A decision procedure for unitary linear quantum cellular automata*

Christoph Dürr† Miklos Santha‡

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 $O(n^{\frac{3r-1}{r+1}}) = O(n^3)$ in the algebraic computational model if the automaton has a continuous neighborhood of size r, where n is the size of the input.

Key words. quantum computation, reversible cellular automata.
AMS subject classification. 81V99, 68Q80.

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 [8, 9] 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 [1, 2]. Deutsch in [5] described a universal simulator for QTMs with exponential overhead. Bernstein and Vazirani [3] were able to construct a universal QTM with only polynomial overhead.

Other quantum computational models were also studied recently. Deutsch [6] has defined the model of *quantum circuits*, and later Yao [19] 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 [4] considered the problem of synchronization, Margolus [14] described space-periodic quantum cellular automata and Lloyd [12, 13] discussed the possibility to realize a special type of quantum cellular automaton. *Linear quantum cellular automata* (LQCAs) were formally defined by Watrous [18] and by Dürr, LêThanh and Santha [7]. 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. Van Dam [17] defined space-periodic LQCAs and gave a universal instance of this model.

*Part of this research was supported by the ESPRIT Working Group RAND2.

†Université Paris-Sud, LRI, bât 490, F-91405 Orsay Cedex, durr@lri.fr. Part of this research was supported by the ISI Foundation.

‡CNRS, URA 410, Université Paris-Sud, LRI, bât 490, F-91405 Orsay Cedex, santha@lri.fr. Part of this work was done while the author visited the Centre de Recerca Matemàtica, Institut d'Estudis Catalans, Bellaterra, Spain.

We should make clear at this point that most of these models are only theoretically motivated. *Real life* quantum computers as built today in laboratories are essentially small quantum circuits or partitioned cellular automata.

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 *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 PLQCAs 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 [3] and by Watrous [18]. In the case of LQCAs no such local constraints are known, still Dürr, LêThanh and Santha [7] gave a polynomial time algorithm to decide if an LQCA is well-formed. Part of this algorithm was improved by Høyer [10] using a different approach.

However, one of the basic postulates of quantum mechanics imposes an even stronger constraint than 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 [3] and [18] 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 sets, 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.[1]

In this paper we give an efficient algorithm to decide if an LQCA is unitary. The complexity of our algorithm is cubic 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 [7] which in quadratic time 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 orthonormality 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 [18] and in [7].

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, non-empty 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

[1] A well-known example is the classical LCA $\textsc{Xor} = (\{0,1\}, 0, (0,1), \delta)$, where $\delta(x,y) = (x+y) \bmod 2$. This cellular automaton is injective for finite configurations but not surjective, thus the associated LQCA is well-formed but not unitary.

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 $\delta : \Sigma^{|N|} \to \mathbb{C}^{\Sigma}$, 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 $[\delta(x_1 \ldots x_r)](y)$ which is denoted by $\langle \delta(x_1 \ldots x_r) | y \rangle$. The quiescent state q satisfies that

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

The set of *configurations* is $\Sigma^{\mathbb{Z}}$, 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 the 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* which we write in matrix form $U_A : \mathcal{C}_A \times \mathcal{C}_A \to \mathbb{C}$, where $U_A(d, c)$ is the transition amplitude of changing configuration c to configuration d in one step. It is defined by

$$U_A(d, c) = \prod_{i \in \mathbb{Z}} \langle \delta(c_{i+N}) | d_i \rangle,$$

where $\delta(c_{i+N})$ is a short notation for $\delta(c_{i+a_1}, \ldots, c_{i+a_r})$. This product is well-defined since c has finite support.

The automaton evolves on *superpositions of configurations* which are elements of the Hilbert space $\ell_2(\mathcal{C}_A)$. 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 $U_A u$. Therefore U_A is also an operator on $\ell_2(\mathcal{C}_A)$. 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 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 a well-formed simple LQCA $A = (\Sigma, q, N, \delta)$. Without loss of generality we assume $N = (0, 1, \ldots, r-1)$. Indeed let $A' = (\Sigma, q, N', \delta)$ be the well-formed simple LQCA with the general neighborhood $N' = (j, \ldots, j + r - 1)$ for some integer j. We claim that A is unitary if and only if A' is unitary. Let $A'' = (\Sigma, q, (j), \delta')$ be the *shift cellular automaton*, where δ' is the identity. A'' is unitary since δ' is unitary. Moreover $U_A = U_{A''} U_{A'}$ which proves the claim.

