Local unitary quantum cellular automata

Carlos A. Pérez-Delgado and Donny Cheung
Institute for Quantum Computing, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
(Received 11 May 2007; revised manuscript received 25 July 2007; published 20 September 2007)

In this paper we present a quantization of cellular automata. Our formalism is based on a lattice of qudits and an update rule consisting of local unitary operators that commute with their own lattice translations. One purpose of this model is to act as a theoretical model of quantum computation, similar to the quantum circuit model. It is also shown to be an appropriate abstraction for space-homogeneous quantum phenomena, such as quantum lattice gases, spin chains, and others. Some results that show the benefits of basing the model on local unitary operators are shown: universality, strong connections to the circuit model, simple implementation on quantum hardware, and a wealth of applications.

DOI: 10.1103/PhysRevA.76.032320 PACS number(s): 03.67.Lx

I. INTRODUCTION

The cellular automaton (CA) is a computational model that has been studied for many decades [1,2]. It is a simple yet powerful model of computation that has been shown to be Turing complete [2]. It is based on massive parallelism and simple, locally constrained instructions, making it ideal for various applications. In particular, CA are very effective at simulating many classical physical systems, including gas dispersion, fluids dynamics, ice formation, and even biological colony growth [3]. Although usually simulated in software, CA hardware implementations have also been developed. All of these characteristics make CA a strong tool for moving from a physical system in nature, to a mathematical model, to an implemented physical simulation.

More recently, the idea of *quantum* cellular automata (QCA) has emerged. Several theoretical mathematical models have been proposed [4–8]. However, there is a lack of applications developed within these models. On the other hand, *ad hoc* models for specific applications like quantum lattice gases [9,10], among others [11], have been developed. Several proposals for scalable quantum computation (QC) have been developed that use ideas and tools related to QCA [12–16]. Some of these have been shown to be capable of universal computation [17,18]. Other QCA tools have been used to solve, or propose solutions to, particular problems in physics [19–23].

However, there does not exist a comprehensive model of QCA that encompasses these different views and techniques. Rather, each set of authors defines QCA in their own particular fashion. In short, there is a lack of a generally accepted QCA model that has all the attributes of the CA model mentioned above: simple to describe; computationally powerful and expressive; efficiently implemented in quantum software *and* hardware; and able to efficiently and effectively model appropriate physical phenomena.

The purpose of this paper is to propose such a model. The model we present here is based on intuitive and well-established ideas: qudits as the basic building blocks (cells), and local unitary operators as the basic evolution method (local update rule).

The choice of local unitary operators as the basic evolution operator ensures that the model is simple and easily explained to anyone familiar with the field of quantum information. However, the choice is not made merely for sake of simplicity: it provides us with an *efficient implementation* of QCA on quantum hardware, while still enjoying an expressive richness strong enough to simulate any appropriate physical system.

Formally, what we mean by efficient implementation is that there exists a uniform family of quantum circuits that can each simulate the evolution of a finite region of the QCA, for a specified number of steps. Furthermore, we require that the *depth* of each circuit be strictly linear in the number of steps, and constant on the size of the region being simulated. This last requirement is to ensure that the QCA retains the quintessential quality of CA: *massive parallelism*.

We will refer to this formalization as the local unitary quantum cellular automata (LUQCA) model, when we need to make the distinction from other formal definitions of QCA.

In Sec. IV we will see how any QCA properly defined in the model presented here can be efficiently implemented. The fact that there is such a guarantee, without any further restraints, is one of the strongest features of the model herein presented. In Sec. VIII we will see that, in general, previous models cannot make such a guarantee. We will also discuss what methods can be used to translate the QCA in these models into the LUQCA model.

We will see in Secs. IV and V how insisting on efficient implementations does not at all limit the expressive power of our QCA model. Section V will also show how most, if not all, physical systems of interest with the proper characteristics—time and space homogeneity—can be modeled using local unitary QCA. We will also prove computational completeness in Sec. IV. Section VIII discusses how valid QCA presented in other models can be rephrased in the local unitary QCA scheme.

We begin in Sec. II by briefly describing classical CA in detail. Following that, we will endeavor to quantize this model in the most natural way possible. The rest of the paper presents results pertaining to the strengths of this model.

II. CELLULAR AUTOMATA

In the classical model of cellular automata, we begin with a finite set of states Σ and an infinite lattice of *cells*, each of

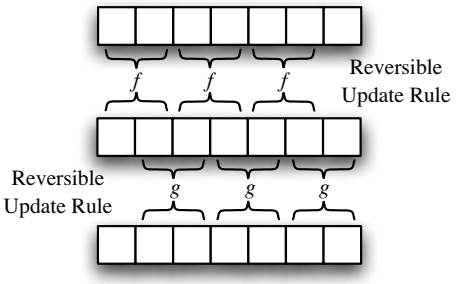


FIG. 1. Partitioned cellular automaton.

which is in one of the states in Σ. We have discrete time steps, and at each time step t the state of the lattice evolves according to some rule. This rule gives the state of each cell at time $t+1$ as a function of the states of the cells in its *neighborhood*, which is simply a finite set of cells corresponding to a particular cell.

Definition 1 (CA). A cellular automaton is a 4-tuple $(L,\Sigma,\mathcal{N},f)$ consisting of a d-dimensional lattice of cells indexed by integers $L=\mathbb{Z}^d$, a finite set Σ of cell states, a finite neighborhood scheme $\mathcal{N}\subseteq\mathbb{Z}^d$, and a local transition function $f:\Sigma^{\mathcal{N}}\to\Sigma$.

The transition function f simply takes, for each lattice cell position $x\in L$, the states of the neighbors of x, which are the cells indexed by the set $x+\mathcal{N}$ at the current time step $t\in\mathbb{Z}$ to determine the state of cell x at time $t+1$. There are two important properties of cellular automata that should be noted. First, cellular automata are *space homogeneous*, in that the local transition function performs the same function at each cell. Also, cellular automata are *time homogeneous*, in that the local transition function does not depend on the time step t.

We may also view the transition function as one which acts on the entire lattice, rather than on individual cells. In this view, we denote the state of the entire CA as a *configuration* $C\in\Sigma^L$ which gives the state of each individual cell. This gives us a *global* transition function which is simply a function that maps $F:\Sigma^L\to\Sigma^L$.

Reversible and partitioned CA

As a first step toward developing a theory of unitary CA we will revisit the theory of classical reversible automata.

A CA is reversible if for any configuration $C\in\Sigma^L$ and time step $t\in\mathbb{Z}$ there exists a unique predecessor configuration C' such that $C=F(C',t)$. It is known that any Turing machine can be simulated using a reversible CA [24], so no computational power is lost by this restriction.

One method that is used to construct reversible cellular automata is *partitioning*. In a partitioned CA, the transition function is composed of local, reversible operations on individual units of a partition of the lattice. (See Fig. 1.)

In order to formally define partitioned CA, we must expand the definition of cellular automata, as partitioned CA are neither time homogeneous nor space homogeneous in general. They are, however, periodic in both space and time, and thus we set both a time period $T\geqslant 1$ and a space period, given as a d-dimensional sublattice S of $L=\mathbb{Z}^d$. The sublattice S can be defined using a set $\{v_k:k=1,\ldots,d\}$ of d linearly independent vectors from $L=\mathbb{Z}^d$ as

$$S=\left\{\sum_{k=1}^{d}a_k v_k : a_k\in\mathbb{Z}\right\}.$$

Definition 2. For a given fixed sublattice $S\subseteq\mathbb{Z}^d$, we define a block $B\subseteq\mathbb{Z}^d$ as a finite subset of $\mathbb{Z}^d$ such that $(B+s_1)\cap(B+s_2)=\varnothing$ for any $s_1,s_2\in S$ with $s_1\neq s_2$, and such that

$$\bigcup_{s\in S}(B+s)=\mathbb{Z}^d.$$

The main idea of the partitioned CA is that, at different time steps, we act on a different block partition of the lattice. We are now ready to formally define the partitioned CA.

Definition 3. A partitioned CA is a 6-tuple $(L,S,T,\Sigma,\mathbf{B},\mathcal{F})$ consisting of (1) a d-dimensional lattice of cells indexed by integers $L=\mathbb{Z}^d$; (2) a d-dimensional sublattice $S\subseteq L$; (3) a time period $T\geqslant 1$; (4) a finite set Σ of cell states; (5) a block scheme $\mathbf{B}$, which is a sequence $\{B_0,B_1,\ldots,B_{T-1}\}$ consisting of T blocks relative to the sublattice S; and (6) a local transition function scheme $\mathcal{F}$, which is a set $\{f_0,f_1,\ldots,f_{T-1}\}$ of reversible local transition functions which map $f_t:\Sigma^{B_t}\to\Sigma^{B_t}$.

At time step $t+kT$ for $0\leqslant t<T$ and $k\in\mathbb{Z}$, we perform f_t on every block B_t+s, where $s\in S$. In order to find the reverse of a partitioned CA, we simply give the reverse block scheme $\mathbf{B}=\{B_{T-1},\ldots,B_1,B_0\}$ and the reverse function scheme $\mathcal{F}=\{f_{T-1}^{-1},\ldots,f_1^{-1},f_0^{-1}\}$.

Although the partitioned CA is not time or space homogeneous, it can be converted into a regular CA, on the lattice S (which is isomorphic to $\mathbb{Z}^d$), with cell states Σ^B, where the new local transition function simulates T time steps of the partitioned CA in one time step.

In the original partitioned CA scheme as described by Margolus [25], the sublattice was fixed as $S=2\mathbb{Z}^d$, and the block scheme was fixed with two partitions: $B_0=\{(x_1,x_2,\ldots,x_d):0\leqslant x_j\leqslant 1\}$ and $B_1=\{(x_1,x_2,\ldots,x_d):1\leqslant x_j\leqslant 2\}$.

