

AN INTRODUCTION TO CELLULAR AUTOMATA AND THEIR APPLICATIONS

DIMACS Reconnect 2000

Group #2 – Sam Northshield (contact), Mike Daven, Alex Kheyfits, Melkamu Zeleke

June 12, 2001

1 Introduction – for the student

A *cellular automaton* or *CA* is a mathematical “machine” or “organism” that lends itself to some very remarkable and beautiful ideas. One of the most exciting aspects of this area of mathematics is that cellular automata arise from very basic mathematical principles. Though they are remarkably simple at the start, CAs have a variety of applications and can show up in places you would never expect! From simple games of chance to computer simulations of biology to art, we will see that these simple ideas are evidence that mathematics is very real and is all around us.

In this module, we will show you how cellular automata are “built” with basic mathematical building blocks. Where necessary, definitions you may know will be reviewed and new concepts may be introduced. After we have seen a few examples of cellular automata, we take a look at the most famous example, the Game of Life. We will then examine CAs from a different perspective, applying these ideas to several problems of probability. We will use a different sort of “machine” to solve several problems that involve games of chance. The final section will equip you with the material necessary to pursue this topic in an advanced class or a semester of independent study.

2 Introduction – for the teacher

This module was designed to be part of a class for first- or second-year liberal arts students. Our intention is to introduce the theory of finite-state automata in an interesting way to those whose studies are in non-technical fields.

In Section 3 we begin with Pascal’s triangle, a concept with which students may already be familiar. In Sections 4 and 5 we use modular arithmetic to create Pascal’s triangle module 2, which we soon see can be viewed as a one-dimensional cellular automata. In Section 7 we take a brief look at the most famous example in cellular automata, the Game of Life. In Section 8 we give a few interesting applications of simple finite-state “machines”, also introducing some basic probability and graph theory. Finally in Section 9 we formalize the material presented in the previous sections in such a way that students will be prepared to study cellular automata at a more advanced level.

This module is self-contained, so there are no prerequisites. We expect that this material could be completed in (2,3 weeks), with time for review of the exercises contained herein.

This module was produced as part of the Reconnect Program 2000, organized by the DIMACS Institute at Rutgers University. Our thanks to Lawrence Gray of the University of Minnesota for introducing us to this topic.

3 Pascal's Triangle

Pascal's triangle is an array of numbers which has found uses in many areas of mathematics.

$$\begin{array}{cccccc} & & & & & 1 \\ & & & & & 1 & 1 \\ & & & & 1 & 2 & 1 \\ & & 1 & 3 & 3 & 1 \\ 1 & 4 & 6 & ? & 1 \end{array}$$

Do you see the pattern?

Exercise 3.1

Can you find the missing number?

Exercise 3.2

Can you find the next row?

Answer: The missing number is 4 and is found by adding the two numbers above the empty space.

Answer: The next row is formed by the addition rule for finding missing spaces: 1, 5, 10, 10, 5, 1.

The 1's on either end also follow this rule if there are implicit zeros. So the triangle really looks like

$$\begin{array}{cccccccc} & & & & \dots & 0 & 1 & 0 & \dots \\ & & & & \dots & 0 & 1 & 1 & 0 & \dots \\ & & & & \dots & 0 & 1 & 2 & 1 & 0 & \dots \\ & & & & \dots & 0 & 1 & 3 & 3 & 1 & 0 & \dots \\ & & & & \dots & 0 & 1 & 4 & 6 & 4 & 1 & 0 & \dots \end{array}$$

One of the most common uses of numbers from Pascal's triangle is in the binomial formulas, such as:

$$\begin{aligned} (a + b)^2 &= a^2 + 2ab + b^2, \\ (a + b)^3 &= a^3 + 3a^2b + 3ab^2 + b^3, \\ \text{and } (a + b)^4 &= a^4 + 4a^3b + 6a^2b^2 + 4ab^3 + b^4. \end{aligned}$$

The coefficients of $(a + b)^n$ come from row n of Pascal's triangle.

Exercise 3.3

Use Pascal's triangle to find $(a + b)^5$ and $(a + b)^7$.

Perhaps a better picture is:

$$\begin{array}{cccccccccccc} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 2 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 3 & 0 & 3 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 4 & 0 & 6 & 0 & 4 & 0 & 1 & 0 & 0 \end{array}$$

One can think of each successive row as a new *state* resulting from application of a *transition rule* to the state before.

Exercise 3.4

(a) What are the states that we are talking about? Give 3 examples of new states.

(b) What is the transition rule?

(c) Apply the transition rule to the following state:

$$(\dots 2\ 4\ 2\ 4\ 1\ 2\ 1\ 2\ 0\ 1\ 1\ 0\dots)$$

(d) Compare the two states:

$$(\dots 0\ 0\ 0\ 1\ 2\ 3\ 0\ 0\ 0\ 0\ 0\dots)$$

$$(\dots 0\ 0\ 0\ 0\ 1\ 2\ 3\ 0\ 0\ 0\ 0\dots)$$

Are they different?

(e) Apply the transition rule to the two states in (d); are the results different?

It is perhaps unsatisfactory to define a state to be an infinite (in both directions!) row of numbers since two different such rows are indistinguishable unless placed next to each other (see Exercise 3.4(d) above). To get around this difficulty, we can define a state to be an assignment of numbers to each integer: for example, $A(2) = 1$, $A(3) = 2$, $A(4) = 3$, and $A(n) = 0$ for all n not equal to 2, 3, or 4.

Exercise 3.5

If A is the state corresponding to the first row of Exercise 3.4(d), what is the state corresponding to the second row?

The transition rule could then be described by the equation

$$B(n) = A(n - 1) + A(n + 1) .$$

4 Addition Modulo 2

We are all familiar with ordinary addition of whole numbers. Here we present addition which only works for a very small set of whole numbers, namely $\{0, 1\}$. The addition rule is

$$0 + 0 = 0 , 0 + 1 = 1 , 1 + 0 = 1 , \text{ and } 1 + 1 = 0 .$$

We write, for a, b equal to 0 or 1, $a + b \pmod 2$.

Essentially, we are throwing out all properties of numbers except their even-ness or odd-ness and adding those.

Exercise 4.1

Find $a + b + c \pmod 2$ for all eight possible configurations of a, b , and c .

Exercise 4.2

Explain why there are exactly eight configurations possible in the exercise above.

5 Pascal's Triangle Modulo 2

Suppose we now replace addition in the transition rule in Section 3 by addition modulo 2. That is, given a state A (being an assignment of 0 or 1 to each of the integers), the transition rule changes it to the state B defined by

$$B(n) = A(n-1) + A(n+1) \pmod{2}.$$

Exercise 5.1

Let A be defined by

$$A(n) = \begin{cases} 1 & \text{if } n \text{ is odd} \\ 0 & \text{if } n \text{ is even} \end{cases}$$

To what does the transition rule take A ? Apply the transition rule to the result. Repeat. Do you see a pattern?

Let us start with $A(n) = \{1 \text{ if } n = 0 ; 0 \text{ otherwise}\}$. This is our initial state. If we draw that state as a row with an asterisk for a one and a blank for a zero, we get

_ _ _ _ _ * _ _ _ _ _

By applying the transition rule repeatedly and drawing consecutive states vertically downward, we get

```

0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 1 0 1 0 0 0 0
0 0 0 1 0 0 0 1 0 0 0
0 0 1 0 1 0 1 0 1 0 0
0 1 0 0 0 0 0 0 0 1 0

```

Here are a few further diagrams:

This is where we had a few cutouts, including one like this ...