2.1 Example

The figures in this paper will illustrate our algorithm with the following LQCA:
QFLIP$= (\{a, b\}, a, (0, 1), \delta)$ with $\langle\delta(x, y)|z\rangle$ defined for all $x, y, z \in \{a, b\}$ by the table:

$xy\backslash z$	a	b
aa	1	0
ba	0	1
ab	$1/\sqrt{2}$	$1/\sqrt{2}$
bb	$1/\sqrt{2}$	$-1/\sqrt{2}$

Using the algorithm in [7] it can be shown that QFLIP is well-formed, and in this article we show that its evolution operator U_{QFLIP} is even unitary. This LQCA has an interesting property. For $n \geq 0$, let c^n be the configuration which is b in all cells of index $i \in [-n, -1]$ and is a elsewhere. Then we have for all $n \geq 1$, $U_{\text{QFLIP}}(c^1, c^n) = (1/\sqrt{2})^n$. Thus an infinite number of configurations lead with non-zero amplitude to the single configuration c^1.

For the all quiescent configuration c^0, we have $U_{\text{QFLIP}}(c^0, c^0) = 1$, thus there is a unique configuration leading to it. Therefore when the LQCA QFLIP runs backwards in time, every cell with index $i \leq -1$ depends on cell -1. From this we conclude that there cannot be a LQCA with finite neighborhood whose evolution operator is exactly U^*_{QFLIP}.

This makes the model of LQCA different from QTMs, since for every well-formed QTM M, there exists a QTM which runs M backward in time with a constant time overhead. It explains a bit why it seems difficult to simulate any LQCA by a QTM.

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* $O(n^{\frac{3r-1}{r+1}}) = O(n^3)$ *if it is unitary.*

Since in [7] 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 of $UU^*\mathbf{c}$ 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 then give an algorithm to solve the problem. The different steps of the algorithm are presented in sections 5, 6 and 7. The proof of the main theorem is then summarized in section 8.

4 The reduction

The different reduction steps are illustrated in the figures 1 to 3.

Our problem is the following. We have to decide if all row vectors of the evolution operator associated to a given LQCA have unit norm, under the assumption that the operator is norm preserving. The naive method fails because one would have to compute the norm for an infinite number of rows. Moreover for every row there can be an infinite number of non-zero entries and every entry is defined by a product on an unbounded number of terms. The purpose of this section is to reduce our problem to a finite one.

The *configuration graph* is the infinite directed graph $G_\infty(V, E)$ defined by $V = \Sigma^{r-1} \times \mathbb{Z}$ and $E = \{((xt, i), (ty, i+1)) : x, y \in \Sigma, t \in \Sigma^{r-2}, i \in \mathbb{Z}\}$. To our knowledge this type of graph has been first used by Sutner and Maas [16] to show that a particular robot motion planning problem in the presence of moving obstacles is PSPACE-hard. It was used again by Sutner [15] to prove that the predecessor of every recursive configuration is also recursive.

A non-empty sequence (possibly infinite to the left, to the right or in both directions) of vertices $(\ldots, (w_i, i), \ldots)$ in G_∞ is a *path* if and only if for at at most a finite number of indices i we have $w_i \neq q^{r-1}$ and there is an edge between every two immediate vertices. Note that a sequence with a single vertex is already a path. We denote by F, L, R and P respectively the set of paths which are finite, infinite to the left, infinite to the right and infinite to both directions. Figure 1 illustrates a path of P.

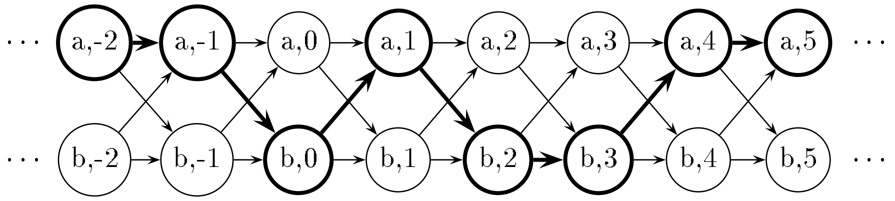

Figure 1: The configuration graph of the LQCA QFLIP. The bold path corresponds to the configuration $\ldots aababbaa \ldots$.

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 g_d for the edges of G_∞, where $g_d((xt, i), (ty, i+1)) = |\langle \delta(xty) | d_i \rangle|^2$. We extend the weight function g_d to paths and to sets 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_∞ which is illustrated in figure 2.

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.

