

# A Physically Universal Cellular Automaton

Luke Schaeffer  
lrs@mit.edu

Massachusetts Institute of Technology

June 11, 2014

## Abstract

Several cellular automata (CA) are known to be universal in the sense that one can simulate arbitrary computations (e.g., circuits or Turing machines) by carefully encoding the computational device and its input into the cells of the CA. In this paper, we consider a different kind of universality proposed by Janzing. A cellular automaton is *physically universal* if it is possible to implement any transformation on a finite region of the CA by initializing the complement of the region and letting the system evolve. We answer an open problem of Janzing by constructing an example of a physically universal CA.

## 1 Introduction

Computation with cellular automata (CA) has an interesting history. Early work by von Neumann focused on self-reproducing structures and universal construction in cellular automata [11], although many of the proposed CAs were also computationally universal. In 1982, Berlekamp, Conway and Guy [2] showed that Conway's Game of Life (see also Gardner's article [5]), a particularly simple binary cellular automaton, is Turing-complete. Shortly after, Margolus [8] gave a reversible cellular automaton analogue for Fredkin and Toffoli's (reversible) billiard ball model of computation. More recently, Matthew Cook [3] proved rule 110 (a binary, one-dimensional CA) Turing-complete.

A cellular automaton is Turing-complete if it can simulate a Turing machine, but there is a fair amount of latitude in the details of the simulation. For instance, in Life, the simulation uses multicellular patterns called *gliders* to transmit and manipulate information. Likewise, Margolus's machine is based on "billiard balls" composed of pairs of particles moving along the same trajectory. Rule 110 builds its simulation on multicellular patterns called *spaceships*. In each case, the simulation requires the input tape to be specially formatted (as gliders, spaceships, or pairs of particles) and produces output in a similar format.

Janzing [7] proposed an alternative notion of universality specific to cellular automata. Informally, a CA is *physically universal* if it is possible to implement any transformation on a finite set of cells by initializing the surrounding cells and letting the system evolve (we will see a formal definition in the next section). In other words, computations in a physically universal CA operate directly on cells, without requiring any special input encoding.

For instance, in a physically universal CA, there is some way to configure the cells around a 5-by-5 square region such that after some number of steps,  $t$ , the contents of the region will

be rotated 90 degrees. Of course, we are not limited to geometric transformations. Given two disjoint 1-by- $n$  regions, we can treat them as  $n$ -bit integers, compute their product, and write back the high-order bits of the product in one region and the low-order bits in the other. We can transform the input region by *any* function.

As motivation for physical universality, consider Deutsch’s constructor theory [4], which “seeks to express all fundamental scientific theories in terms of a dichotomy between possible and impossible physical transformations”. A physically universal CA is a toy universe in which all transformations (on a finite region) are possible. As a special case of physical universality, we can construct any desired pattern, and although this does not imply the existence of self-reproducing machines, it is closely related to von Neumann’s work on universal construction. Janzing proposed physically universal CAs as a model of a world in which the boundary between a controller and the physical system it controls can be shifted. He discusses the thermodynamic and Kolmogorov complexity implications of physical universality at length in [7].

Physical universality imposes fairly stringent conditions on a CA. In particular, the CA must be reversible (i.e., it is possible to run the CA backwards), because otherwise information in the input may be lost before we can reach it. For instance, there are exponentially many configurations in Life which evolve to the empty configuration in one step. If one of these configurations occurs in the middle of the input region, there is no way to distinguish it from the others, because we cannot interact with it before it collapses. Reversibility rules out all of the examples above except for Margolus’s CA.

Despite being reversible and capable of simulating circuits, Margolus’s CA is not physically universal. It is possible to construct immovable, impenetrable walls in Margolus’s CA, which contradict physical universality in two different ways. First, information cannot pass through the wall, so input cells within the wall cannot interact with the cells encoding the transformation, so it is impossible to implement an arbitrary transformation. Second, there is no way to move or destroy a wall, so it is impossible to construct certain output configurations.

Since these well-known CAs (and many others) are not physically universal, Janzing asks whether physically universal CAs exist. The goal of this paper is to answer this question in two dimensions by presenting an example of a physically universal CA.

## 2 Physical Universality

Consider a cellular automaton where each cell has a state in  $\Sigma$ , a finite set. We will assume the cells are positioned at integer points  $\mathbb{Z}^d$  in  $d$  dimensions, but many of the following definitions are more general. Let  $\Omega := \mathbb{Z}^d$  be the set of all cells. Given a set  $X \subseteq \Omega$ , we say  $\bar{X} := \Omega \setminus X$  is the *complement of  $X$*  in  $\Omega$ .

A *configuration* of some subset of cells  $X \subseteq \Omega$  is a map assigning each cell in  $X$  a state in  $\Sigma$ , and we let  $\Sigma^X$  denote the set of all configurations of  $X$ . The notation  $\gamma \rightarrow \gamma'$  indicates that a configuration  $\gamma \in \Sigma^\Omega$  changes into  $\gamma' \in \Sigma^\Omega$  after one timestep. Similarly,  $\gamma \xrightarrow{n} \gamma'$  means that after  $n$  timesteps the configuration goes from  $\gamma$  to  $\gamma'$ .

Suppose  $Z \subseteq \Omega$  is a set of cells and we partition  $Z$  into disjoint subsets  $X$  and  $Y$ . A configuration  $\gamma \in \Sigma^Z$  naturally splits into configurations  $\gamma|_X \in \Sigma^X$  and  $\gamma|_Y \in \Sigma^Y$  called the *restriction of  $\gamma$  to  $X$  (resp.  $Y$ )*. Conversely, given configurations  $\alpha \in \Sigma^X$  and  $\beta \in \Sigma^Y$ , we let  $\alpha \oplus \beta$  denote the combined configuration of  $\Sigma^Z$ .

**Definition 1.** Let  $X, Y \subseteq \Omega$  be finite sets. Let  $f: \Sigma^X \rightarrow \Sigma^Y$  be an arbitrary function. Then we say a configuration  $\phi \in \Sigma^{\bar{X}}$  *implements the transformation  $f$  in time  $T$*  if for any configuration  $x \in \Sigma^X$  there exists a configuration  $\psi \in \Sigma^{\bar{Y}}$  such that

$$x \oplus \phi \xrightarrow{T} \psi \oplus f(x).$$

In other words, a configuration  $\phi$  of  $\bar{X}$  implements a transformation  $f$  if any initial configuration of  $x$ , combined with  $\phi$ , evolves (in  $T$  timesteps) to a configuration where  $Y$  contains  $f(x)$ . We do not care about the contents of  $\bar{Y}$  after  $T$  timesteps.

**Definition 2.** We say a cellular automaton is *physically universal* if for all  $X, Y \subseteq \Omega$  and all transformations  $f: \Sigma^X \rightarrow \Sigma^Y$ , there exists a configuration  $\phi$  of  $\bar{X}$  and constant  $t_0$  such that  $\phi$  implements  $f$  in time  $t_0$ .

We say the CA is *efficiently* physically universal if the implementation runs in time  $t_0$ , where  $t_0$  is polynomial in

- the *diameter* of  $X$  (i.e., the width of the smallest hypercube containing the set) and diameter of  $Y$ ,
- the distance between  $X$  and  $Y$ , and
- the computational complexity of  $f$  under some appropriate model of computation (e.g., the number of logical gates in a circuit for  $f$ ).

Janzing’s definition differs from our definition in two ways, but it turns out the definitions are equivalent. First, he assumes the input and output region are the same, i.e.,  $X = Y$ . Clearly we can take any region  $Z$  containing  $X \cup Y$  and extend  $f$  to a function  $\hat{f}$  on the region  $Z$ , which does not depend on input outside region  $X$ , agrees with  $f$  on  $Y$ , and takes values everywhere else. Janzing also assumes the transformation is bijective, but by a well-known result of Bennett [1], we can embed an arbitrary function  $f$  into a reversible function on twice as many bits, such that for some initial configuration of half the bits,  $f$  is a restriction of the reversible function to the other half of the bits.

Let us make a few observations about efficient physical universality. First, the time to implement an arbitrary transformation must grow linearly with the diameter of the input, since information can only travel one cell per step. For the same reason, information must (in general) travel from  $X$  to  $Y$ , so the number of steps is at least linear in the distance between them. Finally, there is an enormous number of transformations from  $X$  to  $Y$ . Even if  $Y$  is a single cell, we need at least  $|\Sigma|^{|X|}$  bits specify an arbitrary transformation. We do not have that many bits in polynomial space, so we need a third parameter, e.g., computational complexity, to give us more space for some transformations.

### 3 Cellular Automaton

In this section, we define and motivate a cellular automaton; in Section 4 we will prove the CA is physically universal. We use two states,  $\Sigma = \{0, 1\}$ , in our CA. In figures, we will represent 0 cells as white and 1 cells as black. Our CA is a kind of *block cellular automaton*, meaning that we partition the grid into blocks of cells and update each block according to a *local update rule*. In this case, we use a partition of the grid into 2-by-2 blocks. On the next timestep, we shift the partition one step north and one step east, then back again for the timestep after that. This is known as a *Margolus neighbourhood*.

