & 1 & c_{13} & c_{14} & \cdots \\ s_2 & 1 & 0 & c_{23} & c_{24} & \cdots \\ s_3 & c_{13} & c_{23} & 0 & 1 & \cdots \\ s_4 & c_{14} & c_{24} & 1 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{array}$$

we now pick a nonzero c_{ij} and do the following: if $c_{ij} = 1$ and i odd, we add row $i+1$ to row j and the same for the columns. If $c_{ij} = 1$ and i even, we add row $i-1$ to row j and the same for the columns. This only changes the one c_{ij} we are considering, the others remain unchanged. After each step we get a new matrix $\widetilde{C}$ and pick another nonzero c_{ij} from the same 2×2 block (in this example we only have one block). By doing this for all blocks in the first two rows, we create pairs of operators that commute with the first pair. Now we have to check if the process destroyed the anticommutation within the pairs. The diagonal entries of the matrix trivially stay zero, because all operators commute with themselves. We only have to check the other elements of the block (due to the symmetry, we only have to check

one). So if $c_{13} = 1$ we have $1 \mapsto 1 + c_{24}$. We can write $1 \mapsto 1 + c_{13}c_{24}$. Including the whole block of c_{ij} we have $1 \mapsto 1 + c_{13}c_{24} + c_{14}c_{23} + c_{23}c_{14} + c_{24}c_{13} = 1$ as all operations are carried out modulo 2. The new pairs thus fulfill the anticommutation condition. We can repeat this process for the new pairs until all operators from different pairs commute. We started with $2n$ operators, thus we arrived at n pairs which together with their counterparts on the other subsystem encode n pairs of maximally entangled qubits. □

Now it remains to show, how the stabilizer generator length evolves under the action of CQCAs and to deduce the asymptotic entanglement generation rate.

Definition 5.3.5. *The asymptotic entanglement generation rate from stabilizer states for CQCAs is defined as*

$$\frac{\Delta E}{\Delta t}(\omega_\xi) = \lim_{t\to\infty} \frac{1}{t} E(\omega_\xi, t), \tag{5.34}$$

where $E(\omega_\xi, t)$ is the bipartite entanglement at time t.

We will now prove the following theorem:

Theorem 5.3.6. *The asymptotic entanglement generation per step of a general centered CQCA T is the highest exponent in its trace polynomial,* $\mathrm{dg}(\mathrm{tr}\,\mathbf{t})$.

For the proof we need the following lemma:

Lemma 5.3.7. *Given a CQCA T and a pure translation invariant stabilizer state ω_ξ. Then the length $2n+1$ of the stabilizer generator of a stabilizer state grows asymptotically with*

$$\frac{\Delta n}{\Delta t} = \lim_{t\to\infty} \frac{1}{t} n(t, \xi) = \mathrm{dg}(\mathrm{tr}\,\mathbf{t}). \tag{5.35}$$

Proof. We know that CQCAs map pure translation-invariant stabilizer states onto pure translation-invariant stabilizer states. The image of a state with stabilizer generators $\mathbb{S} = \{\mathrm{w}(\tau_x \xi), x \in \mathbb{Z}\}$ under the action of t steps of a CQCA T is a state with stabilizer generators $\mathbb{S}^t = \{\mathrm{w}(\tau_x \boldsymbol{t}^t \xi), x \in \mathbb{Z}\}$. Furthermore, we know that any stabilizer state can be generated from the "all spins up" state by a CQCA $\boldsymbol{b}_\xi$. So we have $\mathbb{S}^t = \{\mathrm{w}(\tau_x \boldsymbol{t}^t \boldsymbol{b}_\xi(0,1)), x \in \mathbb{Z}\}$. The length of the stabilizer generator is determined by the highest order of the stabilizer generator polynomials, $\mathrm{dg}(\xi)$. Namely the stabilizer generator is of length $2 \cdot \mathrm{dg}(\xi) + 1$. So we have to calculate $\mathrm{dg}(\boldsymbol{t}^t \xi) = \mathrm{dg}(\boldsymbol{t}^t \boldsymbol{b}_\xi \binom{0}{1})$ and $n(t, \xi) = \mathrm{dg}(\boldsymbol{t}^t \boldsymbol{b}_\xi \binom{0}{1})$.

For an arbitrary product of CQCAs $\prod_{i=1}^k \boldsymbol{t}_i$ we can define the series $(a_l)_{1\leqslant l\leqslant k} = \mathrm{dg}(\prod_{i=1}^l \boldsymbol{t}_i)$. It is subadditive, i.e. $a_{n+m} \leqslant a_n + a_m$, because the concatenation of CQCAs is essentially the multiplication and addition of polynomials and both operations are are subadditive in the exponents. For subadditive series a_n, Fekete's lemma [91] states that the limit $\lim_{n\to\infty} \frac{a_n}{n}$ exists. In our case the series is always

positive, so the limit is positive and finite. An easy way to determine the limit is to take a subseries, which of course has the same limit. The subseries of the $t = 2^k$th ($k \in \mathbb{N}$) steps is a good candidate, because we can make use of the Cayley-Hamilton theorem to obtain

$$\boldsymbol{t}^{2^k} = \boldsymbol{t}(\operatorname{tr}\boldsymbol{t})^{2^k-1} + \mathbb{1}\sum_{i=1}^{k}(\operatorname{tr}\boldsymbol{t})^{2^k-2^i}$$

and

$$\begin{aligned}\mu(k) &:= \operatorname{dg}(\boldsymbol{t}^{2^k}\boldsymbol{b}\tbinom{0}{1}) \\ &= \operatorname{dg}(\boldsymbol{t}\boldsymbol{b}\tbinom{0}{1}(\operatorname{tr}\boldsymbol{t})^{2^k-1} + \boldsymbol{b}\tbinom{0}{1}\sum_{i=1}^{k}(\operatorname{tr}\boldsymbol{t})^{2^k-2^i}) \\ &= \eta\cdot c(k) + \zeta\cdot d(k)\end{aligned}$$

with $\eta = \boldsymbol{t}\boldsymbol{b}\binom{0}{1}$, $\zeta = \boldsymbol{b}\binom{0}{1}$, $c(k) = (\operatorname{tr}\boldsymbol{t})^{2^k-1}$, and $d(k) = \sum_{i=1}^{k}(\operatorname{tr}\boldsymbol{t})^{2^k-2^i}$. The coefficients $c(k)$ and $d(k)$ fulfill the recursion relations

$$\begin{aligned}c(k+1) &= (\operatorname{tr}\boldsymbol{t})^{2^{k+1}-1} = (\operatorname{tr}\boldsymbol{t})^{2^k-1}(\operatorname{tr}\boldsymbol{t})^{2^k} = c(k)(\operatorname{tr}\boldsymbol{t})^{2^k}, \\ d(k+1) &= \sum_{i=1}^{k+1}(\operatorname{tr}\boldsymbol{t})^{2^{k+1}-2^i} = \sum_{i=1}^{k}(\operatorname{tr}\boldsymbol{t})^{2^k+2^k-2^i} + (\operatorname{tr}\boldsymbol{t})^0 = d(k)(\operatorname{tr}\boldsymbol{t})^{2^k} + 1.\end{aligned}$$

Therefore,

$$\mu(k+1) = \mu(k)(\operatorname{tr}\boldsymbol{t})^{2^k} + \zeta.$$

At this point we need a binary case distinction. Either (1.) $\operatorname{dg}(\mu(k))$ is bounded by $\operatorname{dg}(\zeta)$, implying peridocity of $\boldsymbol{t}$, or (2.) $\operatorname{dg}(\mu(k))$ is unbounded and passes $\operatorname{dg}(\zeta)$ so no cancellation can occur and we can easily calculate the limit.

1. If $\operatorname{dg}(\mu(k))$ is uniformly bounded by $\operatorname{dg}(\zeta)$, this implies $\operatorname{dg}(\mu(k)) \leqslant \operatorname{dg}(\zeta)$ for all k. Then $n(t = 2^k, \xi) = \operatorname{dg}(\mu(k))$ is bounded and $\frac{\Delta n}{\Delta t} = 0$. But $n(t, \xi)$ bounded also implies $\boldsymbol{t}$ periodic and therefore $\operatorname{dg}(\operatorname{tr}\boldsymbol{t}) = 0$. Thus we have $\frac{\Delta n}{\Delta t} = 0 = \operatorname{dg}(\operatorname{tr}\boldsymbol{t})$.

2. If $\operatorname{dg}(\mu(k))$ is not uniformly bounded by $\operatorname{dg}(\zeta)$, there exists a $\mu(k_0)$ such that $\operatorname{dg}(\mu(k_0)) > \operatorname{dg}(\zeta)$. Then $\operatorname{dg}(\mu(k_0+1)) = \operatorname{dg}(\mu(k_0)) + \operatorname{dg}(\operatorname{tr}\boldsymbol{t})(2^{k_0})$ and by recursion $\operatorname{dg}(\mu(k+1)) = \operatorname{dg}(\mu(k)) + \operatorname{dg}(\operatorname{tr}\boldsymbol{t})(2^k)\,\forall k \geqslant k_0$. Now we can calculate

the limit:

$$\begin{aligned}
\lim_{k\to\infty}\frac{1}{2^k}\mathrm{dg}(\boldsymbol{t}^{2^k}\mathbf{b}\binom{0}{1}) &= \lim_{k\to\infty}\frac{1}{2^k}\mathrm{dg}(\mu(k)) \\
&= \lim_{k\to\infty}\frac{1}{2^k}\left(\mathrm{dg}(\mu(k_0))+\sum_{i=1}^{k-k_0}\mathrm{dg}(\mathrm{tr}\,\boldsymbol{t})(2^{k-i})\right) \\
&= \underbrace{\lim_{k\to\infty}\frac{1}{2^k}\mathrm{dg}(\mu(k_0))}_{=0}+\lim_{k\to\infty}\sum_{i=1}^{k-k_0}\mathrm{dg}(\mathrm{tr}\,\boldsymbol{t})\frac{1}{2^i} \\
&= \mathrm{dg}(\mathrm{tr}\,\boldsymbol{t}).
\end{aligned}$$

Thus in all cases $\frac{\Delta n}{\Delta t}=\mathrm{dg}(\mathrm{tr}\,\boldsymbol{t})$. □

Now we can proceed to the proof of Theorem 5.3.6.

Proof of Theorem 5.3.6. As shown in Theorem 5.3.4, a stabilizer state of stabilizer generator length $2n+1$ encodes n maximally entangled logical qubits with respect to any bipartite cut. In Lemma 5.3.7 we showed that the minimal length of a stabilizer generator grows asymptotically with $2\cdot\mathrm{dg}(\mathrm{tr}\,\boldsymbol{t})$ under the action of a CQCA T. Together these results prove the theorem. □

This means that starting from the "all-spins-up" stabilizer product state, the entanglement grows linearly with $\mathrm{dg}(\mathrm{tr}\,\boldsymbol{t})$ under the action of a CQCA T, because $\boldsymbol{b}=\mathbb{1}$. Starting from an arbitrary translation-invariant pure stabilizer state, the entanglement might decrease in the first k steps, e.g. if $\boldsymbol{t}^k=\boldsymbol{b}^{-1}$, but then starts to increase linearly with $\mathrm{dg}(\mathrm{tr}\,\boldsymbol{t})$. This behavior is shown in Figure 5.5.

We can also calculate the entanglement of a finite region, i.e. L consecutive spins, with the rest of the chain as shown in Figure 5.6. To do this calculation, we use the same method as above and arrive at the following theorem.

Theorem 5.3.8. *Given a pure translation-invariant stabilizer state of stabilizer generator length $2n+1$, a region of length L shares $2n$ pairs of maximally entangled qubits with the rest of the chain if $2n\leqslant L$ and L pairs if $2n>L$.*

Proof. The proof can be carried out using the same method as in the bipartite case. In the case $2n\leqslant L$ the cut stabilizers are only cut on one side. But all stabilizers that are cut on the left side commute with those cut on the right side. Thus we have two independent cuts of the bipartite case and therefore $2n$ pairs of maximally entangled qubits. In the case $2n>L$ some stabilizers are cut on both sides. We arrange them in a $(2n+L)\times 2L$-matrix as in the proof of Theorem 5.3.4 and use the same technique to produce the mutually commuting anticommuting pairs which encode the qubits. We always find L pairs of maximally entangled qubits. □

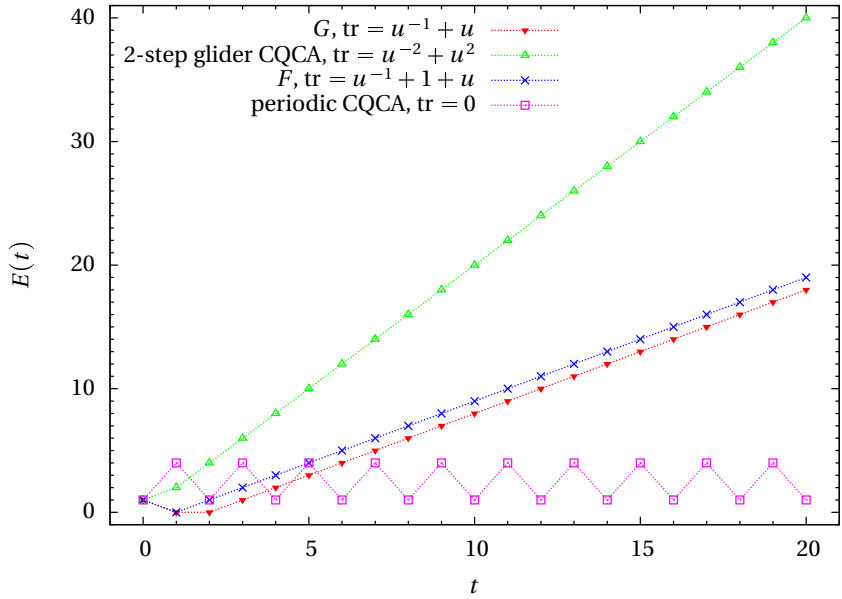


Figure 5.5: Entanglement generation for the stabilizer state with $w(\xi) = \sigma_2 \otimes \sigma_1 \otimes \sigma_2$ in a bipartite setting with different CQCAs; one can see that entanglement can also be destroyed, but grows asymptotically linear with the number of timesteps n. The coefficient is given by the degree of the trace of the CSCA matrix.

For the evolution of entanglement under the action of a CQCA T, this means that starting with a product stabilizer state, the entanglement grows with $2 \cdot \operatorname{tr} \boldsymbol{t}$ until it reaches L. Then it remains constant. If we start with a general translation-invariant stabilizer state, the entanglement might decrease at first. After some time it starts increasing and reaches L, where it remains if the CQCA is not periodic. Results are shown in Figure 5.7.

In both cases, the asymptotic entanglement generation only depends on the degree of the trace of the automaton. A CQCA with gliders generates entanglement just as fast as a fractal automaton if the degree of the trace is the same. Of course periodic automata destroy all entanglement they just generated in the next steps. This corresponds to the fact that their trace is a constant. The entanglement generation rate is however not directly governed by the neighborhood of the CQCA. Of course the size of the neighborhood bounds the possible rate of entanglement gen-

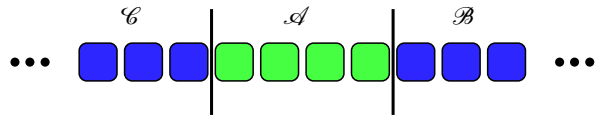


Figure 5.6: A finite region $\mathscr{A}$ of 4 spins is cut out of the chain leaving two infinite ends $\mathscr{B}$ and $\mathscr{C}$.

eration from above, but even automata with a huge neighborhood can be periodic and thus generate no entanglement at all.

CQCAs saturate the bound on the entanglement generation rate for translation-invariant operations acting on translation-invariant states derived in [39] and explained in Section 2.3. This means that there is no translation-invariant operation that can generate more entanglement per step while having the same size of the neighborhood, than CQCAs with maximal degree with respect to the neighborhood (of course there are non-CQCA operations that have the same entanglement generation rate).

5.3.2 Entanglement generation starting from translation-invariant quasifree states

In this section we study the entanglement generation of the glider automorphism G acting on a family of pure quasifree states that interpolates between the all-spins-up state (discussed in the previous section) and a glider-invariant state (discussed in Section 5.2.4).

Let ω be a pure translation-invariant quasifree state, and let $\omega_{[1,L]}$ denote its restriction to the lattice points $\{1,2,\ldots,L\}$. The entanglement entropy of the restricted state $\omega_{[1,L]}$ can be calculated from the eigenvalues $\{\lambda_i\}_{i=1\ldots 2L}$ of the restricted majorana two-point matrix $\left[M_{n,m}\right]_{n,m=1}^{2L}$ by the formula [92, 93]:

$$S(\omega_{[1,L]}) = -\sum_{i=1}^{2L} \lambda_i/2 \log(\lambda_i/2). \tag{5.36}$$

The family of states that we will consider as initial states are the pure translation-invariant quasifree states ω_A described by the symbol (see Section 5.2.4)):

$$Q^{(\omega_A)}(p) = \begin{bmatrix} 1-\chi_{[-\pi A,0]}(p)+\chi_{[0,\pi A]}(p) & i\left[1-\chi_{[-\pi A,0]}(p)-\chi_{[0,\pi A]}(p)\right] \\ -i\left[1-\chi_{[-\pi A,0]}(p)-\chi_{[0,\pi A]}(p)\right] & 1-\chi_{[-\pi A,0]}(p)+\chi_{[0,\pi A]}(p) \end{bmatrix},$$

where $\chi_{[a,b]}$ denotes the characteristic function of the interval $[a,b]$, and A is some real number between 0 and 1. The state corresponding to $A=0$ is the all-spins-up

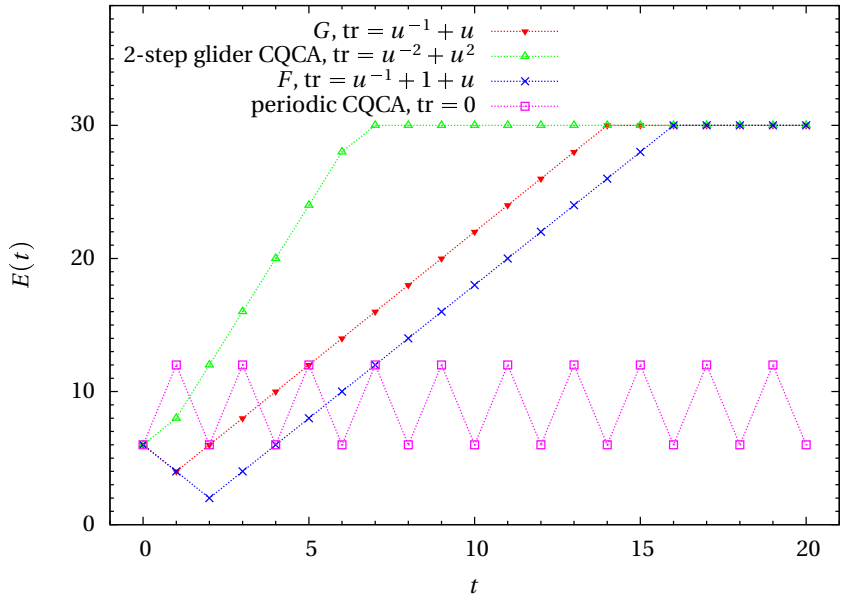


Figure 5.7: Evolution of entanglement of a subchain of 30 consecutive spins for an initial stabilizer state with $w(\xi) = \sigma_2 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_1 \otimes \sigma_2$ for different CQCA actions; the entanglement first grows as in the bipartite case, but then saturates at 30 qubit pairs.

state, while the state corresponding to $A = 1$ is a glider-invariant state. We have shown in Section 5.2.4 that by applying the glider automorphism n-times on ω_A one obtains a quasifree state $\omega_A^{(n)}$ belonging to the symbol

$$Q^{(\omega_A^{(n)})}(p) = \begin{bmatrix} 1 - \chi_{[-\pi A,0]}(p) + \chi_{[0,\pi A]}(p) & i[1 - \chi_{[-\pi A,0]}(p) - \chi_{[0,\pi A]}(p)]e^{2inp} \\ -i[1 - \chi_{[-\pi A,0]}(p) - \chi_{[0,\pi A]}(p)]e^{-2inp} & 1 - \chi_{[-\pi A,0]}(p) + \chi_{[0,\pi A]}(p) \end{bmatrix}.$$

Using this result and Formula (5.36), we calculated the entanglement generation numerically.[24] The results are shown in Figures 5.8 and 5.9. We can observe that the

[24] More precisely, we use Equation (5.28) to determine the translation invariant two-point matrix M from the symbol $Q^{(\omega_A^{(n)})}(p)$. Then we calculate the eigenvalues of the restriction M_L to L spins. Using Equation (5.36) we obtain the entanglement generation.

entanglement generation is linear in time, its rate is maximal when $A = 0$, and the rate can be arbitrarily small (when A approaches 1). This is illustrated in Figure 5.8. For longer sub-chains it takes more time steps for the entanglement to saturate. We show this in Figure 5.9.

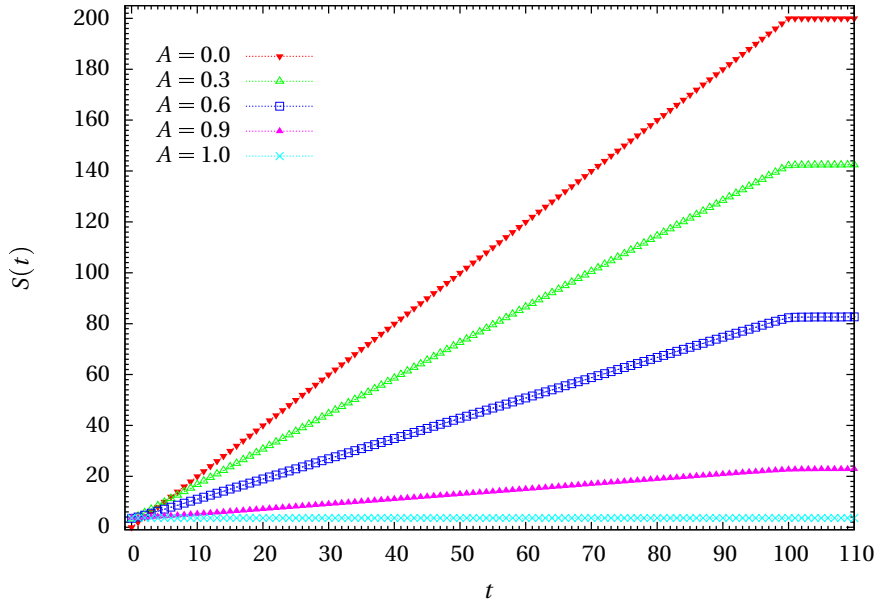


Figure 5.8: The entanglement entropy of a subchain of length 200 after applying the glider time-evolution t times; different colors mark the different initial value of the parameter A of the initial quasifree state.

5.4 Notes on finite systems

Throughout our analysis we have used an infinite spin chain to obtain translation invariance, which would be broken by the ends of the spin chain. For finite chains we generally have to take the effects of the ends of the chain into account. But when we deal with observables that are localized far away from the ends of the chain and time periods T that are much smaller than the length of the chain, for locality reasons the time evolution of the observables has to be the same as for the infinite chain. Information from the ends of the chain can only travel with the same finite speed, as all other information, so, e.g. a one-site observable localized in the middle

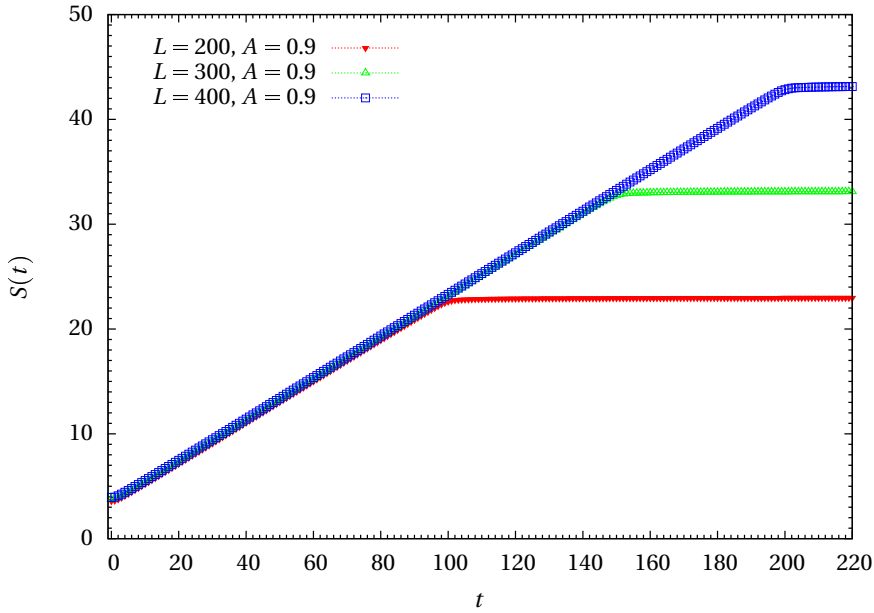


Figure 5.9: The entanglement entropy of a subchain of L consecutive spins in dependence of the time steps; for larger L more entanglement is possible. Different colors refer to different length of the chain. A is fixed at 0.9.

of a chain of length $L = 2l + 1$ can only interfere with influences from the ends after $l/2$ time steps for a nearest neigbor CQCA.

Admittedly, the applications of CQCAs make heavy use of the effects occurring at the ends of the spin chains. Thus, our above statement is not applicable. To take effects at the ends of the spin chains into account, we have to deal with the fact that they break the translation invariance. The one-site images of the sites at the ends of the chain whose neighborhoods would reach over one end of the chain, if they were the same as on the rest of the chain, have to be adapted. Like in the case of stabilizer states above, in general the cut images do not fulfill the necessary commutation relations any more. There is no general theory yet of how to adapt the CQCAs at the ends of the spin chain. However, for special cases the cut images still fulfill the commutation relations. Fortunately, the much used glider CQCA is of this type. So, in this case the only influence from the ends of the chain is that the outermost sites miss the influence from one neighbor. As we still have an automorphism and thus a reversible operation on the whole chain, no information can be

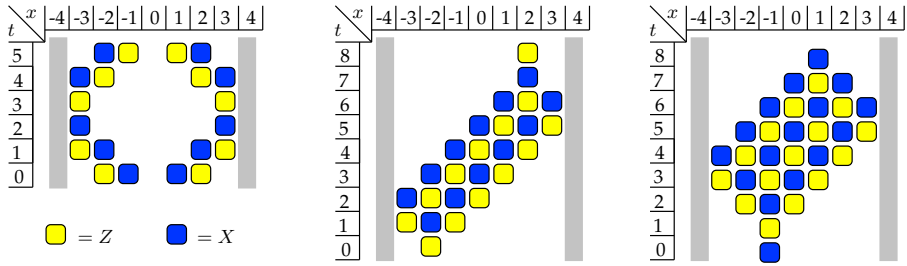


Figure 5.10: Time evolution of the glider CQCA on a finite chain of 7 spins; in the left-hand illustration we can see that the gliders are reflected at the ends of the chain. In the other two illustrations we see that on a finite chain the glider CQCA mirrors the position of single-site Pauli matrices. As all Pauli products can be decomposed into single-site Pauli matrices, this holds for all observables. (All observables can be decomposed into a sum of Pauli products, which are each mirrored in the same number of time steps, so the whole observable is mirrored.)

lost at the ends of the chain. So the ends have to be reflective. We can observe this in the case of an incoming glider in the left part of Figure 5.10. This leads to the fact that observables are mirrored by the chain. For single-site Pauli matrices, this can be seen in the right-hand part of Figure 5.10. Arbitrary observables are sums of tensor products of single-site Pauli matrices. Because the single-site Pauli matrices are mirrored, the tensor products will be mirrored, too. Thus, also sums are mirrored and therefore any observable is mirrored on the chain. This property and the spreading of observables in this process is used in applications of CQCAs such as [74, 73].

5.5 Applications

CQCAs are used in different quantum computational schemes. However, since they are essentially classical cellular automata, they are not capable of performing computations efficiently that a classical computer can not perform efficiently. Nevertheless, they can be of great use when accompanied by non-Clifford operations. This is used in different approaches.

5.5.1 Measurement based quantum computation

The most famous model of quantum computation involving a CQCA (indirectly) is the idea of measurement based quantum computation or the “one way quantum

computer" by Raussendorf and Briegel [2]. The resource state used to perform the quantum computation by successively measuring the single qubits is a so called cluster state on a 2-dimensional (finite) lattice of qubits. It can be generated by a CQCA, namely a 2-dimensional version of the glider CQCA (5.1). Thus one time-step of a CQCA together with measurements of single qubits suffices for universal quantum computation.

5.5.2 Raussendorf's scheme of translation invariant quantum computation

In this scheme of translation-invariant quantum computation [73], the standard glider CQCA is used. The property of generating patterns from single site observables, thus spreading them over the spin-chain, (a finite chain is used here) is used to immunize observables against special global transformations. The time steps in which certain observables are immune against these operations depend on their initial position on the chain. Thus temporal control (when to apply the global gates) can be turned into spatial control and any quantum operation can be conducted on the system. An example of such behavior is be the effect of a global Pauli σ_2 operation (a local σ_2 on each site). It gives a sign on each σ_1 and σ_3 tensor factor. So in our example shown in Figure 5.10 the σ_3^{-2} gains a phase of -1 when the σ_2 is applied in one of the steps $\{0,1,6,7,8\}$, while it gains no phase in the other steps. The σ_1^{-1} gains the phase in steps $\{0,1,2,3,6,7,8\}$. The steps where a phase is gained belong to the contraction respectively expansion of the observables, while the steps where no phase is gained belong to the transmission of the expanded observable over the chain. So if we for example run the automaton for $2L+2=16$ time-steps and apply a global σ_2 in step 3, σ_1^{-1} gains a phase but σ_3^{-2} does not. The application of σ_2 in steps $\{1,3,6,7,\}$ add a phase to σ_3^{-2}. So we turned temporal control into spatial control. To achieve universal quantum computation, we use arbitrary translation invariant local rotations instead of the global σ_2.

5.5.3 The quantum computational scheme by Fitzsimons and Twamley

In [74] the same property of the glider CQCA is used. However, in this case the non-Clifford operations are not translation-invariant, but only conducted at the ends of the chain. Separate control of the ends of the chain is justified since due to the missing neighbor, the physical properties of the systems at the end differ from those in the middle of the chain. The CQCA is used to transport qubits to the end of the chain, which can then be manipulated. Two qubit gates are achieved by first decoupling one spin (at the end of the chain) from the chain and then transporting the other one to its neighboring position. In the following the gate is applied and the

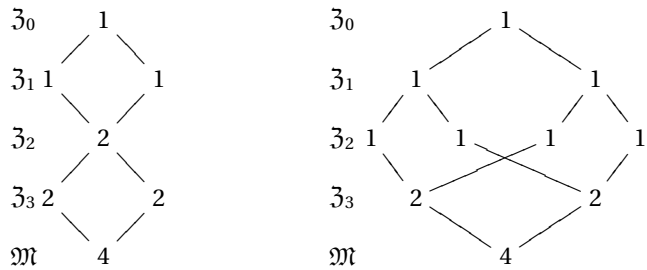
qubits are transported back. This scheme was experimentally realized in a NMR-system [76].

5.6 Bratteli diagrams of causal CQCAs

It suggests itself to use Bratteli diagrams to analyze causal qubit CQCAs. To convert a centered CQCA into a causal CQCA we concatenate it with τ^n, where $n = \mathrm{dg}(\mathrm{tr})\boldsymbol{t}$ (The neighborhood size of the CQCA is $2n+1$.): $\tilde{T} = \tau^n T$. We found that the Bratteli diagrams is not connected to the class of a CQCA. CQCAs of different classes can have the same Bratteli diagram, while CQCAs with the same trace can have different Bratteli diagrams. The following example CQCAs are both periodic with trace 0 but show different Bratteli diagrams (see Figures 5.11a and 5.11b):

$$\boldsymbol{t}_1 = \begin{pmatrix} \frac{1}{u^3}+\frac{1}{u^2}+\frac{1}{u} & 1+\frac{1}{u^4} \\ \frac{1}{u^2} & \frac{1}{u^3}+\frac{1}{u^2}+\frac{1}{u} \end{pmatrix}, \tag{5.37}$$

$$\boldsymbol{t}_2 = \begin{pmatrix} \frac{1}{u^2} & 1+\frac{1}{u^4} \\ 0 & \frac{1}{u^2} \end{pmatrix}. \tag{5.38}$$



(a) Example CQCA $\boldsymbol{t}_1$ (5.37)

(b) Example CQCA $\boldsymbol{t}_2$ (5.38)

Figure 5.11: CQCAs with the same trace can have different Bratteli diagrams.

The memory usage of a causal global-shift-free CQCA is minimal with respect to the memory depth: a centered CQCA T obviously has index 0. Therefore, by the index formula (3.15), the causal CQCA $\tilde{T} := \tau^n T$ has index n. By Corollary 3.7.5 the maximal memory depth for a channel with n qubits of memory is $2n$. As $\tilde{T}$ is global-shift-free the causal inverse $\tilde{T}'$ also has a memory depth of $2n$ (Lemma 4.5.3). By $\tilde{T}\tilde{T}' = \tau^{2n}$ and Equation (3.15) it also needs n qubits of memory.

