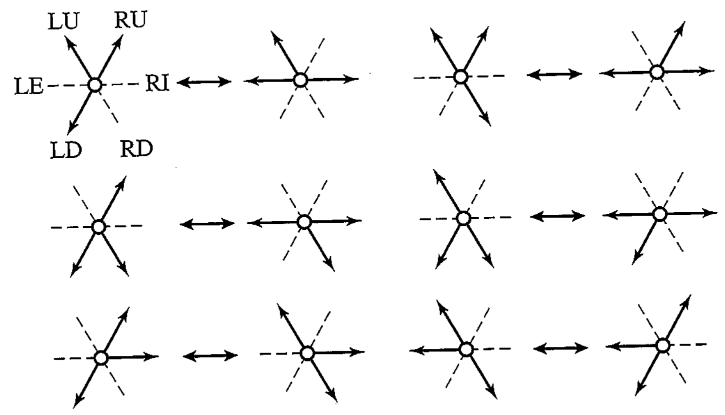

Figure 14.4 Examples of collision rules for three particles, with one particle unchanged and no stationary particles. Each direction or channel is represented by 32 bits, but we need only the first 8 bits. The various channels are summarized in Table 14.1.

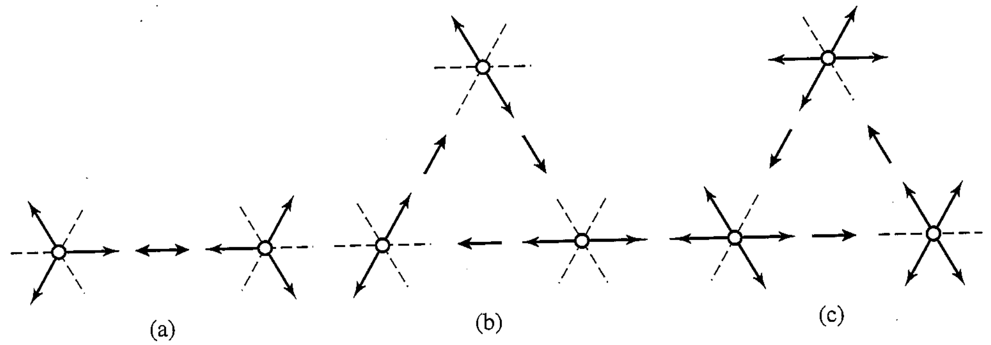

Figure 14.5 (a) Example of collision rule for three particles with zero net momentum. (b) Example of two-particle collision rule. (c) Example of four-particle collision rule. The rules for states that are not shown is that the velocities do not change after a collision. An open circle represents a lattice site and the absence of a stationary particle.

The rules for the collisions are given in the declaration of the class variables in class `LatticeGas`. Because `rule` is declared `static final`, we cannot normally overwrite its values. However, an exception is made for static initializers that are run when the class is first loaded. To construct the rules, we use the bitwise *or* operator `|` and use named constants for each of the possible states. As an example, the state corresponding to one particle moving to the right, one moving to the left and down, and one moving to the left and up is given by `LU` + `LD` + `RI`, which we write as `LU|LD|RI` or 00010101. The collision rule in Figure 14.5(a) is that this state transforms to one particle moving to the right and down, one moving left, and one moving to the right and up. Hence, this collision rule is given by `rule[LU|LD|RI]` = `RU|LE|RD`. The other rules are given in a similar way. Stationary particles can also be created or destroyed. For example, what are the states before and after the collision for `rule[LU|RI]` = `RU|S`?

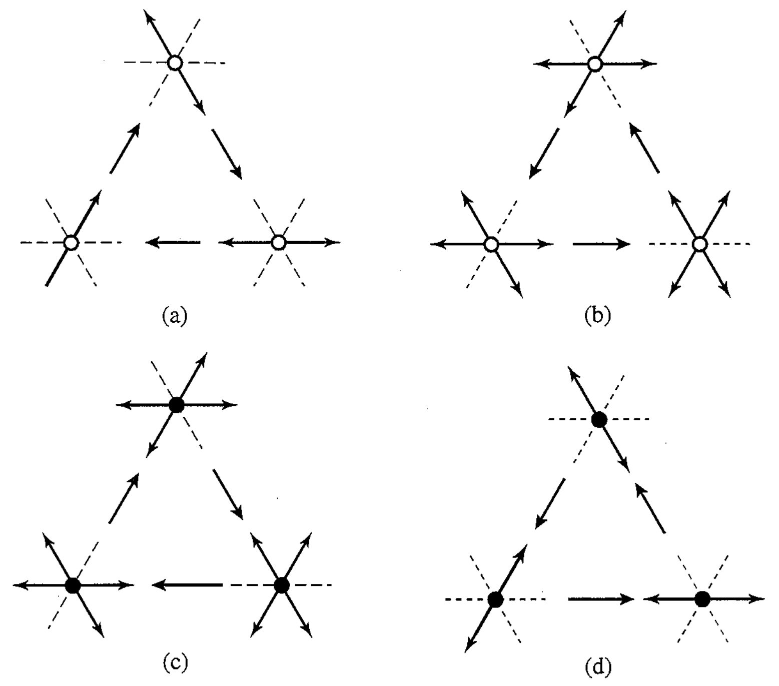

Figure 14.6 (a) and (c) and (b) and (d) are duals of each other. An open circle represents the absence of a stationary particle, and a filled circle represents the presence of a stationary particle. Note that the collision rule in (c) is similar to (b), and the collision rule in (d) is similar to (a), but in the opposite direction.

To every rule corresponds a dual rule that flips the bits corresponding to the presence and absence of a particle. This duality means that we need to only specify half of the rules. The dual rules can be constructed by flipping all bits of the input and output. Our convention is to list the rules starting without a stationary particle. Then the corresponding dual rules are those that start with a stationary particle. The dual rules are implemented by the statement

```
rule[i^(RU|LU|LE|LD|RD|RI|S)] = rule[i]^(RU|LU|LE|LD|RD|RI|S);
```

where ^ is the bitwise exclusive or operator, which equals 1 if both bits are different and is 0 otherwise. Two examples of dual rules are given in Figure 14.6.

The rules in Figures 14.5(b) and 14.5(c) cycle through the states in a particular direction. Although these rules are straightforward, they are not invariant under reflection. To help eliminate this bias, we cycle in the opposite direction when a stationary particle is present (see Figure 14.6).

We adopt the rule that when a particle moves onto a barrier site, we set the velocity **v** of this particle equal to −**v** (see Figure 14.7). Because of our ordering of the velocities, the rule for updating a barrier can be expressed compactly using bit manipulation. Reflection off a barrier is accomplished by shifting the higher-order bits to the right by three bits (>>>3) and shifting the lower-order bits to the left by three bits (<<3). Check the rules given in Listing 14.13. Other possibilities are to set the angle of incidence equal to the angle of reflection or to set the velocity to an arbitrary direction. The latter case would correspond to a collision off a rough surface.