Department of Computer Science
Faculty of Mathematics, Physics and Informatics
Comenius University, Bratislava

On Closure Properties of Quantum Finite Automata

(Diploma thesis)

Martin Macko

Thesis advisor:
prof. Dr. Jozef Gruska, DrSc.

Bratislava, 2006

By this I declare that I wrote this thesis by oneself, only with the help of the referenced literature, under the careful supervision of my thesis advisor.

. . . . . .

Motto:
Quidquid latine dictum sit, altum viditur.

Traditional

Acknowledgments

I would like to thank Peter Košinár and Tomáš Záthurecký for their helpful comments and discussions and prof. Branislav Rovan for lending me copies of [Fre81] and [KF91]. Very special thanks to my advisor prof. Jozef Gruska for his supervision and invaluable help with this work.

Abstract

We prove several new closure properties of the classes of languages recognized by 1.5-way and 2-way quantum finite state automata (1.5QFA and 2QFA) and 2-way finite automata with quantum and classical states (2QCFA). We show that none of these classes of languages is closed under homomorphism, the classes of languages recognized by 1.5QFA and by simple 2QFA are closed under inverse non-erasing homomorphism and the class of languages recognized by simple 1.5QFA is closed under general inverse homomorphism. We also show that the homomorphic closures of the classes of languages recognized by 1.5QFA, 2QFA and 2QCFA are equal to the class of recursively enumerable languages and the homomorphic closure of the class of languages recognized by 1-way quantum finite state automata (1QFA) is equal to the class of regular languages.

Keywords: Quantum computation, Quantum finite automata, 1.5QFA, 2QFA, 2QCFA.

Contents

Chapter 1

Introduction

The phenomena of quantum mechanics give us a new kind of computational power over the classical mechanics. Very remarkable successes in the theory of quantum computing are Peter Shor's algorithms for factoring and discrete logarithm in polynomial time for quantum computers [Sho94, Sho97] and Grover's quantum algorithm for searching an unordered list of size n with only $O(\sqrt{n})$ accesses to the list [Gro96, Gro98].

However, these algorithms use the power of universal quantum machines, such as quantum Turing machines [Deu85, BV93], quantum circuits [Deu89, Yao93] or quantum cellular automata [Mar86, Wat95]. Recent experimental quantum computers are much less powerful, as the most successful realizations of quantum computers have so far only up to 7 quantum bits [VSB$^+$01].

In this thesis we consider quantum finite automata – a simpler model of quantum computation which may be easier to implement than universal quantum machines. Different models of quantum finite automata have been studied. The simplest ones are one-way quantum finite state automata (1QFA) introduced by Kondacs and Watrous [KW97]. This very simple model of quantum automata is not very powerful and the class of languages recognized by it is a proper subset of the class of regular languages. In the same paper, Kondacs and Watrous introduced also a more powerful model – two-way quantum finite state automata (2QFA). These automata can simulate any deterministic finite automaton and even recognize some non-regular languages. Therefore, 2QFA are strictly more powerful than 1QFA. More recently, Amano and Iwama [AI99] introduced a variation of 2QFA – 1.5-way quantum finite state automata (1.5QFA) – which cannot move their head to the left. A limitation of 1.5QFA and 2QFA is that they allow superpositions of different head positions on the tape and hence the number of their quantum states grows with the growing length of the input word. This probably could make the models more difficult to implement. Ambainis and Watrous [AW02] introduced two-way finite automata with quantum and classical states – a model where the size of the quantum part does not depend on the input word length and still is strictly more powerful than finite state automata. 2QCFA can be seen as an intermediate model between 1QFA and 2QFA.

Many properties of quantum finite automata have been studied so far. Every 1QFA can be simulated by a classical finite automaton, and there is no 1QFA recognizing the regular language $\{a, b\}^*a$ [KW97]. The class of languages recognized by 1QFA is closed under complement and inverse homomorphism [BP02] and is not closed under homomorphism [BP02] nor any binary Boolean operation with both arguments significant [AĶV01]. Although 1QFA can recognize the language a^*b^* with probability approximately 0.68 [AF98], which cannot be recognized by any reversible finite automaton, every language recognized by 1QFA with probability greater than roughly 0.78 is recognized by some reversible finite automaton [AĶ03]. Sometimes, 1QFA can be much more space-efficient than deterministic and even probabilistic finite automata. There is an 1QFA checking divisibility by a prime number p with only $O(\log p)$ states, what is equivalent to $O(\log \log p)$ quantum bits of memory [AF98].

On the contrary to 1QFA, 2QFA and 2QCFA are strictly more powerful than classical finite automata. Beyond the regular languages they can recognize the language $\{a^k b^k \mid k \geq 0\}$ [KW97,

AW02]. 2QCFA can also recognize the language of all palindromes $\{w \in \{a,b\}^* \mid w = w^R\}$ [AW02], which cannot be recognized even by any two-way probabilistic finite automaton [DS90, KF91]. 1.5QFA can also recognize some non-regular languages, however, it is not known whether they recognize all regular languages or not. The classes of languages recognized by these three models are trivially closed under complement and the class of languages recognized by 2QCFA is also closed under inverse homomorphism [Kos̆05]. All these three classes are rather complicated, as even the emptiness problem is undecidable for all these three models [AI99, Fre81].

In this thesis we prove several new closure properties of different models of quantum finite automata. We show that the classes of languages recognized by 1.5QFA and simple 2QFA are closed under inverse non-erasing homomorphism and the class of languages recognized by simple 1.5QFA is closed under general inverse homomorphism. Further we show that none of the classes of languages recognized by 1.5QFA, 2QFA nor 2QCFA is closed under homomorphism.

As neither 1QFA, 1.5QFA, 2QFA nor 2QCFA recognize a class of languages closed under homomorphism, a natural question is, what are the smallest classes closed under homomorphism containing these classes as subsets. We show that the homomorphic closures of the classes of languages recognized by 1.5QFA, 2QFA and 2QCFA, respectively, are equal to the class of recursively enumerated languages, and the homomorphic closure of the class of languages recognized by 1QFA is equal to the class of regular languages. This shows a significant gap between the homomorphic closures of 1.5QFA, 2QFA and 2QCFA on one hand, and the homomorphic closure of 1QFA on the other hand.

The remainder of this thesis has the following organization. In chapter 2, we shortly present preliminaries of quantum computating theory. In chapter 3, we provide formal definitions of the four models of quantum finite automata. In chapter 4, we discuss closure properties of 1.5QFA, simple 1.5QFA and simple 2QFA under inverse homomorphism. chapter 5 deals with the homomorphic closure of the class of languages recognized by 1.5QFA and chapter 6 shortly generalizes this result for the case of 2QFA and 2QCFA and discusses the homomorphic closure of 1QFA. Finally, in chapter 7, we conclude with mention of some open problems.

Chapter 2

Preliminaries

In this chapter we shortly present preliminaries of the quantum computing theory we use in the rest of this thesis. To fully understand details of the quantum computing theory, we encourage the reader to read the books [Gru99, NC00], which provide a detailed overview of a wide range of quantum computing topics.

2.1 Hilbert spaces

Closed quantum systems are well modeled by Hilbert spaces. We use them as a basic mathematical framework to formally describe the principles of quantum mechanics. Hilbert spaces are complex vector spaces with inner products $\langle\psi|\phi\rangle$.

Definition 2.1 *An* inner-product space $\mathcal{H}$ *over a field* F *is a vector space over the field* F *with an inner product* $\langle\cdot|\cdot\rangle\colon \mathcal{H}\times\mathcal{H}\to\mathbb{C}$ *satisfying*

1. $\langle\psi|\phi\rangle = \langle\phi|\psi\rangle^*$, *(conjugate symmetry)*
2. $\langle\psi|\psi\rangle \geq 0$ *and* $\langle\psi|\psi\rangle = 0$ *if and only if* $\psi = \mathbf{0}$, *and* *(positive definiteness)*
3. $\langle\psi|c_1\phi_1 + c_2\phi_2\rangle = c_1\langle\psi|\phi_1\rangle + c_2\langle\psi|\phi_2\rangle$, *(right linearity)*

for each $\psi, \phi, \phi_1, \phi_2 \in \mathcal{H}$ *and* $c_1, c_2 \in F$.

The inner product $\langle\psi|\phi\rangle$ of the inner-product space $\mathcal{H}$ introduces on the space $\mathcal{H}$ for every $\phi \in \mathcal{H}$ the *norm*

$$\|\phi\|_{\mathcal{H}} = \sqrt{\langle\phi|\phi\rangle},$$

and for every $\phi, \psi \in \mathcal{H}$ the *metric*

$$\mathrm{dist}_{\mathcal{H}}(\phi,\psi) = \|\phi - \psi\|.$$

If there is no ambiguity we usually write just $\|\phi\|$ and $\mathrm{dist}(\phi,\psi)$ instead of $\|\phi\|_{\mathcal{H}}$ and $\mathrm{dist}_{\mathcal{H}}(\phi,\psi)$, respectively. The notion of metric allows us to introduce a metric topology and concepts such as continuity on $\mathcal{H}$. Vectors of the inner-product space $\mathcal{H}$ with unit norm are called *(pure) states* of the space $\mathcal{H}$.

Definition 2.2 *An inner-product space* $\mathcal{H}$ *is* complete *if for any sequence* $\{\phi_j\}_{j=1}^{\infty}$ *with all* $\phi_j \in \mathcal{H}$ *such that* $\lim_{j,k\to\infty}\|\phi_j - \phi_k\| = 0$ *there is a unique element* $\phi \in \mathcal{H}$, *for which we have*

$$\lim_{j\to\infty}\|\phi - \phi_j\| = 0.$$

Definition 2.3 Hilbert space *is a complete inner-product space over the field* $\mathbb{C}$ *of complex numbers.*

There are many different kinds of Hilbert spaces in the quantum theory. To model the quantum systems used in this thesis, we use the so-called $\ell_2(D)$ Hilbert spaces. We define them as follows.

Definition 2.4 *Let D be a countable set. We say, that $\ell_2(D)$ is the vector space of all complex valued functions on D bounded by the so-called ℓ_2-norm, i.e.*

$$\ell_2(D) = \left\{ x\colon D \to \mathbb{C} \;\middle|\; \sqrt{\sum_{j\in D} x(j)\big(x(j)\big)^*} < \infty \right\}.$$

For any countable set D, the vector space $\ell_2(D)$ is a Hilbert space with respect to the inner product $\langle\cdot|\cdot\rangle\colon \ell_2(D) \times \ell_2(D) \to \mathbb{C}$, defined by

$$\langle x_1|x_2\rangle = \sum_{j\in D} \big(x_1(j)\big)^* x_2(j).$$

The concept of orthogonality is very important in the inner-product space theory. The main role of orthogonality in quantum computing is that only mutually orthogonal quantum states are well distinguishable by quantum measurements.

Definition 2.5 *Vectors ϕ and ψ of an inner-product space $\mathcal{H}$ are* orthogonal*, denoted by $\phi \perp \psi$, if $\langle\phi|\psi\rangle = 0$. A subset S of $\mathcal{H}$ is* orthogonal *if every two elements of S are orthogonal. The set S is* orthonormal *if it is orthogonal and all its elements have norm* 1*.*

Definition 2.6 *An orthonormal subset Θ of an inner-product space is its* orthonormal basis *if none of its proper supersets is orthonormal.*

We can show that all bases of an inner-product space $\mathcal{H}$ have the same cardinality. This cardinality is called the *dimension* of the space $\mathcal{H}$. If the cardinality is finite, we say that the inner-product space is *finite-dimensional*. Otherwise, we say that the space is *infinite-dimensional*.

Definition 2.7 *A* subspace *$\mathcal{E}$ of an inner-product space $\mathcal{H}$ is a subset of $\mathcal{H}$ closed under addition and scalar multiplication.*

By a simple verification of the conditions from definition 2.1 we can show that a subspace of an inner-product space is itself an inner-product space.

Lemma 2.8 *For every complete subspace $\mathcal{E}$ of a complete inner-product space $\mathcal{H}$ there is a unique subspace $\mathcal{E}^\perp$ such that $\langle\phi_1|\phi_2\rangle = 0$ for each $\phi_1 \in \mathcal{E}$ and $\phi_2 \in \mathcal{E}^\perp$, and each $\psi \in \mathcal{H}$ can be uniquely written is the form $\psi = \phi_1 + \phi_2$, where $\phi_1 \in \mathcal{E}$ and $\phi_2 \in \mathcal{E}^\perp$. We write $\mathcal{H} = \mathcal{E} \oplus \mathcal{E}^\perp$ and say that $\mathcal{E}$ and $\mathcal{E}^\perp$ form an* orthogonal decomposition *of the inner-product space $\mathcal{H}$.*

We can generalize the concept of orthogonal decomposition of a complete inner-product space $\mathcal{H}$ to an orthogonal decomposition

$$\mathcal{H} = \mathcal{E}_1 \oplus \ldots \oplus \mathcal{E}_k$$

of $\mathcal{H}$ into k mutually orthogonal subspaces $\mathcal{E}_1, \ldots, \mathcal{E}_k$, such that each $\psi \in \mathcal{H}$ has a unique representation $\psi = \phi_1 + \ldots + \phi_k$, where $\phi_j \in \mathcal{E}_j$ for all j with $1 \le j \le k$.

2.2 Dirac's notation

For any Hilbert space $\mathcal{H}$ and any its state ϕ a linear mapping $f_\phi\colon \mathcal{H} \to \mathbb{C}$ can be defined by assigning $f_\phi(\psi) = \langle\phi|\psi\rangle$. Moreover, we can show that for any continuous linear mapping $f\colon \mathcal{H} \to \mathbb{C}$ there is a unique state $\phi_f \in \mathcal{H}$, such that $f(\psi) = \langle\phi_f|\psi\rangle$ for all $\psi \in \mathcal{H}$. The space of all such linear mappings of the Hilbert space forms a Hilbert space again. We call this space a *dual* Hilbert space

or a *conjugate* Hilbert space and denote it as $\mathcal{H}^*$. The inner product of the conjugate Hilbert space is defined for all $f, g \in \mathcal{H}^*$ by

$$\langle f|g\rangle = \langle \phi_f|\phi_g\rangle.$$

Thanks to the duality between the Hilbert space $\mathcal{H}$ and its conjugate Hilbert space $\mathcal{H}^*$ the Dirac's *bra-ket* notation can be introduced. A vector ϕ of the Hilbert space $\mathcal{H}$ is denoted by a *ket-vector* $|\phi\rangle$. The corresponding mapping f_ϕ of the conjugate Hilbert space is denoted by a *bra-vector* $\langle\phi|$. For every $\phi, \psi \in \mathcal{H}$ we have

$$\langle\phi|\,(|\psi\rangle) = f_\phi(\psi) = \langle\phi|\psi\rangle.$$

In the case of an n-dimensional Hilbert space, we can indetify the ket-vector $|\psi\rangle$ with an n-dimensional column vector and a bra-vector $\langle\phi|$ with an n-dimensional row vector. The inner product $\langle\phi|\psi\rangle$ is then a complex number, while the *outer product* $|\psi\rangle\langle\phi|$ is an $n \times n$ matrix. For the outer product we have

$$|\psi\rangle\langle\phi|\,(|\omega\rangle) = |\psi\rangle(\langle\phi|\omega\rangle) = \langle\phi|\omega\rangle|\psi\rangle$$

for all $\psi, \phi, \omega \in \mathcal{H}$.

2.3 Linear operators

A *linear operator* of a Hilbert space $\mathcal{H}$ is a linear mapping $\mathsf{A}\colon\mathcal{H} \to \mathcal{H}$. An application of the linear operator A of the Hilbert space to a vector $|\phi\rangle$ from that space is denoted by $|\mathsf{A}\phi\rangle$ or simply by $\mathsf{A}|\phi\rangle$. Any linear operator A of the Hilbert space $\mathcal{H}$ is also a linear operator of the conjugate Hilbert space $\mathcal{H}^*$. It maps every linear mapping $\langle\psi|$ of the conjugate space to a linear mapping denoted by $\langle\phi|\,\mathsf{A}$.

We can represent any linear operator A of an n-dimensional Hilbert space $\mathcal{H}$ with the basis $\Theta = \{|\theta_j\rangle\}_{j=1}^n$ by an n-dimensional matrix with the number $\langle\theta_j|\,\mathsf{A}|\theta_k\rangle$ in its j-th row and k-th column. In this case the j-th row of the matrix is the vector $\langle\theta_j|\,\mathsf{A}$ and the k-th column of the matrix is the vector $\mathsf{A}|\theta_k\rangle$.

Projection operators are of special importance among linear operators. If $\mathcal{H} = \mathcal{E}_1 \oplus \ldots \oplus \mathcal{E}_k$ is an orthogonal decomposition of a Hilbert space $\mathcal{H}$, then every vector ϕ from the space $\mathcal{H}$ has a unique representation $\phi = \phi_1 + \ldots + \phi_k$ with $\phi_j \in \mathcal{E}_j$ for all j, where $1 \leq j \leq k$. The mappings defined by the identities $\mathsf{P}_{\mathcal{E}_j}(\phi) = \phi_j$ for all j are called *projection operators* onto the corresponding subspaces $\mathcal{E}_j$.

A special role among linear operators have adjoint and self-adjoint operators. We define them in the following two definitions.

Definition 2.9 *The* norm *of a linear operator* A *of a Hilbert space* $\mathcal{H}$ *is defined as*

$$\|\mathsf{A}\| = \sup\big\{\|\mathsf{A}|\phi\rangle\| \;\big|\; \phi \in \mathcal{H} \;\wedge\; \|\phi\| = 1\big\}.$$

If $\|\mathsf{A}\|$ *is finite, then the operator* A *is called* bounded.

Definition 2.10 *The* adjoint *operator to a bounded linear operator* A *of a Hilbert space* $\mathcal{H}$ *is an operator* A^* *such that*

$$\langle\psi|\mathsf{A}\phi\rangle = \langle\mathsf{A}^*\psi|\phi\rangle$$

for all $\psi, \phi \in \mathcal{H}$. *If* $\mathsf{A} = \mathsf{A}^*$, *then the operator* A *is called* self-adjoint.

The self-adjoint operators correspond to *Hermitian matrices*, i.e., the matrices A with $\mathsf{A} = \mathsf{A}^*$. A bounded linear operator A is *unitary*, if $\mathsf{A}\mathsf{A}^* = \mathsf{A}^*\mathsf{A} = \mathsf{I}$, where I is an identity operator.

A self-adjoint operator A of a finite dimensional Hilbert space $\mathcal{H}$ has a so-called *spectral representation*. If $\lambda_1, \ldots, \lambda_k$ are its distinct eigenvalues, then we can express the operator A in the form $\mathsf{A} = \lambda_1\mathsf{P}_{\mathcal{E}_1} + \ldots + \lambda_k\mathsf{P}_{\mathcal{E}_k}$, where $\mathcal{E}_j$ is the subspace of the Hilbert space $\mathcal{H}$ spanned by the eigenvectors corresponding to λ_j.

2.4 Evolution of quantum systems

In the evolution of a quantum system some transformation of the initial state is performed. In the formal representation of the quantum system, some operator A is used to map one state into another. An evolution of an isolated quantum system corresponds to a transformation by a unitary operator in its Hilbert space. As unitary operators preserve the norm, they can be seen as performing rotations on quantum states.

For a finite dimensional quantum system, we can show that any linear operator representing quantum evolution in that system has to be unitary as follows: A quantum evolution operator A has to map quantum states into quantum states. Therefore, for any state x of the quantum system the identity $\langle \mathsf{A}x|\mathsf{A}x\rangle = \langle x|x\rangle = 1$ holds. As a consequence we have $\langle x|x\rangle = \langle \mathsf{A}^*\mathsf{A}x|x\rangle$ and hence

$$\mathsf{A}^*\mathsf{A} = \mathsf{I}.$$

Therefore, the operator A is unitary.

Note, that the equality $\mathsf{A}^*\mathsf{A} = \mathsf{I}$ corresponds to the condition that row vectors of A are orthonormal and the equality $\mathsf{A}\mathsf{A}^* = \mathsf{I}$ corresponds to the assertion that column vectors of A are orthonormal. If A is a finite dimensional, then $\mathsf{A}\mathsf{A}^* = \mathsf{I}$ if and only if $\mathsf{A}^*\mathsf{A} = \mathsf{I}$.

In the general case, the evolution of a quantum system is described by the linear Schrödinger equation

$$i\hbar\frac{\partial|\psi(t)\rangle}{\partial t} = H(t)|\psi(t)\rangle,$$

where $\hbar$ is the Planck constant, $H(t)$ is an observable of the system called Hamiltonian of the system in time t, and $|\psi(t)\rangle$ is the state of the system in time t. If the Hamiltonian is time independent, the formal solution of the Schrödinger equation has the form

$$|\phi(t)\rangle = \mathsf{U}(t)|\phi(0)\rangle,$$

where $\mathsf{U}(t) = e^{-Ht/\hbar}$ is an evolution operator that can be represented by a unitary matrix.

2.5 Observables and measurements

In order to extract information from a quantum system we have to *observe* the system – to perform a *measurement* of the system. In this thesis we consider only so-called *sharp observables*, which correspond to *projection measurements*.

The numerical outcome of a measurement of a pure state $|\psi\rangle$ of a Hilbert space $\mathcal{H}$ with respect to an observable specified by a self-adjoint operator A is one of the eigenvalues λ_j of A. As the side effect of the measurement the state $|\psi\rangle$ collapses into a state $|\phi\rangle$. The eigenvalue λ_j is obtained with probability

$$P(\lambda_j) = \|\mathsf{P}_{\mathcal{E}_j}|\psi\rangle\|^2 = \langle\psi|\,\mathsf{P}_{\mathcal{E}_j}|\psi\rangle,$$

where $\mathcal{E}_j$ is the subspace of $\mathcal{H}$ spanned by the eigenvectors corresponding to λ_j. The new state $|\phi\rangle$, into which $|\psi\rangle$ collapses, has the form

$$|\phi\rangle = \frac{\mathsf{P}_{\mathcal{E}_j}|\psi\rangle}{\sqrt{\langle\psi|\,\mathsf{P}_{\mathcal{E}_j}|\psi\rangle}}.$$

The measurement of $|\psi\rangle$ with respect to A irreversibly destroys ψ, unless ψ is an eigenvector of A.

An equivalent approach to model the observation through a special orthogonal decomposition, called observable, of the Hilbert space can be done.

Definition 2.11 *An* observable *of a Hilbert space $\mathcal{H}$ is a pair $\mathcal{O} = (\{\mathcal{E}_j\}_{j=1}^k, \mu)$, where $\mathcal{E}_1 \oplus \ldots \oplus \mathcal{E}_k = \mathcal{H}$ is an orthogonal decomposition of the Hilbert space $\mathcal{H}$ and $\mu\colon \{\mathcal{E}_j\}_{j=1}^k \to \mathbb{R}$ is an injective mapping.*

Let $|\phi\rangle$ be a state and $\mathcal{O} = (\{\mathcal{E}_j\}_{j=1}^k, \mu)$ be an observable of a Hilbert space $\mathcal{H}$. We can express $|\phi\rangle$ uniquely as a linear superposition of its projections with respect to corresponding subspaces of the orthogonal decomposition of the Hilbert space

$$|\phi\rangle = \sum_{j=1}^{k} \alpha_j |\phi_{\mathcal{E}_j}\rangle,$$

where $|\phi_{\mathcal{E}_j}\rangle$ is a projection of $|\phi\rangle$ into $\mathcal{E}_j$. A measurement of $|\phi\rangle$ by the observable $\mathcal{O}$ has the following consequences:

1. One of the subspaces $\mathcal{E}_1, \ldots, \mathcal{E}_k$, say $\mathcal{E}_j$, is selected, and the value $\mu(\mathcal{E}_j)$ is acquired. The probability of selecting the subspace $\mathcal{E}_j$ is $|\alpha_j|^2$.
2. After the measurement, the state $|\phi\rangle$ collapses into the state $\dfrac{|\phi_{\mathcal{E}_j}\rangle}{\||\phi_{\mathcal{E}_j}\rangle\|}$.
3. The only classical information obtained by the measurement is the value $\mu(\mathcal{E}_j)$ of the function μ. Any information not in $|\phi_{\mathcal{E}_j}\rangle$ is irreversibly lost.

A measurement with respect to the observable $\mathcal{O} = (\{\mathcal{E}_j\}_{j=1}^k, \mu)$ causes the quantum system to behave randomly and to destroy its original state ϕ unless the state belongs entirely into one of the subspaces of the orthogonal decomposition of the Hilbert space.

Chapter 3

Models of quantum finite automata

In this chapter, we formally define the models of quantum automata discussed in this thesis. In section 3.1, we define two-way quantum finite state automata introduced in [KW97], which are the quantum analogue of classical two-way finite state automata. The following two sections define two special cases of 2QFA. In section 3.2 we define 1.5-way quantum finite state automata introduced in [AI99], which are forbidden to move their head to the left, but still may let their head stationary on the currently read tape symbol. In section 3.3 we define the last variant of 2QFA, namely one-way quantum finite state automata introduced in [KW97], which are forced to move their head to the right in every evolution step. Only the last one of these three models is purely finite, since evolution of 2QFA and 1.5QFA may lead into a large superposition of many heads on different tape symbols and so the size of their quantum memory may be proportional to the input word length.

Finally, in section 3.4 we define two-way finite automata with quantum and classical states, an intermediate model between 1QFA and 2QFA, introduced in [AW02]. This model is a generalized classical or probabilistic two-way finite state automaton, which is extended by additional finite quantum memory.

3.1 Two-way quantum finite state automaton

The model of two-way quantum finite state automata introduced by Kondacs and Watrous in [KW97] is a quantum analogue to the classical two-way finite automata. In this section we provide a formal definition of this model.

Definition 3.1 *A* two-way quantum finite state automaton *(2QFA) we define as a sextuple* $M = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$*, where* Q *is a finite set of states,* Σ *is a finite input alphabet,* $\delta: Q \times \Gamma \times Q \times D \to \mathbb{C}$ *is a transition function,* $q_0 \in Q$ *is an initial state, and* $Q_{acc} \subseteq Q$ *and* $Q_{rej} \subseteq Q$ *are sets of accepting and rejecting states, respectively. It is assumed that* $Q_{acc} \cap Q_{rej} = \emptyset$*,* $q_0 \notin Q_{acc} \cup Q_{rej}$*, symbols* $¢, \$ \notin \Sigma$*,* $\Gamma = \Sigma \cup \{¢, \$\}$ *and* $D = \{-1, 0, 1\}$*.*

Elements of Q_{acc} and Q_{rej} are called *halting* states and elements of $Q_{non} = Q \setminus (Q_{acc} \cup Q_{rej})$ *non-halting* states. Symbols ¢ and \$ are used to mark the left and the right end of an input word, respectively. These symbols together with the input alphabet form *tape alphabet* Γ.

