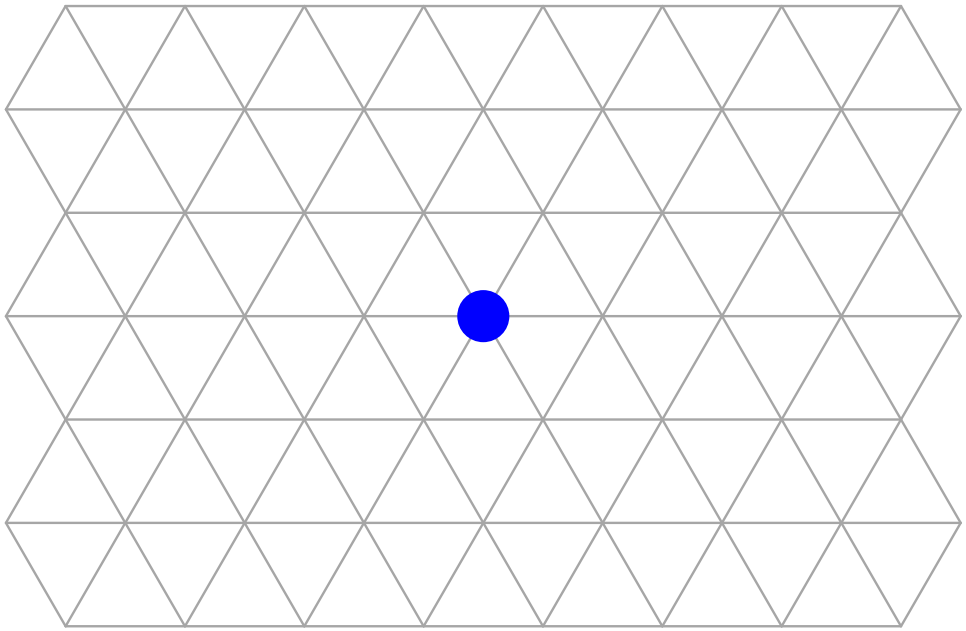
Mobile RAM and Shape Formation by Programmable Particles

(Euro-Par 2020)

Giuseppe A. Di Luna, Paola Flocchini, Nicola Santoro, Giovanni Viglietta, and Yukiko Yamauchi

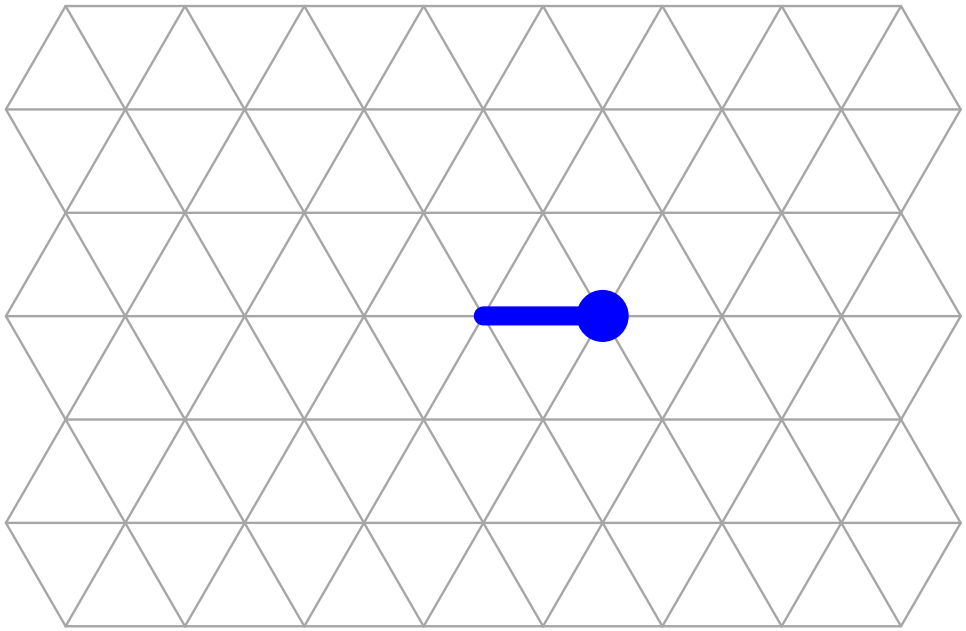
August 28, 2020

Amoebots



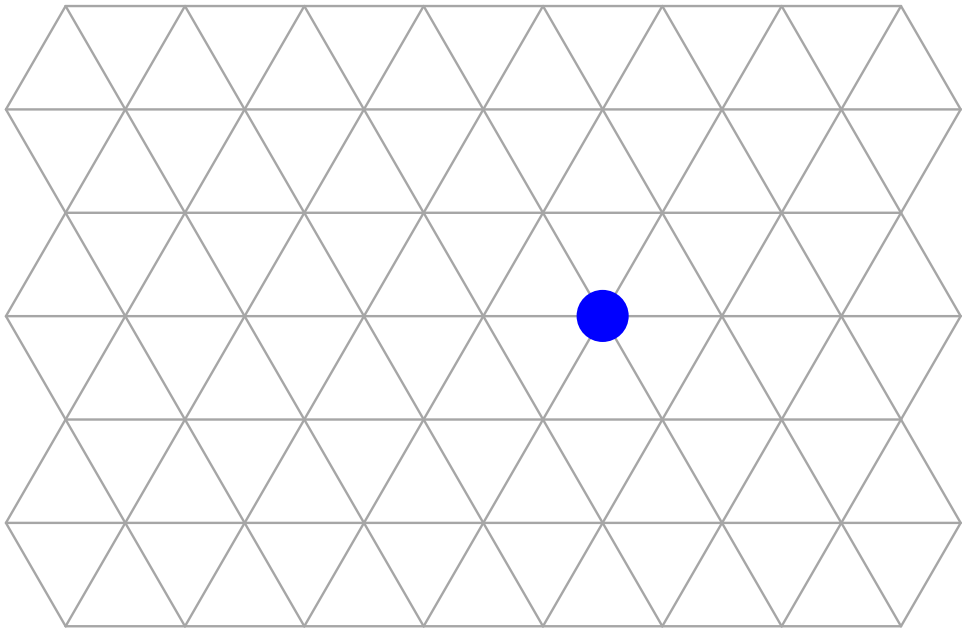
In this model, particles occupy nodes of a triangular grid.

Amoebots



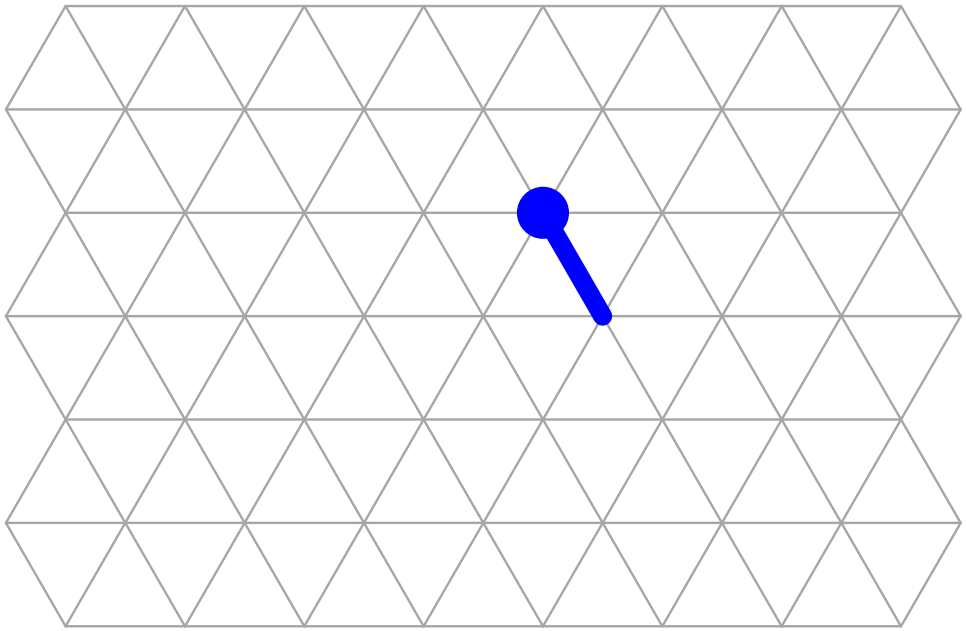
A particle can move by *expanding* and *contracting*.

Amoebots



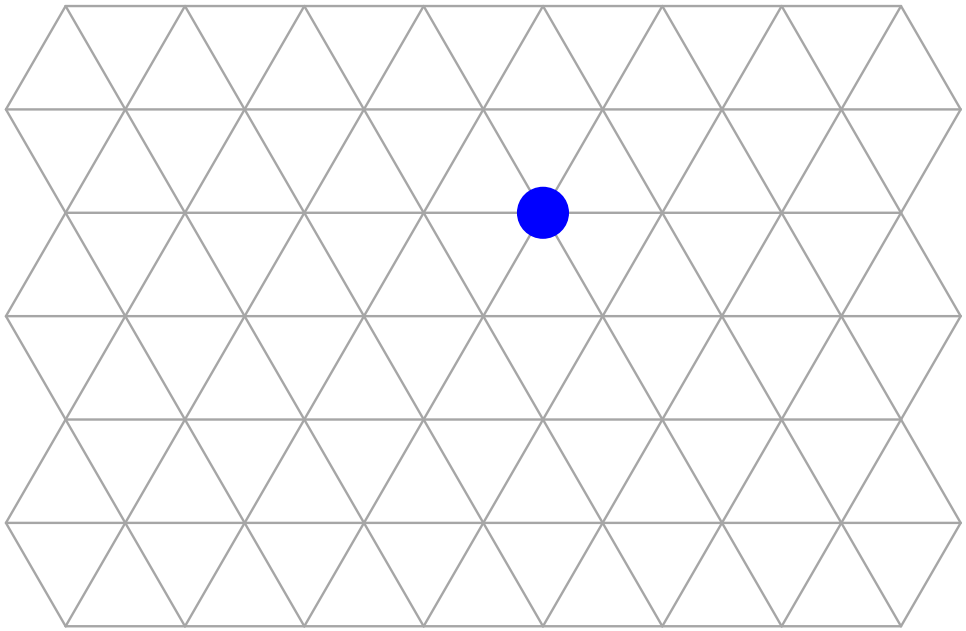
A particle can move by *expanding* and *contracting*.

Amoebots



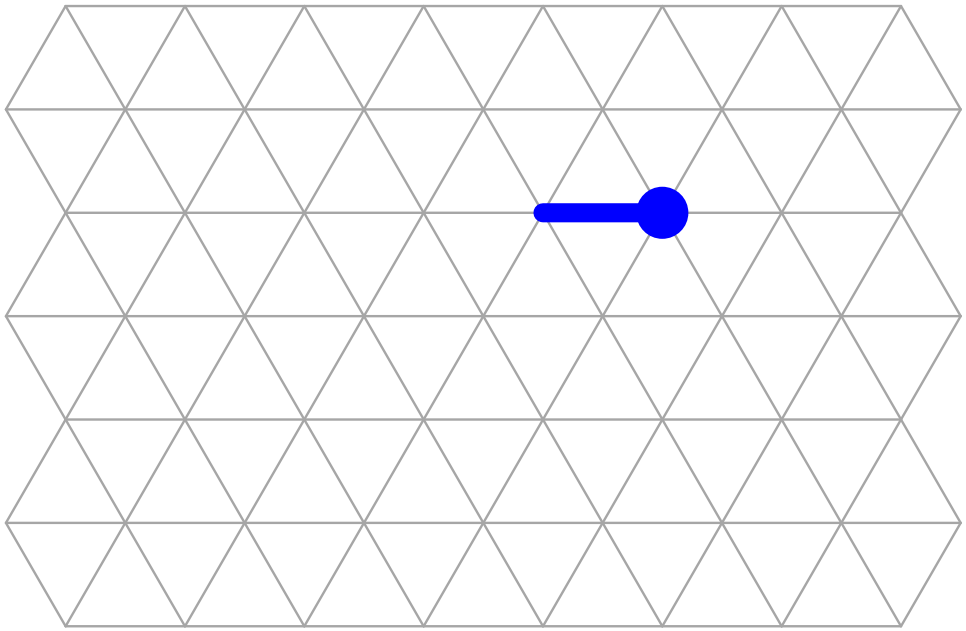
A particle can move by *expanding* and *contracting*.

Amoebots



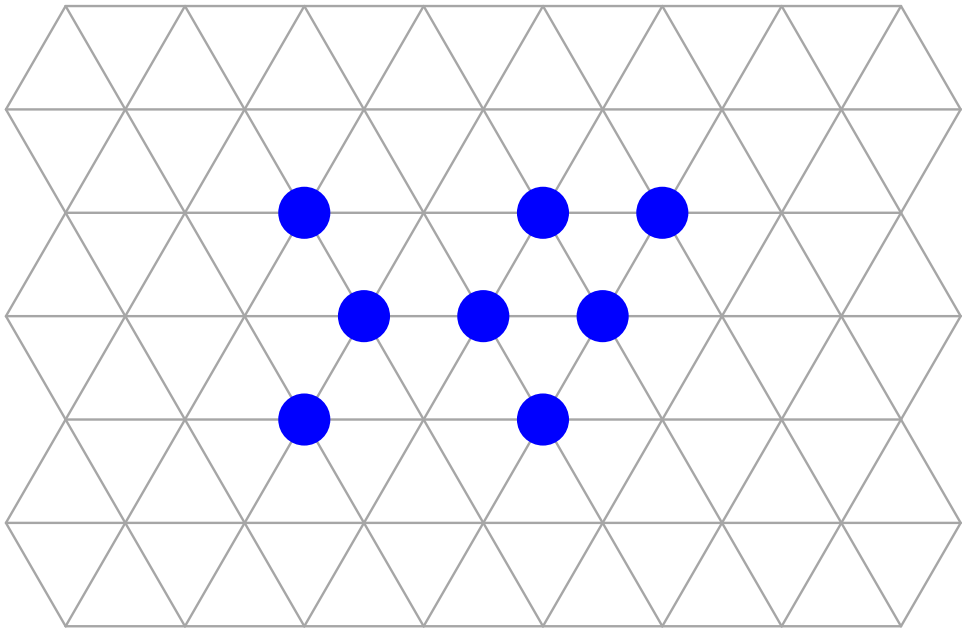
A particle can move by *expanding* and *contracting*.

Amoebots



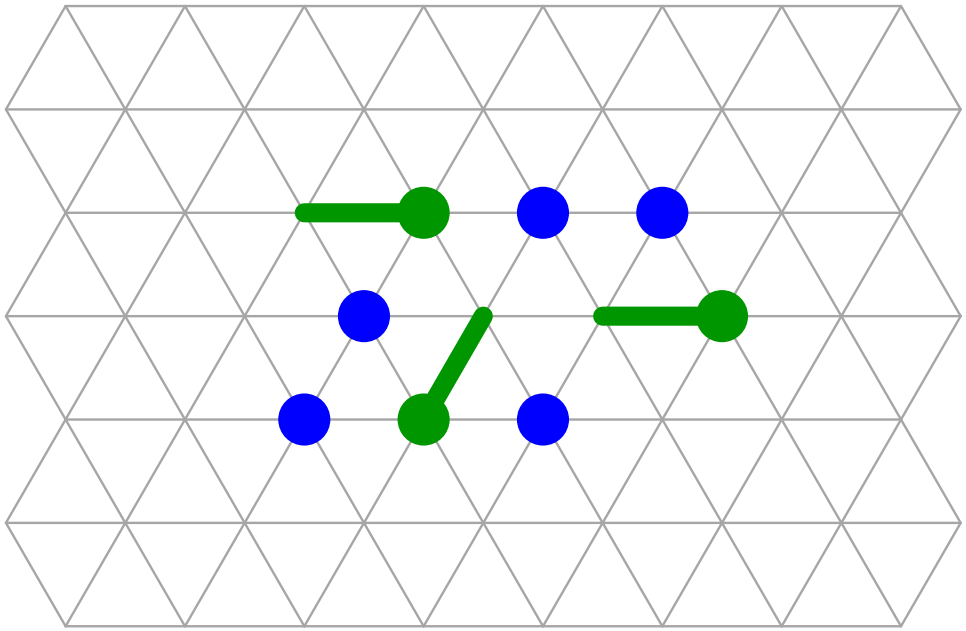
A particle can move by *expanding* and *contracting*.

Amoebots



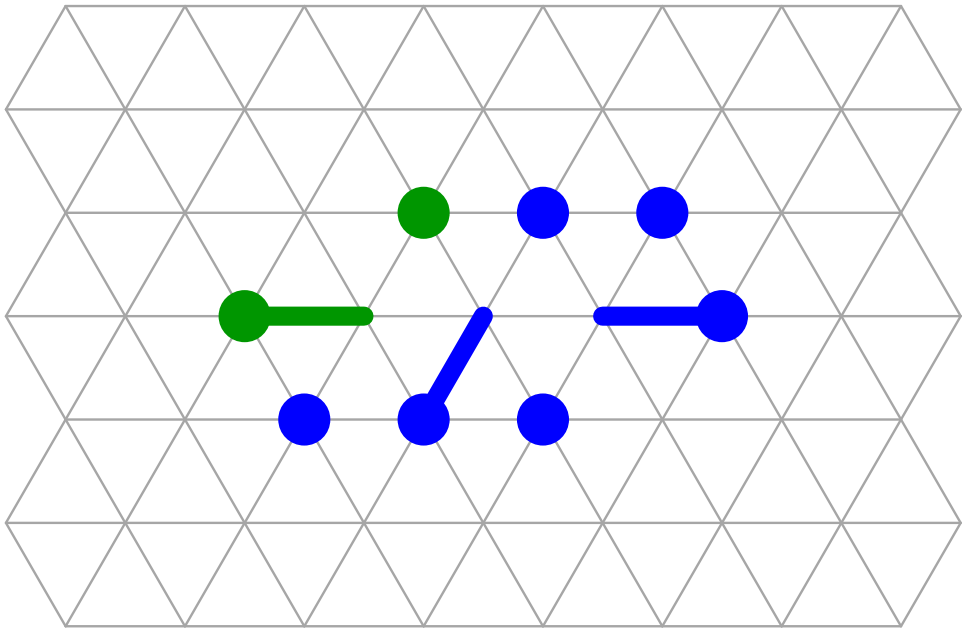
A *system* of particles is given.

Amoebots



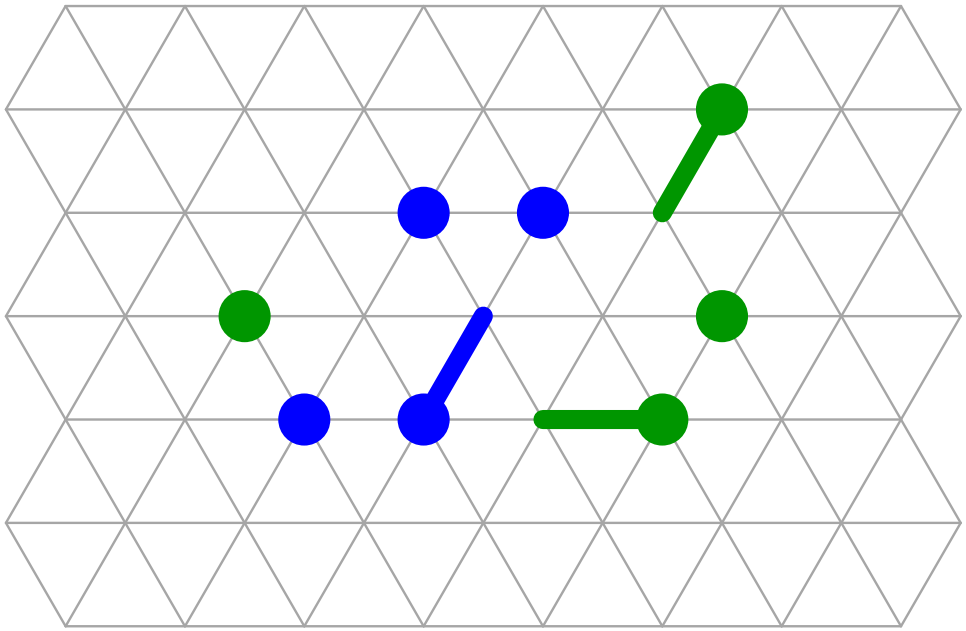
Particles move *asynchronously* following an algorithm.

Amoebots



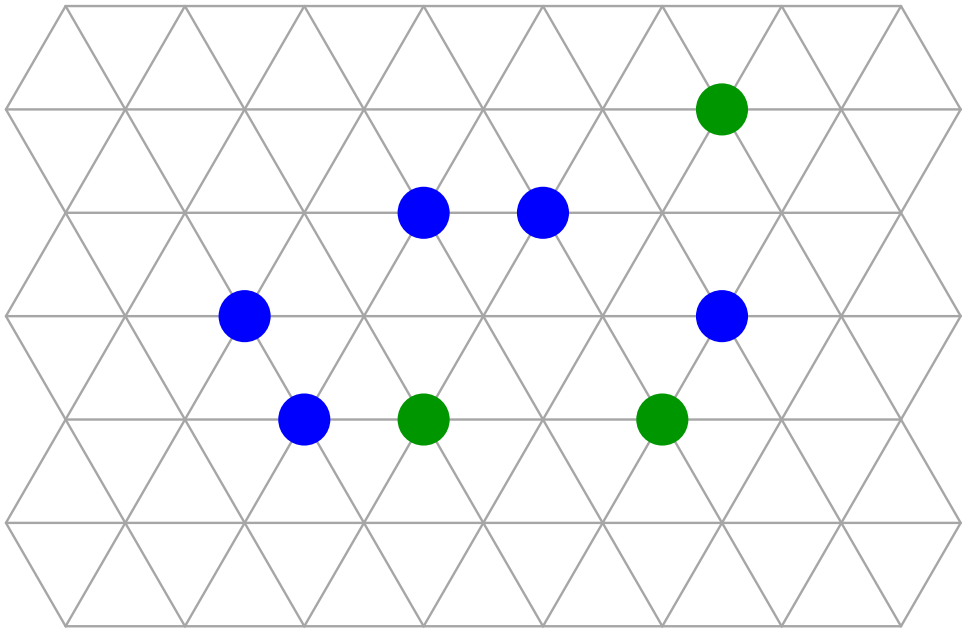
Particles move *asynchronously* following an algorithm.

Amoebots



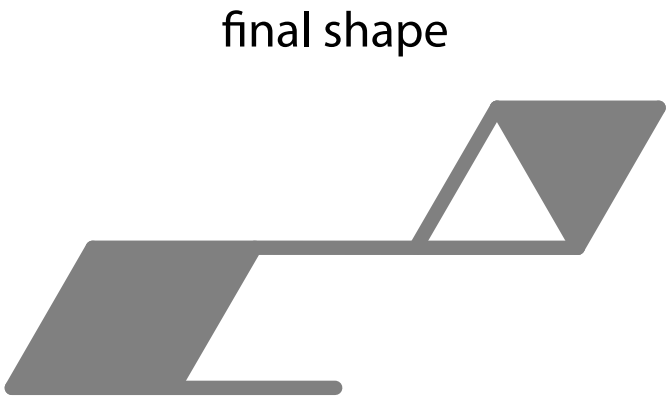
At each step, any set of particles is activated by an *adversary*.

Amoebots



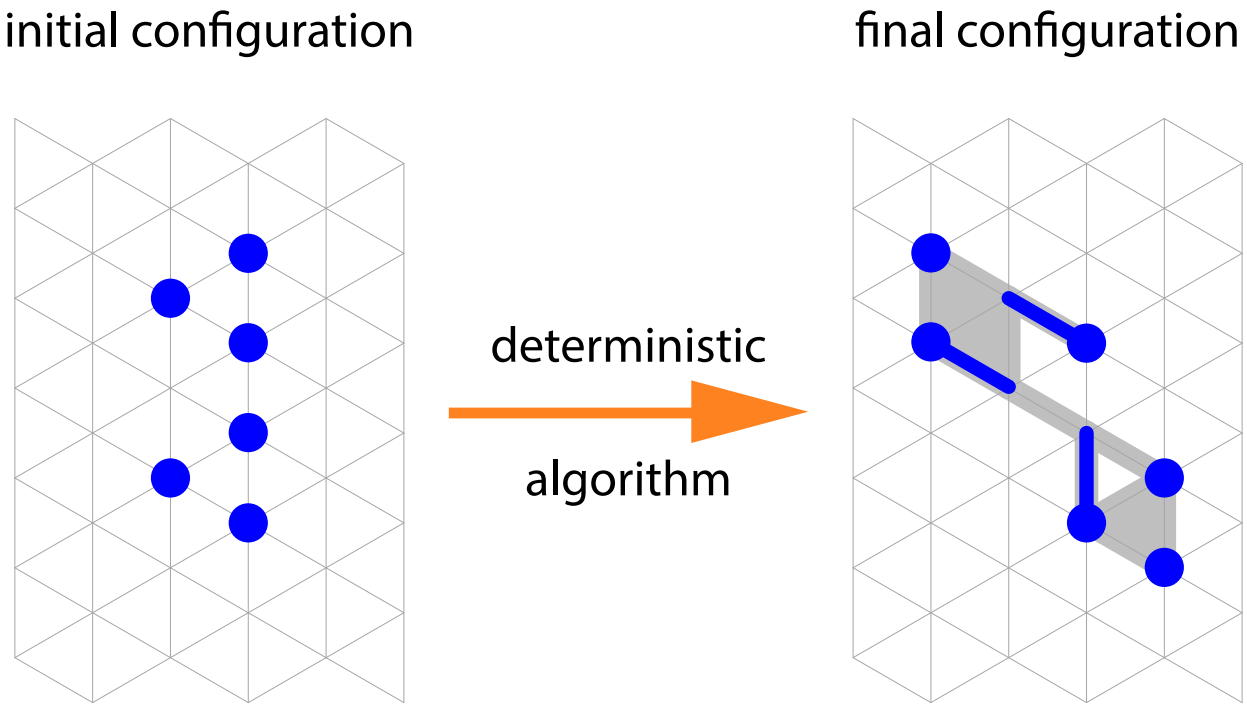
At each step, any set of particles is activated by an *adversary*.

Shape Formation



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

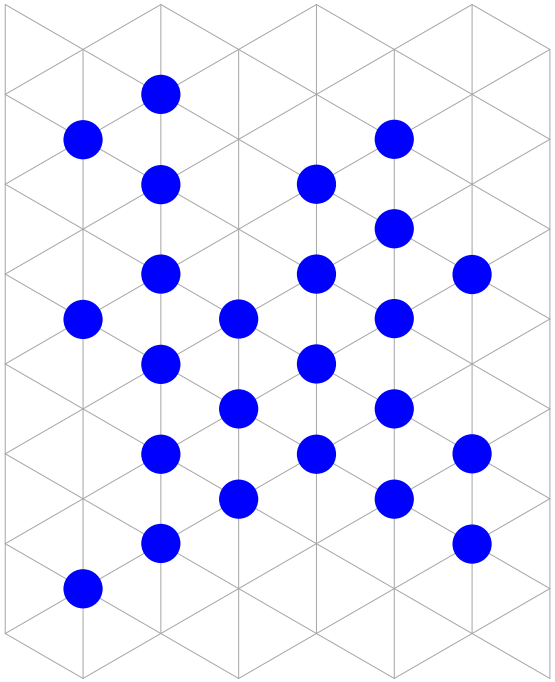
Shape Formation



The shape-formation algorithm should be *deterministic*.