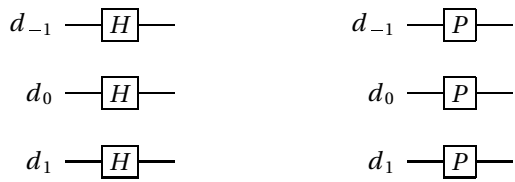
The two example Bratteli diagrams (Figures 5.11a and 5.11b) are are actually the only two maximal depth Bratteli diagrams for $n = 2$. In general there are 2^{n-1} different maximal depth Bratteli diagrams for a Clifford memory channel with n qubits

of memory: In each step of the diagram either pairs of nodes have to be joined or each node has to be split in two. So in each step there are two possibilities. But in total the number of splits and joins has to be equal, so only half of the choices are actually free. The rest is determined by the others. Thus there are $2^n/2$ choices and therefore 2^{n-1} different Bratteli diagrams.

5.7 Circuit and channel implementations of CQCAs

In order to implement a CQCA on a quantum computer it is useful to have a description of the CQCA as a quantum circuit. In [33] it was shown that all CQCAs can be decomposed into a set of basic CQCAs, namely the one-qubit Hadamard and phase transformations H and P, the shear transformations S_n and shifts τ. These were already introduced in the beginning of this chapter. A circuit for each of these basis transformations will allow us to find a circuit for every CQCA.

The implementation of the local CQCAs is simple—they directly correspond to the gates of the same name. Their circuits are shown in Figures 5.12a and 5.12b.



(a) H (b) P

Figure 5.12: Circuits to implement the local CQCAs H and P

The implementation of the shear transformation is also straightforward using the controlled-Z (CZ) gate. The CZ is especially useful for CQCAs, because it commutes with all its translates—a property the CNOT can not offer. Furthermore, it is symmetric in control and target qubit. A CZ between qubits i and j will be denoted by $CZ(i,j)$. However, we will only use translation invariant applications of the CZ and therefore it is only $|i-j|$ that matters. In the Laurent polynomial notation we will write $CZ(n)$ to denote a CZ gate between all qubits and their right nth neighbors (1st = nearest etc.). The shear transformation S_n is implemented by a translation-invariant $CZ(n)$. See also Figure 5.13a.

The implementation of the shift τ poses slightly more problems, as it can not be implemented by a partition of the CQCA into layers of unitary transformations. However, this problem can be circumvented by introducing ancilla qubits [37]. We will see that the implementation of the shift in one lattice direction on the data qubits is realized by shifting the information on the ancilla qubits in the opposite

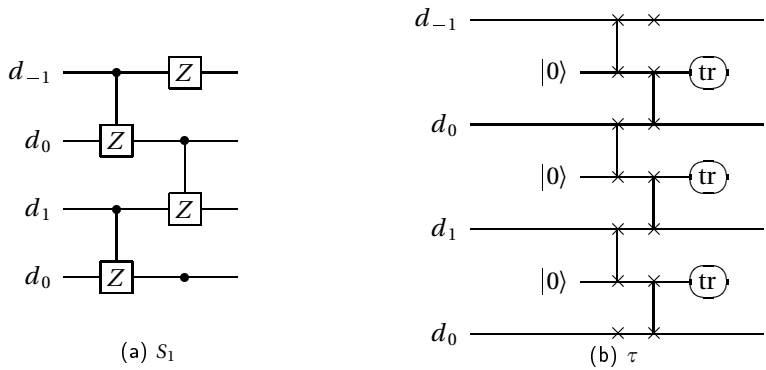


Figure 5.13: Circuits to implement shear transformation S_1 and the shift CQCA τ

direction. This is a special case of the result that any quantum cellular automaton can be implemented locally, when we simultaneously compute the inverse QCA on a copy of the lattice [38].

To implement the shift we add an ancilla qubit in a fixed state between every pair of data qubits. We then first swap the ancilla i with data qubit i followed by swapping ancilla i with data qubit $i+1$. Each of the two steps is carried out simultaneously for all qubits. After the two steps data qubit i has moved to position $i+1$ and ancilla qubit i to $i-1$. The observables are shifted in the opposite direction. Each of the steps only couples every ancilla to one adjacent data qubit, so all the operations of one step commute. Of course the order of the two steps matters, as the operations of different steps do not all commute. Changing the order of the two steps implements the shift in the opposite direction. The circuit is shown in Figure 5.13b.

CQCAs can be implemented by Clifford memory channels, if they only propagate in one direction, i.e. if they are causal operations. As seen in the last section this can be easily achieved by multiplying with the appropriate shift. We will now look at examples of CQCAs from the different classes according to the classification introduced in Section 5.1. They are all nearest neighbor, so we have to multiply them all with the right shift τ to make them causal (the right shift shifts observables to the left). We will take our usual examples F and G_s and the periodic shear transformation S_1. The matrices of their causal versions are

$$\tau\mathbf{g}_s = \begin{pmatrix} 0 & u^{-1} \\ u^{-1} & 1+u^{-2} \end{pmatrix}, \tag{5.39}$$

$$\tau\mathbf{f} = \begin{pmatrix} 0 & u^{-1} \\ u^{-1} & 1+u^{-1}+u^{-2} \end{pmatrix}, \tag{5.40}$$

and

$$\tau s_1 = \begin{pmatrix} u^{-1} & 0 \\ 1+u^{-2} & u \end{pmatrix}. \tag{5.41}$$

Let us first consider the glider CQCA. The circuit is shown in Figure 5.14. The first part is the shift consisting of swaps with ancillas as described above. The second part starts with a Hadamard gate on each qubit followed by controlled Z gates between all neighboring qubits. The controlled Z are split into two groups that are each translation invariant only by shifts by two qubits. This is simply to avoid overlapping gates and give the circuit a clearer layout. The controlled Z gates commute and can all be applied at the same time. The resulting circuit is exactly the circuit used to create one-dimensional cluster states (see e.g. [94]). See also Section 5.5.

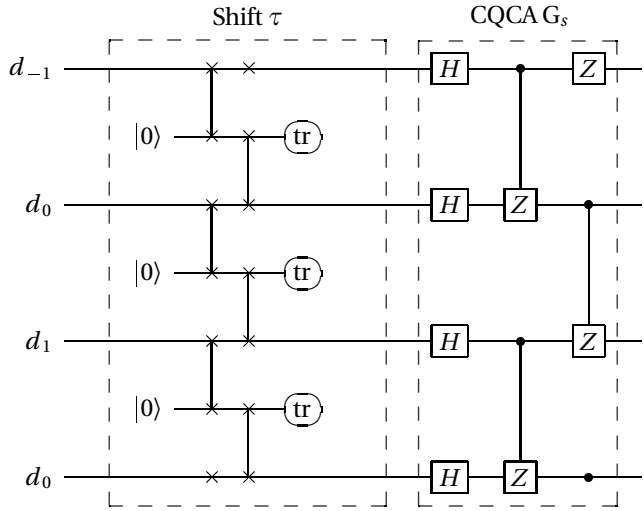


Figure 5.14: Circuit to implement the shifted glider CQCA τG_s

The circuit for the fractal CQCA τF (see Figure 5.15) only differs from the glider circuit by the addition of a phase gate between Hadamard and controlled Z

The circuit of the shifted periodic CQCA τS_1 (Figure 5.16) is again very similar to the glider circuit. Here the Hadamard gate is missing and we only have the controlled Z gates. It is easy to see why this CQCA is periodic up to a shift: multiple steps of the CQCA correspond to multiple repetitions of the same circuit. As it is translation invariant we can commute all the shift parts to the front leaving a circuit of only controlled Z gates. Controlled Z gates are their own inverses, so n layers

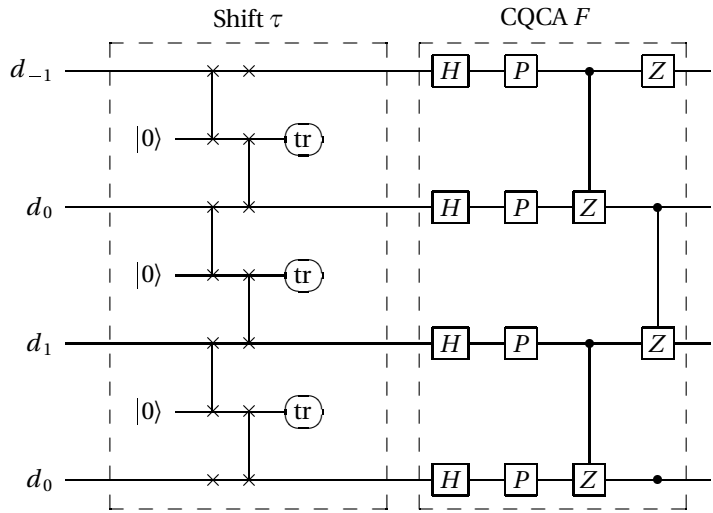


Figure 5.15: Circuit to implement the shifted fractal CQCA τF

of controlled Z gates from n repetitions of the CQCA collapse to one (n odd) or zero (n even) layers. Thus S_1 is periodic with period two.

To implement the CQCAs by memory channels the finite-depth circuits we developed above have to be brought into the form of a memory channel. An easy way to carry out this transformation is to use the commutation relations of gates and shift all gates on "later" (larger index) qubits that commute with gates on "earlier" qubits to later times. This approach is used in [46, 47] to derive memory channel encoders from pearl-necklace encoders for quantum convolutional codes. We demonstrate this with the circuit (Figure 5.14) of the shifted glider CQCA: the swap operations come in pairs that can be aligned in a causal structure. The CZ gates all commute so they can also be aligned in a causal structure. The swap pair of qubits i with the adjacent ancillas can be shifted past the CZ of qubits $i-2$ and $i-1$. The resulting circuit is shown in Figure 5.17. The circuit uses two qubits of memory while the index of the CQCA indicates that only one qubit is needed (see Theorem 4.2.1). Starting from the pearl-necklace structure and obtaining a memory channel structure by shifting gates might give us a channel with a memory that is larger than necessary, as shown in [95]. The interpretation of the circuit as a memory channel introduces an additional problem. The output of the ith box is on wire $i-1$, while the input is on wire i. To be interpreted as a memory channel the ith box has to work on the ith

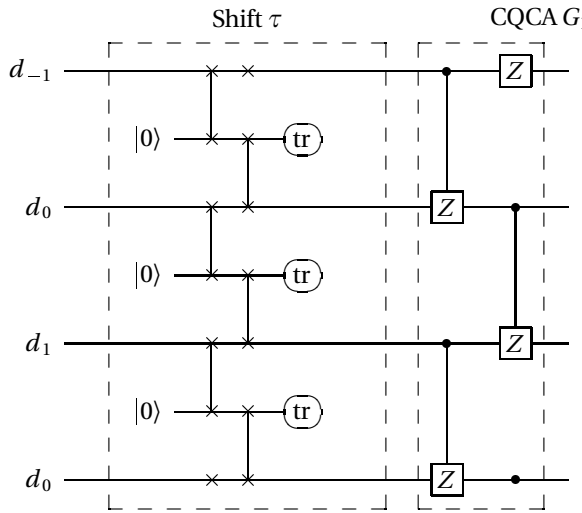


Figure 5.16: Circuit to implement the shifted shear transformation CQCA τG_1

qubit and a memory system. Thus a memory channel using the box in the circuit would implement an additional unwanted right shift: $\tau \mathrm{G}_s \tau = \tau^2 \mathrm{G}_s$. That shows that it is necessary for the channel to use two qubits of memory.

We will now derive a memory channel implementing a causal CQCA using the method introduced in Section 4.3.1 and show that it uses only one qubit of memory. In the next step we will also circumvent the problem of the additional shift.

We start by studying the images of single cell Pauli matrices under the action of the CQCA $\tau \mathrm{G}_s$:

$$
\begin{array}{cccccccccccc|ccc}
\cdots & I & I & X & I & I & \cdots & \to & \cdots & I & Z & I & I & I\cdots \\
\cdots & I & I & Z & I & I & \cdots & \to & \cdots & Z & X & Z & I & I\cdots \\
\cdots & I & I & I & X & I & \cdots & \to & \cdots & I & I & Z & I & I\cdots \\
\cdots & I & I & I & Z & I & \cdots & \to & \cdots & I & Z & X & Z & I\cdots \\
\cdots & I & I & I & I & X & \cdots & \to & \cdots & I & I & I & Z & I\cdots \\
\cdots & I & I & I & I & Z & \cdots & \to & \cdots & I & I & Z & X & Z\cdots
\end{array}
$$

The Pauli products right of the cut indicate that one qubit of memory should be

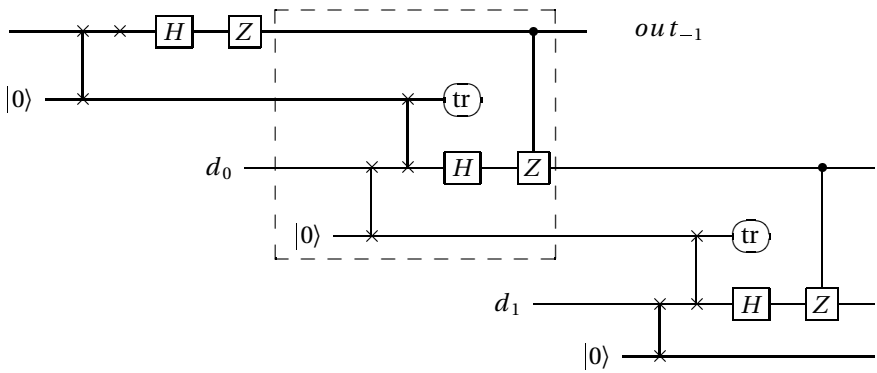


Figure 5.17: Causal circuit to implement the shifted glider CQCA τG_s. The stroked box corresponds to one use of the memory channel implementing the CQCA. It is easy to see that the channel uses two qubits of memory. Using the box as a memory channel it would implement an additional unwanted shift τ.

sufficient for the encoder (See 4.3.1. The following transformation

$$\begin{array}{cccc} o & m & & m & i \\ X & I & \rightarrow & Z & I, \\ Z & I & \rightarrow & X & Z, \\ I & X & \rightarrow & Z & X, \\ I & Z & \rightarrow & I & Z \end{array} \tag{5.42}$$

fulfills the requirements. Using its phase space description and the decomposition algorithm from Section 3.6.3 we arrive at a circuit consisting of only a single CNOT and a single Hadamard gate per input qubit. It is shown in Figure 5.18. The new circuit not only uses only one qubit of memory and less gates per input qubit, but also removes the need of the ancilla qubits. This is possible, because we are no longer limited to commuting operations and can use CNOT operations. The new circuit can not be transformed into a finite depth circuit by just shifting gates in time. Swapping the positions of two of the CNOT gates would introduce an error which would need to be corrected with additional gates. A string of only CNOT gates would be non-forgetful, because a single X would be mapped to an infinite string of X tensor factors. The Hadamard gates swap X and Z thereby making the circuit strictly forgetful. In this circuit the box is actually a valid memory channel implementation of τG_s. However to be able to put the box into a circuit it contains a swap between memory and data qubit. This swap amounts to an additional left shift of

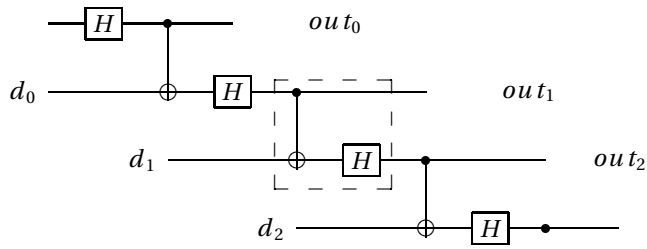


Figure 5.18: Memory efficient causal circuit to implement the shifted glider CQCA τG_s; the stroked box corresponds to one use of the memory channel implementing the CQCA. This circuit only uses one qubit of memory. However, the circuit implements an unwanted left shift and therefore G_s instead of τG_s.

the output system. Thus again we have a problem with the implementation—we actually implement G_s and not the desired causal shifted version.

The problem that the boxes do not implement τG_s as a circuit and as a memory channel at the same time comes from our attempt to use the data systems to also implement the memory system. We make room for the memory that is passed on from one box to the next by shifting the data wires. We will now explicitly add wires for the memory. When we talk about memory channels we usually think of a memory system that is passed on from one use of the channel to the next. The system itself is thought of as being the same all the time. In the circuit picture this corresponds to a wire that is at the top or bottom of the circuit. All the boxes would interact with this wire making the circuit non-local. The idea of swapping the memory through the circuit one position in each step using the data wires creates an unwanted shift as we have already seen. So we are left with either swapping the memory wire through the circuit thereby abusing the notation of the circuit model, or preparing and deleting new memory qubits in every step.

The easiest way to build a primitive for the CQCA that can both be used in the circuit picture and as a memory channel is the following: we use the box primitive from the memory efficient circuit and an ancilla qubit that we prepare in some state. After applying the gates of the box we swap memory and data output and then memory and ancilla qubit. Thus the memory qubit is on the added ancilla wire and the data qubit on the old memory output. The old data output wire is traced out. This is shown in Figure 5.19. We then concatenate the boxes using the new memory output and the old input producing a circuit that implements the CQCA and has a memory channel structure at the same time. We can further simplify the circuit and

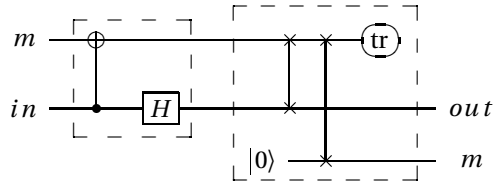


Figure 5.19: Changing the memory efficient implementation to implement the shift that was lost in the conversion from memory channel to circuit.

substitute the preparation and measuring of ancillas by crossing wires as shown in Figure 5.20. This construction is more in the spirit of memory channels than in the spirit of quantum circuits.

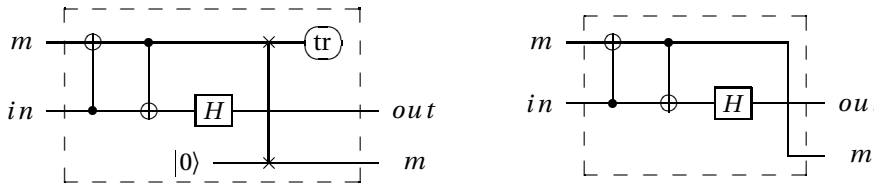


Figure 5.20: Gate efficient implementations of our primitive for τG_s; the left circuit uses preparation and measurement of ancillas and is in the spirit of the circuit model, while the right circuit abuses the circuit model notation and is in the spirit of memory channels, where the memory system is passed on from one step to the next.

Now we might ask if there is a possibility to design a circuit that implements τG_s without ancillas. We do not require the circuit to be of finite depth anymore, so implementing the shift is not ruled out by default as in the finite depth case. However, this task is still impossible. Consider a circuit on n wires. The initial memory input is on the uppermost wire. But the memory output is on the lowermost wire. So by default the data has to shift up by one wire which can not be done using local unitary operators. If we add ancillas we circumvent this problem, because the memory is always carried on the ancilla wires and shifting the memory wire through the circuit does not have an effect on the data wires. If we think of an optical lattice every second site would be a data site, while the other sites would be ancilla sites. The

ancilla sites run a backward shift to compensate for the forward shift included in the dynamics of the data qubits.

We now determine a causal up-to-shift inverse for $\tau \mathrm{G}_s$. Finding the inverse of a CQCA is straightforward by inverting the corresponding matrix. We only need an inverse up to a shift, but require the inverse to also be a causal CQCA. We obtain

$$\tau^2(\tau^{-1}\mathbf{g}_s^{-1}) = \begin{pmatrix} 1+u^{-2} & u^{-1} \\ u^{-1} & 0 \end{pmatrix}. \tag{5.43}$$

The circuit is very similar to the original CQCA. The primitives are shown in Figure 5.21

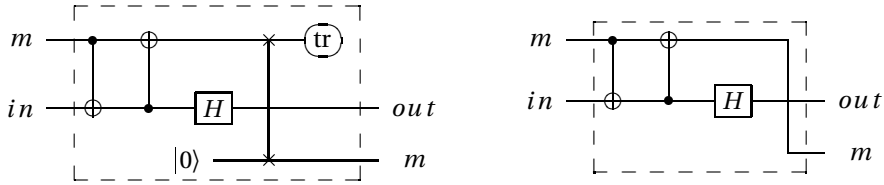


Figure 5.21: Implementations of our primitive for $\tau \mathrm{G}_s^{-1}$; the left circuit uses preparation and measurement of ancillas and is in the spirit of the circuit model, while the right circuit abuses the circuit model notation and is in the spirit of memory channels, where the memory system is passed on from one step to the next. The only difference to $\tau \mathrm{G}_s$ is that the position of the $CNOT$ gates is exchanged.

The circuit for $\tau \mathrm{G}_s$ and its causal inverse $\tau \mathrm{G}_s^{-1}$ is shown in Figure 5.22. The whole circuit implements τ^2 which is the minimal shift we can have, since $\tau \mathrm{G}_s$ has memory depth 2.

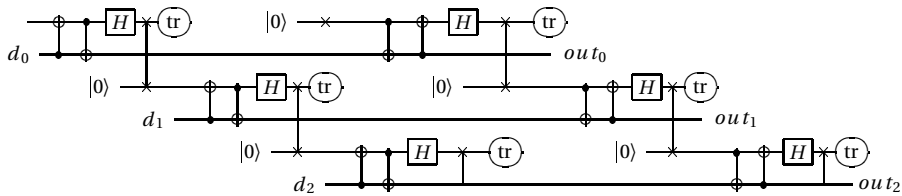


Figure 5.22: Circuit for $\tau \mathrm{G}_s$ and its causal inverse up to a shift $\tau \mathrm{G}_s^{-1}$.

6 The fractal structure of cellular automata on Abelian groups

Most of the material in this chapter has been published in:
Johannes Gütschow, Vincent Nesme, and Reinhard F. Werner
Self-similarity of cellular automata on Abelian groups,
Journal of Cellular Automata, 7(2), 2012, [arXiv:1011.0313]

In this section we will study the self-similarity of the spacetime picture of CQCAs. Our analysis does not need most of the structure underlying symplectic cellular automata. We will therefore extend our results to a wider class of linear classical cellular automata (CAs). However, our main example **f** is the classical counterpart of a CQCA introduced in Section 5.1.3. The fractal structure of cellular automata has been a topic of interest for several decades. In many works on linear CAs, the authors present ways to calculate the fractal dimension or to predict the state of an arbitrary cell at an arbitrary time step, with much lower complexity than by running the CA step by step. However, their notions of linearity are quite different to ours (and to each other). Often only CAs that use states in $\mathbb{Z}_p$ are studied. Other approaches are more general, but still make certain assumptions on the time evolution or the underlying structure of the CA. Here we try to loosen these restrictions as far as possible. We consider one-dimensional linear CAs whose alphabet is an abelian group. We will show that they can be described by $n \times n$ matrices with polynomial entries in a similar way as introduced in the special case of CSCAs. We will use this description to derive a recursion relation for the iterations of the CA. This recursion relation enables us to formulate the evolution of the spacetime diagram as a matrix substitution system, which in turn gives us the means to calculate the fractal dimension of the spacetime diagram.

Most of the methods we employ are commonly used in the study of CAs. To prove that the spacetime diagram converges we use elementary graph theory and study the graph associated with the matrix substitution system of the CA. The matrix substitution system is obtained by a recursion formula for the iterations of the CA and grouping of cells.

This Section is organized as follows: in Section 6.1 we will give our definition of a linear cellular automaton, introduce the formalism we will be working with and

state the main result: every linear cellular automaton has a self-similar structure. In Section 6.3, we will give an intuitive idea as to why the spacetime diagram of our example **f** exhibits a fractal structure. We will then proceed, in Section 6.4, to expose an algorithm taking as input the local transition rule and outputting a description of the spacetime diagram. This allows us to compute salient features of these fractals, such as their fractal dimension and their average color.

6.1 Definitions

6.1.1 Generalities on summable automata

Monoids

Now we generalize our description of CSCAs to a broader class of CAs. But we want to keep the convolutional structure so that we can use the Laurent polynomial description introduced in Chapter 5. Thus the state of a cell should be determined by the sum of the influences from other cells in the timestep before. We call automata, for which it is sensible to consider the influence of a single cell on every other cell and where the global transition function can be reconstructed by "summing" all these influences, "summable automata". So, if Σ denotes the alphabet, instead of the usual local transition function $\Sigma^{\mathcal{N}} \to \Sigma$, a summable automaton is naturally defined by a function $\Sigma \to \Sigma^{\mathcal{N}}$. We now investigate the minimal structure on Σ in order to make such a definition work. The influences from individual cells have to be "summed", so we need an operation on Σ. Since the strip is infinite, an infinitary operation would suffice, but that would not give us enough structure to work with. Instead, it seems reasonable to consider a binary operation $+$. In the same spirit, when we think of the superposition of influences coming from each cell, no notion of order between the cells is involved. Even if in the one-dimensional case a natural order could be put on the cells, it would be less than clear what to do in higher dimensions. We therefore require that $+$ be associative and commutative. The last requirement comes from the fact that, given only the global transition function, we want to be able to isolate the influence of one cell; that is why we demand that $+$ have an identity element, which makes now $(\Sigma, +)$ an Abelian monoid.[1] Of course, in order for this construction to be relevant, the transition function must be a morphism.

Let $\mathcal{N}$ be some finite subset of $\mathbb{Z}$ and f a morphism from Σ to $\Sigma^{\mathcal{N}}$. The set $\mathcal{N}$ is to the neighborhood of f. From f one can define the global transition function as an

[1] A monoid is a set equipped with an associative operation and a neutral element.

endomorphism F of $\Sigma^{\mathbb{Z}}$ by

$$F : \left(\begin{array}{ccc} \Sigma^{\mathbb{Z}} & \to & \Sigma^{\mathbb{Z}} \\ r = (r_n)_{n\in\mathbb{Z}} & \mapsto & \left(\sum_{i\in I} f(r_{n-i})_i \right)_{n\in\mathbb{Z}} \end{array} \right). \tag{6.1}$$

Let τ be the right shift on $\Sigma^{\mathbb{Z}}$, i.e. $\tau(r)_n = r_{n-1}$.[2] As in the case of QCA we assume translation invariance (of F) and have $F \circ \tau = \tau \circ F$. Also, $F(r)_n$ depends only on the values r_{n-i} for $i \in \mathcal{N}$. Since $\mathcal{N}$ is finite, F is a one-dimensional cellular automaton on the alphabet Σ, with neighborhood included in $-\mathcal{N}$. Conversely, if F is an endomorphism of $\Sigma^{\mathbb{Z}}$ defining a cellular automaton over the alphabet Σ, then one can choose a neighborhood $\mathcal{N}$, and define, for $i \in \mathcal{N}$,

$$f(s)_i = F(\bar{s})_i, \tag{6.2}$$

where $\bar{s}$ is the word of $\Sigma^{\mathbb{Z}}$ defined by

$$\bar{s}_n = \begin{cases} s & \text{if } n = 0 \\ e & \text{otherwise} \end{cases}, \tag{6.3}$$

e denoting the neutral element of Σ.

Groups

We will now consider the case when Σ is a (finite abelian) group. For p prime, let Σ_p be the subgroup of Σ that contains only elements whose order is a power of p; then Σ is isomorphic to $\prod_p \Sigma_p$, and every endomorphism of $\Sigma^{\mathbb{Z}}$ factorizes into a product of endomorphisms of the $\Sigma_p^{\mathbb{Z}}$'s (see also Section 2.2.4). It is therefore enough to study the case of the (abelian) p-groups: let us assume that Σ is a p-group (see also Theorem 2.2.8).

It is a well-known fact (see for instance section I-8 of [31]) that Σ is isomorphic to $\mathbb{Z}_{p^{k_1}} \times \mathbb{Z}_{p^{k_2}} \times \cdots \times \mathbb{Z}_{p^{k_d}}$ with $k = k_d \geqslant k_{d-1} \geqslant \ldots \geqslant k_1$. Now let us consider an endomorphism α of Σ and let e_j denote $(0,\ldots,0,1,0,\ldots,0)$, where the 1 lies in position j. When $i \leqslant j$, there is a natural embedding $s_{i,j}$ of $\mathbb{Z}_{p^{k_i}}$ into $\mathbb{Z}_{p^{k_j}}$, namely the multiplication by $p^{k_j-k_i} \in \mathbb{N}$. Since e_j has order p^{k_j}, $\alpha(e_j)_i \in \mathbb{Z}_{p^{k_i}}$ has to be in the image of $s_{j,i}$ when $i \geqslant j$. We can therefore associate to α the endomorphism of $\mathbb{Z}_{p^k}^d$ given by the matrix $A(\alpha) \in \mathscr{M}_d(\mathbb{Z}_{p^k})$ defined by

$$A(\alpha)_{i,j} = p^{k_j-k_i}\alpha\left(e_j\right)_i. \tag{6.4}$$

[2] As we deal with classical CAs here, the shift direction is inverted with respect to the quantum case, where τ shifts observables to the left and states to the right. Here, the classical configuration is treated as a state.

For example, if we have $G = \mathbb{Z}_{32} \times \mathbb{Z}_4 \times \mathbb{Z}_2$ and α defined by $\alpha(1,0,0) = (3,2,1)$, $\alpha(0,1,0) = (24,0,1)$, and $\alpha(0,0,1) = (16,2,0)$, the corresponding matrix of $\mathscr{M}_3(\mathbb{Z}_{32})$ is

$$A(\alpha) = \begin{pmatrix} 3 & 3 & 1 \\ 16 & 0 & 1 \\ 16 & 2 & 0 \end{pmatrix}.$$

Moreover, $\alpha \mapsto A(\alpha)$ is an embedding of $\mathbb{Z}_{p^k}$-algebras, between the algebra of linear endomorphisms of Σ and $\mathscr{M}_d\left(\mathbb{Z}_{p^k}\right)$. Indeed, it is clearly linear, one-to-one, and

$$\begin{array}{rcl} \left[A(\alpha)A(\beta)\right]_{i,j} & = & \sum_{l=1}^{d} A(\alpha)_{i,l} A(\beta)_{l,j} \\ & = & \sum_{l=1}^{d} p^{k_l - k_i} \alpha(e_l)_i p^{k_j - k_l} \alpha(e_j)_l \\ & = & p^{k_j - k_i} \sum_{l=1}^{d} \alpha(e_l)_i \beta\left(e_j\right)_l \\ & = & p^{k_j - k_i} \left[\alpha \circ \beta\left(e_j\right)\right]_i \\ \left[A(\alpha)A(\beta)\right]_{i,j} & = & A(\alpha \circ \beta)_{i,j}. \end{array} \tag{6.5}$$

Let us give a summary of the construction we have just exposed.

Proposition 6.1.1. *For every finite abelian p-group G and endomorphism α of G, there are positive integers k and d, an embedding s of G into $\mathbb{Z}_{p^k}^d$, and an endomorphism $A(\alpha)$ of $\mathbb{Z}_{p^k}^d$ such that the following diagram commutes:*

$$\begin{array}{ccc} G & \xrightarrow{\alpha} & G \\ \downarrow s & & \downarrow s \\ \mathbb{Z}_{p^k}^d & \xrightarrow{A(\alpha)} & \mathbb{Z}_{p^k}^d \end{array} \tag{6.6}$$

This implies that to study the behavior of CAs on abelian groups, it is sufficient to study the case where these groups are of the form $\mathbb{Z}_{p^k}^d$.

R-modules

We now actually consider the more general case where R is a finite commutative ring with characteristic p^l where p is prime, and Σ is a free R-module of dimension d, i.e. isomorphic to R^d.[3] The first reason for doing so is that it does not complicate the mathematics. It will also appear more efficient to understand, for instance, $\mathbb{F}_{2^4}$ as a 1-dimensional vector space over itself than as a 4-dimensional vector space

[3] Modules are generalized of vector spaces, where rings take the place of fields. A free module is a module with a basis.

over $\mathbb{F}_2$: the former simply bears more information, and therefore implies more restrictions on the form of a CA, so that more can be deduced.

Now we generalize the algebraic Fourier transformation (2.56) we used in the case of CSCAs to our new setting and introduce a polynomial description of the CAs we are considering here. Analogous to the CQCA case, for any ring B, $B\left[u,u^{-1}\right]$ denotes the ring of Laurent polynomials over B; it is the ring of linear combinations of integer powers (negative as well as nonnegative) of the unknown u. Applying this to $B=\mathrm{Hom}_R(\Sigma)$, we can associate to the function f the Laurent polynomial $\iota(f)\in\mathrm{Hom}_R(\Sigma)\left[u,u^{-1}\right]$[4] defined by

$$\iota(f)=\sum_{n\in\mathbb{Z}} f(\cdot)_n u^n. \tag{6.7}$$

Remark 6.1.2. *In the manner of a Fourier transform, ι turns convolution into product. Indeed, looking at Equation (6.1), if we think of r as a function from $\mathbb{Z}$ to Σ, F is easily seen to be a convolution of r and f. Since f is R-linear, $f(r(n))_i=r(n)f(1)_i$ holds. Now we introduce $\tilde{f}$ with $\tilde{f}(i)=f(1)_i$. Then*

$$(F(r))(n)=\sum_{i\in\mathbb{Z}} r(n-i)\tilde{f}(i) \tag{6.8}$$

holds. In the polynomial picture this convolution turns into a matrix-vector multiplication.