The 2QFA M ran on an input word $w \in \Sigma^*$ can be seen as it were operating on a circular *tape*. This tape is circular in the sense that moving to the right from the last tape square the automaton head jumps to the first tape square. Reversely, moving to the left from the first tape square the automaton head jumps to the last tape square. For the non-empty input word w, the corresponding tape x has length $|w| + 2$ and takes the form $x = ¢w\$$. For technical reasons, if $w = \varepsilon$ the corresponding tape is $x = ¢\$\$$.

Definition 3.2 *The* set of configurations *of 2QFA M on a tape x of length n is $C_n^M = Q \times \mathbb{Z}_n$. A* superposition *of M on this tape is any norm 1 element of the finite-dimensional Hilbert space $\mathcal{H}_n^M = \ell_2(C_n^M)$. For clarity, we write any base element $|(q,k)\rangle$ of $\mathcal{H}_n^M$ with $q \in Q$ and $k \in \mathbb{Z}_n$ as $|q,k\rangle$ only.*

By $C_{n,acc}^M$ ($C_{n,rej}^M$, $C_{n,non}^M$) we mean the set of accepting (rejecting, non-halting) configurations of 2QFA M on tape x of length n (i.e. $C_{n,acc}^M = Q_{acc}^M \times \mathbb{Z}_n$, $C_{n,rej}^M = Q_{rej}^M \times \mathbb{Z}_n$, and $C_{n,non}^M = Q_{non}^M \times \mathbb{Z}_n$).

We use the Dirac notation to express superpositions. For every $c \in C_n^M$, $|c\rangle$ denotes the unit vector with value 1 at c and 0 elsewhere. Any other element of $\mathcal{H}_n^M$ may be expressed as a linear combination of its basis vectors. For a superposition $|\phi\rangle = \sum_{c \in C_n^M} \alpha_c |c\rangle$, the number α_c is the amplitude associated with c in the superposition $|\phi\rangle$. We have defined Hilbert spaces in chapter 2. For a detailed overview of Hilbert spaces theory, see [NC00] or [Gru99].

Notation 3.3 For a word w by notation $(w)_i$, where $0 \leq i < |w|$, we mean the i-th symbol of the word w counting from zero (i.e. $(abcd)_1 = b$). By notation $(w)_{i\ldots j}$ we mean the subword $(w)_i \ldots (w)_j$ of the word w, and by notation $\#_\alpha(w)$ we mean the number of symbols α in the word w.

The transition function δ of the 2QFA M is to be interpreted, such that for each $q, p \in Q$, $\alpha \in \Gamma$ and $d \in D$, the number $\delta(q, \alpha, p, d)$ represents the amplitude with which the automaton in state q currently scanning symbol α changes its state to p and moves its head in direction d. For a tape x the transition function induces a *time-evolution operator* of M for the tape x on the Hilbert space $\mathcal{H}_{|x|}^M$.

Definition 3.4 *For a 2QFA M on a tape x of length n we define a* time-evolution operator U_x^M *on the Hilbert space $\mathcal{H}_n^M$ as follows:*

$$\mathsf{U}_x^M |q,k\rangle = \sum_{\substack{p \in Q \\ d \in D}} \delta(q, (x)_k, p, d) |p, (k+d) \bmod n\rangle, \text{ where } (q,k) \in C_n^M, \tag{3.1}$$

and extend to all vectors of $\mathcal{H}_n^M$ by linearity.

Definition 3.5 *A 2QFA M is* well-formed *if for every tape $x \in \Gamma^3\Gamma^*$ the corresponding operator U_x^M defined by definition 3.4 is unitary.*

The time-evolution operator U_x^M specifies how the automaton M evolves on the tape x, assuming that the automaton superposition is not observed by any outside observer. To yield information about automaton evolution it uses an observable which correspond to determining whether the automaton is in an accepting, rejecting or non-halting state.

Definition 3.6 *For a 2QFA M on a tape x of length n let $\mathcal{O}_n^M$ be an* observable *corresponding to the decomposition $\mathcal{E}_{acc}^M \oplus \mathcal{E}_{rej}^M \oplus \mathcal{E}_{non}^M$ of $\mathcal{H}_n^M$, where $\mathcal{E}_{acc}^M = Span(\{|c\rangle \mid c \in C_{n,acc}^M\})$, $\mathcal{E}_{rej}^M = Span(\{|c\rangle \mid c \in C_{n,rej}^M\})$ and $\mathcal{E}_{non}^M = Span(\{|c\rangle \mid c \in C_{n,non}^M\})$. Outcome of this observation is "accept", "reject" or "non-halting" accordingly.*

Definition 3.7 Evolution *of a well-formed 2QFA M on a tape x of length n begins in the superposition equal to $|q_0, 0\rangle$. In every step of the evolution the unitary transformation U_x^M is applied on the superposition, followed by a measurement using the observable $\mathcal{O}_n^M$. The evolution continues until the result of the measurement is "accept" or "reject", when the evolution halts.*

Notation 3.8 For a given well-formed 2QFA M we write the probability that the evolution of M on the input word w halts with result "accept" ("reject") as $P_{acc}^M(w)$ ($P_{rej}^M(w)$).

If there is no ambiguity we omit M in the symbols defined in previous paragraphs and write only C_n, $\mathcal{H}_n$, U_x, $\mathcal{O}_n$ and $P_{acc}(w)$ instead of C_n^M, $\mathcal{H}_n^M$, U_x^M, $\mathcal{O}_n^M$ and $P_{acc}^M(w)$, respectively.

Definition 3.9 *A well-formed 2QFA M recognizes a language $L \subseteq \Sigma^*$ with bounded error if for every word $w \in \Sigma^*$ we have $P^M_{acc}(w) + P^M_{rej}(w) = 1$, and there is $\varepsilon > 0$, such that for every word $w \in L$ is $P^M_{acc}(w) > 1/2 + \varepsilon$ and for every word $w \in \Sigma^* \setminus L$ is $P^M_{rej}(w) > 1/2 + \varepsilon$.*

Notation 3.10 If there is a language a well-formed 2QFA M recognizes, we denote this language by $L(M)$. Note, that for every well-formed 2QFA there is at most one language recognized by it.

Kondacs and Watrous [KW97] introduced an effective criterion for well-formedness of 2QFA. This criterion is analogous to Bernstein and Vazirani's criterion for well-formedness of quantum Turing machines [BV93].

Lemma 3.11 [KW97, Proposition 1] *A 2QFA $M = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$ is well-formed if and only if for every $\alpha, \alpha_1, \alpha_2 \in \Gamma$ and $q_1, q_2 \in Q$ the following hold:*

$$\sum_{p \in Q,\ d \in D} \delta^*(q_1, \alpha, p, d)\delta(q_2, \alpha, p, d) = \begin{cases} 1 & \text{if } q_1 = q_2 \\ 0 & \text{if } q_1 \neq q_2 \end{cases} \tag{3.2}$$

$$\sum_{p \in Q} \delta^*(q_1, \alpha_1, p, 1)\delta(q_2, \alpha_2, p, 0) + \delta^*(q_1, \alpha_1, p, 0)\delta(q_2, \alpha_2, p, -1) = 0 \tag{3.3}$$

$$\sum_{p \in Q} \delta^*(q_1, \alpha_1, p, 1)\delta(q_2, \alpha_2, p, -1) = 0. \tag{3.4}$$

The condition (3.2) of the previous lemma is the local probability and orthogonality condition of the automaton evolution unitarity. The other two conditions (3.3) and (3.4) are the separability conditions. These three conditions together are equivalent to the requirement that the evolution of M is unitary on every tape.

Verification of the well-formedness conditions from the previous lemma may be in general a relatively tedious task. Kondacs and Watrous introduced a method, by which some well-formed automata can be specified more easily. We call them *simple*. However, it is still an open question, whether the class of languages recognized by simple 2QFA is equal to the class of languages recognized by all 2QFA.

Definition 3.12 *A 2QFA $M = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$ is* simple*, if for each $\alpha \in \Gamma$ there is a linear operator V_α on the Hilbert space $\ell_2(Q)$ and a function $D\colon Q \to \{-1, 0, +1\}$ such that for each $p, q \in Q$, $\alpha \in \Gamma$ and $d \in D$*

$$\delta(q, \alpha, p, d) = \begin{cases} \langle p|\, \mathsf{V}_\alpha |q\rangle & \text{if } D(p) = d \\ 0 & \text{if } D(p) \neq d. \end{cases} \tag{3.5}$$

Informally, the method decomposes the transition function of the automaton into two parts, one transforming the states and the other moving the tape head.

It is easy to see, from lemma 3.11, that for a simple 2QFA the following holds.

Lemma 3.13 *A simple 2QFA M is well-formed if and only if for every $q_1, q_2 \in Q$ and $\alpha \in \Gamma$ is*

$$\sum_{p \in Q} \langle p|\, \mathsf{V}_\alpha |q_1\rangle^* \langle p|\, \mathsf{V}_\alpha |q_2\rangle = \begin{cases} 1 & \text{if } q_1 = q_2 \\ 0 & \text{if } q_1 \neq q_2. \end{cases}$$

This holds if and only if every operator V_α is unitary.

Notation 3.14 We write the class of languages recognized by 2QFA as $\mathcal{L}_{\text{2QFA}}$ and the class of languages recognized by simple 2QFA as $\mathcal{L}_{\text{simple}-\text{2QFA}}$.

3.2 1.5-way quantum finite state automaton

Amano and Iwama [AI99] defined 1.5QFA as a special case of 2QFA which cannot move its head to the left. However, they did not restrict 1.5QFA to move its head to the right from the right end-marker and, due to the tape circularity, allowed the head to return back to the left end-marker and read the input word once again.

As 1.5QFA is meant to be a model of quantum automata which head should read the input word just once, we narrow the Amano's and Iwama's definition of 1.5QFA, such that the automaton may not move its head beyond the right end-marker symbol. Nevertheless, the results showed in [AI99] hold for the narrowed definition of 1.5QFA, too.

Definition 3.15 *An* 1.5-way quantum finite state automaton *(1.5QFA) we define as a 2QFA* $M = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$*, such that for every* $q, p \in Q$ *and* $\alpha \in \Gamma$ *is* $\delta(q, \alpha, p, -1) = 0$ *and for every* $q \in Q_{non}$ *and* $p \in Q$ *is* $\delta(q, \$, p, 1) = 0$*.*

Note, that we allow $\delta(q, \$, p, 1) \neq 0$ for $q \in Q_{acc} \cup Q_{rej}$, as the amplitude of the automaton being in any halting state is zero at the beginning of every evolution step. This allows us to extend the definition 3.12 of simple 2QFA to the 1.5QFA case. If we asked $\delta(q, \$, p, 1)$ be zero for all $q \in Q$, it would be impossible to construct the unitary operator $\mathsf{V}_\$$ satisfying the condition from the following definition.

Definition 3.16 *An 1.5QFA is* simple*, if it is a simple 2QFA and* $\langle p| \mathsf{V}_\$ |q\rangle = 0$ *for all* $q \in Q_{non}$ *and* $p \in Q$ *with* $D(p) = 1$*.*

As a special case of lemma 3.11 we can formulate a criterion for well-formedness of 1.5QFA.

Lemma 3.17 *An 1.5QFA* $M = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$ *is well-formed if and only if for every* $\alpha, \alpha_1, \alpha_2 \in \Gamma$ *and* $q_1, q_2 \in Q$ *the following hold:*

$$\sum_{p \in Q,\ d \in \{0,1\}} \delta^*(q_1, \alpha, p, d)\delta(q_2, \alpha, p, d) = \begin{cases} 1 & \text{if } q_1 = q_2 \\ 0 & \text{if } q_1 \neq q_2 \end{cases} \tag{3.6}$$

$$\sum_{p \in Q} \delta^*(q_1, \alpha_1, p, 1)\delta(q_2, \alpha_2, p, 0) = 0. \tag{3.7}$$

Notation 3.18 We write the class of languages recognized by 1.5QFA as $\mathcal{L}_{1.5\text{QFA}}$ and the class of languages recognized by simple 1.5QFA as $\mathcal{L}_{\text{simple}-1.5\text{QFA}}$.

3.3 One-way quantum finite state automaton

One-way quantum finite state automaton is a special case of 2QFA which may move its head only to the right. Moreover, the evolution of 1QFA may not continue after reading the right end-marker. Therefore, its superposition after reading this end-marker must be fully halting and so it must belong to the subspace $\mathcal{E}_{acc} \oplus \mathcal{E}_{rej}$. We achieve this by zeroing the amplitude with which the automaton in a non-halting state reading the right end-marker symbol changes its state to any non-halting state.

Definition 3.19 *An* one-way quantum finite state automaton *(1QFA) we define as a 2QFA* $M = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$*, such that for every* $q, p \in Q$ *and* $\alpha \in \Gamma$ *is* $\delta(q, \alpha, p, -1) = \delta(q, \alpha, p, 0) = 0$ *and for every* $q, p \in Q_{non}$ *is* $\delta(q, \$, p, 1) = 0$*.*

It is not difficult to show that for any well-formed 1QFA, the following lemma holds.

Lemma 3.20 *Let $M = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$ be a well-formed 1QFA. For every $\alpha \in \Gamma$ there is a unitary operator V_α on the Hilbert space $\ell_2(Q)$, such that*

$$\delta(q, \alpha, p, d) = \begin{cases} \langle p| \mathsf{V}_\alpha |q\rangle & \text{if } d = 1 \\ 0 & \text{if } d \neq 1 \end{cases}$$

for all $p, q \in Q$ and $d \in D$.

As 1QFA are allowed to move their heads only to the right, they never get into a superposition of two or more heads reading different tape symbols.

Evolution of a well-formed 1QFA $M = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$ on a tape x, may be seen as an evolution performed in the Hilbert space $\ell_2(Q)$. This evolution starts in the superposition equal to $|q_0\rangle$. In its k-th step, where $0 \leq k < |x|$, the operator $\mathsf{V}_{(x)_k}$, as defined in the previous lemma, is applied on the current superposition. This transformation is followed by a measurement by the observable corresponding to the orthogonal decomposition $\mathcal{E}_{acc} \oplus \mathcal{E}_{rej} \oplus \mathcal{E}_{non}$ of $\ell_2(Q)$, where $\mathcal{E}_{acc} = Span(\{|q\rangle \mid q \in Q_{acc}\})$, $\mathcal{E}_{rej} = Span(\{|q\rangle \mid q \in Q_{rej}\})$ and $\mathcal{E}_{non} = Span(\{|q\rangle \mid q \in Q_{non}\})$. If the result of the measurement is in the subspace $\mathcal{E}_{acc}$ or $\mathcal{E}_{rej}$ the automaton accepts or rejects, respectively. Otherwise it continues in the next evolution step by applying the next V_α operator. Note, that after applying the operator $\mathsf{V}_\$$ the evolution halts, as $\mathsf{V}_\$|q\rangle \in \mathcal{E}_{acc} \oplus \mathcal{E}_{rej}$ for any $|q\rangle \in \mathcal{E}_{non}$.

On the contrary to 2QFA and 1.5QFA, every 1QFA uses only a constant size of quantum memory. The 2QFA and 1.5QFA can be in large superpositions of many heads on different tape symbols and so their quantum memory can be proportional to the input word length.

Notation 3.21 We write the class of languages recognized by 1QFA as $\mathcal{L}_{1QFA}$.

3.4 Two-way finite automaton with quantum and classical states

A two-way finite automaton with quantum and classical states is a classical two-way finite state automaton which has an additional quantum memory of a constant size. The size of this memory is independent of the input word. The automaton may perform quantum transformations and measurements on its quantum memory. The transformations and the measurements performed by the automaton are determined by the classical states of the automaton and the read symbols. The results of the measurements determine how the classical part of the automaton evolves.

Notation 3.22 For a finite set A we denote by $Operators(A)$ the set of all unitary operators acting on the Hilbert space $\ell_2(A)$, and by $Decompositions(A)$ the set of all orthogonal decompositions of $\ell_2(A)$ into $|A|$ (not necessarily proper) subspaces represented by $|A|$-tuples $(\mathcal{E}_1, \ldots, \mathcal{E}_{|A|})$, where $\mathcal{E}_j$ are the particular subspaces.

Definition 3.23 *A* 2-way finite automaton with quantum and classical states *(2QCFA) is an octuple $M = (Q, S, \Sigma, \delta, q_0, s_0, S_{acc}, S_{rej})$, where Q and S are finite sets of quantum and classical states, respectively, Σ is a finite input alphabet, $\delta\colon S \times \Gamma \to Operators(Q) \times Decompositions(Q) \times S^{|Q|} \times D^{|Q|}$ is a transition function, $q_0 \in Q$ and $s_0 \in S$ are initial quantum and classical states, respectively, and $S_{acc} \subseteq S$ and $S_{rej} \subseteq S$ are sets of accepting and rejecting classical states, respectively. It is assumed that $S_{acc} \cap S_{rej} = \emptyset$, symbols $¢, \$ \notin \Sigma$, $\Gamma = \Sigma \cup \{¢, \$\}$ and $D = \{-1, 0, 1\}$.*

Elements of S_{acc} and S_{rej} are called *halting* states and elements of $S_{non} = S \setminus (S_{acc} \cup S_{rej})$ *non-halting* states. Symbols ¢ and \$ are used to mark the left and the right end of an input word, respectively. These symbols together with the input alphabet form the *tape alphabet* Γ. The automaton operates on a *tape*. For the input word w, the corresponding tape has the form $¢w\$$.

Definition 3.24 *Let M be a 2QCFA. A* configuration *of M on an input word w is a quadruple $(s, |\phi\rangle, ¢w\$, k)$, where $s \in S$ is the current classical state, $|\phi\rangle \in \ell_2(Q)$ is the current superposition of the quantum memory, $¢w\$$ is the automaton tape content, and k with $0 \leq k \leq |w|$ is the current position of the automaton head on the tape.*

A configuration containing accepting classical state is an accepting configuration. Similarly, a configuration with rejecting classical state is a rejecting configuration. All other configurations are non-halting configurations.

Definition 3.25 *Let $(s, |\phi\rangle, x, k)$ be a non-halting configuration of a 2QCFA M. If $\delta(s, (x)_k) = (\mathsf{V}, (\mathcal{E}_1, \ldots, \mathcal{E}_{|Q|}), (s_1, \ldots, s_{|Q|}), (d_1, \ldots, d_{|Q|}))$, then one* evolution step *of the automaton M in this configuration consists of applying the operator V on the superposition $|\phi\rangle$ followed by a measurement by the observable corresponding to the orthogonal decomposition $\mathcal{E}_1 \oplus \ldots \oplus \mathcal{E}_{|Q|}$ of $\ell_2(Q)$. If the result of the measurement is in the subspace $\mathcal{E}_j$, where $1 \leq j \leq |Q|$, the automaton changes its classical state to the state s_j and moves its head in direction d_j. The resulting configuration of the automaton is $(s_j, \mathsf{P}_j\mathsf{V}|\phi\rangle, x, k + d_j)$, where P_j is a normalized projection operator to subspace $\mathcal{E}_j$.*

Remark 3.26 We assume that the transition function of any 2QCFA is defined such that the automaton head never moves to the left when reading the left end-marker symbol ¢, and similarly never moves to the right when reading the right end-marker symbol \$. Formally, for every $s \in S$ we assume that $\delta(s, ¢) \in \mathit{Operators}(Q) \times \mathit{Decompositions}(Q) \times S^{|Q|} \times \{0, 1\}^{|Q|}$ and $\delta(s, \$) \in \mathit{Operators}(Q) \times \mathit{Decompositions}(Q) \times S^{|Q|} \times \{-1, 0\}^{|Q|}$.

Definition 3.27 *A 2QCFA M on an input word w begins its* evolution *with the configuration $(s_0, |q_0\rangle, ¢w\$, 0)$. It performs evolution steps, as defined in the previous definition, until it reaches any accepting or rejecting configuration, when it halts and accepts or rejects, respectively.*

Notation 3.28 For a given 2QCFA M we write the probability that M accepts (rejects) on the input word w as $P^M_{acc}(w)$ ($P^M_{rej}(w)$). If there is no ambiguity we omit the symbol M in the notation.

Definition 3.29 *A 2QCFA M recognizes a language $L \subseteq \Sigma^*$ with bounded error if for every word $w \in \Sigma^*$ we have $P^M_{acc}(w) + P^M_{rej}(w) = 1$, and there is $\varepsilon > 0$, such that for every word $w \in L$ is $P^M_{acc}(w) > 1/2 + \varepsilon$ and for every word $w \in \Sigma^* \setminus L$ is $P^M_{rej}(w) > 1/2 + \varepsilon$.*

Notation 3.30 If there is a language a 2QCFA M recognizes, we denote this language by $L(M)$. Note, that for every 2QCFA there is at most one language recognized by it.

Notation 3.31 We write the class of languages recognized by 2QCFA as $\mathcal{L}_{\mathrm{2QCFA}}$.

Chapter 4

Closure under inverse homomorphism

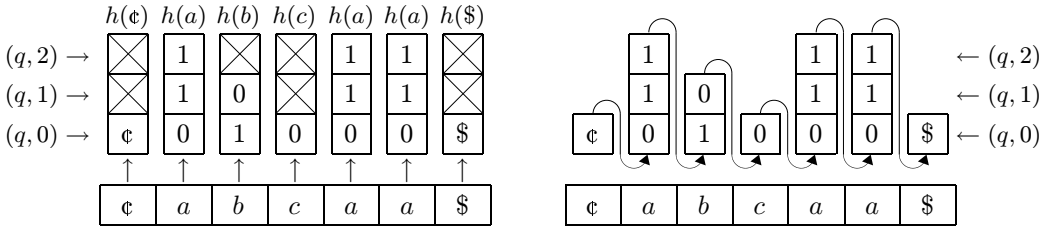
Brodsky and Pippenger [BP02] showed that the class of languages recognized by one-way quantum finite state automata is closed under general inverse homomorphism. The class of languages recognized by two-way finite automata with quantum and classical states is closed under general inverse homomorphism [Kos̆05], too. In this chapter, we investigate this closure property for the case of 1.5-way and 2-way quantum finite state automata.

4.1 1.5QFA and inverse non-erasing homomorphism

We prove that the class of languages recognized by 1.5QFA is closed under inverse non-erasing homomorphism. Take a well-formed 1.5QFA $M = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$ recognizing the language $L(M)$, and a homomorphism $h: \Sigma'^* \to \Sigma^*$, such that $|h(\alpha)| \geq 1$ for all $\alpha \in \Sigma'$. We construct a new 1.5QFA M' recognizing the language $h^{-1}(L(M))$.

The new automaton ran on an input word w will simulate the original automaton on the input word $h(w)$. For each symbol α of its input word, it will keep the superposition of the original automaton heads reading any symbol of the subword $h(\alpha)$ in a superposition of heads reading the symbol α. The different positions of the simulated heads within the word $h(\alpha)$ will be distinguished by different states of the heads reading the symbol α of the new automaton input word.

Figure 4.1: 1.5QFA M' on the input word $abcaa$, with $h(a) = 011$, $h(b) = 10$ and $h(c) = 0$.



Let $\Gamma' = \Sigma' \cup \{¢, \$\}$, extend the non-erasing homomorphism h to $\Gamma'^* \to \Gamma^*$ such that $h(¢) = ¢$ and $h(\$) = \$$, and let $s = \max\{|h(\alpha)| \mid \alpha \in \Gamma'\}$. We define the new automaton M' as the sextuple $(Q', \Sigma', \delta', q'_0, Q'_{acc}, Q'_{rej})$, where $Q' = Q \times \mathbb{Z}_s$, $q'_0 = (q_0, 0)$, $Q'_{acc} = \{(q, k) \mid q \in Q_{acc} \wedge k \in \mathbb{Z}_s\}$ and $Q'_{rej} = \{(q, k) \mid q \in Q_{rej} \wedge k \in \mathbb{Z}_s\}$. The transition function of M' for every $p, q \in Q$ and $\alpha \in \Gamma'$

we define as follows:

$$\begin{aligned}
\delta'((p,k),\alpha,(q,k),0) &= \delta(p,(h(\alpha))_k,q,0);\ 0 \le k < |h(\alpha)|, && (4.1)\\
\delta'((p,k),\alpha,(q,k+1),0) &= \delta(p,(h(\alpha))_k,q,1);\ 0 \le k < |h(\alpha)|-1, && (4.2)\\
\delta'((p,k),\alpha,(q,0),1) &= \delta(p,(h(\alpha))_k,q,1);\ k = |h(\alpha)|-1, && (4.3)\\
\delta'((p,k),\alpha,(p,k),0) &= 1;\ |h(\alpha)| \le k < s, && (4.4)\\
\delta'(\text{anything else}) &= 0. && (4.5)
\end{aligned}$$

As the automaton M is an 1.5QFA, $\delta(q,\$,p,1) = 0$ for all $q \in Q_{non}$ and $p \in Q$. Therefore, by definition of M', $\delta'((q,k),\$,(p,j),1) = 0$ for all $(q,k) \in Q'_{non}$ and $(p,j) \in Q'$. Also, $\delta'((q,k),\alpha,(p,j),-1) = 0$ for all $(q,k),(p,j) \in Q'$ and $\alpha \in \Gamma'$. Hence, the constructed automaton M' meets all conditions of definition 3.15 and so M' is an 1.5QFA.

To prove the closure property of the class of languages recognized by 1.5QFA we need to show that the constructed 1.5QFA M' is well-formed and recognizes the language $L(M') = h^{-1}(L(M))$. We show it in the following two lemmas.

Lemma 4.1 *The 1.5QFA M' is well-formed.*

Proof: As the automaton M is a well-formed 1.5QFA the identities (3.6) and (3.7) of lemma 3.17 holds for it. We prove these identities for the constructed automaton M', too.

As the first one we prove the equality (3.6) of lemma 3.17. Let $\alpha \in \Gamma'$ and $(q_1,k_1),(q_2,k_2) \in Q'$. Without loss of generality, we can assume that k_1 is not greater then k_2. To cover all instances of the identity it is sufficient to consider only tree cases. Namely if $|h(\alpha)| \le k_2$, if $k_1 = k_2 = |h(\alpha)|-1$, and if $k_1 < |h(\alpha)|-1 \wedge k_2 \le |h(\alpha)|-1$.

