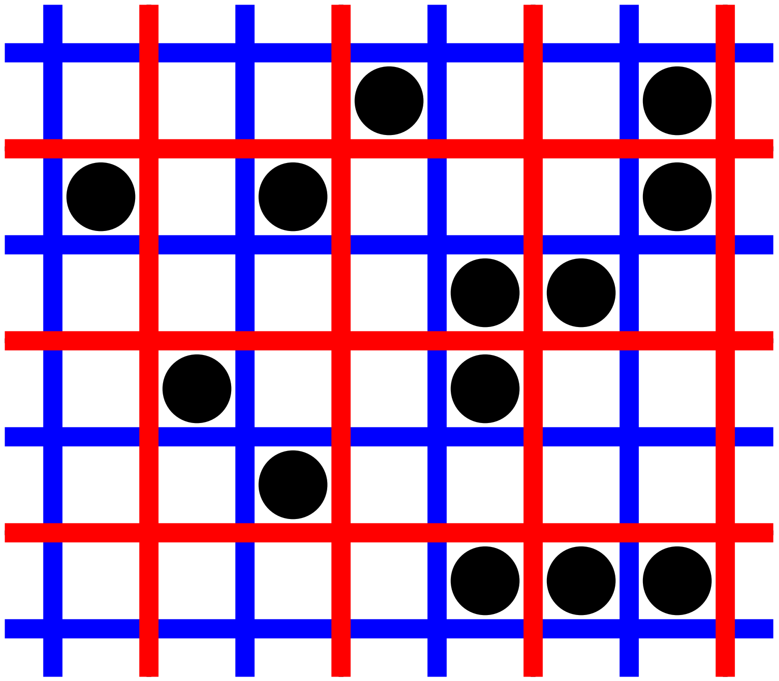
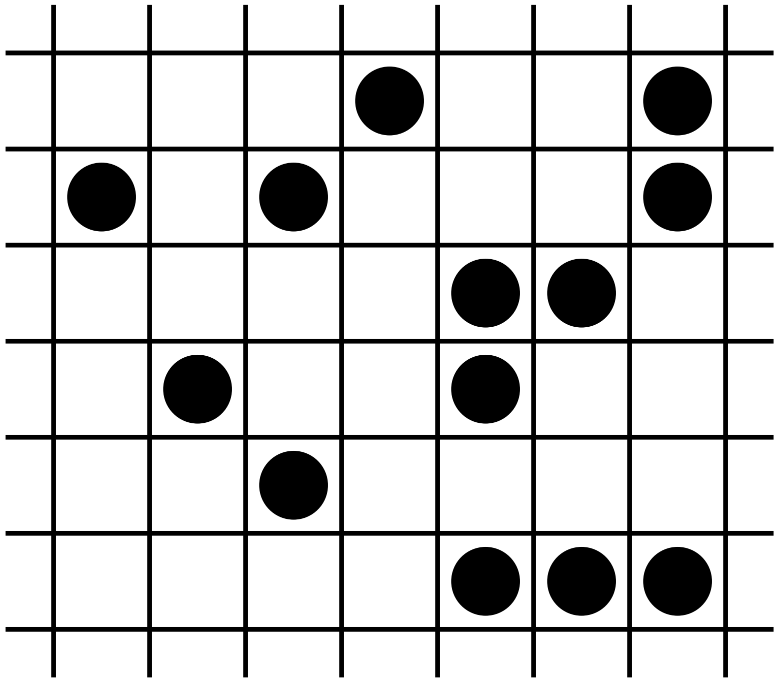
Another technique (similar to PCA) to guarantee reversibility: **Margolus neighborhood**.

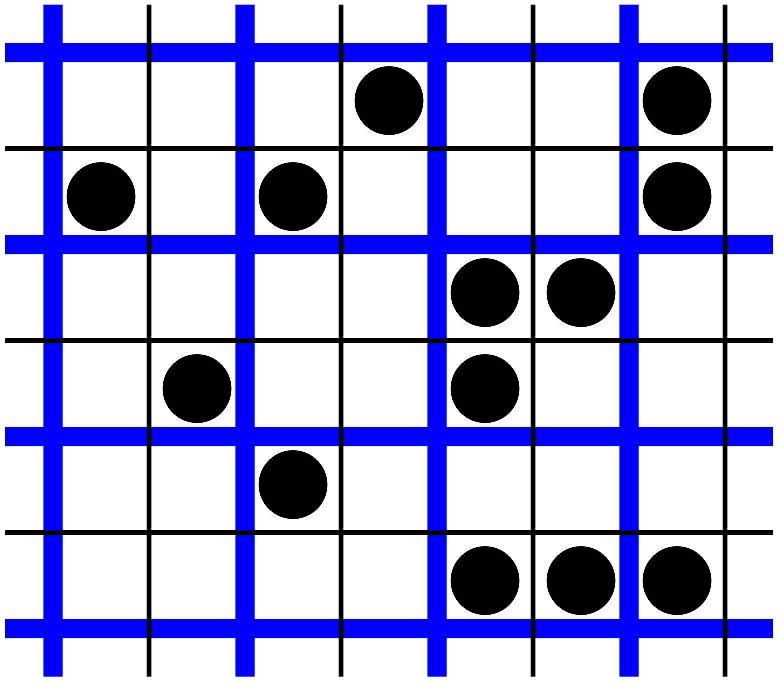
Two well known two-dimensional examples that use this neighborhood are the **Billiard Ball** CA by Margolus and a lattice gas CA called **HPP**.



In the Margolus neighborhood the updating is done in two steps:

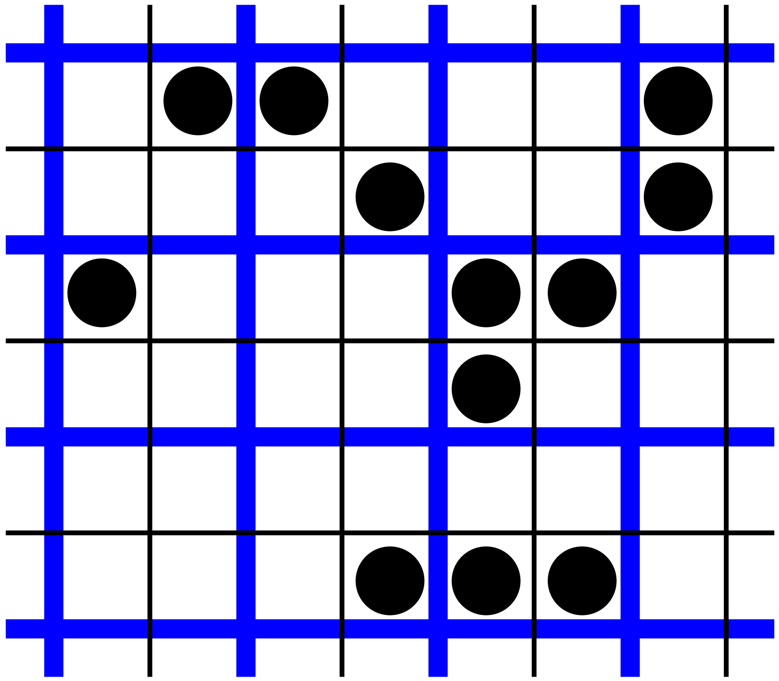


In the Margolus neighborhood the updating is done in two steps:



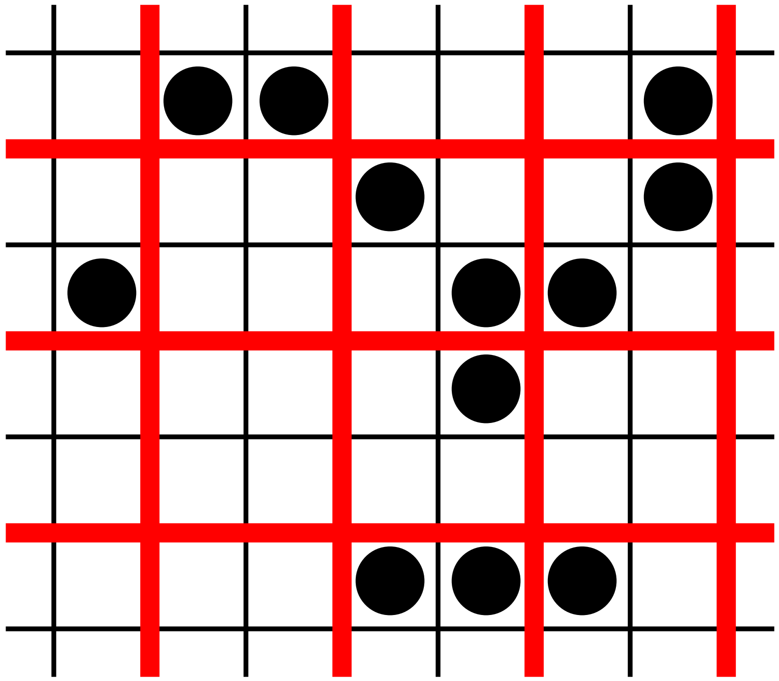
1. Partition the plane into 2×2 blocks and apply some permutation π_1 of S^4 inside each block.

In the Margolus neighborhood the updating is done in two steps:



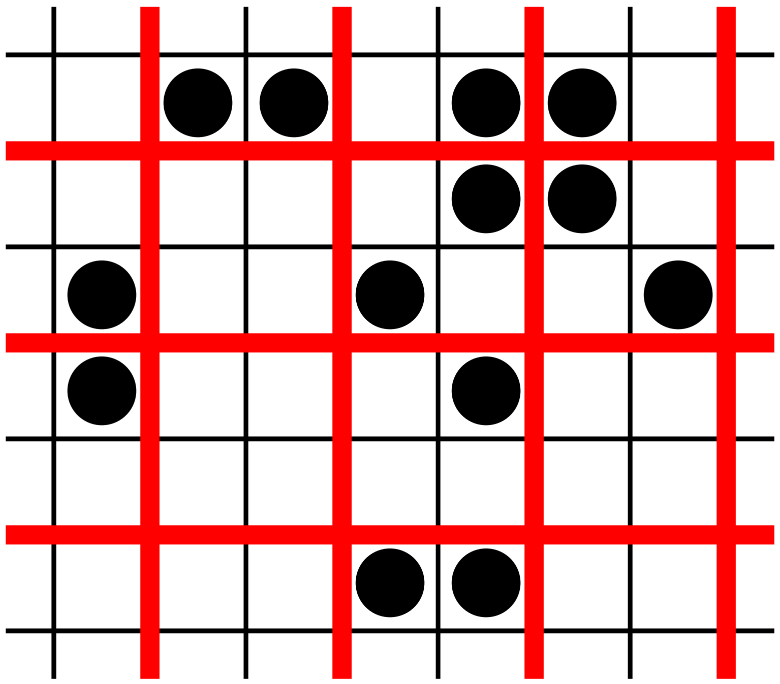
1. Partition the plane into 2×2 blocks and apply some permutation π_1 of S^4 inside each block.

In the Margolus neighborhood the updating is done in two steps:



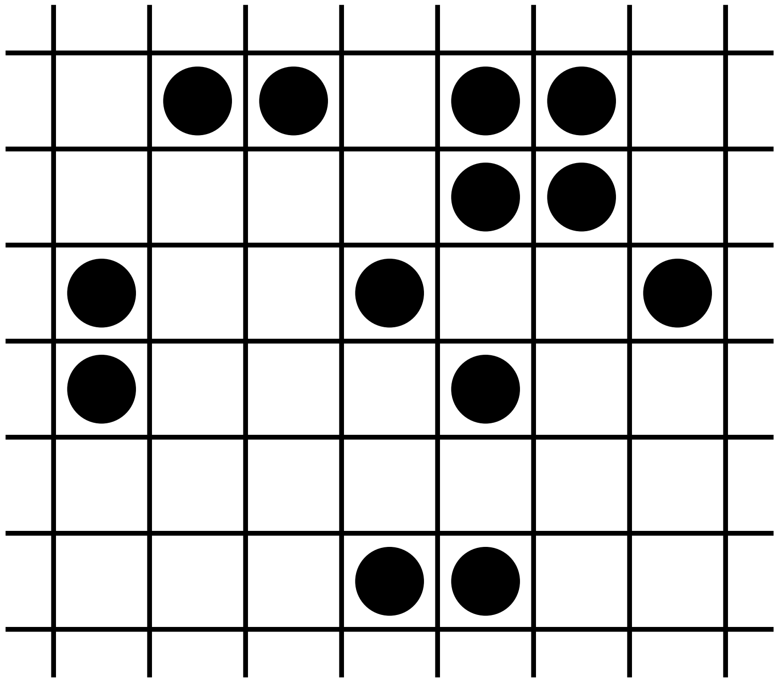
2. Shift the partitioning horizontally and vertically, and apply another permutation π_2 of S^4 on the new blocks.

In the Margolus neighborhood the updating is done in two steps:



2. Shift the partitioning horizontally and vertically, and apply another permutation π_2 of S^4 on the new blocks.

In the Margolus neighborhood the updating is done in two steps:

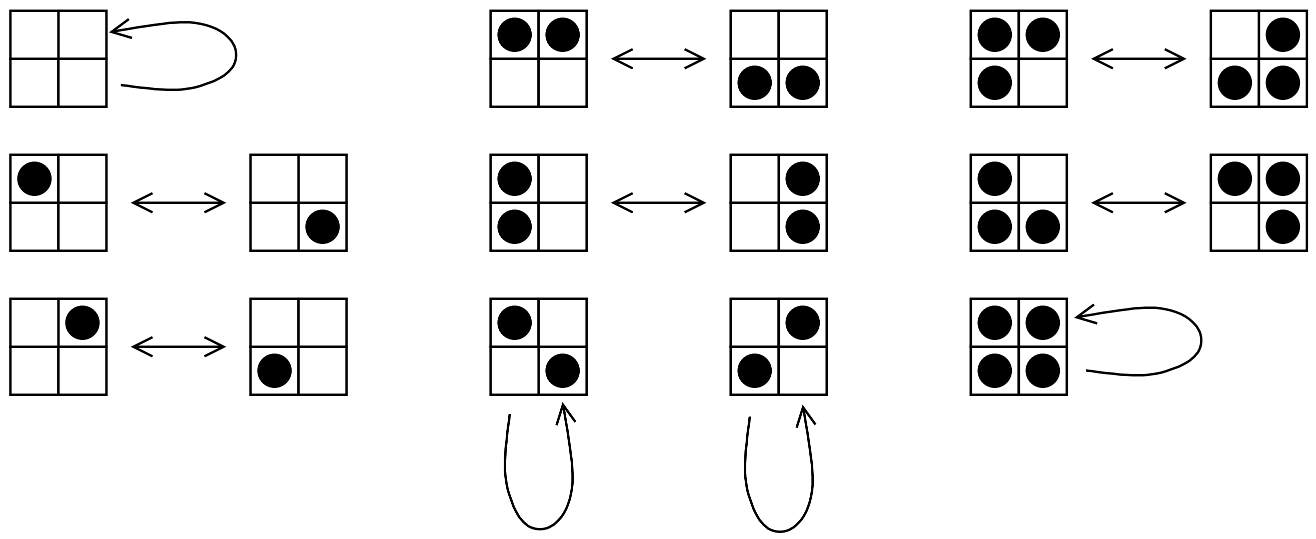


The composition of the two block permutation is one iteration of the CA. It is trivially reversible.

Usually the two permutations are the same $\pi_1 = \pi_2$.

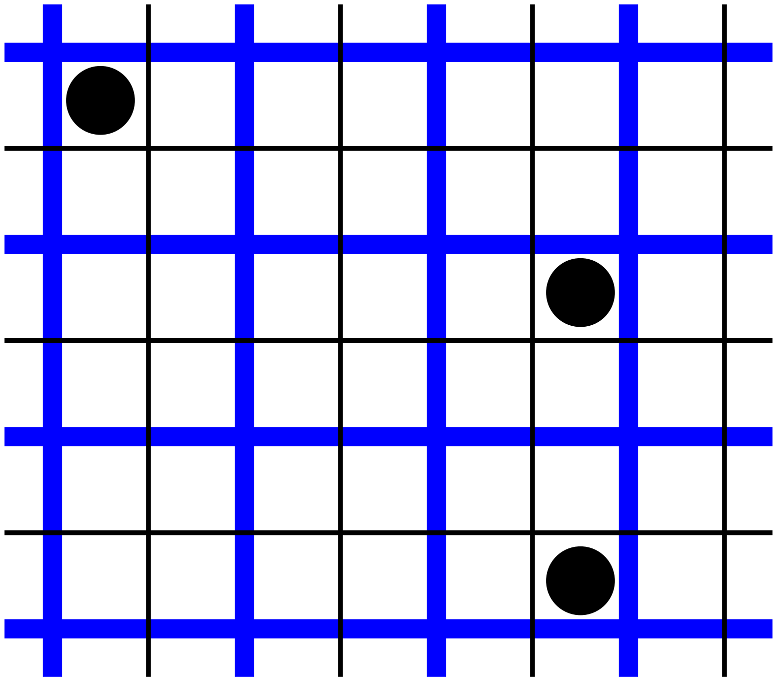
Example 1.

Two states and the following permutation $\pi = \pi_1 = \pi_2$ on all rounds:

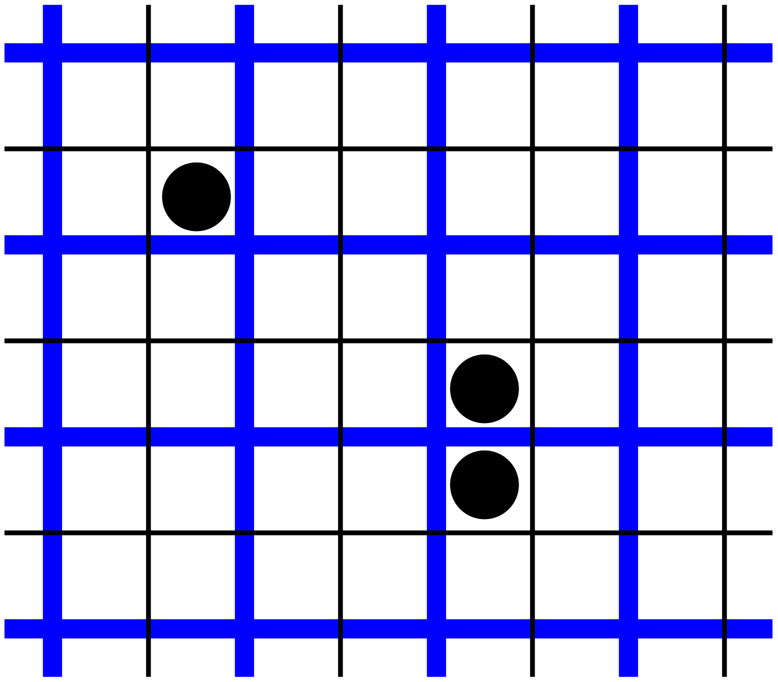


This is a simple half turn of the block: the color of a corner moves to the opposite corner.

