Chapter 1

Introduction to Symbolic Dynamics and Cellular Automata

Cellular automata are mathematical systems that are simple to describe but which have the potential to exhibit complex behavior over time. Instead of using the uncountable and unbounded set of real numbers, or points in higher dimensional space with real coordinates, we discretize our space into finitely many descriptive possibilities for measuring a property of interest. Often, symbols such as colors, letters of the alphabet, or finite sets of integers are used to represent the state of a space at a location in the space. We then have a simpler space that we refer to as *symbol space*. The goal is to replace uncountably many possible numbers representing locations and changing values of a quantity under a process (for example, measuring the temperature of a thin rod) by one of finitely many possible states (such as red for hot and blue for cold, measured only at ten different points along the rod). Symbol space is defined in Section 1.1.2 and is discussed in detail in Chapter 2. In the case of the heated rod, symbolic coding means that we replace the temperature distributed over the rod by a 10-tuple of Rs (hot areas) and Bs (cool areas). We show the initial symbolic representation of a rod in Figure 1.1 and note that it is convenient to write the symbolic coding of the rod as follows: $x = RBBB \ldots BR$. The point x encodes the entire rod, not just one point along the rod. We will see other ways to present

Figure 1.1. The temperature of the rod on the left is coded hot (red) or cold (blue) at ten spots along the rod, shown on the right. In symbolic dynamics the coding for the rod can be written as $x = RBBB\ldots BR$.

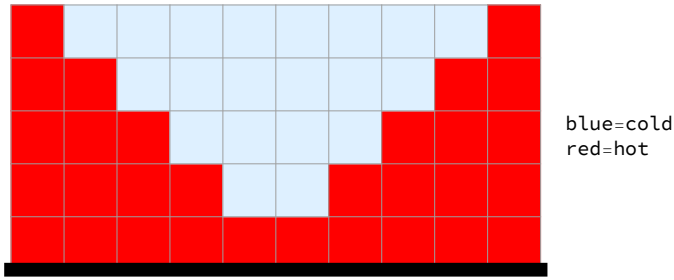

Figure 1.2. The rod is hot everywhere after four time steps, moving down one row at a time, under the rule described above.

symbol space. Symbol space and cellular automata are used to model processes that change over time, so a point in symbol space is not just one of the two colors, red or blue, hot or cold, rather it is a sequence of finitely many colors. The map we use represents how the temperature (in this example) changes over time. Applying a map to a symbol space leads to *symbolic dynamics*. In the setting described here, each of ten points along a metal rod is measured as either hot or cold. We then assume there is an elementary principle that applies, and this depends for example on the material of the rod. For now, we assume that cold refers to any temperature that is "not hot" (below some threshold), and heat travels quickly through the material. Therefore, we assume that a cold spot next to a hot spot gets hot, and a hot spot next to a cold spot stays hot unless it is squeezed between two cold spots. This is the description of a map representing one time step on the points along the rod, and one that leads to a cellular automaton (the singular form of cellular automata). We show the rod at the bottom of Figure 1.2; in four time units it has turned completely hot. It follows that if we call the map f, then $f(x) = RRB\ldots BRR$, as shown in the second row from the top of Figure 1.2, and $f \circ f \circ f \circ f(x) = RR\ldots R$, as shown in the bottom row.

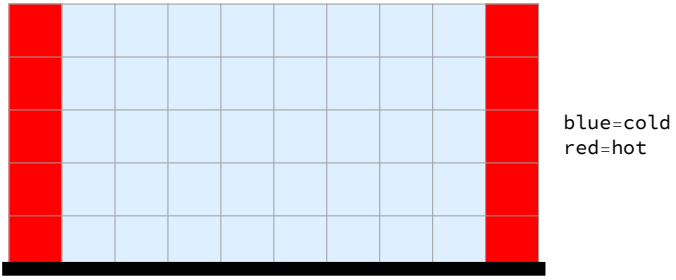

Figure 1.3. The temperature of the rod remains unchanged in four time steps under a small local rule change described below.

Cellular automata are typically studied as maps from a symbol space to itself that are updated at each location in space independently, locally, and using the same update rules. To decide if a location along the rod is hot or cold, we only need to consider its temperature and that of its adjacent locations; this is called a *local rule* because it only depends on nearby locations and not those far away. A small change of local behavior has a big impact on the outcome. We could change our hypothesis to say that a cold spot must be sandwiched between two hot spots in order to get hot (but a hot spot next to a cold spot stays hot). Under this hypothesis, as illustrated in Figure 1.3, in four time units the temperature along the rod has not changed at all.

Working with symbolic dynamics offers a simple mathematical tool for modeling complex behaviors, both mathematical and physical. Cellular automata (CAs) are one type of symbolic dynamical system; there are others and we give an overview to place CAs in the mathematical context of this book. Nature and mathematics are incredibly complex; one approach to aid our understanding is to discretize continuous phenomena, cutting a disordered space into finitely many labeled regions to help digest a complex phenomenon in bite size pieces.

Example 1.1. An example of symbolic coding that we use all the time is the decimal expansion of a real number; each point on the uncountable real line gets boiled down to a collection of digits 0 through 9. Moreover, when we work with real numbers on a computer, they are almost always rounded off to a rational approximation. The rounding off is the same as assigning each real number to a small interval on the real line. The computer typically uses a round off value for an irrational number,

and even for rational numbers, such as using .667 for 2/3, when making computations.

When we multiply a decimal number by 10 we move the decimal point to the right one place and this is a demonstration of symbolic dynamics. This following shows how the map $f(x) = 10x$ works: for example if

$$\begin{aligned} x &= .1234567891011\ldots, && \text{then} \\ f(x) &= 1.2345678910\ldots, && \text{and} \\ f^4(x) &= 1234.5678910\ldots. \end{aligned}$$

It is a straightforward task to iterate f as many times as we want.

Example 1.2.

(1) We can expand real numbers in other bases. Consider the binary (base 2) expansion of the decimal number 12.375:

$$\begin{aligned} 12.375 &= 12\ 3/8 \\ &= 8 + 4 + (0 \times 2) + (0 \times 1) + (0 \times 1/2) + 1/4 + 1/8 \\ &= 1100.011_2. \end{aligned}$$

We see that 12.375×4 has base 2 expansion obtained by moving the (binary) point two digits to the left, so it is 110001.1. This can be translated back to decimal as follows:

$$\begin{aligned} 110001.1 &= 32 + 16 + 1 + 1/2 \\ &= 49.5, \end{aligned}$$

providing a nice demonstration of the uses of symbolic dynamics.

(2) Computing 12.375×8 is now an easy exercise, so we leave it to the reader to execute that in base 2 coding and then translate back to a decimal number.