Shape Formation

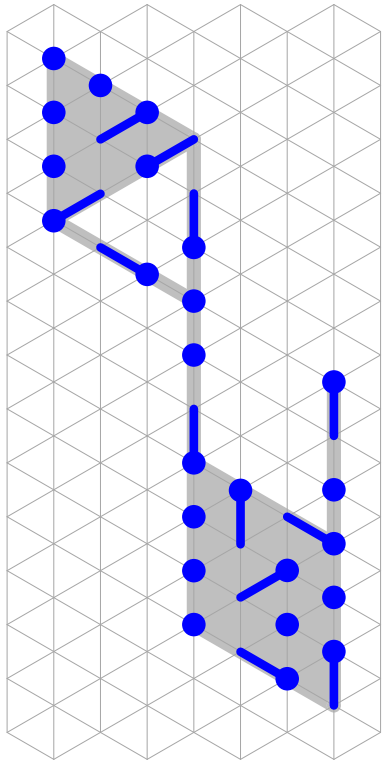
initial configuration



deterministic

algorithm

final configuration



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, where the starting shape is a triangle and the final shape is a collection of triangles:

Derakhshandeh, Gmyr, Richa, Scheideler, Strothmann

Universal shape formation for programmable matter

SPAA 2016

Deterministic shape-formation algorithm for asynchronous Amoebots, where the starting shape is simply connected and the final shape is a collection of triangles and segments:

Di Luna, Flocchini, Santoro, Viglietta, Yamauchi

Shape formation by programmable particles

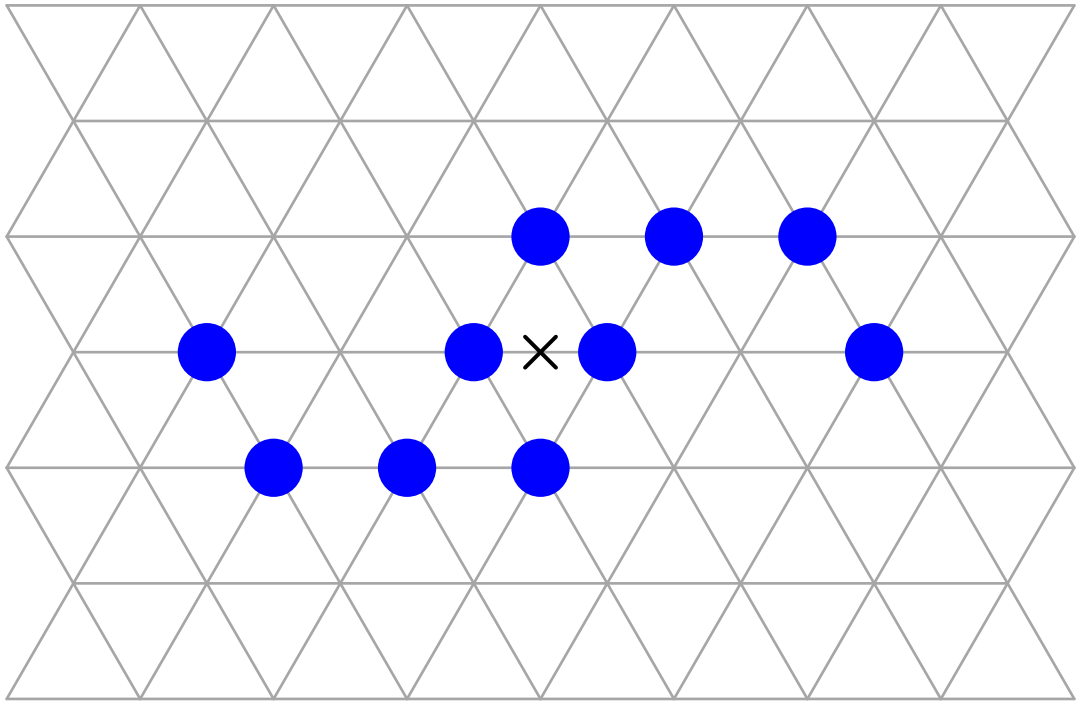
DISC 2017 (BA), OPODIS 2017

Our Particle Model

The n particles in the system:

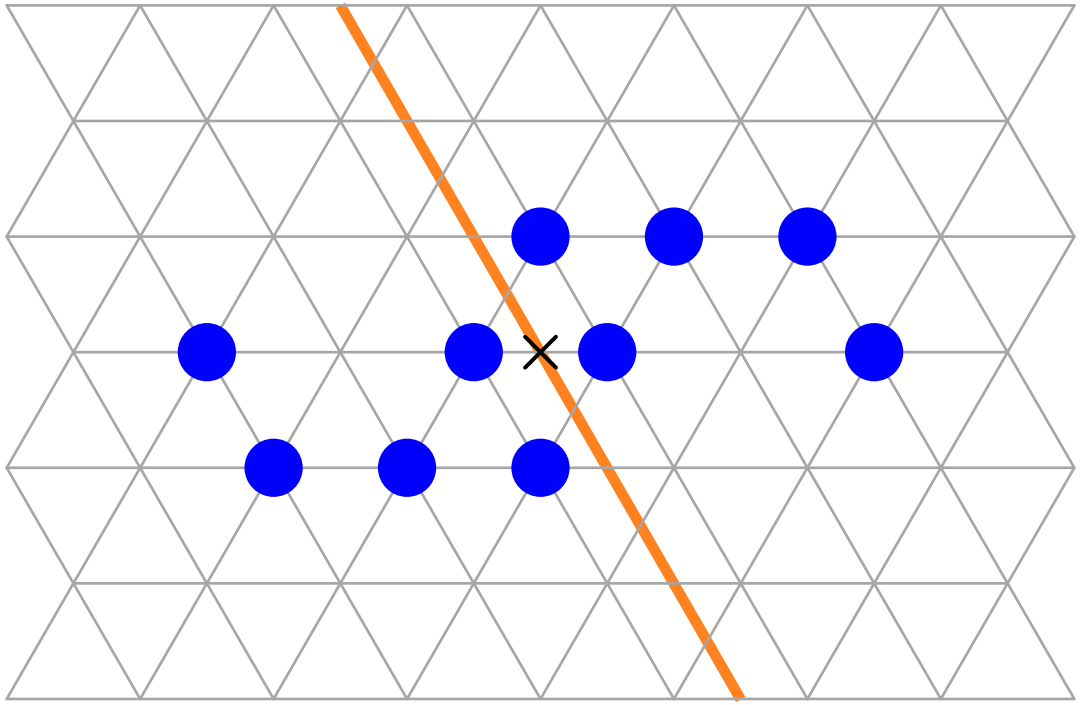
- initially form any simply connected shape
- know the final shape but do *not* know n
- have a constant amount of internal memory
- are anonymous and start in the same state
- can only see and communicate with adjacent particles
- do not have a *compass*
- may not agree on a *clockwise direction*
-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

Unbreakable Symmetry



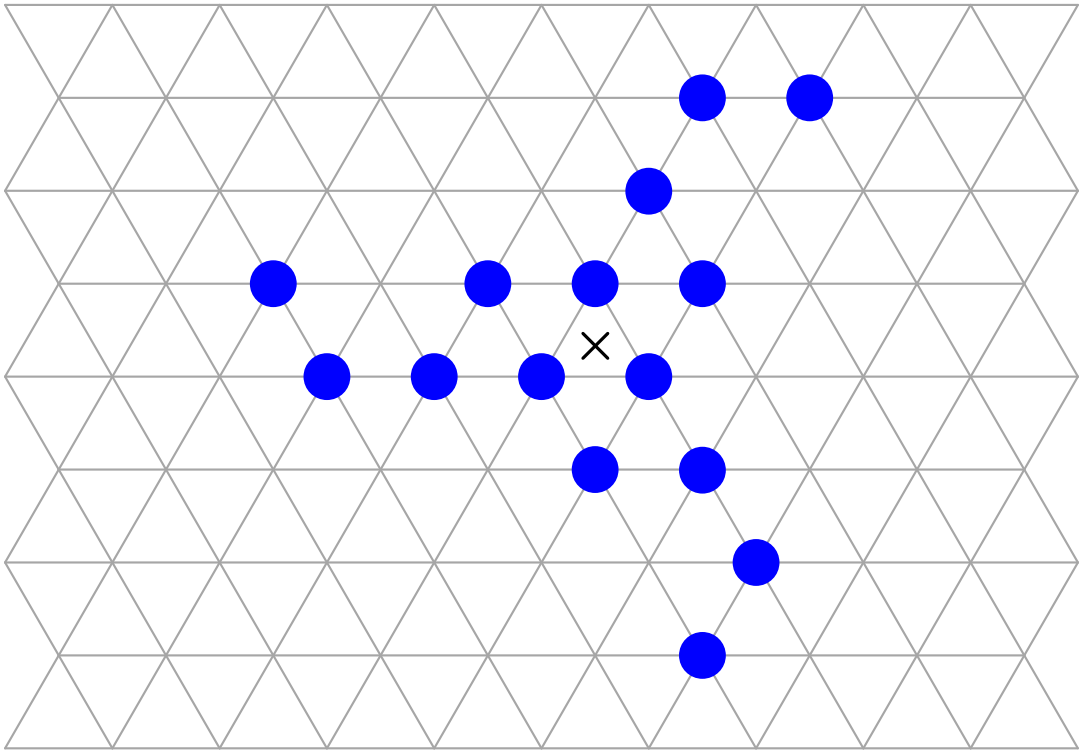
If the system has a center of symmetry not on a grid node...

Unbreakable Symmetry



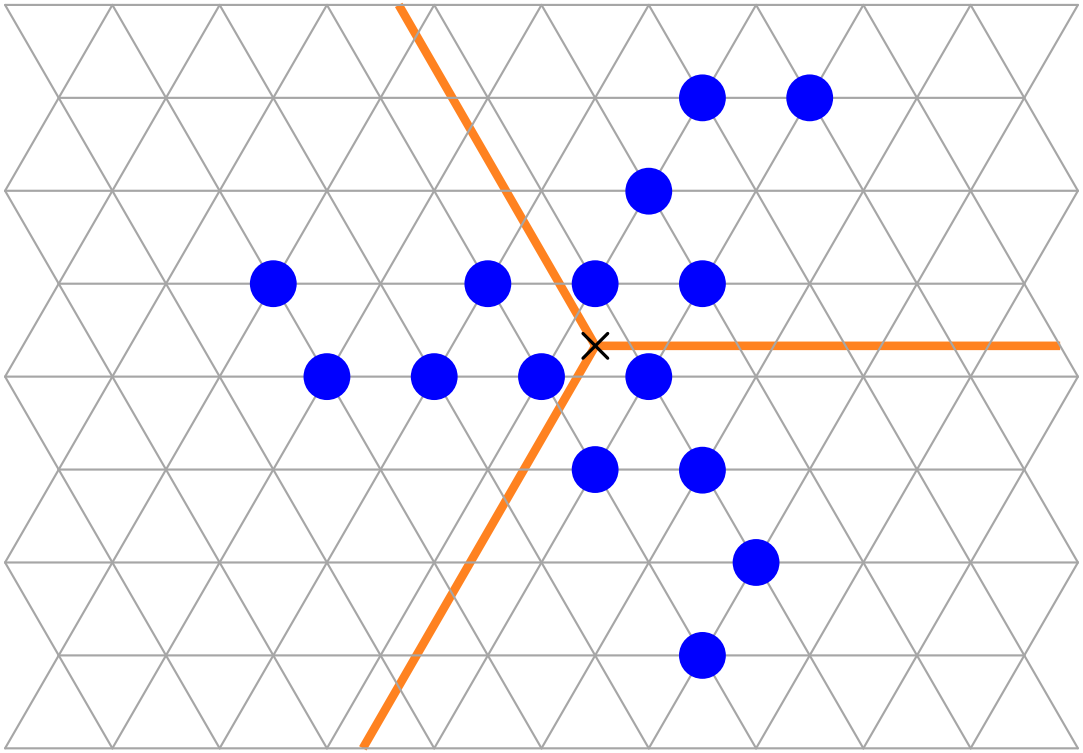
Then this symmetry is impossible to break.

Unbreakable Symmetry



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

Unbreakable Symmetry



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

Statement of Results

Observation

If the system initially has an unbreakable 2- or 3-symmetry, it cannot form shapes that do not have the same type of symmetry.

Theorem

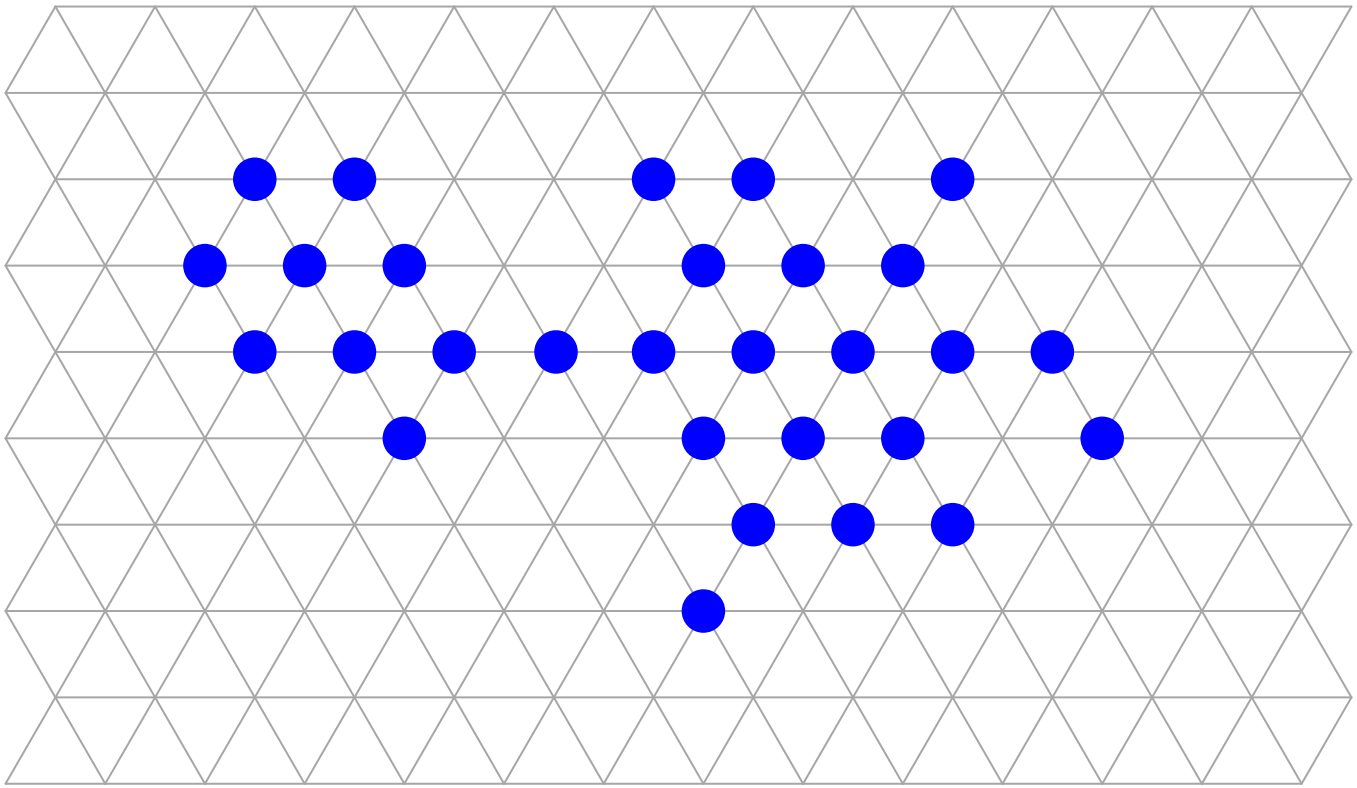
For all other cases, there is a universal shape-formation algorithm, provided that the system initially forms a simply connected shape, and the final shape and its scaled-up copies are Turing-computable (with some bland extra assumptions).

The extra assumptions are satisfied by connected shapes, so:

Corollary

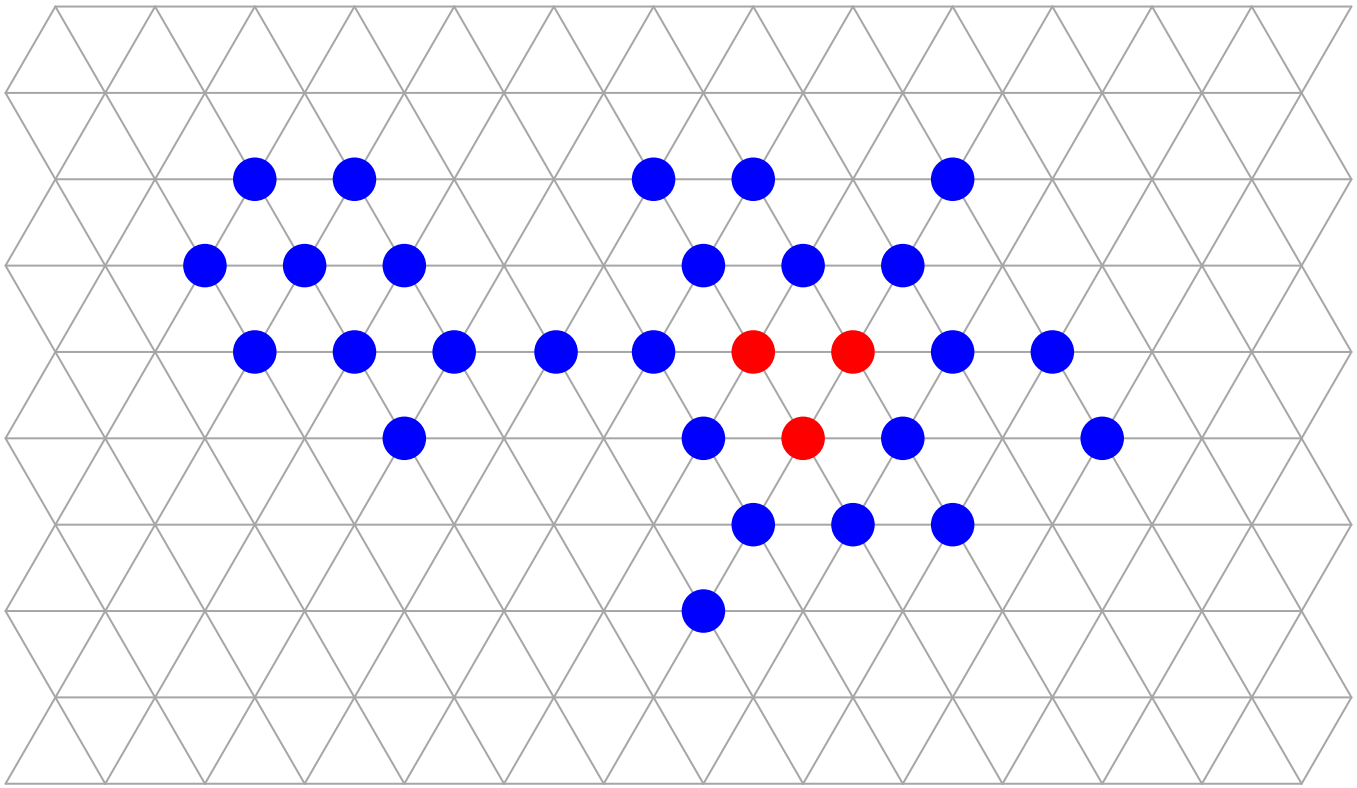
A system that initially forms a simply connected shape can form a final shape whose scaled-up copies are Turing-computable and connected if and only if this does not contradict the Observation.

Universal Shape-Formation Algorithm



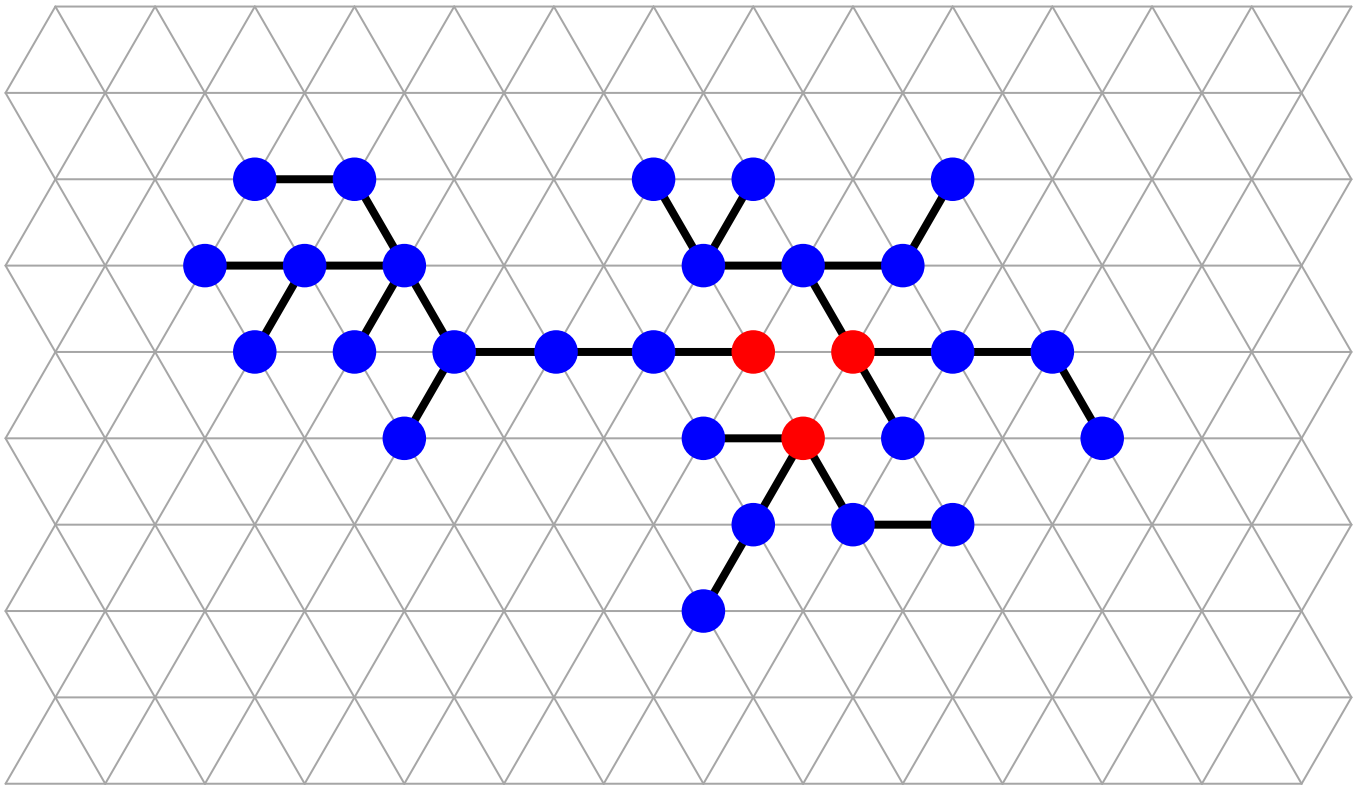
Start with a simply connected system (i.e., with no "holes").

Universal Shape-Formation Algorithm



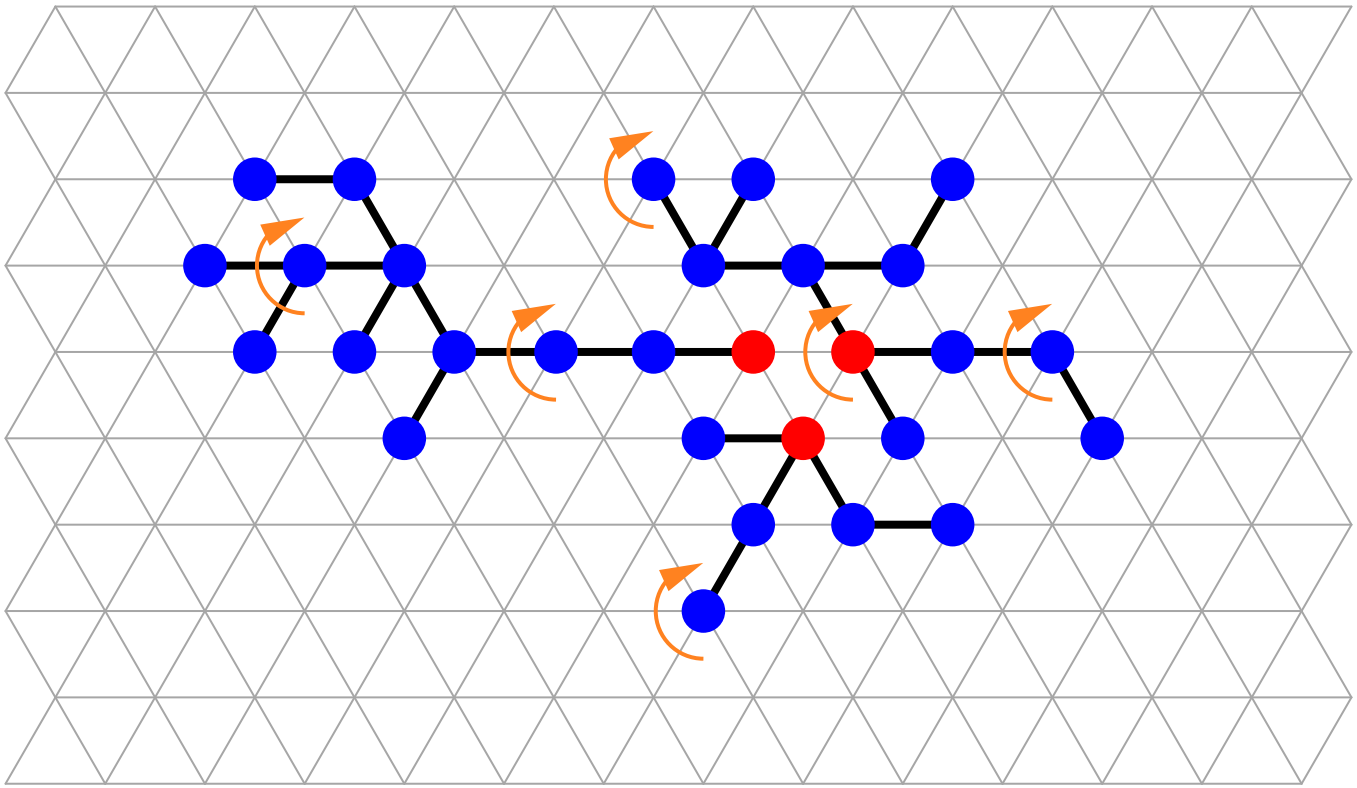
Phase 1 (old): attempt to elect a leader.

Universal Shape-Formation Algorithm



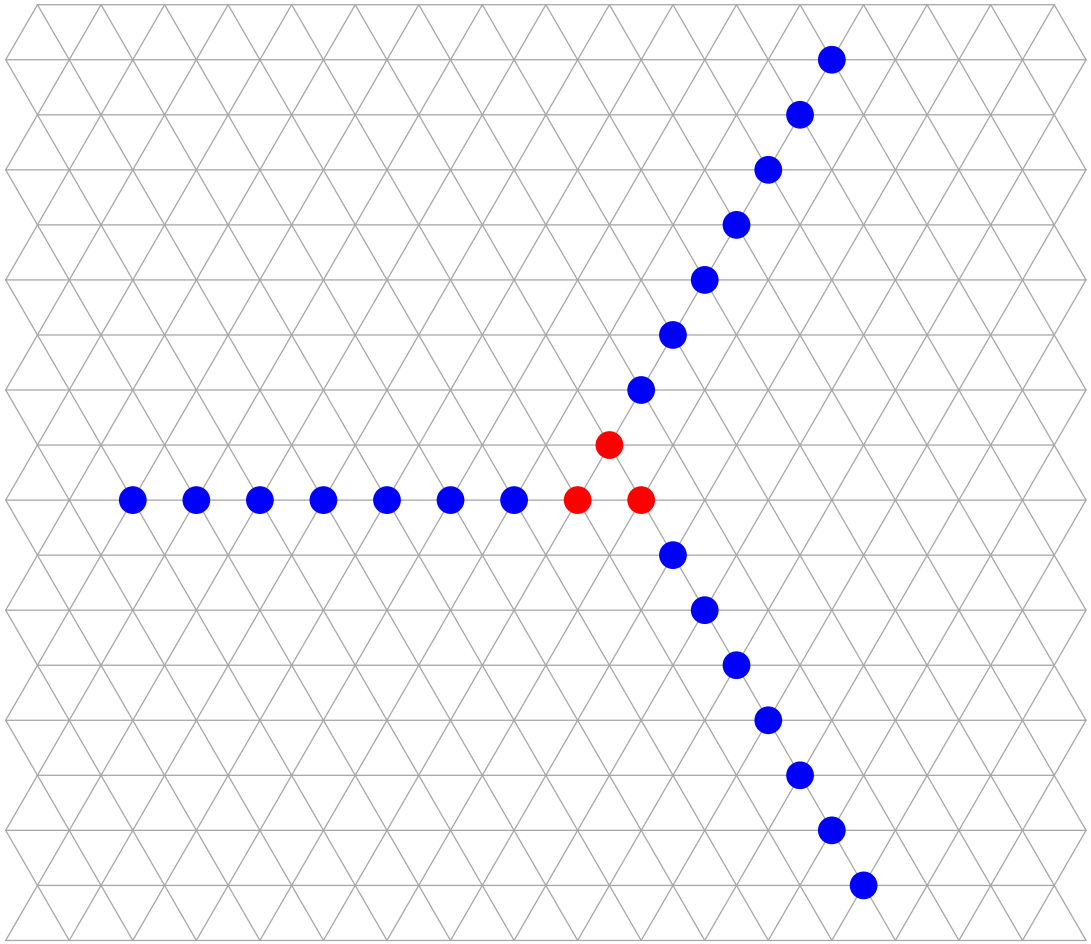
Phase 2 (old): construct a spanning forest.

Universal Shape-Formation Algorithm



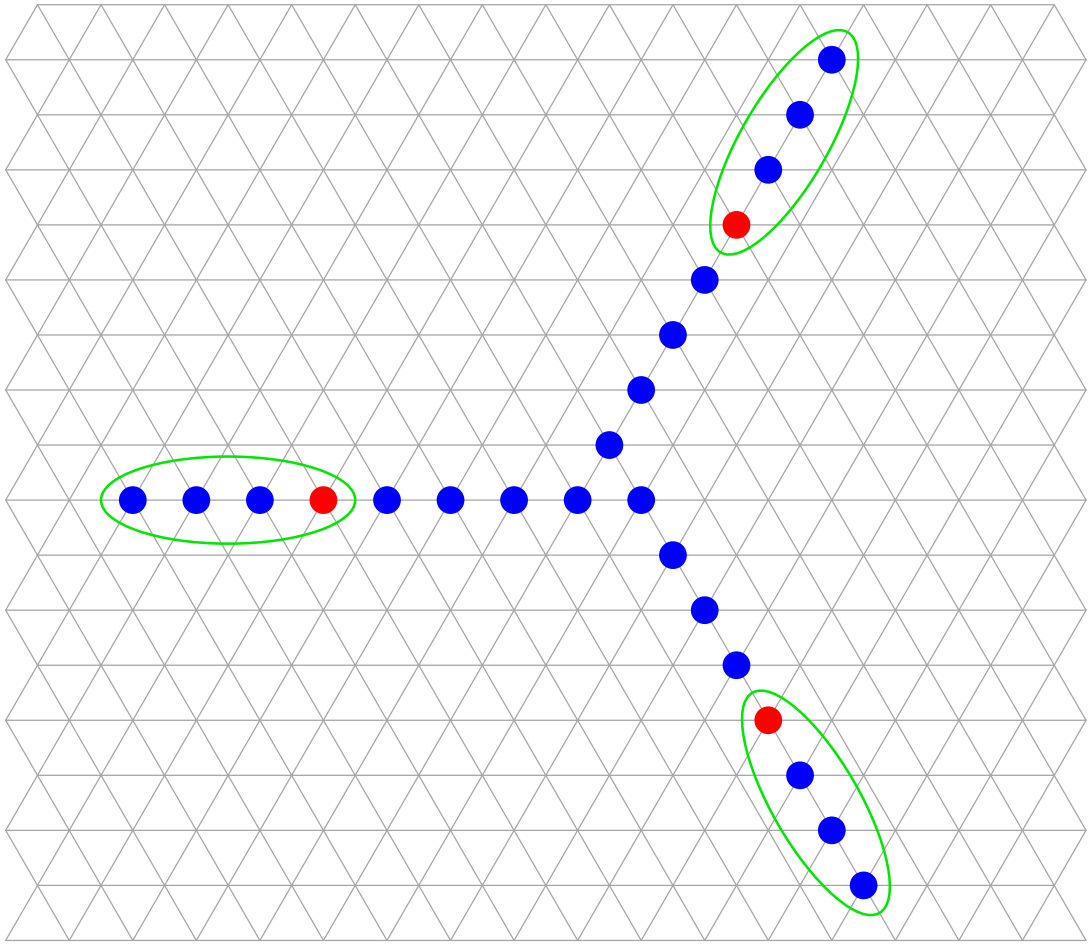
Phase 3 (old): agree on a clockwise direction.

