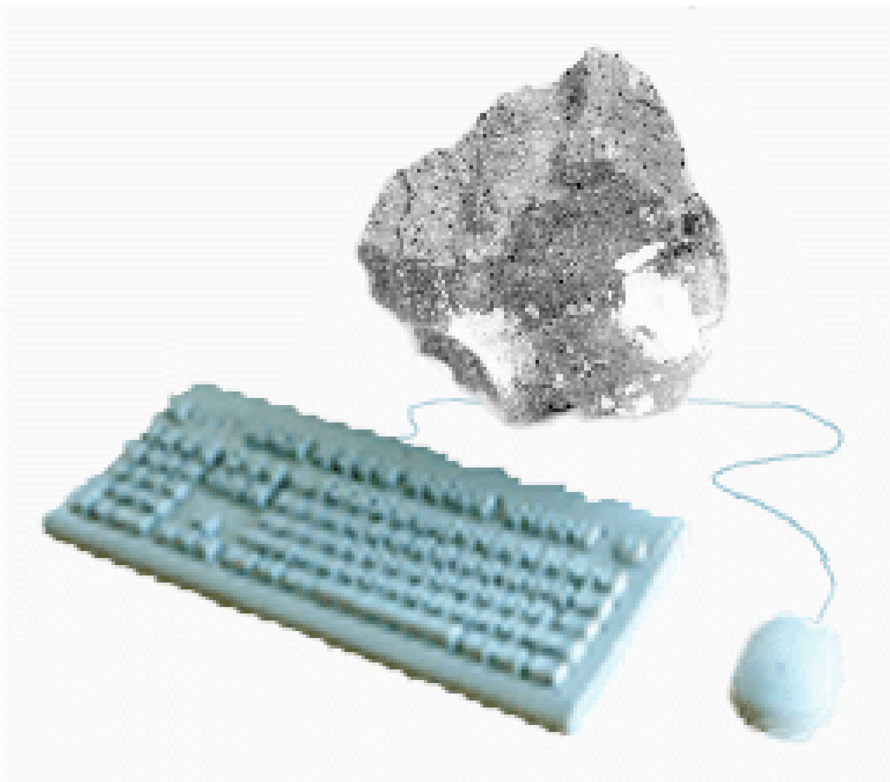
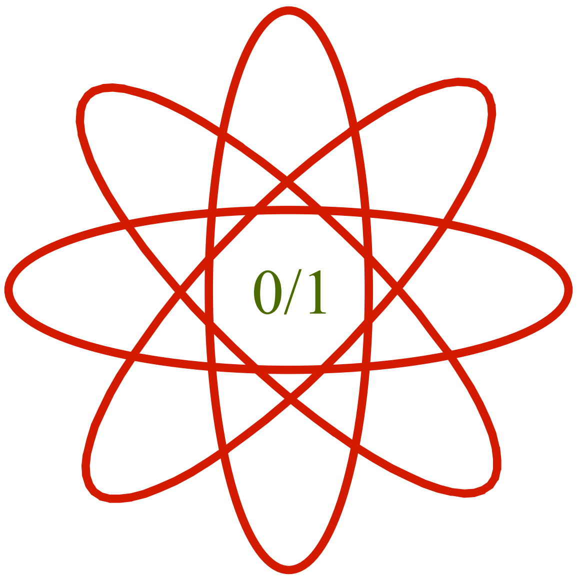
Computing Beyond Silicon Summer School

Physics becomes the computer



Norm Margolus

Physics becomes the computer



Emulating Physics

- » *Finite-state, locality, invertibility, and conservation laws*

Physical Worlds

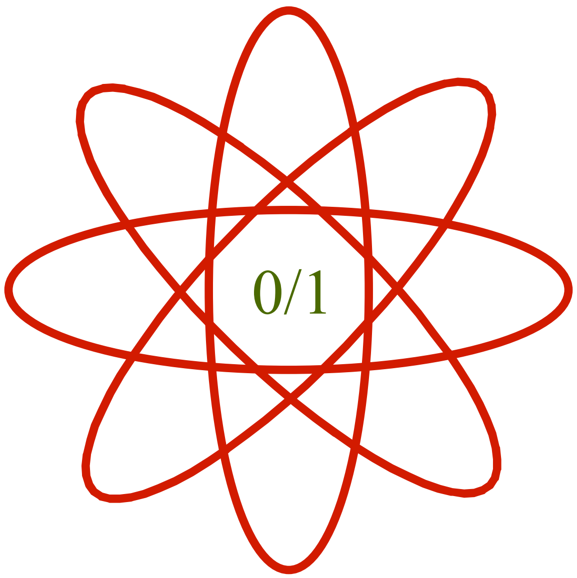
- » *Incorporating comp-universality at small and large scales*

Spatial Computers

- » *Architectures and algorithms for large-scale spatial computations*

Nature as Computer

- » *Physical concepts enter CS and computer concepts enter Physics*

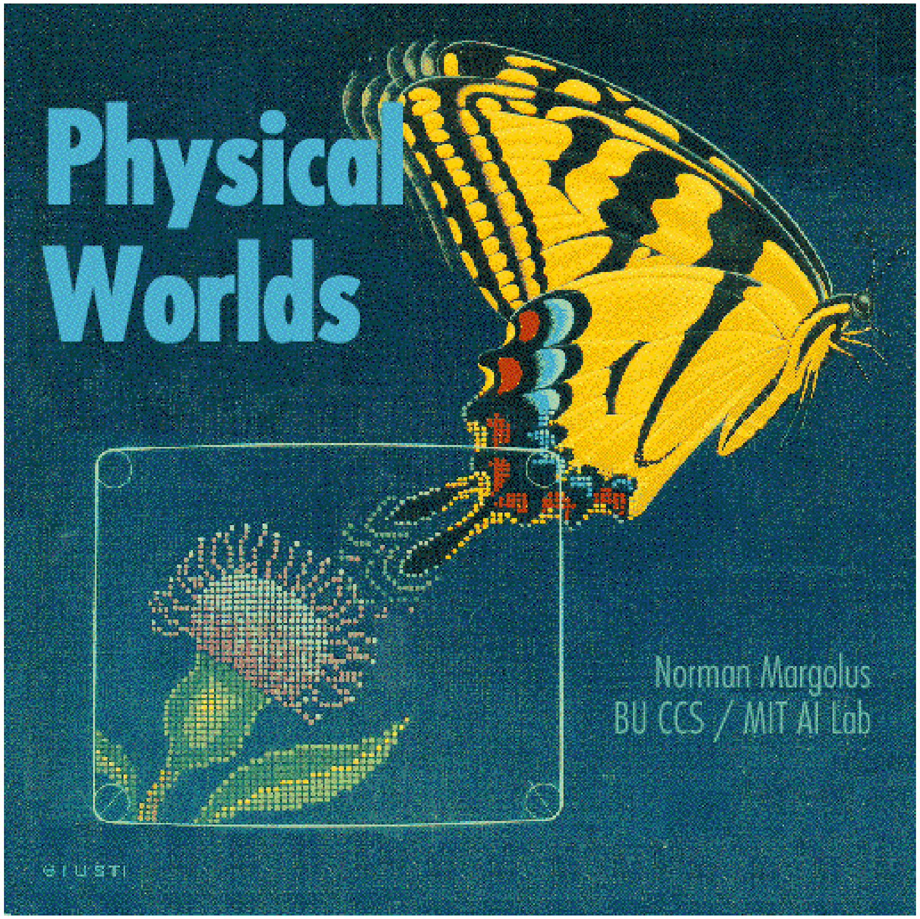


Emulating Physics

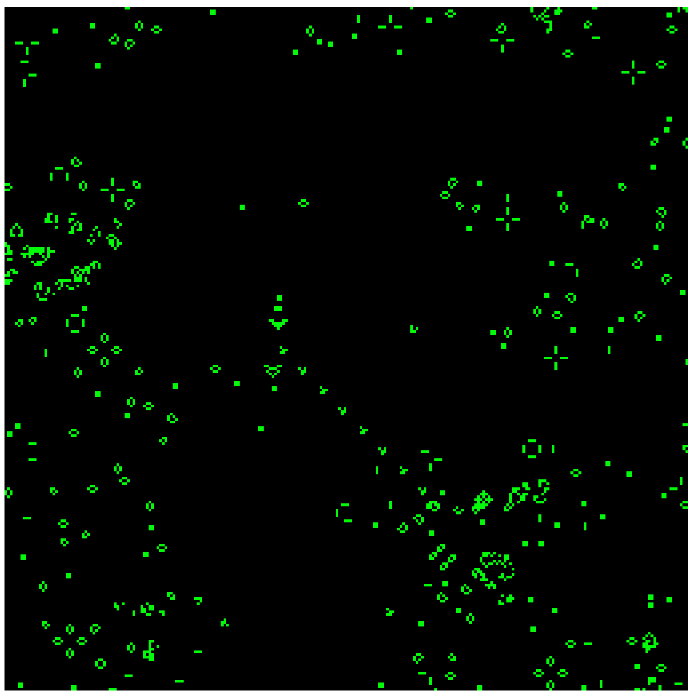
Why emulate physics?

- Comp must adapt to microscopic physics
- Comp models may help us understand nature
- Rich dynamics

- Start with locality: *Cellular Automata*

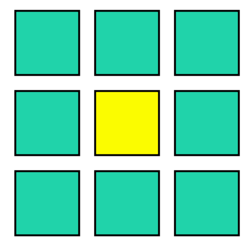


Conway's "Game of Life"



256x256 region of a larger grid.
Glider gun inserted near middle.

In each 3x3 neighborhood, count the ones, not including the center:

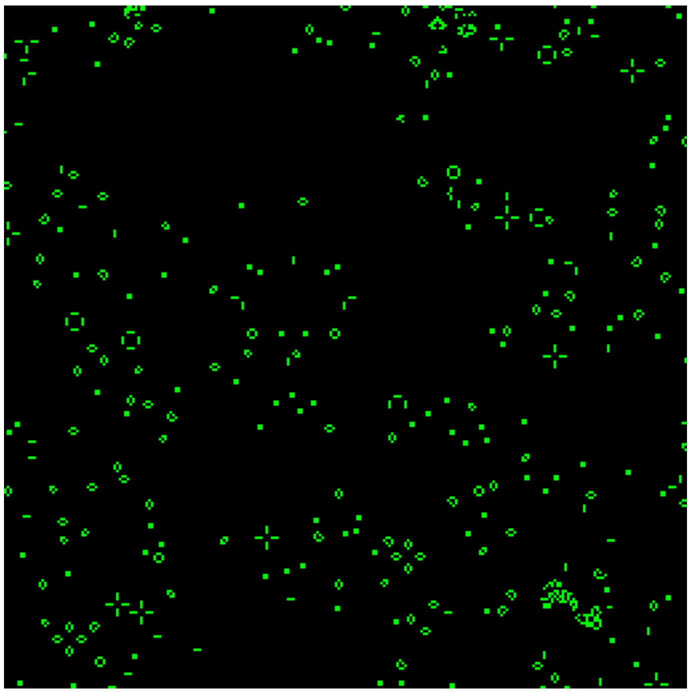


If total = 2: center unchanged

If total = 3: center becomes 1

Else: center becomes 0

Conway's "Game of Life"



256x256 region of a larger grid. About 1500 steps later.

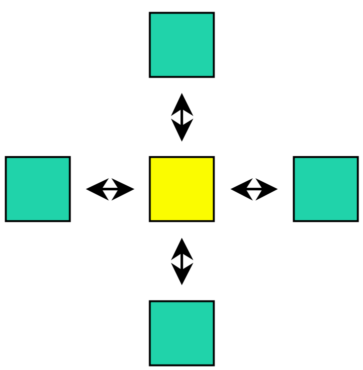
- Captures physical locality and finite-state

But,

- Not reversible (doesn't map well onto microscopic physics)
- No conservation laws (nothing like momentum or energy)
- No interesting large-scale behavior

Reversibility & other conservations

- Reversibility is conservation of information
- Why does exact conservation seem hard?