ι is an isomorphism of R-algebras between the linear cellular automata on the alphabet Σ with internal composition rules $(+,\circ)$ and $\mathrm{Hom}_R(\Sigma)\left[u,u^{-1}\right]$, which can be identified with $\mathscr{M}_d\left(R\left[u,u^{-1}\right]\right)$ because $\Sigma\simeq R^d$. We are going to think and work in this former algebra, so from now on a linear cellular automaton $\boldsymbol{t}=\iota(f)$ will be, for us, an element of $\mathscr{M}_d\left(R\left[u,u^{-1}\right]\right)$.

6.1.2 Related work

Many papers have been published concerning the self-similarity and fractal structure of cellular automata spacetime diagrams. Here we give a short review and point out the differences in our approach. When we mention d and k we are referring to $\mathscr{M}_d\left(\mathbb{Z}_k\left[u,u^{-1}\right]\right)$.

[96] In this work, Willson considers the case $d=1$, $k=2$. In order to determine the fractal dimension of the spacetime diagram, he analyses how blocks of length n in the configuration of time step t are mapped to such blocks in steps $2t$ and $2t+1$, a technique we also use in Section 6.4.

[97, 98] Takahashi generalizes Willson's work to the case $d=1$, with no restriction on the value of k.

[4] $\mathrm{Hom}_R(\Sigma)$ denotes the set of all R-linear maps on Σ.

[99, 100, 101] Haeseler, Peitgen and Skordev study the fractal time evolution of CAs with special scaling properties, the weakest of them being "weakly p-Fermat", where p is some integer, which includes the case $d = 1$, $k = p$. Let us briefly introduce the p-Fermat property and show why the CAs that we study do not have to be p-Fermat. Let π_p be the scaling map

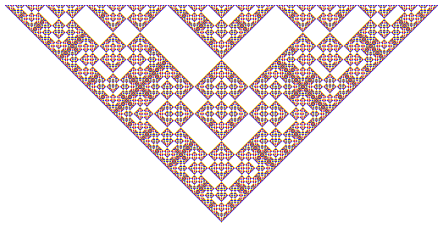
$$\pi_p(\xi)_x = \begin{cases} \xi_y & \text{if } x = py \\ e & \text{otherwise} \end{cases}. \tag{6.9}$$

A CA $\boldsymbol{t}$ is weakly p-Fermat if for all $s \in \Sigma$, $n \in \mathbb{N}$ and $x \in \mathbb{Z}$, $\boldsymbol{t}^{np}(\bar{s})_x = e \Leftrightarrow \pi_p \boldsymbol{t}^n(\bar{s})_x = e$.

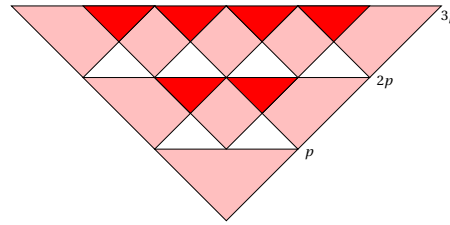
Let us now consider the fractal CSCA **f** introduced in 5.1.3

$$\mathbf{f} = \begin{pmatrix} 0 & 1 \\ 1 & u^{-1} + 1 + u \end{pmatrix} \in \mathscr{M}_2(\mathbb{Z}_2[u, u^{-1}]). \tag{6.10}$$

We will use this example throughout the section. It generates the time evolution depicted in Figure 6.1a (and in more detail in Figure 5.2). A general



(a) Time evolution of **f** starting from $\xi = \binom{1}{0}$.



(b) Time evolution of a general nearest neighbor p-Fermat CA.

Figure 6.1: This figure shows that **f** cannot be a p-Fermat CA. In a p-Fermat CA at least the white areas are filled by the neutral element e; **f** has a different pattern.

nearest-neighbor p-Fermat CA produces a time evolution that reproduces itself after p steps in at most three copies located at positions $\{-p; 0; p\}$. After $2p$ steps we have five copies at most. This creates areas filled with the neutral element e shared by all p-Fermat CAs for a fixed p. In Figures 6.1a and 6.1b we can easily see that **f** does not exhibit these areas; therefore it is not p-Fermat. Furthermore p-Fermat CAs that are not periodic are irreversible, while we also allow reversible CAs, **f** again being an example.

[102, 103] Allouche, Haeseler, Peitgen and Skordev study recurrences in the space-time diagram of linear cellular automata from the angle of k-automatic sequences (which we will not define here). However, they require Σ to be an

Abelian ring and the CA to be a ring homomorphism, which is again essentially the case $d = 1$.

[104, 105] Moore studies CAs with an alphabet A on a staggered spacetime, where every cell c is only influenced by two cells a and b of the last time step. The update rule is $c = a \bullet b$. He requires $(A, \bullet)$ to be a quasigroup and studies different special cases. First let us note that these CAs are either irreversible or trivial, while ours do not have to be. Thus, although it is possible to bring our CAs in the form of staggered CAs, the results of Moore do not apply. In his setting, our CAs would be of the form $c = a \bullet b = f(a) + g(b)$ for some homomorphisms f and g. For $(A, \bullet)$ to be a quasigroup means

$$\forall a, b \in A \; \exists! x, y \in A \quad a \bullet x = b \; \wedge \; y \bullet a = b. \tag{6.11}$$

In our case, these equalities translate respectively as $g(x) = b - f(a)$ and $f(y) = b - g(a)$. The right-hand sides can be arbitrary elements of A, therefore f and g have to be isomorphisms, as indeed required in [104].

The goal of Moore's work is different as well: he does not directly study the fractal properties of the spacetime diagram, but rather the complexity of the prediction—"What will be the state of this cell after t steps?". Describing the spacetime diagram with a matrix substitution system is one way of proving that prediction is an easy task—for instance it makes it **NC**.[5]

[106] Macfarlane uses Willson's approach and generalizes parts of it to some examples of matrix-valued CAs, including **f**. However, the transition matrix is obtained heuristically—"by scrutiny of Figure 9"—from the spacetime diagram, instead of being algorithmically derived from the transition rule (as we do here). The conclusion (Section 6) suggests that the analysis of **f** is easily generalizable to matrices of various sizes over various rings. So, in a sense, our analysis is but an elaboration of the concluding remark of [106], although much work is needed to actually perform the suggested generalizations.

The heart of our proof is in Section 6.4. To prove self-similarity, we want to establish a scaling property on the spacetime diagram of our CA. What we would like to prove is that for some fixed integer m, for every $i, j \in [\![0; m-1]\!]$[6], the state of a cell $m \cdot x + i$ at time $m \cdot y + j$ depends only of that of the cell x at time y. This would show a self-similarity in diagrams of order m^n when n goes to infinity. Unfortunately, that statement is not true. However, the alphabet can be expanded and the transition rule extended[7] in such a way that the desired property is now true.

[5] **NC** is the class of of decision problems that is decidable in polylogarithmic time using a parallel computer with a polynomial number of processors.

[6] $[\![a; b]\!]$ denotes the integer interval $\{a, a+1, \ldots, b\}$.

[7] Beware that the extended "transition rule" is actually not a CA.

This is done by the introduction of α in Equation (6.25). It can also be noted that the first thing we do in Section 6.4 is getting rid of the complicated multiplication of matrices and go back to a simple linear recurrence, as stated in Proposition 6.4.1.

6.2 A visit to the zoo of linear CA

We use **f** as our example throughout the chapter. Despite the special properties it has, i.e. it is reversible, defined over a field of characteristic two, and described by a 2×2 matrix, the analysis applies of course to all other linear CAs. In this section we give a short overview over the variety of spacetime diagrams these CAs generate. Let us start with small changes to **f**. For our first example we keep $m = k = 2$, but change the determinant to u. We only change one entry of the matrix:

$$\mathbf{f}_u = \begin{pmatrix} 0 & u \\ 1 & u^{-1} + 1 + u \end{pmatrix}. \tag{6.12}$$

The spacetime diagram is displayed in Figure 6.2 and shows how much difference a small change in the update rule can make for the spacetime diagram.

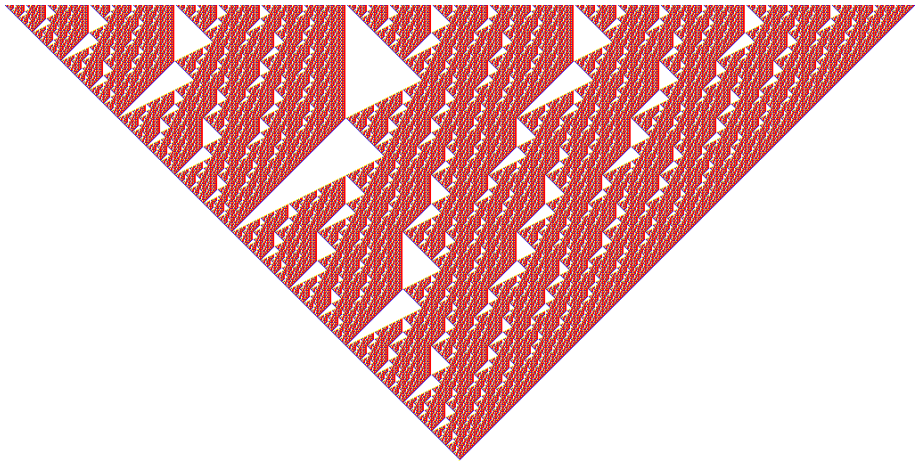


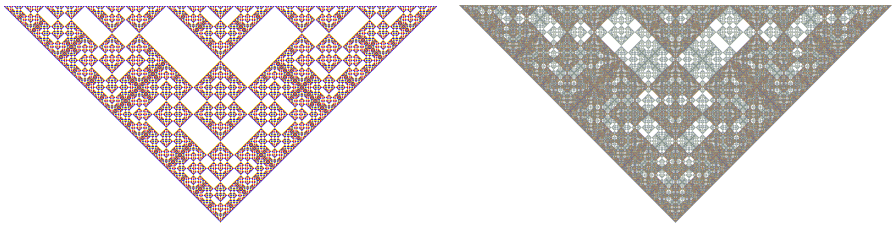
Figure 6.2: Spacetime diagram of the non-Clifford CA $\mathbf{f}_u$

Let us now modify **f** in a more subtle fashion:

$$\mathbf{f}_{k=4} = \begin{pmatrix} 0 & 1 \\ 1 & u^{-1} + 1 + u \end{pmatrix}. \tag{6.13}$$

The hidden difference with **f** is the underlying ring, which has now been extended from $\mathbb{Z}_2$ to $\mathbb{Z}_4$. The CA $\mathbf{f}_{k=4}$ contains in some sense more information than **f**, since

$\mathbf{f}$ is induced from $\mathbf{f}_{k=4}$ by the projection $\mathbb{Z}_4 \twoheadrightarrow \mathbb{Z}_2$. Consequently, the spacetime diagram of $\mathbf{f}$ is nothing but a projection of that of $\mathbf{f}_{k=4}$, as illustrated in Figure 6.3b.



(a) Time evolution of $\mathbf{f}$ with scalars in $\mathbb{Z}_2$

(b) Time evolution of $\mathbf{f}$ with scalars extended to $\mathbb{Z}_4$

Figure 6.3: Spacetime diagrams for two modifications of $\mathbf{f}$; (a) is a projection of (b) induced by $\mathbb{Z}_4^2 \twoheadrightarrow \mathbb{Z}_2^2$.

Even more striking is the spacetime diagram of $\mathbf{f}_{k=3}$, a CA that only differs from $\mathbf{f}$ in the use of $\mathbb{Z}_3^2$ instead of $\mathbb{Z}_2^2$ as the alphabet.

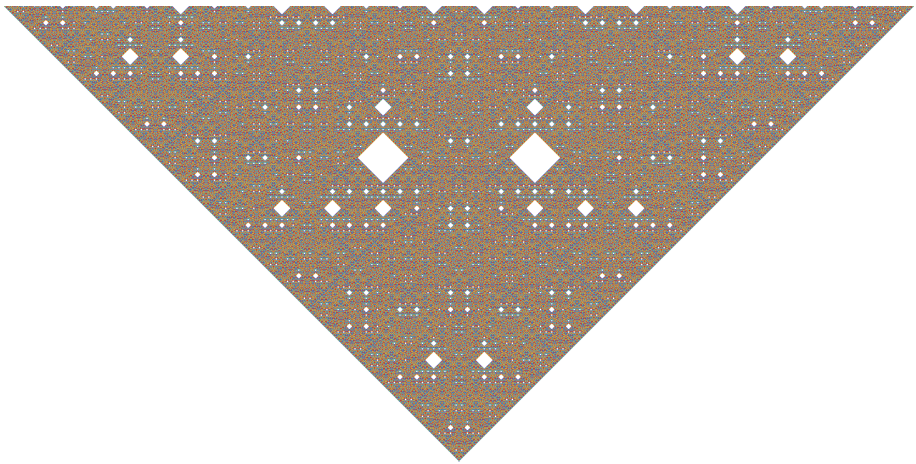


Figure 6.4: Spacetime diagram of $\mathbf{f}_{k=3}$

The last CA we want to present lies in $\mathscr{M}_2\left(\mathbb{F}_4[u,u^{-1}]\right)$, where $\mathbb{F}_4$ is the finite field of order 4, here identified with $\mathbb{F}_2[\omega]/(\omega^2+\omega+1)$. The corresponding matrix is

$$\boldsymbol{t}_{\mathbb{F}_4} = \begin{pmatrix} 0 & \omega \\ u^{-1} & (\omega+1)u^{-1}+\omega+u \end{pmatrix}. \tag{6.14}$$

If one wants to avoid calculations in $\mathbb{F}_4$, this CA can also be translated to a CA in

$\mathscr{M}_4(\mathbb{Z}_2[u,u^{-1}])$, namely

$$\tilde{\boldsymbol{t}}_{\mathbb{F}_4} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ u^{-1} & 0 & u^{-1}+u & u^{-1}+1 \\ 0 & u^{-1} & u^{-1}+1 & 1+u \end{pmatrix}.$$

Its spacetime diagram, as can be seen in Figure 6.5, contains patches of checker-

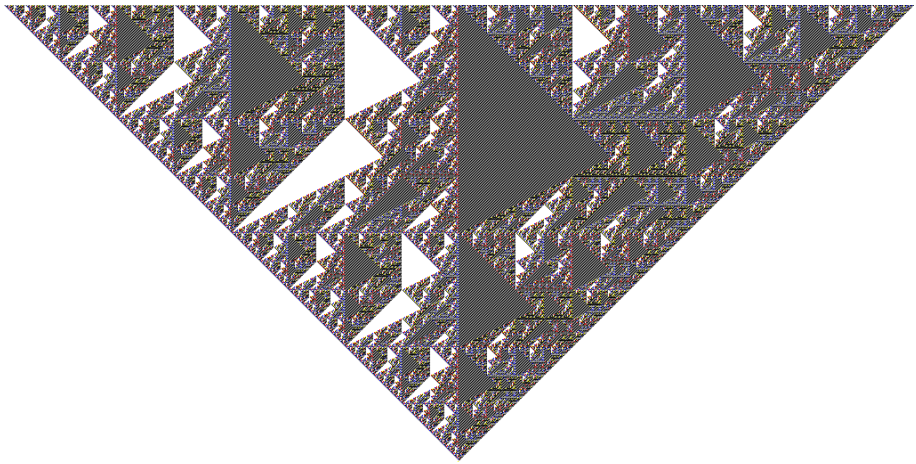


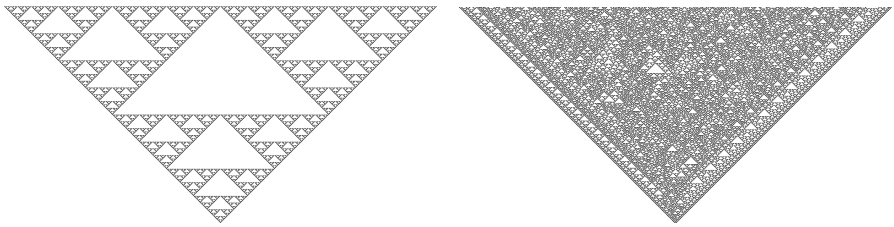
Figure 6.5: Time evolution of $\boldsymbol{t}_{\mathbb{F}_4}$; the gray triangles indicate that the fractal dimension is 2. However, the spacetime diagram can hardly be considered trivial.

board pattern. Somehow, they trivialize most of the usual properties of the figure: for instance they make its fractal dimension 2, even if the fractal structure can hardly be considered trivial. In order to access more interesting properties, it is possible to blank this pattern out, considering it as just "another shade of white". This can be trivially done on the matrix substitution system, by removing the states from which the blank state is inaccessible.

6.2.1 Why linearity?

In our analysis we have only covered linear CAs so far. While our method of using matrices to describe the CAs only applies to linear CAs one could of course analyze non-linear CAs using different methods. However, in general, non-linear CAs do not produce self-similar spacetime diagrams. Furthermore, the time-evolution largely depends on the initial state. It is no longer a superposition of the shifted

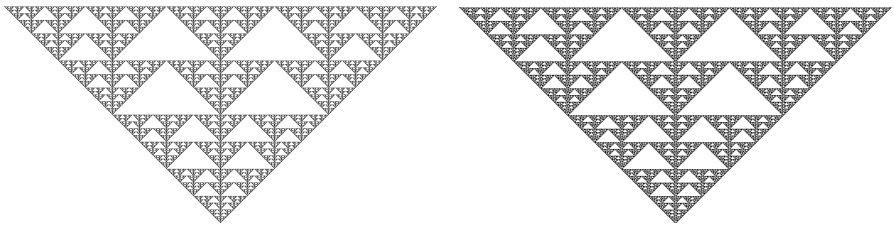
spacetime images of one-cell initial states as in the linear case, but it can be totally different. In Figure 6.6 we demonstrate this using the non-linear CA named *rule 22*, according to its Wolfram code. Here even the self-similarity is lost. Rule 150 is close



(a) Initial state 1. (b) Initial state 10011.

Figure 6.6: Time evolution of the 1D CA with Wolfram rule 22; on the left hand side a single cell initial state is used. On the right hand side we use a multi cell initial state. As rule 22 is non-linear the right hand side is not a superposition of the one cell spacetime diagram. It is not even self-similar.

to rule 22, in the sense that their local transition functions differ in only one case, but it is linear. Figure 6.7 illustrates that the spacetime diagram emerging from several nonzero cells is indeed the superposition of the shifted single initial cell spacetime images and the spacetime image is always self-similar. In our analysis the ini-



(a) Initial state 1. (b) Initial state 10011.

Figure 6.7: Time evolution of the 1D CA with rule 150; on the left hand side a single cell initial state is used. On the right hand side we use a multi cell initial state. The spacetime diagram is self-similar in both cases.

tial state is irrelevant for the substitution scheme as long as it is sufficiently small; only the coloring depends on it. However, large initial states are not covered by the substitution system we derive. But as the CAs we deal with are linear, the spacetime image of an arbitrary (but finite) initial state is nothing but a superposition of single cell spacetime images. So for large times the spatial size of the blank spaces of all the single initial cell spacetime images will be much larger than the size of the

initial state and the superposition will have the same blank spaces. However, in the non-blank areas cancellation may occur. The spacetime image of an arbitrary initial cell thus converges to a subset of the image derived by the substitution system. A class of CAs that show excessive cancellation are the glider CQCAs which act as a lattice translation on some initial states (the gliders) while for all single cell initial states they produce a spacetime image of fractal dimension 2 (see Section 5.1.2). For non-linear CAs we can not deduce much about the image of multi cell initial states just from the single cell images. In particular, the self-similarity depends on the initial state.

6.2.2 Colored spacetime diagrams

The mainstream setting when studying the fractal structure of spacetime diagrams is monochromatic; we introduce colors in the picture.

Instead of considering simple compact subsets of the plane, we will have a finite set of colors $\mathscr{C}$ and compact subsets of $(\mathbb{R}^2)^{\mathscr{C}}$. Let $b \notin \mathscr{C}$ be the additional "blank" color and $\mathfrak{c} : \Sigma \to \mathscr{C} \cup \{b\}$ a coloring of Σ such that $\mathfrak{c}(0) = b$. To determine a *colored spacetime diagram*, we need, furthermore, to be given an automaton $\boldsymbol{t} \in \mathscr{M}_d\left(R\left[u, u^{-1}\right]\right)$, an initial state $\xi \in R^d$, and an integer n. The corresponding colored spacetime diagram is then the rescaled diagram obtained by iteratively applying $\boldsymbol{t}$ n times on ξ.

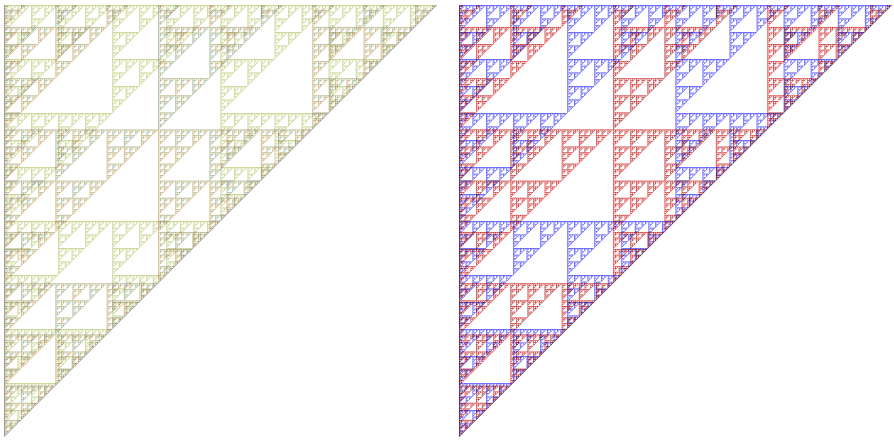
Formally, for $n, i, j \in \mathbb{N}$, let $S_{n,i,j}$ be the full square centered in $\frac{1}{n}(i, j)$ and whose edges, parallel to the axes, are of length $\frac{1}{n}$. To each positive integer n and color $c \in \mathscr{C}$ we associate a compact subset of the plane $\mathscr{P}_n(c)$ which is the union of the $S_{n,i,j}$'s such that $0 \leqslant j \leqslant n$ and $\mathfrak{c}\left(\boldsymbol{t}^j(\xi)_i\right) = c$. The colored spacetime diagram of order n is then the function $\mathscr{P}_n : c \mapsto \mathscr{P}_n(c)$. A sequence of colored patterns $(\mathscr{P}_n)_{n\in\mathbb{N}}$ of spacetime diagrams is said to converge to some colored pattern $\mathscr{P}_\infty$ if for every $c \in \mathscr{C}$, $(\mathscr{P}_n(c))_{n\in\mathbb{N}}$ converges to $\mathscr{P}_\infty(c)$ for the Hausdorff distance.

We can now state the main result of this chapter.

Theorem 6.2.1. *Let G be a finite abelian p-group. For every cellular automaton over G that is also a group homomorphism, there exists a positive integer m such that for every fixed initial state the colored spacetime diagrams of order p^{mn} converge when n goes to infinity.*

In general, to obtain information about the fractal structure of a cellular automaton over some finite group G, one must write G as a product of p-groups and study each p-component of the spacetime diagram independently. According to Theorem 6.2.1, each component generates a fractal pattern. Since the logarithms of the prime numbers are rationally independent, it is possible to find a sequence of resized spacetime diagrams that converges towards a superposition of these different components with arbitrary independent rescaling coefficients. However, there is

no direct generalization of the theorem. For instance, even in the simple case of Pascal's triangle modulo 6, there is no real number $\lambda > 0$ such that the diagrams of order $\lfloor \lambda^n \rfloor$ converge; however, those of order t_n will converge as soon as the fractional parts of $\log_3(t_n)$ and $\log_2(t_n)$ both converge, and then their limits determine the limit pattern. The situation is very briefly described in Section 5 of [98]. We illustrate this in Figure 6.8 and in a video to be found in [107].



(a) Pascal's triangle modulo 6

(b) Superposition of Pascal's triangle modulo 2 (blue) and modulo 3 (red)

Figure 6.8: One can easily see that Pascal's triangle mod 6 is a superposition of Pascal's triangle mod 2 and mod 3.

6.2.3 Matrix substitution systems

We will show how to find a suitable description of the limit pattern in the rest of this chapter. We now explain exactly what it means to generate a colored picture by rules of substitution, and how to take the limit of all these pictures. This is a generalization of the usual monochromatic description that can be found for instance in [108, 96, 99], and which corresponds in our setting to the case where all the colors are mapped to "black".

Let V be a finite alphabet; because we want colors, we do not have to include a special "empty" letter as in the usual definition of a matrix substitution system. A matrix substitution system is then a function $\mathscr{D} : V \to V^{[\![1;r]\!]^2}$, for some integer r. Together with a set of colors $\mathscr{C}$ and a coloring $\mathfrak{c} : V \to \mathscr{C}$, the matrix substitution system defines colored patterns, much in the same way cellular automata do. With the previous notations, at each step n, the pattern $\mathscr{P}_n$ is the union of squares $S_{r^n,i,j}$

of different colors, for different i's and j's; each one of them is indexed by some letter in V.

Then at step $n+1$, each colored square of color c indexed by $v \in V$ present in the nth step pattern is replaced by r^2 smaller squares that pave it; these smaller squares are given by $\mathscr{D}(v)$ and indexed accordingly. To such a matrix substitution system we can associate a multigraph $\Gamma = (V, E)$ where the set of vertices is V and we put as many edges from v to w as there are w's in $\mathscr{D}(v)$.

A *plain* matrix substitution system is one of the usual kind: there are only two colors W and B, and V contains a special letter ε such that $\mathscr{D}(\varepsilon)$ is a matrix full of ε's and $\mathfrak{c}(\varepsilon) = b$. We have $\mathfrak{c}(\varepsilon) = W$ and $\mathfrak{c}(v) = B$ whenever $v \neq \varepsilon$. In this case it is well known that, starting from any letter except ε, $(\mathscr{P}_n(B))_{n\in\mathbb{N}}$ converges.

We want to generalize the usually property of convergence of the patterns defined by plain matrix substitution systems. This will be done by the conjunction of the two following propositions. Let us first remind ourselves of some notions on graphs: a *cycle* is a set of nodes that are connected by edges such that the first and the last node of the path are the same. The *period* of a graph Γ is the greatest common divisor of the lengths of all the cycles in Γ; a graph is *aperiodic* if it has period 1. A *strongly connected component* is a set of nodes such that there is a path from every node of the strongly connected component to any other node of the strongly connected component.

Proposition 6.2.2. *If every strongly connected component of Γ is aperiodic, then $(\mathscr{P}_n)_{n\in\mathbb{N}}$ converges.*

Proof. To each color $c \in \mathscr{C}$ we associate the plain matrix substitution system $\mathscr{D}^c$, obtained from $\mathscr{D}$ simply by turning some letters into ε. For $v \in V$, let $X_c(v)$ be the set of integers n such that there exists a path of length n in Γ connecting v to a letter of the color c. Since the strongly connected component containing v is aperiodic, $X_c(v)$ is either finite or cofinite. To see this we have to consider different cases:

- There is no letter of color c reachable from v. Then $X_c(v)$ is empty and thus finite.
- There is a letter of color c reachable from v. Let us first assume there exists a path from v to this letter that contains a strongly connected component with more than one letter. In this component cycles of coprime lengths exist. Thus the path from v to the letter of color c can be extended by sums of the length of the cycles. The question posed is "which path length can be composed". It is equivalent to the coin problem (also known as the Frobenius problem). This tells us that there is a maximum number, the Frobenius number of the set of cycle length, that can not be composed by combinations of the cycles. Thus all but a finite number of path length can be achieved and $X_c(v)$ is cofinite. If there exists no path of the type in our first assumption, but a path that

uses a letter with a self loop, $X_c(v)$ is again cofinite, because we can generate arbitrary path length using the self loop. The only remaining case is that all possible paths only use letters in components with only one letter. Then there are no cycles and thus only finitely many possible path lengths (the alphabet is finite), making $X_c(v)$ finite.

Those letters $v \in V$ such that $X_c(v)$ is finite will be mapped to a combination of letters that will not contain the color c after finitely many steps and for all further recursion steps. Thus they act as the blank state of a plain matrix substitution system and we send them to ε. This defines $\mathscr{D}^c$. If $X_c(v)$ is finite and v' can be reached from v, then $X_c(v')$ is also finite; therefore, $\mathscr{D}^c$ is indeed a substitution system.

Let us now compare two sequences of figures. The first one is $(\mathscr{P}_n(c))$, the subpattern of color c defined by $\mathscr{D}$. The second one is $\left(\mathscr{P}_n^c\right)$, the one obtained from $\mathscr{D}^c$; we know that it converges to some compact $\mathscr{D}^c_{\mathscr{O}}$. Let $M \in \mathbb{N}$ be such that for every $v \in V$, either $X_c(v)$ or its complement is strictly bounded by M. By construction, $\mathscr{P}_{n+M}(c)$ is included in $\mathscr{P}_n^c$, and for every black square of $\mathscr{P}_n^c$, there is a black subsquare in $\mathscr{P}_{n+M}(c)$.[8] The Hausdorff distance between $\mathscr{P}_{n+M}(c)$ and $\mathscr{P}_n^c$ therefore converges to 0, so that $(\mathscr{P}_n(c))$ converges to $\mathscr{D}^c_{\mathscr{O}}$. □

For a graph Γ, let $\Gamma^k = (V, E^k)$ where E^k is the set of couples (v, w) such that there exists in Γ a path of length k from v to w.

Proposition 6.2.3. *For every (multi)graph Γ, there exists k such that every strongly connected component of Γ^k is aperiodic.*

Proof. Each strongly connected component Δ of Γ has a period $p(\Delta)$, so that $\Delta^{p(\Delta)}$ is aperiodic. Let k_0 be the least common multiple of the $p(\Delta)$'s; then each strongly connected component of Γ induces an aperiodic graph in Γ^{k_0}. However, it is possible that in the process, the component broke down into several connected components, so that Γ^{k_0} might not have the required property. The procedure then has to be repeated from Γ^{k_0} to obtain $\Gamma^{k_0 k_1}$, and so on. Since the strongly connected components of $\Gamma^{k_0 \cdots k_{i+1}}$ are included in those of $\Gamma^{k_0 \cdots k_i}$, this process reaches a fixed point, which is a graph with the required property. □

Therefore, a colored matrix substitution system defines a convergent colored pattern when considering the steps that are a multiple of some well-chosen integer m. So, in order to prove Theorem 6.2.1, all we need to do is to find such a substitution system. This will be done for a special case in the next section, and for the general case in Section 6.4.

[8] Because M bounds $X_c(v)$ if $|X_c(v)|$ is finite, all the states that are blanked in $\mathscr{D}^c$ are also blank. If $|X_c(v)| = \infty$ then M bounds $\overline{X_c(v)}$ and we have a c in the image for sure.

6.3 A special recursion scheme for f

The aim of this section is to give our most direct and natural explanation of the fractal structure generated by **f**. Modulo some caveat, it applies effortlessly to all invertible elements $\boldsymbol{t}$ of $\mathscr{M}_2(R[u,u^{-1}])$, where R is a finite Abelian ring of characteristic 2. This section is not vital to the proof of the general case presented in 6.4, and can therefore be skipped by the impatient reader.

We will deduce, informally, the basic structure of the spacetime diagrams from a simple recursion relation for the 2^nth powers of $\boldsymbol{t}$. The characteristic polynomial of $\boldsymbol{t}$, $P_{\boldsymbol{t}}(X)$, is equal to $X^2 + (\operatorname{tr}\boldsymbol{t}) + \det\boldsymbol{t}$. According to the Cayley-Hamilton theorem, $P_{\boldsymbol{t}}(\boldsymbol{t}) = 0$, so $\boldsymbol{t}^2 + (\operatorname{tr}\boldsymbol{t})\,\boldsymbol{t} + (\det\boldsymbol{t})\,\mathbb{1} = 0$. Multiplying this equation by $\boldsymbol{t}^{-1}$, we have $\boldsymbol{t} = (\det\boldsymbol{t})\,\boldsymbol{t}^{-1} + (\operatorname{tr}\boldsymbol{t})\,\mathbb{1}$. Let us denote $\widetilde{\boldsymbol{t}} = (\det\boldsymbol{t})\,\boldsymbol{t}^{-1}$, which we will name the *dual* of $\boldsymbol{t}$. Since we are in characteristic 2, by repeatedly taking the square of this equality, we obtain

$$\boldsymbol{t}^{2^n} = \widetilde{\boldsymbol{t}}^{2^n} + (\operatorname{tr}\boldsymbol{t})^{2^n}\,\mathbb{1}\ \forall n \in \mathbb{N}. \tag{6.15}$$

Taking the trace of this equation, we have $\operatorname{tr}\boldsymbol{t}^{2^n} = \operatorname{tr}\widetilde{\boldsymbol{t}}^{2^n}$; in particular, $\operatorname{tr}\boldsymbol{t} = \operatorname{tr}\widetilde{\boldsymbol{t}}$ so Equation (6.15) is also valid when swapping $\boldsymbol{t}$ and $\widetilde{\boldsymbol{t}}$. Let $\mathcal{N}_{\boldsymbol{t}}$ be a finite set, and the λ_i's elements of R such that $\operatorname{tr}\boldsymbol{t} = \sum_{i\in\mathcal{N}_{\boldsymbol{t}}} \lambda_i u^i$. Then we have

$$(\operatorname{tr}\boldsymbol{t})^{2^n} = \sum_{i\in\mathcal{N}_{\boldsymbol{t}}} (\lambda_i)^{2^n}\, u^{2^n i}\ \forall n \in \mathbb{N}. \tag{6.16}$$

We do not yet specify the initial state; as a matter of fact, it will prove to be largely irrelevant. The only thing we require is that it is nontrivial (and finite).