III. LOCAL UNITARY QCA

Now, with a formal notion of CA, we can proceed to give a *quantization*. As mentioned earlier, we will have very specific goals in mind.

A. Model requirements

First, we want to develop an intuitive model that is simple both to work with and to develop algorithms for. At the same time, we want this model to be an obvious *extension* of classical CA and to reduce to classical CA behavior under reasonable limits.

Second, we want to keep our model grounded in physical

realities. This has a couple of strong consequences; namely, we approach CA, even classical CA, not as abstract mathematical structures, but as models representing real physical systems. As a consequence, we expect our model to reliably model quantum systems with appropriate behavior, e.g., spin chains. Also, an algorithm described in our model should be easy to translate to an actual physical implementation on such quantum systems. We show in Sec. VI that this is so.

B. A first approach

The first step in our quantization of CA is to change the state space of a cell to reflect a quantum system. There are several methods for doing so; however, we believe that the most natural way to approach this is to convert the alphabet of the cellular automaton Σ into orthogonal basis states of a Hilbert space $\mathcal{H}_\Sigma$. Formally, every cell $x \in L$ is assigned a *qudit*, $|x\rangle \in \mathcal{H}_\Sigma$. This gives us a strong intuitive tool, as the notion of a lattice of qudits should be familiar to anyone working in quantum-information theory.

As we shall see, our approach is also physically grounded, in that it is possible to describe this model in terms of a quantum system evolving according to a Hamiltonian. As an example, spin chains can be directly described by such mathematical constructions. Lattice gases, though not originally modeled in this way, can also be easily described by such mathematical constructs. Perhaps the most obvious physical example is the pulse-driven quantum computer.

We also wish to quantize the standard classical CA update rule. However, this process cannot necessarily proceed in the most obvious manner. In a classical CA, every cell is instantaneously updated in parallel. We wish to replace this classical cell update rule with a quantum analog that acts appropriately on the qudit lattice described above. For a quantum unitary operation to act as a quantum cell update rule, this operator needs to satisfy the following two restrictions.

(1) The operator must act on a finite subset of the lattice. Precisely, $U_x : \mathcal{H}(\mathcal{N}_x) \to \mathcal{H}(\mathcal{N}_x)$ where $\mathcal{N}_x = \mathcal{N} + x \subseteq L$ is the finite neighborhood about the cell x.

(2) The operator must commute with lattice translations of itself. Precisely, we require that $[U_x, U_y] = 0$ for all $x, y \in \mathbb{Z}^n$.

The first condition is an immediate condition for any rule, quantum or otherwise, to qualify as a CA update rule. The second condition allows the operators U_x for $x \in \mathbb{Z}^n$ to be applied in parallel without the need to consider any special or particular ordering of the operators.

It should be clear that any evolution defined in such manner represents a valid quantum evolution which can be ascribed to some physical system. The global evolution of the lattice can be described as

$$U = \prod_x U_x,$$

which is well defined, due to the two conditions given above.

The question that remains is whether this model properly describes what we intuitively would regard as QCA. Properly, there are two questions: (1) Can all entities described by the model above be properly classified as QCA? (2) Can all systems that are identified as QCA be properly described in the model above?

The answer to the first question is *yes*, since the update rules are local and can be applied in unison throughout the lattice. Also, the global unitary operator for the evolution of the lattice is properly defined and space homogeneous, as desired.

The answer to the second question is, unfortunately, *no*. We now present a simple system that one might consider to be a valid QCA, but that cannot be described in the above model.

The counterexample is as follows. We start with a one-dimensional lattice of qudits. For each lattice cell $x \in L$, we associate with it a quantum state $|\psi_x\rangle \in \mathcal{H}_\Sigma$. Although, in general, the configuration of a QCA may not be separable with respect to each cell, the configuration can still be described in terms of a linear superposition of these separable configurations. Thus, it suffices to consider such configurations.

At each time step we wish to have every value shifted one cell to the right. In other words, after the first update each cell x should now store the state $|\psi_{x-1}\rangle$. After k steps each cell x should contain the state $|\psi_{x-k}\rangle$. In fact, such a transition function cannot be implemented by any local unitary process.

To see why this is so, suppose that we had a transition function f, which is the product of a finite number of operations, $f = f_n f_{n-1} \cdots f_1$, such that each operator f_j is the (potentially infinite) product of local unitary operators over disjoint neighborhoods. Note that this gives us the most general description possible of a depth-n quantum circuit implementation of this linear QCA using only local unitary operators. Now, consider an individual cell x_0. By analyzing the dependencies of the individual local unitary operators which make up the transition function f, it is possible to find a range of cells, $P = \{x : a \leqslant x \leqslant b\}$ for some $a, b \in \mathbb{Z}$ such that $x_0 \in P$, and the value of the quantum state at cell x_0 after the application of the transition function depends only on the cells of P.

We now divide the transition function f into two functions, $f = hg$, where g applies sufficiently many local unitary operators from f over the cells of P so that the new value of the quantum state at cell x_0 is computed without violating any of the dependency relations from f. Then, h simply applies the remainder of the local unitary operators, as appropriate. Note that, since g necessarily contains any local unitary operators from f that operate on the cell x_0, the operation h does not. Since h does not perform any operation between cells $x < x_0$ and cells $x > x_0$, in order to implement the shift-right transition function, the cells $\{x : x_0 < x \leqslant b\}$ must contain enough quantum information after g has been applied to reconstruct the information in the cells $\{x : x_0 \leqslant x \leqslant b\}$. This is clearly impossible.

In order to resolve this issue, we need to analyze the classical CA parallel update rules more closely. In the classical CA, the local update rule for a given cell reads the value of the cell and the values of its neighboring cells. It performs a computation based on these values, and then updates the cell's value accordingly. Herein lies the problem: *read* and *update* are modeled in a classical CA as a single atomic action that can be applied throughout the lattice in parallel simultaneously. However, in a physical setting, these

two operations cannot be implemented in this manner. When simulating CA in classical computer architectures, the canonical solution is to use two lattices in memory: one to store the current value, and one to store the computed updated value. Even if we consider hardware implementations of CA, these need to keep the values of the inputs to the transition function while this function is being calculated.

The formal CA model does not need to consider this implementation detail, as it is a mathematical construction and has no claims to directly model a physical system implementing a CA. When developing a QCA model, one cannot take the same liberty. The name itself, QCA, includes reference to an underlying *quantum* physical reality. It is our intention that this model faithfully, if abstractly, represents real physical systems. Although there is some value in mathematical constructions which do not correspond directly to any physical systems, this is not the goal of the constructions presented in this paper.

C. A different approach

We now make an adjustment to our QCA model, given the importance of maintaining independent read and update operations. Instead of having one unitary operator replacing the single atomic operation in the CA model, we define our QCA update rule as consisting of two unitary operators. The first operator, corresponding to the read operation, will be as defined above: a unitary operator U_x, $x \in L$, acting on the neighborhood $\mathcal{N}_x$, which commutes with all lattice translations of itself, U_y, $y \in L$. The second operator V_x, $x \in L$, corresponds to the update operation, and will act only on the single cell x itself.

The intuition is as follows: in our physical model, instead of having separate lattices for the read and update functions, we expand each lattice cell to also contain any space resources necessary for computing the updated value of the cell. The operator U_x reads the values of the neighborhood $\mathcal{N}_x$, performs a computation, and stores the new value in a way that does not prevent neighboring operators U_y from correctly reading its own input values. This allows each cell to be operated upon independently, in parallel, without any underlying assumptions of synchronization. After all the operations U_x have been performed, the second unitary V_x performs the actual update of the lattice cell.

With this model for the update operation, we can again approach the two questions given above as to whether this model adequately describes what we might intuitively regard as QCA.

First, it is clear that all entities described by this updated model can still be properly classified as QCA. The local update rule $R_x = V_x U_x$ is still a valid quantum unitary operation, and the global update rule

$$R = VU = \left(\bigotimes_{x} V_x \right) \left(\prod_{x} U_x \right)$$

is both well defined and space homogeneous.

Now, in order to properly investigate whether all physical systems that can be described as QCA can be described within this model, it is necessary to verify the following. We must first compare our model to existing CA models, both classical and quantum, in order to ensure that our model subsumes all proper CA described in these models. Second, we must also show that any known physical system that behaves according to quantum mechanics and satisfies the CA preconditions of being driven by a local, space-homogeneous interaction can be described by our model.

As an example, the qubit shift-right QCA mentioned above can now be described in this model, by including ancillary computation space with each lattice cell.

We will tackle this question in more depth in the upcoming sections. First, we present a formal definition of the QCA model which we will adopt, as described in this section.

Definition 4 (QCA). A quantum cellular automaton is a 5-tuple $(L, \Sigma, \mathcal{N}, U_0, V_0)$ consisting of (1) a d-dimensional lattice of cells indexed by integers $L = \mathbb{Z}^d$, (2) a finite set Σ of orthogonal basis states, (3) a finite neighborhood scheme $\mathcal{N} \subseteq \mathbb{Z}^d$, (4) a local read function $U_0 : (\mathcal{H}_\Sigma)^{\otimes \mathcal{N}} \to (\mathcal{H}_\Sigma)^{\otimes \mathcal{N}}$, and (5) a local update function $V_0 : \mathcal{H}_\Sigma \to \mathcal{H}_\Sigma$. The read operation carries the further restriction that any two lattice translations U_x and U_y must commute for all $x, y \in L$.

Each cell has a finite Hilbert space associated with it, $\mathcal{H}_\Sigma = \mathrm{span}(\{|x\rangle\}_{x \in \Sigma})$. The reduced state of each cell ρ_x is a density operator over this Hilbert space.

