Note that the exponent contains all connected diagrams, including those that can be disconnected by removal of a single line.

(c) Using the properties of the grand partition function (see Problem 7.7), find diagrammatic expressions for the average number of particles and the pressure of this gas.

(d) Keeping only the first diagram in each sum, express $\overline{N}(\mu)$ and $P(\mu)$ in terms of an integral of the Mayer f-function. Eliminate μ to obtain the same result for the pressure (and the second virial coefficient) as derived in the text.

(e) Repeat part (d) keeping the three-dot diagrams as well, to obtain an expression for the third virial coefficient in terms of an integral of f-functions. You should find that the Λ-shaped diagram cancels, leaving only the triangle diagram to contribute to $C(T)$.

8.2 The Ising Model of a Ferromagnet

In an ideal *para*magnet, each microscopic magnetic dipole responds only to the external magnetic field (if any); the dipoles have no inherent tendency to point parallel (or antiparallel) to their immediate neighbors. In the real world, however, atomic dipoles *are* influenced by their neighbors: There is always some preference for neighboring dipoles to align either parallel or antiparallel. In some materials this preference is due to ordinary magnetic forces between the dipoles. In the more dramatic examples (such as iron), however, the alignment of neighboring dipoles is due to complicated quantum-mechanical effects involving the Pauli exclusion principle. Either way, there is a contribution to the energy that is greater or less, depending on the relative alignment of neighboring dipoles.

When neighboring dipoles align parallel to each other, even in the absence of an external field, we call the material a **ferromagnet** (in honor of iron, the most familiar example). When neighboring dipoles align antiparallel, we call the material an **antiferromagnet** (examples include Cr, NiO, and FeO). In this section I'll discuss ferromagnets, although most of the same ideas can also be applied to antiferromagnets.

The long-range order of a ferromagnet manifests itself as a net nonzero magnetization. Raising the temperature, however, causes random fluctuations that decrease the overall magnetization. For every ferromagnet there is a certain critical temperature, called the **Curie temperature**, at which the net magnetization becomes zero (when there is no external field). Above the Curie temperature a ferromagnet becomes a paramagnet. The Curie temperature of iron is 1043 K, considerably higher than that of most other ferromagnets.

Even below the Curie temperature, you may not notice that a piece of iron is magnetized. This is because a large chunk of iron ordinarily divides itself into **domains** that are microscopic in size but still contain billions of atomic dipoles. Within each domain the material is magnetized, but the magnetic field created by all the dipoles in one domain gives neighboring domains a tendency to magnetize in the opposite direction. (Put two ordinary bar magnets side by side and you'll see why.) Because there are so many domains, with about as many pointing one

way as another, the material as a whole has no net magnetization. However, if you heat a chunk of iron in the presence of an external magnetic field, this field can overcome the interaction between domains and cause essentially *all* the dipoles to line up parallel. Remove the external field after the material has cooled to room temperature and the ferromagnetic interaction prevents any significant realigning. You then have a "permanent" magnet.

In this section I'd like to model the behavior of a ferromagnet, or rather, of a single domain within a ferromagnet. I'll account for the tendency of neighboring dipoles to align parallel to each other, but I'll neglect any long-range magnetic interactions between dipoles. To simplify the problem further, I'll assume that the material has a preferred axis of magnetization, and that each atomic dipole can only point parallel or antiparallel to this axis.* This simplified model of a magnet is called the **Ising model**, after Ernst Ising, who studied it in the 1920s.† Figure 8.3 shows one possible state of a two-dimensional Ising model on a 10×10 square lattice.

Notation: Let N be the total number of atomic dipoles, and let s_i be the current state of the ith dipole, with the convention that $s_i = 1$ when this dipole is pointing up, and $s_i = -1$ when this dipole is pointing down. The energy due to the interaction of a pair of neighboring dipoles will be $-\epsilon$ when they are parallel and $+\epsilon$ when they are antiparallel. Either way, we can write this energy as $-\epsilon s_i s_j$,

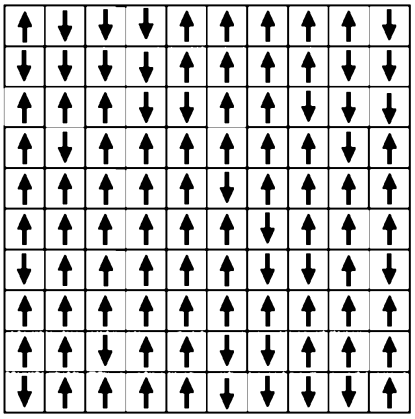

Figure 8.3. One of the many possible states of a two-dimensional Ising model on a 10×10 square lattice.

*I should point out that in many respects this model is *not* an accurate representation of a real ferromagnet. Even if there really is a preferred axis of magnetization, and even if the elementary dipoles each have only two possible orientations along this direction, quantum mechanics is more subtle than this naive model. Because we do not measure the orientation of each individual dipole, it is only the *sum* of their magnetic moments that is quantized—not the moment of each individual particle. At low temperatures, for instance, the relevant states of a real ferromagnet are long-wavelength "magnons" (described in Problem 7.64), in which all the dipoles are nearly parallel and a unit of opposite alignment is spread over many dipoles. The Ising model therefore does not yield accurate predictions for the low-temperature behavior of a ferromagnet. Fortunately, it turns out to be much more accurate near the Curie temperature.

†For a good historical overview of the Ising model see Stephen G. Brush, "History of the Lenz-Ising Model," *Reviews of Modern Physics* **39**, 883–893 (1967).

assuming that dipoles i and j are neighbors. Then the total energy of the system from *all* the nearest-neighbor interactions is

$$U = -\epsilon \sum_{\substack{\text{neighboring} \\ \text{pairs } i,j}} s_i s_j. \tag{8.38}$$

To predict the thermal behavior of this system, we should try to calculate the partition function,

$$Z = \sum_{\{s_i\}} e^{-\beta U}, \tag{8.39}$$

where the sum is over all possible sets of dipole alignments. For N dipoles, each with two possible alignments, the number of terms in this sum is 2^N, usually a *very* large number. Adding up all the terms by brute force is not going to be practical.

Problem 8.15. For a two-dimensional Ising model on a square lattice, each dipole (except on the edges) has four "neighbors"—above, below, left, and right. (Diagonal neighbors are normally not included.) What is the total energy (in terms of ϵ) for the particular state of the 4×4 square lattice shown in Figure 8.4?

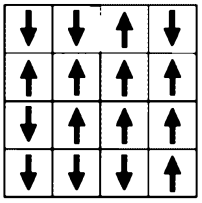

Figure 8.4. One particular state of an Ising model on a 4×4 square lattice (Problem 8.15).

