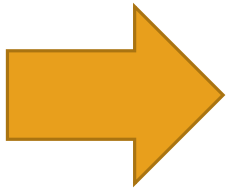
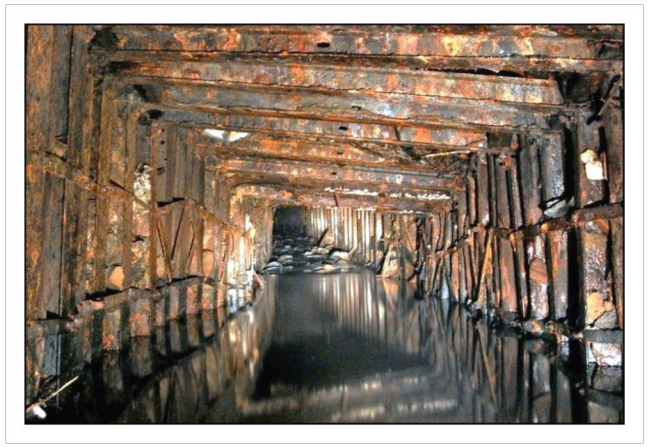
Self-Organizing Particle Systems: an Algorithmic Approach to Programmable Matter

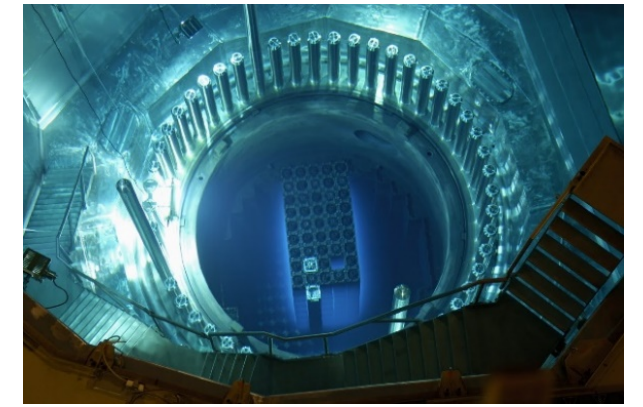
JOSHUA J. DAYMUDE – ARIZONA STATE UNIVERSITY

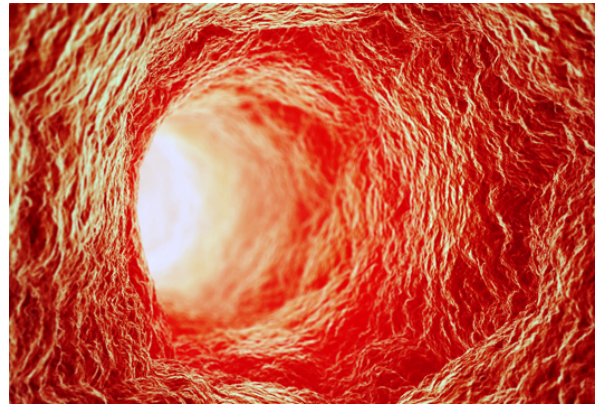
WSSR 2018 – NOVEMBER 4, 2018

Inspirations & Applications

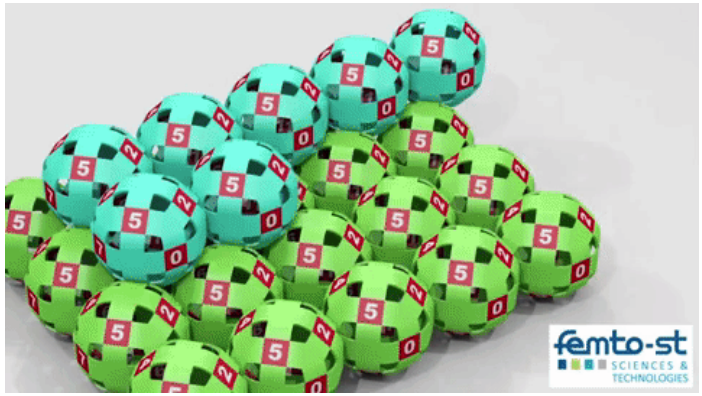




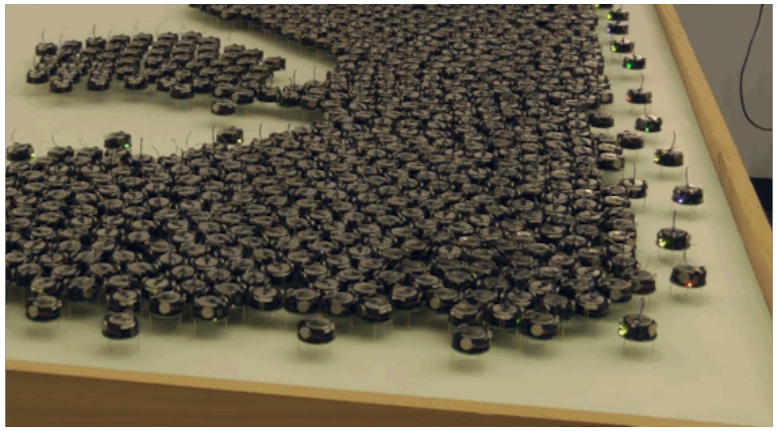


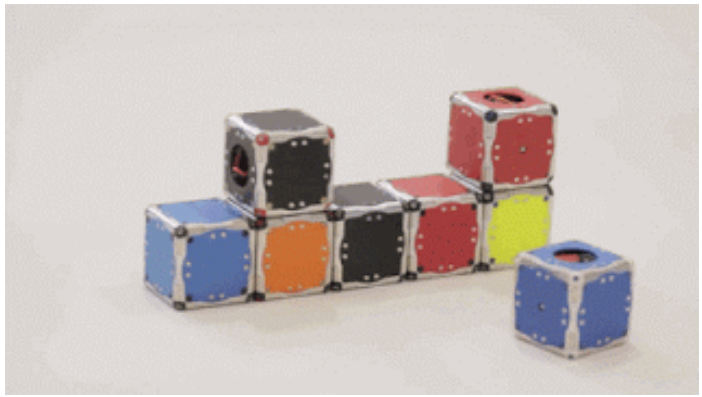


Current Programmable Matter

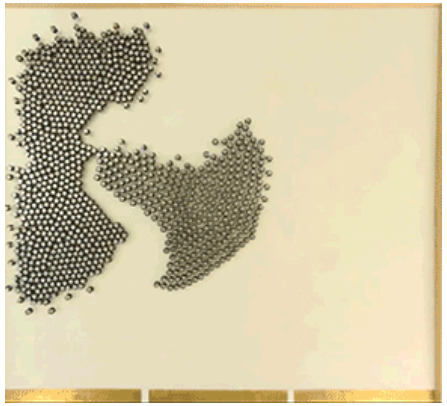


PB 2016: "Design of Quasi-Spherical Modules for Building Programmable Matter"





RGR 2013: "M-blocks: Momentum driven, magnetic modular robots"

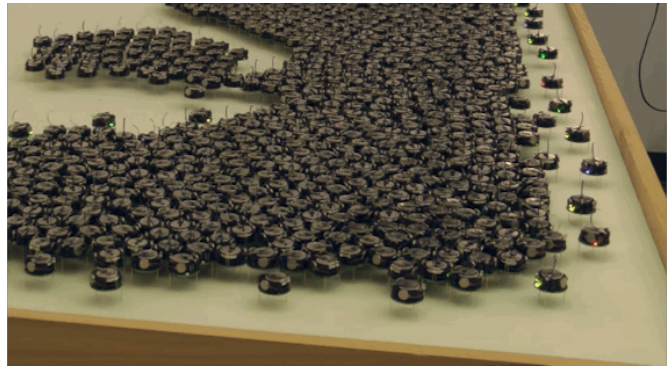


RCN 2014: "Programmable self-assembly in a thousand-robot swarm"

Current Programmable Matter

Programmable matter systems can be **passive** or **active**:

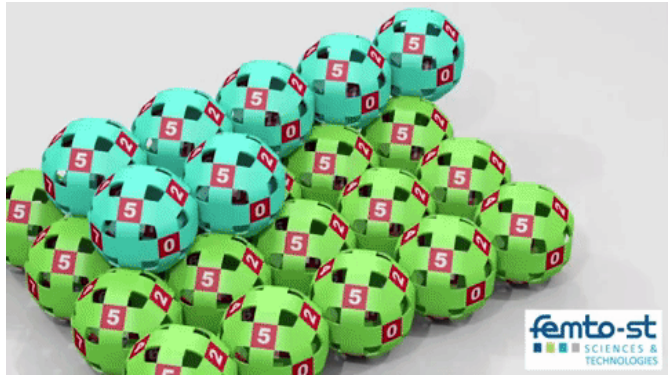
- **Passive**: Little/no control over decisions & movements, depends on the environment.
- **Active**: Can control actions & movements to solve problems.



RCN 2014: "Programmable self-assembly in a thousand-robot swarm"

"Self-Organizing Particle Systems" (SOPS):

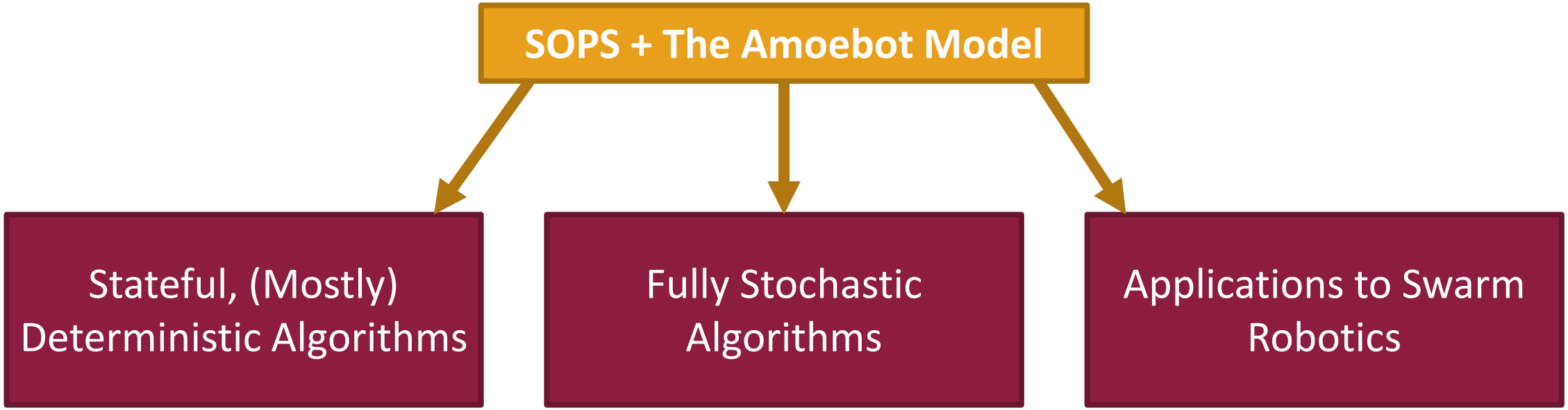
- Abstraction of **active** programmable matter.
- Each "particle" is a simple unit that can move and compute.
- Using **distributed algorithms**, limited particles coordinate to achieve sophisticated behavior.



PB 2016: "Design of Quasi-Spherical Modules for Building Programmable Matter"

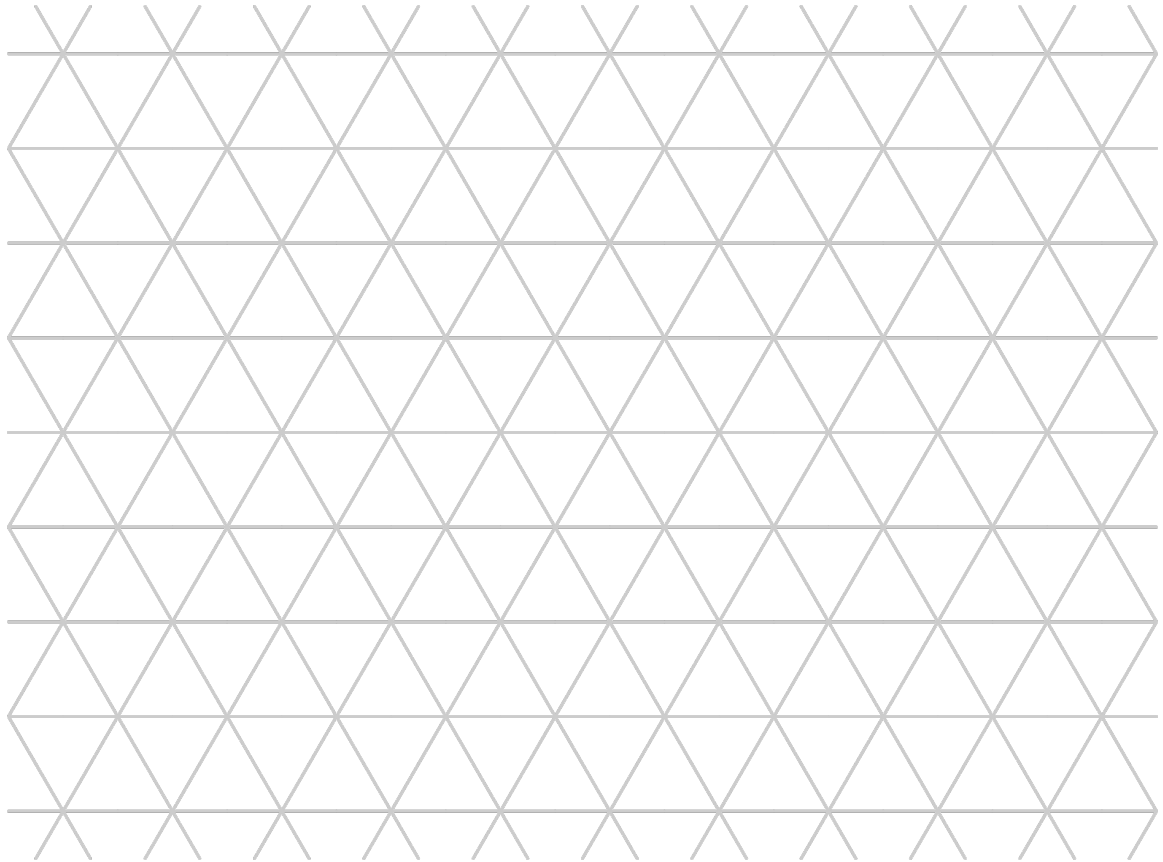
The Big Picture

What complex, collective behaviors are achievable by systems of simple, restricted programmable particles?