Universal Shape-Formation Algorithm



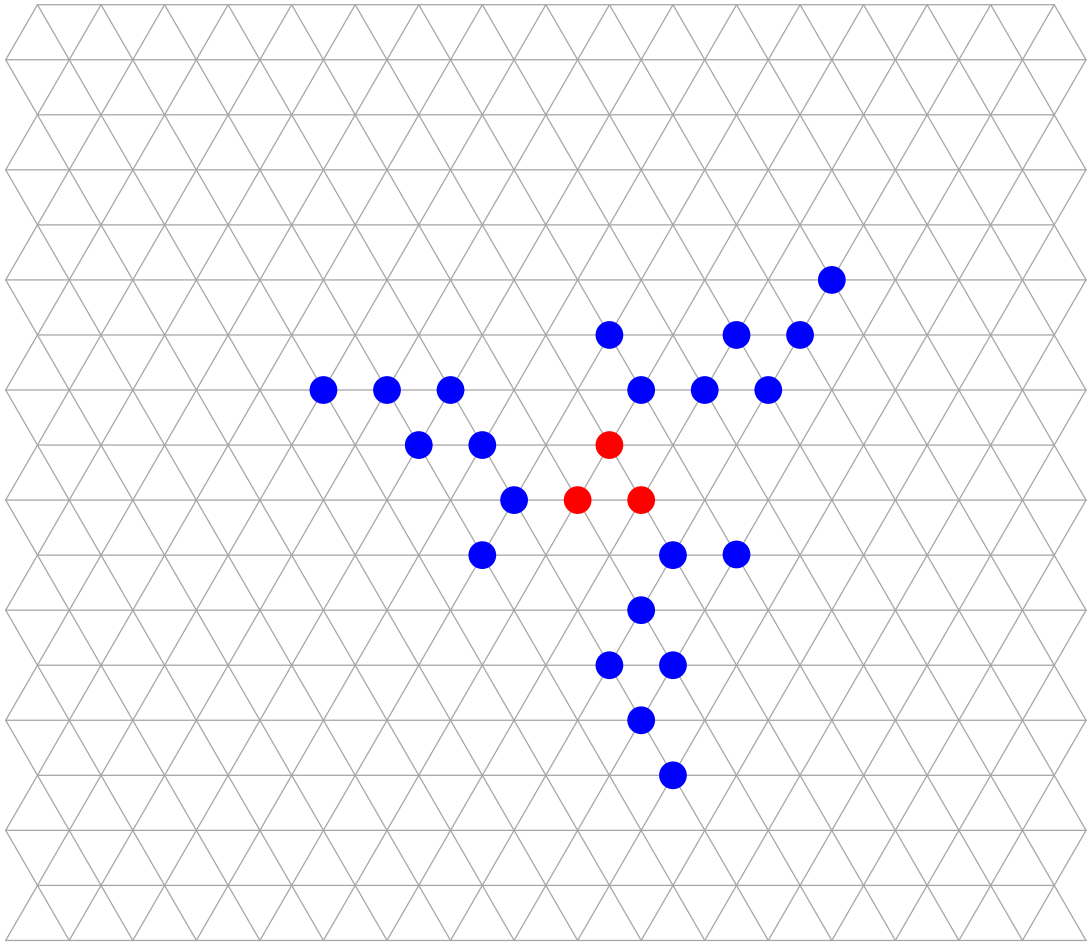
Phase 4 (old): form one line per leader.

Universal Shape-Formation Algorithm



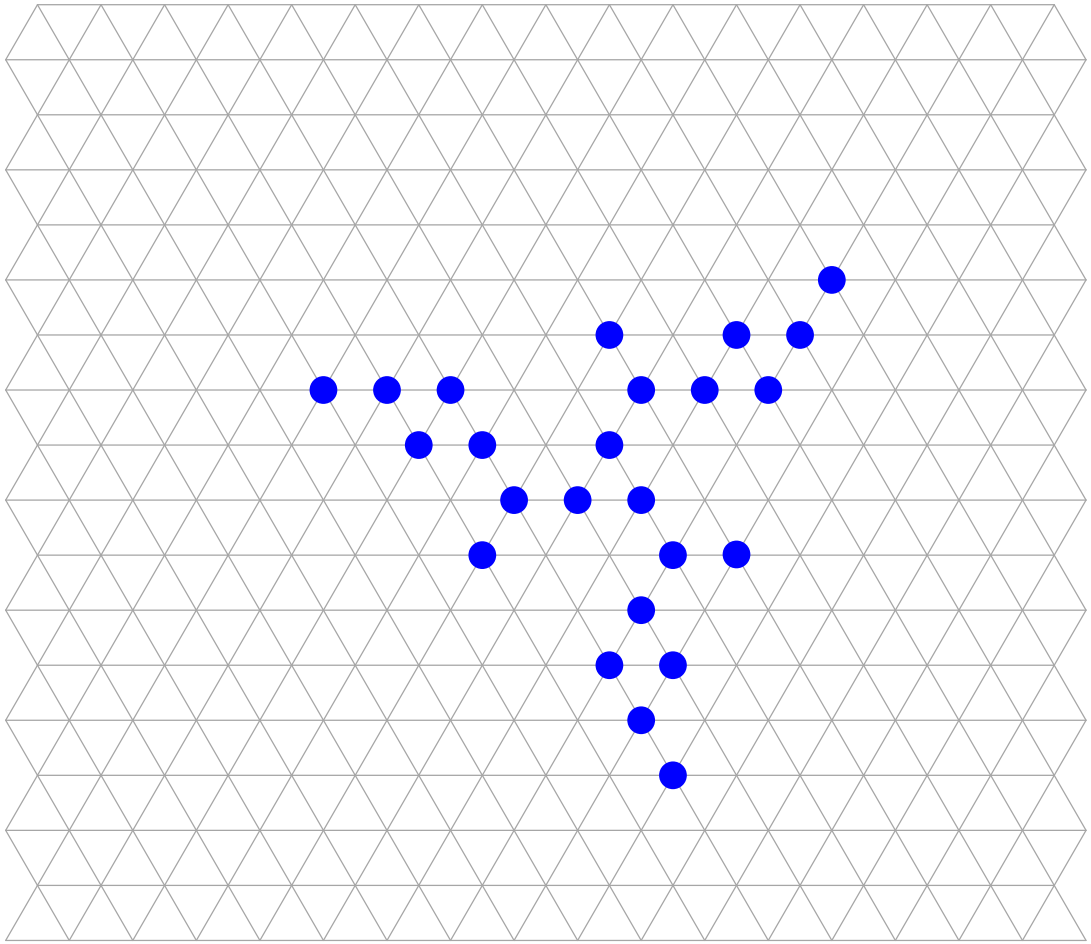
Phase 5 (new): simulate a RAM to compute the final shape.

Universal Shape-Formation Algorithm



Phase 6 (new): keep computing while forming the final shape.

Universal Shape-Formation Algorithm



Phase 6 (new): keep computing while forming the final shape.

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:
  - increment the value stored in a register by 1
  - if the value stored in a register is positive, decrement it by 1
  - test the value of a register and branch if it is 0

Random-Access Machines

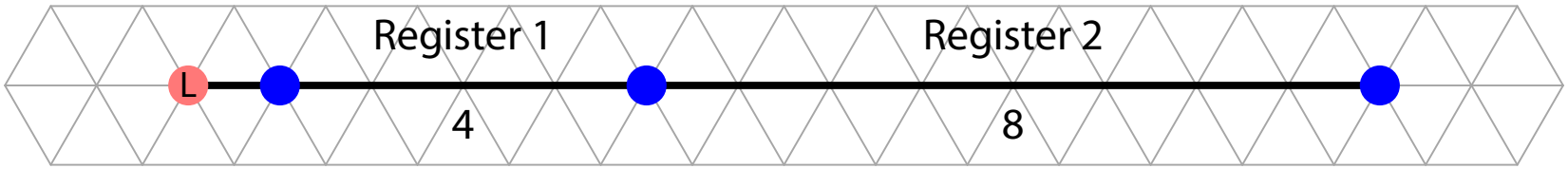
A random-access machine is a model of computation with:

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

Theorem (Minsky, 1967)

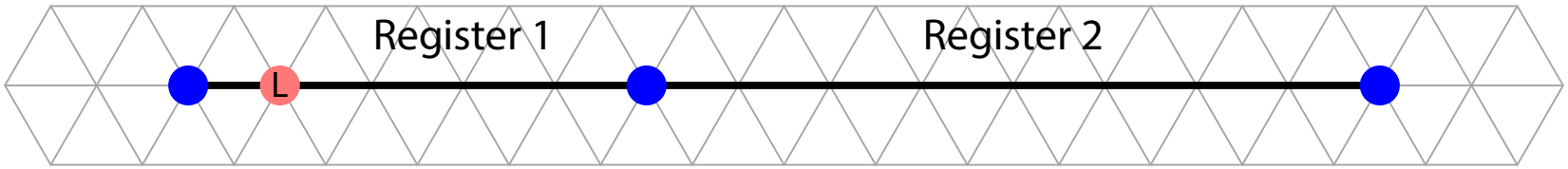
Any Turing machine can be simulated by a random-access machine with only 2 registers, the first of which initially contains the input.

Simulating a Random-Access Machine with 2 Registers



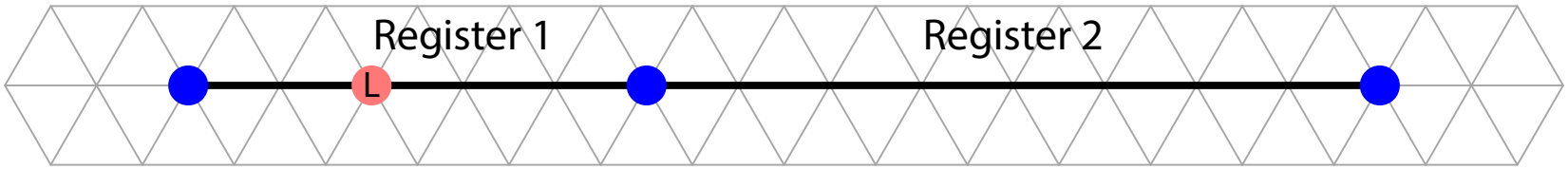
A random-access machine with 2 registers can be simulated by 4 particles: a *leader*, which executes the program, and 3 particles whose distances correspond to the values stored in the 2 registers.

Simulating a Random-Access Machine with 2 Registers



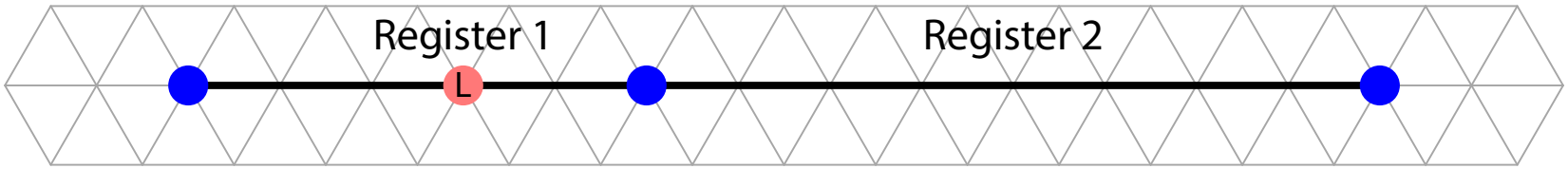
If the leader has to increment the value of the first register,
it pulls the last two particles to the right by one step,
and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers



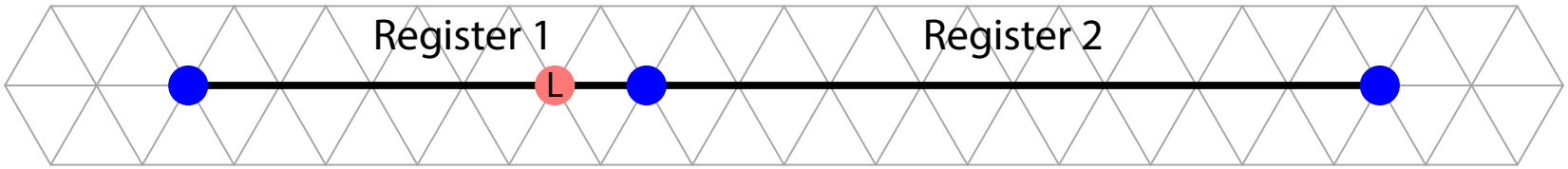
If the leader has to increment the value of the first register,
it pulls the last two particles to the right by one step,
and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers



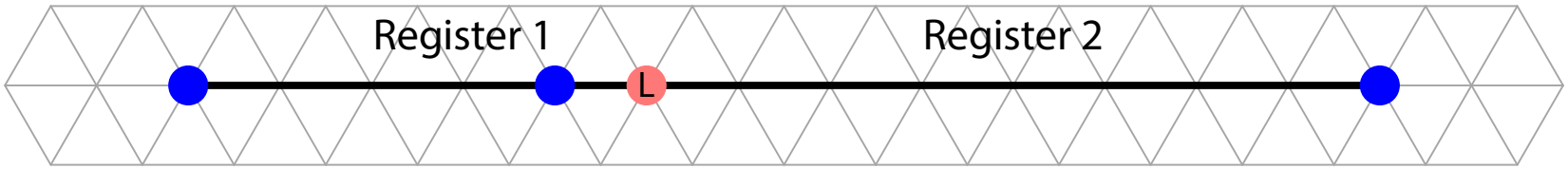
If the leader has to increment the value of the first register,
it pulls the last two particles to the right by one step,
and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers



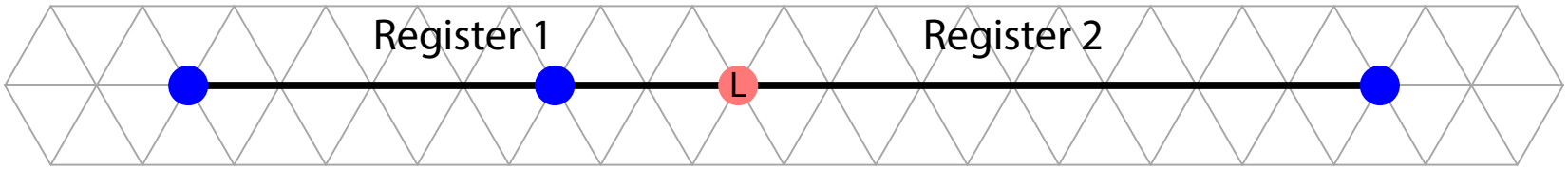
If the leader has to increment the value of the first register,
it pulls the last two particles to the right by one step,
and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers



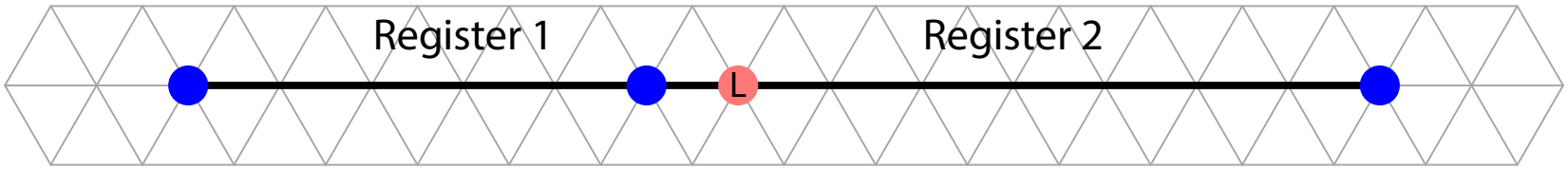
If the leader has to increment the value of the first register,
it pulls the last two particles to the right by one step,
and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers



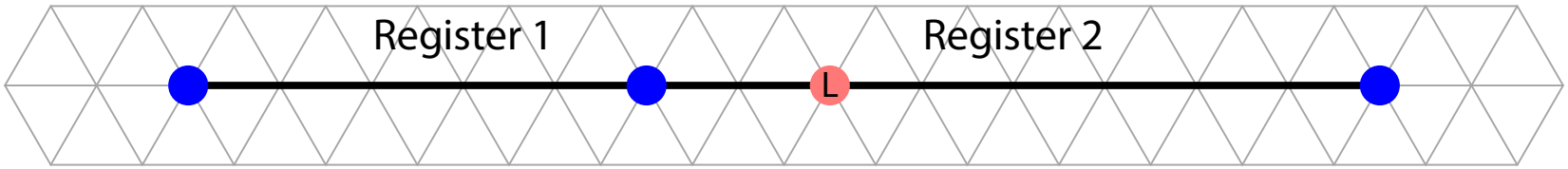
If the leader has to increment the value of the first register,
it pulls the last two particles to the right by one step,
and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers



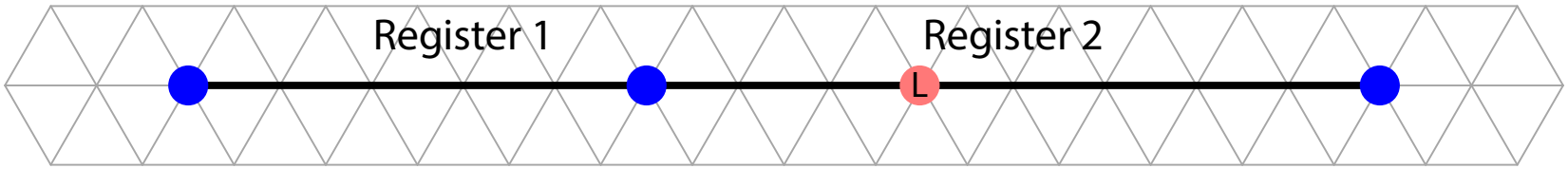
If the leader has to increment the value of the first register,
it pulls the last two particles to the right by one step,
and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers



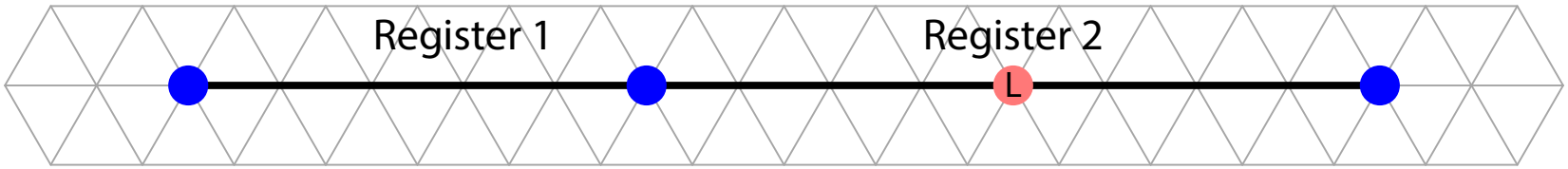
If the leader has to increment the value of the first register,
it pulls the last two particles to the right by one step,
and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers



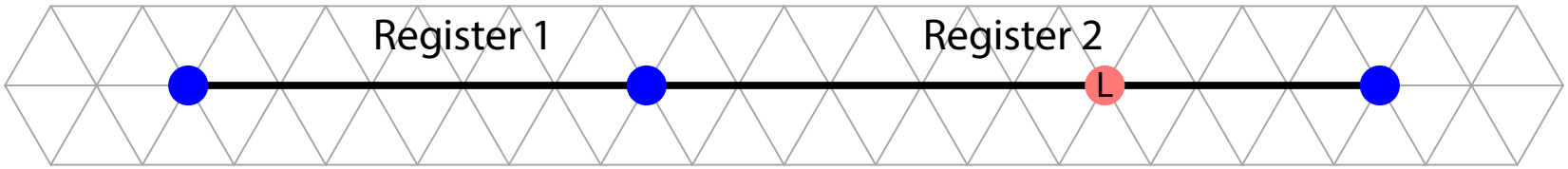
If the leader has to increment the value of the first register,
it pulls the last two particles to the right by one step,
and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers



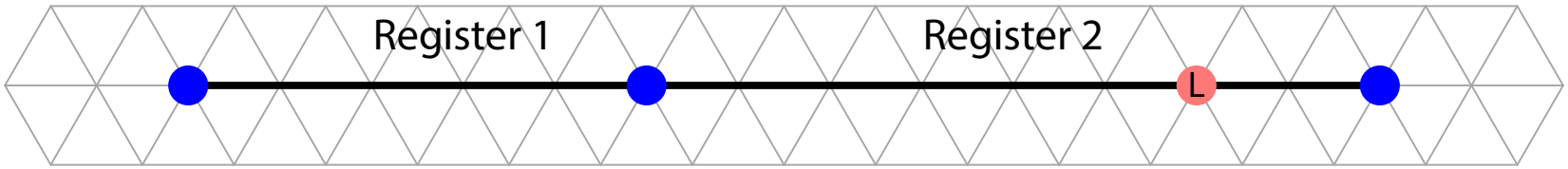
If the leader has to increment the value of the first register,
it pulls the last two particles to the right by one step,
and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers



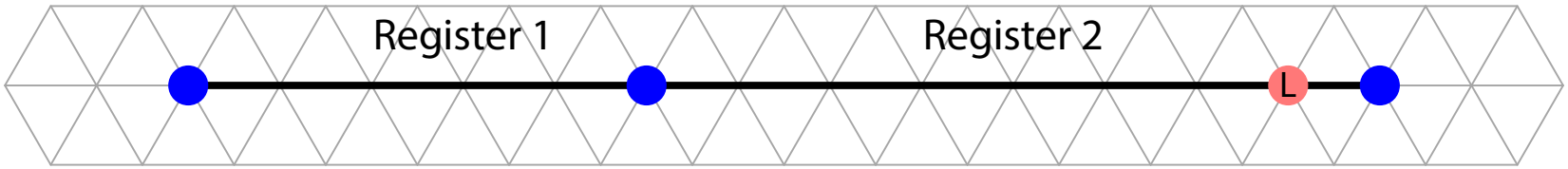
If the leader has to increment the value of the first register,
it pulls the last two particles to the right by one step,
and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers



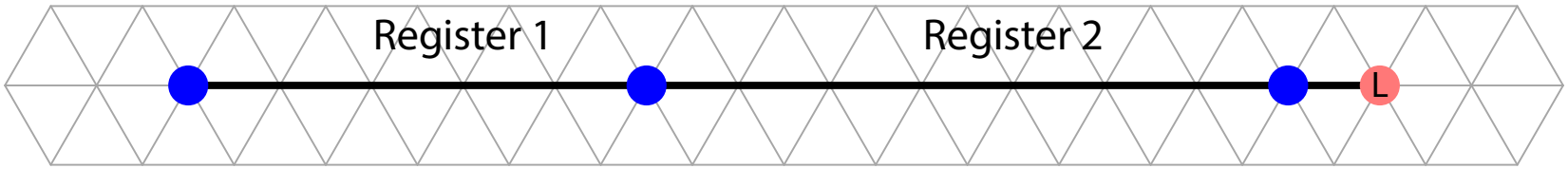
If the leader has to increment the value of the first register,
it pulls the last two particles to the right by one step,
and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers



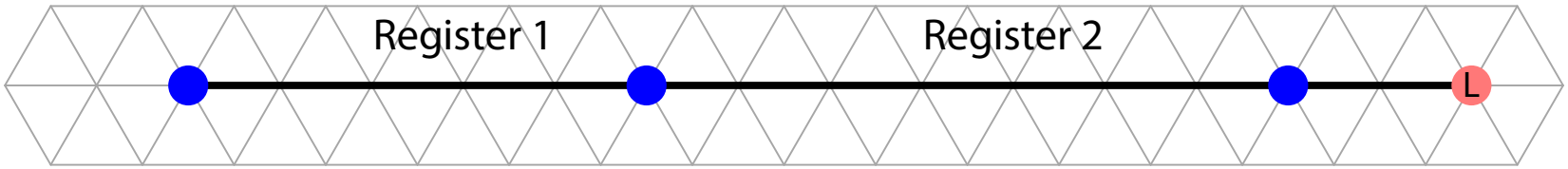
If the leader has to increment the value of the first register,
it pulls the last two particles to the right by one step,
and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers



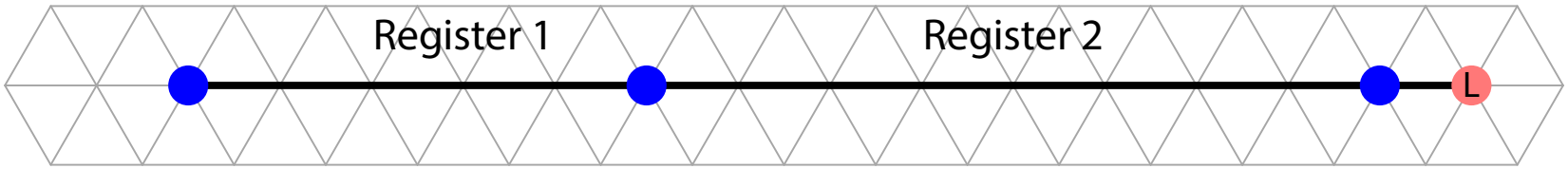
If the leader has to increment the value of the first register,
it pulls the last two particles to the right by one step,
and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers



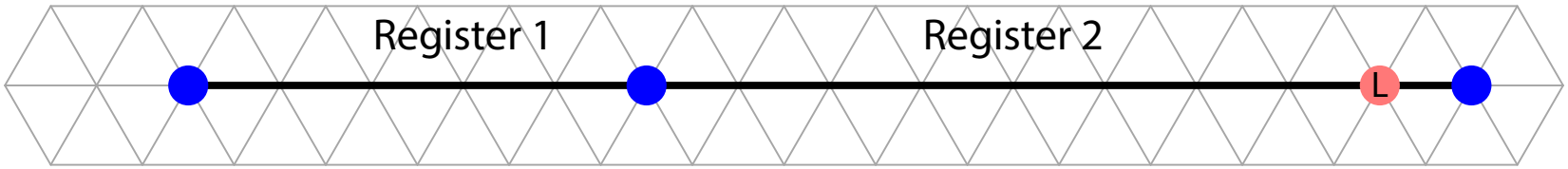
If the leader has to increment the value of the first register, it pulls the last two particles to the right by one step, and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers



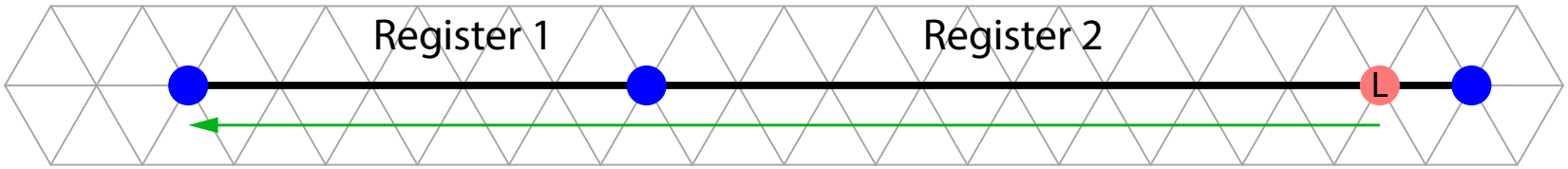
If the leader has to increment the value of the first register,
it pulls the last two particles to the right by one step,
and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers



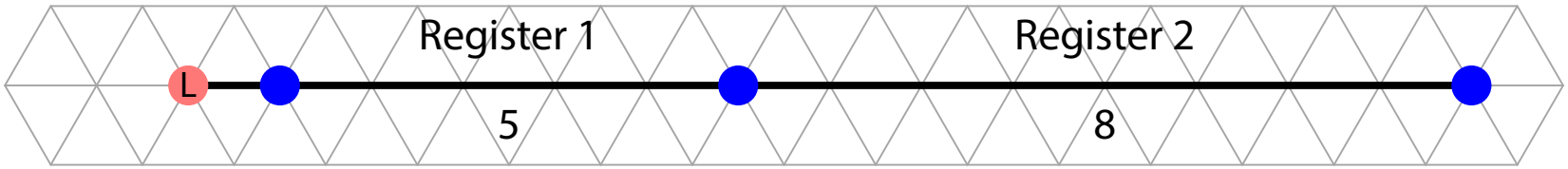
If the leader has to increment the value of the first register,
it pulls the last two particles to the right by one step,
and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers



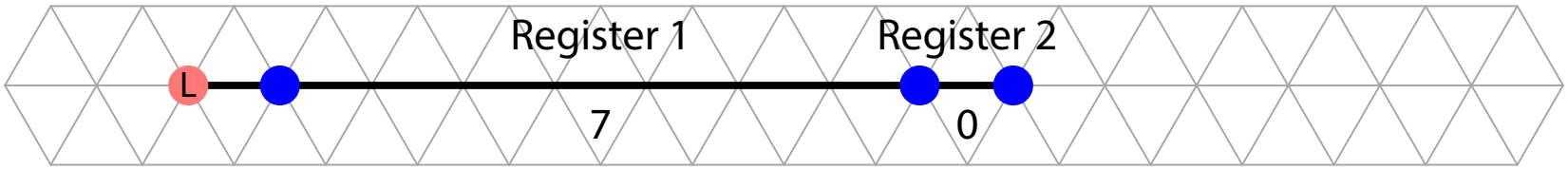
If the leader has to increment the value of the first register,
it pulls the last two particles to the right by one step,
and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers



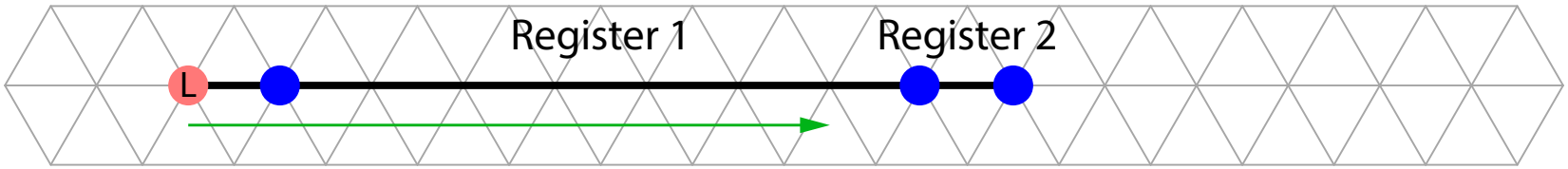
If the leader has to increment the value of the first register,
it pulls the last two particles to the right by one step,
and then goes back to its original position.