Problem 8.16. Consider an Ising model of 100 elementary dipoles. Suppose you wish to calculate the partition function for this system, using a computer that can compute one billion terms of the partition function per second. How long must you wait for the answer?

Problem 8.17. Consider an Ising model of just two elementary dipoles, whose mutual interaction energy is $\pm\epsilon$. Enumerate the states of this system and write down their Boltzmann factors. Calculate the partition function. Find the probabilities of finding the dipoles parallel and antiparallel, and plot these probabilities as a function of kT/ϵ. Also calculate and plot the average energy of the system. At what temperatures are you more likely to find both dipoles pointing up than to find one up and one down?

Exact Solution in One Dimension

So far I haven't specified how our atomic dipoles are to be arranged in space, or how many nearest neighbors each of them has. To simulate a real ferromagnet, I should arrange them in three dimensions on a crystal lattice. But I'll start with a much simpler arrangement, with the dipoles strung out along a one-dimensional line (see Figure 8.5). Then each has only two nearest neighbors, and we can actually carry out the partition sum exactly.

For a one-dimensional Ising model (with no external magnetic field), the energy is

$$U = -\epsilon\big(s_1 s_2 + s_2 s_3 + s_3 s_4 + \cdots s_{N-1} s_N\big), \tag{8.40}$$

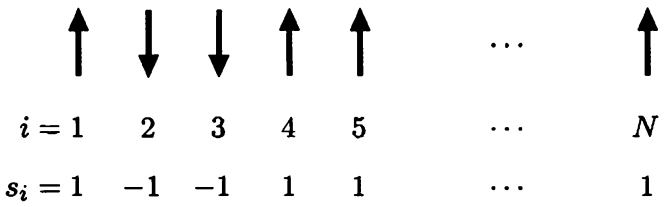

Figure 8.5. A one-dimensional Ising model with N elementary dipoles.

and the partition function can be written

$$Z = \sum_{s_1}\sum_{s_2}\cdots\sum_{s_N} e^{\beta\epsilon s_1 s_2} e^{\beta\epsilon s_2 s_3}\cdots e^{\beta\epsilon s_{N-1} s_N}, \tag{8.41}$$

where each sum runs over the values -1 and 1. Notice that the final sum, over s_N, is

$$\sum_{s_N} e^{\beta\epsilon s_{N-1} s_N} = e^{\beta\epsilon} + e^{-\beta\epsilon} = 2\cosh\beta\epsilon, \tag{8.42}$$

regardless of whether s_{N-1} is $+1$ or -1. With this sum done, the sum over s_{N-1} can now be evaluated in the same way, then the sum over s_{N-2}, and so on down to s_2, yielding $N-1$ factors of $2\cosh\beta\epsilon$. The remaining sum over s_1 gives another factor of 2, so the partition function is

$$Z = 2^N(\cosh\beta\epsilon)^{N-1} \approx (2\cosh\beta\epsilon)^N, \tag{8.43}$$

where the last approximation is valid when N is large.

So we've got the partition function. Now what? Well, let's find the average energy as a function of temperature. By a straightforward calculation you can show that

$$\overline{U} = -\frac{\partial}{\partial\beta}\ln Z = -N\epsilon\tanh\beta\epsilon, \tag{8.44}$$

which goes to $-N\epsilon$ as $T \to 0$ and to 0 as $T \to \infty$. Therefore the dipoles must be randomly aligned at high temperature (so that half the neighboring pairs are parallel and half are antiparallel), but lined up parallel to each other at $T = 0$ (achieving the minimum possible energy).

If you're getting a sense of *déjà vu*, don't be surprised. Yes indeed, both Z and $\overline{U}$ for this system are exactly the same as for a two-state paramagnet, if you replace the magnetic interaction energy μB with the neighbor-neighbor interaction energy ϵ. Here, however, the dipoles like to line up with each other, instead of with an external field.

Notice that, while this system *does* become more ordered (less random) as its temperature decreases, the order sets in gradually. The behavior of $\overline{U}$ as a function of T is perfectly smooth, with no abrupt transition at a nonzero critical temperature. Apparently, the one-dimensional Ising model does *not* behave like a real three-dimensional ferromagnet in this crucial respect. Its tendency to magnetize is not great enough, because each dipole has only two nearest neighbors.

So our next step should be to consider Ising models in higher dimensions. Unfortunately, though, such models are *much* harder to solve. The two-dimensional Ising model on a square lattice was first solved in the 1940s by Lars Onsager. Onsager evaluated the exact partition function as $N \to \infty$ in closed form, and found that this model *does* have a critical temperature, just like a real ferromagnet. Because Onsager's solution is extremely difficult mathematically, I will not attempt to present it in this book. In any case, nobody has ever found an exact solution to the *three*-dimensional Ising model. The most fruitful approach from here, therefore, is to give up on exact solutions and rely instead on approximations.

Problem 8.18. Starting from the partition function, calculate the average energy of the one-dimensional Ising model, to verify equation 8.44. Sketch the average energy as a function of temperature.

The Mean Field Approximation

Next I'd like to present a very crude approximation, which can be used to "solve" the Ising model in any dimensionality. This approximation won't be very accurate, but it does give some qualitative insight into what's happening and why the dimensionality matters.

Let's concentrate on just a single dipole, somewhere in the middle of the lattice. I'll label this dipole i, so its alignment is s_i which can be -1 or 1. Let n be the number of nearest neighbors that this dipole has:

$$n = \begin{cases} 2 & \text{in one dimension;} \\ 4 & \text{in two dimensions (square lattice);} \\ 6 & \text{in three dimensions (simple cubic lattice);} \\ 8 & \text{in three dimensions (body-centered cubic lattice);} \\ 12 & \text{in three dimensions (face-centered cubic lattice).} \end{cases} \tag{8.45}$$

Imagine that the alignments of these neighboring dipoles are temporarily frozen, but that our dipole i is free to point up or down. If it points up, then the interaction energy between this dipole and its neighbors is

$$E_\uparrow = -\epsilon \sum_{\text{neighbors}} s_{\text{neighbor}} = -\epsilon\, n \bar{s}, \tag{8.46}$$

where $\bar{s}$ is the *average* alignment of the neighbors (see Figure 8.6). Similarly, if

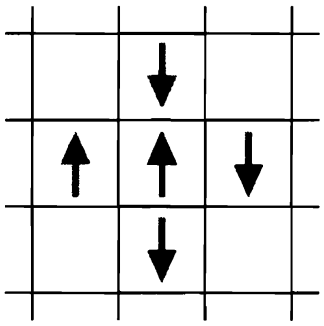

Figure 8.6. The four neighbors of this particular dipole have an average s value of $(+1-3)/4 = -1/2$. If the central dipole points up, the energy due to its interactions with its neighbors is $+2\epsilon$, while if it points down, the energy is -2ϵ.

