

## 10 Cellular Automata and Lattice Gases

We started our discussion of partial differential equations by considering how they arise as continuum approximations to discrete systems (such as cars on a highway, or masses connected by springs). We then studied how to discretize continuum problems to develop numerical algorithms for their solution. In retrospect this might appear silly, and it can be: why bother with the continuum description at all, if the goal is to solve the problem with a digital computer rather than an analog pencil?

Consider a gas. Under ordinary circumstances, it can be described by the continuous distribution of pressure, temperature, and velocity. However, on short length scales (for example, when studying the head of a disk drive flying a micron above the platter), this approximation breaks down and it becomes necessary to keep track of the individual gas molecules. If the energy is high enough (say, in a nuclear explosion) even that is not enough; it's necessary to add the excited internal degrees of freedom in the molecules. Or if the gas is very cold, quantum interactions between the molecules become significant and it's necessary to include the molecular wave functions. Such *molecular dynamics* calculations are done routinely, but require a great deal of effort: for the full quantum case, it's a challenge to calculate the dynamics for as long as a picosecond.

Any gas can be described by such a microscopic description, but that is usually too much work. Is it possible to stop between the microscopic and macroscopic levels, without having to pass through a continuum description? The answer is a resounding yes. This is the subject of *lattice gases*, or more generally, of *cellular automata (CA)*, which in the appropriate limits reduce to both the microscopic and the macroscopic descriptions. The idea was developed by Ulam [Finkel & Edelman, 1985], von Neumann [von Neumann, 1966], and colleagues [Shannon & McCarthy, 1956] in the 1950s, and has since flourished with the development of the necessary theoretical techniques and computational hardware.

A classic example of a CA is Conway's *Game of Life* [Gardner, 1970], in which occupied sites on a grid get born, survive, or die in succeeding generations based on the number of occupied neighboring sites. Any CA has the same elements: a set of connected sites, a discrete set of states that are allowed on the sites, and a rule for how they are updated. We will start by studying a slightly more complicated system that recovers the Navier-Stokes fluid equations, and then will consider the more general question of how cellular automata relate to computation.

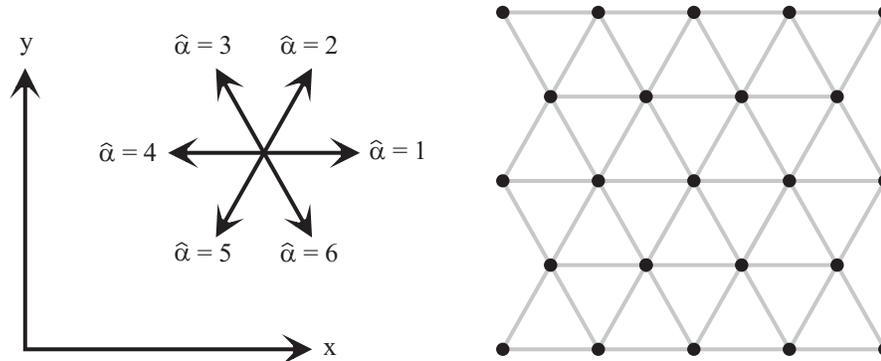


Figure 10.1. Direction indices for a triangular lattice.

### 10.1 LATTICE GASES AND FLUIDS

Hydrodynamics was one of the early successes of the theory of cellular automata, and remains one of the best-developed application areas. A cellular automata model of a fluid (traditionally called a lattice gas) is specified by the geometry of a lattice, by the discrete states permitted at each site, and by an update rule for how the states change based on their neighbors. Both partial differential equations and molecular dynamics models use real numbers (for the values of the fields, or for the particle positions and velocities). A lattice gas discretizes everything so that just a few bits describe the state of each site on a lattice, and the dynamics reduce to a simple look-up table based on the values of the neighboring sites. Each site can be considered to be a parcel of fluid. The rules for the sites implement a “cartoon” version of the underlying microscopic dynamics, but should be viewed as operating on a longer length scale than individual particles. We will see that the conservation laws that the rules satisfy determine the form of the equivalent partial differential equations for a large lattice, and that the details of the rules set the parameter values.

