A decision procedure for well-formed linear quantum cellular automata*

Christoph Dürr, Huong LêThanh
Université Paris-Sud, LRI, Bât. 490
91405 Orsay Cedex, France
e-mail: {durr,huong}@lri.fr, http://www.lri.fr/∼durr

Miklos Santha†
CNRS, URA 410, Université Paris-Sud, LRI, Bât. 490
91405 Orsay Cedex, France
e-mail: santha@lri.fr

Abstract

In this paper we introduce a new quantum computation model, the linear quantum cellular automaton. Well-formedness is an essential property for any quantum computing device since it enables us to define the probability of a configuration in an observation as the squared magnitude of its amplitude. We give an efficient algorithm which decides if a linear quantum cellular automaton is well-formed. The complexity of the algorithm is $O(n^2)$ in the algebraic model of computation if the input automaton has continuous neighborhood.

key words: quantum computation, cellular automata, de Bruijn graphs

1 Introduction

In order to analyze the complexity of algorithms, computer scientists usually choose some computational model, implement the algorithm on it and count the number of steps as a function of the size of the input. Different models, such as Turing machines (TM), random access machines, circuits, or cellular automata can be used. They are all universal in the sense that they can simulate each other with only a polynomial overhead. However, these models are based on classical physics, whereas physicists believe that the universe is better described by quantum mechanics.

Feynman [13, 14] and Benioff [4, 5] were the first who pointed out that quantum physical systems are apparently difficult to simulate on classical computers, suggesting that there may be a gap between computational models based on classical physics and models based on quantum mechanics. Deutsch [10] introduced the first formal model of quantum computation, the quantum Turing machine (QTM). He also described a universal simulator for QTMs with an exponential overhead. More recently, Bernstein and Vazirani constructed a universal QTM with only a polynomial simulation overhead [7].

The power of QTMs was compared to that of classical probabilistic TMs in a sequence of papers [17, 11, 2, 7]. The most striking evidence that QTMs can indeed be more powerful than probabilistic TMs was obtained by Shor[22], who built his work on an earlier result of Simon [21]. Shor has shown that the problems of computing the discrete logarithm and factoring can be

*This research was supported by the ESPRIT Working Group 7097 RAND
†and by the French-Hungarian Research Program "Balaton" No. 94026 of the Ministère des Affaires Etrangères

efficiently solved on a QTM, whereas no polynomial time algorithm is known for these problems on a probabilistic TM.

Other quantum computational models were also studied. Yao [26] has defined the quantum version of the Boolean circuit model, and has shown that QTMs working in polynomial time can be simulated by polynomial size quantum circuits. Also, physicists were interested in quantum cellular automata: Biafore [6] considered the problem of synchronization, Margolus [20] described space-periodic quantum cellular automata and Lloyd [18, 19] discussed the possibility to realize a special type of quantum linear cellular automaton (LQCA). However these models are somehow different from the model of LQCA we consider in this article, and the physical realizability of our model has not yet been studied.

Well-formedness is an essential notion in quantum computation. A quantum computational device is at any moment of its computation in a superposition of configurations, where each configuration has an associated complex amplitude. If the device is observed in some superposition of configurations then a configuration in the superposition will be chosen at random. The probability a configuration will be chosen with is equal to the squared magnitude of its amplitude. Therefore it is essential that superpositions of unit norm be transformed into superpositions of unit norm, or equivalently, that the time evolution operator of the device preserve the norm. This property is called the well-formedness. In the case of a QTM, Bernstein and Vazirani gave easily checkable local constraints on the finite local transition function of the machine which were equivalent to its well-formedness. The existence of such relatively simple, local criteria is due to the local nature of the evolution of a TM: during a transition step only a fixed number of elements can be changed in a configuration.

In this paper we will define formally linear quantum cellular automata and will give an efficient algorithm which decides if an LQCA is well-formed. Our algorithm is of complexity $O(n^2)$ if the input LQCA has continuous neighborhood (most papers in the literature in the classical context deal only with such automata). The problem of well-formedness in the case of an LQCA is much harder than in the case of a QTM. One cannot hope for local conditions on the local transition function as in the case of a QTM, since the transitions of a linear cellular automaton are global: a priori no constant bound can be given on the number of cells which are changing states in a step. It turns out that well-formedness is related to the reversibility of linear classical cellular automata. Thus our work is closely related to the decision procedure for reversible linear cellular automata of Sutner [23].

In fact, quantum mechanics imposes an even stronger constraint on any quantum computational device: its time evolution operator has to be unitary. For QTMs [7], space-periodic LQCAs [9] and partitioned LQCAs [25] well-formedness implies unitarity, but not for the model of LQCAs we consider here. Building on the present algorithm we gave in a subsequent paper [12] an efficient procedure which decides if the evolution operator of a LQCA is unitary.

