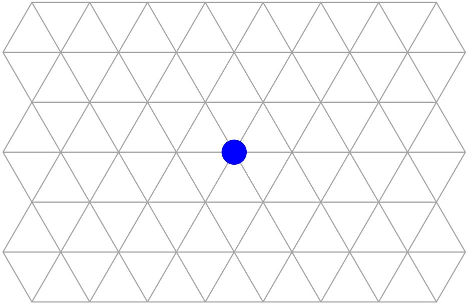
Shape Formation by Programmable Particles

WTCS 2018

Giovanni Viglietta

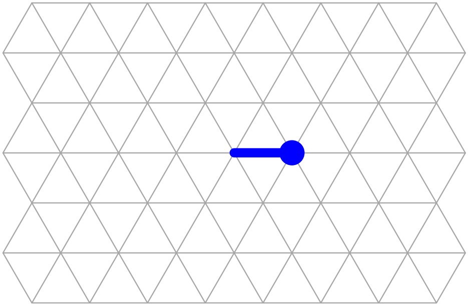
(JAIST, Nomi City, Japan)

Iizuka City – September 10, 2018



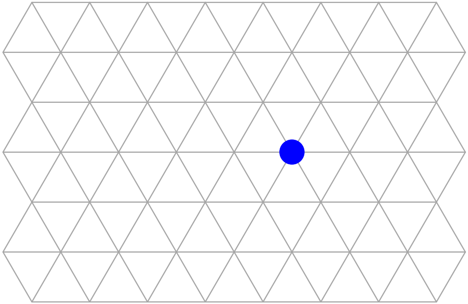
In this model, particles occupy nodes of a triangular grid.

Amoebots



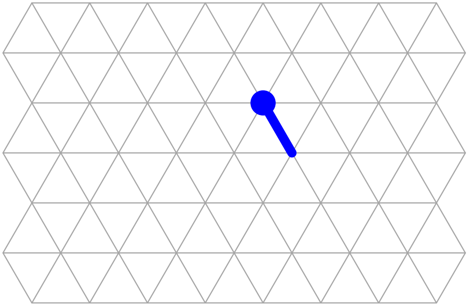
A particle can move by *expanding* and *contracting*.

Amoebots

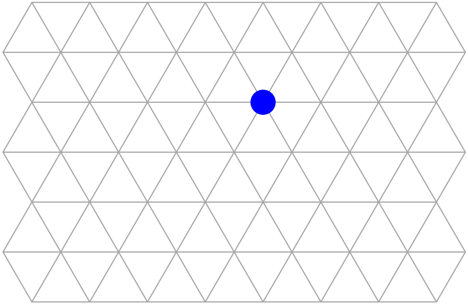


A particle can move by *expanding* and *contracting*.

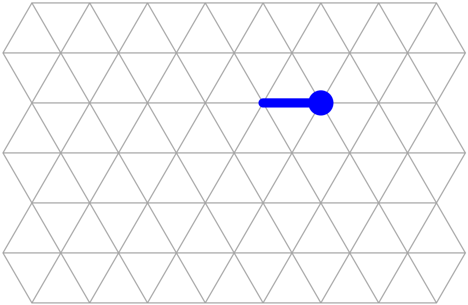
Amoebots



A particle can move by *expanding* and *contracting*.

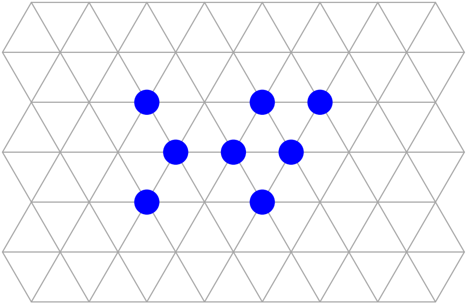


A particle can move by *expanding* and *contracting*.



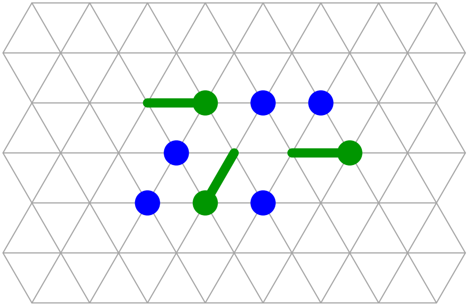
A particle can move by *expanding* and *contracting*.

Amoebots



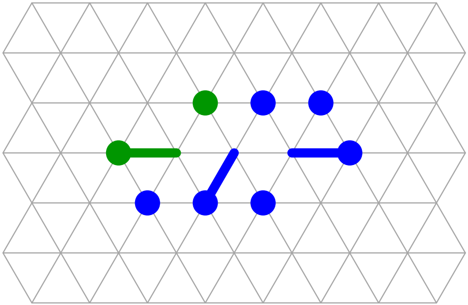
A *system* of particles is given.

Amoebots



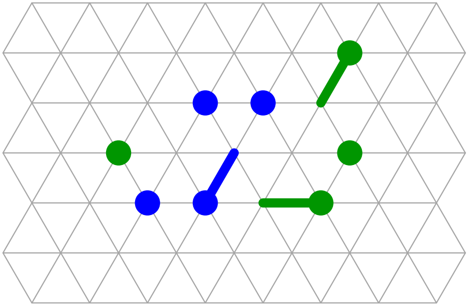
Particles move *asynchronously* following an algorithm.

Amoebots



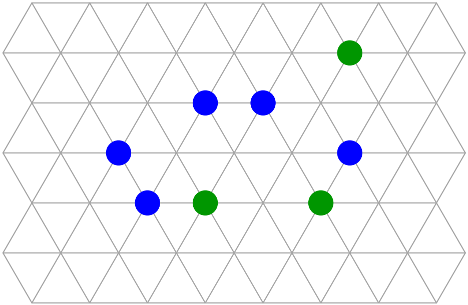
Particles move *asynchronously* following an algorithm.

Amoebots

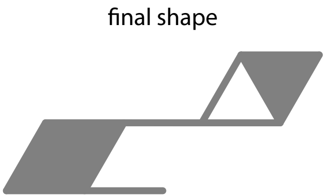


At each step, any set of particles is activated by an *adversary*.

Amoebots

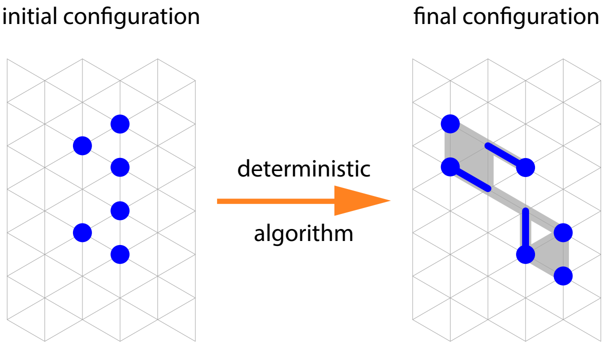


At each step, any set of particles is activated by an *adversary*.



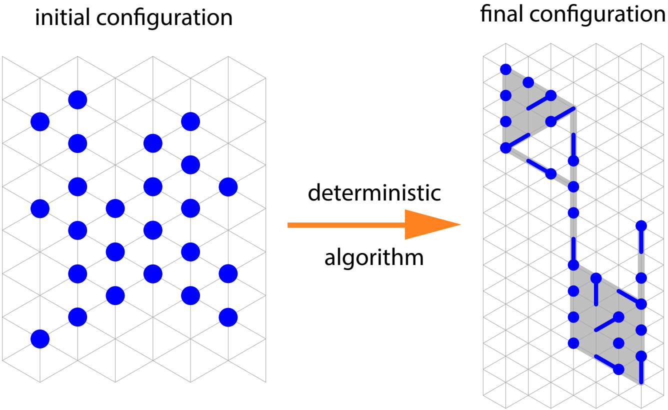
The goal is to form a *shape* that is given as input to all particles.

Shape Formation



The shape formation algorithm should be *deterministic*.

Shape Formation



The shape can be scaled up depending on the size of the system.

Related Literature

Original paper introducing Amoebots:

- Derakhshandeh, Gmyr, Strothmann, Bazzi, Richa, Scheideler

  Leader election and shape formation with self-organizing programmable matter

  DNA 2015

Randomized shape-formation algorithm for sequentially activated Amoebots starting from a triangular shape:

- Derakhshandeh, Gmyr, Richa, Scheideler, Strothmann

  Universal shape formation for programmable matter

  SPAA 2016

Deterministic algorithm, general shapes, asynchronous Amoebots:

- Di Luna, Flocchini, Santoro, Viglietta, Yamauchi

  Shape formation by programmable particles

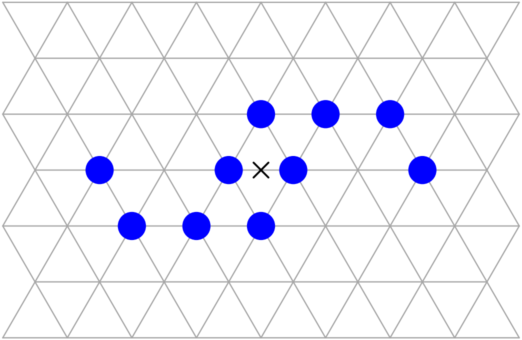
  DISC 2017 (BA), OPODIS 2017

Our Particle Model

The n particles in the system:

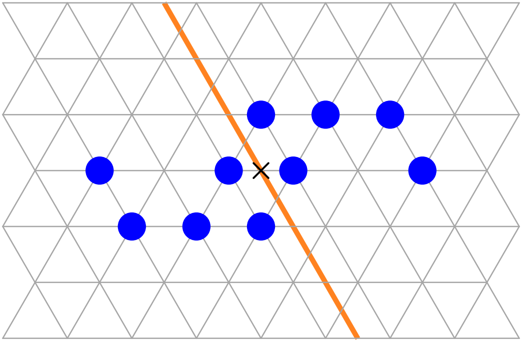
- initially form any simply connected shape
- know the final shape but do *not* know n
- have a constant amount of internal memory
- are anonymous and start in the same state
- can only see and communicate with adjacent particles
- do not have a *compass*
- may not agree on a *clockwise direction*
- are activated asynchronously
- execute the same deterministic algorithm
- cannot occupy the same node

Unbreakable Symmetry



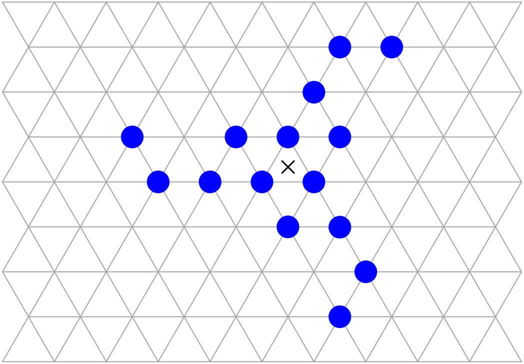
If the system has a center of symmetry not in a grid node...

Unbreakable Symmetry



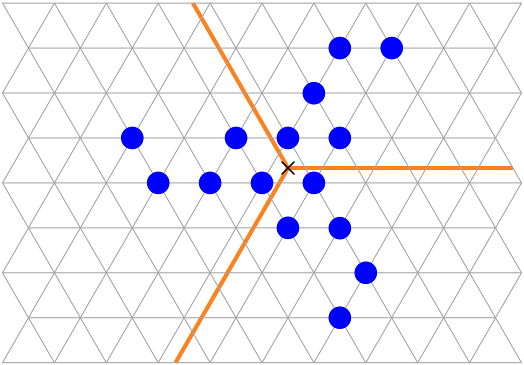
Then this symmetry is impossible to break.

Unbreakable Symmetry



The same holds for systems with a 3-fold rotational symmetry.

Unbreakable Symmetry



If the center is not in a grid node, the symmetry is unbreakable.

Statement of Results

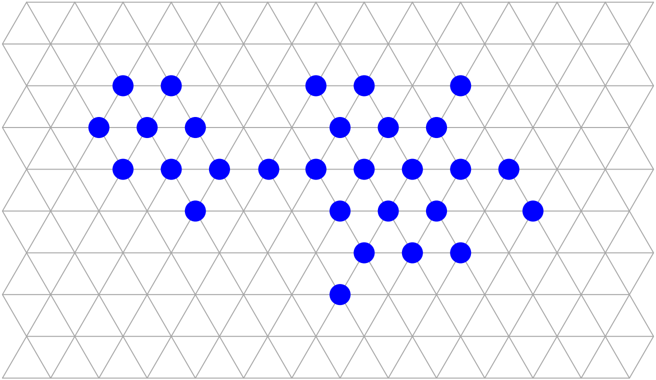
Theorem

If the system initially has an unbreakable symmetry, it cannot form shapes that do not have the same symmetry.

Theorem

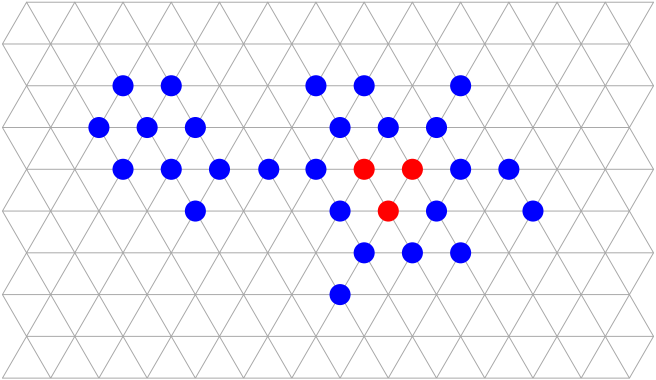
For all other cases, there is a universal shape-formation algorithm, provided that the system initially forms a simply connected shape, and the final shape and its scaled-up copies are Turing-computable (with some bland extra assumptions).

Universal Shape-Formation Algorithm

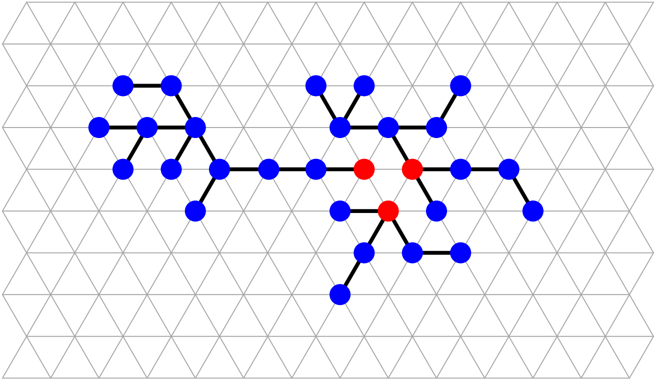


Start with a sufficiently large simply connected system.

Universal Shape-Formation Algorithm

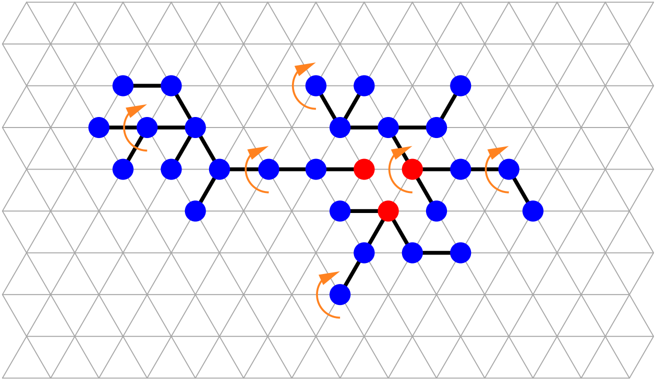


Phase 1: attempt to elect a leader.

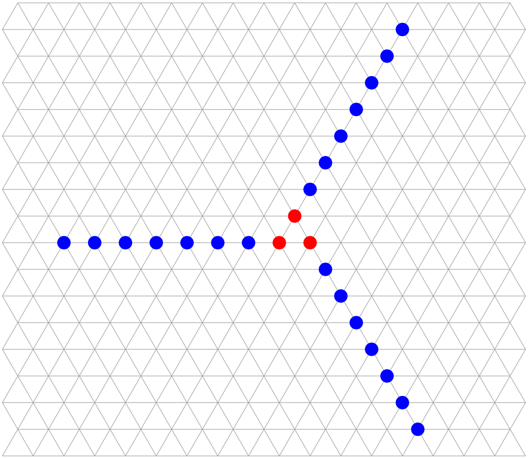


Phase 2: construct a spanning forest.

Universal Shape-Formation Algorithm

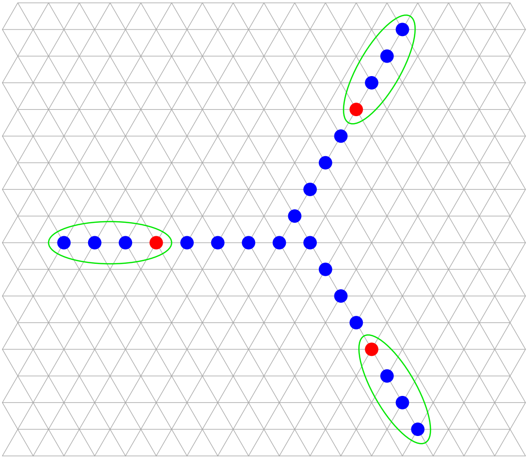


Phase 3: agree on a clockwise direction.



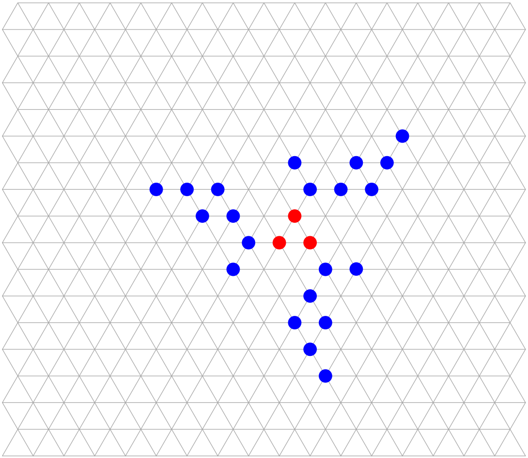
Phase 4: form one line per leader.

Universal Shape-Formation Algorithm



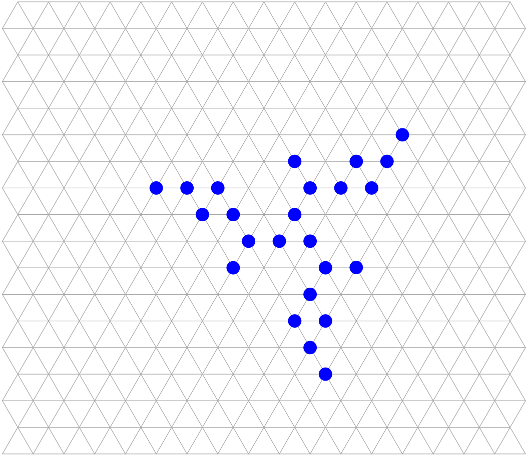
Phase 5: simulate Turing machines to compute the final shape.