Let the symbol  $\boxplus$  denote a set of four cells in a 2-by-2 square. The state of a block is a function from  $\boxplus$  to  $\Sigma$ , so we write  $\Sigma^{\boxplus}$  for the set of all configurations of the block. In a Margolus neighbourhood cellular automaton, the local update rule takes the form  $\rho: \Sigma^{\boxplus} \rightarrow \Sigma^{\boxplus}$ , which expresses the evolution of a 2-by-2 block over one timestep.

There are some advantages to using the Margolus neighbourhood over the more common *Moore neighbourhood*, where each cell is updated based on the current state of it and its eight nearest neighbours. It is much easier to express CAs with reversibility and conservation properties in a Margolus neighbourhood. For instance, a Margolus neighbourhood CA is reversible if and only if the update rule  $\rho$  is reversible, and the reverse CA has a Margolus neighbourhood. Compare this to a Moore neighbourhood CA, where it is undecidable whether an arbitrary CA is reversible, and the reverse may not be a Moore neighbourhood CA.

Margolus and Toffoli [10] give a very general conversion from block cellular automata to Moore neighbourhood CAs. Their conversion augments the set of states with extra information, so each cell knows its current position within the block (e.g., NE, NW, SE, SW). Then the Moore neighbourhood rule can tell which neighbours of a cell are in the same block, and update accordingly. This is suitable for simulating a block CA, but note that the extra information must have the correct initial configuration (e.g., a “northeast corner” cell should not be next to another cell of the same type) or the CA will fail. The resulting Moore neighbourhood CA is therefore not physically universal, since some configurations of the input region (or output region) are forbidden.

Another way to convert a Margolus neighbourhood CA to a Moore neighbourhood is to collapse each 2-by-2 block to a cell (with state  $\Sigma^{\boxplus}$ ). The configuration of a block after two timesteps is a function of its current state and the eight surrounding blocks, so we can make one timestep in the Moore neighbourhood CA simulate *two* timesteps of the Margolus neighbourhood CA. It will turn out that our (Margolus neighbourhood) CA takes an even number of steps to implement any transformation, so the corresponding Moore neighbourhood CA is also physically universal.

Let us introduce the local update rules for our CA. We begin with the following two rules:

$$\begin{array}{|c|c|} \hline \square & \square \\ \hline \square & \square \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|} \hline \square & \square \\ \hline \square & \square \\ \hline \end{array}, \quad \begin{array}{|c|c|} \hline \blacksquare & \square \\ \hline \square & \square \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|} \hline \square & \blacksquare \\ \hline \square & \square \\ \hline \end{array}$$

The rule on the left says that empty (white) space remains empty. If we start with a single black cell in the northwest corner of a block, the rule on the right moves it to the southeast corner. Since the partition also moves one step southeast (or equivalently northeast/northwest/southwest), the black cell is again in the northwest corner and therefore it takes another step southeast. Every step, the black cell will move one unit southeast.

We also have the following three similar rules.

$$\begin{array}{|c|c|} \hline \blacksquare & \blacksquare \\ \hline \square & \square \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|} \hline \blacksquare & \square \\ \hline \square & \square \\ \hline \end{array} \quad \begin{array}{|c|c|} \hline \square & \square \\ \hline \blacksquare & \square \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|} \hline \square & \blacksquare \\ \hline \square & \square \\ \hline \end{array} \quad \begin{array}{|c|c|} \hline \square & \square \\ \hline \square & \blacksquare \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|} \hline \blacksquare & \blacksquare \\ \hline \square & \square \\ \hline \end{array}$$

It follows that a lone black cell will move in some direction (northeast, northwest, southeast or southwest as dictated by the corner it starts in) until it encounters another black cell. We think of black cells as *particles* and white cells as empty space, so the remaining update rules describe the interactions between two or more particles. In keeping with the particle metaphor, our CA satisfies the following “physical” laws.

**Symmetry:** Particle interaction is symmetric. To be precise, given a symmetry  $\pi: \boxplus \rightarrow \boxplus$  of  $\boxplus$  (e.g., rotation or reflection),  $\rho(x \circ \pi) = \rho(x) \circ \pi$  for all  $x \in \Sigma^{\boxplus}$ , where  $\circ$  denotes function composition.

**Conservation:** The number of particles is conserved. The number of black cells within a block does not change after one update step, and therefore the number of black cells across the whole grid is constant over time.

**Reversibility:** The CA is reversible. A Margolus neighbourhood CA is reversible if and only if the update rule,  $\rho$ , is bijective.

We argued that reversibility is necessary in a physically universal CA. There is no reason to believe symmetry or conservation are necessary for physical universality, but they are nice properties to have.

Up to symmetry, there are only four kinds of collisions: two opposing particles, two particles at right angles, three particles, and four particles. Particle conservation forces the four-particle rule to be

$$\begin{array}{|c|c|} \hline \blacksquare & \blacksquare \\ \hline \blacksquare & \blacksquare \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|} \hline \blacksquare & \blacksquare \\ \hline \blacksquare & \blacksquare \\ \hline \end{array},$$

and severely restricts the two- and three-particle collisions. Symmetry narrows down the set of

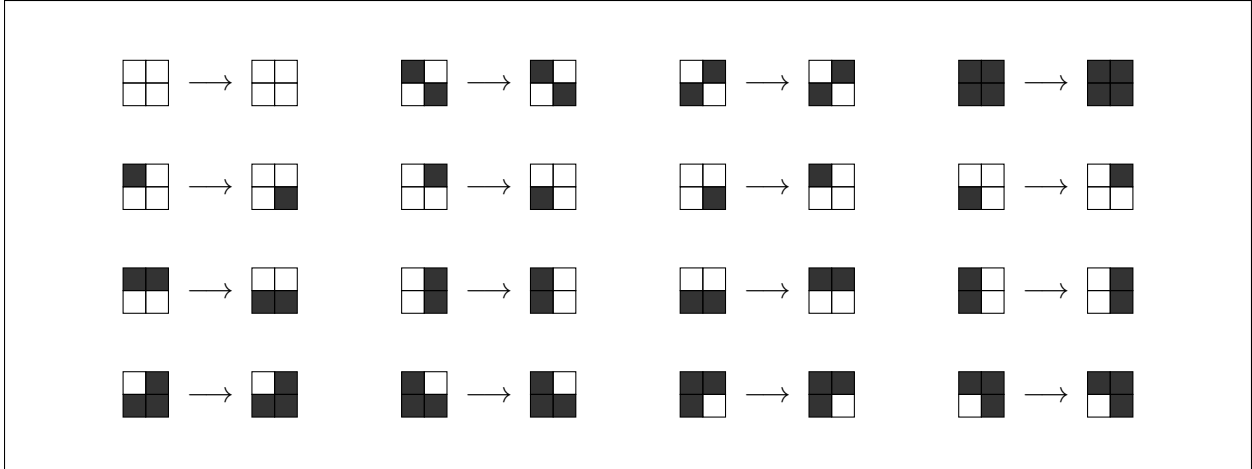


Figure 1: A full description of  $\rho$ , the local update rule.

possible rules even further. For instance, if we have one of these two rules,

$$\begin{array}{|c|c|} \hline \blacksquare & \blacksquare \\ \hline \blacksquare & \square \\ \hline \end{array} \rightarrow \begin{array}{|c|c|} \hline \blacksquare & \blacksquare \\ \hline \square & \square \\ \hline \end{array} \qquad \begin{array}{|c|c|} \hline \blacksquare & \square \\ \hline \square & \square \\ \hline \end{array} \rightarrow \begin{array}{|c|c|} \hline \square & \square \\ \hline \square & \square \\ \hline \end{array}$$

then we have both by symmetry, since they are 180 degree rotations of each other. Both rules cannot hold simultaneously because the local update rule is deterministic, therefore neither rule holds. Due to the reversibility of the CA, the reverse situation is similar. That is, we cannot have the rules

$$\begin{array}{|c|c|} \hline \blacksquare & \blacksquare \\ \hline \square & \square \\ \hline \end{array} \rightarrow \begin{array}{|c|c|} \hline \square & \square \\ \hline \blacksquare & \blacksquare \\ \hline \end{array} \qquad \begin{array}{|c|c|} \hline \square & \square \\ \hline \square & \square \\ \hline \end{array} \rightarrow \begin{array}{|c|c|} \hline \square & \blacksquare \\ \hline \blacksquare & \blacksquare \\ \hline \end{array}$$

since having one would imply the other (by symmetry) and having both contradicts reversibility.

