quant-ph/9604007 v2 12 Jul 1996

A decision procedure for unitary linear quantum cellular automata

Christoph Dürr*†
Université Paris-Sud, LRI, bât 490
91405 Orsay Cedex, France
http://www.lri.fr/~durr
durr@lri.fr

Miklos Santha*‡
CNRS, URA 410
Université Paris-Sud, LRI, bât 490
91405 Orsay Cedex, France
santha@lri.fr

To appear in FOCS'96

Abstract

Linear quantum cellular automata were introduced recently as one of the models of quantum computing. A basic postulate of quantum mechanics imposes a strong constraint on any quantum machine: it has to be *unitary*, that is its time evolution operator has to be a unitary transformation. In this paper we give an efficient algorithm to decide if a linear quantum cellular automaton is unitary. The complexity of the algorithm is $O(n^{\frac{4r-3}{r+1}}) = O(n^4)$ if the automaton has a continuous neighborhood of size r.

1 Introduction

The classical models of computation, such as Turing machines, random access machines, circuits, or cellular automata are all universal in the sense that they can simulate each other with only polynomial overhead. These models are based on classical physics, whereas physicists believe that the universe is better described by quantum mechanics.

Feynman [Fey82, Fey86] pointed out first that there might be a substantial gap between computational models based on classical physics and those based on quantum mechanics. The *quantum Turing machine* (QTM), the first model of quantum computation, was introduced by Benioff [Ben82a, Ben82b]. Deutsch in [Deu85] described a universal simulator for QTMs with exponential overhead. Bernstein and Vazirani [BV93] were able to construct a universal QTM with only polynomial overhead.

Other quantum computational models were also studied recently. Deutsch [Deu89] has defined the model of *quantum circuits*, and later Yao [Yao93] has shown that QTMs working in polynomial time can be simulated by polynomial size quantum circuits. Physicists were also interested in *quantum cellular automata*: Biafore [Bia94] considered the problem of synchronization, Margolus [Mar94] described space-periodic quantum cellular automata and Lloyd [Llo93, Llo94] discussed the possibility to realize a special type of quantum cellular automaton. *Linear quantum cellular automata* (LQCAs) were formally defined by Watrous [Wat95] and by Dürr, Lê Thanh and Santha [DLS96]. In the former paper it was shown that a subclass of LQCAs, *partitioned linear quantum cellular automata* (PLQCAs) can be simulated by QTMs with linear slowdown.

A quantum computational device is at any moment of its computation in a *superposition of configurations*, where each configuration has an associated complex amplitude. A superposition is *valid* if it has unit norm. If the device is *observed* then a configuration will be chosen at random, where the probability of a configuration to be chosen is equal to the squared magnitude of its amplitude. Therefore it is essential that valid superpositions be transformed into valid superpositions, or equivalently, that the *time evolution operator* of the device preserve the norm. This property is called the *well-formedness*, and thus it is a natural problem to decide if a given quantum machine is well-formed. In the case of QTMs and PQLCAs there exist easily checkable constraints on the finite local transition function of the machine which are equivalent to its well-formedness. Such constraints were identified respectively by Bernstein and Vazirani [BV93] and by Watrous [Wat95]. In the case of LQCAs no such local constraints are known, still Dürr, Lê Thanh and Santha [DLS96] gave a polynomial

*Part of this research was supported by the ESPRIT Working Group RAND2.
†Part of this research was supported by the ISI Foundation.
‡Part of this work was done while the author visited the Centre de Recerca Matemàtica, Institut d'Estudis Catalans, Bellaterra, Spain.

time algorithm to decide if an LQCA is well-formed. Part of this algorithm was improved by Høyer [Høy96] using a different approach.

However, one of the basic postulates of quantum mechanics imposes an even stronger constraint then norm-preserving on the time evolution operator. It actually requires that this operator — as any other quantum operator — be a unitary transformation. We will call a machine which satisfies this constraint *unitary*. In [BV93] and [Wat95] it was proven that norm-preserving already implies unitarity in the case of QTMs and PLQCAs. It is also trivially true for machines with finite configuration set, such as quantum circuits. But this is not true for LQCAs; it is quite simple to construct a well-formed LQCA which is not unitary.

In this paper we give an efficient algorithm to decide if an LQCA is unitary. The complexity of our algorithm is $O(n^4)$ if the input LQCA has continuous neighborhood (most papers in the literature about classical linear cellular automata deal only with such cases). Our algorithm will use the procedure of [DLS96] which in time $O(n^2)$ decides if the LQCA is well-formed. The present paper actually gives an algorithm which decides if a well-formed LQCA is also unitary.

Well-formedness is equivalent to the orthonormality of the column vectors of the time evolution operator, unitarity requires the same also from the row vectors. Deciding unitarity is much harder than deciding well-formedness. One way of seeing this is that whereas the column vectors have finite support, the row vectors can have an infinite number of non zero components.

2 The computational model

Let us fix for the paper the following notation. If u and v are vectors in some inner-product space over the complex or the real numbers, then $\langle u|v\rangle$ will denote the inner product of u and v, and $\|u\|$ the norm of u. If M is a matrix in such a space, then M^* denotes its conjugate transpose.

