Efficient Analysis of Population Protocols and Chemical Reaction Networks

Doctoral Defense

Martin Helfrich

December 5, 2023

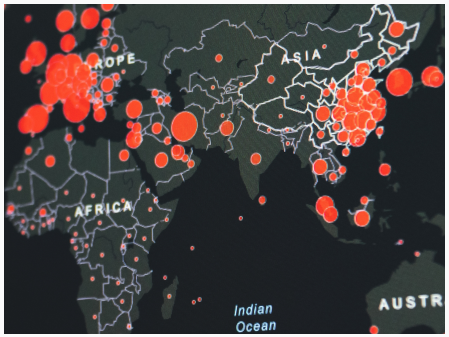
Introduction: Motivation

Introduction: Models

"Population protocols and chemical reaction networks are formal models in which many simple entities interact resulting in a hard to analyse system."

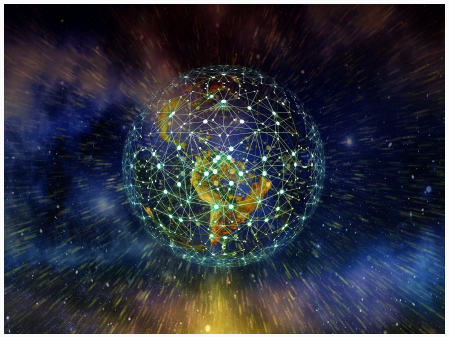
Introduction: Use Cases

Epidemiology

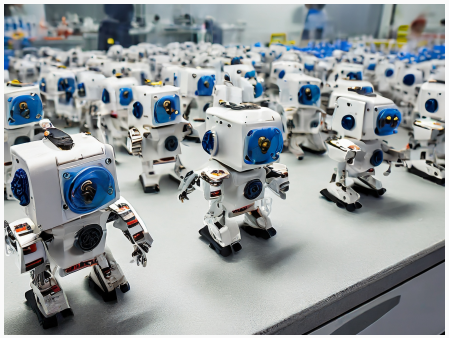


Sensor Networks

Social Networks



PP
CRN



Robot Swarms

Self-Organizing Systems

Drug Discovery

Introduction: Thesis Overview

Goal of Thesis:

Enable efficient (or "practical") analysis of these models via:

1. Theoretical results
2. New efficient analysis methods
3. Easily accessible tools

Introduction: Thesis Overview

Goal of Thesis:

Enable efficient (or "practical") analysis of these models via:

1. Theoretical results
2. New efficient analysis methods
3. Easily accessible tools

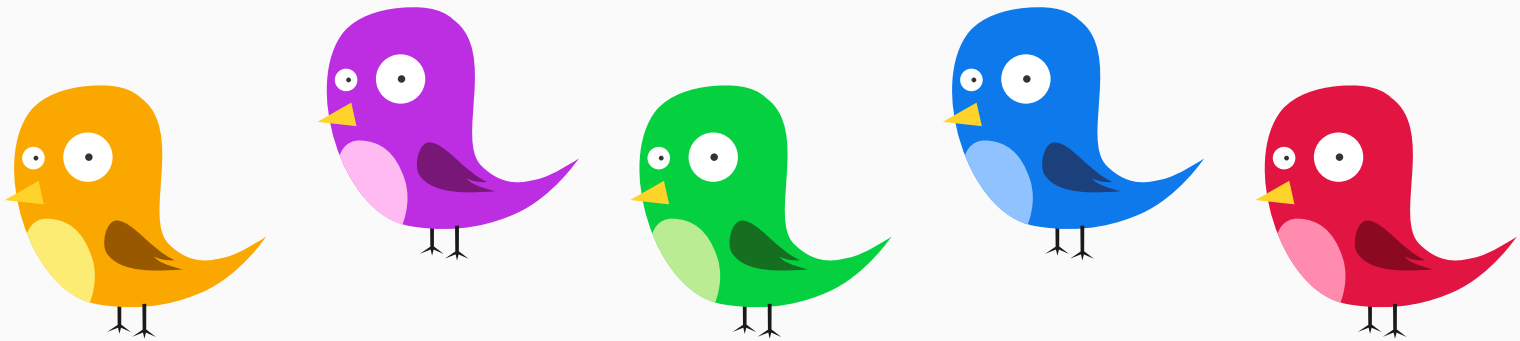
Results of Thesis:

- **Efficient verification of population protocols**
  $\rightarrow$ in this talk: focus on tool Peregrine
- **Synthesis of efficient population protocols**
  $\rightarrow$ not in this talk
- **Efficient transient analysis of chemical reaction networks**
  $\rightarrow$ in this talk: focus on segmental simulation idea