Let us now consider **f**; we have $\det\mathbf{f} = 1$ and $\lambda_i = \chi_{\{-1;0;1\}}(i)$. We start with the spacetime diagram corresponding to 2^n steps; it is rescaled to a triangle with vertex coordinates $\{(0,0),(-1,1),(1,1)\}$. Taking Equations (6.15) and (6.16), we can see that the state at the 2^nth time step can be decomposed into a sum of several copies of the initial state (the positions are governed by the coefficients of the trace) and a configuration that can be derived by applying $\widetilde{\boldsymbol{t}}^{2^n}$ to the initial state. In the next 2^n steps, this configuration will contract itself to the initial state, which is shifted according to $\log_u \det\boldsymbol{t}$, as $\widetilde{\boldsymbol{t}}$ is the inverse of $\boldsymbol{t}$ composed with the shift $(\det\boldsymbol{t})\mathbb{1}$. The copies of the original initial state evolve according to $\boldsymbol{t}$. This is illustrated in Figure 6.9. The figure suggests to divide the spacetime diagram into four parts A, B, C, and D, as shown in Figure 6.10a, which overlap only on a single cell strip at the borders.

A_2, A_3, and A_4 are copies of A_1, and V is marked by an upside down A because it is the reverse evolution of the initial state under the CA $\widetilde{\mathbf{f}}$—so it should logically be named $\widetilde{V}$ or $\overset{V}{\sim}$, but since it always appears upside-down while A always appears straight on its feet, there is no risk of confusion.

Let us assume that the sequence of rescaled spacetime diagrams up to step 2^n actually converges. This means that A_1 should be, in the limit, a copy of the whole

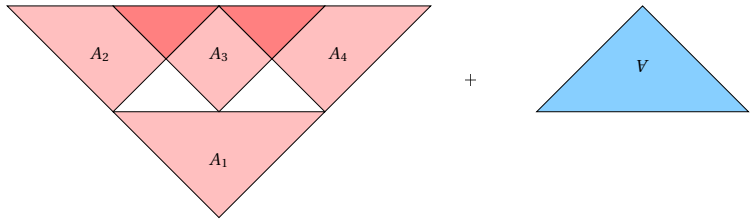
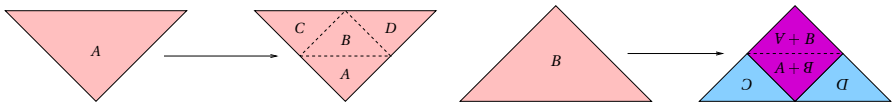


Figure 6.9: The whole figure is the sum of $|\mathcal{N}|+2$ parts.



(a) The first substitution rule.

(b) The second substitution rule.

Figure 6.10: The first and the second substitution rules

picture A, downsized by a factor 2. Therefore we rename it A. This gives us the first substitution rule, represented in Figure 6.10a. The other three parts are still unknown, and we will name these patterns B, C and D. Equation 6.15 tells us what the other substitution rules are.

Remark 6.3.1. *Indeed, the whole figure A should be the "sum" of five tiles. What this means exactly is largely informal to this point, but if you look at iterations to 2^n for a large n, this sum would be just the usual sum in $(\mathbb{Z}/2\mathbb{Z})^2$. The first tile, A_1, comes from our assumption that indeed the sequence of pictures converges. The three tiles A_2, A_3 and A_4 correspond respectively to the monomials of trace u^{-1}, 1 and u. Indeed, according to Equation 6.15, the configuration obtained after 2^n iterations of $\mathbf{t}$ is the sum of the configuration obtained after 2^n iterations of $\mathbf{t}^{-1}$ and of $|\mathcal{N}|$ copies of the initial configuration. Each of these copies will give rise to a pattern equal to A.*

Since Equation 6.15 remains true after swapping $\boldsymbol{t}$ and $\widetilde{\boldsymbol{t}}$, V likewise admits a partition into $\mathcal{V}$, $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$. Summing all the parts shown in Figure 6.9, we have Figure 6.14. Superimposing our first substitution rule (Figure 6.10a) with our second step of the decomposition (Figure 6.11), we obtain the new substitution rules represented by Figures 6.10b, 6.12a and 6.12b.

All the other substitution rules are deducible from these ones, because they are linear: for instance the substitutions for $C+D$ is the sum of the substitution for C and D, as shown in Figure 6.13a. We don not know *a priori* what the sum of two patterns is, but we know that summing a pattern with itself should give 0, for we are working in characteristic 2. In this case $A+A$ and $B+B$ cancel out. In Figure 6.14, one can see how the characteristic white spaces emerge from the above substitution rule.

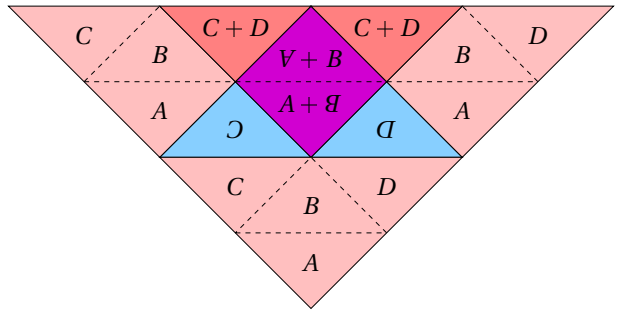
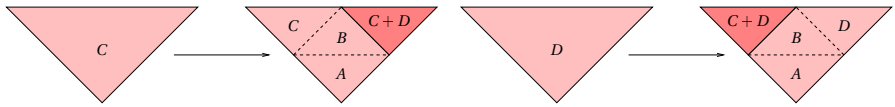


Figure 6.11: Second step of the decomposition



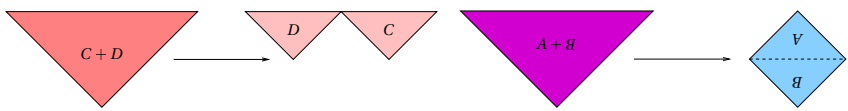
(a) The third substitution rule.

(b) The fourth substitution rule.

Figure 6.12: The third and the fourth substitution rules

The problem with this scheme is that what is happening goes beyond simple juxtaposition of patterns. There can be cancellation at the border between patterns. Of course, we know that $C + C$ is blank, but do we know that $C + D$, for instance, is not? Generally we do not. If the initial state is itself blank, then the whole figure would be, and all tiles being blank is certainly a fixed point for all the substitution rules.

In this case, however, everything turns out well. It should first be noticed that in every part of the picture not tagged as "blank", an A pattern can be found by refining a few more steps. Formally, let $G = (V, E)$ be the graph whose vertices are the different possible tiles (i.e. in this case, A, B, C, D, V, $Ƃ$, $Ɔ$, $Ɑ$, and the sums modulo 2 of tiles having compatible shapes, including the blank tile 0), and



(a) The fith substitution rule.

(b) The sixth substitution rule.

Figure 6.13: The fifth and sixth substitution rules are the superpositions of the third and fourth and the first and second rules respectively. One can see how white spaces emerge because even superpositions of the same pattern cancel out.

edges represent the transition rule in the following way: each vertex has four edges coming out of it, each one pointing to one of its subtiles. In our case, the graph has the property that the set of vertices accessible from A, minus 0, form a strongly connected component.

We may then distinguish two cases: either A has a point in its interior, or it has points only on its border triangle. In the first case, the unique non-empty compact defined by the substitution rule is actually the figure we are looking for. Indeed, it follows from the property of connexity—cf. Proposition 6.2.2—that every non-zero tile actually appearing in the decomposition of the figure has a non-empty interior. Thus, no matter what happens at the boundary between tiles, the figure constructed this way will always converge to the same compact. If, on the contrary, A has points only at its border, the initial configuration is a *glider* and the spacetime image will consist of only of a line. however, not all CAs can have configurations that have points only on a line. For CSCAs only automata with gliders and periodic automata can have such configurations—fractal CSCAs can not have them.

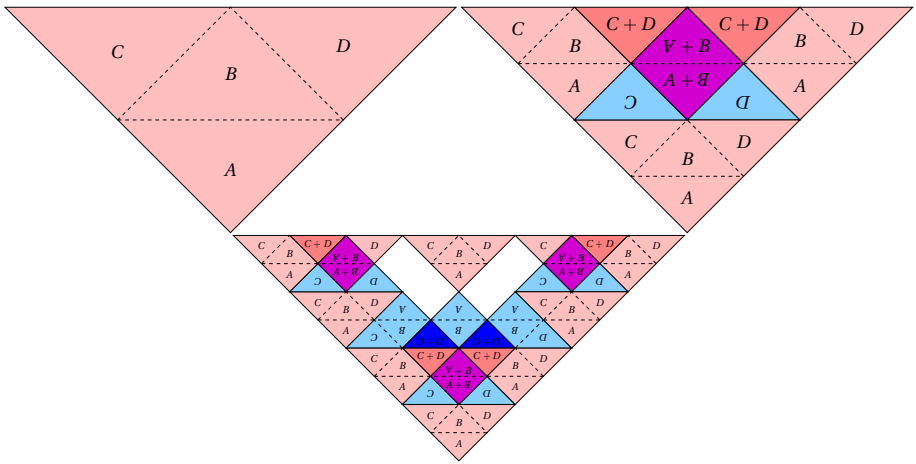


Figure 6.14: Three decomposition steps for the spacetime diagram of **f**; one can clearly see the characteristic white spaces emerging.

6.4 Recursion and matrix substitution system

We will now present a general method to calculate the fractal dimension and average color of the spacetime diagrams of linear CAs in $\mathscr{M}_d(\mathbb{Z}_{p^l}[u,u^{-1}])$. We will again illustrate the method with our example **f**, whereas the derivation is carried out for the general case. Thus, the algorithm works as well on all CAs obeying our

definition, e.g. the CAs presented in Section 6.2, as it works on $\mathbf{f}$. Of course, with larger neighborhoods and groups of higher order the substitution system becomes larger and larger, so that one might want to use a computer to derive the substitution system.

Our approach is the following: from the minimal polynomial Π of the CA $\boldsymbol{t}$ (or any other polynomial fulfilling $\Pi(\boldsymbol{t}) = 0$) we derive a recursion relation for the $\boldsymbol{t}_x^y$'s, the coefficients in u^x of $\boldsymbol{t}^y$. We then forget about every other piece of information we might have on $\boldsymbol{t}$, to concentrate only on this recursion: this shows that the fractal structure, except for contingent blank spaces, can be essentially derived just from the minimal polynomial of $\boldsymbol{t}$. We further develop our recursion scheme for $\boldsymbol{t}$ until we can express every $\boldsymbol{t}_x^y$ in terms of the $\boldsymbol{t}_i^j$ of the first m time steps with coefficients $\alpha_j(x-i,y)$. With a simple grouping of cells we deduce a matrix substitution system that enables us to generate the spacetime diagram of step $t = k^{n+1}$ directly from step $t = k^n$. Using this substitution system we can calculate the fractal dimension, the average coloring, and given an initial state also the whole space time diagram.

Now let $\Pi(X) \in R[u, u^{-1}][X]$ be a monic polynomial such that $\Pi(\boldsymbol{t}) = 0$:

$$\Pi(X) = X^m - \sum_{j=0}^{m-1} \lambda_{\Pi,j} X^j. \tag{6.17}$$

According to the Cayley-Hamilton theorem, which we can apply in our case because $\boldsymbol{t}$ is an endomorphism of a finite-dimensional free module over an abelian ring (see Theorem 3.1. of [31]), the characteristic polynomial of $\boldsymbol{t}$ fulfills this condition. Therefore, we can always find such a polynomial. Let $\mathcal{I}$ be the finite set of exponents i's such that the coefficient in u^i of Π, seen as an element of $R[X][u, u^{-1}]$, is nonzero, so that we can write

$$\lambda_{\Pi,j} = \sum_{i \in \mathcal{I}} \lambda_{\Pi,i,j} u^i. \tag{6.18}$$

$\mathcal{I}$ is not to be confused with the neighborhood of the CA, $\mathcal{N}$, which will not play any role from now on.

Now we want to deduce a decomposition of powers of $\boldsymbol{t}$. First we need some results about powers of sums in Rings with prime power characteristic. For any $x, y, n \in \mathbb{N}$ we have $(x+y)^p \equiv x^p + y^p [p]$. Furthermore we know that if $x \equiv y [p^n]$, then $x^p \equiv y^p [p^{n+1}]$ and therefore $x^{p^k} \equiv x^{p^k} [p^{n+k}]$.[9] Now let $x, y \in R$ and $n \in \mathbb{N}$. Starting from $(x+y)^{p^n} \equiv (x^{p^n} + y^{p^n})[p]$ we use the above results to get to $(x+y)^{p^{n+l-1}} \equiv (x^{p^n} + y^{p^n})^{p^{l-1}} [p^l]$. As the characteristic of R is p^l we actually have $(x+y)^{p^{n+l-1}} = (x^{p^n} + y^{p^n})^{p^{l-1}}$.

[9] This can easily be proven using the divisibility properties of binomial coefficients.

Starting from Equation (6.17) and using the fact that the powers of $\boldsymbol{t}$ commute pairwise we have

$$\begin{aligned} 0 &= \left(\boldsymbol{t}^m - \sum_{j=0}^{m-1} \lambda_{\Pi,j}\, \boldsymbol{t}^j\right)^{p^{n+2(l-1)}} \\ &= \boldsymbol{t}^{p^{n+2(l-1)}m} + \underbrace{(-1)^{p^{n+2(l-1)}}}_{=-1} \cdot \left(\sum_{j=0}^{m-1} \lambda_{\Pi,j}^{p^{n+l-1}}\, \boldsymbol{t}^{p^{n+l-1}j}\right)^{p^{l-1}}. \end{aligned}$$

Using (6.18) we have

$$\boldsymbol{t}^{p^{n+2(l-1)}m} = \left(\sum_{j=0}^{m-1} \left(\sum_{i\in\mathcal{I}} \lambda_{\Pi,i,j}^{p^n}\, u^{p^n i}\right)^{p^{l-1}} \boldsymbol{t}^{p^{n+l-1}j}\right)^{p^{l-1}}. \tag{6.19}$$

For each i, j, the sequence $(\lambda_{\Pi,i,j}^{p^n})$ is ultimately periodic. Therefore there exist integers N and M such that for all i, j,

$$\lambda_{\Pi,i,j}^{p^{M+N}} = \lambda_{\Pi,i,j}^{p^M}.$$

Let $k = p^N$; substituting n by $M + Nn$ in (6.19), we obtain

$$\boldsymbol{t}^{k^n p^{M+2(l-1)}m} = \left(\sum_{j=0}^{m-1} \left(\sum_{i\in\mathcal{I}} \lambda_{\Pi,i,j}^{p^M}\, u^{k^n p^M i}\right)^{p^{l-1}} \boldsymbol{t}^{k^n p^{M+l-1}j}\right)^{p^{l-1}}. \tag{6.20}$$

Hence, if we note $m' = p^{M+2(l-1)}m$ and expand this equation, we find that there is some finite subset $\mathcal{I}'$ of $\mathbb{Z}$ and some elements $\mu_{i,j}$ of R, for $i \in \mathcal{I}'$ and $j \in [\![0; m'-1]\!]$, such that for all $n \in \mathbb{N}$,

$$\boldsymbol{t}^{k^n m'} = \sum_{j=0}^{m'-1} \sum_{i\in\mathcal{I}'} \mu_{i,j}\, u^{k^n i}\, \boldsymbol{t}^{k^n j}, \tag{6.21}$$

where the sum over j comes from expanding the outer bracket in (6.20) and the sum over i from expanding the inner bracket.

We have now applied everything we needed to know about the multiplicative structure on the ring of matrices. As announced at the end of Section 6.1.2, we will now discard the ring structure and concentrate only on the linear recurrence relation that we have just derived. Remember that $\boldsymbol{t}^j \in \mathscr{M}_d\left(R\left[u, u^{-1}\right]\right)$, and we are interested in the coefficient of $\boldsymbol{t}^j$ in u^i, denoted $\boldsymbol{t}_i^j$, so that $\boldsymbol{t}^j = \sum_{i\in\mathcal{I}} \boldsymbol{t}_i^j\, u^i$. By first

multiplying both sides with $\boldsymbol{t}^y$ and then taking the coefficient of u^x we obtain the following relation:

$$\boldsymbol{t}_x^{k^n m'+y} = \sum_{i\in\mathcal{I}'} \sum_{j=0}^{m'-1} \mu_{i,j} \boldsymbol{t}_{x-k^n i}^{y+k^n j}, \tag{6.22}$$

which we rewrite in the form

$$\boldsymbol{t}_x^y = \sum_{(i,j)\in\mathcal{I}'\times[\![0;m'-1]\!]} \mu_{i,j} \boldsymbol{t}_{x+f_{i,j}(y)}^{g_{i,j}(y)}, \tag{6.23}$$

where $f_{i,j}(y) = -k^n i$ and $g_{i,j}(y) = y - k^n(m'-j)$. Of course, this works with any n, but we will choose $n = \lfloor \log_k \frac{y}{m'} \rfloor$. In order to emphasize that the rest of the proof will use only a minimal structure, we state in the next proposition what will actually be proven, and change the notation from $\boldsymbol{t}$, which was an element of $\mathscr{M}_d\left(R\left[u,u^{-1}\right]\right)$, to Ξ, an element of a more arbitrary R-module.[10] It is straightforward to check that $\boldsymbol{t}$ fulfills the hypotheses of the proposition.

Proposition 6.4.1. *Let M be a finite R-module, k a positive integer, Λ a finite set of indices, and for $i\in\Lambda$, $\mu_i \in R$, $f_i : [\![m;+\infty[\![\to \mathbb{Z}$ and $g_i : [\![m;+\infty[\![\to \mathbb{N}$ such that for all $y\in[\![m;+\infty[\![$ and $t\in[\![0;k-1]\!]$,*

- *$g_i(y) < y$;*
- *$f_i(ky+t) = k f_i(y)$ and $g_i(ky+t) = k g_i(y) + t$.*

For $x\in\mathbb{Z}\times\mathbb{N}$, let $\Xi_x^y \in M$ be such that when $y\geqslant m$,

$$\Xi_x^y = \sum_{i\in\Lambda} \mu_i \Xi_{x+f_i(y)}^{g_i(y)}. \tag{6.24}$$

It follows that there exists a finite set E and a function $e : \mathbb{Z}\times\mathbb{N}\to E$ such that

- *Ξ_x^y is a function of $e(x,y)$;*
- *for $s,t\in[\![0;k-1]\!]$, $e(kx+s,ky+t)$ is a function of s, t, and $e(x,y)$.*

The introduction of a new function e in this proposition comes from the need of a scaling property, expressing that the state at point $(kx+s,ky+t)$ can be deduced from the state at point (x,y). Such a property does not follow immediately from Equation (6.24), but it is possible to expand the state space from M to E, and to put more information into e than into Ξ, so as to fulfill the scaling property. An immediate consequence of this proposition is that the spacetime diagrams of Ξ_x^y of order k^n can be described by colored matrix substitution systems, so that Theorem 6.2.1 will follow from Proposition 6.2.3. Let us now prove Proposition 6.4.1. If $y\geqslant m$, we

[10] This Ξ is not connected to the phase space used in the other chapters.

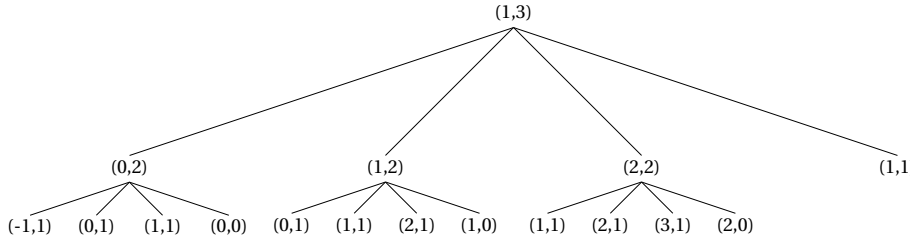


Figure 6.15: Recursive calls to (6.24) for $\boldsymbol{t} = \mathbf{f}$

can recursively apply Equation (6.24) to give an expression of Ξ_x^y in terms of a linear combination of $\Xi_{x'}^{y'}$'s with $y' < y$.

Let us consider the case $\boldsymbol{t} = \mathbf{f}$ as an example. In this case, $R = \mathbb{F}_2$, $k = m = 2$, and $\Pi_{\mathbf{f}}(X) = X^2 + (u^{-1} + 1 + u)X + 1$. The period of the $\lambda_{\Pi,i,j}^x$ is one as they are elements of $\mathbb{F}_2$. Thus $M = 0$ and $N = 1$ and therefore $m' = p^{M+2(l-1)} = 2 \cdot 2^{2(l-1)}$ and $k = 2$. Furthermore $(x+y)^2 = x^2 + y^2$, which means $l - 1 = 0$ and thus $m' = 2$. Finally, we note that $\mu_{i,j} = \lambda_{\Pi,i,j}$ and $\mathcal{I}' = \mathcal{I}$. Now we can use Equation (6.24) to recursively compute coefficient Ξ_x^y respectively $\mathbf{f}_x^y$, where we use $g_{i,j}(y) = y - 2^n(2-j)$, $f_{i,j}(y) = -2^n i$ and $n = \lfloor \log_2 y/2 \rfloor$. Explicitly we have

$$\begin{aligned}
\Xi_x^y &= \sum_{(i,j)\in\{-1,0,1\}x\{0,1\}} \mu_{i,j} \Xi_{x+f_{i,j}(y)}^{g_{i,j}(y)} \\
&= \Xi_{x+f_{0,0}(y)}^{g_{0,0}(y)} + \Xi_{x+f_{-1,1}(y)}^{g_{-1,1}(y)} + \Xi_{x+f_{0,1}(y)}^{g_{0,1}(y)} + \Xi_{x+f_{1,1}(y)}^{g_{1,1}(y)} \\
&= \Xi_x^{y-2^{n+1}} + \Xi_{x+2^n}^{y-2^n} + \Xi_x^{y-2^n} + \Xi_{x-2^n}^{y-2^n}.
\end{aligned}$$

If we use this on Ξ_1^3 we have $n = 0$ and obtain $\Xi_1^3 = \Xi_1^1 + \Xi_0^2 + \Xi_1^2 + \Xi_2^2$. Whereas the decomposition halts here for Ξ_1^1, it goes on for the other three terms of the sum, again with $n = 0$. For instance, Ξ_0^2 is transformed into $\Xi_0^0 + \Xi_{-1}^1 + \Xi_0^1 + \Xi_1^1$. This gives the tree of recursive calls in Figure 6.15, where each node is labeled (x,y) for Ξ_x^y. In this particular case, because $R = \mathbb{F}_2$, the only possible coefficients are 0 and 1, so they are simply represented by the absence or the presence of the corresponding term.

Starting from any point $(x,y) \in \mathbb{Z} \times \mathbb{N}$ and using recursively equation (6.24), we obtain the expression

$$\Xi_x^y = \sum_i \sum_{j=0}^{m-1} \alpha_{\Pi,i,j}(x,y) \Xi_i^j, \tag{6.25}$$

which we take as a definition of $\alpha_{\Pi,i,j}(x,y)$. On Figure 6.15, this corresponds to (parity) counting the leaves tagged (i,j); for instance, there are 4 leaves tagged $(1,1)$, so $\alpha_{\Pi_{\mathbf{f}},1,1}(1,3) = 0$ and $(1,1)$ does not appear as a node in Figure 6.16.

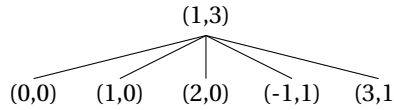


Figure 6.16: Ξ_1^3 as a sum of the simplest terms, when $\boldsymbol{t} = \mathbf{f}$

Since the relation (6.24) is invariant under translations of the parameter x, we have $\alpha_{i,j}(x,y) = \alpha_{0,j}(x-i,y)$. Noting $\alpha_j := \alpha_{0,j}$, we have the following equation:

$$\Xi_x^y = \sum_{i\in\mathbb{Z}} \sum_{j=0}^{m-1} \alpha_j(x-i,y)\Xi_i^j. \tag{6.26}$$

Let us now show that $\alpha_\bullet(kx+s, ky+t)$, for $s,t \in [\![0;k-1]\!]$, is a function of s, t, and the $\alpha_\bullet(x',y)$'s, where x' ranges over some neighborhood of x. By substituting x with $kx+s$ and y with $ky+t$ in Equation (6.24), we have

$$\Xi_{kx+s}^{ky+t} = \sum_{i\in\Lambda} \mu_i \Xi_{k(x+f_i(y))+s}^{kg_i(y)+t}. \tag{6.27}$$

Because the indices and exponents of Ξ on the left and right side of this equation have undergone the same transformation $(x,y) \mapsto (kx+s, ky+t)$, we arrive recursively at the point

$$\Xi_{kx+s}^{ky+t} = \sum_{i\in\mathbb{Z}} \sum_{j=0}^{m-1} \alpha_j(x-i,y)\Xi_{ki+s}^{kj+t} \tag{6.28}$$

which we want to compare to the following equation, directly deduced from (6.26):

$$\Xi_{kx+s}^{ky+t} = \sum_{i\in\mathbb{Z}} \sum_{j=0}^{m-1} \alpha_j(kx+s-i, ky+t)\Xi_i^j. \tag{6.29}$$

Of course, there can be terms in (6.28) with $kj+t \geqslant m$, so that the decomposition is not over: it then needs to be performed to its end. For instance, in our example $\boldsymbol{t} = \mathbf{f}$, if instead of the tree of recursive calls for Ξ_1^3 we want that of $\Xi_3^6 = \Xi_{2\times1+1}^{2\times3}$, we know that the upper part of the tree will be Figure 6.17.

As for the lower part, the decomposition is not finished and needs to be performed to its end. Thanks to linearity, we do not need to remember the whole tree, just its leaves. As a tree growing from its leaves, we just need to apply the transformation $(i,j) \mapsto (2i+1, 2j)$ to the tree in Figure 6.16 in order to find the correct decomposition for Ξ_3^6, which gives Figure 6.18.

However, even keeping track only of the leaves would require an unbounded alphabet, that is why we defined $\alpha_{\Pi,j}(x,y) = \alpha_{\Pi,i,j}(x+i,y)$, i.e. from the tree for Ξ_x^y

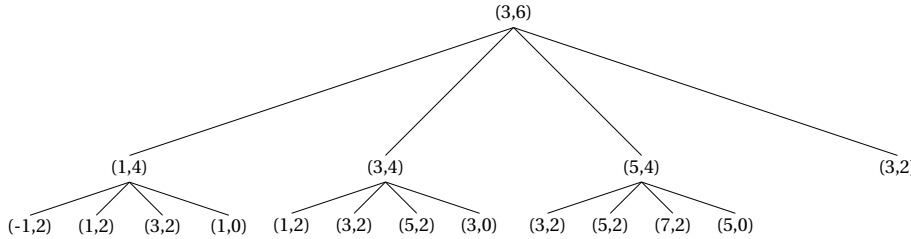


Figure 6.17: Recursive calls to (6.24) for $\boldsymbol{t} = \mathbf{f}$, minus some leaves

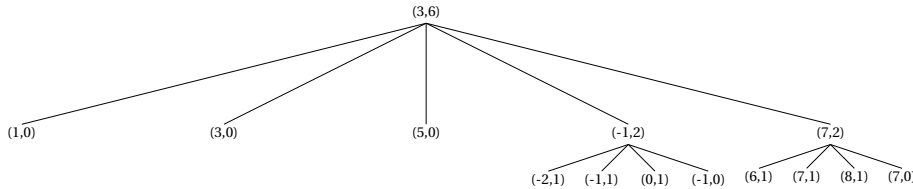


Figure 6.18: Trees grow from their leaves

we want to keep track only of the leaves labeled by $(0, j)$. Alas, that will not work so easily: in Figure 6.18, $(0,1)$ appears once as a son of $(-1,2)$, which comes from node $(-1,1)$ in Figure 6.16. So it it seems that we cannot just forget about everything else.

As we just saw on one example, $\alpha_{\Pi,\cdot}(kx+s, ky+t)$ typically does not depend on $\alpha_{\Pi,\cdot}(x,y)$. Instead the final decomposition of (6.28) will relate the coefficients $\alpha_{\Pi,j}(kx+s-i, ky+t)$ of the Ξ_i^j in (6.29) to sums of the $\alpha_{\Pi,j}^k(x-i,y)$'s. Therefore $\alpha_{\Pi,\cdot}(kx+s, ky+t)$ depends on the $\alpha_{\Pi,\cdot}(x+i,y)$'s for i ranging over some finite set $\mathscr{S}$ depending only on Π. However, this is not much of a problem, as a simple grouping will take care of it—a technique commonly attributed to [96]. Let us show that $\mathscr{S}$ is finite. Since $j \in \{0;\dots;m-1\}$ and $t \in \{0;\dots;k-1\}$, the $kj+t$ appearing as an exponent of Ξ in (6.28) is in $\{0;\dots;km-1\}$. We therefore have to use at most $(k-1)m$ recursive calls to (6.24) in order to get down to coefficients Ξ_x^y with $y < m$, each one of them decreasing the exponent by at least 1. Each one of them also increases the index by $f_i(y)$; since both Λ and $[\![0;km-1]\!]$ are finite, the set of possible $f_i(y)$'s is also bounded by some M, and the total variation in the index, i.e. $\mathscr{S}$, is then bounded by $(k-1)mM$; let us say $\mathscr{S} \subseteq [\![d_{\min}; d_{\max}]\!]$.

We now introduce $\beta_{\Pi,\bullet}(x,y) = (\alpha_\bullet(x-i,y))_{i\in\mathscr{S}'}$, where $\mathscr{S}' = [\![\delta_{\min};\delta_{\max}]\!]$, $\delta_{\max}$ being such that $\delta_{\max} \geqslant d_{\max} + \left\lceil \frac{\delta_{\max}}{k} \right\rceil$ and $\delta_{\min}$ such that $\delta_{\min} \leqslant d_{\min} - 1 + \left\lceil \frac{\delta_{\min}+1}{k} \right\rceil$. This time, for $s,t \in \{0;\dots;k-1\}$, $\beta_{\Pi,\bullet}(kx+s, ky+t)$ does really depend only on

$\beta_{\Pi,\bullet}(x,y)$. Indeed,

$$\beta_{\Pi,\bullet}(kx+s,ky+t)=(\alpha_\bullet(kx+s-i,ky+t))_{i\in\mathscr{S}'},$$

and each $\alpha_\bullet(kx+s-i,ky+t)$ depends on $\left(\alpha_\bullet\left(x+\left\lfloor\frac{s-i}{k}\right\rfloor-j,y\right)\right)_{j\in\mathscr{S}}$ only; the choice of $\mathscr{S}'$ has been made so that $j-\left\lfloor\frac{s-i}{k}\right\rfloor\in\mathscr{S}'$. This concludes the proof of Proposition 6.4.1, as we can choose $E=M^{[\![0;m-1]\!]\times\mathscr{S}'}$, with $e(x,y)(j,i)=\alpha_j(x-i,y)$.