We recall here the definition of a *linear quantum cellular automaton* (LQCA) which is the quantum generalization of the classical one-dimensional cellular automaton. A more detailed description of this model can be found in [Wat95] and in [DLS96].

An LQCA is a 4-tuple $A = (\Sigma, q, N, \delta)$. The *cells* of the automaton are organized in a line, and are indexed by the elements of $\mathbb{Z}$. The finite, nonempty set Σ is the set of *(cell-)states*, and $q \in \Sigma$ is a distinguished *quiescent* state. The *neighborhood* $N = (a_1, \ldots, a_r)$ is a strictly increasing sequence of integers, for some $r \geq 1$, giving the addresses of the neighbors relative to each cell. This means that the neighbors of cell i are indexed by $i+a_1, \ldots, i+a_r$. An automaton is *simple* if its neighborhood is an interval of integers, that is $a_r = a_1 + r - 1$. In this paper we deal only with simple automata, and we will only explain briefly in the conclusion how our results apply to the general case.

The states of the cells are changing simultaneously at every time step according to the *local transition function*. This is the mapping $\delta : \Sigma^{|N|} \times \Sigma \to \mathbb{C}$, which satisfies that for every $(x_1, \ldots, x_r) \in \Sigma^r$, there exists $y \in \Sigma$ such that $\delta(x_1 \ldots x_r, y) \neq 0$. If at some time step the neighbors of a cell are in states $x_1, \ldots, x_r$ then at the next step the cell will change into state y with amplitude $\delta(x_1 \ldots x_r, y)$. The quiescent state q satisfies that

$$\delta(q^r, y) = \begin{cases} 1 & \text{if } y = q, \\ 0 & \text{if } y \neq q. \end{cases}$$

The set of *configurations* is by definition $\Sigma^{\mathbb{Z}}$, where for every configuration c, and for every integer i, the state of the cell indexed by i is $c(i)$. A configuration c is *finite* if its *support* $\{i : c(i) \neq q\}$ is finite. We are dealing only with LQCAs which evolve on finite configurations. We will denote the set of finite configurations by $\mathcal{C}_A$, and from now on we use the word *configuration* to mean a *finite configuration*. For a configuration c, let $\mathrm{idom}(c)$ be the *interval domain* of c, which is smallest integer interval containing the support of c. For the sake of definiteness, we define the empty interval as $[0, -1]$ which is the interval domain of the everywhere quiescent configuration.

The local transition function induces the *time evolution operator* $U_A : \mathcal{C}_A \times \mathcal{C}_A \to \mathbb{C}$, where $U_A(d, c)$ is the transition amplitude of changing configuration c to configuration d in one step. It is defined by

$$U_A(d, c) = \prod_{i \in \mathbb{Z}} \delta(c(i + a_1) \ldots c(i + a_r), d(i)).$$

This product is well-defined since both d and c have finite support.

The automaton evolves on *superpositions of configurations* which are by definition elements of the Hilbert space $l_2(\mathcal{C}_A)$. If at some time step the automaton is in the superposition $u \in \mathbb{C}^{\mathcal{C}_A}$, then at the next time step it will be in the superposition $U_A u$. By definition, A is *well-formed* if U_A is norm-preserving, and we say that it is *unitary* if U_A is a unitary transformation.

We will work in the algebraic computational model where by definition complex numbers take unit space,

and arithmetic and logical operations take unit time. The description size of an automaton is clearly dominated by the local transition table δ. Therefore we define the *size* of the automaton to be $n = |\Sigma^{r+1}|$.

For the rest of the paper we will fix an LQCA $A = (\Sigma, q, N, \delta)$. For the ease of notation we will suppose without loss of generality that $N = (0, 1, \ldots, r-1)$.

3 The main result

The main result of the paper is the following theorem.

Theorem 1 *There exists an algorithm which takes a simple LQCA as input, and decides in time* $O(n^{\frac{4r-3}{r+1}}) = O(n^4)$ *if it is unitary.*

Since in [DLS96] a $O(n^2)$ algorithm is given to decide if an LQCA is well-formed, we will give only an algorithm which decides if a well-formed LQCA is unitary. The following lemma states that we only have to verify that the rows of the time evolution operator are of unit norm.

Lemma 1 *Let* $U \in \mathbb{C}^{\mathcal{C}_A \times \mathcal{C}_A}$ *be a linear operator. If* U *is norm-preserving then its rows have norm at most* 1. *If all the rows are of unit norm then* U *is unitary.*

Proof Let c be a configuration, and $\mathbf{c}$ the superposition which has amplitude 1 for c and 0 elsewhere. Then the norm of the row indexed by c in U is $\|U^*\mathbf{c}\|$. Since U is norm-preserving $\|U^*\mathbf{c}\| = \|UU^*\mathbf{c}\|$ and the projection of $UU^*\mathbf{c}$ on $\mathbf{c}$ has norm

$$|\langle \mathbf{c}|UU^*\mathbf{c}\rangle| = \langle U^*\mathbf{c}|U^*\mathbf{c}\rangle = \|U^*\mathbf{c}\|^2.$$

But the projection on $UU^*\mathbf{c}$ on a unit vector has norm at most $\|U^*\mathbf{c}\|$, and therefore $\|U^*\mathbf{c}\| \leq 1$.