- If $|h(\alpha)| \le k_2$, then

$$\begin{aligned}
(3.6) &= \sum_{(p,j)\in Q'} \delta'^*((q_1,k_1),\alpha,(p,j),0)\delta'((q_2,k_2),\alpha,(p,j),0) \ +\\
&+ \sum_{(p,j)\in Q'} \delta'^*((q_1,k_1),\alpha,(p,j),1)\delta'((q_2,k_2),\alpha,(p,j),1) \overset{(4.5)}{=}\\
&= \delta'^*((q_1,k_1),\alpha,(q_2,k_2),0)\delta'((q_2,k_2),\alpha,(q_2,k_2),0) + 0 \overset{(4.4),(4.5)}{=}\\
&= \begin{cases} 1\cdot 1 + 0 = 1 & \text{if } q_1 = q_2 \wedge k_1 = k_2\\ 0\cdot 1 + 0 = 0 & \text{if } q_1 \ne q_2 \vee k_1 \ne k_2. \end{cases}
\end{aligned}$$

- If $k_1 = k_2 = |h(\alpha)|-1$, then

$$\begin{aligned}
(3.6) &= \sum_{(p,j)\in Q'} \delta'^*((q_1,k_1),\alpha,(p,j),0)\delta'((q_2,k_1),\alpha,(p,j),0) \ +\\
&+ \sum_{(p,j)\in Q'} \delta'^*((q_1,k_1),\alpha,(p,j),1)\delta'((q_2,k_1),\alpha,(p,j),1) \overset{(4.5)}{=}\\
&= \sum_{p\in Q} \delta'^*((q_1,k_1),\alpha,(p,k_1),0)\delta'((q_2,k_1),\alpha,(p,k_1),0) \ +\\
&+ \sum_{p\in Q} \delta'^*((q_1,k_1),\alpha,(p,0),1)\delta'((q_2,k_1),\alpha,(p,0),1) \overset{(4.1),(4.3)}{=}\\
&= \sum_{\substack{p\in Q\\ d\in\{0,1\}}} \delta^*(q_1,(h(\alpha))_{k_1},p,d)\delta(q_2,(h(\alpha))_{k_1},p,d) \overset{(3.6)\text{ for } M}{=} \begin{cases} 1 & \text{if } q_1 = q_2\\ 0 & \text{if } q_1 \ne q_2. \end{cases}
\end{aligned}$$

- Finally, if $k_1 < |h(\alpha)| - 1$ and $k_2 \leq |h(\alpha)| - 1$, then

$$\begin{aligned}(3.6) &= \sum_{(p,j)\in Q'} \delta'^*((q_1,k_1),\alpha,(p,j),0)\delta'((q_2,k_2),\alpha,(p,j),0) \ + \\ &+ \sum_{(p,j)\in Q'} \delta'^*((q_1,k_1),\alpha,(p,j),1)\delta'((q_2,k_2),\alpha,(p,j),1) \overset{(4.5)}{=} \\ &= \sum_{p\in Q} \delta'^*((q_1,k_1),\alpha,(p,k_1+1),0)\delta'((q_2,k_2),\alpha,(p,k_1+1),0) \ + \\ &+ \sum_{p\in Q} \delta'^*((q_1,k_1),\alpha,(p,k_1),0)\delta'((q_2,k_2),\alpha,(p,k_1),0) \ = \ \heartsuit.\end{aligned}$$

  To state the value of the previous expression, we need to consider the following three relations between k_1 and k_2:

  - If $k_1 = k_2$, then

$$\heartsuit \overset{(4.2)}{=} \sum_{p\in Q} \delta^*(q_1,(h(\alpha))_{k_1},p,1)\delta(q_2,(h(\alpha))_{k_1},p,1) \ + \ \overset{(4.1)}{+} \sum_{p\in Q} \delta^*(q_1,(h(\alpha))_{k_1},p,0)\delta(q_2,(h(\alpha))_{k_1},p,0) \overset{(3.6)\text{ for } M}{=} \begin{cases} 1 & \text{if } q_1 = q_2 \\ 0 & \text{if } q_1 \neq q_2. \end{cases}$$

  - If $k_1 + 1 = k_2$, then

$$\heartsuit \overset{(4.2),(4.1),(4.5)}{=} \sum_{p\in Q} \delta^*(q_1,(h(\alpha))_{k_1},p,1)\delta(q_2,(h(\alpha))_{k_1+1},p,0) \overset{(3.7)\text{ for } M}{=} 0.$$

  - And if $k_1 + 2 \leq k_2$, then $\heartsuit \overset{(4.5)}{=} 0 + 0 = 0$.

So far, we have proved the equality (3.6) of the lemma 3.17. We prove the other equality of the lemma. For $\alpha_1, \alpha_2 \in \Gamma'$ and $(q_1,k_1),(q_2,k_2) \in Q'$ we have

$$\begin{aligned}(3.7) &= \sum_{(p,j)\in Q'} \delta'^*((q_1,k_1),\alpha_1,(p,j),1)\delta'((q_2,k_2),\alpha_2,(p,j),0) \overset{(4.5)}{=} \\ &= \sum_{p\in Q} \delta'^*((q_1,k_1),\alpha_1,(p,0),1)\delta'((q_2,k_2),\alpha_2,(p,0),0) \ = \ \diamondsuit.\end{aligned}$$

If $k_1 \neq |h(\alpha_1)|-1$ or $k_2 \neq 0$, then by (4.5) is $\diamondsuit = 0$. In the following we assume that $k_1 = |h(\alpha_1)|-1$ and $k_2 = 0$:

$$\begin{aligned}\diamondsuit &= \sum_{p\in Q} \delta'^*((q_1,|h(\alpha_1)|-1),\alpha_1,(p,0),1)\delta'((q_2,0),\alpha_2,(p,0),0) \overset{(4.3),(4.1)}{=} \\ &= \sum_{p\in Q} \delta^*(q_1,(h(\alpha_1))_{|h(\alpha_1)|-1},p,1)\delta(q_2,(h(\alpha_2))_0,p,0) \overset{(3.7)\text{ for } M}{=} 0.\end{aligned}$$

By proving the equalities of lemma 3.17 we have showed that the constructed automaton M' is well-formed. □

Lemma 4.2 *The 1.5QFA M' recognizes the language $L(M') = h^{-1}(L(M))$.*

Proof: Let $w \in \Sigma'^*$ be an input word of 1.5QFA M' and x the corresponding tape of length n. We show that the evolution of the automaton M' on the tape x mimics the evolution of M on the tape $h(x)$.

Denote $m = |h(x)|$ and $f_j = |h((x)_{0\ldots j-1})|$, for all j with $0 \leq j \leq n$. It is easy to see that

$$(h((x)_j))_l = (h(x))_{f_j+l}, \text{ for } 0 \leq j < n \wedge 0 \leq l < |h((x)_j)|, \text{ and} \tag{4.6}$$

$$f_{(j-1) \bmod n} + |h((x)_{(j-1) \bmod n})| - 1 = (f_j - 1) \bmod m, \text{ for } 0 \leq j < n. \tag{4.7}$$

Write P^t_{acc} (P^t_{rej}, P^t_{non}) the probability that the result of the measurement using the observable $\mathcal{O}^M_m$ in the t-th step of the evolution of M is "accept" ("reject", "non-halting"). Similarly, for the automaton M' and the observable $\mathcal{O}^{M'}_n$, write such probability R^t_{acc} (R^t_{rej}, R^t_{non}). Now, assume

$$|\phi^t\rangle = \sum_{q \in Q,\ j \in \mathbb{Z}_m} a^t_{q,j} |q, j\rangle \tag{4.8}$$

is the superposition of M after t steps of its evolution on the tape $h(x)$. We prove by induction on t that

$$|\psi^t\rangle = \sum_{\substack{(q,k) \in Q' \\ j \in \mathbb{Z}_n}} b^t_{q,k,j} |(q,k), j\rangle, \text{ where } b^t_{q,k,j} = \begin{cases} a^t_{q,f_j+k} & \text{if } k < |h((x)_j)| \\ 0 & \text{if } k \geq |h((x)_j)| \end{cases} \tag{4.9}$$

is the superposition of M' after t steps of its evolution on the tape x. As a consequence we show that $P^t_\heartsuit = R^t_\heartsuit$ for every $t \geq 1$ and $\heartsuit \in \{acc, rej, non\}$.

By definition 3.7, the superposition of the automaton M at the beginning of its evolution is $|\phi^0\rangle = |q_0, 0\rangle$. Similarly, by the same definition, the superposition of M' at the beginning of its evolution is $|\psi^0\rangle = |q'_0, 0\rangle \overset{\text{def. of } M'}{=} |(q_0, 0), 0\rangle$. As $a^0_{q,j} \neq 0$ only for $q = q_0$ and $j = 0$ and $b^0_{q,k,j} \neq 0$ only for $q = q_0$, $k = 0$ and $j = 0$, (4.9) holds for $t = 0$.

In the inductive case, we assume (4.9) holds for t and show it for $t+1$. In the $t+1$-st step of the evolution of M the unitary transition $\mathsf{U}^M_{h(x)}$ is applied on the superposition $|\phi^t\rangle$. This application leads to the superposition

$$\mathsf{U}^M_{h(x)} |\phi^t\rangle \overset{(4.8)}{=} \sum_{\substack{q \in Q \\ j \in \mathbb{Z}_m}} a^t_{q,j} \mathsf{U}^M_{h(x)} |q, j\rangle \overset{(3.1)}{=} \sum_{\substack{q \in Q \\ j \in \mathbb{Z}_m}} a^t_{q,j} \sum_{\substack{p \in Q \\ d \in \{0,1\}}} \delta\left(q, (h(x))_j, p, d\right) |p, (j+d) \bmod m\rangle \ =$$

$$= \sum_{\substack{p \in Q \\ j \in \mathbb{Z}_m}} \sum_{\substack{q \in Q \\ d \in \{0,1\}}} \delta\left(q, (h(x))_{(j-d) \bmod m}, p, d\right) a^t_{q,(j-d) \bmod m} |p, j\rangle = \sum_{\substack{p \in Q \\ j \in \mathbb{Z}_m}} c^t_{p,j} |p, j\rangle. \tag{4.10}$$

Analogically, the superposition after applying the unitary transition $\mathsf{U}^{M'}_x$ in the $t+1$-st step of the evolution of M' is

$$\begin{aligned} \mathsf{U}^{M'}_x |\psi^t\rangle &= \sum_{\substack{(p,l) \in Q' \\ j \in \mathbb{Z}_n}} \sum_{\substack{(q,k) \in Q' \\ d \in \{0,1\}}} \delta'\left((q,k), (x)_{(j-d) \bmod n}, (p,l), d\right) b^t_{q,k,(j-d) \bmod n} |(p,l), j\rangle \ = \\ &= \sum_{\substack{(p,l) \in Q' \\ j \in \mathbb{Z}_n}} d^t_{p,l,j} |(p,l), j\rangle. \end{aligned}$$

Amplitudes $d^t_{p,l,j}$ of particular configurations in the superposition $\mathsf{U}^{M'}_x |\psi^t\rangle$ may be expressed by means of the amplitudes $c^t_{p,j}$ of $\mathsf{U}^M_{h(x)} |\phi^t\rangle$. We have three cases to consider:

- If $0 < l < |h((x)_j)|$, then

$$\begin{aligned} d^t_{p,l,j} &\overset{(4.5)}{=} \sum_{q \in Q} \left(b^t_{q,l,j} \delta'((q,l), (x)_j, (p,l), 0) + b^t_{q,l-1,j} \delta'((q,l-1), (x)_j, (p,l), 0) \right) \ = \\ &\overset{(4.1)}{=} \sum_{q \in Q} \left(b^t_{q,l,j} \delta(q, (h((x)_j))_l, p, 0) \overset{(4.2)}{+} b^t_{q,l-1,j} \delta(q, (h((x)_j))_{l-1}, p, 1) \right) \ = \\ &\overset{\text{IH},(4.6)}{=} \sum_{q \in Q} \left(a^t_{q,f_j+l} \delta(q, (h(x))_{f_j+l}, p, 0) + a^t_{q,f_j+l-1} \delta(q, (h(x))_{f_j+l-1}, p, 1) \right) \overset{(4.10)}{=} c^t_{q,f_j+l}. \end{aligned}$$

- Assume, that $l = 0$. To simplify the expressions let $j' = (j-1) \bmod n$ and $l' = |h((x)_{j'})| - 1$. In this case, we have

$$\begin{aligned}
d^t_{p,l,j} &\overset{(4.5)}{=} \sum_{q\in Q}\Big(b^t_{q,0,j}\delta'((q,0),(x)_j,(p,0),0) + b^t_{q,l',j'}\delta'((q,l'),(x)_{j'},(p,0),1)\Big) = \\
&\overset{(4.1)}{=} \sum_{q\in Q}\Big(b^t_{q,0,j}\delta(q,(h((x)_j))_0,p,0) \overset{(4.3)}{+} b^t_{q,l',j'}\delta(q,(h((x)_{j'}))_{l'},p,1)\Big) = \\
&\overset{\text{IH},(4.6)}{=} \sum_{q\in Q}\Big(a^t_{q,f_j}\delta(q,(h(x))_{f_j},p,0) + a^t_{q,f_{j'}+l'}\delta(q,(h(x))_{f_{j'}+l'},p,1)\Big) = \\
&\overset{(4.7)}{=} \sum_{q\in Q}\Big(a^t_{q,f_j}\delta(q,(h(x))_{f_j},p,0) + a^t_{q,(f_j-1) \bmod m}\delta(q,(h(x))_{(f_j-1) \bmod m},p,1)\Big) = \\
&\overset{(4.10)}{=} c^t_{q,f_j}.
\end{aligned}$$

- Finally, if $l \geq |h((x)_j)|$, then $d^t_{p,l,j} \overset{(4.5)}{=} b^t_{p,l,j}\delta'\left((p,l),(x)_j,(p,l),0\right) \overset{\text{IH},(4.4)}{=} 0 \cdot 1 = 0$.

Therefore, the superposition $\mathsf{U}^{M'}_x|\psi^t\rangle$ we can write as

$$\mathsf{U}^{M'}_x|\psi^t\rangle = \sum_{\substack{(p,l)\in Q'\\ j\in\mathbb{Z}_n}} d^t_{p,l,j}|(p,l),j\rangle, \text{ where } d^t_{p,l,j} = \begin{cases} c^t_{p,f_j+l} & \text{if } l < |h((x)_j)| \\ 0 & \text{if } l \geq |h((x)_j)|. \end{cases} \tag{4.11}$$

Now, let $\heartsuit \in \{acc, rej, non\}$ and compare the probabilities of particular results of the measurements in the t-th step of the evolutions of M and M' on the respective tapes:

$$R^t_\heartsuit = \sum_{\substack{(q,k)\in Q'_\heartsuit\\ j\in\mathbb{Z}_n}} \|d^t_{q,k,j}\|^2 \overset{(4.11)}{=} \sum_{\substack{q\in Q_\heartsuit\\ j\in\mathbb{Z}_n}} \sum_{0\leq k<|h((x)_j)|} \|c^t_{q,f_j+k}\|^2 = \sum_{\substack{q\in Q_\heartsuit\\ j\in\mathbb{Z}_m}} \|c^t_{q,j}\|^2 = P^t_\heartsuit. \tag{4.12}$$

Let P^M_{non} ($\mathsf{P}^{M'}_{non}$) be a normalized projection operator projecting the Hilbert space $\mathcal{H}^M_m$ ($\mathcal{H}^{M'}_n$) to its subspace $\mathcal{E}^M_{non}$ ($\mathcal{E}^{M'}_{non}$). This operator represents a transformation done by a measurement using the observable $\mathcal{O}^M_m$ ($\mathcal{O}^{M'}_n$) with the result "non-halting". For the superposition of M after $t+1$ steps of its evolution on the tape $h(x)$ we have:

$$|\phi^{t+1}\rangle = \mathsf{P}^M_{non}\mathsf{U}^M_{h(x)}|\phi^t\rangle \overset{(4.10)}{=} \mathsf{P}^M_{non}\sum_{\substack{q\in Q\\ j\in\mathbb{Z}_m}} c^t_{q,j}|q,j\rangle = \sum_{\substack{q\in Q_{non}\\ j\in\mathbb{Z}_m}} \frac{c^t_{q,j}}{\sqrt{P^t_{non}}}|q,j\rangle.$$

In sequel to previous and (4.8) the amplitudes of the configurations in the superposition $|\phi^{t+1}\rangle$ for $j \in \mathbb{Z}_m$ are

$$a^{t+1}_{q,j} = \begin{cases} \frac{c^t_{q,j}}{\sqrt{P^t_{non}}} & \text{for } q \in Q_{non} \\ 0 & \text{for } q \notin Q_{non}. \end{cases} \tag{4.13}$$

Similarly, the superposition of the automaton M' after $t+1$ steps of its evolution on the tape x is:

$$\begin{aligned}
|\psi^{t+1}\rangle &= \mathsf{P}^{M'}_{non}\mathsf{U}^{M'}_x|\psi^t\rangle \overset{(4.11)}{=} \mathsf{P}^{M'}_{non}\sum_{\substack{(q,k)\in Q'\\ j\in\mathbb{Z}_n}} d^t_{q,k,j}|(q,k),j\rangle = \sum_{\substack{(q,k)\in Q'_{non}\\ j\in\mathbb{Z}_n}} \frac{d^t_{q,k,j}}{\sqrt{R^t_{non}}}|(q,k),j\rangle \overset{(4.11),\ (4.12)}{=} \\
&= \sum_{\substack{q\in Q_{non}\\ j\in\mathbb{Z}_n}} \sum_{0\leq k<|h((x)_j)|} \frac{c^t_{q,f_j+k}}{\sqrt{P^t_{non}}}|(q,k),j\rangle \overset{(4.13)}{=} \sum_{\substack{q\in Q\\ j\in\mathbb{Z}_n}} \sum_{0\leq k<|h((x)_j)|} a^{t+1}_{q,f_j+k}|(q,k),j\rangle.
\end{aligned}$$

Therefore, the amplitudes of the configurations in the superposition $|\phi^{t+1}\rangle$ correspond to (4.9) and the inductive case is proved. For $(q,k) \in Q'$ and $j \in \mathbb{Z}_n$ we have:

$$b^{t+1}_{q,k,j} = \begin{cases} a^{t+1}_{q,f_j+k} & \text{if } k < |h((x)_j)| \\ 0 & \text{if } k \geq |h((x)_j)|. \end{cases}$$

Thus, it is proved that for any given input word w the probabilities $P^t_\heartsuit$ and $R^t_\heartsuit$ are equal for every $t \geq 1$ and $\heartsuit \in \{acc, rej, non\}$. Let $\heartsuit \in \{acc, rej\}$. Since for any input word w the probability $P^{M'}_\heartsuit(w)$ that the evolution of M' halts with the result $\heartsuit$ is

$$P^{M'}_\heartsuit(w) \;=\; \sum_{k\geq 1}\Big(\prod_{1\leq t<k} R^t_{non}\Big)R^k_\heartsuit \;\overset{(4.12)}{=}\; \sum_{k\geq 1}\Big(\prod_{1\leq t<k} P^t_{non}\Big)P^k_\heartsuit \;=\; P^M_\heartsuit(h(w)),$$

the language $L(M')$ recognized by M' is equal to the language $h^{-1}(L(M))$. □

Theorem 4.3 *The class of languages recognized by 1.5QFA is closed under inverse non-erasing homomorphism.*

Proof: According to the previous two lemmas, for every well-formed 1.5QFA recognizing the language L over the alphabet Σ and every non-erasing homomorphism $h\colon \Sigma'^* \to \Sigma^*$, there is a well-formed 1.5QFA recognizing the language $h^{-1}(L)$. □

4.2 Simple 2QFA and inverse non-erasing homomorphism

In this section, we deal with the class of languages recognized by simple 2QFA. We prove that this class is closed under inverse non-erasing homomorphism. Take a well-formed simple 2QFA $M = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$ recognizing the language $L(M)$ and a homomorphism $h\colon \Sigma'^* \to \Sigma^*$, such that $|h(\alpha)| \geq 1$ for all $\alpha \in \Sigma'$. We construct a new simple 2QFA M' recognizing the language $h^{-1}(L(M))$.

The new simple 2QFA works similarly to the 1.5QFA constructed in the previous section. On an input word w it simulates the original automaton on the input word $h(w)$. For each symbol α of the input word, it keeps the superposition of the original automaton heads reading any symbol of the subword $h(\alpha)$ in a superposition of heads reading the symbol α. The different positions of the simulated heads within the word $h(\alpha)$ are distinguished by different states of the heads reading the symbol α.

To implement left head moves correctly, the automaton head must recognize which state it should change to when moving to the left. As homomorphic images of different symbols may have different lengths, the automaton cannot determine the length of the homomorphic image of the symbol, say γ, one square to the left from the currently read symbol. Thus it may not know which states represent the last symbol of $h(\gamma)$. To allow left head moves, the new automaton uses a special set of states representing last symbols of homomorphic images. We distinguish these states by $\bullet$. All other symbols of homomorphic images are represented as previously.

Let $\Gamma' = \Sigma' \cup \{¢, \$\}$, extend the non-erasing homomorphism h to $\Gamma'^* \to \Gamma^*$ such that $h(¢) = ¢$ and $h(\$) = \$$, and let $s = \max\{|h(\alpha)| \mid \alpha \in \Gamma'\}$. Furthermore, for every $\alpha \in \Gamma'$ define function g_α as follows:

$$g_\alpha(k) = \begin{cases} \bullet & \text{if } k = -1 \\ k & \text{if } 0 \leq k < |h(\alpha)| \\ 0 & \text{if } k = |h(\alpha)|. \end{cases}$$

According to definition 3.12 of simple 2QFA and lemma 3.13, there is for every $\alpha \in \Gamma$ a unitary linear operator V_α on the Hilbert space $\ell_2(Q)$ and a function $D\colon Q \to \{-1, 0, 1\}$ such that (3.5) holds for the transformation function δ of M. We define the new simple 2QFA as $M' = (Q', \Sigma', \delta', q'_0, Q'_{acc}, Q'_{rej})$, where $Q' = Q \times (\mathbb{Z}_s \cup \{\bullet\})$, $Q'_{acc} = Q_{acc} \times (\mathbb{Z}_s \cup \{\bullet\})$ and $Q'_{rej} = Q_{rej} \times (\mathbb{Z}_s \cup \{\bullet\})$. If $D(q_0) = -1$, we declare $q'_0 = (q_0, \bullet)$, otherwise $q'_0 = (q_0, 0)$. We define

Figure 4.2: States of the new simple 2QFA distinguished by the function D.

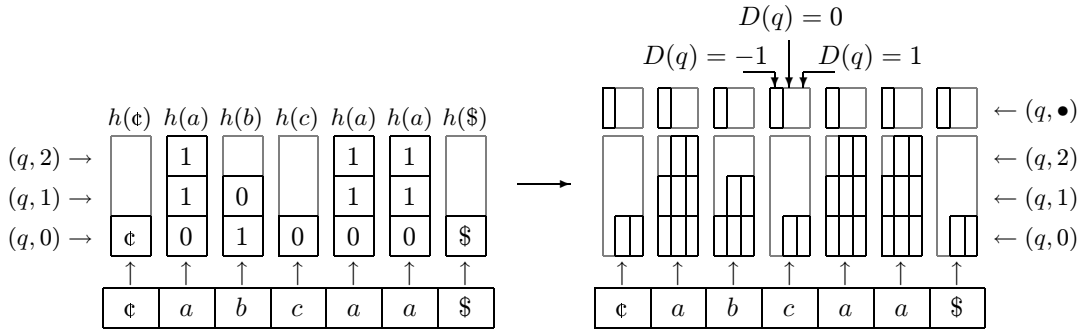
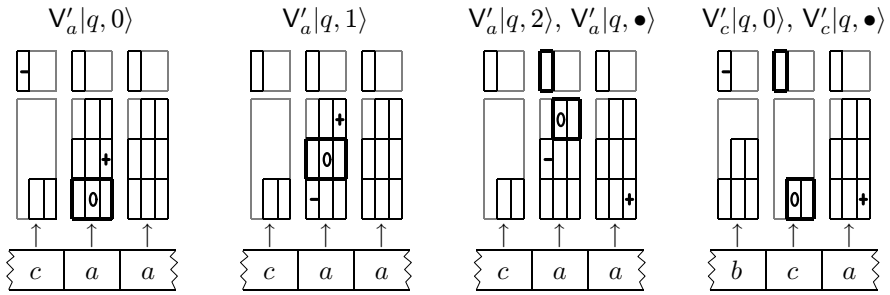


Figure 4.3: Transitions of the constructed simple 2QFA M'.



the transition function δ' of M' with help of operators V'_α, where $\alpha \in \Gamma'$, and function D' as in definition 3.12. For clarity, we omit the parentheses in vectors $|(q,k)\rangle \in \ell_2(Q')$ and write just $|q,k\rangle$, instead.

For every $\alpha \in \Gamma'$ define the operator V'_α on the Hilbert space $\ell_2(Q')$ as follows: for each $k \in \mathbb{Z}_s$ and $q \in Q$, such that $0 \leq k < |h(\alpha)| - 1$ or $k = |h(\alpha)| - 1 \wedge D(q) \neq -1$, let

$$\mathsf{V}'_\alpha |q,k\rangle = \sum_{p \in Q} \langle p| \mathsf{V}_{(h(\alpha))_k} |q\rangle |p, g_\alpha(k + D(p))\rangle, \tag{4.14}$$

for each $q \in Q$, such that $D(q) = -1$, let

$$\mathsf{V}'_\alpha |q,\bullet\rangle = \sum_{p \in Q} \langle p| \mathsf{V}_{(h(\alpha))_k} |q\rangle |p, g_\alpha(k + D(p))\rangle, \text{ where } k = |h(\alpha)| - 1, \tag{4.15}$$

and for all other $q \in Q$ and $k \in \mathbb{Z}_s \cup \{\bullet\}$ let

$$\mathsf{V}'_\alpha |q,k\rangle = |q,k\rangle. \tag{4.16}$$

Furthermore, define function $D' \colon Q' \to \{-1, 0, 1\}$, such that for each $(q,k) \in Q'$ we have

$$\begin{aligned} D'(q,k) &= -1, \text{ if } D(q) = -1 \wedge k = \bullet, \\ D'(q,k) &= 1, \text{ if } D(q) = 1 \wedge k = 0, \text{ and} \\ D'(q,k) &= 0, \text{ otherwise.} \end{aligned}$$

Lemma 4.4 *The simple 2QFA M' is well-formed.*

Proof: According to lemma 3.13, the simple 2QFA M' is well-formed if and only if all V'_α are unitary. Take a symbol α from Γ'. We show that vectors $\mathsf{V}'_\alpha|q,k\rangle$, where $(q,k) \in Q'$, are orthonormal.