Simulating a Random-Access Machine with 2 Registers



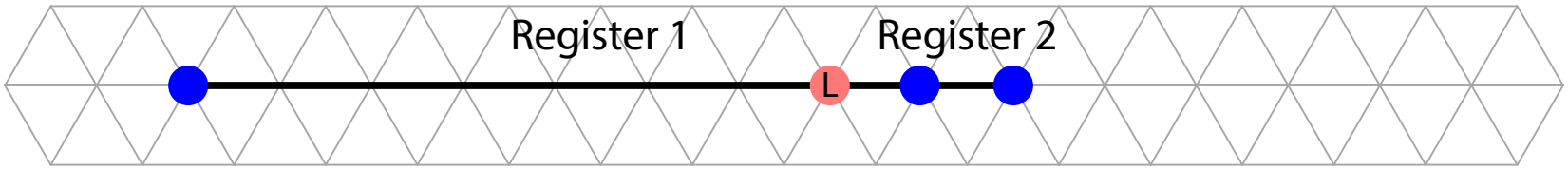
If the leader has to test if the value of the second register is 0, it reaches the second-to-last particle and exchanges messages with it, asking if the last particle is adjacent to it.

Simulating a Random-Access Machine with 2 Registers



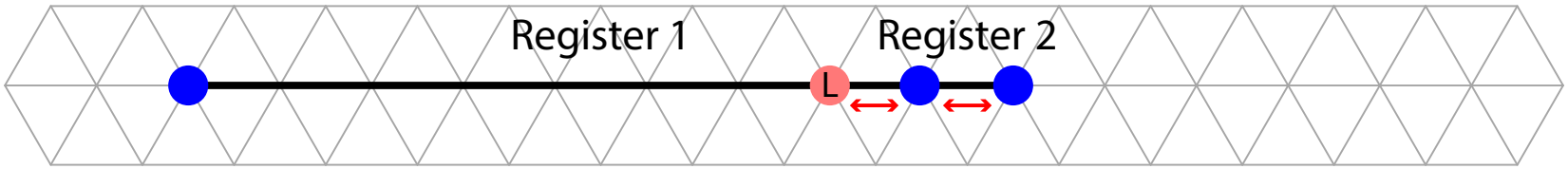
If the leader has to test if the value of the second register is 0, it reaches the second-to-last particle and exchanges messages with it, asking if the last particle is adjacent to it.

Simulating a Random-Access Machine with 2 Registers



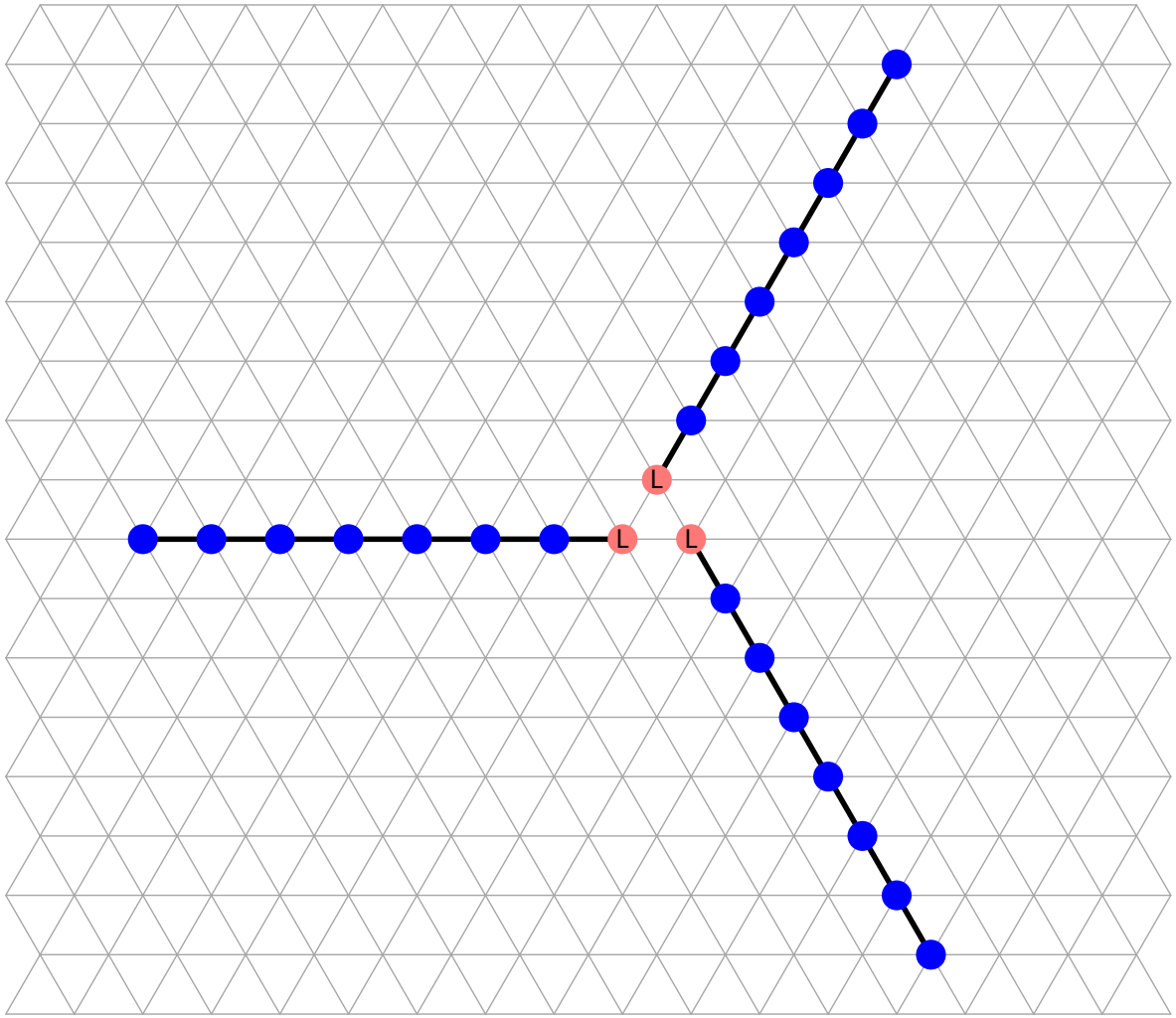
If the leader has to test if the value of the second register is 0, it reaches the second-to-last particle and exchanges messages with it, asking if the last particle is adjacent to it.

Simulating a Random-Access Machine with 2 Registers



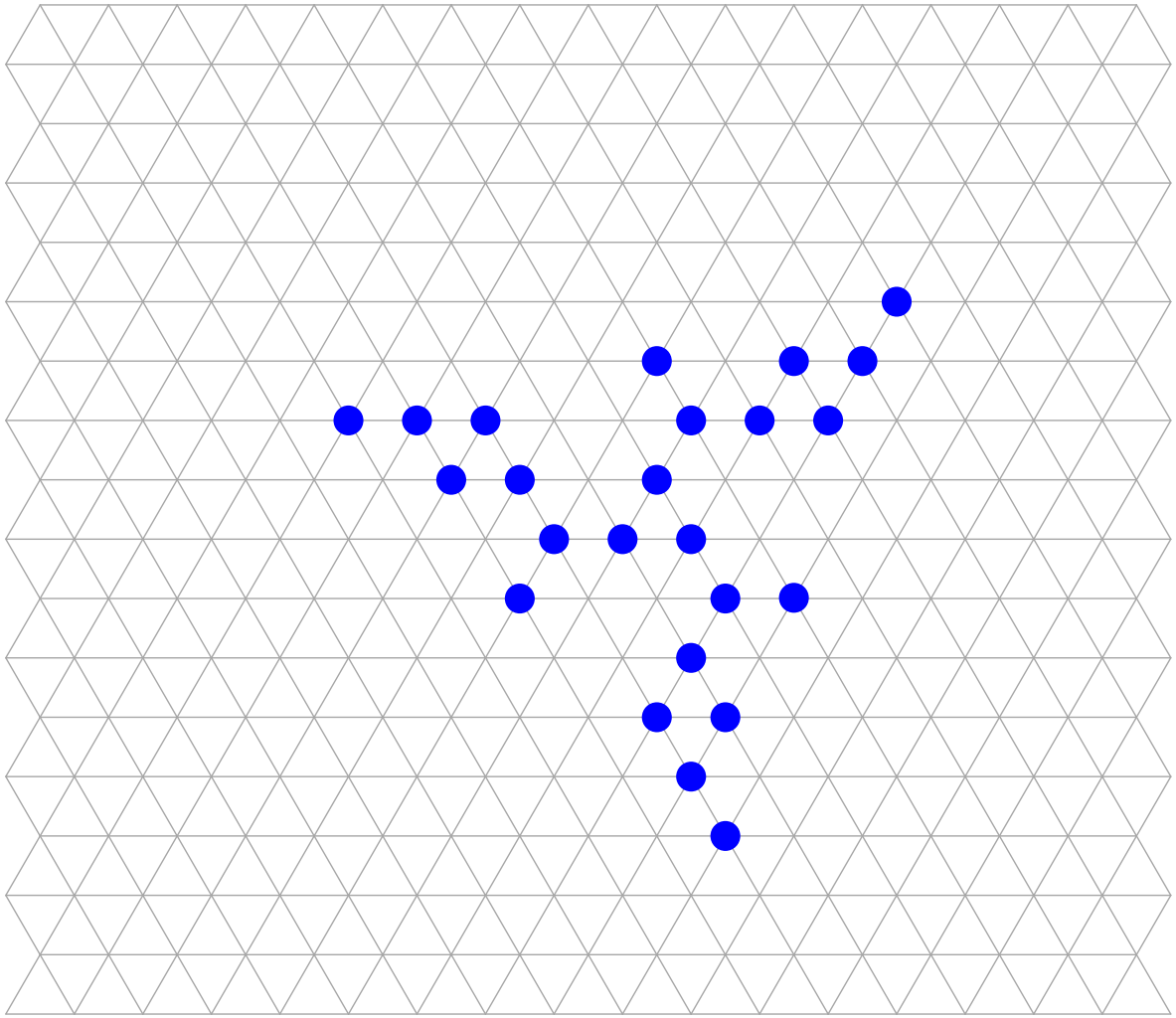
If the leader has to test if the value of the second register is 0, it reaches the second-to-last particle and exchanges messages with it, asking if the last particle is adjacent to it.

Shape-Formation Phase



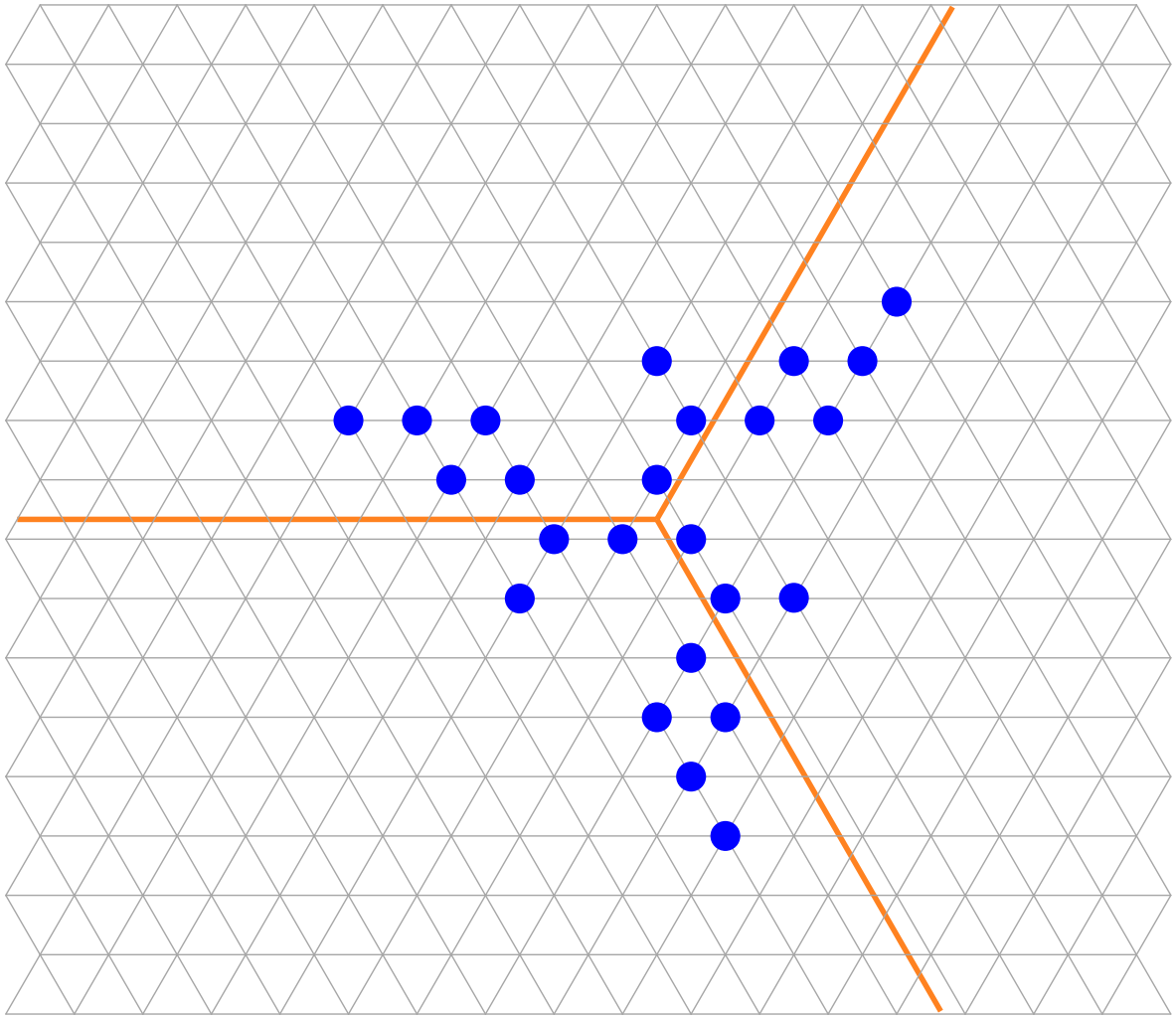
If $k > 1$ leaders have been elected in the previous phases, it means that the initial shape has an unbreakable k-fold symmetry.

Shape-Formation Phase



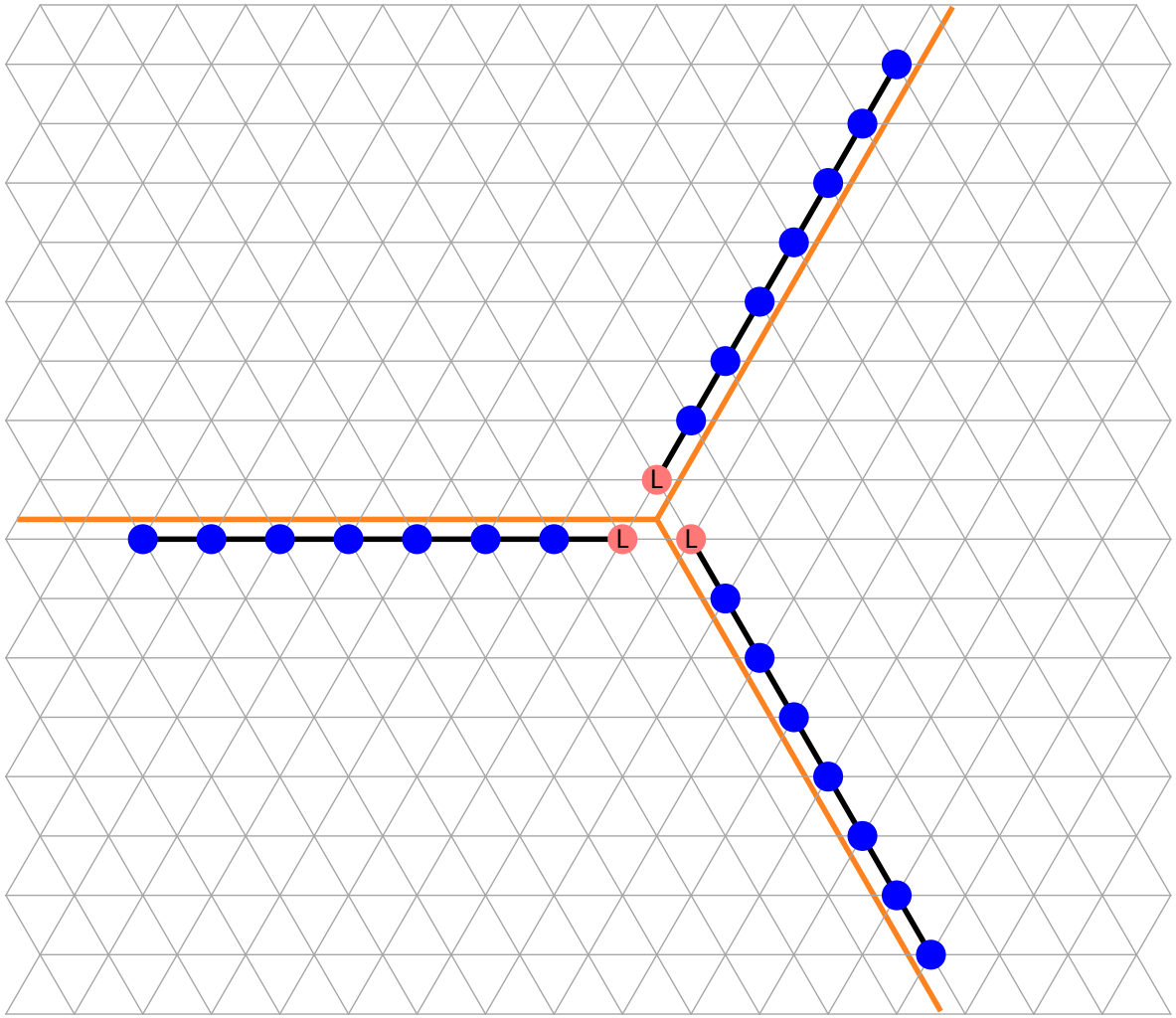
Hence, we may assume that also the shape to be formed has the same k-fold symmetry.

Shape-Formation Phase



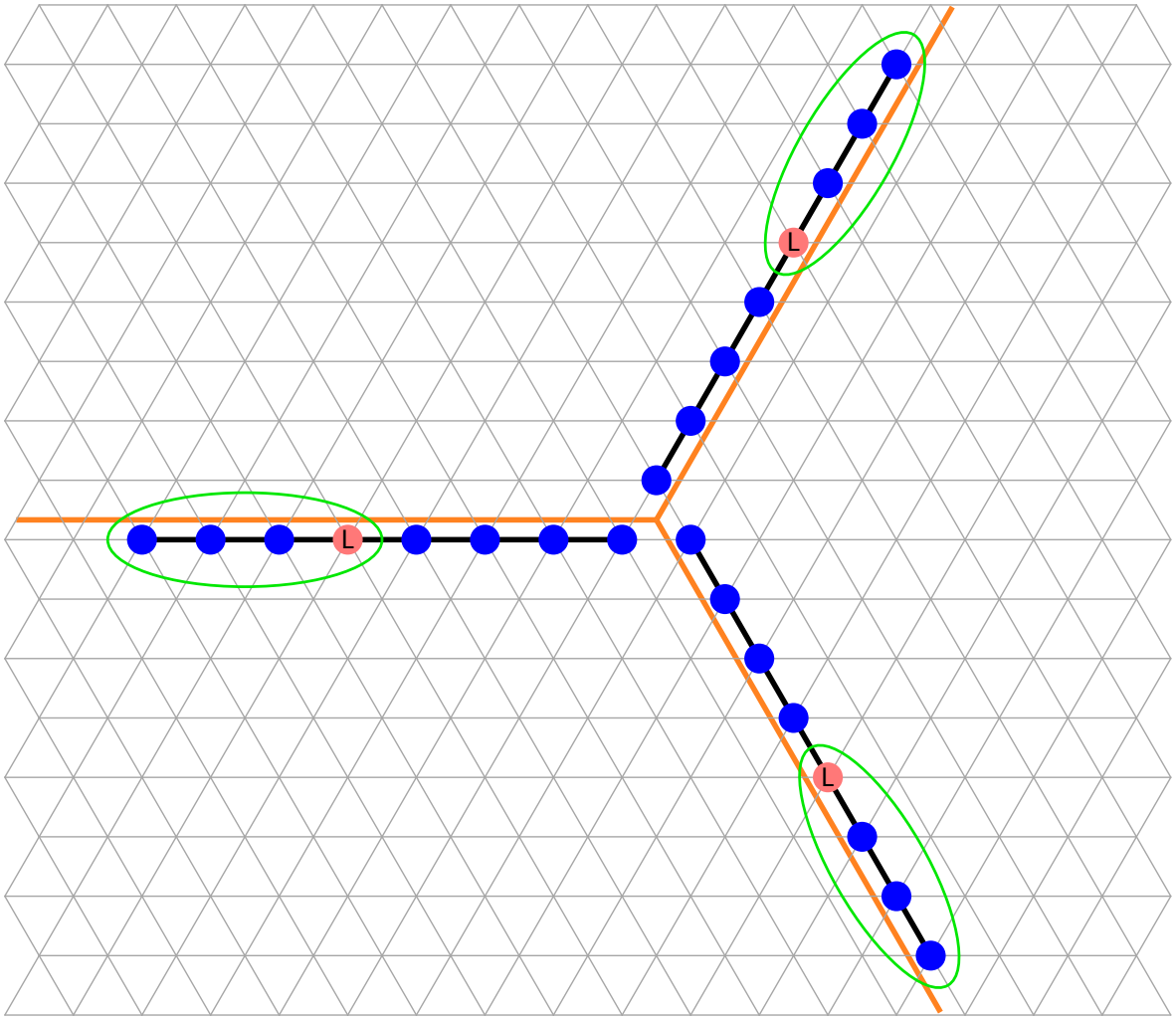
The plane is partitioned into k sectors, and each leader is tasked with forming the part of the shape that falls in its sector.

Shape-Formation Phase



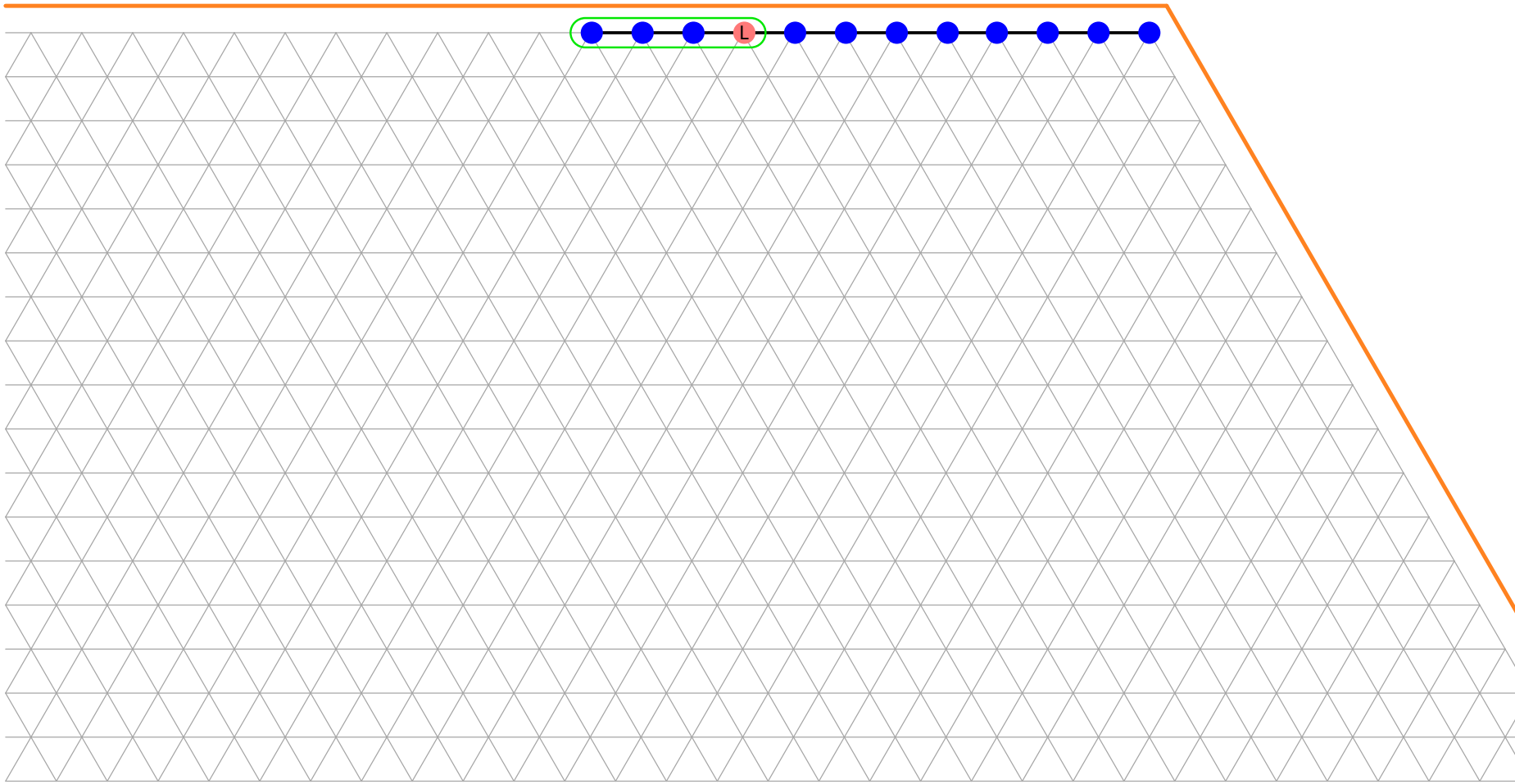
The plane is partitioned into k sectors, and each leader is tasked with forming the part of the shape that falls in its sector.

Shape-Formation Phase



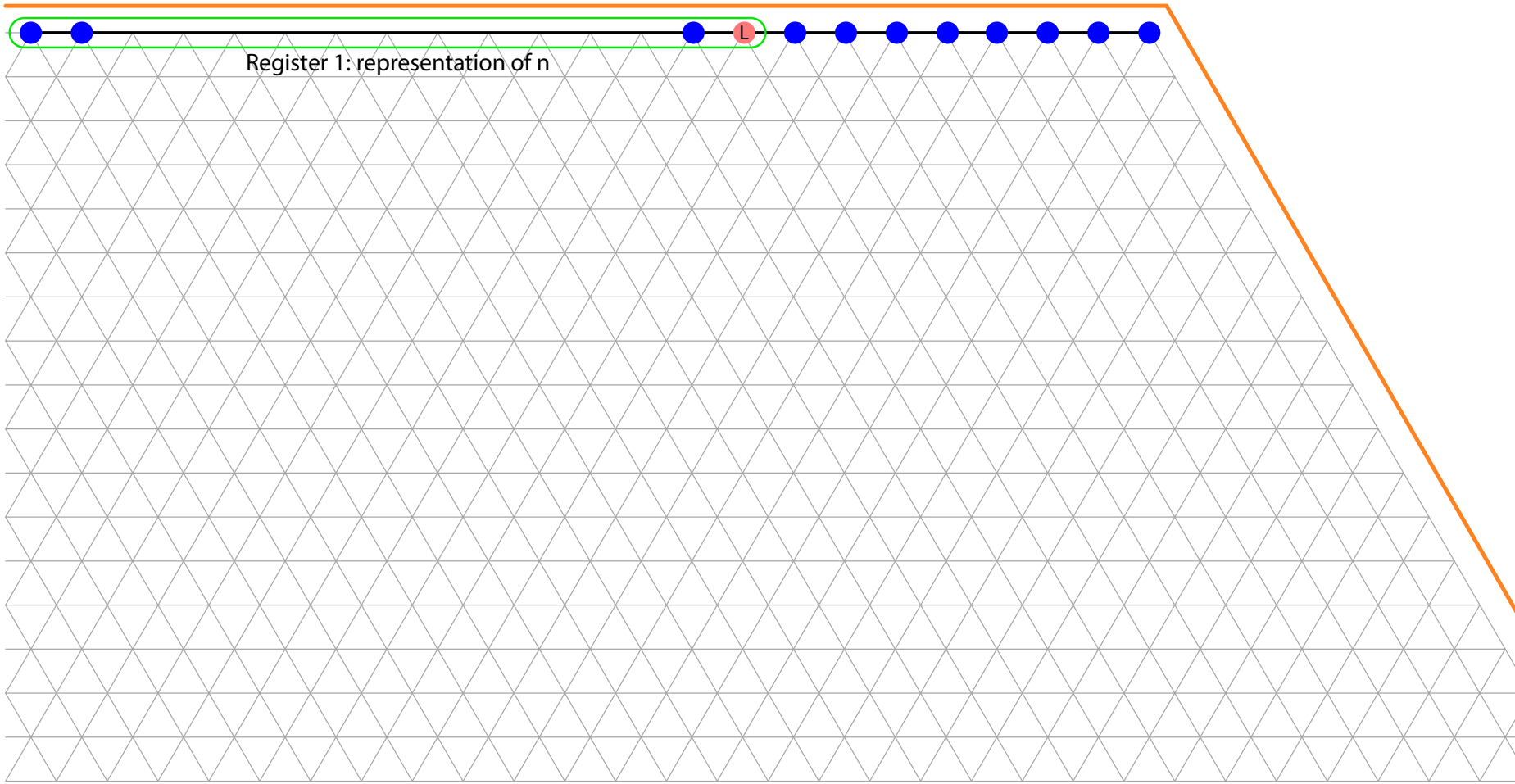
Assume there is an algorithm that, given n, generates the points of the shape. Let each leader simulate a RAM for that algorithm.