For the second part of the lemma observe that the projection of $UU^*\mathbf{c}$ on $\mathbf{c}$ has norm 1. Since U is norm-preserving the projection on any other basis vector $\mathbf{c}'$ must be 0. Thus $\langle \mathbf{c}'|UU^*\mathbf{c}\rangle$ is 1 if $c = c'$ and 0 otherwise, or in other words $UU^* = I$, which concludes the proof. □

The outline of the proof is the following. First we give a sequence of reduction steps in section 4 to a graph theoretical problem and to another one from linear algebra. We define an infinite de Bruijn graph whose infinite paths are in bijection with the (column) configurations. Every (row) configuration induces a weight on the edges of that graph such that the total weight of the infinite paths is the norm of the row. This total weight can be decomposed in some sense into the weight of generating the infinite to the left, quiescent part of the row configuration; the weight of generating its non trivial part; and the weight of generating its infinite to the right, quiescent part. The weights of the quiescent parts can be represented as vectors in some finite dimensional real vector space, we call them the border vectors. We show that they are configuration independent, and reduce their computation to computing the weight of all finite paths in a finite graph. Section 5 contains an algorithm for this problem. The weight of generating the non trivial part of a configuration is configuration dependent, and can be represented by a sequence of linear transformations in the same vector space. Our problem is finally reduced to the following linear algebra question: given an affine hyperplane, a vector, and a finite set of linear applications, is it true that any finite composition of the applications maps the vector into the affine hyperplane? We give an algorithm for this problem in section 6.

Theorem 1 is a direct consequence of theorems 2,3 and 4.

4 The reduction

The *configuration graph* is the infinite, directed graph $G_\infty = (V, E)$ where $V = \Sigma^{r-1} \times \mathbb{Z}$, and $E = \{((xt, i), (ty, i+1)) : x, y \in \Sigma, t \in \Sigma^{r-2}, i \in \mathbb{Z}\}$. A non empty (possibly infinite to the left, to the right, or to both direction) sequence of vertices $(\ldots, (z_i, i), \ldots)$ in G_∞ is a *path* if $z_i = q^{r-1}$ for all but finitely many indices, and there is an edge between two consecutive vertices. Observe that a sequence containing a single vertex is a path. We denote by F, L, R and P respectively the set of finite, infinite to the left, infinite to the right and infinite paths.

We say that two paths p_1 and p_2 are *compatible* if the last vertex of p_1 and the first vertex of p_2 exist and they are the same. In that case the *composition* $p_1 \otimes p_2$ is the concatenation of the two sequences after identifying the extreme vertices. If P_1 and P_2 are sets of paths, then

$$P_1 \otimes P_2 = \left\{ p_1 \otimes p_2 : \begin{array}{l} p_1 \in P_1, p_2 \in P_2, \\ p_1 \text{ and } p_2 \text{ are compatible} \end{array} \right\}.$$

Let d be an arbitrary configuration. It induces a *weight* function w_d for the edges of G_∞, where by definition $w_d((xt, i), (ty, i+1)) = |\delta(xty, d(i))|^2$. We extend the weight function w_d to paths and to set of paths. The weight of a path is the product of the respective edge weights, and the weight of a path set is the sum of the respective path weights. The weight of a path consisting of a single vertex is 1, and the weight of the empty path set is 0. We denote this *weighted configuration graph* by G_∞^d.

Although the weight of an infinite path is an infinite product, it is well defined since all but a finite number of edges have weight 1. The following lemma establishes a strong relationship between the weight of an infinite path in G^d_∞ and the entries of the time evolution matrix.

Lemma 2 *There is a bijection g between the set of configurations $\mathcal{C}_A$ and the set of infinite paths P in G^d_∞ such that for every configuration c and d*

$$w_d(g(c)) = |U_A(d,c)|^2.$$

Proof Let $g : \mathcal{C}_A \to P$ be defined for all configuration c by

$$g(c) = (\ldots,(c(i)\ldots c(i+r-2),i),\ldots).$$

Then it is a bijection, and the following equalities can easily be verified.

$$\begin{aligned} w_d(g(c)) = & \\ = & \prod_{i\in\mathbb{Z}} w_d((c(i)\ldots c(i+r-2),i), \\ & \qquad (c(i+1)\ldots c(i+r-1),i+1)) \\ = & \prod_{i\in\mathbb{Z}} |\delta(c(i)\ldots c(i+r-1),d(i))|^2 \\ = & \left|\prod_{i\in\mathbb{Z}} \delta(c(i)\ldots c(i+r-1),d(i))\right|^2 \\ = & |U_A(d,c)|^2. \end{aligned}$$

□

We got the following reduction of our problem.