The initial state of the QCA is defined in the following way. Let f be any computable function that maps lattice vectors to pure quantum states in $(\mathcal{H}_\Sigma)^{\otimes k^d}$, where d is the dimension of the QCA lattice, and k is the length of a side of a d-dimensional hypercube, which we use to define blocks that are initialized to particular states. Then for any lattice vector $\mathbf{z} = (z_1 k, z_2 k, \ldots, z_d k) \in \mathbb{Z}^d$ the initial state of the lattice hypercube delimited by $(z_1 k, z_2 k, \ldots, z_d k)$ and $((z_1+1)k-1, (z_2+1)k-1, \ldots, (z_d+1)k-1)$ is set to $f(\mathbf{z})$.

Intuitively, each block represents a volume of the QCA that is initialized to a particular pure state. Each block is initialized independently. In particular, f can have a block size of one cell, initializing every cell to the same state in Σ. It can also have more complicated forms such as having every pair of cells in a one-dimensional QCA initialized to some maximally entangled state. Particularly useful are functions f that initialize a finite region about the origin to some interesting state—the input of the QCA—and the rest of the lattice to some quiescent state (see below).

The local update rule acting on a cell x consists of the operation U_x followed by the single-cell operation V_x. Both U_x and V_x are restricted to being computable unitary operators. The global evolution operator R is as previously defined.

D. Quiescent states

Our QCA definition follows the classical CA convention in defining the model over an *infinite* lattice. However, we will often be concerned only with finite regions of the QCA. One reason, for example, is that any physical implementation of a QCA using quantum hardware will, by necessity, simulate only a finite region of the QCA. Another reason is for simulating physical phenomena. For instance, in Sec. V, we

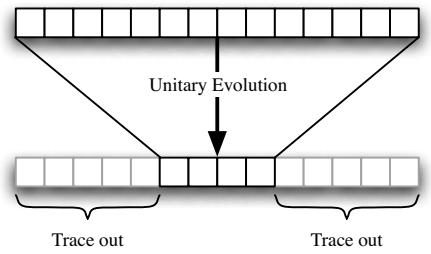


FIG. 2. Past light cone of a region S: This represents a one-dimensional local unitary QCA. In order to obtain the state of the region of interest, the dark region at the bottom, one must consider not just the region itself, but anything that might affect the state of the region within the course of the simulation: its past light cone. One may then trace out the unneeded regions.

will be interested in simulating finite-size chains of spin-$\frac{1}{2}$ particles.

Sometimes, it can be appropriate to simply use finite QCA with cyclic boundary conditions. In this case, we envision the lattice as a closed torus. This is a standard and well-known practice with CA. For example, we can use this technique if the spin chain we wish to simulate is *closed*, that is, it wraps around itself. For other applications, this will not be appropriate, for example, when trying to simulate an *open* spin chain. This is a chain which does not wrap around, but rather has two distinct end points. Another example will be the spin-signal amplification algorithm in Sec. VII, which uses a finite-size-cube ancilla system.

In such cases, the most appropriate way to proceed is to make use of a *quiescent* state, which is a cell state that is guaranteed to remain invariant under the update rule, regardless of the states of its neighbors. For instance, in the case of finite spin-$\frac{1}{2}$ chains, we can use three state cells. We use the state labels $|+1\rangle$ and $|-1\rangle$ to refer to the presence of a spin-$\frac{1}{2}$ particle in a given cell position in the states $\frac{1}{2}(\mathbb{1}+\sigma_z)$ and $\frac{1}{2}(\mathbb{1}-\sigma_z)$, respectively. A third state, labeled $|0\rangle$, denotes the absence of any particle in that cell location. One then need only ensure that the update rule correctly acts on states $|+1\rangle$ and $|-1\rangle$, while leaving state $|0\rangle$ unaffected.

Quiescent states are also very useful for the purposes of *simulation*, and physical implementation. Normally, if one were interested in the state of a region S of the lattice after k steps of the QCA update rule, one would need to look at the *past light cone* of S. If the local update rule has a neighborhood of radius r, then one needs to include kr additional cells in each direction beyond the border of S. This is because any information in the past light cone of S has the ability to affect cells within S, as shown in Fig. 2. Note that since the size of the region needed by the simulation is determined by the number of time steps of the QCA we wish to simulate, one needs to fix the number of steps in the simulation beforehand. However, if a given QCA has a quiescent state, and all cells outside the finite region under consideration are initialized to this quiescent state, then the simulation of this QCA need only include this region for any number of simulated time steps.

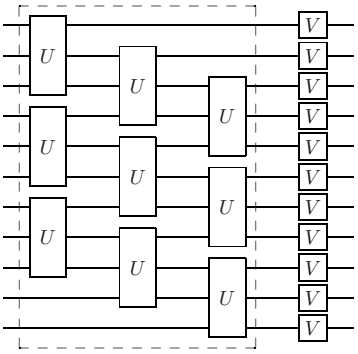


FIG. 3. Quantum circuit simulation of a QCA update step: The dotted area represents the read phase. A read operator U must be applied to each qubit, and its two neighbors. Since U commutes with its translations, we are at liberty to apply the U operators in any order. The update phase consists of the operator V being applied to every qubit.

IV. QUANTUM CIRCUITS AND UNIVERSALITY

In this section we explore two important aspects of the QCA model we introduced in Sec. III. These aspects relate to QCA as a model of computation. First, it is important to show that QCA are capable of universal quantum computation. We demonstrate this using a simulation of an arbitrary quantum circuit using a two-dimensional QCA.

We also show that any QCA can be simulated using families of quantum circuits. A quantum circuit is defined as a finite set of gates acting on a finite input. One can then define a uniform family of quantum circuits, with parameters S and t, such that each circuit simulates the finite region S of the QCA for t update steps. By uniformity we mean that there exists an effective procedure, such as a Turing machine, that on input (S,t) outputs the correct circuit.

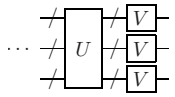
We will show that our simulation is efficient, as defined in Sec. I. Specifically, in order to simulate a QCA on a given region, for a fixed number of time steps, we give a quantum circuit simulation with a depth which is linear with respect to the number of time steps, and constant with respect to the size of the simulated region.

A. Simulation of QCA by quantum circuits

We begin by showing the latter of the two results described above. We proceed incrementally, showing first how to produce a quantum circuit that can simulate a single update step of a simple QCA.

Lemma 1. Any finite region of a one-dimensional QCA with a symmetric neighborhood of radius 1, where cells are individual qubits, can be simulated by a quantum circuit.

Proof. The simulation of an individual update step of this QCA is simple. Recall that the operators U_x, each acting on three qubits, all commute with each other. Therefore, the U_x operators may be applied in an arbitrary order. The operators V_x can all be applied to their respective qubits once all U_x operators have been applied. Figure 3 gives a visual representation of this construction. In order to simulate an arbitrary number of steps, we simply need to repeatedly apply the above construction. Finally, although we represented the



$$U = U_n U_{n-1} \ldots U_2 U_1$$

$$V = V_m V_{m-1} \ldots V_2 V_1$$

FIG. 4. Decomposition of general qudit gates.

operators U in our diagram as single, three-qubit operators, to complete the simulation we decompose U into an appropriate series of one and two qubit gates from a universal gate set. ■

In the case of one-dimensional QCA with a nearest-neighbor scheme, and cells consisting of one qubit, the operator U is simply acting on three qubits. Still this operator U needs to be decomposed into a series of one- and two-qubit gates $U=U_nU_{n-1}\cdots U_2U_1$ taken from a set of universal gates.

In order to extend the construction to allow cells with qudits of arbitrary dimension d, we first replace the single-qubit wires in Fig. 3 with qudit wires as in Fig. 4. Then each gate U_x and V_x is decomposed into one- and two-qubit gates as in the aforementioned figure. The same construction technique works in order to deal with arbitrary dimensions and arbitrary cell neighborhood sizes.

Note that m and n are constants, determined by the structure of the QCA. For very complicated QCA these constants can potentially be large. However, once the QCA has been defined these parameters are set, and hence do not asymptotically affect the complexity of simulating a region of the QCA for a particular length of time.

As our simulation above does not set a region size to be simulated, any region size can be simulated with an appropriate construction. An arbitrary number of time steps can be simulated by simply iterating the above construction. With this in mind, as well as the previous lemma, we can now state the following.

Theorem 1. For every QCA Q there exists a family of quantum circuits, parametrized by (S,t), each acting on $O(m \log|\Sigma|)$ inputs, and with circuit depth $O(t)$ which simulates a finite region S of Q consisting of $m=|S|$ cells, for t time steps.

This is a very important result, as it demonstrates that the local unitary QCA model does not admit automata which are somehow "not physical." More precisely, any behavior that can be described by a QCA can be described by the more traditional quantum circuit model. Furthermore, such descriptions retain the high parallelism inherent to QCA.

B. Simulation of quantum circuits by QCA

Next, we show the converse result from the one above, thus showing that local unitary QCA are capable of efficient universal quantum computation.

Theorem 2. There exists a universal QCA Q_u that can simulate any quantum circuit with at most a linear slowdown, by using an appropriately encoded initial state.

Proof. We proceed by constructing the QCA Q_u over a two-dimensional lattice. We will basically "draw" the circuit onto the lattice. The qubits will be arranged top to bottom, and the wires will be visualized as going from left to right.

Each cell will consist of a number of fields, or registers. The cell itself can be thought of as the tensor product of quantum systems corresponding to these registers.

The first register, the state register, consists of a single qubit that corresponds directly to the value on one of the wires of the quantum circuit at a particular point in the computation. This value will be shifted toward the right as time moves forward.

Next is the gate register. This register will be initialized to a value corresponding to a gate that is to be applied to the state register, at the appropriate time.

There is also a clock register, which will keep track of the current time step of the simulation. There are two phases to the simulation, an *operate* and a *carry* step.

There is finally a single-qubit-active register that keeps a record of which cells are currently actively involved in the computation. This register is either set to *true* or *false*.