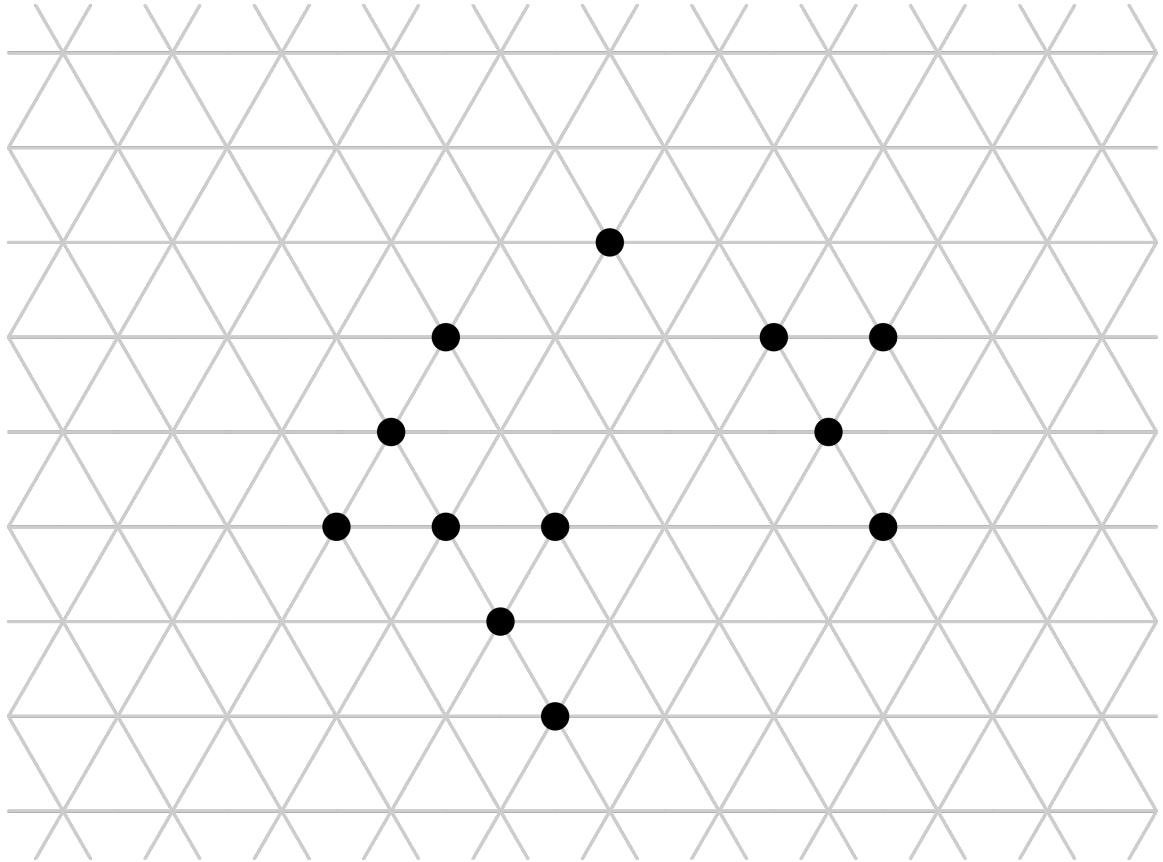
The (Geometric) Amoebot Model

- Space is modeled as the triangular lattice.



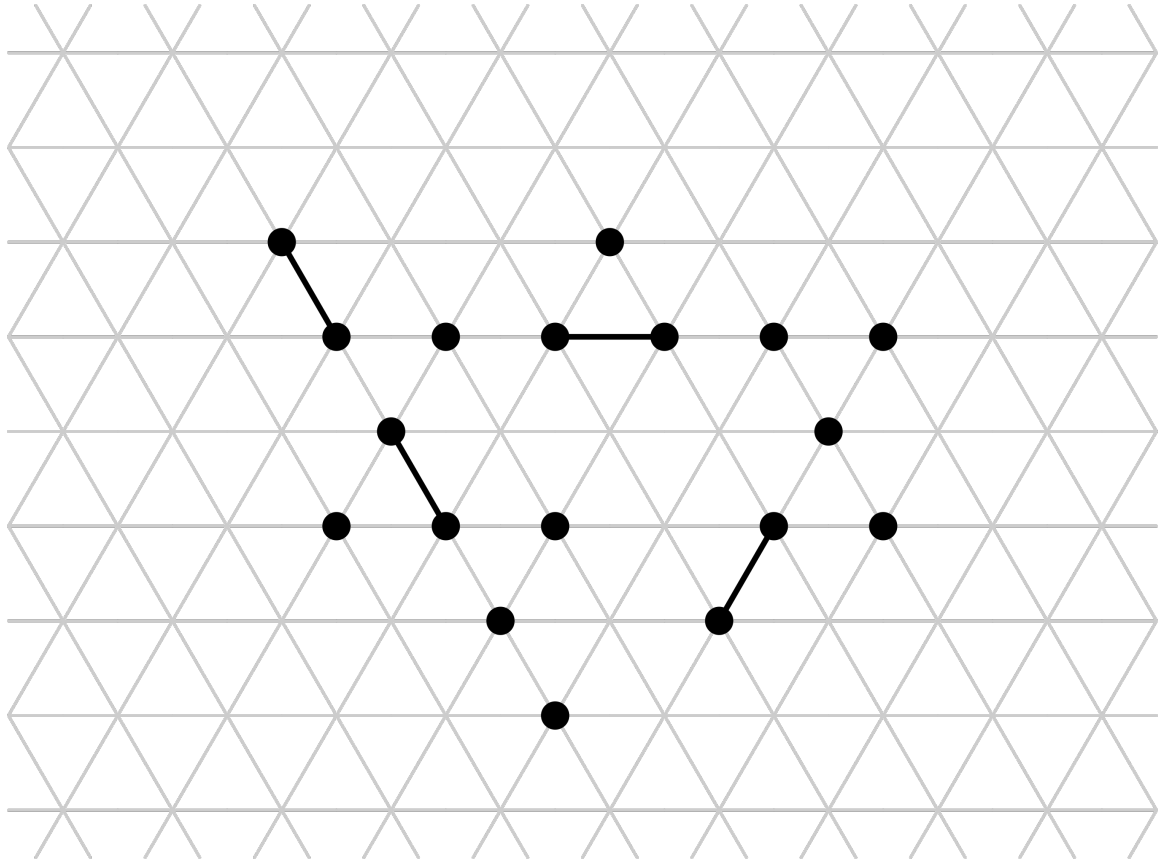
The (Geometric) Amoebot Model

- Space is modeled as the triangular lattice.
- Particles can occupy one node (contracted)...



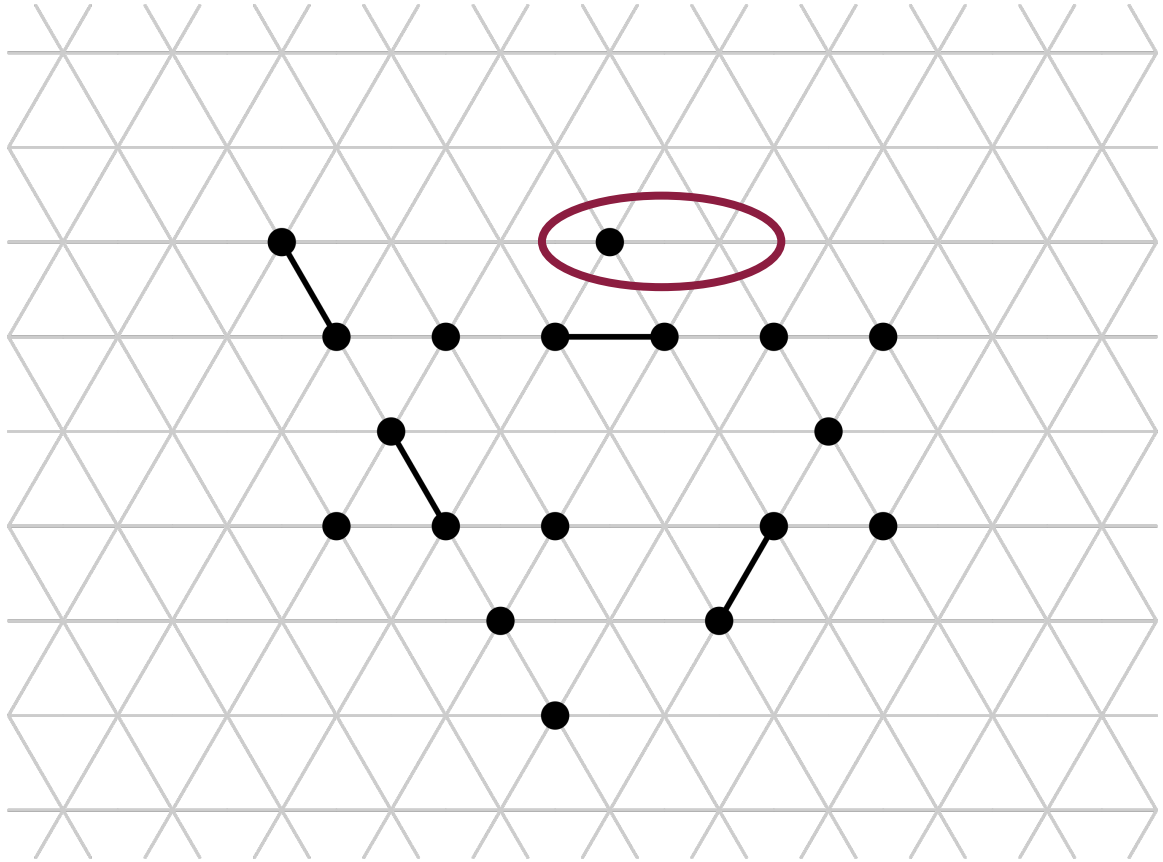
The (Geometric) Amoebot Model

- Space is modeled as the triangular lattice.
- Particles can occupy one node (contracted) or two adjacent nodes (expanded).



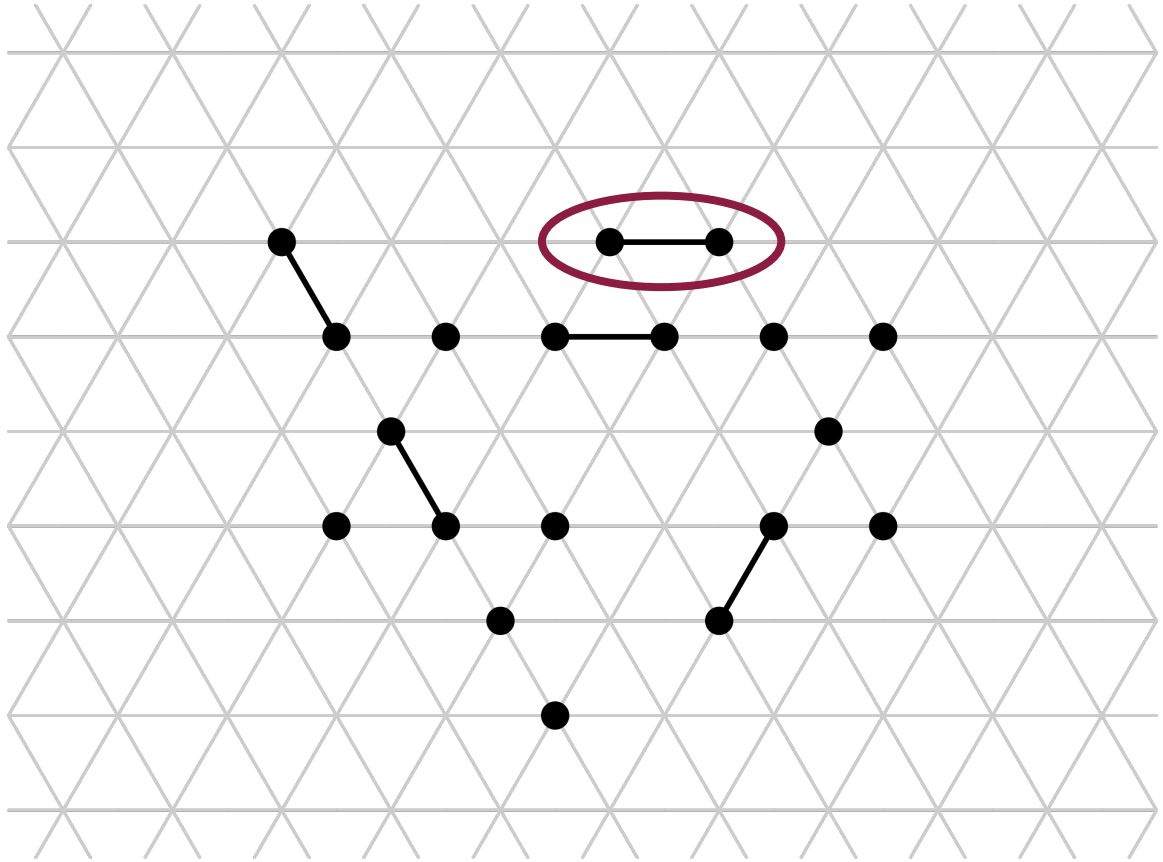
The (Geometric) Amoebot Model

- Space is modeled as the triangular lattice.
- Particles can occupy one node (contracted) or two adjacent nodes (expanded).
- Particles move by expanding and contracting.