Population Protocols

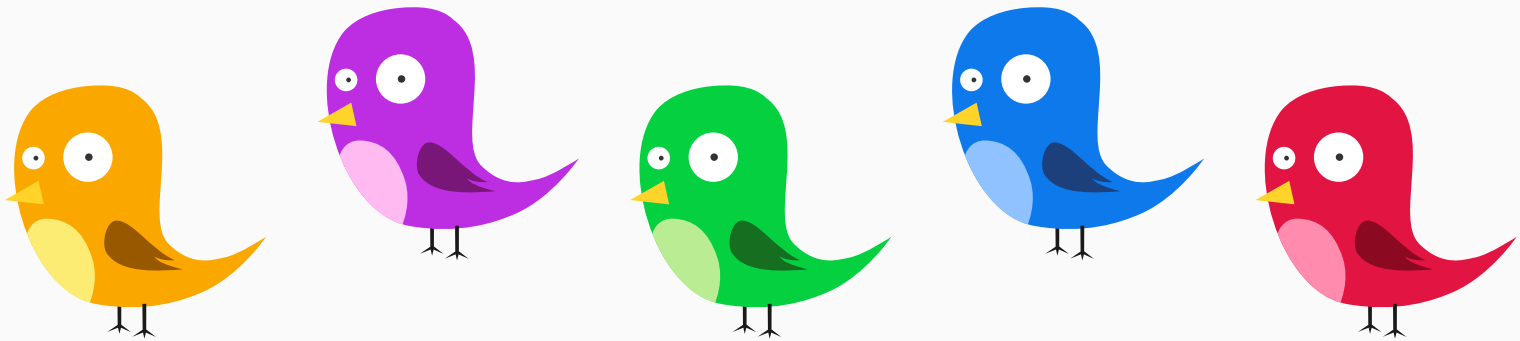
Population Protocols: Intro

(birds by Michael Blondin)



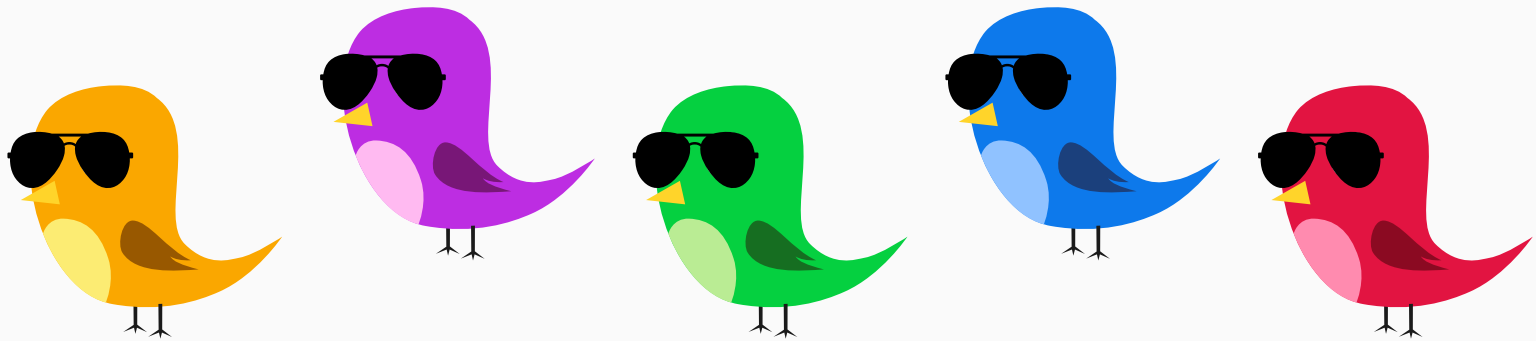
Population Protocols: Intro (birds by Michael Blondin)

- Anonymous mobile agents with very few resources



Population Protocols: Intro (birds by Michael Blondin)

- Anonymous mobile agents with very few resources

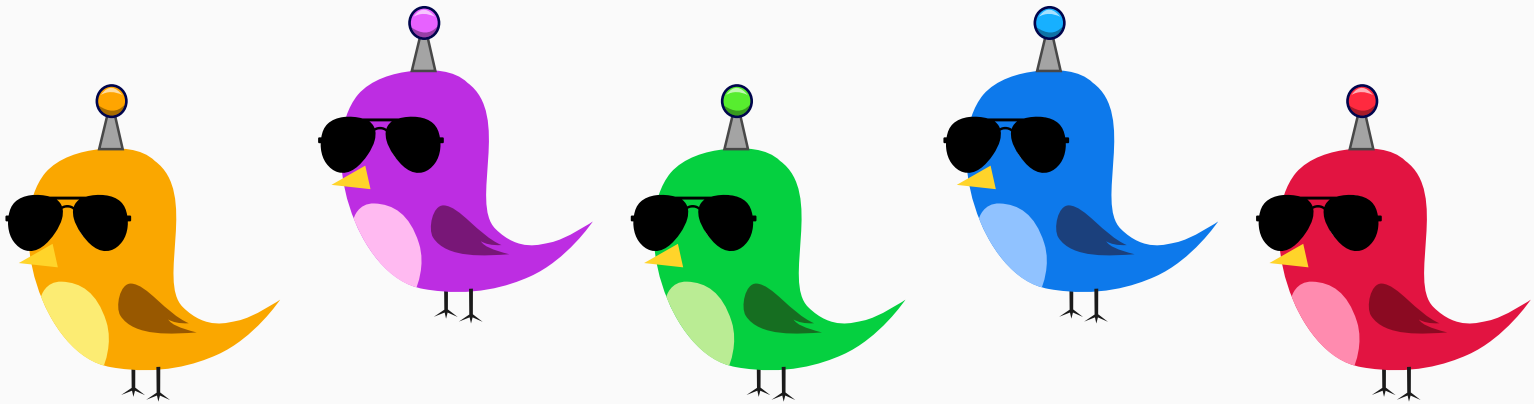


Population Protocols: Intro (birds by Michael Blondin)