dipole i points down, then the interaction energy is

$$E_\downarrow = +\epsilon n \bar{s}. \tag{8.47}$$

The partition function for just this dipole is therefore

$$Z_i = e^{\beta\epsilon n\bar{s}} + e^{-\beta\epsilon n\bar{s}} = 2\cosh(\beta\epsilon n\bar{s}), \tag{8.48}$$

and the average expected value of its spin alignment is

$$\bar{s}_i = \frac{1}{Z_i}\left[(1)e^{\beta\epsilon n\bar{s}} + (-1)e^{-\beta\epsilon n\bar{s}}\right] = \frac{2\sinh(\beta\epsilon n\bar{s})}{2\cosh(\beta\epsilon n\bar{s})} = \tanh(\beta\epsilon n\bar{s}). \tag{8.49}$$

Now look at both sides of this equation. On the left is $\bar{s}_i$, the thermal average value of the alignment of any typical dipole (except those on the edge of the lattice, which we'll neglect). On the right is $\bar{s}$, the average of the actual instantaneous alignments of this dipole's n neighbors. The idea of the **mean field approximation** is to assume (or pretend) that these two quantities are the same: $\bar{s}_i = \bar{s}$. In other words, we assume that at every moment, the alignments of all the dipoles are such that every neighborhood is "typical"—there are no fluctuations that cause the magnetization in any neighborhood to be more or less than the expected thermal average. (This approximation is similar to the one I used to derive the van der Waals equation in Section 5.3. There it was the density, rather than the spin alignment, whose average value was not allowed to vary from place to place within the system.)

In the mean field approximation, then, we have the relation

$$\bar{s} = \tanh(\beta\epsilon n\bar{s}), \tag{8.50}$$

where $\bar{s}$ is now the average dipole alignment over the entire system. This is a transcendental equation, so we can't just solve for $\bar{s}$ in terms of $\beta\epsilon n$. The best approach is to plot both sides of the equation and look for a graphical solution (see Figure 8.7). Notice that the larger the value of $\beta\epsilon n$, the steeper the slope of the hyperbolic tangent function near $\bar{s} = 0$. This means that our equation can have either one solution or three, depending on the value of $\beta\epsilon n$.

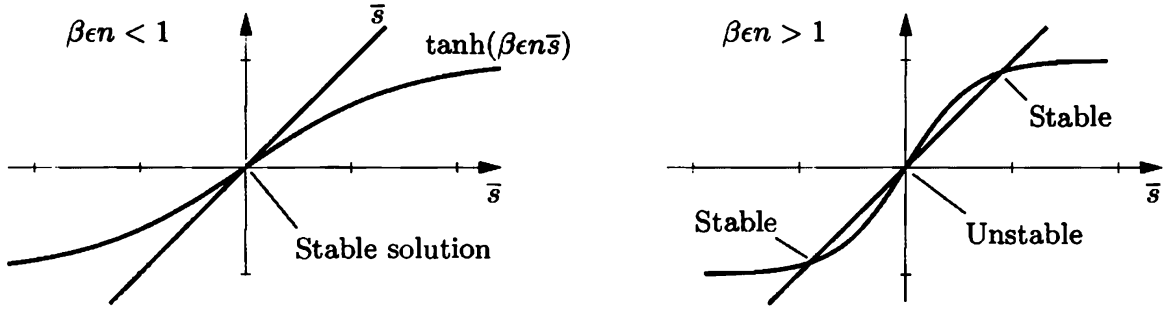

Figure 8.7. Graphical solution of equation 8.50. The slope of the tanh function at the origin is $\beta\epsilon n$. When this quantity is less than 1, there is only one solution, at $\bar{s} = 0$; when this quantity is greater than 1, the $\bar{s} = 0$ solution is unstable but there are also two nontrivial stable solutions.

When $\beta\epsilon n < 1$, that is, when $kT > n\epsilon$, the only solution is at $\bar{s} = 0$; there is no net magnetization. If a thermal fluctuation were to momentarily increase the value of $\bar{s}$, then the hyperbolic tangent function, which dictates what $\bar{s}$ *should* be, would be less than the current value of $\bar{s}$, so $\bar{s}$ would tend to decrease back to zero. The solution $\bar{s} = 0$ is stable.

When $\beta\epsilon n > 1$, that is, when $kT < n\epsilon$, we still have a solution at $\bar{s} = 0$ and we also have two more solutions, at positive and negative values of $\bar{s}$. But the solution at $\bar{s} = 0$ is unstable: A small positive fluctuation of $\bar{s}$ would cause the hyperbolic tangent function to exceed the current value of $\bar{s}$, driving $\bar{s}$ to even higher values. The stable solutions are the other two, which are symmetrically located because the system has no inherent tendency toward positive or negative magnetization. Thus, the system will acquire a net nonzero magnetization, which is equally likely to be positive or negative. When a system has a built-in symmetry such as this, yet must choose one state or another at low temperatures, we say that the symmetry is **spontaneously broken**.

The critical temperature T_c below which the system becomes magnetized is

$$kT_c = n\epsilon, \tag{8.51}$$

proportional to both the neighbor-neighbor interaction energy and to the number of neighbors. This result is no surprise: The more neighbors each dipole has, the greater the tendency of the whole system to magnetize. Notice, though, that even a *one*-dimensional Ising model should magnetize below a temperature of $2\epsilon/k$, according to this analysis. Yet we already saw from the exact solution that there is no abrupt transition in the behavior of a one-dimensional Ising model; it magnetizes only as the temperature goes to zero. Apparently, the mean field approximation is no good at all in one dimension.* Fortunately, the accuracy improves as the dimensionality increases.

Problem 8.19. The critical temperature of iron is 1043 K. Use this value to make a rough estimate of the dipole-dipole interaction energy ϵ, in electron-volts.

Problem 8.20. Use a computer to plot $\bar{s}$ as a function of kT/ϵ, as predicted by mean field theory, for a two-dimensional Ising model (with a square lattice).

Problem 8.21. At $T = 0$, equation 8.50 says that $\bar{s} = 1$. Work out the first temperature-dependent correction to this value, in the limit $\beta\epsilon n \gg 1$. Compare to the low-temperature behavior of a real ferromagnet, treated in Problem 7.64.

Problem 8.22. Consider an Ising model in the presence of an external magnetic field B, which gives each dipole an additional energy of $-\mu_B B$ if it points up and $+\mu_B B$ if it points down (where μ_B is the dipole's magnetic moment). Analyze this system using the mean field approximation to find the analogue of equation 8.50. Study the solutions of the equation graphically, and discuss the magnetization of this system as a function of both the external field strength and the temperature. Sketch the region in the T-B plane for which the equation has three solutions.