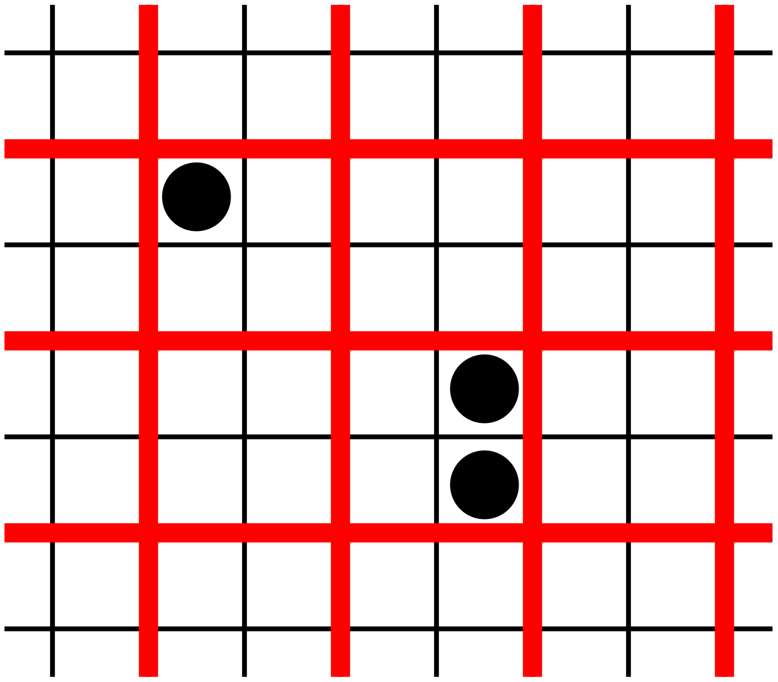
Interpreting the black state as a particle, the particle moves diagonally across $\mathbb{Z}^2$ with constant speed. The direction depends on the position inside the 2×2 block. The particles do not interact:



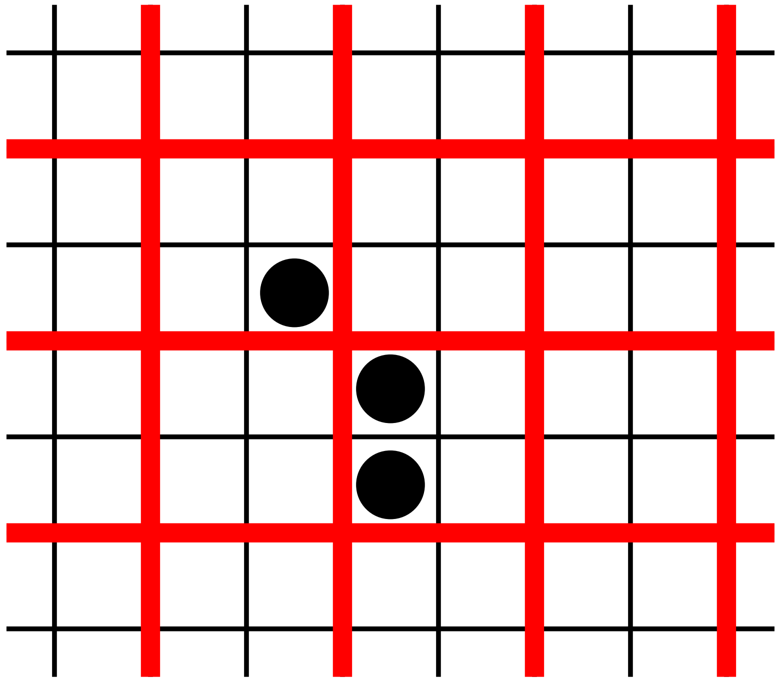
Interpreting the black state as a particle, the particle moves diagonally across $\mathbb{Z}^2$ with constant speed. The direction depends on the position inside the 2×2 block. The particles do not interact:



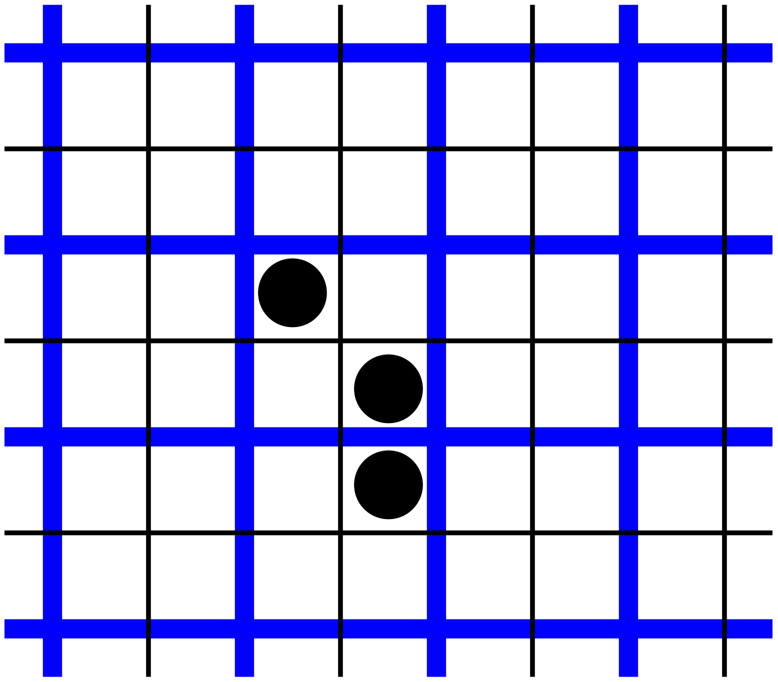
Interpreting the black state as a particle, the particle moves diagonally across $\mathbb{Z}^2$ with constant speed. The direction depends on the position inside the 2×2 block. The particles do not interact:



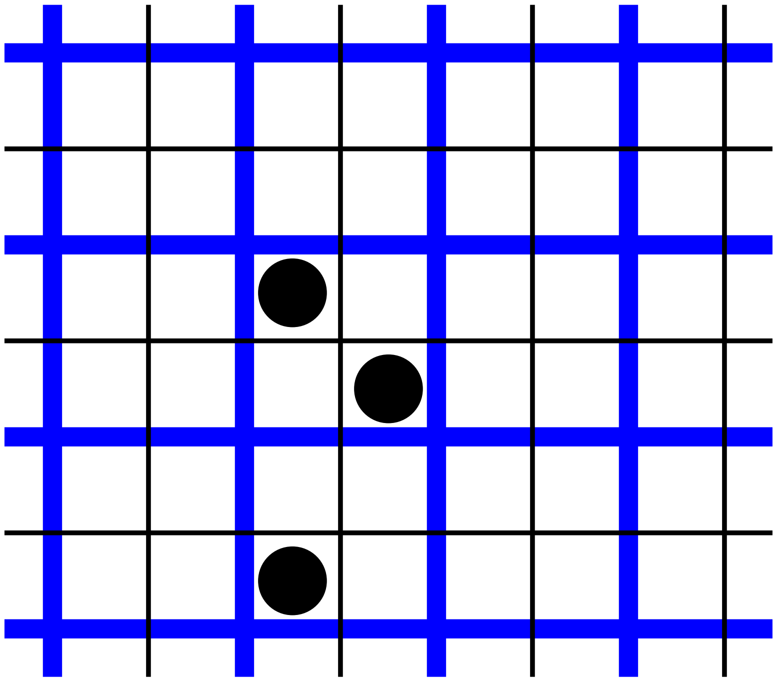
Interpreting the black state as a particle, the particle moves diagonally across $\mathbb{Z}^2$ with constant speed. The direction depends on the position inside the 2×2 block. The particles do not interact:



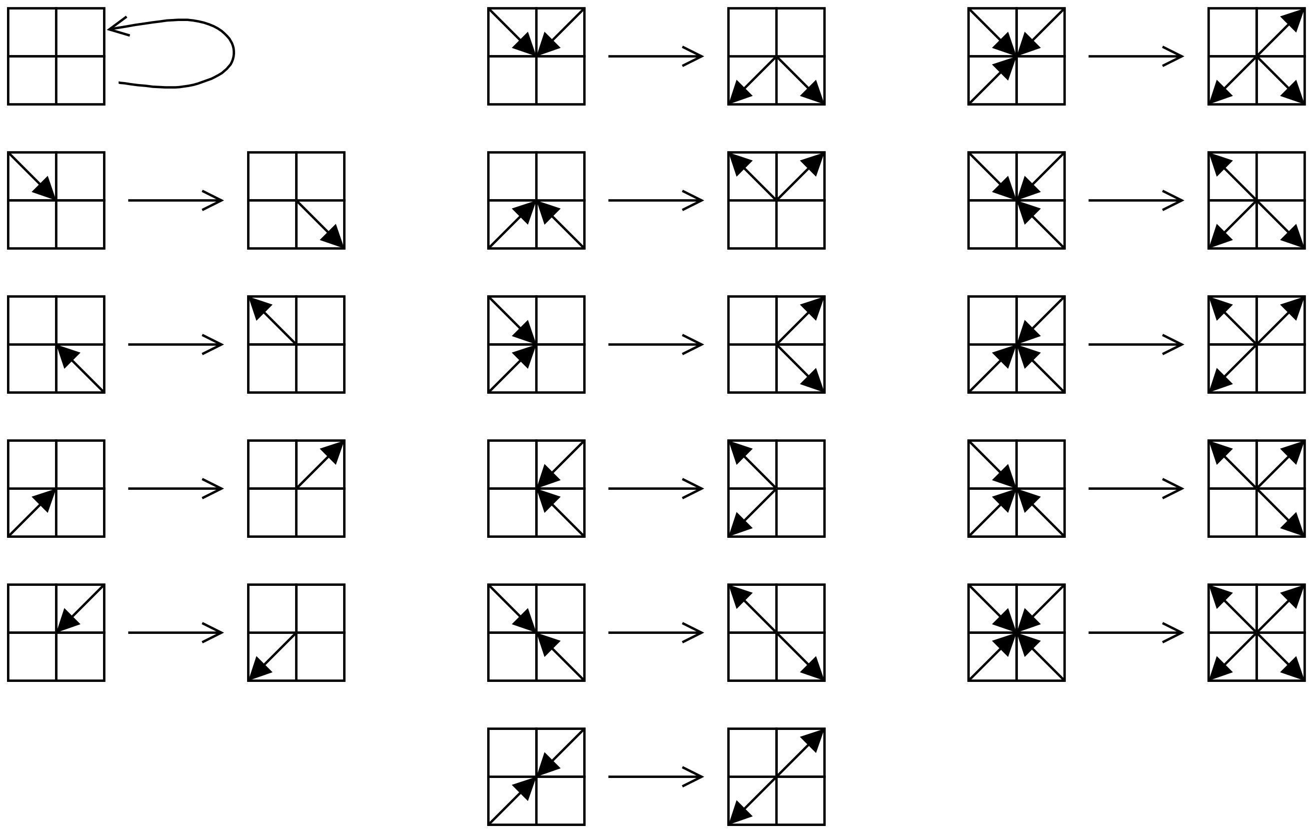
Interpreting the black state as a particle, the particle moves diagonally across $\mathbb{Z}^2$ with constant speed. The direction depends on the position inside the 2×2 block. The particles do not interact:



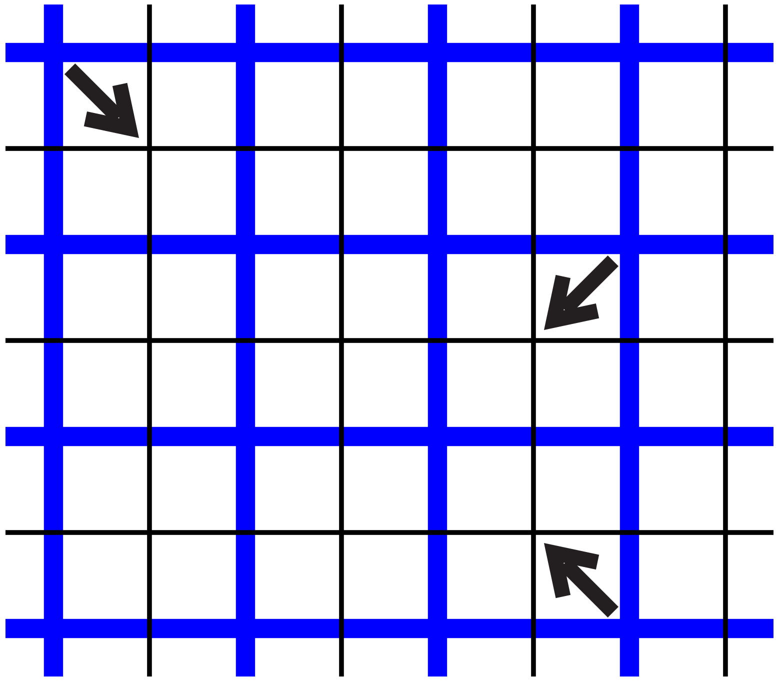
Interpreting the black state as a particle, the particle moves diagonally across $\mathbb{Z}^2$ with constant speed. The direction depends on the position inside the 2×2 block. The particles do not interact:



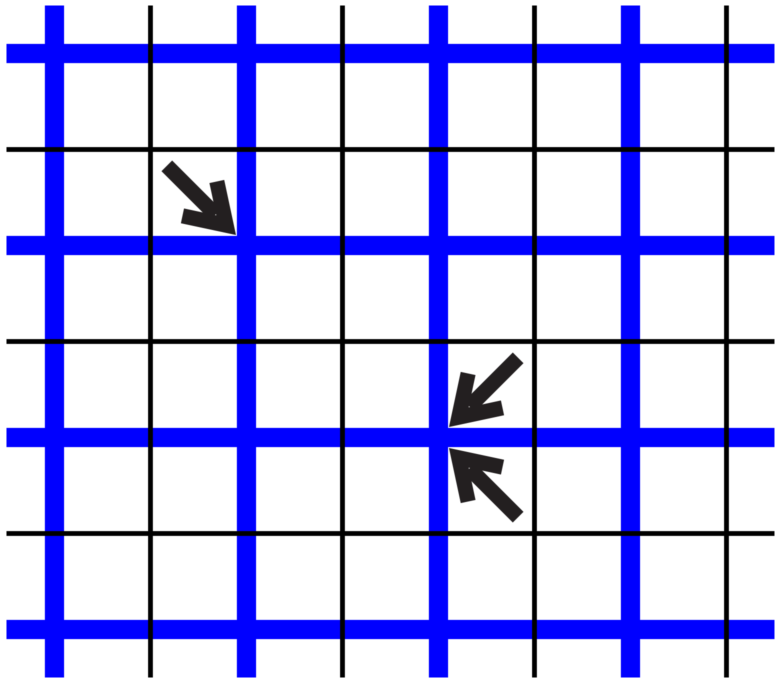
Conveniently drawing a particle as a diagonal arrow pointing to its direction of motion (=towards the center of the 2×2 block), the permutation becomes



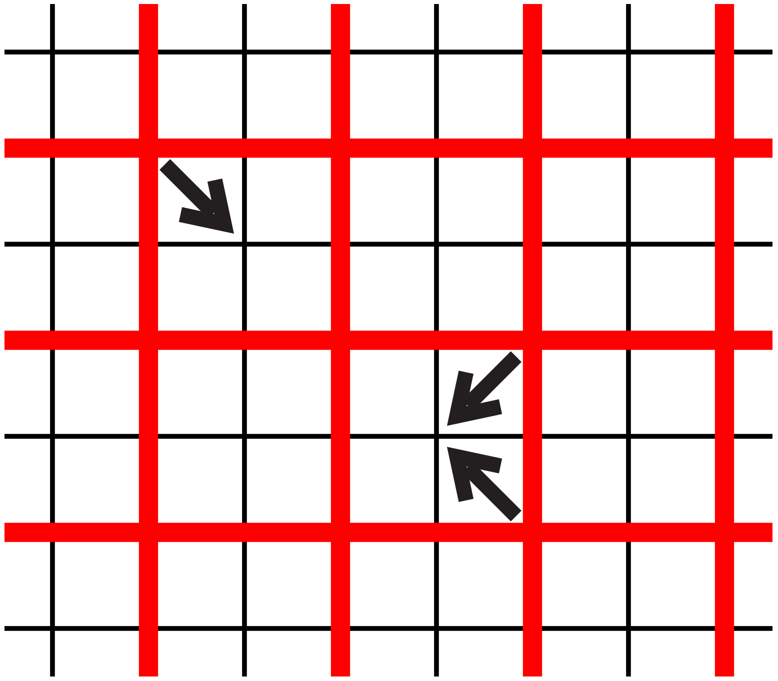
Our sample iteration becomes:



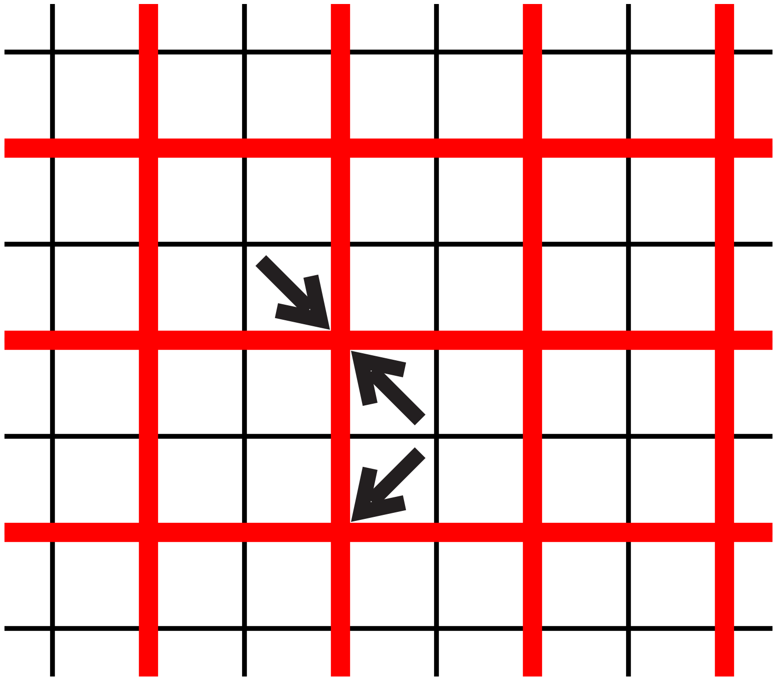
Our sample iteration becomes:



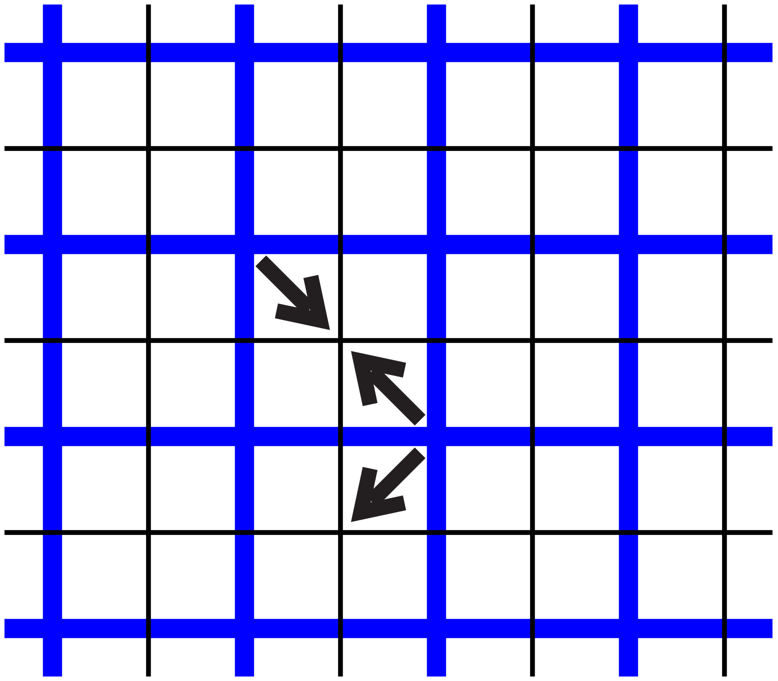
Our sample iteration becomes:



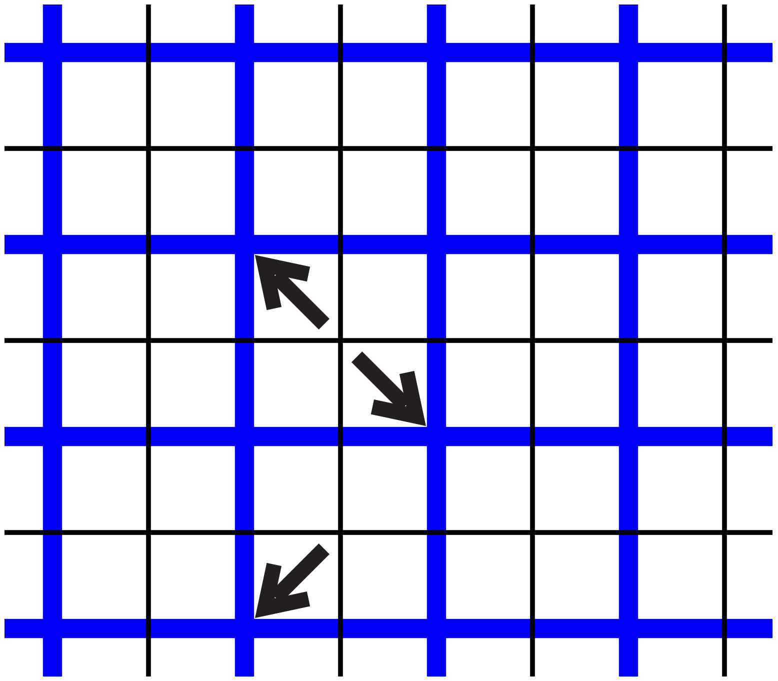
Our sample iteration becomes:



Our sample iteration becomes:

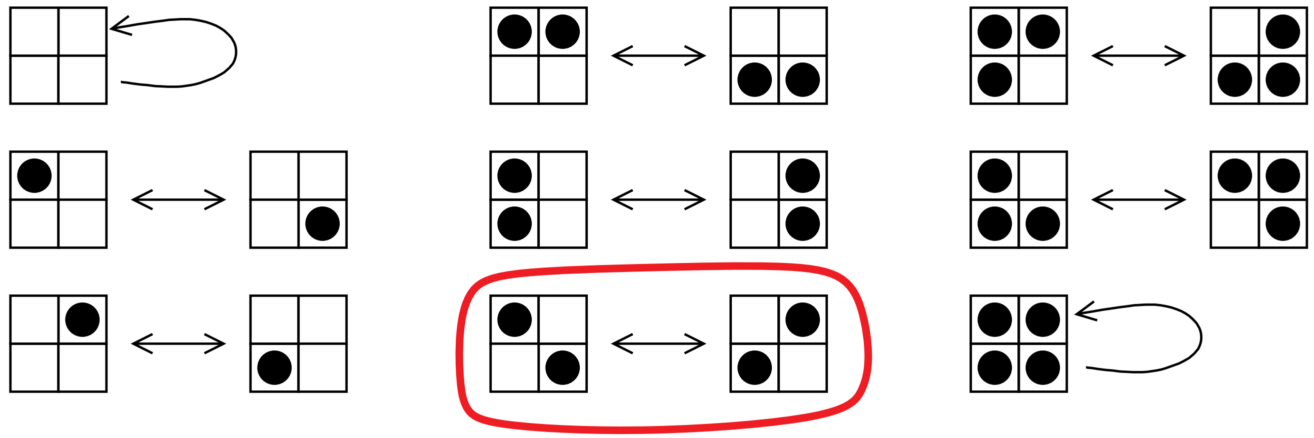


Our sample iteration becomes:



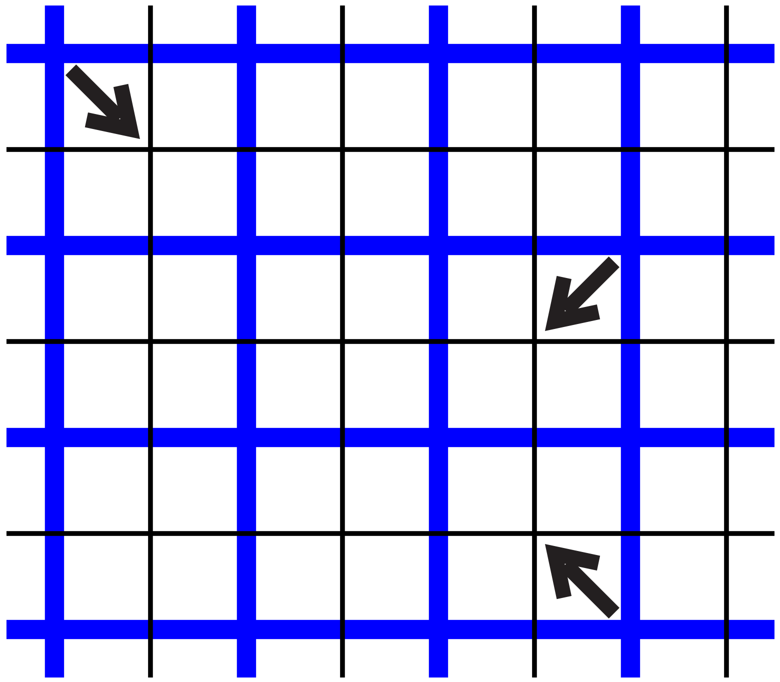
In this CA every particle moves uninterrupted in its direction, and there are no interactions between particles. Each block can contain up to four particles, all moving to different directions.

Example 2. Let us introduce particle interaction in the case when two particles collide head-on. The new permutation $\pi = \pi_1 = \pi_2$ is the following:

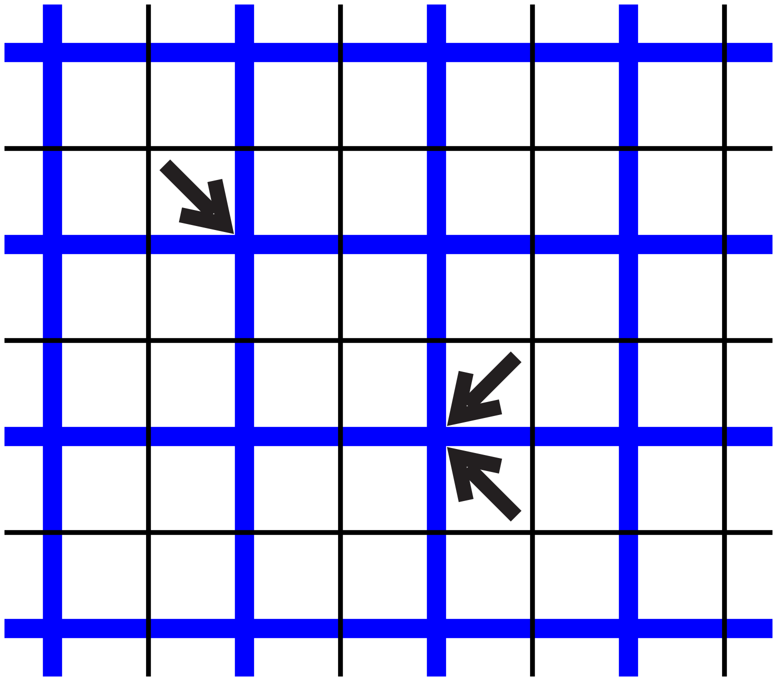


The only change is in a block with two diagonally aligned black and white cells: In such head-on collision both particles turn 90°.

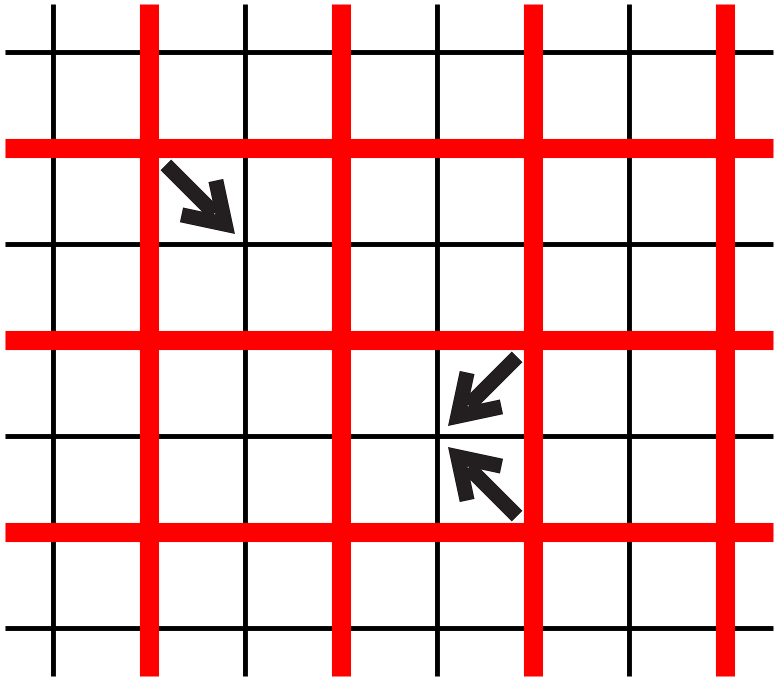
The resulting CA is the **HPP lattice gas**.



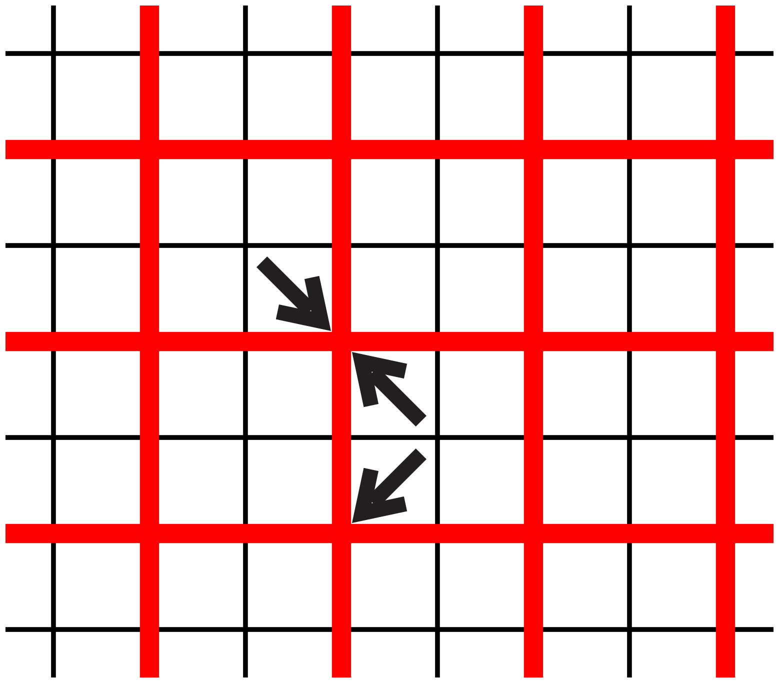
The resulting CA is the **HPP lattice gas**.



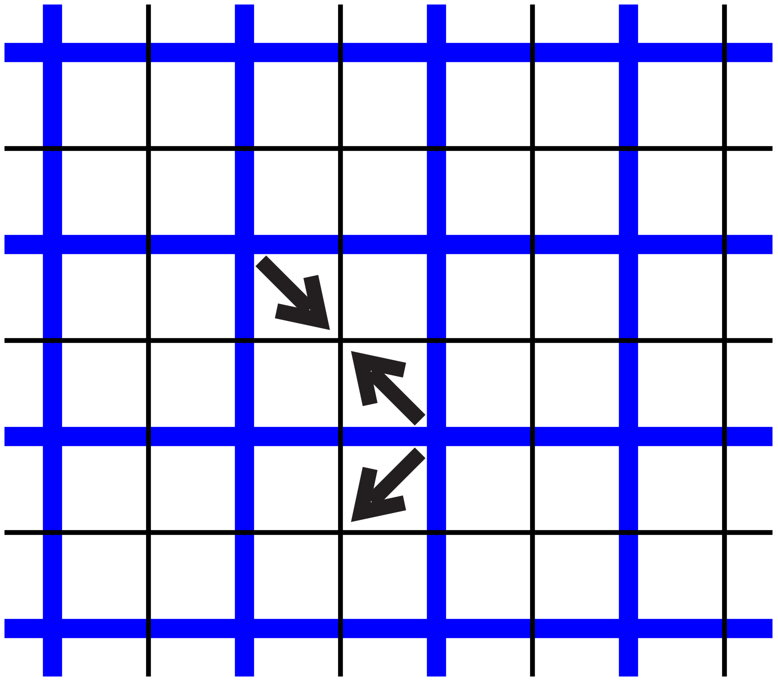
The resulting CA is the **HPP lattice gas**.



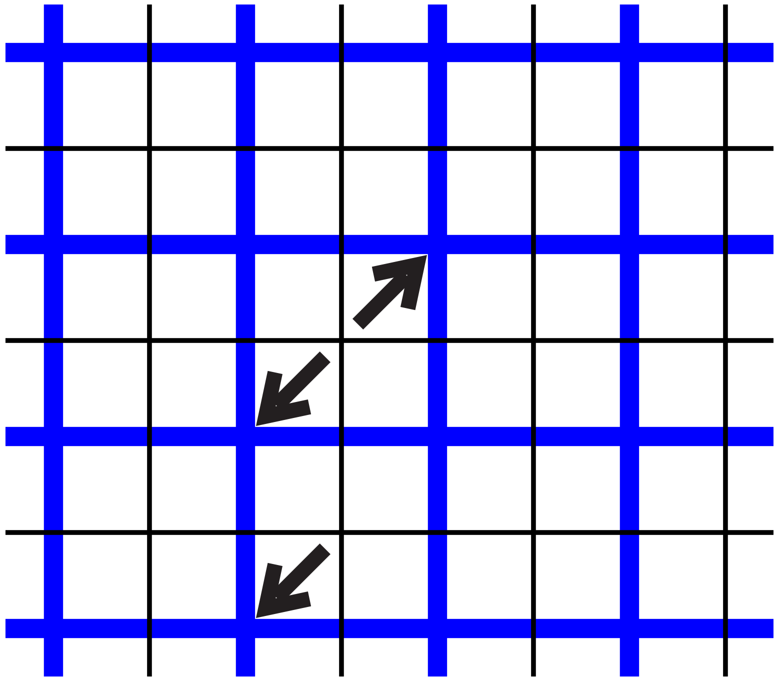
The resulting CA is the **HPP lattice gas**.



The resulting CA is the **HPP lattice gas**.



The resulting CA is the **HPP lattice gas**.



HPP provides a simplistic simulation of gas or fluid dynamics. The particles represent molecules. HPP is **reversible** as is the physical system it attempts to simulate.

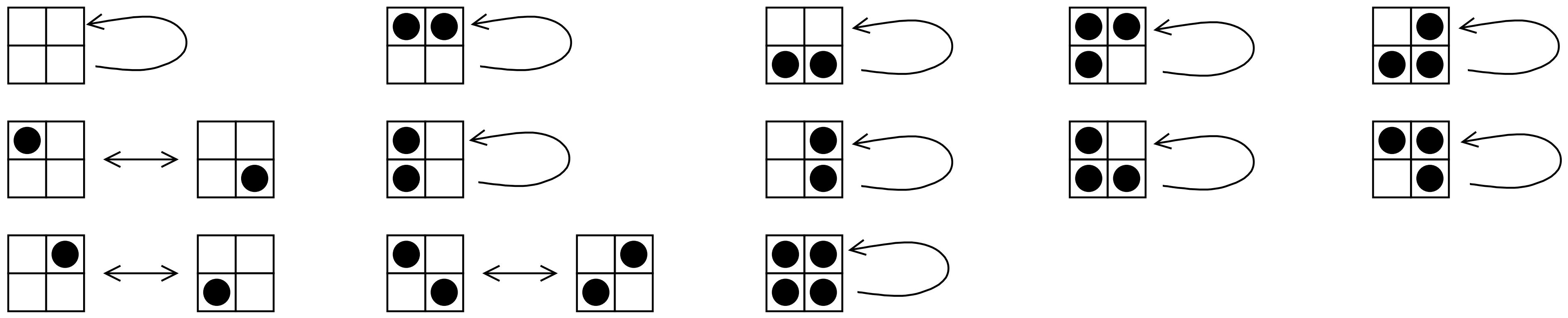
HPP also has **conservation laws**:

(1) The total **mass** (=number of particles) remains invariant. Hence also the total **energy** is preserved, since each particle has the same kinetic energy.

(2) The total **momentum** of the system is preserved. (Momentum is the sum of the velocity vectors of the particles.) The only update where particle directions change is in a two-particle block where the total momentum before and after the update is zero.

Our next section studies such **conserved quantities** as the mass, energy and momentum in HPP.

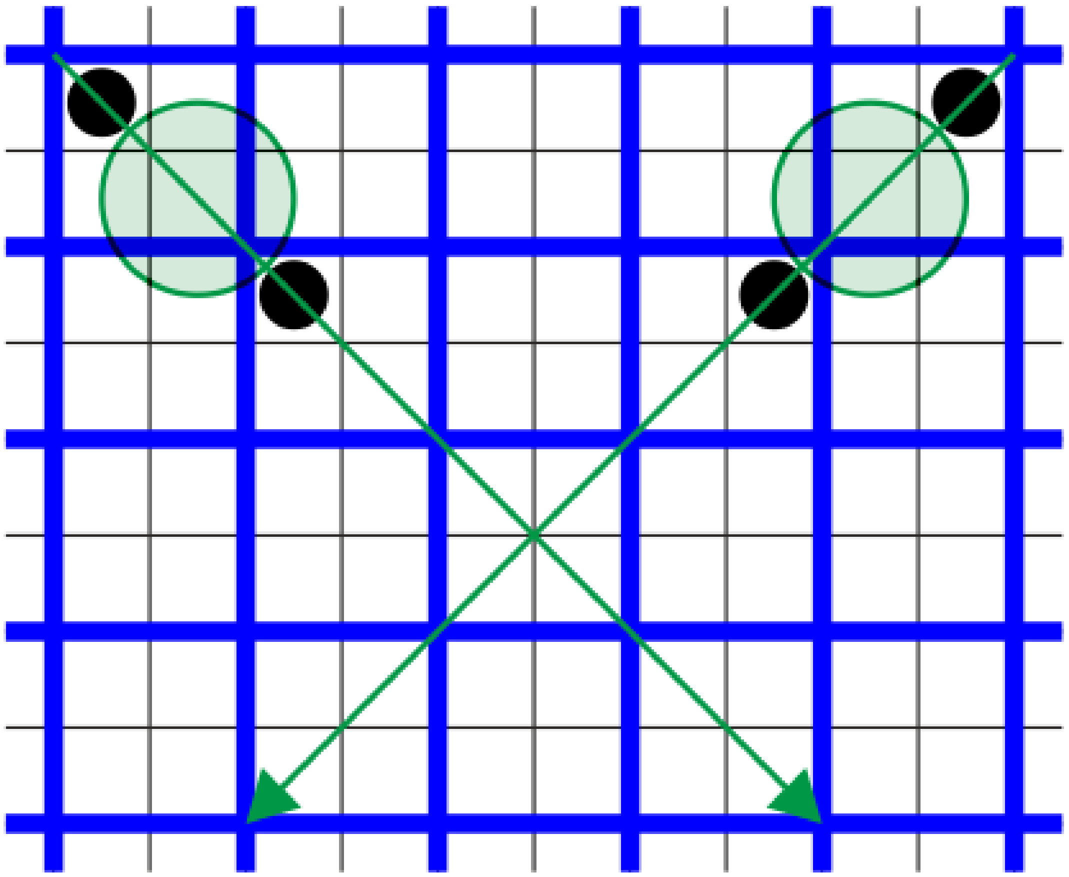
Example 3. Let $\pi = \pi_1 = \pi_2$ be as follows:



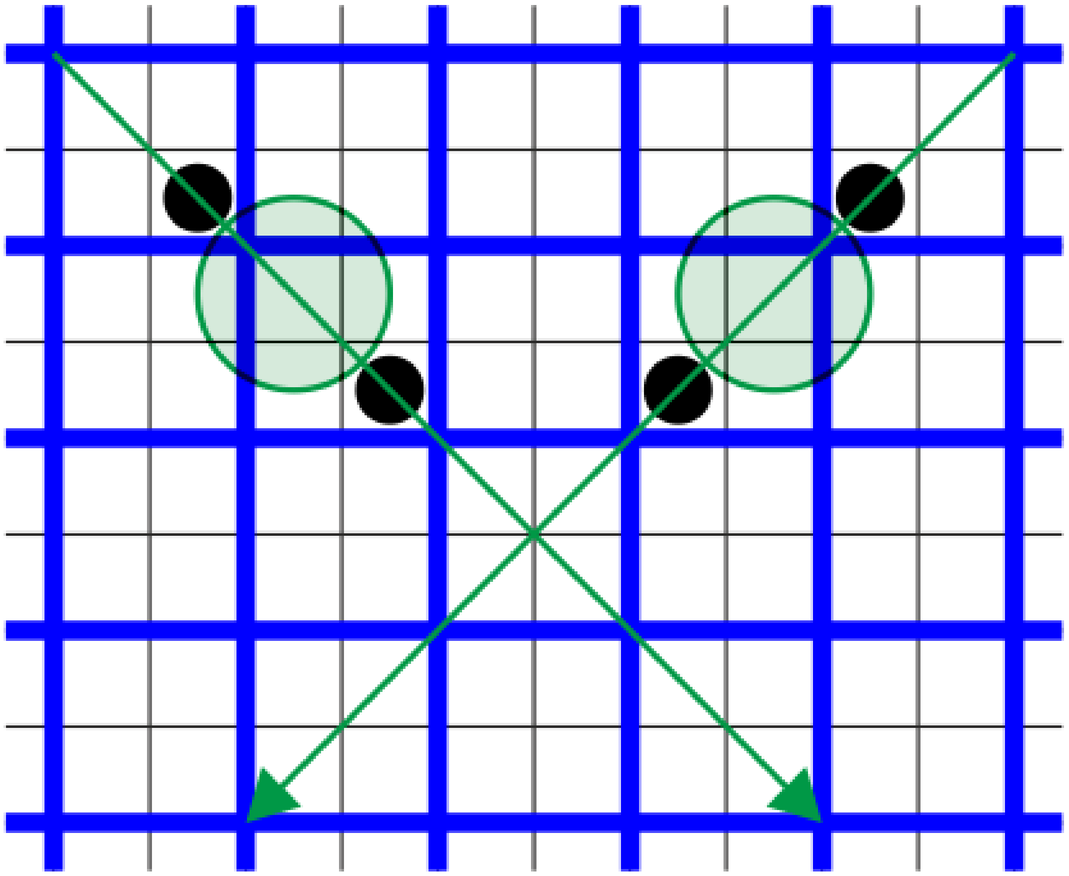
Again the numbers of tokens are conserved. Now a collision of any number of particles makes them reverse their directions, except when exactly two particle collide head on then they turn $90°$.