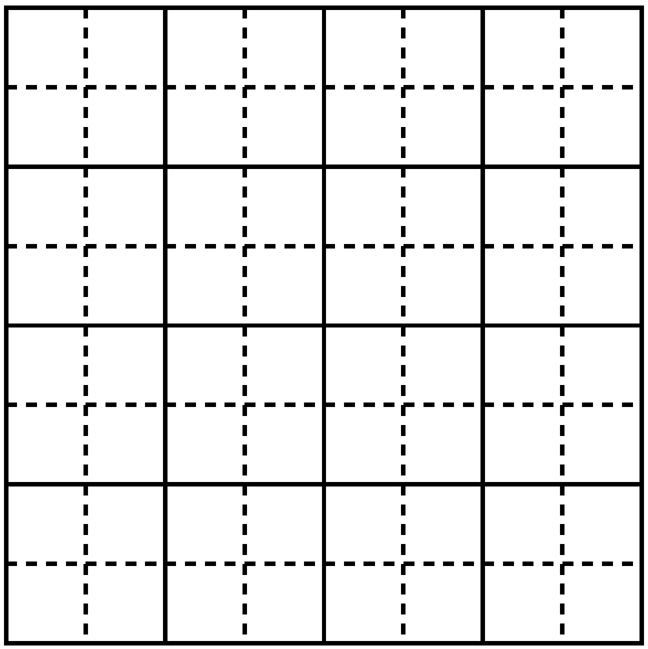
The same information is visible at multiple positions

For rev, one n^{th} of the neighbor info must be left at the center

Adding conservations

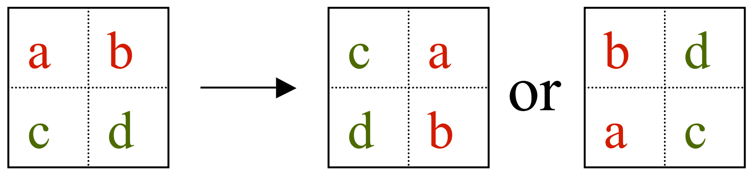
- With traditional CA's, conservations are a *non-local* property of the dynamics.
- Simplest solution: redefine CA's so that conservation is a *manifestly local* property
- *CA ≡ regular computation in space & time*
  - » Regular in space: repeated structure
  - » Regular in time: repeated sequence of steps

Diffusion rule

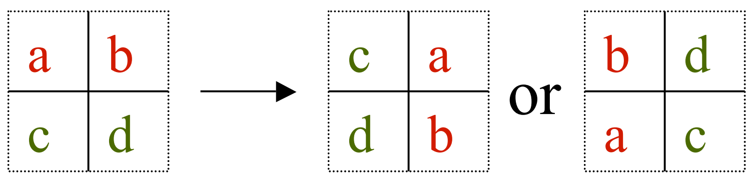


Use 2x2 blockings. Use solid blocks on even time steps, use dotted blocks on odd steps.

Even steps: *rotate cw or ccw*

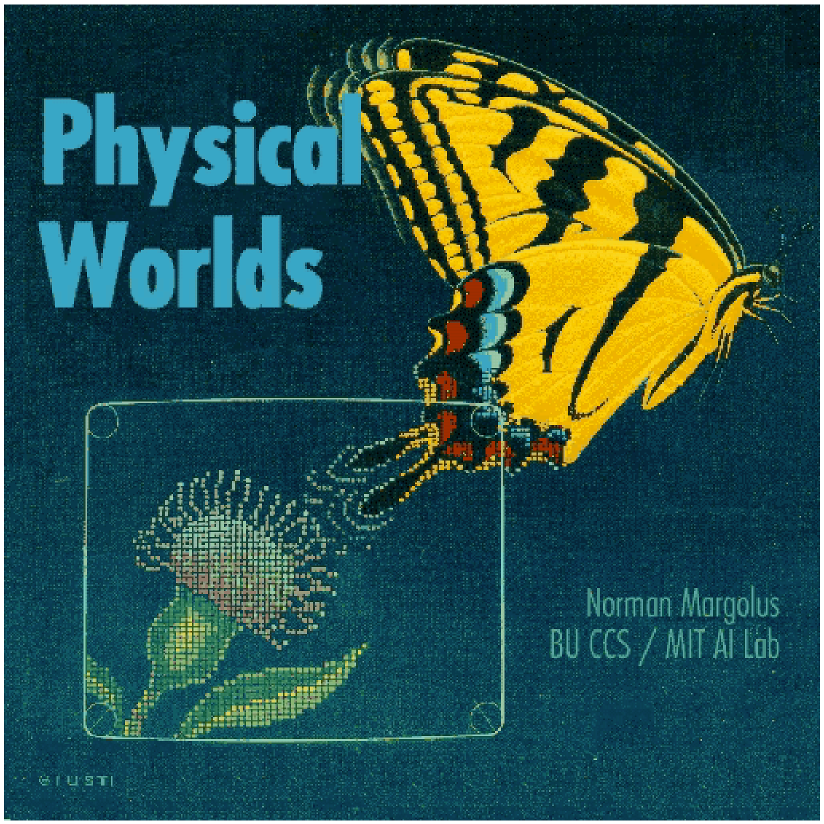


Odd steps: *rotate cw or ccw*



We "randomly" choose to rotate blocks 90-degrees cw or ccw (we actually use a fixed sequence of choices for each spot).

Diffusion rule



Even steps: *rotate cw or ccw*

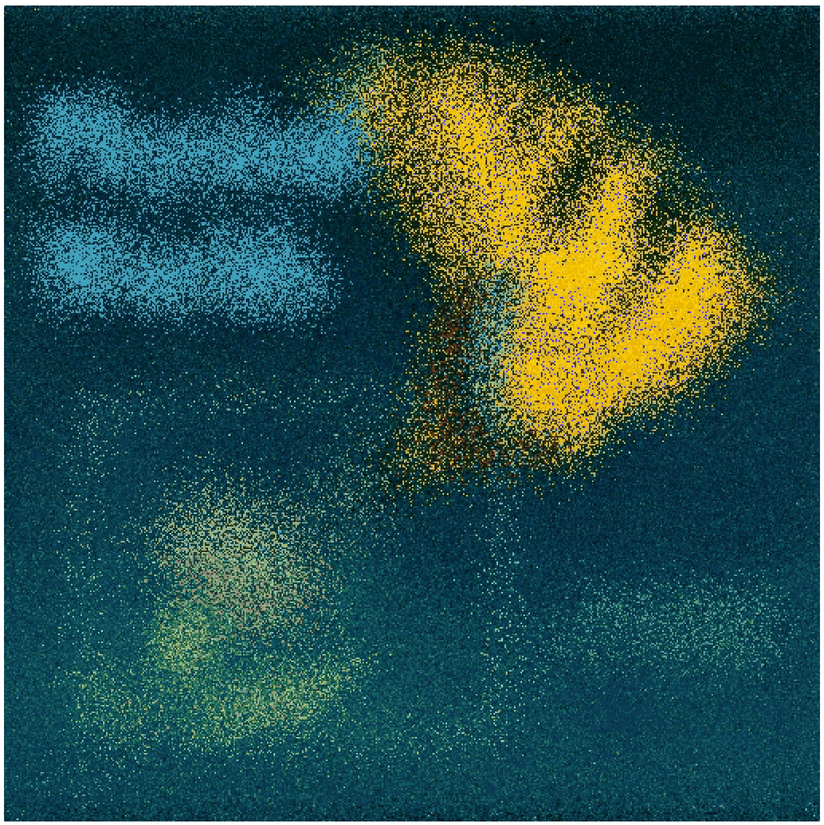
a	b	→	c	a	or	b	d
c	d		d	b		a	c

Odd steps: *rotate cw or ccw*

a	b	→	c	a	or	b	d
c	d		d	b		a	c

We "randomly" choose to rotate blocks 90-degrees cw or ccw (we actually use a fixed sequence of choices for each spot).

Diffusion rule



Even steps: *rotate cw or ccw*

a	b
c	d

→

c	a
d	b

or

b	d
a	c

Odd steps: *rotate cw or ccw*

a	b
c	d

→

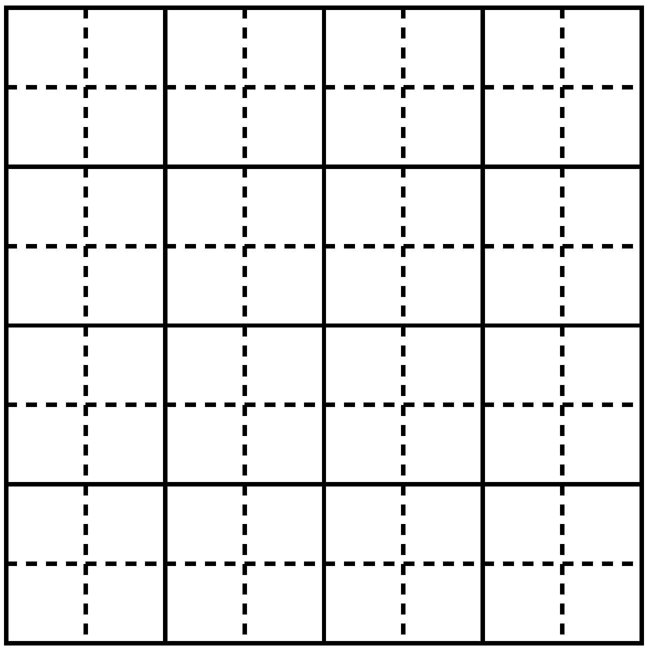
c	a
d	b

or

b	d
a	c

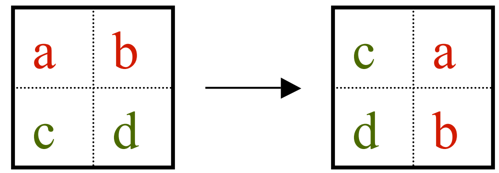
We "randomly" choose to rotate blocks 90-degrees cw or ccw (we actually use a fixed sequence of choices for each spot).

TM Gas rule

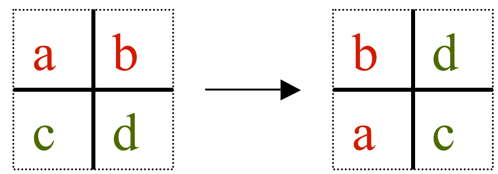


Use 2x2 blockings. Use solid blocks on even time steps, use dotted blocks on odd steps.

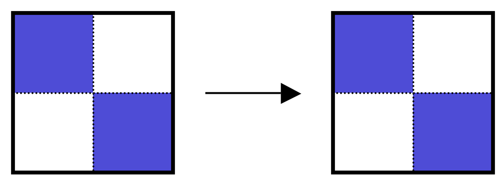
Even steps: *rotate cw*



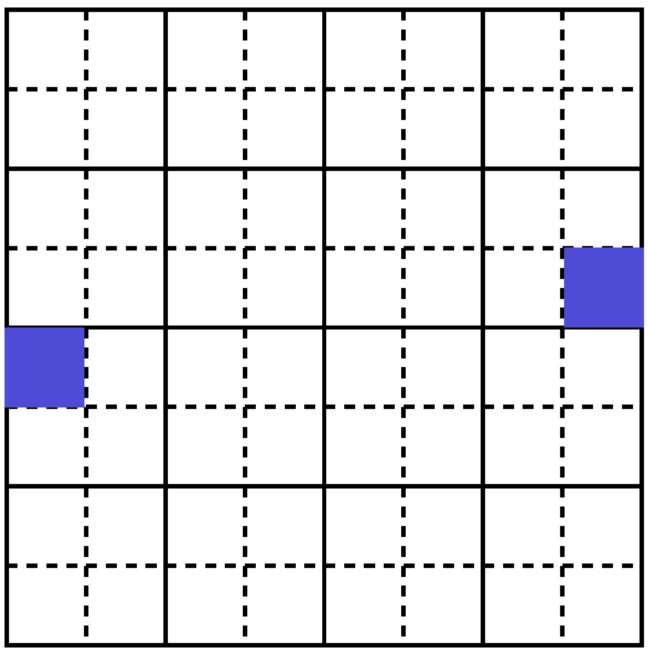
Odd steps: *rotate ccw*



Except: *2 ones on diag, nc*

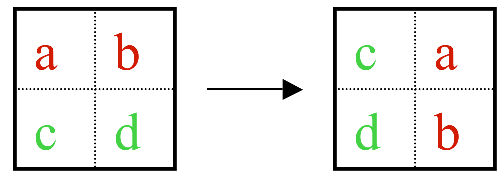


TM Gas rule

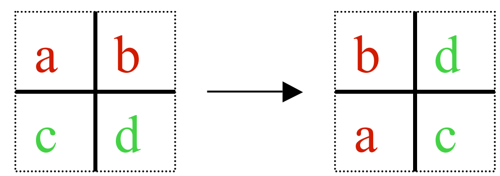


Even step: update solid blocks.