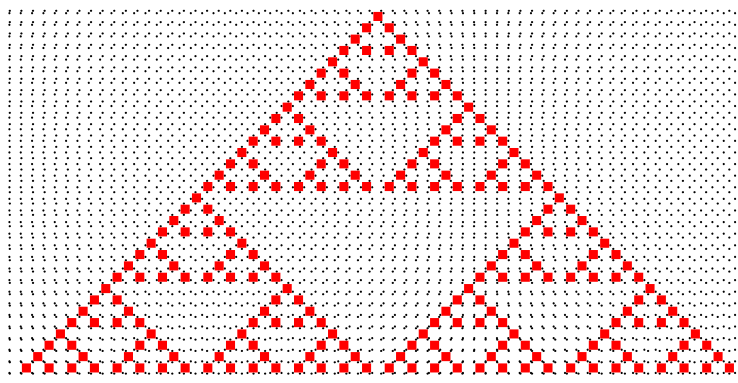


Figure 5.1: One of several ways to picture Pascal's triangle.

The figure above is sometimes called "Pascal's triangle mod 2" (for, I hope, obvious reasons).

Exercise 5.2

Interestingly, we may sometimes run a cellular automaton backwards. The question is, if B is the state following A by applying a transition rule, is it possible to find the reverse transition rule which, when applied to B , gives A ? Hint: The only way that one can reverse a transition rule is if the rule is “one-to-one” – that is, there are no two distinct states that can precede a given state. Is there a state other than $(\dots 0001000\dots)$ preceding $(\dots 000101000\dots)$?

A curious sequence is obtained by counting the number of 1’s in consecutive rows of Pascal’s triangle mod 2:

$$1, 2, 2, 4, 2, 4, 4, 8, 2, 4, 4, 8, 4, 8, 8, 16, 2, 4, 4, 8, \dots$$

Exercise 5.3

What are the next 50 terms of this sequence?

To do the above exercise, one must have noticed some patterns (e.g., every number in the sequence is a power of 2). Is there a way to predict the n^{th} term of this sequence without calculating all the previous terms? For example, how many 1’s are there in the 1000^{th} row of Pascal’s triangle mod 2?

Exercise 5.4

(A challenging exercise)

Show that the number of ones in the n^{th} row of Pascal’s triangle mod 2 is 2 to the power j , where j is the number of ones in the binary expansion of n . Recall that the binary expansions of $1, 2, 3, 4, 5, 6, 7, 8 \dots$ are $1, 10, 11, 100, 101, 110, 111, 1000, \dots$

Exercise 5.5

Use the result of Exercise 5.4 (even if you didn’t do it) to find the number of 1’s in the n^{th} row of Pascal’s triangle mod 2 if $n = 10, 25, 31, 66, 134$, and 123456 .

Exercise 5.6

(A challenging exercise based on Exercise 5.4)

Recall that $n!$ (“ n -factorial”) is the product of the first n natural numbers. It is even, for n greater than or equal to 2. Furthermore, the larger n is, the more 2 divides $n!$. Show that 2 to the n^{th} power is the largest power of 2 which divides $n!x$, where x is the number of 1’s in the n^{th} row of Pascal’s triangle mod 2.

6 Cellular Automata

We are interested in generalizing the previous example. There are two ways to do this: one is to change the initial state, the other to change the transition rule. Before we proceed, let’s formalize what we have so far.

We have a initial state (say A) and, implicitly, a set of possible states. In the case above, the set of possible states is simply the set of sequences of 0’s and 1’s – more explicitly, it is the set of all possible assignments of 0’s and 1’s to the integers. Furthermore, we have a transition rule which assigns to each possible state, another state.

The rest of this section is devoted to examples.

Exercise 6.1

What happens if we start with a different starting state? Trace out the first 5 states following each of the following initial states:

- (a) (... 000001001001000000 ...) (3 ones, 3 apart)
- (b) (... 000011001100000000 ...) (two pairs of ones)
- (c) (... 111000111000111000 ...) (alternating triples of ones and zeros)
- (d) (... 010001011010001101 ...) (random)

Another possible change is obtained by changing the transition rule. Such a rule can be thought of as a “machine” or computer which takes as input a string of 0’s and 1’s and gives a 0 or 1 as output.

For example, we may define $B(n) = A(n - 1) + A(n) + A(n + 1) \pmod{2}$. It is similar to the rule above in that it is “local”; the new value at n depends only on the old values at and/or near n . We shall call this rule “Rule 111” (the first rule, the one for Pascal’s triangle, is denoted “Rule 101”).

Exercise 6.2

Repeat Exercise 6.1 but with Rule 111 instead of Rule 101.

Exercise 6.3

How would you interpret Rule 10101? Repeat Exercise 6.1 with this rule.

It has been proposed that there are 4 types of transition rules for cellular automata resulting in different qualitative behavior. The possibilities are that the evolution of the automaton leads to a homogeneous state, leads to a set of separated simple stable or periodic structures, leads to a chaotic pattern, or leads to complex localized structures, sometimes long-lived. Here are some examples (these all follow binary rules of the form $B(n) = F(A(n - 2), A(n - 1), A(n), A(n + 1), A(n + 2))$, where binary means that cells are either 0 or 1.)

Insert figures here

Figure 6.1: The 4 transition rules for cellular automata.

Exercise 6.4

Use a computer implementation of cellular automata (see references) to draw pictures similar to those above. Part of this exercise is to figure out the transition rules.

7 The Game of Life

John H. Conway developed “the Game of Life” in the 1960’s. It is an example of a two-dimensional CA. It is called two-dimensional since a possible state looks like an infinite checkerboard with sites either occupied (with a checker, say) or not. For theoretical purposes, we usually assume that the checkerboard is infinite in all directions. For practical purposes, such as running the game on a computer, we must assume that the checkerboard is finite. First we define “neighbors”: two cells (i.e., squares) are neighbors if they differ by one space (left/right/up/down/diagonal) – that is, one chess king’s move apart. Thus, every cell has eight neighboring cells.

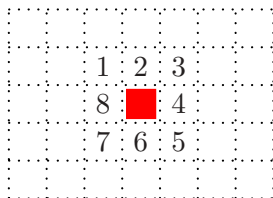


Figure 7.1: The neighbors of a cell are one chess king’s move away.

Here, of course, the infinite case is preferable; how does one define neighbors for cells on the edge of a finite board? There are two common ways to do this. In the first case, one simply counts neighbors regardless of whether the cell is on the edge or not. For example, a corner square of a checkerboard has only 3 neighbors. Often we wish to preserve the fact that each cell has eight neighbors in spite of being on the edge. The way to work this out is to “wrap” the board around so that the cell to the left of a cell on the left edge of row k is the rightmost cell of row k , and so on.

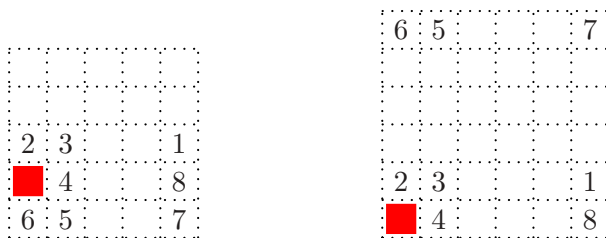


Figure 7.2: There are always eight neighbors on a torus.

Exercise 7.1

Explain why all cells in the arrangements above have exactly 8 neighbors. This setup is sometimes said to be “on a torus”. Why?

A “state” is the checkerboard plus an assignment, for each cell, whether that cell is occupied or not.