This CA by N.Margolus simulates the **billiard ball model of computation (BBM)** by E. Fredkin.

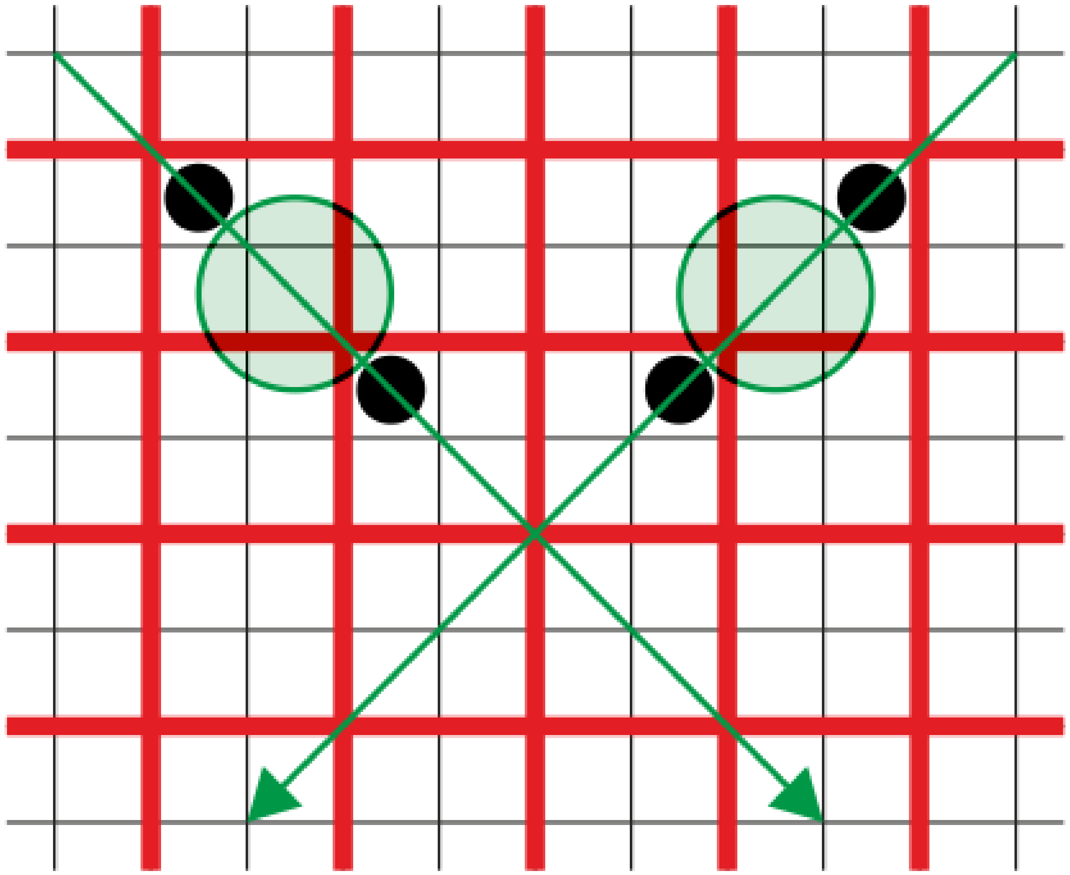
The BBMCA can simulate collisions of balls of identical positive radius. The collisions are “soft” meaning that a collision takes time.



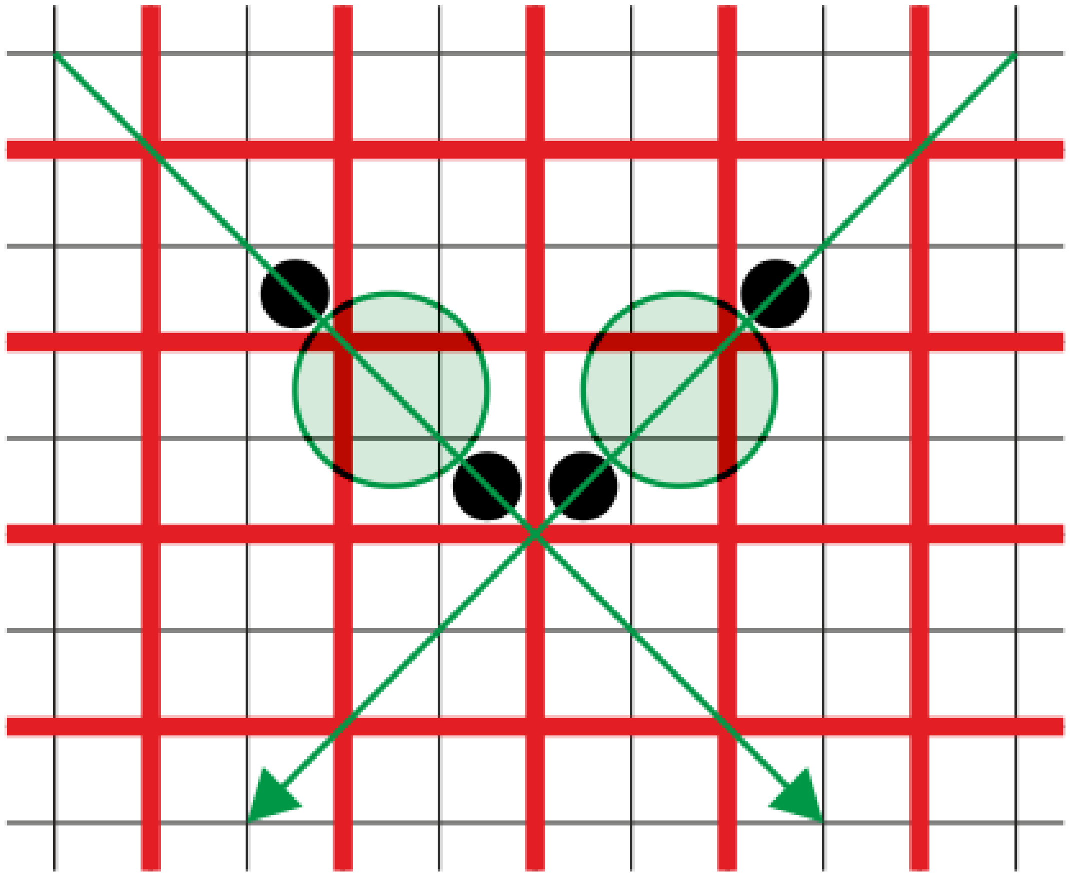
The BBMCA can simulate collisions of balls of identical positive radius. The collisions are “soft” meaning that a collision takes time.



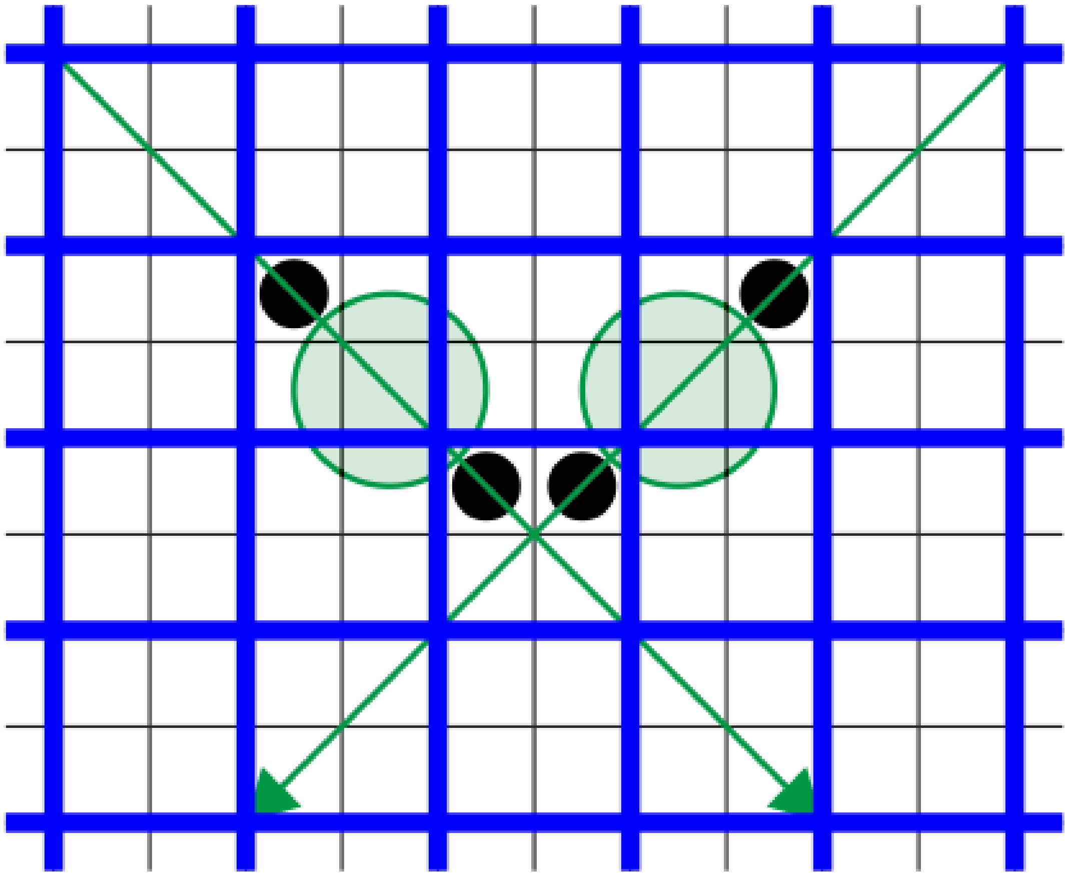
The BBMCA can simulate collisions of balls of identical positive radius. The collisions are "soft" meaning that a collision takes time.



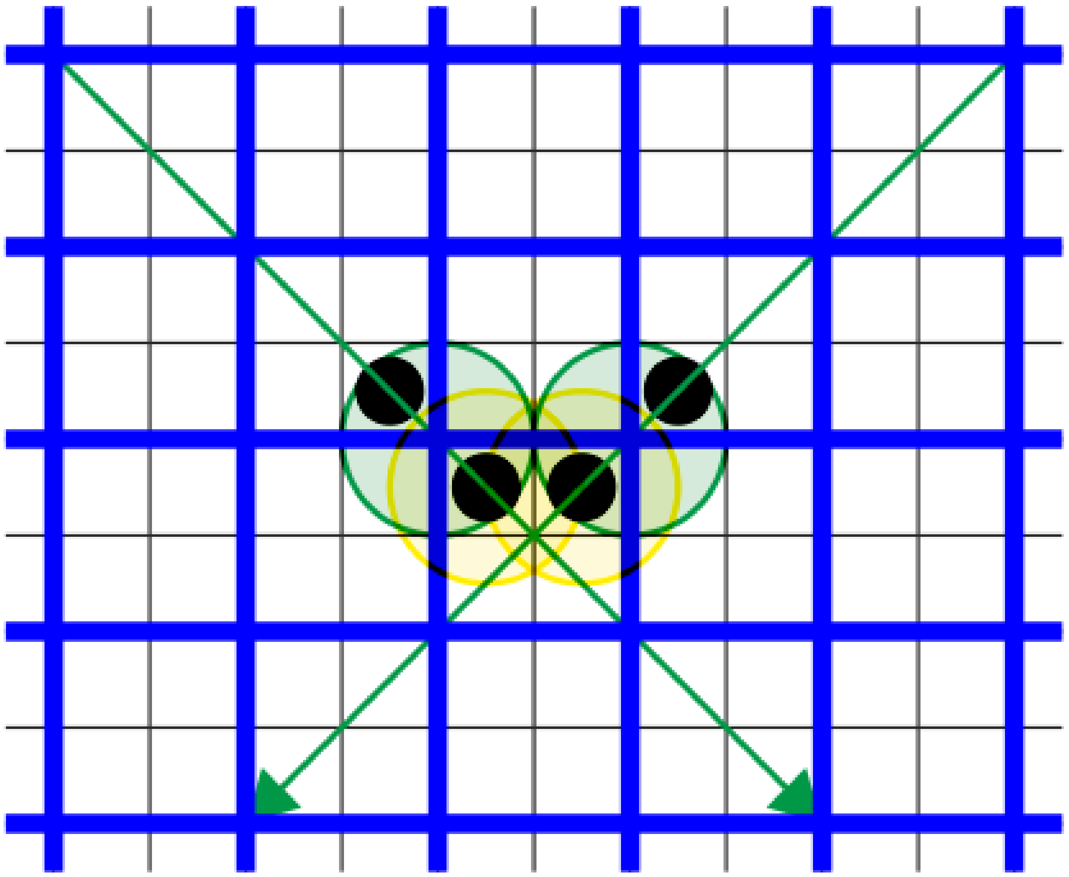
The BBMCA can simulate collisions of balls of identical positive radius. The collisions are "soft" meaning that a collision takes time.



The BBMCA can simulate collisions of balls of identical positive radius. The collisions are "soft" meaning that a collision takes time.

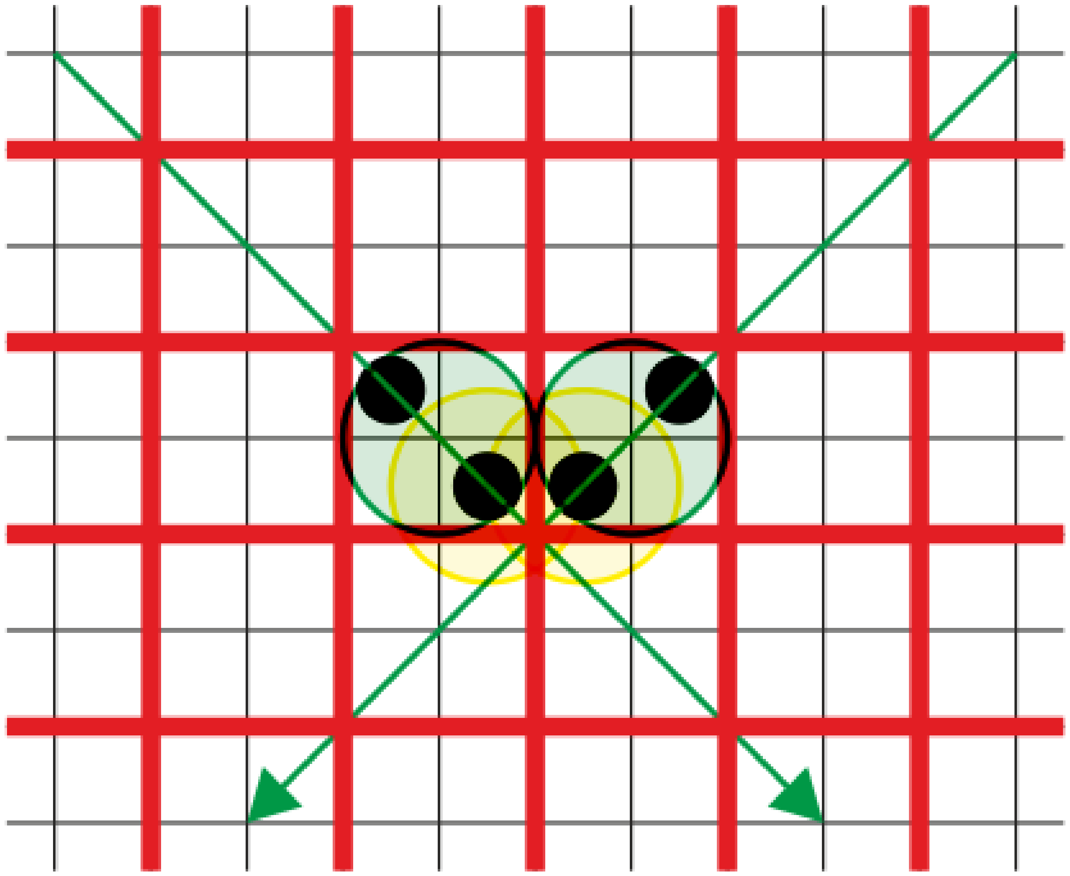


The BBMCA can simulate collisions of balls of identical positive radius. The collisions are "soft" meaning that a collision takes time.

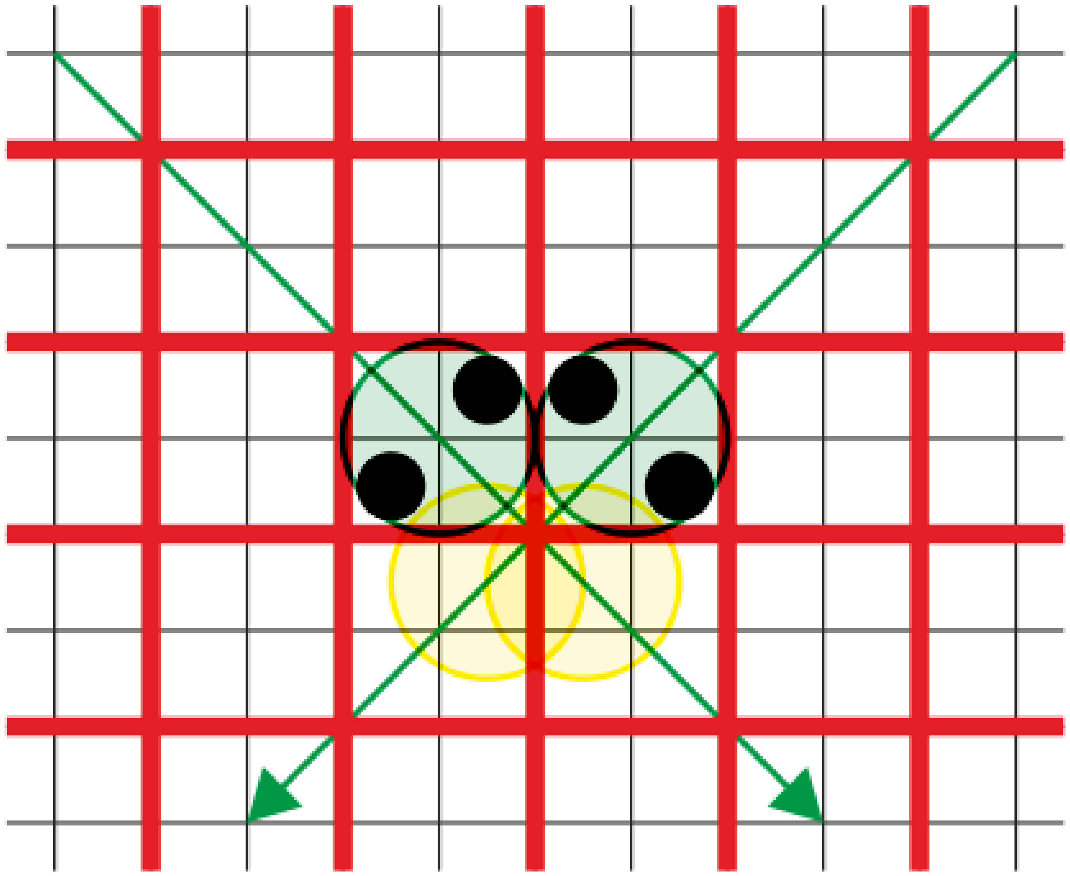


The yellow ball indicates where the ball without a collision would be.