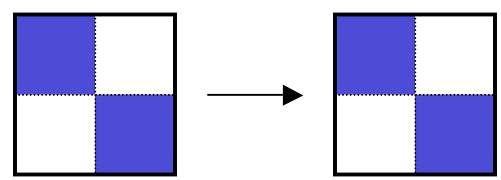
Even steps: *rotate cw*



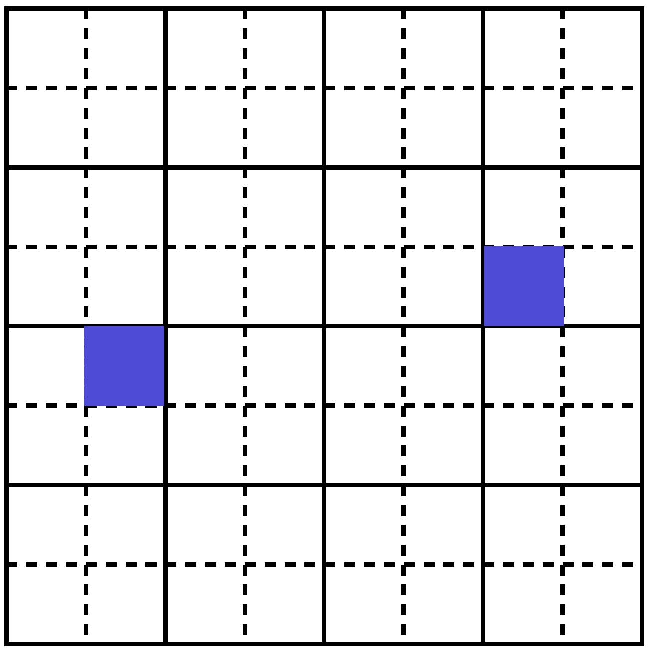
Odd steps: *rotate ccw*



Except: *2 ones on diag, nc*

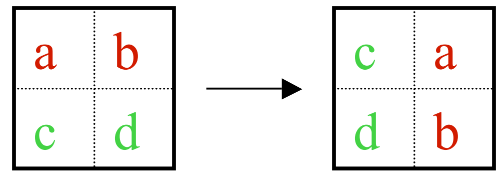


TM Gas rule

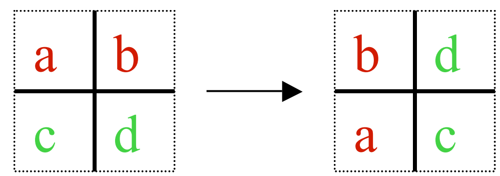


Odd step: update dotted blocks

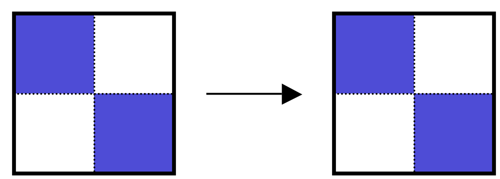
Even steps: *rotate cw*



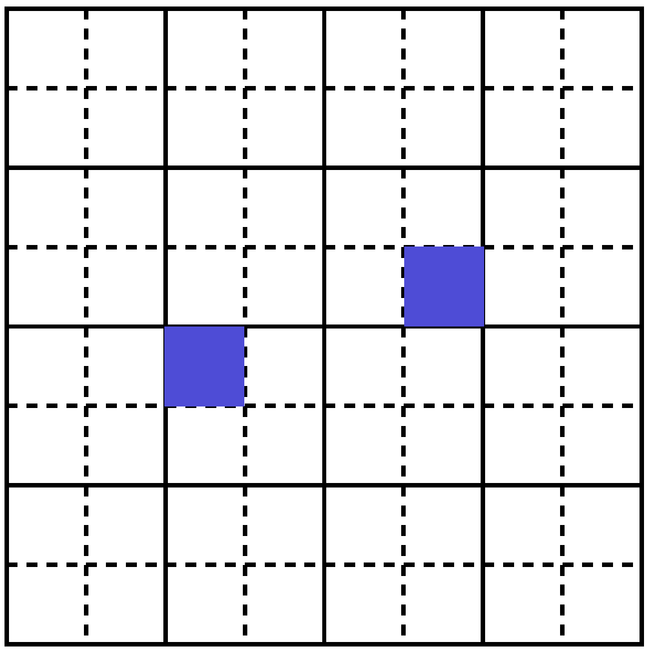
Odd steps: *rotate ccw*



Except: *2 ones on diag, nc*

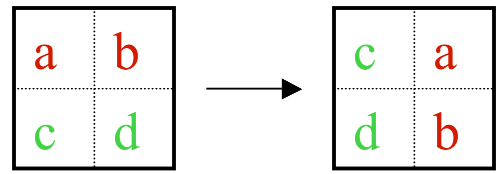


TM Gas rule

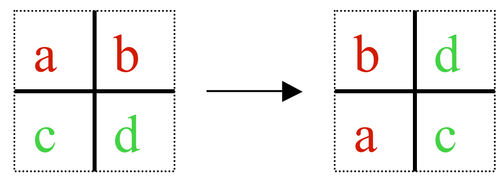


Even step: update solid blocks

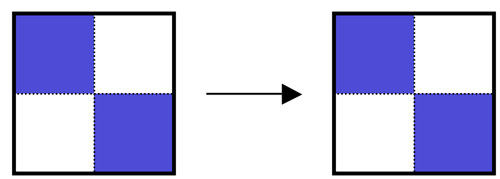
Even steps: *rotate cw*



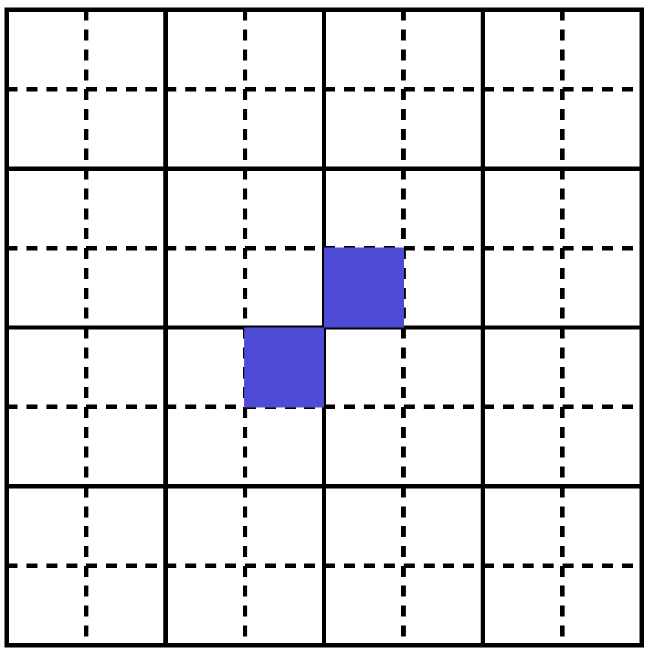
Odd steps: *rotate ccw*



Except: *2 ones on diag, nc*

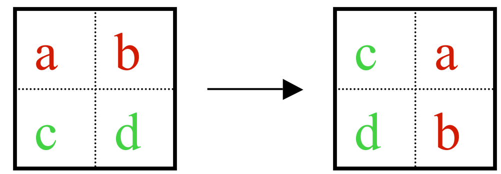


TM Gas rule

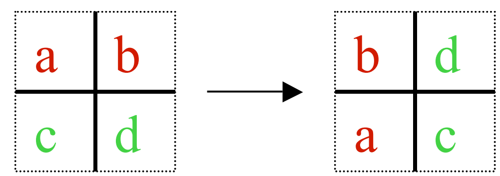


Odd step: update dotted blocks

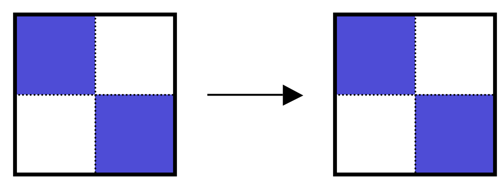
Even steps: *rotate cw*



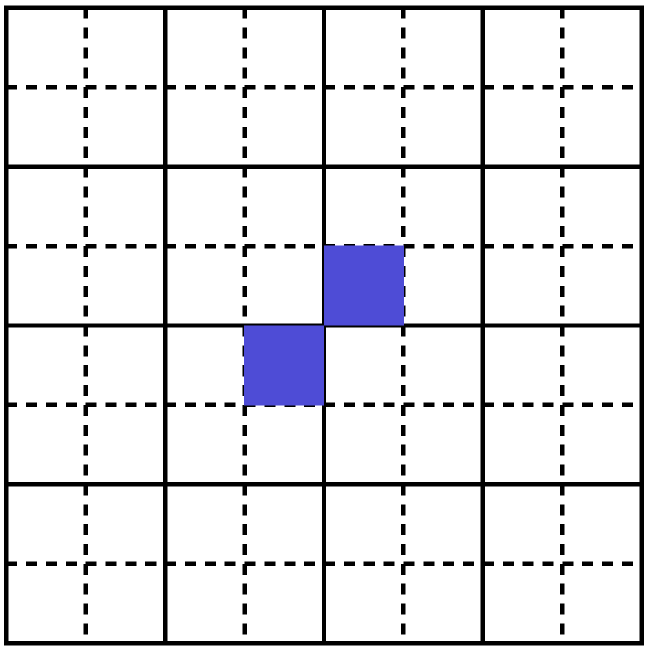
Odd steps: *rotate ccw*



Except: *2 ones on diag, nc*

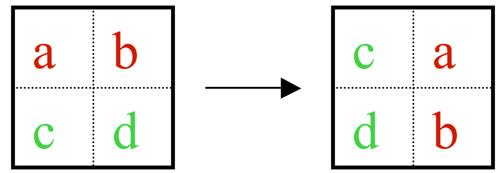


TM Gas rule