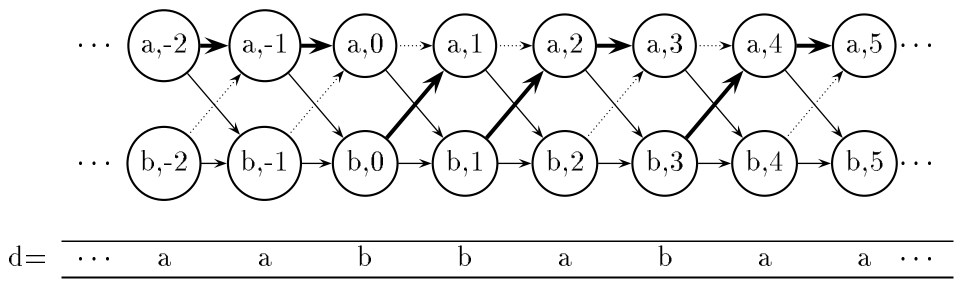

Figure 2: The configuration graph of the LQCA QFLIP weighted by the configuration $d = \dots aabbabaa \dots$. The bold edges have weight 1, the normal edges have weight 1/2 and the dotted-line edges have weight 0.

Lemma 2 *For any configuration d the row indexed by d in U_A has norm $\sqrt{g_d(P)}$.*

Proof We will show that there is a bijection h 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

$$g_d(h(c)) = |U_A(d,c)|^2.$$

Summing up over all configurations c will immediately conclude the lemma.

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

$$h(c) = (\dots, (c_i \dots c_{i+r-2}, i), \dots).$$

Then it is a bijection, and the following equalities conclude the proof.

$$\begin{aligned} g_d(h(c)) &= \prod_{i\in\mathbb{Z}} g_d((c_i \dots c_{i+r-2}, i), (c_{i+1} \dots c_{i+r-1}, i+1)) \\ &= \prod_{i\in\mathbb{Z}} |[\delta(c_{i+N})](d_i)|^2 \\ &= \left| \prod_{i\in\mathbb{Z}} [\delta(c_{i+N})](d_i) \right|^2 \\ &= |U_A(d,c)|^2. \end{aligned}$$

□

With Lemma 1 we got the following reduction of our problem.

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

Let us fix an interval $[j,k]$ and a configuration d with $\mathrm{idom}(d) \subseteq [j,k]$. This interval induces a subset of the path sets L, F and R. For every $w, w' \in \Sigma^{r-1}$ we set

$$\begin{aligned} L^j_w &= \{p \in L : \text{the last vertex of } p \text{ is } (w,j)\}, \\ F^{j,k}_{w,w'} &= \left\{ p \in F : \begin{array}{l} \text{first vertex of } p \text{ is } (w,j), \\ \text{last vertex of } p \text{ is } (w',k+1) \end{array} \right\}, \end{aligned}$$

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

Since the set of infinite paths can be decomposed as

$$P = \bigcup_{w,w' \in \Sigma^{r-1}} L^j_w \otimes F^{j,k}_{w,w'} \otimes R^k_{w'},$$

we have

$$g_d(P) = \sum_{w,w' \in \Sigma^{r-1}} g_d(L^j_w) \cdot g_d(F^{j,k}_{w,w'}) \cdot g_d(R^k_{w'}).$$

The following lemma shows that $g_d(L^j_w)$ and $g_d(R^k_w)$ are independent from j, k and d.

Lemma 3 *For any intervals $[j,k]$, $[j',k']$ and configurations d, d' such that* $\mathrm{idom}(d) \subseteq [j,k]$, $\mathrm{idom}(d') \subseteq [j',k']$ *and for every $w \in \Sigma^{r-1}$, we have*

$$g_d(L^j_w) = g_{d'}(L^{j'}_w) \quad \textit{and} \quad g_d(R^k_w) = g_{d'}(R^{k'}_w).$$

Proof We prove only the first equation, the proof for the second one is analogous. Let $m = j' - j$. We define a bijection from L^j_w to $L^{j'}_w$ which preserves the weight. If $p = (\ldots, (w_i, i), \ldots, (w_j, j))$ is a path in L^j_w, then we define its image as $p' = (\ldots, (w_i, i+m), \ldots, (w_j, j+m))$. This is clearly a bijection and we also have $g_d(p) = g_{d'}(p')$ since $d_i = q$ for $i < j$ and $d'_i = q$ for $i < j'$. □

We define the *left* and *right border vectors*, respectively $\vec{l} = (l_w)_{w \in \Sigma^{r-1}}$ and $\vec{r} = (r_w)_{w \in \Sigma^{r-1}}$ as follows: for $w \in \Sigma^{r-1}$, $l_w = g_d(L^j_w)$ and $r_w = g_d(R^k_w)$, where $[j,k]$ is an arbitrary interval and d an arbitrary configuration satisfying $\mathrm{idom}(d) \subseteq [j,k]$. The next lemma states that $\vec{l}$ and $\vec{r}$ are in $\mathbb{R}^{\Sigma^{r-1}}$.