The local read operator U_x proceeds as follows. The neighborhood scheme is the von Neumann neighborhood of radius 1, i.e., the cells directly above, below, and to either side of the cell. The read operator acts nontrivially only on the one cell directly above, and the one directly to the left. However, the bigger neighborhood is needed to ensure unitary evolution, and translation invariance.

If the clock register is set to operate, then a quantum gate is applied to the state register of the current cell (and possibly the state register of the upward neighbor). For this, we fix a finite set of universal gates consisting of the controlled phase gate and some set of single-qubit operators. The choice of the controlled phase gate, as opposed to, say, controlled-NOT, is to ensure that U_x commutes with translations of itself. Any one-qubit unitary gates that form a universal set will work.

If the clock register is set to carry, then the state register will be swapped with the state register of the left neighbor if and only if the following conditions occur: the active register is set to true on the left neighbor, and set to false on the current cell, and the clock register is set to carry on all the neighbors (above, below, and to either side). These extra checks are required to ensure the operator U_x commutes with translations of itself.

Figure 5 gives a visual representation of the update rule operator U_x. Operator V_x simply updates the clock register, applying a NOT gate at each time step.

Finally, the initial state is set as follows. There is one horizontal row for each wire in the quantum circuit. Every column represents a time step in the quantum circuit. The cells are initialized to have their gate registers set to the appropriate gate, if there is a gate, in the wire corresponding to its row, and in the time step corresponding to its column. The clock register is set to operate, and the state register is set to $|0\rangle$ initially on all cells. The first column of the quantum circuit is set to active, and all other cells are set to inactive.

This construction can only natively simulate circuits with nearest-neighbor gates. In order to encode arbitrary circuits, it is necessary to translate the circuit into one using only nearest-neighbor gates by adding swap gates where needed. This is the cause of the worst-case linear slowdown, mentioned in the statement of this theorem. ■

The previous result is important in that it proves that the QCA model is computationally complete. It also gives a

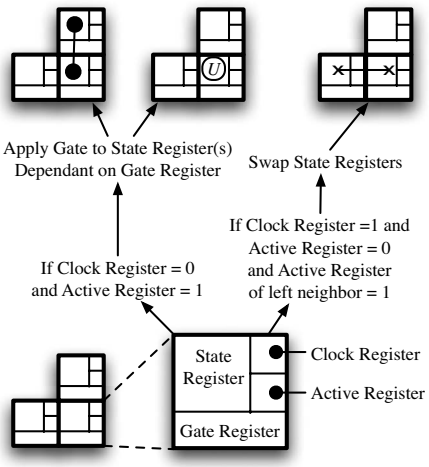


FIG. 5. Universal QCA update rule.

recipe for implementing quantum circuit algorithms on two-dimensional QCA. It is important to mention that it is also possible to show that one-dimensional LUQCA are universal for quantum computing. For a complete proof see [26].

In the following sections, by showing how physical systems can implement QCA, we complete a formula for implementing quantum algorithms on physical systems using QCA methods. We will see, however, that the strongest virtue of this QCA model lies not in its ability to simulate quantum circuits. Rather, it lies in the algorithms that take natural advantage of the QCA structure.

V. MODELING PHYSICAL SYSTEMS

We stated before that one of our goals in developing a QCA formalism is to create a useful modeling tool for quantum systems. Classical CA are used for simulating various phenomena based on classical information, such as sea ice formation, fluid dynamics, or voter systems [3,25]. Similarly, we expect QCA to be able to model different types of physical systems based on quantum information, with dynamics which are based on time and space homogeneous local interactions.

Physical systems that fall in this category include Ising and Heisenberg interaction spin chains, solid state NMR, and quantum lattice gases. We will be looking at some of these systems in this section.

A. Spin chains

Spin chains are perhaps the most obvious candidate for physical systems being modeled with QCA. Indeed, Ising interaction spin chains, and in general, any spin chain with a coupling Hamiltonian that commutes with its own lattice translations, can be implemented easily.

Suppose we have a linear spin chain of length N, indexed by $n \in \mathbb{Z}$. Each spin n is coupled to its nearest neighbor $n+1$, with a coupling Hamiltonian $J\sigma_z^{(n)}\sigma_z^{(n+1)}$, where J is the coupling strength constant. Note that the coupling Hamiltonian does commute with its lattice translations. The Hamiltonian for the entire spin chain is

$$H_{\mathrm{I}} = \sum_{n=1}^{N-1} J\sigma_z^{(n)}\sigma_z^{(n+1)}.$$

It is a simple matter to give a discrete-time approximation to such a spin chain. First, we fix a time step interval Δt. Our QCA model will allow for simulation of the spin chain for time steps in multiples of Δt. Hence, while the choice of Δt is arbitrary, it is important in determining the resolution of the simulation.

For a simulation of the Ising spin chain, the QCA lattice consists of a one-dimensional array, where each cell is a single qubit. The neighborhood of each cell n simply consists of the cell and its right neighbor $n+1$. The local rule operator U_n is given as

$$U_n = e^{-iJ\sigma_z^{(n)}\sigma_z^{(n+1)}\Delta t}.$$

The operator V_n is simply the identity operator. Note that the operator U_n commutes with its translations, that is, $[U_n, U_m]=0$, for all $n,m \in \mathbb{Z}$. Furthermore, the global operator

$$U = \prod_{n=1}^{N-1} U_n$$

satisfies

$$U = e^{-iH_{\mathrm{I}}\Delta t}.$$

Hence, the QCA construction faithfully simulates the Ising spin chain for times that are integer multiples of Δt, as desired.

A more complicated endeavor is to construct a QCA simulation of a spin chain whose coupling Hamiltonians do not commute with each other. In particular, we examine the Heisenberg spin chain as an example. Let the coupling Hamiltonian between spins n and $n+1$ be

$$H_{\mathrm{H}}^{(n,n+1)} = J(\sigma_x^{(n)}\sigma_x^{(n+1)} + \sigma_y^{(n)}\sigma_y^{(n+1)} + \sigma_z^{(n)}\sigma_z^{(n+1)} - \mathbb{1}\otimes\mathbb{1}).$$

Here, note that $H_{\mathrm{H}}^{(n,n+1)}$ does not commute with its translations $H_{\mathrm{H}}^{(m,m+1)}$. The Hamiltonian of the total system is

$$H_{\mathrm{H}} = \sum_{n=1}^{N-1} H_{\mathrm{H}}^{(n,n+1)}$$

A QCA simulation of the Heisenberg spin chain presented above is still possible, however, with the help of two powerful tools: *Trotterization* and *cell coloring*. The first technique is well known in physics; the second is a tool developed for QCA. Together, they allow for simulation of complicated and almost arbitrary Hamiltonians by QCA.

Trotterization is a technique by which a Hamiltonian is approximated using a combination of noncommuting Hamiltonians whose sum adds up to the original Hamiltonian. In other words, it is possible to approximate with bounded error the evolution due to the Hamiltonian $H=H_a+H_b$ by alter-

nately evolving the system under the Hamiltonians H_a and H_b even if these two do not commute. Precisely, we can give a first-order approximation

$$e^{-i(H_a+H_b)\Delta t} = (e^{-iH_a\Delta t/k}e^{-iH_b\Delta t/k})^k + \delta.$$

In the case that $\|[H_a,H_b]\|\Delta t^2 \ll 1$, the error δ is $O(\Delta t^2/k)$. Higher-order techniques can achieve error rates of $O(\Delta t^m + 1/k^m)$ at the cost of using $O(2^m)$ gates. Though the number of gates increases exponentially, the time required for each gate *decreases* exponentially as well.

In the case of our QCA simulation of the Heisenberg spin chain Hamiltonian H_{H} above, we have

$$H_a = \sum_{n=1}^{\left\lceil\frac{(N-1)}{2}\right\rceil} H_{\mathrm{H}}^{(2n-1,2n)}$$

and

$$H_b = \sum_{n=1}^{\left\lceil\frac{(N-1)}{2}\right\rceil} H_{\mathrm{H}}^{(2n,2n+1)}.$$

Note that $H_{\mathrm{H}}=H_a+H_b$. The Hamiltonians H_a and H_b consist of the couplings from the even spins to their left and right neighbors, respectively.

Our QCA evolution will consist of alternately evolving the lattice cells under H_a and H_b, using a technique called cell coloring. Each cell will have two fields. The first field is a state register, consisting of one qubit, which will hold the state of the spin represented by the cell. The second field, called the active color register, will also consist of a single qubit. Initially, the color register of each cell n is set to the value n mod 2.

The QCA lattice used in this simulation is also one dimensional, and the neighbor set of each cell includes both the cell to the immediate right, and the immediate left of the given cell. Let U_n' be the Trotter step acting on the current cell state register and the right-neighbor state register. Using the first-order approximation, we have

$$U_n' = e^{-iH^{(n,n+1)}\Delta t/k}$$

for an appropriate value k. It is also possible to use higher-order approximations.

The local update rule operator U_n then consists of applying the operator U_n' if and only if the current cell's active color register is set to 1, and both left and right neighbors have their color registers set to zero. The operator V_n simply toggles the active color register.

It should be clear that this QCA construction simulates the Heisenberg spin chain. Moreover, by using an appropriate operator U_n', it is possible to simulate any Hamiltonian with nearest neighbor couplings with this technique.

It is appropriate here to mention that one-dimensional spin structures such as these can be efficiently simulated using classical computers [27]. There are also efficient ways to calculate the lowest energy eigenstates and eigenvalues using classical numerical techniques such as the density matrix renormalization group method [28]. This, of course, also implies that the QCA presented in this section can be efficiently solved and simulated classically.

However, this conclusion cannot be easily generalized to larger classes of QCA. First, we note that we have used one-dimensional spin networks here for expository purposes. From our constructions, it should be clear that these QCA simulations generalize easily to higher dimensions. On the other hand, no efficient classical simulation is known for spin networks of dimension higher than 1.