Watrous [25] has considered a subclass of LQCAs, partitioned linear quantum cellular automata. He has shown that a QTM can be simulated by a machine from that class with constant slowdown, and conversely, a partitioned LQCA can be simulated by a QTM with linear slowdown. The efficient simulation of a general LQCA by a QTM is left open in his paper. As it is shown by Watrous, the problem of well-formedness in the case of a partitioned LQCA is easy. The local transition function of a partitioned LQCA can be described by a finite dimensional complex square matrix, and the automaton is well-formed if and only if this finite matrix preserves the norm. No analogous result is known in the case of a general LQCA.

Our paper is organized as follows. In section 2 we first define linear cellular automata and give the basic notions of quantum computation in a finite space. Then we describe quantum linear cellular automata, define the notion of well-formedness, and prove that the inner product of two successor superpositions of configurations can be reduced to the inner product of two finite tensors. In section 3 first we give an example which shows that the trivial sufficient condition on the finite local transition function is not necessary for well-formedness. Then we describe the decision procedure for well-formed quantum linear cellular automata, prove its correctness, and analyze its complexity. The procedure consists of two separate algorithms, one which checks the unit norms, and another which checks the orthogonality of the column vectors of the infinite dimensional time evolution matrix of the automaton. In section 4 we describe a few open problems and finally in the appendix we give a shorter proof of one of the main theorems of Watrous' paper.

2 The computation model

2.1 Linear cellular automata

A *linear cellular automaton* (LCA) is a 4-tuple $A = (\Sigma, q, N, \delta)$. The *cells* of the automaton are organized in a line and are indexed by $\mathbb{Z}$. Σ is a finite non-empty set of *(cell-)states*. At every step of the computation, each cell is in a particular state. The *neighborhood* $N = (a_1, \ldots, a_r)$ is a strictly increasing sequence of signed 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 $i + a_1, \ldots, i + a_r$. We call $r = |N|$ the *size* of the neighborhood. Cells are simultaneously changing their states at each time step according to the states of their neighbors. This is described by the *local transition function* $\delta : \Sigma^{|N|} \to \Sigma$. If at a given step the neighbors of a cell are respectively in states $x_1, \ldots, x_r$ then at the next step the state of the cell will be $\delta(x_1, \ldots, x_r)$. The state $q \in \Sigma$ of A is the distinguished *quiescent* state, which satisfies by definition $\delta(q, \ldots, q) = q$.

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. The *support* of a configuration c is $supp(c) = \{i \in \mathbb{Z} : c_i \neq q\}$. A configuration c will be called *finite* if it has a finite support. We are dealing only with LCA's which work on finite configurations. Therefore from now on by *configuration* we will mean *finite configuration*. The set of configurations will be denoted $\mathcal{C}_A$.

The local transition function induces a *global transition function*, $\Delta : \mathcal{C}_A \to \mathcal{C}_A$, mapping a configuration to its *successor*. For every configuration c, and for every integer i, we have by definition

$$[\Delta(c)](i) = \delta(c_{i+N}),$$

where $\delta(c_{i+N})$ is a short notation for $\delta(c_{i+a_1}, \ldots, c_{i+a_r})$.

Configurations will often be represented by finite functions. We call an *interval* a finite subset of consecutive integers $[j, k] = \{j, j+1, \ldots, k\}$ of $\mathbb{Z}$ for any j and k (if $j > k$ this defines the empty interval $\emptyset$). For our purposes it will be convenient to deal with representations whose domains are intervals. Therefore for a configuration c, and for an interval I, let c_I be the restriction of c to I. Also, let $idom(c)$, the *interval domain* of c, be the smallest interval which contains $supp(c)$. For an interval $I = [j, k]$ with $j \leq k$, we define $ext(I)$, the *extension* of I (with respect to the neighborhood N) as the interval $[j - a_r, k - a_1]$. The extension of $\emptyset$ is $\emptyset$. If $I = idom(c)$ then the support of its successor $\Delta(c)$ is contained in $ext(I)$. Clearly, for every configuration c and intervals I and I', if $idom(c) \subseteq I$ and $ext(idom(c)) \subseteq I'$ then c_I and $\Delta(c)_{I'}$ specify respectively c and $\Delta(c)$.

We will call an LCA *simple* if the elements of its neighborhood form an interval, that is $a_r - a_1 = r - 1$. In the literature LCA's are often by definition simple.

A LCA is *trivial* if its neighborhood consist of a single cell. We can suppose without loss of generality that this single neighbor is the cell itself, that is $N = (0)$.