Corollary 1 *In a well-formed LQCA, for every configuration d, we have $w_d(P) \le 1$. Moreover the automaton is unitary if and only if for every d, $w_d(P) = 1$.*

Let us fix a configuration d where $\mathrm{idom}(d) = [j,k]$ for some $j \le k+1$. This configuration induces a subset of the path sets L, F and R. For every $z, z' \in \Sigma^{r-1}$ we set

$$L^d_z = \{p \in L : \text{the last vertex of } p \text{ is } (z,j)\},$$

$$F^d_{z,z'} = \left\{ p \in F : \begin{array}{l}\text{first vertex of } p \text{ is } (z,j),\\ \text{last vertex of } p \text{ is } (z',k+1)\end{array} \right\},$$

$$R^d_{z'} = \{p \in R : \text{the first vertex of } p \text{ is } (z',k+1)\}.$$

Since the set of infinite paths can be decomposed as

$$P = \bigcup_{z,z'\in\Sigma^{r-1}} L^d_z \otimes F^d_{z,z'} \otimes R^d_{z'},$$

we have

$$w_d(P) = \sum_{z,z'\in\Sigma^{r-1}} w_d(L^d_z)\cdot w_d(F^d_{z,z'})\cdot w_d(R^d_{z'}).$$

The following lemma shows that $w_d(L^d_z)$ and $w_d(R^d_z)$ are independent from d.

Lemma 3 *For any configurations d, d' and for every $z \in \Sigma^{r-1}$, we have*

$$w_d(L^d_z) = w_{d'}(L^{d'}_z) \quad \text{and} \quad w_d(R^d_z) = w_{d'}(R^{d'}_z).$$

Proof We prove only the first equation, the proof for the second one is analogue. Let $\mathrm{idom}(d) = [j,k]$, $\mathrm{idom}(d') = [j',k']$ and $m = j' - j$. We define a bijection from L^d_z to $L^{d'}_z$ which preserves the weight. If $p = (\ldots,(z_i,i),\ldots,(z_j,j))$ is a path in L^d_z, then we define its image as $p' = (\ldots,(z_{i-m},i),\ldots,(z_j,j'))$. This is clearly a bijection and we also have $w_d(p) = w_{d'}(p')$ since $d(i) = q$ for $i < j$ and $d'(i) = q$ for $i < j'$. □

We define the *left* and *right border vectors*, respectively $\vec{l}, \vec{r}$ as follows: for $z \in \Sigma^{r-1}$, $\vec{l}(z) = w_d(L^d_z)$ and $\vec{r}(z) = w_d(R^d_z)$, where d is an arbitrary configuration. The next lemma states that $\vec{l}$ and $\vec{r}$ are in $\mathbb{R}^{\Sigma^{r-1}}$.

Lemma 4 *For all $z \in \Sigma^{r-1}$, $\vec{l}(z)$ and $\vec{r}(z)$ are finite.*

Proof Suppose there is a z such that $\vec{l}(z) = \infty$. (The case $\vec{r}(z) = \infty$ is symmetric.) Set $z = x_1\ldots x_{r-1}$, and let $z' = x'_1\ldots x'_{r-1}$ be such that $\vec{r}(z') > 0$. There exists such a vector since for example $\vec{r}(q^{r-1}) \ge 1$. For $i = 1,\ldots,r$ we denote $x_i\ldots x_{r-1}x'_1\ldots x'_{i-1}$ by z_i, and for $i = 1,\ldots,r-1$ we denote $x_i\ldots x_{r-1}x'_1\ldots x'_i$ by v_i. Note that $z_1 = z$ and $z_r = z'$. For $i = 1,\ldots,r-1$, let $y_i \in \Sigma$, be such that $\delta(v_i,y_i) \ne 0$. Let j be an arbitrary integer, and set $k = j+r-2$. We define the configuration d to be the quiescent state outside the interval $[j,k]$ and $d(j+i-1) = y_i$, for $i = 1,\ldots,r-1$. Then in G^d_∞ already the set of paths going through the vertices $(z_1,j),\ldots,(z_r,k+1)$ has infinite weight. Since each path has non-negative weight, P has also infinite weight, which by corollary 1 contradicts the fact that U_A is norm-preserving. □

The first part of our algorithm will be the computation of the border vectors. For the second part, we reduce now our problem to a question in linear algebra.

For every $a \in \Sigma$, let $M_a \in \mathbb{R}^{\Sigma^{r-1}\times\Sigma^{r-1}}$ be the linear operator whose matrix is defined for all $z, z' \in \Sigma^{r-1}$ as

$$M_a(z,z') = \begin{cases} |\delta(xty,a)|^2 & \text{if } z = xt, z' = ty \\ & \text{for some } x,y \in \Sigma \\ & \text{and } t \in \Sigma^{r-2}, \\ 0 & \text{otherwise.} \end{cases}$$

We extend this definition to finite sequences over Σ. If ϵ denotes the empty word, then M_ϵ is the identity operator. Let $s > 1$ be an integer, and $b = b_1 \dots b_s$ be an element of Σ^s. Then

$$M_b = M_{b_s} \times \cdots \times M_{b_1}.$$

Lemma 5 *Let d be a configuration with* $\mathrm{idom}(d) = [j,k]$. *Then*

$$w_d(P) = \langle M_{d(j)\dots d(k)} \vec{l} | \vec{r} \rangle.$$

Proof We have

$$\begin{aligned} w_d(P) &= \sum_{z,z' \in \Sigma^{r-1}} w_d(L^d_z) \cdot w_d(F^d_{z,z'}) \cdot w_d(R^d_{z'}) \\ &= \sum_{z,z'} \vec{l}(z) \cdot w_d(F^d_{z,z'}) \cdot \vec{r}(z') \\ &= \sum_{z'' \in \Sigma^{r-1}} M_{d(j)\dots d(k)} \vec{l}(z'') \cdot \vec{r}(z'') \\ &= \langle M_{d(j)\dots d(k)} \vec{l} | \vec{r} \rangle. \end{aligned}$$

□

Since for every $b \in \Sigma^*$, there exists a configuration d whose non-quiescent part is b, corollary 1 and lemma 5 imply the following reduction.