*There do exist more complicated versions of the mean field approximation that lack this fatal flaw, predicting correctly that the one-dimensional Ising model magnetizes only at $T = 0$. See, for example, Pathria (1996).

Problem 8.23. The Ising model can be used to simulate other systems besides ferromagnets; examples include antiferromagnets, binary alloys, and even fluids. The Ising model of a fluid is called a **lattice gas**. We imagine that space is divided into a lattice of sites, each of which can be either occupied by a gas molecule or unoccupied. The system has no kinetic energy, and the only potential energy comes from interactions of molecules on adjacent sites. Specifically, there is a contribution of $-u_0$ to the energy for each pair of neighboring sites that are both occupied.

(a) Write down a formula for the *grand* partition function for this system, as a function of u_0, T, and μ.

(b) Rearrange your formula to show that it is identical, up to a multiplicative factor that does not depend on the state of the system, to the *ordinary* partition function for an Ising ferromagnet in the presence of an external magnetic field B, provided that you make the replacements $u_0 \to 4\epsilon$ and $\mu \to 2\mu_B B - 8\epsilon$. (Note that μ is the chemical potential of the gas while μ_B is the magnetic moment of a dipole in the magnet.)

(c) Discuss the implications. Which states of the magnet correspond to low-density states of the lattice gas? Which states of the magnet correspond to high-density states in which the gas has condensed into a liquid? What shape does this model predict for the liquid-gas phase boundary in the P-T plane?

Problem 8.24. In this problem you will use the mean field approximation to analyze the behavior of the Ising model near the critical point.

(a) Prove that, when $x \ll 1$, $\tanh x \approx x - \frac{1}{3}x^3$.

(b) Use the result of part (a) to find an expression for the magnetization of the Ising model, in the mean field approximation, when T is very close to the critical temperature. You should find $M \propto (T_c - T)^\beta$, where β (not to be confused with $1/kT$) is a **critical exponent**, analogous to the β defined for a fluid in Problem 5.55. Onsager's exact solution shows that $\beta = 1/8$ in two dimensions, while experiments and more sophisticated approximations show that $\beta \approx 1/3$ in three dimensions. The mean field approximation, however, predicts a larger value.

(c) The magnetic susceptibility χ is defined as $\chi \equiv (\partial M/\partial B)_T$. The behavior of this quantity near the critical point is conventionally written as $\chi \propto (T - T_c)^{-\gamma}$, where γ is another critical exponent. Find the value of γ in the mean field approximation, and show that it does not depend on whether T is slightly above or slightly below T_c. (The exact value of γ in two dimensions turns out to be $7/4$, while in three dimensions $\gamma \approx 1.24$.)

Monte Carlo Simulation

Consider a medium-sized, two-dimensional Ising model on a square lattice, with 100 or so elementary dipoles (as shown in Figure 8.3). Although even the fastest computer could never compute the probabilities of *all* the possible states of this system, maybe it isn't necessary to consider all of them—perhaps a random sampling of only a million or so states would be enough. This is the idea of **Monte Carlo summation** (or integration), a technique named after the famous European gambling center. The procedure is to generate a random sampling of as many states as

possible, compute the Boltzmann factors for these states, and then use this random sample to compute the average energy, magnetization, and other thermodynamic quantities.

Unfortunately, the procedure just outlined does not work well for the Ising model. Even if we consider as many as one billion states, this is only a *tiny* fraction—about one in 10^{21}—of all the states for a modest 10×10 lattice. And at low temperatures, when the system wants to magnetize, the *important* states (with nearly all of the dipoles pointing in the same direction) constitute such a small fraction of the total that we are likely to miss them entirely. Sampling the states purely at random just isn't efficient enough; for this reason it's sometimes called the *naive* Monte Carlo method.

A better idea is to use the Boltzmann factors themselves as a guide during the random generation of a subset of states to sample. A specific algorithm that does this is as follows: Start with any state whatsoever. Then choose a dipole at random and consider the possibility of flipping it. Compute the energy difference, ΔU, that would result from the flip. If $\Delta U \leq 0$, so the system's energy would decrease or remain unchanged, go ahead and flip this dipole to generate the next system state. If $\Delta U > 0$, so the system's energy would increase, decide at random whether to flip the dipole, with the probability of the flip being $e^{-\Delta U/kT}$. If the dipole does *not* get flipped, then the new system state will be the same as the previous one. Either way, continue by choosing another dipole at random and repeat the process, over and over again, until every dipole has had many chances to be flipped. This algorithm is called the **Metropolis algorithm**, after Nicholas Metropolis, the first author of a 1953 article* that presented a calculation of this type. This technique is also called Monte Carlo summation with **importance sampling**.

The Metropolis algorithm generates a subset of system states in which low-energy states occur more frequently than high-energy states. To see in more detail why the algorithm works, consider just two states, 1 and 2, which differ only by the flipping of a single dipole. Let U_1 and U_2 be the energies of these states, and let us number the states so that $U_1 \leq U_2$. If the system is initially in state 2, then the probability of making a transition to state 1 is $1/N$, simply the probability that the correct dipole will be chosen at random among all the others. If the system is initially in state 1, then the probability of making a transition to state 2 is $(1/N)e^{-(U_2-U_1)/kT}$, according to the Metropolis algorithm. The ratio of these two transition probabilities is therefore

$$\frac{\mathcal{P}(1 \to 2)}{\mathcal{P}(2 \to 1)} = \frac{(1/N)e^{-(U_2-U_1)/kT}}{(1/N)} = \frac{e^{-U_2/kT}}{e^{-U_1/kT}}, \tag{8.52}$$

simply the ratio of the Boltzmann factors of the two states. If these were the *only*

*N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of State Calculations for Fast Computing Machines," *Journal of Chemical Physics* **21**, 1087–1092 (1953). In this article the authors use their algorithm to calculate the pressure of a two-dimensional gas of 224 hard disks. This rather modest calculation required several days of computing time on what was then a state-of-the-art computer.

two states available to the system, then the frequencies with which they occur would be in exactly this ratio, as Boltzmann statistics demands.*

Next consider two other states, 3 and 4, that differ from 1 and 2 by the flipping of some other dipole. The system can now go between 1 and 2 through the indirect process $1 \leftrightarrow 3 \leftrightarrow 4 \leftrightarrow 2$, whose forward and backward rates have the ratio

$$\frac{\mathcal{P}(1 \to 3 \to 4 \to 2)}{\mathcal{P}(2 \to 4 \to 3 \to 1)} = \frac{e^{-U_3/kT}}{e^{-U_1/kT}} \frac{e^{-U_4/kT}}{e^{-U_3/kT}} \frac{e^{-U_2/kT}}{e^{-U_4/kT}} = \frac{e^{-U_2/kT}}{e^{-U_1/kT}}, \tag{8.53}$$

again as demanded by Boltzmann statistics. The same conclusion applies to transitions involving any number of steps, and to transitions between states that differ by the flipping of more than one dipole. Thus, the Metropolis algorithm does indeed generate states with the correct Boltzmann probabilities.