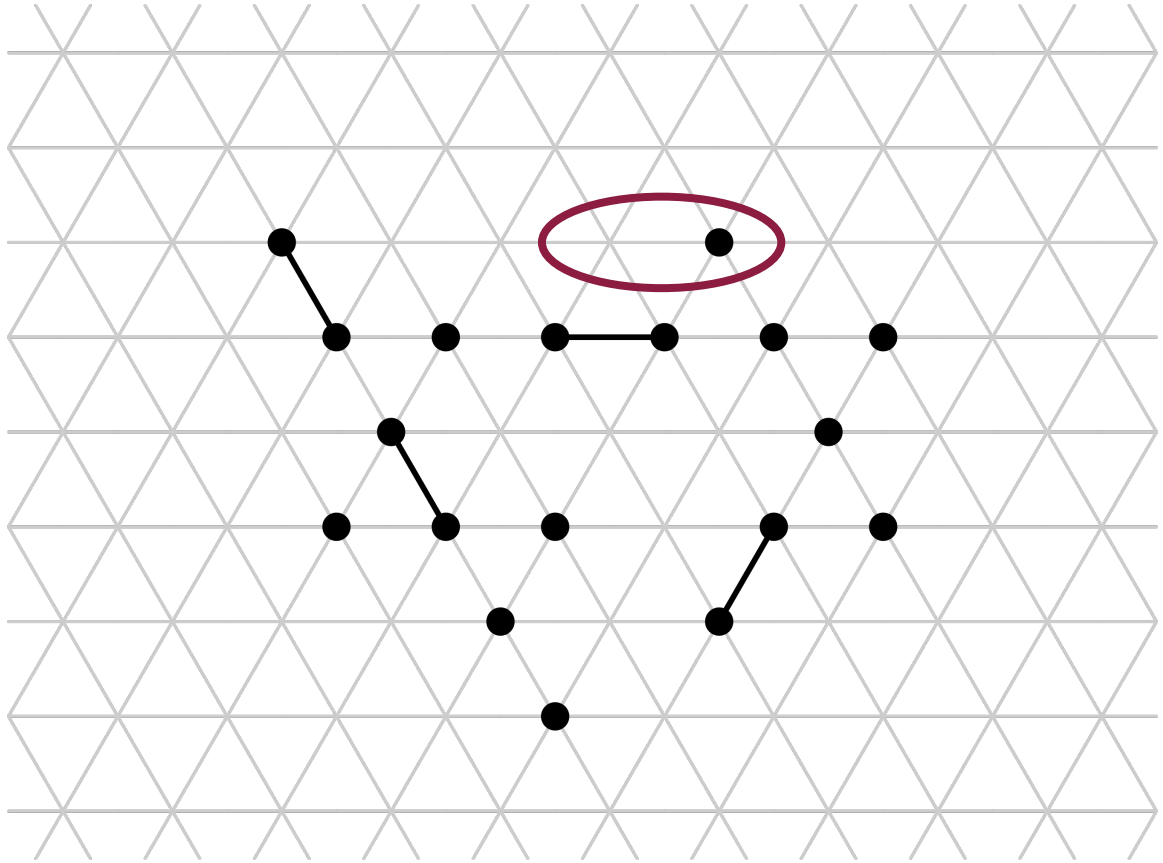
The (Geometric) Amoebot Model

- Space is modeled as the triangular lattice.
- Particles can occupy one node (contracted) or two adjacent nodes (expanded).
- Particles move by expanding and contracting.



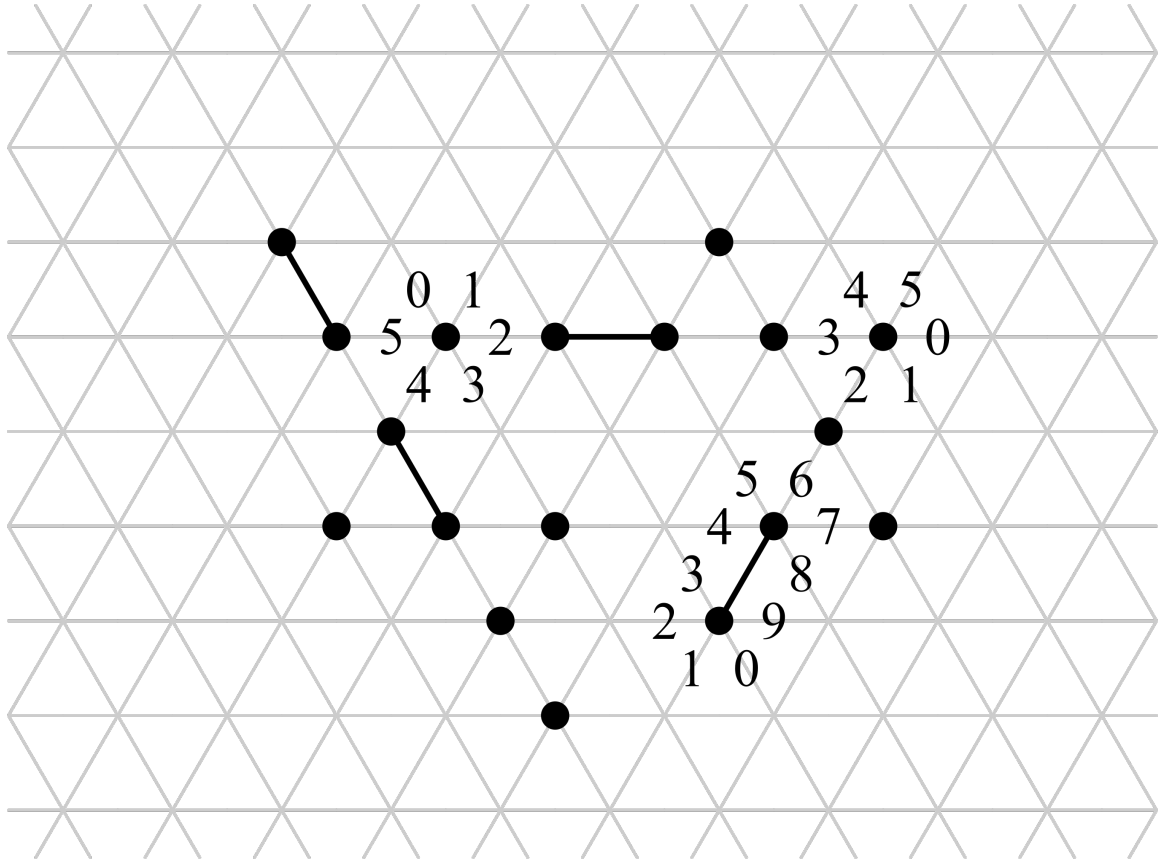
The (Geometric) Amoebot Model

- Space is modeled as the triangular lattice.
- Particles can occupy one node (contracted) or two adjacent nodes (expanded).
- Particles move by expanding and contracting.



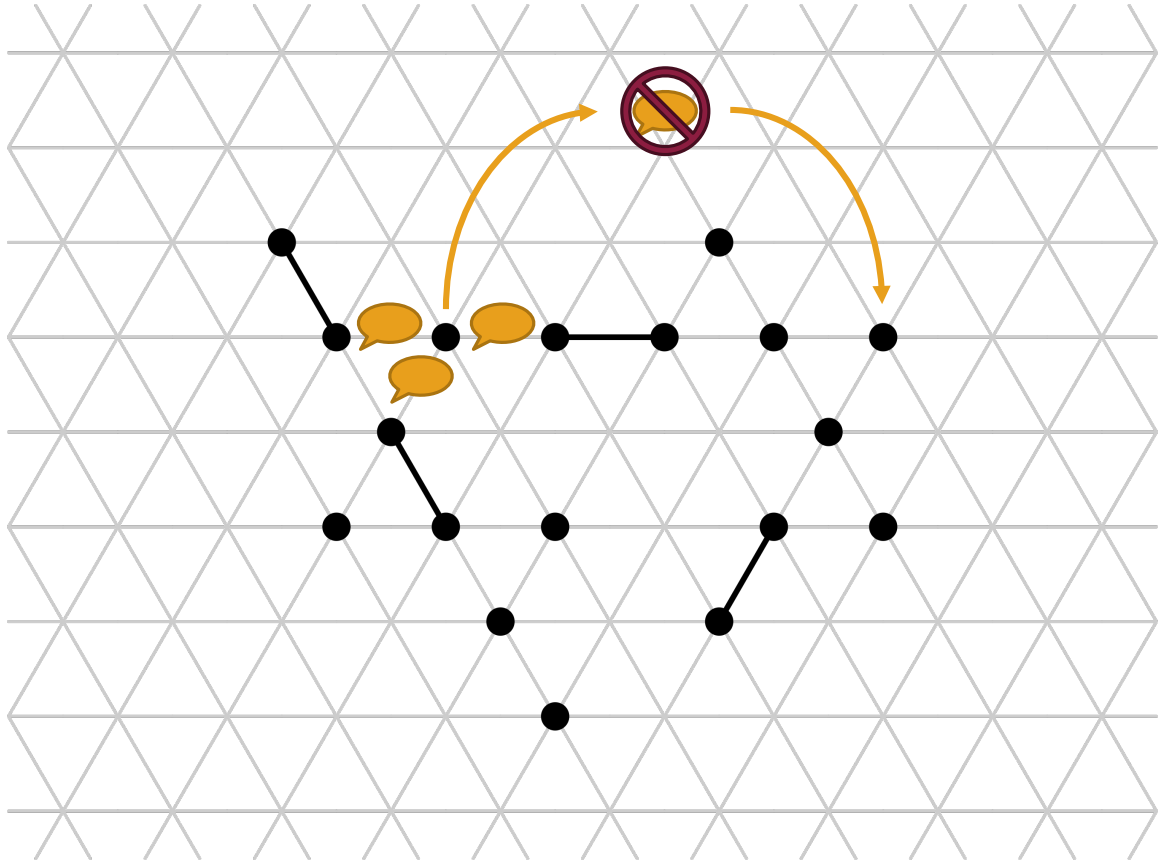
The (Geometric) Amoebot Model

- Space is modeled as the triangular lattice.
- Particles can occupy one node (contracted) or two adjacent nodes (expanded).
- Particles move by expanding and contracting.
- Particles do not have a global compass, but locally label their neighbors in clockwise order.



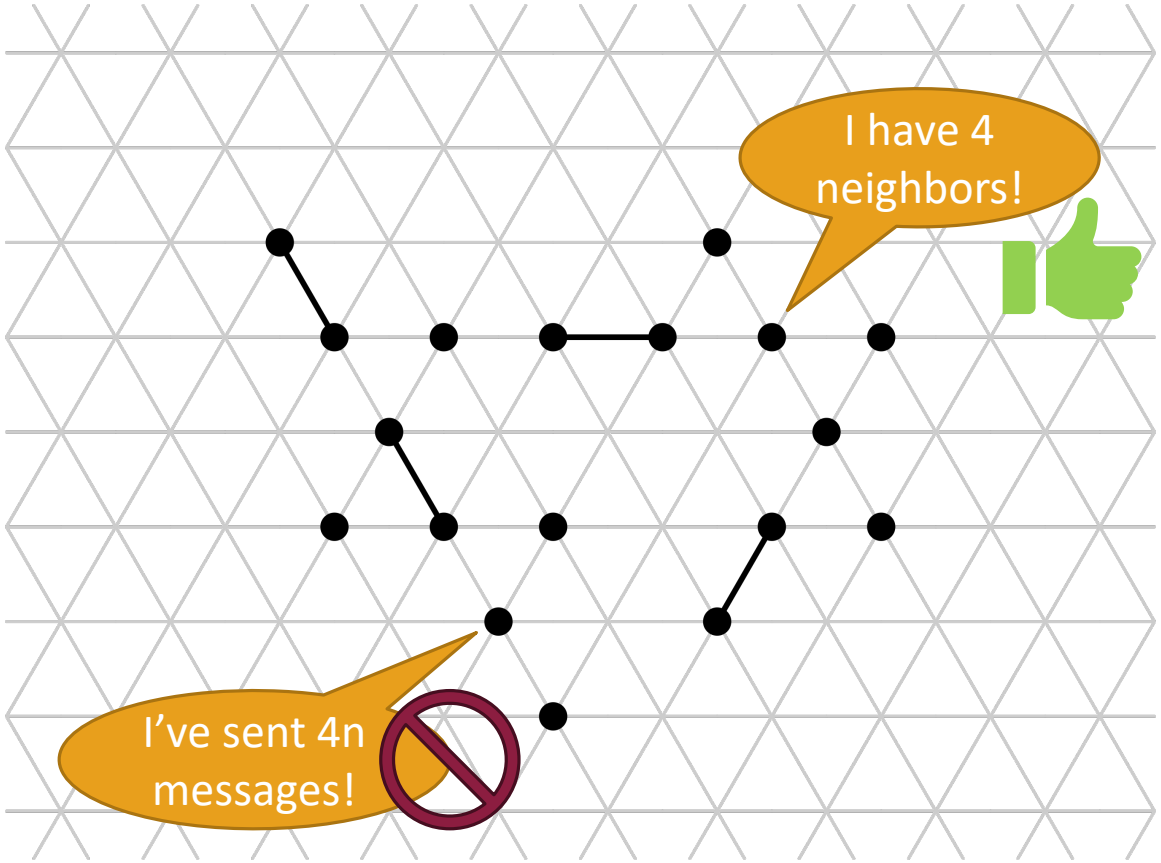
The (Geometric) Amoebot Model

- Space is modeled as the triangular lattice.
- Particles can occupy one node (contracted) or two adjacent nodes (expanded).
- Particles move by expanding and contracting.
- Particles do not have a global compass, but locally label their neighbors in clockwise order.
- Particles can communicate only with their neighbors.