2.2 Basic notions of quantum computation

Let E be a finite set and let us consider the complex vector space $\mathbb{C}^E$ with the usual inner product which is defined for vectors $u, v \in \mathbb{C}^E$ by

$$\langle u, v\rangle = \sum_{e \in E} u(e) \cdot \overline{v(e)}.$$

The vectors in $\mathbb{C}^E$ will be called *superpositions* over E, and for a superposition u and an element $e \in E$, we will say that $u(e)$ is the *amplitude* of e in that superposition. The norm $\|u\|$ of a superposition u defined by this inner product is

$$\|u\| = \sqrt{\sum_{e \in E} |u(e)|^2} = \sqrt{\langle u, u \rangle}.$$

Two superpositions u and v are *orthogonal*, in notation $u \perp v$, if $\langle u, v \rangle = 0$. A superposition is *valid* if it has unit norm. If a valid superposition u over the set E is *observed* then one of the element of E will be chosen randomly and will be returned as the result of this observation. The probability that the element e is returned is $|u(e)|^2$. After the observation the superposition u is changed into the trivial superposition in which e has amplitude 1 and all the other elements 0.

Let I be an interval, and for each $i \in I$, let u_i be a superposition over E. The *tensor product* $\otimes_{i \in I} u_i$ is a superposition over E^I, that is an element of the complex vector space $\mathbb{C}^{E^I}$, where by definition, for all $x \in E^I$,

$$\left[\bigotimes_{i \in I} u_i\right](x) = \prod_{i \in I} u_i(x_i).$$

For our purposes the useful property of this operator is that the inner product of two tensors is the product of the respective inner products. Indeed, since I is finite, we have

$$\left\langle \bigotimes_{i \in I} u_i, \bigotimes_{i \in I} v_i \right\rangle = \prod_{i \in I} \langle u_i, v_i \rangle. \tag{1}$$

2.3 Linear quantum cellular automata

A linear quantum cellular automaton differs from a classical one in the sense that the automaton evolves on a superposition of configurations. The local transition function δ maps the state vector of a neighborhood into a superposition of new states, giving the amplitude with which a cell moves into a specific state given the state of its neighbors.

A *linear quantum cellular automaton* (LQCA) is a 4-tuple $A = (\Sigma, q, N, \delta)$, where the states set Σ and the neighborhood N are as before. It is called *simple* if the integers in N form an interval. The *local transition function* is $\delta : \Sigma^{|N|} \to \mathbb{C}^\Sigma$ such that for every $(x_1, \ldots, x_r) \in \Sigma^r$, we have $\|\delta(x_1, \ldots, x_r)\| > 0$. The distinguished *quiescent* state $q \in \Sigma$ satisfies for all $x \in \Sigma$

$$[\delta(q, \ldots, q)](x) = \begin{cases} 1 & \text{if} \quad x = q, \\ 0 & \text{if} \quad x \neq q. \end{cases}$$

Cells are again simultaneously changing their states at time steps but the outcome of the changes is not unique. If the neighbors of a cell are respectively in states $x_1, \ldots, x_r$ then at the next step, the cell will be in a superposition of states, where for every $y \in \Sigma$, the state of the cell will be p with amplitude $[\delta(x_1, \ldots, x_r)](y)$.

The local transition function induces a global one, which maps a superposition of configurations into its *successor* superposition. We call it the linear *time evolution operator* $U_A : \mathcal{C}_A \times \mathcal{C}_A \to \mathbb{C}$. For every $c, d \in \mathcal{C}_A$, the automaton enters d from c in one step with amplitude

$$U_A(d, c) = \prod_{i \in \mathbb{Z}} [\delta(c_{i+N})](d_i).$$

This infinite product is well-defined since we deal with finite configurations, so for all but a finite number of integers i, $c_{i+N} = q^r$ and $d_i = q$. Therefore in the product only a finite number of terms can be different from 1. Moreover if there is an i such that $c_{i+N} = q^r$ and $d_i \neq q$ then $U_A(d, c) = 0$. Thus in order to have non-zero transition amplitude it is necessary that $idom(d)$ be contained in $ext(idom(c))$.

Let I be any interval which contains $ext(idom(c))$. Then by the previous observations and by definition of tensor product we have

$$U_A(d,c) = \begin{cases} \left[\bigotimes_{i\in I} \delta(c_{i+N})\right](d_I) & \text{if } idom(d) \subseteq I, \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, superpositions of configurations form the Hilbert space defined by

$$\ell_2(\mathcal{C}_A) = \left\{ u \in \mathbb{C}^{\mathcal{C}_A} : \sum_{c\in\mathcal{C}_A} u(c)\cdot\overline{u(c)} < \infty \right\},$$

with the inner product defined for $u_1, u_2 \in \mathbb{C}^{\mathcal{C}_A}$ by

$$\langle u_1, u_2\rangle = \sum_{c\in\mathcal{C}_A} u_1(c)\cdot\overline{u_2(c)}.$$

As usual, u_1 and u_2 are *orthogonal* (in notation $u_1 \perp u_2$) if $\langle u_1, u_2\rangle = 0$.

As in the finite case, a superposition v of configurations is *valid* if $\|v\| = \sqrt{\langle v, v\rangle} = 1$. Also, as in the finite case, if an LQCA is *observed* in a valid superposition of configurations v, the result of the observation will be the configuration c with probability $|v(c)|^2$. Immediately after the observation whose outcome is c, the automaton will change its superposition into the classical one which gives amplitude 1 to c and 0 to all the others.