Strictly speaking, though, this conclusion applies only after the algorithm has been running infinitely long, so that every state has been generated many times. *We* want to run the algorithm for a relatively short time, so that most states are never generated at all! Under these circumstances we have no guarantee that the subset of states actually generated will accurately represent the full collection of all system states. In fact, it's hard to even define what is *meant* by an "accurate" representation. In the case of the Ising model, our main concerns are that the randomly generated states give an accurate picture of the expected energy and magnetization of the system. The most noticeable exception in practice will be that at low temperatures, the Metropolis algorithm will rapidly push the system into a "metastable" state in which nearly all of the dipoles are parallel to their neighbors. Although such a state *is* quite probable according to Boltzmann statistics, it may take a very long time for the algorithm to generate other probable states that differ significantly, such as a state in which every dipole is flipped. (In this way the Metropolis algorithm is analogous to what happens in the real world, where a large system never has time to explore all possible microstates, and the relaxation time for achieving true thermodynamic equilibrium can sometimes be very long.)

With this limitation in mind, let's now go on and implement the Metropolis algorithm. The algorithm can be programmed in almost any traditional computer language, and in many nontraditional languages as well. Rather than singling out one particular language, let me instead present the algorithm in "pseudocode," which you can translate into the language of your choice. A pseudocode program for a basic two-dimensional Ising simulation is shown in Figure 8.8. This program produces only graphical output, showing the lattice as an array of colored squares—one color for dipoles pointing up, another color for dipoles pointing down. Each time a dipole is flipped the color of a square changes, so you can see exactly what sequence of states is being generated.

The program uses a two-dimensional array called `s(i,j)` to store the values of the spin orientations; the indices `i` and `j` each go from 1 to the value of `size`, which can be changed to simulate lattices of different sizes. The temperature `T`,

*When the transition rates between two states have the correct ratio, we say that the transitions are in **detailed balance**.

```
program ising                          Monte Carlo simulation of a 2D Ising
                                       model using the Metropolis algorithm

size = 10                              Width of square lattice
T = 2.5                                Temperature in units of ϵ/k
initialize
for iteration = 1 to 100*size^2 do     Main iteration loop
  i = int(rand*size+1)                 Choose a random row number
  j = int(rand*size+1)                 and a random column number
  deltaU(i,j,Ediff)                    Compute ΔU of hypothetical flip
  if Ediff <= 0 then                   If flipping reduces the energy ...
    s(i,j) = -s(i,j)                   then flip it!
    colorsquare(i,j)
  else
    if rand < exp(-Ediff/T) then       otherwise the Boltzmann factor
      s(i,j) = -s(i,j)                 gives the probability of flipping
      colorsquare(i,j)
    end if
  end if
next iteration                         Now go back and start over ...
end program

subroutine deltaU(i,j,Ediff)           Compute ΔU of flipping a dipole
                                       (note periodic boundary conditions)
  if i = 1 then top = s(size,j) else top = s(i-1,j)
  if i = size then bottom = s(1,j) else bottom = s(i+1,j)
  if j = 1 then left = s(i,size) else left = s(i,j-1)
  if j = size then right = s(i,1) else right = s(i,j+1)
  Ediff = 2*s(i,j)*(top+bottom+left+right)
end subroutine

subroutine initialize                  Initialize to a random array
  for i = 1 to size
    for j = 1 to size
      if rand < .5 then s(i,j) = 1 else s(i,j) = -1
      colorsquare(i,j)
    next j
  next i
end subroutine

subroutine colorsquare(i,j)            Color a square according to s value
                                       (implementation depends on system)
```

Figure 8.8. A pseudocode program to simulate a two-dimensional Ising model, using the Metropolis algorithm.

measured in units of ϵ/k, can also be changed for different runs. After setting these two constants, the program calls the subroutine `initialize` to assign the initial value of each `s` randomly.*

The heart of the program is the "main iteration loop," which executes the Metropolis algorithm 100 times per dipole so that each dipole will have many chances to be flipped. The value 100 can be changed as appropriate. (Note that * represents multiplication, while ^ represents exponentiation.) Within the loop, we first choose a dipole at random; the function `rand` is assumed to return a random real number between 0 and 1, while `int()` returns the largest integer less than or equal to its argument. The subroutine `deltaU`, defined later in the program, computes the energy change upon hypothetically flipping the chosen dipole; this energy change (in units of ϵ) is returned as `Ediff`. If `Ediff` is negative or zero, we flip the dipole, while if `Ediff` is positive, we use it to compute a Boltzmann factor and compare this to a random number to decide whether to flip the dipole. If the dipole gets flipped, we call the subroutine `colorsquare` to change the color of the corresponding square on the screen.

The subroutine `deltaU` requires further explanation. There is always a problem, when a simulation uses a relatively small lattice, in dealing with "edge effects." In the Ising model, dipoles on the edge of the lattice are less constrained to align with their neighbors than are dipoles elsewhere. If we're modeling a very small system whose size is the same as that of our simulated lattice, then we should treat the edges as edges, with fewer neighbors per dipole. But if we're really interested in the behavior of much larger systems, then we should try to minimize edge effects. One way to do this is to make the lattice "wrap around," treating the right edge as if it were immediately left of the left edge and the bottom edge as if it were immediately above the top edge. Physically this would be like putting the array of dipoles on the surface of a torus. Another interpretation of this wrapping is to imagine that the lattice is flat and infinite in all directions, but that its state is always perfectly periodic, so that moving up, down, left, or right by a certain amount (the value of `size`) always takes you to an equivalent place where the dipoles have exactly the same alignments at all times. Based on this latter interpretation, we say that we are using **periodic boundary conditions**. Back to the subroutine `deltaU`, notice that it correctly identifies all four nearest neighbors, whether or not the chosen dipole is on an edge. The change in energy upon flipping is then twice the product of `s(i,j)` with `s` of the neighbor, summed over the four neighbors.