Also, it is not known whether arbitrary one-dimensional LUQCA can be simulated efficiently classically. In fact, due to the universality of one-dimensional LUQCA [26], this will not be the case unless classical computers can efficiently simulate quantum systems ($BPP=BQP$), which is generally regarded as unlikely.

B. Quantum lattice gases

Quantum lattice gases have been studied for over a decade now [9,10,29–32]. In essence, they are the quantum analog of classical lattice gases. The basic principles are the same in both the classical and quantum cases: one starts with a discrete CA-based model that describes particles on the lattice, and their movement. One can then take the *continuous limit* of such a CA model and show that, in this limit, the behavior of the CA mimics a well-known differential equation.

Taking the continuous limit of a classical CA is a well-known procedure. It involves giving the lattice a physical interpretation, where each cell is thought to represent a point in space. The distance between two adjacent cells is taken to be Δx and each time step of the CA is assumed to take Δt time. One then takes the limit, in a well-prescribed manner, where $\Delta x \to 0$ and $\Delta t \to 0$. There exist classical CA whose continuous limits represent gas diffusion, as well as various other fluid dynamics [3].

In the quantum case, Meyer [9], and Boghosian and Taylor [10] give a construction of a quantum lattice gas whose continuous limit is the Schrödinger equation for a freely moving particle. We now show how any type of lattice gas can be represented under the local unitary QCA model.

We begin by introducing the quantum walk QCA Q_W. This QCA models multiparticle quantum walks on a lattice. Each cell is allowed to have *two* particles, in orthogonal states (these two states can be thought of as orthogonal spins). The lattice can have any number of particles in total.

The construction is as follows. The QCA Q_W is one dimensional. Each cell has two single-qubit registers, called *up* and *down*. Each register will represent the presence of a particle in the lattice site, with the appropriate spin, by being in the state $|1\rangle$, and the absence of the corresponding particle by being in the state $|0\rangle$.

The local update operator U_x acts on the down register of the current cell, and up register of the right neighbor, swapping the two values. Operator V operates on both fields of the cell with operator

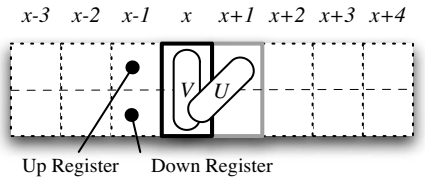


FIG. 6. Quantum walk on a lattice.

$$V_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & q & p & 0 \\ 0 & p & q & 0 \\ 0 & 0 & 0 & \phi \end{pmatrix},$$

where $p,q \in \mathbb{C}$ satisfying $|p|^2+|q|^2=1$, $p\bar{q}+\bar{p}q=0$, and $|\phi|^2=1$. The update rule is summarized in Fig. 6.

The dynamics of this QCA are the same as in the lattice gas described by Boghosian and Taylor in [10].

Let $\Psi_u(x,t)$ and $\Psi_d(x,t)$ be the amplitude corresponding to the presence of a particle with spin up and spin down, respectively, in cell position x, at time t. Let $\Psi(x,t)$ be the total amplitude corresponding to the presence of a particle in cell x at time t, that is $\Psi(x,t)=\Psi_u(x,t)+\Psi_d(x,t)$. Then, we have that

$$\Psi_u(x,t+\Delta t) = q\Psi_u(x-\Delta x,t) + p\Psi_d(x+\Delta x,t)$$

and

$$\Psi_d(x,t+\Delta t) = q\Psi_d(x+\Delta x,t) + p\Psi_u(x-\Delta x,t).$$

We can proceed according to Boghosian and Taylor [10], and take the continuous limit of our QCA $\Delta x^2 \to 0$ and $\Delta t \to 0$, using the Chapman-Enskog method [3]. Doing so reveals that $\Psi(x,t)$ obeys the equation

$$\frac{\partial}{\partial t}\Psi(x,t) = \frac{i}{2m}\frac{\partial^2}{\partial x^2}\Psi(x,t),$$

which is the equation for a freely moving particle of mass $m=ip/q$ in one dimension.

Using the same construction techniques, we can also describe a freely moving particle in two or three dimensions. We can construct QCA that simulate other quantum lattice gases like the ones proposed in [9]. Most, if not all, quantum lattice gases, whether single- or multiparticle, can be described as local unitary QCA.

This concludes our discussion on the expressive powers of the QCA model presented here. In the next section we continue with a discussion of how to take these mathematical models and implement them in quantum hardware.

VI. QUANTUM COMPUTATION

In previous sections, we discussed how our unitary QCA can be used to model physical systems, and how universal quantum computation can be accomplished using only QCA primitives. In this section we will look into bridging the gap by using QCA as a theoretical framework for implementing quantum computation.

A clear advantage of working in the QCA model over quantum circuits, in regard to physical implementations of quantum computation, is that QCA make considerably fewer demands on the underlying hardware. In particular, as opposed to direct implementations of quantum circuits, the global evolution of the lattice in the QCA model does not assume independent control over qudits. Rather, all qudits are to be addressed collectively in parallel. However, it should be noted that the models of cellular automata described in this paper do not explicitly address the issue of initialization. Any proposed physical realization of the QCA must also describe the set of initial states which are constructible. This may require some degree of nonglobal control over a physical apparatus, such as having individual cells initialized to a certain basis state, or it may require some interaction with the environment, such as having the lattice cooled to a ground state.

The QCA model also more closely resembles what is currently achievable in several current quantum computer implementations. For example, if qudits are represented by physical spins, and the control of such spins is achieved through the use of magnetic pulses, as is the case in NMR or electron spin resonance (ESR), then it is more reasonable to consider all spins as being subjected to the same pulse sequences, rather than having the ability to address spins individually. The same can be said about many other physical quantum computer proposals.

In this section we will concentrate on implementing QCA on NMR, since most of the groundwork for this implementation has already been laid out.

Colored QCA

In Sec. V, we considered cell coloring as a useful QCA programming technique. As with other computation models, where a programming technique can be formalized into its own subset model and then shown to be equivalent to the general model (such as multitrack Turing machines), we can do the same with colored QCA.

First, we will define the notion of a symmetric transition function for QCA. It is the quantum analog of symmetric CA, in which the transition function depends only on the total number of neighboring cells in each of the possible cell states. Essentially, a transition update function is symmetric when it affects only the value of the target cell in a manner which depends only on how many of the cell's neighbors are in particular states, rather than on which state any particular neighbor is in.

Definition 5. Given a QCA $Q=(L,\Sigma,\mathcal{N},U_0,V_0)$, we call the update function $U_0:(\mathcal{H}_\Sigma)^{\otimes\mathcal{N}}\to(\mathcal{H}_\Sigma)^{\otimes\mathcal{N}}$ symmetric if it can be expressed as a collection of single-cell operations on cell 0 controlled by the computational basis states of the neighborhood $\mathcal{N}\backslash\{0\}$, and U_0 commutes with every operator $\text{SWAP}_{x,y}$, which simply swaps the contents of cells x and y, where $x,y \in \mathcal{N}\backslash\{0\}$. If Q has a symmetric update function, then we call Q a symmetric QCA.

Next, we wish to formalize the notion of a colored QCA. For this model, we will fix the neighborhood scheme to include only directly adjacent cells. That is, $\mathcal{N}=\{x \in \mathbb{Z}^d : \|x\|_1$

$\leqslant 1\}$. However, first we will define the set of permissible colorings of a lattice.

Definition 6. Given a lattice $L=\mathbb{Z}^d$ and a neighborhood scheme $\mathcal{N}$, we define a correct k coloring for a lattice as a periodic mapping $C:L\rightarrow\{0,1,\ldots,k-1\}$, such that no two neighboring cells in L are assigned the same color.

We may think of cell color as an inherent property of each cell. However, it may also be helpful to consider cell color as classical information which is being stored with each cell in such a way that the local transition function does not alter this information. We can now finally give a definition for the colored QCA. Recall that the neighborhood scheme $\mathcal{N}$ is fixed.

Definition 7 (CQCA). A colored QCA or CQCA is a 5-tuple $(L,C,\Sigma,\mathcal{U},c)$ consisting of a lattice $L=\mathbb{Z}^d$, a correct k coloring C, a finite set Σ of cell states, a sequence of T symmetric unitary operators $\mathcal{U}=(U_0^{(0)},U_0^{(1)},\ldots,U_0^{(T-1)})$, with $U_0^{(j)}$: $(\mathcal{H}_\Sigma)^{\otimes\mathcal{N}}\rightarrow(\mathcal{H}_\Sigma)^{\otimes\mathcal{N}}$, and a sequence of T colors $c=(c_0,c_1,\ldots,c_{(T-1)})$, labeled by integers $0\leqslant c_j<k$.

The local transition operation consists of applying $U_x^{(j)}$ to each cell x with color $C(x)=c_j$ at time step $t=j+nT$, where $0\leqslant j<T$ and $n\in\mathbb{Z}$.

Note that since C is a correct k coloring, any two operators $U_x^{(j)}$ acting nontrivially on two cells of the same color at the same time will commute.

CQCA can be simply considered as a shorthand for the cell coloring technique we introduced in Sec. V. As such, it should be clear that CQCA are a subset of unitary QCA.

Theorem 3. For every CQCA Q there is a QCA Q' that simulates the same evolution exactly.

Proof. We may incorporate the color information of each cell of the CQCA Q within an additional color register for each cell of the QCA Q'. Now, it suffices to add one extra clock register to each cell, initialized to 0. The update operator U_x simply applies $U_x^{(j)}$ conditional on both $C(x)$ and the clock register of cell x being set to j. In order to ensure that U_x commutes with its translations, we must ensure that the colors of all the neighbors of x are consistent with the coloring C before applying the appropriate operator. Otherwise, U_x should act as the identity operator. The read operator V_x simply increments the clock register, modulo T. ■

What is more surprising is the converse result: that all unitary QCA can be rephrased in the CQCA formalism.