In the end, eight cellular automata satisfy our conditions, corresponding to all possible choices of

$$\begin{array}{l} \begin{array}{|c|c|} \hline \blacksquare & \blacksquare \\ \hline \square & \square \\ \hline \end{array} \rightarrow \begin{array}{|c|c|} \hline \blacksquare & \blacksquare \\ \hline \square & \square \\ \hline \end{array} \text{ OR } \begin{array}{|c|c|} \hline \square & \square \\ \hline \blacksquare & \blacksquare \\ \hline \end{array}, \\ \begin{array}{|c|c|} \hline \blacksquare & \square \\ \hline \square & \square \\ \hline \end{array} \rightarrow \begin{array}{|c|c|} \hline \square & \square \\ \hline \blacksquare & \blacksquare \\ \hline \end{array} \text{ OR } \begin{array}{|c|c|} \hline \blacksquare & \blacksquare \\ \hline \square & \square \\ \hline \end{array}, \\ \begin{array}{|c|c|} \hline \blacksquare & \blacksquare \\ \hline \blacksquare & \square \\ \hline \end{array} \rightarrow \begin{array}{|c|c|} \hline \blacksquare & \blacksquare \\ \hline \blacksquare & \square \\ \hline \end{array} \text{ OR } \begin{array}{|c|c|} \hline \square & \square \\ \hline \blacksquare & \blacksquare \\ \hline \end{array}. \end{array}$$

Once we fix  $\rho$  for these three values, the rest of the update rule follows by symmetry. For our cellular automaton, we chose

$$\begin{array}{|c|c|} \hline \blacksquare & \blacksquare \\ \hline \square & \square \\ \hline \end{array} \rightarrow \begin{array}{|c|c|} \hline \square & \square \\ \hline \blacksquare & \blacksquare \\ \hline \end{array}, \quad \begin{array}{|c|c|} \hline \square & \square \\ \hline \square & \square \\ \hline \end{array} \rightarrow \begin{array}{|c|c|} \hline \square & \blacksquare \\ \hline \blacksquare & \blacksquare \\ \hline \end{array}, \quad \begin{array}{|c|c|} \hline \blacksquare & \blacksquare \\ \hline \blacksquare & \square \\ \hline \end{array} \rightarrow \begin{array}{|c|c|} \hline \blacksquare & \blacksquare \\ \hline \blacksquare & \square \\ \hline \end{array}.$$

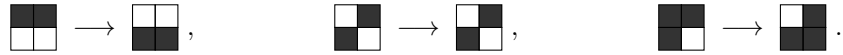
For reference, Figure 1 contains the full set of rules.

It is worth noting that at least five of the remaining seven rules have been previously studied. In particular, Margolus's billiard ball CA [8] corresponds to the opposite choice for perpendicular two-particle interactions. Unfortunately, with these three rules,

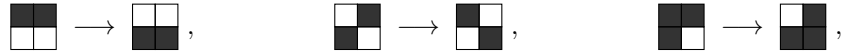
$$\begin{array}{|c|c|} \hline \blacksquare & \blacksquare \\ \hline \square & \square \\ \hline \end{array} \rightarrow \begin{array}{|c|c|} \hline \blacksquare & \blacksquare \\ \hline \square & \square \\ \hline \end{array}, \quad \begin{array}{|c|c|} \hline \blacksquare & \square \\ \hline \square & \square \\ \hline \end{array} \rightarrow \begin{array}{|c|c|} \hline \blacksquare & \square \\ \hline \square & \square \\ \hline \end{array}, \quad \begin{array}{|c|c|} \hline \blacksquare & \blacksquare \\ \hline \blacksquare & \blacksquare \\ \hline \end{array} \rightarrow \begin{array}{|c|c|} \hline \blacksquare & \blacksquare \\ \hline \blacksquare & \blacksquare \\ \hline \end{array},$$

and their reflections/rotations, one can construct impenetrable walls. It is impossible to alter the cells in the wall, and the wall can isolate a portion of the input from the environment, so this rule is not physically universal.

Suppose we change the three particle interaction instead:



Then  $\rho$  swaps opposite corners regardless of their contents; there are no particle interactions. However, if we change the rule for two opposing particles, we get



the Hardy-Pazzis-Pomeau (HPP) gas. The HPP gas was proposed in [6] as a simplified model for real gases, because it conserves the number of particles (and hence, the total energy) as well as momentum.

## 4 Proof of Physical Universality

As discussed earlier, we may assume the input region  $X$  and output region  $Y$  are the same, and take  $X$  to be a  $2n \times 2n$  square of cells aligned to the block boundaries. We assume our boolean function is given as a circuit of  $m$  NOR gates. Our goal is to produce an initial configuration  $\phi \in \Sigma^{\bar{X}}$  which implements  $f$  in time polynomial in  $n$  and  $m$ .

We divide the computation into stages chronologically as follows.

1. It turns out that particles in our CA behave somewhat like a gas, in that particles diffuse out of any bounded region. We will show that if all particles are initially in a square box (e.g.,  $X$ ), then after some number of timesteps, all particles will be outside the box, moving away in one of four groups of particles.
2. We place particles in the initial configuration of  $\bar{X}$  to intercept the particles as they escape from the box, preserving the information represented by the presence or absence of a particle. We rearrange this information into a column formation for easier computation.
3. Next, we show how to implement a NOR operation on the column of bits. Since the NOR gate is universal, we can implement arbitrary circuits, which we use to perform the following transformations.
  - Simulate the CA in reverse to determine the initial configuration of the input, based on the particles that escaped  $X$ .
  - Apply the function  $f$ .
  - Simulate the CA in reverse again, so we know where to send particles to achieve the desired output configuration.
4. Finally, we carefully position particles such that they converge on  $X$  and interact to produce the desired result.

We structure the proof to reflect these stages. For each stage, we place a few more particles in the initial configuration at positions designed to cause collisions between particles at precise locations. By controlling the collisions, we can accomplish the objectives for each stage. We are careful to avoid all other collisions. See Figure 2 for an overview of the stages.

### 4.1 Particle Diffusion

Particles in our CA will tend to diffuse out of any finite region into the empty space surrounding it. The CA is deterministic, so all particles escape after finite time, and we can give an explicit bound on that time. Furthermore, there are exactly  $4n^2$  places where the  $4n^2$  particles in the input region can escape, so we know where to intercept the particles.

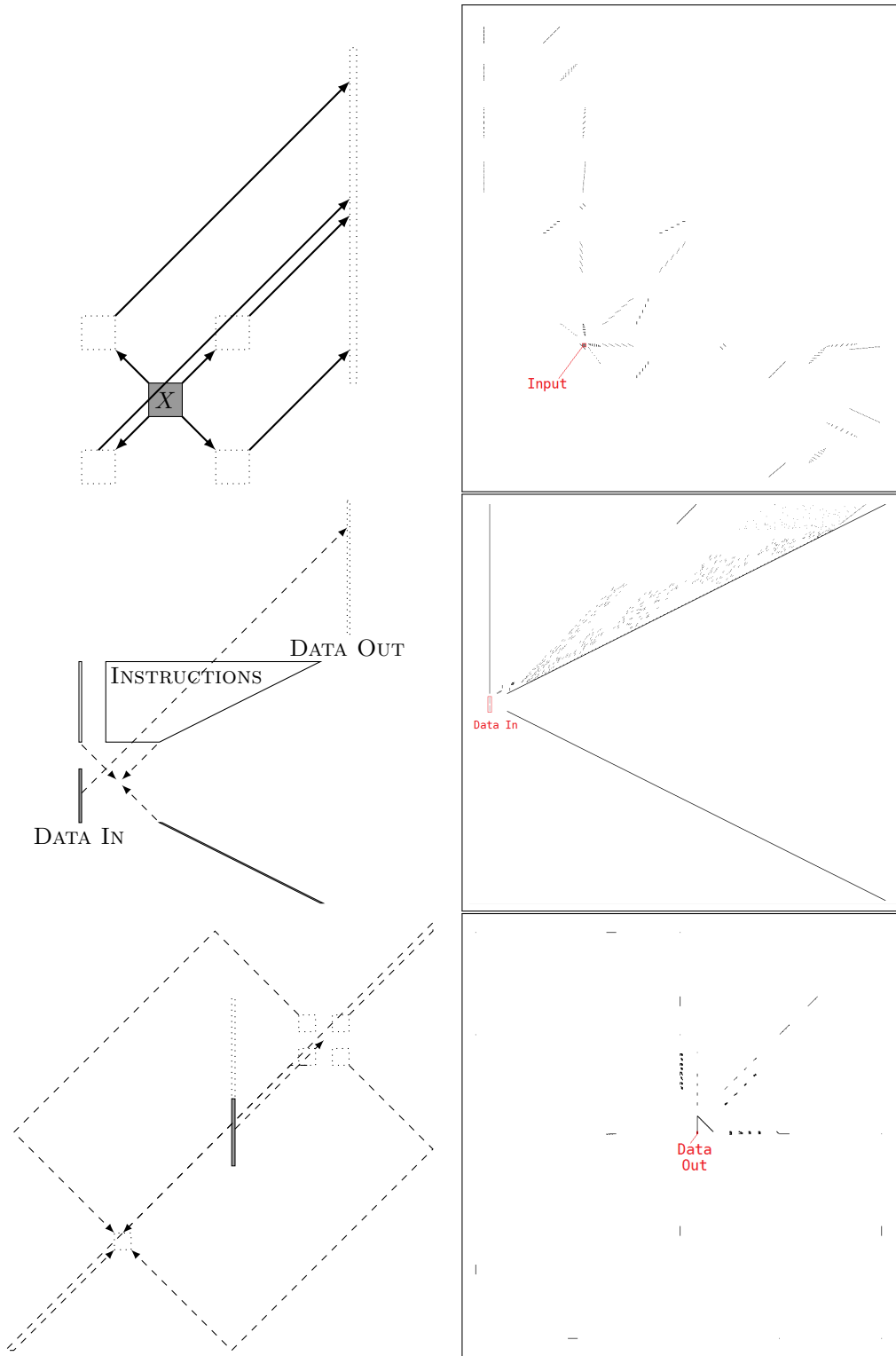


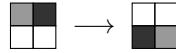
Figure 2: The three stages: input extraction, computation, and output insertion. For each one, the diagram on the left shows the flow of information, and the image on the right is the actual configuration at the beginning of the stage a sample implementation.