A historically important example of a lattice gas is the FHP rule (named after its inventors, Frisch, Hasslacher, and Pomeau [Frisch *et al.*, 1986]). This operates in 2D on a triangular lattice, and the state of each site is specified by six bits (Figure 10.1). Each bit represents a particle on one of the six links around the site, given by the unit vectors  $\hat{\alpha}$ . On each link a particle can either be present or absent, and all particles have the same unit velocity: in the absence of collisions, they travel one lattice step ahead in one time step. The simple update rule proceeds in two stages, chosen to conserve particle number and momentum (Figure 10.2). First, collisions are handled. At the beginning of the step a particle on a link is considered to be approaching the site, and after the collision step a particle on a link is taken to be leaving the site. If a site has two particles approaching head-on, they scatter. A random choice is made between the two possible outgoing directions that conserve momentum (always choosing one of them would break the symmetry of the lattice). Because of the large number of sites in a lattice gas it is usually adequate to approximate the random decision by simply switching between the two outgoing choices on alternate site updates. If three particles approach symmetrically, they scatter. In all other configurations not shown the particles pass through the collision

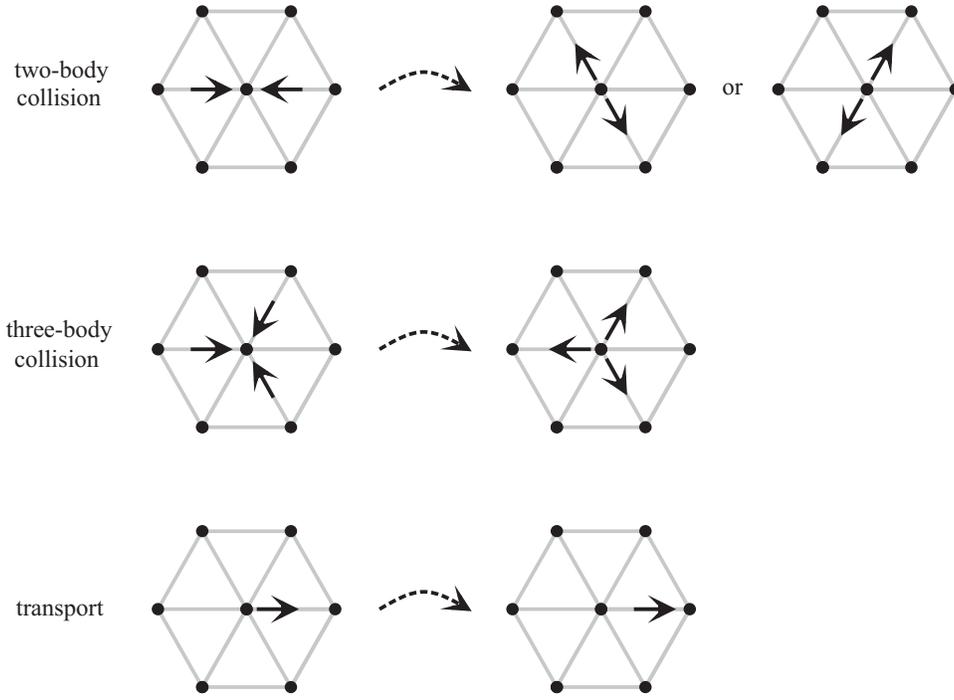


Figure 10.2. Update rules for an FHP lattice gas.

unchanged. After the collision step there is a transport step, in which each particle moves by one unit in the direction that it is pointing and arrives at the site at the far end of the link. While these rules might appear to be somewhat arbitrary, it will turn out that the details will not matter for the form of the governing equations, just the symmetries and conservation laws.

A simpler rule related to FHP is HPP (named after Hardy, de Pazzis, and Pomeau [Hardy *et al.*, 1976]), which operates on a square lattice. Each site is specified by four bits, and direction-changing collisions are allowed only when two particles meet head-on (unlike FHP, here there is only one possible choice for the exit directions after scattering). We will see that HPP and FHP, although apparently quite similar, behave very differently.

Let's label time by an integer  $T$ , the lattice sites by a vector  $\vec{X}$ , and the lattice directions by a unit vector  $\hat{\alpha}$ . If we start an ensemble of equivalent lattices off with the same update rule but different random initial conditions, we can define  $f_\alpha(\vec{X}, T)$  to be the fraction of sites  $\vec{X}$  at time  $T$  with a particle on link  $\hat{\alpha}$ . In the limit of a large ensemble, this fraction becomes the probability to find a particle on that link. Defining this probability will let us make a connection between the lattice gas and partial differential equations.