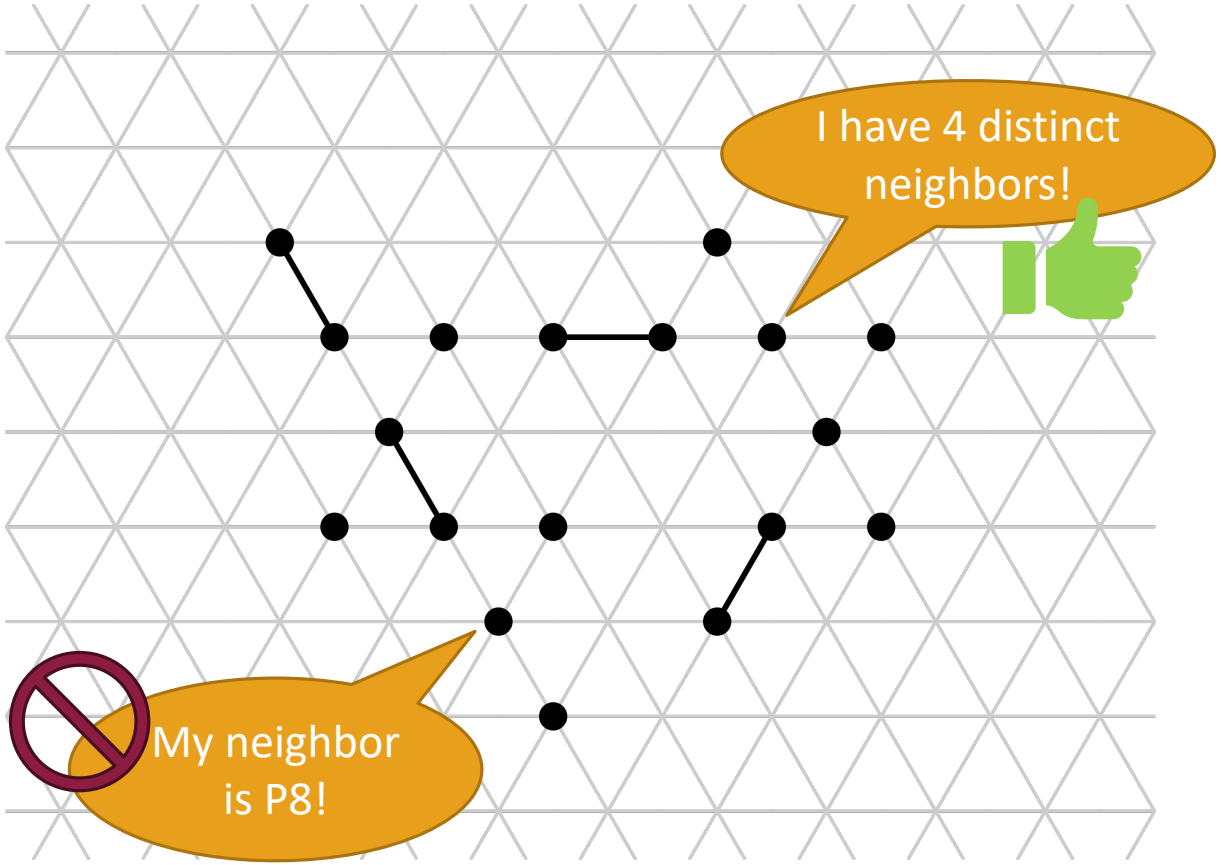
The (Geometric) Amoebot Model

- A particle only has constant-size memory.



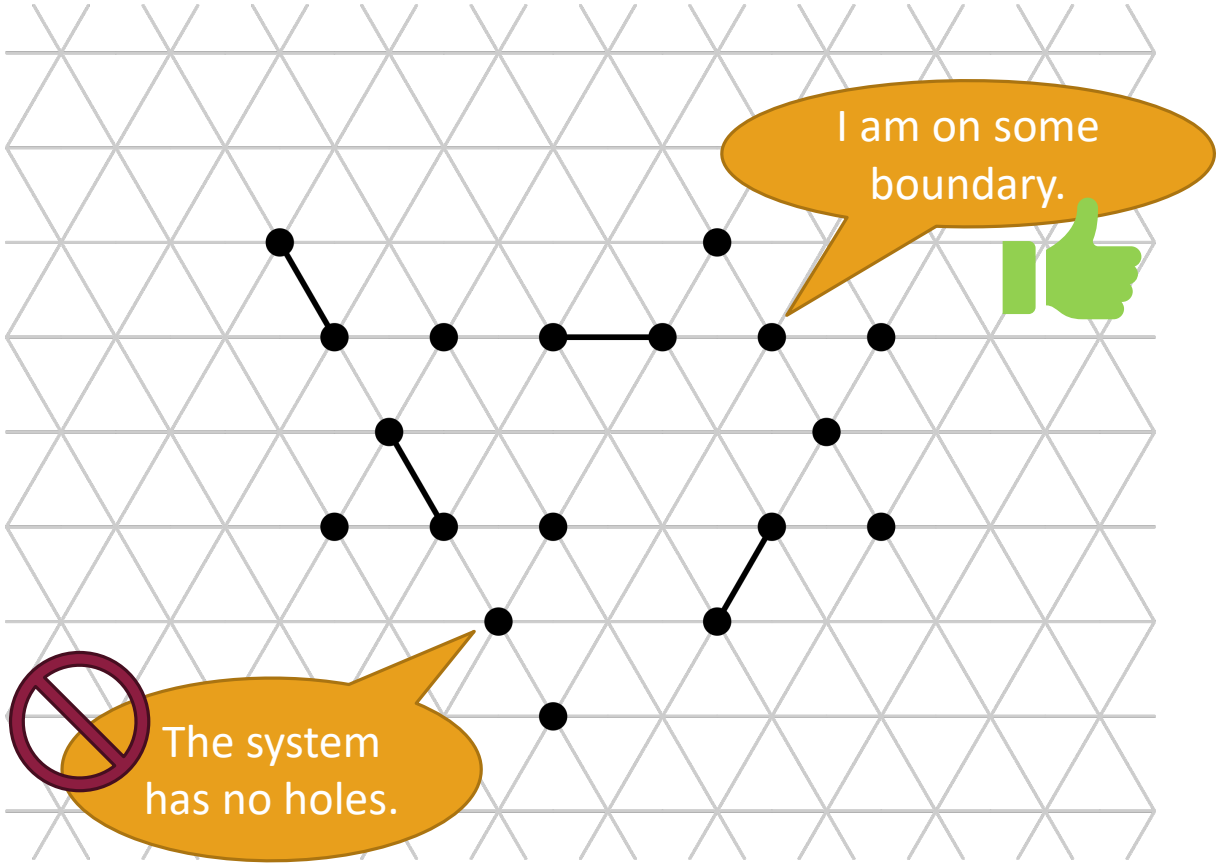
The (Geometric) Amoebot Model

- A particle only has constant-size memory.
- No unique identifiers.



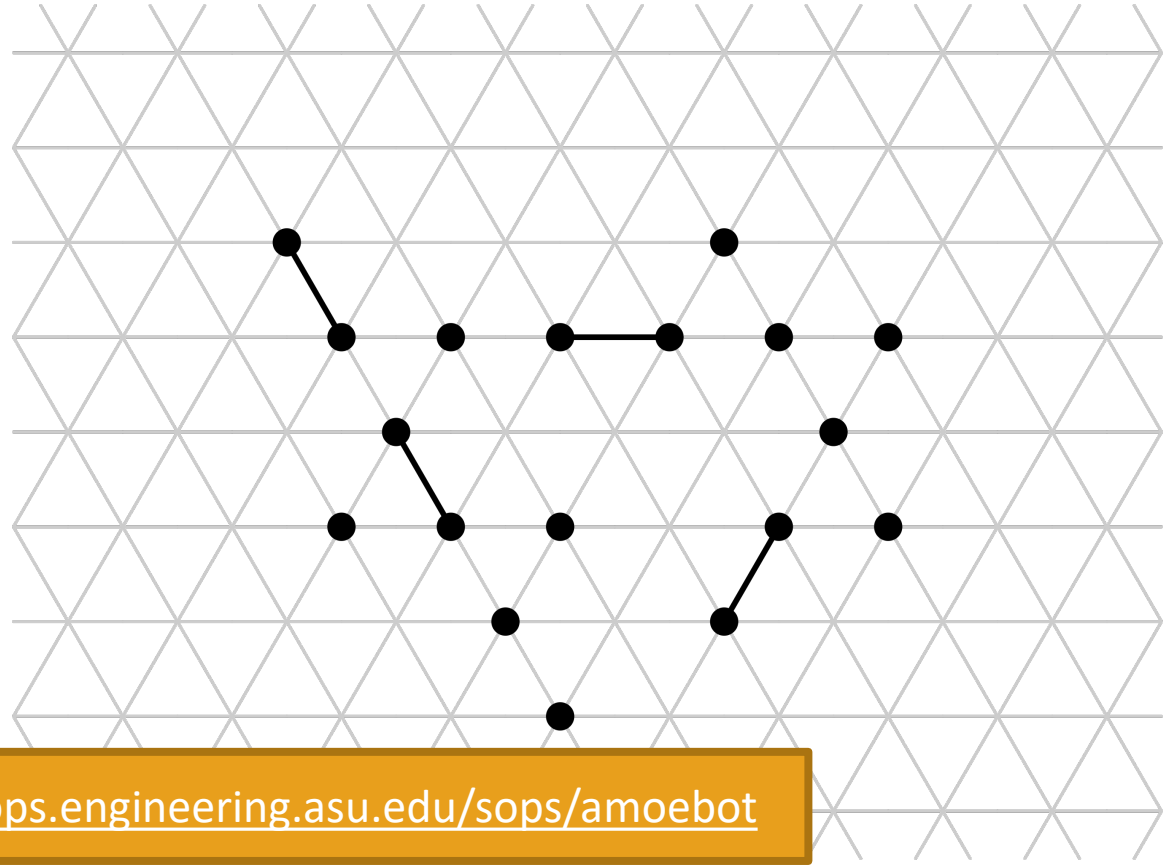
The (Geometric) Amoebot Model

- A particle only has constant-size memory.
- No unique identifiers.
- No global information.



The (Geometric) Amoebot Model

- A particle only has constant-size memory.
- No unique identifiers.
- No global information.
- Asynchronous model of time: one atomic action may include finite computation and communication and at most one movement.



Read more at: sops.engineering.asu.edu/sops/amoebot

Stateful, (Mostly) Deterministic Algorithms

Arizona State University

UNIVERSITÄT PADERBORN

UNIVERSITY of HOUSTON


Andréa W. Richa


Zahra Derakhshandeh


Christian Scheideler


Kristian Hinnenthal


Thim Strothmann

Irina Kostitsyna


Robert Gmyr

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

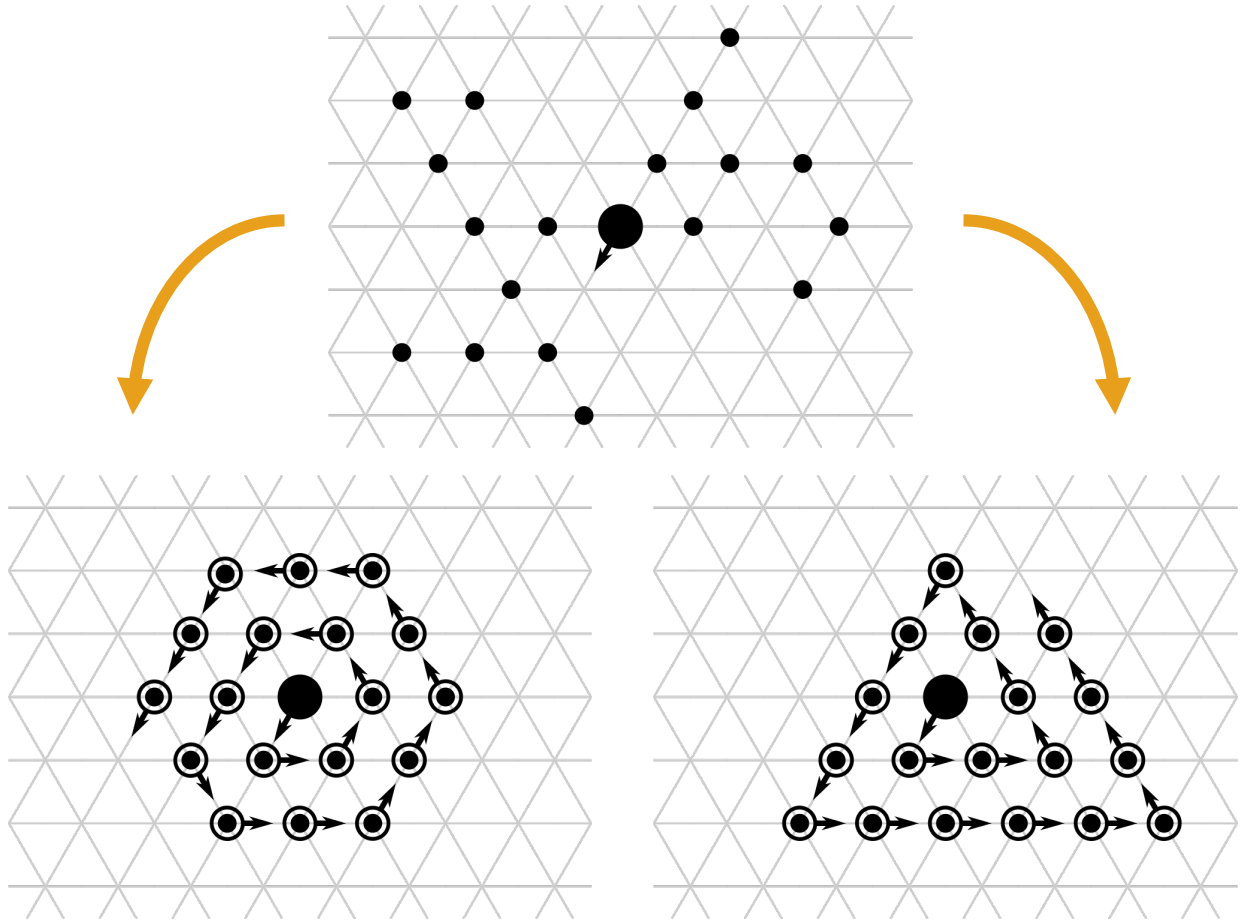
Stateful, (Mostly) Deterministic Algorithms

At a glance:

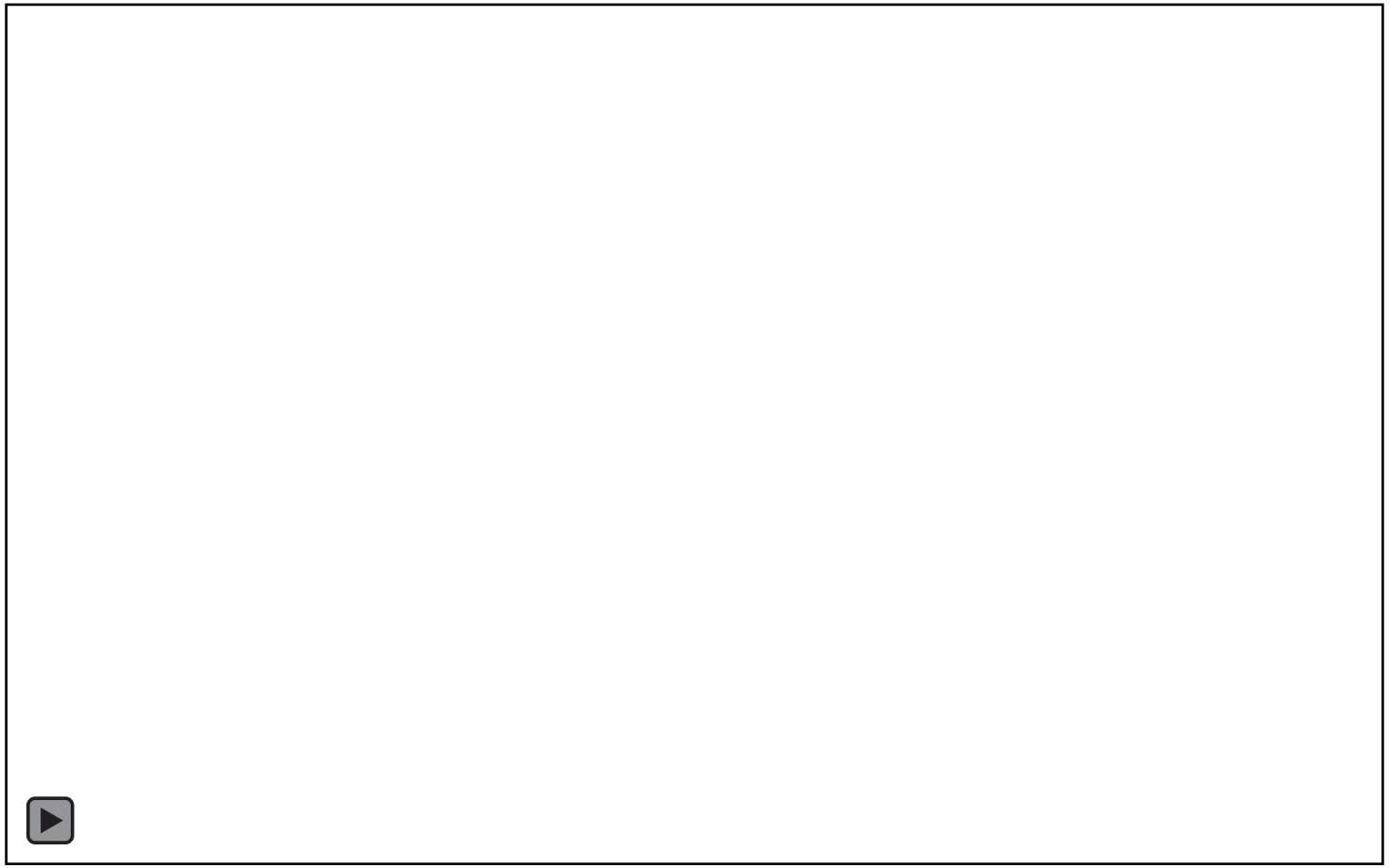
- Particles running these algorithms utilize their constant-size memories to store state (variables, tokens, etc.)
- Particles running these algorithms coordinate through communication.
- In these algorithms, particle actions/movements are based on a combination of their own state and the states of their neighbors.
- These algorithms come with provable correctness and runtime guarantees.
- These algorithms, to date, are not even resilient to a single particle crash failure (with one important exception: DFPSV 2018: "Line Recovery by Programmable Particles").

Algorithm 1: Basic Shape Formation

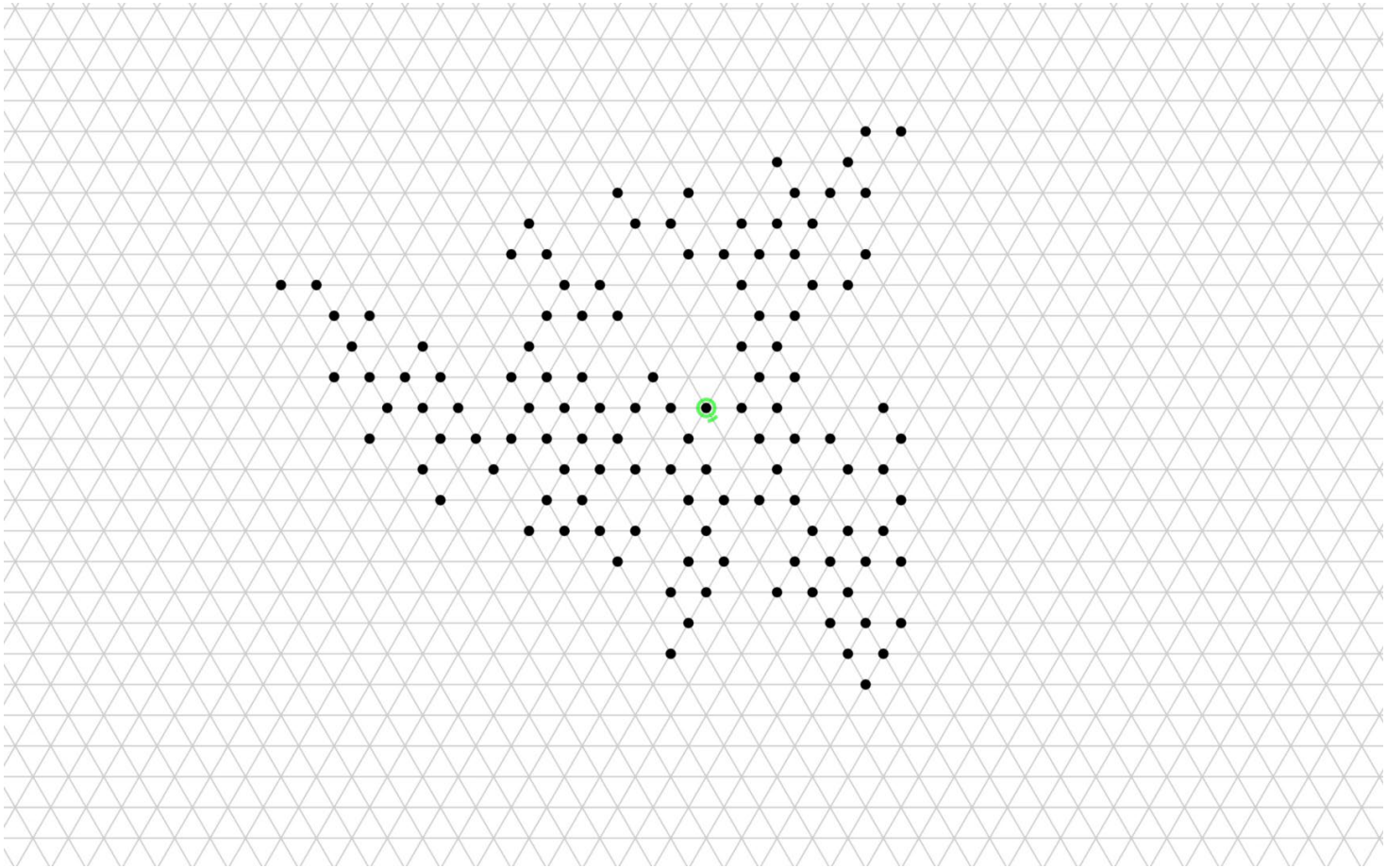
- **Problem:** Transform any connected initial configuration of contracted particles into a line, regular hexagon, or regular triangle.
- **Assumptions:** Given a unique leader (seed) particle.
- **Main Idea:** Grow the final structure particle-by-particle, starting at the seed.
- **Correctness:** Guaranteed.
- **Runtime:** Requires $\mathbf{O}(n)$ asynchronous rounds in the worst case, and matches the lower bound for the worst case amount of work: $\mathbf{\Omega}(n^2)$.



Algorithm 1: Basic Shape Formation

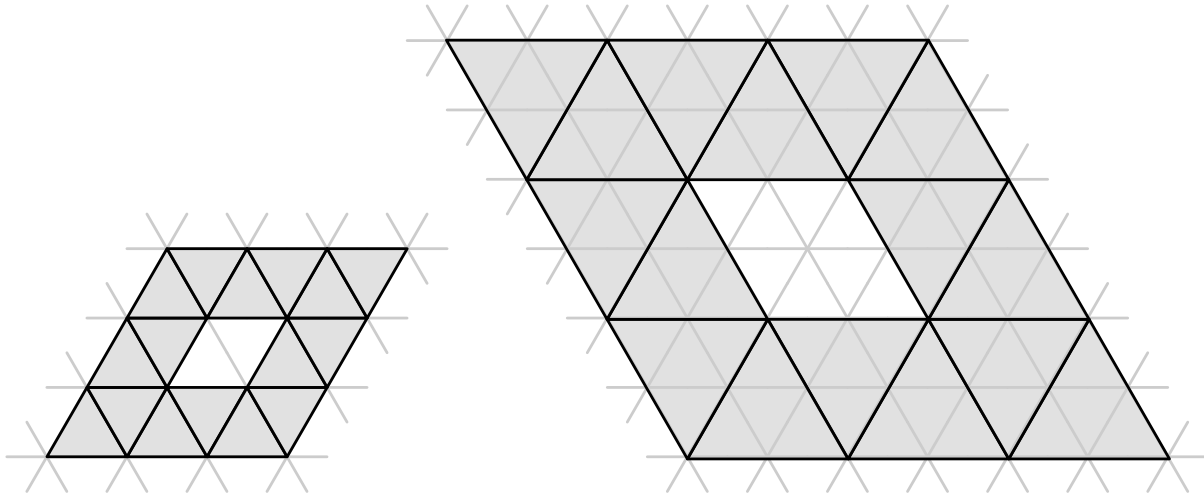


Algorithm 1: Basic Shape Formation



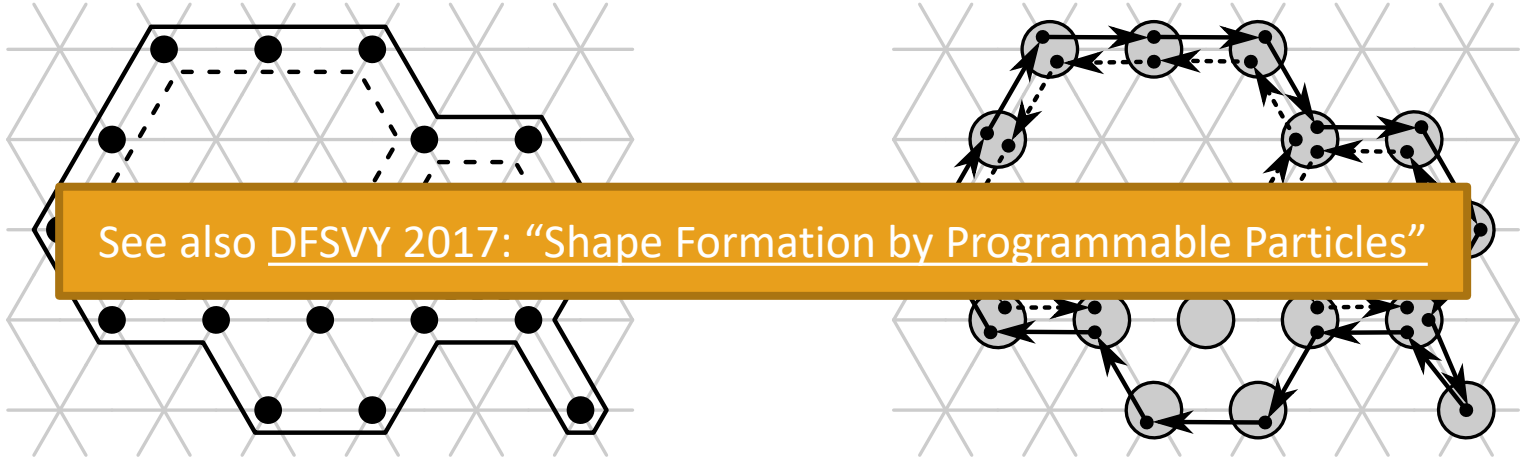
Algorithm 2: General Shape Formation

- **Problem:** Transform initial triangle of contracted particles into a target sequentially constructible shape S.
- **Assumptions:** Given a unique leader (seed) particle. Particles know constant-size representation of S.
- **Main Idea:** Organize the system into triangles of particles, and move these triangles to their place in the final shape.
- **Correctness:** Guaranteed.
- **Runtime:** Forms any "sequentially constructible" shape in $\mathbf{O}(n^{1/2})$ asynchronous rounds, which matches the lower bound for any local-control algorithm.