In order to prove a theorem about all possible configurations of the input, we need to reason about the evolution of a collection of configurations simultaneously. Our approach is to represent the set of configurations in our original CA, which we call the *concrete CA*, with a single configuration of a new three-state *abstract CA*. We will relate the evolution of a single abstract configuration to the evolution of the collection of concrete configuration. The abstract CA is purely a tool for the proof; it will not be physically universal, or even reversible.

The abstract CA is over a set of three states,  $\Gamma = \{0, 1, \top\}$ , drawn as white, black and gray in figures. We think of it as an extension of the concrete CA where white and black have the same meaning as before, and gray is used to represent cells with unknown, uncertain or “wildcard” state. Suppose  $\phi \in \Sigma^X$  is a concrete configuration and  $\Phi \in \Gamma^X$  is an abstract configuration (of the same region). Then we say  $\phi$  is *consistent* with  $\Phi$  (or sometimes,  $\Phi$  is *consistent* with  $\phi$ ) if every non-gray cell in  $\Phi$  has the same colour as the corresponding cell in  $\phi$ .

Given a set of concrete configurations,  $S \subseteq \Sigma^X$ , there is a *minimal consistent configuration*  $\Phi_S \in \Gamma^X$  which is consistent with every member of  $S$ , and has the minimal number of gray cells. We think of  $\Phi_S$  as a representation of the entire set  $S$ , but we lose a lot of information about the specific correlations between cells. For instance, if  $S$  is the set of all 2-by-2 block configurations with exactly one black cell, then we are uncertain about every cell. The minimal consistent configuration  $\Phi_S$  is an all-gray block, the same as if  $S$  was the set of *all* block configurations.

The evolution of the abstract CA is as one might expect. Formally, we define  $\rho' : \Gamma^{\square} \rightarrow \Gamma^{\square}$  such that  $\rho'(\Phi)$  is the minimal configuration consistent with  $\rho(\phi)$  for all  $\phi$  consistent with  $\Phi$ . In other words, we fill in the gray cells of  $\Phi$  as black or white in all possible ways, evolve this collection of concrete configurations forwards one timestep, then let  $\rho'(\Phi)$  be the minimal configuration consistent with all the evolutions. For example,  $\rho'$  contains the following rule



because the two configurations consistent with  evolve as follows,



and  is the minimal configuration consistent with both outcomes.

We can say a few general things about  $\rho'$  based on the conservativity of our CA. Consider the number of black cells, or *population*, of a configuration. For abstract configurations, the population is a *range* from the number of black cells, to the number of black or gray cells. Since our CA is conservative, the range in population for an abstract  $\Phi$  is contained in the range for  $\rho'(\Phi)$ . In particular, if all the cells in  $\Phi$  are white or gray, then there cannot be any black cells in  $\rho'(\Phi)$ , since the population could be zero. The width of the range is the number of gray cells, so this also tells us that the number of gray cells cannot decrease as we evolve an abstract configuration (we can also get this from reversibility).

The payoff of these definitions is the following theorem (which we state without proof) relating the evolution of a collection of concrete configurations to the evolution of a single abstract configuration.

**Theorem 3.** *Let  $\Phi \in \Gamma^{\Omega}$  be an abstract configuration, and suppose  $\Phi$  evolves (in the abstract CA) to  $\Phi'$  after  $n$  timesteps. Let  $\phi \in \Sigma^{\Omega}$  be a concrete configuration, and suppose it evolves (in the concrete CA) to  $\phi'$  after  $n$  timesteps. Then  $\phi'$  is consistent with  $\Phi'$  if  $\phi$  is consistent with  $\Phi$ .*

One might hope for  $\Phi'$  to be the minimal configuration meeting this condition. This is true for one or two timesteps, but breaks down for three or more timesteps, as shown in Figure 3. In the figure, the leftmost (abstract) configuration  $\Phi$  in Figure 3 evolves to the configuration in the



middle over three timesteps. However, if we take a concrete configuration consistent with  $\Phi$  and evolve it forward three timesteps, it is consistent with the rightmost configuration in Figure 3, so the middle configuration is not minimal.

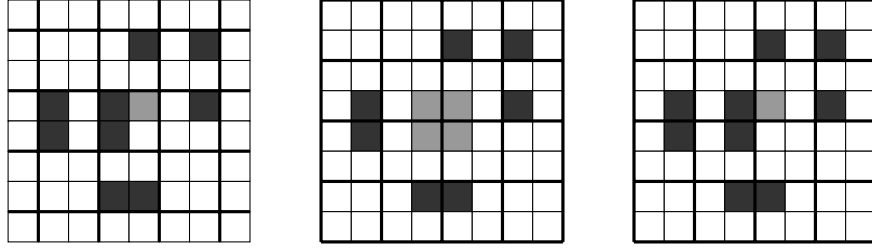


Figure 3: An example of a configuration (left) such that after only three timesteps, the abstract evolution (middle) differs from the minimal consistent configuration (right).

The theorem, and more generally, the idea of creating an abstract CA an “uncertain” state, is not specific to our CA. We could easily take an arbitrary binary cellular automaton and construct the corresponding abstract CA as above. In most cases, the result is uninformative; the “uncertain” state tends to proliferate, which tells us nothing about the concrete state. In that case, Theorem 3 may tell us little more than “non-blank cells may spread up to  $t$  steps away in  $t$  timesteps”. In the abstract version of Conway’s Life, for example, aside from two simple patterns<sup>1</sup>, it is challenging to construct any configuration where the uncertain cells neither go extinct, nor overwhelm everything else.

Our specific abstract CA does not suffer from the problems described above because interaction between cells is rare. The CA just swaps opposite corners within each block unless exactly three of the cells are black. As a result,  $\rho'$  swaps opposite corners with each abstract block unless it is consistent with some three-particle concrete block. This allows us to compute most of the rules on white and gray blocks, shown in Figure 4. Observe that on white and gray blocks,  $\rho'$  is identical to  $\rho$ , except for the three-particle interactions. It follows that, for the most part, gray cells behave like particles, which we exploit to prove the following theorem.



Figure 4: The update rule  $\rho'$  (suppressing symmetries) on white and gray blocks.

**Theorem 4** (Diffusion Theorem). *Let  $X \subseteq \Omega$  be a  $2n$ -by- $2n$  square region of cells. Let  $\phi \in \Gamma^\Omega$  be a configuration where  $\bar{X}$  is white and  $X$  is gray. Then we can partition the gray cells into four groups,  $C_{NE}$ ,  $C_{NW}$ ,  $C_{SE}$  and  $C_{SW}$  such that*

- *each group contains  $n^2$  gray particles,*
- *the gray particles in  $C_x$  all move in the direction  $x$  each step, and*
- *the cells outside  $C_{NE} \cup C_{NW} \cup C_{SE} \cup C_{SW}$  are all white.*

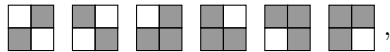
*It follows (by Theorem 3) that for any concrete configuration  $\psi$  consistent with  $\phi$  (i.e., with all particles confined to the box  $X$ ), the region  $X$  will be empty after  $2n$  timesteps.*

<sup>1</sup>Specifically, a blinker and a block, identical to their counterparts in concrete Life, but with uncertain cells.

*Proof.* The initial abstract configuration  $\phi$  contains only white and gray cells. Since  $\rho'$  maps white and gray blocks to white and gray blocks (see Figure 4), the configuration will always be white and gray. Recall that the gray cells behave like particles as long as there are no three-particle interactions.

Divide the gray particles into groups  $C_{NE}$ ,  $C_{NW}$ ,  $C_{SE}$  and  $C_{SW}$ . We let  $C_{NE}$  contain particles in the southwest corners of their respective blocks because a particle in the southwest corner will move to the northeast corner (i.e., in the northeast direction) if there are no other particles. Similarly for  $C_{NW}$ ,  $C_{SE}$  and  $C_{SW}$ .

We claim that the following blocks do not appear in the evolution of  $\phi$



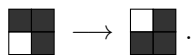
and all the gray cells are in  $C_{NE}$ ,  $C_{NW}$ ,  $C_{SE}$ ,  $C_{SW}$ , which move one cell per timestep in their respective directions.