Let $A = \{(q,k) \in Q' \mid 0 \leq k < |h(\alpha)| - 1\}$, $B = \{(q,k) \in Q' \mid k = |h(\alpha)| - 1 \wedge D(q) \neq -1\}$, $C = \{(q,\bullet) \in Q' \mid D(q) = -1\}$, and $E = Q' \setminus (A \cup B \cup C)$. For each $(q,k) \in A \cup B$ we have $\|\mathsf{V}'_\alpha|q,k\rangle\|^2 = \sum_{p \in Q} \|\langle p|\,\mathsf{V}_{(h(\alpha))_k}|q\rangle\|^2 = 1$ by (4.14) and unitarity of $\mathsf{V}_{(h(\alpha))_k}$. Similarly, we have $\|\mathsf{V}'_\alpha|q,\bullet\rangle\| = 1$ for each $(q,\bullet) \in C$. For all $(q,k) \in E$ we have $\|\mathsf{V}'_\alpha|q,k\rangle\| = 1$ directly from (4.16). Hence, all vectors $\mathsf{V}'_\alpha|q,k\rangle$ have norm 1.

Let $(q,k) \in A \cup B \cup C$ and $(p,j) \in E$. We see that $\langle p,j|\,\mathsf{V}'_\alpha|q,k\rangle = 0$, by inspection. Therefore, the vector $\mathsf{V}'_\alpha|p,j\rangle$ is orthogonal to the vector $\mathsf{V}'_\alpha|q,k\rangle$. So are vectors $\mathsf{V}'_\alpha|q,k\rangle$, where $(q,k) \in E$, to each other. Now, let $(p,k),(q,k) \in A$ with p and q different. By unitarity of $\mathsf{V}_{(h(\alpha))_k}$ we have $\sum_{r \in Q} \left(\langle r|\,\mathsf{V}_{(h(\alpha))_k}|q\rangle\right)^* \left(\langle r|\,\mathsf{V}_{(h(\alpha))_k}|p\rangle\right) = 0$, and hence $\mathsf{V}'_\alpha|p,k\rangle \perp \mathsf{V}'_\alpha|q,k\rangle$. Similarly, $\mathsf{V}'_\alpha|p,j\rangle \perp \mathsf{V}'_\alpha|q,k\rangle$ for $(p,j),(q,k) \in B \cup C$ with p and q different. For every $(p,j),(q,k) \in A \cup B \cup C$ with j and k different, there is no base vector $|r,l\rangle$ with $(r,l) \in Q$, such that both $\langle r,l|\,\mathsf{V}'_\alpha|p,j\rangle$ and $\langle r,l|\,\mathsf{V}'_\alpha|q,k\rangle$ are non-zero. Therefore, the vectors $\mathsf{V}'_\alpha|p,j\rangle$ and $\mathsf{V}'_\alpha|q,k\rangle$ are orthogonal.

We have showed that all vectors $\mathsf{V}'_\alpha|q,k\rangle$, where $(q,k) \in Q'$, are orthogonal and have norm 1. Hence, all V'_α are unitary and M' is well-formed. □

Lemma 4.5 *The simple 2QFA M' recognizes the language $h^{-1}(L(M))$.*

Proof: Let $w \in \Sigma'^*$ be an input word of 2QFA M' and x the corresponding tape of length n. By a similar argument as in the proof of lemma 4.2, we can show that the evolution of the automaton M' on the tape x mimics the evolution of M on the tape $h(x)$.

Denote $m = |h(x)|$ and $f_j = |h((x)_{0\ldots j-1})|$, for all j with $0 \leq j \leq n$, and let sets A, B, C and E be defined as in the proof of the previous lemma. Now, assume that

$$|\phi^t\rangle = \sum_{q \in Q,\ j \in \mathbb{Z}_m} a^t_{q,j}|q,j\rangle$$

is the superposition of M (on the Hilbert space $\mathcal{H}^M_m$) after t steps of its evolution on the tape $h(x)$. Similarly as in lemma 4.2, we can prove by induction on t that

$$|\psi^t\rangle = \sum_{\substack{(q,k) \in Q' \\ j \in \mathbb{Z}_n}} b^t_{q,k,j}|(q,k),j\rangle, \text{ where } b^t_{q,k,j} = \begin{cases} a^t_{q,f_j+k} & \text{if } (q,k) \in A \cup B \\ a^t_{q,f_j+|h((x)_j)|-1} & \text{if } (q,k) \in C \\ 0 & \text{if } (q,k) \in E \end{cases}$$

is the superposition of M' (on the Hilbert space $\mathcal{H}^{M'}_n$) after t steps of its evolution on the tape x.

As a consequence, the probability that the result of the measurement using the observable $\mathcal{O}^M_m$ in the t-th step of the evolution of M is "accept" ("reject", "non-halting") is equal to the probability that the result of the measurement using the observable $\mathcal{O}^{M'}_n$ in the t-th step of the evolution of M' is "accept" ("reject", "non-halting"). Therefore, the probabilities of accepting (rejecting) the word w by M' and the word $h(w)$ by M are equal and the language recognized by M' is equal to the language $h^{-1}(L(M))$. □

Theorem 4.6 *The class of languages recognized by simple 2QFA is closed under inverse non-erasing homomorphism.*

Proof: According to the previous two lemmas, for every well-formed simple 2QFA recognizing the language L over the alphabet Σ and every non-erasing homomorphism $h: \Sigma'^* \to \Sigma^*$, there is a well-formed simple 2QFA recognizing the language $h^{-1}(L)$. □

Note that if the automaton M is simple 1.5QFA, so is the automaton M'. Therefore, the class of languages recognized by simple 1.5QFA is closed under inverse non-erasing homomorphism, too.

4.3 Simple 1.5QFA and general inverse homomorphism

In the previous section we have shown that the class of languages recognized by simple 2QFA is closed under inverse non-erasing homomorphism. Now, we prove a stronger result for simple 1.5QFA, namely we prove that the class of languages recognized by simple 1.5QFA is closed under general inverse homomorphism.

Definition 4.7 *We call a homomorphism* $h\colon \Sigma_1^* \to \Sigma_2^*$ *length-nonincreasing if* $|h(\alpha)| \leq 1$ *for all* $\alpha \in \Sigma_1$.

Foremost, we show that the class of languages recognized by simple 1.5QFA is closed under inverse length-nonincreasing homomorphism, and then we generalize the result for any inverse homomorphism. Take a well-formed simple 1.5QFA $M = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$ recognizing the language $L(M)$ and a homomorphism $h\colon \Sigma'^* \to \Sigma^*$, such that $|h(\alpha)| \leq 1$ for all $\alpha \in \Sigma'$. We construct a new simple 2QFA M' recognizing the language $h^{-1}(L(M))$.

The new automaton ran on the input word w simulates the automaton M on the word $h(w)$. Its evolution on symbols with non-empty homomorphic image is the same as the evolution of M. On symbols with homomorphic image equal to ε, the automaton moves all its heads immediately to the right. If α is an input symbol with k preceding symbols mapped onto ε, the evolution on this symbol is delayed by k evolution steps.

Let $\Gamma' = \Sigma' \cup \{¢, \$\}$ and extend the homomorphism h to $\Gamma'^* \to \Gamma^*$ such that $h(¢) = ¢$ and $h(\$) = \$$. For every $\alpha \in \Gamma$ there is a unitary linear operator V_α on the Hilbert space $\ell_2(Q)$ and a function $D\colon Q \to \{-1, 0, 1\}$, such that (3.5) holds for δ. We define the new simple 2QFA as $M' = (Q, \Sigma', \delta', q_0, Q_{acc}, Q_{rej})$, where δ' is the transition function defined bellow. For each $\alpha \in \Gamma'$ with $|h(\alpha)| = 1$ let operator $\mathsf{V}'_\alpha = \mathsf{V}_{h(\alpha)}$, and for each $\alpha \in \Gamma'$ with $|h(\alpha)| = 0$ let operator $\mathsf{V}'_\alpha = \mathsf{I}$, where I is the identity operator. With help of the operators V'_α and the function D let δ' be defined as in (3.5). As all $\mathsf{V}_{h(\alpha)}$ are unitary, the operators V'_α are unitary, too, and the automaton M' is well-formed.

Lemma 4.8 *The simple 1.5QFA* M' *recognizes the language* $h^{-1}(L(M))$.

Proof: Let $w \in \Sigma'^*$ be an input word of 2QFA M' and x the corresponding tape of length n. By a similar argument as in the proof of lemma 4.2, we can show that the evolution of the automaton M' on the tape x mimics the evolution of M on the tape $h(x)$.

Denote $m = |h(x)|$, and $e_j = j - |h((x)_{0\ldots j-1})|$, for all j with $0 \leq j \leq n$. The number e_j is the number of symbols from the prefix $(x)_{0\ldots j-1}$ mapped onto ε by the homomorphism h. Assume that

$$|\phi^t\rangle = \sum_{q \in Q,\ j \in \mathbb{Z}_m} a^t_{q,j} |q, j\rangle$$

is the superposition of M (on the Hilbert space $\mathcal{H}_m^M$) after t steps of its evolution on the tape $h(x)$. Similarly as in lemma 4.2, we can prove by induction on t that

$$|\psi^t\rangle = \sum_{q \in Q,\ j \in \mathbb{Z}_n} b^t_{q,j} |q, j\rangle, \text{ where } b^t_{q,j} = \begin{cases} a^{t-e_j}_{q,j-e_j} & \text{if } e_j \leq t \wedge h((x)_j) \neq \varepsilon \\ a^{t-e_j}_{q,j-e_j} & \text{if } e_j \leq t \wedge h((x)_j) = \varepsilon \wedge D(q) = 1 \\ 0 & \text{if } e_j > t \text{ or } h((x)_j) = \varepsilon \wedge D(q) \neq 1 \end{cases}$$

is the superposition of M' (on the Hilbert space $\mathcal{H}_n^{M'}$) after t steps of its evolution on the tape x.

As a consequence, we can prove that the probabilities of accepting (rejecting) the word w by M' and the word $h(w)$ by M are equal and the language recognized by M' is equal to the language $h^{-1}(L(M))$. □

Lemma 4.9 *The class of languages recognized by simple 1.5QFA is closed under inverse length-nonincreasing homomorphism.*

Proof: According to lemma 4.8, for every well-formed simple 1.5QFA recognizing the language L over the alphabet Σ and every length-nonincreasing homomorphism $h\colon \Sigma'^* \to \Sigma^*$, there is a well-formed simple 1.5QFA recognizing the language $h^{-1}(L)$. □

Having proved the closure under inverse non-erasing and inverse length-nonincreasing homomorphism, we can prove the closure under general inverse homomorphism.

Lemma 4.10 *For every general homomorphism $h\colon \Sigma_1^* \to \Sigma_2^*$ there is a non-erasing homomorphism h_1 and a length-nonincreasing homomorphism h_2, such that $h^{-1}(L) = h_1^{-1}(h_2^{-1}(L))$ for every language $L \subseteq \Sigma_2^*$.*

Proof: Without loss of generality, assume that the symbol e does not belong to Σ_2. The lemma holds for non-erasing homomorphism $h_1\colon \Sigma_1^* \to (\Sigma_2 \cup \{e\})^*$ and length-nonincreasing homomorphism $h_2\colon (\Sigma_2 \cup \{e\})^* \to \Sigma_2^*$ defined as follows:

$$\begin{array}{ll} h_1(\alpha) = h(\alpha), & \text{if } h(\alpha) \neq \varepsilon, \\ h_1(\alpha) = e, & \text{if } h(\alpha) = \varepsilon, \end{array} \quad \text{and} \quad \begin{array}{ll} h_2(\alpha) = \alpha, & \text{for } \alpha \neq e, \\ h_2(e) = \varepsilon. & \end{array}$$

□

Theorem 4.11 *The class of languages recognized by simple 1.5QFA is closed under general inverse homomorphism.*

Proof: According to the previous lemma, for every homomorphism h there is a non-erasing homomorphism h_1 and a length-nonincreasing homomorphism h_2, such that $h^{-1}(L) = h_1^{-1}(h_2^{-1}(L))$ for every language L.

For every well-formed simple 1.5QFA recognizing the language L there is, according to lemma 4.9, a well-formed simple 1.5QFA recognizing the language $h_2^{-1}(L)$. For this 1.5QFA there is, according to the paragraph after theorem 4.6, a well-formed simple 1.5QFA recognizing the language $h_1^{-1}(h_2^{-1}(L))$. □

Chapter 5

Homomorphic closure of 1.5QFA

In this chapter we investigate the class of languages defined by homomorphic closure of the class of languages recognized by 1.5QFA. We prove that $\mathcal{H}(\mathcal{L}_{1.5QFA})$ is equal to the class of recursively enumerated languages $\mathcal{L}_{RE}$. As a consequence of the proof we show that the class of languages recognized by 1.5QFA is not closed under homomorphism. As 1.5QFA are a special case of 2QFA, the same also holds for the class of languages recognized by 2QFA.

The class of recursively enumerated languages is defined as a class of languages accepted by Turing machines, see [HU90]. In the proof we construct for every Turing machine an 1.5QFA and a homomorphism, such that the homomorphic image of the language recognized by the constructed 1.5QFA is equal to the language accepted by the Turing machine. We construct the 1.5QFA such that it recognizes the set of somehow encoded accepting computations of the Turing machine.

Before we proceed on construction of the 1.5QFA we introduce the encoding we use to encode computations of the Turing machine.

5.1 Encoding input words

Turing machines work with infinite tapes of symbols. For a particular Turing machine the tape contains symbols from a finite set of tape symbols, say Γ. We call this set a tape alphabet. The tape alphabet contains one special symbol to mark untouched squares of the tape. We call this symbol *blank* and write it as $\textsc{b}$.

At the beginning of the computation of the Turing machine, its tape contains a finite input word written on it. All other squares of the tape are blank. The input word may be empty or may contain any symbols from the input alphabet, say Σ. The input alphabet is limited to the tape alphabet and may not contain the blank symbol (i.e., $\textsc{b} \notin \Sigma$). In any moment during the computation, only finitely many symbols of the tape are non-blank. Therefore, in every step of the computation, the tape may be represented by a finite word over the tape alphabet.

In this and the following sections we encode automata tapes extensively. To encode a tape, we encode the word written on the tape. To define the encoding of words over the tape alphabet Γ we take an enumeration function $\overline{\cdot}\colon \Gamma \to \mathbb{Z}_{|\Gamma|}$ of symbols from Γ with $\overline{\textsc{b}} = 0$ and extend it to $\overline{\cdot}\colon \Gamma^* \to \mathbb{N}$ such that the code of a word $w \in \Gamma^*$ is the number $\overline{w}$, where

$$\begin{aligned} \overline{\varepsilon} &= 0 \\ \overline{va} &= |\Gamma| \cdot \overline{v} + \overline{a} \end{aligned}$$

for $v \in \Gamma^*$ and $a \in \Gamma$. Note that $\overline{\textsc{b}w} = \overline{w}$ for every word w and the code of a word w is equal to the code of the word w with trimmed all left blanks. This property of the encoding function lines up words representing the same tape together.

Having defined the encoding function of words over Γ we are prepared to introduce the encoding of input words of the Turing machine. We use this encoding in construction of the desired 1.5QFA. We design the encoding such that

1. it is possible for an 1.5QFA to check if a given word is a valid code of an input word,
2. it is possible for an 1.5QFA to recognize for a word and a (unary encoded) number, if the word is a code of the same input word as the word written on a tape encoded by the number,
3. the input word may be picked up from its code by a homomorphism.

To accomplish the third requirement we design the encoding such that the code of a word contains the encoded word as a subsequence. Further, to accomplish the second requirement, the code of an input word contains a code of the tape with the input word written on as a subword. Together with the code of this tape, the code of the input word contains for every prefix of the input word a code of the tape with this prefix written on as a subword. The codes (numbers) of these tapes are unary encoded and concatenated one by one separated from each other by last symbols of the corresponding prefixes.

To make the encoding clear in detail we provide its formal definition. For an input alphabet $\Sigma \subseteq \Gamma$, with $l \notin \Sigma$, we say that the code of a word $w \in \Sigma^*$ is the word $I(w)$, where

$$\begin{aligned} I(\varepsilon) &= \varepsilon \\ I(va) &= I(v)al^{\overline{va}} \end{aligned} \tag{5.1}$$

for $v \in \Sigma^*$ and $a \in \Sigma$. For instance the code of the word abc is $al^{\overline{a}}bl^{\overline{ab}}cl^{\overline{abc}}$.

It is easy to see that for homomorphism $h\colon (\{l\} \cup \Sigma)^* \to \Sigma^*$, where $h(l) = \varepsilon$ and $h(\alpha) = \alpha$, for $\alpha \in \Sigma$, we have $h(I(w)) = w$, for every word $w \in \Sigma^*$. Also, as we prove in section 5.3.2 the other two requirements on the encoding are fulfilled, too.

5.2 Encoding configurations

A computation of a Turing machine is a finite sequence of its configurations. Every such configuration is represented by a state of the Turing machine, value of the machine tape and position of the machine head on the tape. The tape symbol on the tape square just under the machine head is called currently read symbol. This symbol splits the tape into two parts – the left tape part of symbols to the left and the right tape part of symbols to the right from the currently read symbol. As the tape contains only finitely many non-blank symbols, we may look on both its parts as on finite words over the tape alphabet. The first symbols of these two words are the symbols one square next to the currently read symbol.

A configuration of the Turing machine we encode as a word containing information about the machine state and the currently read symbol and the codes of both parts of the tape. The first symbol of this word encodes the state and the currently read symbol. We call this symbol a head of the configuration code. The rest of the configuration code encodes its tape. For practical reasons in construction of the desired 1.5QFA we actually encode in a configuration code head besides the actual state and the read symbol also the states and the read symbols of the last two configurations of the computation prior to the encoded configuration. If there are no prior configurations to the encoded configuration we mark this fact in the code of this configuration, too.

Formally, for a Turing machine $(K, \Sigma, \Gamma, \delta, q_0, F)$ we define a code of its configuration as a word of the form $C\{l, r\}^*$, where C is a set of all possible code heads. There are three kinds of code heads we need to consider, 1) code heads of initial configurations with no preceding configurations, 2) code heads of configurations with only one preceding configuration, and 3) code heads of remaining configurations. We denote the sets of these three kinds of heads by C_1, C_2 and C_3, respectively. Obviously the set of all possible code heads C is a union of these three sets. We define these sets as follows:

$$C_1 = \left\{ \begin{bmatrix} \bullet \\ \bullet \\ q_0 & \text{B} \end{bmatrix} \right\},\ C_2 = \left\{ \begin{bmatrix} q_0 & \bullet & \text{B} \\ q_1 & a_1 \end{bmatrix} \;\middle|\; q_1 \in K \wedge a_1 \in \Gamma \wedge \delta_1(q_0, \text{B}) = q_1 \right\}, \text{ and}$$

$$C_3 = \left\{ \begin{bmatrix} q_1 & a_1 \\ q_2 & a_2 \\ q_3 & a_3 \end{bmatrix} \;\middle|\; q_1, q_2 \in K \setminus F \wedge q_3 \in K \wedge a_1, a_2, a_3 \in \Gamma \wedge \delta_1(q_1, a_1) = q_2 \wedge \delta_1(q_2, a_2) = q_3 \right\},$$

where the function $\delta_1 \colon K \times \Gamma \to K$ projects the respective part of the transition function of the Turing machine $\delta \colon K \times \Gamma \to K \times \Gamma \times \{-1, 0, 1\}$. Note, that the state q_0 in the previous definition is the initial state of the Turing machine, while the other three states are arbitrary states of the Turing machine. The heads of configurations codes obey the state part (δ_1) of the transition function because they represent real computation steps which obey the transition function.

Also note, that we assume that a computation of the Turing machine starts in the initial state reading a blank symbol from the square one square to the right from the rightmost symbol of the input word and continues until it reaches an accepting state. We assume the computation halts upon reaching an accepting state for the first time and thus, there are no accepting states during the computation prior to the last computation configuration. Classical definitions of Turing machines, as in [HU90], define the initial configuration of the Turing machine slightly differently. They assume the computation starts with the machine head reading the first symbol of an input word. However, it is easy to see that this difference is not significant and the class of languages accepted by the machines of these two definitions are the same.

Finally, take a configuration reached during a computation of the Turing machine. If the configuration has state $q \in Q$, currently read symbol $a \in \Gamma$ and left and right parts of the machine tape equal to words $w_l \in \Gamma^*$ and $w_r \in \Gamma^*$, respectively, its code is the word

$$\begin{bmatrix} ? \\ q\ a \end{bmatrix} w, \text{ with } w \in \{l, r\}^* \ \wedge \ \#_l(w) = \overline{w_l} \ \wedge \ \#_r(w) = \overline{w_r},$$

where '?' represents the first two lines of the code head which depend on the two preceding configurations during the computation as described in definition of the set C.

5.3 Constructing the automaton

For a Turing machine we construct an 1.5QFA recognizing the set of all encoded accepting computations of the Turing machine. The computations we encode using the encoding of input words defined in section 5.1 and the encoding of machine configurations defined in section 5.2.

We encode the computation of the Turing machine on an input word as the code of the input word followed by codes of all computation configurations. In the code of the last (accepting) configuration we omit the tape code as its value is irrelevant to the computation. The resulting code is a concatenation of these codes.

Take a Turing machine $T = (K^T, \Sigma^T, \Gamma^T, \delta^T, q_0^T, F^T)$, such that $q_0^T \notin F^T$, T implements only move-left and move-right steps (i.e. $\delta^T(q, \alpha) \neq (p, \beta, 0)$, for all $p, q \in K^T$ and $\alpha, \beta \in \Gamma^T$), and the computation of T begins on the right side of the input word. We can make this assumption on the Turing machine without loss of generality, because it is easy to see that for every Turing machine there is an equivalent Turing machine meeting these conditions.

Before formally defining the encoding we use to encode computations of the Turing machine we introduce few auxiliary definitions that help us hereinafter to simplify our notations. For the transition function of the Turing machine we define three functions $\delta_1^T \colon K^T \times \Gamma^T \to K^T$, $\delta_2^T \colon K^T \times \Gamma^T \to \Gamma^T$ and $\delta_3^T \colon K^T \times \Gamma^T \to \{-1, 0, 1\}$ projecting respective parts of the transition function. We call them the state part of the transition function, the writing part of the transition function and the motion part of the transition function, respectively. Furthermore, let $B = |\Gamma^T|$ be the number of the tape symbols, define an enumeration function $\bar{\cdot}$ of these symbols meeting the conditions from section 5.1 and extend it to all words over the tape alphabet in sense of that section, and define the encoding function I as is defined in that section. Finally, let C, C_1, C_2 and C_3 be defined as in section 5.2 for the Turing machine T.

The set of codes of all accepting computations of the Turing machine T is the following language over the alphabet $\{l, r\} \cup \Sigma^T \cup C$:

$$L = \left\{ I(v) \overbrace{\begin{bmatrix} \vdots \\ q_0 & a_0 \end{bmatrix}}^{\gamma_0 \in C_1} w_0 \overbrace{\begin{bmatrix} \bullet & a_0 \\ q_1 & a_1 \end{bmatrix}}^{\gamma_1 \in C_2} \left(\prod_{j=2}^{k} w_{j-1} \overbrace{\begin{bmatrix} q_{j-2} & a_{j-2} \\ q_{j-1} & a_{j-1} \\ q_j & a_j \end{bmatrix}}^{\gamma_j \in C_3} \right) \;\middle|\; \right. \tag{5.2}$$

(i) $k > 0 \;\wedge\; q_k \in F^T \;\wedge\; v \in \Sigma^{T*} \;\wedge$
$\Big(\forall j < k\colon w_j \in \{l, r\}^* \;\wedge\; c_j = \delta_2^T(q_j, a_j) \;\wedge\; l_j = \#_l(w_j) \;\wedge\; r_j = \#_r(w_j)\Big) \;\wedge$

(ii) $l_0 = \overline{v} \;\wedge\; r_0 = 0 \;\wedge$

(iii) $\Big(\forall j < k-1\colon \Big(\delta_3^T(q_j, a_j) = 1 \quad \to B \cdot l_j + \overline{c_j} = l_{j+1} \;\wedge\; r_j = B \cdot r_{j+1} + \overline{a_{j+1}}\Big) \;\wedge$
$\Big(\delta_3^T(q_j, a_j) = -1 \to l_j = B \cdot l_{j+1} + \overline{a_{j+1}} \;\wedge\; B \cdot r_j + \overline{c_j} = r_{j+1}\Big)\Big)\Big\}.$

Note, that according to the definitions of C_1, C_2 and C_3, for every word in L we have $q_0 = q_0^T$, $a_0 = \text{B}$ and for every $j < k$ is $q_j \notin F^T$ and $q_{j+1} = \delta_1^T(q_j, a_j)$.

Every code of a computation of the Turing machine from this language begins with code $I(v)$ of the input word v. The code of the input word is followed by codes of all computation configurations. The codes of the first two configurations have the forms $\begin{bmatrix} \vdots \\ q_0^T & \text{B} \end{bmatrix} w_0$ and $\begin{bmatrix} q_0^T & \bullet & \text{B} \\ q_1 & a_1 \end{bmatrix} w_1$, respectively, as they do not have enough preceding configurations, while all others configurations have the form $\begin{bmatrix} q_{j-2} & a_{j-2} \\ q_{j-1} & a_{j-1} \\ q_j & a_j \end{bmatrix} w_j$. Note, that every computation has at least two configurations as the machine initial state does not belong to the set of accepting states. As we already mentioned, we omit the tape code in the code of the last (accepting) configuration $\begin{bmatrix} q_{k-2} & a_{k-2} \\ q_{k-1} & a_{k-1} \\ q_k & a_k \end{bmatrix}$. The computation begins with the input word written on the tape, hence the code of the tape in the first configuration must encode such a tape. This is ensured by the second condition (5.2-ii) in the definition of the language L.

The third condition (5.2-iii) in the previous definition guarantees that the tape of each computation whose code belongs to L obeys the writing and the motion parts of the transition function of the Turing machine. In other words, that the tapes of any two consecutive configurations differ only in the symbol written by the machine head and the head's new position. The left part of the tape is prolonged by the currently written symbol and the first symbol of the right tape part is taken off if the machine head moves right. Conversely, if the machine head moves left, the first symbol is taken off the left tape part and the symbol written by the head is pushed to the right tape part.

After a brief inspection we can see that the language L represents exactly the set of all accepting computations of the Turing machine T and there is a homomorphism h whose image of this language is equal to the language accepted by the Turing machine, i.e., $h(L) = L(T)$. Namely, the equality holds for the homomorphism $h\colon (\{l, r\} \cup \Sigma^T \cup C)^* \to \Sigma^{T*}$, where

$$h(\alpha) = \begin{cases} \alpha & \text{for } \alpha \in \Sigma^T \\ \varepsilon & \text{for } \alpha \notin \Sigma^T. \end{cases} \tag{5.3}$$

Therefore, the following lemma holds:

Lemma 5.1 *For the homomorphism h mentioned in the previous paragraph we have $h(L) = L(T)$.*

We construct the 1.5QFA recognizing the language L of the codes of all accepting computations of the Turing machine by incorporating seven subautomata. Each of these seven subautomata recognizes a subset of the properties of the codes from the language L. The resulting automaton recognizes the intersection of the languages recognized by the subautomata in some manner.