To convert my pseudocode into a real program that runs on a real computer, you first need to pick a computer system and a programming language. The syntax for arithmetic operations, variable assignments, if-then constructions, and for-next loops will vary from language to language, but almost any common programming language should provide easy ways to do these things. Some languages require that

*In principle, the initial state can by anything. In practice, the choice of initial state can be important if you don't want to wait forever for the system to equilibrate to a "typical" state. A random initial state works well at high temperatures; a completely magnetized initial state would work better at low temperatures.

variables be declared and given a type (such as integer or real) at the beginning of the program. Variables that are accessed both in the main program and in subroutines may require special treatment. The least standardized element of all is the handling of graphics; the contents of the subroutine `colorsquare` will vary wildly from system to system. Nevertheless, I hope that you will have little trouble implementing this program on your favorite computer and getting it to run.

Running the `ising` program is great fun: You get to watch the squares constantly changing colors as the system tries to find states with relatively large Boltzmann factors. It is tempting, in fact, to imagine that you are watching a simulation of what really happens in a magnet, as the dipoles change their alignments back and forth with the passage of time. Because of this similarity, a Monte Carlo program using importance sampling is usually called a Monte Carlo **simulation**. But please remember that we have made no attempt to simulate the real time-dependent behavior of a magnet. Instead we have implemented a "pseudodynamics," which flips only one dipole at a time and otherwise ignores the true time-dependent dynamics of the system. The only realistic property of our pseudodynamics is that it generates states with probabilities proportional to their Boltzmann factors, just as the real dynamics of a magnet presumably does.

Figure 8.9 shows some graphical output from the `ising` program for a 20×20 lattice. The first image shows a random initial state generated by the program, while the remaining images each show the final state at the end of a run of 40,000 iterations (100 per dipole), for various temperatures. Although these snapshots are no substitute for watching the program in action, they do show what a typical state at each temperature looks like. At $\mathtt{T} = 10$ the final state is still almost random, with only a slight tendency for dipoles to align with their neighbors. At successively lower temperatures the dipoles tend to form larger and larger clusters* of positive and negative magnetization until, at $\mathtt{T} = 2.5$, the clusters are about as large as the lattice itself. At $\mathtt{T} = 2$ a single cluster has taken over the whole lattice, and we would say that the system is "magnetized." Small clusters of dipoles will still occasionally flip, but they don't last long; we would have to wait a very long time for the whole lattice to flip to a (just as probable) state of opposite magnetization. The $\mathtt{T} = 1.5$ run happens to have settled into the opposite magnetization, and at this temperature fluctuations of individual dipoles are becoming uncommon. At $\mathtt{T} = 1$ we might expect the system to magnetize completely and stay that way, and indeed, sometimes it does. About half the time, however, it instead becomes stuck in a metastable state with two domains, one positive and the other negative, as shown in the figure.

Based on these results, we can conclude that this system has a critical temperature somewhere between 2.0 and 2.5, in units of ϵ/k. Recall that the mean field approximation predicts a critical temperature of $4\epsilon/k$—not bad qualitatively, though off by nearly a factor of 2. But a 20×20 lattice is really quite small; what

*I'm making no attempt here to precisely define a "cluster"—just look at the pictures and use your intuition. A careful definition of the "size" of a cluster is given in Problem 8.29.

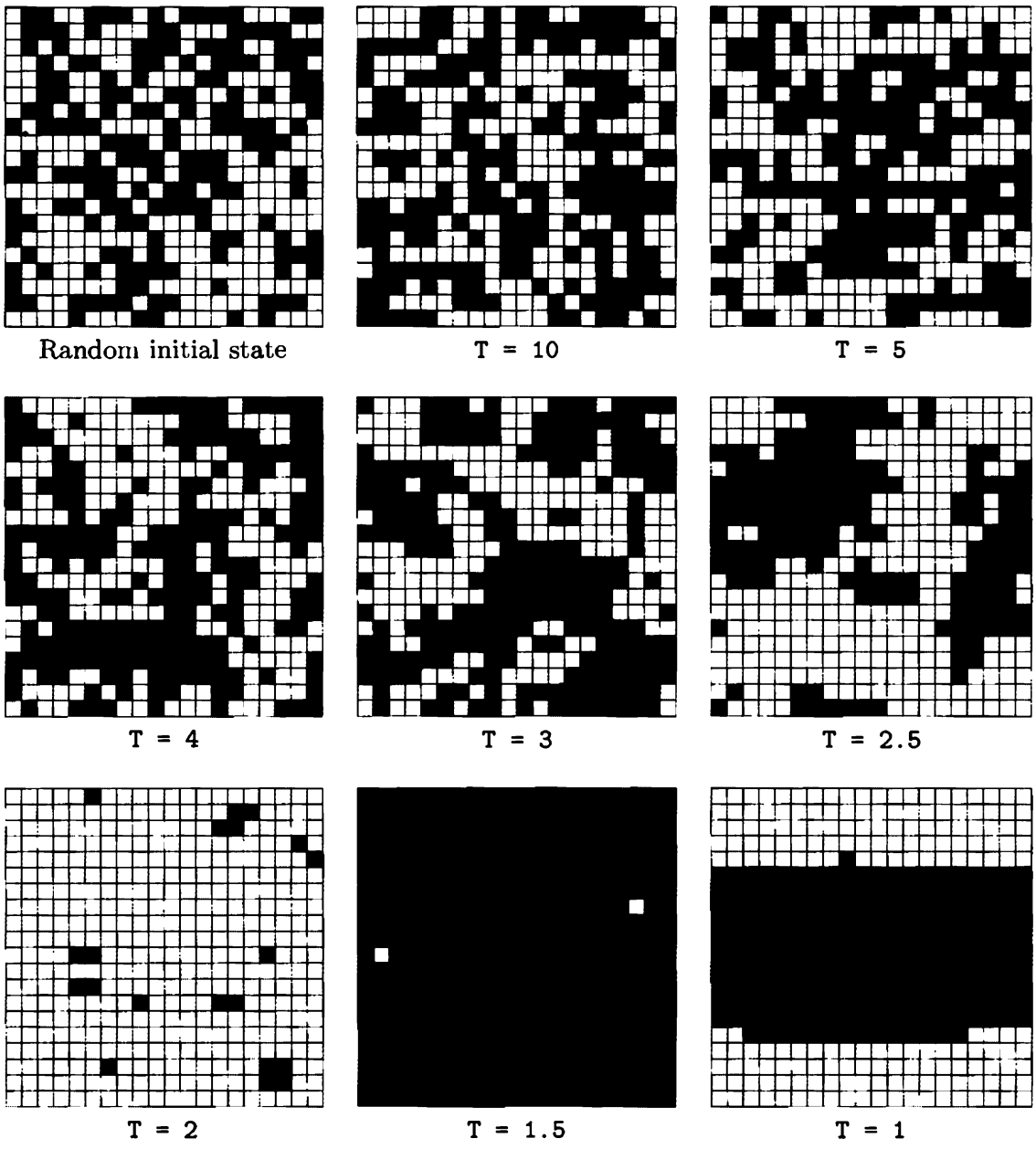

Figure 8.9. Graphical output from eight runs of the `ising` program, at successively lower temperatures. Each black square represents an "up" dipole and each white square represents a "down" dipole. The variable T is the temperature in units of ϵ/k.

happens in larger, more realistic simulations?