Algorithm 3: Leader Election

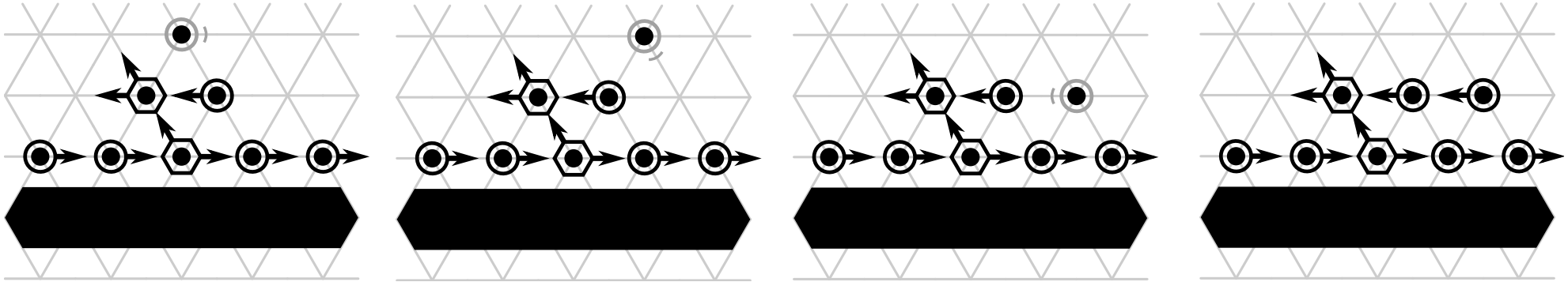
- **Problem:** Given an connected system of contracted particles, a particle must eventually uniquely and irreversibly declare itself the leader.
- **Main Idea:** Candidates compete using distributed "identifiers". The candidate on the unique outer boundary with the highest identifier wins.
- **Correctness:** With high probability.
- **Runtime:** Elects a leader in **O**(length of the outer boundary), with high probability.



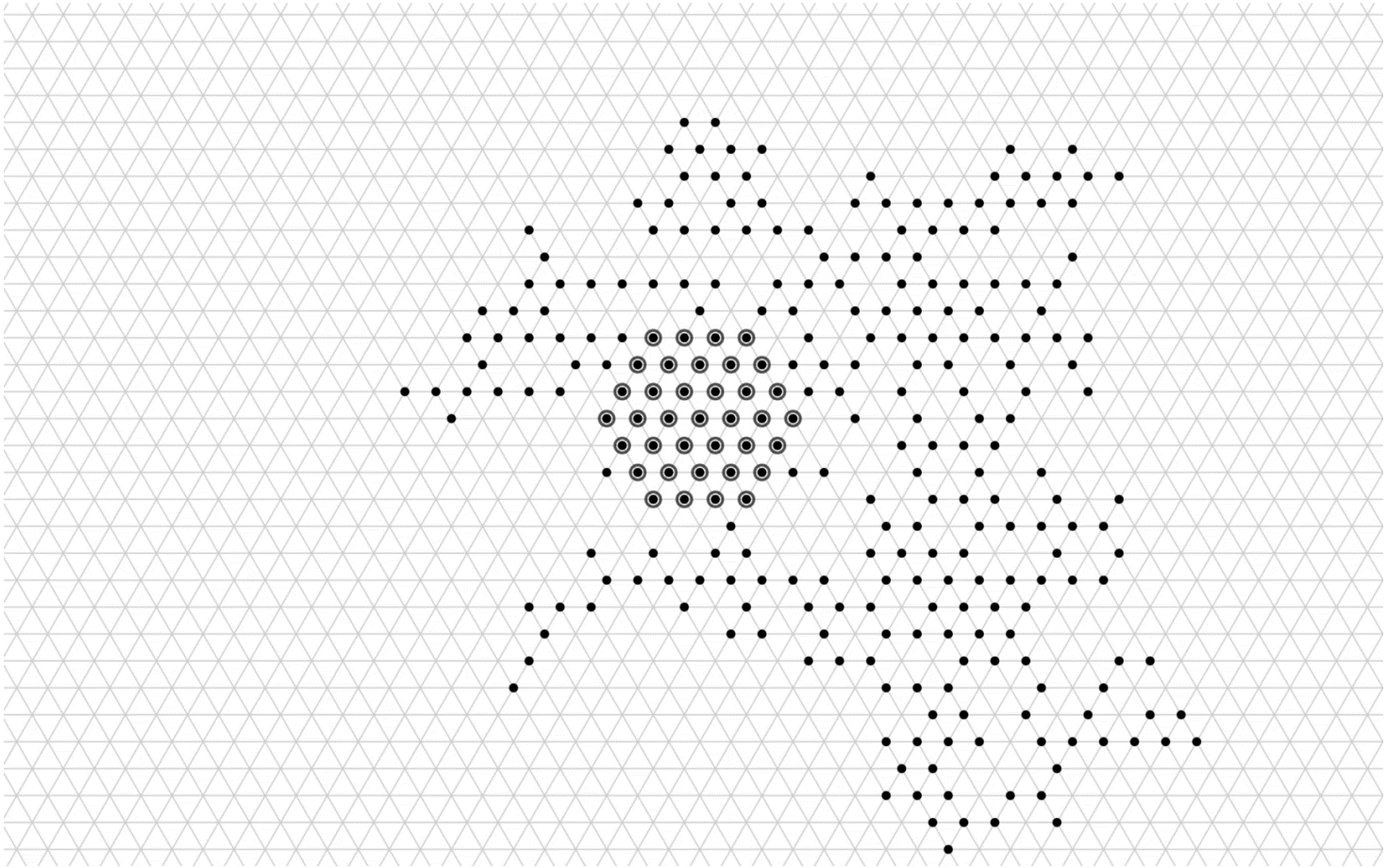
See also DFSVY 2017: "Shape Formation by Programmable Particles"

Algorithm 4: Object Coating

- **Problem:** Use a particle system to evenly coat a static object as evenly as possible.
- **Assumptions:** The object does not contain narrow tunnels.
- **Main Idea:** Particles coat the first layer by following the object and attempt to elect a leader. If elected, this leader marks the start/end of the higher layers.
- **Correctness:** With high probability (leader election).
- **Runtime:** Requires **O**(n) asynchronous rounds (w.h.p.) in the worst case, which matches the lower bound for any local-control algorithm.

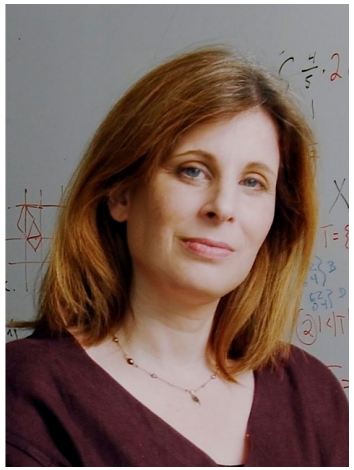


Algorithm 4: Object Coating



Fully Stochastic Algorithms

Andréa W. Richa



Dana Randall

Cem Gökmen

Sarah Cannon

Marta Andrés Arroyo

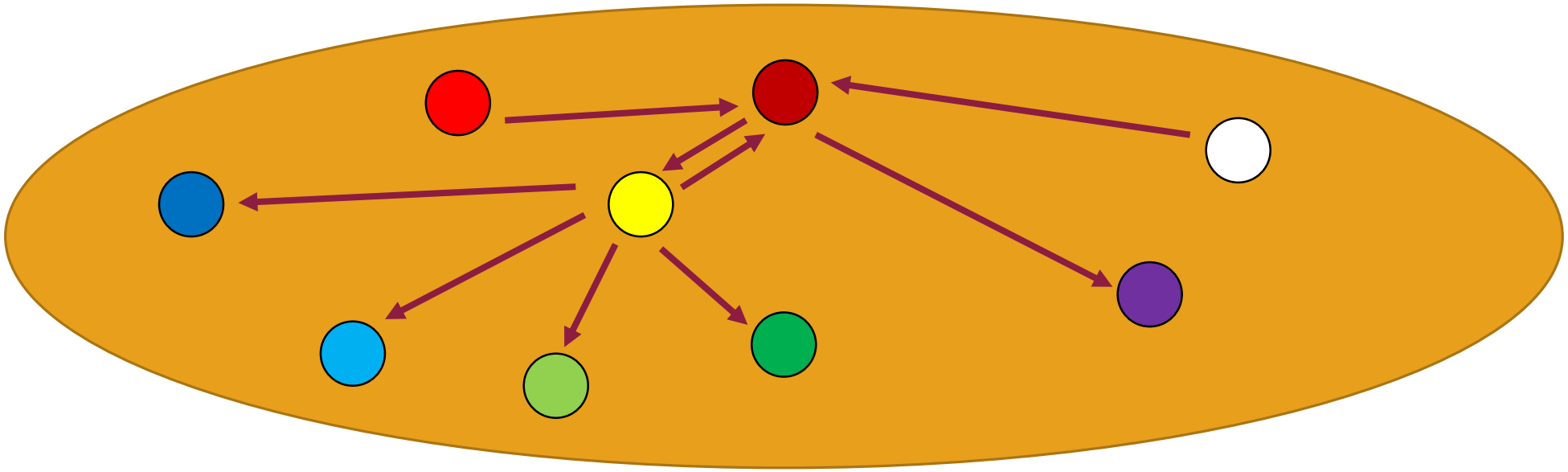
Fully Stochastic Algorithms

At a glance:

- These algorithms are Markov chain processes that are directly translated into distributed algorithms.
- In these algorithms, particle actions/movements are all chosen at random with probabilities biasing them towards the objective.
- Particles running these algorithms use almost no memory (one or two bits per particle).
- Particles running these algorithms hardly ever communicate.
- These algorithms are guaranteed to converge to the desired behavior, but runtime bounds have only been obtained in simulation.

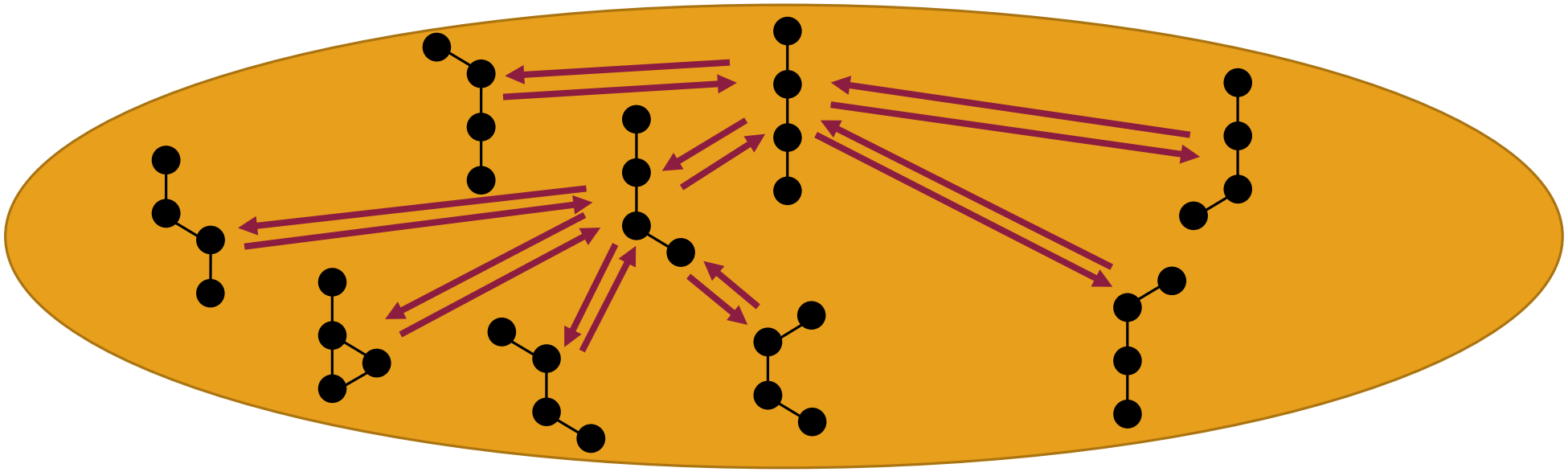
Markov Chains

- A Markov chain is a **memoryless**, **random** process that undergoes transitions between states in a state space.



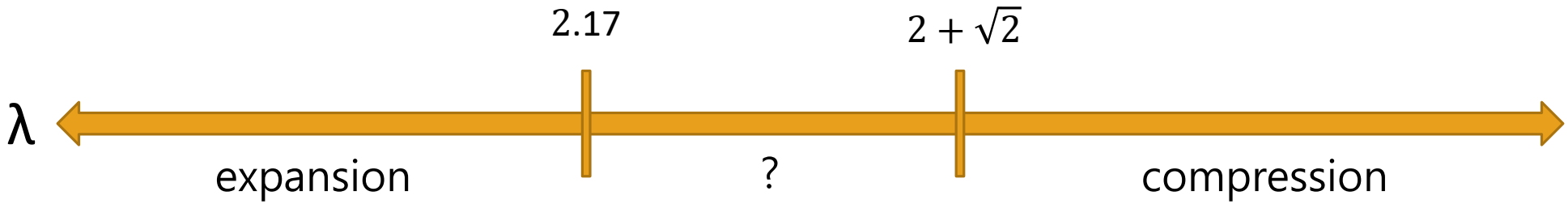
Markov Chains

- A Markov chain is a **memoryless**, **random** process that undergoes transitions between states in a state space.
- Our state space is all possible particle system **configurations**, and transitions between them are individual **particle moves**.