We want to have valid superpositions of configurations at each moment of the computation in order to associate the above probabilities to an observation. The initial configuration of the automaton is clearly valid. Therefore we say that the LQCA A is *well-formed* if its time evolution operator U_A preserves the norm.

It is not hard to see that U_A preserves the norm if and only if its column vectors are *orthonormal*, that is they have *unit norms* and they are *pairwise orthogonal*. We will denote the column vector of index c by $U_A(\cdot, c)$. In the next chapter we will give an algorithm which decides if the column vectors of U_A are orthonormal. An important technical tool in the correctness of the algorithm will be the generalization of equality (1) to successor superpositions of configurations in the infinite Hilbert space. This is stated in the following lemma.

Lemma 1 *Let c and c' be configurations and let I be an interval such that $ext(idom(c)) \cup ext(idom(d)) \subseteq I$. Then we have*

$$\langle U_A(\cdot, c), U_A(\cdot, c')\rangle = \prod_{i\in I} \langle \delta(c_{i+N}), \delta(c'_{i+N})\rangle.$$

Proof

$$\begin{aligned}
\langle U_A(\cdot, c), U_A(\cdot, c')\rangle &= \\
&= \sum_{d\in\mathcal{C}_A} U_A(d,c)\cdot\overline{U_A(d,c')} && (2)\\
&= \sum_{\substack{d\in\mathcal{C}_A,\\ supp(d)\subseteq I}} \left[\bigotimes_{i\in I}\delta(c_{i+N})\right](d_I)\cdot\overline{\left[\bigotimes_{i\in I}\delta(c'_{i+N})\right](d_I)} && (3)\\
&= \sum_{d'\in\Sigma^I} \left[\bigotimes_{i\in I}\delta(c_{i+N})\right](d')\cdot\overline{\left[\bigotimes_{i\in I}\delta(c'_{i+N})\right](d')} && (4)\\
&= \left\langle \bigotimes_{i\in I}\delta(c_{i+N}), \bigotimes_{i\in I}\delta(c'_{i+N}) \right\rangle && (5)\\
&= \prod_{i\in I}\langle\delta(c_{i+N}), \delta(c'_{i+N})\rangle. && (6)
\end{aligned}$$

The equations are justified in the following manner: (2) by definition of the inner product, (3) by the choice of I, (4) by identification of d_I with d', (5) by definition of the tensor product and (6) by equation (1). $\square$

We have the immediate corollary:

Corollary 1 *Let c be a configuration and let I be an interval such that $ext(idom(c)) \subseteq I$. Then we have*

$$\|U_A(\cdot, c)\| = \prod_{i \in I} \|\delta(c_{i+N})\|.$$

3 A decision procedure for well-formed LQCAs

3.1 Trivial LQCAs

It is easy to give sufficient and necessary conditions for the well-formedness of a trivial LQCA which are easily checkable on the local transition function.

Lemma 2 *Let $A = (\Sigma, q, (0), \delta)$ be a trivial LQCA. Then A is well-formed if and only if for every $x, y \in \Sigma$ with $x \neq y$*

$$\delta(x) \perp \delta(y), \tag{7}$$

and for every $x \in \Sigma$

$$\|\delta(x)\| = 1. \tag{8}$$

Proof For every $x \in \Sigma$ let c^x be the configuration which is x at cell 0 and quiescent elsewhere. Then for every $x, y \in \Sigma$ we have $\langle \delta(x), \delta(y) \rangle = \langle U_A(\cdot, c^x), U_A(\cdot, c^y) \rangle$. Thus if A is well-formed conditions (7) and (8) hold.

For the converse suppose that both conditions are satisfied. Then corollary 1 implies that the columns of U_A have unit norm. Now we show that for any two distinct configurations c and c', the associated columns of the evolution operator are orthogonal. Since c and c' are different there exist a cell i, such that $c_i \neq c'_i$. Thus $\delta(c_i) \perp \delta(c'_i)$ by condition (7) and $U_A(\cdot, c) \perp U_A(\cdot, c')$ by lemma 1. $\square$

For non-trivial LQCAs condition (7) can never hold since when $|N| > 1$ we can not have $|\Sigma|^{|N|}$ independent vectors in a space of dimension $|\Sigma|$.

But condition (8) still implies that the column vectors have unit norm by corollary 1. The following example shows that this condition is not necessary.