At each time step, in the absence of collisions, the fraction of particles at site  $\vec{X}$  at time  $T$  pointing in direction  $\hat{\alpha}$  will move one step in that direction:

$$f_\alpha(\vec{X} + \hat{\alpha}, T + 1) = f_\alpha(\vec{X}, T) \quad . \quad (10.1)$$

Let's introduce new rescaled variables  $\vec{x} = \delta_x \vec{X}$  and  $t = \delta_t T$  in terms of the (small) space

step  $\delta_x$  and time step  $\delta_t$ . Substituting in these variables, collision-free transport becomes

$$f_\alpha(\vec{x} + \delta_x \hat{\alpha}, t + \delta_t) - f_\alpha(\vec{x}, t) = 0 \quad . \quad (10.2)$$

If the probability  $f_\alpha$  varies slowly compared to  $\delta_x$  and  $\delta_t$ , we can expand equation (10.2) in  $\delta_x$  and  $\delta_t$ :

$$\frac{\partial f_\alpha(\vec{x}, t)}{\partial t} \delta_t + \hat{\alpha} \cdot \nabla f_\alpha(\vec{x}, t) \delta_x + \mathcal{O}(\delta^2) = 0 \quad . \quad (10.3)$$

Choosing to scale the variables so that  $\delta_x = \delta_t$ , to first order this becomes

$$\frac{\partial f_\alpha(\vec{x}, t)}{\partial t} + \hat{\alpha} \cdot \nabla f_\alpha(\vec{x}, t) = 0 \quad . \quad (10.4)$$

This equation says that the time rate of change of the fraction of particles at a point is equal to the difference in the rate at which they arrive and leave the point by straight transport (remember equation (8.6)). If collisions are allowed, the time rate of change of  $f_\alpha$  will depend on both the spatial gradient and on a collision term  $\Omega_\alpha$  scattering particles in or out from other directions

$$\frac{\partial f_\alpha(\vec{x}, t)}{\partial t} + \hat{\alpha} \cdot \nabla f_\alpha(\vec{x}, t) = \Omega_\alpha(\vec{x}, t) \quad . \quad (10.5)$$

$f_\alpha$  is the distribution function to find a particle. The collision term  $\Omega_\alpha$  will in general depend on the distribution function for pairs of particles as well as the one-particle distribution function, and these in turn will depend on the three-particle distribution functions, and so forth. This is called the *BBGKY hierarchy* of equations (Bogolyubov, Born, Green, Kirkwood, Yvon [Boer & Uhlenbeck, 1961]). The *Boltzmann equation* approximates this by assuming that  $\Omega_\alpha$  depends only on the single-particle distribution functions  $f_\alpha$ .

We derived equation (10.5) by making use of the fact that particles travel one lattice site in each time step, and then assuming that  $f_\alpha$  varies slowly. Now let's add the conservation laws that have been built into the update rules. The total density of particles  $\rho$  at a site  $\vec{x}$  is just the sum over the probability to find one in each direction

$$\sum_\alpha f_\alpha(\vec{x}) = \rho(\vec{x}) \quad , \quad (10.6)$$

and the momentum density is the sum of the probabilities times their (unit) velocities

$$\sum_\alpha \hat{\alpha} f_\alpha = \rho \vec{v} \quad . \quad (10.7)$$

Since our scattering rules conserve particle number (the particles just get reoriented), the number of particles scattering into and out of a site must balance

$$\sum_\alpha \Omega_\alpha(\vec{x}) = 0 \quad . \quad (10.8)$$

And since the rules conserve momentum, the net momentum change from scattering must vanish

$$\sum_\alpha \hat{\alpha} \Omega_\alpha = 0 \quad . \quad (10.9)$$

Therefore, summing equation (10.5) over directions,

$$\sum_{\alpha} \left[ \frac{\partial f_{\alpha}(\vec{x}, t)}{\partial t} + \hat{\alpha} \cdot \nabla f_{\alpha}(\vec{x}, t) \right] = \sum_{\alpha} \Omega_{\alpha} \quad (10.10)$$

$$\frac{\partial}{\partial t} \sum_{\alpha} f_{\alpha} + \sum_{\alpha} \hat{\alpha} \cdot \nabla f_{\alpha} = 0$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0 \quad . \quad (10.11)$$

This is the familiar equation for the continuity of a fluid, and has arisen here because we've chosen scattering rules that conserve mass. A second equation comes from momentum conservation, multiplying equation (10.5) by  $\hat{\alpha}$  and summing over directions

$$\frac{\partial}{\partial t} \sum_{\alpha} \hat{\alpha} f_{\alpha} + \sum_{\alpha} \hat{\alpha} (\hat{\alpha} \cdot \nabla f_{\alpha}) = 0 \quad . \quad (10.12)$$