Corollary 2 *A well-formed LQCA is unitary if and only if for every $b \in \Sigma^*$, we have $\langle M_b \vec{l} | \vec{r} \rangle = 1$.*

In particular, we must have $\langle \vec{l} | \vec{r} \rangle = 1$, which is equivalent to the condition that the row in U_A corresponding to the everywhere quiescent configuration is of unit norm. This single condition can be checked easily, and from now on we will suppose that it is satisfied. It will make our last reduction step simpler.

Let $m = |\Sigma^{r-1}|$. The border vectors can be seen as elements of $\mathbb{R}^m$, and the elements of the set $\mathcal{M} = \{M_a : a \in \Sigma\}$ can also be seen as linear applications in $\mathbb{R}^m$. Let us fix a few notations for the inner product space $\mathbb{R}^m$. Let $\vec{v} \in \mathbb{R}^m$ be a vector, $U \subseteq \mathbb{R}^m$ a finite set of vectors, and $\mathcal{F} \subseteq \mathbb{R}^{m \times m}$ a finite family of linear applications. We set $U + \vec{v} = \{\vec{u} + \vec{v} : \vec{u} \in U\}$, and $\mathcal{F}(U) = \{f(\vec{u}) : \vec{u} \in U, f \in \mathcal{F}\}$. The subspace generated by U will be denoted by $\langle U \rangle$, and let $H_{\vec{v}}$ denote the hyperplane whose normal vector is $\vec{v}$, that is $H_{\vec{v}} = \{\vec{u} : \langle \vec{u} | \vec{v} \rangle = 0\}$.

We define by induction on i, for $i \geq 0$, the sets $\mathcal{F}^i(U)$. Let $\mathcal{F}^0(U) = U$, and $\mathcal{F}^{i+1}(U) = \mathcal{F}^i(U) \cup \mathcal{F}(\mathcal{F}^i(U))$. We say that U is *closed* for $\vec{v}$ under $\mathcal{F}$, if $\bigcup_{i=0}^{\infty} \mathcal{F}^i(\{\vec{v}\}) \subseteq U$.

Since we have $\langle \vec{l} | \vec{r} \rangle = 1$, the affine hyperplane $H_{\vec{r}} + \vec{l}$ is the set of vectors which have unit inner product with $\vec{r}$, that is

$$H_{\vec{r}} + \vec{l} = \{\vec{u} : \langle \vec{u} | \vec{r} \rangle = 1\}.$$

Clearly for every $b \in \Sigma^*$, $\langle M_b \vec{l} | \vec{r} \rangle = 1$ if and only if $\langle \vec{l} | \vec{r} \rangle = 1$, and $H_{\vec{r}} + \vec{l}$ is closed for $\vec{l}$ under $\mathcal{M}$. Therefore we can summarize our reduction steps in the following theorem:

Theorem 2 *A well formed LQCA is unitary if and only if $\langle \vec{l} | \vec{r} \rangle = 1$, and $H_{\vec{r}} + \vec{l}$ is closed for $\vec{l}$ under $\mathcal{M}$.*

5 Computing the border vectors

In this section we will give an algorithm for computing the border vectors. By symmetry, it will be sufficient to give it only for the left vector. The main tool in the computation will be the weighted border graph. Its underlying graph can be seen as a slight modification of the finite version of the configuration graph. This graph was also used in [DLS96] for checking that all the columns of U_A had unit norms. However, there the weights were defined as the norms of the transition state superpositions, whereas here they will be the squared magnitudes of the amplitude of the quiescent state in those superpositions.

The *(left) border graph* is the finite, directed, weighted graph $G_l = (V, E, w)$. The vertex set is $V = \Sigma^{r-1} \cup \{sq^{r-2}\}$ where $s \notin \Sigma$. The edge set is

$$\begin{aligned} E = &\{(xt, ty) : x, y \in \Sigma, t \in \Sigma^{r-2}\} \cup \\ &\{(sq^{r-2}, q^{r-2}y) : y \in \Sigma, y \neq q\}. \end{aligned}$$

The weight function is defined as

$$\begin{aligned} & w((xt, ty)) = |\delta(xty, q)|^2 \\ \text{and} \quad & w((sq^{r-2}, q^{r-2}y)) = |\delta(q^{r-1}y, q)|^2. \end{aligned}$$

A *path* in G_l is a finite, non empty sequence of at least two vertices such that there is an edge between two consecutive vertices. Observe that a single vertex alone here does not form a path. As usual, the weight of a path is the product of the edge weights, and the weight of a set of paths is the sum of the individual path weights. The weight of the empty path set is 0.

For every $z \in \Sigma^{r-1}$, we define P_z as the set of paths in G_l whose first vertex is sq^{r-2} and whose last vertex is z.