Lemma 4 *For all $w \in \Sigma^{r-1}$, l_w and r_w are finite.*

Proof Suppose there is a w such that $l_w = \infty$. (The case $r_w = \infty$ is symmetric.) We will prove that this implies the existence of a configuration such that the associated line vector has infinite norm, thus contradicting by Corollary 1 the hypothesis that A is well-formed.

Let w' be such that $r_{w'} > 0$. There exists such a w', since for example $r_{q^{r-1}} \geq 1$. Let $x_1, x_2, \ldots, x_{2r-1} \in \Sigma$ such that $w = x_1 \ldots x_{r-1}$ and $w' = x_r \ldots x_{2r-2}$. We set $w_i = x_i x_{i+1} \ldots x_{i+r-2}$ for $i = 1, \ldots, r$ and $v_i = x_i x_{i+1} \ldots x_{i+r-1}$ for $i = 1, \ldots, r-1$. Note that $w_1 = w$ and $w_r = w'$. For $i = 1, \ldots, r-1$ let $y_i \in \Sigma$ be such that $\langle \delta(v_i) | y_i \rangle \neq 0$. 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 $(w_1, j), \ldots, (w_r, k+1)$ has infinite weight. Since each path has non-negative weight, P has also infinite weight. □

The first part of our algorithm will be the computation of the border vectors $\vec{l}$ and $\vec{r}$. 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 $w, w' \in \Sigma^{r-1}$ as

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

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 $b = b_1 \ldots b_s$ be an element of Σ^s. We define

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

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

$$g_d(P) = \langle M_{d_j \ldots d_k} \vec{l} | \vec{r} \rangle.$$

Proof We have

$$\begin{aligned}
g_d(P) &= \sum_{w, w' \in \Sigma^{r-1}} g_d(L_w^j) \cdot g_d(F_{w,w'}^{j,k}) \cdot g_d(R_{w'}^k) \\
&= \sum_{w,w'} l_w \cdot g_d(F_{w,w'}^{j,k}) \cdot r_{w'} \\
&= \sum_{w,w'} l_w \cdot M_{d_j \ldots d_k}(w', w) \cdot r_{w'} \\
&= \sum_{w'} \left(\sum_{w} l_w \cdot M_{d_j \ldots d_k}(w', w) \right) \cdot r_{w'} \\
&= \sum_{w'} (M_{d_j \ldots d_k} \vec{l})(w') \cdot r_{w'} \\
&= \langle M_{d_j \ldots d_k} \vec{l} | \vec{r} \rangle.
\end{aligned}$$

□

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 $\langle M_b \vec{l} | \vec{r} \rangle = 1$.*

We also have the following property, which simplifies our next reduction step.

Lemma 6 *For every well-formed LQCA A we have $\langle \vec{l} | \vec{r} \rangle = 1$.*

Proof Let d be the all quiescent configuration. Then in U_A the column indexed by d has the entry 1 at row d and 0 elsewhere. Since the column vectors of U_A are pairwise orthogonal, Lemma 1 implies that the row indexed by d has only zero entries besides column d. Therefore this row has norm 1, and the claim follows from Lemma 2 and 5. □

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 transformations in $\mathbb{R}^m$. Let us fix a few notations for the inner product space $\mathbb{R}^m$. Let $S \subseteq \mathbb{R}^m$ a finite set of vectors. The *linear subspace* and the *affine subspace* generated by S, denoted here respectively by $\langle S \rangle$ and $[S]$ are defined as

$$\begin{aligned}
\langle S \rangle &= \{\lambda_1 \vec{s}_1 + \ldots + \lambda_t \vec{s}_t \,|\, t \geq 0; \vec{s}_1, \ldots, \vec{s}_t \in S; \lambda_1, \ldots, \lambda_t \in \mathbb{R}\}, \\
[S] &= \{\lambda_1 \vec{s}_1 + \ldots + \lambda_t \vec{s}_t \,|\, t \geq 0; \vec{s}_1, \ldots, \vec{s}_t \in S; \lambda_1, \ldots, \lambda_t \in \mathbb{R}; \lambda_1 + \ldots + \lambda_t = 1\}.
\end{aligned}$$

B is said to be a basis of $\langle S \rangle$ (respectively of $[S]$) if $\langle B \rangle = \langle S \rangle$ (respectively $[B] = [S]$) and has minimal cardinality for this property.