6.4.1 Example: f

In the case of **f**, Equation (6.28) becomes

$$\Xi^{2y+t}_{2x+s}=\sum_i \alpha_{\mathbf{f},0}(x-i,y)\Xi^t_{2i+s}+\alpha_{\mathbf{f},1}(x-i,y)\Xi^{2+t}_{2i+s}. \tag{6.30}$$

The first term is now elementary, but the second one has to be decomposed once more, i.e.

$$\Xi^{2+t}_{2i+s}=\Xi^t_{2i+s}+\Xi^{1+t}_{2i+s-1}+\Xi^{1+t}_{2i+s}+\Xi^{1+t}_{2i+s+1}. \tag{6.31}$$

If $t=0$ we are finished. However, if $t=1$ we have

$$\Xi^3_{2i+s}=\Xi^0_{2i+s-1}+\Xi^0_{2i+s}+\Xi^0_{2i+s+1}+\Xi^1_{2i+s-2}+\Xi^1_{2i+s+2}. \tag{6.32}$$

Comparing this with (6.29) we can deduce the substitution rule of $\alpha_{\mathbf{f},\cdot}$. It can then be written in the following way, where for convenience $\alpha_{\mathbf{f},\cdot}$ is represented as

$\alpha_{\mathbf{f},1}$
$\alpha_{\mathbf{f},0}$

:

$\alpha_{\mathbf{f},\cdot}(x,y)$

↓

$\alpha_{\mathbf{f},\cdot}(2x,2y+1)$	$\alpha_{\mathbf{f},\cdot}(2x+1,2y+1)$
$\alpha_{\mathbf{f},\cdot}(2x,2y)$	$\alpha_{\mathbf{f},\cdot}(2x+1,2y)$

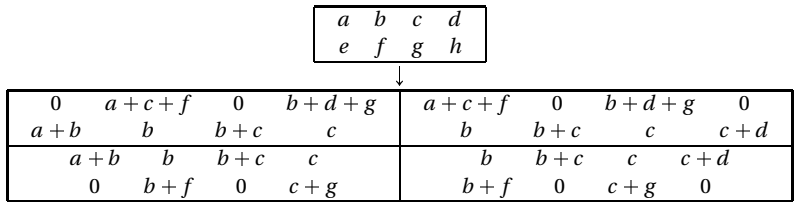
||

$\alpha_{\mathbf{f},0}(x,y)+\alpha_{\mathbf{f},1}(x-1,y)+\alpha_{\mathbf{f},1}(x+1,y)$ $\alpha_{\mathbf{f},1}(x,y)$	0 $\alpha_{\mathbf{f},1}(x,y)+\alpha_{\mathbf{f},1}(x+1,y)$
$\alpha_{\mathbf{f},1}(x,y)$ $\alpha_{\mathbf{f},0}(x,y)+\alpha_{\mathbf{f},1}(x,y)$	$\alpha_{\mathbf{f},1}(x,y)+\alpha_{\mathbf{f},1}(x+1,y)$ 0

Thus our grouping uses the bound $[\![-1,2]\!]$. We will represent the grouping in the form

$\alpha_{\mathbf{f},1}(x-1,y)$	$\alpha_{\mathbf{f},1}(x,y)$	$\alpha_{\mathbf{f},1}(x+1,y)$	$\alpha_{\mathbf{f},1}(x+2,y)$
$\alpha_{\mathbf{f},0}(x-1,y)$	$\alpha_{\mathbf{f},0}(x,y)$	$\alpha_{\mathbf{f},0}(x+1,y)$	$\alpha_{\mathbf{f},0}(x+2,y)$

The alphabet thus has size 256, and the substitution system is described by



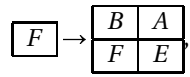
Let us denote by A the matrix having a 1 in position a and 0 elsewhere, B the matrix having a 1 only in position b, and so on. For these matrices, we will denote the sum of matrices by a simple juxtaposition: AB will mean $A+B$, as the matrix multiplication has no meaning in this context.

Since $t_x^0 = \delta_{x,0} t_0^0$, the starting position, with which we describe the whole line number 0, is

$\cdots$	0	H	G	F	E	0	$\cdots$

.

Since we have, for instance, the rule



the graph derived from this substitution system is aperiodic; this means that, in whatever way $A, B, C, \ldots$ are represented, either as colored dots or as white dots, the pattern converges, and the fractal structure is described by this matrix substitution system (see Section 6.2.2).

To calculate the fractal dimension of our spacetime diagram we use the transition matrix of the matrix substitution system, which contains the information about the images of all states. The line corresponding to F would contain a 1 in the rows of A, B, E, and F and zeros elsewhere. As every cell gives rise to four new cells the sum of all entries in each column of the matrix is 4. We thus deal with a sparse 256×256 matrix. The base 2 logarithm of the second largest eigenvalue of this matrix is the fractal dimension of the spacetime diagram (cf. for instance [96]). Here this gives a fractal dimension of $\log_2 1/2\left(3+\sqrt{17}\right) \simeq 1.8325$, as also found in [106].

Let us note that up to this point our analysis for **f** is directly valid for all CA in $\mathscr{M}_2(\mathbb{Z}_2[u,u^{-1}])$ of determinant 1 and trace $u^{-1}+1+u$. The additional information is only used for the actual coloring of the picture. In general all CA with the same minimal polynomial have the same substitution system, and in dimension 2 the minimal polynomial is entirely determined by the trace and the determinant. The fractal we obtain if we use the substitution system starting from

$\cdots$	0	H	G	F	E	0	$\cdots$

is shown in Figure 6.19.

In the case of **f** the connection between the substitution system and the colored picture is very simple; let us take $\xi = \binom{1}{0}$ as the initial state. Then the state of cell x after y iterations is

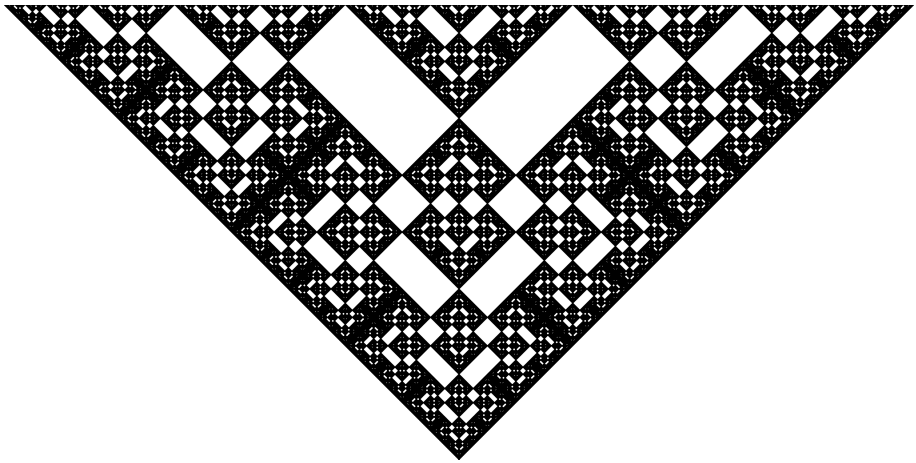


Figure 6.19: The general fractal time evolution of a linear CA with $k = m = 2$, determinant 1 and trace $u^{-1}+1+u$; only the areas where the whole group of αs is 0 is marked white. Thus the image appears to have less white than the colored picture. In the limit of infinite recursion this effect vanishes, thus the fractal that is generated is actually the same.

$$\begin{aligned} \mathbf{f}_x^y \xi &= \sum_i \left(\alpha_{\mathbf{f},0}(x-i,y)\mathbf{f}_i^0 + \alpha_{\mathbf{f},1}(x-i,y)\mathbf{f}_i^1 \right) \xi \\ &= \alpha_{\mathbf{f},0}(x,y)\xi + \sum_i \alpha_{\mathbf{f},1}(x-i,y)\mathbf{f}_i^1 \xi. \end{aligned} \tag{6.33}$$

Since

$$\mathbf{f}_0^1 = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, \ \mathbf{f}_1^1 = \mathbf{f}_{-1}^1 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \text{ and } \mathbf{f}_i^1 = 0$$

when $i \notin \{-1;0;1\}$, we have

$$\boldsymbol{t}_x^y \xi = \begin{pmatrix} \alpha_{\mathbf{f},0}(x,y) \\ \alpha_{\mathbf{f},1}(x,y) \end{pmatrix}.$$

This gives us a color assignment for each state of the matrix substitution system, which corresponds to simply dropping all states that include neither B nor F.

We can now determine the average hue of the spacetime diagram making use of the eigenvector corresponding to the second largest eigenvalue of the transition matrix [96]. Let us say $\binom{1}{0}$, $\binom{0}{1}$ and $\binom{1}{1}$ are respectively coded by the colors c_{10}, c_{01} and c_{11}; let c_q be the white color. We determine which symbols of the alphabet belong to each of the colors by looking only at the part $\binom{\alpha_{\mathbf{f},0}(x,y)}{\alpha_{\mathbf{f},1}(x,y)}$. Then we simply

add up all the weights of symbols with the same color in the eigenvector. We obtain the following unnormalized coefficients: c_{10}: $2(4+\sqrt{17})$, c_{01}: $2(4+\sqrt{17})$, and c_{11}: $5+\sqrt{17}$.

In Figure 6.3a, this color code was used: $c_{10} =$, $c_{01} =$ and $c_{11} =$. We must therefore have the following average hue: .

In [109] a general algorithm is presented that takes an arbitrary linear CA as input and outputs the associated substitution system.

6.5 Higher dimensions

So far we have only studied the one-dimensional case, but the analysis can be carried over to higher dimensions without much additional work. Instead of an element of $\mathscr{M}_d(R[u,u^{-1}])$, a n-dimensional linear cellular automaton would then be an element of $\mathscr{M}_d(R[u_1,u_1^{-1},u_2,u_2^{-1},\dots u_n,u_n^{-1}])$, matrix substitution systems would become $(n+1)$-dimensional array substitution systems, the system of indices in Section 6.4 would be further complicated, and the spacetime diagrams would be harder to display. However, the generalization does not present any theoretical difficulty and makes nice pictures.

For instance, Figure 6.20 is a view of the the fractal structure produced by

$$\Omega = \begin{pmatrix} 0 & 1 \\ 1 & (1+u+u^{-1})(1+v+v^{-1}) \end{pmatrix},$$

a variation on $\mathbf{f}$ belonging to $\mathscr{M}_2(\mathbb{Z}_2[u,u^{-1},v,v^{-1}])$. The neighborhood of Ω is a square centered on $(0,0)$, and so, unsurprisingly, the pyramid of its spacetime image has a square base. We cut it open so as to reveal its insides. The sector that is cut out is delimited by two half-lines. The one that is visible on the right along $(1,0)$ the sector having an angle of $\frac{2}{3}\pi$.

For any positive integer n, the coefficient in v^n of Ω^n, seen as an element of $\mathscr{M}_2(\mathbb{Z}_2[u,u^{-1}])[v,v^{-1}]$, is

$$\begin{pmatrix} 0 & 0 \\ 0 & (1+u+u^{-1})^n \end{pmatrix}.$$

Therefore, the pattern on the face of the pyramids is given by $(1+u+u^{-1}) \in \mathscr{M}_1(\mathbb{Z}_2[u,u^{-1}])$, as can be indeed noticed on the figure, where half of this symmetric pattern is clearly visible on the external face. By contrast, the coefficient of Ω^n in v^0 is exactly $\mathbf{f}^n$, which explains why the cut along $(1,0)$ in Figure 6.20 looks like the half of Figure 6.19 (turned upside-down).

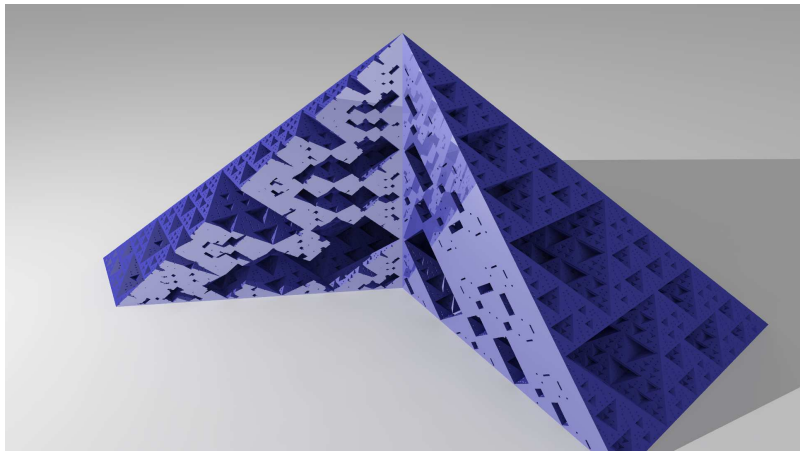


Figure 6.20: Spacetime image of a 2D CA; it is displayed upside down and cut open to reveal the inner structure.

7 Convolutional codes

This last chapter employs the theoretical framework we built up in Chapters 3 and 4 to study convolutional codes. Recall that convolutional codes are of interest because of their promising performance in terms of code rate and their low complexity error-corrections algorithms, as discussed in Section 3.10.1. In this chapter our goal is to introduce an efficient description of convolutional codes in terms of Clifford memory channles and to use it to analyze catastrophic errors, finite depth decoding circuits and to compare the performance of convolutional codes to the performance of block codes. Some of the results of this chapter have already been included in [34].

As we have already argued in Section 3.10 convolutional codes can be encoded by memory channels. Like in the case of block codes we can describe the encoding operation of a stabilizer code by a Clifford channel. The error-correction process can again not be implemented by a Clifford channel. The encoding inverse can of course also be implemented by a Clifford memory channel. The representation of convolutional codes as memory channels is depicted in Figure 7.1.

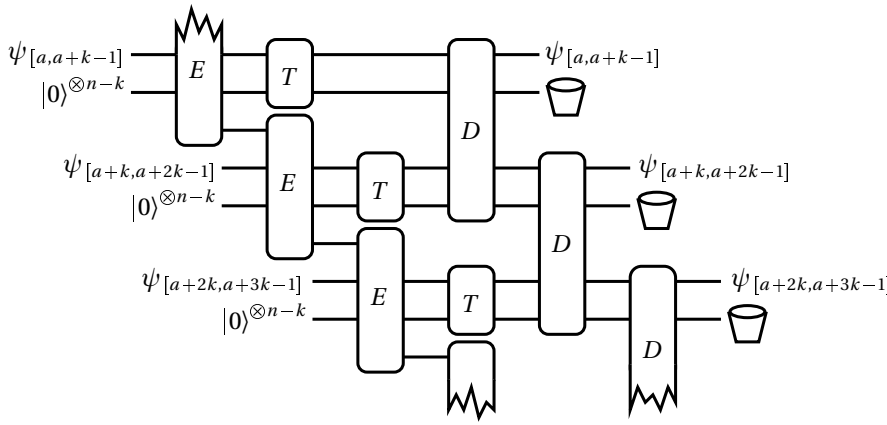


Figure 7.1: Encoding and decoding a convolutional code described in terms of memory channels; the transmission channel S is assumed to have correlation only inside a block. In general it could also be a memory channel but that case is not studied here.

Using this description of quantum convolutional encoders by Clifford memory channels we will show how our results on the forgetfulness of Clifford memory channels can be used to study catastrophic errors. The main result will be that strictly forgetful encoders and encoder inverses are non-catastrophic. Our study of causal operations in Chapter 4 showed that strictly forgetful channels correspond to finite depth causal operations which have a causal inverse. Using this result we will construct an encoding-decoding circuit of finite depth enabling a convolutional coding scheme with a delay between sending and receiving of information which is independent of the length of the transmission. Furthermore, the total quantum memory needed on the decoder side is independent of the length of the transmission. In Section 7.2 we will present an algorithm to construct an encoding channel given the stabilizer generators of the code and determine the minimal internal memory needed for the encoding channel.

In the second part of this chapter we will present a convolutional equivalent of the quantum Hamming bound. Motivated from different assumptions on the transmission channel we will derive bounds for different error-correction requirements. These bounds are then used to study the performance potential of convolutional codes in comparison to block codes under resource constraints on the encoder and decoder side. In this analysis a correctable error rate and the size of the input operation will be given. The measure of performance will be the achievable code rate. We will show that convolutional codes have the potential to outperform block codes if the number of errors per block is not an integer. This potential is especially high if we consider an asymmetric scenario, where only the resources on the encoder are subject to restrictions. If the restrictions are extended to the decoder side the potential to outperform block codes is less pronounced, but still existent.

Throughout this chapter we will assume that the transmission channel introducing the errors is not a memory channel as this would affect the error propagation properties of the code. A non-forgetful transmission channel could introduce errors that can not be corrected by a finite depth coding circuit, because they have unbounded correlation length. Classical correlations of the error positions are also not considered explicitly. An exception is the low error bound, where we assume that an erroneous block is followed by τ error-free blocks which implies a correlation of the error positions (or a very low error rate). However, correlations inside a block are implicitly allowed. We will not consider the process of error-correction as such. As shown in Section 3.9.2 the measurements of the error syndromes with the following application of the necessary correction operation can not be described as a Clifford channel. This part of the decoding could either be done before or after the inverse encoding, implying different resource requirements for the decoder. This will be discussed in Section 7.2.4.

7.1 Catastrophic errors

The information transfer between code blocks will not only prove to be the core reason for the superior rates achievable by convolutional codes but is also at the heart of one of the main problems with convolutional codes. Errors that occur during encoding, transmission or decoding can propagate through the encoded stream and affect an unbounded number of qubits. Such an error is called catastrophic, because its correction needs an operation of unbounded size. In the usual error-correction settings one only considers errors during the transmission of the information and assumes the encoding and decoding operations to be perfect. In this setting only the error propagation properties of the decoder are of interest. We will consider an error-correction process in which the transmitted stream is processed in the inverse encoder before the error syndromes are measured. Therefore, the decoder includes an inverse of the encoder. We will show that the error propagation properties depend on the forgetfulness of the channel. In Section 3 of the appendix we prove that the inverse of a strictly forgetful channel is strictly forgetful, too (Theorem 3.2). Therefore, we can also study the error propagation properties of the encoder.

The existence of encoders that exhibit catastrophic errors is also apparent in Section 7.2.2 where we showed that we can not always find encoded Z operators which respect the translation invariant structure and have finite support. An error operator which includes a Z tensor factor on one of the input qubits corresponds to an operator with unbounded support on the output side. This error could only be corrected after the end of the transmission.

Catastrophic errors have to be avoided for two reasons: if they occur they are practically uncorrectable, because there will always be a limit on the number of received qubits that can be stored and on the number of qubits we can operate on at the same time. But even if the probability for catastrophic errors is low enough to accept them as a trade-off for good performance, the pure possibility of catastrophic errors makes the encoder practically unusable. Encoders with catastrophic errors are non-forgetful Clifford channels, which do not have causal inverses (see Theorem 4.5.1). Therefore, the decoding can only begin after the last qubit has been received. The encoding and decoding algorithm would have a depth proportional to the length of the transmission. In this case the memory on the decoding side limits the length of the transmission. Furthermore, long time storage of quantum information is difficult and quantum memory a very costly resource. With increasing storage time the probability for new errors increases as well. Therefore it is in general desirable to have encoders and decoders that allow a finite depth circuit and a delay between transmission and decoding that is independent of the decoding length. These “on-line” (finite-depth) decoders only exist for non-catastrophic encoders.

7.1.1 Definitions of catastrophic errors and encoders

In the literature one can find several different definitions of catastrophic errors and catastrophic encoders. In this section we will review the most important definitions and their properties and implications to motivate the definition we will use in this thesis.

The first definition of a catastrophic error was formulated in [110]. It states that an error is catastrophic if it affects a finite number of qubits before decoding but an infinite number of qubits after decoding (inverse encoding, no error-correction). This is the basis for the following definition:

Definition 7.1.1 ([71])**.** *Consider an (n,k,m)-convolutional encoder which protects $q \cdot k$ logical qubits. A catastrophic error is an error that affects $\mathcal{O}(1)$ qubits before the end of the decoding operation and that can only be corrected by a unitary transformation whose size of support grows with q, for large q.*

If a decoder for the encoding scheme allows such errors it is called catastrophic. This definition only considers the size of the necessary correction operation, but not its position. This is a flaw in the definition as we will later show.

In [69] the authors differentiate between catastrophic encoders and completely catastrophic encoders. Their definitions build upon the memory state diagram of the encoder *seed transformation*. The seed transformation is the inverse encoding Clifford channel and maps operators on the input side to operators on the output side.

The state diagram of an encoding seed transformation E of a convolutional code is a directed graph that is comprised of all Pauli products $M = \mathrm{w}_m(\xi)$ on the memory algebra as nodes. The memory Pauli products are referred to as "memory-states" in [69]. Two nodes $M = \mathrm{w}_m(\xi)$ and $M' = \mathrm{w}_m(\xi')$ are connected by an edge directed from M to M' with label $(L,P) = (\mathrm{w}_k(\eta), \mathrm{w}_n(\zeta))$ if there exist $\eta \in \mathbb{Z}_2^{2k}$, $\zeta \in \mathbb{Z}_2^{2n}$ and $S^z = \mathrm{w}(\zeta_z)$, $\zeta_z \in \bigoplus_{n-k}\{(0,0),(0,1)\}$, i.e., $S^z \in \{I,Z\}^{n-k}$ such that $P \otimes M' = E\big[M \otimes L \otimes S^z\big]$. The labels L and P are called logical and physical label of the edge. In other words there is an edge between nodes M and M' if there is a combination of an unencoded stabilizer operator S^z, an operator L on the unencoded logical qubits and an operator P on the encoded physical qubits such that the seed transformation maps the complete input operator to the complete output operator.

In this formalism an encoder is called non-catastrophic if and only if all cycles in its state diagram which have physical weight 0 also have logical weight 0. The physical, respectively logical, weight of an edge is the number of non-identity factors in P, respectively L; the weights of a cycle are the summed weights of its edges. This definition means that an encoder is non-catastrophic if and only if there is no cycle that has non-identity logical outputs without any physical inputs. This definition is weaker than Definition 7.1.1, because it only considers the influence of

catastrophic errors on the decoded physical qubits, not on the decoded stabilizers. It is also used in [95, 48].

The second definition extends to the decoded stabilizers: An encoder is completely non-catastrophic if the only cycle with zero physical weight is the self loop at $M = I^{\otimes n}$. We can best understand this definition if we think of the encoding inverse. An error occurring during transmission corresponds to an operator on the physical qubits. If there is a cycle with zero physical weight, but non-zero logical or stabilizer weight, a finitely localized error can steer the encoder into this cycle, where it will stay without any further input of logical operators. For all loops except the self loop at $M = I$ this makes the encoding channel catastrophic. Therefore, this definition is very similar to the definition of a strictly forgetful memory channel– with the exception that the graph uses a reduced seed transformation where the images of the Pauli X operators on the unencoded stabilizer qubits are not specified. We will show in 7.2.3 that we can always find a non-catastrophic completion if the transformation of stabilizers and physical qubits is non-catastrophic.

The downside of these definitions is that they both employ the state diagram which is exponential in the memory dimension. This is an unnecessarily complex object, as it is derived from the seed transformation, that can be described by a set of parameters that is polynomial in the number of memory qubits. We will therefore base our definition of catastrophic errors on Definition 7.1.1. However, we could also prove the equivalence of the latter graph based definition to strict forgetfulness of the encoding channel, makeing the connection to the criterion we are going to introduce in the following. Our criterion is based on the forgetfulness of the encoding channel.

We will now show where Definition 7.1.1 is flawed and change it accordingly. As introduced above the encoder of a convolutional code can be described as a circuit built up by the concatenation of the same unitary U implementing the encoding channel E shifted by multiples of n qubits (see Figure 7.1). In practice we need some operations for initialization and termination of the encoding, which we omit here. In [71] it is claimed that an encoder is free of catastrophic errors if and only if it can be converted into a circuit of finite depth as shown in Figure 7.2.

Conjecture 7.1.2 ([71])**.** *A quantum convolutional encoder E is non-catastrophic if and only if the encoding operation E_q can be decomposed in the following way for large q:*

$$E_q = \tilde{T}_{erm}(q) \left(\prod_{i=0}^{\lfloor q/l_t \rfloor} D^{i l_t}[U_t] \right) \cdots \left(\prod_{i=0}^{\lfloor q/l_1 \rfloor} D^{i l_1}[U_1] \right) \tilde{I}_{nit}(q) \tag{7.1}$$

$\tilde{T}_{erm}(q)$ and $\tilde{I}_{nit}(q)$ are modified initialization and termination operations which can depend on q but whose localization area is bounded independent of q. $\{U_j\}_j$ is a finite set of unitary operations with bounded localization and independent of q. Furthermore $D^i[U_j]$ and $D^{i'}[U_j]$ commute. The l_j are also independent of q.

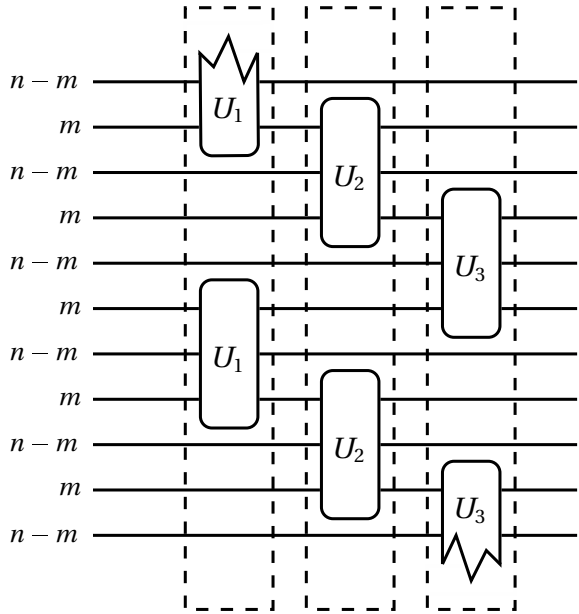


Figure 7.2: Finite depth encoding circuit of a convolutional code; the stroked boxes each denote one layer of unitary operations. All unitaries at the same layer can be applied in the same time, as they commute.

Therefore, the circuit is made up by t layers of unitaries U_j, each invariant under shifts by $i\,l_t\,n$ qubits.

The proof of this conjecture as presented in [71] contains a minor error which can not be fixed because a counter example exists to the necessity of the decomposition for the encoder to be non-catastrophic (in the sense of the Definition 7.1.1). We will present this in Example 7.1.3. However, the part of the proof that shows that the decomposition is sufficient to exclude catastrophic errors is correct.

The proof of the necessity of the decomposition begins from the original encoding circuit. The order of two encoding steps is flipped. In general the encoding steps do not commute and the flipping introduces an error in the encoding. According to the Definition 7.1.1 this error can be corrected with an operation of bounded support. At this point is is wrongly assumed that the localization area is independent of q if the encoder is non-catastrophic according to Definition 7.1.1. However, this is not part of the definition; the position of the operator can still depend on q. In 7.1.3 we present an example of a code with this error-propagation property.

Example 7.1.3 (Counterexample to Conj. 7.1.2)**.** *We start with a (n,k,d) block stabilizer code with stabilizer group $\mathcal{S}$ and encode it like a $(n,k,d,1,\infty)$ convolutional code. The convolutional code uses the stabilizer generators of the block code and their shifts by multiples of n qubits. This code can be encoded by a circuit with unitary operations that have no overlap. Now we add a memory qubit which does not interact with the other qubits. This additional memory qubit is just passed on from one step of the encoding to the next unchanged. The resulting encoding operation E is shown in Figure 7.3.*

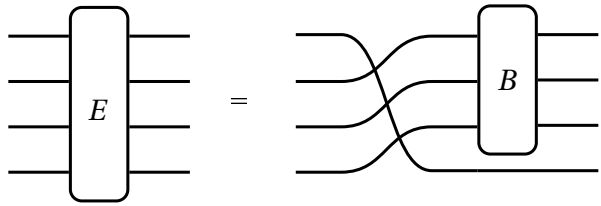


Figure 7.3: Encoding operation of a block code encoded as a convolutional code; the memory qubit is just passed on to the next step of the encoding. The operation B is the encoder of the block code.

Concatenating several steps of the encoding operation it is easy to observe that the qubit we added is not included in the process of encoding (see Figure 7.4).

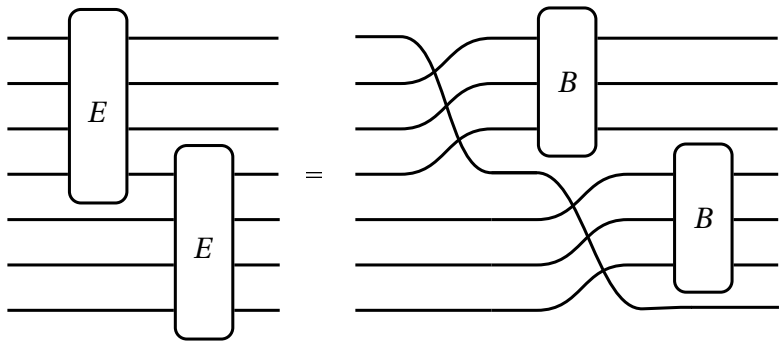


Figure 7.4: Concatenation of the example encoder; one qubit does not take part in the encoding, but is just passed on from one step to the next.

Exchanging two of the encoding steps interrupts the transport of the qubit to the end of the chain. To correct this error one needs an operation that has a support area which is linearly dependent on q, because the information on the memory qubit that

was no longer passed on has to be passed to the end of the chain in the correction operation. Therefore, this circuit can not be transfered into a finite depth circuit. However, by Definition 7.1.1 this error is not catastrophic, because it can be corrected with an operation of bounded length. Therefore Conjecture 7.1.2 is wrong if definition 7.1.1 is used.

Of course the presented encoder is an example without practical use, just constructed to refute Conjecture 7.1.2. Furthermore, catastrophic errors can only occur on one special qubit that is not used in the code. Nevertheless it is necessary to include this case in the definition of catastrophic errors to fix Conjecture 7.1.2.

Definition 7.1.4. *Let E_q be an operation to encode $n \cdot q$ qubits using a (n,k,e,m_E,τ) convolutional code. A catastrophic error is an error that affects $\mathcal{O}(1)$ qubits before the end of encoding, but influences at least one output (memory or data) for arbitrarily large q. Thus the error can only be corrected after the end of the transmission. An encoder which exhibits such errors is called catastrophic. Otherwise it is called non-catastrophic.*

This definition includes the old definition as well as errors like the one in the example. Another way to understand the new definition is the idea that in the absence of catastrophic error propagation finitely localized inputs can only affect outputs in a finite area independent of the number of iterations. Theorem 4.5.1 shows that causal operations only have a causal inverse if they are of finite memory depth. A catastrophic encoder has infinite memory depth. Therefore it does not have a causal inverse and can not be transformed into a finite depth circuit. Using the new definition we ruled out the counter example and the proof of Conjecture 7.1.2 holds.

In the encoder state diagram the example error would create self loops at the nodes corresponding to the affected memory qubit. According to the definitions introduced in [69] the encoding transformation is catastrophic, because the self loops have all possible combinations of physical and logical operators and thus weights. However, the nodes are not at all connected to the rest of the graph, so the state diagram immediately shows that the corresponding qubit can be removed from the encoding process leaving a non-catastrophic encoder which uses one qubit of memory less.

7.1.2 Non-catastrophic encoders in the memory channel implementation

Using the representation of convolutional encoders by Clifford memory channels we will now study which channels correspond to catastrophic encoders and which to non-catastrophic encoders. This leads to a new and efficient criterion for non-catastrophic convolutional encoders and encoder inverses.

Theorem 7.1.5. *A convolutional encoder is non-catastrophic if and only if the Memory channel representing it is strictly forgetful.*

Proof. By Theorem 3.1 a channel is strictly forgetful if and only if every finitely localized observable on the output system is mapped to an observable on the input system which is finitely localized independent of the number of channel uses q for large q. Obviously this is equivalent to the definition of a non-catastrophic encoder. □

Remark 7.1.6. *The proof uses neither the stabilizer structure of the code, nor the Clifford property of the channel. Thus, it holds for general convolutional encoders and the channels that implement them. This is in accordance with our results on memory channel implementations of causal operations in Chapter 4.*

This connection between the existence of catastrophic errors to the forgetfulness of the encoding (or inverse encoding) operation gives us an efficient criterion to check if a given convolutional encoder (encoder inverse) is catastrophic. According to Corollary 3.7.5 a Clifford memory channel is strictly forgetful if and only the matrix $\mathbf{s}_{\mathscr{M}}$ representing the memory-to-memory transformation is nilpotent. Therefore we only have to check if the submatrix $\mathbf{e}_{\mathscr{M}}$ is nilpotent to find out if a given encoder is catastrophic.

Corollary 7.1.7. *An inverse encoder D (encoder E) of a quantum convolutional stabilizer code is non-catastrophic if and only if the memory-to-memory submatrix $\mathbf{d}_{\mathscr{M}}$ ($\mathbf{e}_{\mathscr{M}}$) of the phase space matrix $\mathbf{d}$ ($\mathbf{e}$) representing the inverse encoder (encoder) is nilpotent.*

Proof. By Theorem 7.1.5 a convolutional encoder is non-catastrophic if and only if the channel implementing it is strictly forgetful. A convolutional stabilizer code is encoded and inverse encoded by a Clifford memory channel. By Corollary 3.7.5 a Clifford memory channel $\mathbf{s}$ is strictly forgetful if and only if its memory-to-memory submatrix in the phase space representation $\mathbf{s}_{\mathscr{M}}$ is nilpotent. This completes the proof. □

This criterion can be checked with an algorithm of polynomial complexity in the number of memory qubits if the encoding (inverse encoding) operation is given. As noted above, the criterion for catastrophic encoders introduced in [69] is based on the state diagram of the encoder which is exponential in the number of memory qubits and also has to be derived from the encoding (inverse encoding) operation. Therefore, the memory channel description of convolutional codes enabled us to understand catastrophic encoders more precisely and to develop an efficient criterion that did not exist so far. The results of this section have already been included in [34, 36]. In the next section we will show how to explicitly construct an encoding channel for a given stabilizer group again considering catastrophic errors.

7.2 Construction of the encoding channel

In Section 3.9.3 we showed how block stabilizer codes can be encoded using Clifford channels. Convolutional stabilizer codes can be encoded using Clifford memory channels in a similar way. Using the results from Chapters 3 and 4 the reasoning is straight forward: the $n-k$ basic stabilizer generators of a convolutional stabilizer code can be seen as the images of $n-k$ Pauli Z operators of a block of n qubits on a chain of qubits. The shifted generators are treated as images of shifted blocks accordingly. This partly defines a causal operation. The encoded Pauli operators are taken as the images of k pairs of X and Z on the unused k qubits in each block. The missing Pauli X operators are determined to complete the definition of the causal operation. If all the used Pauli products (i.e. the stabilizer generators, the encoded Pauli operators etc.) are finite, we can determine a memory channel that implements this operation by the algorithm presented in Section 4.3.1. Furthermore, by Theorem 4.5.1, a causal inverse exists. Thus an on-line decoder exists and can be determined using the method introduced in Section 4.5.1.