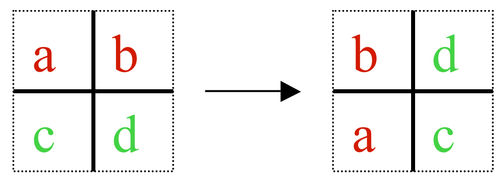


Even step: update solid blocks

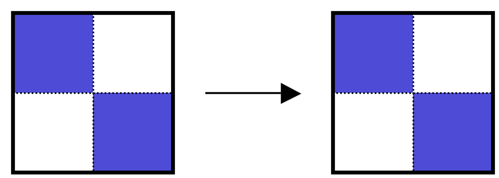
Even steps: *rotate cw*



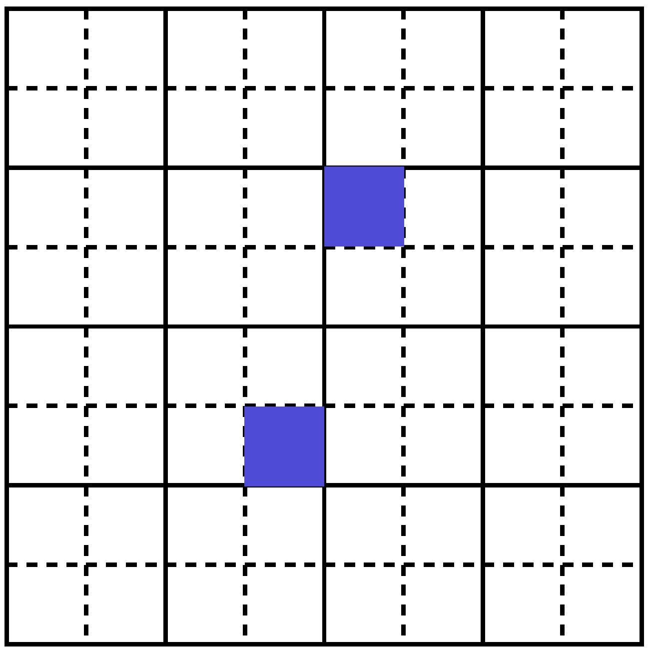
Odd steps: *rotate ccw*



Except: *2 ones on diag, nc*

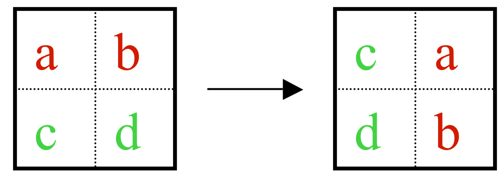


TM Gas rule

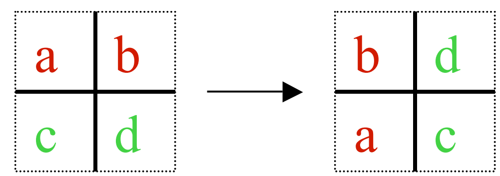


Odd step: update dotted blocks

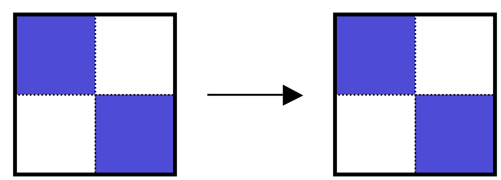
Even steps: *rotate cw*



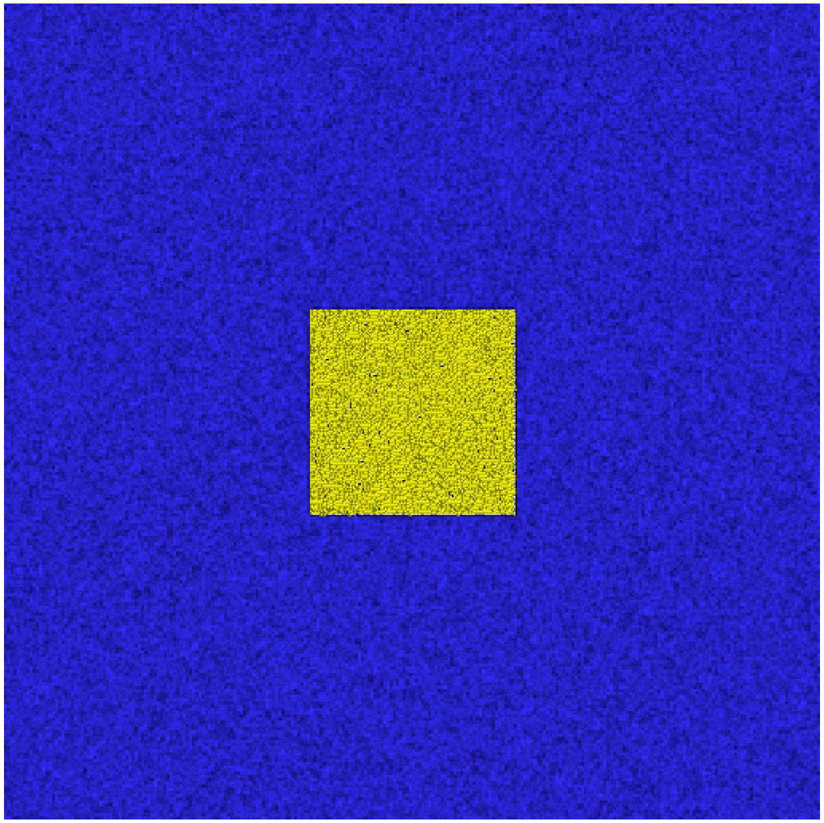
Odd steps: *rotate ccw*



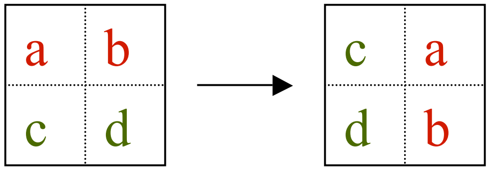
Except: *2 ones on diag, nc*



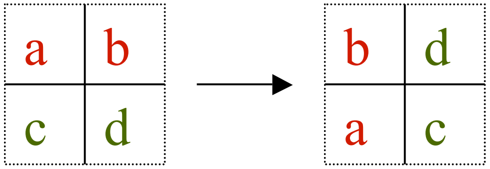
TM Gas rule



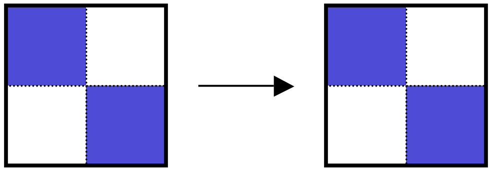
Even steps: *rotate cw*



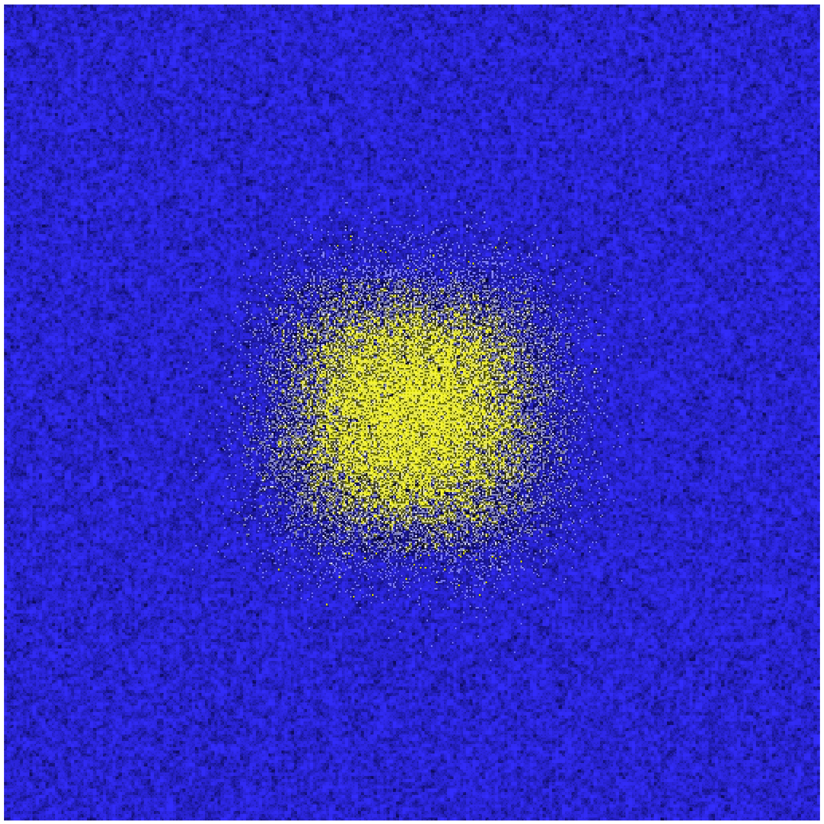
Odd steps: *rotate ccw*



Except: *2 ones on diag, nc*



TM Gas rule



Even steps: *rotate cw*

a	b
c	d

→

c	a
d	b

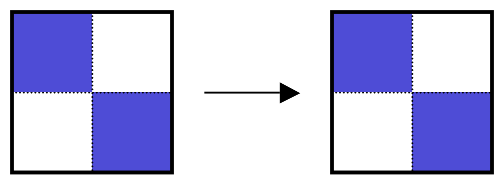
Odd steps: *rotate ccw*

a	b
c	d

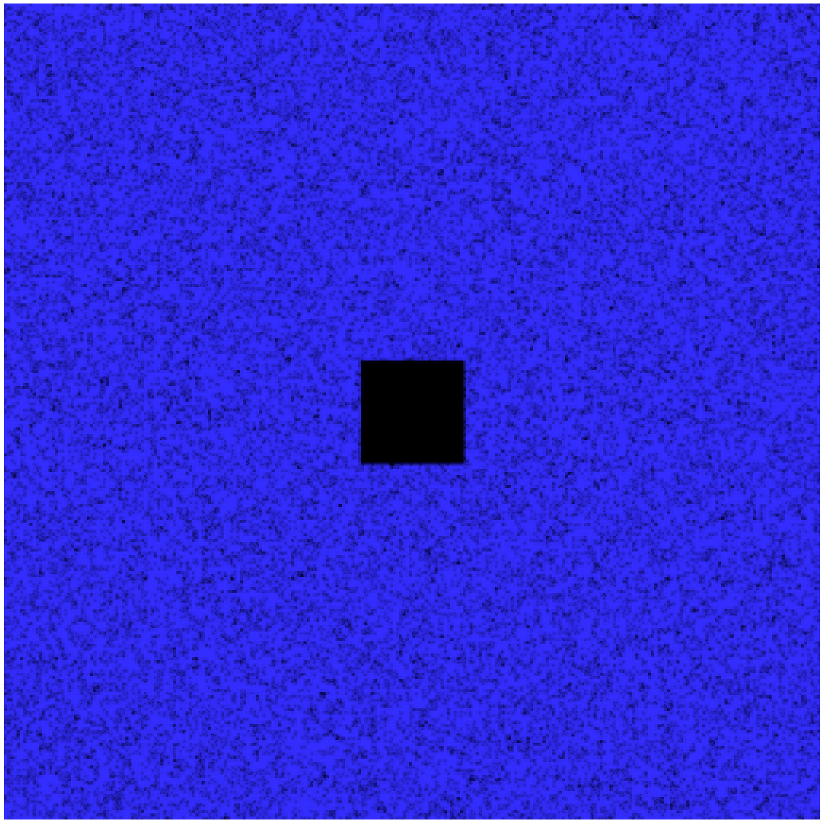
→

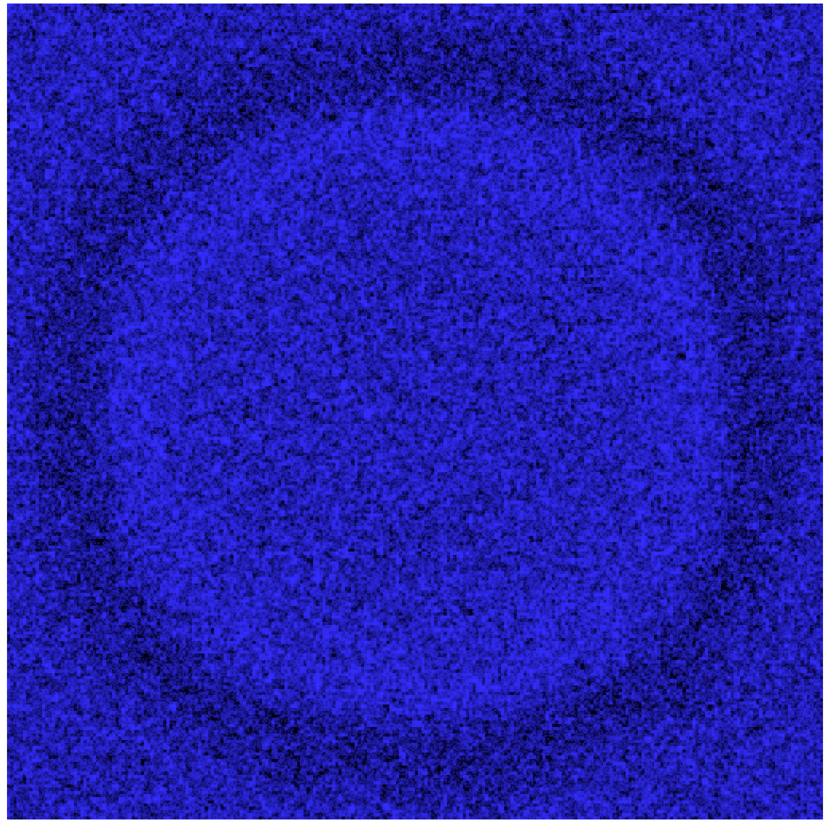
b	d
a	c

Except: *2 ones on diag, nc*

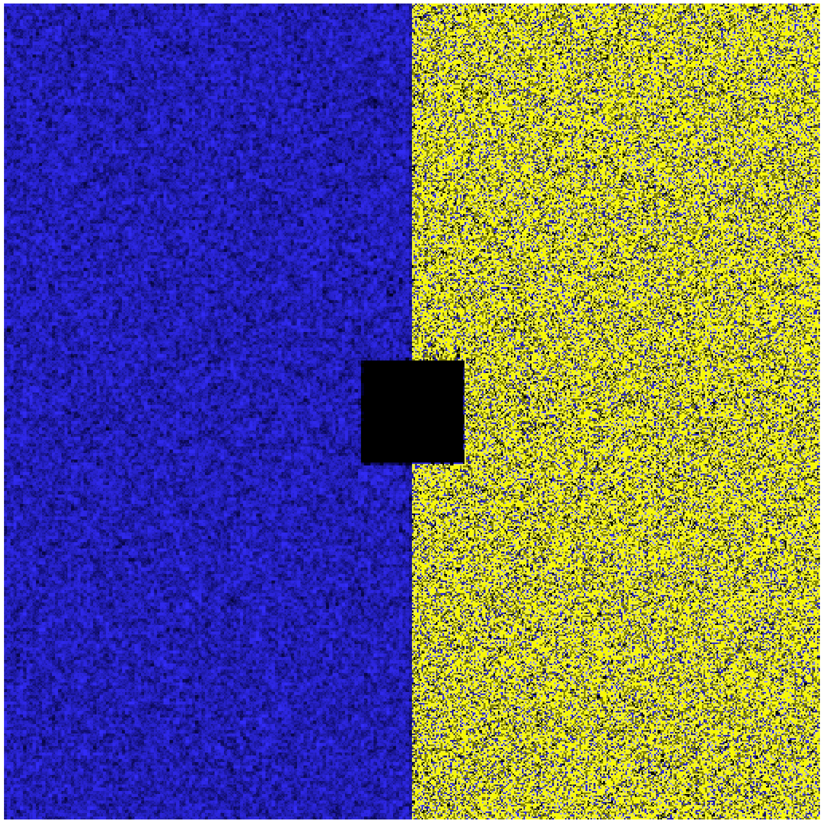