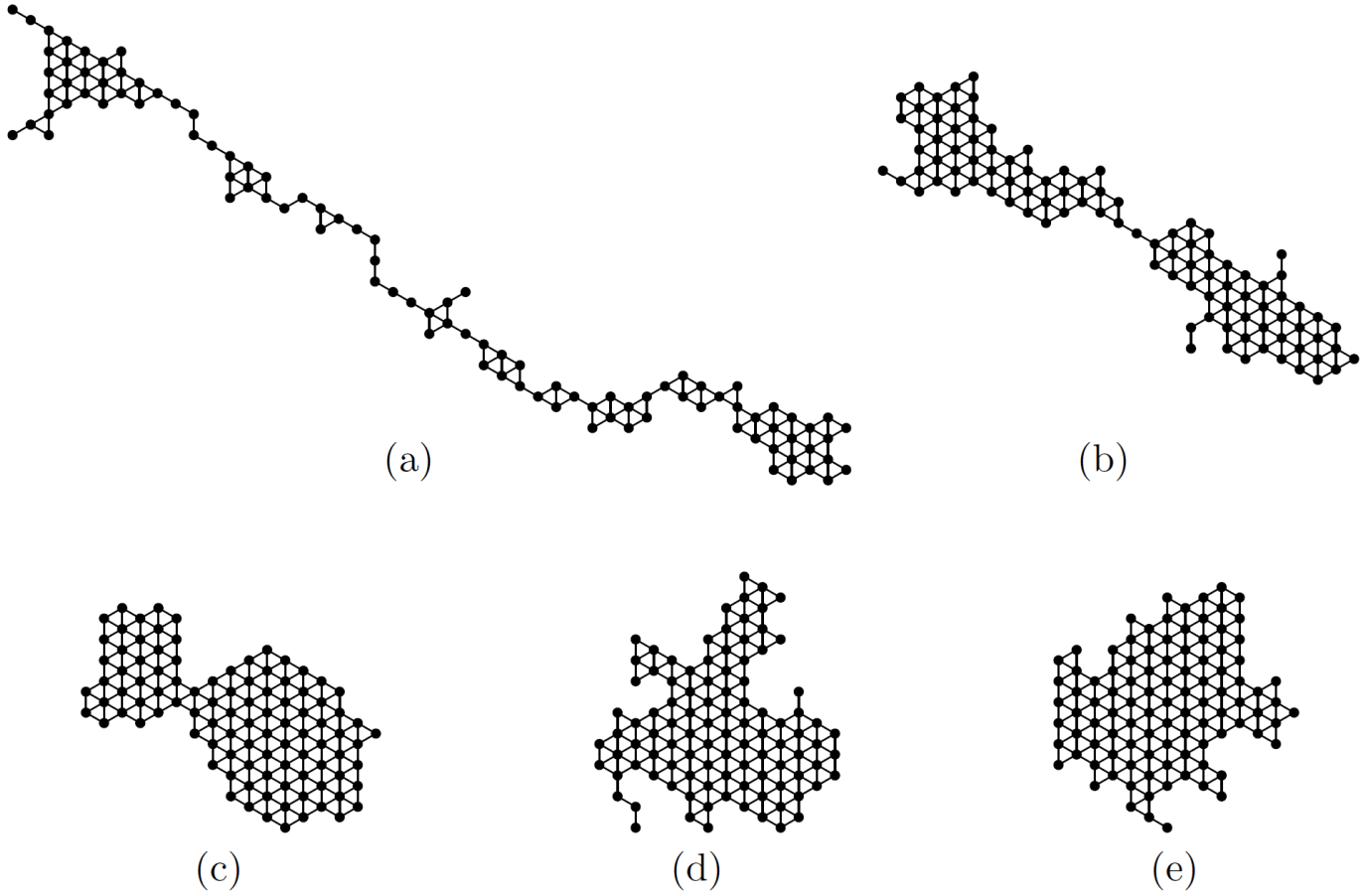
Algorithm 5: Compression (Expansion)

- **Problem:** Gather a particle system as tightly together as possible. (Or do the opposite.)
- **Main Idea:** Particles make moves that are biased towards increasing (or decreasing) their number of neighbors to achieve the global outcome.
- **Correctness:** With all but exponentially small probability.
- **Runtime:** Unknown. Simulations suggest $\mathbf{O}(n^3)$ asynchronous rounds.



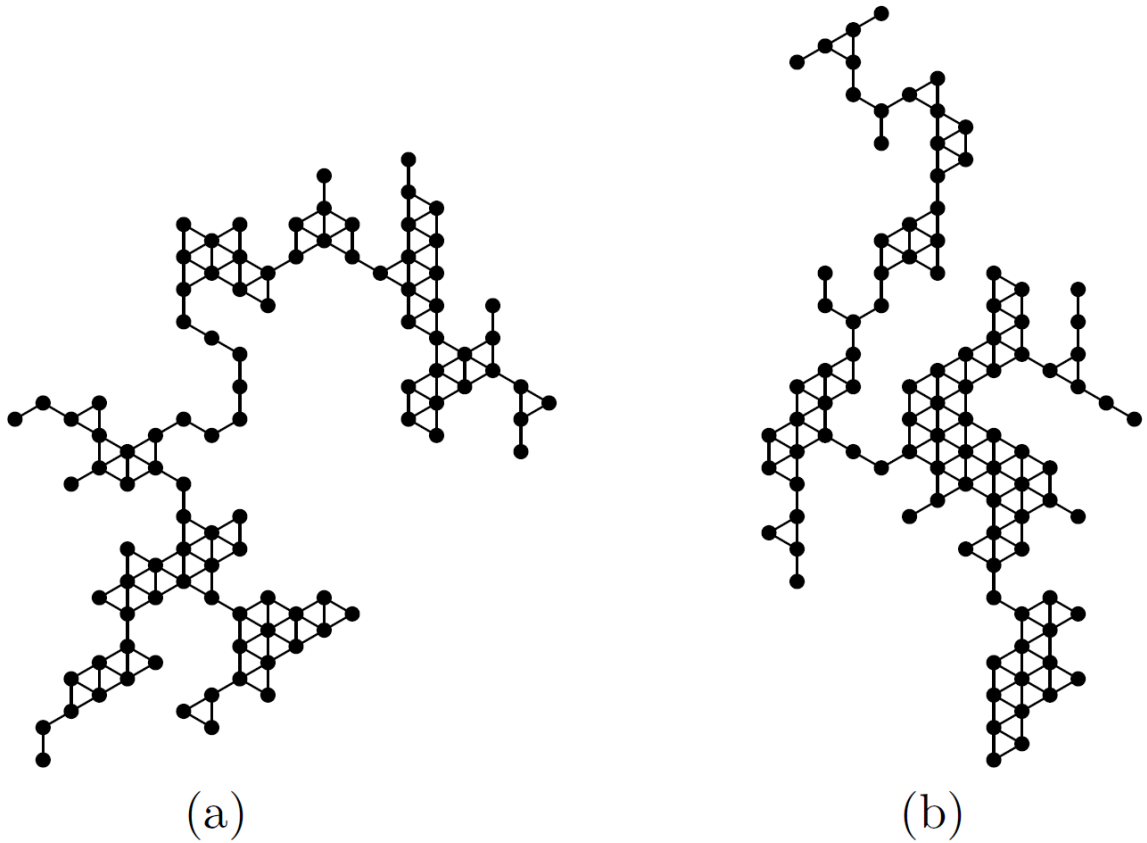
Algorithm 5: Compression (Expansion)

$\lambda = 4$



Algorithm 5: Compression (Expansion)

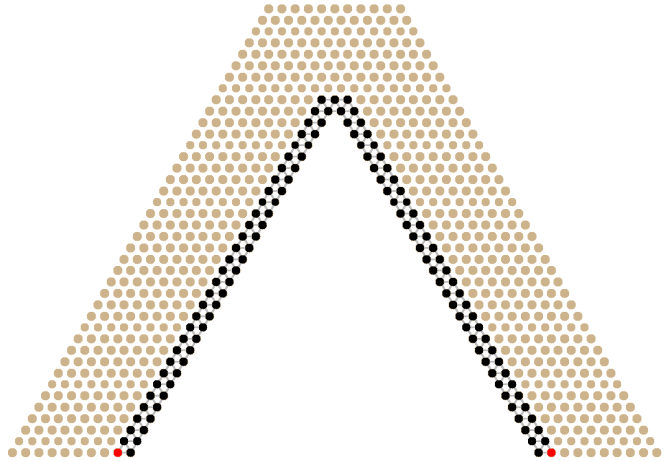
$\lambda = 2$



Algorithm 6: Shortcut Bridging

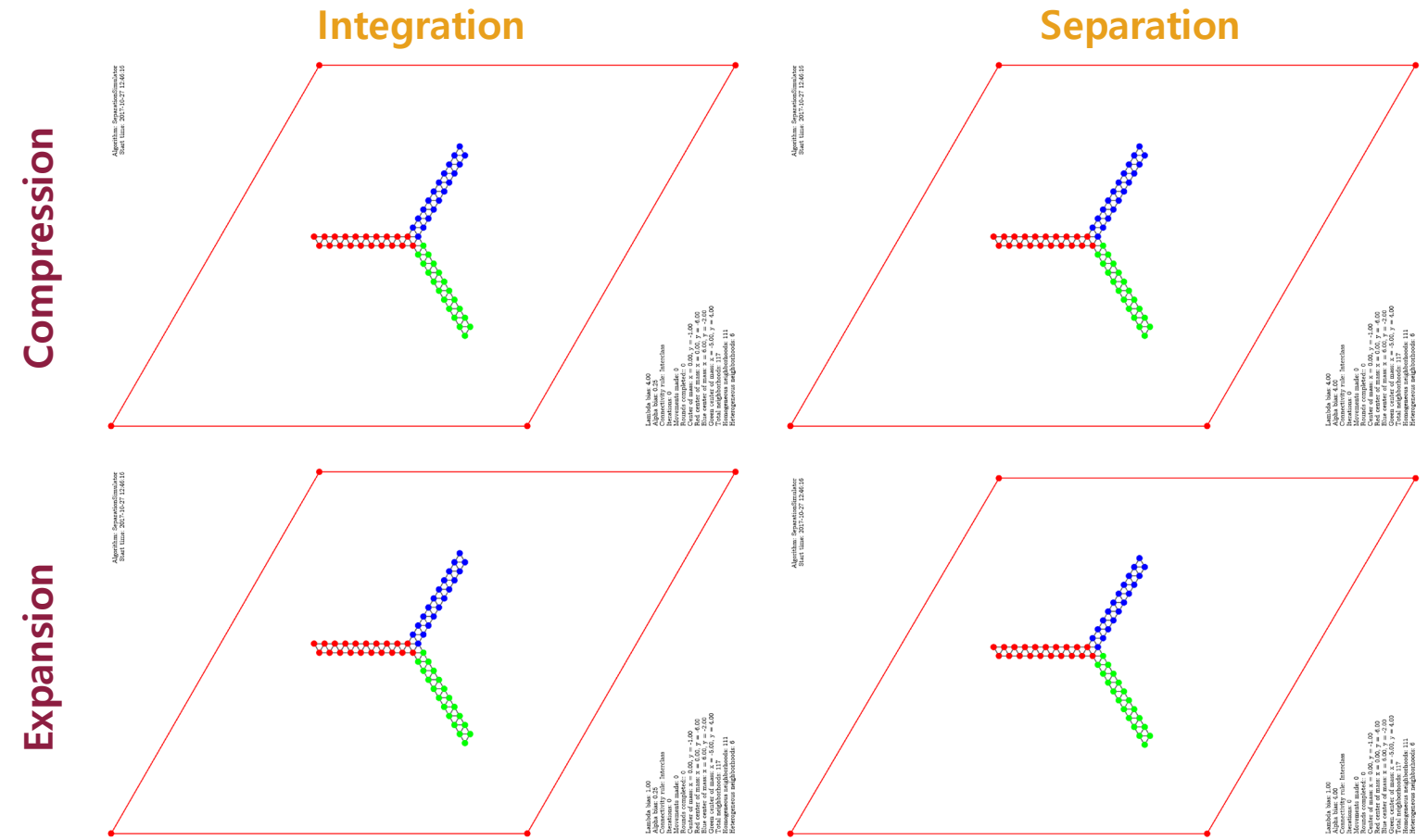
- **Problem:** Maintain bridges that simultaneously balance the tradeoff between the benefit of a shorter path and the cost of more particles in the bridge.
- **Main Idea:** Extends compression by considering gap vs. land locations.
- **Correctness:** With all but exponentially small probability.

RLPKCG 2015: "Army ants dynamically adjust living bridges…"



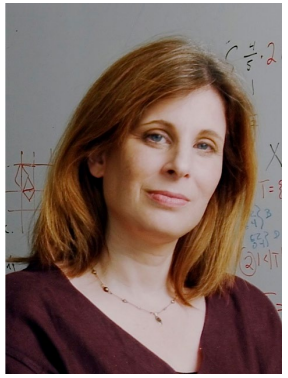
Algorithm 7: Separation (by "color")

- **Problem:** Enable a heterogeneous particle system to dynamically separate or integrate.
- **Main Idea:** Extends compression by considering neighbors of different colors.
- **Correctness:** With all but exponentially small probability.



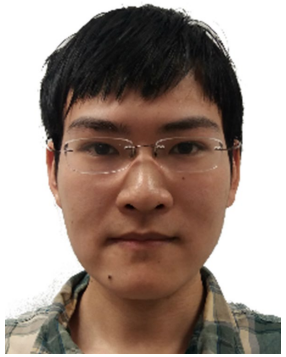
Applications to Swarm Robotics

Georgia Tech


Dana Randall


Cem Gökmen


Dan Goldman


Shengkai Li


Will Savoie


Bahni Dutta

Berkeley UNIVERSITY OF CALIFORNIA

The University Of Sheffield.