Universal Shape-Formation Algorithm



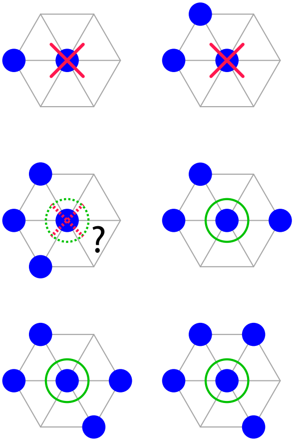
Phase 6: keep computing while forming the final shape.

Universal Shape-Formation Algorithm



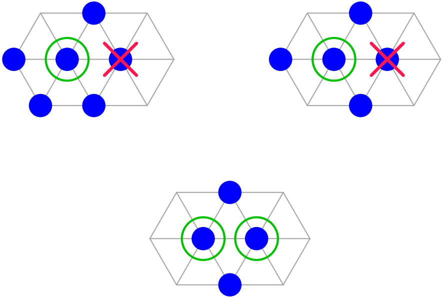
Phase 6: keep computing while forming the final shape.

Lattice Consumption Phase



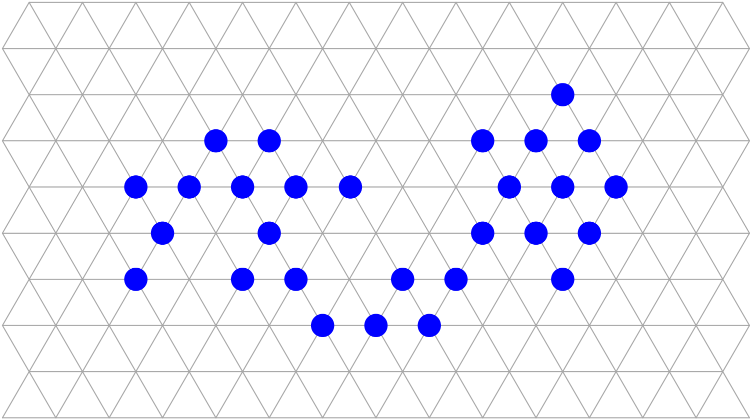
All particles are initially *eligible*. Depending on its eligible neighbors, a particle may decide to *eliminate* itself or stay eligible.

Lattice Consumption Phase



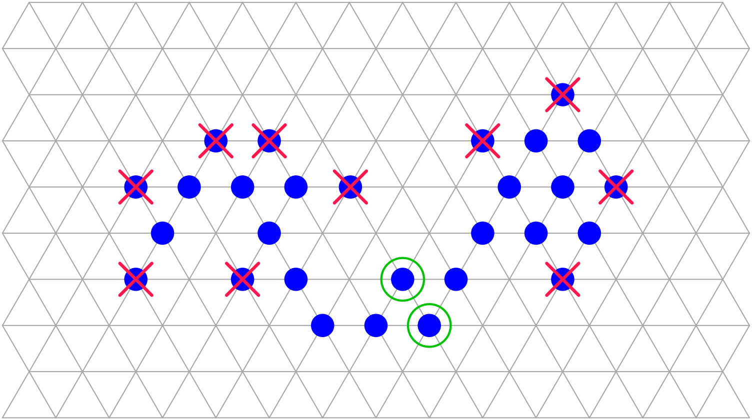
There is just one special case, where the particle has to communicate with a neighbor to ensure that its elimination would not disconnect the set of eligible particles.

Lattice Consumption Phase



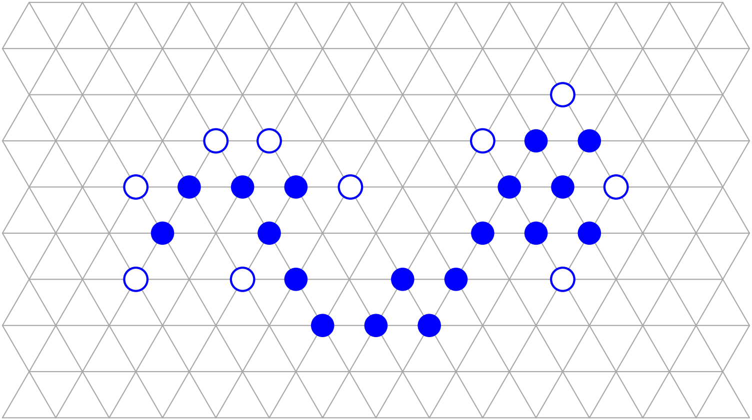
Following this protocol, the set of eligible particles remains simply connected, even if activations happen asynchronously.

Lattice Consumption Phase



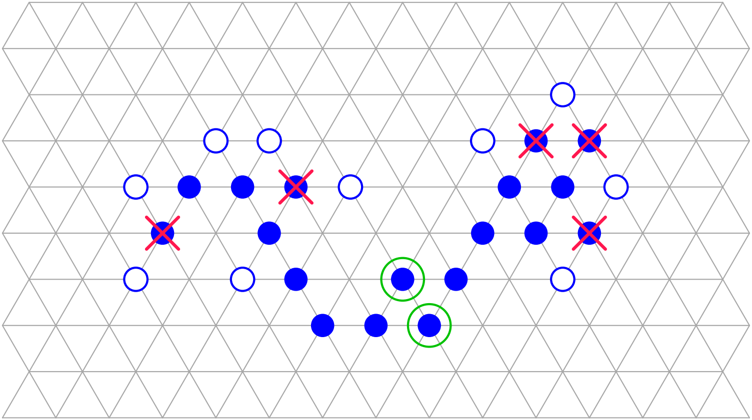
Following this protocol, the set of eligible particles remains simply connected, even if activations happen asynchronously.

Lattice Consumption Phase



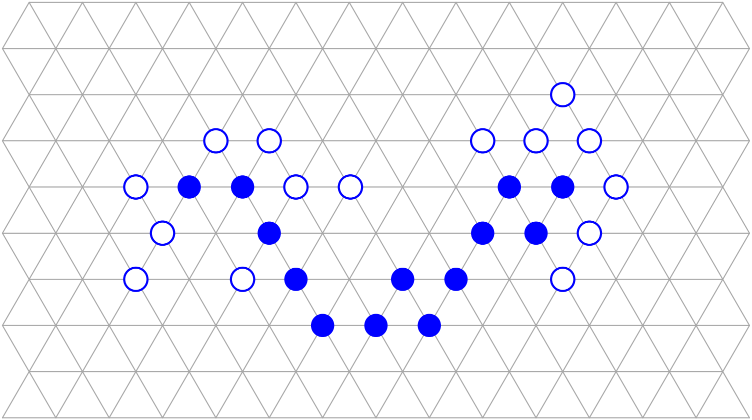
Following this protocol, the set of eligible particles remains simply connected, even if activations happen asynchronously.

Lattice Consumption Phase



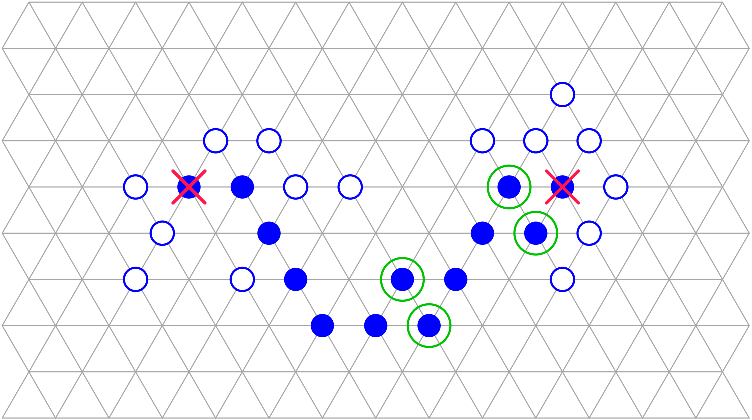
Following this protocol, the set of eligible particles remains simply connected, even if activations happen asynchronously.

Lattice Consumption Phase

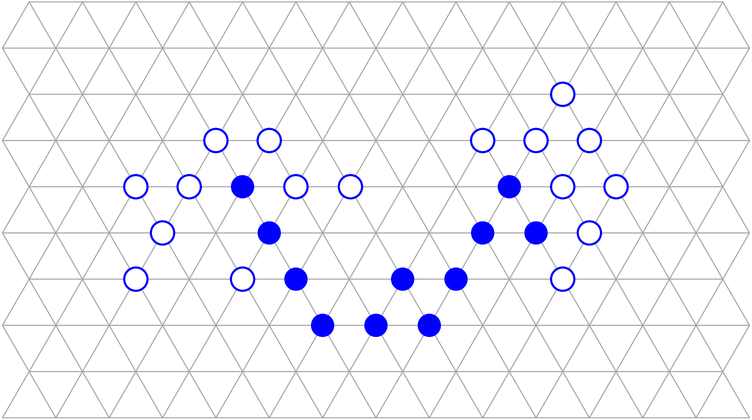


Following this protocol, the set of eligible particles remains simply connected, even if activations happen asynchronously.

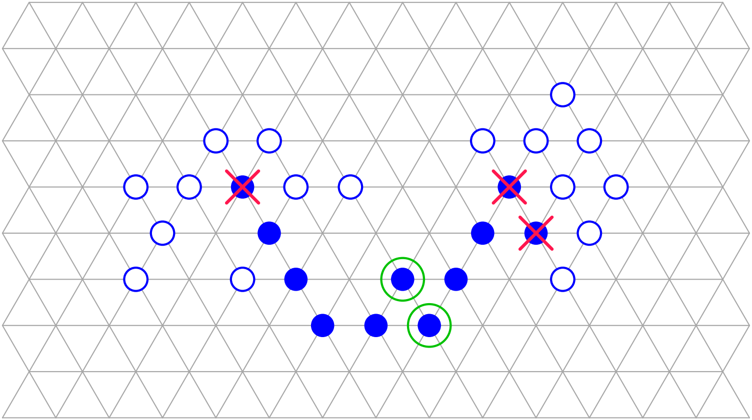
Lattice Consumption Phase



Following this protocol, the set of eligible particles remains simply connected, even if activations happen asynchronously.

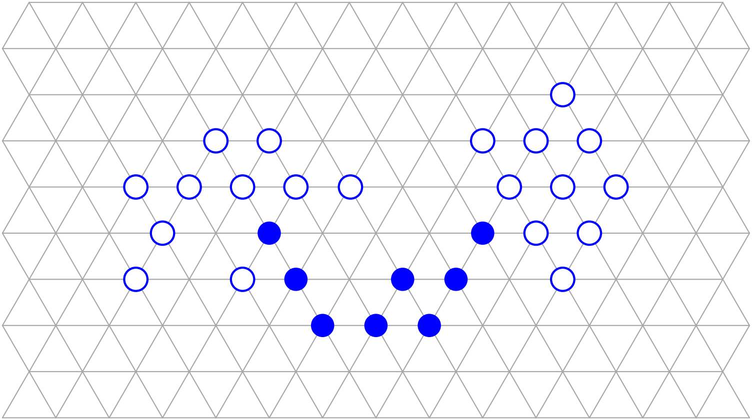


Following this protocol, the set of eligible particles remains simply connected, even if activations happen asynchronously.



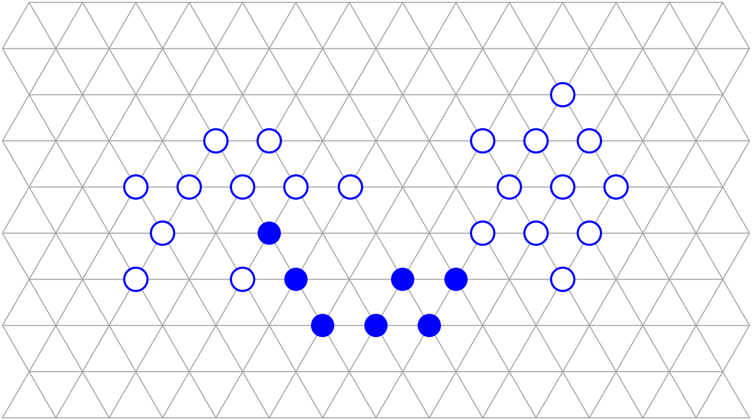
Following this protocol, the set of eligible particles remains simply connected, even if activations happen asynchronously.

Lattice Consumption Phase



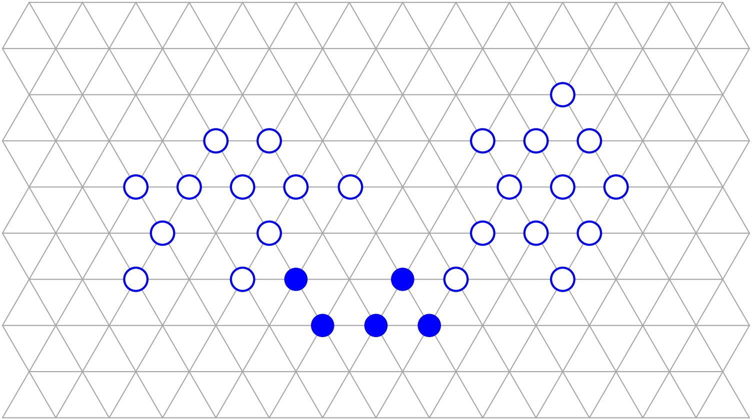
Following this protocol, the set of eligible particles remains simply connected, even if activations happen asynchronously.

Lattice Consumption Phase



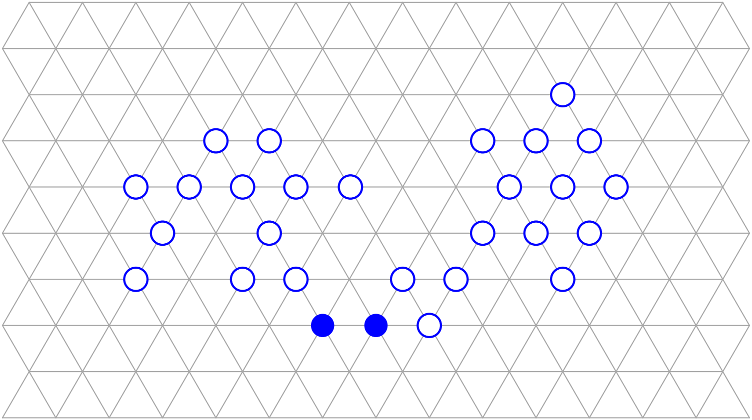
Following this protocol, the set of eligible particles remains simply connected, even if activations happen asynchronously.

Lattice Consumption Phase



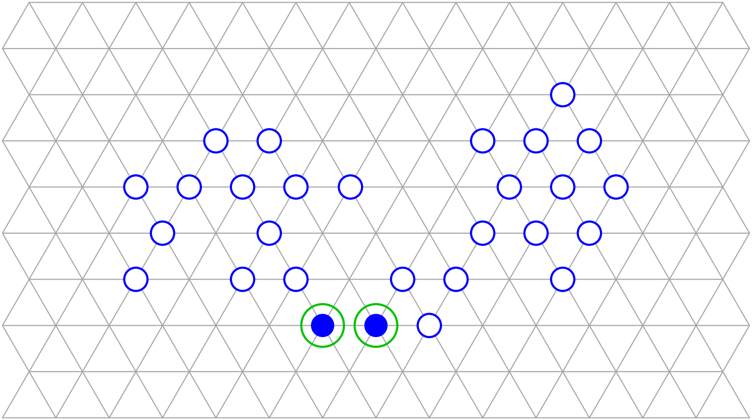
Following this protocol, the set of eligible particles remains simply connected, even if activations happen asynchronously.

Lattice Consumption Phase



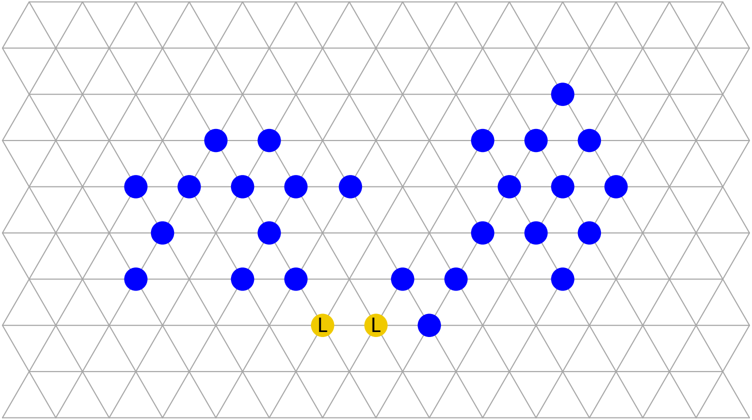
Following this protocol, the set of eligible particles remains simply connected, even if activations happen asynchronously.

Lattice Consumption Phase



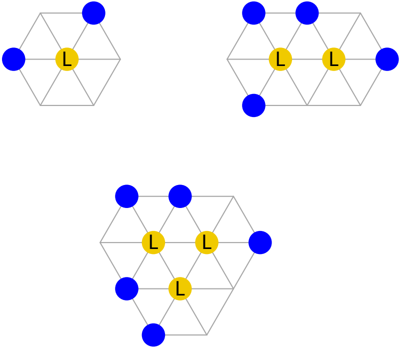
When the process ends, the particles that are still eligible become *candidate leaders*.

Lattice Consumption Phase



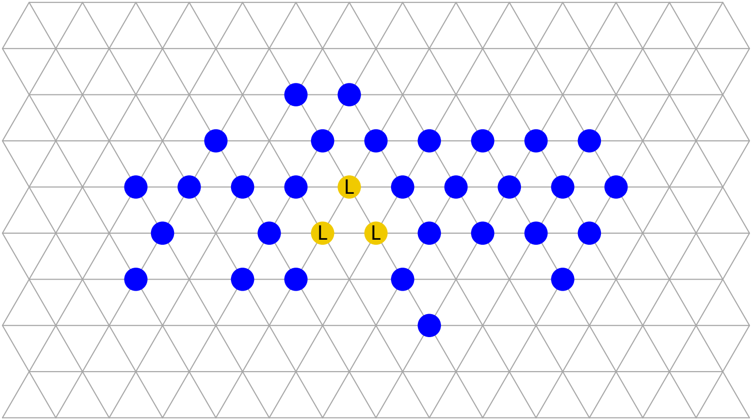
When the process ends, the particles that are still eligible become *candidate leaders*.