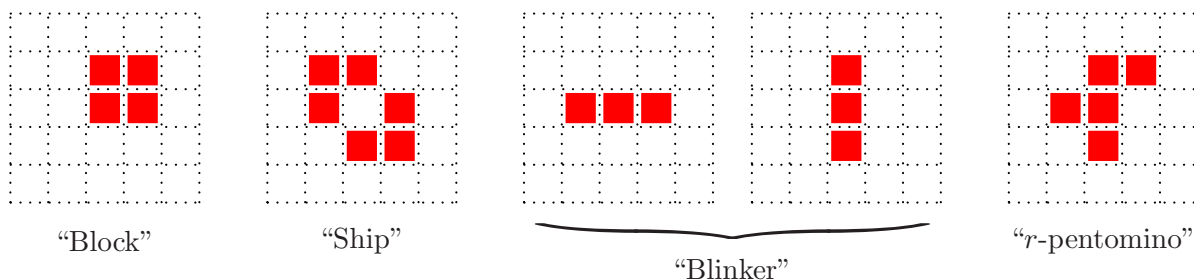
Exercise 7.2

What is the state with the fewest number of occupied sites? How many states are there with exactly one occupied site if the board is 3 by 3? 8 by 8? Infinite?

As above, the transition rule is local. Here it is: an occupant of a cell with one or fewer neighbors will, sadly, die of loneliness. On the other hand, with 4 or more neighbors, an occupant will die of over-excitement. With two neighbors, the occupant will continue into the next generation. The apparently tri-sexual occupants will reproduce occasionally: if a cell, empty or not, has three occupied neighbors, then the cell will be occupied in the next generation. These simple rules allow for virtually unlimited complexity.

Exercise 7.3

Apply the transition rule to each of the following configurations (assume that cells outside the visible region are all empty).



Doing Exercise 7.3, you saw that a couple of configurations were stable. In particular, the “block” and the “ship” remained the same after applying the transition rule. These terms were coined by Conway; see these and more in *Winning Ways for Your Mathematical Plays* [2] (volume 2) with Elwyn Berlekamp and Richard K. Guy.

We are interested in the evolution of this system. That is, given an initial state, what happens after several generations? Several thousand generations? For example, we saw above that the state consisting of one block or one ship will remain the same generation after generation. Thus the evolution of this system is completely known! Also from Exercise 7.3, you saw that a row of three becomes a column of three and vice versa. Thus, starting with row of three, we get back to the original state in two generations. This configuration is called a “blinker” (take a look at it on a computer).

Exercise 7.4

- Why do we use the word “cyclic” when some state occurs more than once?
- Give four examples of initial states that become cyclic.
- Give a convincing argument for why, in either of the two finite setups, the evolution always becomes cyclic.

We say a state is “periodic” if it occurs more than once following some starting state.

Exercise 7.5

- Show that a block and a ship are each periodic.
- Show that if a state is periodic, then the subsequent state is also periodic.
- Show that every state following a periodic state is periodic.

If we start with a periodic state, then that state reappears for the first time after a certain number of generations; the number of such generations is called the “period” of that state. For example, the block has period 1.

Exercise 7.6

- (a) Find the period of the “ship”.
- (b) Find the period of the “blinker”.
- (c) Begin with a row of 10 consecutive occupied cells; the state is eventually periodic. What is the period of the first periodic state? This exercise all but requires a computer.
- (d*) Find a periodic state with period 3.

Although every state is eventually periodic on a finite board (by an exercise above), this is not true on an infinite board. Consider the initial state in the figure below, called a “glider”:

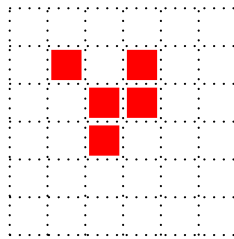


Figure 7.3: A glider.

Exercise 7.7

- (a) Iterate Life for 10 generations starting with a glider.
- (b) Can you explain why, on an infinite board, the glider is not eventually periodic?
- (c) What happens to the glider on the torus? That is, why does this not violate the conclusion of Exercise 7.7(b)?

An interesting phenomenon is that a finite state can result in unbounded growth. For example, consider the “*r*-pentomino”.

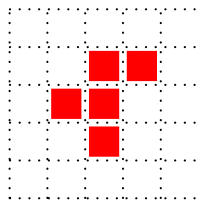


Figure 7.4: The *r*-pentomino.

Exercise 7.8

- (a) How many gliders are produced by an *r*-pentomino? (Use a computer!)

(b) What is the total biomass at the end? That is, how many occupied sites are there after things “settle down”?

If one iterates this (hopefully with the aid of a computer), one sees that the mess eventually produces a couple of gliders going off in distinct directions. Hence the diameter of the occupied sites grows without bound (on an infinite board).

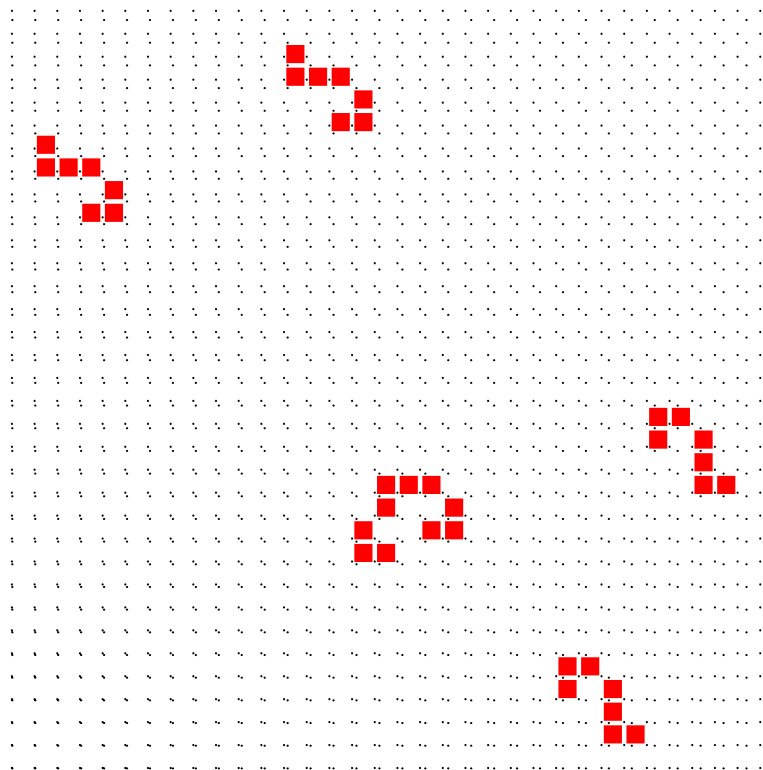


Figure 7.5: A glider gun.

The objects like the “blinker”, the “glider”, and the “block” are indeed objects; we can comprehend them as entities that persist through time (e.g., they blink, glide, or just sit). They are not states or even, in the case of the glider, parts of states.

Exercise 7.9

(A problem for philosophers):

If blinkers, gliders, and blocks are “objects”, what is an object? For the glider, what is it that “persists through time”?

Here are two more objects:

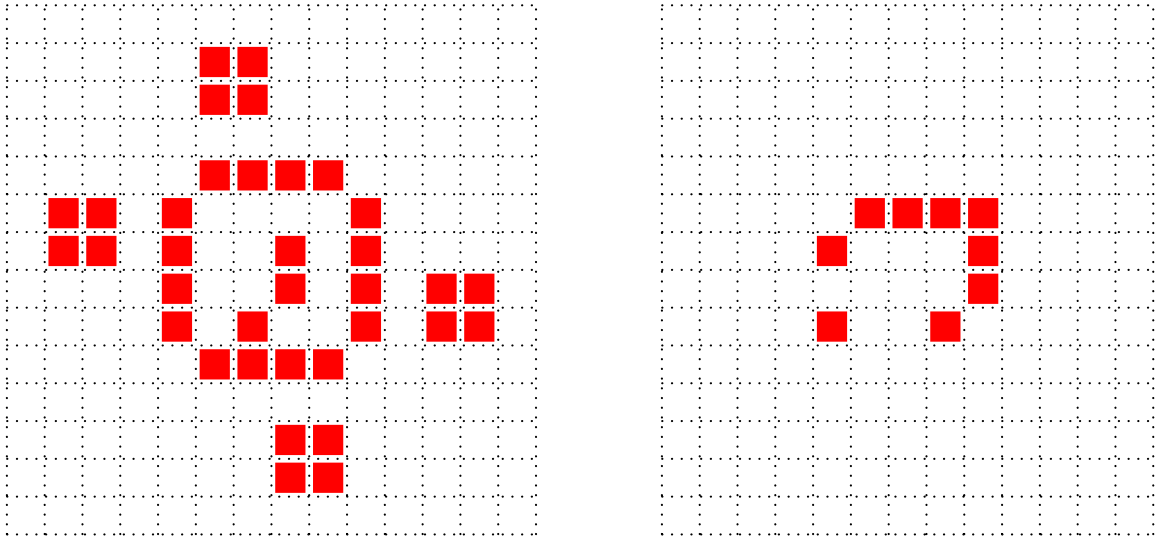


Figure 7.6: On the left, a clock; on the right, a horizontal glider.