Let $\vec{u} \in \mathbb{R}^m$ be a vector and $\mathcal{F} \subseteq \mathbb{R}^{m \times m}$ be a finite family of linear transformations. We set $S + \vec{u} = \{\vec{v} + \vec{u} : \vec{v} \in S\}$, and $\mathcal{F}(S) = \{f(\vec{v}) : \vec{v} \in S, f \in \mathcal{F}\}$. Let $H_{\vec{u}}$ denote the linear subspace whose normal vector is $\vec{u}$, that is $H_{\vec{u}} = \{\vec{v} : \langle \vec{v} | \vec{u} \rangle = 0\}$.

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

From Lemma 6, $H_{\vec{r}} + \vec{l}$ is the set of vectors which have unit inner product with $\vec{r}$, that is

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

It is an affine subspace since $H_{\vec{r}} + \vec{l} = [H_{\vec{r}} + \vec{l}]$. Clearly for every $b \in \Sigma^*$, $\langle M_b \vec{l} | \vec{r} \rangle = 1$ if and only if $H_{\vec{r}} + \vec{l}$ is closed for $\vec{l}$ under $\mathcal{M}$. Therefore by Corollary 2 our reduction steps lead to the following theorem:

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

This characterization is illustrated in figure 3.

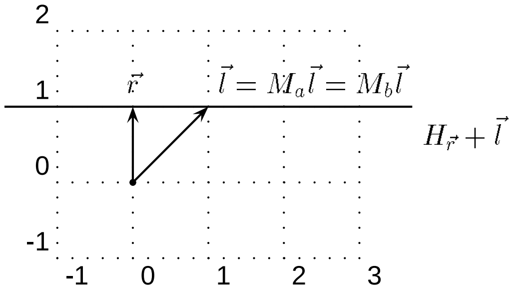

Figure 3: For the LQCA QFLIP, the border vectors $\vec{l}$ and $\vec{r}$, and the affine subspace $H_{\vec{r}} + \vec{l}$. In this example $\vec{l}$ is a fix-point for the operators M_a and M_b, which shows that U_{QFLIP} is unitary.

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 [7] 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 $G_l = (V, E, g)$. The vertex set is $V = \Sigma^{r-1}$ and the edge set is

$$E = \{(xt, ty) : x, y \in \Sigma, t \in \Sigma^{r-2}\}$$

The weight function is defined as

$$g((xt, ty)) = |\langle \delta(xty) | q \rangle|^2.$$

A *path* in G_l is a finite, non empty sequence of at least two vertices such that there is an edge between every 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 $w \in \Sigma^{r-1}$, we define P_w as the set of paths in G_l whose first vertex is q^{r-1}, whose second vertex is different from q^{r-1} and whose last vertex is w.

Lemma 7 *For every $w \in \Sigma^{r-1}$, we have*

$$l_w = \begin{cases} g(P_w) & \text{if } w \neq q^{r-1}, \\ g(P_w) + 1 & \text{if } w = q^{r-1}. \end{cases}$$

Proof Let d be a configuration with interval domain $[j, k]$, and let

$$p_q = (\ldots, (q^{r-1}, i), \ldots, (q^{r-1}, j)).$$

We set $L'_w = L^j_w - \{p_q\}$. We will give a weight preserving bijection from L'_w to P_w which maps p to p'. Let $p = (\ldots, (w_i, i), \ldots, (w_j, j))$ be an element of L'_w, where $w_j = w$. Let $h \leq j$ be the greatest integer such that for every $i \leq h$, we have $w_i = q^{r-1}$. Then we set $p' = (q^{r-1}, w_{h+1}, \ldots, w_j)$. This is clearly an injective mapping, and it is also surjective since by the choice of h, $w_{h+1} \neq q^{r-1}$. It is also weight preserving since the edges in p until the vertex (w_h, h) have all weight 1. Since $g_d(p_q) = 1$ 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 Lemma 7 it is sufficient to compute $g(P_w)$ for $w \in \Sigma^{r-1}$. The main difficulty in this computation is that the paths of P_w are defined by a constraint which forces the second vertex to be different from q^{r-1}. The solution we propose codes this constraint directly in the graph, which we will augment by one vertex for this purpose. Then we compute the total path weight from i to j for all vertices i, j. To do this, we will adapt a standard algorithm which constructs the regular expression associated to a finite state automaton.

Let $G'_l = (V', E', g')$ where $V' = V \cup \{sq^{r-2}\}$ for a letter $s \notin \Sigma$,

$$E' = E \cup \{(sq^{r-2}, q^{r-2}y) : y \in \Sigma \backslash \{q\}\},$$

and $g'(e) = g(e)$ for all edges $e \in E$ and $g'((sq^{r-2}, q^{r-2}y)) = g((q^{r-1}, q^{r-2}y))$. This graph is illustrated in figure 4. For every $w \in \Sigma^{r-1}$, let P'_w be the set of all paths in G'_l from sq^{r-2} to w. Clearly there is a weight-preserving bijection between P'_w and P_w.