The first of these subautomata recognizes codes of computations with a proper structure which satisfy all local conditions of a correct accepting computation. Every such computation must begin in the initial and end in an accepting configuration of the Turing machine. Also, every step of the computation must be performed according to the state part of the machine transition function. However, this subautomaton does not take care of the machine tape and does not check if tape

codes change according to the transition function. Furthermore, the subautomaton checks if the first two lines of configurations codes heads correspond to states and currently read symbols of the two preceding configurations. The language recognized by this subautomaton is actually the language L eased to only the first condition (5.2-i) in the definition of L with taking no care of the input word code.

The next two subautomata check if computations codes begin with a proper input word code. They check for a computation code if the prefix terminated by the code head of the first configuration is a valid code of an input word as defined in section 5.1. Applying the condition (5.2-ii) they also ensure that the tape code of the first computation configuration is a code of a tape containing the input word. The subautomata check these conditions in such a way that the first of them checks the equality (5.1) for even symbols and the other one for odd symbols of the input word. We check the equality (5.1) by a method similar to the method to recognize the language $\{a^i b^i \mid i \geq 0\}$ on 2QFA introduced in [KW97].

The remaining subautomata verify that tape codes of all configurations of a computation change according to the transition function of the Turing machine. These subautomata check for every computation code from the language recognized by the first subautomaton if the condition (5.2-iii) holds. Two of these subautomata check behavior of the left tape part while the other two subautomata behavior of the right tape part. The equalities of (5.2-iii) are tested by a method similar to the method used in previous two subautomata checking the structure of the input word code.

In following sections we primarily construct the auxiliary subautomata and then finally in section 5.4 we construct the desired 1.5QFA.

5.3.1 Subautomaton checking local conditions

As the first component of the desired 1.5QFA we construct an automaton recognizing a language of codes of all computations of the Turing machine that satisfy local conditions of a correct accepting computation. This is the language of codes of all accepting computations eased to only the first condition (5.2-i) with taking no care of the input word code. We can write this language as follows:

$$L_1 = \left\{ v \overbrace{\begin{bmatrix} \vdots \\ \vdots \\ q_0 \ a_0 \end{bmatrix}}^{\gamma_0 \in C_1} w_0 \overbrace{\begin{bmatrix} \bullet \ a_0 \\ q_1 \ a_1 \end{bmatrix}}^{\gamma_1 \in C_2} \left(\prod_{j=2}^{k} w_{j-1} \overbrace{\begin{bmatrix} q_{j-2} \ a_{j-2} \\ q_{j-1} \ a_{j-1} \\ q_j \ a_j \end{bmatrix}}^{\gamma_j \in C_3} \right) \;\middle|\; \begin{array}{l} v \in \left(\Sigma^T \cup \{l\}\right)^* \wedge \ k > 0 \wedge \\ \wedge \ q_k \in F^T \ \wedge \ (\forall j < k \colon w_j \in \{l, r\}^*) \end{array} \right\}.$$

Do not forget, as already mentioned, that according to the definitions of C_1, C_2 and C_3, for every word in L_1 we have $q_0 = q_0^T$, $a_0 = \text{B}$ and for every $j < k$ is $q_j \notin F^T$ and $q_{j+1} = \delta_1^T(q_j, a_j)$. Also note, that each word in L_1 contains exactly one symbol from C_1 and exactly one symbol from C_2.

The task of the automaton we are about to construct is to ensure that every word of the language it recognizes

1. begins with prefix $v\gamma_0 w_0 \gamma_1$, where $v \in \left(\Sigma^T \cup \{l\}\right)^*$, $w_0 \in \{l, r\}^*$, $\gamma_0 \in C_1$ and $\gamma_1 \in C_2$,
2. keeps the histories of preceding states and read symbols in heads of configurations codes consistent, and
3. ends with a code head of an accepting configuration (with no subsequent ls or rs).

There is no need to check if there are any codes of accepting configurations before the last one, because the consistency of histories in the second condition together with definition of C guarantees there are no such configurations.

The first condition says that a computation code starts with something that may be an input word code followed by codes of the first two configurations. Heads of these two configurations are from C_1 and C_2, respectively. By "something may be an input word code" we mean it is a word, which is not necessarily an input word code, but is over the alphabet input words codes are. Checking this condition is quite straightforward. The automaton uses states p and q to check

it. It starts in a superposition $|p\rangle$, and skims all symbols from the input words codes alphabet $\{l\} \cup \Sigma^T$ until its head arrives to a symbol from C_1, where it changes its superposition to $|q\rangle$. In this superposition the automaton skims all ls and rs until arriving to a symbol from C_2, where it begins to check the second condition. Whenever the automaton discovers an illegal symbol in the input word it changes its superposition to $|p_R\rangle$ or $|q_R\rangle$ and rejects.

To verify the second condition of a computation code we need to compare the code heads of every two consecutive configurations of the computation. For every two such configurations the last two lines of the code head must be equal to the first two lines of the following code head. To compare such pairs of lines of configurations heads the automaton uses a set of states carrying information about these pairs. On the head of the former of the two consecutive configurations the automaton state changes to a state representing the last two lines of this configuration head and moves toward the head of the latter configuration. If the first two lines of this head do not correspond to the lines carried in the state, the automaton rejects. Otherwise, it continues with the next two consecutive configurations. For details about making this part of the automaton transition function reversible see its formal definition in algorithm 5.1.

To carry pairs of lines from configurations codes heads the automaton uses the following set of states:

$$H = \left\{ \begin{bmatrix} q_1 & a_1 \\ q_2 & a_2 \end{bmatrix} \;\middle|\; q_1 \in K^T \setminus F^T \wedge q_2 \in K^T \wedge a_1, a_2 \in \Gamma^T \wedge \delta_1^T(q_1, a_1) = q_2 \right\}.$$

In addition to these states the automaton uses their accepting and rejecting versions $H_A = \{\alpha_A \mid \alpha \in H\}$ and $H_R = \{\alpha_R \mid \alpha \in H\}$, respectively.

If the carried pair of lines comes from an accepting configuration the state in the second line of the pair belongs to the set of accepting states of the Turing machine. To verify the third condition it suffices to ensure that the automaton accepts only if such a pair of lines is carried over the right end-marker \$. Also, it must be ensured that no such pair of lines is carried over any other symbol. To recognize these pairs of lines we define a subset of H representing them:

$$F = \left\{ \begin{bmatrix} q_1 & a_1 \\ q_2 & a_2 \end{bmatrix} \;\middle|\; q_1 \in K^T \setminus F^T \wedge q_2 \in F^T \wedge a_1, a_2 \in \Gamma^T \wedge \delta_1^T(q_1, a_1) = q_2 \right\}.$$

Before defining the automaton recognizing L_1 we introduce two auxiliary functions $P_1 \colon C_3 \to H$ and $P_2 \colon C_2 \cup C_3 \to H$ projecting the appropriate pair of lines of a configuration code head. P_1 projects the first two lines and P_2 the last two lines of the code head. Namely:

$$P_1\left(\begin{bmatrix} q_1 & a_1 \\ q_2 & a_2 \\ q_3 & a_3 \end{bmatrix}\right) = \begin{bmatrix} q_1 & a_1 \\ q_2 & a_2 \end{bmatrix},\; P_2\left(\begin{bmatrix} \bullet & \\ q_2 & a_2 \\ q_3 & a_3 \end{bmatrix}\right) = \begin{bmatrix} q_2 & a_2 \\ q_3 & a_3 \end{bmatrix}, \text{ and } P_2\left(\begin{bmatrix} q_1 & a_1 \\ q_2 & a_2 \\ q_3 & a_3 \end{bmatrix}\right) = \begin{bmatrix} q_2 & a_2 \\ q_3 & a_3 \end{bmatrix}.$$

Define 1.5QFA M_1 as the sextuple $(Q, \Sigma, \delta, p, Q_{acc}, Q_{rej})$, where $\Sigma = \{l, r\} \cup \Sigma^T \cup C$, $Q = \{p, q, p_R, q_R\} \cup H \cup H_A \cup H_R$, $Q_{acc} = H_A$ and $Q_{rej} = \{p_R, q_R\} \cup H_R$. According to definition 3.1 the tape alphabet of this automaton is equal to $\Gamma = \Sigma \cup \{¢, \$\}$. For each $\alpha \in \Gamma$ let V_α take values as in algorithm 5.1 and arbitrarily extend it to be unitary on the Hilbert space $\ell_2(Q)$. Also define D as in algorithm 5.1, and let δ be defined as in (3.5). Since each V_α is unitary, M_1 is well-formed by lemma 3.13. Note, that this automaton never goes to a superposition of two or more different states nor to a superposition of two or more different head positions. We prove that this automaton recognizes language L_1.

Lemma 5.2 *If $w \in L(M_1)$ then there are $v \in (\Sigma^T \cup \{l\})^*$, $w_0 \in \{l, r\}^*$, $\gamma_0 \in C_1$ and $\gamma_1 \in C_2$ such that $v\gamma_0 w_0 \gamma_1$ is a prefix of w.*

Proof: Take a word $w \in \Sigma^*$, for which there are no $v \in (\Sigma^T \cup \{l\})^*$, $w_0 \in \{l, r\}^*$, $\gamma_0 \in C_1$ and $\gamma_1 \in C_2$ such that $v\gamma_0 w_0 \gamma_1$ is a prefix of w. As the the word w does not have such prefix it may have only one of the following four forms:

1. $w \in (\Sigma^T \cup \{l\})^*$: In this case the automaton's initial superposition $|p\rangle$ does not change while reading word w. Upon arriving on the right end-marker \$ the superposition changes to $|p_R\rangle$ and the automaton rejects. Therefore, $w \notin L(M_1)$.

Algorithm 5.1 Subautomaton M_1 checking local conditions.

$\beta \in C_1$:
$\mathsf{V}_\beta|p\rangle = |q\rangle$,
$\mathsf{V}_\beta|\alpha\rangle = |\alpha_R\rangle, \quad \alpha \in \{q\} \cup H$,

$\beta \in C_2$:
$\mathsf{V}_\beta|q\rangle = |P_2(\beta)\rangle$,
$\mathsf{V}_\beta|\alpha\rangle = |\alpha_R\rangle, \quad \alpha \in \{p\} \cup H$,

$\beta \in C_3 \ \wedge \ P_1(\beta) = P_2(\beta)$:
$\mathsf{V}_\beta|\alpha\rangle = |\alpha_R\rangle, \quad \alpha \in \{p, q\}$,
$\mathsf{V}_\beta|\alpha\rangle = |\alpha_R\rangle, \quad \alpha \in H \setminus \{P_1(\beta)\}$,
$\mathsf{V}_\beta|\alpha\rangle = |P_2(\beta)\rangle, \ \alpha = P_1(\beta)$,
$\mathsf{V}_\beta|\alpha_R\rangle = |\alpha\rangle, \quad \alpha \in H \setminus \{P_1(\beta)\}$,
$\mathsf{V}_\beta|\alpha_R\rangle = |\alpha_R\rangle, \quad \alpha = P_1(\beta)$,

$\beta \in C_3 \ \wedge \ P_1(\beta) \neq P_2(\beta)$:
$\mathsf{V}_\beta|\alpha\rangle = |\alpha_R\rangle, \quad \alpha \in \{p, q\}$,
$\mathsf{V}_\beta|\alpha\rangle = |\alpha_R\rangle, \quad \alpha \in H \setminus \{P_1(\beta)\}$,
$\mathsf{V}_\beta|\alpha\rangle = |P_2(\beta)\rangle, \ \alpha = P_1(\beta)$,
$\mathsf{V}_\beta|\alpha_R\rangle = |\alpha\rangle, \quad \alpha \in H \setminus \{P_1(\beta), P_2(\beta)\}$,
$\mathsf{V}_\beta|\alpha_R\rangle = |\alpha_R\rangle, \quad \alpha = P_1(\beta)$,
$\mathsf{V}_\beta|\alpha_R\rangle = |P_1(\beta)\rangle, \ \alpha = P_2(\beta)$,

$\mathsf{V}_l|\alpha\rangle = |\alpha\rangle, \quad \alpha \in \{p, q\} \cup H \setminus F$,
$\mathsf{V}_l|\alpha\rangle = |\alpha_R\rangle, \ \alpha \in F$,
$\mathsf{V}_r|\alpha\rangle = |\alpha\rangle, \quad \alpha \in \{q\} \cup H \setminus F$,
$\mathsf{V}_r|\alpha\rangle = |\alpha_R\rangle, \ \alpha \in \{p\} \cup F$,

$\sigma \in \Sigma^T$:
$\mathsf{V}_\sigma|p\rangle = |p\rangle$,
$\mathsf{V}_\sigma|\alpha\rangle = |\alpha_R\rangle, \ \alpha \in \{q\} \cup H$,

$\mathsf{V}_¢|p\rangle = |p\rangle$,
$\mathsf{V}_\$|\alpha\rangle = |\alpha_A\rangle, \ \alpha \in F$,
$\mathsf{V}_\$|\alpha\rangle = |\alpha_R\rangle, \ \alpha \in \{p, q\} \cup H \setminus F$,

$D(\alpha) = 0, \quad \alpha \in Q_{acc} \cup Q_{rej}$,
$D(\alpha) = 1, \quad \alpha \in Q \setminus Q_{acc} \setminus Q_{rej}$.

2. w has prefix $u \in (\Sigma^T \cup \{l\})^*(\{r\} \cup C_2 \cup C_3)$: Again, while reading symbols from $\Sigma^T \cup \{l\}$ the automaton remains in superposition $|p\rangle$ and upon arriving on a symbol from $\{r\} \cup C_2 \cup C_3$ changes to superposition $|p_R\rangle$ and rejects. Therefore, $w \notin L(M_1)$.

3. $w \in (\Sigma^T \cup \{l\})^* C_1 \{l, r\}^*$: After reading a prefix of the form $(\Sigma^T \cup \{l\})^* C_1$ the automaton superposition changes to $|q\rangle$. Then while reading ls and rs the superposition does not change and upon reading the right end-marker \$ it changes to $|q_R\rangle$. Therefore, the automaton rejects and $w \notin L(M_1)$ again.

4. w has prefix $u \in (\Sigma^T \cup \{l\})^* C_1 \{l, r\}^* (\Sigma^T \cup C_1 \cup C_3)$: After reading all symbols of the prefix u except the last one the automaton superposition is $|q\rangle$. Reading its last symbol (a symbol from $\Sigma^T \cup C_1 \cup C_3$) the automaton changes its superposition to $|q_R\rangle$ and rejects. Again, we have $w \notin L(M_1)$.

Hence, every word accepted by the automaton M_1 has the required prefix. □

Lemma 5.3 *Let* $w = v \overbrace{\begin{bmatrix} \vdots \\ q_0 \ a_0 \end{bmatrix}}^{\gamma_0 \in C_1} w_0 \overbrace{\begin{bmatrix} \bullet \\ q_0 \ a_0 \\ q_1 \ a_1 \end{bmatrix}}^{\gamma_1 \in C_2} \left(\prod_{j=2}^{k} w_{j-1} \overbrace{\begin{bmatrix} q_{j-2} \ a_{j-2} \\ q_{j-1} \ a_{j-1} \\ q_j \quad a_j \end{bmatrix}}^{\gamma_j \in C_3} \right)$, *where* $k \geq 1$, $v \in (\Sigma^T \cup \{l\})^*$, *and* $w_j \in \{l, r\}^*$ *for all* $j < k$. *After reading prefix* w *of a word with such prefix, the automaton* M_1 *is in superposition* $|P_2(\gamma_k)\rangle = \left| \begin{bmatrix} q_{k-1} \ a_{k-1} \\ q_k \quad a_k \end{bmatrix} \right\rangle$.

Proof: We prove this lemma by induction on k. For $k = 1$ the word $w = v\gamma_0 w_0 \gamma_1$, where $v \in (\Sigma^T \cup \{l\})^*$, $w_0 \in \{l, r\}$, $\gamma_0 \in C_1$ and $\gamma_1 \in C_2$. While reading v the automaton does not change its initial superposition $|p\rangle$. Reading γ_0 changes its superposition to $|q\rangle$. This superposition does not change until reading γ_1, when it changes to $|P_2(\gamma_1)\rangle = |P_2(\gamma_k)\rangle$.

Now, assume the lemma holds for $k = n \geq 1$, we prove it holds for $k = n + 1$, too. For $k = n + 1$ we have $w = v\gamma_0 w_0 \gamma_1 w_1 \ldots \gamma_n w_n \gamma_{n+1}$. We know that the Turing machine state q_n in the third line of γ_n is equal to the state in the second line of γ_{n+1}. As $\gamma_{n+1} \in C_3$ the state

q_n is not an accepting state of the Turing machine T. By induction hypothesis the automaton is in superposition $|P_2(\gamma_n)\rangle$ after reading prefix $u = v\gamma_0 w_0 \gamma_1 w_1 \dots \gamma_n$. As $q_n \notin F^T$, the state $P_2(\gamma_n)$ belongs to $H \setminus F$ and therefore, the automaton does not change its superposition $|P_2(\gamma_n)\rangle$ while reading the word $w_n \in \{l, r\}^*$. Upon reading symbol γ_{n+1} the superposition changes to $|P_2(\gamma_{n+1})\rangle = |P_2(\gamma_k)\rangle$ as $P_2(\gamma_n) = P_1(\gamma_{n+1})$. □

Lemma 5.4 *Let $w \in \Sigma^*$. If $w \in L_1$ then M_1 accepts w with probability 1 and if $w \notin L_1$ then M_1 rejects w with probability 1.*

Proof: If $w \in L_1$ we know, by lemma 5.3, that the automaton after reading word w is in superposition $|P_2(\gamma_k)\rangle$, where γ_k is the last configuration code head. Since the last configuration of an accepting computation is accepting, $P_2(\gamma_k) \in F$ and reading the right end-marker \$ the superposition changes to $|P_2(\gamma_k)_A\rangle$ and the automaton accepts with probability 1.

Now, assume that $w \notin L_1$ and M_1 accepts w. By lemma 5.2, the word w has prefix $v\gamma_0 w_0 \gamma_1$, where v, w_0 and γs are in the sense of L_1 definition. By lemma 5.3, the superposition of the automaton after reading this prefix is $|P_2(\gamma_1)\rangle$. If the rest of w contains any symbol from $\Sigma^T \cup C_1 \cup C_2$ the superposition changes after the first such symbol to $|\alpha_R\rangle$ for some $\alpha \in H$ and the automaton rejects for sure. Therefore, we have a contradiction with the assumption that the automaton accepts. (Note, that the automaton may have rejected earlier because of some other reason, but may not have accepted.) Hence, the rest of w contains only symbols from $\{l, r\} \cup C_3$ and w has the form of $v\gamma_0 w_0 \gamma_1 w_1 \gamma_2 w_2 \dots w_{k-1} \gamma_k u$ for some $k \geq 1$, where u and all ws are from $\{l, r\}^*$, $\gamma_0 \in C_1$, $\gamma_1 \in C_2$ and $\gamma_j \in C_3$ for $j \geq 2$. There are only tree kinds of reasons why the word w does not belong to the language L_1:

1. There are two consecutive configurations encoded in w with inconsistent states and/or read symbols information in their codes heads. Take the first such inconsistency. I.e., take the smallest j with $1 \leq j < k$ such that $P_2(\gamma_j) \neq P_1(\gamma_{j+1})$. According to lemma 5.3, after reading prefix $v\gamma_0 w_0 \gamma_1 \dots \gamma_j$ of w the automaton is in superposition $|P_2(\gamma_j)\rangle$. Reading w_j does not change the superposition, however, reading γ_{j+1} changes the superposition to $|P_2(\gamma_j)_R\rangle$ as $P_2(\gamma_j) \neq P_1(\gamma_{j+1})$. As the result, the automaton rejects for sure and we have a contradiction.

2. All consecutive configurations encoded in w are consistent, but the last configuration is not an accepting one, i.e., $P_2(\gamma_k) \notin F$. By lemma 5.3, after reading prefix $v\gamma_0 w_0 \gamma_1 \dots \gamma_k$ the automaton is in superposition $|P_2(\gamma_k)\rangle$. Reading u does not change the superposition, but reading the right end-marker \$ changes the superposition to $|P_2(\gamma_k)_R\rangle$ as $P_2(\gamma_k) \in H \setminus F$ and the automaton rejects for sure. Therefore, we have a contradiction.

3. All consecutive configurations are consistent, the last configuration is accepting, but there are trailing ls and/or rs after the last configuration code head. In other words $u \neq \varepsilon$. Again, by lemma 5.3, after reading prefix $v\gamma_0 w_0 \gamma_1 \dots \gamma_k$ the automaton is in superposition $|P_2(\gamma_k)\rangle$. As $P_2(\gamma_k) \in F$ reading a symbol l or r changes the superposition to $|P_2(\gamma_k)_R\rangle$ and the automaton rejects with probability 1. Hence, we have a contradiction again.

We have shown that every evolution of M_1 on a word $w \notin L_1$ is rejecting. Therefore, M_1 rejects every $w \notin L_1$ with probability 1. □

Lemma 5.5 *The automaton M_1 halts on every word $w \in \Sigma^*$ after $O(|w|)$ evolution steps.*

Proof: As for every non-halting state q is $D(q) = 1$ the automaton reaches the end of the input word and halts in $|w| + 1$ evolution steps unless it halts sooner. □

5.3.2 Subautomata checking input word code

In this section we construct two components of the final 1.5QFA. These two components recognize two languages whose intersection is a language of computations codes with properly encoded input words.

Algorithm 5.2 1.5QFA recognizing $L' = \{a^x b a^y \mid c_1 + d_1 \cdot x = c_2 + d_2 \cdot y\}$.

$1 \le k \le N$:

$\mathsf{V}_¢|Z_k\rangle = \frac{1}{\sqrt{N}} \sum_{j=1}^{N} \omega^{kj} |p_{j,c_1 \cdot j}\rangle$,
$\mathsf{V}_¢|p_{k,j}\rangle = |p_{k,j-1}\rangle,\ 1 \le j \le c_1 \cdot k$,
$\mathsf{V}_¢|p_{k,0}\rangle = |p_{k,0}\rangle$,
$\mathsf{V}_¢|q_{k,0}\rangle = |q_{k,0}\rangle$,

$\mathsf{V}_\$|p_{k,0}\rangle = |R_k\rangle$,
$\mathsf{V}_\$|q_{k,0}\rangle = \frac{1}{\sqrt{N}} \sum_{j=1}^{N} \omega^{kj} |Z_j\rangle$,

$D(\alpha) = 1,\ \alpha \in \{p_{k,0}, q_{k,0} \mid 1 \le k \le N\}$,
$D(\alpha) = 0,\ \alpha \in Q \setminus \{p_{k,0}, q_{k,0} \mid 1 \le k \le N\}$,

$\mathsf{V}_a|p_{k,0}\rangle = |p_{k,d_1 \cdot k}\rangle$,
$\mathsf{V}_a|p_{k,j}\rangle = |p_{k,j-1}\rangle,\ 1 \le j \le d_1 \cdot k$,
$\mathsf{V}_a|q_{k,0}\rangle = |q_{k,d_2 \cdot (N-k+1)}\rangle$,
$\mathsf{V}_a|q_{k,j}\rangle = |q_{k,j-1}\rangle,\ 1 \le j \le d_2 \cdot (N-k+1)$,

$\mathsf{V}_b|p_{k,0}\rangle = |q_{k,c_2 \cdot (N-k+1)}\rangle$,
$\mathsf{V}_b|q_{k,j}\rangle = |q_{k,j-1}\rangle,\ 1 \le j \le c_2 \cdot (N-k+1)$,
$\mathsf{V}_b|q_{k,0}\rangle = |R_k\rangle$.

Notation 5.6 In this and the following section we mean by notation ω_N for a given positive number N the N-th root of 1, namely $\omega_N = e^{2\pi i/N}$. If there is no ambiguity we omit the symbol N in notation ω_N.

Before constructing the subautomata checking whether a computation code begins with a properly encoded input word we describe a routine we extensively use in the subautomata. Let c_1, d_1, c_2 and d_2 be natural numbers. By a method similar to the method to recognize the language $\{a^i b^i \mid i \ge 0\}$ introduced in [KW97] we construct an 1.5QFA recognizing the language $L' = \{a^x b a^y \mid c_1 + d_1 \cdot x = c_2 + d_2 \cdot y\}$. Actually, for every $\varepsilon > 0$, we construct an 1.5QFA recognizing the language L' with error bounded by ε. The language L' can be seen as the set of all solutions of the equation $c_1 + d_1 \cdot x = c_2 + d_2 \cdot y$ with unknowns x and y. Inspired by this view of the language L' we call this routine the equation routine.

For every $N \ge 3$, define 1.5QFA $M_N = (Q, \Sigma, \delta, Z_N, Q_{acc}, Q_{rej})$, where $\Sigma = \{a, b\}$, $Q = \{Z_k, R_k, p_{k,j}, q_{k,j} \mid 1 \le k \le N \wedge 0 \le j \le \max\{c_1, d_1, c_2, d_2\} \cdot N\}$, $Q_{acc} = \{Z_N\}$ and $Q_{rej} = \{Z_k \mid 1 \le k < N\} \cup \{R_k \mid 1 \le k \le N\}$. The tape alphabet of the automaton is $\Gamma = \Sigma \cup \{¢, \$\}$. For each $\alpha \in \Gamma$ let operator V_α take values as in the algorithm 5.2 and extend it arbitrarily to be unitary on $\ell_2(Q)$. Also define function D as in the algorithm 5.2, and let δ be defined as in (3.5). Since V_α is unitary for each $\alpha \in \Gamma$, the automaton M_N is well-formed.

Lemma 5.7 *Let $N \ge 3$ and $w \in \Sigma^*$. If $w \in L'$ then M_N accepts w with probability 1, and otherwise M_N rejects w with probability at least $1 - 1/N$. The automaton accepts in a superposition of one head in the state Z_N reading the right end-marker.*

Proof: At the beginning of the automaton evolution on a word w, its head branches into a superposition of N heads each with amplitude $1/\sqrt{N}$. The k-th ($1 \le k \le N$) of these heads starts in the state $p_{k,c_1 \cdot k}$ and deterministically moves along the input word toward the right end-marker. This head remains $c_1 \cdot k$ steps of the evolution on the left end-marker ¢ and then moves right. On each following symbol a the head remains stationary for $d_1 \cdot k$ steps until reaching the first symbol b, where it remains stationary for $c_2 \cdot (N - k + 1)$ steps and changes its state from $p_{k,j}$ series to $q_{k,j}$ series. On each latter symbol a the head stays for $d_2 \cdot (N - k + 1)$ steps. If there are no or more than one bs in the input word the head rejects in state R_k and the automaton rejects with probability 1. If there is exactly one b in the input word, the word has the form $a^x b a^y$ and the k-th head requires exactly

$$s_k = (1 + c_1 k) + (1 + d_1 k)x + (1 + c_2(N - k + 1)) + (1 + d_2(N - k + 1))y$$

steps to move from the left to the right end-marker. Under the assumption that $k \ne k'$, we have $s_k = s_{k'}$ if and only if $c_1 + d_1 \cdot x = c_2 + d_2 \cdot y$. Therefore any two different heads in the superposition reach the right end-marker \$ in the same step of the evolution if and only if $c_1 + d_1 \cdot x = c_2 + d_2 \cdot y$.