Exercise 7.10

- What is the period of the “clock”?
- Verify that the “horizontal glider” earns its name.

One of the most interesting results due to J.H. Conway is that the Game of Life is capable of “universal computation” – that is, it can do anything that any computer can do. For example, there is a finite initial state such that any paragraph of English prose, when properly coded as a sequence of gliders, will result in a “spell-checked” paragraph of English prose (again coded as a sequence of gliders). More details on this and other topics on the Game of Life can be found in [2].

8 Applications of CAs and probability

Until now, we have considered a very particular sort of cellular automaton. It was important to know the transition rule from one state to another state. In this section, we will discuss elementary probability and a few applications of the ideas of CAs. The states and transition rules will appear different, as will the figures we draw, but we will still rely on the fundamental notions we have seen in the previous sections.

Exercise 8.1

Flip a coin 3 times, and draw a tree diagram that represents the possible outcomes.

What are the chances that you get 3 heads in a row? What are the chances that you get two heads and one tails? What are the chances that you get two heads and one tails, not necessarily in that order? These are simple questions, but the answers are not so simple. It will help if we clarify the terminology we will use.

Call the result of each coin toss an *outcome*. Here, an outcome can be either heads or tails. An *event* is a sequence of several outcomes, such as “three heads in a row” or “heads, tails, heads,

tails”. We call the collection of all possible events the *sample space*. The chances that a certain event will occur is called the *probability* of that event.

For each of the questions above, “What are the chances . . . ?”, the answer will be between 0 and 1. Whenever we ask a probability question, this will always be the case. We will usually assume that the coin we use is a fair coin.

Exercise 8.2

- (a) The chances of getting “heads” on a single toss of a fair coin is $1/2$. For the same coin, what are the chances of getting “three heads in a row” in 3 tosses? What are the chances of getting “heads, tails, heads, tails” in 4 tosses?
- (b) With a fair coin, what are the chances of getting “three heads in a row” in 4 tosses? In 5 tosses? In each case, consider what “three heads in a row” really means.
- (c) Suppose that we have a biased coin, where the chances of getting “heads” on a single toss is $1/3$. Repeat parts (a) and (b) for this “unfair” coin.

Example 8.3

A BOLD GAMBLE

You have \$1 and you need \$5. You can reach your goal by a fair gamble. You decide on *the bold strategy*: at each round you stake so much of your current fortune that you come as close to your goal as possible if you win. For example, if you have \$2, you bet it all; if you have \$4, you would only need to bet \$1. You are bold enough to risk whatever it takes to get \$5, but you can never bet more than you have at the moment. The bold gamble can be translated into Figure 8.1.

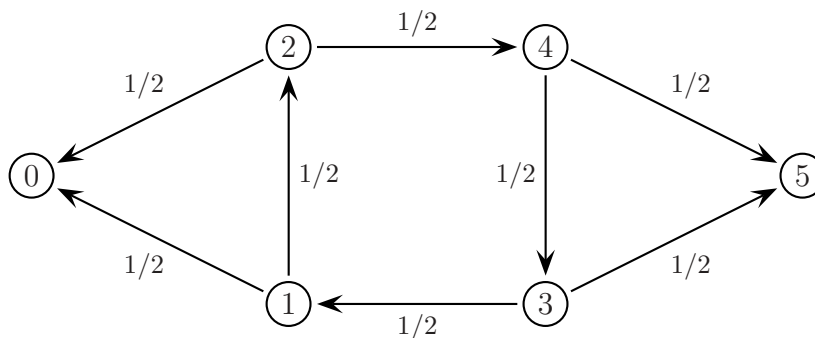


Figure 8.1: If you had \$1, how could you get \$5 by several bold gambles?

An event here is your current fortune, and each event is represented by a *vertex* or *state*. If the event *B* can be obtained from event *A* by a single coin-toss, we draw a directed line or *arc* from *A* to *B* and we call *B* a *neighbor* of *A*. We call the resulting diagram a *directed graph* or *digraph*. So we use a digraph to picture the transition from one state to another; this is a different way of looking at what we have seen in the previous sections.

There are two kinds of states we will want to identify. If a state has both “incoming” and “outgoing” arcs, we say this state is *internal*. If a state has only “incoming” arcs, we say this state is a *boundary* or *absorbing* state. From each internal vertex in our digraph there are two “outgoing” arcs, since we can either win or lose from this position. We have marked each arc with a “ $1/2$ ” to indicate that a win or loss is equally likely for each individual bet you make.

The direction of each arc is important. For instance, the vertex labelled “1” indicates the start of the game, where you hold \$1. At this point, you could bet just \$1 and either win (go to state “2”) or lose (go to state “0”). Suppose you win, and are now in state 2. From here, you will bet your entire \$2. If you win this next bet, you will go to state 4; if you lose, you will go to state 0. Take a moment to confirm that each of the directed edges corresponds to a bet won or lost.

With this digraph and some new terminology, we can return to our original puzzle. What are the chances of successfully winning the \$5 you need? That is, what are the chances of getting from \$1 to \$5?

describe “weighted average, duration, mean value”

To answer this question, we will need to know two important properties that are used in probability. First, the probability of going from an internal state to one of the boundary states is the weighted average of the probabilities of the neighboring states. Second, the expected duration going from an internal state to the one of the boundaries is the average of the duration of its neighbors, plus 1.

describe why it doesn't matter which boundary

redraw digraph in several steps, describe

□

Exercise 8.4

List the internal states and the boundary states in the digraph above.

Exercise 8.5

Why do the states “0” and “5” have no outgoing arcs? Why does each internal state have a single incoming arc?

Exercise 8.6

Suppose you start with \$1, but the object is to win \$6. Draw a digraph with 7 vertices and all of the appropriate edges. In this new digraph, which of the states are internal, and which are absorbing?

Exercise 8.7

For each of the following, give an example of a digraph:

- (a) a digraph with several boundary states
- (b) a digraph with at least one vertex which has only outgoing arcs
- (c) a digraph with no boundary
- (d) a digraph with no internal states

These digraphs don't necessarily have to come from “the bold strategy”.

Of course, as in Exercise 8.7, digraphs may represent a variety of scenarios. Here is another game, and we will again use digraphs to map out all the possibilities.

Example 8.8

A “SIMPLE” COIN GAME?

Aristophanes and Zeilberger have chosen the numbers 011 and 111, respectively. They play the

following game: toss a fair coin with faces 0 and 1 until one of their numbers has appeared. What are the chances that Zeilberger wins this game?