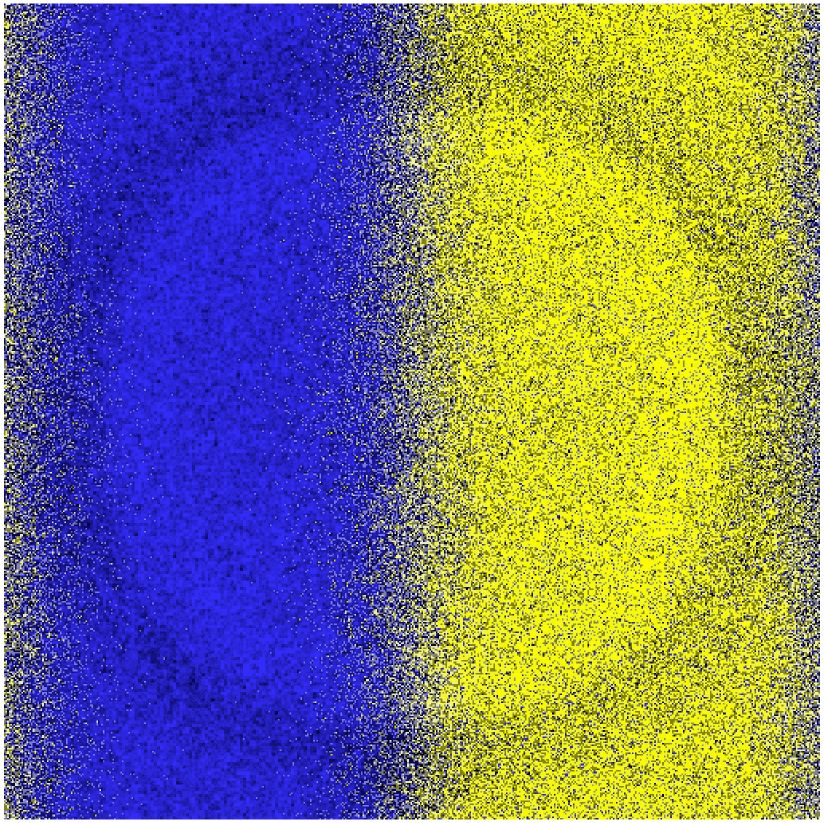
TM Gas rule



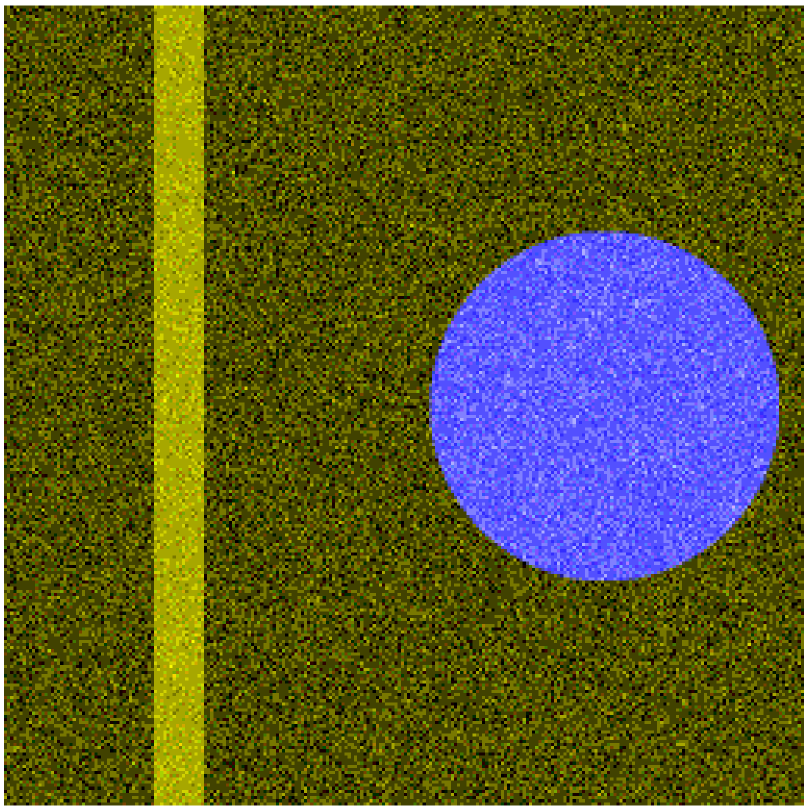


TM Gas rule



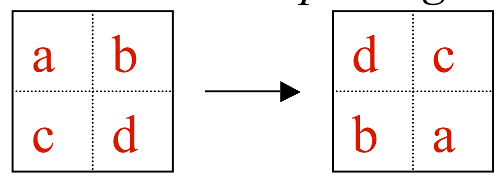


Lattice gas refraction

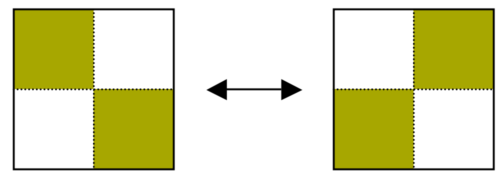


- *Half the time:* HPP gas rule everywhere:

Even & odd: *swap along diags*

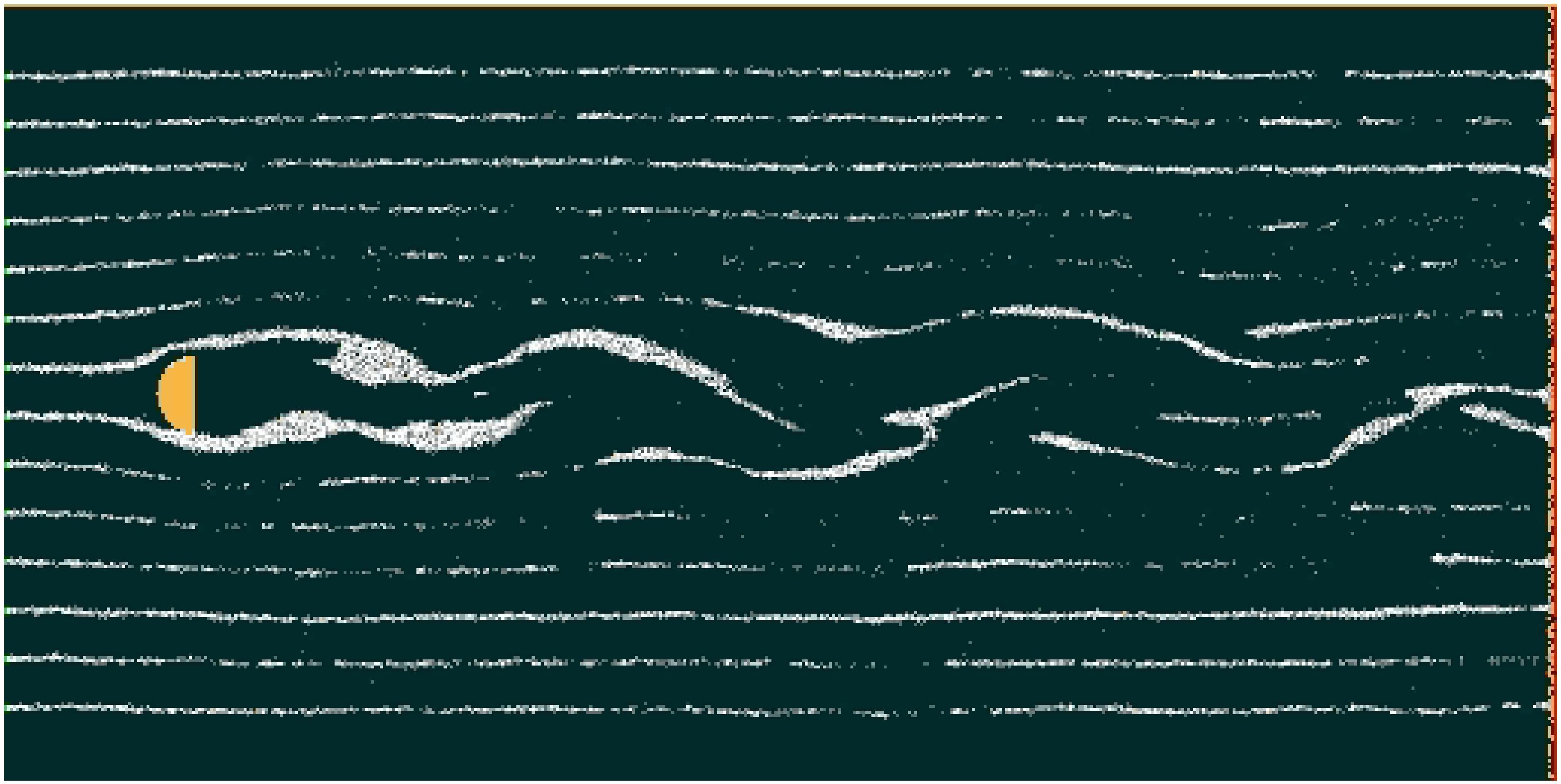


Except: *two ones on diag flip*



- *Half the time:* HPP gas rule outside of blue region, *ID rule* inside (no change).

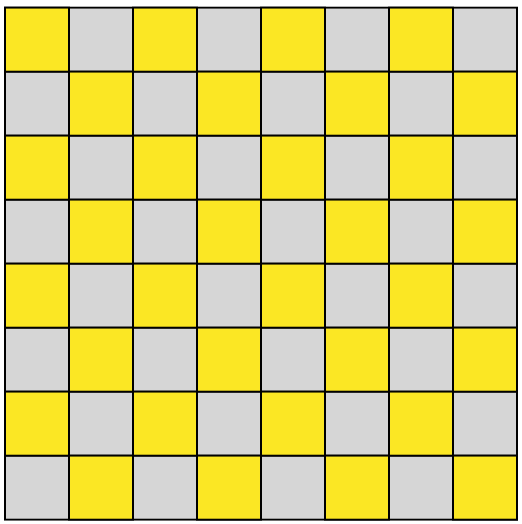
Lattice gas hydrodynamics



Six direction LGA flow past a half-cylinder, with vortex shedding. System is 2Kx1K.

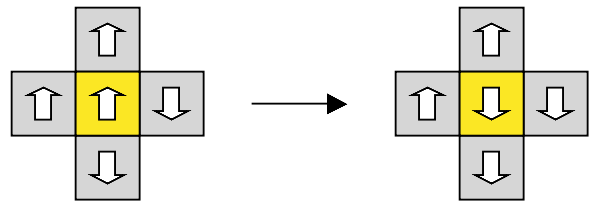
Dynamical Ising rule

Gold/silver checkerboard

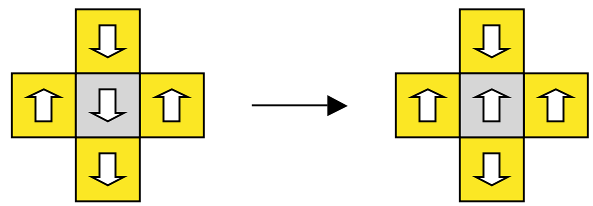


We divide the space into two sublattices, updating the gold on even steps, silver on odd.

Even steps: update gold sublattice

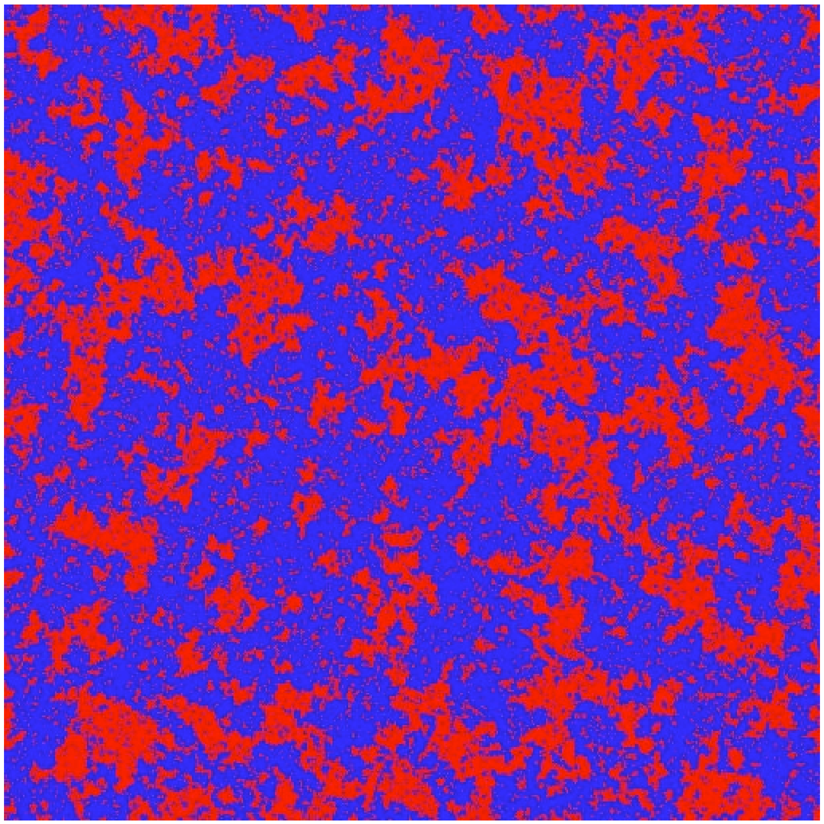


Odd steps: update silver sublattice

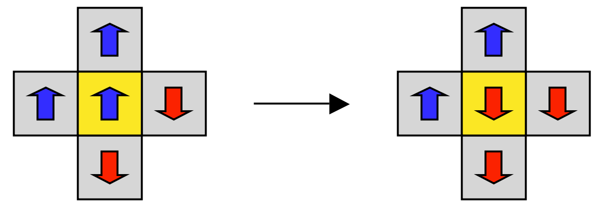


A spin is flipped if exactly 2 of its 4 neighbors are parallel to it. After the flip, exactly 2 neighbors are still parallel.