When physical (or mathematical) processes are modeled on symbol spaces using CAs, they are computable. These models are computable because each coordinate is updated by looking only at it and a finite number of nearby coordinates. Hadamard, Turing, von Neumann, and Wolfram are a few mathematicians who believed that complex changing systems can be analyzed in this discrete and local way, and in doing so

changed mathematics. John H. Conway found these mathematical systems so interesting that he developed a game, called the Game of Life, to encode the basic principles of "life" (birth and death). The Game of Life CA is discussed in Chapter 6. In this chapter we give an overview of how and why we use these techniques, with examples, and give an application of cellular automata to traffic congestion.

1.1. Coding and symbolic dynamics

A code is a rule for converting information from a source into symbols to be communicated. Here we show the output from an experiment, a repeated action whose outcome is not known in advance, that forms the basis for many seemingly random mathematical systems. Looking first at Figure 1.4 then next at the sequence in (1.1), it seems that the "code" presented in Figure 1.4, and the "message" implemented using the code in (1.1), conveys the passage of time:

$$\text{H H T H T T H T T H H H T} \ldots. \tag{1.1}$$

The sequence in (1.1) is shorthand for the outcome of an experiment. It could be repeated forever, so a point is an infinite mathematical sequence. Mathematically, we typically represent the experiment in (1.1) using binary digits:

$$x = 0\,0\,1\,0\,1\,1\,0\,1\,1\,0\,0\,0\,1 \ldots.$$

The sequences of Hs and Ts or 0s and 1s, represent the possible outcomes of an experiment of tossing a coin over and over. A fair coin toss is one where 0s and 1s appear with equal frequency as the experiment is repeated over and over. The *unfair coin toss* setting, where the occurrence of a 0 has probability p with $p \in (0,1)$, $p \neq 1/2$, reappears for study in Chapter 7. In Chapter 2 we analyze how to impose a formal mathematical framework on this coding process. Before that, we conduct our own

Figure 1.4. Heads or tails? We encode the outcomes of coin tosses using H or 0 for heads and T or 1 for tails.

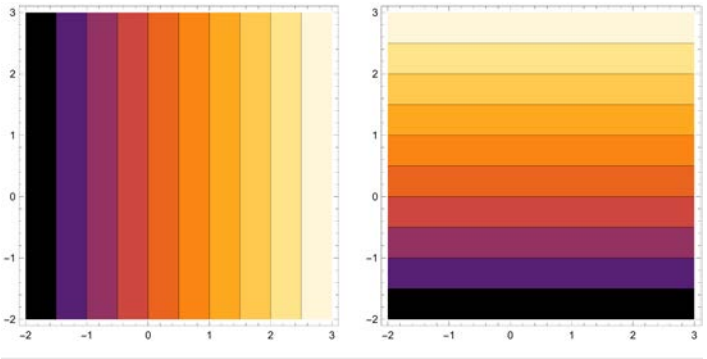

Figure 1.5. We show how the color coding works for $f_1(x, y) = x$, $f_2(x, y) = y$. Each colored region represents a range of values. On the left we show $f_1(x, y)$ and on the right, $f_2(x, y)$.

experiments on familiar math objects to gain further understanding of mathematical coding theory.

We can code mathematical functions using color, and it is possible to combine the coding with the passage of time using multiple graphics and color. Figure 1.5 shows the code we use to distinguish large positive values from zero and negative values (large and small). In Figures 1.6 and 1.7 we use that coding consisting of light and dark colors: the darkest color shows the lowest value (usually negative), and the lightest is the largest value (positive). Each function takes on a continuum of values while the color banding makes their differences apparent. Moreover, the coloring can also reveal how the values of the function develop over time as well as the periodicity of the function, if any exists, as the next example shows.

Example 1.3. In Figure 1.5 we color code the values of the simple functions $f_1(x, y) = x$ and $f_2(x, y) = y$ by letting the darkest points be the smallest value, letting the lightest points be the highest values, and changing color at intervals of length 1/2. For example, we color black the set of points, $B = \{(x, y) : -2 \leq f_j(x, y) \leq -1.5\}$ for $j = 1, 2$. Using the same coding-by-color scheme, in Figure 1.6 we consider two real-valued functions of $(x, y) \in \mathbb{R}^2$, given by

$$f_1(x, y) = \sin x \cos 2y \qquad \text{and}$$
$$f_2(x, y)) = \cos x \sin y.$$

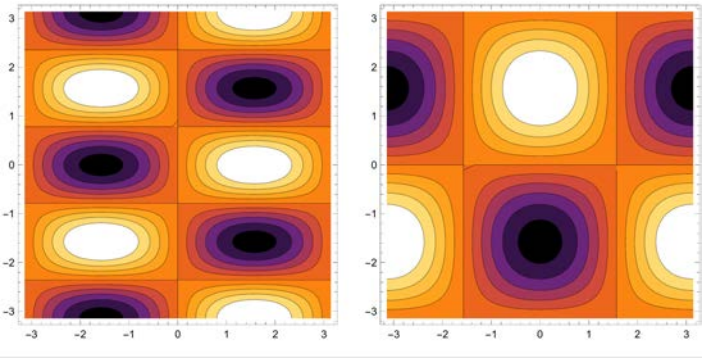

Figure 1.6. We consider the functions $f_1(x,y) = \sin x \cos 2y$ and $f_2(x,y) = \cos x \sin y$ with their values coded according to the same color coding used in Figure 1.5.

The range of each function is not a subset of its domain, but we can consider the function

$$F(x,y) = (f_1(x,y), f_2(x,y))$$

as a map $F : \mathbb{R}^2 \to \mathbb{R}^2$. Then we could view the left side of Figure 1.6 as the values of the first coordinate and the picture on the right as a coding of the second coordinate values of the function F. The advantage of this approach is that we can consider $F^2(x,y) = F \circ F(x,y)$ and look at the coding of the first and second coordinates of $F^2(x,y) = (\phi_1(x,y), \phi_2(x,y))$, as shown in Figure 1.7.

Example 1.4. We consider the function $G : \mathbb{R}^2 \to \mathbb{R}^2$ defined by

$$G(x,y) = (g_1(x,y), g_2(x,y)) = (x^2 - y^2, 2xy).$$

Since the graph of G is impossible to draw in three-dimensional space, we use a similar but coarser (with fewer colors) version of the color coding used in Figures 1.5–1.7. Therefore, we use the following color code in Figure 1.8:

$$\begin{aligned}
\text{darkest blue} &= \{(x,y) \ : \ g_j(x,y) \le -1\},\\
\text{light blue} &= \{(x,y) \ : \ g_j(x,y) \in (-1,0)\},\\
\text{lightest blue} &= \{(x,y) \ : \ g_j(x,y) \in (0,1)\},\\
\text{light yellow} &= \{(x,y) \ : \ g_j(x,y) \ge 1\}.
\end{aligned}$$

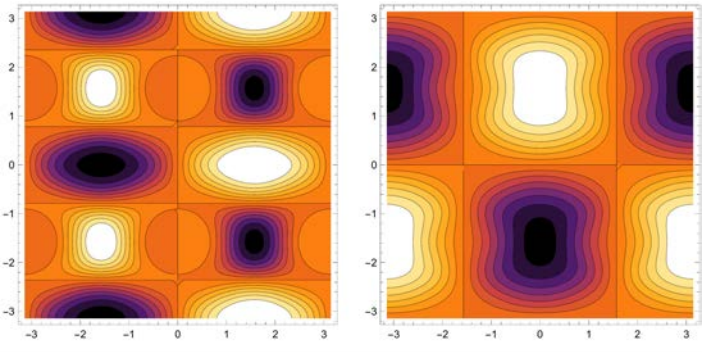

Figure 1.7. The functions f_1 and f_2 can be distinguished further by iterating $F = (f_1, f_2)$ and using the same coding to show how the passage of time distinguishes the coordinate functions $\phi_1(x, y)$ and $\phi_2(x, y)$ of $F \circ F(x, y)$.