Let's draw the appropriate digraph:

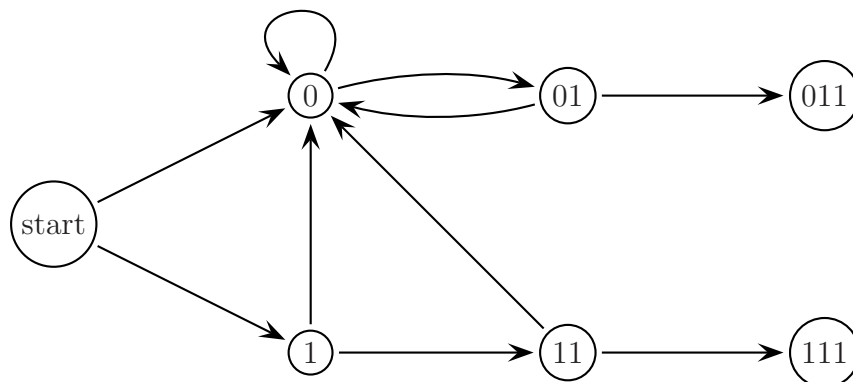


Figure 8.2: What are the chances that 111 will come up before 011? Use this digraph.

At the very start, we toss the die once and get either a 0 or 1; let's say we have rolled a 1. Consider the next roll; either this is a 1 (so we have "11") or a 0. Recall that the sequences we are looking for are "011" and "111". Since "10" does not occur in either final sequence and order is important!, the first roll is discarded and we only take note of the 0. Take a moment to convince yourself that the vertices and arcs in the digraph above describe all the possible events in this game.

finish this game! discuss counting paths, $(1/2)^n$ for paths of length n , etc.

□

Exercise 8.9

What are the chances that Aristophanes will win?

Exercise 8.10

Suppose this game is played with an unfair coin. Let's say the chances of a 0 is $1/3$, while the chances of a 1 is $2/3$. In this case, the arcs should be labelled with either $1/3$ or $2/3$. What are the chances that Aristophanes will win using this coin?

Exercise 8.11

Alice chooses the number 010 and Bob chooses the number 101. Sketch an appropriate digraph and repeat the exercises above.

Of course, the game above could be played with a "normal" coin; just let "1" corresponds to "heads" and "0" correspond to "tails", for instance. By doing something as simple as playing a coin game, we have had the opportunity to ask some very challenging questions and to experience some rather interesting mathematics. The theory of probability is a rich and interesting field which arises from a few familiar first concepts. Here is another game which is easy to state.

Example 8.12

Suppose we roll a fair 6-sided die repeatedly. Here is the question: what are the chances that both a 1 and a 3 both occur before an even number occurs?

In this game, an outcome is a single roll of the die and can be any whole number from 1 to 6. An event is a sequence of 5 consecutive rolls of the die. The event “1,5,3,2,3” works, since both 1 and 3 occur before 2. Another successful event is “4,3,5,1,1”. The 1 and 3 needn’t occur “back-to-back” or “in order”, as long as they occur before any even number. On the other hand, the event “1,2,3,3,2” does not match the question, since 2 is an even number and occurs before the 3. Can you think of other events that work or do not work? There are *many* possibilities, and this is why a carefully-drawn digraph is so important. Here is a diagram to describe how the game is played:

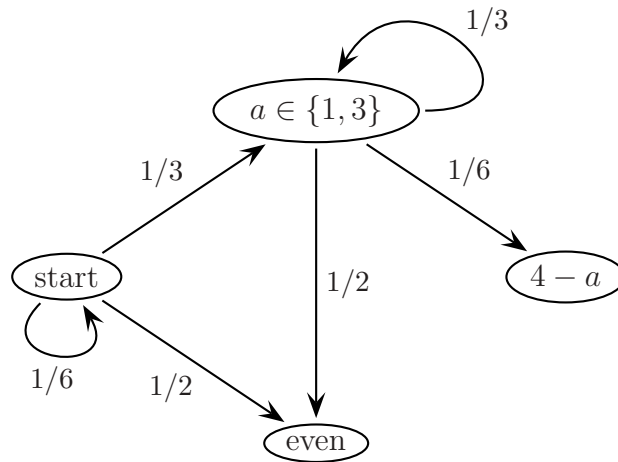


Figure 8.3: Toss a fair 6-sided die, looking for a 1 *and* a 3 before an even number.

In order to answer our question, we have created a digraph that we will call a *chip-firing machine*. We will place markers or *chips* onto a vertex. Since a regular die has six faces, we allow “firing” when a vertex holds six or more chips. By firing, we mean that we distribute the chips on a particular vertex to the neighboring vertices, which we will describe more below. In Figure 8.4, we see how the first firing takes place.

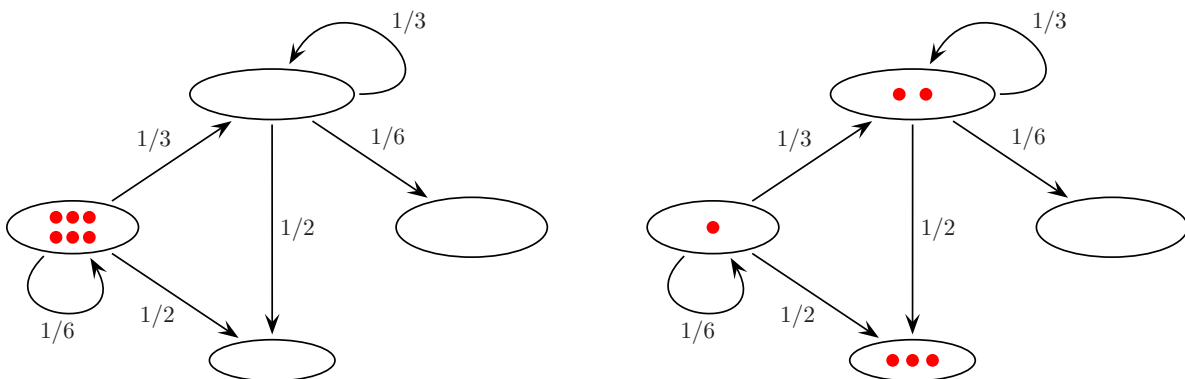


Figure 8.4: At left we see there are 6 chips at the start, so this vertex is ready to fire. After firing, we see how the chips were distributed at right. Several chips must be placed before firing again.

Notice that there is only one internal state in this digraph. Since nothing will happen unless there are at least 6 chips on at least one vertex, we sometimes “pre-load” a vertex with one or more

chips. Without loss of generality, load the internal state in the diagram with five chips, and begin the game by feeding 6 chips into the start.

We are ready to play the game. The game will be finished when the initial loading of the internal state recurs (with 5 chips in this case). The probabilities we have assigned to the edges determine the distribution of the chips during each firing.