Let $B = (\{q, p\}, q, (0, 1), \delta)$ be an LQCA with the local transition function defined as follows. For $x \in \{q, p\}$, we define the superposition $|x\rangle$ over $\{q, p\}$ by

$$|x\rangle(y) = \begin{cases} 1 & \text{if} \quad x = y, \\ 0 & \text{if} \quad x \neq y. \end{cases}$$

Then δ is defined as:

$$\begin{array}{rclrcl} \delta(q,q) & = & |q\rangle, & \delta(q,p) & = & \frac{1}{2}|q\rangle, \\ \delta(p,q) & = & 2|p\rangle, & \delta(p,p) & = & |p\rangle. \end{array}$$

In every configuration the number of pairs qp is equal to the number of pairs pq, therefore for all configurations c, d we have

$$U_B(d, c) = \begin{cases} 1 & \text{if} \quad c = d, \\ 0 & \text{if} \quad c \neq d. \end{cases}$$

Thus the time evolution matrix U_B is just the identity, and B is well-formed. However, $\delta(q, p)$ and $\delta(p, q)$ do not have unit norm.

Nevertheless we can always transform a well-formed LQCA $A = (Q, q, N, \delta)$ into an LQCA $A' = (Q, q, N, \delta')$ such that $U_A = U_{A'}$ and A' satisfies condition (8). We simply renormalize the local transition function for all $w \in \Sigma^{|N|}$ by defining $\delta'(w) = \delta(w)/\|\delta(w)\|$. Then for every configurations c, d and interval I containing $ext(idom(c))$ and $idom(d)$ we have

$$\begin{aligned}
U_{A'}(d, c) &= \left[\bigotimes_{i \in I} \delta'(c_{i+N})\right](d_I) \\
&= \prod_{i \in I} [\delta'(c_{i+N})](d_i) \\
&= \prod_{i \in I} \frac{[\delta(c_{i+N})](d_i)}{\|\delta(c_{i+N})\|} \\
&= \frac{\prod_{i \in I} [\delta(c_{i+N})](d_i)}{\prod_{i \in I} \|\delta(c_{i+N})\|} \\
&= \frac{\prod_{i \in I} [\delta(c_{i+N})](d_i)}{\|U_A(\cdot, c)\|} \\
&= \frac{\prod_{i \in I} [\delta(c_{i+N})](d_i)}{1} \\
&= U_A(d, c).
\end{aligned}$$

The following lemma establishes a particular property of trivial LQCAs which is not true in general.

Lemma 3 *Let $A = (\Sigma, q, (0), \delta)$ be a trivial LQCA. If A is well-formed then U_A is unitary.*

Proof Suppose A is well-formed. By the previous lemma δ is described by a unitary matrix. Let δ^{-1} be the local function described by the inverse of this matrix, that is for all $x, y \in \Sigma$ we have $[\delta^{-1}(y)](x) = \overline{[\delta(x)](y)}$. Let A' be the trivial LQCA $(\Sigma, q, (0), \delta^{-1})$. Clearly $U_{A'} U_A = U_A U_{A'} = I$, which concludes the proof. □

3.2 The algorithm

Before giving the algorithm, let us discuss the size of the input, that is the size of an LQCA $A = (\Sigma, q, N, \delta)$. It is clearly dominated by the size of the description of δ. We will work in the algebraic computational model, where by definition complex numbers take unit space, arithmetic operations and comparisons take unit time. Then δ can be given by a table of size $|\Sigma|^{r+1}$, when the neighborhood is of size $|N| = r$. Therefore we define the *size* of the automaton $n = |\Sigma|^{r+1}$, and we will do the complexity analysis of our algorithm as a function of n.

Our main theorem is an immediate consequence of Theorems 4 and 5.

Theorem 2 *There exists an algorithm P which takes a simple LQCA as input, and decides if it is well-formed. The complexity of the algorithm is $O(n^2)$.*

What can we say about the well formedness of an LQCA which is not necessarily simple? Let $A = (\Sigma, q, N, \delta)$ be an LQCA of size n whose neighborhood is $N = (a_1, \ldots, a_r)$. We can transform A into a simple LQCA $A' = (\Sigma, q, N', \delta')$ such that A and A' have the same time evolution operator. This can be done by taking as neighborhood $N' = (a_1, a_1+1, a_1+2, \ldots, a_r)$, and making the local transition function δ' independent from the new neighbors in N'. Then we can run P on A'.

The size of A' will depend also on another parameter, on the *span* s of A which is defined as $s = a_r - a_1 + 1$. Since $|N'| = s$, the size of A' will be $n' = |\Sigma^{s+1}| = n^{(s+1)/(r+1)}$. Let us define the *expansion factor* e of A as $e = (s+1)/(r+1)$. Then the time taken by P will be $O(n'^2) = O(n^{2e})$. We have therefore the following corollary:

Corollary 3 *There exists an algorithm which takes an LQCA with expansion factor e as input, and decides if it is well-formed. The complexity of the algorithm is $O(n^{2e})$.*