The  $i$ th component of this vector equation is

$$\frac{\partial}{\partial t} \sum_{\alpha} \hat{\alpha}_i f_{\alpha} + \sum_{\alpha} \sum_j \hat{\alpha}_i \hat{\alpha}_j \frac{\partial f_{\alpha}}{\partial x_j} = 0 \quad . \quad (10.13)$$

Defining the momentum flux density tensor by

$$\Pi_{ij} \equiv \sum_{\alpha} \hat{\alpha}_i \hat{\alpha}_j f_{\alpha} \quad , \quad (10.14)$$

this becomes

$$\frac{\partial \rho v_i}{\partial t} + \sum_j \frac{\partial \Pi_{ij}}{\partial x_j} = 0 \quad . \quad (10.15)$$

We now have two equations, (10.11) and (10.15), in three unknowns,  $\rho$ ,  $\vec{v}$ , and  $\Pi$ . To eliminate  $\Pi$  we can find the continuum form of the momentum flux density tensor by using a *Chapman–Enskog expansion* [Huang, 1987], a standard technique for finding approximate solutions to the Boltzmann equation. We will assume that  $f_{\alpha}$  depends only on  $\vec{v}$  and  $\rho$  and their spatial derivatives (and not on time explicitly), and so will do an expansion in all possible scalars that can be formed from them. The lowest-order terms of the deviation from the equilibrium uniform configuration are

$$\begin{aligned} f_{\alpha} = & \frac{\rho}{6} \left( 1 + 2\hat{\alpha} \cdot \vec{v} + A \left[ (\hat{\alpha} \cdot \vec{v})^2 - \frac{1}{2} |\vec{v}|^2 \right] \right. \\ & \left. + B \left[ (\hat{\alpha} \cdot \nabla)(\hat{\alpha} \cdot \vec{v}) - \frac{1}{2} \nabla \cdot \vec{v} \right] + \dots \right) \quad . \end{aligned} \quad (10.16)$$

The terms have been grouped this way to guarantee that the solution satisfies the density and momentum equations (10.6) and (10.7) (this can be verified by writing out the components of each term). In this derivation the only features of the FHP rule that we've used are the conservation laws for mass and momentum, and so all rules with these features will have the same form of the momentum flux density tensor (to this order), differing only in the value of the coefficients  $A$  and  $B$ .

The *Navier–Stokes* governing equation for a  $d$ -dimensional fluid is

$$\frac{\partial \rho \vec{v}}{\partial t} + \mu \rho (\vec{v} \cdot \nabla) \vec{v} = -\nabla p + \eta \nabla^2 \vec{v} + \left( \zeta + \frac{\eta}{d} \right) \nabla (\nabla \cdot \vec{v}) \quad , \quad (10.17)$$

where  $p$  is the pressure,  $\eta$  is the shear viscosity,  $\zeta$  is the bulk viscosity, and  $\mu = 1$  for translational invariance [Batchelor, 1967]. Using the Chapman–Enskog expansion to evaluate the momentum flux density tensor in equation (10.15) and comparing it with the Navier–Stokes equation shows that they agree if  $\zeta = 0$ ,  $\eta = \rho \nu = -\rho B/8$  ( $\nu$  is the kinematic viscosity),  $\mu = A/4$ , and  $p = \rho/2$ . Further, in the Boltzmann approximation it is possible (with a rather involved calculation) to find the values of  $A$  and  $B$  for a given CA rule [Wolfram, 1986].

The simple conservation laws built into our lattice gas have led to the full Navier–Stokes equation; the particular rule determines the effective viscosity of the fluid. While the details of this calculation are complicated, there are some simple and important conclusions. The viscosity for the square lattice (HPP model) turns out to depend on direction and hence is not appropriate for most fluids, but the viscosity of the triangular lattice (FHP model) is isotropic. The simulations in Problems 10.1 and 10.2 will show just how striking the difference is. This profound implication of the lattice symmetry was not appreciated in the early days of lattice gas models, and helped point the way towards the realization that a simple lattice gas model could in fact be very general. In 3D the situation is more difficult because there is not a 3D lattice that gives an isotropic viscosity. However, it can still be achieved by using a quasiperiodic tiling that is not translationally periodic [Boghosian, 1999], or a cut through a higher-dimensional lattice such as the 4D Face-Centered Hyper-Cubic (FCHC) rule [Frisch *et al.*, 1987]. It is not possible to reduce the viscosity in a simple model like FHP (an attribute that is needed for modeling a problem such as air flow), but this can be done by adding more than one particle type or by increasing the size of the neighborhood used for the rule [Dubrulle *et al.*, 1991].