Lemma 6 *For every $z \in \Sigma^{r-1}$, we have*

$$\vec{l}(z) = \begin{cases} w(P_z) & \text{if } z \neq q^{r-1}, \\ w(P_z) + 1 & \text{if } z = q^{r-1}. \end{cases}$$

Proof Let d be a configuration with integer domain $[j,k]$, and let $p_q = (\dots, (q^{r-1}, i), \dots, (q^{r-1}, j))$.

We set $L'_z = L^d_z - \{p_q\}$. We will give a weight preserving bijection from L'_z to P_z which maps p to p'. Let $p = (\ldots,(z_i,i),\ldots,(z_j,j))$ be an element of L'_z, where $z_j = z$. Let $h \leq j$ be the greatest integer such that for every $i \leq h$, we have $z_i = q^{r-1}$. Then by definition we set $p' = (sq^{r-2}, z_{h+1}, \ldots, z_j)$. This is clearly an injective mapping, and it is also surjective since there is no edge in G_l from sq^{r-2} to q^{r-1}. It is also weight preserving since the edges in p until the vertex (z_h, h) have all weight 1. Since $w_d(p_q) = 1$, the lemma follows. □

Theorem 3 *There exists an algorithm which computes the border vectors in time* $O(n^{\frac{3(r-1)}{r+1}})$.

Proof According to the lemma 6 it is sufficient to compute $w(P_z)$ for $z \in \Sigma^{r-1}$. The border vectors have only finite components, nevertheless for their computation we have to extend the non-negative real numbers with ∞. Let $\mathbb{R}^*$ be this set. We define the following computation rules with respect to ∞:

$$\infty + c = c + \infty = \infty,$$

for every $c \in \mathbb{R}^*$,

$$\infty \cdot c = c \cdot \infty = \infty \cdot \infty = \infty$$

for every real number $c > 0$,

$$\infty \cdot 0 = \infty \cdot 0 = 0,$$

and $\infty^0 = 1$. We also define c^* for every $c \in \mathbb{R}^*$ as follows

$$c^* = \begin{cases} 1/(1-c) & \text{if } 0 \leq c < 1, \\ \infty & \text{otherwise.} \end{cases}$$

Let $\{v_1, v_2, \ldots, v_{|V|}\}$ an arbitrary enumeration of the vertices of G_l. For $1 \leq i,j \leq |V|$ and for $0 \leq k \leq |V|$, we define the path sets $P_k(i,j)$ as the set of paths which start in v_i, end in v_j, and all the other vertices in the path are in the set $\{v_h : h \leq k\}$. Let $W_k(i,j)$ denote $w(P_k(i,j))$. Then we claim that $W_k(i,j)$ satisfies the following recursion for $1 \leq i,j \leq |V|$ and $1 \leq k \leq |V|$:

$$W_0(i,j) = \begin{cases} w((v_i,v_j)) & \text{if } (v_i,v_j) \in E, \\ 0 & \text{otherwise.} \end{cases}$$

$$W_k(i,j) = W_{k-1}(i,j) + W_{k-1}(i,k) \cdot (W_{k-1}(k,k))^* \cdot W_{k-1}(k,j).$$

We prove our claim by induction on k. In $P_0(i,j)$ the only path is the edge between v_i and v_j if this edge exists. In $P_k(i,j)$, for $1 \leq k \leq |V|$, every path which contains v_k as intermediate vertex can be decomposed into $e+2$ subpaths for some integer $e \geq 0$ such that the first one goes from v_i to v_k, the last one goes from v_k to v_j, the e other subpaths are cycles from v_k to v_k, and on all this subpaths, all the intermediate vertices are from the set $\{v_h : h \leq k-1\}$. By the inductive hypothesis, the total weight of all paths which go trough the vertex v_k and contain exactly e cycles around it is $W_{k-1}(i,k) \cdot (W_{k-1}(k,k))^e \cdot W_{k-1}(k,j)$. Observe that two paths containing respectively e and e' cycles around v_k are different if $e \neq e'$ because the sequence (v_k) by definition is not a path. Therefore the total weight of $P_k(i,j)$ is

$$\begin{aligned} & W_{k-1}(i,j) + \\ & \quad \sum_{e=0}^{\infty} (W_{k-1}(i,k) \cdot (W_{k-1}(k,k))^e \cdot W_{k-1}(k,j)) \\ = \; & W_k(i,j). \end{aligned}$$

The complexity of the algorithm is $O(|\Sigma|^{3(r-1)}) = O(n^{\frac{3(r-1)}{r+1}})$. □

6 Closed affine hyperplanes

In this section we will give a polynomial time algorithm to decide if the affine hyperplane $H_{\vec{r}} + \vec{l}$ is closed for $\vec{l}$ under the set of linear applications $\mathcal{M} = \{M_a : a \in \Sigma\}$. Let us recall the notation $m = |\Sigma|^{r-1}$, and set $t = |\Sigma|$. For the simplicity of notation, let $H = H_{\vec{r}}$ and $H' = H_{\vec{r}} + \vec{l}$. Set $E_i = \mathcal{M}^i(\{\vec{l}\})$.

The main difficulty to decide if $\bigcup_{i=0}^{\infty} E_i \subseteq H'$ is that $\bigcup_{i=0}^{\infty} E_i$ might contain an infinite number of vectors and H' is not a subspace. More precisely, a priori there is no fixpoint E_j such that if E_j is included in H' then $\bigcup_{i=0}^{\infty} E_i$ is also included in H'. Therefore it is not even clear that there is a finite algorithm for the problem.