3.3 Unit norms of column vectors

In this chapter we will give an algorithm which decides if the column vectors of the time evolution operator have unit norms. Let $A = (\Sigma, q, N, \delta)$ be a simple LQCA whose neighborhood is of size r. We define an edge weighted directed de Bruijn graph $G_A = (V, E, w)$ with vertex set $V = \Sigma^{r-1}$, edge set $E = \{(xz, zy) : x, y \in \Sigma, z \in \Sigma^{r-2}\}$ and with weight function $w : E \to \mathbb{R}$ defined by $w((xz, zy)) = \|\delta(xzy)\|$. The unweighted version of this graph was defined by Sutner in [23]. A *path* is a sequence $p = (v_0, \ldots, v_k)$ of vertices such that for $0 \le i \le k-1$, we have $(v_i, v_{i+1}) \in E$. The *weight* $w(p)$ of a the path p is

$$\prod_{0 \le i \le k-1} w((v_i, v_{i+1})).$$

We call the path $(v_0, \ldots, v_k)$ a *cycle* if $v_0 = v_k$ and $k > 0$. If in addition, $v_0 = q^{r-1}$ then it is called a *q-cycle*. Our algorithm is based on the following lemma.

Lemma 4 *The column vectors of U_A have unit weight if and only if the weight of all q-cycles in G_A is 1.*

Proof Let T denote the set of q-cycles of G_A. We define a mapping $M : \mathcal{C}_A \to T$. Let c be a configuration with interval domain $I = [j, k]$. Let $t = k - j$, and for $i = 0, 1, \ldots, k - j$, let $x_i = c_{j+i}$. Then by definition

$$M(c) = (q^{r-1}, q^{r-2}x_0, q^{r-3}x_0x_1, \ \ldots \ , x_0x_1\ldots x_{r-2}, \ \ldots \ , x_tq^{r-2}, q^{r-1}).$$

We have then

$$\begin{aligned} \|U_A(\cdot, c)\| &= \|\delta(q^{r-1}x_0)\| \cdot \|\delta(q^{r-2}x_0x_1)\| \cdot \ldots \cdot \|\delta(x_tq^{r-1})\| && \text{by corollary 1} \\ &= \|\delta(q^r)\| \cdot \|\delta(q^{r-1}x_0)\| \cdot \ldots \cdot \|\delta(x_tq^{r-1})\| \cdot \|\delta(q^r)\| \\ &= w(M(c)). && \text{by definition} \end{aligned}$$

Since the mapping M is clearly surjective the statement of the lemma follows. □

Verifying if all column vectors of U_A are of unit norm is now reduced to checking if all q-cycles in G_A are of unit weight. The algorithm we give now will just do that.

Theorem 4 *There exists an algorithm R which takes a simple LQCA $A = (\Sigma, q, N, \delta)$ as input, and decides if the column vectors of the time evolution operator U_A have all unit norm. The complexity of the algorithm is $O(n^2)$.*

Proof Algorithm R will construct the graph G_A of lemma 4 and then determines if it has a q-cycle of weight different from 1. This will be done by two consecutive algorithms R_1 and R_2, from which the first will check if there is a column of norm less than 1, and the second will check if there is a column of norm greater than 1. They are both modifications of the Bellman-Ford single source shortest paths algorithm [3, 15, see also [8]] (BF for short), when q^{r-1} is taken for the source. They are based on the fact that BF detects negative cycles going through the source. (Actually for our purposes any shortest paths algorithm can be used which uses sum and min as arithmetic operations, and which detects negative cycles. Floyd's algorithm would be another example).

Algorithm R_1 replaces every sum operation in BF by a product operation, and initializes the shortest path estimate for the source to 1 (the shortest path estimates for the other vertices are initialized to ∞ as in BF), and then runs it on G_A. This way it computes the shortest paths when the weight of a path is defined as the product of the edge weights. To see this let $G_A{}'$

be the same graph as G_A except the edge weights are replaced by their logarithm. Then the weight of a shortest path in ${G_A}'$ given by BF will be the logarithm of the shortest path in G_A given by R_1. For the same reason, negative cycles in ${G_A}'$ through the source will correspond to q-cycles in G_A with weight less than 1 which will therefore be detected by R_1.

Algorithm R_2 replaces every min operation in R_1 by max and the default initial shortest path estimate ∞ by 0, and then runs it on G_A. This way it computes the shortest paths when the weight of a path is defined as the product of the reciprocal of the edge weights. If we define ${G_A}'$ with negative logarithm edge weights then negative cycles in ${G_A}'$ will correspond to cycles in G_A with weight greater than 1 and will be detected by R_2.

The complexity of BF is $O(|V| \cdot |E|)$. In the graph G_A we have $|V| = |\Sigma|^{r-1}$. Every vertex has $|\Sigma|$ outgoing edges, therefore $|E| = |\Sigma|^r$. Thus the complexity of the algorithm R is $O(|\Sigma|^{2r-1}) = O(n^2)$. $\square$

In [16] Høyer gave a linear time algorithm to decide if the column vectors have all unit norm, improving the complexity of our result.