Theorem 4. For every QCA Q there is a CQCA Q' that simulates the same evolution exactly.

Proof. Given the QCA $Q=(L,\Sigma,\mathcal{N},U_0,V_0)$, we will use the same lattice L and alphabet Σ. The neighborhood scheme for the CQCA, $\mathcal{N}'$, is fixed by definition. We also need to provide a correct k coloring of the lattice. To this end, it suffices to provide a coloring C with the property that no neighborhood $\mathcal{N}_x$ of Q or $\mathcal{N}'_x$ of Q' contains two cells with the same color. Now, we need to construct a sequence $\mathcal{U}$ of update operators. Note that single-qudit operations on x and the controlled-NOT operation targeting x are symmetric operations, since any two cells belonging to the same neighborhood have different colors, by construction. Now, given an implementation of the unitary update operation U_0 of Q using single-qudit and nearest-neighbor controlled-NOT operations, we can give a sequence of symmetric operations that perform U_0 on a neighborhood $\mathcal{N}_x$ of a cell x of a specific color. By performing a similar sequence of operations for each color in our coloring C, we effectively perform U_x for each cell x. Since each update operation U_x commutes with the other update operations, we have effectively simulated the update transition operation of Q. Finally, we can perform the single-qudit operations V_x on each cell. ■

This last result is of major importance as it allows us to implement any unitary QCA algorithm on a pulse-driven quantum computer, as proposed by Lloyd [16], and further developed by Benjamin [14,15] and others [33]. The scheme involves using large molecules comprised of two or more different species of spin-$\frac{1}{2}$ particles, arranged in repetitive structures, such as crystals or polymers, to store the quantum data. It then evolves the system using series of magnetic pulses that address all spins of any one particular species.

To implement a given QCA in the pulse-driven computation model, we first convert the QCA into one which uses a two-state alphabet. This can be done by expanding each cell into $\lceil\log|\Sigma|\rceil$ cells to encode the states of Σ with a binary alphabet, then adjusting the neighborhood scheme $\mathcal{N}$ accordingly. We then apply the construction in Theorem 4. With this, and the techniques of Lloyd *et al.*, it would be possible to implement any QCA algorithm using NMR and an appropriate molecule.

We choose NMR and pulse-driven quantum computing devices to show a physical implementation of local unitary QCA. However, this should not be taken to be the only possible implementation of QCA. There are many other physical systems, like optical lattices [34], cavity QED, among others [35,36], that seem better suited to implementing QCA, rather than the more traditional quantum circuits.

VII. ALGORITHMS

We have seen two practical applications that can be achieved with an implementation of QCA in the laboratory. First, there are numerous physical systems that can be naturally simulated using the QCA model. Second, one can also achieve universal quantum computation by simulating quantum circuits on a QCA.

While these are both interesting and important applications of QCA, a very important application in the future of QCA will be the development and implementation of true, *native*, QCA algorithms.

We saw in Sec. IV how a quantum circuit can simulate any QCA, and how a QCA can simulate any quantum circuit. However, these simulations come at a cost of a linear-time slowdown going in either direction. While this slowdown is not as important a concern in terms of asymptotic complexity, in current laboratory conditions, any source of slowdown is to be avoided.

In the next section we analyze a problem that is particularly well suited to a natural solution using QCA, and we show how the tools that we have developed thus far can be used effectively to provide an optimal solution to the problem.

Spin-signal amplification algorithm

We present a description of the problem in simple abstract terms. Suppose we want to *amplify* the signal from a single

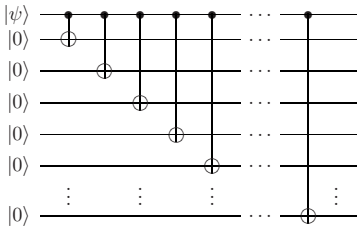


FIG. 7. A simple quantum circuit that implements U.

spin-$\frac{1}{2}$ particle. That is, we have a single spin-$\frac{1}{2}$ particle, and we want to create a large ensemble of spins whose bulk angular momentum resembles the original spin in a particular basis. Note that this is not cloning, since a basis needs to be set beforehand. Succinctly, we want a unitary procedure U that maps the state

$$\underbrace{(\alpha|0\rangle + \beta|1\rangle)}_{\text{amplified spin}} \otimes \underbrace{|0\rangle^{\otimes N}}_{\text{ancilla}}$$

to the state

$$\alpha|0\rangle^{\otimes(N+1)} + \beta|1\rangle^{\otimes(N+1)},$$

where $|0\rangle$ and $|1\rangle$ form the basis in which we wish to amplify. The main application of such an algorithm is to perform a measurement in situations where bulk magnetization is needed in order to achieve a detectable signal, such as with NMR. Hence, the algorithm needs to be extremely efficient: the whole procedure needs to be completed before decoherence can destroy the desired value. The value N will also need to be reasonably large, on the order of 10^7 or 10^8, in order to get a reasonable signal in NMR.

Figure 7 shows a simple quantum circuit solution. However, this circuit approach does have several shortcomings. First and foremost, it requires individually addressing N different spins. For large N, in most laboratory conditions, this is not feasible. Supposing that one could get around this first hurdle, one would still need to perform N independent gates before decoherence destroys the data. Again, this is not likely to be feasible in most experimental settings.

The QCA approach is simple, elegant, and optimally efficient. In order to develop the algorithm we will make use of the colored QCA developed earlier. We will use a two-color (black and white), three-dimensional QCA. Since we are describing an algorithm that has to be implemented in an actual physical setting, we will be using a finite-sized workspace. In order to describe this workspace, we will use three-state qudits for our cells: the logical states $|+1\rangle$ and $|-1\rangle$ will be used to denote the presence and the spin of a spin-$\frac{1}{2}$ particle in the corresponding state, while the state $|0\rangle$ will denote the absence of any particle. This state $|0\rangle$ will be *quiescent*, as defined in Sec. III.

Every cell in the QCA is initialized to a state $|0\rangle$ except for a perfectly cubic region of volume roughly $2N$. The cube will have its cells initialized to the value $|-1\rangle$, except for the top front left corner of the cube, whose value will be initialized to the state $|\psi\rangle$ which we wish to amplify. We will use this top, front, left portion of the cube as an ancilla system.

As we are using a colored QCA, the neighborhood of each cell is fixed to be the set of cells with Manhattan distance 1 from that cell. Hence, each cell has only neighbors of the opposing color. We also need to provide an update rule that is color symmetric. For both colors, the update rule is as follows. We apply a NOT gate (which maps $|+1\rangle$ to $|-1\rangle$, $|-1\rangle$ to $|+1\rangle$, and leaves the quiescent state $|0\rangle$ untouched) if and only if the set of neighbors of a cell have values that sum to -2, -1, or 0. It can be shown that this update rule, when applied repeatedly for $O(\sqrt[3]{N})$ time steps, will achieve the desired result.

We can make a few simple observations about the algorithm. First, as a native QCA algorithm, it does not require individual spin addressability. The algorithm is optimally efficient, if we allow only the use of local operations, in at most three dimensions.

It should also be noted that the problem of single-spin measurement in NMR is generally considered to be a difficult one; the fact that the exposition of the algorithm presented here is simple and succinct is due to the development of the theoretical tools earlier in this work.

It is important to add that it is possible to implement this algorithm in solid state NMR by adapting some of the techniques presented above, and applying some clever manipulations. For a full description of this algorithm, including a discussion on physical implementation, see [21,26,37].

VIII. PREVIOUS QCA MODELS

In this section, we will present a number of other models of QCA that have been developed, and we will relate them to our proposed model.

A. Watrous–van Dam QCA

An attempt to define a quantized version of cellular automata was made by Watrous [6], whose ideas were further explored by van Dam [5], and by Dürr, LêThanh, and Santha [38,39]. The model considers a one-dimensional lattice of cells and a finite set of basis states Σ for each individual cell, and features a transition function which maps a neighborhood of cells to a single quantum state instantaneously and simultaneously. Watrous also introduces a model of partitioned QCA in which each cell contains a triplet of quantum states, and a permutation is applied to each cell neighborhood before the transition function is applied.

Definition 8. A Watrous–van Dam QCA, acting on a one-dimensional lattice indexed by $\mathbb{Z}$, consists of a 3-tuple $(\Sigma,\mathcal{N},f)$ consisting of a finite set Σ of cell states, a finite neighborhood scheme $\mathcal{N}$, and a local transition function $f:\Sigma^{\mathcal{N}}\rightarrow\mathcal{H}_{\Sigma}$.

This model can be viewed as a direct quantization of the classical cellular automata model, where the set of possible configurations of the CA is extended to include all linear superpositions of the classical cell configurations, and the local transition function now maps the cell configurations of a given neighborhood to a quantum state. In the case that a neighborhood is in a linear superposition of configurations, f simply acts linearly. Also note that, in this model, at each

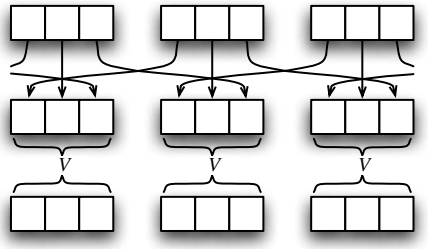


FIG. 8. Watrous partitioned QCA.

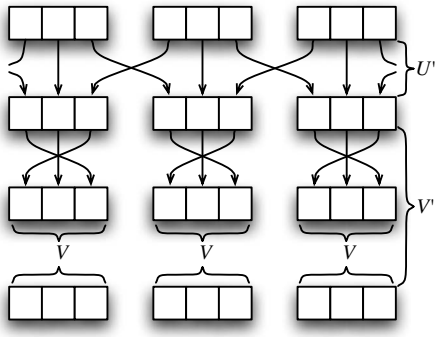


FIG. 9. Watrous QCA expressed as a local unitary QCA.

time step, each cell is updated with its new value simultaneously, as in the classical model.