Upon reaching the right end-marker, the k-th head of the superposition again splits according to the quantum Fourier transformation. If the input word has the form $w = a^x b a^y$, with $c_1 + d_1 \cdot x = c_2 + d_2 \cdot y$, all heads of the automaton superposition reach the right end-marker in the same step

of the evolution. In this case, the superposition of the automaton after performing the quantum Fourier transformation is

$$\frac{1}{N}\sum_{k=1}^{N}\sum_{j=1}^{N}\omega^{kj}|Z_j\rangle = |Z_N\rangle.$$

Hence, the automaton accepts with probability 1.

If the input has the form $w = a^x b a^y$, with $c_1 + d_1 \cdot x \neq c_2 + d_2 \cdot y$, each of the N heads reaches the right end-marker in a different evolution step and so no quantum cancellation between the rejecting states happens. In the step, when the k-th head reaches the right end-marker, the probability that the outcome of the observable $\mathcal{O}^{M_N}$ is "accept" is $1/N^2$. The total probability that the automaton accepts w is $1/N$. □

In this and the next section we use this routine in various modifications as a part of different algorithms. Generally, we use the routine on subwords of longer and more complex words, where the tape symbols of M_N are represented by other symbols. When using the routine, we may fix its coefficients c_1, d_1, c_2 and d_2 and explicitly specify them in the automaton transition function, or let the automaton determine the coefficients on the fly from its input word. If the automaton determines the coefficients on the fly we must ensure it uses the same coefficients in all evolution steps of one routine instance. Sometimes it is worth to use a slightly modified version of the routine, in which the k-th head of the superposition stays for $c_1 \cdot k$ and $c_2 \cdot (N-k+1)$ steps of the evolution on symbols b and \$ instead of on symbols ¢ and b, respectively. Such version is useful especially if the automaton needs to determine the coefficients c_1 and c_2 of the routine on the fly from symbols representing symbols b and \$. As all these modifications are very minor, we usually do not say in descriptions of algorithms which variant of the routine the described algorithm uses. However, this should be clear from formal definitions of automata we use these routines in. Similarly, despite we provided a proof of the previous lemma only for the original version of the routine, it holds for all mentioned modifications of the routine and we use this lemma in any context of an automaton using any of the mentioned modifications of the routine.

Now we construct the two subautomata checking whether computations codes begin with a proper input word code. For every computation code they check if its prefix terminated by the first configuration code head is a valid code of an input word. We discussed in section 5.1 how to encode input words. They also verify if the tape code of the first computation configuration is a code of a tape containing the input word. However, these subautomata do not care of the rest of computations codes. The intersection of the languages recognized by these subautomata is the following language:

$$L_e = \left\{ I(v)\gamma_0 l^{\overline{v}}\gamma_1 u \mid v \in \Sigma^{T*} \wedge \gamma_0 \in C_1 \wedge \gamma_1 \in C_2 \wedge u \in (\{l, r\} \cup \Sigma^T \cup C)^* \right\}.$$

According to the definition of $I(v)$ in section 5.1 the subautomata need to verify a number of equalities to recognize this language. Namely, for a given input word v they need to check if the input word code equals to

$$I(v) = l^{e_0}(v)_0 l^{e_1}(v)_1 l^{e_2} \ldots (v)_{|v|-1} l^{e_{|v|}},$$

$$\text{where } e_0 = 0 \text{ and } e_{j+1} = B \cdot e_j + \overline{(v)_j} \text{ for all } j < |v|. \tag{5.4}$$

In addition to these equalities the subautomata must also check if the number of ls at the end of the input word code is equal to the number of ls between γ_0 and γ_1, i.e., if the tape code of the first configuration equals to $l^{e_{|v|}} = l^{\overline{v}}$.

We construct these subautomata in such a way that the first one checks the equalities $e_{j+1} = B \cdot e_j + \overline{(v)_j}$ for j even and the other one for j odd. We name the subautomata M_e^{even} and M_e^{odd}, respectively. The odd version also verifies if $e_0 = 0$. Which of these subautomata checks if the tape code of the first configuration equals to $l^{e_{|v|}}$ depends on the parity of the input word length. If the length of the input word is even, M_e^{even} checks it, otherwise M_e^{odd} checks it.

Algorithm 5.3 Subautomata checking input word code.

even version: $\mathsf{V}_¢|Z_N\rangle = \frac{1}{\sqrt{N}}\sum_{j=1}^{N}|p_{j,0}\rangle$,
odd version: $\mathsf{V}_¢|Z_N\rangle = \frac{1}{\sqrt{N}}\sum_{j=1}^{N}|q_{j,0}\rangle$,

both versions:

$\sigma \in \Sigma^T \ \wedge\ 1 \le k \le N$:
$\mathsf{V}_\sigma|Z_k\rangle = \frac{1}{\sqrt{N}}\sum_{j=1}^{N}\omega^{kj}|p_{j,0}\rangle$,
$\mathsf{V}_\sigma|p_{k,0}\rangle = |q_{k,\overline{\sigma}\cdot k}\rangle$,
$\mathsf{V}_\sigma|q_{k,j}\rangle = |q_{k,j-1}\rangle,\ 1 \le j \le \overline{\sigma}\cdot k$,
$\mathsf{V}_\sigma|q_{k,0}\rangle = \frac{1}{\sqrt{N}}\sum_{j=1}^{N}\omega^{kj}|Z_j\rangle$,
$\mathsf{V}_\sigma|r_{k,0}\rangle = |R_k\rangle$,

$1 \le k \le N$:
$\mathsf{V}_l|p_{k,0}\rangle = |p_{k,B\cdot k}\rangle$,
$\mathsf{V}_l|p_{k,j}\rangle = |p_{k,j-1}\rangle,\ 1 \le j \le B\cdot k$,
$\mathsf{V}_l|q_{k,0}\rangle = |q_{k,N-k+1}\rangle$,
$\mathsf{V}_l|q_{k,j}\rangle = |q_{k,j-1}\rangle,\ 1 \le j \le N-k+1$,
$\mathsf{V}_l|r_{k,0}\rangle = |r_{k,B\cdot(N-k+1)}\rangle$,
$\mathsf{V}_l|r_{k,j}\rangle = |r_{k,j-1}\rangle,\ 1 \le j \le B\cdot(N-k+1)$,

$D(\alpha) = 1,\ \alpha \in \{p_{k,0}, q_{k,0}, r_{k,0} \mid 1 \le k \le N\}$,
$D(\alpha) = 0,\ \alpha \in Q \setminus \{p_{k,0}, q_{k,0}, r_{k,0} \mid 1 \le k \le N\}$,

$\gamma \in C_1 \ \wedge\ 1 \le k \le N$:
$\mathsf{V}_\gamma|Z_N\rangle = |A\rangle$,
$\mathsf{V}_\gamma|p_{k,0}\rangle = |r_{k,0}\rangle$,
$\mathsf{V}_\gamma|q_{k,0}\rangle = \frac{1}{\sqrt{N}}\sum_{j=1}^{N}\omega^{kj}|Z_j\rangle$,
$\mathsf{V}_\gamma|r_{k,0}\rangle = |R_k\rangle$,

$\gamma \in C_2 \ \wedge\ 1 \le k \le N$:
$\mathsf{V}_\gamma|Z_N\rangle = |A\rangle$,
$\mathsf{V}_\gamma|p_{k,0}\rangle = |P_k\rangle$,
$\mathsf{V}_\gamma|q_{k,0}\rangle = |Q_k\rangle$,
$\mathsf{V}_\gamma|r_{k,0}\rangle = \frac{1}{\sqrt{N}}\sum_{j=1}^{N}\omega^{kj}|Z_j\rangle$,

$\alpha \in \{r, \$\} \cup C_3 \ \wedge\ 1 \le k \le N$:
$\mathsf{V}_\alpha|p_{k,0}\rangle = |P_k\rangle$,
$\mathsf{V}_\alpha|q_{k,0}\rangle = |Q_k\rangle$,
$\mathsf{V}_\alpha|r_{k,0}\rangle = |R_k\rangle$.

Both subautomata check the equalities by sequentially running the routine to solve equations presented in the beginning of this section. The coefficients c_1, d_1, c_2 and d_2 of the routine are chosen according to the equality the automaton is going to check. For instance, checking the equality $e_{j+1} = B\cdot e_j + \overline{(v)_j}$ the coefficients would be $c_1 = \overline{(v)_j}$, $d_1 = B$, $c_2 = 0$ and $d_2 = 1$. Note, that the automaton can determine the coefficient c_1 from symbol the $(v)_j$ on the fly, while checking the particular equality. If the routine rejects while checking an equality the automaton rejects, too. However, if the routine accepts the automaton synchronizes its superposition to just one head in state Z_N and continues on the next equality by running another instance of the routine instead of just accepting. The automaton accepts after checking all equalities it needs to check.

At the beginning of the evolution the automaton checks equalities considering only the input word code. It checks only these equalities until it arrives on the first occurrence of a symbol from C_1 in the input word – the symbol γ_0. Upon moving over the symbol γ_0 during execution of the routine the automaton knows it is in the middle of checking the equality considering the first configuration tape code. The coefficients for this equality are $c_1 = 0$, $d_1 = B$, $c_2 = 0$, $d_2 = B$. Note, that only one of the automata happens to check this equality.

To provide a formal definition, for each $N \ge 3$ we define $M_{e,N}^{\text{even}} = (Q, \Sigma, \delta^{\text{even}}, Z_N, Q_{acc}, Q_{rej})$ and $M_{e,N}^{\text{odd}} = (Q, \Sigma, \delta^{\text{odd}}, Z_N, Q_{acc}, Q_{rej})$, where $Q = \{A\} \cup \{Z_k, P_k, Q_k, R_k, p_{k,j}, q_{k,j}, r_{k,j} \mid 1 \le k \le N \wedge 0 \le j \le B\cdot N\}$, $\Sigma = \{l, r\} \cup \Sigma^T \cup C$, $Q_{acc} = \{A\}$ and $Q_{rej} = \{Z_k \mid 1 \le k < N\} \cup \{P_k, Q_k, Q_k \mid 1 \le k \le N\}$. The tape alphabets of the defined automata are $\Gamma = \Sigma \cup \{¢, \$\}$. For each $\alpha \in \Gamma$ let V_α take values as in the even version of algorithm 5.3 and arbitrarily extend it to be unitary on $\ell_2(Q)$. Also define D as in algorithm 5.3, and let δ^{even} be defined as in (3.5). Now, similarly let values of V_α be as in the odd version of algorithm 5.3 and arbitrarily extend it to be unitary on $\ell_2(Q)$. Define D the same way as before and let δ^{odd} be defined as in (3.5), again. Since V_α is unitary for each $\alpha \in \Gamma$ in both cases, the constructed automata $M_{e,N}^{\text{even}}$ and $M_{e,N}^{\text{odd}}$ are well-formed.

Lemma 5.8 *Let $N \ge 3$ and $w = \prod_{j=0}^{k-1} l^{e_{2j}} a_{2j} l^{e_{2j+1}} a_{2j+1}$, where $k \ge 0$, $a_{2k-1} \in \Sigma^T \cup C_1$ and $a_j \in \Sigma^T$ for all $j < 2k-1$, with $e_{2j+1} = B\cdot e_{2j} + \overline{a_{2j}}$ for all $j < k$. Running the automaton $M_{e,N}^{\text{even}}$*

on an input word with prefix w eventually leads to a superposition of one head in state Z_N reading the last symbol of w, or if the word $w = \varepsilon$ reading the left end-marker ¢.

Proof: We prove the lemma by induction on k. If $k = 0$, the word w is empty and the initial superposition of the automaton leads on one step to a superposition that has the required form. If $k = 1$, the input word has prefix $w = l^{e_0} a_0 l^{e_1} a_1$. The superposition after the first evolution step of the automaton consists of one head in state Z_N reading the left end-marker ¢. By lemma 5.7, running one instance of the equation routine discussed in the beginning of this section with symbols a, b and \$ exchanged by symbols l, a_0 and a_1, respectively, for coefficients $c_1 = \overline{a_0}$, $d_1 = B$, $c_2 = 0$ and $d_2 = 1$, leads on the prefix $l^{e_0} a_0 l^{e_1} a_1$ of the input word with $e_1 = B \cdot e_0 + \overline{a_0}$ to a superposition with one head in state Z_N reading the symbol a_1.

Now, assume the lemma holds for all prefixes w with $k = n \geq 1$, we prove it for a prefix w with $k = n + 1$, too. The prefix w of an input word has the form $w = \prod_{j=0}^{n} l^{e_{2j}} a_{2j} l^{e_{2j+1}} a_{2j+1}$ with all $a_j \in \Sigma^T$, where $j < 2n + 1$. As it also has prefix $\prod_{j=0}^{n-1} l^{e_{2j}} a_{2j} l^{e_{2j+1}} a_{2j+1}$, running the automaton $M_{e,N}^{\text{even}}$ on the input word, by induction hypothesis, eventually leads to a superposition of one head in state Z_N reading the symbol a_{2n-1}. Again, by lemma 5.7, running an instance of the equation routine from this superposition with symbols ¢, a, b and \$ exchanged by symbols a_{2n-1}, l, a_{2n} and a_{2n+1}, respectively, for the same coefficients as previously, leads on the subword $a_{2n-1} l^{e_{2n}} a_{2n} l^{e_{2n+1}} a_{2n+1}$ with $e_{2n+1} = B \cdot e_{2n} + \overline{a_{2n}}$ to a superposition with one head in state Z_N reading the symbol a_{2n+1}, which is the last symbol of w. □

Lemma 5.9 *Let $N \geq 3$ and $w = l^{e_0} a_0 \prod_{j=0}^{k-1} l^{e_{2j+1}} a_{2j+1} l^{e_{2j+2}} a_{2j+2}$, where $k \geq 0$, $e_0 = 0$, $a_{2k} \in \Sigma^T \cup C_1$ and $a_j \in \Sigma^T$ for all $j < 2k$, with $e_{2j+2} = B \cdot e_{2j+1} + \overline{a_{2j+1}}$ for all $j < k$. Running the automaton $M_{e,N}^{\text{odd}}$ on an input word with prefix w eventually leads to a superposition of one head in state Z_N reading the last symbol of w.*

Proof: This lemma we prove by induction on k, too. If $k = 0$, we have $w = a_0$. The automaton $M_{e,N}^{\text{odd}}$ on an input word with prefix a_0 performs a slight variation of the equation routine. It splits the head to a superposition of N heads on the left end-marker ¢ and instantly, on the next symbol, it merges the heads back into a one head in state Z_N reading the symbol a_0.

The inductive case of this lemma can be proved in the same way as in lemma 5.8. □

Lemma 5.10 *Let $N \geq 3$ and $w \in \Sigma^*$. If $w \in L_e$ then both $M_{e,N}^{\text{even}}$ and $M_{e,N}^{\text{odd}}$ accept w with probability 1 and if $w \notin L_e$ then either $M_{e,N}^{\text{even}}$ or $M_{e,N}^{\text{odd}}$ rejects w with probability at least $1 - 1/N$.*

Proof: If $w \in L_e$, we have $w = \left(\prod_{j=0}^{k-1} l^{e_j} a_j \right) l^{e_k} \gamma_0 l^{e_k} \gamma_1 u$, with $k \geq 0$. Without loss of generality we assume that k is even. The proof for k odd is similar. As k is even, by lemma 5.9, running the automaton $M_{e,N}^{\text{odd}}$ on the word w leads to a superposition of one head in state Z_N reading the symbol γ_0. In the next evolution step the only head of the automaton superposition changes from state Z_N to state A and the automaton accepts with probability 1. Similarly, by lemma 5.8, running the automaton $M_{e,N}^{\text{even}}$ on the word w leads to a superposition of one head in state Z_N reading the symbol a_{k-1} if $k > 0$, or reading the left end-marker ¢ if $k = 0$. In either case, by lemma 5.7, running an instance of the equation routine with symbols ¢, a, b and \$ exchanged by symbols a_{k-1} (or ¢ if $k = 0$), l, γ_0 and γ_1, respectively, for coefficients $c_1 = c_2 = 0$ and $d_1 = d_2 = B$, leads on the subword $l^{e_k} \gamma_0 l^{e_k} \gamma_1$ to a superposition with one head in state Z_N reading the symbol γ_1. In the next evolution step the only automaton head state changes from Z_N to A and the automaton accepts for sure.

Now assume the input word $w \notin L_e$. If the input word does not have the form of $(\{l\} \cup \Sigma^T)^* C_1 \{l\}^* C_2 \Sigma^*$, both automata reject the input word for sure upon moving their heads over the first unexpected symbol in w. (Under the assumption the automaton does not reject earlier for any other reason. Note, that the automaton may not have accepted the input word earlier.) They reject the word in a superposition of N heads in one of these three series of states: P_k, Q_k or R_k. If the input word has the correct form, we can write $w = \left(\prod_{j=0}^{k-1} l^{e_j} a_j \right) l^{e_k} \gamma_0 l^{e_k} \gamma_1 u$, for some $k \geq 0$,

$a_j \in \Sigma^T$, $\gamma_0 \in C_1$, $\gamma_1 \in C_2$ and $u \in \Sigma^*$. There are only three reasons that may cause the input word w not to belong to the language L_e:

1. $e_0 > 0$: The first instance of the equation routine during the evolution of the automaton $M_{e,N}^{\text{odd}}$ on the input word w verifies if $e_0 = 0$. It is a slightly modified version of the routine, instead of comparing numbers of symbols in two bunches of ls, the modified version compares the number of symbols in just one such bunch of ls with a constant given explicitly by the transition function. In this case the constant is 0. By a similar proof as to lemma 5.7, we can show, that if $e_0 > 0$ the automaton $M_{e,N}^{\text{odd}}$ rejects the input word w with probability at least $1 - 1/N$.

2. $e_0 = 0$, but at least one of the equalities $e_{j+1} = B \cdot e_j + \overline{a_j}$ does not hold: Take the first of the equalities that does not hold, i.e., take a minimal j for which $e_{j+1} \neq B \cdot e_j + \overline{a_j}$. Assume j is even. By lemma 5.8, running the automaton $M_{e,N}^{\text{even}}$ on the word w leads to a superposition of one head in state Z_N reading the symbol a_{j-1} if $j > 0$, or reading the left end-marker if $j = 0$. By lemma 5.7, running the equation routine on the subword $l^{e_j} a_j l^{e_{j+1}} \alpha$, where $\alpha = a_{j+1}$ if $j < k - 1$, or $\alpha = \gamma_1$ if $j = k - 1$, leads to rejecting the input word by $M_{e,N}^{\text{even}}$ with probability at least $1 - 1/N$. Similarly, if j is odd we can prove with help of lemma 5.9, that the automaton $M_{e,N}^{\text{odd}}$ rejects the word w with probability at least $1 - 1/N$.

3. $e_0 = 0$ and all equalities from the previous case hold, but the number of ls between γs does not correspond to the number of ls between a_{k-1} and γ_1: By a proof similar to the proof from the previous case we can show that at least one of the automata rejects the word w with probability at least $1 - 1/N$. Which of the automata happens to reject the word depends on the parity of k.

Therefore, every word that does not belong to L_e is rejected by at least one of the subautomata. □

Lemma 5.11 *Let $N \geq 3$. Both automata $M_{e,N}^{\text{even}}$ and $M_{e,N}^{\text{odd}}$ halt on every word $w \in \Sigma^*$ after $O(|w|)$ evolution steps.*

Proof: We can easily see from algorithm 5.3 that whole amplitude of a head in any state q with $D(q) = 1$ reading any input symbol α accepts, rejects or moves to the next tape square in at most $B \cdot N + 1$ evolution steps. Also, whole amplitude of a head in the initial state reading the left end-marker symbol ¢ moves to the next tape square in one step. To say this formally, $(\mathsf{AV}_\alpha)^{B \cdot N + 1} |q\rangle = 0$ for all $q \in Q$ with $D(q) = 1$ and $\alpha \in \Sigma$, and $\mathsf{AV}_¢ |Z_N\rangle = 0$, where the function $\mathsf{A}: \ell_2(Q) \to \ell_2(Q)$ is defined such that:

$$\mathsf{A}\Big(\sum_{p \in Q} \gamma_p |p\rangle\Big) = \sum_{\substack{p \in Q \setminus Q_{acc} \setminus Q_{rej} \\ D(p) = 0}} \gamma_p |p\rangle$$

Furthermore, any head in a state q with $D(q) = 1$ reading the right end-marker symbol rejects in one evolution step. Formally, $V_\$ |q\rangle \in \mathit{Span}(\{p \mid p \in Q_{rej}\})$ for every $q \in Q$ with $D(q) = 1$. As a consequence of these two properties, both automata on an input word w halt in at most $(B \cdot N + 1) \cdot |w| + 2$ steps. □

5.3.3 Subautomata checking tape code progress

Finally we construct the remaining four subautomata that verify whether tape codes of computations configurations change according to the transition function of the Turing machine. Two of these subautomata check behavior of the left tape part. The other two subautomata check behavior of the right tape part.

First, we present the former two subautomata. For every two consecutive configurations of a computation they check whether the left tape part of the latter configuration may be obtained by

one computation step of the Turing machine from the left tape part of the former configuration. The computation step is determined by the state and the currently read symbol encoded in the code head of the former configuration. Every such step rewrites the currently read symbol by a new symbol and moves the Turing machine head one square right or left. Remember, we chose the Turing machine such that it always moves its head. Moving the machine head one square right extends the left tape part by the newly written symbol. Moving the machine head one square left shortens the the left tape part by the new currently read symbol – the currently read symbol of the latter configuration.

These two subautomata recognize two languages with the following common intersection with the language L_1:

$$\begin{aligned} L_l \;=\; \Bigg\{ & v \overbrace{\begin{bmatrix} \vdots \\ q_0 \;\; a_0 \end{bmatrix}}^{\gamma_0 \in C_1} w_0 \overbrace{\begin{bmatrix} \cdot & \\ q_0 & a_0 \\ q_1 & a_1 \end{bmatrix}}^{\gamma_1 \in C_2} \left(\prod_{j=2}^{k} w_{j-1} \overbrace{\begin{bmatrix} q_{j-2} & a_{j-2} \\ q_{j-1} & a_{j-1} \\ q_j & a_j \end{bmatrix}}^{\gamma_j \in C_3} \right) \;\Bigg|\; k > 0 \;\wedge\; q_k \in F^T \;\wedge\; v \in (\Sigma^T \cup \{l\})^* \;\wedge \\ & \wedge \Big(\forall j < k \colon w_j \in \{l, r\}^* \;\wedge\; c_j = \delta_2^T(q_j, a_j) \;\wedge l_j = \#_l(w_j) \;\wedge\; r_j = \#_r(w_j) \Big) \wedge \\ & \wedge \Big(\forall j < k-1 \colon \Big(\delta_3^T(q_j, a_j) = 1 \quad \to B \cdot l_j + \overline{c_j} = l_{j+1} \Big) \wedge \\ & \qquad \Big(\delta_3^T(q_j, a_j) = -1 \to l_j = B \cdot l_{j+1} + \overline{a_{j+1}} \Big) \Big) \Bigg\}. \end{aligned}$$

Note, that according to the definition of C_1, C_2 and C_3, for every word in L_l we have $q_0 = q_0^T$, $a_0 = \text{B}$, and $q_j \notin F^T$ and $q_{j+1} = \delta_1^T(q_j, a_j)$ for all $j < k$. The intersected language L_1 is the language recognized by the automaton M_1 defined in section 5.3.1.

To recognize if a word from L_1 belongs to the above language the automata need only to verify the following set of equalities:

$$\begin{aligned} B \cdot l_j + \overline{c_j} &= l_{j+1} \qquad \text{for all } j < k-1 \text{ with } \delta_3^T(q_j, a_j) = 1, \text{ and} \\ l_j &= B \cdot l_{j+1} + \overline{a_{j+1}} \qquad \text{for all } j < k-1 \text{ with } \delta_3^T(q_j, a_j) = -1. \end{aligned} \tag{5.5}$$

They verify the equalities using the equation routine from the previous section with symbols ¢, a, b and \$ exchanged by symbols γ_j, l, γ_{j+1} and γ_{j+2}, respectively. The coefficients to the routine are $c_1 = \overline{c_j}$, $d_1 = B$, $c_2 = 0$ and $d_2 = 1$ if $\delta_3^T(q_j, a_j) = 1$, and $c_1 = 0$, $d_1 = 1$, $c_2 = \overline{a_{j+1}}$ and $d_2 = B$ if $\delta_3^T(q_j, a_j) = -1$. The automata can easily determine the coefficients c_1 and c_2 on the fly from symbols γ_j and γ_{j+1}, respectively.

We construct the automata such that the first one checks the equalities for j even and the other one for j odd. We name the automata M_l^{even} and M_l^{odd}, respectively. Both automata sequentially run one instance of the routine for every other two consecutive configurations of the computation until they reach the end of the input word, where they accept. The even version of the automaton begins with comparing the configuration $\gamma_0 w_0$ to $\gamma_1 w_1$. The odd version begins with comparing $\gamma_1 w_1$ to $\gamma_2 w_2$. If any instance of the routine rejects the automaton running this instance rejects, too. However, if an instance of the routine accepts, the automaton instead of just accepting proceeds to another instance of the routine, comparing the next two consecutive configurations. The automaton accepts on the right end-marker after checking all equalities it needs to check.

We define for each $N \geq 3$ the automata $M_{l,N}^{\text{even}} = (Q, \Sigma, \delta^{\text{even}}, s, Q_{acc}, Q_{rej})$ and $M_{l,N}^{\text{odd}} = (Q, \Sigma, \delta^{\text{odd}}, s, Q_{acc}, Q_{rej})$, where $Q = \{s, S, Z_k, P_k^+, P_k^-, Q_k^+, Q_k^-, p_{k,j}^+, p_{k,j}^-, q_{k,j}^+, q_{k,j}^- \mid 1 \leq k \leq N \wedge 0 \leq j \leq B \cdot N\}$, $\Sigma = \{l, r\} \cup \Sigma^T \cup C$, $Q_{acc} = \{P_k^+, P_k^-, Q_k^+, Q_k^- \mid 1 \leq k \leq N\}$ and $Q_{rej} = \{S, Z_k \mid 1 \leq k < N\}$. The tape alphabets of the defined automata are $\Gamma = \Sigma \cup \{¢, \$\}$. For each $\alpha \in \Gamma$ let V_α take values as in the even version of algorithm 5.4. All these operators are unitary on $\ell_2(Q)$, by inspection. Also define D as in algorithm 5.4, and let δ^{even} be defined as in (3.5). Now similarly let values of V_α as in the odd version of algorithm 5.4. All these operators are unitary on $\ell_2(Q)$, too. Define D the same way as previously and let δ^{odd} be defined as in (3.5), again. Since V_α is unitary for each $\alpha \in \Gamma$ in both cases, the constructed automata $M_{l,N}^{\text{even}}$ and $M_{l,N}^{\text{odd}}$ are well-formed.