3.4 Orthogonality of column vectors

Now we will build an algorithm which decides if the column vectors of the time evolution matrix are orthogonal. Let again $A = (\Sigma, q, N, \delta)$ be a simple LQCA whose neighborhood is of size r. We define the graph $H_A = (V, E)$ with vertex set $V = \Sigma^{r-1} \times \Sigma^{r-1}$ and edge set

$$E = \{ \quad ((x_1 z_1, x_2 z_2), (z_1 y_1, z_2 y_2)) : \\ x_1, x_2, y_1, y_2 \in \Sigma, z_1, z_2 \in \Sigma^{r-2}, \delta(x_1 z_1 y_1) \not\perp \delta(x_2 z_2 y_2) \quad \}.$$

For a path $p = ((u_0, v_0), \ldots, (u_k, v_k))$ of H_A, let $p_1 = (u_0, \ldots, u_k)$, and $p_2 = (v_0, \ldots, v_k)$. Clearly, p_1 and p_2 are paths in G_A. A cycle is called here a *q-cycle* if its first vertex is (q^{r-1}, q^{r-1}).

Lemma 5 *The column vectors of U_A are orthogonal if and only if $p_1 = p_2$ for every q-cycle p in H_A.*

Proof Let $L = \{(c, c') \in \mathcal{C}_A \times \mathcal{C}_A : U_A(\cdot, c) \not\perp U_A(\cdot, c')\}$, and let T denote the set of q-cycles. We will define a mapping $M : L \to T$. For $(c, c') \in L$, let $I = [j, k]$ be an interval such that $ext(idom(c)) \cup ext(idom(c')) \subseteq I$. Let $t = k - j$, and for $i = 0, 1, \ldots, k - j$ we define $x_i = c_{j+i}$, and $y_i = c'_{j+i}$. Then by definition

$$M(c, c') = ((q^{r-1}, q^{r-1}), (q^{r-2} x_0, q^{r-2} y_0), \ \ldots \ , (x_t q^{r-2}, y_t q^{r-2}), (q^{r-1}, q^{r-1})).$$

Since $U_A(\cdot, c) \not\perp U_A(\cdot, c')$, lemma 1 implies that $M(c, c')$ is indeed a q-cycle in H_A. Also, it is clear that M is surjective. Finally $c \neq c'$ if and only if $M(c, c')_1 \neq M(c, c')_2$ since both are equivalent to the existence of $i \in I$ such that $x_i \neq y_i$. $\square$

We can now affirm:

Theorem 5 *There exists an algorithm S which takes a simple LQCA $A = (\Sigma, q, N, \delta)$ as input, and decides if the column vectors of the time evolution operator U_A are orthogonal. The complexity of the algorithm is $O(n^2)$.*

Proof The algorithm S constructs the graph H_A and computes the strongly connected component of the node (q^{r-1}, q^{r-1}). By lemma 5 there exists two distinct configurations such that the corresponding column vectors in U_A are not orthogonal if and only if in this component there is a vertex (u, v) with $u \neq v$. This can be checked easily.

Finding the strongly connected components in a graph can be done in time $O(|E|)$ for example with Tarjan's algorithm [24]. In H_A the size of number of vertices is $|V| = |\Sigma|^{2(r-1)}$. Since every vertex has outdegree $|\Sigma|^2$, the number of edges is $|E| = |\Sigma|^{2r}$. Therefore the complexity of the algorithm S is $O(|\Sigma|^{2r}) = O(n^2)$. $\square$

4 Conclusion

It would be interesting to generalize results concerning reversibility of a linear classical CA for the well-formedness of an LQCA. For example a necessary condition for reversibility is the notion of *balancedness* of the local transition function [1], which means that every state has the same number of preimages. How does balancedness generalizes to the quantum model?

It remains open, as stated also by Watrous, whether a QTM can simulate an LQCA with reasonable slowdown.

Partitioned linear quantum cellular automata

This appendix treats a special kind of LQCA, the partitioned LQCA, which was the main topic of Watrous' paper [Wat95]. Our aim is to provide a new, shorter proof to one of his results, based on our approach.

A *partitioned linear quantum cellular automaton* (PLQCA) is a LQCA $A = (\Sigma, q, N, \delta)$, which satisfies the following restrictions:

1. The state-set Σ is the Cartesian product $\Sigma_1 \times \cdots \times \Sigma_r$ of some finite non-empty sets Σ_i, $i \in \{1, \ldots, r\}$.
2. The local transition function $\delta : \Sigma^r \to \mathbb{C}^\Sigma$ is the composition of two functions, the *classical part* $\delta_p : \Sigma^r \to \Sigma$ and the *quantum* part $\delta_Q : \Sigma \to \mathbb{C}^\Sigma$. For all $x_{i,j} \in \Sigma_j$, $i, j \in \{1, \ldots, r\}$, δ_p is defined by

$$\delta_p((x_{1,1}, \ldots, x_{1,r}), (x_{2,1}, \ldots, x_{2,r}), \ldots, (x_{r,1}, \ldots, x_{r,r})) = (x_{1,1}, x_{2,2}, \ldots, x_{r,r}).$$