Lattice Consumption Phase



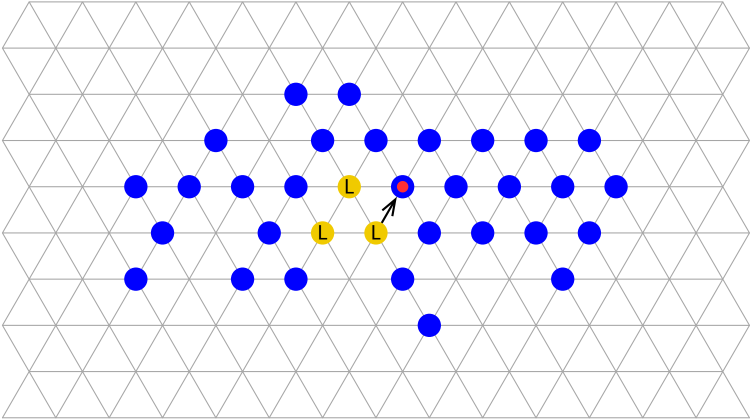
The candidate leaders are all adjacent, and can be at most 3.

Spanning Forest Construction Phase



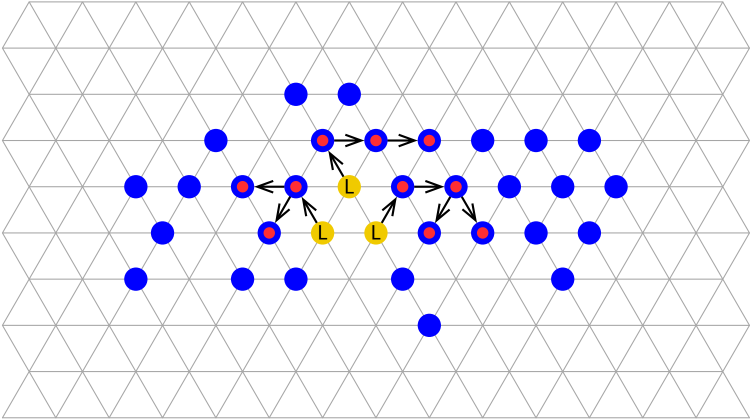
Each candidate leader starts constructing a tree.

Spanning Forest Construction Phase



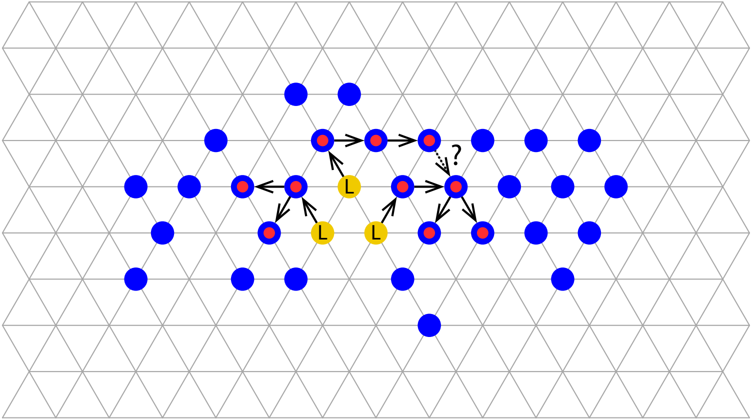
Each candidate leader starts constructing a tree.

Spanning Forest Construction Phase

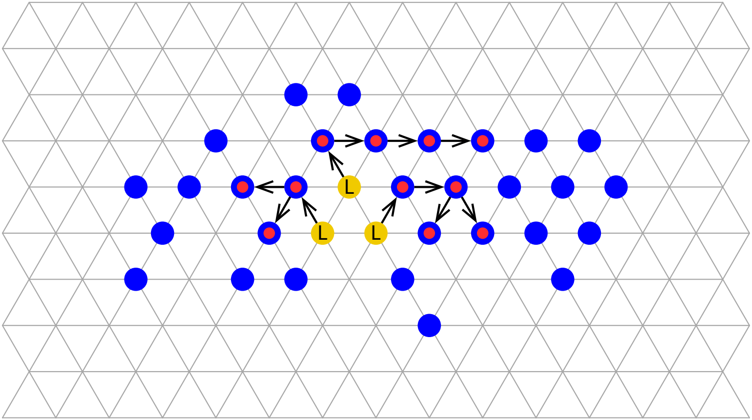


Each node of a tree tries to extend the tree in all directions by sending *merge requests* to its neighbors.

Spanning Forest Construction Phase

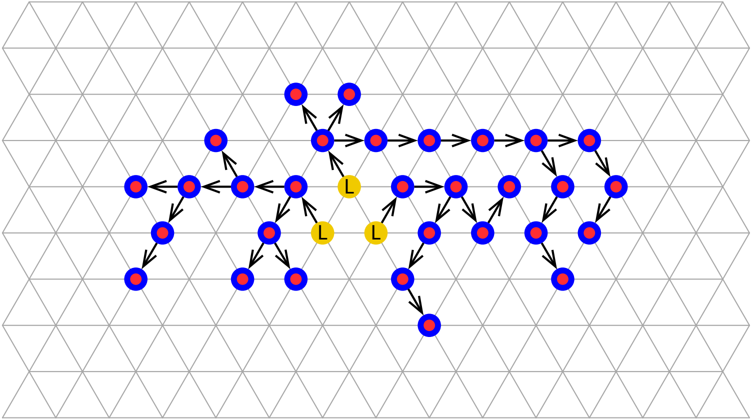


If a node is already part of a tree, it refuses further merge requests.



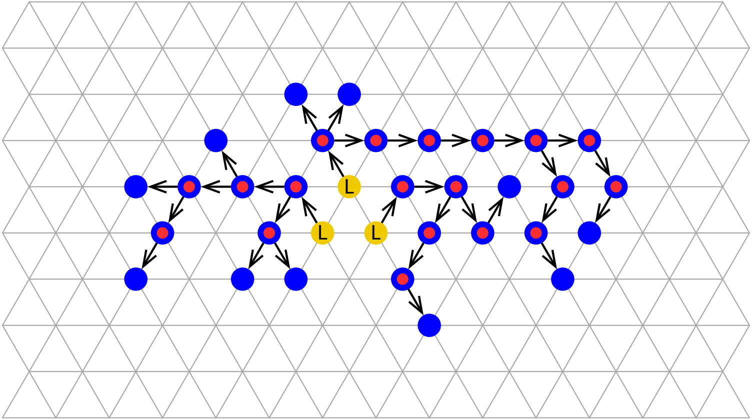
Otherwise, it sets a *parent* variable to the *port number* corresponding to a neighbor that sent a request.

Spanning Forest Construction Phase



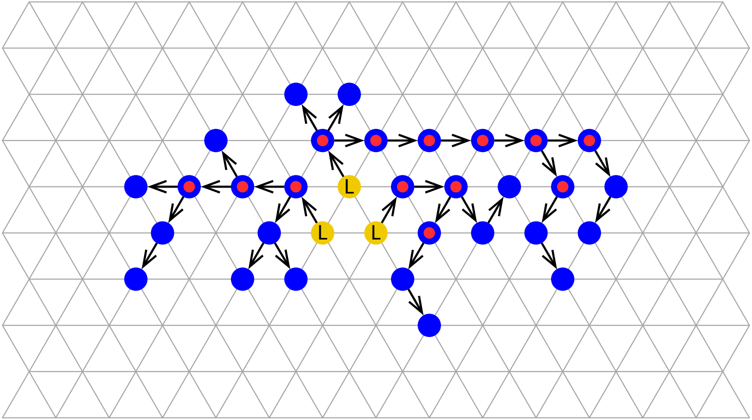
Since the shape is connected, eventually a spanning forest is constructed.

Spanning Forest Construction Phase



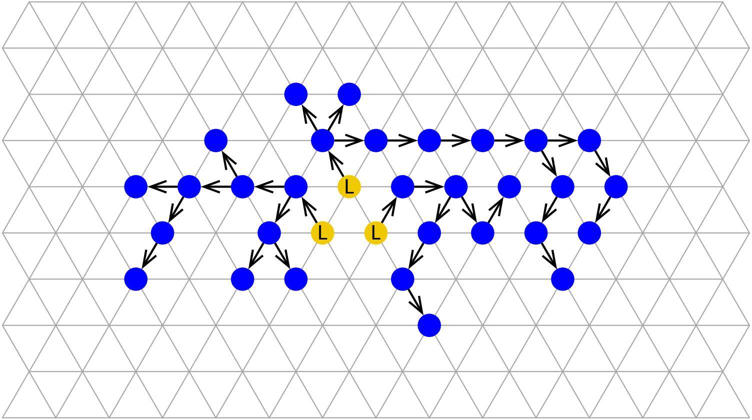
Nodes that cannot expand anymore send a *termination message* to their parents, starting from the leaves.

Spanning Forest Construction Phase



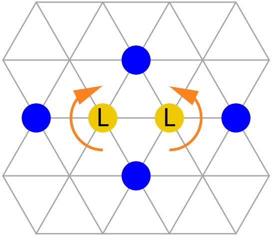
Nodes that cannot expand anymore send a *termination message* to their parents, starting from the leaves.

Spanning Forest Construction Phase



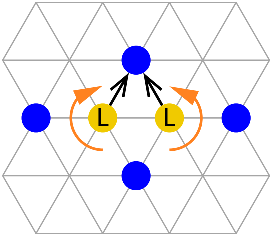
Eventually, the termination messages reach the candidate leaders, and the phase ends.

Handedness Agreement Phase



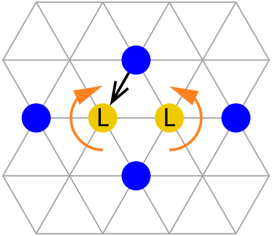
We want two candidate leaders to agree on the same handedness.

Handedness Agreement Phase



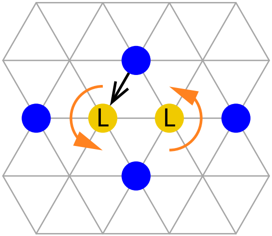
If they have a common neighbor, they send a message to it.
If the same neighbor receives both messages, it means that the candidate leaders have opposite handedness.

Handedness Agreement Phase



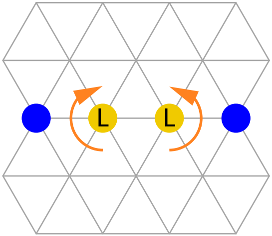
So, the neighbor decides which candidate leader has to change its handedness, and sends it a message.

Handedness Agreement Phase



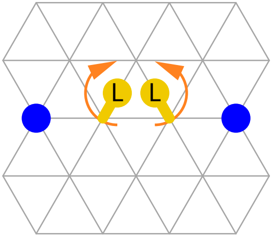
So, the neighbor decides which candidate leader has to change its handedness, and sends it a message.

Handedness Agreement Phase



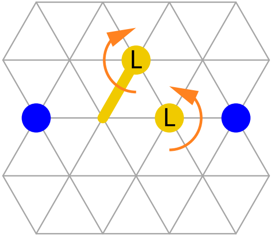
If the candidate leaders have no common neighbor, they try to expand to a neighboring location.

Handedness Agreement Phase



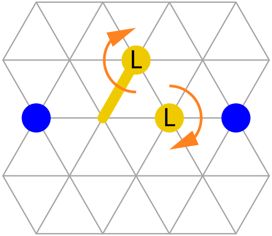
If the candidate leaders have no common neighbor, they try to expand to a neighboring location.

Handedness Agreement Phase



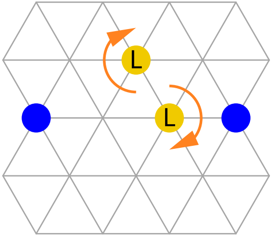
If one of them fails to reach it, it means that they have opposite handedness.

Handedness Agreement Phase



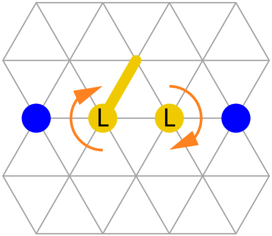
So, the candidate leader that fails to expand changes its own handedness.

Handedness Agreement Phase



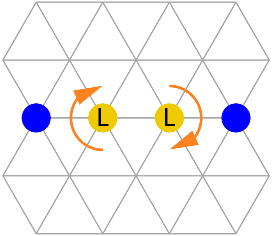
If a candidate leader succeeds to expand, it then contracts and moves back to its original location.

Handedness Agreement Phase



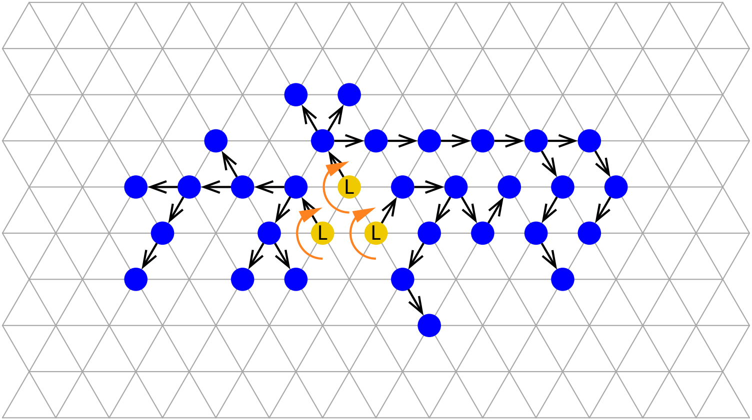
If a candidate leader succeeds to expand, it then contracts and moves back to its original location.

Handedness Agreement Phase



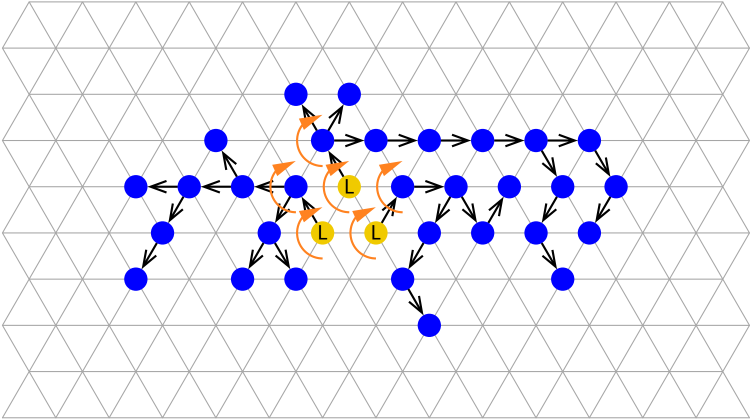
If a candidate leader succeeds to expand, it then contracts and moves back to its original location.

Handedness Agreement Phase



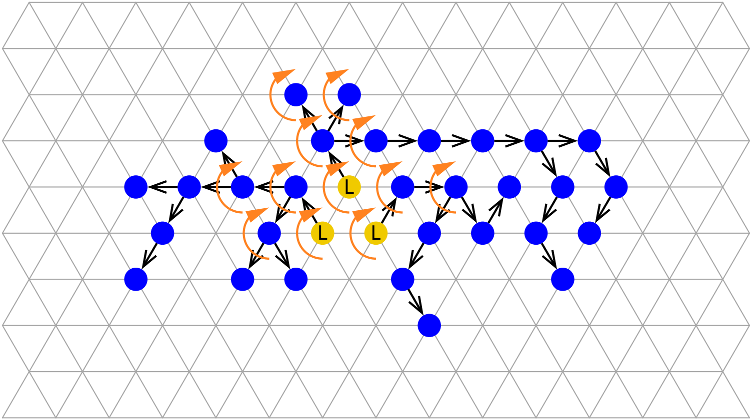
Eventually, all candidate leaders get the same handedness.

Handedness Agreement Phase

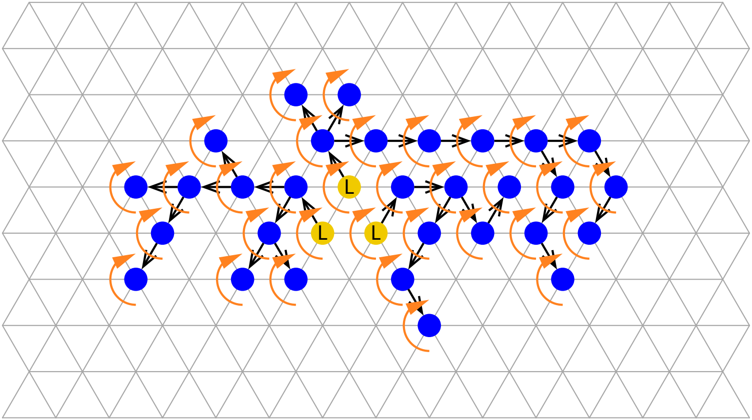


By a similar protocol, the agreed-upon handedness is communicated along the trees until all particles agree.

Handedness Agreement Phase

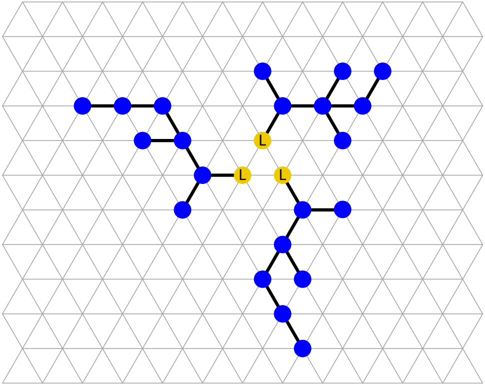


By a similar protocol, the agreed-upon handedness is communicated along the trees until all particles agree.



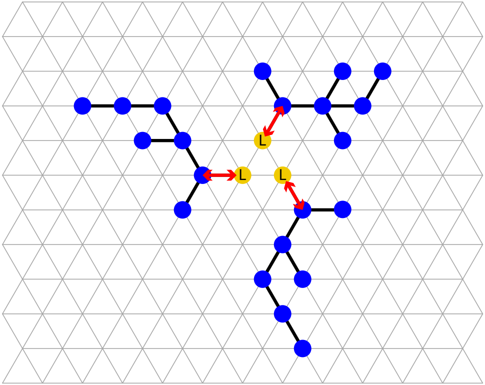
Since several instances of the protocol are taking place across the network, appropriate *locking* and *unlocking* mechanisms have to be implemented, and the absence of *deadlocks* has to be proven.

Leader Election Phase



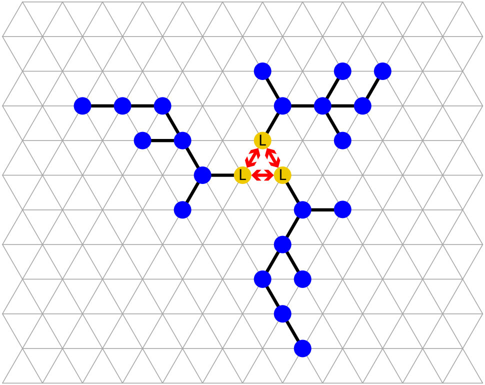
The candidate leaders want to compare their respective trees, in an attempt to break symmetry.