Algorithm 5.4 Subautomata checking progress of the left tape part

We define the following for all k, where $1 \leq k \leq N$, with operators acting as identities on all remaining base states. In the following we write m as a shortcut for $N-k+1$, c as a shortcut for $\delta_2^T(q_3,a_3)$, d as a shortcut for $\delta_3^T(q_3,a_3)$ and e as a shortcut for $\delta_3^T(q_1,a_1)$.

even version:

$\gamma = \begin{bmatrix} \vdots \\ q_3\ a_3 \end{bmatrix} \in C_1 \ \wedge \ d = 1$:
$\mathsf{V}_\gamma|s\rangle = |Z_N\rangle$,
$\mathsf{V}_\gamma|Z_k\rangle = \frac{1}{\sqrt{N}} \sum_{j=1}^{N} \omega^{kj} |p^+_{j,\overline{c}\cdot j}\rangle$,
$\mathsf{V}_\gamma|p^+_{k,j}\rangle = |p^+_{k,j-1}\rangle,\ 1 \leq j \leq \overline{c} \cdot j$,
$\mathsf{V}_\gamma|p^+_{k,0}\rangle = \frac{1}{\sqrt{N}} \sum_{j=1}^{N-1} \omega^{kj} |Z_j\rangle + \frac{|s\rangle}{\sqrt{N}}$,

$\gamma = \begin{bmatrix} \vdots \\ q_3\ a_3 \end{bmatrix} \in C_1 \ \wedge \ d = -1$:
$\mathsf{V}_\gamma|s\rangle = |Z_N\rangle$,
$\mathsf{V}_\gamma|Z_k\rangle = \frac{1}{\sqrt{N}} \sum_{j=1}^{N} \omega^{kj} |p^-_{j,0}\rangle$,
$\mathsf{V}_\gamma|p^-_{k,0}\rangle = \frac{1}{\sqrt{N}} \sum_{j=1}^{N-1} \omega^{kj} |Z_j\rangle + \frac{|s\rangle}{\sqrt{N}}$,

$\gamma = \begin{bmatrix} q_2\ a_2 \\ q_3\ a_3 \end{bmatrix} \in C_2$:
$\mathsf{V}_\gamma|p^+_{k,0}\rangle = |q^+_{k,0}\rangle$,
$\mathsf{V}_\gamma|q^+_{k,0}\rangle = |p^+_{k,0}\rangle$,
$\mathsf{V}_\gamma|p^-_{k,0}\rangle = |q^-_{k,\overline{a_3}\cdot m}\rangle$,
$\mathsf{V}_\gamma|q^-_{k,j}\rangle = |q^-_{k,j-1}\rangle,\ 1 \leq j \leq \overline{a_3} \cdot m$,
$\mathsf{V}_\gamma|q^-_{k,0}\rangle = |p^-_{k,0}\rangle$,

odd version:

$\gamma = \begin{bmatrix} \bullet\ a_2 \\ q_3\ a_3 \end{bmatrix} \in C_2 \ \wedge \ d = 1$:
$\mathsf{V}_\gamma|s\rangle = |Z_N\rangle$,
$\mathsf{V}_\gamma|Z_k\rangle = \frac{1}{\sqrt{N}} \sum_{j=1}^{N} \omega^{kj} |p^+_{j,\overline{c}\cdot j}\rangle$,
$\mathsf{V}_\gamma|p^+_{k,j}\rangle = |p^+_{k,j-1}\rangle,\ 1 \leq j \leq \overline{c} \cdot j$,
$\mathsf{V}_\gamma|p^+_{k,0}\rangle = \frac{1}{\sqrt{N}} \sum_{j=1}^{N-1} \omega^{kj} |Z_j\rangle + \frac{|s\rangle}{\sqrt{N}}$,

$\gamma = \begin{bmatrix} \bullet\ a_2 \\ q_3\ a_3 \end{bmatrix} \in C_2 \ \wedge \ d = -1$:
$\mathsf{V}_\gamma|s\rangle = |Z_N\rangle$,
$\mathsf{V}_\gamma|Z_k\rangle = \frac{1}{\sqrt{N}} \sum_{j=1}^{N} \omega^{kj} |p^-_{j,0}\rangle$,
$\mathsf{V}_\gamma|p^-_{k,0}\rangle = \frac{1}{\sqrt{N}} \sum_{j=1}^{N-1} \omega^{kj} |Z_j\rangle + \frac{|s\rangle}{\sqrt{N}}$,

Continued on the next page...

Lemma 5.12 *Let $N \geq 3$ and $w = v\gamma_0 w_0 \gamma_1 w_1 \dots w_{k-1}\gamma_k \in L_l$, with $k > 0$, $v \in (\Gamma^T \cup \{l\})^*$, $w_j \in \{l,r\}^*$ for all $j < k$, $\gamma_0 \in C_1$, $\gamma_1 \in C_2$ and $\gamma_j \in C_3$ for all j with $2 \leq j \leq k$. For each m with $0 \leq 2m \leq k$, running the automaton $M^{\text{even}}_{l,N}$ on the input word w eventually leads to a superposition of one head in the state Z_N reading the symbol γ_{2m}.*

Proof: We prove the lemma by induction on m. The automaton starts its evolution in a superposition of one head in state s. It keeps the head in this state while reading the whole prefix v of the input word. Upon arriving on the symbol γ_0 the state of the only superposition head changes to Z_N. Hence, the lemma proposition holds for $m = 0$.

In the inductive case we assume the proposition holds for $m = n$ with $0 \leq n \leq k-2$, we prove it holds for $m = n+1$, too. By induction hypothesis running the automaton on the input word w leads to a superposition of one head in the state Z_N reading the symbol γ_{2n}. The symbol γ_{2n} is the head of a configuration code. If $n = 0$ then $\gamma_{2n} \in C_1$ and $\gamma_{2n} = \begin{bmatrix} \vdots \\ q_3\ a_3 \end{bmatrix}$ for $q_3 \in K^T$ and $a_3 \in \Gamma^T$, or if $n > 0$ then $\gamma_{2n} \in C_3$ and $\gamma_{2n} = \begin{bmatrix} q_1\ a_1 \\ q_2\ a_2 \\ q_3\ a_3 \end{bmatrix}$ for $q_1, q_2, q_3 \in K^T$ and $a_1, a_2, a_3 \in \Gamma^T$. In either case q_3 represents the state and a_3 the currently read symbol of the configuration encoded by $\gamma_{2n} w_{2n}$.

If $\delta_3^T(q_3,a_3) = 1$ the automaton runs an instance of the equation routine with symbols ¢, a, b and \$ exchanged by symbols γ_{2n}, l, γ_{2n+1} and γ_{2n+2}, respectively, for coefficients $c_1 = \overline{\delta_2^T(q_3,a_3)}$, $d_1 = B$, $c_2 = 0$ and $d_2 = 1$. The states used to perform this instance of the routine have the form $p^+_{x,y}$ and $q^+_{x,y}$. Running such an instance of the routine on the subword $\gamma_{2n} w_{2n} \gamma_{2n+1} w_{2n+1} \gamma_{2n+2}$ with $\overline{\delta_2^T(q_3,a_3)} + B \cdot \#_l(w_{2n}) = \#_l(w_{2n+1})$ from a superposition with one head in state Z_N reading the symbol γ_{2n} eventually leads to a superposition with one head in state Z_N reading the symbol $\gamma_{2n+2} = \gamma_{2m}$.

Similarly, if $\delta_3^T(q_3,a_3) = -1$ the automaton runs an instance of the routine with the symbols

Algorithm 5.4 Continued

both versions:

$\gamma = \left[\begin{smallmatrix} q_1 & a_1 \\ q_2 & a_2 \\ q_3 & a_3 \end{smallmatrix}\right] \in C_3 \ \wedge \ d = 1$:

$$
\begin{aligned}
\mathsf{V}_\gamma|Z_k\rangle &= \tfrac{1}{\sqrt{N}} \textstyle\sum_{j=1}^{N} \omega^{kj} |p^+_{j,\overline{c}\cdot j}\rangle, \\
\mathsf{V}_\gamma|p^+_{k,j}\rangle &= |p^+_{k,j-1}\rangle, \ 1 \le j \le \overline{c}\cdot k, \\
\mathsf{V}_\gamma|p^+_{k,0}\rangle &= |q^+_{k,0}\rangle, \\
\mathsf{V}_\gamma|p^-_{k,0}\rangle &= |q^-_{k,\overline{a_3}\cdot m}\rangle, \\
\mathsf{V}_\gamma|q^-_{k,j}\rangle &= |q^-_{k,j-1}\rangle, \ 1 \le j \le \overline{a_3}\cdot m, \\
\left.\begin{aligned} \mathsf{V}_\gamma|q^+_{k,0}\rangle &= \tfrac{1}{\sqrt{N}} \textstyle\sum_{j=1}^{N} \omega^{kj} |Z_j\rangle, \\ \mathsf{V}_\gamma|q^-_{k,0}\rangle &= |p^-_{k,0}\rangle, \end{aligned}\right\} & \ e = 1 \\
\left.\begin{aligned} \mathsf{V}_\gamma|q^+_{k,0}\rangle &= |p^-_{k,0}\rangle, \\ \mathsf{V}_\gamma|q^-_{k,0}\rangle &= \tfrac{1}{\sqrt{N}} \textstyle\sum_{j=1}^{N} \omega^{kj} |Z_j\rangle, \end{aligned}\right\} & \ e = -1
\end{aligned}
$$

$\gamma = \left[\begin{smallmatrix} q_1 & a_1 \\ q_2 & a_2 \\ q_3 & a_3 \end{smallmatrix}\right] \in C_3 \ \wedge \ d = -1$:

$$
\begin{aligned}
\mathsf{V}_\gamma|Z_k\rangle &= \tfrac{1}{\sqrt{N}} \textstyle\sum_{j=1}^{N} \omega^{kj} |p^-_{j,0}\rangle, \\
\mathsf{V}_\gamma|p^+_{k,0}\rangle &= |q^+_{k,0}\rangle, \\
\mathsf{V}_\gamma|p^-_{k,0}\rangle &= |q^-_{k,\overline{a_3}\cdot m}\rangle, \\
\mathsf{V}_\gamma|q^-_{k,j}\rangle &= |q^-_{k,j-1}\rangle, \ 1 \le j \le \overline{a_3}\cdot m, \\
\left.\begin{aligned} \mathsf{V}_\gamma|q^+_{k,0}\rangle &= \tfrac{1}{\sqrt{N}} \textstyle\sum_{j=1}^{N} \omega^{kj} |Z_j\rangle, \\ \mathsf{V}_\gamma|q^-_{k,0}\rangle &= |p^+_{k,0}\rangle, \end{aligned}\right\} & \ e = 1 \\
\left.\begin{aligned} \mathsf{V}_\gamma|q^+_{k,0}\rangle &= |p^+_{k,0}\rangle, \\ \mathsf{V}_\gamma|q^-_{k,0}\rangle &= \tfrac{1}{\sqrt{N}} \textstyle\sum_{j=1}^{N} \omega^{kj} |Z_j\rangle, \end{aligned}\right\} & \ e = -1
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{V}_l|p^+_{k,0}\rangle &= |p^+_{k,B\cdot k}\rangle, \\
\mathsf{V}_l|p^+_{k,j}\rangle &= |p^+_{k,j-1}\rangle, \ 1 \le j \le B\cdot k, \\
\mathsf{V}_l|p^-_{k,0}\rangle &= |p^-_{k,k}\rangle, \\
\mathsf{V}_l|p^-_{k,j}\rangle &= |p^-_{k,j-1}\rangle, \ 1 \le j \le k, \\
\mathsf{V}_l|q^+_{k,0}\rangle &= |q^+_{k,N-k+1}\rangle, \\
\mathsf{V}_l|q^+_{k,j}\rangle &= |q^+_{k,j-1}\rangle, \ 1 \le j \le m, \\
\mathsf{V}_l|q^-_{k,0}\rangle &= |q^-_{k,B\cdot(N-k+1)}\rangle, \\
\mathsf{V}_l|q^-_{k,j}\rangle &= |q^-_{k,j-1}\rangle, \ 1 \le j \le B\cdot m,
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{V}_\$|p^+_{k,0}\rangle &= |P^+_k\rangle, \quad \mathsf{V}_\$|P^+_k\rangle = |p^+_{k,0}\rangle, \\
\mathsf{V}_\$|p^-_{k,0}\rangle &= |P^-_k\rangle, \quad \mathsf{V}_\$|P^-_k\rangle = |p^-_{k,0}\rangle, \\
\mathsf{V}_\$|q^+_{k,0}\rangle &= |Q^+_k\rangle, \quad \mathsf{V}_\$|Q^+_k\rangle = |q^+_{k,0}\rangle, \\
\mathsf{V}_\$|q^-_{k,0}\rangle &= |Q^-_k\rangle, \quad \mathsf{V}_\$|Q^-_k\rangle = |q^-_{k,0}\rangle, \\
\mathsf{V}_\$|s\rangle &= |S\rangle, \quad \mathsf{V}_\$|S\rangle = |s\rangle,
\end{aligned}
$$

$$
\begin{aligned}
&D(\alpha) = 1, \ \alpha \in \{s, (p^+_{k,0}), (p^-_{k,0}), (q^+_{k,0}), (q^-_{k,0}) \mid 1 \le k \le N\} \\
&D(\alpha) = 0, \ \alpha \in Q \setminus \{s, (p^+_{k,0}), (p^-_{k,0}), (q^+_{k,0}), (q^-_{k,0}) \mid 1 \le k \le N\}
\end{aligned}
$$

exchanged in the same way as previously, for coefficients $c_1 = 0$, $d_1 = 1$, $c_2 = \overline{a_4}$ and $d_2 = B$, where a_4 is the currently read symbol of the configuration encoded by $\gamma_{2n+1}w_{2n+1}$ – the bottom right symbol from its head code γ_{2n+1}. For this routine instance the automaton uses the states of the form $p^-_{x,y}$ and $q^-_{x,y}$. Running this instance of the routine on the subword $\gamma_{2n}w_{2n}\gamma_{2n+1}w_{2n+1}\gamma_{2n+2}$ with $\#_l(w_{2n}) = \overline{a_4} + B \cdot \#_l(w_{2n+1})$ from a superposition with one head in state Z_N reading the symbol γ_{2n} eventually leads to a superposition with one head in state Z_N reading the symbol $\gamma_{2n+2} = \gamma_{2m}$.

It is important to note that the third line of γ_{2n} is equal to the first line of γ_{2n+2}, and therefore the automaton knows if its superposition contains states of the $p^+_{x,y}$ and $q^+_{x,y}$ or the $p^-_{x,y}$ and $q^-_{x,y}$ form when reading the symbol γ_{2n+2}. This allows the automaton to decide according to the symbol γ_{2n+2}, which of these two forms of states map onto the state Z_N using the last quantum Fourier transformation of the equation routine. □

Lemma 5.13 *Let $N \ge 3$ and $w = v\gamma_0 w_0 \gamma_1 w_1 \dots w_{k-1}\gamma_k \in L_l$, with $k > 0$, $v \in (\Gamma^T \cup \{l\})^*$, $w_j \in \{l, r\}^*$ for all $j < k$, $\gamma_0 \in C_1$, $\gamma_1 \in C_2$ and $\gamma_j \in C_3$ for all j with $2 \le j \le k$. For each m with $1 \le 2m + 1 \le k$, running the automaton $M^{\text{odd}}_{l,N}$ on the input word w eventually leads to a superposition of one head in the state Z_N reading the symbol γ_{2m+1}.*

Proof: The proof of this lemma is similar to the proof of the previous lemma. The only difference worth mentioning is that the odd version of the automaton remains in its initial superposition of one head in state s until arriving on the symbol γ_1 unlike the even version which keeps the head in state s only until reading the symbol γ_0. □

Lemma 5.14 *Let $N \ge 3$ and $w \in L_1$. If $w \in L_l$ then both $M^{\text{even}}_{l,N}$ and $M^{\text{odd}}_{l,N}$ accept w with probability 1 and if $w \notin L_l$ then either $M^{\text{even}}_{l,N}$ or $M^{\text{odd}}_{l,N}$ rejects w with probability at least $1 - 1/N$.*

Proof: As $w \in L_1$ we can write it as $w = v\gamma_0 w_0 \gamma_1 w_1 \dots w_{k-1}\gamma_k$, with $k \geq 1$, $v \in (\Gamma^T \cup \{l\})^*$, $w_j \in \{l, r\}^*$ for all $j < k$, $\gamma_0 \in C_1$, $\gamma_1 \in C_2$ and $\gamma_j \in C_3$ for all j with $2 \leq j \leq k$.

Assume that $w \in L_l$. If k is even then running the even version of the automaton leads to a superposition of one head in state Z_N reading the symbol γ_k, by lemma 5.12. This head on the symbol γ_k splits to N heads, each of which in one of the states $p^+_{x,y}$ or $p^-_{x,y}$ according to the last line of the code head γ_k of the configuration encoded by $\gamma_k w_k$. In the next few evolution steps (at most $B \cdot N + 1$) each of these N heads moves one square right onto the right end-marker symbol, where it accepts in a state of the form P^+_x or P^-_x. Therefore the automaton $M^{\text{even}}_{l,N}$ accepts for sure.

If k is odd then running the automaton $M^{\text{even}}_{l,N}$ leads to a superposition of one head in state Z_N reading the symbol γ_{k-1}. Similarly, as in the previous case, the only head of the superposition splits on the symbol γ_{k-1} to N heads, each of which is in one of the states $p^+_{x,y}$ or $p^-_{x,y}$. All these heads after reading the subword $w_{k-1}\gamma_k$ change their states from a state of the form $p^+_{x,y}$ or $p^-_{x,y}$ to a corresponding state of the form $q^+_{x,y}$ or $q^-_{x,y}$. Upon moving any of the heads over the right end-marker \$ it accepts in one of the states Q^+_x or Q^-_x. Therefore the automaton $M^{\text{even}}_{l,N}$ accepts for sure again.

By lemma 5.13, we can similarly show that the automaton $M^{\text{odd}}_{l,N}$ accepts each word from L_l for sure, too.

Now, assume that $w \notin L_l$. Since $w \in L_1$, the only reasons why $w \notin L_l$ is that at least one of the equalities (5.5) does not hold for this word. Take the first such equality – i.e., take the minimal number j for which (5.5) does not hold. If j is even, then by lemma 5.12, running the even version of the automaton leads to a superposition of one head in state Z_N reading the symbol γ_j. In the next instance (as described in the inductive case of lemma 5.12) of the equation routine, according to lemma 5.7, the automaton $M^{\text{even}}_{l,N}$ rejects with probability at least $1 - 1/N$. Similarly, by lemma 5.13, if j is odd running the odd version of the automaton leads to a superposition of one head in state Z_N reading the symbol γ_j. In the next instance of the routine, according the same lemma as previously, the automaton $M^{\text{odd}}_{l,N}$ rejects w with probability at least $1 - 1/N$. Therefore every word from $L_1 \setminus L_l$ is rejected by at least one of the two automata with probability at least $1 - 1/N$. □

In lemma 5.14 we proved that the automata recognize for all words from L_1 whether they belong to L_l. However, we have not yet showed how the automata behave on words not from L_1. The next lemma proves that both of the automata eventually halt on every input word, although we do not care what answer the automata halt with for words not it L_1.

Lemma 5.15 *Let $N \geq 3$. Both automata $M^{\text{even}}_{l,N}$ and $M^{\text{odd}}_{l,N}$ halt on every word $w \in \Sigma^*$ after $O(|w|)$ evolution steps.*

Proof: By the exactly same argument as in the proof of lemma 5.11. □

The automata checking behavior of the right tape part are very similar to the automata checking behavior of the left tape part. They work in the same manner as the former two, with the only difference that they check the following equalities instead:

$$\begin{aligned} r_j &= B \cdot r_{j+1} + \overline{a_{j+1}} && \text{for all } j < k-1 \text{ with } \delta^T_3(q_j, a_j) = 1, \text{ and} \\ B \cdot r_j + \overline{c_j} &= r_{j+1} && \text{for all } j < k-1 \text{ with } \delta^T_3(q_j, a_j) = -1. \end{aligned} \tag{5.6}$$

The intersection of the languages recognized by these automata and the language L_1 is the language

$$L_r = \left\{ v \overbrace{\begin{bmatrix} \vdots \\ \vdots \\ q_0\ a_0 \end{bmatrix}}^{\gamma_0 \in C_1} w_0 \overbrace{\begin{bmatrix} \vdots \\ q_0\ a_0 \\ q_1\ a_1 \end{bmatrix}}^{\gamma_1 \in C_2} \left(\prod_{j=2}^{k} w_{j-1} \overbrace{\begin{bmatrix} q_{j-2}\ a_{j-2} \\ q_{j-1}\ a_{j-1} \\ q_j\ a_j \end{bmatrix}}^{\gamma_j \in C_3} \right) \;\middle|\; k > 0 \ \wedge\ q_k \in F^T \ \wedge\ v \in (\Sigma^T \cup \{l\})^* \ \wedge \right.$$
$$\wedge \left(\forall j < k \colon w_j \in \{l, r\}^* \ \wedge\ c_j = \delta_2^T(q_j, a_j) \ \wedge l_j = \#_l(w_j) \ \wedge\ r_j = \#_r(w_j) \right) \wedge$$
$$\wedge \left(\forall j < k-1 \colon \left(\delta_3^T(q_j, a_j) = 1 \quad \rightarrow r_j = B \cdot r_{j+1} + \overline{a_{j+1}} \right) \wedge \right.$$
$$\left. \left. \left(\delta_3^T(q_j, a_j) = -1 \rightarrow B \cdot r_j + \overline{c_j} = r_{j+1} \right) \right) \right\}.$$

For a given $N \geq 3$ we construct the automata $M_{r,N}^{\text{even}}$ and $M_{r,N}^{\text{odd}}$ in a similar way we constructed the automata $M_{l,N}^{\text{even}}$ and $M_{l,N}^{\text{odd}}$, respectively.

Lemma 5.16 *Let $N \geq 3$ and $w \in L_1$. If $w \in L_r$ then both $M_{r,N}^{\text{even}}$ and $M_{r,N}^{\text{odd}}$ accept w with probability 1 and if $w \notin L_r$ then either $M_{r,N}^{\text{even}}$ or $M_{r,N}^{\text{odd}}$ rejects w with probability at least $1 - 1/N$.*

Lemma 5.17 *Let $N \geq 3$. Both automata $M_{r,N}^{\text{even}}$ and $M_{r,N}^{\text{odd}}$ halt on every word $w \in \Sigma^*$ after $O(|w|)$ evolution steps.*

5.4 Putting everything together

Having constructed all seven necessary subautomata we can combine them into the final automaton recognizing the language L. The final automaton randomly chooses which of the seven subautomata to run and then simulates the chosen one. Each of the subautomata is chosen equally probable. The evolution of the final automaton halts as soon as the simulated subautomaton halts. It halts with the same answer as the simulated one. In addition to simulating the new automaton, it may in its first evolution step decide to reject immediately.

The automaton decides to reject immediately with probability 6/13 and to simulate any of the subautomata with probability 1/13. Therefore if any of the subautomata rejects an input word with probability close to 1 the new automaton rejects it with probability close to 7/13. Similarly, if all the subautomata accept the input word for sure the new automaton accepts it with probability exactly 7/13.

For a given N, in effort to make notations clearer, number the subautomata M_1, $M_{e,N}^{\text{even}}$, $M_{e,N}^{\text{odd}}$, $M_{l,N}^{\text{even}}$, $M_{l,N}^{\text{odd}}$, $M_{r,N}^{\text{even}}$ and $M_{r,N}^{\text{odd}}$ with numbers 1 to 7, respectively. Also let V_α^k, D^k and $M^k = (Q^k, \Sigma, \delta^k, q_0^k, Q_{acc}^k, Q_{rej}^k)$ represent the respective objects of the k-th subautomaton as defined in previous sections.

For every $N \geq 3$ define the automaton $M_N = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$, where $Q = \{q_0, q_r\} \cup \{(k,p) \mid 1 \leq k \leq 7 \ \wedge\ p \in Q^k\}$, $\Sigma = \{l, r\} \cup \Sigma^T \cup C$, $Q_{acc} = \{(k,p) \mid 1 \leq k \leq 7 \ \wedge\ p \in Q_{acc}^k\}$ and $Q_{rej} = \{q_r\} \cup \{(k,p) \mid 1 \leq k \leq 7 \ \wedge\ p \in Q_{rej}^k\}$. The tape alphabet of the constructed automaton is $\Gamma = \Sigma \cup \{\text{¢}, \$\}$. Define functions $\mathsf{A}^k \colon \ell_2(Q^k) \rightarrow \ell_2(Q)$, where $1 \leq k \leq 7$, such that $\mathsf{A}^k \left(\sum_{q \in Q^k} \gamma_q |q\rangle \right) = \sum_{q \in Q^k} \gamma_q |(k,q)\rangle$, and let $\mathsf{V}_\text{¢}$ for $|q_0\rangle$ as follows:

$$\mathsf{V}_\text{¢} |q_0\rangle = \frac{\sqrt{6}}{\sqrt{13}} |q_r\rangle + \sum_{k=1}^{7} \frac{1}{\sqrt{13}} \mathsf{A}^k \mathsf{V}_\text{¢}^k |q_0^k\rangle,$$

and arbitrarily extend it to be unitary on the Hilbert space $\ell_2(Q)$. For each $\alpha \in \Gamma \setminus \{\text{¢}\}$ and $(k,q) \in Q$ let $\mathsf{V}_\alpha |(k,q)\rangle = \mathsf{A}^k \mathsf{V}_\alpha^k |q\rangle$, $\mathsf{V}_\alpha |q_0\rangle = |q_0\rangle$ and $\mathsf{V}_\alpha |q_r\rangle = |q_r\rangle$. As all V_α^k are unitary on $\ell_2(Q^k)$, the defined operators V_α are unitary on $\ell_2(Q)$, too. Further, define function D as follows:

$$D(q_0) = 0, \ D(q_r) = 0, \text{ and } D((k,q)) = D^k(q), \text{ for all } (k,q) \in Q,$$

and let δ be defined as in (3.5). Since all V_α are unitary, the constructed automaton is well-formed. We defined the automaton such that if it does not reject immediately, it performs the

first evolution step of the simulated subautomaton and then passes the evolution control to the simulated automaton.