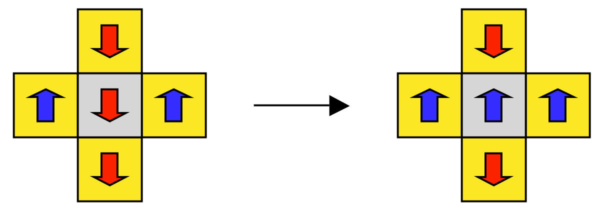
Dynamical Ising rule



Even steps: update gold sublattice

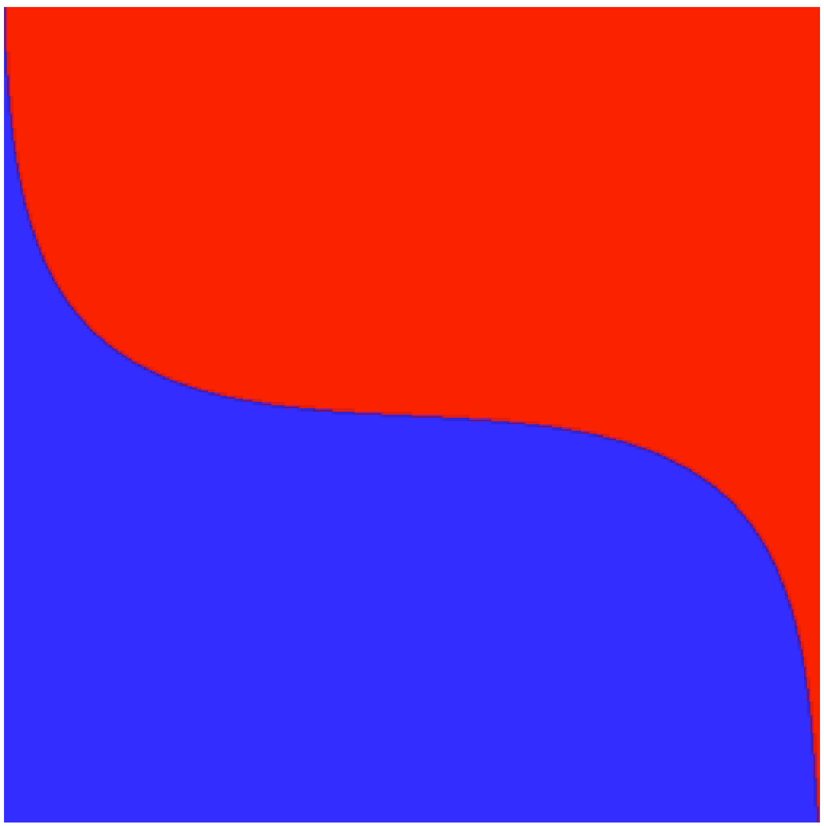


Odd steps: update silver sublattice

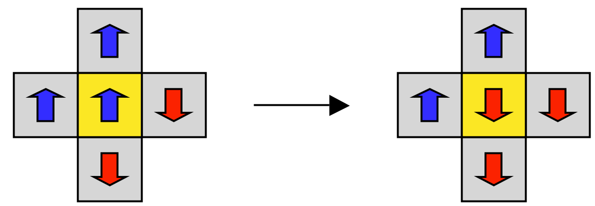


A spin is flipped if exactly 2 of its 4 neighbors are parallel to it. After the flip, exactly 2 neighbors are still parallel.

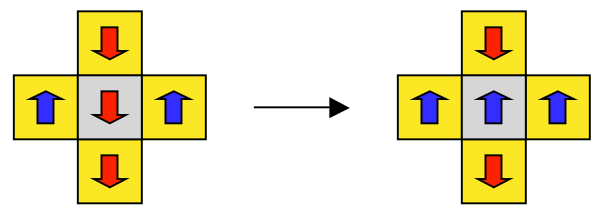
Dynamical Ising rule



Even steps: update gold sublattice



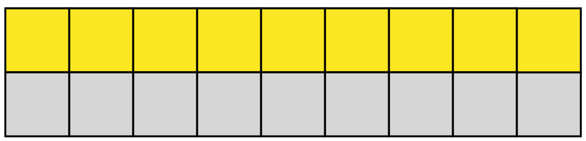
Odd steps: update silver sublattice



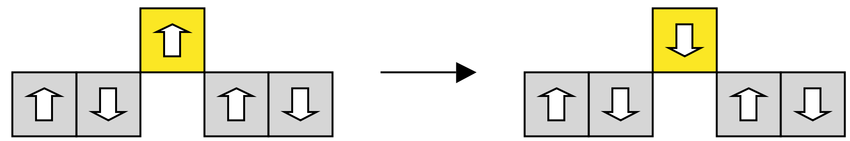
A spin is flipped if exactly 2 of its 4 neighbors are parallel to it. After the flip, exactly 2 neighbors are still parallel.

Bennett's 1D rule

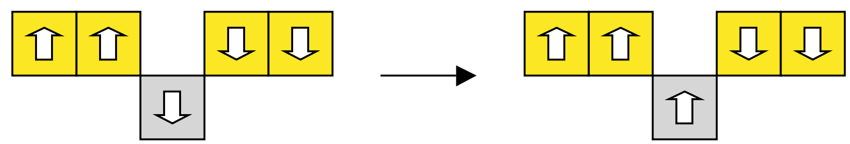
Gold/silver 1D lattice



Even steps: update gold sublattice



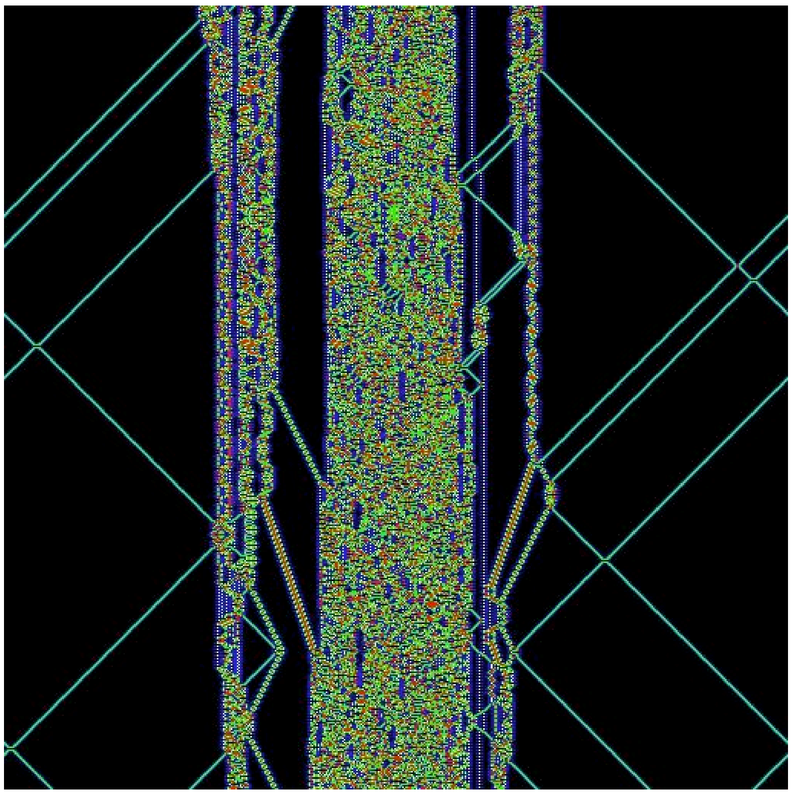
Odd steps: update silver sublattice



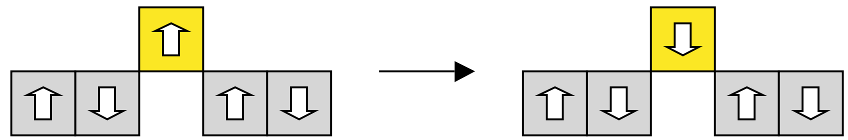
At each site in a 1D space, we put 2 bits of state. We'll call one the "gold" bit and one the "silver" bit. We update the gold bits on even steps, and the silver on odd steps.

A spin is flipped if exactly 2 of its 4 neighbors are parallel to it. After the flip, exactly 2 neighbors are still parallel.

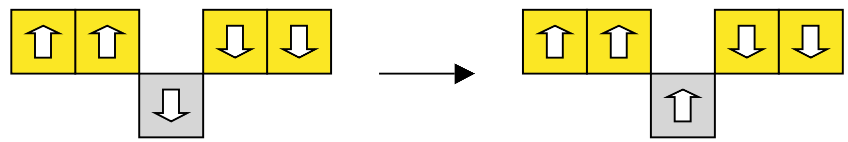
Bennett’s 1D rule



Even steps: update gold sublattice

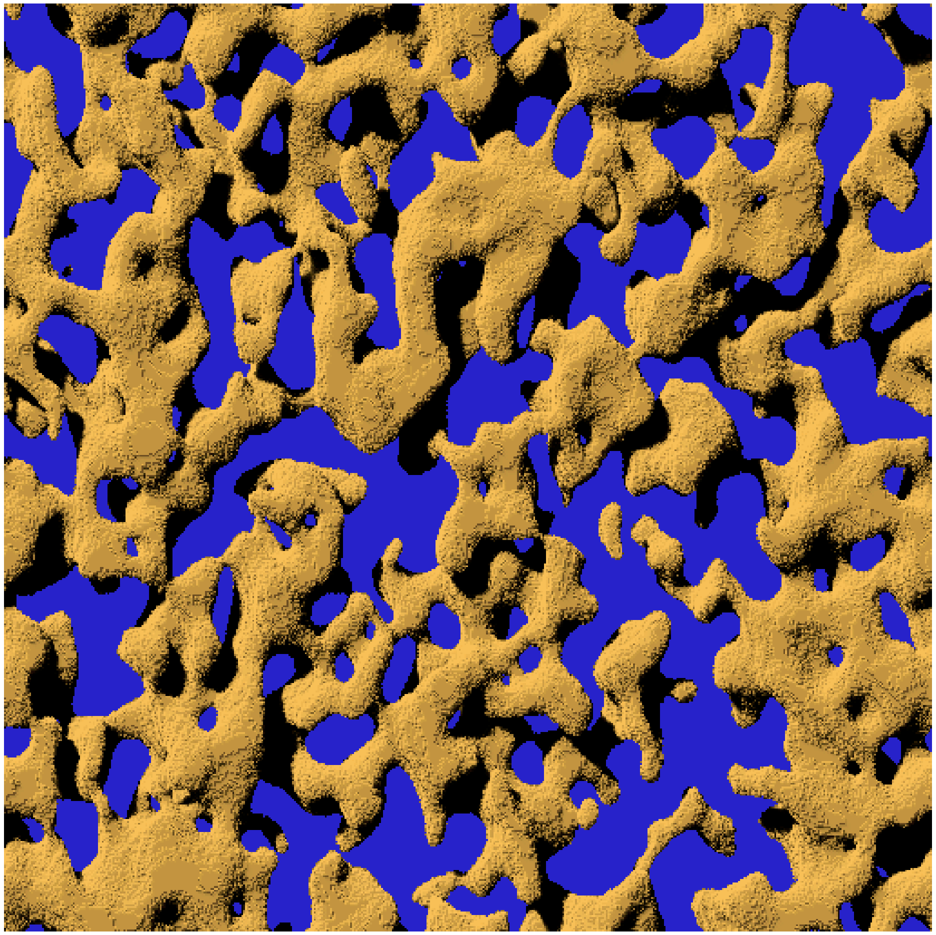


Odd steps: update silver sublattice



A spin is flipped if exactly 2 of its 4 neighbors are parallel to it. After the flip, exactly 2 neighbors are still parallel.