Leader Election Phase



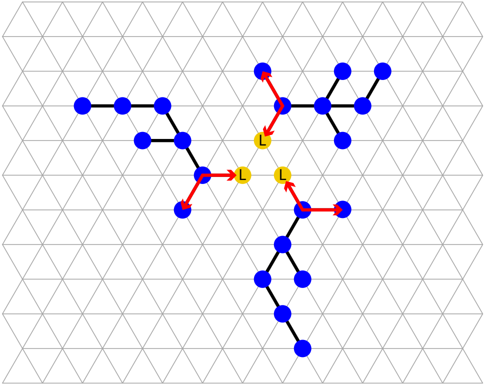
They do so by a breadth-first search, forwarding a message to a node and waiting for it to reply with a representation of its neighborhood.

Leader Election Phase



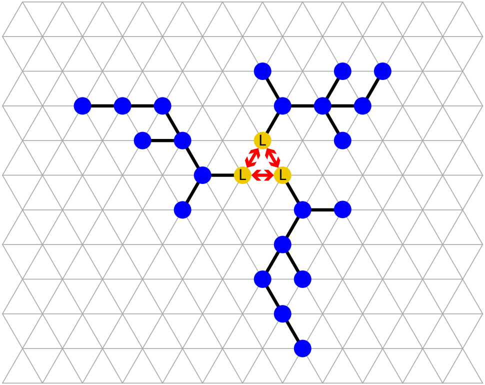
When all candidate leaders have received a reply from a node, they compare it to see if the symmetry can be broken.

Leader Election Phase



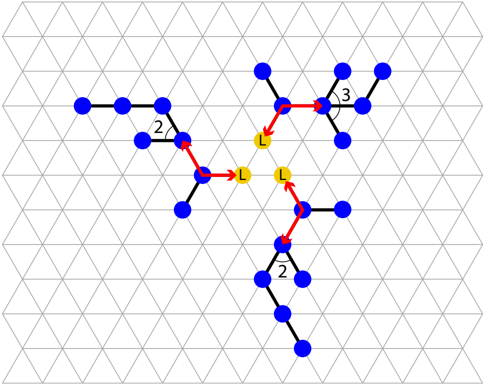
If the replies are all equal, they proceed with the next node.

Leader Election Phase



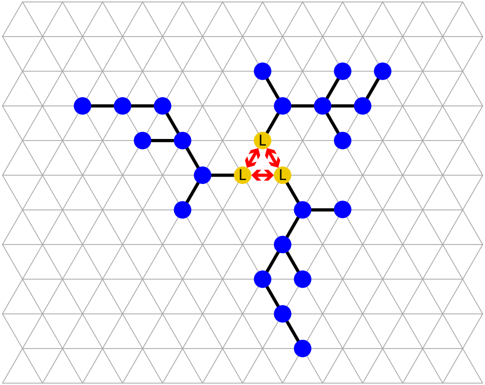
If the replies are all equal, they proceed with the next node.

Leader Election Phase



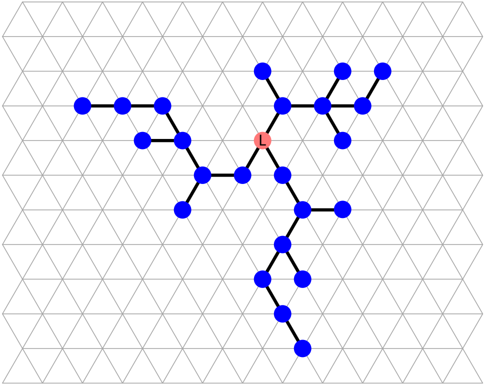
As soon as the replies are not all equal, a unique leader is elected, and the other candidate leaders become its children.

Leader Election Phase



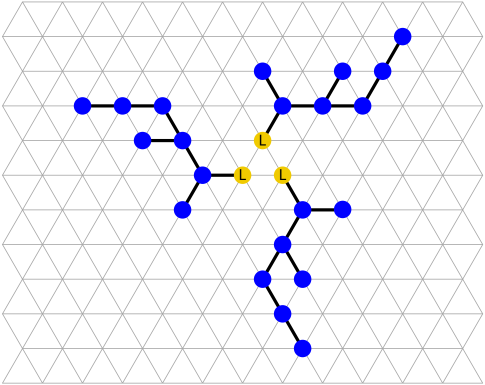
As soon as the replies are not all equal, a unique leader is elected, and the other candidate leaders become its children.

Leader Election Phase



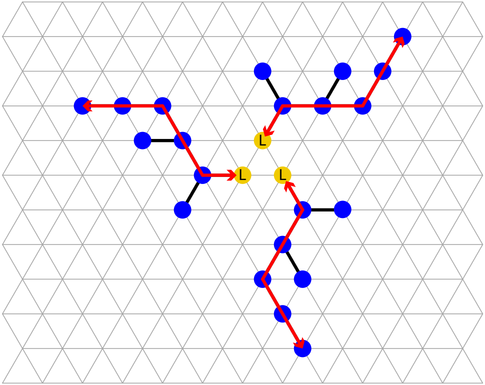
As soon as the replies are not all equal, a unique leader is elected, and the other candidate leaders become its children.

Leader Election Phase



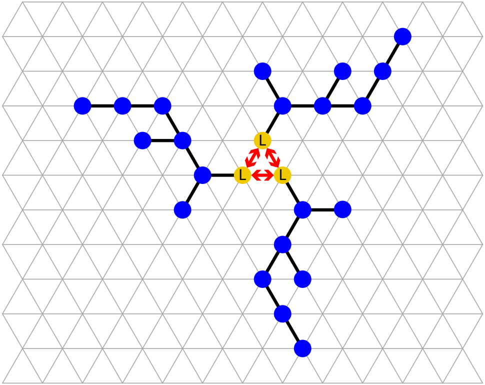
If the last node of each tree has been reached and the replies are still all equal, then the trees must be equal and equally oriented (because all particles agree on the same handedness).

Leader Election Phase



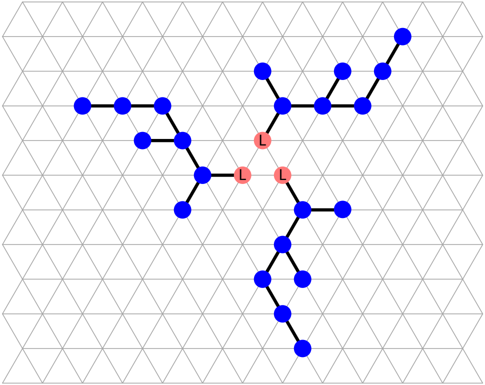
If the last node of each tree has been reached and the replies are still all equal, then the trees must be equal and equally oriented (because all particles agree on the same handedness).

Leader Election Phase



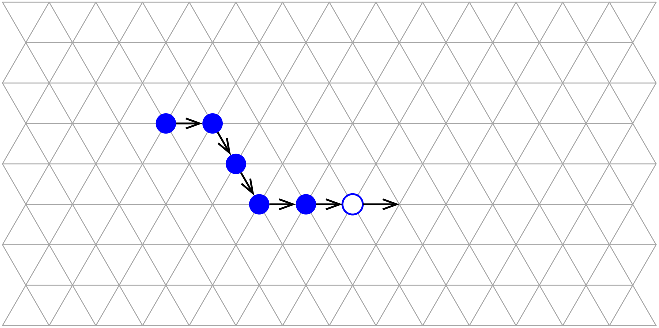
If the last node of each tree has been reached and the replies are still all equal, then the trees must be equal and equally oriented (because all particles agree on the same handedness).

Leader Election Phase



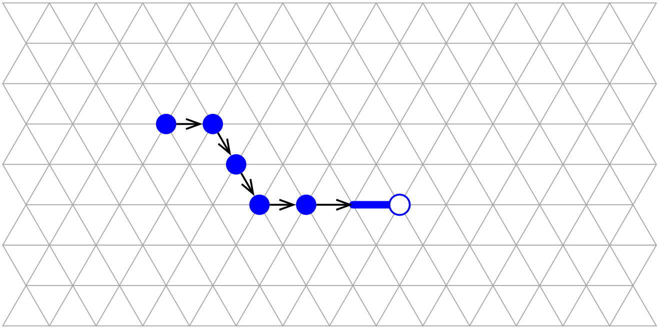
In this case the initial shape has an unbrekable symmetry, and all candidate leaders become leaders.

Basic Locomotion Protocol



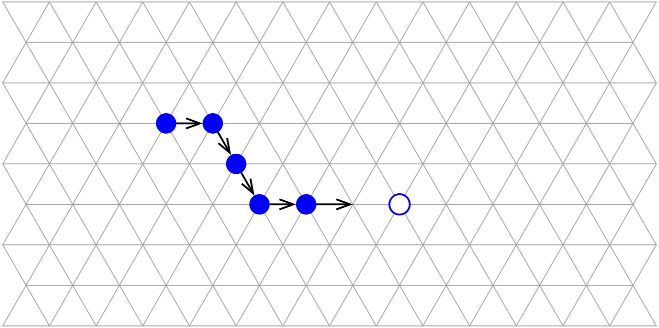
This protocol allows a chain of particles, led by a *pioneer*, to move around without leaving particles behind.

Basic Locomotion Protocol



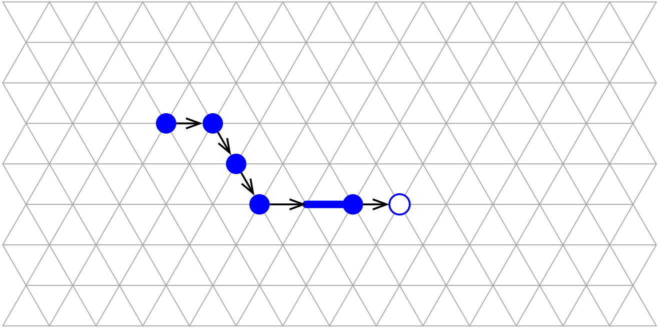
The pioneer expands in some direction and then contracts.

Basic Locomotion Protocol



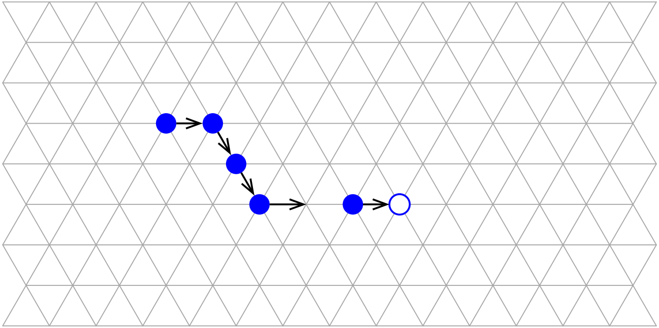
The pioneer expands in some direction and then contracts.

Basic Locomotion Protocol



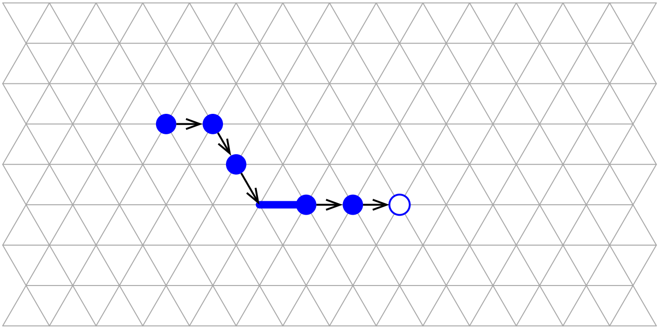
The next particle notices the absence of its parent and moves to the location where it used to be.

Basic Locomotion Protocol



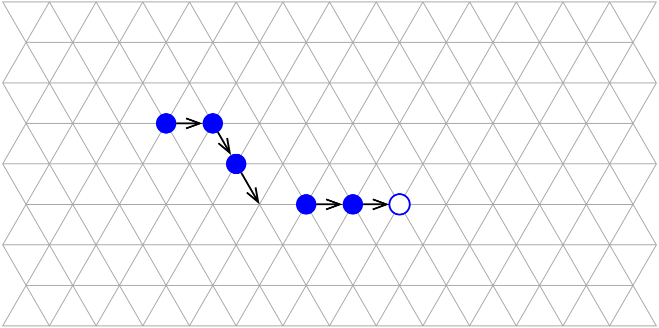
The next particle notices the absence of its parent and moves to the location where it used to be.

Basic Locomotion Protocol



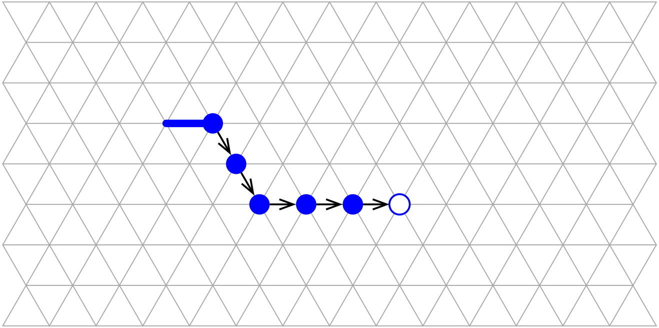
The protocol continues until the last particle has moved.

Basic Locomotion Protocol



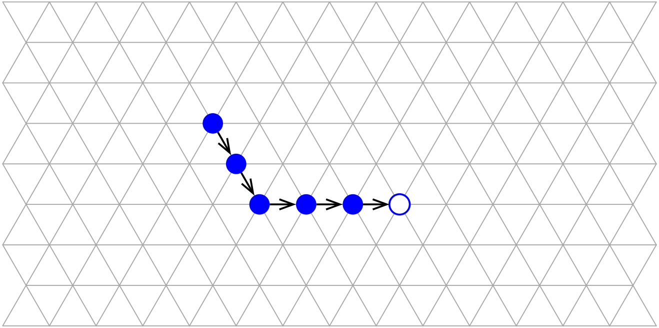
The protocol continues until the last particle has moved.

Basic Locomotion Protocol

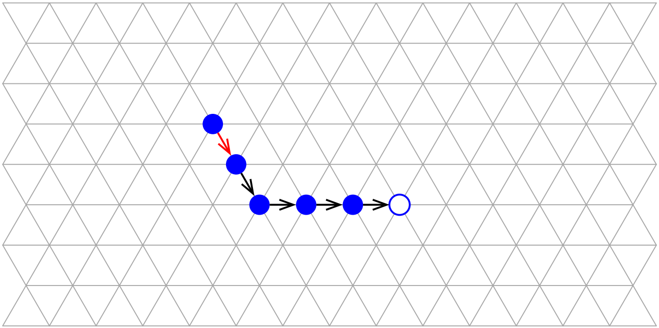


The protocol continues until the last particle has moved.

Basic Locomotion Protocol

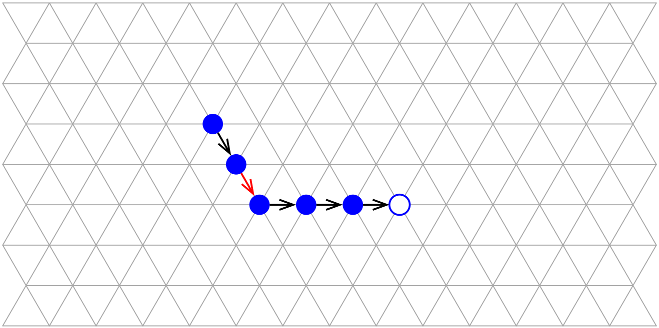


The protocol continues until the last particle has moved.



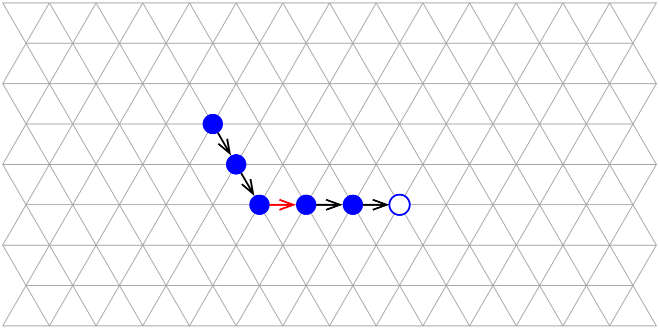
The last particle forwards a *termination* message to the pioneer.

Basic Locomotion Protocol



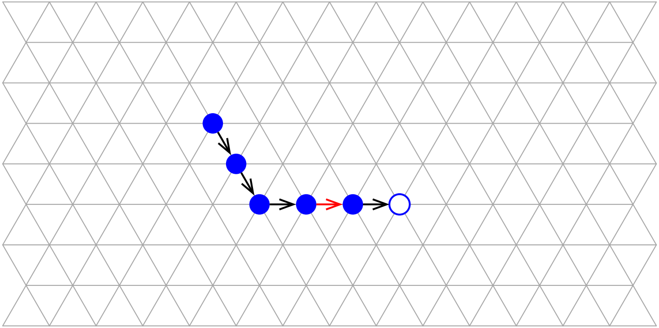
The last particle forwards a *termination* message to the pioneer.

Basic Locomotion Protocol



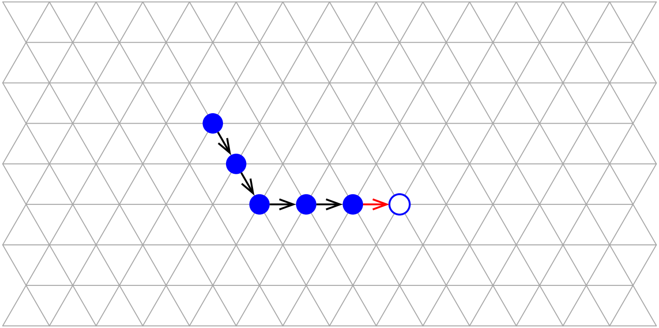
The last particle forwards a *termination* message to the pioneer.

Basic Locomotion Protocol



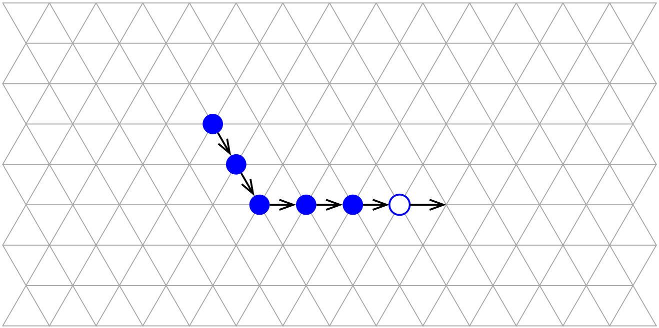
The last particle forwards a *termination* message to the pioneer.