The answer isn't hard to guess. As long as the temperature is sufficiently high, so that the size of a typical cluster is much smaller than the size of the lattice, the behavior of the system is pretty much independent of the lattice size. But a larger lattice allows for larger clusters, so near the critical temperature we should use as large a lattice as possible. With sufficiently long runs with large lattices one can show that the size of the largest clusters approaches infinity at a temperature of $2.27\epsilon/k$ (see Figure 8.10). This, then, is the *true* critical temperature in the thermodynamic limit. And indeed, this result agrees with Onsager's exact solution

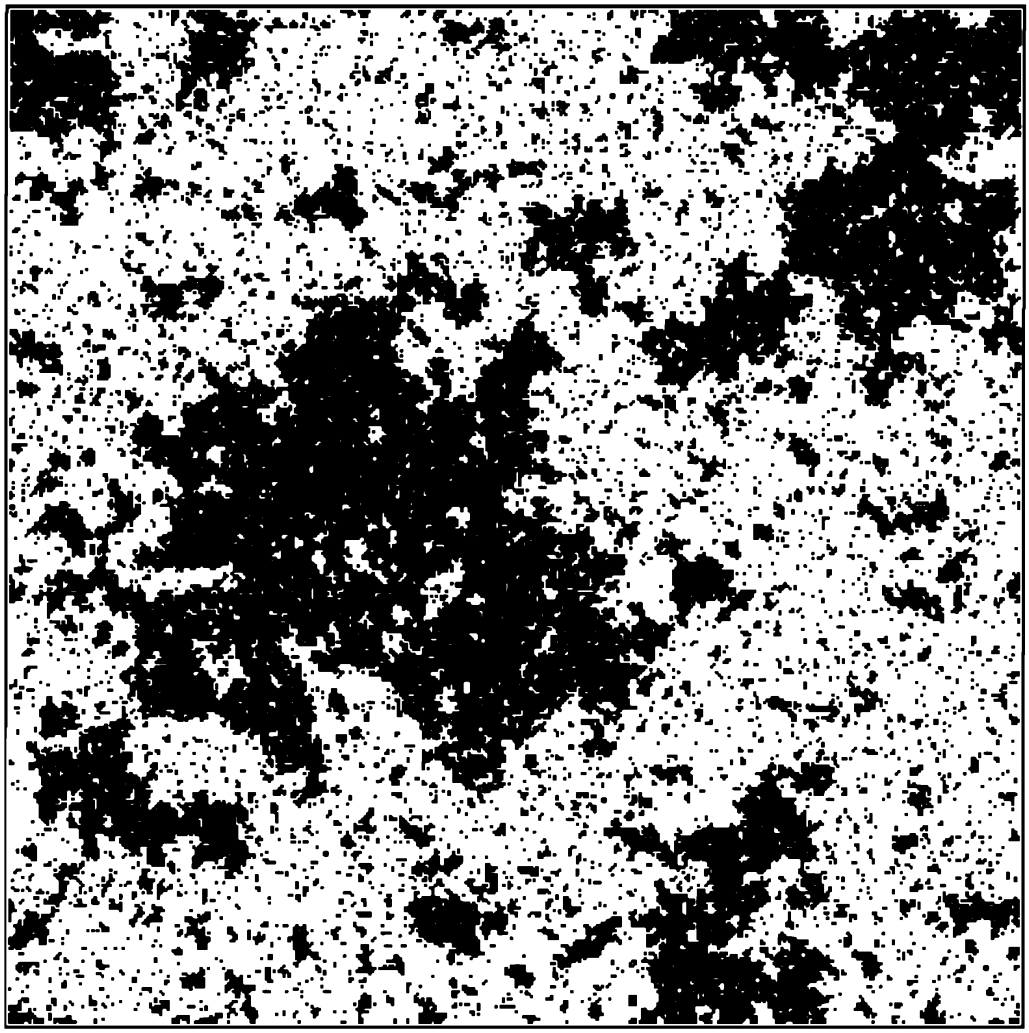

Figure 8.10. A typical state generated by the `ising` program after a few billion iterations on a 400×400 lattice at $T = 2.27$ (the critical temperature). Notice that there are clusters of all possible sizes, from individual dipoles up to the size of the lattice itself.

of the two-dimensional Ising model.

Similar simulations have been performed for the *three*-dimensional Ising model, although this requires much more computer time and the results are harder to display. For a simple cubic lattice one finds a critical temperature of approximately $4.5\epsilon/k$, again somewhat less than the prediction of the mean field approximation. The Monte Carlo method can also be applied to more complicated models of ferromagnets and to a huge variety of other systems including fluids, alloys, interfaces, nuclei, and subnuclear particles.

Problem 8.25. In Problem 8.15 you manually computed the energy of a particular state of a 4×4 square lattice. Repeat that computation, but this time apply periodic boundary conditions.

Problem 8.26. Implement the `ising` program on your favorite computer, using your favorite programming language. Run it for various lattice sizes and temperatures and observe the results. In particular:

(a) Run the program with a 20×20 lattice at T = 10, 5, 4, 3, and 2.5, for at least 100 iterations per dipole per run. At each temperature make a rough estimate of the size of the largest clusters.

(b) Repeat part (a) for a 40×40 lattice. Are the cluster sizes any different? Explain.

(c) Run the program with a 20×20 lattice at T = 2, 1.5, and 1. Estimate the average magnetization (as a percentage of total saturation) at each of these temperatures. Disregard runs in which the system gets stuck in a metastable state with two domains.

(d) Run the program with a 10×10 lattice at T = 2.5. Watch it run for 100,000 iterations or so. Describe and explain the behavior.

(e) Use successively larger lattices to estimate the typical cluster size at temperatures from 2.5 down to 2.27 (the critical temperature). The closer you are to the critical temperature, the larger a lattice you'll need and the longer the program will have to run. Quit when you realize that there are better ways to spend your time. Is it plausible that the cluster size goes to infinity as the temperature approaches the critical temperature?