Shape-Formation Phase



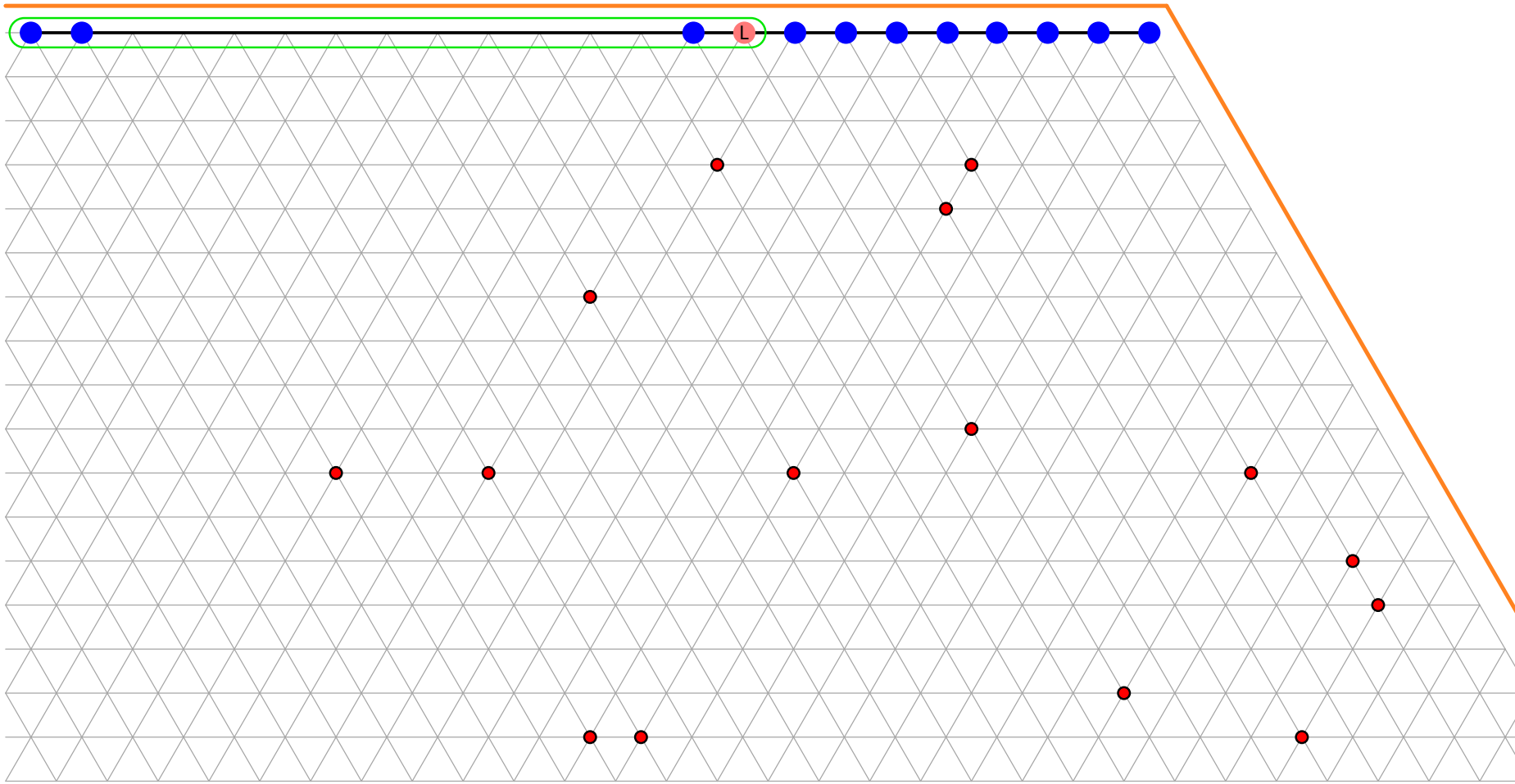
The leader takes position at the beginning of the simulated RAM.

Shape-Formation Phase



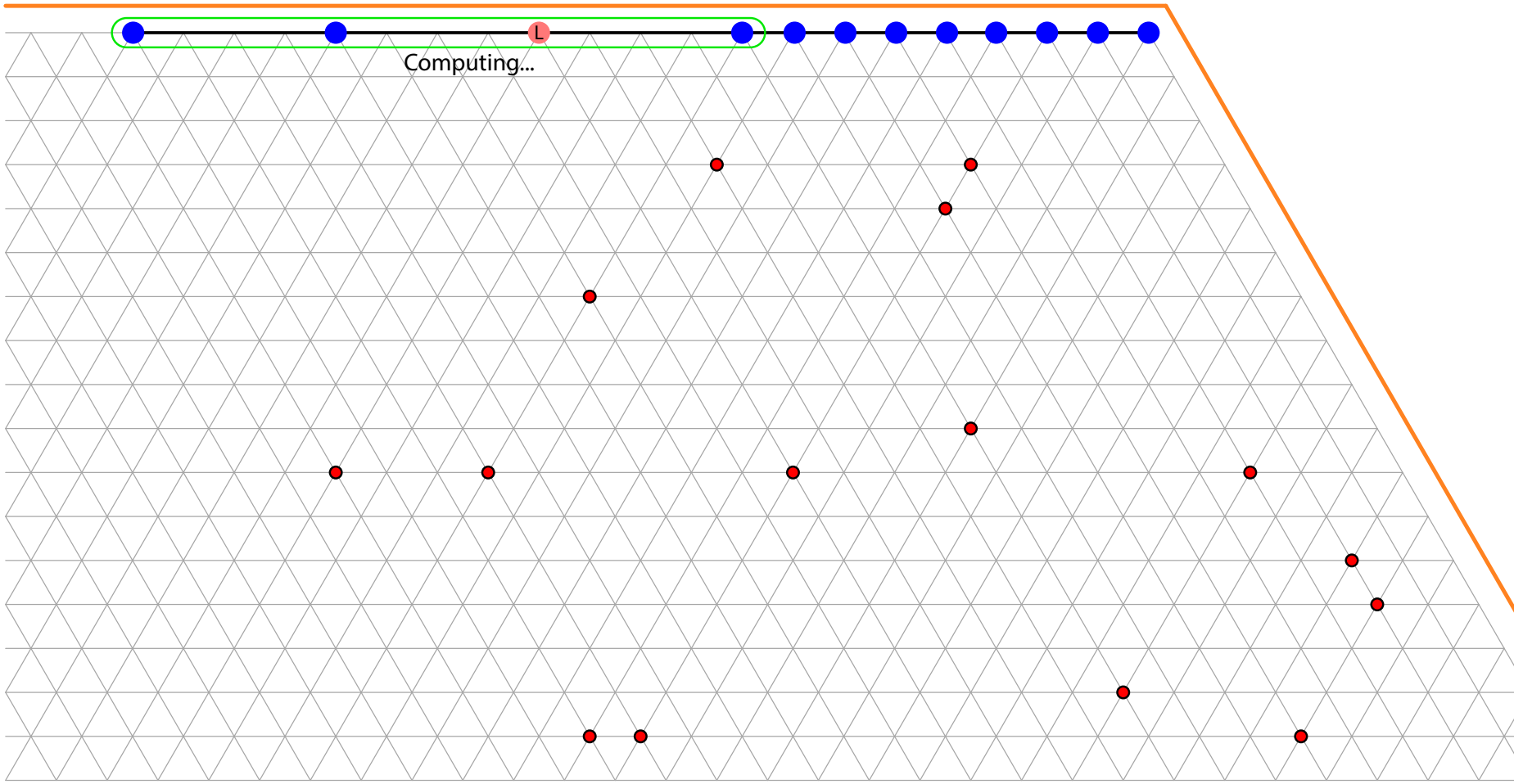
By scanning the previous part of the chain, it constructs a representation of n in the first register, which serves as the input.

Shape-Formation Phase



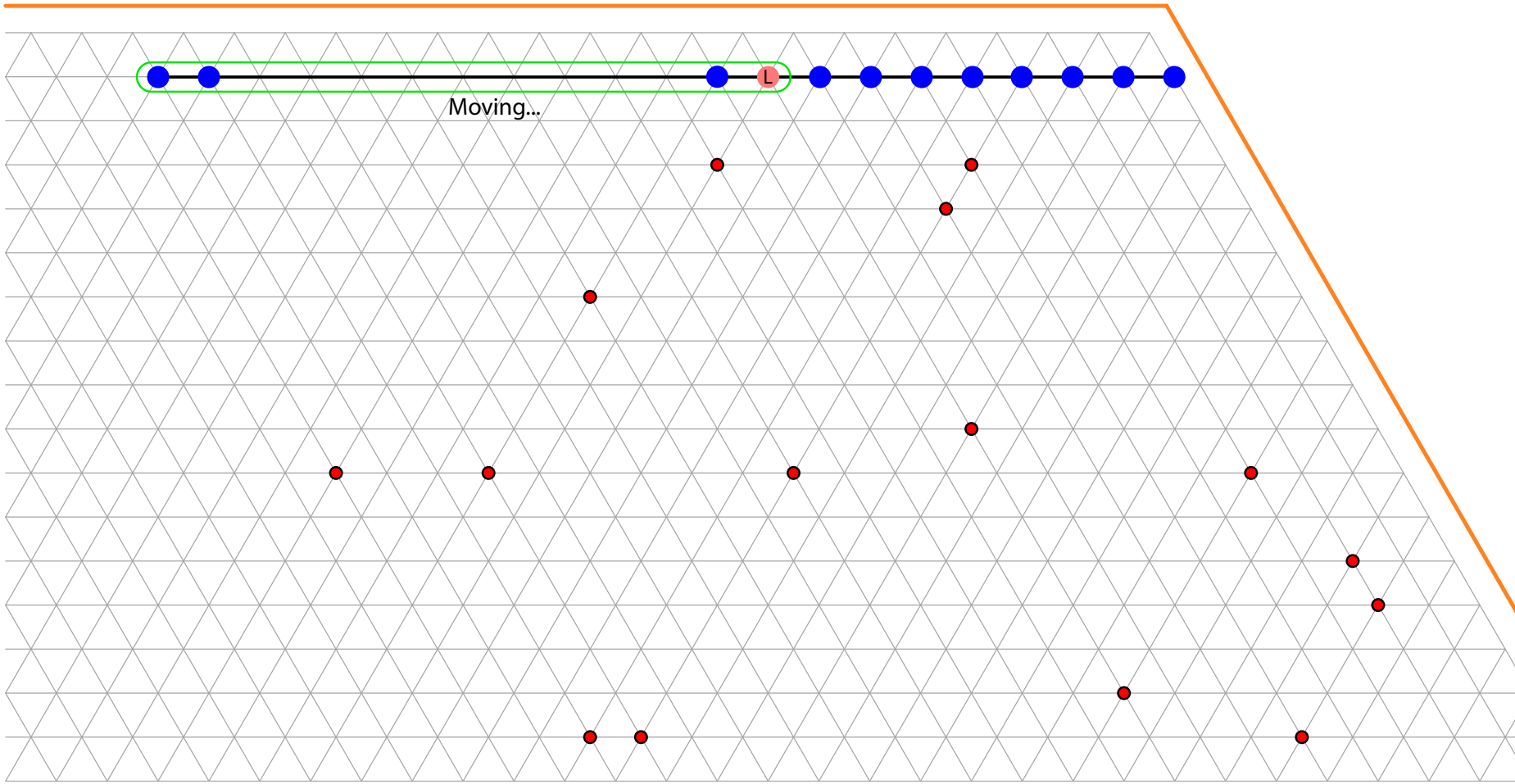
The simulated RAM will generate all the points of the shape and the sequence of moves necessary to reach them.

Shape-Formation Phase



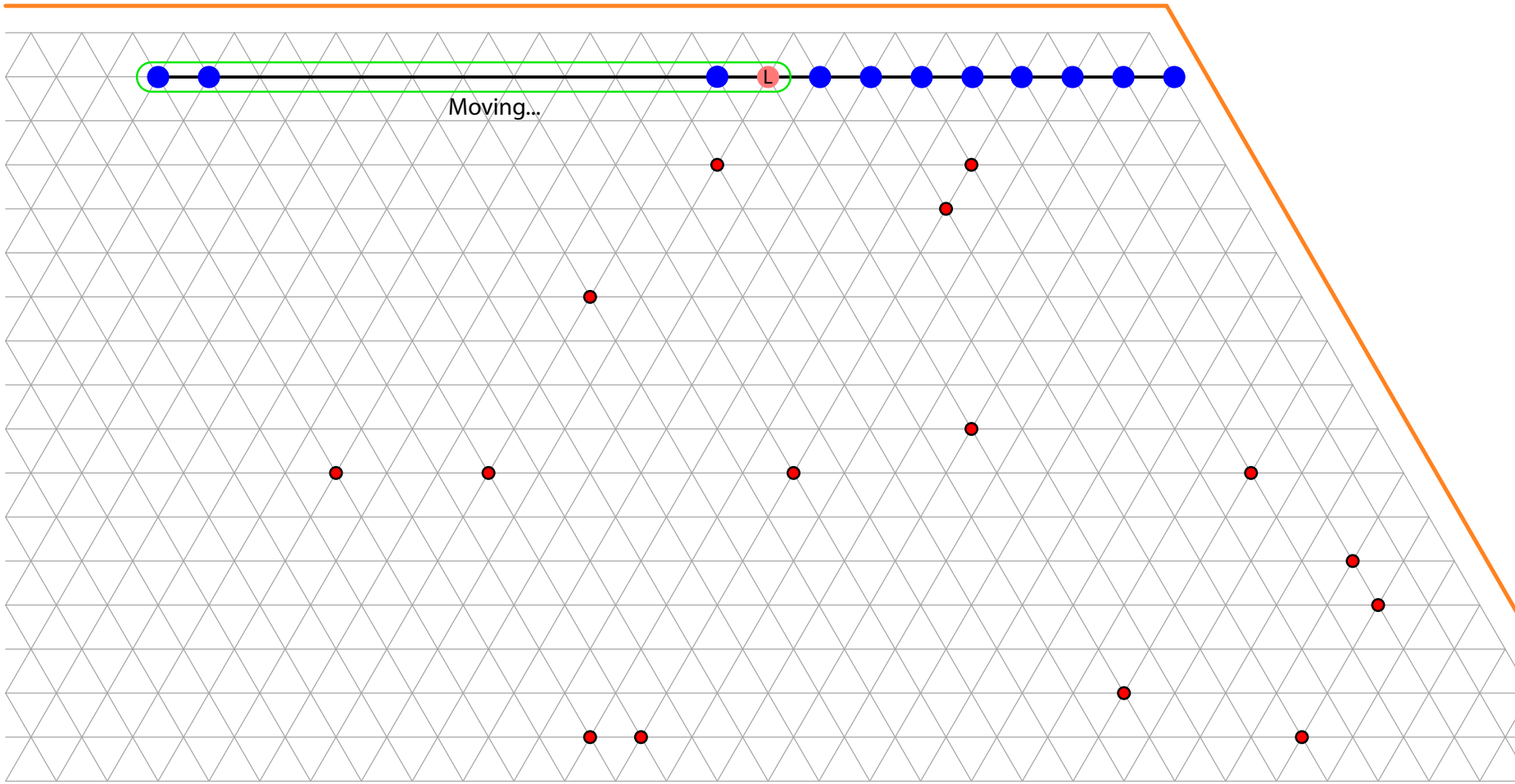
The simulated RAM computes the first point of the shape, while the rest of the chain does not move.

Shape-Formation Phase



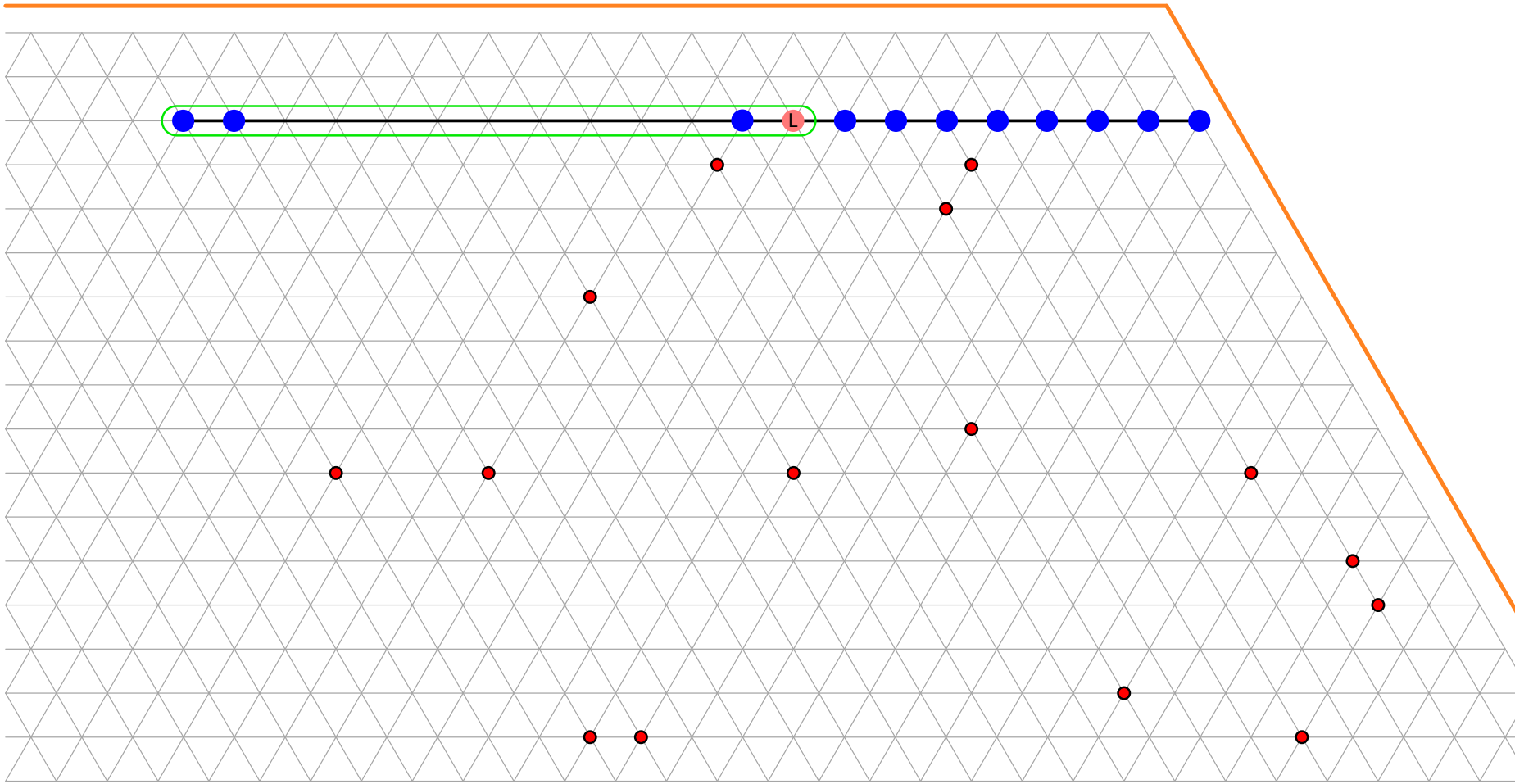
When the RAM has finished, the value of the first register indicates that the chain has to move in some direction.

Shape-Formation Phase



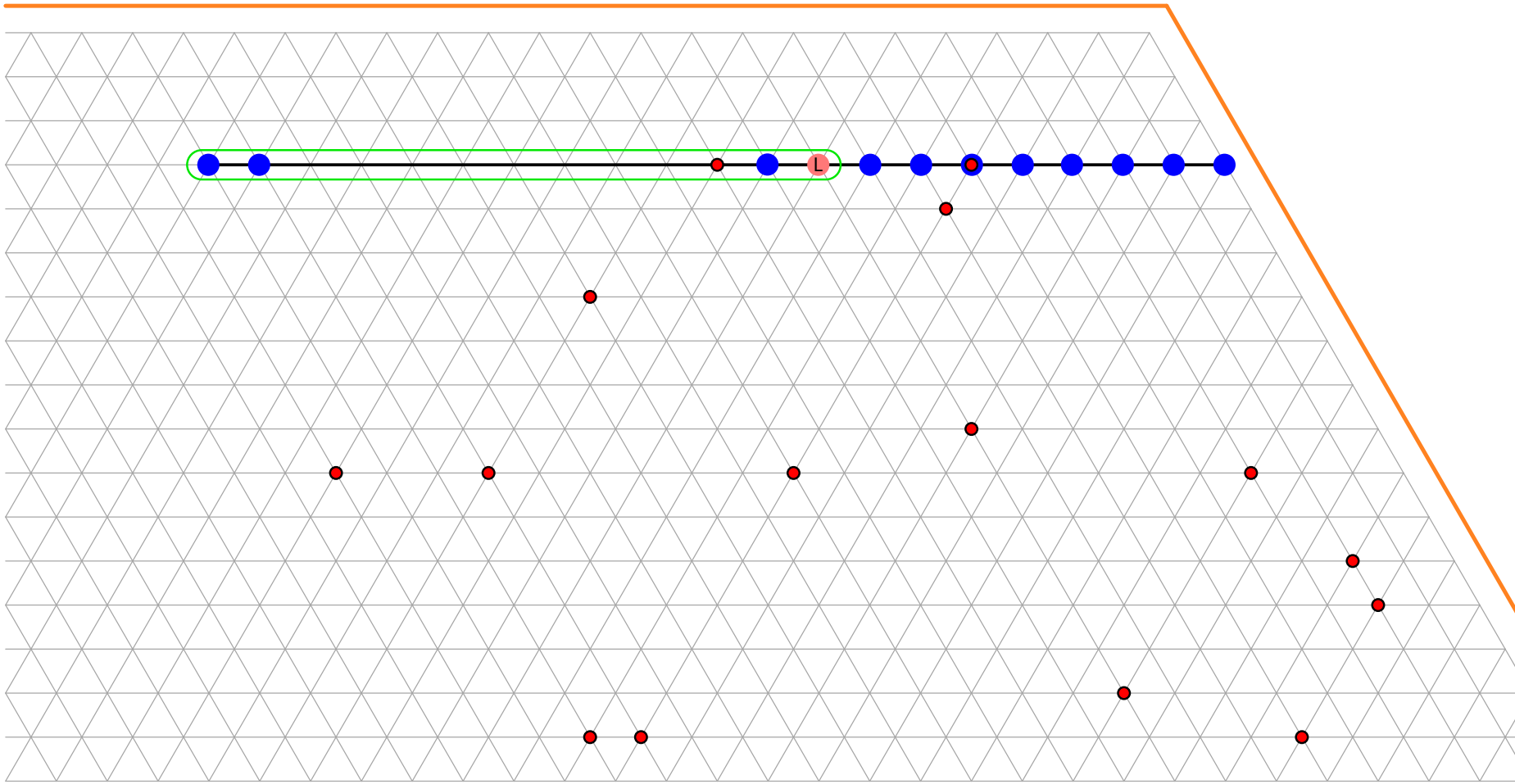
(The movement of the whole chain is coordinated by the leader, and takes place one particle at a time.)

Shape-Formation Phase



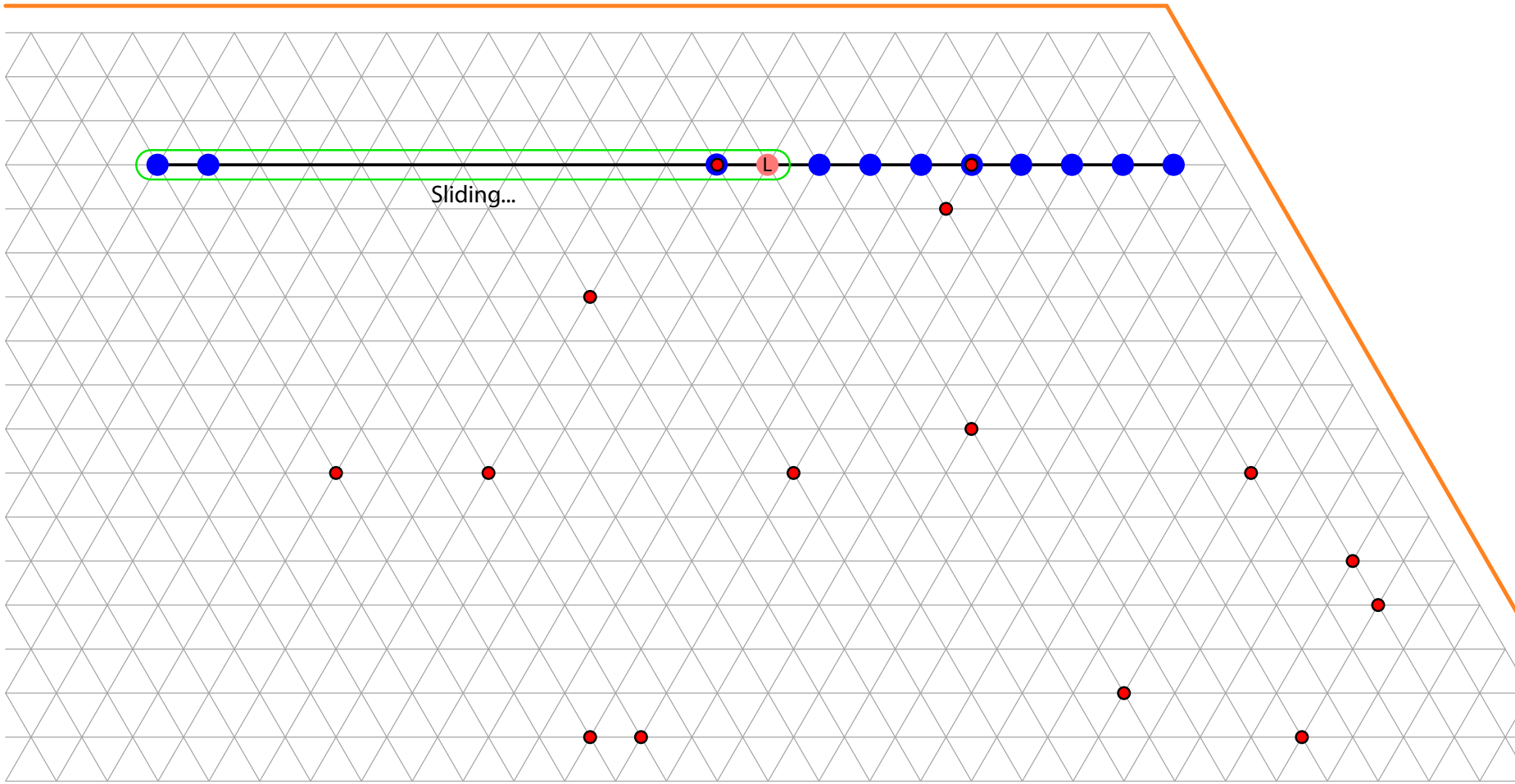
The RAM computes the next movement,
and the whole chain moves as soon as the computation is finished.

Shape-Formation Phase



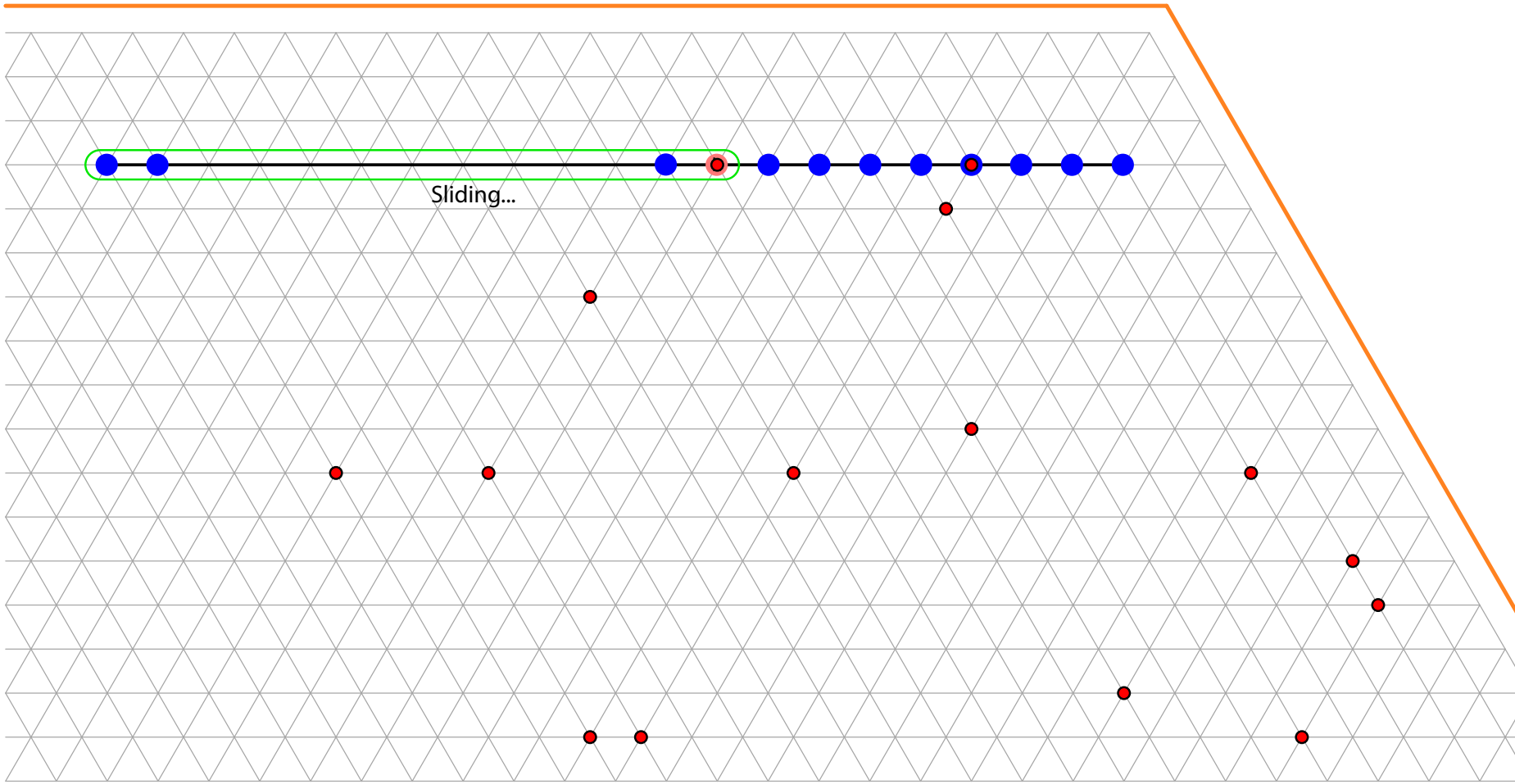
The RAM computes the next movement,
and the whole chain moves as soon as the computation is finished.