For $j = 2$ we draw a plane and give the color code for x and y between -2 and 2, as is done for g_1 in Figure 1.8. This coding can be done by hand, but using a computer works as well. The map on the right in Figure 1.8 shows level curves using the same coloring, for the real-valued map $H(x, y) = 2\sin(x)\cos(y)$; the periodicity of H is apparent with this coding.

Lab 1, at the end of the chapter in Section 1.4, discusses this type of coding and how it changes under iterations.

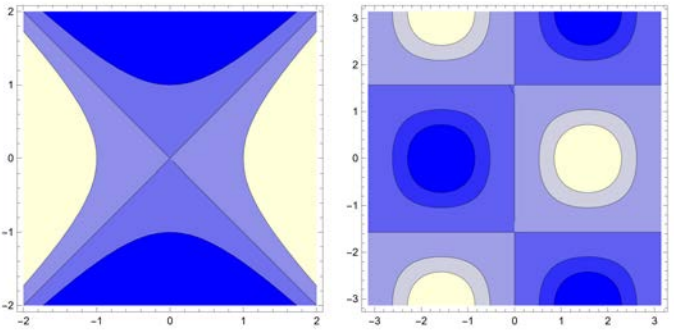

Figure 1.8. A color coding of $g_1(x, y)$ from Example 1.4 on the left, and on the right, a periodic function $H(x, y)$.

1.1.1. Symbolic dynamics. We now turn to the crux of symbolic dynamics. The goal is to follow the coding of a point under successive iterates of a map that often represent the passage of time or the repetition of an experiment. In many of the examples that follow, we use simple modular arithmetic whose notation is as follows.

$$\begin{aligned} &\text{If } x, y \in \mathbb{R}, \quad \text{we say} \\ &y = x \pmod 1 \quad \text{if} \\ &x = m + y \quad \text{for some } m \in \mathbb{Z}. \end{aligned} \tag{1.2}$$

We remark that $y = x \pmod 1$ if and only if $x = y \pmod 1$. In symbolic dynamics, the passage of time is shown by iterating a map $f : X \to X$, on a space X. It is useful to follow the path of a single point as time passes, so we need the following dynamical definition.

Definition 1.5. Let X be a set of points or, equivalently, a space. Suppose $f : X \to X$ is a map. Fix a point $x \in X$, and let f^n denote the n-fold composition of f with itself, e.g., $f^3(x) = f(f(f(x)))$. We assume that $f^0(x) = x$ for all x. The *orbit of* x (under f) is the set of points

$$\{f^n(x)\}_{n \in \mathbb{N} \cup \{0\}} \subset X.$$

If f is invertible, we let $f^{-1} : X \to X$ denote its inverse map. We then define the *two-sided orbit of* x under f to be

$$\{f^n(x)\}_{n \in \mathbb{Z}} \subset X.$$

Example 1.6. We consider the circle in the plane, $C = \{(x, y) : x^2 + y^2 = 1\}$, and parametrize the circle by θ,

$$(x, y) = (\cos(2\pi\theta), \sin(2\pi\theta)).$$

We partition the parametrized circle into two sets, using arc length. We assign to a point the symbol 0 if $\theta \in [0, 1/2)$, and label (x, y) with a 1 if $\theta \in [1/2, 1)$ as shown in Figure 1.9.

We define a map R on the unit circle according to arc length $\theta \in [0, 1)$, so $R : [0, 1) \to [0, 1)$. We choose and fix some $\beta \in (0, 1)$ and rotate through the angle $2\pi\beta$ to define a map $R(\theta)$, a function of a single variable with domain $[0, 1)$. Here we consider the map given by

$$R(\theta) = \theta + \beta \pmod 1, \text{ with } \beta = \sqrt{2}/4.$$

We start with the point $\theta_0 = 0$. We then code this point by assigning a 0 or 1 according to whether its image lies in the top half of the circle

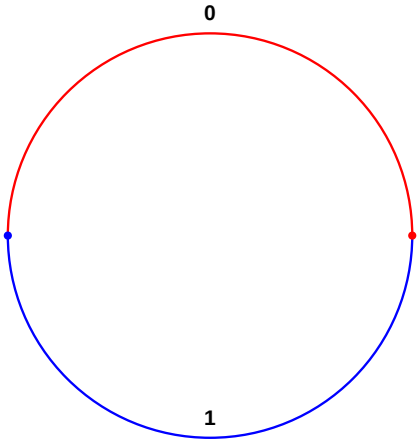

Figure 1.9. For symbolic dynamics, we label a space like the circle once and follow the coding of a single point under iterations of a map.

(we assign it a (0)), or the bottom half (then we assign it a (1)). The symbolic string for 0 then starts with 001, since $\theta_0, R(\theta_0) \in [0, 1/2)$, but $R^2(\theta_0) = \sqrt{2}/2 \in [1/2, 1)$.

This is illustrated in Figure 1.10. On the left we show how $\theta_0 = 0$ proceeds around the circle under the map $S(\theta) = \theta + \frac{1}{20} \pmod 1$, so the orbit of 0 proceeds through the rainbow monotonically in k, and using the exact same coloring, we show how the orbit of $\{R^k(\theta)\}_{k=0,\ldots,19}$ changes rapidly. In particular the coding for the first 20 iterates of θ_0 using R, starting at $k = 0$ is 00100100101101101001. This coding can be roughly seen by following the color progression in Figure 1.10, or by working it out on a computer. We can compare that with the coding for the first 20 iterates of θ_0 using S instead, which is 00000000001111111111.

In this way we map the entire orbit $\{R^k(\theta)\}_{k\in\mathbb{N}_0}$ to obtain an infinite sequence of 0s and 1s. Since the map R is invertible, with $R^{-1}(\theta) = \theta - \beta \pmod 1$, we can find a bi-infinite sequence of 0s and 1s associated to each $\theta \in [0, 1)$. Under certain circumstances, knowing the infinite sequence of 0s and 1s determines θ uniquely in much the same way that knowing all the decimal digits of a real number between 0 and 1 determines its precise value.