If the six forbidden blocks do not appear, then it follows from the rules in Figure 4 that the gray cells in the groups move as described and no other gray cells are created.

The forbidden blocks do not appear in the initial configuration, but suppose for a contradiction that one of these six blocks occurs in some timestep. Without loss of generality, let the northeast corner of the block be white, so  $C_{SW}$  does not overlap with the block. Since  $C_{SW}$  is a square, the block is either too far north/south, or too far east/west to be in  $C_{SW}$ . In the former case, the block is also outside  $C_{SE}$  (since  $C_{SE}$  covers the same north/south range as  $C_{SW}$ ), so the northwest corner is also white. In the latter case, the block must be outside  $C_{NE}$  and therefore the southeast corner of the block is white. In general, if some corner of a block is white then one of the two adjacent corners is also white. All six blocks above violate this condition, so they cannot occur.

A simple induction argument completes the proof. There are no forbidden blocks at  $t = 0$ , so the cells evolve as described, which implies there are no forbidden blocks in the next timestep, and so on.  $\square$

Since the proof ignores the behaviour of  $\rho'$  on the forbidden blocks, so the theorem generalizes to other abstract CAs (and the corresponding concrete CAs). We can change the two-opposing-particle and three-particle interactions in the concrete CA, and it will only affect the two-opposing-particle and three-particle interactions in the abstract CA. In particular, the theorem holds for the HPP gas, and another CA where



We discuss these CAs in the last section.

The diffusion theorem has a number of uses. First, it saves us from having to tediously probe into the input region to extract the information inside it. Second, it helps us prevent unintended collisions between particles by specifying a time after which there are no more collisions. Finally, observe that  $\rho$  is an involution, so the reverse CA is the same as the original CA (starting with the opposite block alignment). Therefore, the diffusion theorem tells us we can put the output region into an arbitrary configuration by colliding four groups.

## 4.2 Input Signal Redirection

After  $2n$  steps, the input particles are in the four groups, as described above. The presence or absence of particles in the groups encodes  $4n^2$  bits of information. Every timestep, each particle in  $C_{NE}$  moves one step northeast. Similarly, if there is no particle some cell of  $C_{NE}$  then in the next timestep there will be no particle in the cell one step northeast. Either way, a bit of information moves northeast each timestep. We call this moving piece of information a *signal*.

We need to preserve the signals, since we need them to reconstruct the original input bits, and eventually perform the transformation. It is clearly impossible to perform any computation while the signals are moving in four different directions. Therefore, our first task is to redirect the signals to all move in the same direction, say, northeast. Then we manipulate the signals so they are in the same column and densely packed. We call this arrangement *column formation*.

We use carefully placed particles to control the signals. Since there are no interactions between pairs of particles, we need to intercept a signal with at least two other particles to affect it. Nothing useful happens when three particles meet a signal, so we consider the two ways (up to symmetry) for a signal and two particles to meet. We call these cases *reflection* and *deflection*.

**Reflection:** Suppose two particles meet head-on, with the signal at right angles. The signal will reflect off of the two particles and move in the opposite direction, as shown by the following rules.



We use reflection to reverse the direction of a particle.

**Deflection:** Suppose the signal meets one particle head-on and another particle at right angles. The signal will continue on, but the particle at right angles will be reflected or not depending on the signal, effectively copying the signal.



Deflection is useful for creating a duplicate (or inverted) signal at right angles. A deflection can produce particles or signals in all four directions, so we usually follow a deflection with one or more reflections to clean up.

These two operations are the building blocks for all the signal redirection. We also think of the following two operations as primitives, although they are built from deflections and reflections.

**Double Deflection:** In this maneuver, we deflect the signal 90 degrees, and then deflect it back along its original heading. This changes the diagonal along which the signal is moving, but not the direction. The deflections do not change the path of the original signal, so unless stated otherwise, a double deflection operation also includes a reflection to divert the original signal.

**Double Reflection:** When we reflect a particle twice, it will end up moving the same direction, along the same diagonal, but further back than before. The time between the two reflections controls how far the particle moves backwards.

Figure 5 illustrates all four basic operations, and shows how a signal  $x$  is manipulated by the other particles.

Let us return to our earlier goal: redirecting the four groups ( $C_{NE}$ ,  $C_{NW}$ ,  $C_{SE}$ ,  $C_{SW}$ ) to move northeast. The plan for each group is as follows.

**Northeast** The signals in  $C_{NE}$  are already moving northeast. We do not need to do anything with them until we have all the other signals moving northeast too.

**Southwest** The group  $C_{SW}$  is moving southwest, so we can use reflections to reverse the signals. One could reflect each particle individually, but it is easier to reflect the entire group by having a line of  $O(n)$  particles moving southeast meet a line of  $O(n)$  particles moving northwest. The two lines form a “mirror” to reflect the group.

**Southeast** The signals in  $C_{SE}$  are moving southeast, so we need to deflect them northeast. There is no shortcut to deflect the entire group at once (as we did for  $C_{SW}$ ), but a single particle moving northwest may intercept many signals in  $C_{SE}$ , and we can reuse it for deflection of all those signals in quick succession.

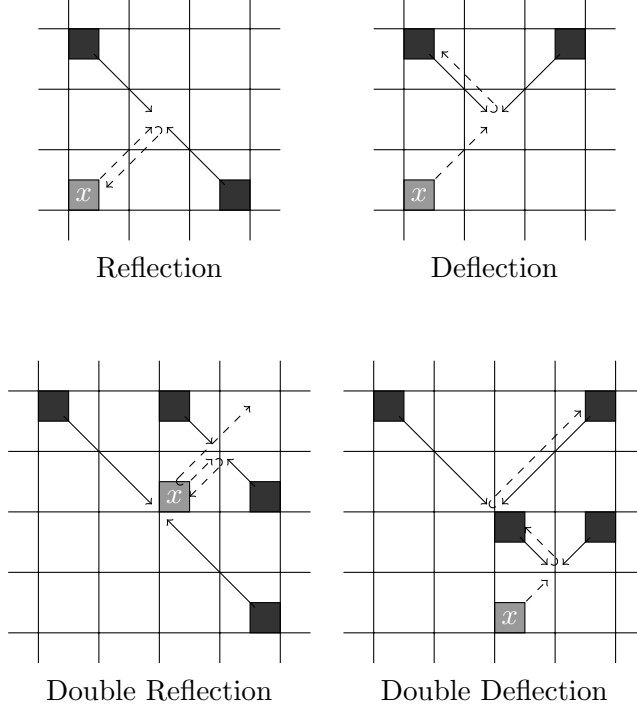


Figure 5: Basic operations for manipulating a signal,  $x$ , representing a bit of data. The flow of information is indicated by dashed lines.

**Northwest** The procedure for deflecting  $C_{NW}$  is essentially the mirror image of the procedure for  $C_{SE}$ . However, particles passing through  $C_{NW}$  (going southeast) may meet the particles passing through  $C_{SE}$  (going northwest). These intersections are harmless unless they meet a signal from  $C_{SW}$  moving northeast (having been redirected by our mirror) and reflect it back southwest. We can avoid this kind of interaction by changing the precise timing of  $C_{SW}$ ,  $C_{SE}$  or  $C_{NW}$  redirections, so we will assume it does not happen.

Now suppose the four groups have been redirected and all the information from the input is contained in a group of  $4n^2$  signals moving northeast. We can reflect any other particles or signals moving northeast, so we assume only the input signals are moving northeast, and we redirect these signals into a column as follows.

1. If there are two or more particles moving along the same (northeast/southwest) diagonal then move them to separate diagonals using a double deflection maneuver. The signals from  $C_{SE}$  and  $C_{NW}$  should already be spread out (over approximately  $O(n^2)$  diagonals) by the earlier deflections, but the signals from  $C_{NE}$  and  $C_{SW}$  are contained in just  $O(n)$  diagonals. Computation on the column formation will be relatively efficient whether the signals are consecutive or not.
2. Identify the westmost signal in the group. Use double reflections to move each particle backwards (i.e., southwest) along the northeast/southwest diagonal until it is in the same column as the westernmost signal.

The plan above lists the collisions required to redirect the signals into column formation, but only at a high level. The exact time and coordinates of each collision are too tedious to specify here, so scheduling is left as an exercise to the reader. We make the following observations about scheduling collisions.

- It is easy to determine the initial configuration required to produce a collision at a given

time and place, assuming nothing interferes with our particles along the way. We simply work backwards from the collision site.

- Any accidental interaction requires at least three particles (or signals). Each of the three particles is either on the way to an (intentional) interaction, or the product some earlier interaction. In almost all cases, we can move one of the three interactions such that the particles do not coincide.
- We have plenty of time and space to spend making sure there are no accidental interactions. We claim that an efficient schedule can perform the redirections above in  $O(n^2)$  time, but scheduling becomes much easier if we give ourselves polynomial time. For instance, when we deflect  $C_{SE}$ , we can wait  $O(n)$  steps after each deflection to ensure the particles and signals from one deflection are completely clear of  $C_{SE}$  before we start the next deflection.
- We can apply the diffusion theorem to prevent unintended collisions. For instance, if it takes  $O(n^2)$  time to redirect the four groups, then all signals and particles used up to that point must be in a region of size  $O(n^2)$ . After another  $O(n^2)$  steps, none of the particles from that region will ever interact with each other again (by the diffusion theorem).