Shape-Formation Phase



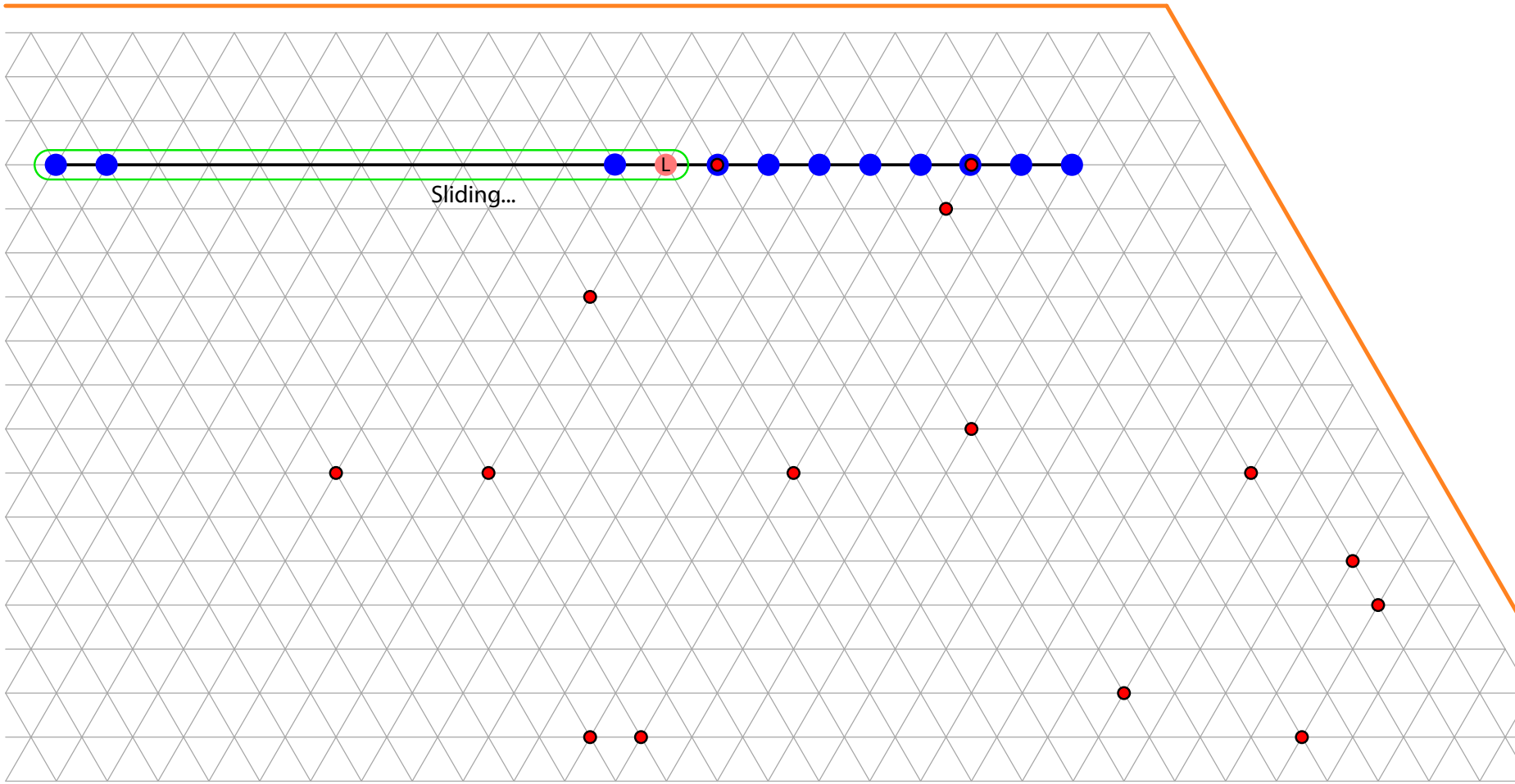
When the chain is on the same line as the first point of the shape, it slides until the last particle of the chain coincides with the point.

Shape-Formation Phase



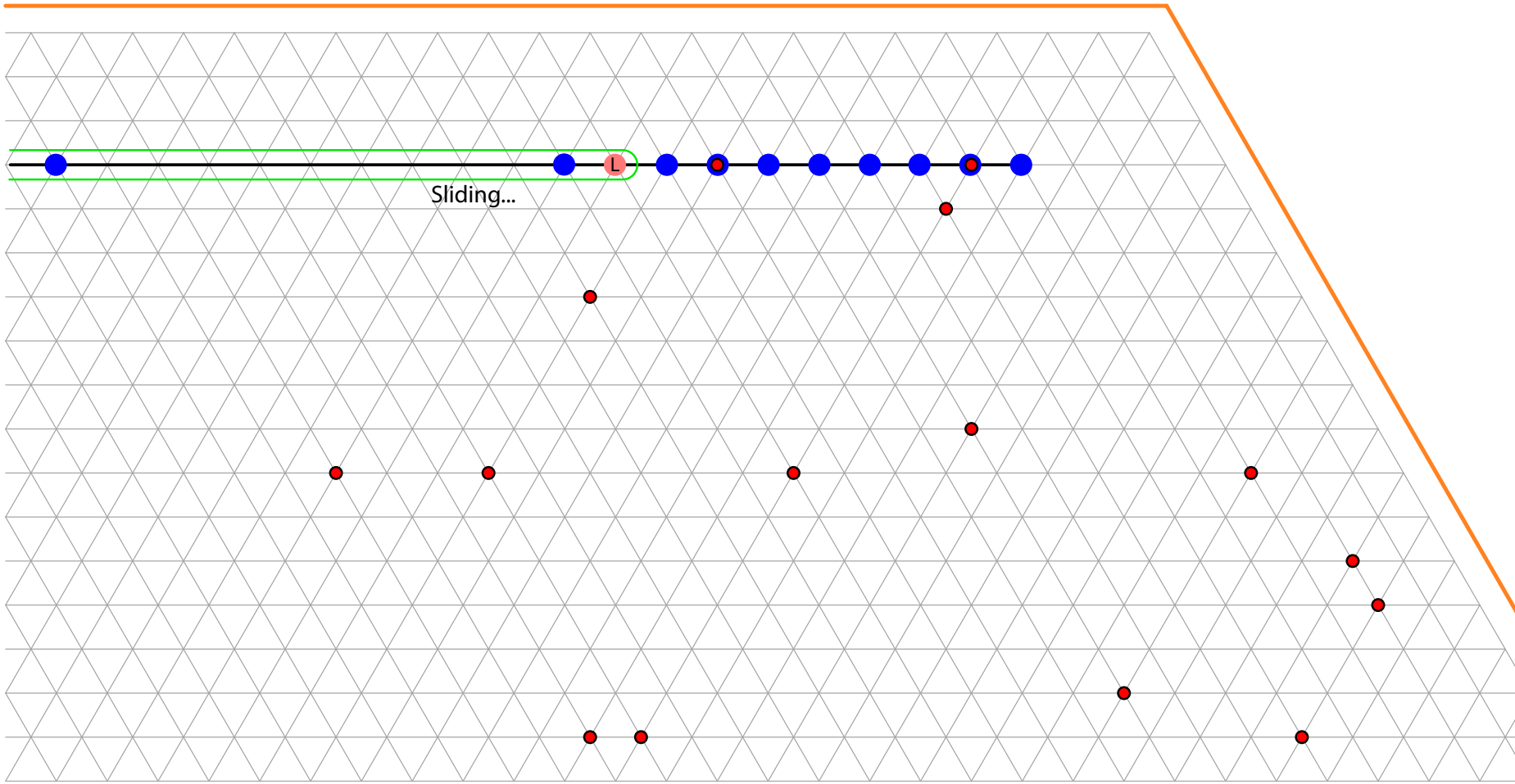
When the chain is on the same line as the first point of the shape, it slides until the last particle of the chain coincides with the point.

Shape-Formation Phase



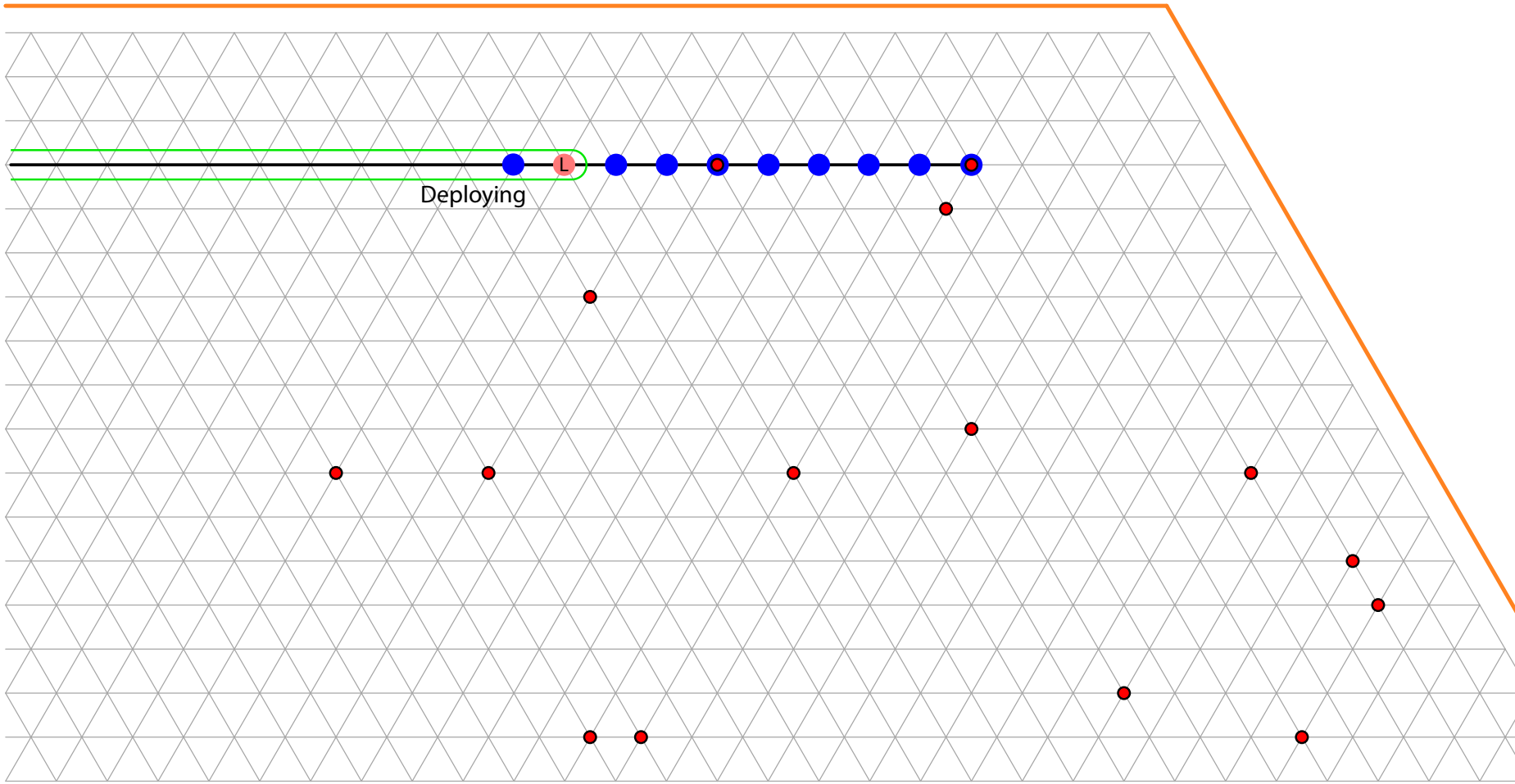
When the chain is on the same line as the first point of the shape, it slides until the last particle of the chain coincides with the point.

Shape-Formation Phase



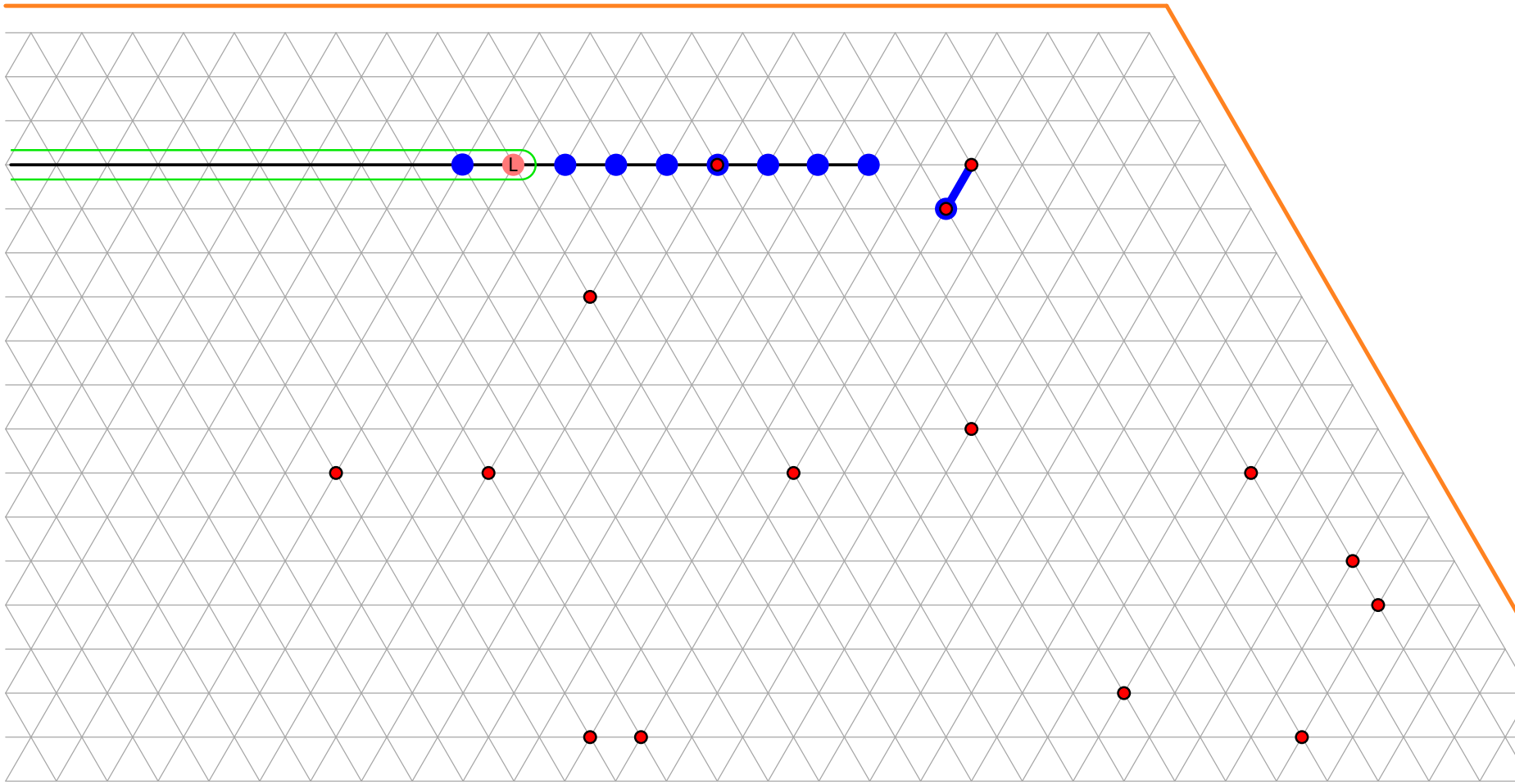
When the chain is on the same line as the first point of the shape, it slides until the last particle of the chain coincides with the point.

Shape-Formation Phase



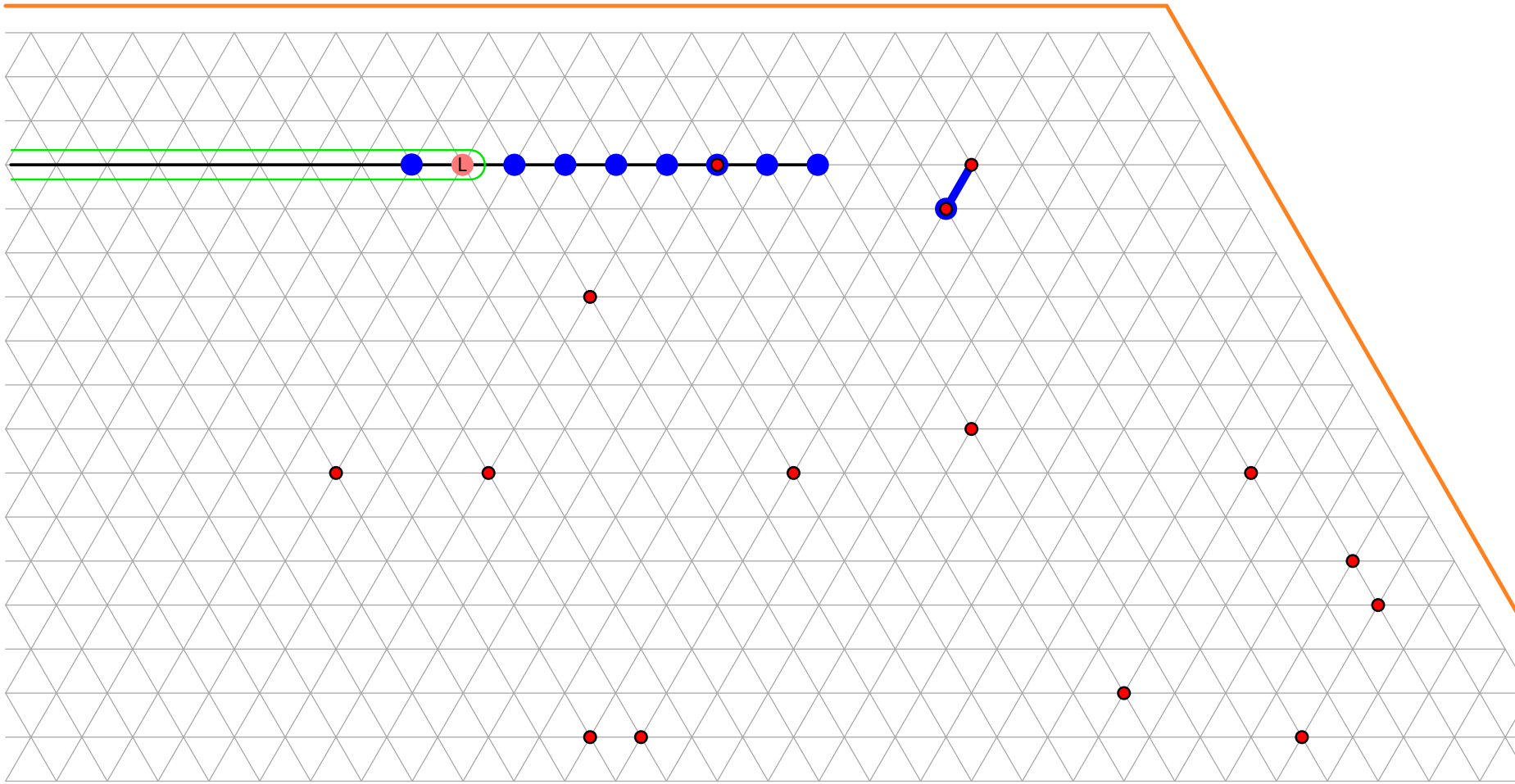
A message is forwarded to the last particle, telling it to stay there, and perhaps expand in some direction to cover two points.

Shape-Formation Phase



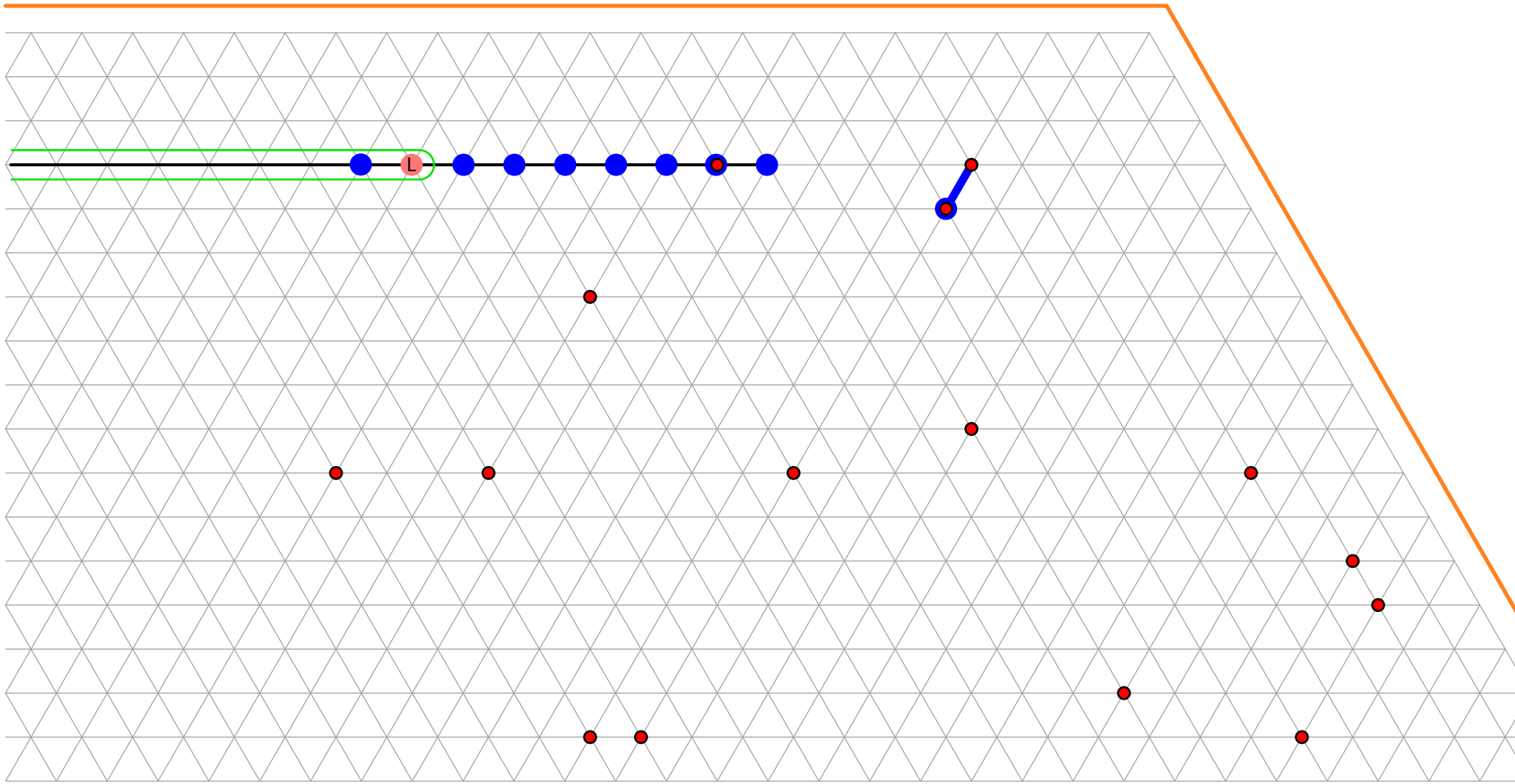
A message is forwarded to the last particle, telling it to stay there, and perhaps expand in some direction to cover two points.

Shape-Formation Phase



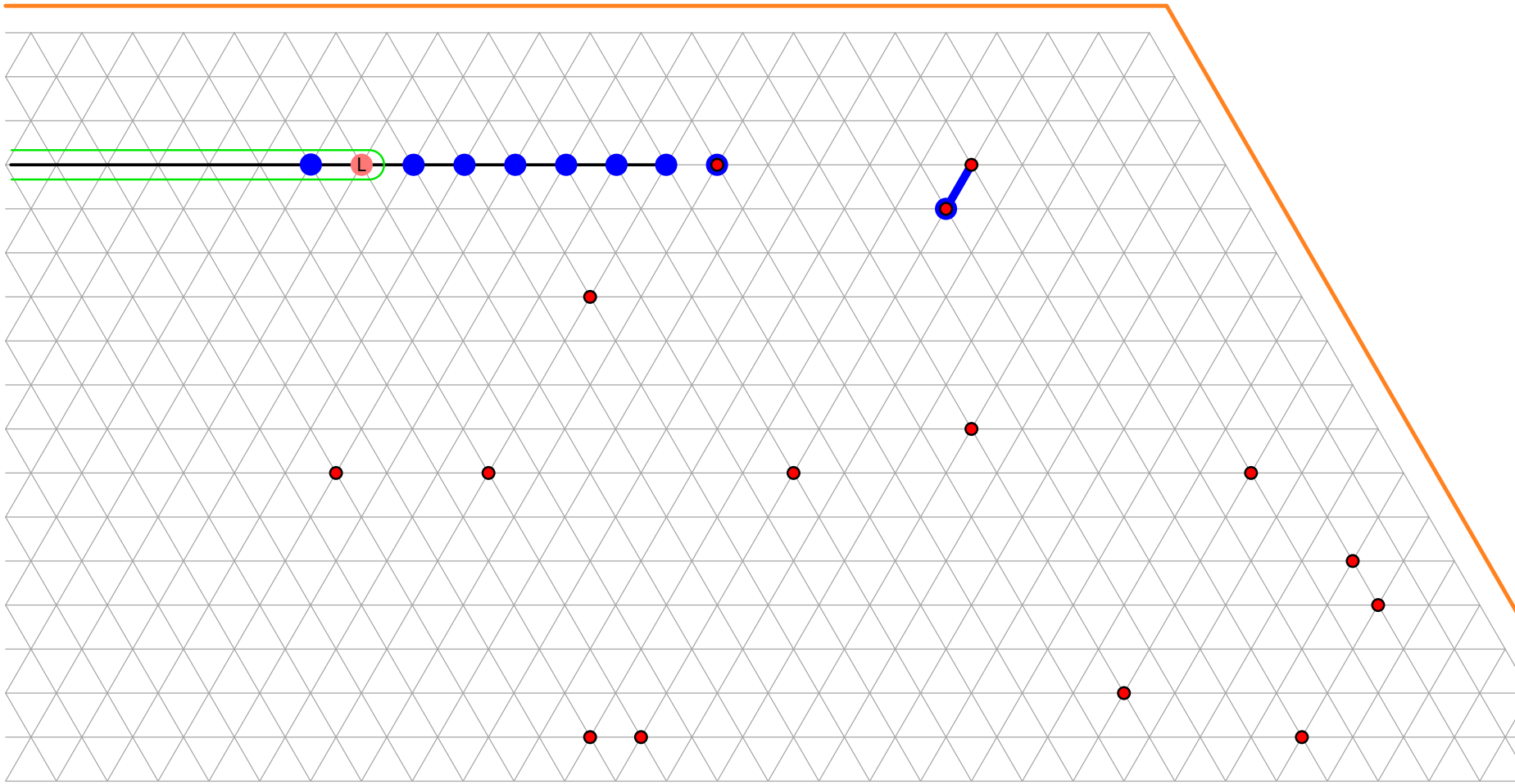
The protocol proceeds in the same fashion with the other points of the shape.

Shape-Formation Phase



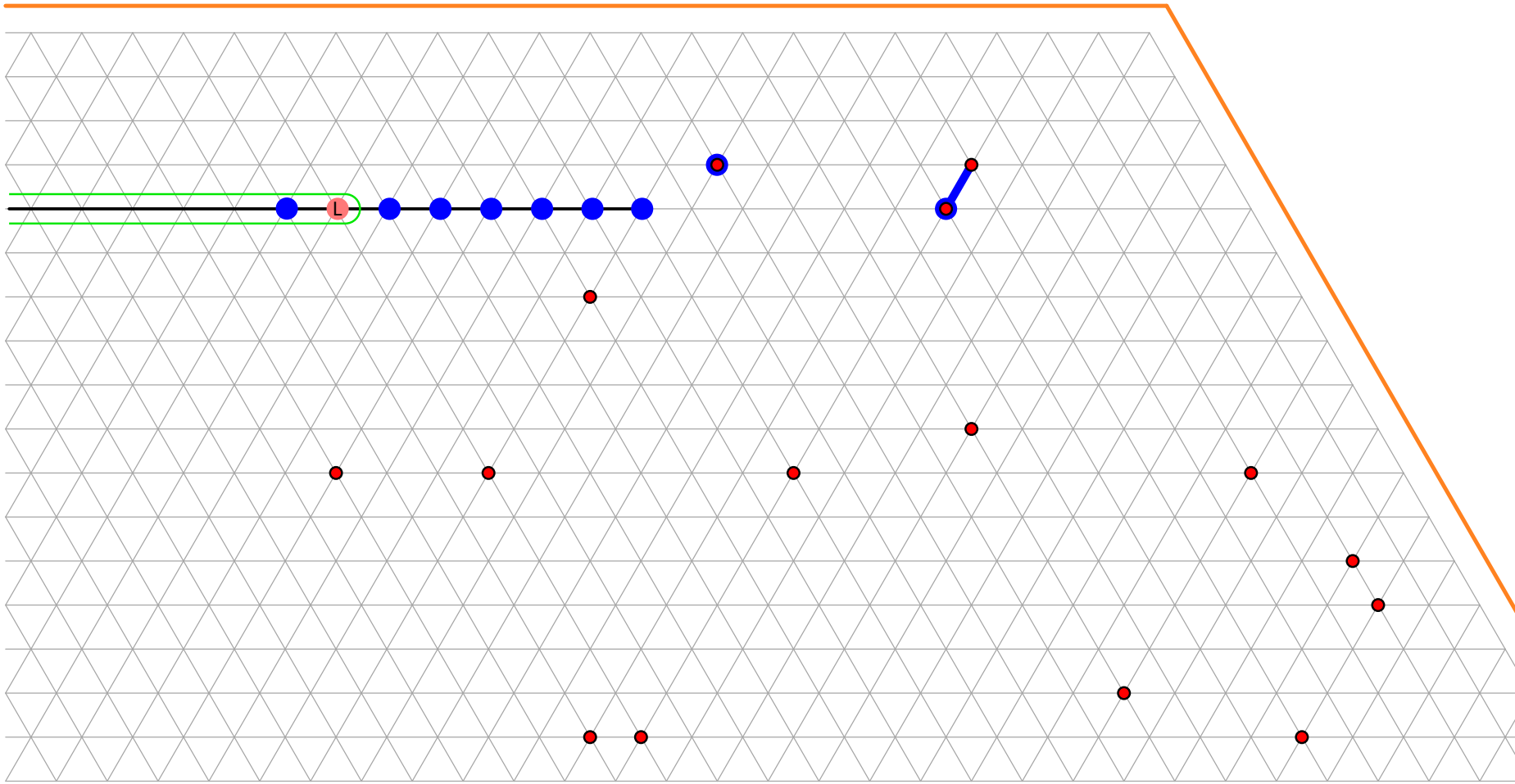
The protocol proceeds in the same fashion with the other points of the shape.