Basic Locomotion Protocol



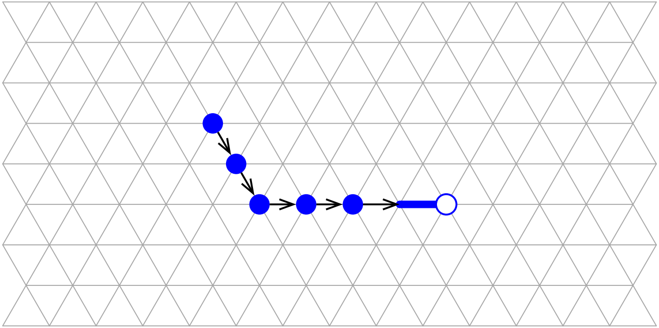
The last particle forwards a *termination* message to the pioneer.

Basic Locomotion Protocol



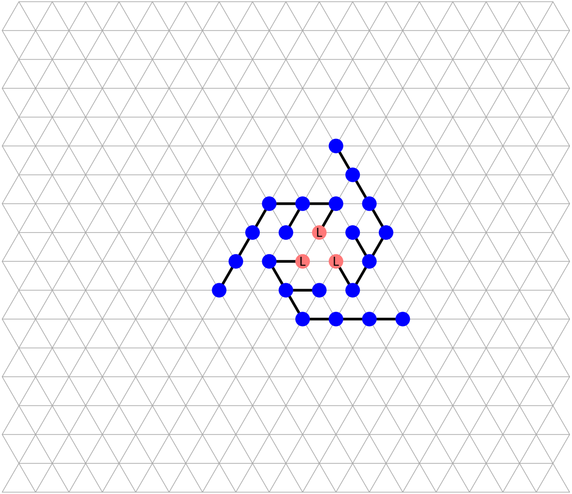
When the pioneer receives the termination message, it moves again, and the protocol repeats.

Basic Locomotion Protocol



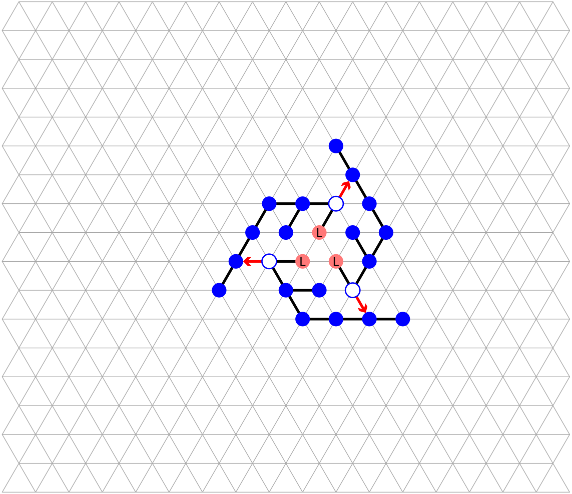
When the pioneer receives the termination message, it moves again, and the protocol repeats.

Straightening Phase



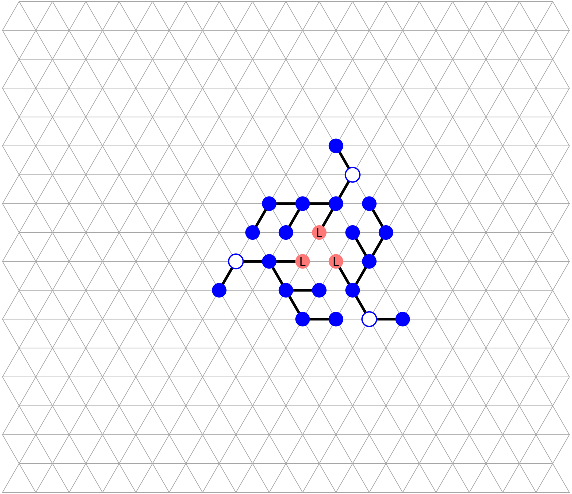
Each leader wants to transform its tree into a line segment.

Straightening Phase



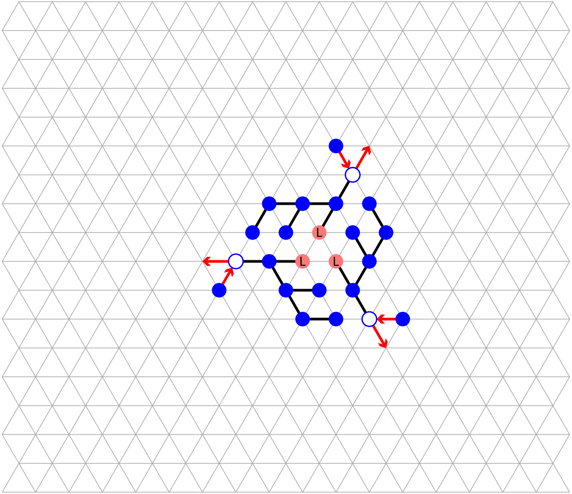
A *pioneer* is sent forth to the designated direction.

Straightening Phase



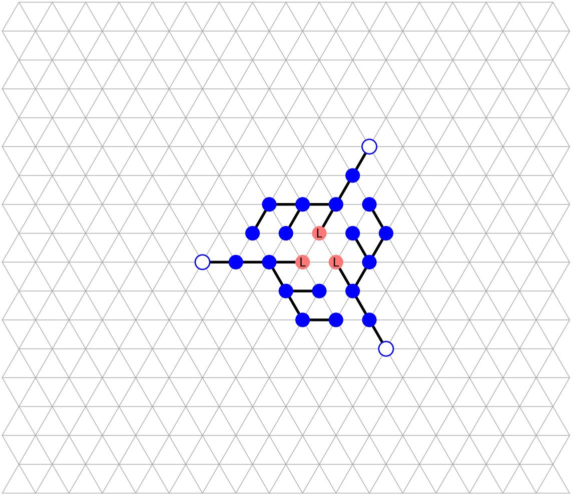
When a pioneer hits another particle,
it becomes its parent and then swaps states with it.

Straightening Phase



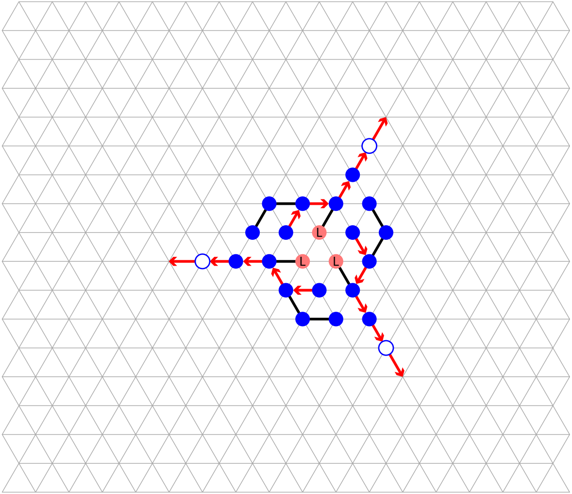
Then each pioneer keeps pulling chains of particles with the basic locomotion protocol, until its tree is straightened.

Straightening Phase

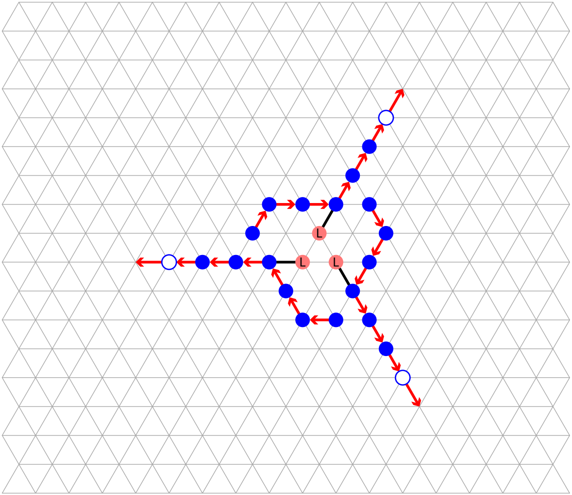


Then each pioneer keeps pulling chains of particles with the basic locomotion protocol, until its tree is straightened.

Straightening Phase

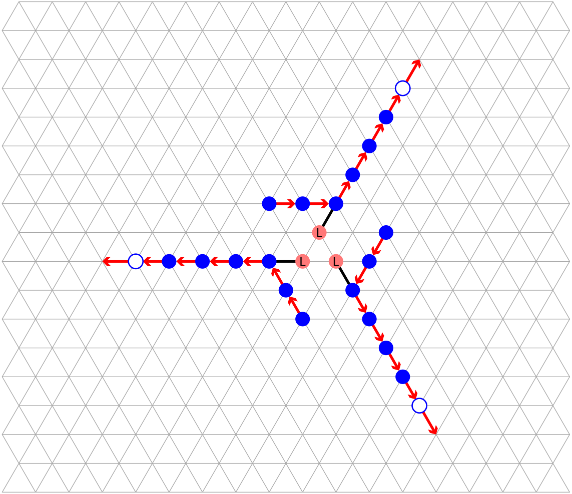


Then each pioneer keeps pulling chains of particles with the basic locomotion protocol, until its tree is straightened.



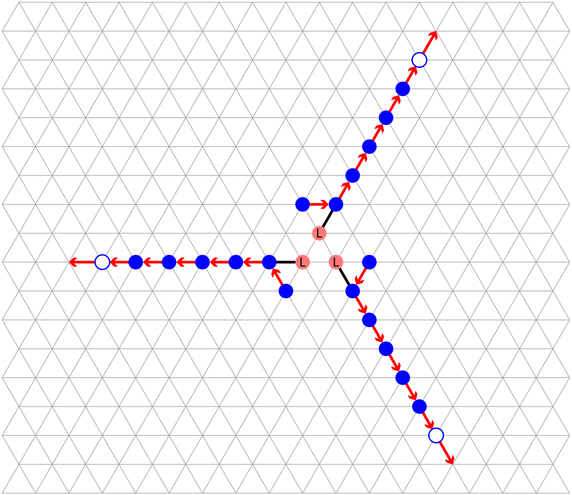
Then each pioneer keeps pulling chains of particles with the basic locomotion protocol, until its tree is straightened.

Straightening Phase



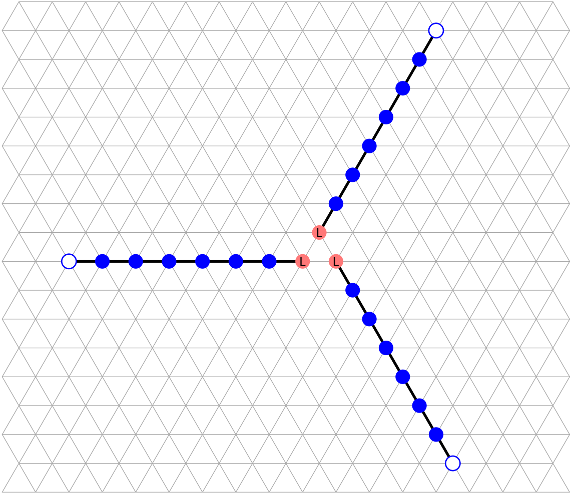
Then each pioneer keeps pulling chains of particles with the basic locomotion protocol, until its tree is straightened.

Straightening Phase



Then each pioneer keeps pulling chains of particles with the basic locomotion protocol, until its tree is straightened.

Straightening Phase



The lines must have the same length, so the leaders communicate with each other to make their pioneers move at the same pace.

Random-Access Machines

A random-access machine is a model of computation with:

- some *registers*, each storing a non-negative integer
- a finite *program* consisting of only 3 types of instructions:
  - <u>increment</u> the value stored in a register by 1
  - if the value stored in a register is positive, <u>decrement</u> it by 1
  - <u>test</u> the value of a register and <u>branch</u> if it is 0

Random-Access Machines

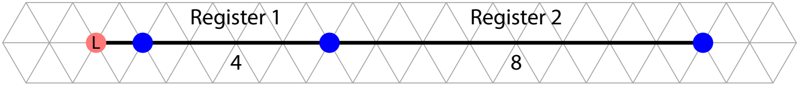
A random-access machine is a model of computation with:

- some *registers*, each storing a non-negative integer
- a finite *program* consisting of only 3 types of instructions:
  - <u>increment</u> the value stored in a register by 1
  - if the value stored in a register is positive, <u>decrement</u> it by 1
  - <u>test</u> the value of a register and <u>branch</u> if it is 0

Theorem (Minsky, 1967)

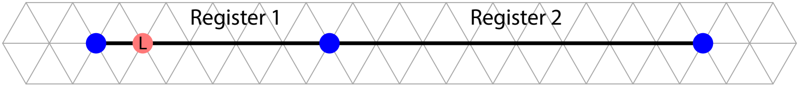
Any Turing machine can be simulated by a random-access machine with only 2 registers, the first of which initially contains the input.

Simulating a Random-Access Machine with 2 Registers



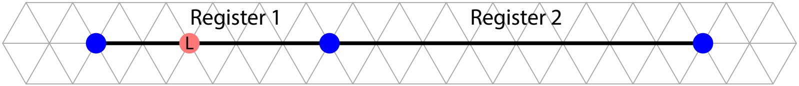
A random-access machine with 2 registers can be simulated by 4 particles: a *leader*, which executes the program, and 3 particles whose distances correspond to the values stored in the 2 registers.

Simulating a Random-Access Machine with 2 Registers



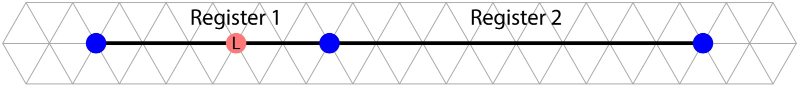
If the leader has to increment the value of the first register, it pulls the last two particles to the right by one step, and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers

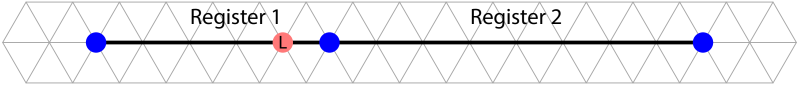


If the leader has to increment the value of the first register, it pulls the last two particles to the right by one step, and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers

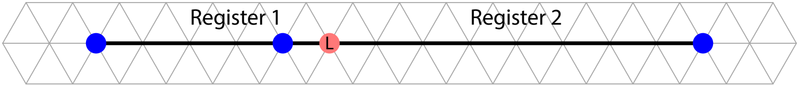


If the leader has to increment the value of the first register, it pulls the last two particles to the right by one step, and then goes back to its original position.



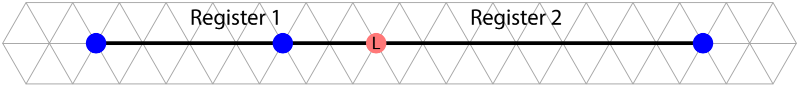
If the leader has to increment the value of the first register, it pulls the last two particles to the right by one step, and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers

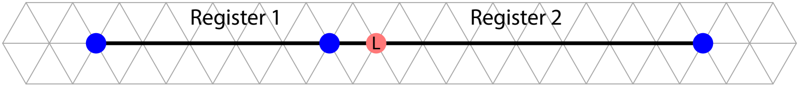


If the leader has to increment the value of the first register, it pulls the last two particles to the right by one step, and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers

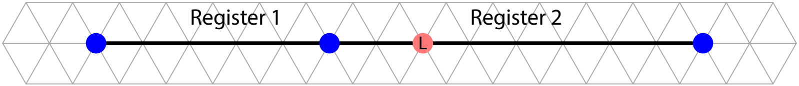


If the leader has to increment the value of the first register, it pulls the last two particles to the right by one step, and then goes back to its original position.



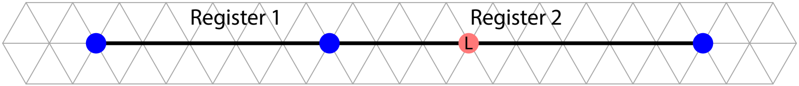
If the leader has to increment the value of the first register, it pulls the last two particles to the right by one step, and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers



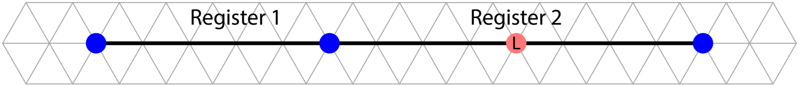
If the leader has to increment the value of the first register, it pulls the last two particles to the right by one step, and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers



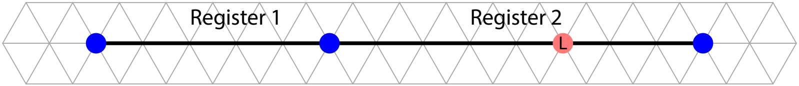
If the leader has to increment the value of the first register, it pulls the last two particles to the right by one step, and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers



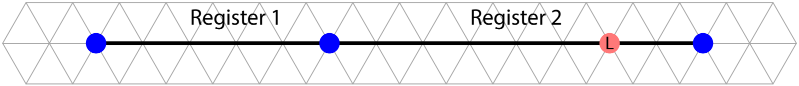
If the leader has to increment the value of the first register, it pulls the last two particles to the right by one step, and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers



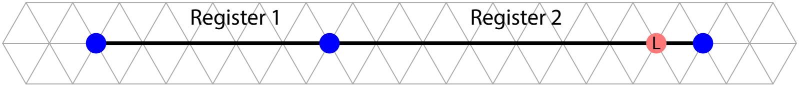
If the leader has to increment the value of the first register, it pulls the last two particles to the right by one step, and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers



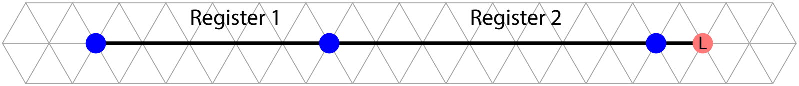
If the leader has to increment the value of the first register, it pulls the last two particles to the right by one step, and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers

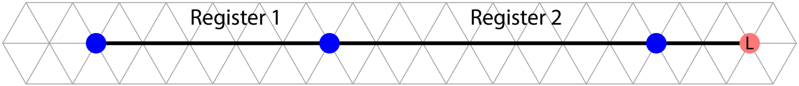


If the leader has to increment the value of the first register, it pulls the last two particles to the right by one step, and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers

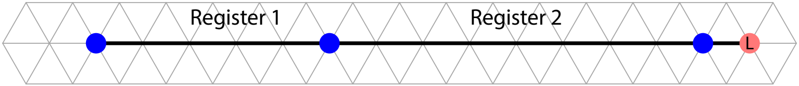


If the leader has to increment the value of the first register, it pulls the last two particles to the right by one step, and then goes back to its original position.