### 4.3 Logical Operations and Computation

The next step is to show that we can perform computations on the signals in column formation. Specifically, we will show how to append a signal corresponding to the NOR of some subset of the other signals. This is enough to perform arbitrary computations, since NOR is a universal gate.

**Theorem 5.** *Let  $X = \{x_1, \dots, x_{n+m}\}$  be a group of signals moving northeast in column formation, where signals  $x_1, \dots, x_n$  encode bits  $b_1, \dots, b_n \in \{0, 1\}$  respectively, and  $x_{n+1}, \dots, x_{n+m}$  are empty.*

*Suppose we are given sets  $S_1, \dots, S_m$  where  $S_i \subseteq \{1, \dots, n + i - 1\}$ . Define  $b_{n+1}, \dots, b_{n+m}$  such that*

$$b_{n+i} = \overline{\bigvee_{j \in S_i} b_j}.$$

*Then we can initialize  $\bar{X}$  such that after  $t$  timesteps, the signal  $x_j$  encodes bit  $b_j$ , for all  $1 \leq j \leq m + n$ . In other words, we can modify the last  $m$  signals so each one is the NOR of a given set of earlier signals.*

*Furthermore,  $t$  is in  $O(n + m)$  and the number of particles in  $\bar{C}$  is*

$$3m + \sum_{i=1}^m |S_i|.$$

*Proof.* Consider an arbitrary signal  $x_{n+i}$  where  $1 \leq i \leq m$ . We construct particles  $p_i$ ,  $q_i$  and  $r_i$ , moving southeast, southwest, and northwest respectively. We position the particles so they all meet (at the same time) with  $x_{i+n}$ , assuming they are not reflected earlier. Furthermore, we position  $p_i$  north of all signals in  $X$  and north of  $p_1, \dots, p_{i-1}$ . The idea is that if  $p_i$  is not reflected back northwest before it meets  $r_i$ , then  $p_i$  and  $r_i$  will reflect  $q_i$  into  $x_{i+n}$  (i.e., signal  $x_{i+n}$  will contain a particle). Otherwise  $q_i$  will continue moving southwest and  $x_{i+n}$  will continue to be empty.

We also construct a particle  $s_{ij}$  for each  $1 \leq i \leq m$  and  $j \in S_i$ . Let  $s_{ij}$  move southwest and intercept  $p_i$  and  $x_j$  at the same time. The result is that  $p_i$  will be reflected if  $b_j = 1$ , unless  $p_i$  has already been reflected. Hence,  $p_i$  only reaches  $x_{i+n}$  if  $b_j = 0$  for all  $j \in S_i$ . It follows that

$$b_{n+i} = \overline{\bigvee_{j \in S_i} b_j},$$

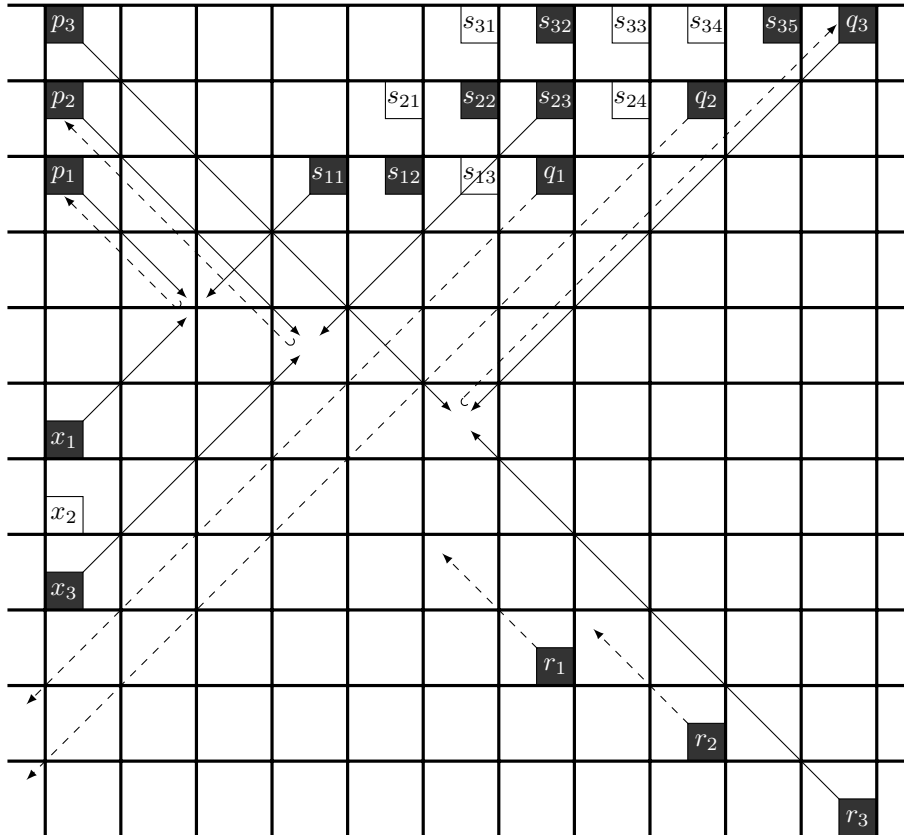


Figure 6: An example of a small computation on bits  $x_1, x_2, x_3$ .

as desired.

Observe that only  $x_1, \dots, x_{n+m}$  and  $p_1, \dots, p_m$  are moving eastwards (i.e., northeast or southeast). Each  $p_i$  is positioned to collide with the corresponding  $x_{i+n}$ , so they must be in the same column. It follows that all eastbound particles are in the same column. Collisions require at least three particles, at least one of which must be eastbound, so all collisions must occur in the same column as the  $x_j$ s. We positioned the particles ( $p_i, q_i, r_i$  and  $s_{ij}$ ) based on where and when they meet the  $x_j$ s, so the only collisions are the ones we have already discussed.

Suppose we place  $p_1$  immediately above  $x_1$  and  $p_2, \dots, p_m$  in order above that. Then clearly  $p_m$  reaches  $x_{m+n}$  in  $t = O(m+n)$  time. It is also clear that we use only  $3 + |S_i|$  particles to modify  $x_{i+n}$ , and hence only

$$3m + \sum_{i=1}^m |S_i|$$

particles altogether. □

Figure 6 shows the construction on three bits,  $x_1 = 1$ ,  $x_2 = 0$  and  $x_3 = 1$ . We add  $x_4 = \overline{x_1 \vee x_2} = 0$ ,  $x_5 = \overline{x_2 \vee x_3} = 0$  and  $x_6 = \overline{x_2 \vee x_5} = 1$  over the course of the computation. The black cells are particles, and white cells are locations where particles are conspicuously absent. For instance, we could place particles at  $s_{13}, s_{21}$ , and so on, to specify a different circuit. We use arrows to indicate the motion of key particles, and we use a dashed arrow if the particle does not interact with any other particles. We can see that  $p_1$  and  $p_2$  are deflected (by  $x_1$  and  $x_3$  respectively), but  $p_3$  makes it all the way to  $q_3$  and  $r_3$ , so it can reflect  $q_3$  into position as  $x_6$ .

We use this theorem to perform a series of computations, with computations for the original transformation,  $f: \Sigma^X \rightarrow \Sigma^X$ , occurring somewhere in the middle. Before we can apply  $f$  to the initial configuration,  $x \in \Sigma^X$ , we must decode  $x$  from the  $4n^2$  signals captured from the four square groups. Similarly, after we have computed the final configuration  $f(x) \in \Sigma^X$ , we need to encode it as  $4n^2$  signals. We arrange the signals into four groups, which converge on  $X$  to produce  $f(x)$ . This last step is the subject of the following section.

#### 4.4 Output Signal Redirection

The final step of the construction is to insert the desired configuration,  $f(x)$ , into  $X$ , the output region. The diffusion theorem tells us we can insert  $f(x)$  by making four groups of carefully designed signals meet in the output. The signals in the four groups were computed in the previous section, so the problem is to redirect a column of  $4n^2$  signals into four groups, converging on  $X$ . This is essentially the reverse of an earlier step, where we redirected the four groups of signals into a single column.

Unfortunately, literally reversing our steps, in an attempt to exploit the reversibility of the CA, is not a practical way to put information into the output. To illustrate, suppose we implement our procedure and let the system evolve up to the point where the contents of the input region are laid out as a column of signals (i.e., just before we begin computation). Then we change some of the signals in the column, and run CA backwards for the same number of timesteps. In this situation, the contents of the input region will not reflect the changes we made to the column. The problem is that many cells, not just the ones in the column, depend on a given cell of the input. Our construction copies information every time we perform a deflection operation, and we leave intermediate steps around in our computation. We have to change all of these cells to change the value of the input cell. Since there are too many cells to practically change, we must find another way to populate the output.