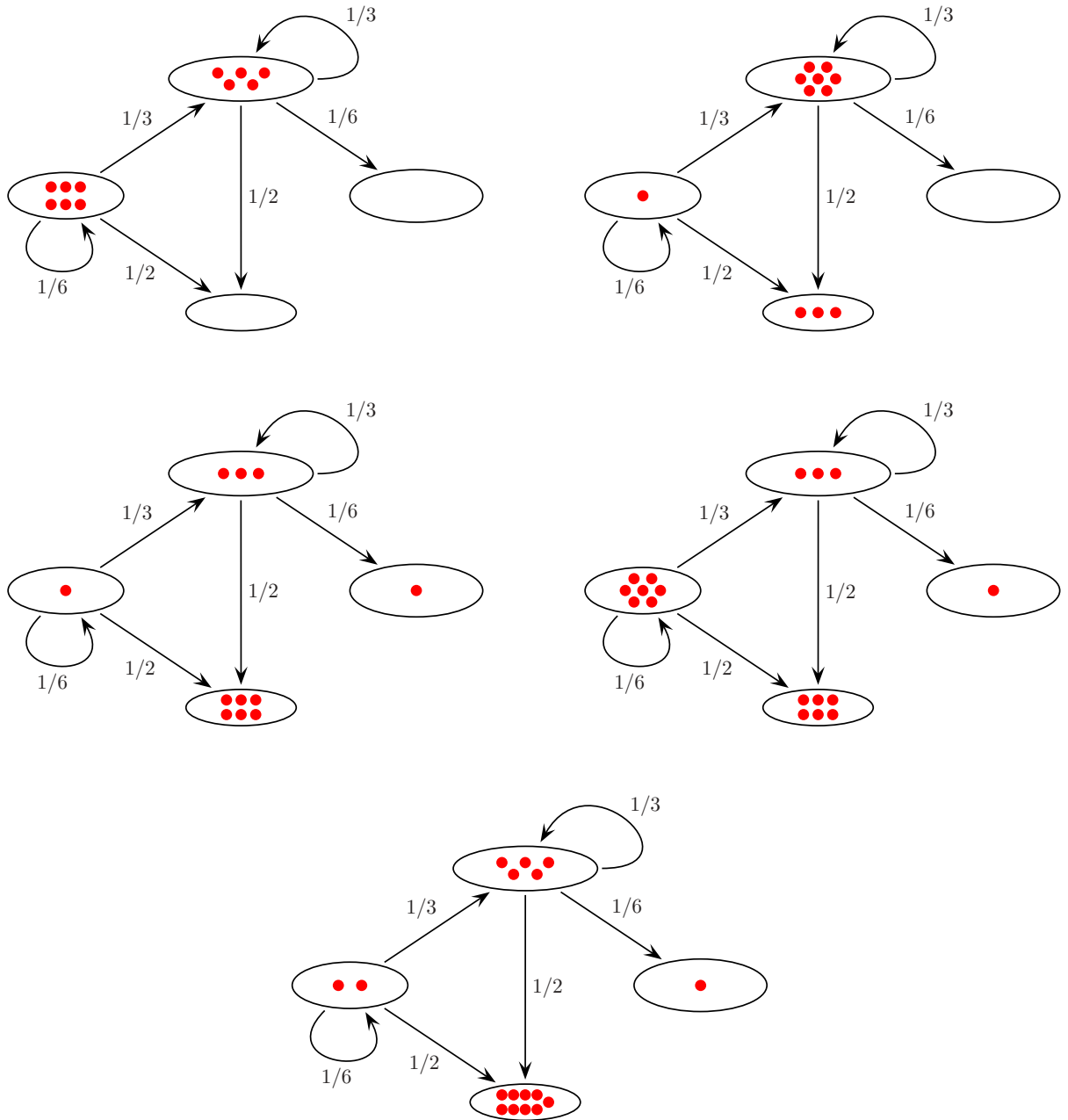


Figure 8.5: The game, start to finish (left-to-right, then down).

say more here

From the chain of diagrams in Figure 8.5, one can see that the chances of getting 1 *and* 3 before an even number is $1/10$. The expected duration of the game (that is, total number of steps made, divided by the number of chips absorbed) is $18/10 = 1.8$. □

Exercise 8.13

Why is an event a sequence of five consecutive rolls of the die? Why pre-load we label the internal vertex with 5 chips?

Exercise 8.14

Suppose we want to know the chances that “1,3” occurs before an even number. That is, we toss a fair 6-sided die repeatedly and look for a 1 then a 3, both before an even number. For example, “1,5,3,2,3” and “1,1,3,6,5” are successful events, while “1,2,3,3,2” and “5,3,3,1,4” are not successful. Sketch an appropriate digraph and label the arcs.

more exercises, especially on duration and mean value?

The way we moved chips around the chip-firing machine, and some of the other digraphs we have seen, might remind you of an abacus, a primitive sort of calculating machine. In fact, several of the problems in this section were drawn from a paper entitled “The probabilistic abacus” (Engel, [4]), and that paper contains several more examples. For a more theoretical discussion, see also [5]. For more on the mathematics of gambling-type games, see [3].

9 Formal definitions

In preceding sections we have considered a few problems that can be described in terms of the same mathematical concept, called a *Cellular Automaton*. An automaton is a (real or imaginable) device capable to perform certain actions according to a specific program, without human interference. We do not discuss here as to who has created the program. A *mathematical* (not necessarily cellular) automaton is a model describing this behavior. This model involves five mathematical objects, three sets and two mappings¹. The cellular automata can be also defined in this way, and this is our goal in this section. While reading this section, the reader should keep in mind the exercises in Section 6.

Recall that in the preceding examples, we have actually dealt with sequences of zeros and ones and changed one such sequence to another, subject to some specified rules. We can imagine an infinite tape separated into equal squares, called *cells*, with the digits 0 and 1 written in these squares, exactly one digit in each cell. This is why the model is called a “cellular” automaton. Thus, we can assume each row of the Pascal triangles in the preceding sections to be written on such tape. When we go downward in the triangle, from one line to another, we change the contents of its cells according to the transition rule described. We assume this tape is infinite in both directions with all empty cells initially occupied by zeros.

The set of symbols used by the automaton, in our case the set $\mathbb{Z}_2 = \{0, 1\}$, is called the (internal) *alphabet* of the automaton. It is possible to use any set of different characters, not just two digits. Moreover, rather than a one-dimensional tape, we can consider a plane or even a many-dimensional

¹Namely, a mathematical automaton is a quintuple $\{X, Y, S, f, g\}$. The sets X , Y , and S are called the *input alphabet*, the *output alphabet*, and the set of (internal) *states*. The map $f : X \times S \rightarrow Y$ is called the *output function*, and the map $G : X \times S \rightarrow S$ is called the *transition function*.

space separated into cells, or (which is, of course, the same) we can place these characters at the points with integer Cartesian coordinates (also called cells or sites) of the n -dimensional Euclidean space. However, for our introductory exposition, it is enough to consider only two symbols and one tape². We call these objects *1-Dimensional Binary Cellular Automata* (1-D BCA) and while referring to CA in this section, we shall always mean 1-D BCA.

Now, given a CA, at every time instance we have an infinite sequence of zeros and ones written on the tape. So we can forget about the tape and consider only this sequence of zeros and ones. Such infinite sequences are called *states* of a cellular automaton. For example, $s' = (\dots, 0, 0, 0, \dots)$, $s'' = (\dots, 1, 1, 1, \dots)$, and $s''' = (\dots, 1, 0, 1, 0, \dots)$ are three different states. The set of all states, that is, the set of all double-infinite binary sequences is denoted by S . Sometimes we need also to specify the *zero term* of this sequence, that is, the zero cell of a state. Hereafter, the initial term (cell) is denoted by a “hat” like $s = (\dots, 0, \hat{0}, 0, \dots)$. If we denote a state as a sequence $s = (\dots, c_{-2}, c_{-1}, c_0, c_1, c_2, \dots)$, we assume the initial cell to be c_0 .

A CA operates in discrete time. This means that at an initial moment in time, let us denote it by t_0 , the CA is in its *initial state* $s_0 = s(t_0)$. Within a certain time unit, all the cells *synchronously* change their holdings and at the next time moment t_1 , the CA transfers to the next state s_1 . In the following time moments, these steps repeat. We often denote these moments by $0, 1, 2, \dots$. When we need to show the dependence of the cells’ holdings on the time moment t , we write $s(t) = (\dots, c_{-2}(t), c_{-1}(t), c_0(t), c_1(t), c_2(t), \dots)$, etc.