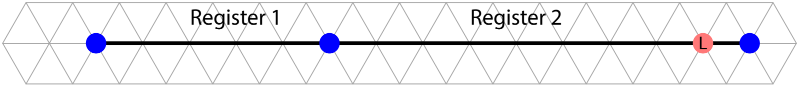
If the leader has to increment the value of the first register, it pulls the last two particles to the right by one step, and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers



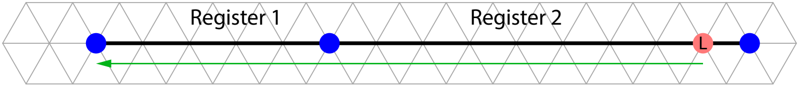
If the leader has to increment the value of the first register, it pulls the last two particles to the right by one step, and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers



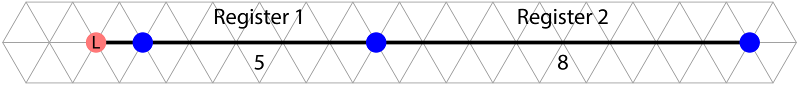
If the leader has to increment the value of the first register, it pulls the last two particles to the right by one step, and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers

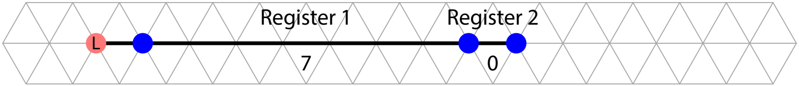


If the leader has to increment the value of the first register, it pulls the last two particles to the right by one step, and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers

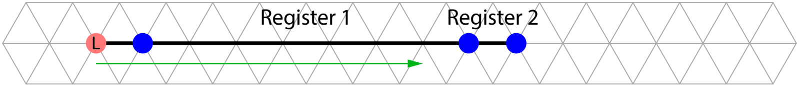


If the leader has to increment the value of the first register, it pulls the last two particles to the right by one step, and then goes back to its original position.



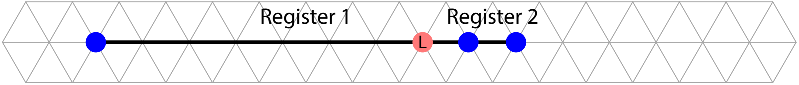
If the leader has to test if the value of the second register is 0, it reaches the second-to-last particle and exchanges messages with it, asking if the last particle is adjacent to it.

Simulating a Random-Access Machine with 2 Registers

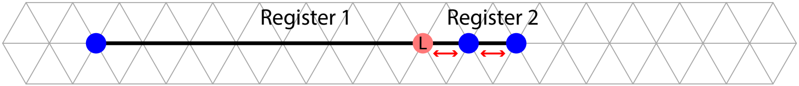


If the leader has to test if the value of the second register is 0, it reaches the second-to-last particle and exchanges messages with it, asking if the last particle is adjacent to it.

Simulating a Random-Access Machine with 2 Registers

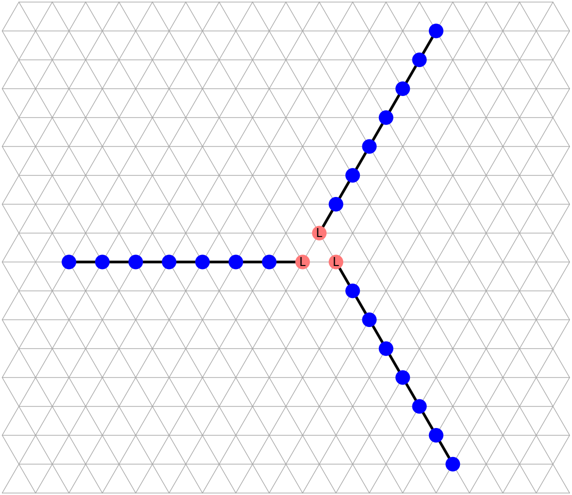


If the leader has to test if the value of the second register is 0, it reaches the second-to-last particle and exchanges messages with it, asking if the last particle is adjacent to it.



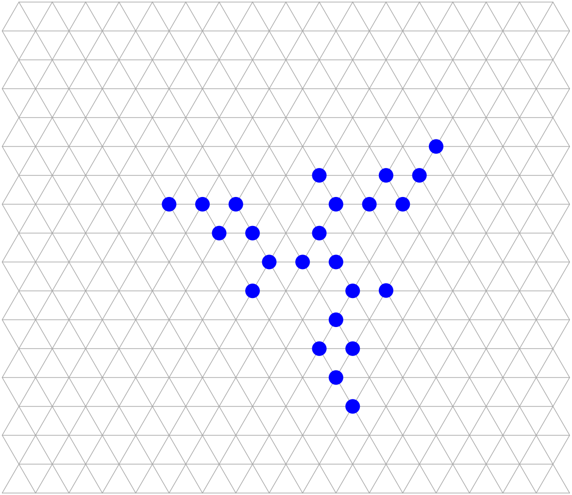
If the leader has to test if the value of the second register is 0, it reaches the second-to-last particle and exchanges messages with it, asking if the last particle is adjacent to it.

Shape Formation Phase



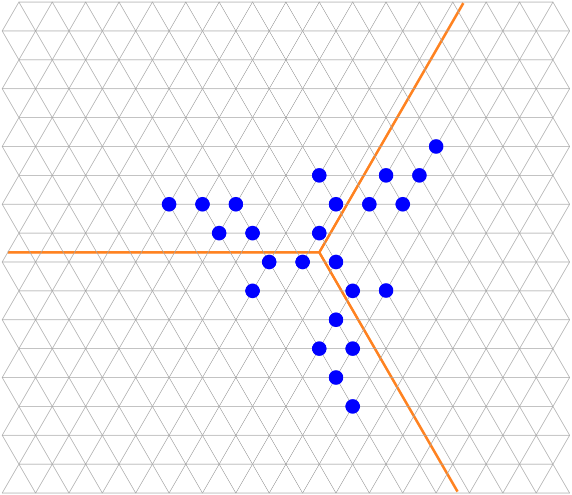
If $k > 1$ leaders have been elected in the previous phases, it means that the initial shape has an unbreakable k-fold symmetry.

Shape Formation Phase



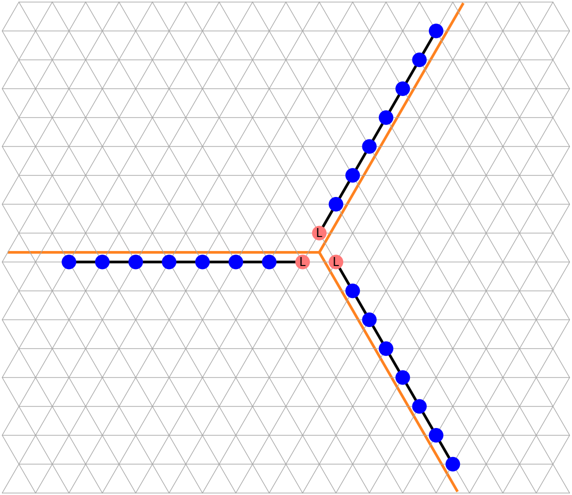
Hence, we may assume that also the shape to be formed has the same k-fold symmetry.

Shape Formation Phase



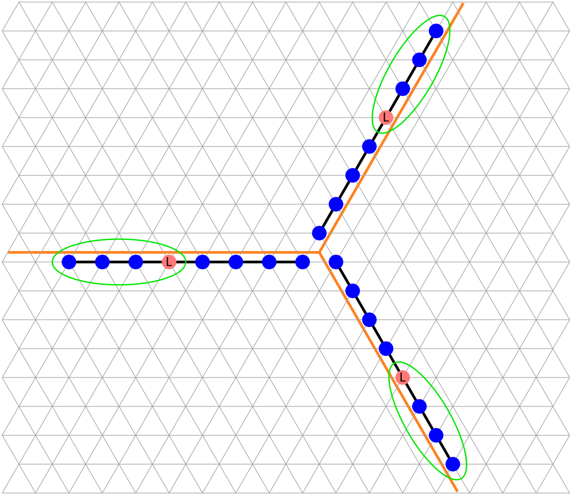
The plane is partitioned into k sectors, and each leader is tasked with forming the part of the shape that falls in its sector.

Shape Formation Phase



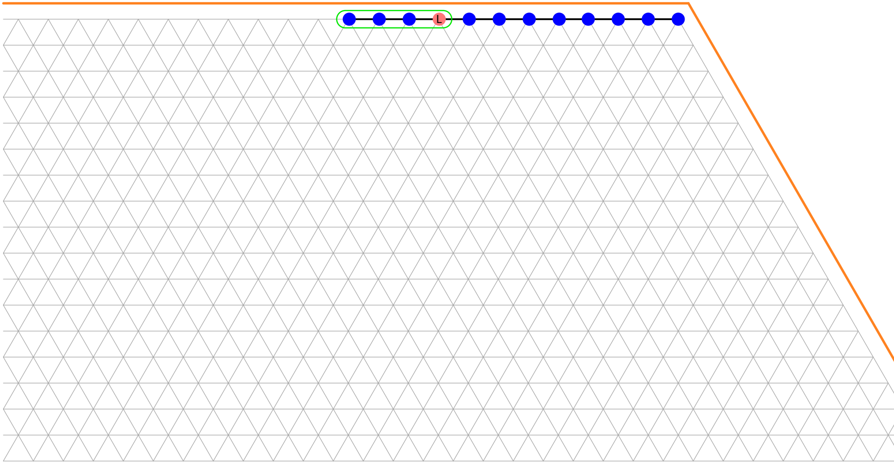
The plane is partitioned into k sectors, and each leader is tasked with forming the part of the shape that falls in its sector.

Shape Formation Phase



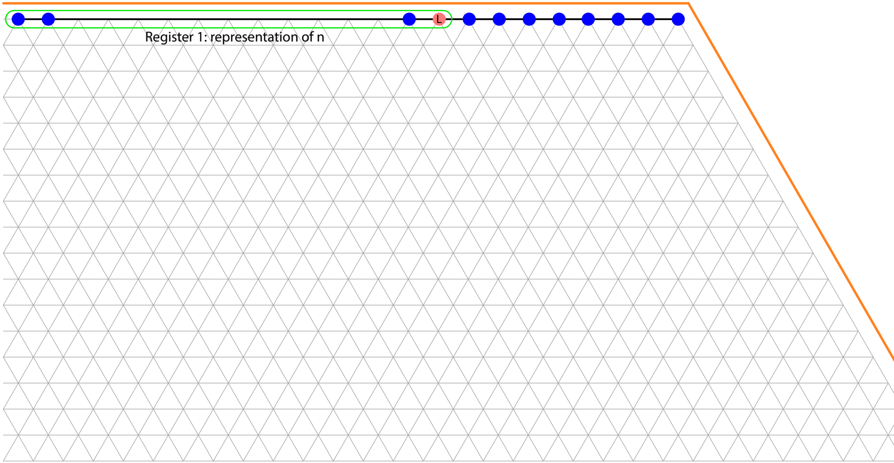
Assume there is an algorithm that, given n, generates the points of the shape. Let each leader simulate a RAM for that algorithm.

Shape Formation Phase



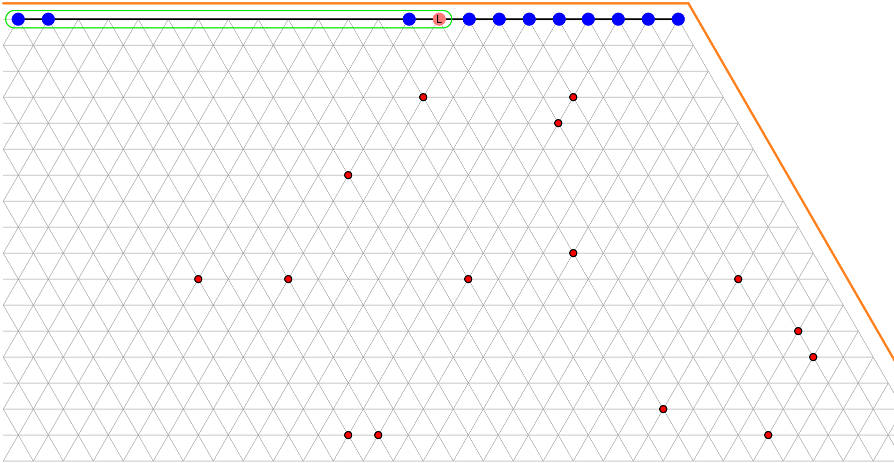
The leader takes position at the beginning of the simulated RAM.

Shape Formation Phase



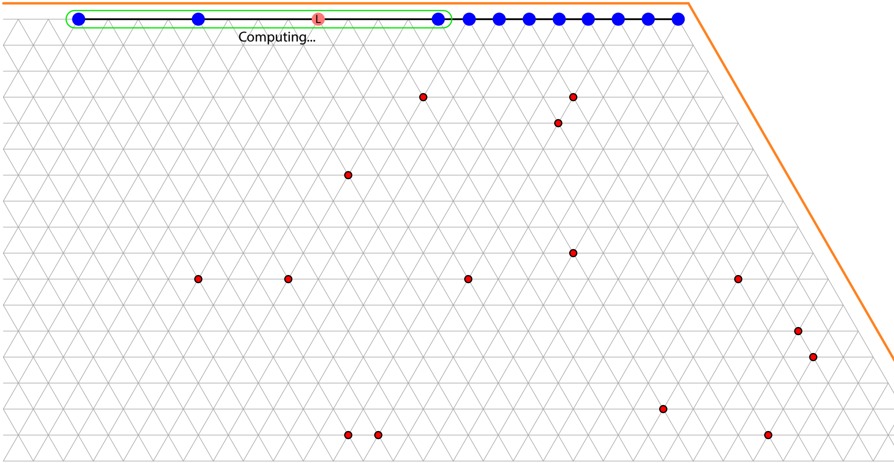
By scanning the previous part of the chain, it constructs a representation of n in the first register, which serves as the input.

Shape Formation Phase



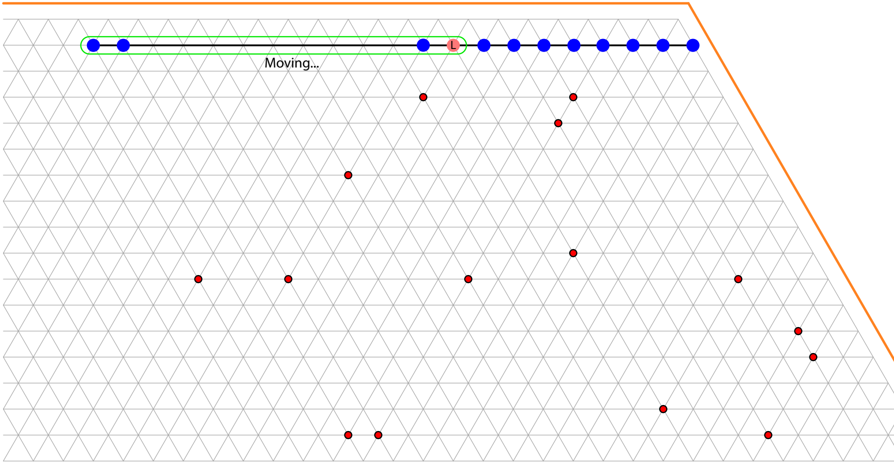
The simulated RAM will generate all the points of the shape and the sequence of moves necessary to reach them.

Shape Formation Phase



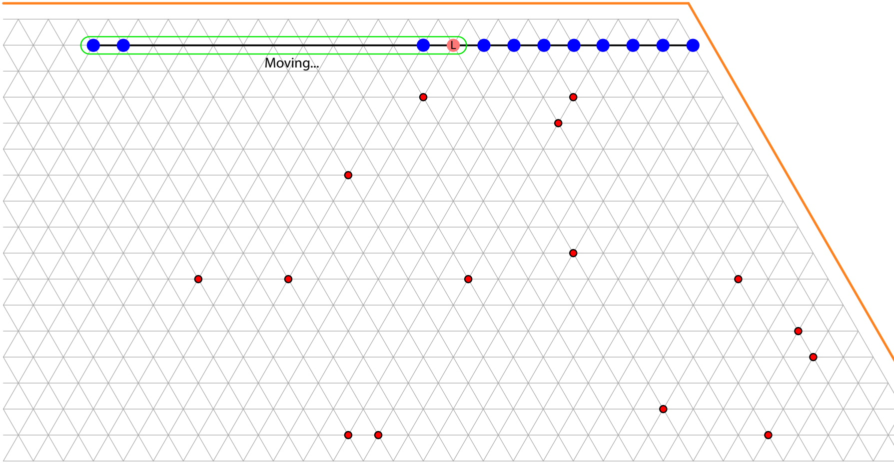
The simulated RAM computes the first point of the shape, while the rest of the chain does not move.

Shape Formation Phase



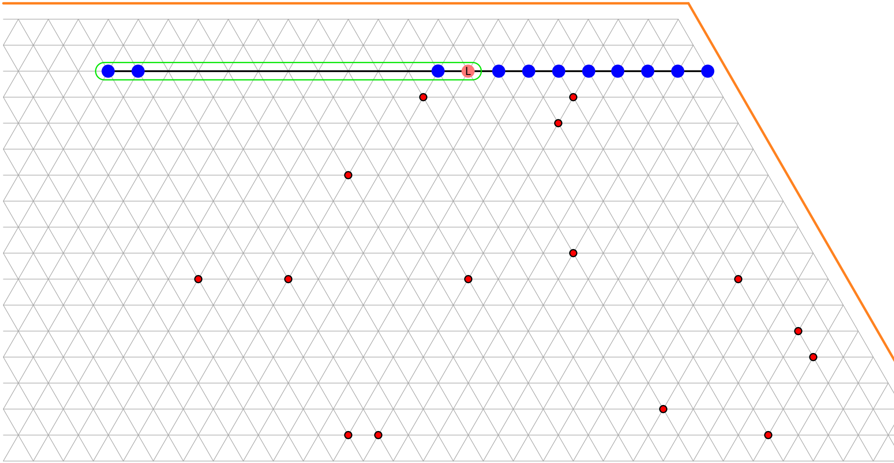
When the RAM has finished, the value of the first register indicates that the chain has to move in some direction.

Shape Formation Phase



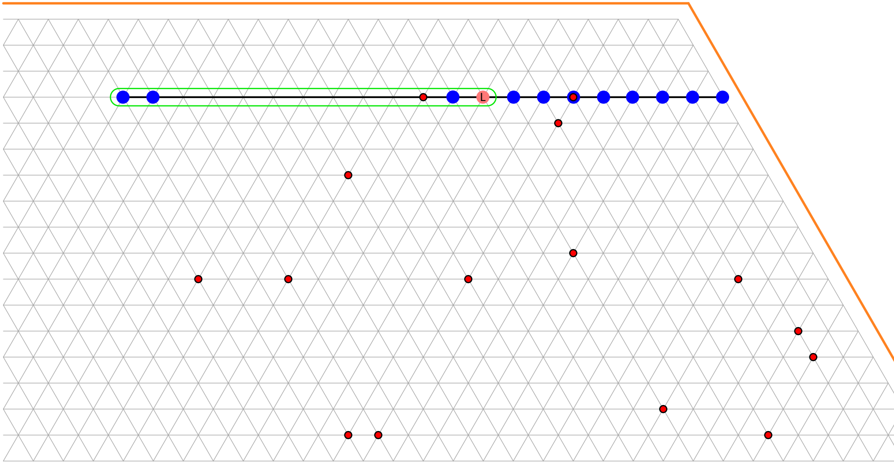
(The movement of the whole chain is coordinated by the leader, and takes place one particle at a time.)