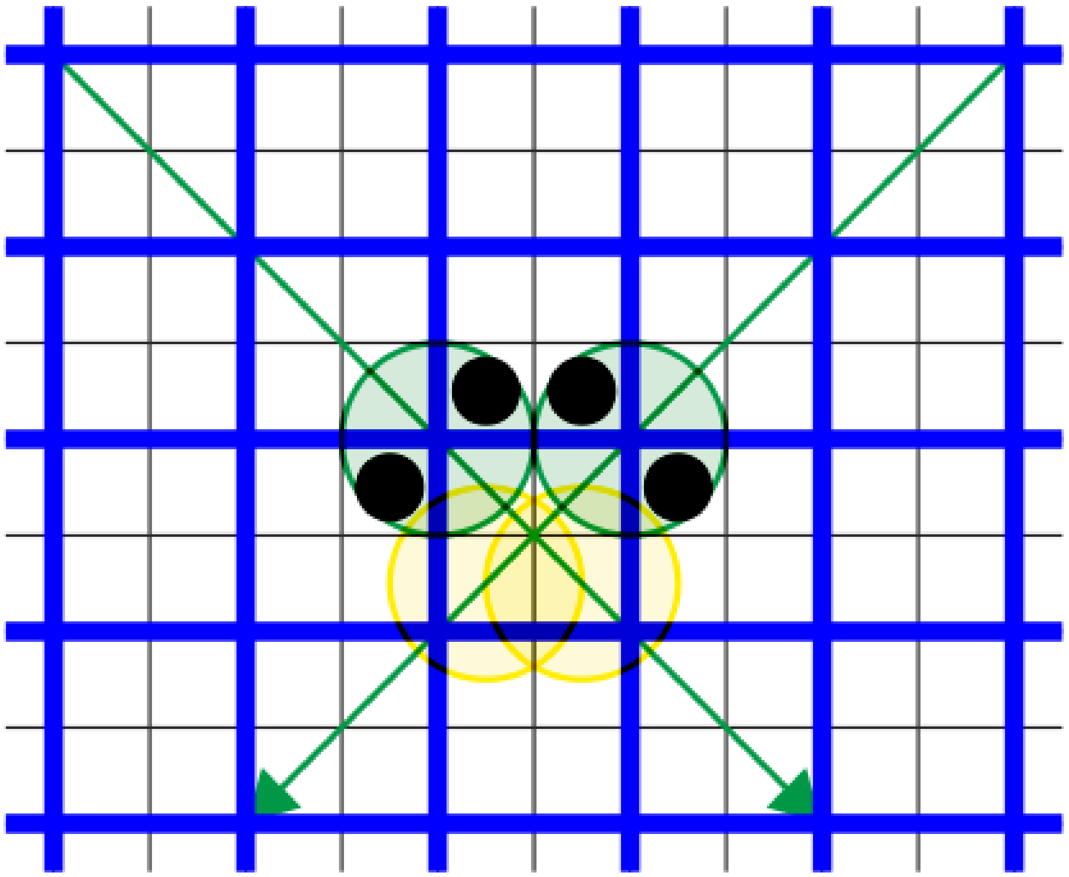
The BBMCA can simulate collisions of balls of identical positive radius. The collisions are "soft" meaning that a collision takes time.



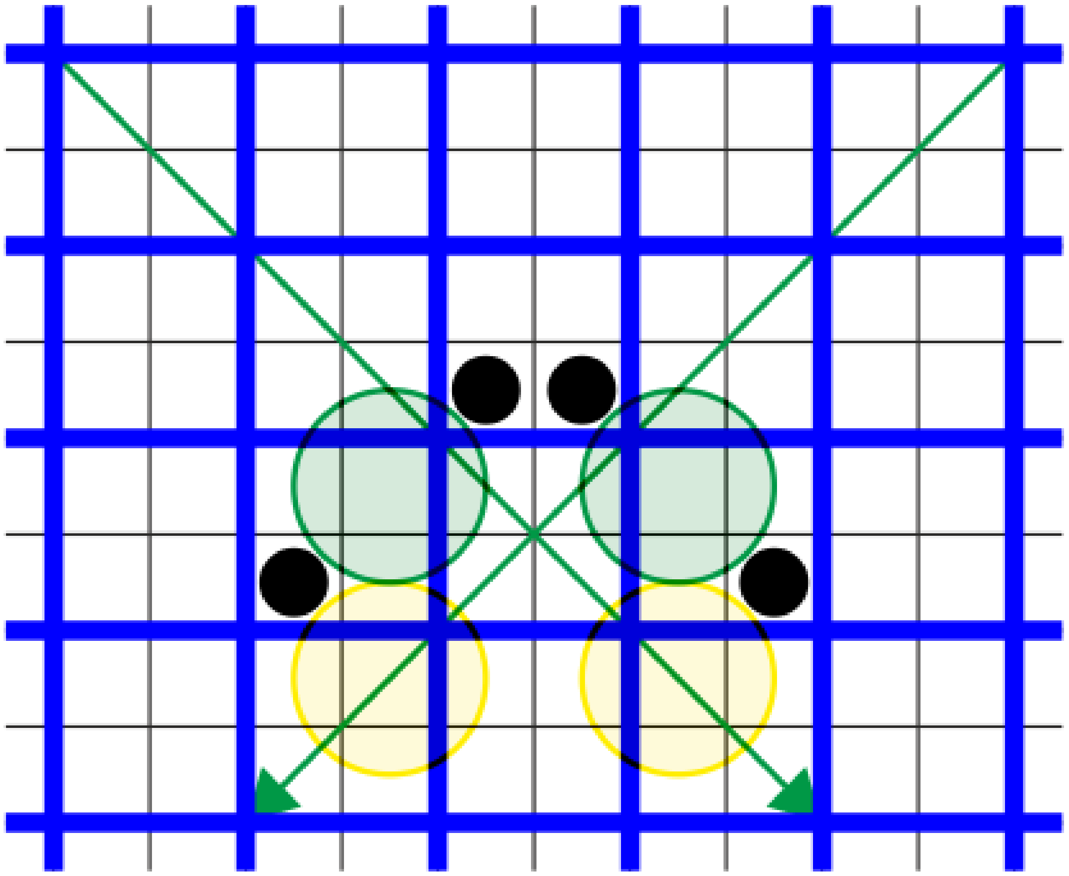
The BBMCA can simulate collisions of balls of identical positive radius. The collisions are "soft" meaning that a collision takes time.



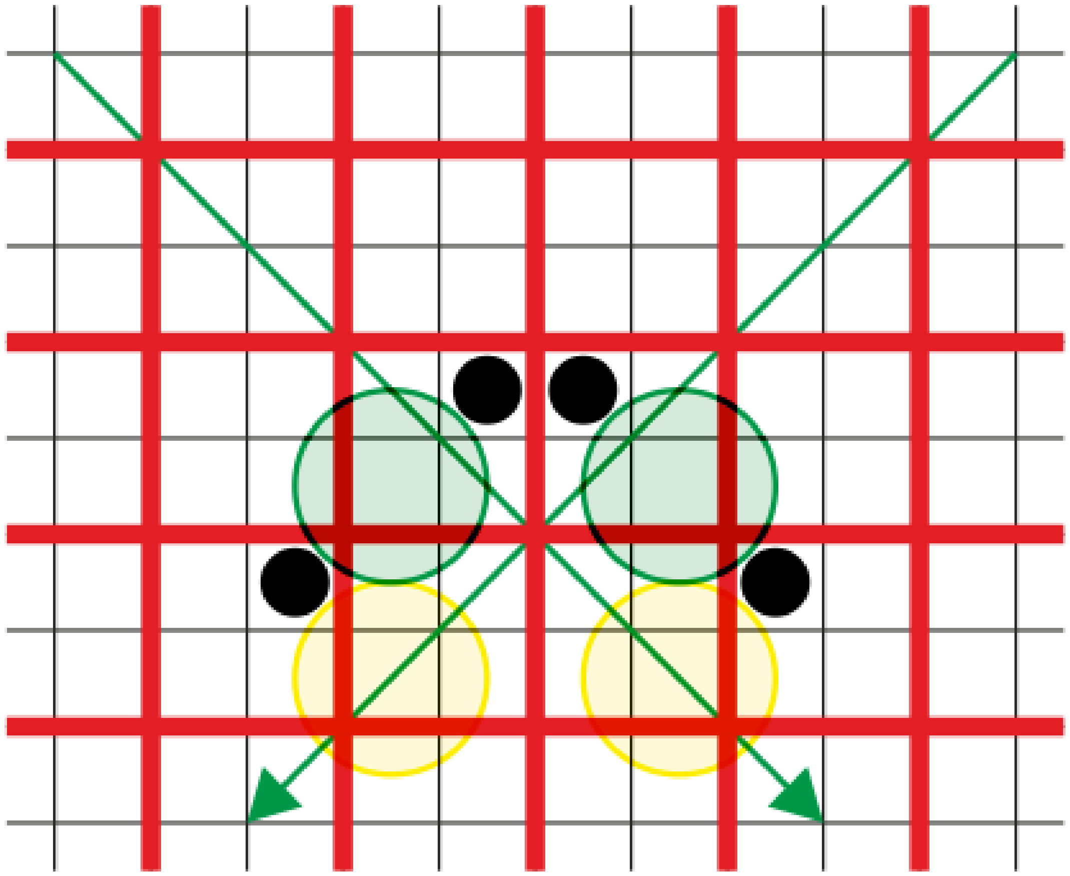
The BBMCA can simulate collisions of balls of identical positive radius. The collisions are "soft" meaning that a collision takes time.



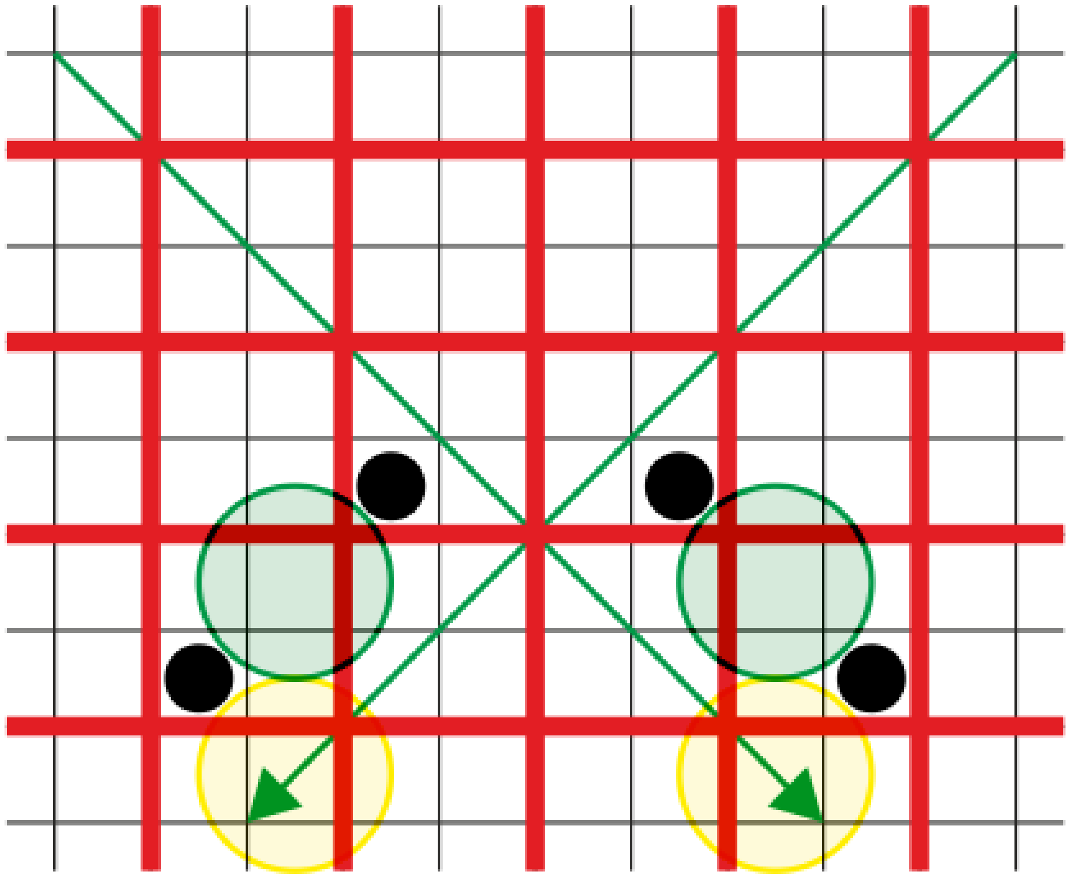
The BBMCA can simulate collisions of balls of identical positive radius. The collisions are “soft” meaning that a collision takes time.



The BBMCA can simulate collisions of balls of identical positive radius. The collisions are "soft" meaning that a collision takes time.

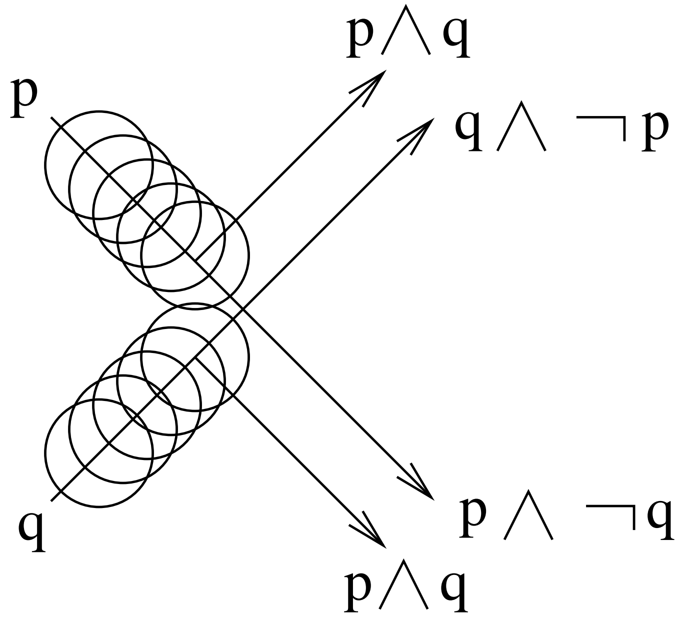


The BBMCA can simulate collisions of balls of identical positive radius. The collisions are "soft" meaning that a collision takes time.

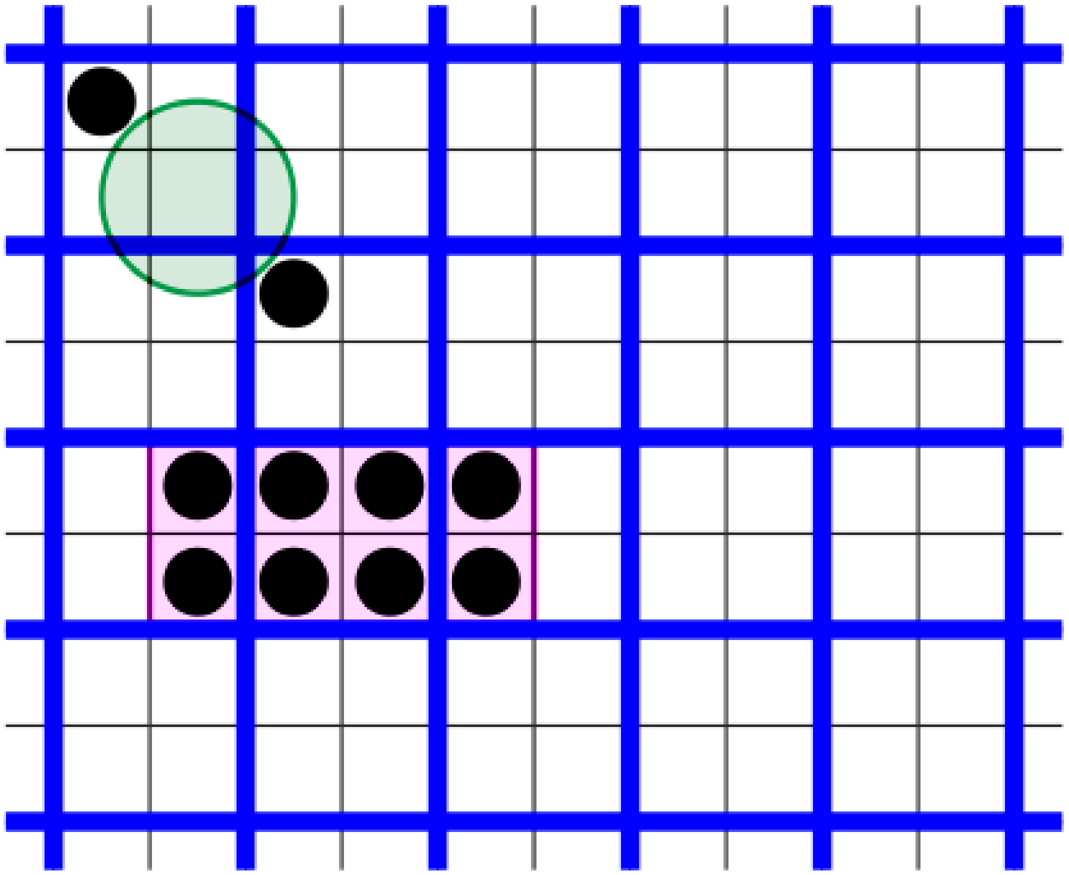


Potential trajectories of balls are wires. Presence/absence of a ball represents the bit 1/0.