The function δ_p defines a LCA $A_p = (\Sigma, N, \delta_p)$ whose global transition function Δ_p is a permutation on configurations such that for all $c \in \mathcal{C}_A$ and $i \in \mathbb{Z}$,

$$[\Delta_p(c)](i) = \delta_p(c_{i+N}).$$

Moreover, the time evolution operator U_{A_p} of A_p is a unitary matrix since for all $c, d \in \mathcal{C}_A$, we have

$$U_{A_p}(d, c) = \begin{cases} 1 & \text{if } \Delta_p(c) = d, \\ 0 & \text{otherwise.} \end{cases}$$

The *local transition matrix* Q is the complex valued matrix, indexed by Σ, defined for all states $x, y \in \Sigma$ by

$$Q(y, x) = [\delta_Q(x)](y).$$

In fact Q completely determines the local transition function δ.

The function δ_Q defines a trivial LQCA $A_Q = (\Sigma, q, (0), \delta_Q)$, with the time evolution operator U_{A_Q}. Clearly, $\mathcal{C}_A$ and q are respectively the set of configurations and the quiescent state also of A_p and A_Q. It turns out that unitarity of the local transition matrix is equivalent to the unitarity of the time evolution operator, as stated in the following theorem.

Theorem 6 ([Wat 95, theorem 3.1 and corollary 3.1]) *Let A be a PLQCA, U_A its time evolution operator and Q its local transition matrix. Then the following statements are equivalent.*

1. *Q is unitary.*
2. *A is well-formed.*
3. *U_A is unitary.*

Proof The local transition function of A is the composition of two separate local transition functions, thus its time evolution operator is also the composition of time evolution operators of the associated LQCAs, that is $U_A = U_{A_Q} U_{A_p}$. Since U_{A_p} is unitary we have that U_A preserves the norm (resp. is unitary) if and only if U_{A_Q} preserves the norm (resp. is unitary).

The theorem follows from lemmas 2 and 3. □

Acknowledgements

We are thankful to Stéphane Boucheron, Bruno Durand, Richard Jozsa and John Watrous for several helpful conversations.

References

[1] S. Amoroso and Y. Patt, Decision Procedures for Surjectivity and Injectivity of Parallel Maps for Tessellation Structures, *Journal of Computer and System Sciences* **6**, 448–464, 1972.

[2] A. Berthiaume and G. Brassard, The Quantum Challenge to Structural Complexity Theory, *Proceedings of the 7th IEEE Conference on Structure in Complexity Theory*, 132–137, 1992.

[3] R. Bellman, On a routing problem, *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.

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

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

[6] M. Biafore, Can Computers Have Simple Hamiltonians? *Proceedings of the 3rd Workshop on Physics and Computation*, 63–69, 1994.

[7] E. Bernstein and U. Vazirani, Quantum complexity theory, to appear in *SIAM Journal on Computing*, 1997. A preliminary version has appeared in *Proceedings of the 25th ACM Symposium on the Theory of Computing*, 11–20, 1993.

[8] T. Cormen, C. Leiserson and R. Rivest, Introduction to Algorithms, *The MIT Press*, 1990.

[9] W. van Dam, A universal quantum cellular automaton, *Fourth Workshop on Physics and Computation*, 1996.

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

[11] D. Deutsch and R. Jozsa, Rapid solution of problems by quantum computation, *Proceedings of the Royal Society of London*, A439:553–558, 1992.

[12] C. Dürr and M. Santha, A decision procedure for unitary linear quantum cellular automata, *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science*, 38–45, 1996.

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

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

[15] L. Ford and D. Fulkerson, Flows in Networks, *Princeton University Press*, 1962.

[16] P. Høyer, Note on linear quantum cellular automata, *manuscript*, `u2pi@imada.ou.dk`, 1996.

[17] R. Jozsa, Characterizing classes of functions computable by quantum parallelism, *Proceedings of the Royal Society of London*, A435:563–574, 1991.

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

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

[20] N. Margolus, Parallel Quantum Computation, *Complexity, Entropy and the Physics of Information*, Addison-Wesley, 273, 1994.

[21] D. Simon, On the Power of Quantum Computation, *Proceedings of the 34th IEEE Symposium on Foundations of Computer Science*, 116–123, 1994.

[22] P. Shor, Algorithms for Quantum Computation: Discrete Log and Factoring *Proceedings of the 26th ACM Symposium on the Theory of Computing*, 124–134, 1994.

[23] K. Sutner, De Bruijn graphs and cellular automata, *Complex Systems*, 5:19–30, 1991.

[24] R. Tarjan, Depth first search and linear graph algorithms, *SIAM Journal on Computing*, 1(2):146–160, 1972.

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

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