Shape-Formation Phase



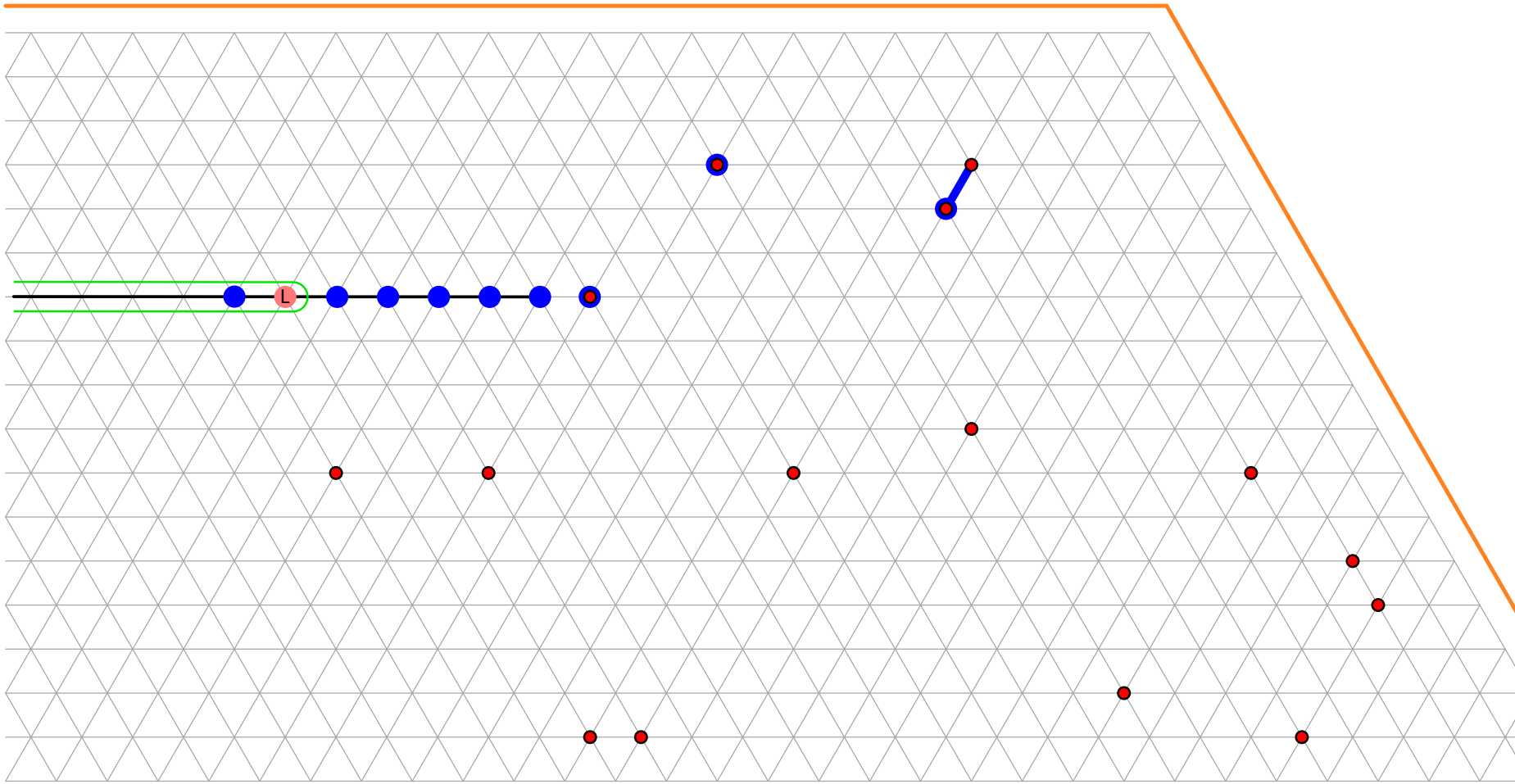
The protocol proceeds in the same fashion with the other points of the shape.

Shape-Formation Phase



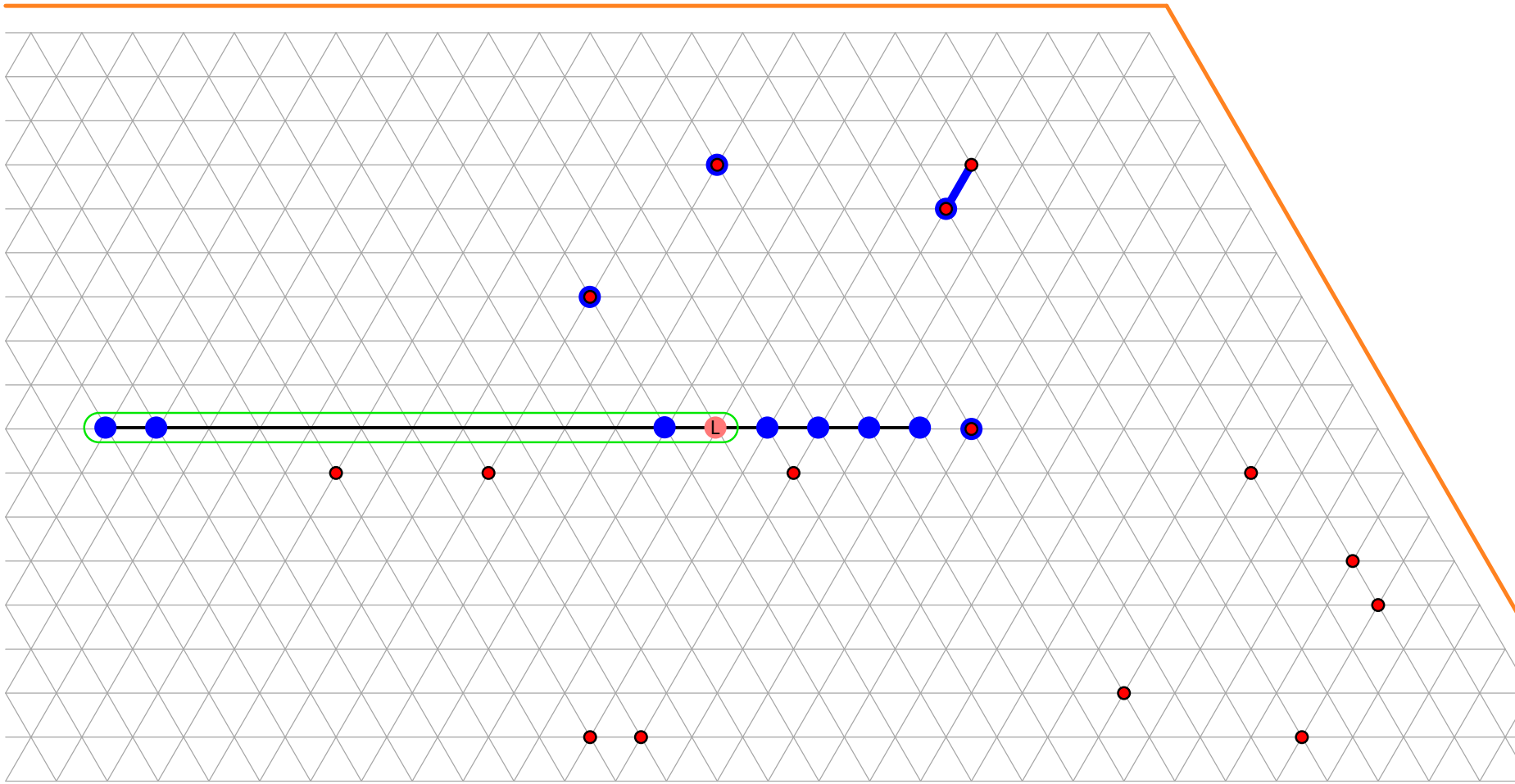
The protocol proceeds in the same fashion with the other points of the shape.

Shape-Formation Phase



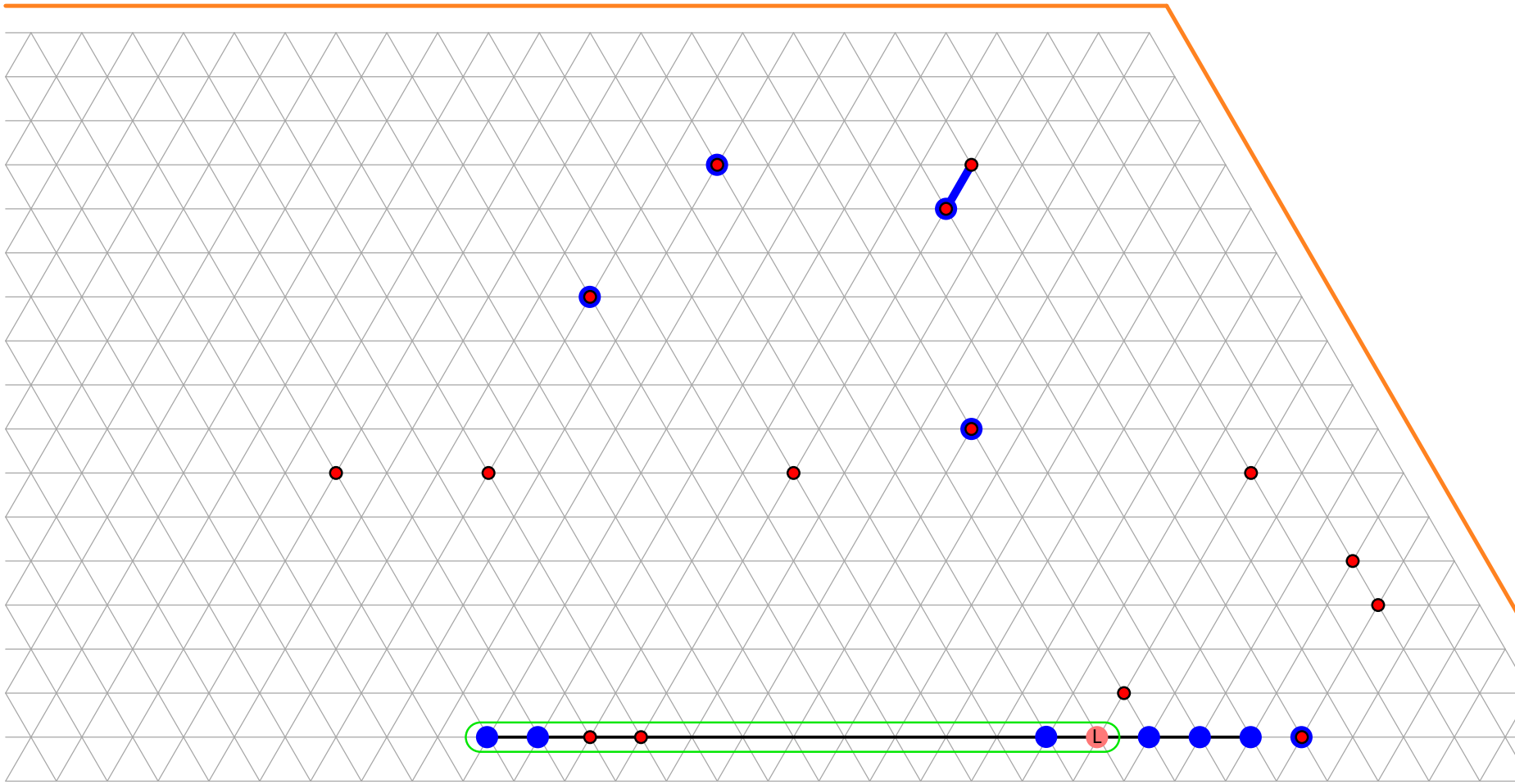
The protocol proceeds in the same fashion with the other points of the shape.

Shape-Formation Phase



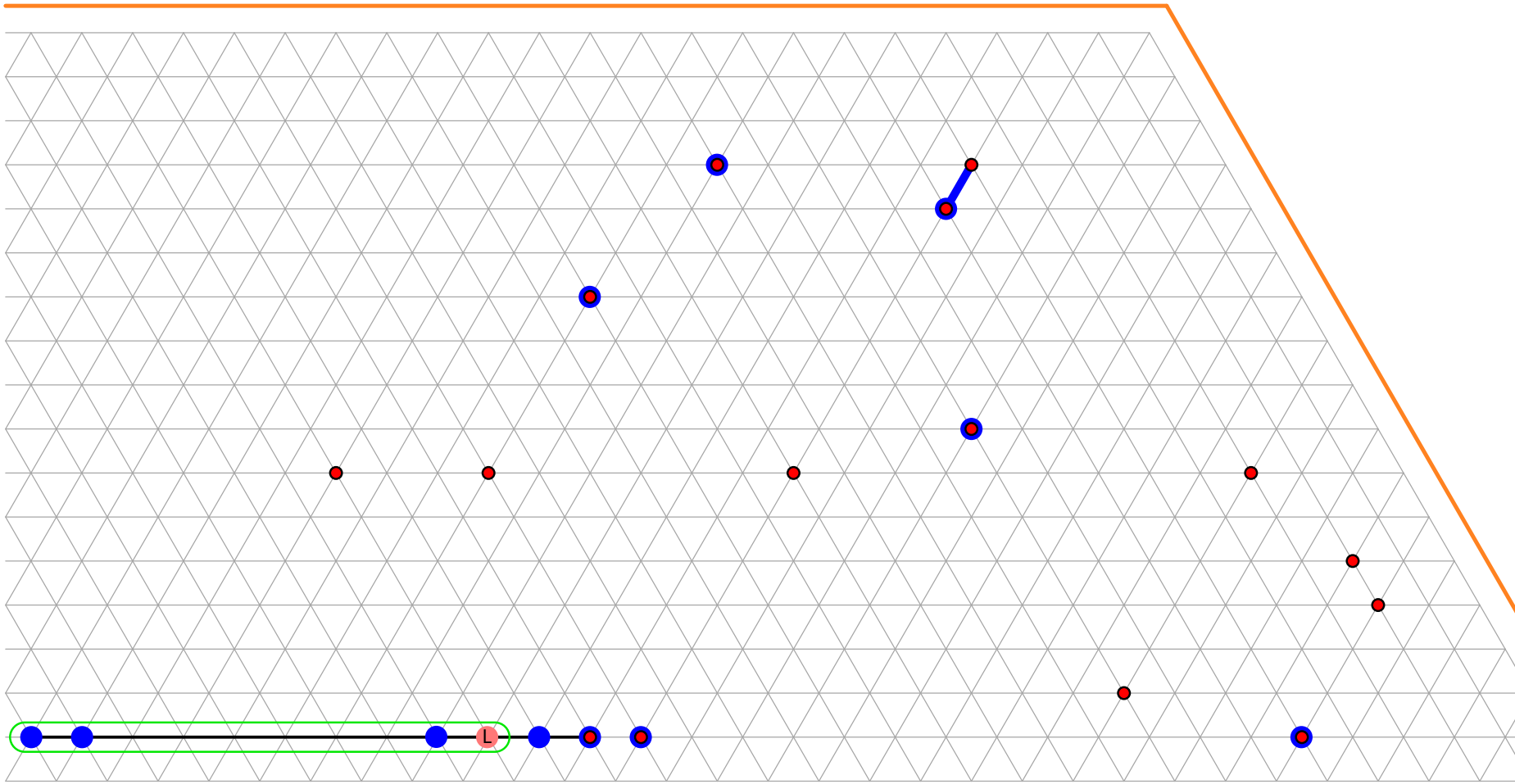
The protocol proceeds in the same fashion with the other points of the shape.

Shape-Formation Phase



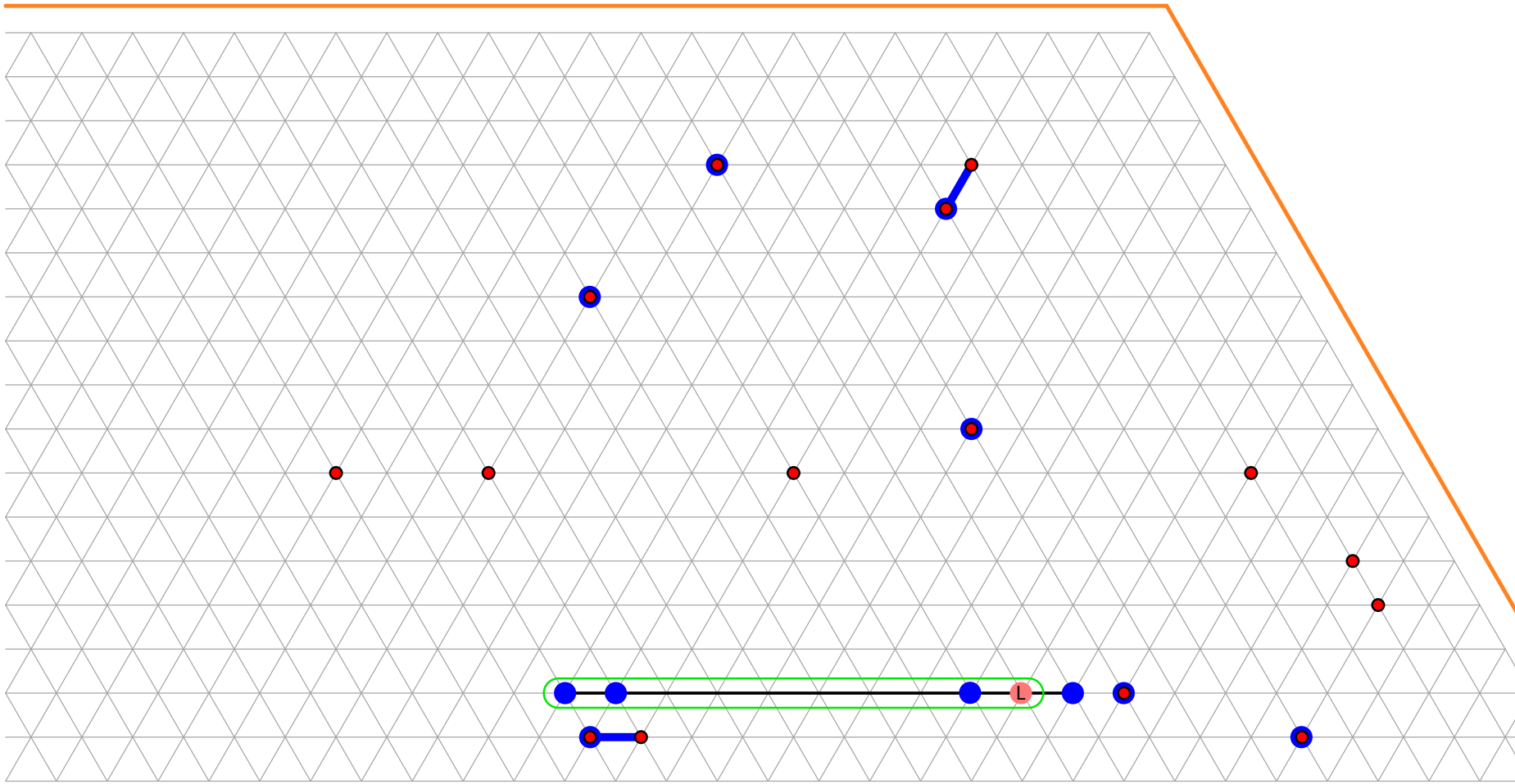
The protocol proceeds in the same fashion with the other points of the shape.

Shape-Formation Phase



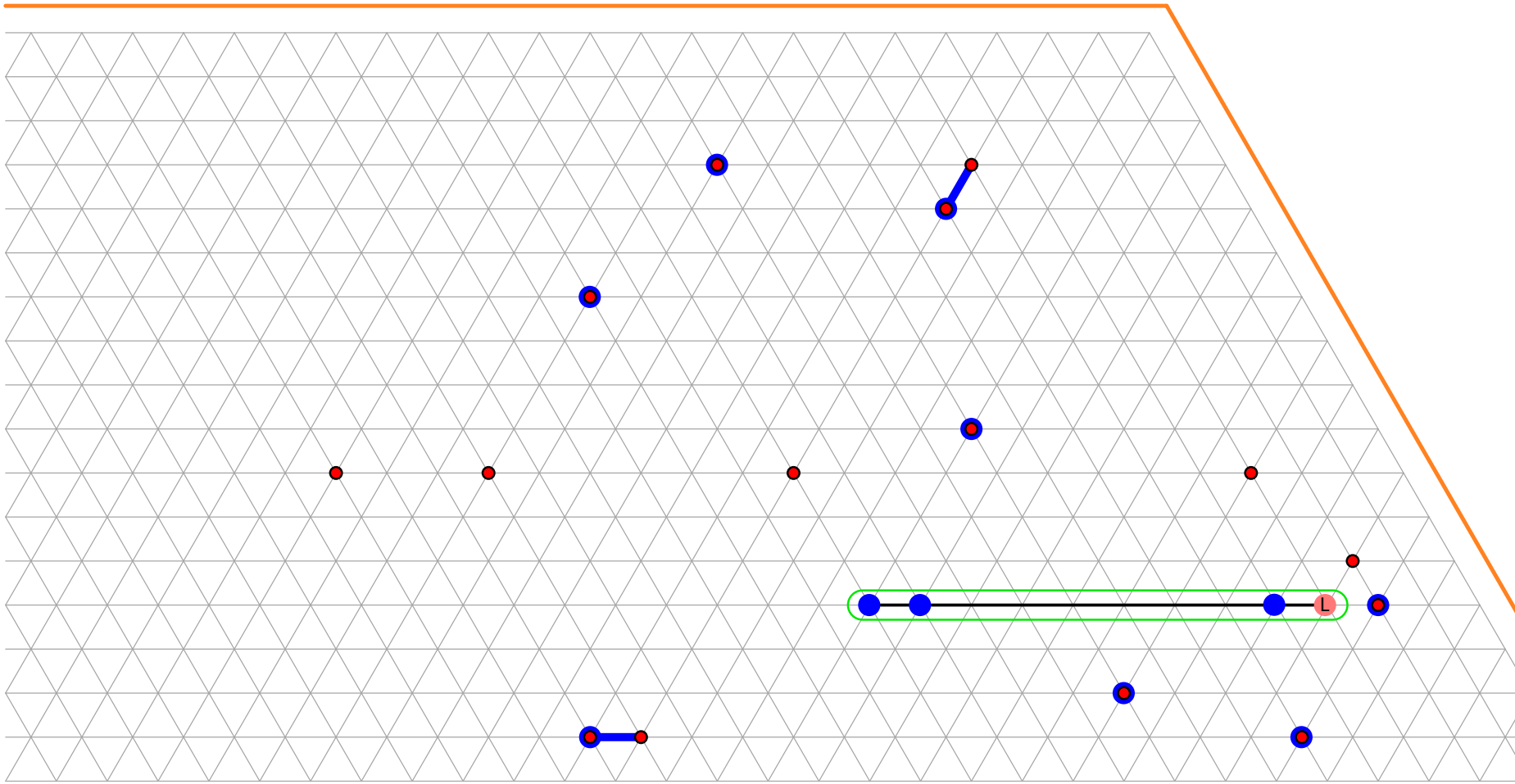
The protocol proceeds in the same fashion with the other points of the shape.

Shape-Formation Phase



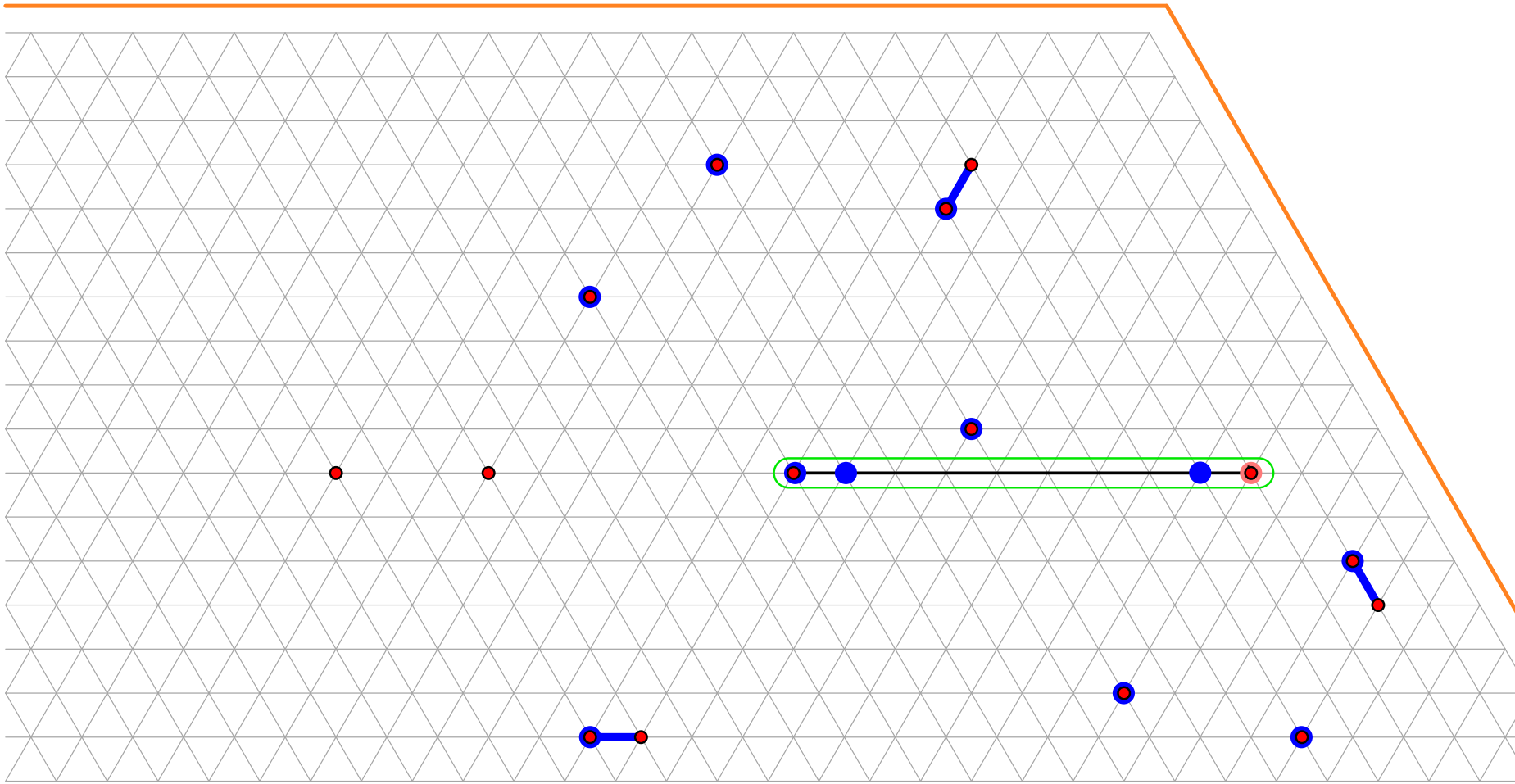
The protocol proceeds in the same fashion with the other points of the shape.

Shape-Formation Phase



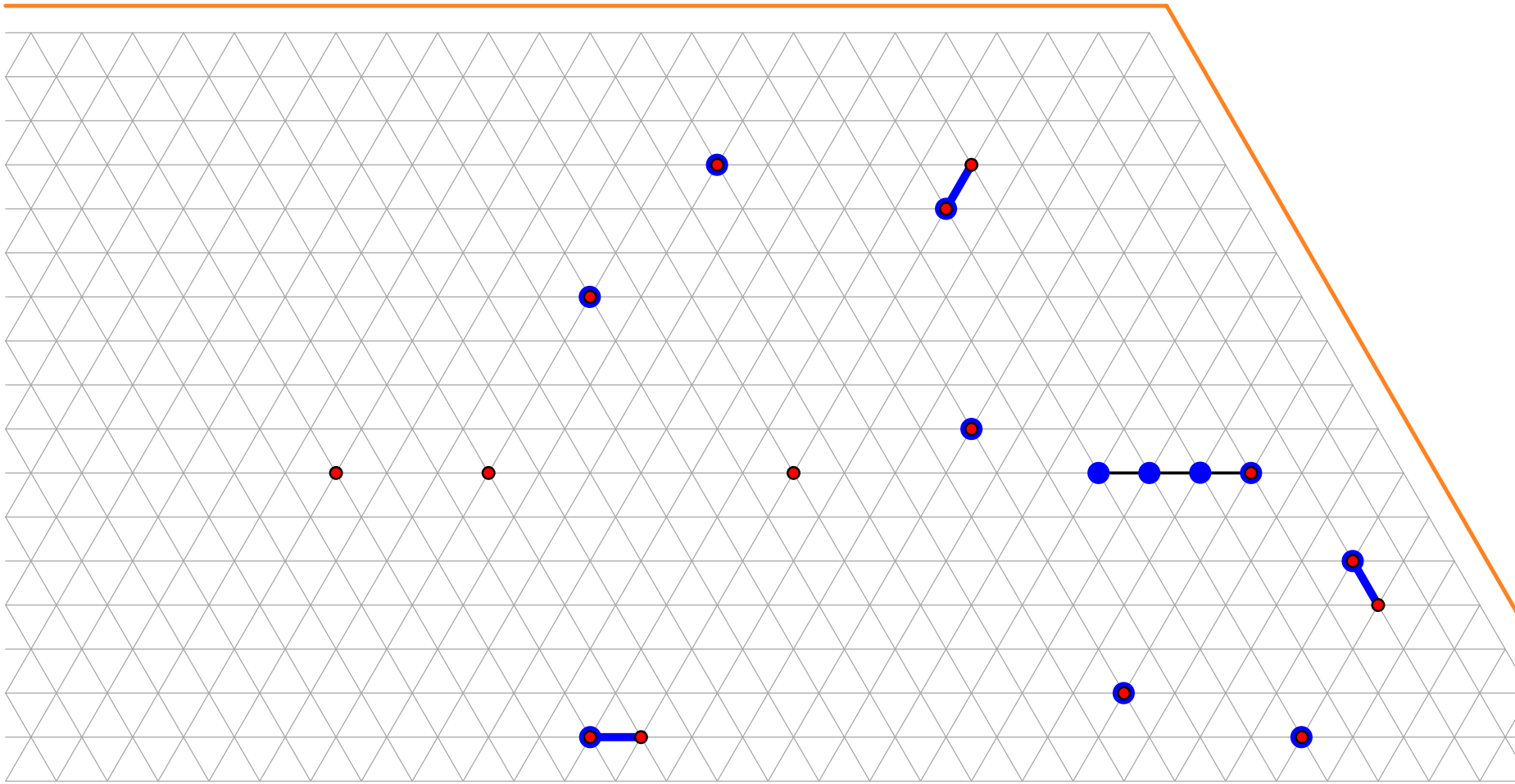
The protocol proceeds in the same fashion with the other points of the shape.