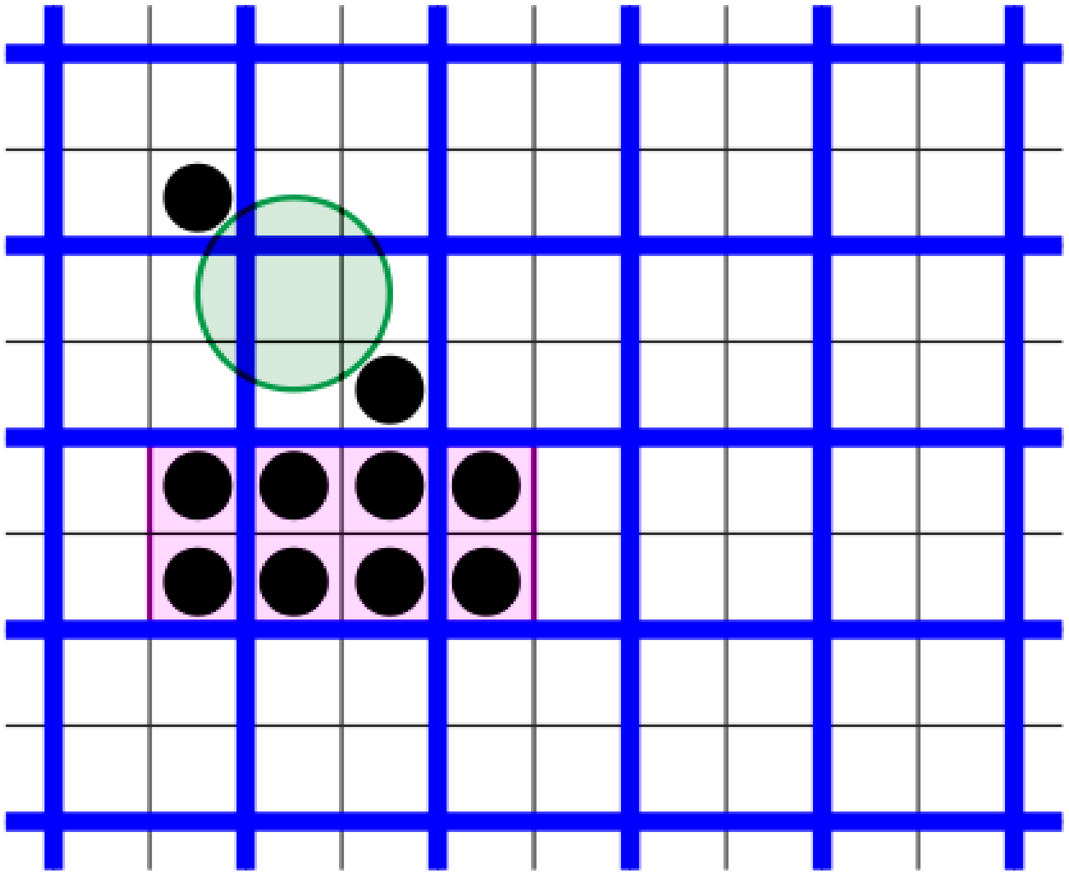
A collision changes the trajectories of balls $\Longrightarrow$ logic gate



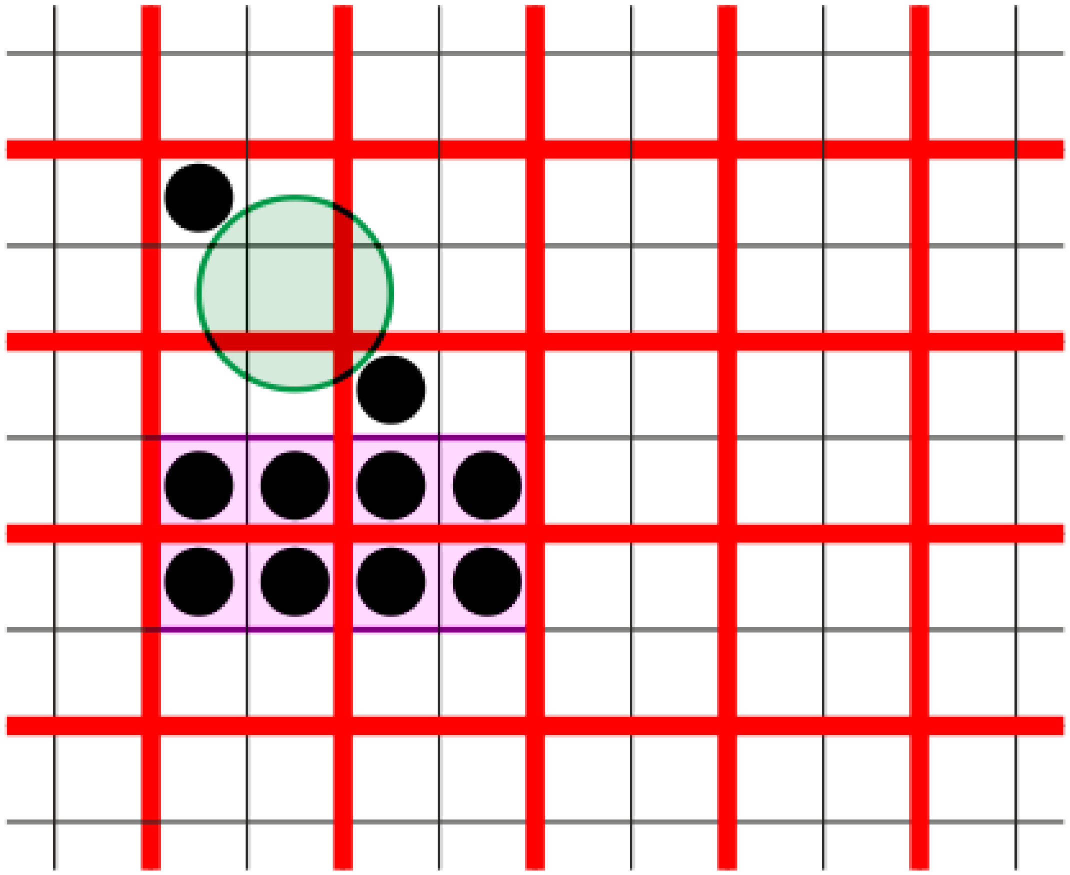
One can make static walls from which balls bounce:



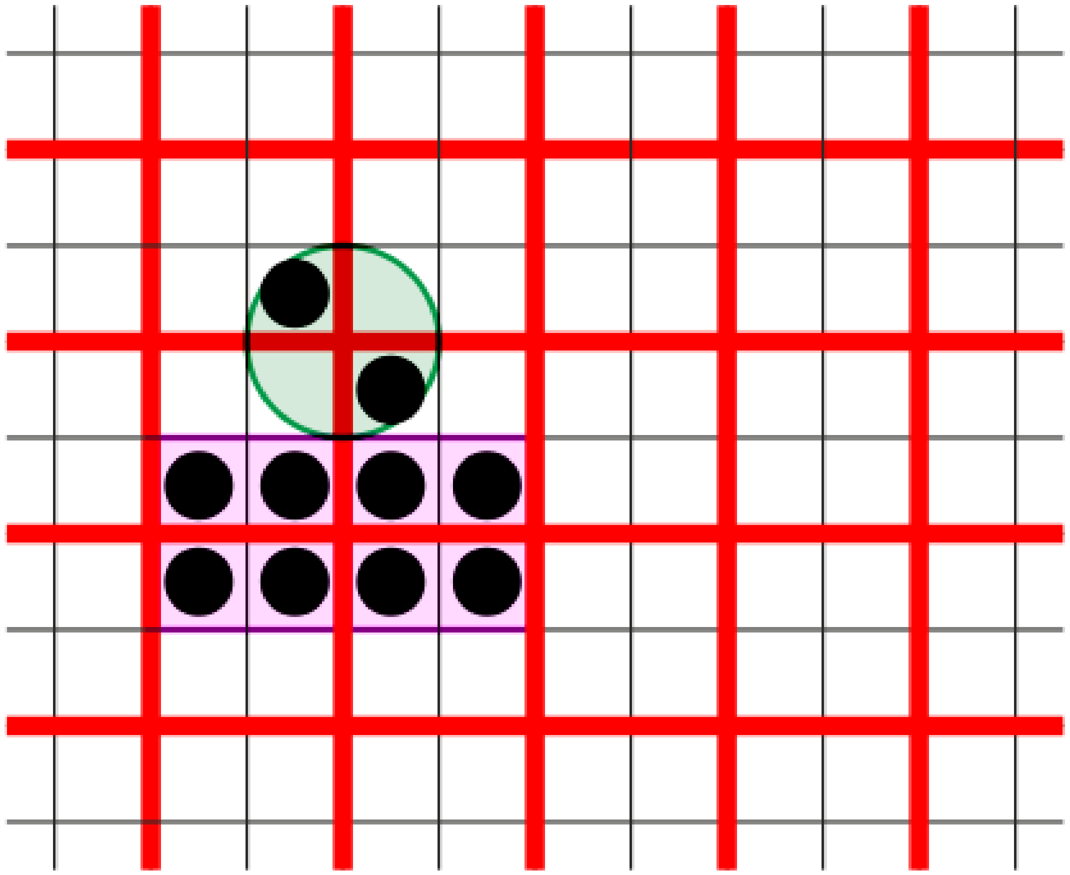
One can make static walls from which balls bounce:



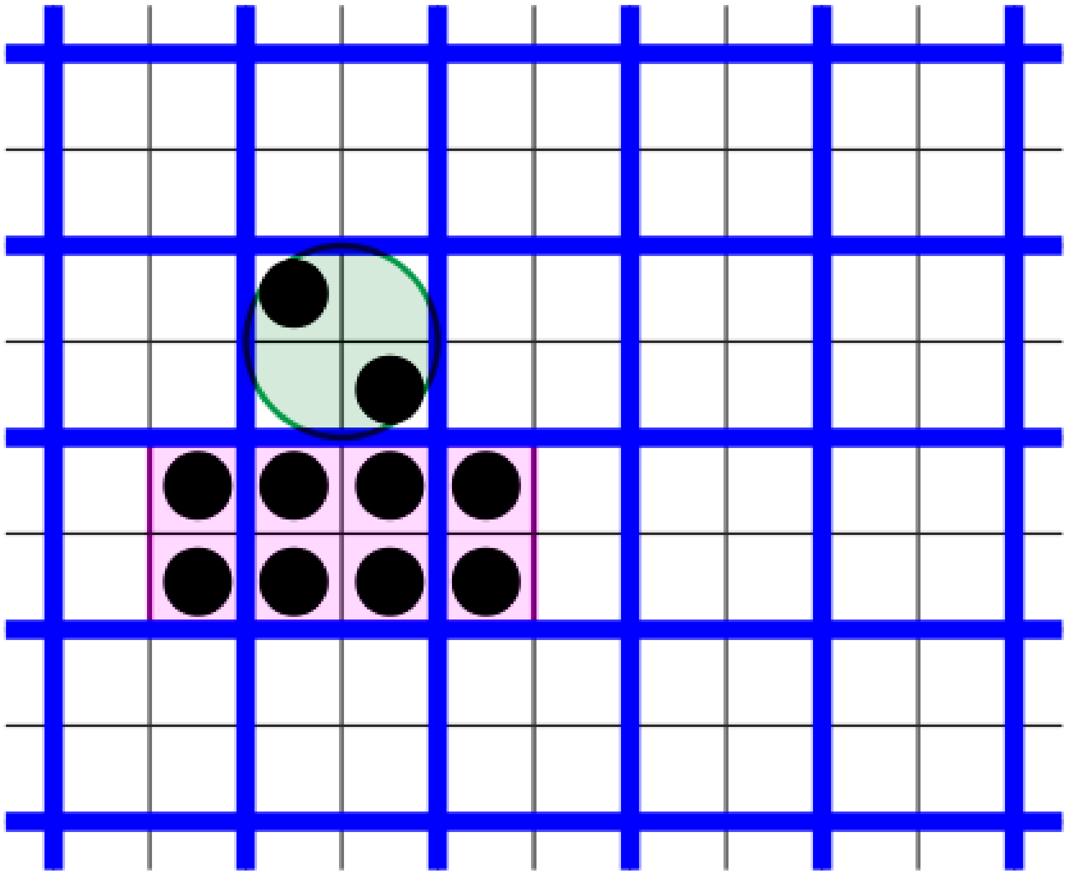
One can make static walls from which balls bounce:



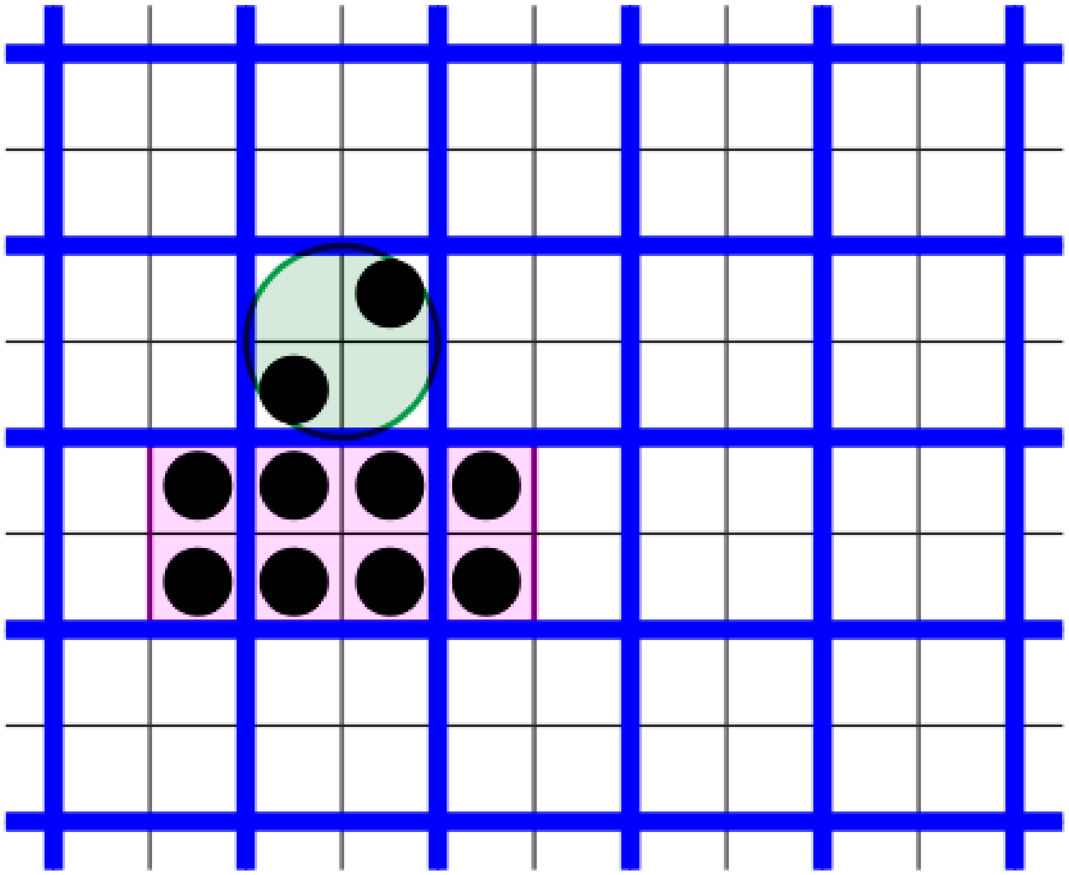
One can make static walls from which balls bounce:



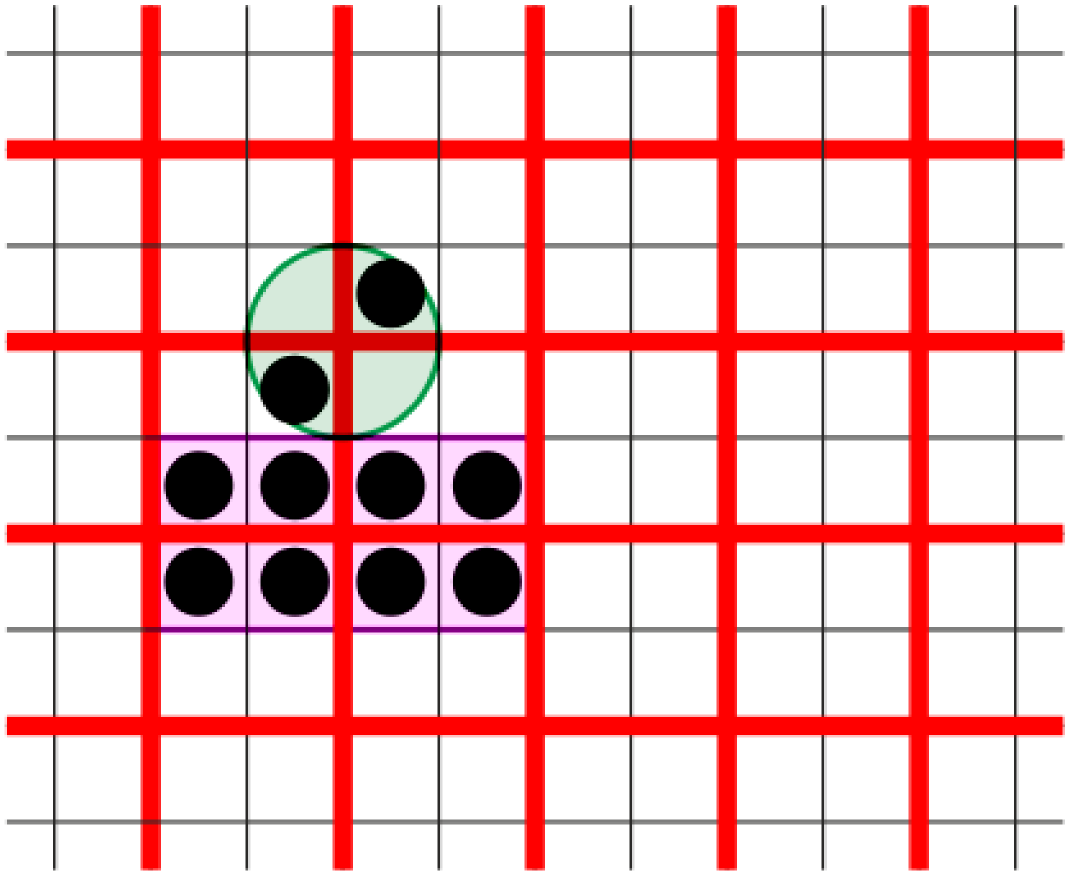
One can make static walls from which balls bounce:



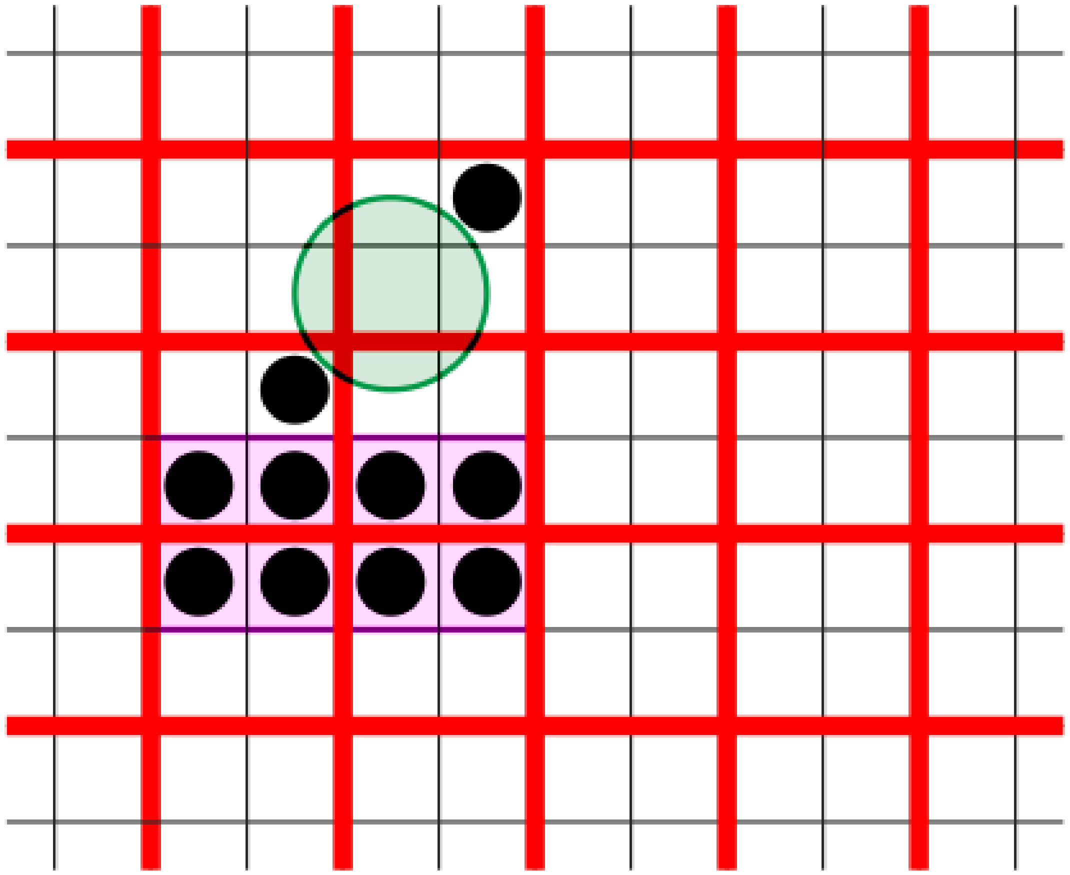
One can make static walls from which balls bounce:



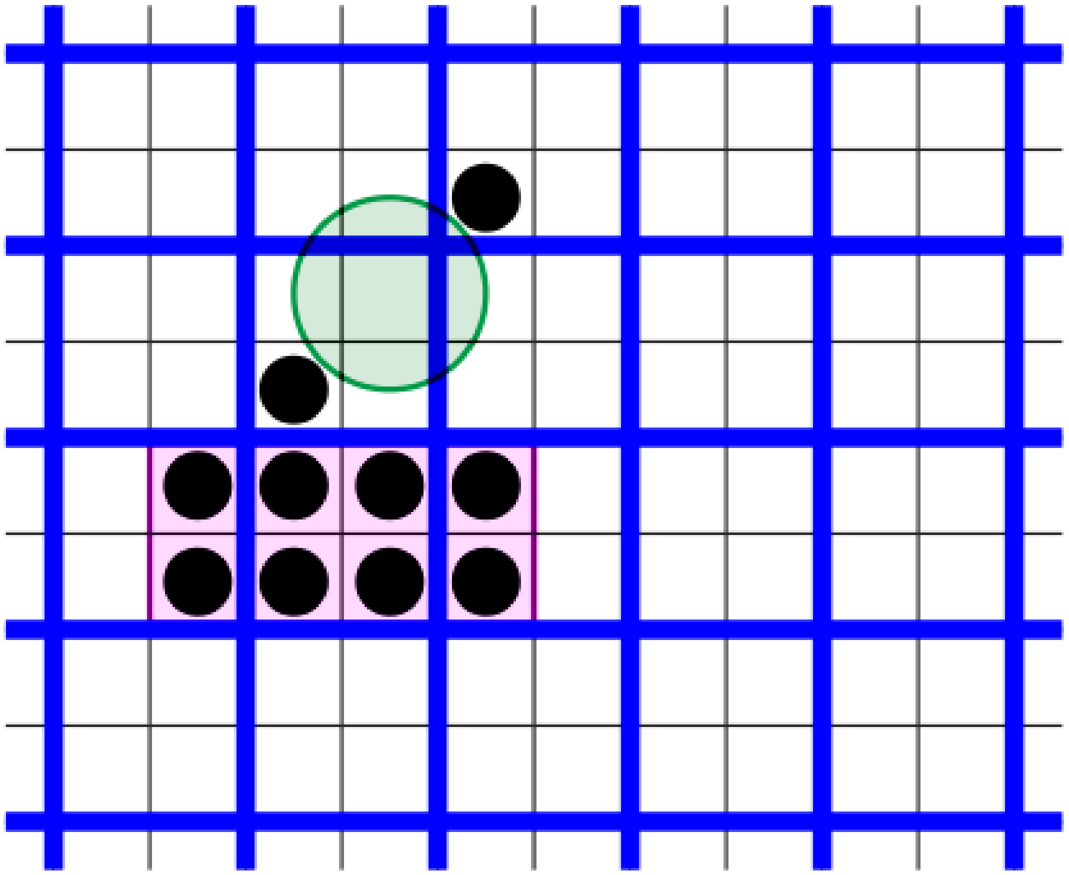
One can make static walls from which balls bounce:



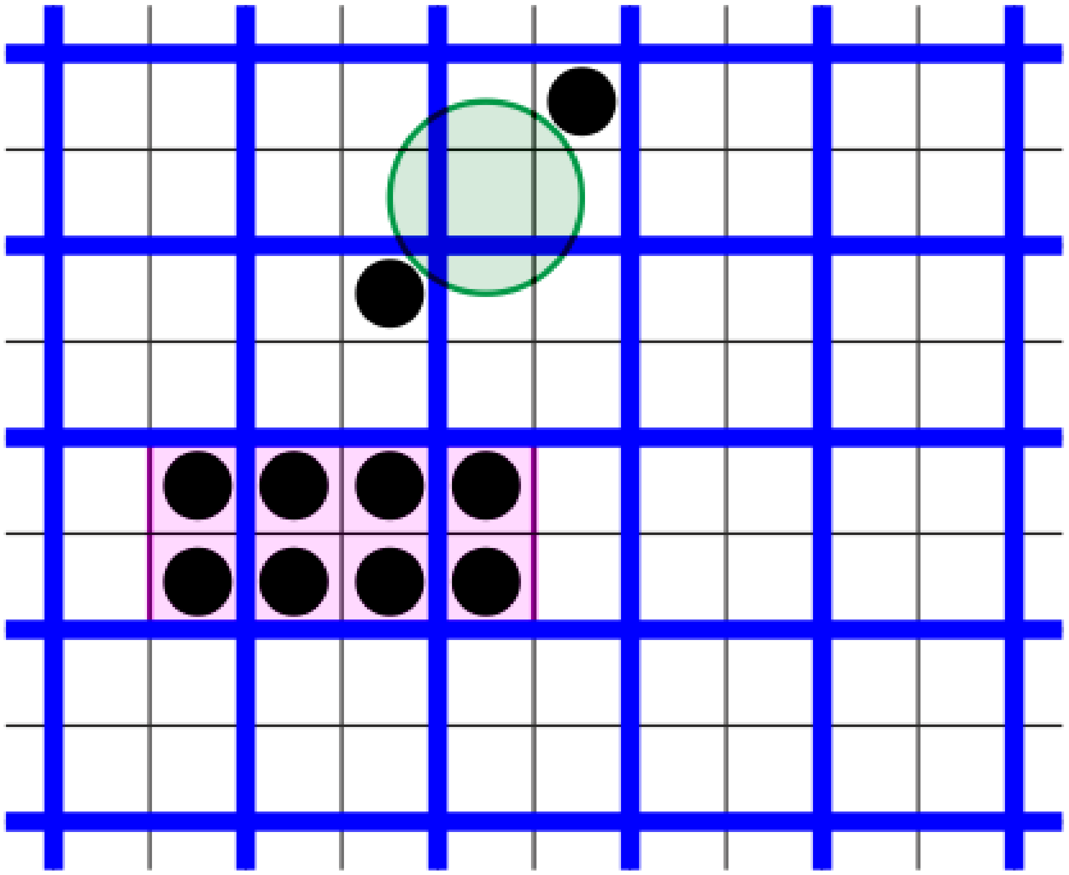
One can make static walls from which balls bounce:



One can make static walls from which balls bounce:

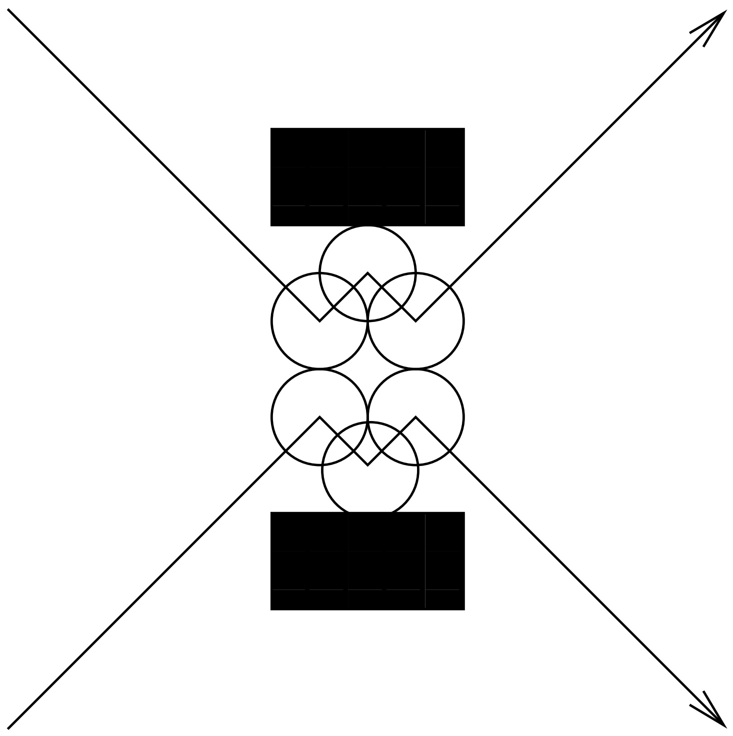


One can make static walls from which balls bounce:

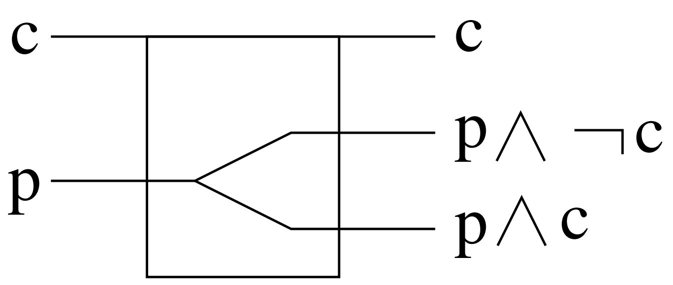


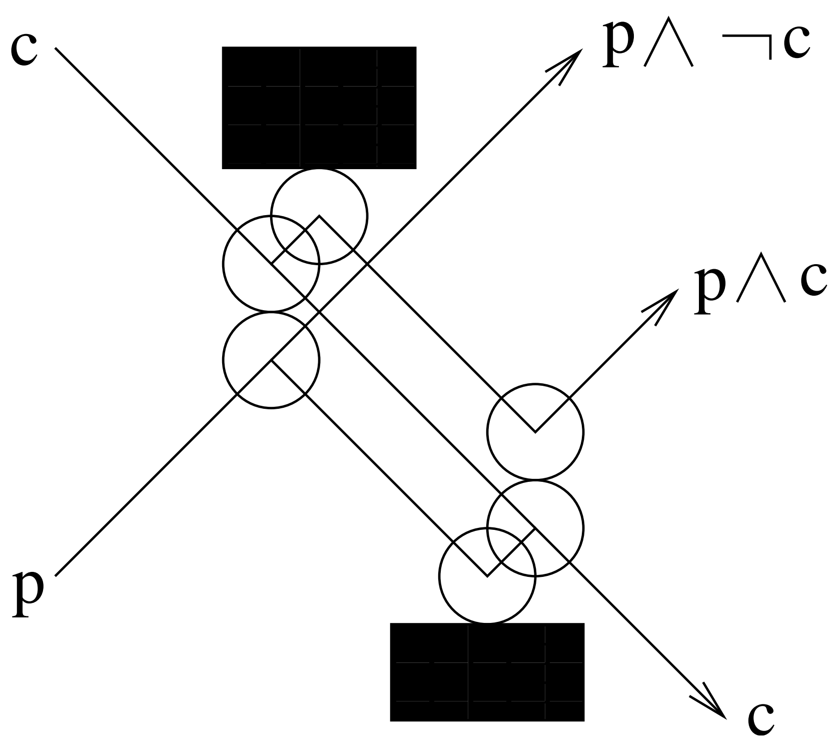
Using walls one can turn wires (=potential trajectories) and delays can be created by increasing the length of the wire by additional turns.

Wires can **cross**:



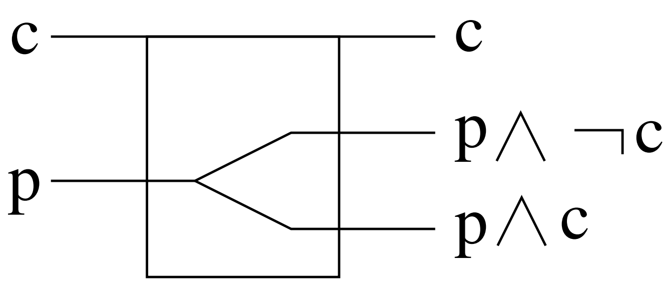
A **switch gate** performs conditional routing:

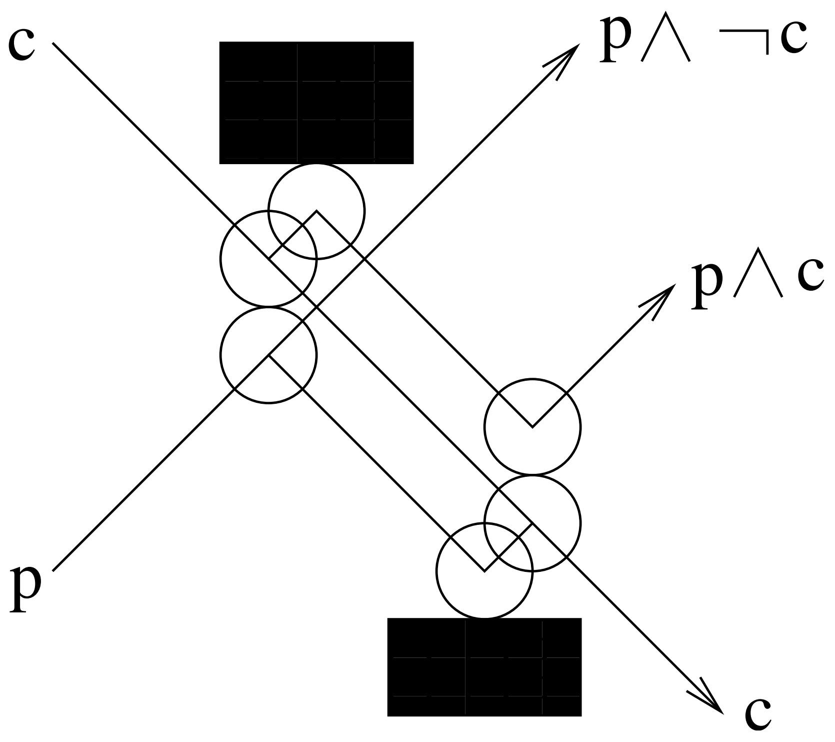




We can use the switch gate in the opposite direction to select between two inputs, under the condition that the non-selected input is 0.

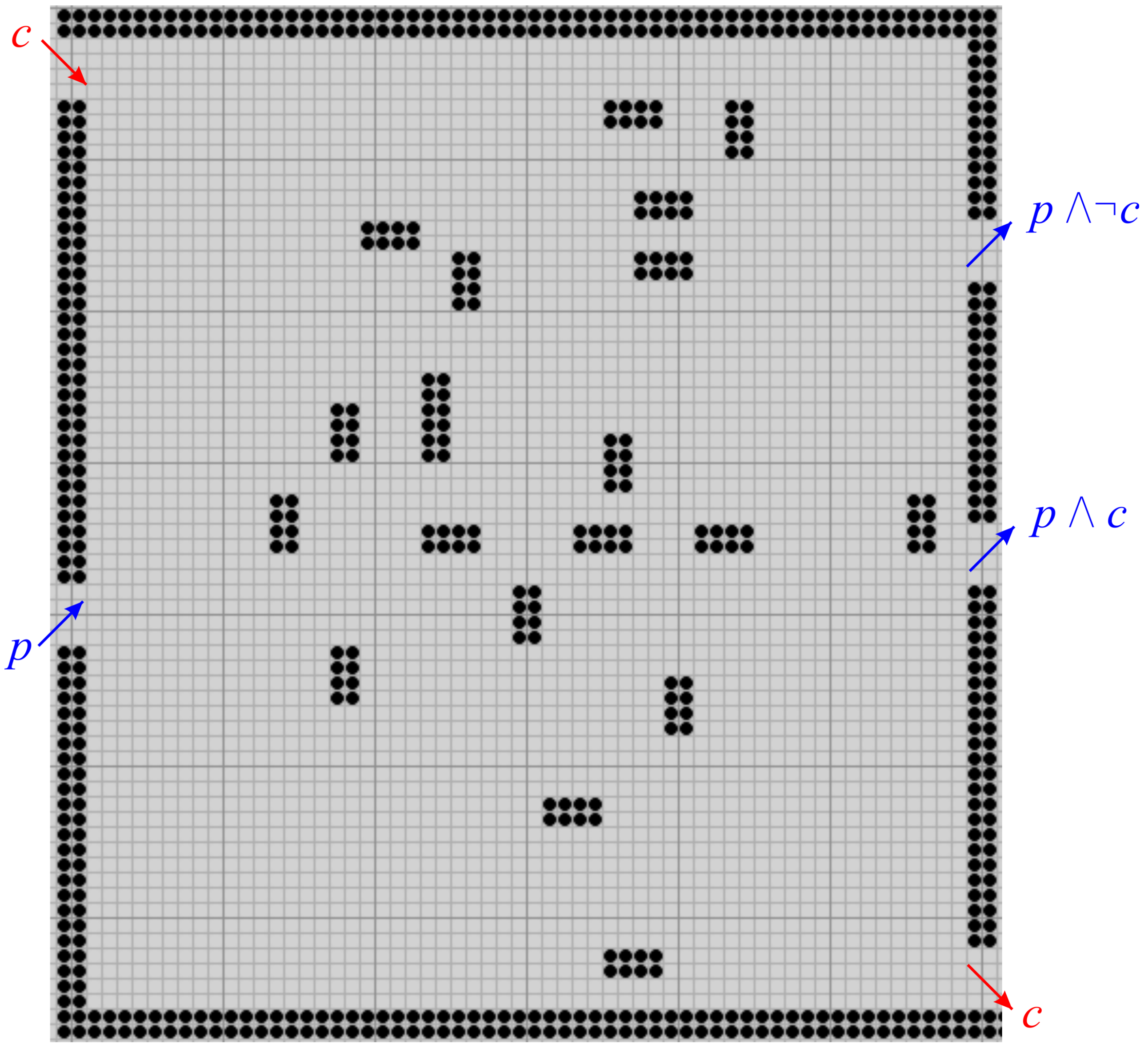
A **switch gate** performs conditional routing:



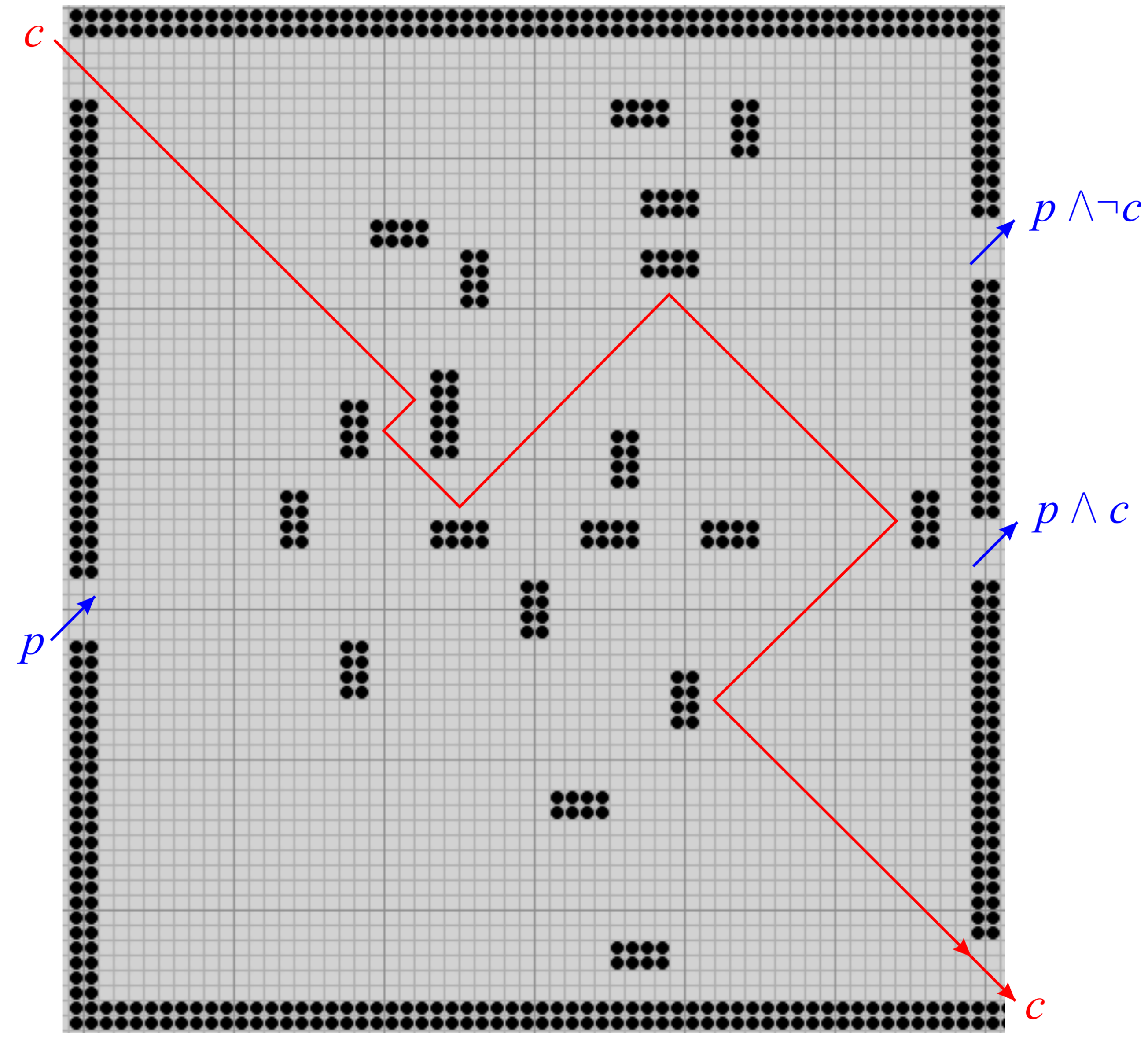


We can use the switch gate in the opposite direction to select between two inputs, under the condition that the non-selected input is 0.