Unfortunately, this definition allows for nonphysical behavior. It is possible to define transition functions that do not represent unitary evolution of the cell tape, either by producing superpositions of configurations which do not have norm 1, or by inducing a global transition function which is not injective, and therefore not unitary. In order to help resolve this problem, Watrous restricts the set of permissible local transition functions by introducing the notion of *well-formed* QCA. A local transition function is well formed simply if it maps any configuration to a properly normalized linear superposition of configurations. Because the set of configurations is infinite, this condition is usually expressed in terms of the ℓ_2 norm of the complex amplitudes associated with each configuration.

In order to describe QCA that perform unitary evolution, Watrous also introduces the idea of a *quiescent* state, which is a distinguished element $\epsilon \in \Sigma$ which has the property that $f:\epsilon^{\mathcal{N}} \mapsto \epsilon^{\mathcal{N}}$. We can then define a quiescent QCA as a QCA with a distinguished quiescent state acting only on finite configurations, which consist of finitely many nonquiescent states. It can be shown that a quiescent QCA that is well formed and injective represents unitary evolution on the lattice. Also, note that this notion of a quiescent state is slightly different from the one introduced in Sec. III.

In order to construct examples of valid QCA in this model, Watrous also introduces a model of partitioned QCA, in which each cell consists of three quantum states, so that the set of finite states can be subdivided as $\Sigma = \Sigma_l \times \Sigma_c \times \Sigma_r$ (Fig. 8). Given a configuration in which each cell, indexed by $k \in \mathbb{Z}$, is in the state $(q_k^{(l)}, q_k^{(c)}, q_k^{(r)})$, the transition function of the QCA in one time step first consists of a permutation which brings the state of cell k to $(q_{k-1}^{(l)}, q_k^{(c)}, q_{k+1}^{(r)})$ for each $k \in \mathbb{Z}$, then performs a local unitary operation V_k on each cell.

Watrous shows that this model of partitioned QCA can be used to simulate a universal quantum Turing machine with polynomial overhead.

The partitioned QCA model given by Watrous can also be expressed in the local unitary QCA model. First, suppose $|\Sigma_l| = |\Sigma_r|$. If this is not the case, we can pad the smaller set with unused symbols so that both sets are of the same size. Then, we separate the permutation into an operation P_1 which operates on two consecutive cells, mapping

$$P_1:(q_k^{(l)}, q_k^{(c)}, q_k^{(r)}), (q_{k+1}^{(l)}, q_{k+1}^{(c)}, q_{k+1}^{(r)}) \mapsto (q_k^{(l)}, q_k^{(c)}, q_{k+1}^{(l)}), (q_k^{(r)}, q_{k+1}^{(c)}, q_{k+1}^{(r)})$$

followed by an operation P_2 which operates on a single cell, mapping

$$P_2:(q_k^{(l)}, q_k^{(c)}, q_k^{(r)}) \mapsto (q_k^{(r)}, q_k^{(c)}, q_k^{(l)}).$$

Note that $P_2 P_1$ performs the desired permutation, and also that P_1 commutes with any lattice translation of P_1. Now we can express the Watrous partitioned QCA in our QCA model by setting $U' = P_1$ and $V' = VP_2$, as shown in Fig. 9.

B. Schumacher-Werner QCA

Schumacher and Werner [4] take a different approach in the definition of their model of QCA, working in the Heisenberg picture rather than the Schrödinger picture. They introduce a comprehensive model of QCA in which they consider only the evolution of the algebra of observables on the lattice, rather than states of the cell lattice itself. By extending local observables of the cell lattice into a closed observable algebra, the Schumacher-Werner model has a number of useful algebraic properties. In this model, the transition function is simply a homomorphism of the observable algebra which satisfies a locality condition. Schumacher and Werner also introduce a model of partitioned QCA called the *generalized Margolus partitioned QCA*, in which the observable algebra is partitioned into subalgebras. This generalizes the Margolus scheme, as described in Sec. II, in which the cell lattice itself is partitioned.

In order to avoid problematic issues dealing with observables over infinite lattices, Schumacher and Werner make use of the *quasilocal* algebra. In order to construct this algebra, we first start with the set of all observables on finite subsets $S \subseteq L$ of the lattice, denoted $\mathcal{A}(S)$, and extend them appropriately into observables of the entire lattice by taking a tensor product with the identity operator over the rest of the lattice. The completion of this set forms the quasilocal algebra.

In this setting, the global transition operator of a QCA is simply defined as a homomorphism $T:\mathcal{A}(L) \to \mathcal{A}(L)$ over the quasilocal algebra which satisfies two specific properties. First, a locality condition must be satisfied: $T(\mathcal{A}(S)) \subseteq \mathcal{A}(S$

$+\mathcal{N})$ for all finite $S \subseteq L$. Second, T must commute with lattice translation operators, so that the QCA is space homogeneous. Now, the QCA can be defined in terms of the lattice L, the neighborhood scheme $\mathcal{N}$, the single-cell observable algebra $\mathcal{A}_0$, which takes the place of the alphabet, and the global transition operator T.

The local transition operator of a QCA is simply a homomorphism $T_0: \mathcal{A}_0 \to \mathcal{A}(\mathcal{N})$ from the observable algebra of a single distinguished cell $0 \in L$ to the observable algebra of the neighborhood of that cell. Schumacher and Werner show that a local homomorphism T_0 will correspond uniquely to a global transition operator T if and only if for each $x \in L$, the algebras $T_0(\mathcal{A}_0)$ and $\tau_x(T_0(\mathcal{A}_0))$ commute elementwise. Here, τ_x is a lattice translation by x. The global transition operator T given by T_0 is defined by

$$T(\mathcal{A}(S)) = \prod_{x \in S} T_x(\mathcal{A}_x).$$

Next, we will describe the generalized Margolus partitioned QCA. Schumacher and Werner present this partitioned scheme as a method of producing valid reversible QCA in their general model. In order to describe this scheme, we will proceed according to the definition of a classical partitioned CA, as given in Sec. II.

We start with the d-dimensional lattice $L=\mathbb{Z}^d$, and we fix the sublattice $S=2\mathbb{Z}^d$ as the set of cells of L with all even coordinates. We also fix the time period as $T=2$. The block scheme $\mathbf{B}$ is given as $\{B_0, B_1\}$, which is given as

$$B_0 = \{(x_1, x_2, \ldots, x_d) \in L : 0 \leqslant x_j \leqslant 1, 1 \leqslant j \leqslant d\},$$

which is simply a cube of size 2^d with corners at cells $\mathbf{0} = (0, 0, \ldots, 0)$ and $\mathbf{1} = (1, 1, \ldots, 1)$, and

$$B_1 = B_0 + \mathbf{1},$$

which is simply a translation of the cube B_0.

Now, as in the regular Schumacher-Werner QCA model, we proceed in the Heisenberg picture. For any block B_0+s, $s \in S$, we have 2^d intersecting blocks from the partition B_1+S. For each block B_1+s' which intersects with B_0+s, there is a vector $v \in \mathbb{Z}^d$ representing the translation taking B_0+s to B_1+s', so that $B_1+s'=B_0+s+v$. Indeed, these 2^d intersecting blocks may be indexed by the vectors v, which are simply all vectors of $\mathbb{Z}^d$ whose entries are each ± 1. Hence, we will set $B_v^{(s)} = B_0+s+v$.

For each block $B_v^{(0)}$, we will fix an observable algebra $\mathcal{B}_v^{(0)}$ as a subalgebra of the observable algebra $\mathcal{A}(B_v^{(0)})$ for the entire block. Then, for each block $B_v^{(s)}$, the observable algebra $\mathcal{B}_v^{(s)}$ is simply the appropriate translation of $\mathcal{B}_v^{(0)}$. Note that, in particular, the observable algebra for the block $B_1+s=B_1^{(s)}$, $\mathcal{A}(B_1^{(s)})$, contains each of the observable algebras $\mathcal{B}_v^{(s+1-v)}$. In order for an assignment of subalgebras to be considered valid, these subalgebras $\mathcal{B}_v^{(s+1-v)}$ must commute and span $\mathcal{A}(B_1^{(s)})$. This occurs if and only if the product of the dimensions of these algebras is $|\Sigma|^{2^d}$.

The transition function then consists first of an isomorphism

$$T_0^{(s)}: \mathcal{A}(B_0^{(s)}) \to \prod_v \mathcal{B}_v^{(s)},$$

followed by the isomorphism

$$T_1^{(s)}: \prod_v \mathcal{B}_v^{(s+1-v)} \to \mathcal{A}(B_1^{(s)}).$$

Note that since T_0 and T_1 are isomorphisms between observable algebras of equal dimension, with an appropriate choice of basis, they can be represented by unitary operators U_0 and U_1 which map vectors from a complex vector space to another complex vector space of equal dimension. However, they do not represent local unitary evolution, since these complex vector spaces are used to describe two different quantum systems. For example, the shift-right QCA, which was shown in Sec. III to be impossible to implement using local unitary operators, can be constructed in the generalized Margolus partitioning QCA model.

Fortunately, it is possible to simulate the generalized Margolus partitioning QCA model within the local unitary QCA model by adding 2^d memory registers to each cell corresponding to the subalgebras $\mathcal{B}_v$ in addition to a clock register indicating which of the two stages of the transition function is being performed. The transition function of the local unitary QCA simply swaps the contents of the data registers of each cell with the appropriate memory registers before applying the unitary operations corresponding to the desired isomorphisms.