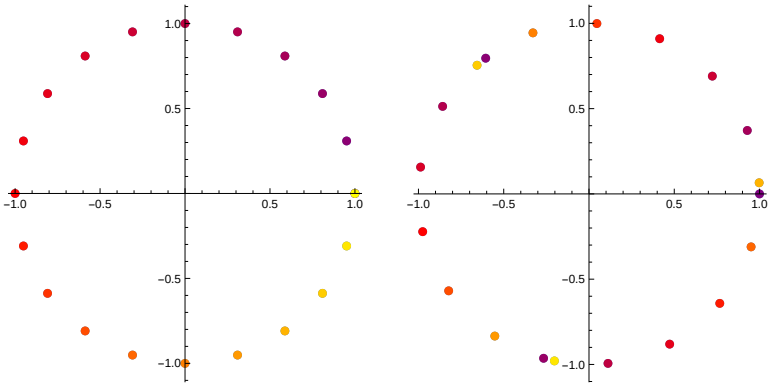

Figure 1.10. We follow the orbit $\{S^j(0)\}$, $j = 0, \ldots, 19$ in each case. In the circle on the left the orbit moves counterclockwise using the map $S(\theta) = \theta + 1/20$, with the colors decreasing in darkness with time (j). On the right, we use the identical algorithm for coloring the orbit but using the map $R(\theta) = \theta + \sqrt{2}/4 \pmod 1$, as in Example 1.6.

1.1.2. The shift map. The shift map σ plays a central role in the field of symbolic dynamics; it often represents the passage of time. We begin with a finite *alphabet* $\mathcal{A} = \{0, 1, \ldots, m-1\}$, which represents a finite list of possible states or *letters*, often coded as integers 0 through $m-1$ for the m different states. We have already seen that $\mathcal{A}$ may be a list of m different colors or letters (H and T) for example. Then the space Σ_m is the space of bi-infinite or two-sided sequences of elements from $\mathcal{A}$. A point in Σ_m is often referred to as a *configuration* since it is a string showing what state each coordinate is in, and collectively the coordinates of x give a configuration of the space Σ_m. If $x \in \Sigma_m$, we write it as

$$x = \ldots x_{-j} \ldots x_{-2}\, x_{-1} \bullet x_0\, x_1\, x_2 \ldots x_k \ldots \tag{1.3}$$

with each coordinate $x_i \in \mathcal{A}$. The dot to the left of x_0 marks an *origin* (0th coordinate) for the point. The (left) shift map is denoted σ and is defined by

$$\begin{aligned} \sigma(x) &= \sigma(\ldots x_{-j} \ldots x_{-1} \bullet x_0\, x_1 \ldots) \\ &= \ldots x_{-1}\, x_0 \bullet x_1\, x_2 \ldots . \end{aligned}$$

The coordinate x_0 often represents the present state, so x_0 is the state of the experiment or process now, while the negative coordinates reveal the past states and the positive coordinates are the future states.

Therefore the map σ is also described coordinate-wise as

$$\sigma(x)_i = x_{i+1}$$

for each $i \in \mathbb{Z}$. The shift can be interpreted as saying that if

$$\sigma^k(x) = y,\ y_i = x_{i+k},$$

then after the passage of k units of time, the coordinate y_0 (the present state of $y = \sigma^k(x)$) is x_k, a state k steps into the future of the point x.

Sometimes we use only one-sided sequences for configurations of the space. In this case we write the space as Σ_m^+, and we write points as follows:

$$\Sigma_m^+ = \{x = {\bullet}x_0\, x_1\, \ldots\} \tag{1.4}$$

with each coordinate an element of the alphabet $\mathcal{A}$. In this setting, it is frequently the case that a point whose present and future are shown by the coordinates of x can have infinitely many different pasts (see Exercise (9)).

A word about the notation. The convention is to denote points in Σ_m and Σ_m^+ as in (1.3) and (1.4), respectively. While the notation is reminiscent of a decimal (or base m) expansion of a real number, and there are occasional important connections to real numbers; the dot appears only to indicate where the 0th cooordinate of a sequence is located.

To extract a finite list of coordinates, known as a finite block, string or *word*, in a point x, we use the following notation: if $k < \ell$, then

$$x_{\{k,\ell\}} = x_k\, x_{k+1} \ldots x_\ell. \tag{1.5}$$

We call a finite block of states,

$$w = w_1\, w_2 \ldots w_n = (w_1, \ldots, w_n), \quad w_j \in \mathcal{A}, \tag{1.6}$$

a *word of length n* or an *n-block*. If w is a word of length n, then

$$C_k^w = \{x\ :\ x_{\{k,k+n\}} = w\} \tag{1.7}$$

denotes the set of configurations with the word w appearing starting at the kth coordinate. Since in the English language we write a word as w = cat rather than w = (c, a, t), we typically choose the left-hand notation in (1.6) for a symbolic word, unless confusion arises.

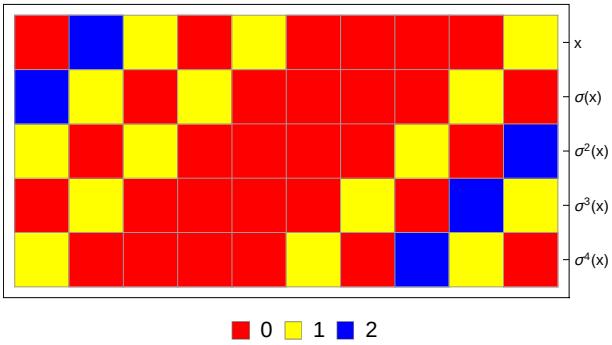

Figure 1.11. The orbit of a single point under iterations of the shift map σ.

The shift map σ keeps a word w intact, just moving it to the left as we iterate the shift map, as shown in Figure 1.11. We describe more precisely what is shown there. The top line shows the following point in the space $\Sigma_3^+ = \{0,1,2\}^{\mathbb{N}}$:

$$x = .0210100001\ldots.$$

The digits are shown in colors as indicated. Beneath x we have four rows showing $y_1 = \sigma(x)$, $y_2 = \sigma(y_1) = \sigma^2(x)$, $y_3 = \sigma(y_2)$, $y_4 = \sigma(y_3) = \sigma^4(x)$. We see that the word $w = 0000$ is shifted to the left but does not get broken apart under the shift map σ. We use the convention that if w is a word, then $x = \overline{w}$ means $x = www\ldots$, infinitely often. Only part of the point x appears in Figure 1.11 and in any picture we draw by hand or on the computer; for this example, we assume the segment shown repeats over and over. In other words, we assume $x = \overline{.0210100001}$. We note that later in the book we will see there are various ways to extend the point x beyond its finite window; for now, we use the convention that $x = .\overline{w}$ means we repeat the word w (given here by 0210100001) infinitely often.