The *transition rule* of a CA is a function f which, given a state s_t at the moment t , calculates the state s_{t+1} at the next moment $t+1$. That is, $f : S \rightarrow S$, or more specifically, $s(t+1) = f(s(t))$. For instance, in Exercise 3.4(b) on page 3, the transition rule for Pascal’s triangle can be written as $c_k(t+1) = c_{k-1}(t) + c_{k+1}(t)$, where k and t are integers and $t \geq 0$. The transition rule in Section 4 is given by the formula $c_k(t+1) = c_{k-1}(t) + c_{k+1}(t) \pmod{2}$. To apply these transition rules, we must specify the initial state $s(0)$. In both examples above, $s = (\dots, 0, \hat{1}, 0, \dots)$; that is, $c_0(0) = 1$ and $c_k(0) = 0$ if $k \leq 1$. To define a transition rule, it is often convenient to assign its action on specific cells as in the examples above. In terms of abstract automata theory, CAs are autonomous automata without output.

Thus, we arrive at the following basic definition.

Definition. A cellular automaton is a triple $\mathfrak{A} = (S, f, s(0))$, where S is a set of double-infinite 0-1 sequences, called the states of the automaton, a map $f : S \rightarrow S$ is called its transition rule, and $s(0) \in S$ is its initial state.

The collection of all consecutive states s_0, s_1, s_2, \dots of an automaton is called its *evolution*.

Example 9.1

Let S consist of sequences containing exactly one “1”, the initial state be $s(0) = (\dots, 0, \hat{1}, 0, \dots)$, and the transition rule be given by the following equations:

$$c_{t+1}(t+1) = \sum_{k=-\infty}^{\infty} c_k(t) \quad \text{and} \quad c_k(t+1) = 0 \text{ for } k \neq t+1.$$

Exercise 9.2

- (a) Verify that this rule moves the “1” consecutively one step to the right.
- (b) Construct a CA that moves the “1” consecutively one step to the left.

²From a mathematical point of view, these extensions do not provide a more general theory.

Example 9.3

We sometimes consider *periodic* states. This means that there exists a finite group of the digits 0 and 1 which repeats endlessly in both directions. Thus, in the state s' above this group consists of one symbol “0”, the length of this period is 1. In s''' above, the period – which is $(1, 0)$ – has the length of 2. In s' and s'' , the initial cell is irrelevant, but it should be noted that the state $s'''' = (\dots, 1, 0, 1, 0, 1, \dots)$ is different from both s' and s'' . It also has the period of the length 2, however now the period is $(0, 1)$. Certainly, the groups $(1, 0, 1, 0)$, $(1, 0, 1, 0, 1, 0)$, etc., are also periods of s'''' , but we always look for the shortest period. Periodic states of length N are denoted by $s = (c_0, c_1, \dots, c_{N-1})$, where N is assumed to be the length of the shortest period and c_0 occupies the initial cell – we do not “hat” it now. Due to the periodicity, $c_{-1} = c_{N-1}$, $c_0 = c_N$, and so forth. We call these equations *periodic boundary conditions*. So we can write down the same state as $s = (\dots, c_{N-1}, c_0, c_1, \dots, c_{N-1}, c_0, c_1, \dots)$.

Exercise 9.4

Determine periods of the following states:

- (a) $(1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1)$
- (b) $(\dots, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, \dots)$
- (c) What about Part (c) of Exercise 6.1?

Answer: For (a), the period is $(1, 1, 0, 1)$, its length is 4, so the state can be written as $(1, 1, 0, 1)$. In both (b) and (c), these states have no period.

For further examples, review all relevant examples from the preceding sections.

Given a time instant t and the state $s(t) = (\dots, c_{-2}(t), c_{-1}(t), c_0(t), c_1(t), c_2(t), \dots)$, a transition rule must calculate the contents of all cells at time $t + 1$. This means that the transition rule must involve, explicitly or implicitly, functions $c_k(t + 1) = F_k(\dots, c_{-2}(t), c_{-1}(t), c_0(t), c_1(t), c_2(t), \dots)$ for some integer k . The functions F_k may themselves depend on t ; that is, they may depend on the preceding states $s(t - 1)$, $s(t - 2)$, etc. However, we consider only a simple case when the functions F_k (that is, the transition rule f) does not depend on t . This means that

$$\begin{aligned}c_k(1) &= F_k(\dots, c_{-2}(0), c_{-1}(0), c_0(0), c_1(0), c_2(0), \dots) , \\c_k(2) &= F_k(\dots, c_{-2}(1), c_{-1}(1), c_0(1), c_1(1), c_2(1), \dots) , \\c_k(3) &= F_k(\dots, c_{-2}(2), c_{-1}(2), c_0(2), c_1(2), c_2(2), \dots) , \text{ etc.}\end{aligned}$$

Moreover, we assume that the functions F_k depend only on the cell itself and its $2r$ closest neighbors but do not depend on k . That is, the transition rule is given by a fixed function F depending on $2r + 1$ arguments:

$$c_k(t + 1) = F(c_{k-r}(t), c_{k-r+1}(t), \dots, c_{k-1}(t), c_k(t), c_{k+1}(t), \dots, c_{k+r-1}(t), c_{k+r}(t))$$

for some integer k . We call such automata *isotropic*. The number $r \geq 1$ is called the *range* of a CA. Thus, in the example with Pascal’s triangle, $r = 1$.

Exercise 9.5

In Example 9.1, what is r ?

Example 9.6

Let $r = 2$ and $F(a, b, c, d, e) = (a - b + c - d + e) \pmod 2$. Assume $s_0 = s(0) = (\dots, 0, \hat{1}, 0, \dots)$, calculate $s(1) = (\dots, c_{-1}(1), c_0(1), c_1(1), \dots)$.

We have the following:

$$\begin{aligned}
 c_0(1) &= F(c_{-2}(0), c_{-1}(0), c_0(0), c_1(0), c_2(0)) \\
 &= (0 - 0 + 1 - 0 + 0) \pmod 2 = 1 \pmod 2 = 1 ; \\
 c_1(1) &= F(c_{-1}(0), c_0(0), c_1(0), c_2(0), c_3(0)) \\
 &= (0 - 1 + 0 - 0 + 0) \pmod 2 = -1 \pmod 2 = 1 ; \\
 c_{-1}(1) &= F(c_{-3}(0), c_{-2}(0), c_{-1}(0), c_0(0), c_1(0)) \\
 &= (0 - 0 + 0 - 1 + 0) \pmod 2 = -1 \pmod 2 = 1 ; \\
 c_2(1) &= F(c_0(0), c_1(0), c_2(0), c_3(0), c_4(0)) \\
 &= (1 - 0 + 0 - 0 + 0) \pmod 2 = 1 \pmod 2 = 1 ; \\
 c_{-2}(1) &= F(c_{-4}(0), c_{-3}(0), c_{-2}(0), c_{-1}(0), c_0(0)) \\
 &= (0 - 0 + 0 - 0 + 1) \pmod 2 = 1 \pmod 2 = 1 .
 \end{aligned}$$

Clearly, $c_k(1) = 0$ if $|k| \geq 3$, since the evaluation of these $c_k(1)$ do not involve $c_0(0) = 1$. Therefore $s(1) = (\dots, 0, 1, 1, \hat{1}, 1, 1, 0, \dots)$.

Exercise 9.7

Find $s(2)$, $s(4)$, $s(4)$, and $s(5)$ for this CA.

Exercise 9.8

* Can you use the Principle of Mathematical Induction to find all states of this CA?

The transition rule in this example can be written as a linear function,

$$c_k(t+1) = c_{k-2}(t) + c_{k-1}(t) + c_k(t) + c_{k+1}(t) + c_{k+2}(t) ,$$

where k is an integer. This is an example of a *linear* rule. Generally, a transition rule is called linear if $F(a, b, \dots, m) = \alpha a + \beta b + \dots + \mu m$, where $\alpha, \beta, \dots, \mu$ are constants.

Exercise 9.9

Design a linear transition rule with $r = 1$ and calculate the evolution of this CA.

Now consider the following transition rule with $r = 1$: $F(a, b, c) = (a^2 + b^2 + c^2) \pmod (2)$. This rule is obviously non-linear.