Shape-Formation Phase



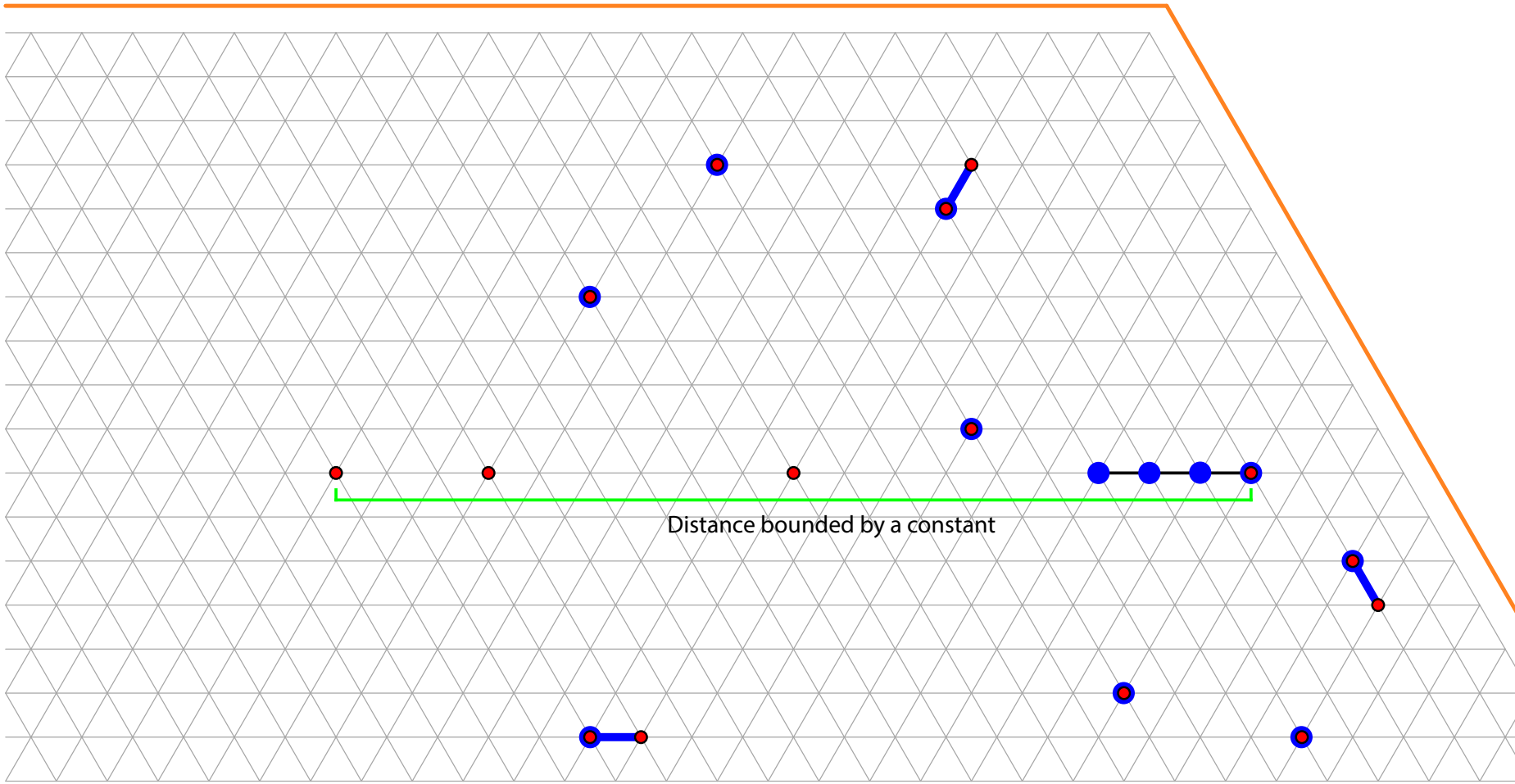
The algorithm ensures that the last 4 points of the shape are “in the same neighborhood”.

Shape-Formation Phase



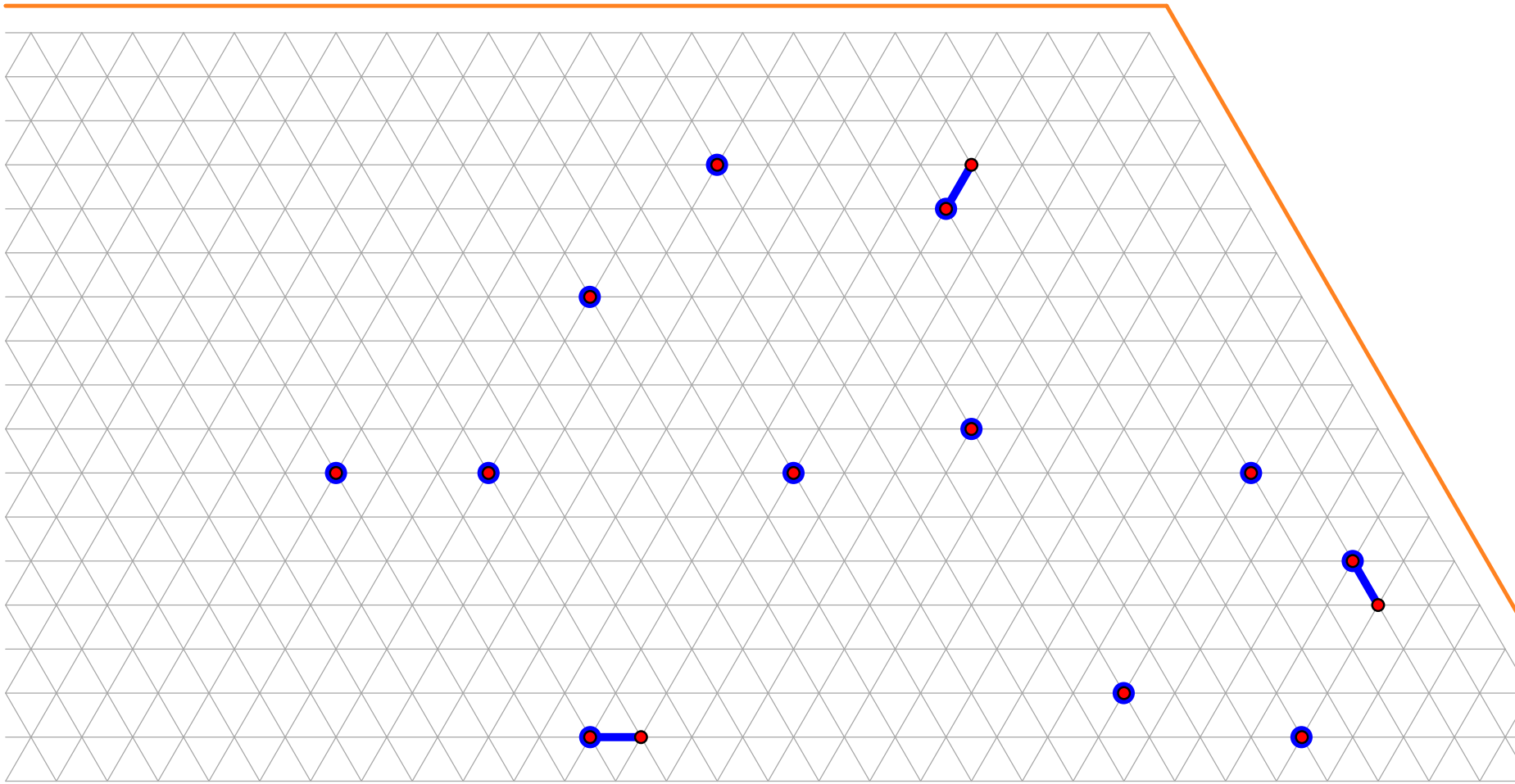
When the leader is on the first of these 4 points,
it makes the RAM contract, erasing the registers.

Shape-Formation Phase



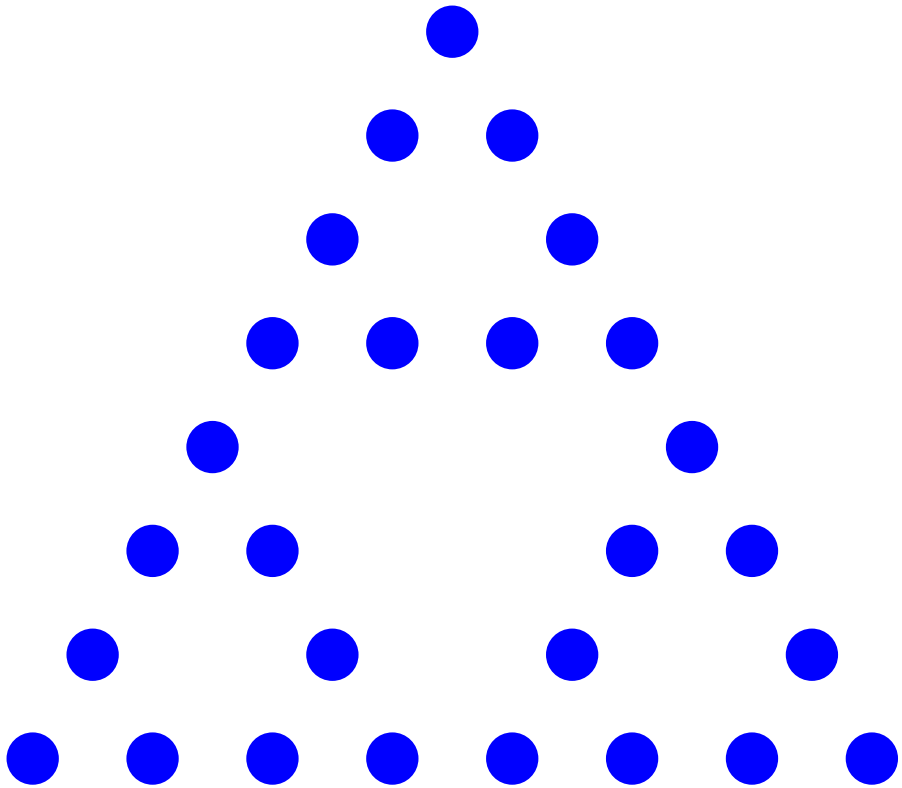
Assuming that the distance of the other 3 points is bounded by a constant, the particles can reach them using constant memory.

Shape-Formation Phase



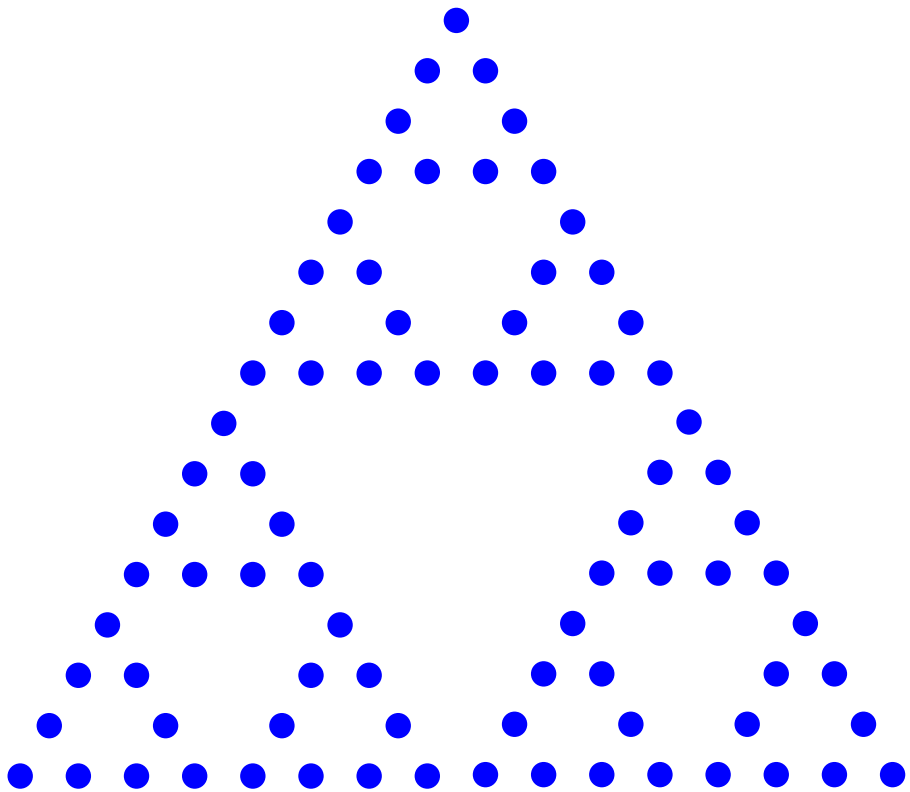
Assuming that the distance of the other 3 points is bounded by a constant, the particles can reach them using constant memory.

Fractal and Curved Shapes



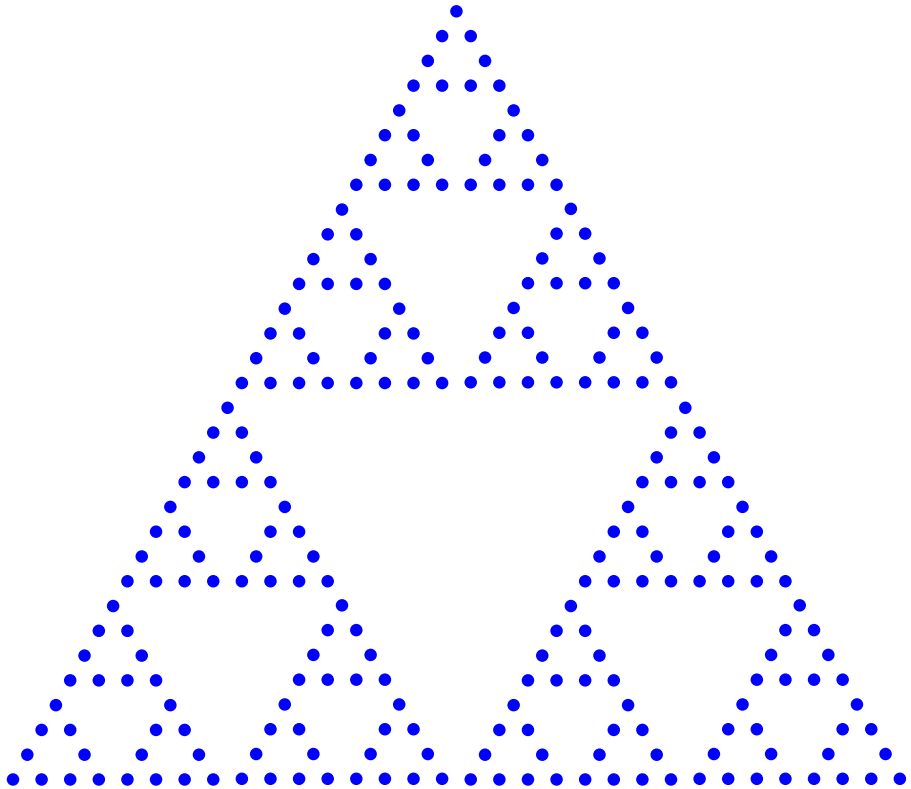
This protocol allows the system to form shapes that scale up like fractals (e.g., the Sierpinski triangle), as well as curved objects, which are better approximated as the number of particles increases.

Fractal and Curved Shapes



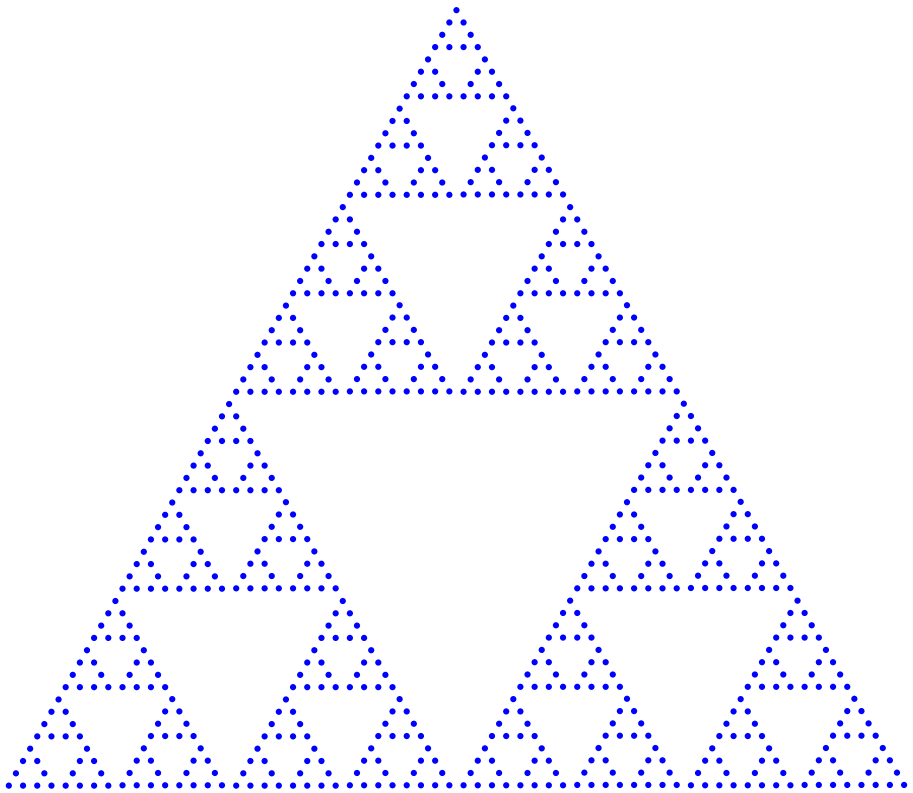
This protocol allows the system to form shapes that scale up like fractals (e.g., the Sierpinski triangle), as well as curved objects, which are better approximated as the number of particles increases.

Fractal and Curved Shapes



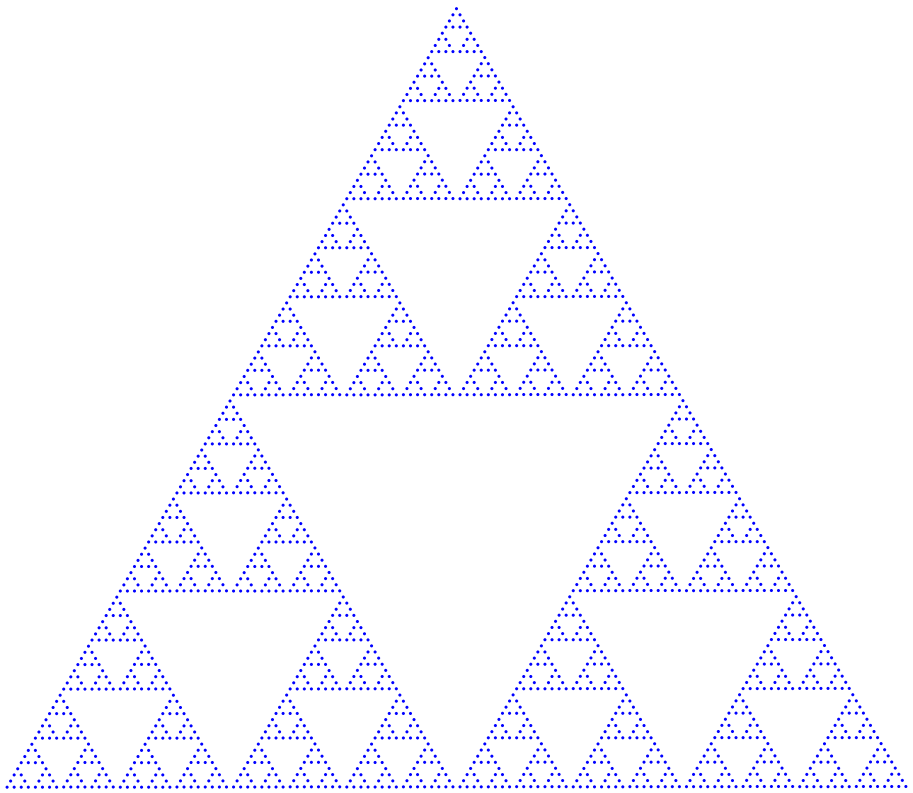
This protocol allows the system to form shapes that scale up like fractals (e.g., the Sierpinski triangle), as well as curved objects, which are better approximated as the number of particles increases.

Fractal and Curved Shapes



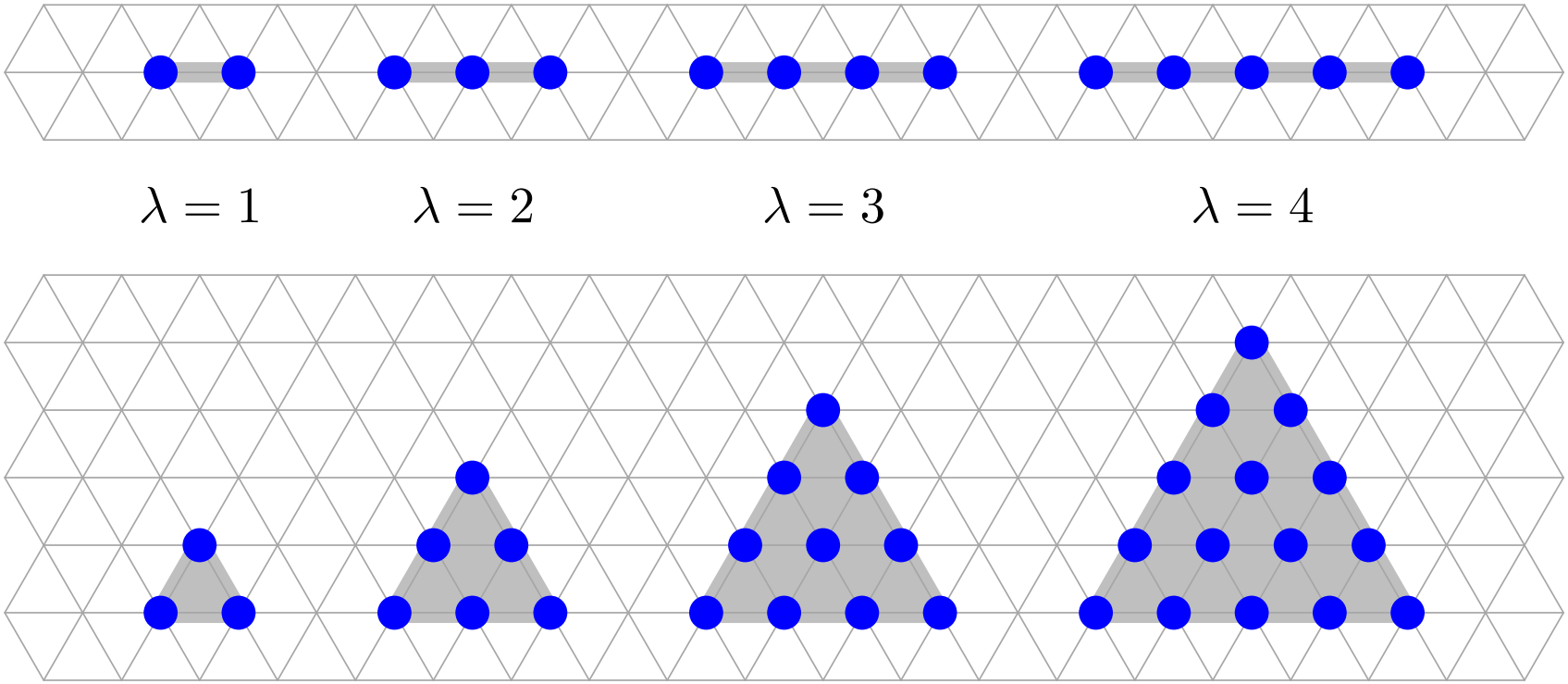
This protocol allows the system to form shapes that scale up like fractals (e.g., the Sierpinski triangle), as well as curved objects, which are better approximated as the number of particles increases.

Fractal and Curved Shapes



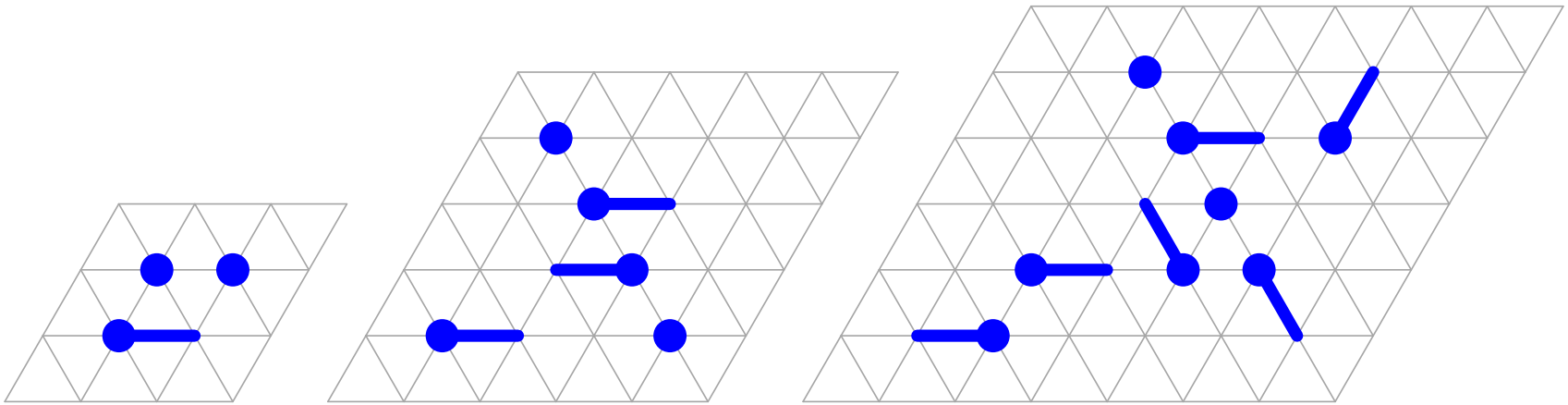
This protocol allows the system to form shapes that scale up like fractals (e.g., the Sierpinski triangle), as well as curved objects, which are better approximated as the number of particles increases.

Fractal and Curved Shapes



By contrast, the old protocol only allowed the formation of shapes that are made up of full triangles and segments.

General Computable Shapes



$n = 3$ $n = 5$ $n = 8$

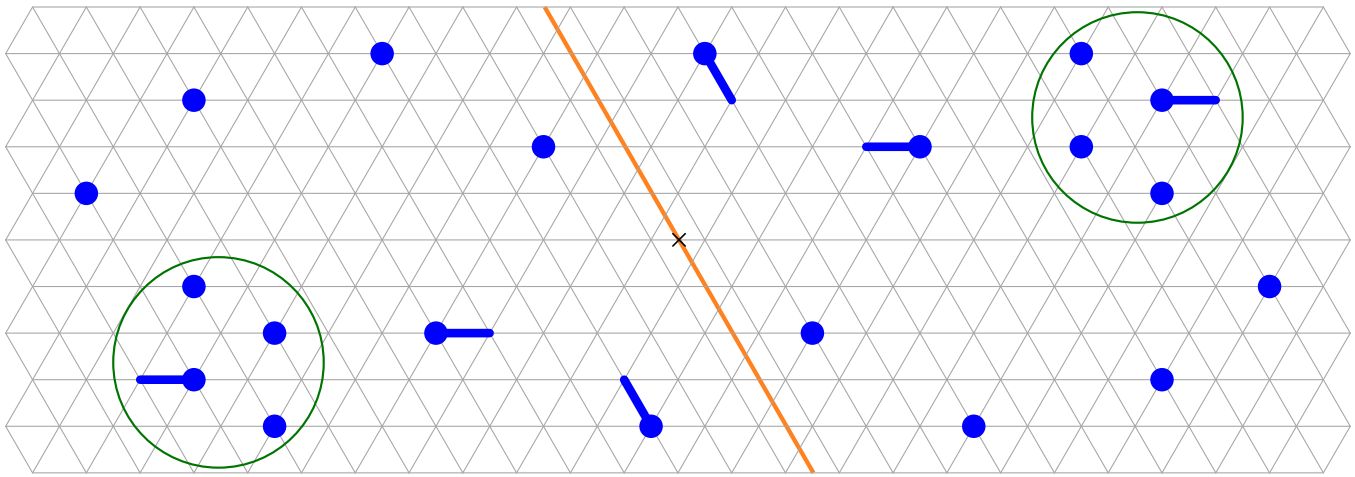
More generally, the new protocol only requires the existence of a computable function that, on input n, produces a configuration of n particles that forms a suitably scaled-up copy of the final shape.

Dismantling the Mobile RAMs

Also, each mobile RAM used to form the shape has to be dismantled when it has finished computing:

Assumption

For each scaled-up copy S_n of the final shape, there exists a configuration C_f of n particles that forms S_n (such that C_f is unbreakably k-symmetric if S_n has to be formed from an unbreakably k-symmetric initial configuration) and, for each symmetric component C'_f of C_f, there exists a ball of diameter independent of n that contains at least four particles of C'_f.



Conclusion

Theorem

Under the previous Assumption, any Turing-computable shape is formable from any simply connected initial configuration.

Only very sparse shapes fail to satisfy the Assumption.
In particular, connected shapes abundantly satisfy it.

Corollary

A necessary and sufficient condition for a connected Turing-computable shape to be formable from a simply connected initial configuration is that, if the initial configuration is unbreakably k-symmetric, then also the corresponding scaled-up copy of the shape is unbreakably k-symmetric.