Lattice gas models have not displaced conventional numerical hydrodynamics codes, which are much more mature, and which make it easier to adjust the numerical parameters. But with much less effort they can be surprisingly competitive, they are easier to apply to more complex geometries, and the required hardware, software, and theory are all advancing rapidly [Rothman & Zaleski, 2004].

## 10.2 CELLULAR AUTOMATA AND COMPUTING

Simulating a cellular automata is an particularly simple type of computation. Rather than the many kinds of memory, instructions, and processors in a conventional computer, it requires just storage for the bits on the lattice, and an implementation of the local update rule. This is an extreme form of a *SIMD* (Single Instruction Multiple Data) parallel computer. The update can be performed by using the bits at each site as an index into a look-up table, so the architecture reduces to memory cycling through a look-up table (where the sequence of the memory retrieval determines the lattice geometry). This means that relatively modest hardware can match or exceed the performance of a supercomputer for CA problems [Toffoli & Margolus, 1991].

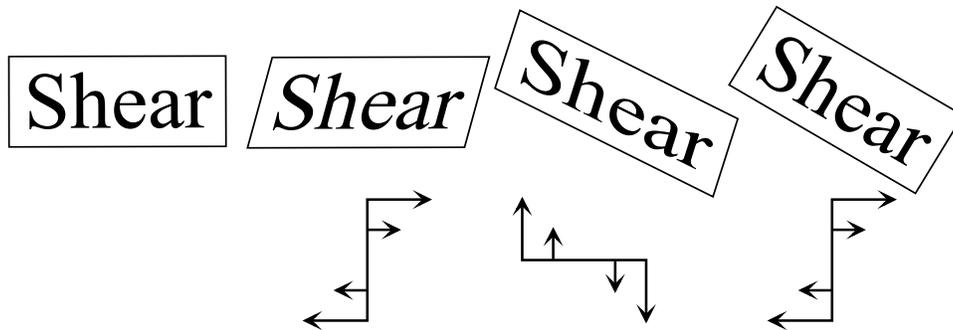


Figure 10.3. Rotation by three shears.

We've seen that a cellular automata computer can simulate fluids. A wide range of other physical systems also can be modeled by cellular automata rules; an important example is the rendering of 3D graphics. An obvious possibility is to replace the ray-tracing calculations used in computer graphics by the propagation of lattice photons. Less obviously, global rotations in 2D and 3D can be done by shear operations, which can be implemented efficiently on a CA-style computer (Figure 10.3) [Toffoli & Quick, 1997]. Therefore, by using cellular automata 3D graphics can feasibly be done from a first-principles description.

Conversely, instead of using a computer to simulate a CA, a CA can be used to simulate a computer. One way to do this is by implementing Boolean logic in CA rules; this was used to first prove their computational universality [Banks, 1971]. This approach is attractive for hardware scaling because it builds in physical constraints; any technology that can perform the local cell updates can execute the same global programs [Gershenfeld *et al.*, 2010].

Alternatively, since CAs can model physical systems, and physical systems can compute, CAs can model physical systems that compute [Margolus, 1984]. Consider the billiard-ball CA in Figure 10.4 (the underlying lattice is not shown). This is similar to a lattice gas: billiard balls move on a lattice with unit velocity, and scatter off of each other and from walls. Two balls colliding generates the AND function, and if one of the streams of balls is continuous it generates the NOT function of the other input. These two elements are sufficient to build up all of logic [Hill & Peterson, 1993]. Memory can be implemented by delays, and wiring by various walls to guide the balls. The balls can be represented by four bits per site (one for each direction), with one extra bit per site needed to represent the walls.

This kind of computing has many interesting features [Fredkin & Toffoli, 1982]. No information is ever destroyed, which means that it is reversible (it can be run backwards to produce inputs from outputs) [Bennett, 1988], and which in turn means that it can be performed (in theory) with arbitrarily little dissipation [Landauer, 1961]. Reversibility is also essential for designing quantum cellular automata, since quantum evolution is reversible. A quantum CA is much like a classical CA, but it permits the sites to be in a superposition of their possible states [Lloyd, 1993]. This is a promising architecture for building quantum computers based on short-range interactions [Nielsen & Chuang, 2000].