It would be much easier if we were asking, instead of an affine hyperplane, if a hyperplane is closed for $\vec{l}$ under linear applications. Then we could work with the generated subspaces $\langle E_i \rangle$, and the existence of a fixpoint would easily follow from dimension arguments. The main idea of the proof is that we can exchange in the problem the affine hyperplane and the set of linear applications for a hyperplane and a set of affine applications, and then the above approach works. Let now make this idea precise.

For every $M \in \mathcal{M}$, we define the application $M' : \mathbb{R}^m \to \mathbb{R}^m$ by

$$M'(\vec{u}) = M(\vec{u} + \vec{l}) - \vec{l},$$

and we set $\mathcal{M}' = \{M' : M \in \mathcal{M}\}$. Observe that M' is an affine transformation, because $M'(\vec{u}) = M(\vec{u}) + M'(\vec{0})$. Also $M'(\vec{0}) = M(\vec{l}) - \vec{l}$. We set $E'_i = E_i - \vec{l}$.

Lemma 7 *$E'_0 = \{\vec{0}\}$ and for all $i \geq 0$, we have $E'_{i+1} = E'_i \cup \mathcal{M}'(E'_i)$.*

Proof By definition $E'_0 = \{\vec{l}\} - \vec{l} = \{\vec{0}\}$. For all $i \geq 0$ we have

$$\begin{aligned} E'_{i+1} &= E_{i+1} - \vec{l} \\ &= (E_i \cup \mathcal{M}(E_i)) - \vec{l} \\ &= (E_i - \vec{l}) \cup (\mathcal{M}(E_i) - \vec{l}) \\ &= E'_i \cup (\mathcal{M}(E_i - \vec{l} + \vec{l}) - \vec{l}) \\ &= E'_i \cup (\mathcal{M}(E'_i + \vec{l}) - \vec{l}) \\ &= E'_i \cup \mathcal{M}'(E'_i). \end{aligned}$$

□

Since $E_i \subseteq H'$ if and only if $E'_i \subseteq H$ we have the following corollary.

Corollary 3 *H' is closed for $\vec{l}$ under $\mathcal{M}$ if and only if H is closed for $\vec{0}$ under $\mathcal{M}'$.*

This corollary already implies that there is a finite algorithm for the problem: we compute the fixpoint E'_j (that is the smallest integer j such that $\langle E'_j \rangle = \langle E'_{j+1} \rangle$) and we check if $E'_j \subseteq H$. Obviously j is at most m, the dimension of the space. The problem is that E'_j may contain an exponential number of vectors. Therefore instead of E'_i we will only compute a basis B_i of $\langle E'_i \rangle$, for all i until we reach a fixpoint. Since a basis has at most m elements, our algorithm will be polynomial. The correctness of this approach relies on the following lemma.

Lemma 8 *Let $X, Y \subseteq \mathbb{R}^m$ such that $\langle X \rangle = \langle Y \rangle$, and $\mathcal{M}'(\{\vec{0}\}) \subseteq \langle X \rangle$. Then we have*

$$\langle X \cup \mathcal{M}'(X) \rangle = \langle Y \cup \mathcal{M}'(Y) \rangle.$$

Proof We will prove that $X \cup \mathcal{M}'(X) \subseteq \langle Y \cup \mathcal{M}'(Y) \rangle$. Clearly $X \subseteq \langle Y \cup \mathcal{M}'(Y) \rangle$. Therefore let $\vec{v} \in \mathcal{M}'(X)$. Then $\vec{v} = M'(\vec{u})$ for some $M' \in \mathcal{M}'$, and $\vec{u} \in X$. Since $X \subseteq \langle Y \rangle$, for some k, there exist $\vec{u}_j \in Y$ and $\alpha_j \in \mathbb{R}$, for $1 \leq j \leq k$, such that $\vec{u} = \sum_{j=1}^{k} \alpha_j \vec{u}_j$. A simple computation shows that

$$\vec{v} = \sum_{j=1}^{k} \alpha_j M'(\vec{u}_j) + (1 - \sum_{j=1}^{k} \alpha_j) M'(\vec{0}).$$

Now $M'(\vec{u}_j) \in \mathcal{M}'(Y)$, therefore $\sum_{j=1}^{k} \alpha_j M'(\vec{u}_j) \in \langle \mathcal{M}'(Y) \rangle$. By hypothesis $(\sum_{j=1}^{k} \alpha_j - 1) M'(\vec{0})$ is in $\langle Y \rangle$. Therefore the sum of these two vectors is in $\langle Y \cup \mathcal{M}'(Y) \rangle$. □

Theorem 4 *There exists an algorithm which decides if H' is closed for $\vec{l}$ under $\mathcal{M}$ in time $O(n^{\frac{4r-3}{r+1}})$.*

Proof By corollary 3 we will decide if H is is closed for $\vec{0}$ under $\mathcal{M}'$ with the following algorithm:

$B_1 :=$ a basis of $\langle \mathcal{M}'(\{\vec{0}\}) \rangle$
$i := 1$
while $\langle B_i \rangle \neq \langle B_i \cup \mathcal{M}'(B_i) \rangle$
  $B_{i+1} :=$ a basis of $\langle B_i \cup \mathcal{M}'(B_i) \rangle$
  $i := i + 1$
$B := B_i$
if $B \subseteq H$ accept
else reject

At every iteration $\dim(\langle B_i \rangle)$ increases therefore the algorithm terminates in at most $m-1$ iterations. We prove that it is correct.