Lemma 5.18 *Let $N \geq 27$ and $w \in \Sigma^*$. If $w \in L$ then M_N accepts w with probability $7/13$ and if $w \notin L$ then M_N rejects w with probability at least $7/13 - 1/N$. Moreover, $P_{acc}^{M_N}(w) + P_{rej}^{M_N}(w) = 1$ and hence the automaton M_N recognizes the language L.*

Proof: From definitions of the languages from previous sections we know that $L = L_1 \cap L_e \cap L_l \cap L_r$. The automaton M_N rejects w with probability $6/13$ immediately in its first evolution step, or it starts to simulate one of the seven subautomata by performing its first evolution step. Simulating any of the automata happens with probability $1/13$.

If $w \in L$ then w belongs to all of the languages L_1, L_e, L_l and L_r. By lemmas 5.4, 5.10, 5.14 and 5.16 all of the subautomata accept w with probability 1. Therefore the automaton M_N accepts w with probability $7/13$. If $w \notin L$ then w does not belong to at least one of the languages L_1, L_e, L_l and L_r. If $w \notin L_1$ then by lemma 5.4 the automaton M^1 rejects w for sure. If $w \notin L_e$ then by lemma 5.10 at least one of M^2 and M^3 rejects w with probability at least $1 - 1/N$. Similarly if $w \notin L_l$ ($w \notin L_r$) then by lemma 5.14 (5.16) at least one of M^4 and M^5 (M^6 and M^7) rejects w with probability at least $1 - 1/N$. In either case at least one of the seven subautomata rejects w with probability at least $1 - 1/N$. As the automaton M_N happens to simulate this subautomaton with probability $1/13$ it rejects in such a simulation with probability at least $1/13(1 - 1/N) > 1/13 - 1/N$. Total probability of rejecting w by M_N is at least $7/13 - 1/N$.

By lemmas 5.5, 5.11, 5.15 and 5.17 all the automata halt with probability 1 and therefore we have $P_{acc}^{M^k}(w) + P_{rej}^{M^k}(w) = 1$ for all k with $1 \leq k \leq 7$. Hence:

$$P_{acc}^{M_N}(w) + P_{rej}^{M_N}(w) = \frac{6}{13} + \sum_{k=1}^{7} \frac{1}{13} \left(P_{acc}^{M^k}(w) + P_{rej}^{M^k}(w) \right) = 1.$$

As $7/13 - 1/N > 1/2$ for $N \geq 27$ the automaton M_N recognizes L. □

Lemma 5.19 *Let $N \geq 3$. The automaton M_N halts on every word $w \in \Sigma^*$ after $O(|w|)$ evolution steps.*

Proof: A straightforward consequence of lemmas 5.5, 5.11, 5.15 and 5.17. □

We constructed the automaton recognizing the language L with probability of a correct answer $7/13 - \varepsilon$. As the reader can see, the subautomata M_e^{even} and M_e^{odd} stop their work where the subautomata M_l^{even} and M_l^{odd} start their work. Therefore, it is possible for the pairs of the subautomata to concatenate their work and construct only two subautomata recognizing languages with the same intersection as the languages recognized by the original four subautomata, with the same probability of a correct answer. Furthermore we can let the work done by the subautomaton M_1 be done by any other of the subautomata as this subautomaton is actually just an one-way reversible finite automaton.

This method allows us to reduce the number of subautomata of the final automaton to just four. The probability of a correct answer of such final automaton would be $4/7 - \varepsilon$. However, to make the construction more understandable, we decided to present its simplified version.

Theorem 5.20 *For every Turing machine T there is an 1.5QFA M and a homomorphism h such that $h(L(M)) = L(T)$.*

Proof: For the machine T, there is an equivalent Turing machine $T' = (K^T, \Sigma^T, \Gamma^T, \delta^T, q_0^T, F^T)$, such that $q_0 \notin F^T$, T' implements only move-left and move-right steps and the computation of T' begins on the right side of the input word. For the Turing machine T' and $N = 47$, we construct all seven subautomata from previous sections and then construct the automaton M_{47}

as described above. According to lemma 5.18, the automaton M_{47} recognizes the language L. For homomorphism $h\colon (\{l,r\} \cup \Sigma^T \cup C)^* \to (\Sigma^T)^*$ defined as follows:

$$\begin{aligned} h(\alpha) &= \varepsilon, \text{ for } \alpha \in \{l,r\} \cup C, \text{ and} \\ h(\alpha) &= \alpha, \text{ for } \alpha \in \Sigma^T, \end{aligned}$$

we have $h(L(M_{47})) = h(L) = L(T') = L(T)$, by lemma 5.1. □

Corollary 5.21 *Homomorphic closure of the class of languages recognized by 1.5QFA is equal to the class of recursively enumerated languages.*

Corollary 5.22 *The class of languages recognized by 1.5QFA is not closed under homomorphism.*

Proof: Trivially $\mathcal{L}_{1.5QFA} \subsetneq \mathcal{L}_{RE}$ and hence $\mathcal{L}_{1.5QFA} \subsetneq \mathcal{H}(\mathcal{L}_{1.5QFA})$. □

Chapter 6

Homomorphic closure of other classes

The result from the previous chapter we can generalize for the case of two-way quantum finite state automata and the case of two-way finite automata with quantum and classical states, too. Also, we can prove a similar result for the class of languages recognized by one-way quantum finite state automata. In section 6.3 we show that the homomorphic closure of this class is equal to the class of regular languages. This property introduces a significant gap between the homomorphic closures of the classes of languages recognized by 1.5QFA, 2QFA and 2QCFA on one hand, and the homomorphic closure of the class languages recognized 1QFA on the other hand.

6.1 Homomorphic closure of 2QFA

Although theorem 5.20 from section 5.4 is sufficient to show that the homomorphic closure of the class of languages recognized by 2QFA is equal to the class of recursively enumerated languages, and hence is not closed under homomorphism, there is a better 2QFA to recognize the language L defined in section 5.3 than the 1.5QFA M_N defined there.

The 1.5QFA M_N defined in the section 5.4 randomly chooses which of the subautomata to simulate and recognizes the language L with probability of a correct answer $7/13 - \varepsilon$ (or $4/7 - \varepsilon$ according to the paragraph before theorem 5.20). A 2QFA, thanks to its ability to move its head back to the left end-marker symbol, can simulate all seven subautomata sequentially and recognize the language L with probability of correct answer $1 - \varepsilon$.

To construct such a 2QFA we need to patch the subautomata such that each of them accepts every word from the language it recognizes at one moment (in one evolution step) in a superposition of just one head in a particular state (independent of the input word) reading the left end-marker symbol, instead of accepting in many different evolution steps or in a superposition of many different states on many different tape symbols.

The automaton M_1 accepts every word from the language L_1 in a superposition of one head reading the right end-marker, but the head may be in any of many different states. Fortunately, the state of this head depends only on the last symbol of the accepted input word and hence the automaton may synchronize the head state by returning back to this symbol and then moving its only head back to the left end-marker. The new 2QFA recognizing the language L will simulate this subautomaton as the first one, so that all the next subautomata may rely on the form of the input word.

The automata $M_{e,N}^{\text{even}}$ and $M_{e,N}^{\text{odd}}$ accept every word from the language L_e in a superposition of one head in state A reading the first occurrence of a symbol from C_1 or C_2 depending on the parity of the number of symbols from Σ^T before. To recognize between the two cases, the automaton can move its only head over the prefix of the accepted input word terminated by the first occurrence of a symbol from C_1 or C_2 again, and count the parity of the number of symbols from Σ^T in this

prefix. Using this parity the automaton can ensure it halts in a superposition of one head in some specific state reading the left end-marker.

The remaining four subautomata checking the progress of the tape accept the words from L_l and L_r, respectively, in possibly N different evolution steps, such that in each of them several superposition heads accepts on the right end-marker. Each of the heads accepts in a different state. However, they always accept during one instance of the equation routine. To synchronize the superposition to only one head during the equation routine, the automaton can "undo" the part of the routine already done. Each head of the equation routine superposition upon reaching the right end-marker changes its state to a symmetric one (i.e., $p^+_{x,y}$ to $p^+_{N-x+1,y}$) and undoes all previously done steps. By this approach, all the heads of the superposition meet in one moment at the symbol where the routine began and synchronize into a superposition of one head in state Z_N. So far, the automaton halts on every accepted word in a superposition of one head in state Z_N reading the last or the last but one input word symbol from C, depending on the parity of the input word. Similarly as previously, the automaton can ensure it halts in a superposition of one head in some specific state reading the left end-marker by counting the parity of the number of symbols from C in the input word located to the left from the symbol, where the undone instance of the equation routine started.

Each of the patched subautomata begins and halts its evolution in a superposition of one head in some particular state reading the left end-marker symbol, hence a 2QFA can easily simulate all the subautomata sequentially.

Lemma 6.1 *For every $\varepsilon > 0$ there is a 2QFA such that it accepts every word from L with probability 1 and rejects every word from $\Sigma^* \setminus L$ with probability at least $1 - \varepsilon$. Moreover, it halts on every word $w \in \Sigma^*$ in $O(|w|)$ evolution steps.*

Theorem 6.2 *For every Turing machine T there is a 2QFA M and a homomorphism h such that $h(L(M)) = L(T)$.*

Corollary 6.3 *Homomorphic closure of the class of languages recognized by 2QFA is equal to the class of recursively enumerated languages.*

Corollary 6.4 *The class of languages recognized by 2QFA is not closed under homomorphism.*

Proof: Trivially $\mathcal{L}_{2QFA} \subsetneq \mathcal{L}_{RE}$ and hence $\mathcal{L}_{2QFA} \subsetneq \mathcal{H}(\mathcal{L}_{2QFA})$. □

6.2 Homomorphic closure of 2QCFA

For the class of languages recognized by 2QCFA, we can also show that its homomorphic closure is equal to the class of recursively enumerated languages, and hence the class is not closed under homomorphism. For every Turing machine we can construct a 2QCFA recognizing the language L (5.2) and a homomorphism (5.3) defined in section 5.3, such that the homomorphic image of the language L is equal to the language accepted by the Turing machine.

In section 5.4, we constructed the 1.5QFA recognizing the language L. The automaton recognized the language with help of seven subautomata each of which recognized a subset of the properties of the language L. The first of the subautomata M_1, defined in section 5.3.1, recognized the language L_1 of codes of all Turing machine computations satisfying local conditions of a correct accepting computation without taking any care of the Turing machine tape codes. This automaton is actually an one-way reversible finite automaton, and so it may be easily simulated by a 2QCFA.

The remaining six subautomata, defined in sections 5.3.2 and 5.3.3, recognized the respective languages by checking the equalities (5.4) of a correct input word code and the equalities (5.5) and (5.6) guaranteeing a progress of the Turing machine tape that correctly obeys the Turing machine transition function.

A 2QCFA performing the task of these six subautomata can be constructed, such that it checks all the equalities checked by these subautomata by a method based on the method to recognize the language $\{a^k b^k \mid k \geq 0\}$ introduced in [AW02].

Lemma 6.5 [AW02, Theorem 5] *For every $\varepsilon > 0$, there is a 2QCFA M that accepts every $w \in \{a^k b^k \mid k \geq 0\}$ with probability 1 and rejects every $w \notin \{a^k b^k \mid k \geq 0\}$ with probability at least $1 - \varepsilon$. The automaton halts in expected time $O(|w|^4)$.*

By a slight variation of this method we can construct a 2QCFA recognizing the language $\{a^x b a^y \mid c_1 + d_1 \cdot x = c_2 + d_2 \cdot y\}$ for any natural numbers c_1, d_1, c_2 and d_2, and use it as a routine of the 2QCFA performing the task of the mentioned subautomata. This routine is a version of the equation routine from section 5.3.2 for the 2QCFA case.

Lemma 6.6 *For every $\varepsilon > 0$ there is a 2QCFA that accepts every word from L with probability 1 and rejects every word from $\Sigma^* \setminus L$ with probability at least $1 - \varepsilon$. Moreover, it halts on every word $w \in \Sigma^*$ in $O(|w|^4)$ evolution steps.*

Theorem 6.7 *For every Turing machine T there is a 2QCFA M and a homomorphism h such that $h(L(M)) = L(T)$.*

Corollary 6.8 *Homomorphic closure of the class of languages recognized by 2QCFA is equal to the class of recursively enumerated languages.*

Corollary 6.9 *The class of languages recognized by 2QCFA is not closed under homomorphism.*

Using the method introduced by Freivalds in [Fre81] to recognize the language $\{a^k b^k \mid k \geq 0\}$ and the language $\{a^{j_1} b^{j_1} a^{j_2} b^{j_2} \dots a^{j_k} b^{j_k} \mid k \geq 0 \wedge (\forall l \leq k \colon j_l \geq 1)\}$ by two-way probabilistic finite state automata (2PFA) in exponential time, we can generalize our result for 2QCFA to the case of 2PFA, too. For a definition of 2PFA see [Kuk73].

Lemma 6.10 [Fre81, Lemma 1] *For every $\varepsilon > 0$ there is a 2PFA that accepts every word from L with probability at least $1 - \varepsilon$ and rejects every word from $\Sigma^* \setminus L$ with probability at least $1 - \varepsilon$.*

Theorem 6.11 *For every Turing machine T there is a 2PFA M and a homomorphism h such that $h(L(M)) = L(T)$.*

Corollary 6.12 *Homomorphic closure of the class of languages recognized by 2PFA is equal to the class of recursively enumerated languages.*

Corollary 6.13 *The class of languages recognized by 2PFA is not closed under homomorphism.*

6.3 Homomorphic closure of 1QFA

In [BP02], it is shown that the class of languages recognized by 1QFA is not closed under homomorphism. A natural question is, what is the smallest class of languages containing the class $\mathcal{L}_{1QFA}$ as a subset and closed under homomorphism. We show that this class is equal to the class of regular languages.

The class of regular languages is the class of languages accepted by one-way (deterministic) finite automata (1FA). For the definition and properties of 1FA see [HU90]. We show that for every 1FA there is an 1QFA and a non-erasing homomorphism, such that the homomorphic image of the language recognized by the 1QFA is equal to the language recognized by the 1FA. The constructed 1QFA will recognize the language of all accepting computations of the 1FA.

Take an one-way deterministic finite automaton $A = (K, \Sigma, \delta, q_0, F)$ with no ε-moves. We construct an 1QFA recognizing the language

$$L = \left\{ \prod_{j=0}^{k-1} (q_j, \alpha_j, q_{j+1}) \;\middle|\; k \geq 0 \wedge (\forall j < k \colon (q_j, \alpha_j, q_{j+1}) \in \Gamma') \wedge q_k \in F \right\}.$$

Let $K_A = \{q_A \mid q \in K\}$ and $K_R = \{q_R \mid q \in K\}$ be two sets of new symbols and define 1QFA $M = (Q, \Sigma', \delta', q_0, Q_{acc}, Q_{rej})$, where $Q = K \cup K_A \cup K_R$, $\Sigma' = \{(p, \alpha, r) \mid p, r \in K \wedge \alpha \in \Sigma \wedge \delta(p, \alpha) = r\}$,

Algorithm 6.1 1QFA recognizing computations of a finite automaton.

$(p,\alpha,r) \in \Sigma' \ \wedge \ p \neq r$:
$\mathsf{V}_{(p,\alpha,r)}|q\rangle = |q_R\rangle$, $q \in K \setminus \{p\}$,
$\mathsf{V}_{(p,\alpha,r)}|p\rangle = |r\rangle$,
$\mathsf{V}_{(p,\alpha,r)}|r_R\rangle = |p\rangle$,

$(p,\alpha,p) \in \Sigma'$:
$\mathsf{V}_{(p,\alpha,p)}|q\rangle = |q_R\rangle$, $q \in K \setminus \{p\}$,
$\mathsf{V}_{(p,\alpha,p)}|p\rangle = |p\rangle$,

$\mathsf{V}_{¢}|q\rangle = |q\rangle$, $q \in K$,
$\mathsf{V}_{\$}|q\rangle = |q_A\rangle$, $q \in F$,
$\mathsf{V}_{\$}|q\rangle = |q_R\rangle$, $q \in K \setminus F$,

$Q_{acc} = K_A$ and $Q_{rej} = K_R$. According to definition 3.1 the tape alphabet of this automaton is equal to $\Gamma' = \Sigma' \cup \{¢, \$\}$. For each $\alpha \in \Gamma'$ let V_α take values as in algorithm 6.1 and arbitrarily extend it to be unitary on the Hilbert space $\ell_2(Q)$. Also define $D(q) = 1$ for all $q \in Q$, and let δ' be defined as in (3.5). Since each V_α is unitary, M is well-formed by lemma 3.13.

Lemma 6.14 *Let $w = \prod_{j=0}^{k-1}(q_j, \alpha_j, q_{j+1})$, where $k \geq 0$, $(q_j, \alpha_j, q_{j+1}) \in \Sigma'$ for all $j < k$ and q_0 is the initial state of the automaton A. The automaton M is after reading the prefix w of a word with such prefix in the superposition $|q_k\rangle$.*

Proof: We prove the lemma by induction on k. If $k = 0$, the word w is empty and the initial superposition $|q_0\rangle$ of the automaton has the required form. Now, assume the lemma holds for all prefixes w with $k = n \geq 0$, we prove it for a prefix w with $k = n + 1$, too. The prefix w of an input word has the form $w = \prod_{j=0}^{n}(q_j, \alpha_j, q_{j+1})$. As it also has prefix $w' = \prod_{j=0}^{n-1}(q_j, \alpha_j, q_{j+1})$, the automaton M is after reading the prefix w' in the superposition $|q_n\rangle$, by induction hypothesis. As the symbol following the prefix w' – the last symbol of w is equal to the symbol q_n, the automaton after reading this symbol changes its superposition to $|q_{n+1}\rangle = |q_k\rangle$. □

Lemma 6.15 *Let $w \in \Sigma^*$. If $w \in L$, the automaton M accepts w for sure and if $w \notin L$, it rejects w for sure.*

Proof: If $w \in L$, by lemma 6.14, the automaton M is after reading the input word w in the superposition $|q\rangle$, where (p, α, q) is the last symbol of w. Reading the right end-marker \$ the automaton changes its superposition to $|q_A\rangle$ and accepts, as $q \in F$.

Now assume the input word $w \notin L$. As w does not belong to the language L, there are two consecutive symbols in w of the form $(p, \alpha, q)(r, \beta, s)$ with $q \neq r$, or the the last symbol of w has the form (p, α, q) with $q \notin F$.

If there are two consecutive symbols of the form $(p, \alpha, q)(r, \beta, s)$ with $q \neq r$ in the word w, take the first such pair. By lemma 6.14, the automaton M is after reading the prefix terminated by the first symbol of this pair in the superposition $|q\rangle$. As $q \neq r$, reading the symbol (r, β, s) the automaton changes its superposition to $|q_R\rangle$ and rejects.

If no two consecutive symbols of w has the mentioned form and $w \notin L$, the last symbol of w has the form (p, α, q) with $q \notin F$. After reading the input word w the automaton M, by lemma 6.14, is in the superposition $|q\rangle$. As $q \notin F$, reading the right end-marker \$ the automaton changes its superposition to $|q_R\rangle$ and rejects. □

Theorem 6.16 *For every one-way deterministic finite automaton A with no ε-moves there is an 1QFA M and a non-erasing homomorphism h such that $h(L(M)) = L(A)$.*

Proof: Construct the automaton M for the 1FA A as described above. According to lemma 6.15, the automaton M recognizes the language L. For the non-erasing homomorphism $h: \Sigma'^* \to \Sigma^*$ defined, such that $h((p, \alpha, r)) = \alpha$ for all $(p, \alpha, r) \in \Sigma'$, we have $h(L(M)) = h(L) = L(A)$. □

Corollary 6.17 *Homomorphic closure of the class of languages recognized by 1QFA is equal to the class of regular languages.*

Proof: As $\mathcal{L}_{1QFA} \subseteq \mathcal{R}$ [KW97] and the class of regular languages is closed under homomorphism, we have $\mathcal{H}(\mathcal{L}_{1QFA}) \subseteq \mathcal{R}$. Therefore, $\mathcal{H}(\mathcal{L}_{1QFA}) = \mathcal{R}$. □

Chapter 7

Conclusion

In this thesis we have proved several new closure properties of different quantum finite automata. However, a number of questions have been left open.

What other closure properties the discussed models of quantum finite automata have? Is the class of languages recognized by 1.5QFA closed under general inverse homomorphism? Is the class of languages recognized by 2QFA closed under inverse non-erasing homomorphism, or even under general inverse homomorphism? How about the class of languages recognized by simple 2QFA?

We have shown that the classes of languages recognized by 1.5QFA, 2QFA and 2QCFA are not closed under general homomorphism. Are they closed under non-erasing homomorphism? Our 1.5QFA in section 5, by which we have shown that the homomorphic closure of the class of languages recognized by 1.5QFA is equal to the class of recursively enumerable languages, has a quite large error probability. Is it possible to reduce it, or is the homomorphic closure of the class of languages recognized by 1.5QFA with small error probability different than the homomorphic closure of the class of languages recognized by 1.5QFA with large error probability?

Bibliography

[AF98] Andris Ambainis and Rūsiņš Freivalds. 1-way quantum finite automata: strengths, weaknesses and generalizations. In *FOCS '98: Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, page 332, Washington, DC, USA, 1998. IEEE Computer Society.

[AI99] Masami Amano and Kazuo Iwama. Undecidability on quantum finite automata. In *STOC '99: Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 368–375, New York, NY, USA, 1999. ACM Press.

[AĶ03] Andris Ambainis and Arnolds Ķikusts. Exact results for accepting probabilities of quantum automata. *Theor. Comput. Sci.*, 295(1-3):3–25, 2003.

[AĶV01] Andris Ambainis, Arnolds Ķikusts, and Māris Valdats. On the class of languages recognizable by 1-way quantum finite automata. In *STACS '01: Proceedings of the 18th Annual Symposium on Theoretical Aspects of Computer Science*, pages 75–86, London, UK, 2001. Springer-Verlag.

[AW02] Andris Ambainis and John Watrous. Two-way finite automata with quantum and classical states. *Theoretical Computer Science*, 287(1):299–311, 2002.

[BP02] Alex Brodsky and Nicholas Pippenger. Characterizations of 1-way quantum finite automata. *SIAM J. Comput.*, 31(5):1456–1478, 2002.

[BV93] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. In *STOC '93: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 11–20, New York, NY, USA, 1993. ACM Press.

[Deu85] David Deutsch. Quantum theory, the church-turing principle and the universal quantum computer. *Royal Society of London Proceedings Series A*, 400:97–117, July 1985.

[Deu89] David Deutsch. Quantum computational networks. *Royal Society of London Proceedings Series A*, 425:73–90, September 1989.

[DS90] Cynthia Dwork and Larry Stockmeyer. A time complexity gap for two-way probabilistic finite-state automata. *SIAM J. Comput.*, 19(6):1011–1123, 1990.

[Fre81] Rūsiņš Freivalds. Probabilistic two-way machines. In *Proceedings on Mathematical Foundations of Computer Science*, pages 33–45, London, UK, 1981. Springer-Verlag.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, New York, NY, USA, 1996. ACM Press.

[Gro98] Lov K. Grover. A framework for fast quantum mechanical algorithms. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 53–62, New York, NY, USA, 1998. ACM Press.

[Gru99] Jozef Gruska. *Quantum Computing*. McGraw-Hill, London, UK, 1999.

[HU90] John E. Hopcroft and Jeffrey D. Ullman. *Introduction To Automata Theory, Languages, And Computation*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[KF91] Jānis Kaņeps and Rūsiņš Freivalds. Running time to recognize nonregular languages by 2-way probabilistic automata. In *Proceedings of the 18th international colloquium on Automata, languages and programming*, pages 174–185, New York, NY, USA, 1991. Springer-Verlag New York, Inc.

[Košo5] Peter Košinár. Konečné kvantové výpočtové modely. Master's thesis, Faculty of Mathematics, Physics and Informatics, Comenius University, Bratislava, Slovakia, 2005.

[Kuk73] Yu. I. Kuklin. Two-way probabilistic automata. In *Avtomatika i vyčislitelnaja tekhnika*, volume 5, pages 35–36, 1973.

[KW97] Attila Kondacs and John Watrous. On the power of quantum finite state automata. In *IEEE Symposium on Foundations of Computer Science*, pages 66–75, 1997.

[Mar86] Norman Margolus. Quantum computation. *Annals of the New York Academy of Science*, 480:287–297, 1986.

[NC00] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[Sho94] Peter W. Shor. Polynominal time algorithms for discrete logarithms and factoring on a quantum computer. In *ANTS-I: Proceedings of the First International Symposium on Algorithmic Number Theory*, page 289, London, UK, 1994. Springer-Verlag.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.

[VSB+01] L. Vandersypen, M. Steffen, G. Breyta, C. Yannoni, M. Sherwood, and I. Chuang. Experimental realization of shor's quantum factoring algorithm using nuclear magnetic resonance. *Nature*, 414:883–887, December 2001.

[Wat95] John Watrous. On one-dimensional quantum cellular automata. In *FOCS '95: Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS'95)*, page 528, Washington, DC, USA, 1995. IEEE Computer Society.

[Yao93] Andrew C. Yao. Quantum circuit complexity. In *34th Annual Symposium on Foundations of Computer Science*, pages 352–361, 1993.

Slovenský abstrakt

V tejto diplomovej práci sme dokázali niekoľko nových uzáverových vlastnosti tried jazykov rozpoznávaných jeden-a-pol-smernými a dvojsmernými konečnými kvantovými automatmi (1.5QFA a 2QFA) a dvojsmernými konečnými automatmi s kvantovými a klasickými stavmi (2QCFA). Ukázali sme, že žiadna z týchto tried jazykov nie je uzavretá na homomorfizmus. Ďalej sme dokázali, že triedy jazykov rozpoznávaných pomocou 1.5QFA a zjednodušených 2QFA (simple 2QFA) sú uzavreté na inverzný nevymazávajúci homomorfizmus a trieda jazykov rozpoznávaných pomocou zjednodušených 1.5QFA (simple 1.5QFA) je uzavretá na všeobecný inverzný homomorfizmus. Tiež sme dokázali, že homomorfné uzávery tried jazykov rozpoznávaných pomocou 1.5QFA, 2QFA a 2QCFA sú zhodné s triedou rekurzívne vyčísliteľných jazykov a homomorfný uzáver triedy jazykov rozpoznávaných pomocou jednosmerných konečných kvantových automatov (1QFA) je zhodný s triedou regulárnych jazykov.

Kľúčové slová: Kvantové výpočty, Konečné kvantové automaty, 1.5QFA, 2QFA, 2QCFA.