C. Other models

Meyer [9,40], Boghosian and Taylor [10], and Love and Boghosian [31], among others, explored the idea of using QCA as a model for simulating quantum lattice gases. As classical CA are used to model classical physical systems, it is natural to develop QCA models which are capable of modeling quantum physical systems. In order to simulate lattice gases, Meyer uses a model of QCA in which each lattice cell is represented by a computational basis state in a Hilbert space, and the set of states that a given cell can take is replaced with a complex number representing the amplitude of the basis state corresponding to that cell. In this regard, Meyer's QCA modeling of lattice gases greatly differs from the one presented here, and is not suitable as the basis for a more general model of QCA.

Lloyd [16] introduced a model of physical computation based on a chain consisting of a repeating sequence of a fixed number of distinguishable nuclear species. In this model, pulses are programmed that are capable of distinguishing the species and performing nearest-neighbor unitary operations. This model has been further developed by others [14,41,42]. It has been shown that this model is sufficient for implementing universal quantum computation.

The model, sometimes referred to as *pulse-driven quantum computers*, or globally controlled quantum arrays (GCQAs), is different from QCA in that it allows for time-dependent evolution. Still, they are closely related in their use of only space-homogeneous update rules. For the sake of applying results about one model to the other, it is also pos-

sible to argue that a pulse-driven quantum computer is a degenerate case of a QCA where the update rule is applied *once*. Also, this physical scheme provides a natural platform for implementing QCA.

D. Comparison and discussion

In this section, we have gone through a brief discussion of the major QCA models already in the literature today. While each has its own strengths, it is also true that each has weaknesses that are addressed by the LUQCA model.

We have seen how general Watrous–van Dam QCA have the problem of permitting ill-defined QCA. It is tempting to simply restrict attention to well-defined QCA in these models. However, deciding whether a given QCA is well defined or not is a hard problem.

Even when a QCA is guaranteed to be well defined, as is the case for the partitioned Watrous–van Dam QCA, these allow for evolution that is not unitary and local, e.g., shift-right automata as described in Sec. III B. This same criticism applies to the Schumacher-Werner model of QCA.

Fortunately, it is possible to simulate any valid QCA in these models with local unitary QCA by adding ancillary space to each cell, in order to perform the necessary evolution in a unitary fashion.

Meyer's definition of QCA, while being suitable for his purposes, is not general enough to allow for all the behavior that is possible with local unitary QCA, i.e., universal computation. Again, QCA in this model can easily be simulated by LUQCA.

Finally, we address globally controlled quantum arrays. There are many similarities and connections between QCA and this model of computation. One important connection is how globally controlled arrays can be used to implement QCA. However, these two models are quite distinct. A GCQA is centered around the idea of doing computation on large arrays of simple quantum systems, without locally addressing them. GCQAs divide their lattice of cells, or qudits, into *subsets*, each of which can be addressed collectively.

The first major distinction with QCA comes from the fact that sequences of pulses applied to these subsets of qudits are arbitrary, and do not necessarily follow a time-homogeneous pattern. The second is that, although Lloyd's construction is space homogeneous, GCQAs are not constrained in such a fashion. More recently, GCQAs have been proposed that have less spatially homogeneous structures [43].

As a model of computation one can say that QCA are more restricted than GCQAs. At the same time, QCA are more than just a model of computation; they serve also as models of physical phenomena. It can be argued that QCA are, in a sense, a more *fundamental* construct.

IX. CONCLUSIONS

In this paper, we have presented a model of quantum cellular automata based on local unitary operators. We have shown that it has distinct advantages over previous cellular automata quantizations. In particular, we have shown that given any LUQCA it is always possible to give an efficient, low-depth quantum circuit that faithfully represents it. This leads to the conclusion that any universal quantum computer could implement a LUQCA efficiently.

More importantly, however, we have also shown that it is possible to implement LUQCA in experimental setups that are arguably simpler than traditional quantum-circuit-based algorithms: for instance, globally addressing spins in NMR or ESR. At the same time, we have shown that our model is universal for quantum computation. We gave an explicit proof of an efficient simulation of quantum circuits using a two-dimensional QCA and mentioned that a proof of universality for one-dimensional LUQCA exists as well [26]. We also showcased the LUQCA as a modeling and simulation tool.

Finally, we have shown that QCA in previous models can be efficiently translated into QCA within the model presented here. For example, a universal QCA in previous models [6,18] can easily become a universal QCA within the local unitary model (see [26] for an explicit construction). All of these facts suggest that the LUQCA is a very strong model.

The purpose of this paper has been to motivate, develop, and showcase a model of quantum cellular automata based on strictly local, translation-commuting, unitary operators. It is our conjecture that the construction given here is the most general of this form.

Ultimately, it is our hope that this paper serves to help unify the several methods, results, and views surrounding QCA into one single, cohesive paradigm.

ACKNOWLEDGMENTS

The authors would like to thank Niel de Beaudrap and John Watrous for their invaluable comments on preliminary versions of this paper. Research for this paper was supported in part by DTO-ARO, ORDCF, Ontario-MRI, CFI, MITACS, and CIFAR.

[1] J. von Neumann, *Cerebral Mechanisms in Behavior* (John Wiley and Sons, New York, 1951), pp. 1–41.

[2] J. von Neumann, *Theory of Self-Reproducing Automata* (University of Illinois Press, Urbana, IL, 1966).

[3] B. Chopard and M. Droz, *Cellular Automata Modeling of Physical Systems* (Cambridge University Press, Cambridge, U.K., 1998).

[4] B. Schumacher and R. F. Werner, e-print arXiv:quant-ph/0405174.

[5] W. van Dam, MS thesis, University of Nijmegen, 1996.

[6] J. Watrous, in *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS), 1995* (IEEE Computer Society, Washington, D.C., 1995), pp. 528–537.

[7] C. A. Pérez-Delgado and D. Cheung, e-print arXiv:quant-ph/0508164.
[8] G. Grossing and A. Zeilinger, Complex Syst. **2**, 197 (1988).
[9] D. A. Meyer, J. Stat. Phys. **85**, 551 (1996).
[10] B. M. Boghosian and W. Taylor, Phys. Rev. E **57**, 54 (1998).
[11] D. M. Forrester, K. E. Kurten, and F. V. Kusmartsev, Phys. Rev. B **75**, 014416 (2007).
[12] J. Fitzsimons and J. Twamley, Phys. Rev. Lett. **97**, 090502 (2006).
[13] K. G. H. Vollbrecht and J. I. Cirac, Phys. Rev. A **73**, 012324 (2006).
[14] S. C. Benjamin, Phys. Rev. A **61**, 020301(R) (2000).
[15] S. C. Benjamin and S. Bose, Phys. Rev. Lett. **90**, 247901 (2003).
[16] S. Lloyd, Science **261**, 1569 (1993).
[17] R. Raussendorf, Phys. Rev. A **72**, 022301 (2005).
[18] D. J. Shepherd, T. Franz, and R. F. Werner, Phys. Rev. Lett. **97**, 020502 (2006).
[19] A. Imre, G. Csaba, L. Ji, A. Orlov, G. H. Bernstein, and W. Porod, Science **311**, 205 (2006).
[20] M. Khatun, T. Barclay, I. Sturzu, and P. D. Tougaw, J. Appl. Phys. **98**, 094904 (2005).
[21] C. A. Pérez-Delgado, M. Mosca, P. Cappellaro, and D. G. Cory, Phys. Rev. Lett. **97**, 100501 (2006).
[22] K. Walus and G. A. Jullien, Proc. IEEE **94**, 1225 (2006).
[23] P. J. Love and B. Boghosian, Physica A **362**, 210 (2006).
[24] T. Toffoli, Ph.D. thesis, The University of Michigan, 1977.
[25] T. Toffoli and N. Margolus, *Cellular Automata Machines* (MIT Press, Cambridge, MA, 1987).
[26] C. A. Pérez-Delgado, Ph.D. thesis, University of Waterloo, 2007.
[27] G. Vidal, Phys. Rev. Lett. **93**, 040502 (2004).
[28] *Density-Matrix Renormalization, a New Numerical Method in Physics*, edited by I. Peschel, X. Want, M. Kaulke, and K. Hallberg, Lecture Notes in Physics Vol. 528 (Springer-Verlag, Berlin, 1999).
[29] B. Boghosian, Physica D **120**, 30 (1998).
[30] J. Yepez, *QCQC '98: Selected Papers fro the First NASA International Conference on Quantum Computing and Quantum Communications*, Lecture Notes in Computer Science Vol. 1509 (Springer-Verlag, Berlin, 1999), p. 34.
[31] P. J. Love and B. M. Boghosian, Quantum Inf. Process. **4**, 335 (2005).
[32] P. J. Love, B. Boghosian, and D. A. Meyer, Philos. Trans. R. Soc. London, Ser. A **362**, 1667 (2004).
[33] G. K. Brennen and J. E. Williams, Phys. Rev. A **68**, 042311 (2003).
[34] S. C. Benjamin, B. W. Lovett, and J. H. Reina, Phys. Rev. A **70**, 060305(R) (2004).
[35] B. E. Kane, Nature (London) **393**, 133 (1998).
[36] M. Khatun, T. Barclay, I. Sturzu, and P. D. Tougaw, J. Phys. D **39**, 1489 (2006).
[37] C. A. Pérez-Delgado, D. Cheung, M. Mosca, P. Cappellaro, and D. Cory, in *Proceedings of the Asian Conference on Quantum Information Science (AQIS), 2006* (Chinese Academy of Science, Beijing, 2006).
[38] C. Dürr and M. Santha, in *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science, 1996* (IEEE Computer Society, Washington, D.C., 1996), pp. 38–45.
[39] C. Dürr, H. LêThanh, and M. Santha, Random Struct. Algorithms **11**, 381 (1997).
[40] D. A. Meyer, Phys. Lett. A **223**, 337 (1996).
[41] S. C. Benjamin and S. Bose, Phys. Rev. A **70**, 032314 (2004).
[42] S. C. Benjamin, New J. Phys. **6**, 61 (2004).
[43] J. Fitzsimons and J. Twamley, e-print arXiv:0707.1119v1.