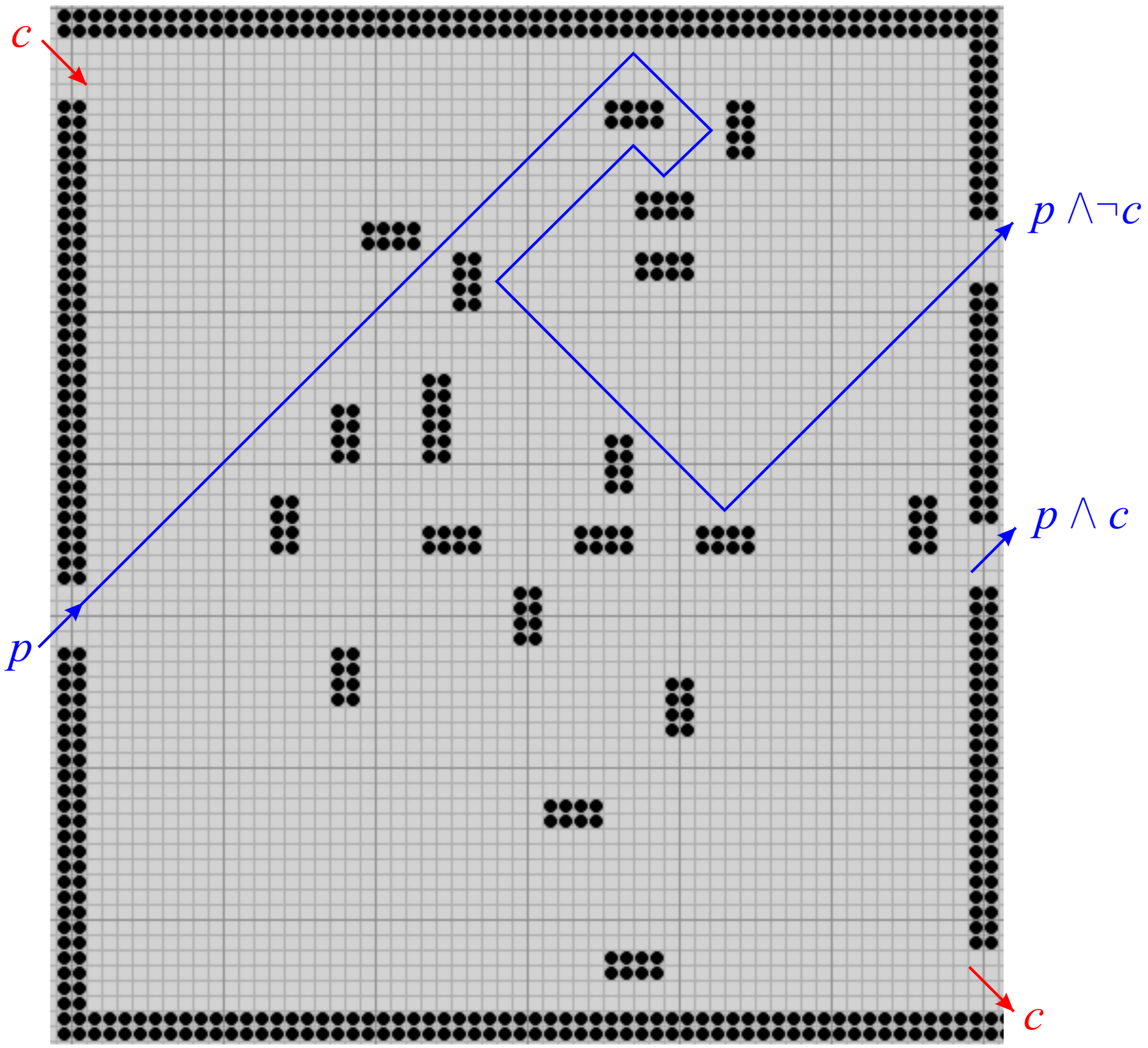
The switch gate above works with "hard" balls (=bouncing does not cause delays) but with the "soft" balls as in BBMCA the timing of the output c depends on whether $p = 0$ or $p = 1$.



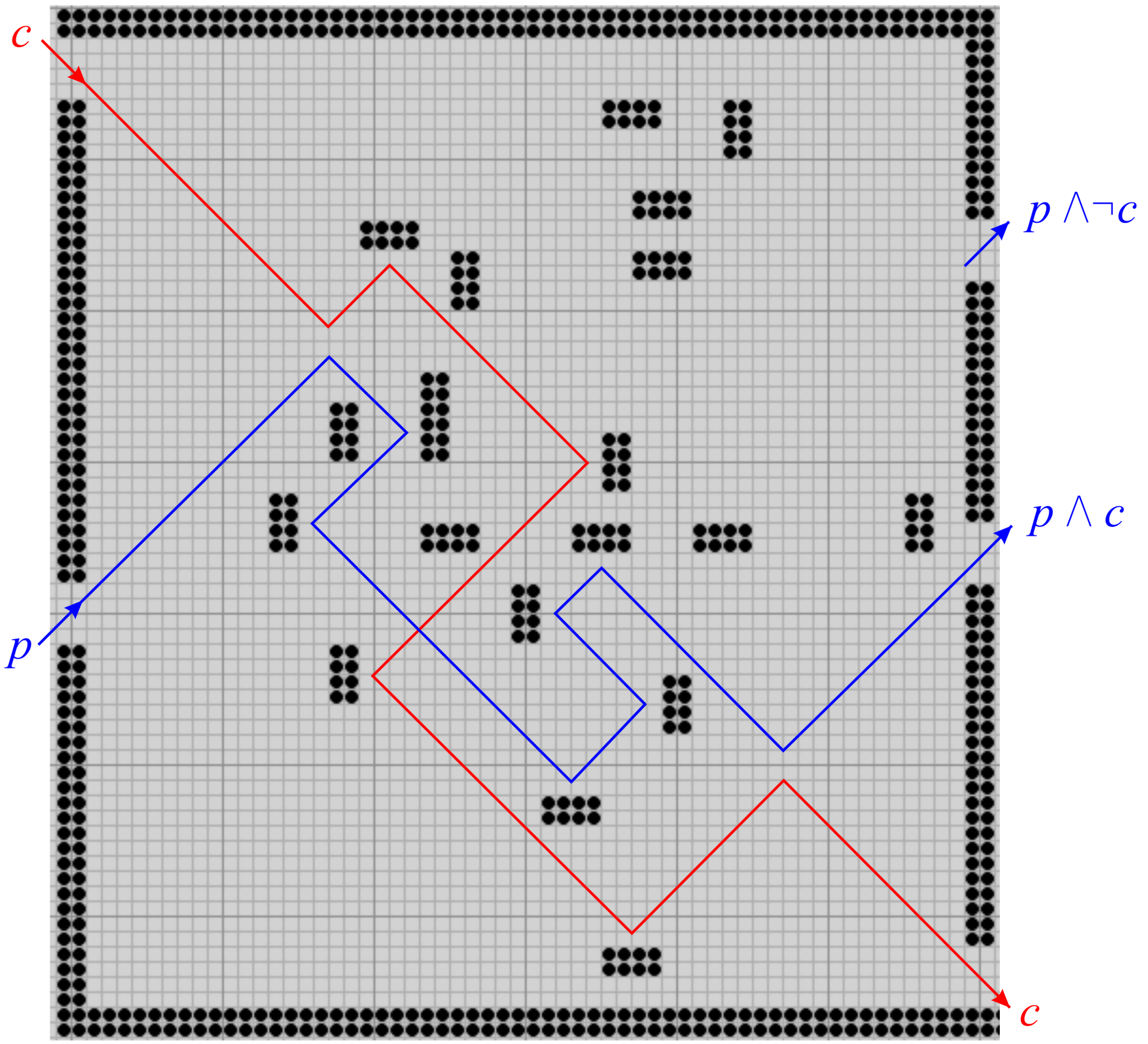
Here's a switch gate that works with BBMCA. The delay from input to output is always the same 100 generations.



The trajectory of $c = 1$ when $p = 0$.

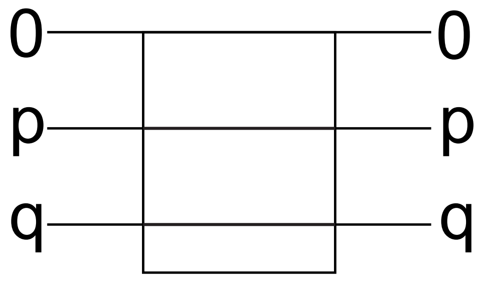


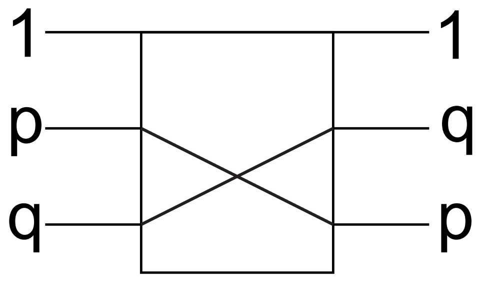
The trajectory of $p = 1$ when $c = 0$.



The trajectories when both $c = 1$ and $p = 1$.

The **Fredkin gate** is a controlled switch gate with three inputs and corresponding outputs. If the control wire is $c = 1$ then the the other two signals are swapped, otherwise not:





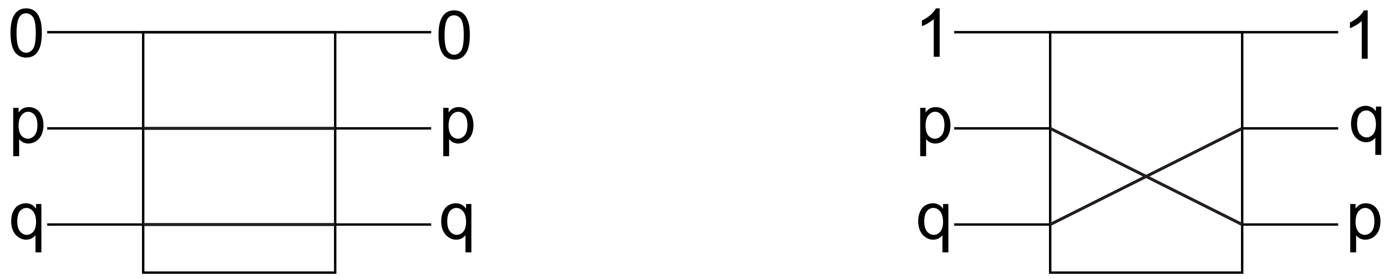
The Fredkin gate is universal as it implements AND, NOT and OR:

$$q_{in} = 0 \qquad \Longrightarrow \qquad q_{out} = c_{in} \text{ AND } p_{in},$$

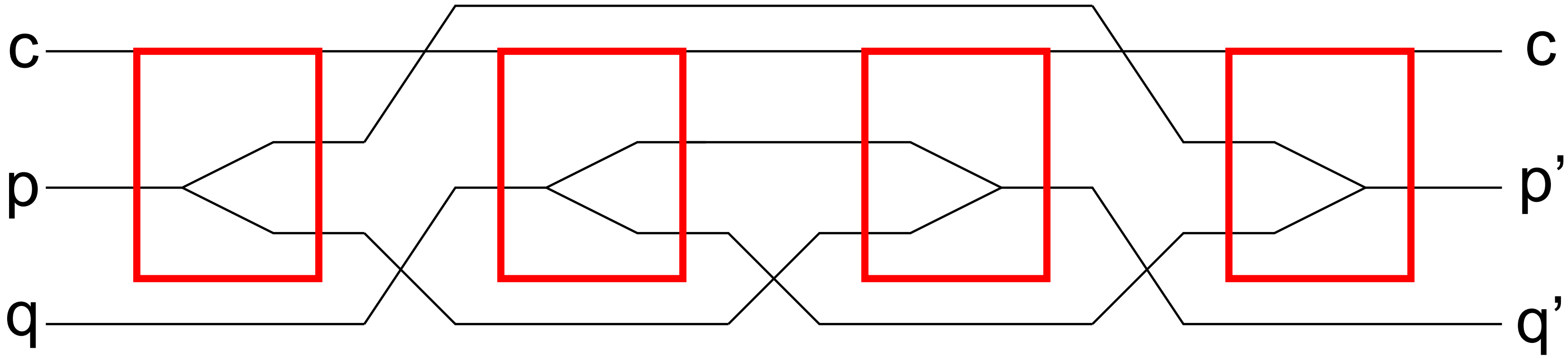
$$q_{in} = 1 \qquad \Longrightarrow \qquad p_{out} = c_{in} \text{ OR } p_{in},$$

$$q_{in} = 1, p_{in} = 0 \qquad \Longrightarrow \qquad q_{out} = \text{ NOT } c_{in},$$

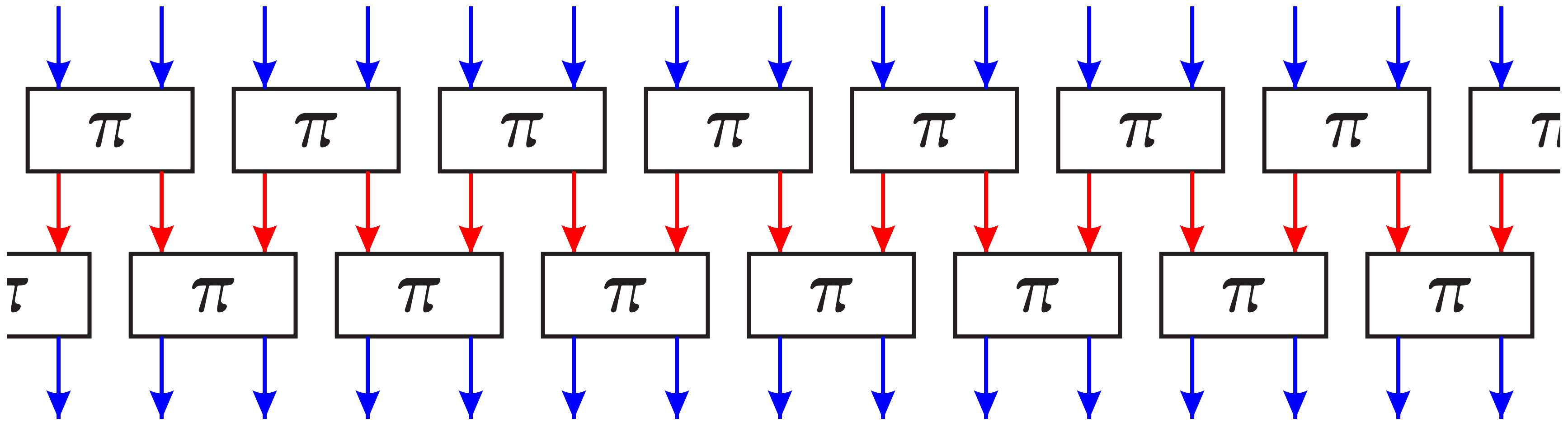
The **Fredkin gate** is a controlled switch gate with three inputs and corresponding outputs. If the control wire is $c = 1$ then the the other two signals are swapped, otherwise not:



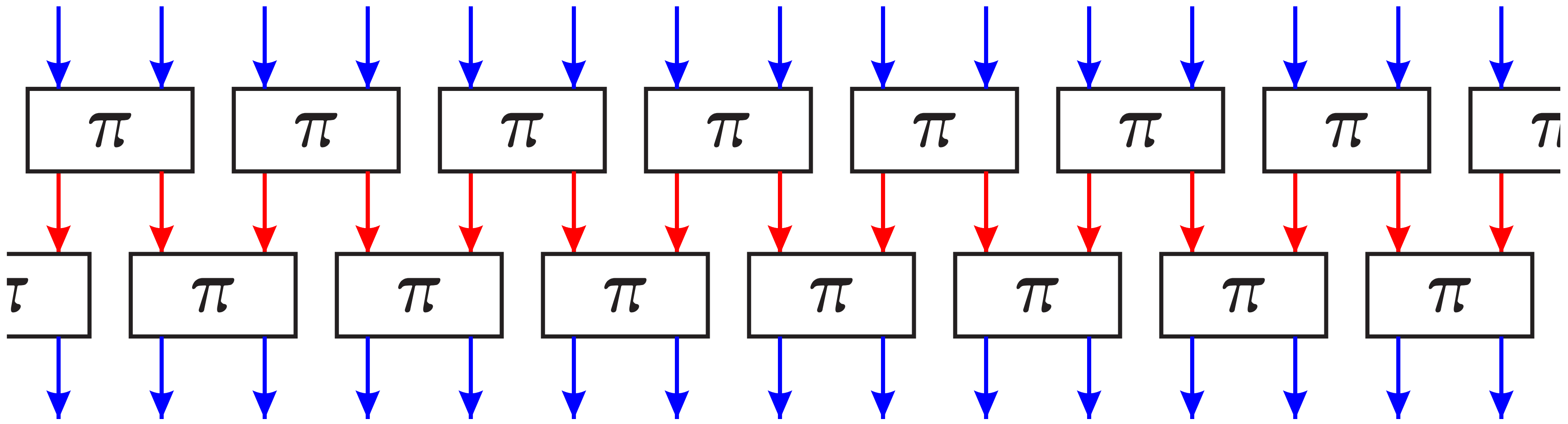
The Fredkin gate can be implemented using four switch gates, two of which are used in the opposite direction:



Remark: The Margolus neighborhood can be used in other dimensions than $d = 2$. For example, in the one-dimensional case one partitions $\mathbb{Z}$ into segments of length two, applies a bijective function $\pi : S^2 \longrightarrow S^2$ in each segment, and repeats the operation using a partitioning that is translated by one cell:



Remark: The Margolus neighborhood can be used in other dimensions than $d = 2$. For example, in the one-dimensional case one partitions $\mathbb{Z}$ into segments of length two, applies a bijective function $\pi : S^2 \longrightarrow S^2$ in each segment, and repeats the operation using a partitioning that is translated by one cell:



One can also use the idea of the Margolus neighborhood with other partitions: Divide the space in any regular manner and apply locally a bijection in each part independently of other parts. For the next round the partition is changed to allow information propagation in space.