Shape Formation Phase



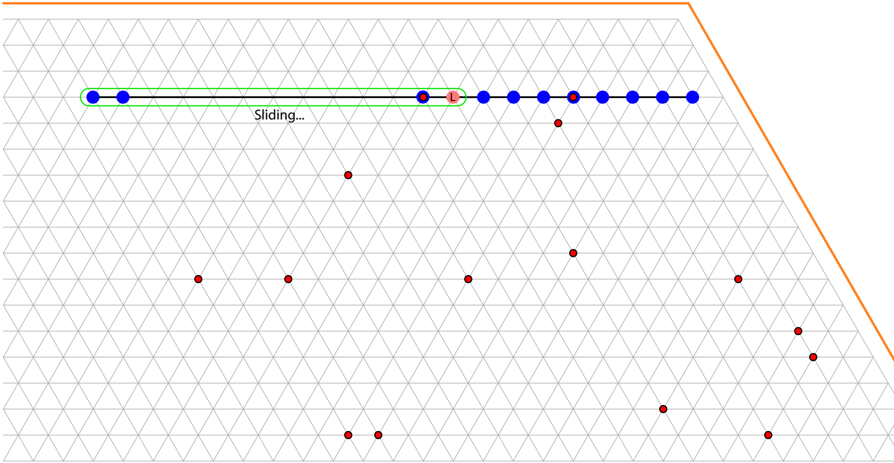
The RAM computes the next movement,
and the whole chain moves as soon as the computation is finished.

Shape Formation Phase



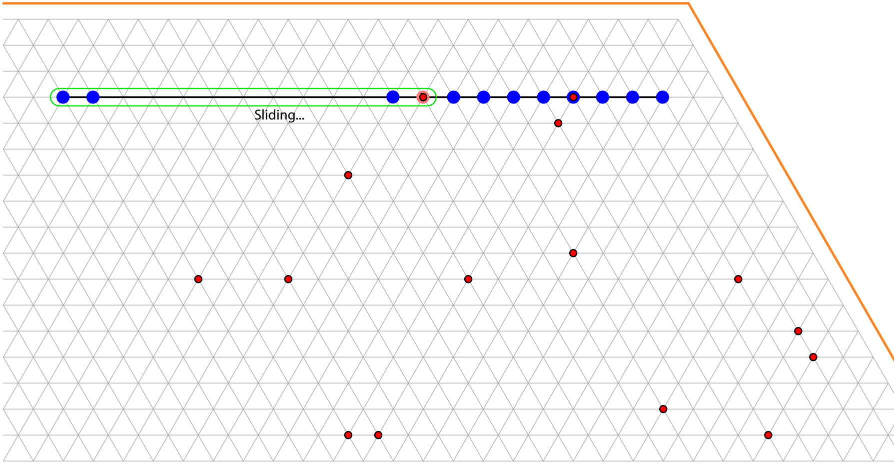
The RAM computes the next movement,
and the whole chain moves as soon as the computation is finished.

Shape Formation Phase



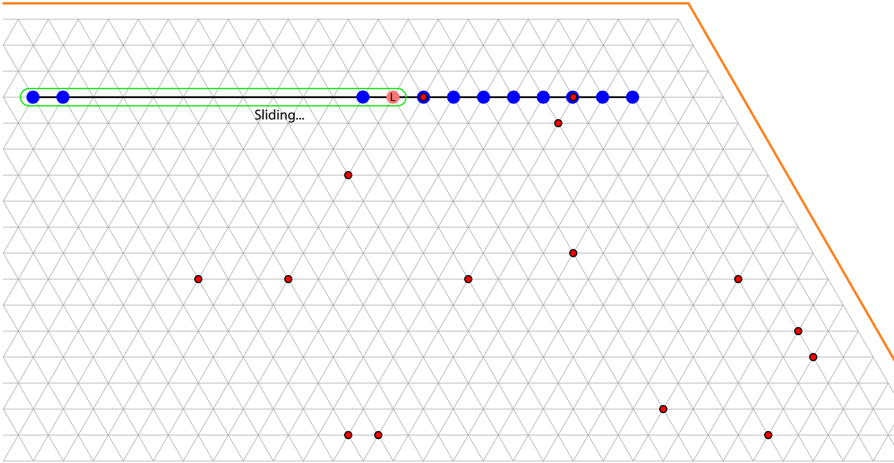
When the chain is on the same line as the first point of the shape, it slides until the last particle of the chain coincides with the point.

Shape Formation Phase



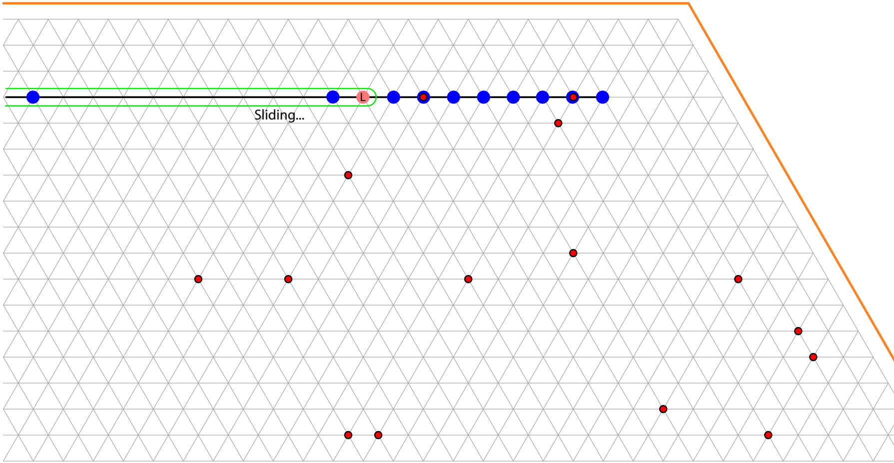
When the chain is on the same line as the first point of the shape, it slides until the last particle of the chain coincides with the point.

Shape Formation Phase



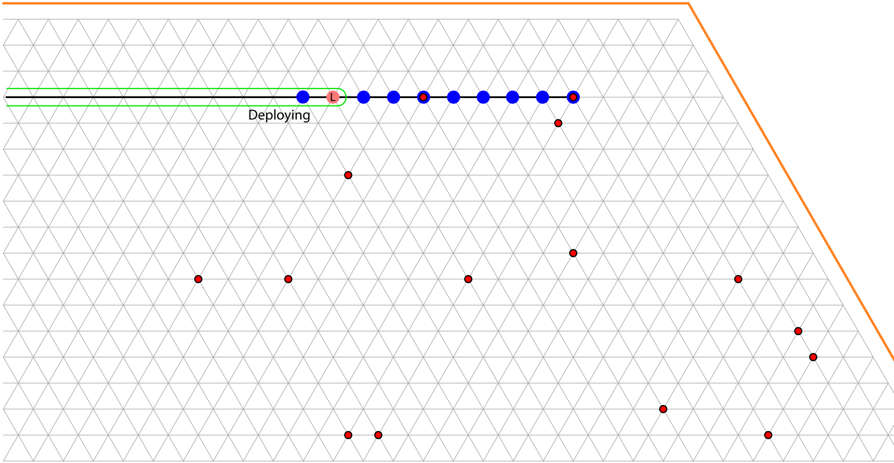
When the chain is on the same line as the first point of the shape, it slides until the last particle of the chain coincides with the point.

Shape Formation Phase

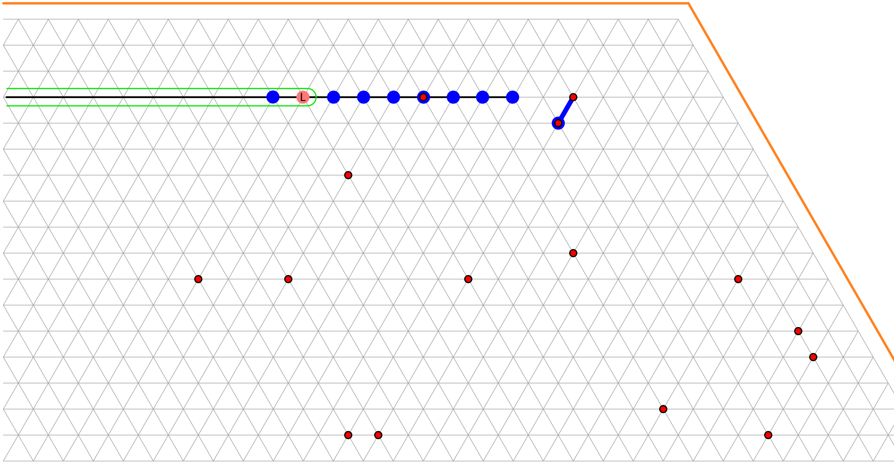


When the chain is on the same line as the first point of the shape, it slides until the last particle of the chain coincides with the point.

Shape Formation Phase

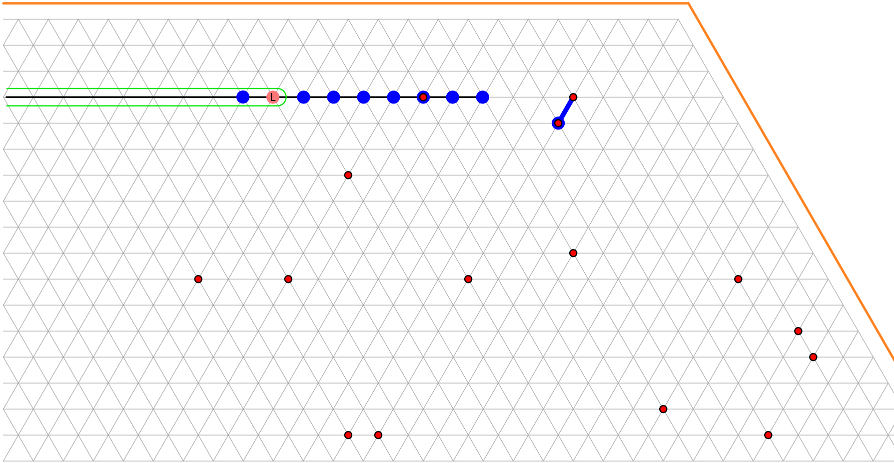


A message is forwarded to the last particle, telling it to stay there, and perhaps expand in some direction to cover two points.



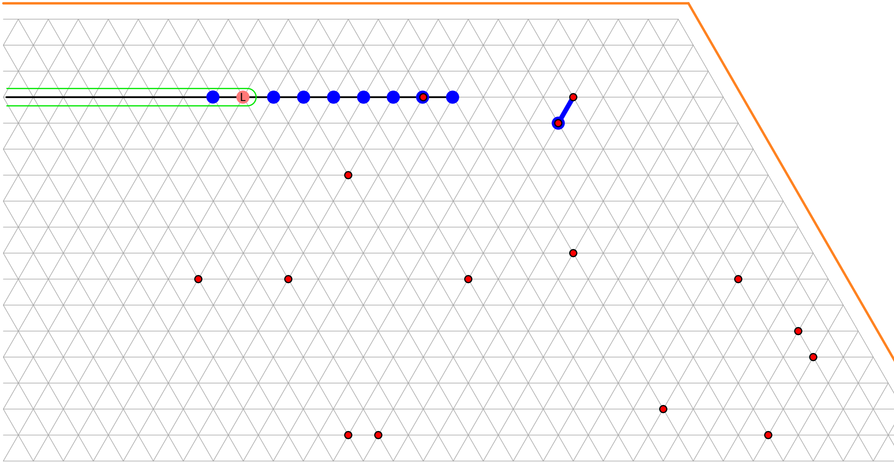
A message is forwarded to the last particle, telling it to stay there, and perhaps expand in some direction to cover two points.

Shape Formation Phase



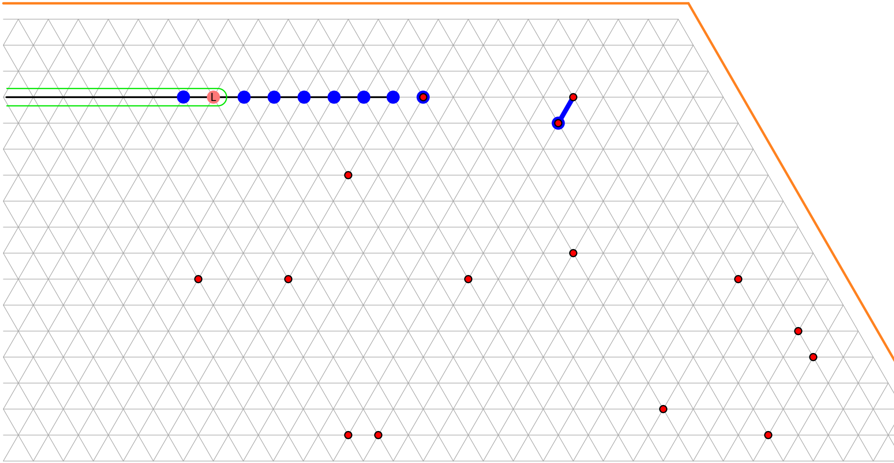
The protocol proceeds in the same fashion with the other points of the shape.

Shape Formation Phase



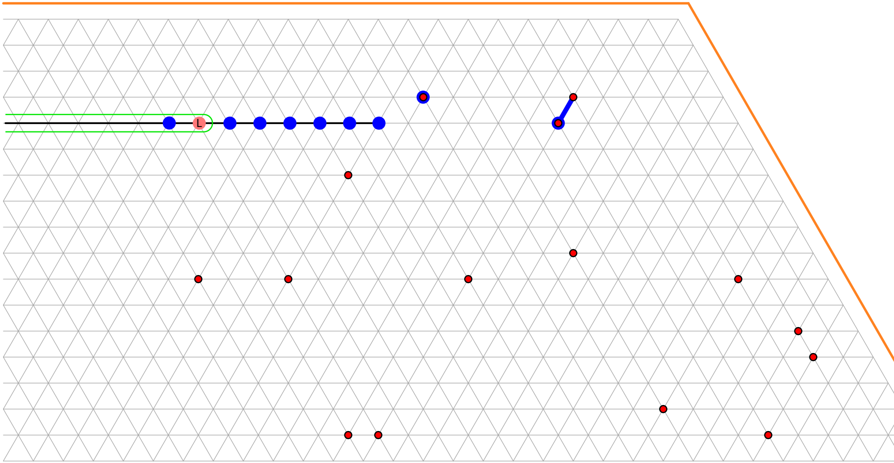
The protocol proceeds in the same fashion with the other points of the shape.

Shape Formation Phase



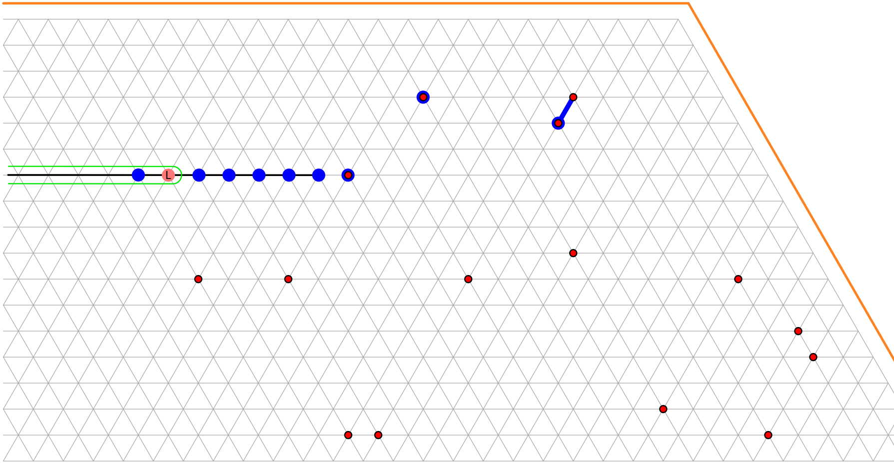
The protocol proceeds in the same fashion with the other points of the shape.

Shape Formation Phase



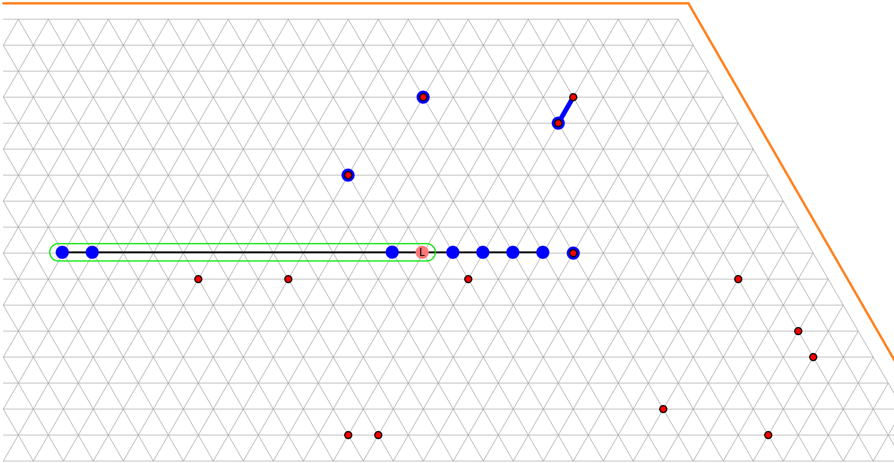
The protocol proceeds in the same fashion with the other points of the shape.

Shape Formation Phase



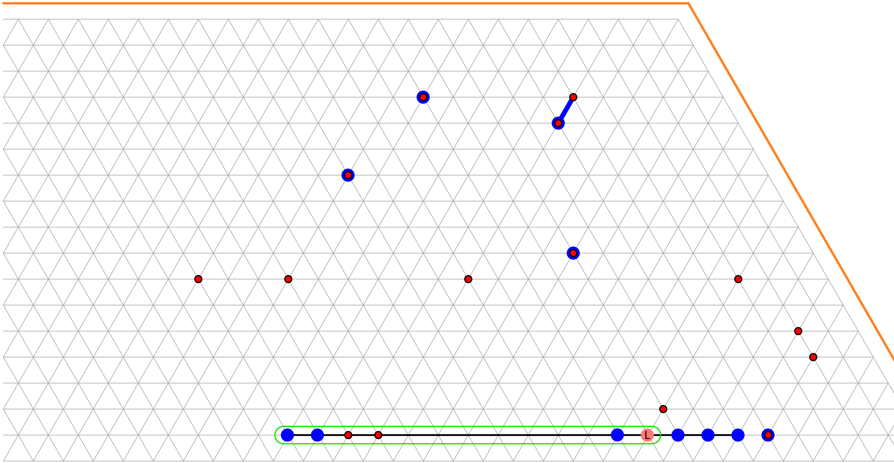
The protocol proceeds in the same fashion with the other points of the shape.