We study the space Σ_m and the map σ in much more detail in the chapters that follow.

1.2. Cellular automata

Cellular automata were introduced by Ulam and von Neumann [**73**, **74**] in order to construct mathematical models of some physical processes. A simple nonmathematical description is that at each time step, a CA

updates each coordinate of a configuration of the space simultaneously, using the same set of local rules. In Figure 1.12 we use the same method of presenting our map as in Figure 1.11 to show the output of a cellular automaton. We postpone the formal mathematical definition until the next chapter, giving instead a description that is equivalent to the definition. First, because a point is a string of symbols from an alphabet $\mathcal{A}$ that can be displayed in a one-dimensional array, we refer to these CA as being one dimensional. In the next chapter we discuss the notion of dimension in more detail.

A *one-sided* point is written as $x = \centerdot x_0, x_1, x_2, \ldots$ and is described in (1.4), while a *two-sided* or *bi-infinite sequence* is a sequence indexed by both positive and negative integers. It is denoted by $x = \ldots x_{-2}x_{-1} \centerdot x_0x_1\, x_2, \ldots$ as in (1.3). Our choice of space for a CA depends on both the physical and mathematical context.

A one-dimensional cellular automaton is a map F from Σ_m or Σ_m^+ defined in (1.3) and (1.4) to itself, $m \geq 2$, for which each coordinate of $y = F(x)$ has the property that y_n only depends on finitely many coordinates of x, independently of $n \in \mathbb{Z}$. We write it as

$$y_n = f(x_{n-r}, x_{n-r+1}, \ldots, x_n, x_{n+1}, \ldots, x_{n+r})$$

for some $r \in \mathbb{N}_0$. In Figure 1.12, we note that the point x, forming the top line in each of the figures, is the same point appearing in Figure 1.11. We define $F(x)$ in this example by specifying each coordinate as follows:

$$[F(x)]_j = x_j + x_{j+1} \pmod 3, \tag{1.8}$$

where on a grid of size N we use $[F(x)]_N = x_N + x_0 \pmod 3$. For the definition of $y \pmod 3$ we refer the reader to (1.2) or (2.1). In this way we compute that if $x = .0210100001$ is what is shown, we implement the CA in (1.8) by using $x = .\overline{0210100001}$, so that we can fill in all the coordinate boxes in Figure 1.12.

In Figure 1.13 we start with the point $x = .\overline{2110000002}$, and we view the output from the two symbolic maps, σ and F, side by side. We see that even though both rules are very simple, the output of the point under four iterations appears to be more scrambled by F than by the shift map. This is not always the case, and these differences reveal the complexity of cellular automata.

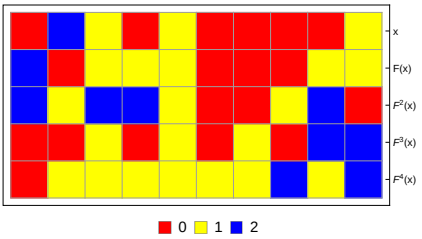

Figure 1.12. The orbit of a single point under iterations of the CA map F.

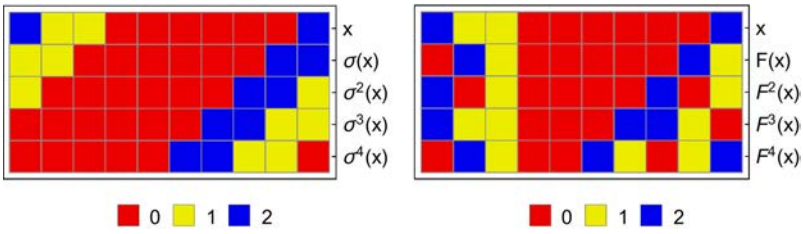

Figure 1.13. The same initial point x has very different outcomes under σ and F.

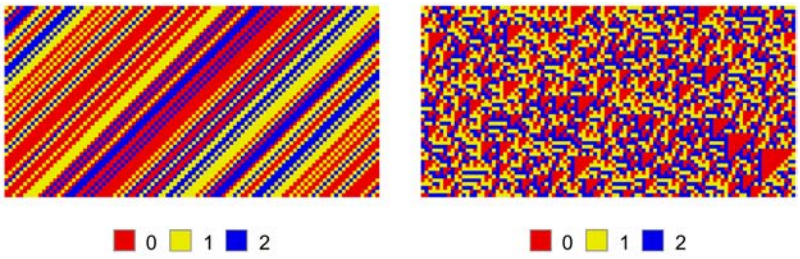

Figure 1.14. Using the same randomly chosen initial point x on the right, left, and in Figure 1.15, and showing the first 100 coordinates and 40 iterations of the same maps σ and F, the differences are more pronounced.

We repeat this using a randomly generated point x_0 and showing 100 coordinates of x_0 and 40 iterates, but still showing the orbit under σ on the left and the orbit of x_0 under F on the right, in Figure 1.14.

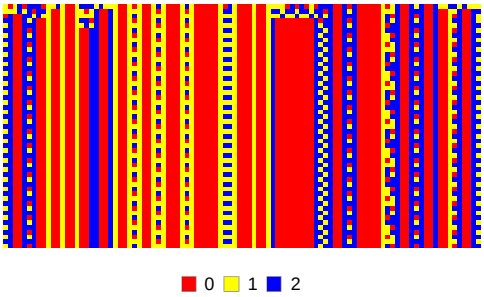

Figure 1.15. The same initial point x used in Figure 1.14 has very different values under a third CA.

1.2.1. An application of a 1D CA. There are many ways to describe a cellular automaton using mathematical, computer science, and physical descriptions. This book uses mathematical language for the most part. However, if the CA is being used to model a physical system, then often instead of a math formula, a physical description is used to define the CA. Here we give a description of a 1D CA that is considered by many to provide an accurate model for traffic flow. Versions of this CA, including in higher dimensions, provide informative models of swarming. In this and other settings a coordinate of a configuration x is often called a *cell*.

The Taxi CA example on Σ_2. We consider traffic flowing in a line along a road in one lane, from left to right across the page, some cars following others closely while other drivers leave more space between their vehicle and the car immediately ahead of them. At each coordinate i, if a cell has value 1 (yellow), it is occupied by a vehicle, and 0 or black represents an unoccupied cell. If a cell with value 1 has a cell with value 0 immediately to its right, this means there is space for the car to move forward, so the 1 moves rightwards leaving a 0 behind. A car cannot move forward to the right if a car is blocking its way, so a 1 with another 1 to its right remains a 1 (the car stays in place as it cannot move forward). Similarly, a 0 (unoccupied site on the road) that does not have a 1 to its left stays a 0 because there is no car to come up and fill that space. One can observe that traffic that is bunched up around minor road construction, for example, is unevenly spaced out. However it seems to be evenly spaced out after the passage of time and this is shown by the CA below. Figure 1.16 shows a line of taxis unevenly spaced at time 0, and one time unit later. We note that a specific taxi moves forward only when it can,

Figure 1.16. The top row represents a line of taxis (yellow coordinates) driving down a road (black coordinates) to the right (→), and the second row represents the taxis one time unit later.