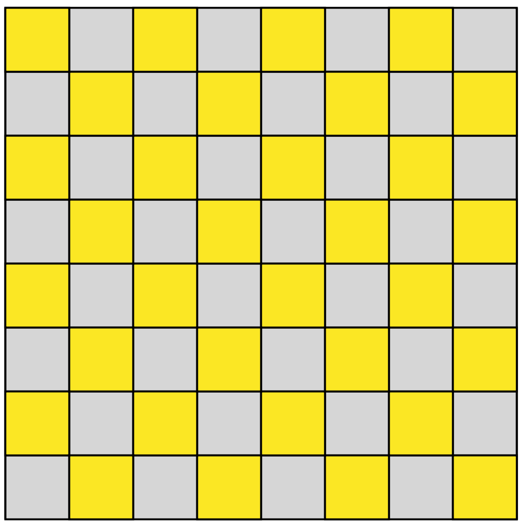
3D Ising with heat bath



If the heat bath is initially much cooler than the spin system, then domains grow as the spins cool.

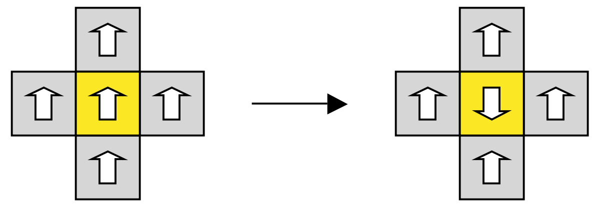
2D "Same" rule

Gold/silver checkerboard

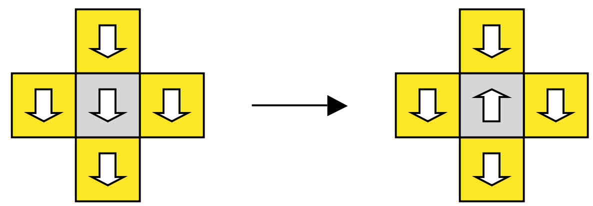


We divide the space into two sublattices, updating the gold on even steps, silver on odd.

Even steps: update gold sublattice

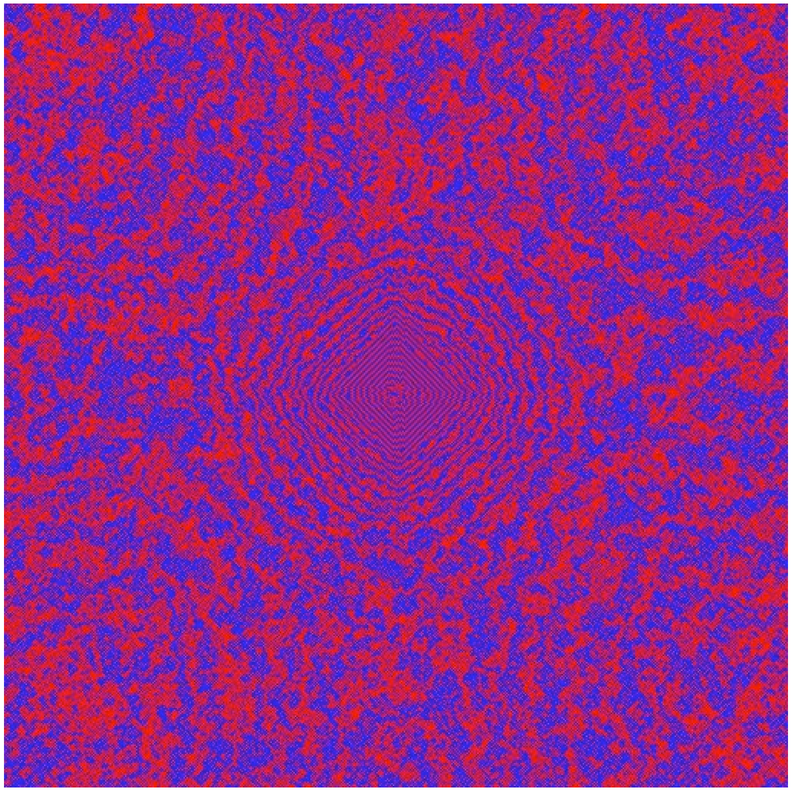


Odd steps: update silver sublattice

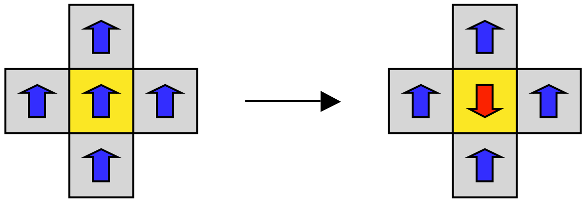


A spin is flipped if all 4 of its neighbors are the same. Otherwise it is left unchanged.

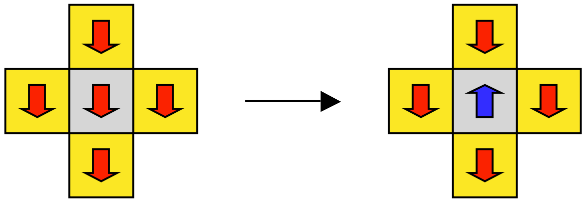
2D "Same" rule



Even steps: update gold sublattice

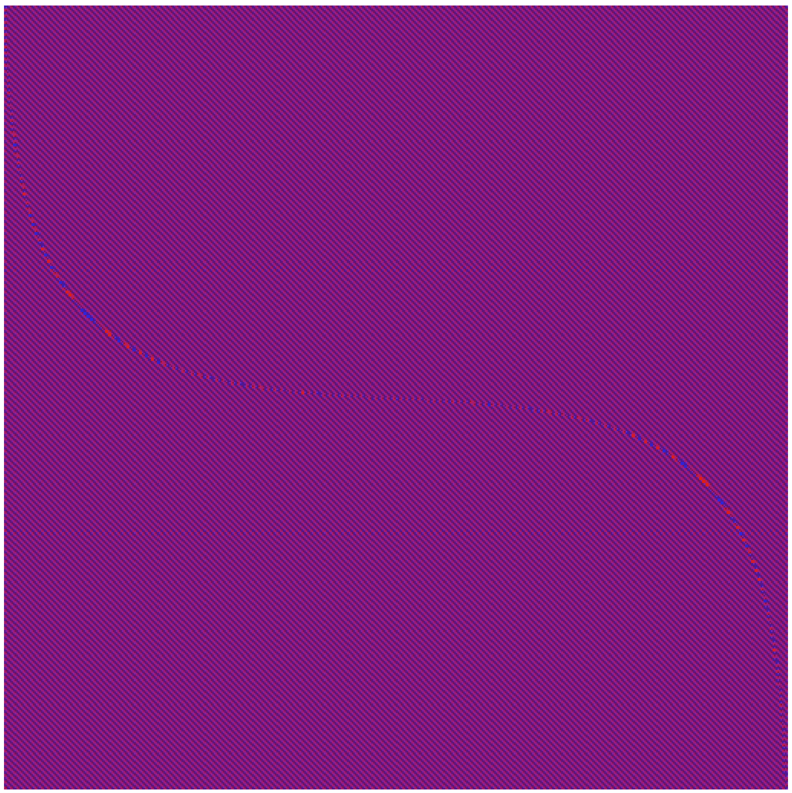


Odd steps: update silver sublattice

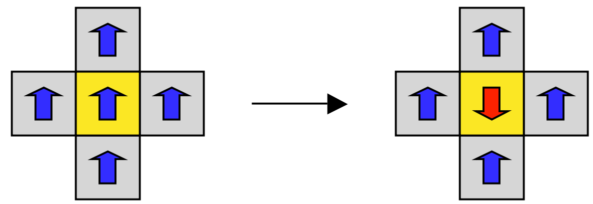


A spin is flipped if all 4 of its neighbors are the same. Otherwise it is left unchanged.

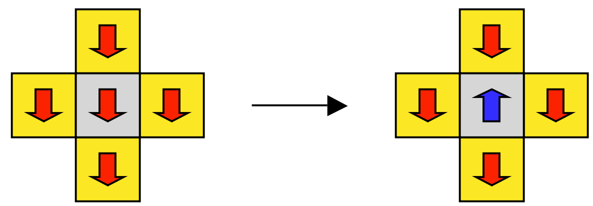
2D “Same” rule



Even steps: update gold sublattice

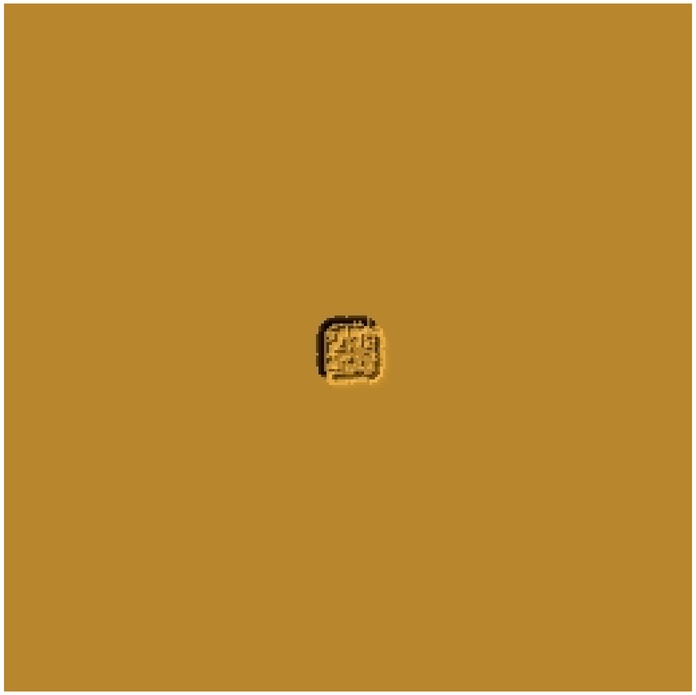


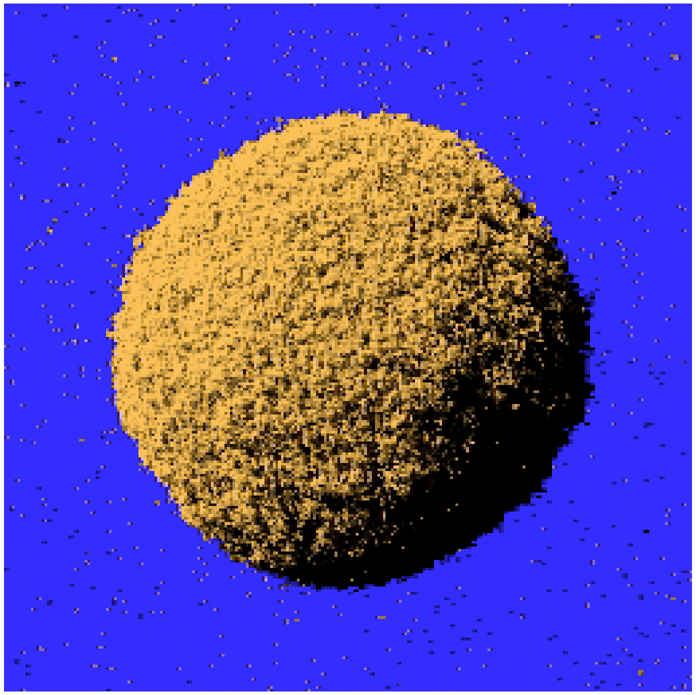
Odd steps: update silver sublattice



A spin is flipped if all 4 of its neighbors are the same. Otherwise it is left unchanged.

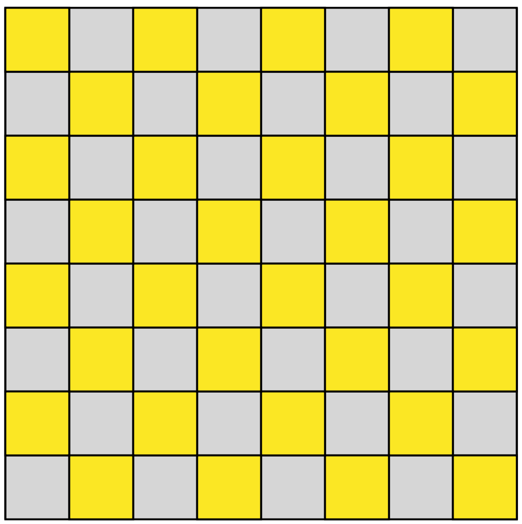
3D “Same” rule





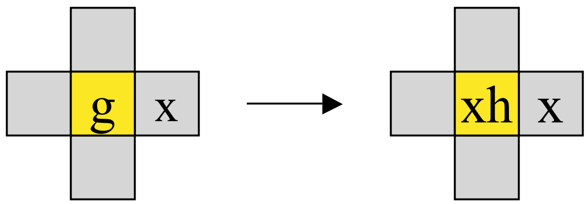
Reversible aggregation rule

Gold/silver checkerboard

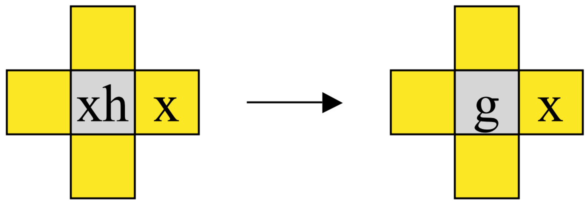


We update the gold sublattice, then let gas and heat diffuse, then update silver and diffuse.