Shape Formation Phase



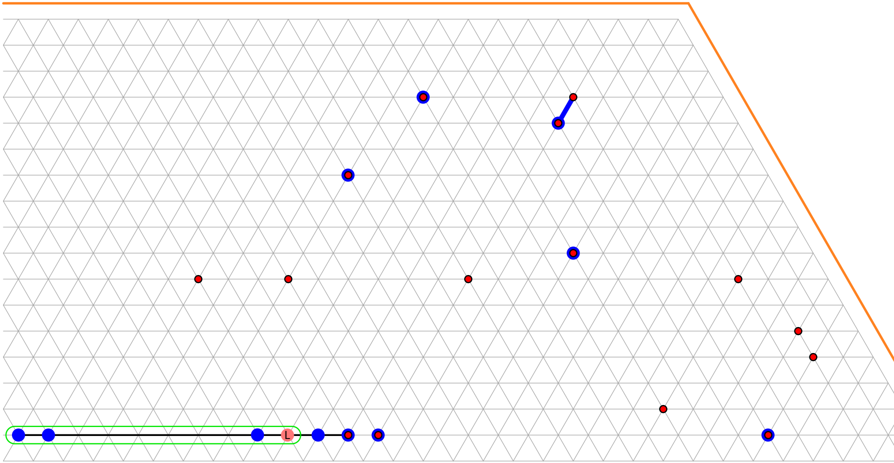
The protocol proceeds in the same fashion with the other points of the shape.

Shape Formation Phase



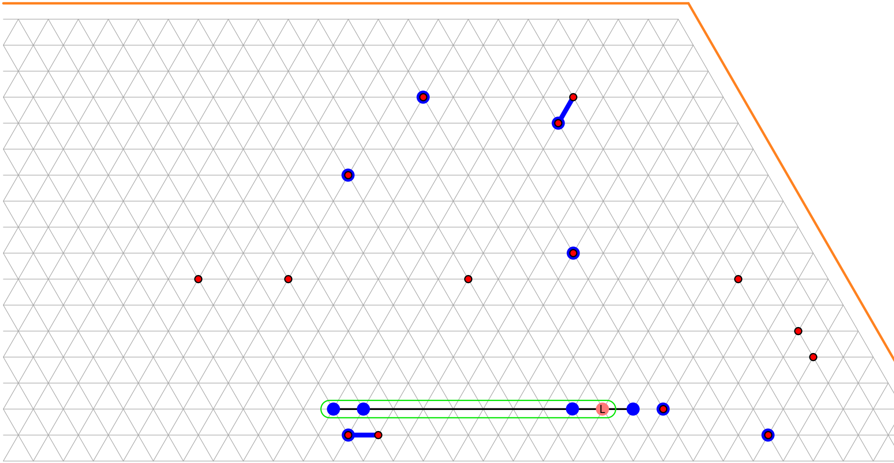
The protocol proceeds in the same fashion with the other points of the shape.

Shape Formation Phase



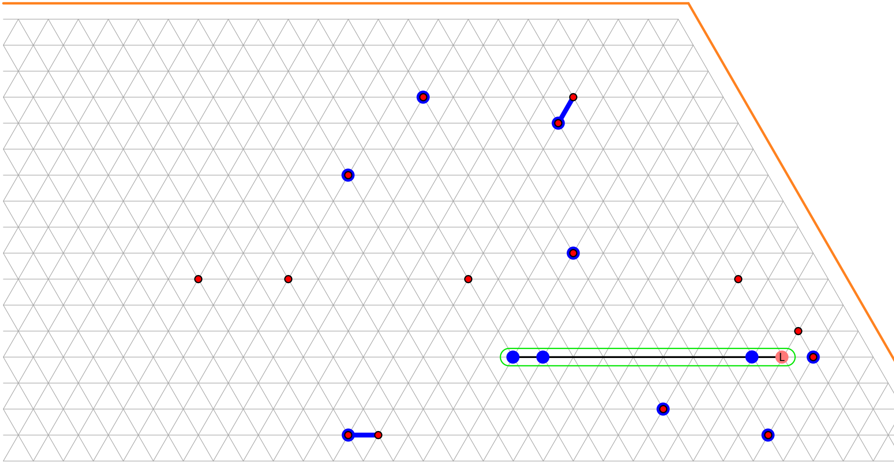
The protocol proceeds in the same fashion with the other points of the shape.

Shape Formation Phase



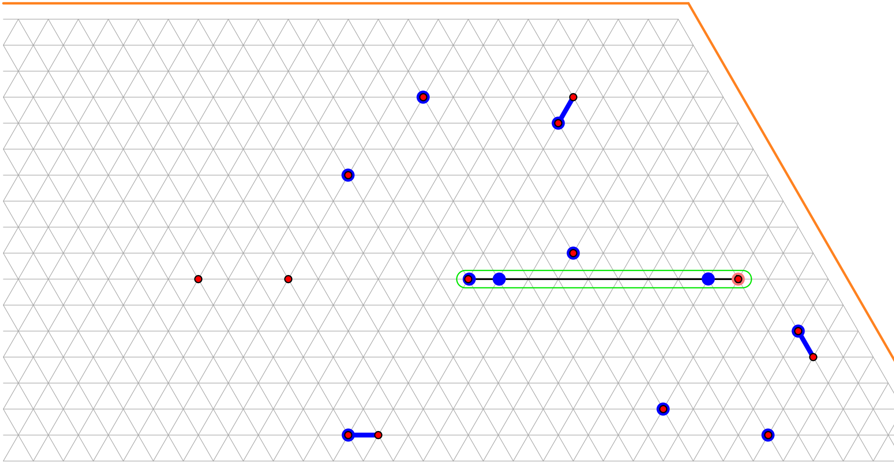
The protocol proceeds in the same fashion with the other points of the shape.

Shape Formation Phase



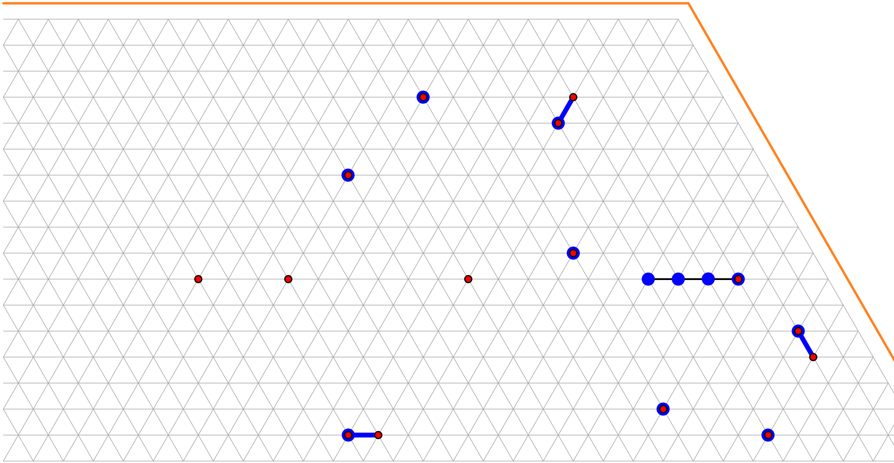
The protocol proceeds in the same fashion with the other points of the shape.

Shape Formation Phase



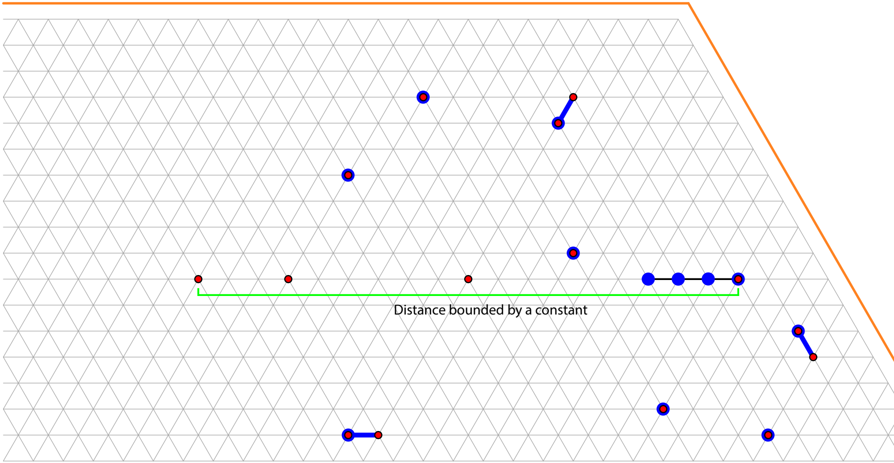
The algorithm ensures that the last 4 points of the shape are on the same line parallel to the chain of particles.

Shape Formation Phase



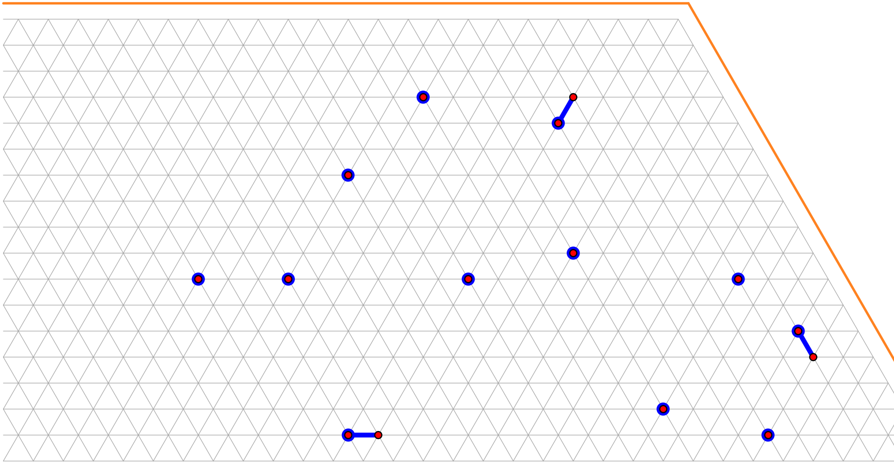
When the leader is on the first of these 4 points,
it makes the RAM contract, erasing the registers.

Shape Formation Phase



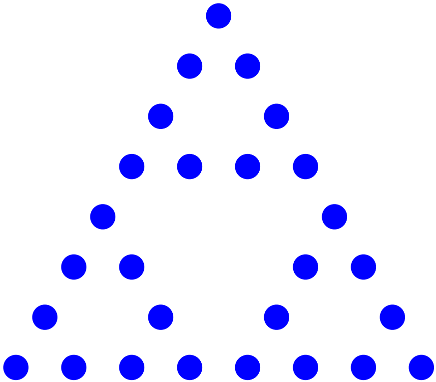
Assuming that the distance of the other 3 points is bounded by a constant, the particles can reach them using constant memory.

Shape Formation Phase



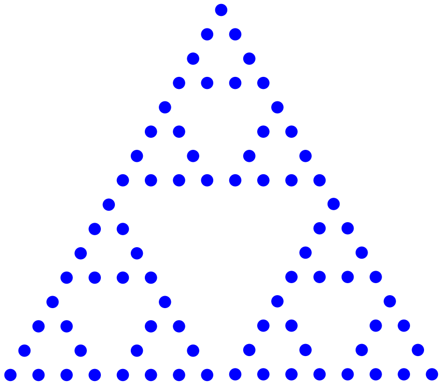
Assuming that the distance of the other 3 points is bounded by a constant, the particles can reach them using constant memory.

Fractal Shapes



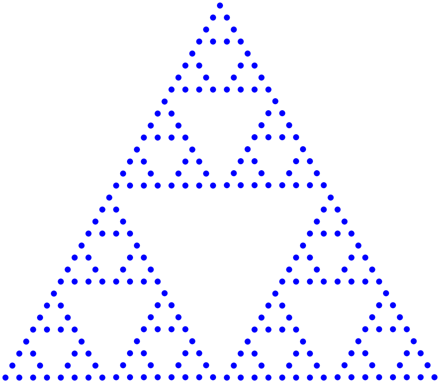
This protocol allows the system to form shapes that scale up like fractals (e.g., the Sierpinski triangle).

Fractal Shapes



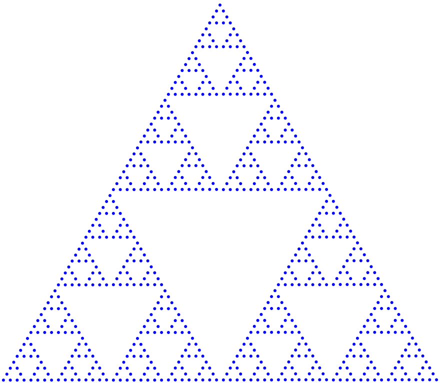
This protocol allows the system to form shapes that scale up like fractals (e.g., the Sierpinski triangle).

Fractal Shapes



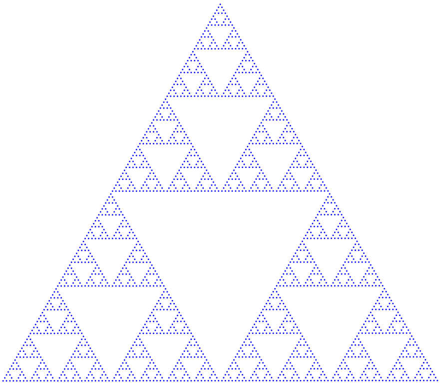
This protocol allows the system to form shapes that scale up like fractals (e.g., the Sierpinski triangle).

Fractal Shapes



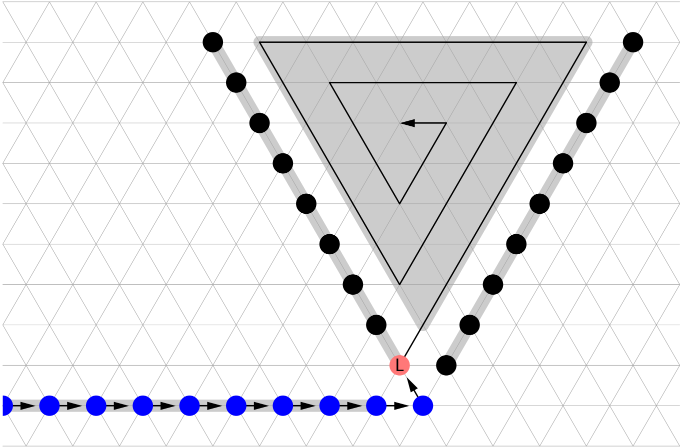
This protocol allows the system to form shapes that scale up like fractals (e.g., the Sierpinski triangle).

Fractal Shapes



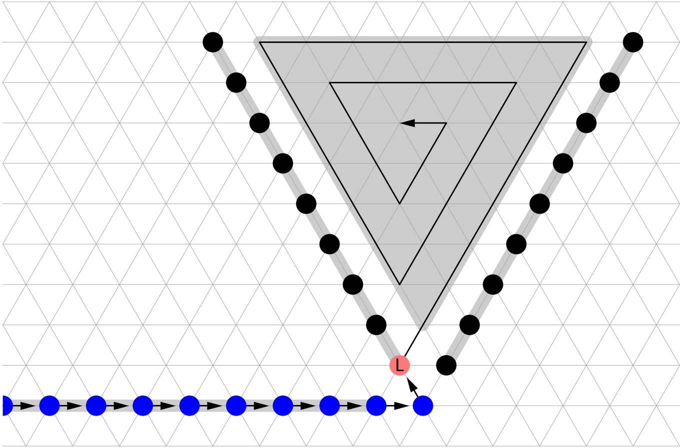
This protocol allows the system to form shapes that scale up like fractals (e.g., the Sierpinski triangle).

Forming Segments and Full Triangles



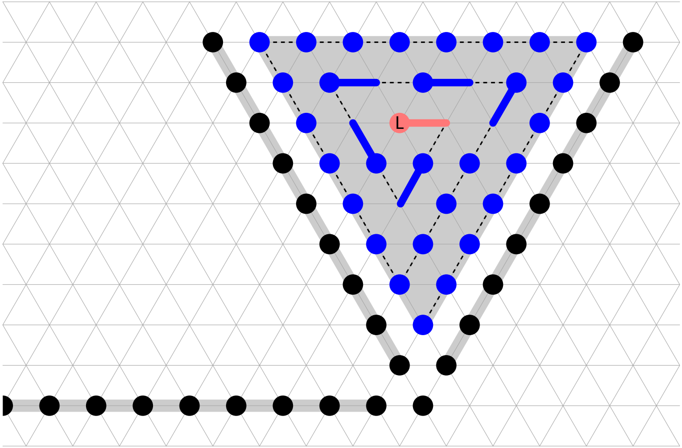
To form a general shape, the total number of *moves* taken by the particles depends on the algorithm that computes its points.

Forming Segments and Full Triangles



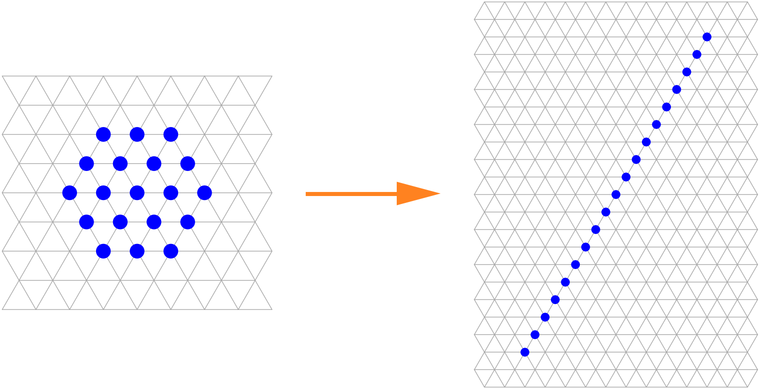
But if the shape consists only of segments and full triangles, a special protocol allows to form it in $O(n^2)$ total moves.

Forming Segments and Full Triangles



But if the shape consists only of segments and full triangles, a special protocol allows to form it in $O(n^2)$ total moves.

Matching Lower Bound



This example shows that $O(n^2)$ total moves are optimal.

Conclusion

Theorem

There is a universal shape-formation algorithm that allows a system of at least 4 particles, initially in a simply connected configuration (possibly with an unbreakable symmetry), to form any Turing-computable shape (with the same symmetry) such that, at every scale, each symmetric component has at least 4 points lying on a segment of constant length.

Theorem

If the shape to be formed consists only of segments and full triangles, the system can form it in $O(n^2)$ moves (optimally) and $O(n^2)$ rounds.

Open problem: are $O(n^2)$ rounds optimal?