In this section we will present an algorithm to derive an encoding Clifford channel for a convolutional code from its stabilizer generators. We start by deriving a partly defined channel from the stabilizer generators. This channel will then be completed by determining encoded Pauli matrices and adding the corresponding transformations. We will study under which conditions the encoder is guaranteed to be non-catastrophic. Finally we will analyze the memory requirements of the encoder and the influence of different decoding schemes on the resources needed on the decoding side. The individual steps are illustrated by our example codes.

7.2.1 Encoding the stabilizer generators

Here we present an algorithm to derive a partly defined memory channel that implements a partly defined causal operation. We use this to derive a partly defined encoding channel from the stabilizer generators as to implement the first part of our algorithm to derive an encoding channel for a convolutional stabilizer code. As we define convolutional codes in terms of stabilizer generators and encoded Pauli matrices it is easier to consider a transformation that maps operators from the input side (Pauli Z on the ancilla qubits and Pauli X and Z on the data qubits) to operators on the output side (stabilizer generators and encoded Pauli matrices). Therefore, we actually construct a memory channel inverse that implements an anti-causal operation, the inverse encoding. At the end of the process of determining the encoding operation we invert the inverse encoding and obtain the encoding. We use the algorithm to derive a channel implementation of a causal operation introduced in Section 4.3.1 as a basis and adapt it to the case of partially defined anti-causal operations.

- The algorithm takes as input a list of k transformation rules $\xi_i \mapsto \eta_i$, $\xi_i \in \mathbb{Z}_2^{2n}$, $\eta_i \in \mathbb{Z}_2^{2n(\tau+1)}$. These rules partly describe a causal operation T. The operation is assumed to be translation invariant with respect to shifts by n qubits. We assume ξ_i to be elements of a standard basis of $\mathbb{Z}_2^{2n}$. For a convolutional code we would have $\xi_i \widehat{=} I^{\otimes(i-1)} \otimes Z \otimes I^{\otimes(n-i)}$.

- The algorithm returns the partly defined phase space matrix $\mathbf{s}$ of a channel S that implements the k input transformation rules. Instead of a partly defined phase space matrix one can also think of a set of transformation rules that partly defines the channel. The ordering of systems is in accordance with the usual convention for memory channels: $S : \mathfrak{A}_i \otimes \mathfrak{M} \to \mathfrak{M} \otimes \mathfrak{A}_{i-1}$.

- The algorithm uses the following steps
  - Generate all shifted images: shift η_i to the left by j blocks (jn qubits) for $0 \leqslant j \leqslant \tau$. Everything that is shifted out at the left side will just be cut, while the right side will be padded with I, respectively $(0,0)$ in the phase space picture. E.g. $XYZ \mapsto YZI$ respectively $(1,0,1,1,0,1) \mapsto (1,1,0,1,0,0)$. We obtain an extended set η_i, $1 \leqslant i \leqslant k(\tau+1)$.
  - Generate the (partly defined) phase space representation of $\mathfrak{M}$ by cutting every element η_i $1 \leqslant i \leqslant k(\tau+1)$ by n qubits on the left. We call the result $\tilde{\eta}_i$, $1 \leqslant i \leqslant k(\tau+1)$. Each $\tilde{\eta}_i$ is of length of at most $n\tau$ qubits. Actually, the last k phase space vectors $\tilde{\eta}_i$, $k\tau+1 \leqslant i \leqslant k(\tau+1)$ will always be 0, because the according η_i have only one non-zero block.
  - Now find a basis ζ_j, $1 \leqslant j \leqslant l$ for $\tilde{\eta}_i$ which is a subset of a standard basis. A standard basis is a basis such that the commutation relations are the same as the ones of $(1,0,\ldots)$, $(0,1,0,\ldots)$, etc. Let Ξ_ζ be the space spanned by the ζ_j. Then, following Section 4.4.1, we can see that $\Xi_\zeta = \mathrm{rad}\Xi_\zeta \oplus \Xi_\zeta/\mathrm{rad}\Xi_\zeta$ where the first summand is the maximally isotropic subspace of Ξ_ζ and the second summand is a symplectic space. Let d_c be the dimension of $\mathrm{rad}\Xi_\zeta$ and let d_s be the symplectic dimension of $\Xi_\zeta/\mathrm{rad}\Xi_\zeta$. Then, by the embedding procedure of Section 4.4.1, Ξ_ζ can be embedded into $\Xi_m = \mathbb{Z}_2^{2(d_s+d_c)}$. Thus we can identify our basis vectors ζ_j with elements ζ_j^m of the standard basis of $\Xi_m = \mathbb{Z}_2^{2m}$ where $m = d_s + d_c$ is the number of memory qubits needed for the channel.[1] It is in general not necessary to use a standard basis, but then the resulting phase space transformation would not be symplectic with respect to the standard symplectic form. Transforming an arbitrary basis into a standard basis can be carried out by the methods explained in Chapter 5

[1] Using this formula, the memory requirements of a convolutional encoder can be computed just from the stabilizer generators. Details will be presented in Section 7.2.4.

in Section 5.3.1 to derive commuting pairs of anticommuting operators from cut stabilizer operators.

- Decompose the $\tilde{\eta}_i$ in terms of the basis ζ_j and obtain η_i^m, $1 \leqslant i \leqslant k\tau$, each being a phase space vector of length l. Add 0 entries at the places where the corresponding standard basis elements are missing in ζ_j (e.g. we identify the standard basis elements that span $\Xi_\zeta/\mathrm{rad}\Xi_\zeta$ with the Pauli products $X_1, Z_1, \ldots, X_{d_s}, Z_{d_s}$ and the elements that span $\mathrm{rad}\Xi_\zeta$ with $X_{d_s+1}, \ldots, X_{d_s+d_c}$ Then the positions of $Z_{d_s+1}, \ldots, Z_{d_s+d_c}$ will be filled with 0). We then have phase space vectors of length $2m$ (m qubits).
- Determine the input-to-output transformation $\mathbf{s}_{i-o}$: take the n leftmost qubits of η_i, $1 \leqslant i \leqslant k$ and use them as columns of a matrix. Add undefined columns where the rule in the causal transformation is missing. This (partly defined) $2n \times 2n$-matrix is the input-to-output matrix $\mathbf{s}_{i-o}$.
- Determine the input-to-memory transformation $\mathbf{s}_{i-m}$: use the first k η_i^m as columns of a matrix. Again add undefined columns where the rule in the causal transformation is missing. This (partly defined) $2m \times 2n$-matrix is the input-to-memory transformation $\mathbf{s}_{i-m}$.
- Determine the memory-to-output transformation $\mathbf{s}_{m-o}$: take the leftmost n qubits of the memory basis ζ_j, $1 \leqslant j \leqslant l$ and use them as the columns of a matrix. Again fill with undefined columns. This partly defined $2n \times 2m$-matrix is the memory-to-input matrix $\mathbf{s}_{m-o}$.
- Determine the memory-to-memory transformation $\mathbf{s}_{m-m}$: take ζ_j, $1 \leqslant j \leqslant l$ and shift them to the left by one block (n qubits). Decompose them in terms of ζ_j, fill with 0 at the appropriate places, and use the resulting vectors as columns of a matrix. Again add undefined columns where needed. This $2m \times 2m$-matrix is the memory-to-memory matrix $\mathbf{s}_{m-m}$.
- Compose the four matrices to obtain $\mathbf{s}$:

$$\mathbf{s} = \left(\begin{array}{c|c} \mathbf{s}_{m-o} & \mathbf{s}_{i-o} \\ \hline \mathbf{s}_{m-m} & \mathbf{s}_{i-m} \end{array}\right). \tag{7.2}$$

In both obtaining the memory-to-input and the memory-to-memory transformation we used the translation invariance of the encoding operation S.

Let us now illustrate the algorithm using our example codes: we start with the two transformation rules of code 1:

$$\begin{aligned} ZII &\mapsto XXXXZY = \eta_1, \\ IZI &\mapsto ZZZZYX = \eta_2. \end{aligned}$$

From those we can construct the partly defined memory channel.

- Generate all shifted and cut stabilizers:

$$\begin{array}{ccc|cccccc} X & X & X & X & Z & Y & & & \\ Z & Z & Z & Z & Y & X & & & \\ & & & X & X & X & X & Z & Y \\ & & & Z & Z & Z & Z & Y & X. \end{array}$$

  Therefore we obtain the extended set η_i

$$\begin{aligned} \eta_1 &= XXXXZY, \\ \eta_2 &= ZZZZYX, \\ \eta_3 &= XZYIII \\ \eta_4 &= ZYXIII. \end{aligned}$$

- Now we cut each η_i by n qubits on the left to obtain the (partly defined) phase space representation of the memory algebra. We obtain

$$\begin{aligned} \tilde{\eta}_1 &= XZY, \\ \tilde{\eta}_2 &= ZYX. \end{aligned}$$

- Now we have to find a basis for the $\tilde{\eta}_i$, which is in this case a trivial task, as $\tilde{\eta}_1 = XZY$ and $\tilde{\eta}_2 = ZYX$ already form a basis. Therefore, $\zeta_1 = \tilde{\eta}_1$ and $\zeta_2 = \tilde{\eta}_2$. We identify $XZY \hat{=} X = \zeta_1^m$ and $ZYX \hat{=} Z = \zeta_2^m$. The space Ξ_ζ is a two-dimensional symplectic space. Thus it has symplectic dimension $d_s = 1$ and radical dimension $d_c = 0$. Therefore, the encoder needs one qubit of memory and the memory phase space is a two-dimensional symplectic space $\Xi_m = \mathbb{Z}_2^2$.

- Decomposing $\tilde{\eta}_i$ in terms of ζ_j we obtain $\eta_1^m = (1,0)$ and $\eta_2^m = (0,1)$.

- The n leftmost qubits of η_i, $i = 1,2$ form the columns of the partly defined input-to-output matrix

$$\mathbf{s}_{i-o} = \begin{pmatrix} ? & 1 & ? & 0 & ? & ? \\ ? & 0 & ? & 1 & ? & ? \\ ? & 1 & ? & 0 & ? & ? \\ ? & 0 & ? & 1 & ? & ? \\ ? & 1 & ? & 0 & ? & ? \\ ? & 0 & ? & 1 & ? & ? \end{pmatrix}.$$

  The according transformation rules are

$$\begin{aligned} ZII &\mapsto XXX, \\ IZI &\mapsto ZZZ. \end{aligned}$$

- The input-to-memory transformation is constructed from the first $k = 2$ vectors η_i^m: we obtain

$$\mathbf{s}_{i-m} = \begin{pmatrix} ? & 1 & ? & 0 & ? & ? \\ ? & 0 & ? & 1 & ? & ? \end{pmatrix},$$

  where we padded with unknown columns in the places with an unknown transformation rule. The according known transformation rules are

$$\begin{aligned} ZII &\mapsto X, \\ IZI &\mapsto Z. \end{aligned}$$

- The memory-to-output transformation is created from the leftmost n qubits of the memory basis ζ_j:

$$\mathbf{s}_{m-o} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 0 \end{pmatrix}.$$

  The respective rules are

$$\begin{aligned} X &\mapsto XZY, \\ Z &\mapsto ZYX. \end{aligned}$$

- The memory-to-memory transformation is constructed by shifting ζ_i to the left by one block. As the vectors ζ_i have a length of only one block the memory elements are both mapped to the identity on the memory and the matrix is

$$\mathbf{s}_{m-m} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

- Finally we assemble the matrices and obtain

$$\mathbf{s} = \begin{pmatrix} 1 & 0 & ? & 1 & ? & 0 & ? & ? \\ 0 & 1 & ? & 0 & ? & 1 & ? & ? \\ 1 & 0 & ? & 1 & ? & 0 & ? & ? \\ 0 & 1 & ? & 0 & ? & 1 & ? & ? \\ 1 & 0 & ? & 1 & ? & 0 & ? & ? \\ 0 & 1 & ? & 0 & ? & 1 & ? & ? \\ 0 & 0 & ? & 1 & ? & 0 & ? & ? \\ 0 & 0 & ? & 0 & ? & 1 & ? & ? \end{pmatrix}.$$

To obtain the actual encoding unitary this partly defined transformation has to be completed and inverted, because we map operators from the input side to operators on the output side and therefore describe the inverse operation.

Example code 2 is a bit more involved. Again we start with the stabilizer generators and obtain the anti-causal transition rules

$$\begin{aligned} ZII &\mapsto XXXIXZIXYXYZ = \eta_1, \\ IZI &\mapsto ZZZIZYIZXZXY = \eta_2. \end{aligned}$$

Following the algorithm we can construct the partly defined memory channel.

- First we generate all shifted stabilizers and obtain the set of phase space vectors η_i:

$$\begin{aligned} \eta_1 &= XXXIXZIXYXYZ, \\ \eta_2 &= ZZZIZYIZXZXY, \\ \eta_3 &= IXZIXYXYZIII, \\ \eta_4 &= IZYIZXZXYIII, \\ \eta_5 &= IXYXYZIIIIII, \\ \eta_6 &= IZXZXYIIIIII, \\ \eta_7 &= XYZIIIIIIIII, \\ \eta_8 &= ZXYIIIIIIIII. \end{aligned}$$

- Now we cut each η_i by n qubits on the left to obtain the (partly defined) phase space representation of the memory algebra. We obtain

$$\begin{aligned} \tilde{\eta}_1 &= IXZIXYXYZ, \\ \tilde{\eta}_2 &= IZYIZXZXY, \\ \tilde{\eta}_3 &= IXYXYZIII, \\ \tilde{\eta}_4 &= IZXZXYIII, \\ \tilde{\eta}_5 &= XYZIIIIII, \\ \tilde{\eta}_6 &= ZXYIIIIII. \end{aligned}$$

- Now we construct a standard basis ζ_j of $\tilde{\eta}_i$ using the algorithm introduced in

Section 5.3.1 on the commutativity matrix of $\tilde{\eta}_i$. We obtain the basis

$$\begin{aligned}
\zeta_1 &= IXZIXYXYZ,\\
\zeta_2 &= IZYIZXZXY,\\
\zeta_3 &= XXXIZXZXY,\\
\zeta_4 &= IYYZIIXYZ,\\
\zeta_5 &= IXXYIIZXY,\\
\zeta_6 &= YYYIXYXYZ.
\end{aligned}$$

The commutativity matrix has full rank, therefore the memory phase space $\Xi_m = \mathbb{Z}_2^6$ is a symplectic space of symplectic dimension 3; the encoder needs 3 qubits of memory.

- Decomposing $\tilde{\eta}_i$ in terms of ζ_j we obtain

$$\begin{aligned}
\eta_1^m &= (1,0,0,0,0,0) = XII,\\
\eta_2^m &= (0,1,0,0,0,0) = ZII,\\
\eta_3^m &= (1,1,0,1,1,0) = YZX,\\
\eta_4^m &= (1,0,0,1,0,0) = XZI,\\
\eta_5^m &= (0,1,1,0,0,0) = ZXI,\\
\eta_6^m &= (1,1,1,0,0,1) = YXZ.
\end{aligned}$$

- The n leftmost qubits of η_i, $i = 1,2$ form the columns of the partly defined input-to-output matrix. We obtain

$$\mathbf{s}_{i-o} = \begin{pmatrix}
? & 1 & ? & 0 & ? & ?\\
? & 0 & ? & 1 & ? & ?\\
? & 1 & ? & 0 & ? & ?\\
? & 0 & ? & 1 & ? & ?\\
? & 1 & ? & 0 & ? & ?\\
? & 0 & ? & 1 & ? & ?
\end{pmatrix},$$

where we padded with unknown columns in the places with an unknown transformation rule. The according transformation rules are

$$\begin{aligned}
ZII &\mapsto XXX,\\
IZI &\mapsto ZZZ.
\end{aligned}$$

- The input-to-memory transformation is constructed from the first $k = 2$ η_i^m:

we obtain

$$\mathbf{s}_{i-m} = \begin{pmatrix} ? & 1 & ? & 0 & ? & ? \\ ? & 0 & ? & 1 & ? & ? \\ ? & 0 & ? & 0 & ? & ? \\ ? & 0 & ? & 0 & ? & ? \\ ? & 0 & ? & 0 & ? & ? \\ ? & 0 & ? & 0 & ? & ? \end{pmatrix}.$$

The according known transformation rules are

$$\begin{aligned} ZII &\mapsto XII, \\ IZI &\mapsto ZII. \end{aligned}$$

- The memory-to-output transformation is created from the leftmost n qubits of the memory basis ζ_j:

$$\mathbf{s}_{m-o} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

The respective rules are

$$\begin{aligned} XII &\mapsto IXZ, \\ ZII &\mapsto IZY, \\ IXI &\mapsto XXX, \\ IZI &\mapsto IYY, \\ IIX &\mapsto IXX, \\ IIZ &\mapsto YYY. \end{aligned}$$

- The memory-to-memory transformation is constructed by shifting ζ_i to the left by one block. The results $\tilde{\zeta}_i$ are decomposed in terms of ζ_i. The transformation rules are $\zeta_i \mapsto \tilde{\zeta}_i$. They are expressed in the memory basis:

$$\begin{aligned} XII &\mapsto XZY, \\ ZII &\mapsto XZI, \\ IXI &\mapsto XZI, \\ IZI &\mapsto IYY, \\ IIX &\mapsto IZZ, \\ IIZ &\mapsto YZX. \end{aligned}$$

The corresponding matrix is

$$\mathbf{s}_{m-m} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}.$$

- Finally we assemble the matrices and obtain

$$\mathbf{s} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 1 & ? & 0 & ? & ? & ? \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & ? & 1 & ? & ? & ? \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & ? & 0 & ? & ? & ? \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & ? & 1 & ? & ? & ? \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & ? & 0 & ? & ? & ? \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & ? & 1 & ? & ? & ? \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & ? & 0 & ? & ? & ? \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & ? & 1 & ? & ? & ? \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & ? & 0 & ? & ? & ? \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & ? & 0 & ? & ? & ? \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & ? & 0 & ? & ? & ? \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & ? & 0 & ? & ? & ? \end{pmatrix}.$$

To obtain the actual encoding unitary this partly defined transformation has to be completed and inverted.

7.2.2 Encoded Pauli matrices

Finding valid encoded Pauli matrices for convolutional codes is a more demanding task than for block codes. They do not only have to commute with stabilizer generators from their own block, but also with generators from all other blocks and furthermore commute with their own translates. The latter turns out to be the hardest requirement, as it leads to non-linear equations. As we have seen in Section 3.10.2 known schemes to determine encoded Pauli matrices are not always able to produce non-catastrophic encoders because a special solution to the non-linear self-commutation Equation 3.108 is assumed.

One approach to find non-catastrophic encoders would be to try to solve Equation 3.108 without a special simplified ansatz. However, here we will use a different approach. We will start from the partly defined channel implementing the encoding operation and complete it to include the encoded Pauli matrices.

From Lemma 3.6.11 we know that we can complete the partly defined encoding transformation **e** to a symplectic isomorphism. Therefore we can complete the encoding channel that is just defined on the stabilizer generators to a unitary Clifford

transformation. This immediately gives us encoded Pauli matrices. However, it is not clear that these encoded Pauli matrices are finite and that the resulting encoder is non-catastrophic. In the following section we will show under which circumstances we can prove that a non-catastrophic encoder exists.

7.2.3 Existence of non-catastrophic encoders

We have shown in the last sections that non-catastrophic encoders are a key requirement for successful quantum convolutional coding. However, finding non-catastrophic encoders for general convolutional codes is still an open problem. Although a general theory is still to be developed, there are some results on the existence of non-catastrophic encoders in the literature. In [111] it is shown that for every convolutional code derived from a self orthogonal classical convolutional code one can find a subcode which allows a non-catastrophic encoding. More recently it was shown that all completions of partially defined encoders with a full rank memory commutativity matrix are non-catastrophic [48]. Furthermore it was shown that partially defined encoders which have no edges of the form $M \otimes L \otimes S^z \mapsto \mathbb{1}^{\otimes n} \otimes M$ in their state diagram can be completed to non-catastrophic encoders.

Using our channel description of convolutional stabilizer codes we have not yet proved more, but we can easily prove the case of a full rank memory commutativity matrix using the algorithm introduced in Section 7.2.1. The memory commutativity matrix c is a $l \times l$ binary matrix where $c_{ij} = 0$ if memory basis elements i and j commute and $c_{ij} = 1$ if they anti-commute. The matrix dimension l stems from the number of independent memory elements. The commutativity matrix can be computed for every set of multiplicatively independent Pauli products, so we do not need the complete description of the memory, but can also compute the commutativity matrix of the generators of a partly defined memory algebra. Obviously, the commutativity matrix c_{ij} of a multiplicatively independent set of Pauli operators $\mathrm{w}(\xi_i)$ is nothing more than the symplectic form restricted to the phase space vectors representing the operators: $c_{ij} = \sigma(\xi_i, \xi_j)$. Therefore the dimension of the commutativity matrix equals the dimension of the memory phase space Ξ_ζ; $\dim \Xi_\zeta = 2d_s + d_c$. The rank of c_{ij} equals the dimension of the symplectic part of Ξ_ζ: $\mathrm{rank}(c_{ij}) = \dim(\Xi_\zeta / \mathrm{rad}\Xi_\zeta) = 2d_s$. If c_{ij} has full rank we have $d_c = 0$ and the vectors ζ_i form a full standard basis of $\mathbb{Z}_2^{2d_s}$. Therefore, the memory phase space is already complete and the memory-to-memory transformation $\mathbf{e}_{\mathscr{M}}$ is already completely defined. By construction the memory-to-memory transformation $\mathbf{e}_{\mathscr{M}}$ obtained from the stabilizer generators is nilpotent, because they are finitely localized. Completing the partly defined encoding operation can not change a completely defined $\mathbf{e}_{\mathscr{M}}$, therefore the complete encoding channel is strictly forgetful and the encoder therefore non-catastrophic.

Coming back to our example codes we notice that both codes have $d_c = 0$ and thus all possible completions of the encoder matrix are non-catastrophic. There-

fore, we can use the completion algorithm (Section 3.6.2) to find non-catastrophic encoder.

The transformation of code 1 can be extended with the rules

$$\begin{aligned} IIIX &\mapsto IXZY, \\ IIIZ &\mapsto YIZX. \end{aligned}$$

This leads to the encoded Pauli operators

$$\begin{aligned} \bar{X} &= IXZYXZ, \\ \bar{Z} &= YIZXZY. \end{aligned}$$

In the case of code 2 valid extending rules are

$$\begin{aligned} IIIIIX &\mapsto YZIXIX, \\ IIIIIZ &\mapsto IZXYZX. \end{aligned}$$

The corresponding encoded Pauli operators are

$$\begin{aligned} \bar{X} &= YZIIIYZIZYIIZXY, \\ \bar{Z} &= IZXYZXXYZ. \end{aligned}$$

In both cases we find non-catastrophic encoders while the algorithm introduced in [71] fails to do so. So while our algorithm is not guaranteed to produce a non-catastrophic encoder in all cases it does output non-catastrophic encoders in cases where the algorithm introduced in [71] fails to do so. To complete out encoders we would need to determine the last missing transformations, but as their existence is guaranteed this does not give us any new information and we omit this step here.

Once we have determined an encoding channel it is an easy task to obtain a causal inverse. If the encoding channel is strictly forgetful the encoding transformation is a finite depth causal operation. Therefore, by Theorem 4.5.1, a causal inverse exists. This inverse can be constructed as described in Section 4.5.1.

7.2.4 Memory requirements

The resource requirement of a convolutional coding scheme is a crucial factor for its implementation. Quantum computers still only work on very few qubits. Parallel qubits and quantum memory are likely to stay costly resources in the future. Only recently, the first studies on the memory requirements of quantum convolutional codes have been published: [46, 47, 48, 95]. In [48] the authors present a method to determine the needed memory for a convolutional encoder from the memory commutativity matrix. Here we state an equivalent criterion and show how to quickly obtain a representation of the memory algebra.

Lemma 7.2.1. *Let $\mathcal{S} = \mathrm{sp}\{S_{j,i} = I^{\otimes j\cdot n} \otimes S_{0,i}, 1 \leqslant i \leqslant n-k, 0 \leqslant j\}$ with $S_{0,i} = \mathrm{w}(\eta_i)$ be the stabilizer group of a quantum convolutional code. Furthermore, let $\tilde{\eta}_i$, $1 \leqslant i \leqslant (n-k)\tau$ be the set of all stabilizer generators cut by $1,\ldots,\tau-1$ blocks from the left. Let Ξ_ζ be the space spanned by those η_i and let ζ be a basis of Ξ_ζ. Then the minimal memory of an encoding channel for the code defined by $\mathcal{S}$ is $m = d_s + d_c$, where d_s is the symplectic dimension of $\Xi_\zeta/\mathrm{rad}\Xi_\zeta$ and d_c is the dimension of $\mathrm{rad}\Xi_\zeta$.*

Proof. This result has already been proven in Section 7.2.1 in the third step of the algorithm to determine the enocoding channel. □

This criterion is easy to check, because is suffices to find a basis for the cut stabilizers. This step is also necessary to check the memory commutativity matrix criterion. Actually, our criterion is exactly the same as the memory commutativity matrix criterion. From the formalism of Clifford causal operations introduced in Chapter 4, we know that the cut images form a representation of the memory algebra. The cut stabilizer operators therefore form a representation of a part of the whole memory algebra. The same is true for the partly defined memory basis used in [48]. The only difference is the representation. The cut stabilizers form a representation on $\mathbb{C}^{2\tau}$, while the memory basis in [48] is a representation on $\mathbb{C}^{2m}$. To compute the dimension of $\Xi_\zeta/\mathrm{rad}\Xi_\zeta$ and $\mathrm{rad}\Xi_\zeta$ we use the symplectic form restricted to Ξ_ζ, which is the memory commutativity matrix: $c_{i,j} = \sigma(\eta_i,\eta_j)$. This gives us $d_s = 1/2\mathrm{rank}c_{i,j}$ and $d_c = \dim c_{i,j} - \mathrm{rank}c_{i,j}$. Therefore, $m = d_s + d_c = \dim c_{i,j} - 1/2\mathrm{rank}c_{i,j}$, which is exactly the formula derived in [48]. The two methods to derive the needed memory are therefore equivalent in the result, while our method uses a simpler algorithm which does not require do determine the memory dynamics first.

In our resource analysis we will not restrict ourselves to the internal memory of the encoding and decoding process, but the total resources needed by quantum computers to carry out the encoding and decoding processes. Our aim is to produce a basis to compare the performance of block codes to the performance of convolutional codes. As introduced in Section 3.4 for the implementation by a quantum computer the total spatial resources, i.e. the number of qubits that have to be stored and operated on at the same time, are a limiting factor. Therefore, we will compare the performance of codes that use the same spatial resources. To that end we need to determine the resources needed for block codes and convolutional codes. To simplify the analysis we will not consider the internal structure of the input operation which might reduce the memory usage for some codes. The resource requirements for block codes are illustrated in Figure 7.5; for convolutional codes see Figure 7.6.

The resource requirements on the encoder side are straight forward: for a block code we need n qubits. For a convolutional code we need $n+m$ qubits. On the decoder side a block code again needs n qubits. For a convolutional code the resource

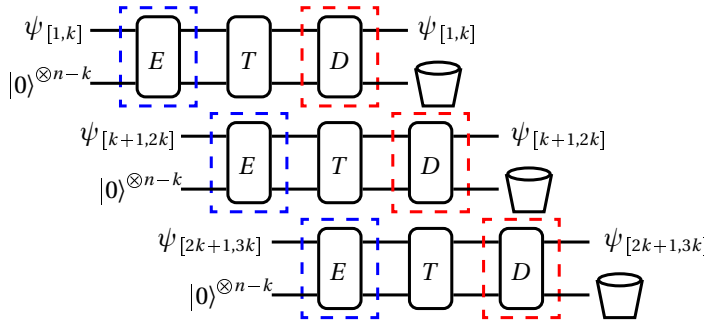


Figure 7.5: Resource requirements of a block code; encoder and decoder both need n qubits.

requirements depend on the chosen model of error-correction: if we measure the syndromes before inverting the encoding we need $n(\tau+1)$ qubits because we have to jointly measure all $n(\tau+1)$ qubits of the stabilizer generators (See Figure 7.7).

If we choose to first invert the encoding and then measure the "decoded" stabilizer generators to obtain the syndrome, the syndrome measurement only affect single qubits. Therefore, either the encoding inverse, or the correction operation are the most resource intensive operations. To obtain the resources needed for the inverse encoding we can employ the index formula (Section 4.5.2): $m_d = n\tau - m_e$, where m_d is the number of memory qubits on the decoder side and m_e on the encoder side. As we processed the uncorrected information in the inverse encoder, the error can spread on several output qubits. Each block has k output qubits; due to the forgetfulness of the inverse encoding channel at most $\tau+1$ blocks can be affected. Therefore, the correction operations affect up to $k\tau$ qubits at once. In total the decoding quantum computer needs to operate on at least $\max(n+m_d, k(\tau+1)) = \max(n+n\tau-m_e, k(\tau+1))$ qubits. For an illustration see Figure 7.8. We will always use the second decoding scheme as it is less resource intensive.

The memory requirement derived from the stabilizer depends on the representation of the stabilizer group. Multiplying stabilizer generators does not change the stabilizer group. However, replacing a stabilizer generator S_1 by its product with a shifted copy of another generator S_2 can enlarge its support. We can easily construct an extreme case to show that this can affect the needed memory. Imagine that S_2 is shifted far enough that the resulting generator is just the original generator S_1 followed by l blocks of identity elements and then the shifted generator S_2. To encode it we need a series of independent memory transformation rules that produces no output for l blocks before it outputs S_2. Those rules need l qubits of

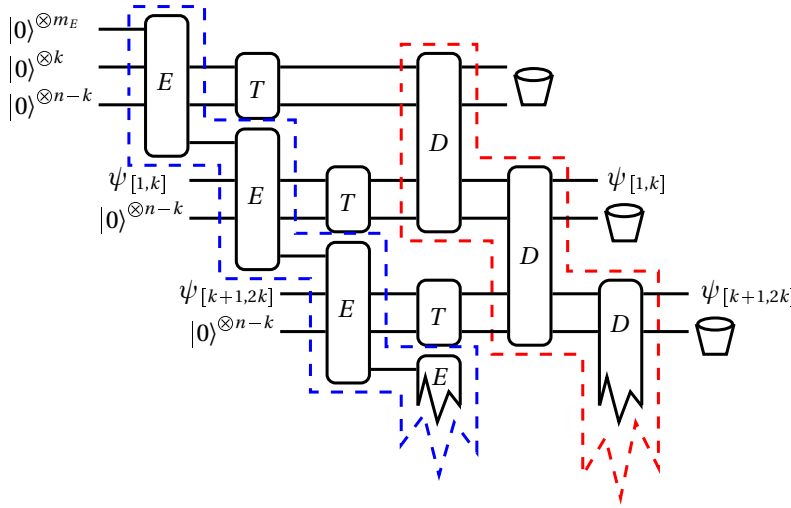


Figure 7.6: Resource requirements for a convolutional code; the encoder needs $n+m_e$ qubits, while the decoder needs $n+m_d$ qubits. The figure shows the beginning of the encoding, where the memory is initialized with a given state.

memory. However, we derived an algorithm to find a set of stabilizer generators that minimizes the memory requirements in cooperation with the authors of [48]. The algorithm is presented in [48] and will not be repeated here.

7.3 Quantum convolutional Hamming bound

For quantum block codes the quantum Hamming bound [112] constitutes an upper bound on the number of arbitrary single qubits errors t that can be corrected when k data qubits are encoded into n physical qubits. In other words, the bound determines the highest t such that all arbitrary t-qubit errors can be corrected. It thus also gives a bound on the code-rate if we bound the block-size and the error rate $e = t/n$. Given the Hamming bound one can search for codes that saturate this bound and thus operate on the limit of what is theoretically achievable. Such codes are called perfect.[2] In this section we will derive similar bounds for convolutional codes under different assumptions on the occurring errors.

[2] If special assumptions on the occurring errors are made the construction of codes that can correct more errors might be possible. However, the search for such codes has been without success so far to our knowledge. See also [113].

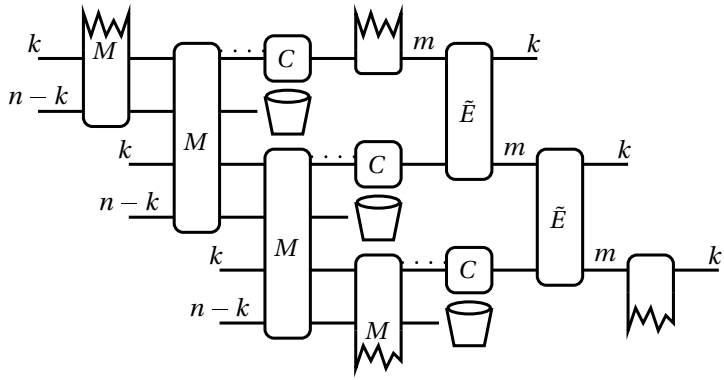


Figure 7.7: In this decoding scheme first the stabilizer generators are measured. Then the detected errors are corrected and finally the encoding is inverted. The dotted lines represent transfer of classical information.