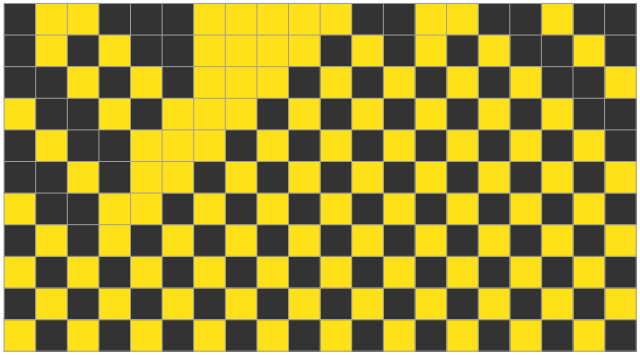

Figure 1.17. The flow of traffic under this CA evens out eventually with most initial points. Again the top row shows where they start and each subsequent row shows their location one time unit later. The driving direction is to the right (→).

and for this figure we use periodic boundary conditions, meaning the taxis are essentially driving on a loop. Figure 1.17 shows that after only a small number of time steps the taxis are as evenly spaced as possible, using only these simple rules given above.

Since each coordinate x_i is updated by looking at the value of x_i, x_{i+1}, and x_{i-1} (in front of the car and behind the car), and independently of the value of i (the specific location on the road), we can recast this model as a 1D CA. We formalize this later, but the complete mathematical description of the CA appears in Table 1.1. We see later that this CA has a unique name of F_{184}. Mathematically, we typically assume that the road is infinitely long, and the initial point has a random distribution of 0s and 1s whose proportions are largely determined by whether or not it is rush hour.

Table 1.1. A math presentation for the Taxi CA, showing how it acts on triples of states.

$x_{i-1}x_ix_{i+1}$	$[F(x)]_i$
111	1
110	0
101	1
100	1
011	1
010	0
001	0
000	0

1.2.2. Two-dimensional CAs. While we leave the details of examples such as this next one to Chapter 6, we briefly mention a two-dimensional CA. It is a single map acting on a point consisting of an infinite array of 0s (light yellow) and 1s (dark blue) lying in a planar (two-dimensional) grid. At each time step or application of the CA, the grid is updated. To view it in color in a finite array, we assume that the parts not shown are all 0s or that the extension to an infinite grid is obvious from the context. Lab 2 in Section 1.4 gives some hands-on lessons on how to update a 2D CA.

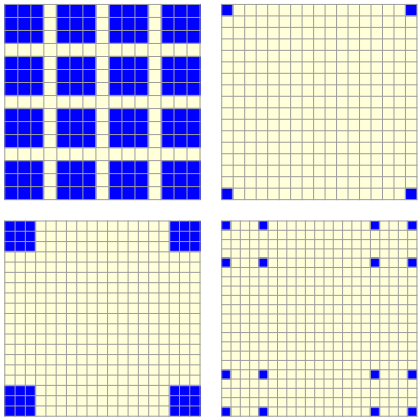

Figure 1.18. A two-dimensional CA is a configuration of states in a planar grid. This shows four iterations of a point, moving from left to right in each row to iterate the point in the upper left picture.

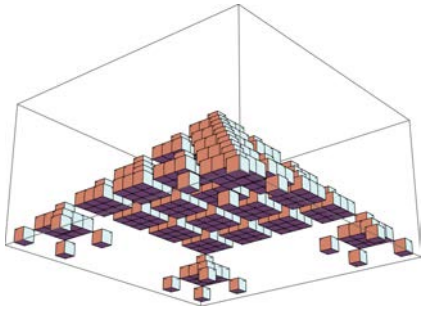

Figure 1.19. One way to show the development of a 2D CA is to show it in three dimensions. The bottom four layers correspond to the four points shown in Figure 1.18, with time moving downwards. The bottom is the last iterate shown in Figure 1.18.

1.3. Exercises

(1) Consider the following two real-valued functions of $(x, y) \in \mathbb{R}^2$, given by

$$f_1(x, y) = x + y \qquad \text{and}$$
$$f_2(x, y)) = x^2 + y.$$

For each f_j, $j = 1, 2$ draw a plane and color it with a coding consisting of three colors:

$$C_1 = \{(x, y) \ : \ f_j(x, y) > 1)\},$$
$$C_2 = \{(x, y) \ : \ -1 < f_j(x, y) < 1\},$$
$$C_3 = \{(x, y) \ : \ f_j(x, y) < -1\}.$$

A computer is not needed to determine the coding since the level curves for the boundary values of each colored region are easy to draw.

(2) Recall that a complex number is written as $z = x + iy$, with $x, y \in R$, and $i^2 = -1$. We write $\mathrm{Re}(z) = x$ for the real part of z and $\mathrm{Im}(z) = y$ for the imaginary part of z.

(a) For the function $f(z) = 2z$ give a contour plot coding, exactly as in (1), of the functions $f_1(x, y) = \mathrm{Im}\, f(z)$ and $f_2(x, y) = \mathrm{Re}\, f(z)$, using $z = x + iy$ and letting x and y range from -2 to 2. (Refer to Figure 1.5 for assistance.)

(b) Do the same for $f(z) = z^2$.

(3) (a) Draw by hand the graphs of the functions: $f_b(x) = bx \pmod 1$ on the domain $I = [0,1)$, for $b = 2, 3$, and $3/2$, referring to (1.2) if needed.
(b) Partition the interval I into two disjoint sets,

$$A_0 = [0, 1/2) \quad \text{and} \quad A_1 = [1/2, 1), \tag{1.9}$$

and color, label, or describe the points whose coding under each f_b starts with 00, 01, 10, and 11, using that $x \in [0,1)$ has coding ij if $x \in A_i$ and $f(x) \in A_j$.
(c) For the function f_b given in (a) and $b = 3$ give the subinterval(s) that are coded by 111 under iterations of f_b. (See (b) for the first iteration of f_b.)

(4) Using the maps and their graphs from Exercise (3) above, draw the unit interval $I = [0,1)$ and label the partition from (1.9).
(a) For $x_1 = 1/3$, write x_1 as an infinite one-sided binary sequence according to its binary (base 2) expansion.
Hint: Show $\sum_{k=1}^{\infty} 1/4^k = 1/3$.
(b) For $x_2 = 6/7$, write x_2 as a one-sided binary sequence according to its binary (base 2) expansion.
(c) Write down a coding, according to whether the point is in A_0 (code it as a 0) or lands in A_1 (code it as a 1) for the following sequences, using the map $f(x) = 2x \pmod 1$:

$$\{f^n(x_1)\}_{n\geq 1} \quad \text{and} \quad \{f^n(x_2)\}_{n\geq 1}.$$

(d) Compare the sequences with applying the shift map σ to the points x_0 and x_1 written in their binary expansion.