Figure 4: The graphs G'_l (left-hand) and G'_r (right-hand), associated to the LQCA QFLIP. From these graphs we can compute $\vec{l} = \binom{1}{1}$ and $\vec{r} = \binom{1}{0}$.

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 $\sum_{e'=0}^{\infty} c^{e'}$, that is

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

Let $\{v_1, v_2, \ldots, v_{|V'|}\}$ be 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 have index less or equal to k. Let $W_k(i,j)$ denote $g(P_k(i,j))$. Then we claim that $W_k(i,j)$ satisfies the following recursion for $1 \leq i, j \leq |V'|$, and $1 \leq k \leq |V'|$:

$$\begin{aligned} W_0(i,j) &= \begin{cases} g'((v_i, v_j)) & \text{if } (v_i, v_j) \in E', \\ 0 & \text{otherwise,} \end{cases} \\ W_k(i,j) &= W_{k-1}(i,j) + W_{k-1}(i,k) \cdot (W_{k-1}(k,k))^* \cdot W_{k-1}(k,j). \end{aligned}$$

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.

Assume that this equation is true for $k-1$. We note that for every path of $P_k(i,j)$ there exists a unique integer e such that vertex v_k appears exactly e times the path. Thus we can write

$$P_k(i,j) = P_{k-1}(i,j) \cup \bigcup_{e=1}^{\infty} P_{k-1}(i,k) \otimes \underbrace{P_{k-1}(k,k) \otimes \cdots \otimes P_{k-1}(k,k)}_{e-1} \otimes P_{k-1}(k,j),$$

where the unions are disjoint and $\otimes$ is the path composition operator defined in section 4. By induction hypothesis we have

$$W_k(i,j) = W_{k-1}(i,j) + \sum_{e'=0}^{\infty} \left(W_{k-1}(i,k) \cdot (W_{k-1}(k,k))^{e'} \cdot W_{k-1}(k,j) \right),$$

which concludes the induction.

This proves the correction of the following algorithm: Let $m = |V'| = |\Sigma|^{r-1}$. Initialize W_0. For $k = 1, \ldots, m$ compute W_k using W_{k-1}. Finally output the border vector $\vec{l}$ defined by $\vec{l}(w) = W_m(sq^{r-2}, w)$ for $w \neq q^{r-1}$ and $\vec{l}(q^{r-1}) = W_m(sq^{r-2}, q^{r-1}) + 1$. Proceed in similar fashion for $\vec{r}$. The complexity of the algorithm is $O(|\Sigma|^{3(r-1)}) = O(n^{\frac{3(r-1)}{r+1}})$. $\square$

6 Closed affine subspace

In this section we will give a polynomial algorithm for the following problem.

Closed Affine Subspace

Input: Two vectors $\vec{l}, \vec{r} \in \mathbb{R}^m$ such that $\langle \vec{l} | \vec{r} \rangle = 1$ and a set of linear transformations $\mathcal{M} = \{M_a : a \in \Sigma\}$ in $\mathbb{R}^m$, where $m = |\Sigma|^{r-1}$.

Question: Is $H_{\vec{r}} + \vec{l}$ closed for $\vec{l}$ under $\mathcal{M}$, i.e. for all $b \in \Sigma^*$ do we have $M_b \vec{l} \in H_{\vec{r}} + \vec{l}$?

We set $t = |\Sigma|$. For the simplicity of notation, let $H = H_{\vec{r}} + \vec{l}$ and let $E_i = \mathcal{M}^i(\{\vec{l}\})$. Since $H = [H]$, we have $E_i \subseteq H$ if and only if $[E_i] \subseteq H$. Therefore we have to decide if $[\bigcup_{i=0}^{\infty} E_i] \subseteq H$. Dimension arguments imply the existence of a fixpoint, a set E_j, such that $[E_j] = [\bigcup_{i=0}^{\infty} E_i]$. Moreover we need only to keep track of a basis of $[E_i]$, that is a set B_i of linearly independent vectors, with $[B_i] = [E_i]$.