In the case of block codes, the blocks are independent, so the error-correction capabilities of one bock do not depend on the errors in previous blocks. In the case of convolutional codes however, the blocks of stabilizer generators overlap. Therefore, the syndrome measurement for one block is also influenced by errors in all blocks which support the stabilizer generator that is measured. We distinguish different error-scenarios with the following extremal cases:

- Every block of n qubits may contain up to t errors which can all be corrected perfectly. This setting is similar to the block code error-scenario.

- After one block with errors several error-free blocks have to follow to ensure that the error can be corrected. This setting makes use of the overlapping blocks of stabilizer generators used in convolutional codes. In this case for perfect error-correction t arbitrary single qubit are allowed in a block, given that the τ preceeding and following blocks are error-free. Therefore, the erroneous blocks need to have a distance of at least the length of a stabilizer generator $(\tau+1)$. This setting has the potential to allow for codes with higher rates than block codes, because these can not make use of the error-free blocks. If we think of a transmission channel with uncorrelated errors only $t = 1$ makes sense. However, in the case of transmission channels, where the errors are correlated and always come in pairs, $t = 2$ makes perfect sense.

Starting from these extremal cases we will study a setting where we allow a fractional rate of errors per block.

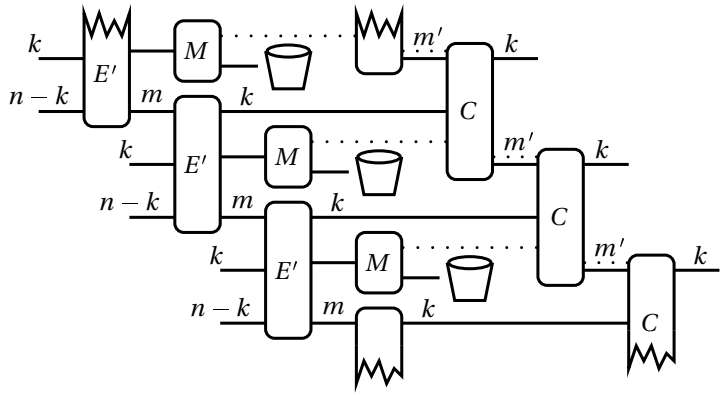


Figure 7.8: In this decoding scheme first the encoding is inverted. the the decoded stabilizer generators are measured and finally the detected errors are corrected. The dotted lines represent transfer of classical information.

7.3.1 The quantum Hamming bound for block codes

Let us start with the well-known quantum Hamming bound for non-degenerate block codes. Let us consider an $[n,k,d]$ code, a code that encodes k data qubits into n physical qubits and has a minimal distance of d, thus correcting $\lfloor (d-1)/2 \rfloor$ single qubit errors per block. As introduced in Section 3.9.1 for an error to be correctable by a non-degenerate code its action has to be distinguishable from the action of all the other errors (including no errors at all). We distinguish the errors by measuring the stabilizer generators. The outcomes $(+1,-1)$ generate the error syndromes. To distinguish all errors we need at least one syndrome per error. The number of possible error syndromes is 2 to the power of the number of stabilizer generators: 2^{n-k}. On the other hand we calculate the number of possible errors considering independent errors on up to t qubits at once. If we have i errors, there are $\binom{n}{i}$ possible combinations of locations. Each error can be X, Z or Y (all the other errors will be projected on the Pauli operators by the syndrome measurements, so we only have to consider this basis, see Section 3.9.3). So there are $\binom{n}{i}3^i$ different i-qubit errors. All summed up we have $\sum_{i=0}^{t}\binom{n}{i}3^i$ different errors that have to be correctable and need a unique syndrome. If the condition

$$\sum_{i=0}^{t}\binom{n}{i}3^i \leqslant 2^{n-k} \tag{7.3}$$

is not met, a $[n,k,2t+1]$ code cannot exist. This condition is called the quantum Hamming bound. The proof of the quantum Hamming bound only holds in the

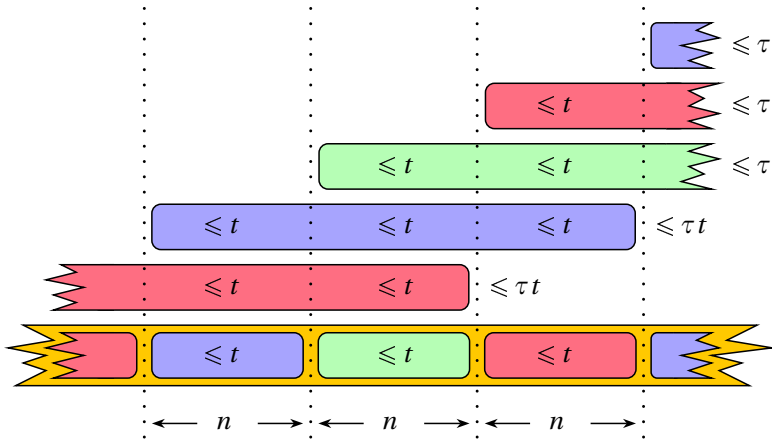


Figure 7.9: Error distribution scenario for the high error convolutional quantum hamming bound

case of non-degenerate codes, because we assumed that every error needs a unique correction operation and therefore a unique syndrome for its identification. For degenerate codes this is not necessary. However, there are no known degenerate codes that violate the quantum Hamming bound.

7.3.2 The convolutional Hamming bound in the high error case

In the high error regime we consider t errors per block of stabilizer generators. As the blocks of generators overlap we can not study individual blocks. So we consider q blocks and then use the limit $q \to \infty$ to find the asymptotic behavior for long transmissions. The code has to correct all error-pattern with up to $t \cdot q$ errors that contain at most t errors in every block. Thus in every block we have the same situation as in the block code case. Thus there are $\sum_{i=0}^{t} \binom{n}{i} 3^i =: e_b$ possible error patterns in every block. Each block can contain one of those error-patterns. This gives us e_b^q different error patterns in total. Again we need more possible error syndromes than possible error patterns to distinguish and correct the erros. When counting the possible error syndromes, we have to take the overlap of the stabilizers into account. The stabilizer generators have a length of $\tau + 1$ blocks. In each block we have a set of $n-k$ stabilizer generators, but also τ sets of $n-k$ stabilizer generators each that originate in other blocks, but overlap with the block we are considering. If we consider q blocks, only the boundaries of the region overlap with stabilizers from other blocks. Thus we have $(q+\tau)(n-k)$ stabilizer generators that can be affected by the errors in the region and can thus be used to determine the

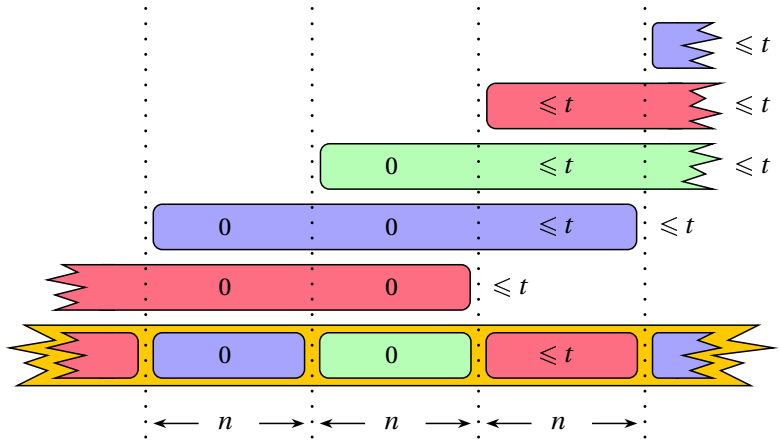


Figure 7.10: Error distribution scenario for the low error convolutional quantum hamming bound

error syndrome. This amounts to a total of $2^{(q+p\tau)(n-k)}$ error syndromes. Therefore we obtain

$$e_b^q \leqslant 2^{(q+\tau)(n-k)}.$$

Taking the limit $q \to \infty$ for large regions we obtain

$$\begin{aligned} \Leftrightarrow \lim_{q\to\infty} e_b^q &\leqslant \lim_{q\to\infty} \left(2^{n-k}\right)^{q+\tau} \\ \Leftrightarrow \lim_{q\to\infty} e_b^q &\leqslant \lim_{q\to\infty} \left(2^{n-k}\right)^{q} \left(2^{n-k}\right)^{\tau} \\ \Leftrightarrow e_b &\leqslant \lim_{q\to\infty} 2^{n-k} \left(2^{n-k}\right)^{\tau/q}. \end{aligned}$$

The high-error quantum convolutional Hamming bound therefore is

$$\Leftrightarrow \sum_{i=0}^{t} \binom{n}{i} 3^i \leqslant 2^{n-k}. \tag{7.4}$$

Apparently the convolutional quantum hamming bound for high error rates and the block quantum Hamming bound are the same. Therefore, for fixed n and k and errors in every block convolutional codes can correct the same number of errors as block codes.

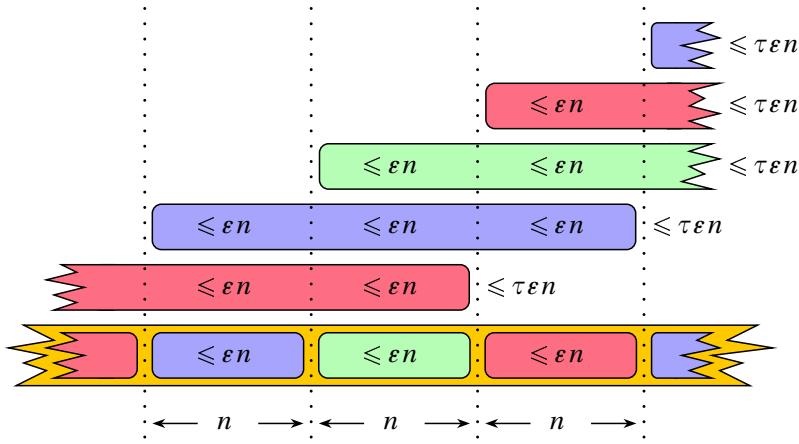


Figure 7.11: Error distribution scenario for the arbitrary error convolutional quantum hamming bound

7.3.3 The convolutional Hamming bound in the low error case

Now let us consider the case where an erroneous block is followed by τ error-free blocks. For block codes this does not give us any advantage, because all blocks are independent. But for convolutional codes we can make use of the overlapping stabilizers. We consider a block containing up to t errors.[3] Thus we have to deal with $\sum_{i=0}^{t} \binom{n}{i} 3^i$ different errors. In each block $\tau+1$ blocks of stabilizer generators overlap. So we can use $(n-k)(\tau+1)$ stabilizer generators to determine the error syndrome. If we count all the generators whose support overlaps with the block under consideration, we have to make sure that there are no other blocks with errors within the support of any of these generators. That is why we require an erroneous block to be followed by τ error-free blocks. The Hamming bound for this case is

$$\sum_{i=0}^{t} \binom{n}{i} 3^i \leqslant 2^{(n-k)(\tau+1)}. \tag{7.5}$$

Enlarging the space between individual errors to more then τ error-free blocks would not allow for higher rates, because then there would be stabilizer generators which would not intersect with any error and thus could not be used for the syndrome measurement. This is similar to the case of block codes where the bound does not increase if we consider less then one error per block.

[3] As said before $t > 1$ only makes sense if we assume correlated errors.

7.3.4 The general quantum convolutional Hamming bound

Now we will generalize the high error bound to the case of an arbitrary number of errors. We consider an error rate of $0 < \varepsilon < 1$ errors per qubit and therefore $\varepsilon n\tau$ errors in τ consecutive blocks. It does not make sense to considers more than $\tau+1$ consecutive blocks, because then some individual errors might not be in the support of all of the stabilizers of the consecutive blocks and the error counting and the syndrome counting would not work. Thus the code has to be able to correct $t = \lceil \varepsilon n(\tau+1) \rceil$ errors in $\tau+1$ consecutive blocks. On a longer series of qubits all error patterns can be corrected that allow a partitioning in macroblocks consisting of $\tau+1$ blocks each such that every macroblock contains at most t errors. In another partitioning there might be a macroblock that contains more than t errors, but it will be preceded and followed by blocks with less than t errors. The overlap of stabilizers between adjacent macroblocks allows to correct more than t errors in an isolated block in this case.

The number of possible error patterns in a macroblock is

$$e_{\tau+1} = \sum_{i=0}^{t} \binom{n(\tau+1)}{i} 3^i. \tag{7.6}$$

There are $(n-k)(\tau+1+\tau)$ stabilizers that overlap with any given macroblock, where the last τ comes from stabilizers from adjacent blocks. Now we use the same approach as in the high error scenario, just using macroblocks instead of individual blocks this time. We consider q macroblocks and study the limit $q \to \infty$: we have

$$\begin{aligned}
e_{\tau+1}^q &\leqslant 2^{(n-k)(\tau+1)q+(n-k)\tau} \\
\Leftrightarrow \lim_{q\to\infty} e_{\tau+1}^q &\leqslant \lim_{q\to\infty} 2^{(n-k)(\tau+1)q+(n-k)\tau} \\
\Leftrightarrow \lim_{q\to\infty} e_{\tau+1}^q &\leqslant \lim_{q\to\infty} \left(2^{(n-k)(\tau+1)}\right)^q + \left(2^{(n-k)\tau/q}\right)^q \\
\Leftrightarrow \lim_{q\to\infty} e_{\tau+1} &\leqslant \lim_{q\to\infty} 2^{(n-k)(\tau+1)} + 2^{(n-k)\tau/q} \\
\Leftrightarrow e_{\tau+1} &\leqslant 2^{(n-k)(\tau+1)}.
\end{aligned}$$

Therefore, the general quantum convolutional Hamming bound is

$$\sum_{i=0}^{t} \binom{n(\tau+1)}{i} 3^i \leqslant 2^{(n-k)(\tau+1)}. \tag{7.7}$$

This generalized bound does not cover the special low error case introduced above. Here we only require the existence of a partition into macroblocks such that each macroblock only contains up to t errors, while in the low error case we require each erroneous block to be followed by τ error-free blocks, which is a much stronger assumption. However for sufficiently low error rates e this assumption is justified, because the probability of an error which is not correctable is low.

7.4 Performance of convolutional codes under limited resources

In this section we analyze the maximal theoretical performance of convolutional and block codes under resource constraints using the Hamming bounds. The resource we consider is the total quantum memory of a quantum computer used for encoding (and decoding) as introduced in Section 3.4 and discussed for convolutional codes in Section 7.2.4. The performance measure we consider is the code rate for a given correctable error-rate. The decoding complexity, i.e. the complexity of the algorithm that determines the necessary correction operation given the error-syndrome, is not part of our analysis. For a block code the resource requirement on input and output side are n qubits. For convolutional codes we need $n + m_e$ qubits on the input side and $m_d = \max(n(\tau+1) - m_e, k(\tau+1))$ on the output side. Thus we compare block codes with n qubit blocks to convolutional codes with $n' = n - m_e$ qubit blocks if we consider the encoder $n' = n - m_d$ quit blocks if we consider the decoder.

Let us first consider an asymmetric scenario, where the encoding is conducted on a quantum computer with limited resources while the decoding does not have any limitations. Such a scenario would make sense for example in the transmission of quantum information from a satellite to a ground station.

To compare block and convolutional codes we consider a rate of e errors per qubit. Therefore, a block code with block length n has to correct $t = \lceil n \cdot e \rceil$ errors per block, while a convolutional code with block length n and memory depth τ has to correct $\tilde{t} = \lceil n(\tau+1) \cdot e \rceil$ errors per $\tau + 1$ blocks. In Figures 7.12 and 7.13 we show the maximal rate allowed by the Hamming bounds for a range of combinations of e and n respectively $n+m$. The comparison is made between codes that use the same total input resources. A block code with n_b qubits is compared to a convolutional code using $n_c + m = n_b$ qubits as input resources.

A comparison of the rates for convolutional and block codes using the same input resources (in terms of qubits) is shown in Figure 7.14. The figure shows that in the considered asymmetric scenario the convolutional bound is usually higher than the block bound. In the cases where the block code bound is higher than the convolutional bound $n \cdot e$ is close to an integer. In the other scenarios the block codes can correct more errors than they are required to, because they are always able to correct an integer number of errors per block. A convolutional code does not have this restriction and can therefore perform better in those settings.

Now let us consider the symmetric scenario where the bound on $n + m$ has to hold both for the encoder and the decoder. Given $n + m$ we consider all codes where both encoder and decoder use at most $n + m$ qubits of memory. In this case the index formula $n\tau = m_e + m_d$ and the error-correction restrict the memory depth τ for a given resource bound $n + m$ and therefore the parameter range

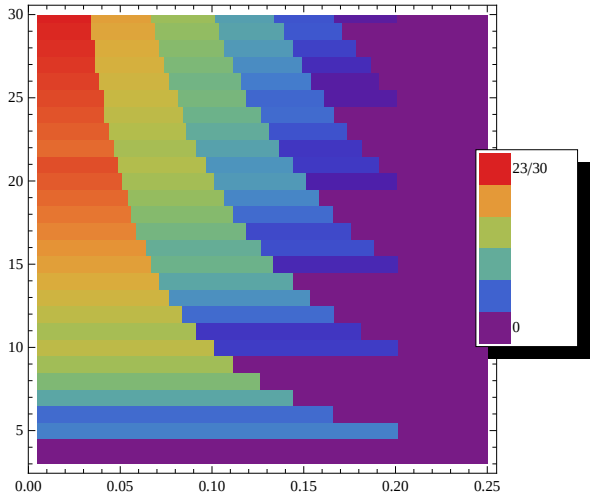


Figure 7.12: Maximal rate allowed by the Hamming bound given the size of the input operation n (vertical) and an error probability e (horizontal); the code has to be able to correct $\lceil n \cdot e \rceil$ errors per block.

for the codes. Figure 7.15 shows the code rates for given error rates and resource bounds. Comparing with the asymmetric case (Figure 7.13) one can easily see that the rates are in general lower. Especially in the case of very strict resource limitations and the case of very high error rates the performance of convolutional codes in the asymmetric case and the performance of block codes can not be met. In the case of high error rates the explanation is simple: the code rate k/n has to be small and $n-k$ has to be large, so there are many stabilizer generators per data qubit. This also makes n large. But the index formula requires that $m_e + m_d \geqslant n$ and therefore the resource requirements are at least $n + n/2$ qubits. In the asymmetric case the encoder has the freedom of only using one qubit of memory, shifting the majority of the needed memory to the decoder that is not considered in the asymmetric resource analysis. In the case of low resources the asymmetry of the memory requirements as governed by the index formula comes into play. On the one hand the length of the stabilizers has to be maximized with respect to the encoder memory to enable error-correction, but on the other hand this leads to a high resource requirement on the decoder side. Therefore, for low resource scenarios on both encoder and decoder side convolutional codes can not use their potentially long stabilizers generators to outperform block codes.

Comparing convolutional codes with block codes we observe that while convolu-

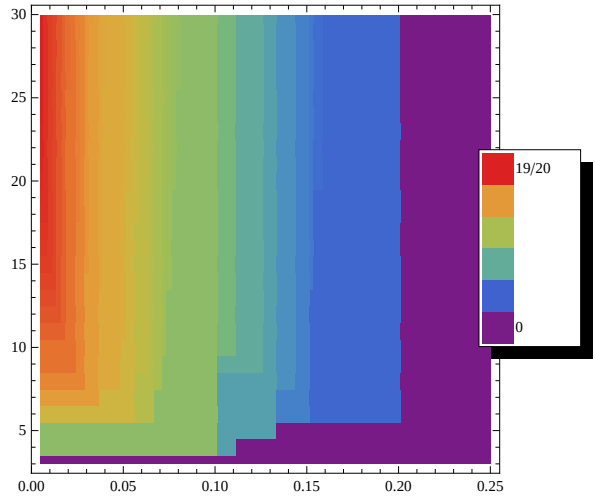


Figure 7.13: Maximal rate allowed by the convolutional Hamming bound given the size of the input operation $n+m$ (vertical) and an error probability e (horizontal); the code has to be able to correct $\lceil n(\tau+1)\cdot e \rceil$ errors in any macroblock of $\tau+1$ consecutive blocks, where $n(\tau+1)$ is the length of the stabilizer generators.

tional codes have the potential to better perform in the asymmetric case than in the symmetric case they still have the potential to outperform block codes for a wide range of error rates. This is shown in Figure 7.16.

Concluding, we can state that both in an asymmetric and in a symmetric setting convolutional codes have the potential to outperform block codes in terms of code rate under resource restrictions on the encoder respectively encoder and decoder. Their potential is especially high in an asymmetric setting. Convolutional codes make use of their long stabilizer generators when the correctable error rate is such that the number of correctable errors per block is not an integer.

However, we have to note that the bounds given here are upper bounds which do not guarantee the existence of codes that saturate them. For block codes so called perfect codes that saturate the quantum hamming bound are know [114, 115, 116]. In the following we will show that our example codes violate the block quantum Hamming bound and therefore outperform any possible block code with the same input resources.

Example code 1 is a rate $1/3$ $(3,1,3,1,1)$ $((n,k,d,m,\tau))$ convolutional code. The block quantum Hamming bound tells us that a block-code with comparable prop-

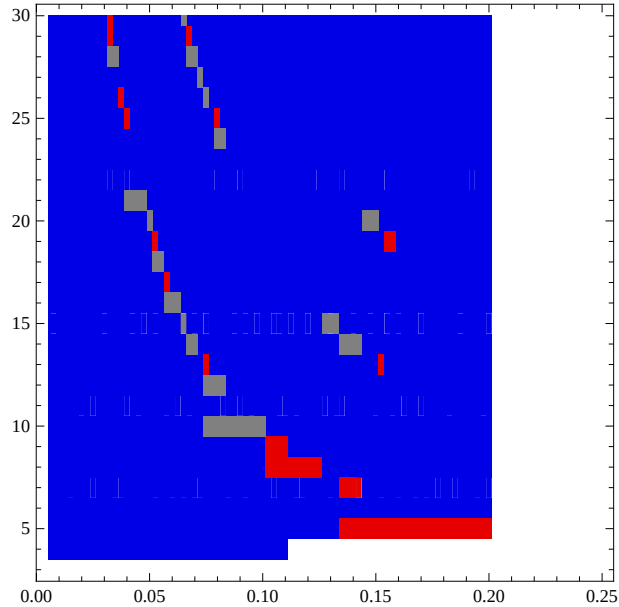


Figure 7.14: Comparison of the block and convolutional hamming bound in an asymmetric scenario; In the blue region the convolutional bound is higher, while in the red region the block bound is. In the gray areas the bound are equal.

erties (a $[4,1,3]$ code) does not exist. Therefore, this code really outperforms block codes in a limited encoder resource setting. However if we consider the resources used by the decoder we notice, that the code also exceeds our symmetric convolutional bound: The encoder memory is $m_e = m = 1$. Together with $n = 3$ and $\tau = 1$ this gives a decoder memory of $m_d = 2$ qubits. Therefore, the total needed decoder resources are 5 qubits. Figure 7.15 tells us that such a code should not exist. This contradiction comes from the fact that our general bound does not consider the special assumptions on the error distribution used for this code. Therefore the general convolutional Hamming bound does not apply to this code. The low error bound is not violated, as

$$\sum_{i=0}^{t} \binom{n}{i} 3^i = \sum_{i=0}^{1} \binom{3}{i} 3^i = 1 + 3 \cdot 3 \leqslant 2^4 = 2^{(3-1)(1+1)} = 2^{(n-k)(\tau+1)}.$$

The second example is a rate $1/3$ $(3,1,5,3,4)$ code which corrects all error patterns that have at most two erroneous sites in the support of each stabilizer genera-

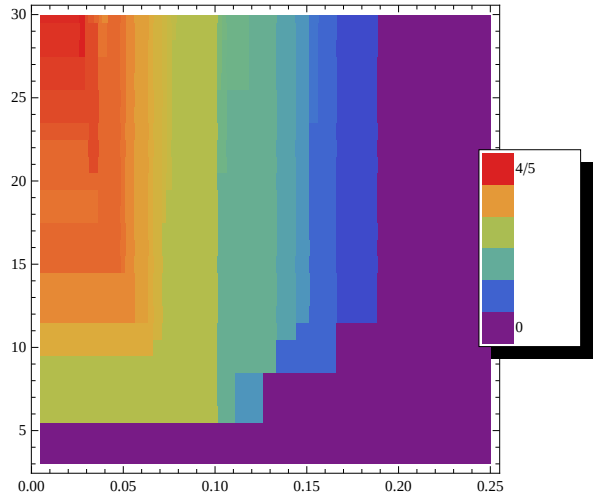


Figure 7.15: Maximal rate allowed by the general convolutional Hamming bound given a bound on the size $n+m$ of the input and output operations (vertical) and an error probability e (horizontal); the code has to be able to correct $\lceil n(\tau+1)\cdot e\rceil$ errors in any macroblock of $\tau+1$ consecutive blocks, where $n(\tau+1)$ is the length of the stabilizer generators.

tor. No block code which operates on only 6 qubits has rate $1/3$, no matter how low the error rate. Therefore the second example also exceeds the block quantum Hamming bound. The resources on the output are quite large, because the high memory depth $\tau=3$ gives us a decoder memory of $m_d = n\tau - m_e = 6$. Therefore, on the decoder side 9 qubits are needed. In this case the convolutional quantum Hamming bound allows for convolutional codes with rate $1/3$ and maximal encoder resource requirements of 6 qubits respectively maximal overall resource requirements of 9 qubits. Their error-correction capabilities can be even better than the 2 errors in 21 qubits the example code can correct.[4]

[4] The error-correction rate of the code depends on the error model we consider. We allowed for two errors in four consecutive blocks. Then the distance of the code requires the three preceding ad the three following blocks to be error free. We only count the three preceding blocks, as the following error free blocks are preceding another strip of blocks that may contain errors. Therefore, the code can correct 2 errors every 7 blocks or 21 qubits.

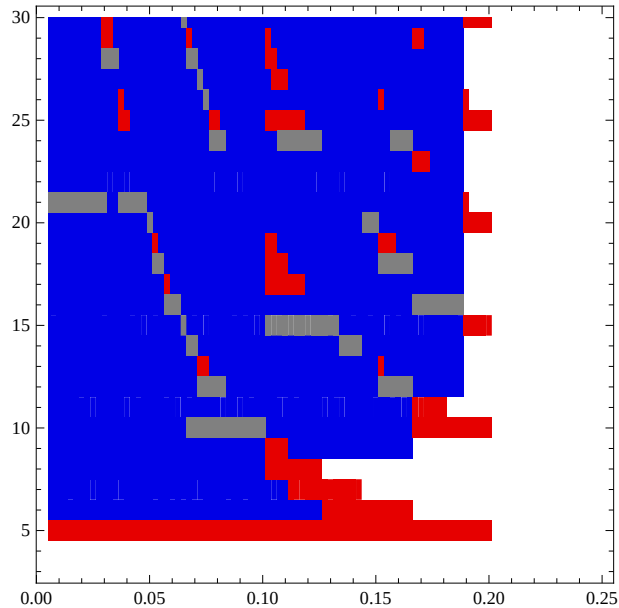


Figure 7.16: Comparison of the block and convolutional hamming bound in a symmetric scenario; in the blue region the convolutional bound is higher, while in the red region the block bound is higher. In the gray areas the bound are equal.

8 Outlook and open problems

In the following paragraphs, we give a short overview over open questions and possible extensions of the work presented in this thesis. As it is probably the case with every thesis, at the end it seems that there is more left to do than there was at the beginning. Hopefully, the work in this thesis will not go unnoticed and some ideas and open problems will be picked up. The structure of this Chapter follows the chapter structure of the thesis.

Causal operations

One of the most important questions is, whether the bound (4.27) on the memory depth is strict. A good approach to tackle this problem is to investigate whether one can construct a reversible memory channel for any given Bratteli diagram. Another interesting question regarding Bratteli diagrams of channels is the relation between the Bratteli diagram of a causal operation T and the diagram of its causal inverse T'.

The decomposition Theorem 4.1.1 could be generalized to non-reversible operations. Furthermore, it would be nice to identify more cases, in which the memory is isomorphic to a full matrix algebra to construct a resource efficient implementation. A good candidate is the case of reversible but not finite depth causal operations. So far, we have only proven the case of finite depth causal operations.

The results concerning the forgetfulness have only been proven for the case of translation-invariant operations. To generalize them to a non-translation-invariant setting, one has to generalize the theory of forgetful memory channels introduced in [51] to concatenations of different channels. It is also probable that most of the results can be extended to the non-reversible case.

Finally, the formalism could be used to investigate the uniqueness of the memory channel decomposition of causal operations. Starting from a forgetful memory channel, we can obtain a causal operation. This operation can be decomposed into a forgetful channel. The question is if this process can reduce the memory the channel uses. It seems that in the reversible scenario this is not the case. However, certain non-forgetful channels can also produce causal operations, because their memory never interacts with the input and output systems. Such a channel would benefit from the transformation to a causal process and back, because the memory states which never influence the outputs would be removed.

Clifford quantum cellular automata

In the case of Clifford quantum cellular automata it would be an obvious next step to study the extension to higher lattice dimensions. However, first examples showed no new and interesting dynamics but a mere superposition of the one-dimensional evolutions. It is of interest to investigate whether the different classes of Clifford quantum cellular automata have different capabilities as building blocks of quantum computational schemes. So far, we have only found differing potential in the entanglement generation that is governed by the degree of the trace of the CQCA and therefore only singles out periodic CQCAs. However, the blank spaces emerging in the spacetime image of fractal CQCAs could have an impact on the usability in quantum computational schemes.

Quantum convolutional codes

While we introduced an efficient description of convolutional codes by Clifford memory channels and were able to reproduce important results with simplified proofs, the construction of non-catastrophic encoders for all codes remains unsolved. Furthermore, the formalism is not yet capable of describing entanglement assisted convolutional codes [68, 67] and quantum turbo codes [69].

The performance analysis carried out here should only be the basis for a more detailed study. For a deeper analysis of the bound it would be necessary to not only consider the average error rate e, but also the probability for errors that can not be corrected. Comparable codes would not only have to use the same resource but also satisfy a common bound $\varepsilon > 0$ on the probability of uncorrectable errors. In this approach one could include the low-error bound into the general bound. For error-rates that imply a probability of uncorrectable errors below ε, the low-error bound would be used. Else, the general bound applies.

Furthermore, the resource requirements of some codes could potentially be reduced by a further decomposition of the encoding operation using the algorithm presented in Section 3.6.3. This can be possible if the operation has a causal structure in itself. Some codes have this property, e.g. the 5-qubit convolutional code introduced in [110]. In this code the stabilizer generators of a block are shifted copies of one basic generator. Therefore, the code has an intra-block causal structure that can be used to reduce the memory requirements. This code does not outperform block codes in terms of encoder resources, as a 5-qubit block code with the same properties exists. However, using the internal structure, it might have a representation that uses less resources than the corresponding block code.

Finally, it is important to find families of codes that saturate the bounds. Only then the bounds have a practical relevance—the upper bound as such could be higher than necessary because of bad proof techniques. However, the example codes show that convolutional codes can actually outperform block codes and give

hope for larger families of codes with these properties. A starting point would be to analyze the performance of existing families of convolutional codes such as [66, 117, 70, 67, 69] with respect to the Hamming bounds and block codes.

Theorems and proofs

1 A condition for complete positivity

Here we state the proof of a criterion for complete positivity for linear maps. First, let us cite a short lemma from Paulsen's textbook on completely bounded maps [25].

Lemma 1.1 (Lemma 3.13. in [25])**.** *Let $\mathfrak{A}$ be a C*-algebra. Then $A \in \mathcal{M}_n(\mathfrak{A}) = \mathfrak{A} \otimes \mathcal{M}_n(\mathbb{C})$ is positive if and only if the operators $A_{i,j}$ defined by $A = \sum_{i,j=1}^n A_{i,j} \otimes |i\rangle\langle j|$ are of the form $A_{i,j} = \sum_{l=1}^n a_{i,l}^* a_{j,l}$ for some $a_1, \ldots, a_n \in \mathfrak{A}$.*

Using this lemma we modify the condition for complete positivity of a map $T : \mathfrak{B} \mapsto \mathfrak{A}$.

$$
\begin{aligned}
T \text{ is CP} \quad &\Leftrightarrow \quad (T \otimes \mathrm{id}_n)[B] \geqslant 0 \quad \forall B \geqslant 0 \in \mathfrak{B}\, \forall n \\
&\Leftrightarrow \quad (T \otimes \mathrm{id}_n)[\sum_{i,j=1}^n \sum_{l=1}^n \left(b_{i,l}^* b_{j,l}] \otimes |i\rangle\langle j| \right) \geqslant 0\, \forall n \\
&\Leftrightarrow \quad \sum_{i,j=1}^n T[\sum_{l=1}^n b_{i,l}^* b_{j,l}] \otimes |i\rangle\langle j| \geqslant 0\, \forall n \\
&\Leftrightarrow \quad \exists c_i \in \mathfrak{A} \text{ s.t. } T[\sum_{l=1}^n b_{i,l}^* b_{j,l}] = \sum_{l=1}^n c_{i,l}^* c_{j,l}
\end{aligned}
$$

We can now proof the following Lemma:

Lemma 1.2. *Let $T : \mathfrak{B} \mapsto \mathfrak{A}$ be a linar map. T is completely positive if and only if*

$$\sum_{i,j} a_i^* T[b_i^* b_j] a_j \geqslant 0$$

for all $a_i \in \mathfrak{A}$ and all $b_i \in \mathfrak{B}$.