(5) Repeat Exercise (4)(c) using the map $f_{3/2}(x) = 3/2x \pmod 1$, using the points x_1 and x_2 given in problem 4(a) and 4(b).

(6) Give the next two lines that would appear at the bottom of both sides of Figure 1.13. In other words, give $F^5(x)$ and $F^6(x)$ out as far as possible, using the 0th coordinate as the $n+1$st, 1st coordinate at $n+2$, etc. On the left use the shift map σ and on the right use F as given in equation (1.8).

(7) A useful test for the complexity of a CA is to iterate the CA starting with a bi-infinite initial point $x \in \Sigma_2$ that consists of all zeros except for one 1 at the 0th coordinate of x. This is sometimes referred to as the *single site seed* for the CA. It is

particularly useful for getting a fast impression of the complexity of the CA, in terms of how many nonzero states there are, and how complicated they become under iteration. Make a few copies of the grid on the following page, and work out the first 5 iterates of the following CAs on Σ_2. A computer is not necessary for this problem but can be used.

(a) $[F(x)]_i = x_{i-1} + x_i + x_{i+1}$
(b) $[F(x)]_i = x_i + x_{i+1} + x_{i+2}$
(c) $[F(x)]_i = 1 - x_i$

(8) Order the CAs from Exercise (7) in terms of the level of complexity from the small amount of information available now, from simplest to most complex. For now, provide your own definition of "simple" and "complex" cellular automata.

(9) Show that a one-sided shift has the following property: for every $m \geq 2$, for every $x \in \Sigma_m^+$,

$$\sigma^{-1}(x) = \{y \ : \ \sigma(x) = y\}$$

has more than one pre-image. If we define the *infinite past* of x to be

$$\bigcup_{n \geq 1} \sigma^{-n}(x) = \{y \ : \ \sigma^n(x) = y \text{ for some } n \in \mathbb{N}\}$$

then show that the infinite past of each x is infinite. If $|A|$ denotes the cardinality of a set A, then show that for each fixed $n \in \mathbb{N}$, $|\sigma^{-n}(x)| < \infty$ but

$$\lim_{n \to \infty} |\sigma^{-n}(x)| = \infty.$$

(10) Using the Taxi CA from Section 1.2.1 and the initial point

$$x = 000011111.111110011,$$

with ten cars bunched up together where a little space in front of another vehicle is present, work out on a computer or on the grid in Figure 1.20 by hand, how long it takes to evenly space the cars, and if it is possible. You will need to use periodic boundary conditions, so assume the cars are driving on a loop.

(11) We describe a one-dimensional CA that represents the first pass at modeling virus dynamics in this book. The space is Σ_2, bi-infinite sequences of 0s and 1s, where 0 represents a healthy cell and 1 is an infected cell. The initial infection is represented

by a thin sprinkling of 1 in a point that is otherwise 0 everywhere; in particular we sprinkle the 1s in an area concentrated around x_0 to accentuate the initial infection (so we assume that initial points have only finitely many 1s). The local rule to define the CA updates a coordinate as follows: if $x_j = 0$ (healthy cell at that site), then $y_j = F(x)_j = 1$ if and only if x_j is contiguous to an infected cell. This means if and only if $x_{j-1} = 1$ or $x_{j+1} = 1$ (or both). We assume that the immune response is working, so if $x_j = 1$, it always updates to a healthy cell.

(a) Draw an initial point as described, and give a few updates below it.

(b) Describe some qualitative dynamical properties: Does the level of initial infection affect the outcome? Does the organ modeled, in this case, a one-dimensional capillary, ever (or often) completely clear the virus?

(c) With no conditions imposed on the initial value of x, discuss if there are any points $x \in \Sigma_2$ such that $F(x) = x$. Are there points such that $F \circ F(x) = x$ but $F(x) \neq x$?

1.4. Labs

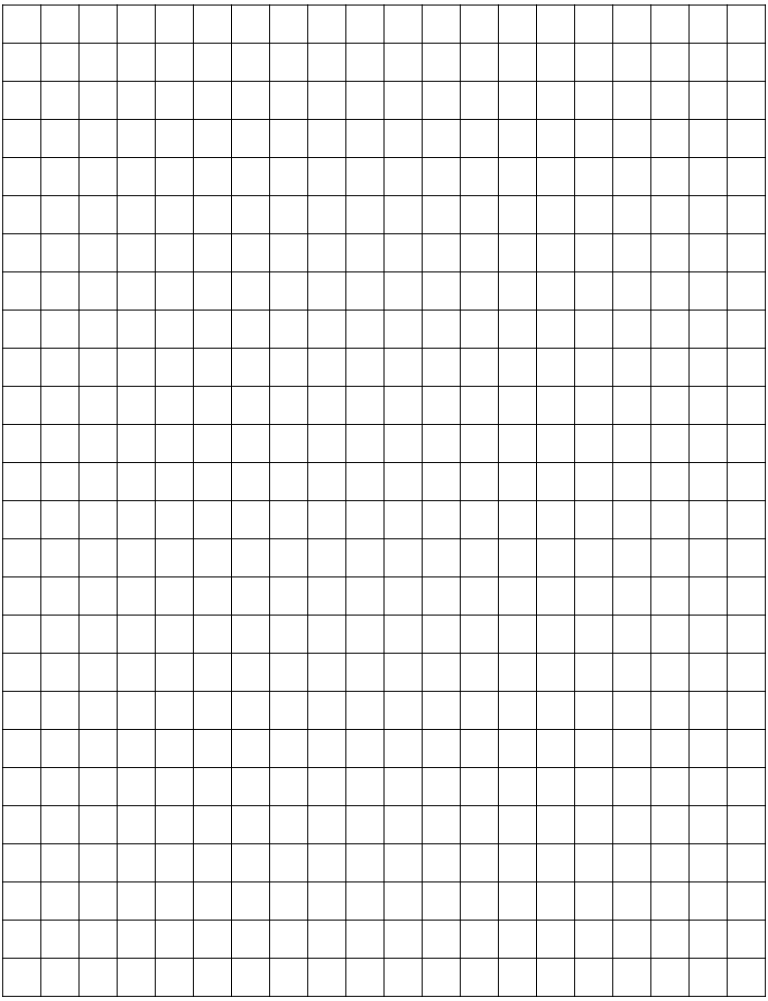

Figure 1.20. Blank grid for 1D CAs. This is to be copied and used to run CA examples by hand.