As before, let the four groups of signals be called  $C_{NE}$ ,  $C_{NW}$ ,  $C_{SW}$ , and  $C_{SE}$ , where the subscript indicates the direction of motion of the group when it eventually converges on  $X$ . Initially, the signals are in column formation, moving northeast, along with signals leftover

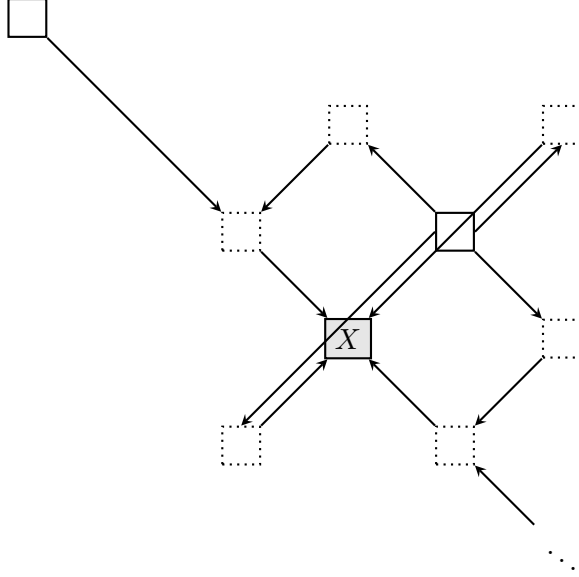


Figure 7: Idealized paths of signal groups returning to  $X$ .

from the intermediate computations. We make a few preliminary adjustments to the signals as follows.

1. Reflect all signals moving northeast except the  $4n^2$  designated signals. We do not want any interference.
2. Use a series of double deflections to move the  $4n^2$  designated signals so they are approximately northeast of  $X$ . That is, they are not too far from the northeast/southwest diagonal passing through the center of  $X$ .
3. Use further double deflections and double reflections to rearrange the signals into four square formations corresponding to the four groups. Within a group, each signal should have the same offset from its intended destination, except within  $C_{SW}$ . We will move  $C_{SW}$  into final position by reflection, so we expect each signal to have the same offset to the *reflection* (over a northwest/southeast diagonal) of its destination. For  $C_{SW}$  and  $C_{NE}$ , we expect the offsets to be strictly northeast (no steps northwest or southeast).

At this point, we wait until the signals are, as a group, some suitably large distance  $D$  northeast of  $X$ . The plan is to have the signals diverge at this point, and converge on  $X$  in approximately  $3D$  steps, at some target time  $t$ . Conceptually, we redirect the four groups as follows.

**Southwest** Signals in  $C_{SW}$  continue northeast until they are  $2D$  units from  $X$ , and then we reflect them back towards the origin. No other manipulations are necessary because the signals in  $C_{SW}$  are already in the correct square formation.

**Northeast** We reflect signals in  $C_{NE}$  back to the southwest, and let them travel  $2D$  units before reflecting them northeast again. The two reflections are sufficient because  $C_{NE}$  is already in a square formation.

**Southeast** We deflect signals in  $C_{SE}$  northwest for about  $D$  units. Then we deflect the signals southwest for  $D$  units, and finally southeast for the final approach. The final deflection (to the southeast) is different from the other deflections; we use the incoming signal from the northeast and a particle from the southwest to reflect a particle coming from the northwest because a particle from the southeast would have to pass through  $X$  at an awkward time.



In other words, there is a group of particles in square formation moving southeast towards  $X$ , and the final deflection whittles away particles to form signals, instead of reflecting new particles into empty space.

**Northwest** We redirect signals in  $C_{NW}$  as the mirror image  $C_{SE}$ . We will see that it is not an exact mirror image in practice.

See Figure 7 for an idealized illustration of the paths. In practice, we cannot redirect all four groups simultaneously. Even if we could, deflecting  $C_{SE}$  and  $C_{NW}$  would likely create unwanted collisions. As a result, there may be some distance (or time) between the ideal initial redirection of a signal, and the actual initial redirection. We assume the distance is small relative to  $D$ , and use the subsequent redirections to correct the error as follows.

- Define  $D$  such that there is no error in group  $C_{SW}$ . The other groups are supposed to divert at distance  $D$ , but may experience some delay. Since we assume the delay is small relative to  $D$ , the other groups should be gone when it is time to reflect  $C_{SW}$ .
- As long as the initial reflection of  $C_{NE}$  is less than  $D$  steps late, it will pass through  $X$  before the designated time  $t$ . As the time remaining (until the groups converge) decreases, the distance between  $C_{NE}$  and  $X$  increases. At some point the time remaining is equal to the distance, and we schedule the second reflection relative to that point.
- We adjust signals in  $C_{SE}$  using the second deflection, at a location approximately  $2D$  cells north of its final location. Recall that we want each signal in  $C_{SE}$  to meet a particle moving southeast, so we must deflect the signal southwest when it is in the same row of cells as the particle. Then the signal is to the east of the particle, so the two will eventually collide.

Observe that if we delay the first deflection, it does not change the time of the second deflection, since the signal is moving north (either northeast or northwest) at the same rate before and after the first deflection. The delay does move the location of the second deflection east.

It is possible for the “parity” of a signal to be such that these operations are impossible. That is, if a particle is  $x$  blocks right of its target position and  $y$  blocks above, then it will be impossible to maneuver it into position if  $x+y$  is odd. Fortunately, we completely control parity in the computation stage, so it is not a serious obstacle.

We argue that intentional collisions only occur near (i.e., much less than  $D$  units away from) vertices in Figure 7. Any unintentional collisions must also occur near vertices, since there must be intentional collisions in three directions. We divide unintentional collisions into two types: *local collisions* where at least one signal has travelled much less than  $D$  units (i.e., the signal comes from the vertex at which the collision occurs) and *non-local collisions*.

We claim unwanted local collisions can be eliminated by rescheduling intentional collisions. Unwanted non-local collisions can only occur at  $X$  or the vertex northeast of  $X$ , since these are the only vertices with three equidistant neighbours. There are no unwanted collisions at  $X$ , since the only signals moving along the northwest/southeast diagonal are  $3D$  steps away, and will arrive at time  $t$ . There is a threat of unwanted collisions at the vertex northeast of  $X$ , but we can schedule the deflections of  $C_{SE}$  and  $C_{NW}$  such that no two signals are deflected into the same southeast/northwest diagonal. Then there is no chance of a signal moving southeast meeting a signal moving northwest in that area. We conclude that there are no unwanted collisions.

Since only intentional collisions occur, we will redirect the signals into four groups, which will converge on  $X$  to create the desired configuration at time  $t$ . We have successfully implemented the transformation  $f$  on the region  $X$ , and since both were arbitrary, it follows that our CA is physically universal.

Moreover, note that all operations take polynomial time. Redirecting the input takes  $O(n^2)$  time where  $n$  is the diameter (in blocks) of the input. Performing the computation is  $O(m)$  (where the transformation takes  $m$  gates), after  $O(n^3)$  to decode the input. Encoding the

output is  $O(n^3)$ , then redirecting it to the output takes time proportional to the rest of the computation. It is certainly polynomial in  $n$  and  $m$ , so we have shown our CA is efficiently physically universal. To give a sense of the constants involved, we estimate a transformation with  $n = 5$  and  $m$  negligible would take about  $t = 50000$  timesteps by the procedure above.

## 5 Future Work

There are many unanswered questions and conjectures about physical universality, our cellular automaton, and this specific physical universality construction.

1. Are any CAs in the literature physically universal? The Hardy-Pazzis-Pomeau gas [6] is a good candidate because the diffusion theorem holds, it is possible to arbitrarily redirect particles, and one can construct universal gates<sup>2</sup>. We conjecture the HPP gas is physically universal as well.
2. Do physically universal CAs exist in other dimensions? We conjecture that slight modifications of our CA are physically universal in higher dimensions. We also conjecture the existence of one-dimensional physically universal CAs, although we need more than two states because all twenty-four reversible, binary, Margolus-neighbourhood CAs are incapable of simulating circuits.
3. Are there physically universal CAs with no analogue of the diffusion theorem? We already know enough to give an unsatisfying example: a colour-inverted version of our CA, where white becomes black and black becomes white. Single black cells are stationary, so they do not diffuse out of the region. Our colour-inverted construction surrounds the input region with a wave of black cells, which flood into the input region, forcing the white cells out in predictable locations.

More generally, is there a physically universal CA with a qualitatively different approach to computation?

4. Is there a physically universal CA (or another construction for our CA) which achieves better performance (i.e., number of timesteps<sup>3</sup>) with respect to the size of the input ( $n$ ) or the number of NOR gates ( $m$ ) in the optimal circuit for the transformation?
  - (a) The dependence on the complexity of the transformation,  $O(m)$ , is good, but not optimal. Using  $m$ , the size of a circuit for the transformation, as the measure of complexity misses the opportunity for parallel execution in cellular automata. Circuit depth is also an unsuitable measure of complexity because it ignores the cost of communicating between bits separated by a great distance.