Proof. First assume that (1.2) holds. It follow that for any positive element $B \in \mathfrak{B}$ we have $B = b^* b$. Now choose $a \in \mathfrak{A}$ invertible. By assumption $a^* T[b^* b] a \geqslant 0$ and therefore $a^* T[b^* b] a = d^* d$ for some $d \in \mathfrak{A}$. By invertibility of a we get $T[b^* b] = (a^*)^{-1} d^* d\, a^{-1} = e^* e$ with $e = d\, a^{-1}$ and T is CP.

Now assume T is CP.

$$\begin{aligned}\sum_{i,j=1}^{n} a_i^* T[b_i^* b_j] a_j &= \sum_{i,j=1}^{n} a_i^* \sum_{l=1}^{n} c_{i,l}^* c_{j,l} a_j \\ &= \sum_{l=1}^{n} \left(\sum_{i=1}^{n} a_i^* c_{i,l}^* \right) \left(\sum_{j=1}^{n} a_j c_{i,l} \right) \\ &= \sum_{l=1}^{n} d_l^* d_l \geqslant 0\end{aligned}$$

with $d_l = \sum_{j=1}^{n} a_j c_{i,l}$. □

2 Results for CQCAs on qubits

We state here some results for CQCAs on chains of qubits that are needed in Section 5.2.2. All results are stated for CQCAs, because it is more convenient than the pure phase space formulation. Nevertheless, the results are also true for CSCAs and the proofs use the phase space formulation.

Lemma 2.1. *A finite tensor product of only one kind of Pauli matrices (and the identity) occurs at most once for every Pauli matrix σ_j, $j = 1,2,3$ in the history of any non-periodic CQCA T.*

Proof. We begin with the case $j = 1$. For two observables of the form $\bigotimes \sigma_i$, $i = 1,0$ to occur in the same time evolution of a CQCA T, the condition

$$\boldsymbol{t}^n = \begin{pmatrix} x & y \\ z & v \end{pmatrix} \begin{pmatrix} a \\ 0 \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix} \tag{1}$$

has to be true. It immediately follows that

$$\boldsymbol{t}^n = \begin{pmatrix} 1 & y \\ 0 & 1 \end{pmatrix},$$

which is a periodic automaton. The case $j = 3$ works analogous. For $j = 2$ we employ the fact, that we can build a CQCA T which fulfills Equation (1) via

$$\boldsymbol{t} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \boldsymbol{b} \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

from any CQCA B that fulfills $\boldsymbol{b}\binom{a}{a} = \binom{b}{b}$. Moreover, as the conjugation with a CQCA does not change the trace, all CQCAs B have to be periodic. □

Lemma 2.2. *Let T be a fractal CQCA on a spin chain and let $\bigotimes \sigma_i$ be a finite tensor product of Pauli matrices. For every $k \in \mathbb{N}$ there exist an $m \in \mathbb{N}$ such that $T^m \bigotimes \sigma_i$ contains at least k Pauli matrices.*

Proof. We assume that the number of elements is bounded by some k_{max}. The area over which these k_{max} elements are distributed is not bounded: if it were, the elements would either be restricted to a finite area for an infinite number of time steps, implying periodicity, or the area would move as a whole implying gliders for some power of T, which is not possible as we have shown in Lemma 5.1.12. So we see that any group of Pauli matrices will eventually be distributed over any area. But as we require the number of elements to be bounded, the distance between any two groups of Pauli matrices becomes larger than the neighborhood of the automaton. Then the starting argument applies to each of the new groups and forces them to break apart further until only isolated single-cell observables are left. But these will expand, thus the number of elements can not be bounded. □

3 General results for strictly forgetful channels

Here, we present two results we use to derive the results on error propagation.

Theorem 3.1. *Let $S : \mathfrak{B} \otimes \mathfrak{M} \to \mathfrak{M} \otimes \mathfrak{A}$ be a channel and $S_n : \mathfrak{B}^{\otimes n} \otimes \mathfrak{M} \to \mathfrak{M} \otimes \mathfrak{A}^{\otimes n}$ its n-fold concatenation. Then S is strictly forgetful if and only if every finitely localized observable on the output system $\mathfrak{B}^{\otimes n} \otimes \mathfrak{M}$ is mapped to an observable which is finitely localized on the input system $\mathfrak{M} \otimes \mathfrak{A}^{\otimes n}$, the localization area being independent of n for large n.*

Proof.

⇒ Let S be strictly forgetful. Let us assume there is an observable with finite localization area on the output system whose image is not independent of n for arbitrarily large n. This implies that after arbitrarily many steps the memory still transports information about the observable. Thus, the image of the observable on the memory system not be the identity after finitely many steps. This implies that the condition for forgetfulness of the channel can not be met and the channel is not forgetful which contradicts our assumption. Thus, all images of finitely localizes observables are finitely localized independent of n for large n.

⇐ Now let every image of every finitely localized observable be finitely localized independent of n for large n. Then for some finite n the image of all these observables on the memory has to be the identity. Thus the channel is strictly forgetful.

□

Theorem 3.2. *Let $S : \mathfrak{B} \otimes \mathfrak{M} \to \mathfrak{M} \otimes \mathfrak{A}$ be a reversible strictly forgetful channel. If follows that its inverse $S^{-1} : \mathfrak{M} \otimes \mathfrak{A} \to \mathfrak{B} \otimes \mathfrak{M}$ is also strictly forgetful.*

Proof. Strictly forgetful channels map finitely localized observables to finitely localized observables that are localized independently of the number of concatenations n for large n . For large n the image of a finitely localized observable commutes with every observable which is localized in the last m subsystems:

$$\Big[\underbrace{S_n\big[\mathbb{1}_{\mathfrak{B}}^{\otimes n-m} \otimes B_m \otimes M\big]}_{=\mathbb{1}_{\mathfrak{M}} \otimes \mathbb{1}_{\mathfrak{A}}^{\otimes n-l} \otimes A_l \in \mathfrak{M} \otimes \mathfrak{A}^{\otimes n}}, M' \otimes A_m \otimes \mathbb{1}_{\mathfrak{A}}^{\otimes n-m}\Big] = 0, \quad \text{for large } n. \tag{2}$$

B_m is localized on the last m subsystems. Equation (2) has to hold for all A_m, B_m, M and M'. (It might look like B_m depends on n. This is only the reason, because we count from the point of view of the input system. We could also fix B_0 at subsystem 0 and shift A_m to negative indices on the input system.) If we apply the inverse channel to the commutator, which has the same effect as applying it on the individual observables because the channel is reversible and thus a homomorphism, we get

$$\big[\mathbb{1}_{\mathfrak{B}}^{\otimes n-m} \otimes B_m \otimes M, S_n^{-1}\big[M' \otimes A_m \otimes \mathbb{1}_{\mathfrak{A}}^{\otimes n-m}\big]\big] = 0, \quad \text{for large } n. \tag{3}$$

This is the same condition we had for S_n. The only difference is the exchanged role of input and output system. Therefore the inverse channel is also strictly forgetful. □

Bibliography

[1] R. P. Feynman. Quantum mechanical computers. *Foundations of Physics*, 16(6):507–531, June 1986.

[2] R. Raussendorf and H. J. Briegel. A One-Way Quantum Computer. *Physical Review Letters*, 86(22):5188–5191, 2001.

[3] R. Raussendorf. Quantum cellular automaton for universal quantum computation. *Physical Review A*, 72(2):4, August 2005, arXiv:quant-ph/0412048.

[4] D. Shepherd, T. Franz, and R. F. Werner. Universally Programmable Quantum Cellular Automaton. *Physical Review Letters*, 97(2):2–5, July 2006, arXiv:quant-ph/0512058v3.

[5] A. Childs. Universal Computation by Quantum Walk. *Physical Review Letters*, 102(18):9, May 2009, arXiv:0806.1972.

[6] N. B. Lovett, S. Cooper, M. Everitt, M. Trevers, and V. Kendon. Universal quantum computation using the discrete-time quantum walk. *Physical Review A*, 81(4):9, April 2010, arXiv:0910.1024.

[7] J. Cirac and P. Zoller. Quantum Computations with Cold Trapped Ions. *Physical Review Letters*, 74(20):4091–4094, May 1995.

[8] C. Monroe, D. M. Meekhof, B. E. King, W. M. Itano, and D. J. Wineland. Demonstration of a Fundamental Quantum Logic Gate. *Physical Review Letters*, 75(25):4714–4717, December 1995.

[9] G. Brennen, C. Caves, P. Jessen, and I. Deutsch. Quantum Logic Gates in Optical Lattices. *Physical Review Letters*, 82(5):1060–1063, February 1999, arXiv:quant-ph/9806021.

[10] A. Negretti, P. Treutlein, and T. Calarco. Quantum computing implementations with neutral particles. *Quantum Information Processing*, 10(6):721–753, September 2011, arXiv:1105.0088.

[11] L. DiCarlo, J. M. Chow, J. M. Gambetta, L. S. Bishop, B. R. Johnson, D. I. Schuster, J. Majer, A. Blais, L. Frunzio, S. M. Girvin, and R. J. Schoelkopf. Demonstration of two-qubit algorithms with a superconducting quantum processor. *Nature*, 460(7252):240–4, July 2009.

[12] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 124–134. IEEE, 1994.

[13] L. Grover. Quantum Mechanics Helps in Searching for a Needle in a Haystack. *Physical Review Letters*, 79(2):325–328, July 1997, arXiv:quant-ph/9605043.

[14] A. Harrow, A. Hassidim, and S. Lloyd. Quantum Algorithm for Linear Systems of Equations. *Physical Review Letters*, 103(15):15, October 2009, arXiv:0811.3171.

[15] T. Baumgratz, D. Gross, M. Cramer, and M. B. Plenio. Scalable reconstruction of density matrices. July 2012, arXiv:1207.0358.

[16] P. Arrighi, V. Nesme, and R. F. Werner. Quantized Neighbourhoods. October 2009, arXiv:0910.4461.

[17] M. Takesaki. *Theory of operator algebras I*, volume 2. Springer, Berlin, 1st edition, 2001.

[18] K. R. Davidson. *C*-algebras by example*. American Mathematical Society, Providence, 1996.

[19] I. Gelfand and M. Naimark. On the imbedding of normed rings into the ring of operators in Hilbert space. *Matematicheskii Sbornik*, 12(54):197–213, 1943.

[20] O. Bratteli. Inductive limits of finite dimensional C*-algebras. *Transactions of the American Mathematical Society*, 171(September), 1972.

[21] O. Bratteli and D. W. Robinson. *Operator Algebras and Quantum Statistical Mechanics 1*, volume 1. Springer, 1997.

[22] O. Bratteli and D. W. Robinson. *Operator Algebras and Quantum Statistical Mechanics 2*, volume 2. Springer, 1997.

[23] M. Takesaki. *Theory of operator algebras II*, volume 2. Springer, Berlin, 2002.

[24] M. Takesaki. *Theory of operator algebras III*, volume 2. Springer, Berlin, 2002.

[25] V. Paulsen. *Completely bounded maps and operator algebras*. Cambridge University Press, Cambride, 1st edition, 2002.

[26] E. M. Žmud. Symplectic Geometries over Finite Abelian Groups. *Mathematics of the USSR Sbornik*, 15(1):7–29, 1971.

[27] E. M. Žmud. Symplectic Geometric and Projective Representations of Finite Abelian Groups. *Mathematics of the USSR Sbornik*, 16(1):1–16, 1972.

[28] D.-M. Schlingemann. *Quantum Information Processing with Graph States*. Habilitation, Universität Braunschweig, 2005.

[29] R. Berndt. *Einführung in die symplektische Geometrie*. Vieweg, 1998.

[30] I. Vaisman. *Symplectic geometry and secondary characteristic classes*. Birkhäuser, 1987.

[31] S. Lang. *Algebra*. Addison-Wesley publishing company, 3rd edition, 1993.

[32] N. Jacobson. *Lectures in abstract algebra. Vol. I. Basic concepts*, volume 1. D Van Nostrand Company Inc, 1952.

[33] D.-M. Schlingemann, H. Vogts, and R. F. Werner. On the structure of Clifford quantum cellular automata. *Journal of Mathematical Physics*, 49(11):112104, 2008, arXiv:0801.4604v1.

[34] J. Gütschow. *Cliffordkanäle als Zellularautomaten und Faltungscodierer*. Diploma thesis, Universität Braunschweig, 2008.

[35] J. Gütschow. Clifford Channels with Memory. 2010.

[36] J. Gütschow. Representation of Convolutional Stabilizer Codes as Clifford Memory Channels. Internal report for the CORNER project, 2010.

[37] B. Schumacher and R. F. Werner. Reversible quantum cellular automata. May 2004, arXiv:quant-ph/0405174.

[38] D. Gross, V. Nesme, H. Vogts, and R. F. Werner. Index Theory of One Dimensional Quantum Walks and Cellular Automata. *Communications in Mathematical Physics*, 310(2):419–454, January 2012, arXiv:0910.3675.

[39] J. Gütschow, S. Uphoff, R. F. Werner, and Z. Zimborás. Time asymptotics and entanglement generation of Clifford quantum cellular automata. *Journal of Mathematical Physics*, 51(1):015203, June 2010, arXiv:0906.3195.

[40] H. Araki and E. H. Lieb. Entropy inequalities. *Communications in Mathematical Physics*, 18(2):160–170, June 1970.

[41] H. Vogts. *Discrete Time Quantum Lattice Systems*. PhD thesis, Universität Braunschweig, 2009.

[42] R. Alicki and M. Fannes. *Quantum Dynamical Systems*. Oxford University Press, 2001.

[43] T. Eggeling, D.-M. Schlingemann, and R. F. Werner. Semicausal operations are semilocalizable. *Europhysics Letters (EPL)*, 57(6):782–788, March 2002, arXiv:quant-ph/0104027.

[44] A. Bisio, G. M. D'Ariano, P. Perinotti, and G. Chiribella. Minimal computational-space implementation of multiround quantum protocols. *Physical Review A*, 83(2):4, February 2011, arXiv:1006.1780.

[45] A. Bisio, G. M. D'Ariano, P. Perinotti, and M. Sedlak. Memory cost of quantum protocols. December 2011, arXiv:1112.3853.

[46] M. Houshmand, S. Hosseini-Khayat, and M. M. Wilde. Minimal Memory Requirements for Pearl-Necklace Encoders of Quantum Convolutional Codes. *IEEE Transactions on Computers*, 1:30, April 2010, arXiv:1004.5179.

[47] M. Houshmand and S. Hosseini-Khayat. Minimal-memory realization of pearl-necklace encoders of general quantum convolutional codes. *Physical Review A*, 83(2):16, February 2011, arXiv:1009.2242.

[48] M. Houshmand, S. Hosseini-Khayat, and M. M. Wilde. Minimal-memory, non-catastrophic, polynomial-depth quantum convolutional encoders. May 2011, arXiv:1105.0649.

[49] J. Gütschow. Performance of Quantum Convolutional Coding with Memory constraints. Poster presented at the International Iranian Conference on Quantum Information, 2010.

[50] G. Bowen and S. Mancini. Quantum channels with a finite memory. *Physical Review A*, 69(1):7, January 2004, arXiv:quant-ph/0305010.

[51] D. Kretschmann and R. F. Werner. Quantum channels with memory. *Physical Review A*, 72(6):21, December 2005, arXiv:quant-ph/0502106.

[52] D.-M. Schlingemann. Finite Weyl Systems. In *Encyclopedia of Mathematical Physics*. Elsevier, 2005.

[53] A. S. Holevo. Remarks on the classical capacity of quantum channel. December 2002, arXiv:quant-ph/0212025.

[54] M. Fukuda and A. S. Holevo. On Weyl-covariant channels. October 2005, arXiv:quant-ph/0510148.

[55] T. S. Cubitt, M. B. Ruskai, and G. Smith. The structure of degradable quantum channels. *Journal of Mathematical Physics*, 49(10):102104, February 2008, arXiv:0802.1360.

[56] M. Nathanson and M. B. Ruskai. Pauli diagonal channels constant on axes. *Journal of Physics A: Mathematical and Theoretical*, 40(28):8171–8204, July 2007, arXiv:quant-ph/0611106.

[57] T. C. Bschorr, D. G. Fischer, and M. Freyberger. Channel estimation with noisy entanglement. *Physics Letters A*, 292(1-2):15–22, December 2001, arXiv:quant-ph/0108036.

[58] J. Dehaene, M. V. den Nest, B. De Moor, and F. Verstraete. Local permutations of products of Bell states and entanglement distillation. *Physical Review A*, 67(2):6, July 2002, arXiv:quant-ph/0207154.

[59] J. Dehaene and B. De Moor. Clifford group, stabilizer states, and linear and quadratic operations over GF(2). *Physical Review A*, 68(4):9, October 2003, arXiv:quant-ph/0304125.

[60] S. Aaronson and D. Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5):1–14, November 2004, arXiv:quant-ph/0406196.

[61] E. Hostens, J. Dehaene, and B. De Moor. Stabilizer states and Clifford operations for systems of arbitrary dimensions and modular arithmetic. *Physical Review A*, 71(4):10, April 2005, arXiv:quant-ph/0408190.

[62] E. Knill and R. Laflamme. Theory of quantum error-correcting codes. *Physical Review A*, 55(2):900–911, February 1997, arXiv:quant-ph/9604034.

[63] C. Bennett, D. DiVincenzo, J. Smolin, and W. K. Wootters. Mixed-state entanglement and quantum error correction. *Physical Review A*, 54(5):3824–3851, November 1996, arXiv:quant-ph/9604024.

[64] D. Gottesman. *Stabilizer Codes and Quantum Error Correction*. PhD thesis, May 1997, arXiv:quant-ph/9705052.

[65] H. Chau. Quantum convolutional error-correcting codes. *Physical Review A*, 58(2):905–909, August 1998, arXiv:quant-ph/9712029.

[66] M. Grassl and M. Rötteler. Constructions of Quantum Convolutional Codes. In *2007 IEEE International Symposium on Information Theory*, number D, pages 816–820. IEEE, June 2007, arXiv:quant-ph/0703182.

[67] M. M. Wilde and T. A. Brun. Entanglement-assisted quantum convolutional coding. *Physical Review A*, 81(4):1–22, April 2010, arXiv:0712.2223.

[68] M. M. Wilde and T. A. Brun. Unified quantum convolutional coding. In *2008 IEEE International Symposium on Information Theory*, number 4, pages 359–363. IEEE, July 2008, arXiv:0801.0821.

[69] D. Poulin, J.-P. Tillich, and H. Ollivier. Quantum serial turbo codes. *IEEE Transactions on Information Theory*, 55(6):2776–2798, December 2009, arXiv:0712.2888.

[70] G. D. Forney, M. Grassl, and S. Guha. Convolutional and Tail-Biting Quantum Error-Correcting Codes. *IEEE Transactions on Information Theory*, 53(3):865–880, March 2007, arXiv:quant-ph/0511016.

[71] H. Ollivier and J.-P. Tillich. Quantum convolutional codes: fundamentals. January 2004, arXiv:quant-ph/0401134.

[72] A. Ahlbrecht, F. Richter, and R. F. Werner. How long can it take for a quantum channel to forget everything? pages 1–18, May 2012, arXiv:1205.0693.

[73] R. Raussendorf. Quantum computation via translation-invariant operations on a chain of qubits. *Physical Review A*, 72(5):1–6, November 2005, arXiv:quant-ph/0505122v2.

[74] J. Fitzsimons and J. Twamley. Globally controlled quantum wires for perfect qubit transport, mirroring and computing. *Physical Review Letters*, 97(9):5, January 2006, arXiv:quant-ph/0601120.

[75] V. Eisler and I. Peschel. Entanglement in a periodic quench. *Annalen der Physik*, 17(6):410–423, 2008.

[76] J. Fitzsimons, L. Xiao, S. Benjamin, and J. Jones. Quantum Information Processing with Delocalized Qubits under Global Control. *Physical Review Letters*, 99(3):4, July 2007, arXiv:quant-ph/0606188.

[77] A. Bultheel and M. Van Barel. *Linear algebra, rational approximation, and orthogonal polynomials*. North Holland, 1997.

[78] E. H. Lieb, T. Schultz, and D. Mattis. Two soluble models of an antiferromagnetic chain. *Annals of Physics*, 16(3):407–466, 1961.

[79] E. Barouch, B. McCoy, and M. Dresden. Statistical Mechanics of the XY Model. I. *Physical Review A*, 2(3):1075–1092, September 1970.

[80] E. Barouch and B. McCoy. Statistical Mechanics of the XY Model. II. Spin-Correlation Functions. *Physical Review A*, 3(2):786–804, February 1971.

[81] H. Araki. On the XY-model on two-sided infinite chain. *Publications of the Research Institute for Mathematical Sciences Kyoto University*, 20(2):277–296, 1984.

[82] H. Araki. On Quasifree States of CAR and Bogoliubov Automorphisms. *Publications of the Research Institute for Mathematical Sciences Kyoto University*, 6(3):385–442, 1971.

[83] W. H. Aschbacher and J.-M. Barbaroux. Exponential spatial decay of spin-spin correlations in translation invariant quasifree states. *Journal of Mathematical Physics*, 48(11):113302, 2007.

[84] W. Rudin. *Functional Analysis*. McGraw-Hill Book Company, New York, 1973.

[85] M. Reed and B. Simon. *Methods of Modern Mathematical Physics, Vol. 2: Fourier Analysis, Self-Adjointness*, volume 2. Academic Press, New York, 1975.

[86] M. Cramer, C. M. Dawson, J. Eisert, and T. J. Osborne. Exact relaxation in a class of non-equlibrium quantum lattice systems. *Physical Review Letters*, 100(3):30602, January 2008.

[87] S. Uphoff. *Stationary States of Clifford Quantum Cellular Automata*. Diploma thesis, Universität Braunschweig, 2008.

[88] P. Calabrese and J. Cardy. Evolution of entanglement entropy in one-dimensional systems. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(04):P04010, April 2005, arXiv:cond-mat/0503393.

[89] G. Brennen and J. Williams. Entanglement dynamics in one-dimensional quantum cellular automata. *Physical Review A*, 68(4):12, October 2003, arXiv:quant-ph/0306056.

[90] D. Fattal, T. S. Cubitt, Y. Yamamoto, S. Bravyi, and I. Chuang. Entanglement in the stabilizer formalism. June 2004, arXiv:quant-ph/0406168.

[91] M. Fekete. Über die Verteilung der Wurzeln bei gewissen algebraischen Gleichungen mit ganzzahligen Koeffizienten. *Mathematische Zeitschrift*, 17:228–249, 1923.

[92] M. Fannes. The entropy density of quasi free states. *Communications in Mathematical Physics*, 31(4):279–290, December 1973.

[93] G. Vidal, J. I. Latorre, E. Rico, and A. Kitaev. Entanglement in Quantum Critical Phenomena. *Physical Review Letters*, 90(22):227902, June 2003.

[94] M. A. Nielsen. Cluster-state quantum computation. *Reports on Mathematical Physics*, 57(1):147–161, February 2006, arXiv:quant-ph/0504097.

[95] M. M. Wilde, M. Houshmand, and S. Hosseini-Khayat. Examples of minimal-memory, non-catastrophic quantum convolutional encoders. In *Proceedings of the International Symposium on Information Theory 2011*, page 5, November 2011, arXiv:1011.5535.

[96] S. Willson. Computing fractal dimensions for additive cellular automata. *Physica D: Nonlinear Phenomena*, 24(1-3):190–206, January 1987.

[97] S. Takahashi. Cellular automata and multifractals: dimension spectra of linear cellular automata. *Physica D: Nonlinear Phenomena*, 45(1-3):36–48, 1990.

[98] S. Takahashi. Self-similarity of linear cellular automata. *Journal of Computer and System Sciences*, 44(1):114–140, February 1992.

[99] F. von Haeseler, H.-O. Peitgen, and G. Skordev. Cellular automata, matrix substitutions and fractals. *Annals of Mathematics and Artificial Intelligence*, 8(3-4):345–362, September 1993.

[100] F. von Haeseler, H.-O. Peitgen, and G. Skordev. Self-similar structure of rescaled evolution sets of cellular automata I. *International Journal of Bifurcation and Chaos*, 11(4):913–926, 2001.

[101] F. von Haeseler, H.-O. Peitgen, and G. Skordev. Self-similar structure of rescaled evolution sets of cellular automata II. *International Journal of Bifurcation and Chaos*, 11(4):927–942, 2001.

[102] J.-P. Allouche. Linear cellular automata, finite automata and Pascal's triangle. *Discrete Applied Mathematics*, 66(1):1–22, April 1996.

[103] J.-P. Allouche, F. von Haeseler, H.-O. Peitgen, A. Petersen, and G. Skordev. Automaticity of double sequences generated by one-dimensional linear cellular automata. *Theoretical Computer Science*, 188(1-2):195–209, November 1997.

[104] C. Moore. Quasi-Linear Cellular Automata. *Physica D: Nonlinear Phenomena*, 103:100–132, 1997.

[105] C. Moore. Non-Abelian cellular automata. *Physica D: Nonlinear Phenomena*, 111:27–41, 1998.

[106] A. J. Macfarlane. Linear reversible second-order cellular automata and their first-order matrix equivalents. *Journal of Physics A: Mathematical and General*, 37:10791–10814, 2004.

[107] J. Gütschow, V. Nesme, and R. F. Werner. Additional electronic material on the fractal structure of cellular automata on abelian groups (http://www.johannes-guetschow.de/fractal-ca), 2010.

[108] B. B. Mandelbrot, Y. Gefen, A. Aharony, and J. Peyrière. Fractals, their transfer matrices and their eigen-dimensional sequences. *Journal of Physics A: Mathematical and General*, 18:335–354, 1985.

[109] J. Gütschow, V. Nesme, and R. F. Werner. The fractal structure of cellular automata on Abelian groups. *Automata 2010*, page 29, November 2010, arXiv:1011.0313.

[110] H. Ollivier and J.-P. Tillich. Description of a Quantum Convolutional Code. *Physical Review Letters*, 91(17):4, October 2003, arXiv:quant-ph/0304189.

[111] M. Grassl and M. Rötteler. Non-catastrophic Encoders and Encoder Inverses for Quantum Convolutional Codes. In *2006 IEEE International Symposium on Information Theory*, pages 1109–1113. IEEE, July 2006, arXiv:quant-ph/0602129.

[112] A. Ekert and C. Macchiavello. Quantum Error Correction for Communication. *Physical Review Letters*, 77(12):2585–2588, September 1996, arXiv:quant-ph/9602022.

[113] G. Chiribella, M. Dall'Arno, G. M. D'Ariano, C. Macchiavello, and P. Perinotti. Quantum error correction with degenerate codes for correlated noise. July 2010, arXiv:1007.3655.

[114] R. Laflamme, C. Miquel, J. P. Paz, and W. H. Zurek. Perfect Quantum Error Correcting Code. *Physical Review Letters*, 77(1):198–201, July 1996, arXiv:quant-ph/9602019.

[115] D. Gottesman. Pasting Quantum Codes. July 1996, arXiv:quant-ph/9607027.

[116] A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane. Quantum error correction via codes over GF (4). *IEEE Transactions on Information Theory*, 44(4):1369–1387, August 1998, arXiv:quant-ph/9608006.

[117] G. D. Forney and S. Guha. Simple rate-1/3 convolutional and tail-biting quantum error-correcting codes. In *International Symposium on Information Theory, 2005.*, pages 1028–1032. IEEE, January 2005, arXiv:quant-ph/0501099.

[118] J. Gütschow. Entanglement generation of Clifford quantum cellular automata. *Applied Physics B*, 98(4):623–633, December 2009, arXiv:1001.1062.

[119] J. Gütschow, T. Rybár, and R. F. Werner. Memory requirements for general reversible quantum stream processors. 2012.

Curriculum vitae

Full name	Johannes Gütschow
Date of birth	December 23rd, 1981 in Hamburg

Positions and education

03/2012 – now: Research assistant, *Potsdam Institute for Climate Impact Research*

05/2008 – 02/2012: PhD Student, *Institute for Theoretical Physics*, Leibniz University Hannover and *Institute for Mathematical Physics* University of Braunschweig
Project: Quantum information processing with Clifford quantum cellular automata
Supervised by Prof. Reinhard F. Werner

04/2008: Diploma in Physics, *Institute for Mathematical Physics*, University of Braunschweig
Project: Cliffordkanäle als Zellularautomaten und Faltungscodierer
Supervised by Prof. Reinhard F. Werner

09/2002 – 04/2008: Studies in physics, University of Braunschweig

2001: Abitur, *Gymnasium Osdorf*, Hamburg
Completed high school education.

List of publications

[39] Johannes Gütschow, Sonja Uphoff, Reinhard F. Werner, and Zoltán Zimborás
Time asymptotics and entanglement generation of Clifford quantum cellular automata
Journal of Mathematical Physics, 51(1):015203, January 2010
arXiv:0906.3195
Selected as JMP research highlight in Feb. 2010. Selected for the Virtual Journal of Quantum Information Volume 10, Issue 2 (Feb 2010)

[118] Johannes Gütschow
Entanglement generation of Clifford quantum cellular automata
Applied Physics B, 98(4):623-633, December 2009
arXiv:1001.1062

[109] Johannes Gütschow, Vincent Nesme, and Reinhard F. Werner
The fractal structure of cellular automata on Abelian groups
Proceedings of Automata 2010, 51-70, June 2010
arXiv:1011.0313
Extended version:
Self-similarity of cellular automata on Abelian groups
Journal of Cellular Automata, 7(2):83-113, March 2012

[119] Johannes Gütschow, Tomáš Š. Rybár, and Reinhard F. Werner
Memory requirements for general reversible quantum stream processors
in preparation

Acknowledgements

This thesis would not have been finished or even begun without the support of a lot of people whom I want to thank here for all their support.

First of all I want to thank Reinhard Werner for advising my on this thesis and always having good ideas and helpful comments (usually more than I could implement). Without his lecture on quantum mechanics, which I attended during my undergrad studies, I would probably still think that nobody understands quantum mechanics and its whole purpose is to calculate Clebsch Gordon coefficients. This lecture showed me that one can understand Quantum mechanics and motivated me to give it a try. This lead me to joining Reinhard Werner's group for my diploma and PhD. I also want to thank Tobias Osborne for reviewing this thesis.

I want to thank the Rosa Luxemburg Foundation for support through a PhD-scholarship and several interesting seminars and conferences broadening my horizon beyond math and physics.

During my work on this thesis I had interesting and helpful discussions in our group, with collaborators and on conferences. Most important for this thesis were the discussions with my collaborators on the papers which form a basis of some parts of this thesis. Therefore, I want to specially thank Sonja Uphoff, Vincent Nesme and Zoltán Zimborás. Furthermore, I want to thank Tomáš Rybár: our joint work on causal operations is just not published in a journal yet, because we both had to finish our PhDs.

In our group in Braunschweig and later in Hannover I discussed many question on memory channels and forgetfulness with David Groß, Florian Richter and Thomas Salfeld. Many thanks go to them for a lot of good ideas and helpful comments.

A lot of people had an impact on my research through discussions we had during lunch, at conferences or with a beer in the pub. Among them are Michael Bremner, Friederike Trimborn, Bernhard Neukirchen, Dirk Schlingemann, Holger Vogts, Laleh Memarzadeh, Cosmo Lupo, Mark Wilde, Monireh Houshmand, Andre Ahlbrecht, Tobias Osbourne, Albert Werner, Volkher Scholz, Torsten Franz, and all other members of the quantum information group in Hannover.

Thanks also go to the organizers of all the conferences I attended during my PhD. The conferences were always motivating and I usually returned with new insights and ideas. I want to especially thank the organizers of the Benasque Quantum Information Summerschools 2009 and 2011 and the organizers of the 2010 International Iranian Conference on Quantum Information.

For proof reading this thesis I want to thank Sarah Harrison, Sonja Uphoff, Peter Finch, Michael Bremner, and Claudine Chen. Thank you very much for pointing out all the inconsistencies and incomprehensible sentences I produced and for correcting my English. Further thanks go to Michal Bremner for help on structuring the thesis and to Sönke Schmidt for sharing his knowledge about all the formal requirements needed to hand in the thesis.

I also want to thank Katja Frieler at the Potsdam Institute for Climate Impact Research, where I work now, for giving me the possibility to take some time off to finish this thesis.

Further thanks go to Friederike Trimborn, Berhard Neukirchen and Christopher Cedzich for sharing an office with me and always having time for questions, scientific discussions, a non-scientific chat or a discussion about the latest political developments in the world.

With Michael Bremner and Friederike Trimborn I shared many walks to 24grad to get our daily dosage of really good coffee. Our discussions about science, politics and the rest of the world were a highlight of the day. 24grad deserves thanks for the great coffee that not only kept my brain working but also stimulated my taste buds. I want to thank Michael Bremner, Friederike Trimborn, Bernhard Neukirchen, Peter Finch, Tobias Osborne, Cédric Bény and Robert Matjeschk for many shared lunch breaks and discussions from science to movies and back.

I want to thank Wiebke Möller for organizing all the administrative stuff and making sure that I always had a place to stay during conferences.

Finally, I want to thank all my friends, housemates, family and most importantly Sonja Uphoff for supporting me during the time of my PhD and being tolerant towards my bad moods and constant excuses related to finishing this thesis.