Problem 8.27. Modify the `ising` program to compute the average energy of the system over all iterations. To do this, first add code to the `initialize` subroutine to compute the initial energy of the lattice; then, whenever a dipole is flipped, change the energy variable by the appropriate amount. When computing the average energy, be sure to average over *all* iterations, not just those iterations in which a dipole is actually flipped (why?). Run the program for a 5×5 lattice for T values from 4 down to 1 in reasonably small intervals, then plot the average energy as a function of T. Also plot the heat capacity. Use at least 1000 iterations per dipole for each run, preferably more. If your computer is fast enough, repeat for a 10×10 lattice and for a 20×20 lattice. Discuss the results. (Hint: Rather than starting over at each temperature with a random initial state, you can save time by starting with the final state generated at the previous, nearby temperature. For the larger lattices you may wish to save time by considering only a smaller temperature interval, perhaps from 3 down to 1.5.)

Problem 8.28. Modify the `ising` program to compute the total magnetization (that is, the sum of all the s values) for each iteration, and to tally how often each possible magnetization value occurs during a run, plotting the results as a histogram. Run the program for a 5×5 lattice at a variety of temperatures, and discuss the results. Sketch a graph of the most likely magnetization value as a function of temperature. If your computer is fast enough, repeat for a 10×10 lattice.

Problem 8.29. To quantify the clustering of alignments within an Ising magnet, we define a quantity called the **correlation function**, $c(r)$. Take any two dipoles i and j, separated by a distance r, and compute the product of their states: $s_i s_j$. This product is 1 if the dipoles are parallel and -1 if the dipoles are antiparallel. Now average this quantity over *all* pairs that are separated by a fixed distance r, to obtain a measure of the tendency of dipoles to be "correlated" over this distance. Finally, to remove the effect of any overall magnetization of the system, subtract off the square of the average s. Written as an equation, then, the correlation function is

$$c(r) = \overline{s_i s_j} - \overline{s_i}^2,$$

where it is understood that the first term averages over all pairs at the fixed distance r. Technically, the averages should also be taken over all possible states of the system, but don't do this yet.

(a) Add a routine to the `ising` program to compute the correlation function for the current state of the lattice, averaging over all pairs separated either vertically or horizontally (but not diagonally) by r units of distance, where r varies from 1 to half the lattice size. Have the program execute this routine periodically and plot the results as a bar graph.

(b) Run this program at a variety of temperatures, above, below, and near the critical point. Use a lattice size of at least 20, preferably larger (especially near the critical point). Describe the behavior of the correlation function at each temperature.

(c) Now add code to compute the *average* correlation function over the duration of a run. (However, it's best to let the system "equilibrate" to a typical state before you begin accumulating averages.) The **correlation length** is defined as the distance over which the correlation function decreases by a factor of e. Estimate the correlation length at each temperature, and plot a graph of the correlation length vs. `T`.

Problem 8.30. Modifiy the `ising` program to simulate a *one*-dimensional Ising model.

(a) For a lattice size of 100, observe the sequence of states generated at various temperatures and discuss the results. According to the exact solution (for an infinite lattice), we expect this system to magnetize only as the temperature goes to zero; is the behavior of your program consistent with this prediction? How does the typical cluster size depend on temperature?

(b) Modify your program to compute the average energy as in Problem 8.27. Plot the energy and heat capacity vs. temperature and compare to the exact result for an infinite lattice.

(c) Modify your program to compute the magnetization as in Problem 8.28. Determine the most likely magnetization for various temperatures and sketch a graph of this quantity. Discuss.

Problem 8.31. Modify the `ising` program to simulate a *three*-dimensional Ising model with a simple cubic lattice. In whatever way you can, try to show that this system has a critical point at around `T` $= 4.5$.

Problem 8.32. Imagine taking a two-dimensional Ising lattice and dividing the sites into 3×3 "blocks," as shown in Figure 8.11. In a **block spin transformation**, we replace the nine dipoles in each block with a single dipole, whose state is determined by "majority rule": If more than half of the original dipoles point up, then the new dipole points up, while if more than half of the original dipoles point down, then the new dipole points down. By applying this transformation to the entire lattice, we reduce it to a new lattice whose width is 1/3 the original width. This transformation is one version of a **renormalization group transformation**, a powerful technique for studying the behavior of systems near their critical points.*

*For more about the renormalization group and its applications, see Kenneth G. Wilson, "Problems in Physics with Many Scales of Length," *Scientific American* **241**, 158–179 (August, 1979).

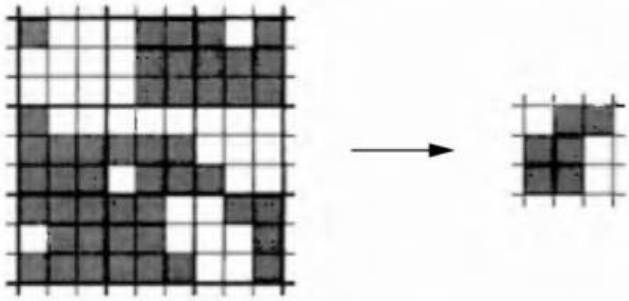

Figure 8.11. In a **block spin transformation**, we replace each block of nine dipoles with a single dipole whose orientation is determined by "majority rule."

(a) Add a routine to the `ising` program to apply a block spin transformation to the current state of the lattice, drawing the transformed lattice alongside the original. (Leave the original lattice unchanged.) Have the program execute this routine periodically, so you can observe the evolution of both lattices.

(b) Run your modified program with a 90×90 original lattice, at a variety of temperatures. After the system has equilibrated to a "typical" state at each temperature, compare the transformed lattice to a typical 30×30 piece of the original lattice. In general you should find that the transformed lattice resembles an original lattice at a *different* temperature. Let us call this temperature the "transformed temperature." When is the transformed temperature greater than the original temperature, and when is it less?

(c) Imagine starting with a very large lattice and applying many block spin transformations in succession, each time taking the system to a new effective temperature. Argue that, no matter what the original temperature, this procedure will eventually take you to one of three **fixed points**: zero, infinity, or the critical temperature. For what initial temperatures will you end up at each fixed point? [Comment: Think about the implications of the fact that the critical temperature is a fixed point of the block spin transformation. If averaging over the small-scale state of the system leaves the dynamics unchanged, then many aspects of the behavior of this system must be independent of any specific microscopic details. This implies that many different physical systems (magnets, fluids, and so on) should have essentially the same critical behavior. More specifically, the different systems will have the same "critical exponents," such as those defined in Problems 5.55 and 8.24. There are, however, two parameters that can still affect the critical behavior. One is the dimensionality of the space that the system is in (3 for most real-world systems); the other is the dimensionality of the "vector" that defines the magnetization (or the analogous "order parameter") of the system. For the Ising model, the magnetization is one-dimensional, always along a given axis; for a fluid, the order parameter is also a one-dimensional quantity, the difference in density between liquid and gas. Therefore the behavior of a fluid near its critical point should be the same as that of a three-dimensional Ising model.]