Even steps: update gold sublattice



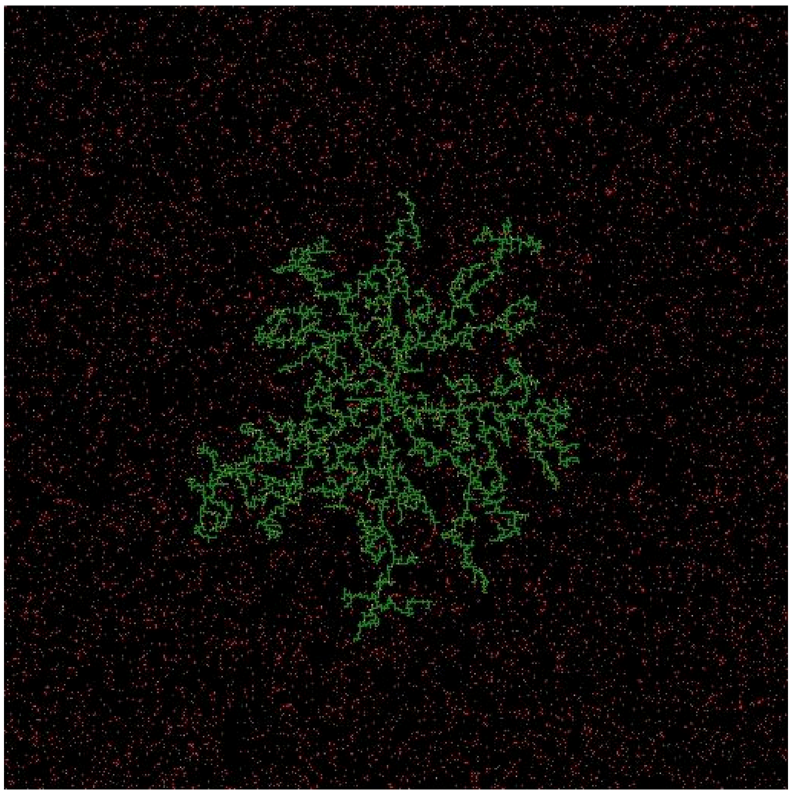
Odd steps: update silver sublattice



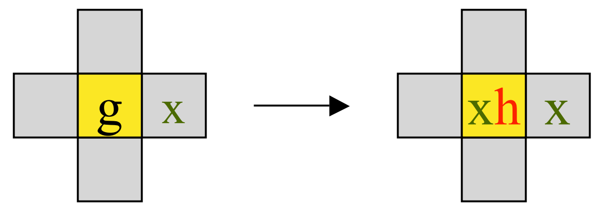
When a gas particle diffuses next to exactly one crystal particle, it crystallizes and emits a heat particle. The reverse also happens.

for more info, see cond-mat/9810258

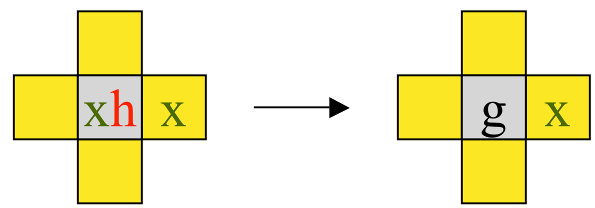
Reversible aggregation rule



Even steps: update gold sublattice



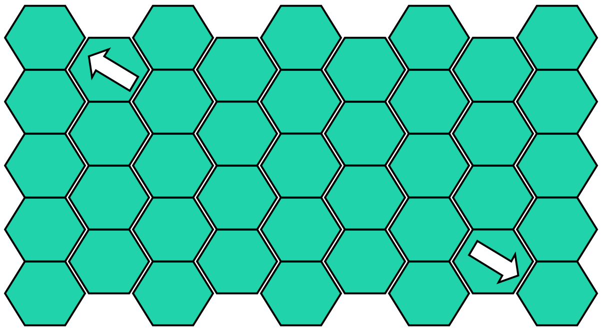
Odd steps: update silver sublattice



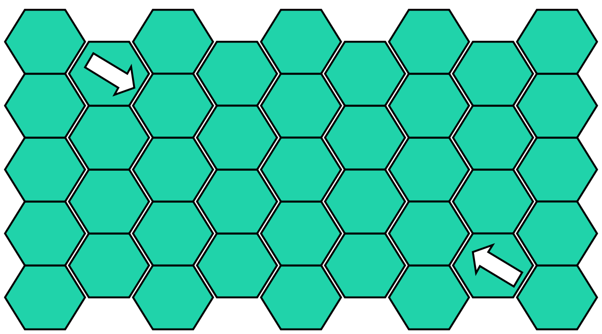
When a gas particle diffuses next to exactly one crystal particle, it crystallizes and emits a heat particle. The reverse also happens.

for more info, see cond-mat/9810258

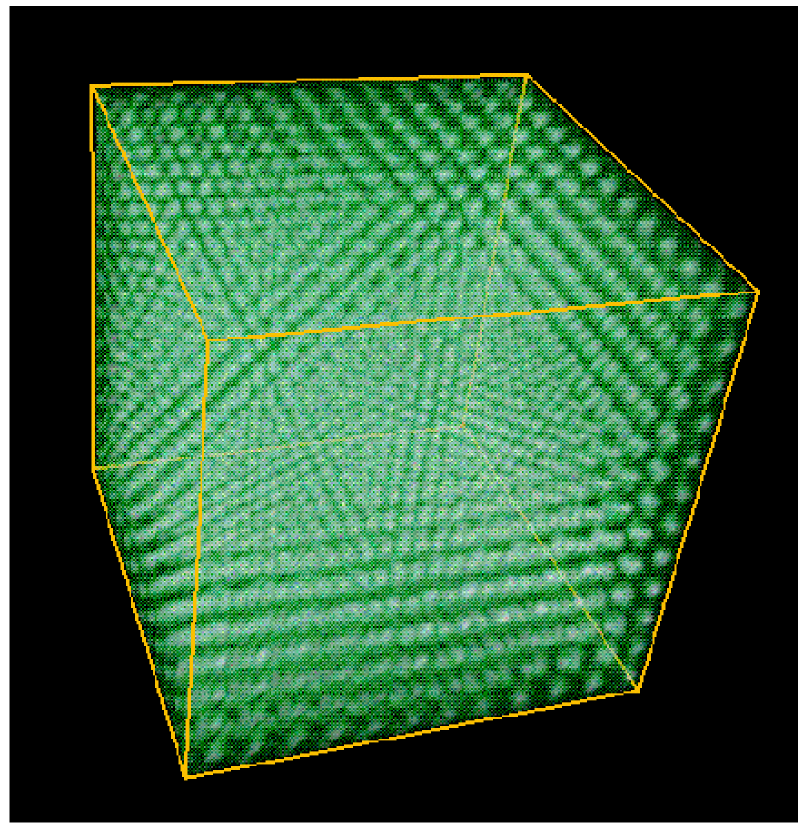
Adding forces irreversibly



becomes:

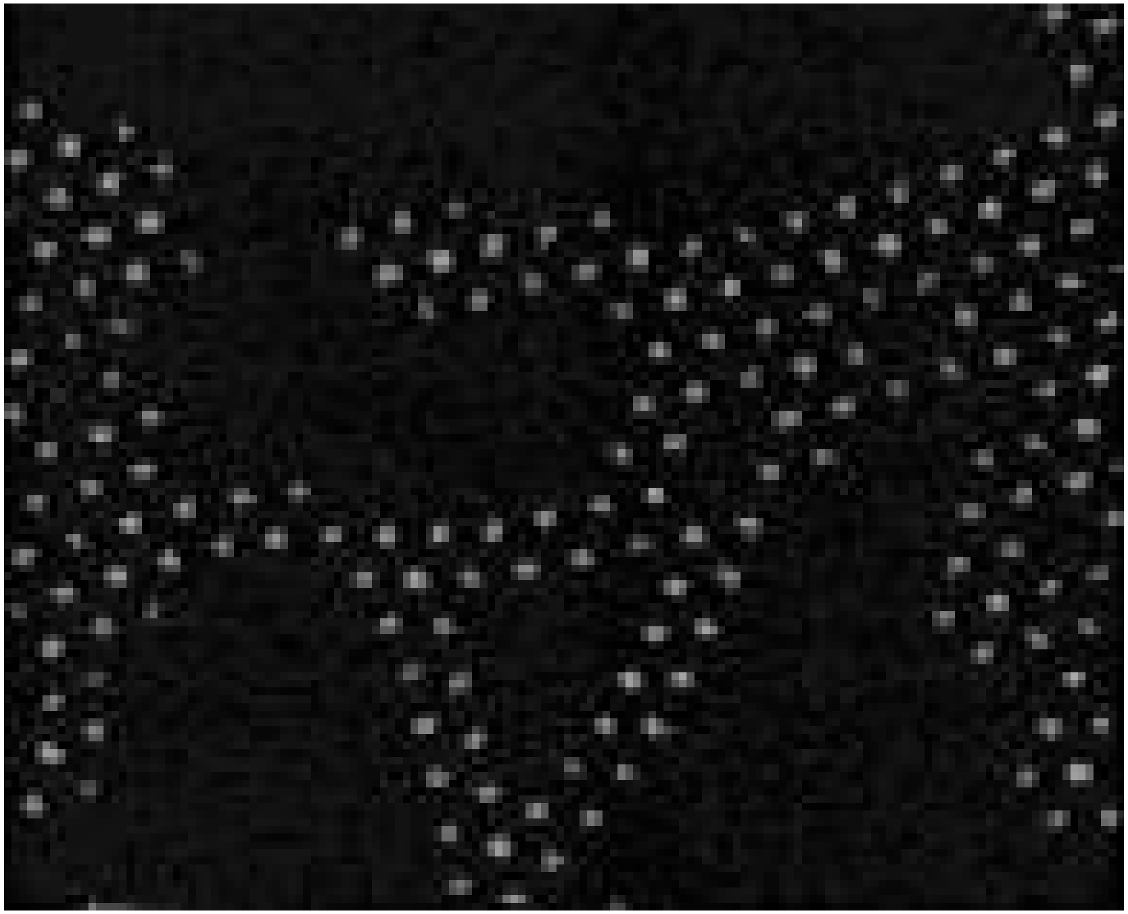


Particles six sites apart along the lattice attract each other.



3D momentum conserving crystallization.

Adding forces irreversibly

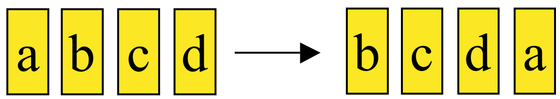


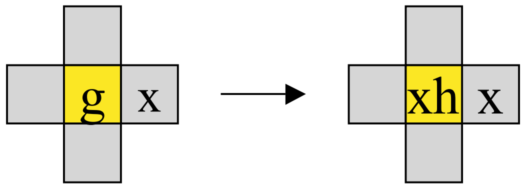
Crystallization using irreversible forces (Jeff Yepez, AFOSR)

Conservations summary

To make conservations manifest, we employ a sequence of steps, in each of which either

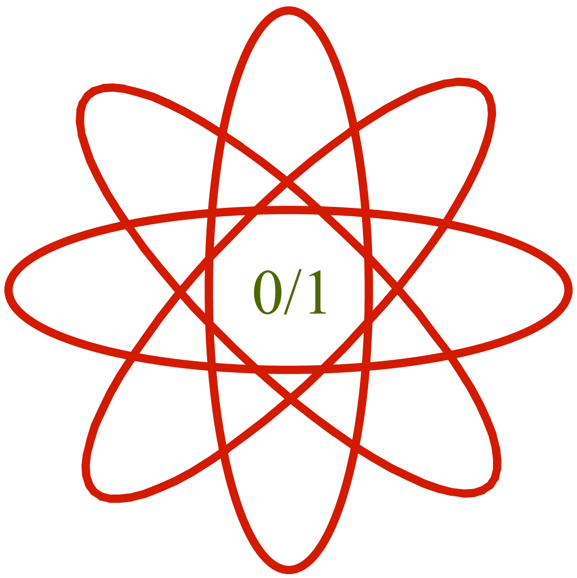
1. *the data are rearranged without any interaction,* or
2. *the data are partitioned into disjoint groups of bits that change as a unit. Data that affect more than one such group don't change.*





Conservations allow computations to map efficiently onto microscopic physics, and also allow them to have interesting macroscopic behavior.

Physics becomes the computer



Emulating Physics

- » *Finite-state, locality, invertibility, and conservation laws*

Physical Worlds

- » *Incorporating comp-universality at small and large scales*

Spatial Computers

- » *Architectures and algorithms for large-scale spatial computations*

Nature as Computer

- » *Physical concepts enter CS and computer concepts enter Physics*