Exercise 9.10

Find the evolution of the CA with this transition rule and the initial state $s(0) = (\dots, 0, \hat{1}, 0, \dots)$. Find a linear transition rule that generates an equivalent CA, that is, an automaton with the same evolution.

Exercise 9.11

* Is it always possible to find an equivalent linear transition rule for a general transition rule $F(c_{k-r}(t), c_{k-r+1}(t), \dots, c_{k-1}(t), c_k(t), c_{k+1}(t), \dots, c_{k+r-1}(t), c_{k+r}(t))$?

Exercise 9.12

* For a general transition rule $F(c_{k-r}(t), c_{k-r+1}(t), \dots, c_{k-1}(t), c_k(t), c_{k+1}(t), \dots, c_{k+r-1}(t), c_{k+r}(t))$, is it always possible to find an equivalent transition rule of kind $f(a, b, \dots, m) = f(\alpha a + \beta b + \dots + \mu m)$? That is, a rule depending only on a linear combination of its arguments.

Next, we introduce several important classes of CA. If the function F does not actually depend on $c_k(t)$, that is, it is a function of only $2r$ arguments, then the automaton is said to be “without memory”. An automaton or a corresponding transition rule is said to be *symmetric* if

$$F(c_{k-r}(t), c_{k-r+1}(t), \dots, c_{k-1}(t), c_k(t), c_{k+1}(t), \dots, c_{k+r-1}(t), c_{k+r}(t)) \\ = F(c_{k+r}(t), c_{k+r-1}(t), \dots, c_{k+1}(t), c_k(t), c_{k-1}(t), \dots, c_{k-r+1}(t), c_{k-r}(t)) .$$

An automaton or a corresponding transition rule is called *legal* if $F(0, 0, \dots, 0) = 0$ (an empty state can generate only itself).

Examples.

Representations of cellular automata. – Digraphs (diagrams of states), Tables of transfers, Vertices of the unit cube.

To “materialize” an automaton, we can use different ways, depending on our task. For instance, given a transition rule, the corresponding CA can be represented by a directed graph, as we have seen in preceding sections.

More examples.

To store a CA in computer memory, it is convenient to use *transfer tables*. We again consider a 1-D BCA with the range r , that is, its transfer function (the rule) depends on $2r + 1$ arguments and the internal alphabet is $\mathbb{Z}_2 = \{0, 1\}$. We can assign such a transfer rule as a chart with $2r + 1$ columns for the arguments and an additional column for the values of the transfer function. Since the entry of each cell in this chart may be either 0 or 1, there are 2^{2r+1} possibilities to fill in these rows for the arguments; that is, the chart must contain 2^{2r+1} rows:

$c_{k-r}(t)$	$c_{k-r+1}(t)$	\dots	$c_k(t)$	\dots	$c_{k+r-1}(t)$	$c_{k+r}(t)$	$c_k(t+1)$
0	0	\dots	0	\dots	0	0	
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	
1	1	\dots	1	\dots	1	1	

where $c_k(t+1) = F(c_{k-r}(t), \dots, c_k(t), \dots, c_{k+r}(t))$.

Since there are 2^{2r+1} cells in the last column and two possible entries, there are $2^{2^{2r+1}}$ ways to complete the chart. So there are $2^{2^{2r+1}}$ one-dimensional binary cellular automata. For example, the following chart exhibits the trivial CA, which transforms any state to an empty one, with all

cells containing only zeros:

$c_{k-r}(t)$	$c_{k-r+1}(t)$	\cdots	$c_k(t)$	\cdots	$c_{k+r-1}(t)$	$c_{k+r}(t)$	$c_k(t+1)$
0	0	\cdots	0	\cdots	0	0	0
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
1	1	\cdots	1	\cdots	1	1	0

Exercise 9.13

How many 1-D BCA are there without memory? Legal CA? Symmetric CA?

Answer: $2^{2^{2r}}$; $2^{2^{2r+1}-1}$; $2^{2^{2r}}$.

Exercise 9.14

How many 1-D CA are there with the alphabet containing k symbols?

Answer: $k^{k^{2r+1}}$.

We again consider binary CA, so let $k = 2$. The $2^{2^{2r+1}}$ different right-most columns, representing $2^{2^{2r+1}}$ various automata or transition rules, can be viewed as the binary expansions of the whole numbers $0, 1, 2, \dots, 2^{2^{2r+1}} - 1$. These decimal numbers are often considered as labels of different transition rules. For example, since the decimal number 17 has the binary representation 00010001, the rule 17 can be written as

$c_{k-1}(t)$	$c_k(t)$	$c_{k+1}(t)$	$c_k(t+1)$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

When $k = 2$, the charts above represent the truth tables of Boolean functions, that is, mappings $f : \mathbb{Z}_2 \times \mathbb{Z}_2 \times \cdots \times \mathbb{Z}_2 \rightarrow \mathbb{Z}_2$, where $\mathbb{Z}_2 \times \mathbb{Z}_2 \times \cdots \times \mathbb{Z}_2$ is the Cartesian product of $2^{2^{2r+1}}$ copies of \mathbb{Z}_2 . In this sense, the theory of 1-D BCA is equivalent to the theory of Boolean functions. In particular, we can define Boolean operations: *negation*, *conjunction*, *disjunction*, *conditional implication* on BCA. For example, the negation of the rule 17 above is the rule 238, since $\neg(00010001) = (11101110)$, which is the binary representation of 238.

We call a CA *elementary* (or an *atom*) if the binary representation of its rule contains exactly one “1”. Therefore, there are $2^{2^{2r+1}}$ atoms (8 for $r = 1$).

Theorem 9.15 *Each CA can be represented by the disjunction of the atoms.*

??? Define dual, monotone, self-dual CA.
Examples.

??? Representation by Boolean polynomials.

??? Completeness.

References

- [1] P. Bak, *How Nature Works: the science of self-organized criticality*, New York: Springer-Verlag, 1996.

In this book, the author describes different models for evolution and natural phenomena. These models are based in the area of cellular automata.

- [2] E.R. Berlekamp, J.H. Conway, and R.K. Guy, *Winning Ways for Your Mathematical Plays*, London: Academic Press, 1982.

This books contains mathematical formulations of games, and more. In particular, the Game of Life (Chapter 25) is the most famous of all cellular automata.

- [3] L.E. Dubins and L.J. Savage, *How to Gamble If You Must: inequalities for stochastic processes*, New York: McGraw-Hill, 1965.

- [4] A. Engel, The probabilistic abacus, *Ed. Stud. Math.* **6** (1975), 1-22.

- [5] A. Engel, Why does the probabilistic abacus work?, *Ed. Stud. Math.* **7** (1976), 59-69.

The ideas in these two papers were the inspiration for the chip-firing games and other games in Section 8, and they contain many more useful examples.

- [6] S. Wolfram, Computer software in Science and Mathematics, *Scientific American* **251** (September 1984), 188-203.

- [7] S. Wolfram, Geometry of binomial coefficients, *Amer. Math. Monthly* **91** (1984), 566-570.

- [8] S. Wolfram, *Theory and Applications of Cellular Automata*, New York: World Scientific, 1986.

Steven Wolfram is one of the pioneers in the area of cellular automata, and is the creator of the computer algebra system *Mathematica*. Wolfram described the future of computer simulation, and this article is still relevant today.

- [9] <http://psoup.math.wisc.edu> – The Primordial Soup Kitchen.

This is one of the premiere websites on cellular automata. Links, articles and software available. See especially the following freeware programs:

- MCell (“Mirek’s Celebration”)
- Automalab
- Cellab

- [10] <http://www.dimacs.rutgers.edu> – The DIMACS Institute.

DIMACS is the Center for Discrete Mathematics and Theoretical Computer Science. Reconnect is one of many programs organized by DIMACS, which is located at Rutgers University.