A better way to measure complexity is relative to other cellular automata. We say a physically universal CA is *competitive* if it can simulate any other CA with at most a constant factor slowdown. Does a competitive physically universal CA exist?

- (b) The dependence on the input size is  $O(n^3)$  in our construction, which is most likely not optimal. The bottleneck (in our case) is working backwards to find the original input, because we have to simulate  $O(n^2)$  cells backwards for  $O(n)$  timesteps. If we could perform the simulation more efficiently, we might improve the performance to the next bottleneck at  $O(n^2)$ . Rearranging the particles into column formation is unavoidably  $O(n^2)$ , and we currently take  $O(n^2)$  time to deflect  $n^2$  particles.

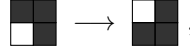
The ultimate lower bound is  $O(n)$ , since it takes at least  $O(n)$  steps for information to travel from one corner of the region to another. On the other hand, there is evidence that the key steps, rearranging particles and simulating a cellular automata, can be made to run in  $O(n)$  time in other CAs.

---

<sup>2</sup>In fact, Moore and Nordahl [9] showed that the HPP gas is  $P$ -complete.

<sup>3</sup>The number of non-blank cells in the initial configuration is another possible measure of performance. We will not discuss it, but there are clearly interesting questions in that area.

- Consider a variation of our cellular automaton where the rule for three-particle interaction is



with rotational symmetry. In this CA, a signal meeting one particle head-on and another particle from the right will turn to the right without disrupting the other particles. It is possible to deflect an  $n$ -by- $n$  square of particles to the right in  $O(n)$  time.

- It is also possible to simulate Conway's Game of Life within itself with only a constant factor slowdown using a metapixel construction. Take, for instance, the p5760 unit Life cell, invented by David Bell in 1996. It is a 500-by-500 square structure, which takes 5760 timesteps to simulate one step of a single cell in the Game of Life. Other metapixel constructions can be programmed to simulate other cellular automata.

Is there a physically universal CA which combines these traits to achieve better performance?

5. Can we extend physical universality to unbounded computations? Under the current definition, we specify the time  $t$  at which the computation will finish, independent of the contents of the input region. Suppose we drop this requirement and allow the CA to signal that computation is complete by some other means (e.g., the first time a certain cell is non-empty, the output region will contain the desired output).
  - Suppose we are given the transformation  $f$  as a Turing machine  $M$ , and promised that  $M$  will produce an output that fits in  $X$  if it halts. The representation of  $f$  is irrelevant under the usual definition of physical universality, since we only care if there *exists* an initial configuration which implements the transformation. What if we have to *construct* an initial configuration which implements the transformation?
  - In the definition of physical universality, we know the input region, and therefore the size of the input, when we construct a configuration to implement the transformation. Is there a physically universal CA such that we specify the configuration of  $\bar{X}$  in some form before we know the size of  $X$ ? For instance, in the one-dimensional case we could specify a prologue and epilogue, to be placed on either side of the input, without knowing the size of the input.
  - Even without an explicit deadline  $t$  in the definition of physical universality, the diffusion theorem bounds the length of time a finite initial configuration of our CA can perform computation. One possible solution is to allow infinite configurations which are periodic (in some sense). Can we perform unbounded computations in our CA with infinite periodic initial configurations?
6. Consider the following variation of physical universality.

**Definition 6.** We say a cellular automaton over  $\Sigma$  is *reversibly physically universal* if for any region  $X$  and any bijection  $f: \Sigma^X \rightarrow \Sigma^X$ , there exists a constant  $t_0$  and configurations  $y, y' \in \Sigma^{\bar{X}}$  such that for any configuration  $x \in \Sigma^X$ , the configuration  $x \oplus y$  evolves to  $f(x) \oplus y'$  in  $t_0$  steps.

Note that  $f$  must be a bijection, otherwise there exist  $x_1, x_2 \in \Sigma^X$  such that  $f(x_1) = f(x_2)$ , so  $x_1 \oplus y$  and  $x_2 \oplus y$  evolve to  $f(x_1) \oplus y' = f(x_2) \oplus y'$ , contradicting reversibility. This is motivated by physical universality in a quantum setting, where unitarity forces all interaction to be reversible in this sense.

Do reversible physically universal CAs exist? Our CA is unlikely to be reversibly physically universal because deflection, the basic operation we depend on to rotate signals 90 degrees, works by copying a signal. That is, if we deflect a signal  $x$ , the original signal actually continues unaffected and we create two new signals,  $x$  and  $\neg x$ , moving in orthogonal directions. A *reversible* physically universal CA is not allowed to let the input configuration

affect the environment, so it must catch those signals and return them to the output region. If we change three-particle interaction in our CA to

$$\begin{array}{|c|c|} \hline \blacksquare & \blacksquare \\ \hline \blacksquare & \blacksquare \\ \hline \blacksquare & \blacksquare \\ \hline \blacksquare & \blacksquare \\ \hline \end{array} \rightarrow \begin{array}{|c|c|} \hline \blacksquare & \blacksquare \\ \hline \blacksquare & \blacksquare \\ \hline \blacksquare & \blacksquare \\ \hline \blacksquare & \blacksquare \\ \hline \end{array},$$

then, as discussed earlier, there is a clean way to redirect a signal by 90 degrees. We also know the diffusion theorem holds, so we understand how to extract information from the input region and inject a desired configuration into the output region. Let us assume, for the sake of argument, that we can arbitrarily redirect signals in this CA. In that case, it is equivalent to a circuit/gate model where  $\rho$ , represented by a four input/four output gate, is the only available gate. The gate is universal, so we can perform any reversible computation.

The cellular automaton is almost a reversible physically universal CA, but there is a problem: the new CA, like its predecessor, conserves particles. The total number of particles remains constant, and the number of particles in  $\bar{X}$  is fixed at time  $t = 0$  and at time  $t = t_0$ . Hence, the transformation must be such that the number of particles in  $x$  and  $f(x)$  differ by a constant, otherwise we cannot implement it. Since the full region cannot map to a configuration with more particles, nor an empty region map to a configuration with fewer particles, the constant must be zero. In other words, the CA is unable to implement any transformation which does not conserve the number of particles. We believe a non-conservative CA can overcome this limitation, and conjecture that reversible physically universal CAs exist.

7. What can we say in the quantum case? A physically universal QCA should be able to implement any unitary transformation on any finite region, but the following details are unclear.
  - Should the implementation of the unitary be exact, or is it enough to produce a sequence of approximations?
  - Do we need  $\bar{X}$  to be a general quantum state? Can we still implement unitary transformations if we restrict ourselves to separable (or even classical, if we only have to approximate the unitary) configurations of  $\bar{X}$ ?
8. With our CA, after the transformation is finished, particles continue to escape outwards at the speed of light. Are there physically universal CAs such that the size of the active region does not grow so rapidly?
  - If the size of the active region is constant then the CA must eventually repeat a configuration. Since the CA is deterministic, once the CA repeats a configuration, it is periodic forever. A physically universal CA of this type could start with configuration  $x$  in the input region, evolve to  $f(x)$  in the output region, and then eventually evolve back to  $x$  in the input region.
  - If the CA is unbounded then the number of cells is at least logarithmic; if we run for more than  $|\Sigma|^n$  timesteps without repeating then the active region is at least  $n$  cells wide by pigeonhole principle. There exist reversible CAs with this kind of logarithmic growth, but none are known to be physically universal.

## 6 Acknowledgements

This research was supported by the National Science Foundation Alan T. Waterman Award, grant number 2745557. The author would also like to thank Scott Aaronson for the problem, and for help preparing this paper.

## References

- [1] C. H. Bennett. Logical reversibility of computation. *IBM J. Res. Dev.*, 17(6):525–532, November 1973.
- [2] Elwyn Berlekamp, John H. Conway, and Richard K. Guy. *Winning Ways for Your Mathematical Plays*, volume 2. A. K. Peters, 1st edition, 1982.
- [3] Matthew Cook. Universality in Elementary Cellular Automata. *Complex Systems*, 15(1):1–40, 2004.
- [4] David Deutsch. Constructor Theory. *ArXiv e-prints*, October 2012.
- [5] Martin Gardner. The fantastic combinations of John Conway’s new solitaire game “life”. *Scientific American*, 223:120–123, October 1970.
- [6] J. Hardy, O. De Pazzis, and Y. Pomeau. Molecular dynamics of a classical lattice gas: Transport properties and time correlation functions. *Physical Review A*, 13(5):1949–1961, 1976.
- [7] Dominik Janzing. Is there a physically universal cellular automaton or Hamiltonian? *ArXiv e-prints*, September 2010.
- [8] Norman Margolus. Physics-like models of computation. *Physica D: Nonlinear Phenomena*, 10(12):81–95, 1984.
- [9] Cristopher Moore and Mats G. Nordahl. Lattice Gas Prediction is P-complete. *Contributions to Mineralogy and Petrology*, page 4001, April 1997.
- [10] Tommaso Toffoli and Norman Margolus. *Cellular Automata Machines: A New Environment for Modeling*. MIT Press, Cambridge, MA, USA, 1987.
- [11] John von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, Champaign, IL, USA, 1966.