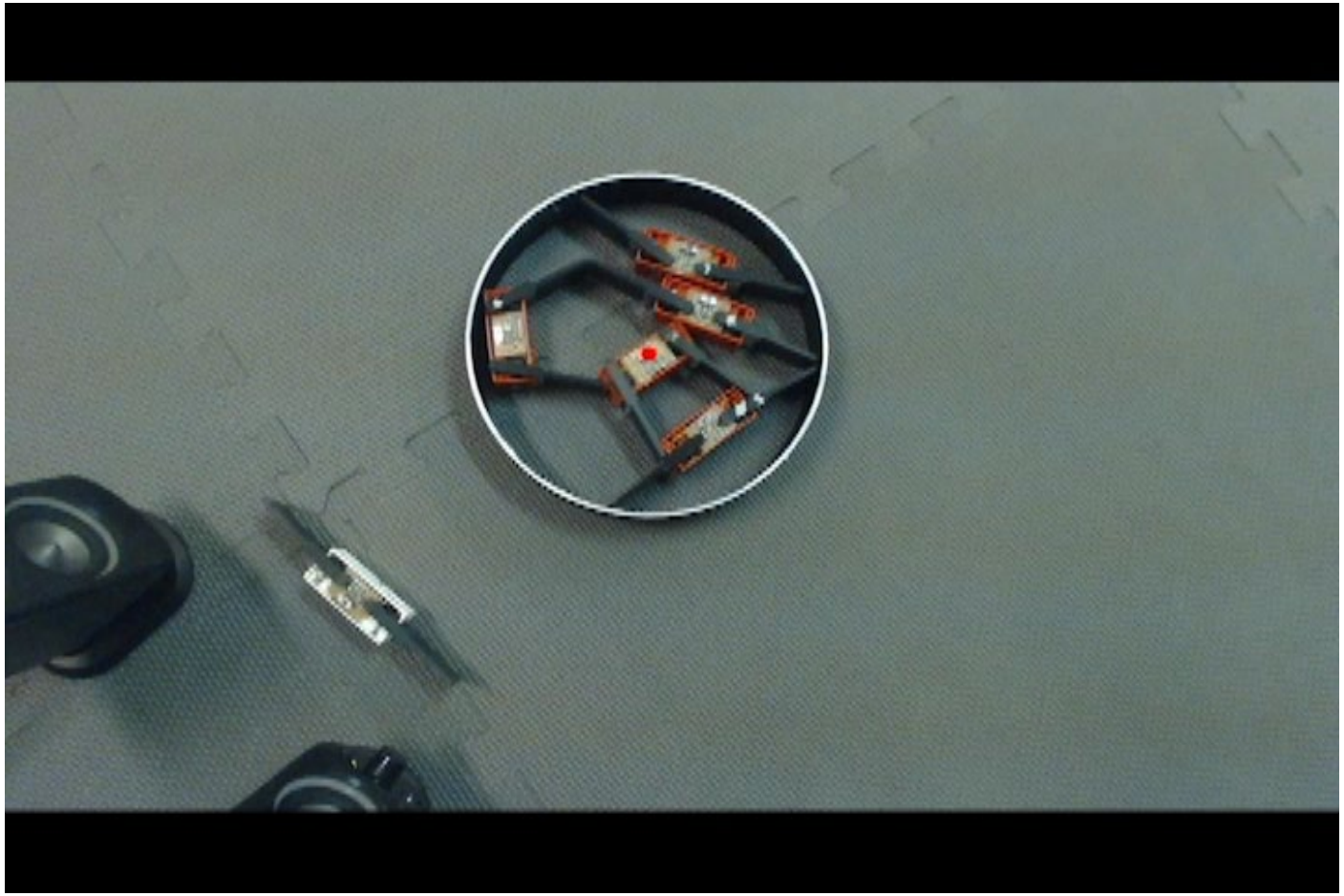
ASU Arizona State University


Sarah Cannon


Roderich Gross

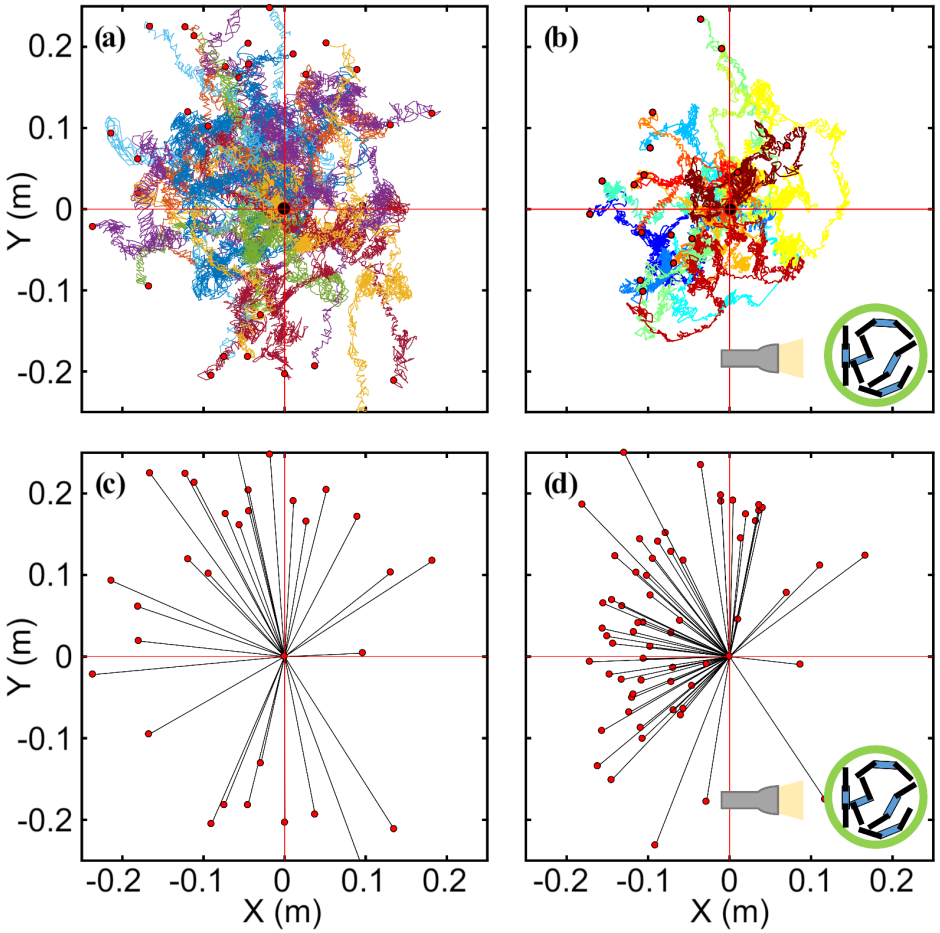

Andréa W. Richa

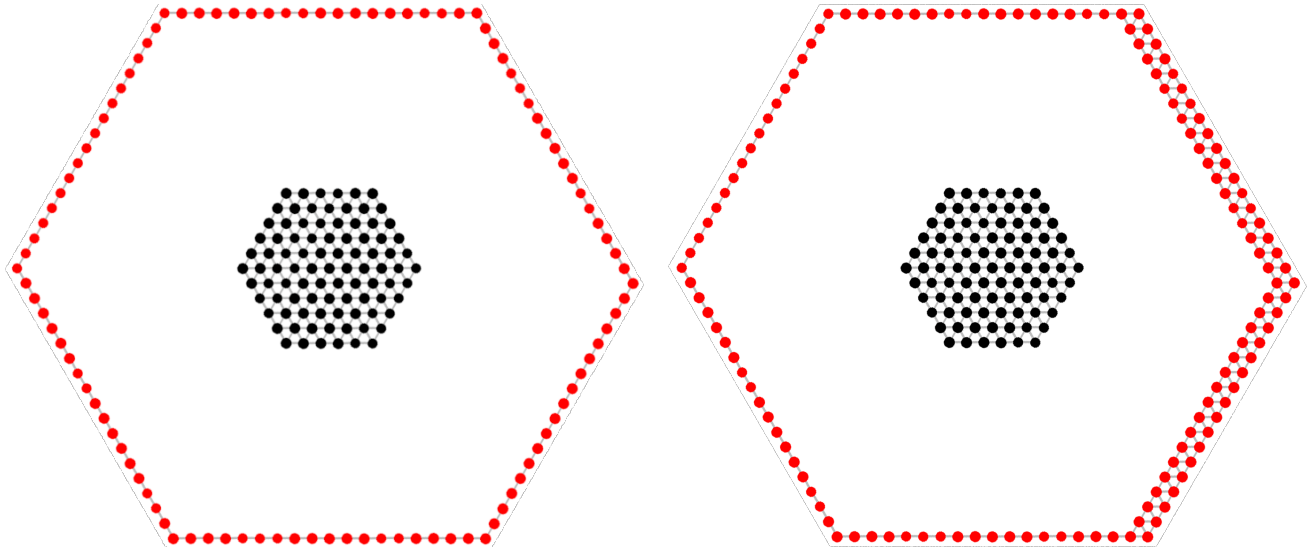
Algorithm 8: Phototaxing



Algorithm 8: Phototaxing

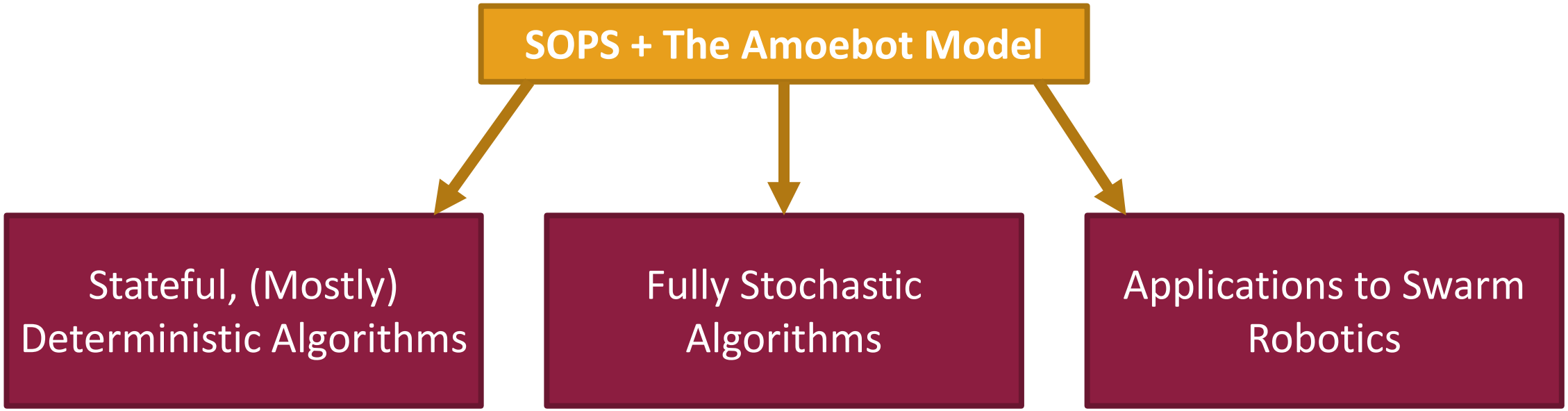
Algorithm 8: Phototaxing





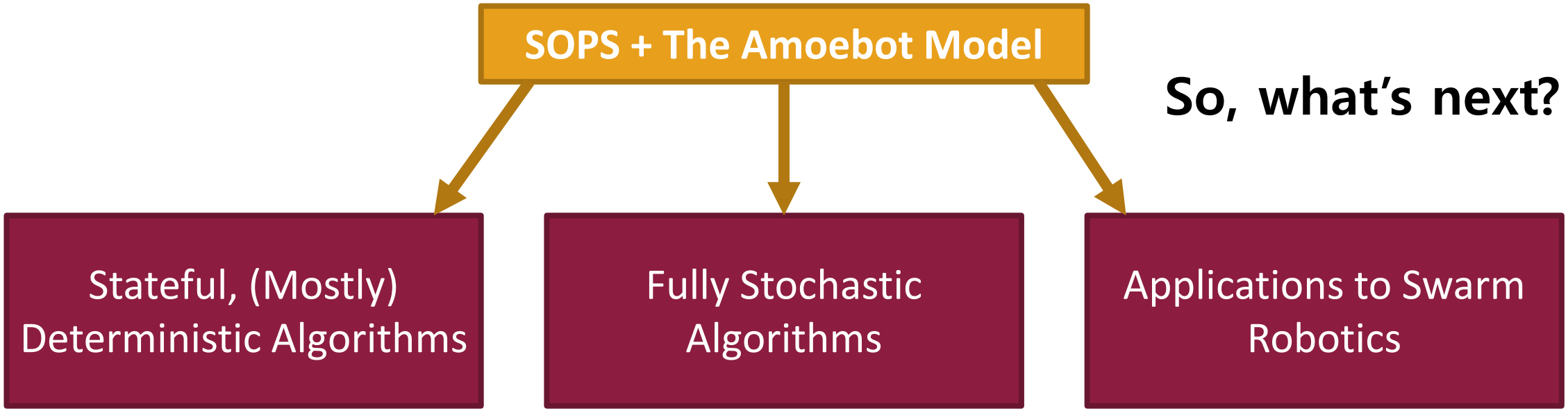
Conclusion (The Big Picture)

What complex, collective behaviors are achievable by systems of simple, restricted programmable particles?



Conclusion (The Big Picture)

Many complex, collective behaviors can be achieved by simple, restricted programmable particles.



So, what's next?

Ongoing Work

Extending to 3D:

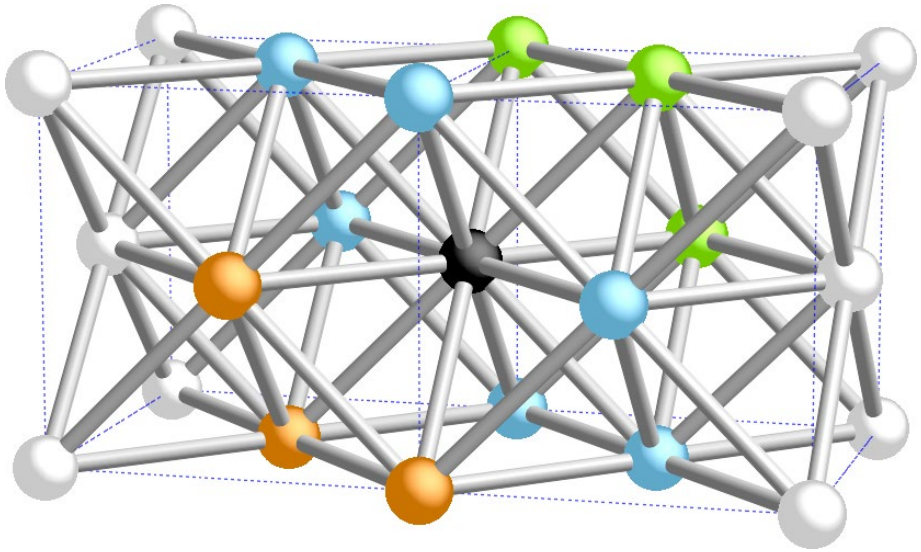
- How should the amoebot model be extended to handle three dimensions?
- Modeling work is nearly done; simulator in progress.

Fault Tolerance:

- How should the amoebot model consider crash and Byzantine failures?
- Can the current algorithms be made fault-tolerant?

Energy Management:

- How could the amoebot model consider energy being supplied to and used by the particles?



Thank you!

sops.engineering.asu.edu

joshdaymude.wordpress.com

Arizona State University