LAB #1: COLOR CODING FOR REAL-VALUED FUNCTIONS (🖳)

Consider the function of two real variables,

$$G(x,y) = (\sin x, \cos y) = (g_1(x,y), g_2(x,y)).$$

Let $|(u,v)| = \sqrt{u^2+v^2}$ denote the Euclidean length of a vector in the plane. Use the following two-color code for the real-valued function $A(x,y) = |G(x,y)|$ deciding on a color to label B and one to label R:

$$\text{Color B} = \{(x,y) \ : \ A(x,y) \in (0,1)\},$$
$$\text{Color R} = \{(x,y) \ : \ A(x,y) \geq 1\}.$$

(1) Draw by hand the color-coded plane for $A(x,y)$

(2) This is similar to (1), but for g_j, $j = 1,2$ (replacing A with g_j, $j = 1,2$) and using the following coding rules. Choose colors for B and R, and using $g_1(x,y) = \sin(x)$, draw the color coded plane according to the coding

$$\text{Color B} = \{(x,y) \ : \ g_1(x,y) \in (0,1]\},$$
$$\text{Color R} = \{(x,y) \ : \ g_1(x,y) \in -[1,0]\}.$$

Repeat this for $g_2(x,y) = \cos(y)$.

(3) For $n=2,3$ draw the color coded plane for $A_n(x,y)=|G^n(x,y)|$, where $G^n(x,y) = G \circ G \circ \cdots \circ G$, n times. This may require a computer.

(4) We now work out the limiting coding. Namely,

$$\text{Color L}_1 = \{(x,y) \ : \ \lim_{n\to\infty} A_n(x,y) \in (0,1/2)\},$$
$$\text{Color L}_2 = \{(x,y) \ : \ \lim_{n\to\infty} A_n(x,y) \geq 1/2\}.$$

Hint: The limit is actually a constant, so the entire plane is either coded with L_1 or L_2. Which is it, and why?

(5) Repeat Exercise (1) for the map,

$$H(x,y) = (h(x), h(y)) = (3x \,(\text{mod } 1), 3y \,(\text{mod } 1))$$

again using the function $A(x,y) = |H(x,y)|$. Use three contours and colors for this function. For a more challenging function, let $J(x,y) = (3x \,(\text{mod } 1), 2y \,(\text{mod } 1))$ and encode the contours of the absolute value function for this using some natural (and small) set of contours.

LAB #2: SOME SIMPLE 2D CAS AND HOW TO UPDATE THEM BY HAND (🖳)

This lab shows how to work with a simple two-dimensional cellular automaton by hand, as a computer is optional here.

(1) In Figure 1.21 we show a small piece of a doubly bi-infinite array of values of 0 = white and 1 = red states. We label a coordinate by $x_{i,j}$, and we show a point consisting of all 0s except for the coordinate in the center, so $x_{0,0} = 1$.

(2) The updating rule F for this CA is that $F(x)_{i,j} = 1$ if any coordinate above or below, to the right or to the left is a 1, and 0 otherwise (see the first two boxes in Figure 1.21). Therefore $[F(x)]_{i,j} = 1$ if and only if

$$x_{i,j+1} = 1, x_{i,j-1} = 1, x_{i+1,j} = 1, \text{ or } x_{i-1,j} = 1.$$

(3) The configuration for $F(x)$ is shown: Fill in the rest of the grids for $F^2(x) - F^5(x)$, assuming the squares outside the marked space are 0s to begin with, according to the rule described, in Figure 1.21. You will can extend the grids using another sheet of paper, or assume that every state outside the grid is a 0 (unless it turns to a 1) and only show what happens inside the region shown.

(4) Repeat lab steps (1)–(3), using Figure 1.22 and the CA defined as follows: $[G(x)]_{i,j} = 1$ if and only if

$$x_{i,j} + x_{i,j-1} + x_{i+1,j} + x_{i-1,j} = 1 \pmod 2.$$

Refer to (1.2) or (2.1) for the notation. Use an initial configuration with two 1s placed centrally, and assume everything outside the viewing region is 0. Add more boxes to the grids as needed to expand the nonzero coordinates.

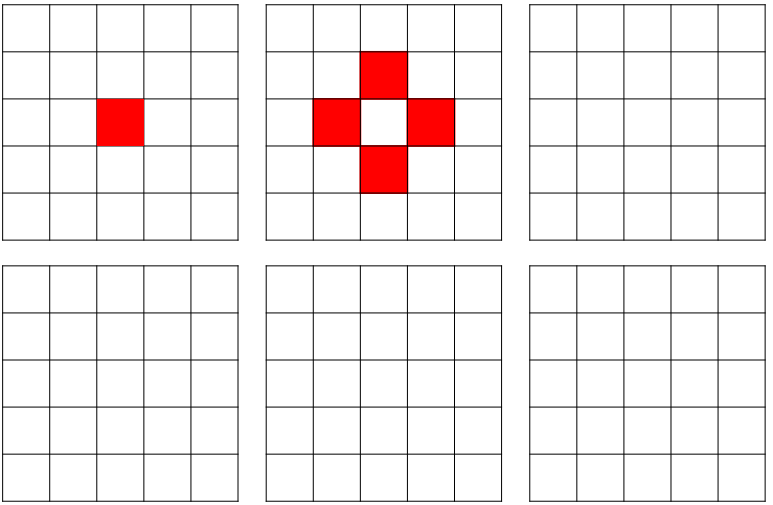

Figure 1.21. In this 2D CA, the initial point is a field of 0s (white) with one 1 (red) at the coordinate $(0, 0)$. We update the configuration by assigning the coordinate 1 if and only if there is a 1 directly above or below or immediately to the right or left.

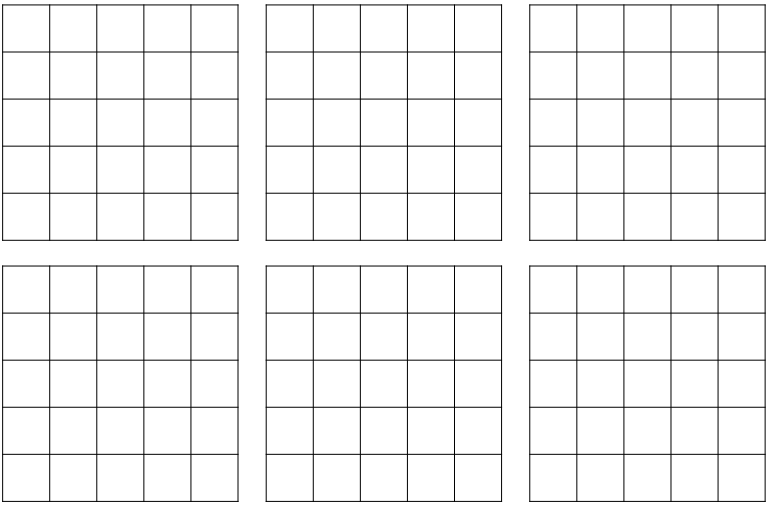

Figure 1.22. This is a blank grid for the project outlined in Lab 2.