Theorem 4 *There exists an algorithm which decides if H is closed for $\vec{l}$ under $\mathcal{M}$ in time $O(n^{\frac{3r-2}{r+1}})$.*

Proof We claim that this is realized by the following algorithm:

$B_0 := \{\vec{l}\}$
$i \ := 1$
while $[B_i] \neq [B_i \cup \mathcal{M}(B_i)]$
  $B_{i+1} :=$ a basis of $[B_i \cup \mathcal{M}(B_i)]$
  $i := i + 1$
$B := B_i$
if $B \subseteq H$ accept
else reject

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

We show $[B_i] = [E_i]$ for every $i \geq 0$ by induction. The statement holds by definition for $i = 0$. Suppose $[E_i] = [B_i]$ for some i. Since $\mathcal{M}$ contains only linear operators, we have for any set S, $[\mathcal{M}(S)] = [\mathcal{M}([S])]$. Therefore $[\mathcal{M}(E_i)] = [\mathcal{M}(B_i)]$ which implies $[E_{i+1}] = [B_{i+1}]$.

Since B is a fixpoint, that is $[B] = [B \cup \mathcal{M}(B)]$, this implies that $[B] = [\bigcup_{i=0}^{\infty} E_i]$ and the correctness follows.

We now turn to the analysis of the complexity. We will build inductively the basis so that for all i, $B_i \subseteq B_{i+1}$. At the i-th iteration, to build B_{i+1}, initially we set $B_{i+1} = B_i$. Then we compute every vector in $\mathcal{M}(B_i \backslash B_{i-1})$ and add it to B_{i+1} if it is not in the affine subspace generated by B_{i+1}. At the end of the algorithm, these steps were applied to all vectors $M\vec{b}$ for $M \in \mathcal{M}$ and $\vec{b} \in B$, thus at most $|\mathcal{M}| \cdot |B| = O(tm)$ times. Computing $M\vec{b}$ takes $O(m^2)$ with standard matrix multiplication, and checking affine independence takes also $O(m^2)$ with the algorithm described in the next section. Thus the overall complexity is $O(tm^3)$. The theorem follows since $t = n^{\frac{1}{r+1}}$ and $m = n^{\frac{r-1}{r+1}}$. □

7 Maintaining a basis

In this section we give a dynamic algorithm for the following problem. We want to maintain a basis B of a d-dimensional linear subspace in R^m, such that the following requests for a given vector $\vec{u} \in \mathbb{R}^m$ can be treated efficiently:

Membership query: "Is $\vec{u} \in \langle B \rangle$?"

Add to basis: Replace B by $B \cup \{\vec{u}\}$.

We can define the problem for affine subspaces as well. Fortunately the latter can easily be reduced to the former: Let $f : \mathbb{R}^m \to \mathbb{R}^{m+1}$ be the function which maps a vector $\vec{u}$ to $\vec{u}'$ with $u'_i = u_i$ for $i = 1, \dots, m$ and $u'_{m+1} = 1$. Then every vector $\vec{v}$ satisfies $\vec{v} \in [B]$ if and only if $f(\vec{v}) \in \langle f(B) \rangle$.

A solution to this problem requires a tricky data structure to encode the basis. The naive way would be to represent B by a matrix such that its column vectors are exactly those of B and to apply the Gaussian elimination algorithm (see for example [11]) to check whether $\vec{u} \in \langle B \rangle$. This would require $O(md^2)$ time steps.

Transforming the matrix in an upper triangular form is the bottleneck of this approach. The representation we choose for B will improve this complexity.

Theorem 5 *There is a dynamic algorithm for maintaining a basis of a d-dimensional subspace in $\mathbb{R}^m$ which treats each request in time $O(m(m-d))$.*

Proof We represent a non empty basis B by the couple (T, B), where $T \in \mathbb{R}^{m \times m}$ is an orthogonal matrix and $\langle T(B) \rangle = \mathbb{R}^d \times \{0\}^{m-d}$. The empty basis is represented by $(I, \emptyset)$, where I is the identity matrix.

Since $T(\langle B \rangle) = \langle T(B) \rangle$, $\vec{u} \in \langle B \rangle$ if and only if $T\vec{u} \in \mathbb{R}^d \times \{0\}^{m-d}$. Thus verifying $\vec{u} \in \langle B \rangle$ is reduced to checking if the last $m - d$ components of $T\vec{u}$ are all 0 which can be done in time $O(m(m-d))$.

Suppose $\vec{u} \notin \langle B \rangle$. We will show that there is a orthogonal matrix M affecting only components from $d+1$ to m which satisfies $(MT\vec{u})_{d+1} \neq 0$ and $(MT\vec{u})_i = 0$ for $i = d+2, \dots, m$. Thus $\langle MT(B) \rangle = \langle T(B) \rangle$ and $\langle MT(B \cup \{\vec{u}\}) \rangle = \mathbb{R}^{d+1} \times \{0\}^{m-(d+1)}$. Therefore $(MT, B \cup \{\vec{u}\})$ represents $B \cup \{\vec{u}\}$.