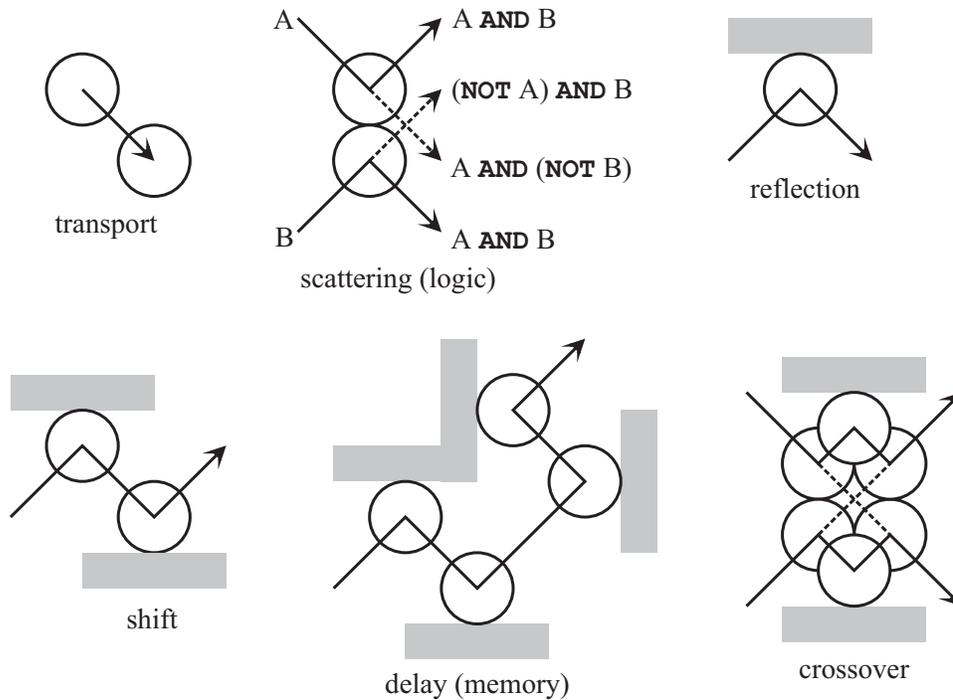


Figure 10.4. Billiard ball logic.

For some, cellular automata are much more than just an amusing alternative to traditional models of computation [Fredkin, 1990]. Most physical theories are based on real numbers. This means that a finite volume of space contains an infinite amount of information, since its state must be specified with real numbers. But if there is an energetic cost to creating information (as there is in most theories), then this implies an infinite amount of energy in a finite space. This is obviously unacceptable; something must bound the information content of space. While such a notion can arise in quantum field theories, CAs start as discrete theories that do not have this problem, and so in many ways they are more satisfying than differential equations as a way to specify governing equations. There is nothing less basic about them than differential equations; which is more “fundamental” depends on whether you are solving a problem with a pencil or a computer.

### 10.3 SELECTED REFERENCES

[Hasslacher, 1987] Hasslacher, Brosl (1987). Discrete Fluids. *Los Alamos Science*, 175–217.

A very good introductory paper for lattice gases.

[Doolen *et al.*, 1990] Doolen, Gary D. Frisch, Uriel, Hasslacher, Brosl, Orszag, Steven, & Wolfram, Stephen (eds) (1990). *Lattice Gas Methods for Partial Differential Equations*. Santa Fe Institute Studies in the Sciences of Complexity. Reading, MA: Addison-Wesley.

This collection includes many of the important articles, including [Wolfram, 1986] and [Frisch *et al.*, 1987], which work out the connection between lattice gases and hydrodynamics.

[Rothman & Zaleski, 2004] Rothman, Daniel H., & Zaleski, Stephane. (2004). *Lattice-Gas Cellular Automata: Simple Models of Complex Hydrodynamics*. Cambridge: Cambridge University Press.

Reviews the basic theory and extends it to porous media and fluids with multiple components.

## 10.4 PROBLEMS

- (10.1) Simulate the HPP lattice gas model. Take as a starting condition a randomly seeded lattice, but completely fill a square region in the center and observe how it evolves. Use periodic boundary conditions. Now repeat the calculation, but leave the lattice empty outside of the central filled square.
- (10.2) Simulate the FHP model, with the same conditions as the previous model, and describe the difference. Alternate between the two possible two-body rules on alternate site updates.