Since $\mathcal{M}'(\{\vec{0}\}) \subseteq \langle B_1 \rangle$, and $\langle B_i \rangle \subseteq \langle B_{i+1} \rangle$, we have $\mathcal{M}'(\{\vec{0}\}) \subseteq \langle B_i \rangle$ for every $i \geq 1$. We prove that $\langle B_i \rangle = \langle E'_i \rangle$ for every $i \geq 1$ by induction. The statement holds by definition for $i = 1$, and the inductive step follows from lemmas 7 and 8. Since B is a fixpoint, that is $\langle B \rangle = \langle B \cup \mathcal{M}'(B) \rangle$, this implies that $\langle B \rangle = \langle \bigcup_{i=0}^{\infty} E'_i \rangle$ and the correctness follows.

We now turn to the analysis of the complexity. Since $|\mathcal{M}'| = t$ and $|B_i| \leq m$ for every i, we have $|\mathcal{M}'(B_i)| \leq tm$. Therefore computing $\mathcal{M}'(B_i)$ takes time $O(tm^3)$ using highschool multiplication. A basis of a subspace generated by at most $(t+1)m$ vectors can also be computed in time $O(tm^3)$ using Gaussian elimination. Therefore the time spent per iterations is $O(tm^3)$, and the overall time is $O(tm^4)$. Since $t = n^{\frac{1}{r+1}}$ and $m = n^{\frac{r-1}{r+1}}$, the result follows. □

7 Conclusion

The analysis of the complexity of our algorithm was not tight. If $M(n)$ denotes the complexity of multiplying two n-dimensional matrices then it can be shown that the algorithm works in time $n^{\frac{r-1}{r+1}} M(n^{\frac{r}{r+1}})$.

A not necessarily simple LQCA with neighborhood $N = (a_1, \ldots, a_r)$ can be transformed into a simple one with the same time evolution operator. The size of the new neighborhood will be $s = a_r - a_1 + 1$. If we define the *expansion factor* of an LQCA as $e = (s+1)/(r+1)$ then the algorithm works in the general case in time $O(n^{4e})$.

In the case of space periodic configurations van Dam [Dam96] has shown that LQCAs can be efficiently simulated by QTMs. This question remains still open for finite configurations.

Acknowledgments

We are thankful to Laurent Rosaz, Huong Lê Thanh and Umesh Vazirani for several helpful conversations.

References

[Ben82a] P. Benioff, Quantum mechanical Hamiltonian models of Turing machines, *J. Stat. Phys.* **29**, 515–546, 1982.

[Ben82b] P. Benioff, Quantum mechanical Hamiltonian models of Turing machines that dissipates no energy, *Physical Review Letters* **48**, 1581–1585, 1982.

[Bia94] M. Biafore, Can Computers Have Simple Hamiltonians? *Proceedings of the Workshop on Physics and Computation*, 63–69, 1994.

[BV93] E. Bernstein and U. Vazirani, Quantum complexity theory, *Proceeding of the 25th ACM Symposium on the Theory of Computing*, 11–20, 1993.

[Dam96] W. van Dam, A universal Quantum Cellular Automata, submitted to the *4th Workshop on Physics and Computation*, `wimvdam@cs.kun.nl`, 1996.

[Deu85] D. Deutsch, Quantum theory, the Church-Turing principle and the universal quantum computer, *Proceeding of the Royal Society of London*, A400:97–117, 1985.

[Deu89] D. Deutsch, Quantum computational networks, *Proceeding of the Royal Society of London*, A425:73–90, 1989.

[DLS96] C. Dürr, H. Lê Thanh and M. Santha, A decision procedure for well-formed linear quantum cellular automata, *Proceeding of the 13th Annual Symposium on Theoretical Aspects of Computer Science*, 281–292, 1996.

[Fey82] R. Feynman, Simulating physics with computers, *International Journal of Theoretical Physics* **21**, 467–488, 1982.

[Fey86] R. Feynman, Quantum Mechanical Computers, *Foundations of Physics* **16**, 507, 1986.

[Høy96] P. Høyer, Note on Linear Quantum Cellular Automata, *manuscript*, `u2pi@imada.ou.dk`, 1996.

[Llo93] S. Lloyd, A potentially realizable Quantum Computer, *Science* **261**, 1569–1571, 1993.

[Llo94] S. Lloyd, Envisioning a Quantum Supercomputer, *Science* **263**, 695, 1994.

[Mar94] N. Margolus, Parallel Quantum Computation, *Complexity, Entropy and the Physics of Information*, Addison-Wesley, 1994.

[Wat95] J. Watrous, On one dimensional quantum cellular automata, *Proceeding of the 36th IEEE Symposium on Foundations of Computer Science*, 528–537, 1995.

[Yao93] A. Yao, Quantum circuit complexity, *Proceeding of the 34th IEEE Symposium on Foundations of Computer Science*, 352–361, 1993.