We define M as the composition of two operators M_1 and M_2 we describe now. By hypothesis $\vec{u} \notin \langle B \rangle$, therefore there exists an index $k \in \{d+1, \dots, m\}$ such that $(T\vec{u})_k \neq 0$. Define M_1 to be the permutation matrix which exchanges k and $d+1$.

Let $\vec{u}' = M_1 T \vec{u}$. Note that $\vec{u}'_{d+1} \neq 0$. Then for an arbitrary vector $\vec{v}$ we define $(M_2\vec{v})_i = \vec{v}_i$ for $i = 1, \dots, d+1$ and $(M_2\vec{v})_i = \vec{v}_i - \vec{v}_{d+1}\vec{u}'_i / \vec{u}'_{d+1}$ for $i = d+2, \dots, m$. Clearly M_1 and M_2 are orthogonal linear operators, and since

$$M_2 \vec{u}' = (u'_1, u'_2, \dots, u'_{d+1}, 0, \dots, 0)$$

M satisfies the required property. We can compute $M_2 M_1 T$ in time $O(m(m-d))$, which concludes the proof. □

8 Putting all together

We are now able to prove Theorem 1, that is to give an algorithm to decide if a given LQCA is unitary. By Theorem 2 to solve this problem we have to compute the associated border vectors and decide the corresponding Closed Affine Subspace problem. According to Theorem 3 the border vectors can be computed in time $O(n^{\frac{3(r-1)}{r+1}})$, and due to Theorem 4 the last problem can be solved in time $O(n^{\frac{3r-2}{r+1}})$, which concludes the proof.

9 Conclusion

A not necessarily simple LQCA can be transformed into a simple one with the same time evolution operator. Let the original neighborhood be $N = (a_1, \ldots, a_r)$. 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 $O(n^{e\frac{3r-1}{r+1}}) = O(n^{3e})$.

In the case of space-periodic configurations van Dam [17] has shown that LQCAs can be efficiently simulated by QTMs. Watrous [18] gave an equivalent result for partitioned LQCAs. This question remains still open for the model of this paper.

Acknowledgments

We are thankful to Laurent Rosaz, Huong LêThanh and Umesh Vazirani for several helpful conversations. We wish also to thank the anonymous referees for helpful comments and in particular for simplifying section 6.

References

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

[2] ——, *Quantum mechanical Hamiltonian models of Turing machines that dissipate no energy*, Physical Review Letters, 48 (1982), pp. 1581–1585.

[3] E. Bernstein and U. Vazirani, *Quantum complexity theory*, SIAM Journal on Computing, 26 (1997), pp. 1411–1473.

[4] M. Biafore, *Can computers have simple Hamiltonians?*, in Proceedings of the Workshop on Physics and Computation, 1994, pp. 63–69.

[5] D. Deutsch, *Quantum theory, the Church-Turing principle and the universal quantum computer*, Proceedings of the Royal Society of London, A400 (1985), pp. 97–117.

[6] ——, *Quantum computational networks*, in Proceedings of the Royal Society of London, vol. A425, 1989, pp. 73–90.

[7] C. Dürr, H. LêThanh, and M. Santha, *A decision procedure for well-formed linear quantum cellular automata*, Random Structures & Algorithms, 11 (1997), pp. 381–394.

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

[9] ——, *Quantum mechanical computers*, Foundations of Physics, 16 (1986), p. 507.

[10] P. Høyer, *Note on linear quantum cellular automata.* manuscript, 1996.

[11] A. Lenstra and H. L. Jr., *Handbook of Theoretical Computer Science*, vol. A, Elsevier Science Publishers and The Mit Press, 1990, ch. Algorithms in number theory, pp. 683–685.

[12] S. Lloyd, *A potentially realizable quantum computer*, Science, 261 (1993), pp. 1569–1571.

[13] ——, *Envisioning a quantum supercomputer*, Science, 263 (1994), p. 695.

[14] N. Margolus, *Complexity, Entropy and the Physics of Information*, Addison-Wesley, 1994, ch. Parallel quantum computation.

[15] K. Sutner, *De Bruijn graphs and linear cellular automata*, Complex Systems, 5 (1991), pp. 19–30.

[16] K. Sutner and W. Maas, *Motion planning among time-dependent obstacles*, Acta Informatica, 26 (1988), pp. 93–122.

[17] W. van Dam, *A universal quantum cellular automaton*, in Proceedings of the Fourth Workshop on Physics and Computation, 1996, pp. 323–331.

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

[19] A. Yao, *Quantum circuit complexity*, Proceedings of the 34th IEEE Symposium on Foundations of Computer Science, (1993), pp. 352–361.