- Anonymous mobile agents with very few resources

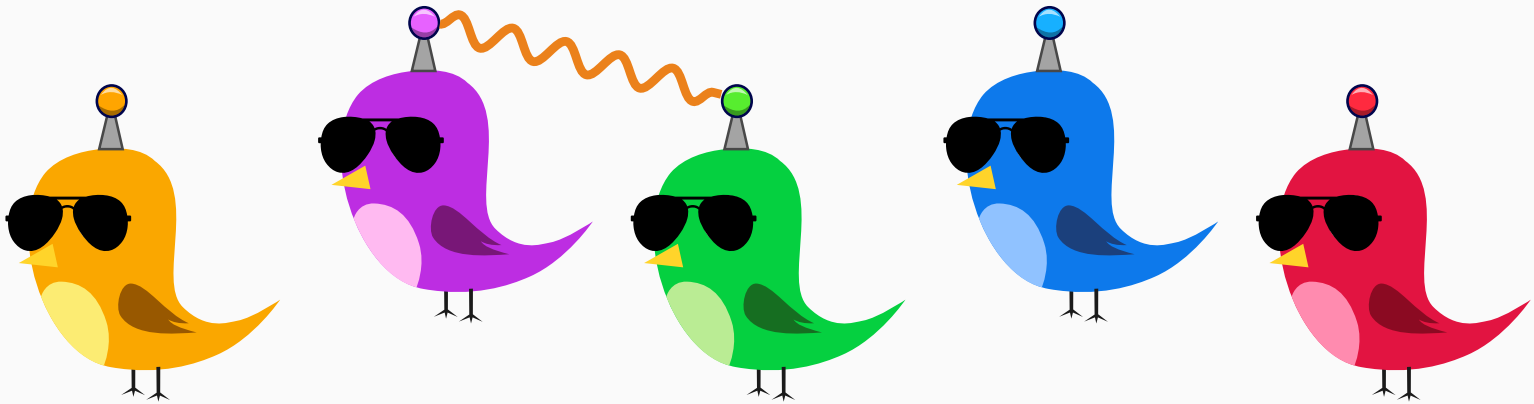
Population Protocols: Intro (birds by Michael Blondin)

- Anonymous mobile agents with very few resources



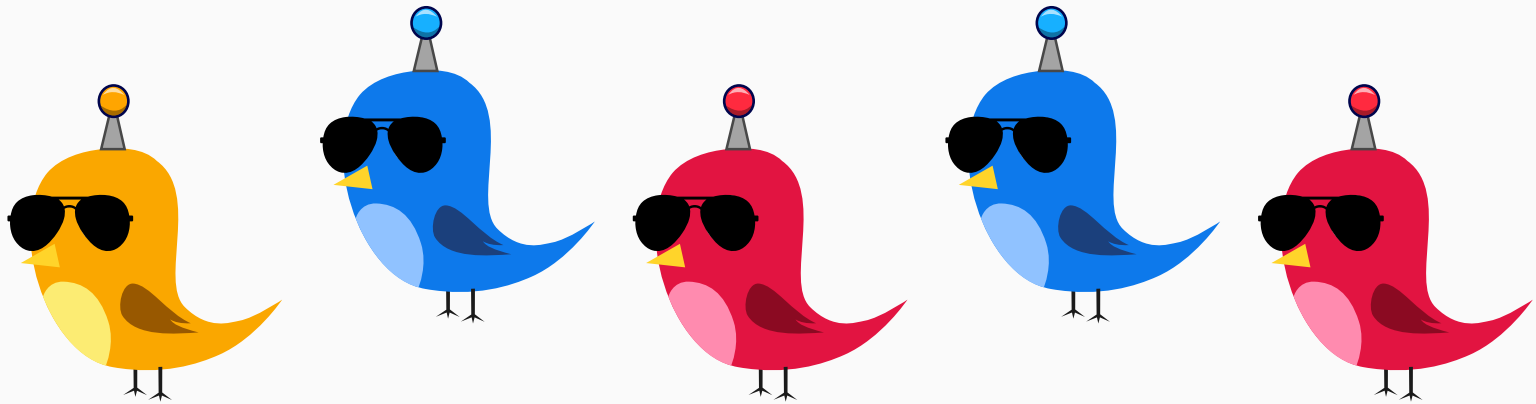
Population Protocols: Intro (birds by Michael Blondin)

- Anonymous mobile agents with very few resources
- Agents change states via random pairwise interactions



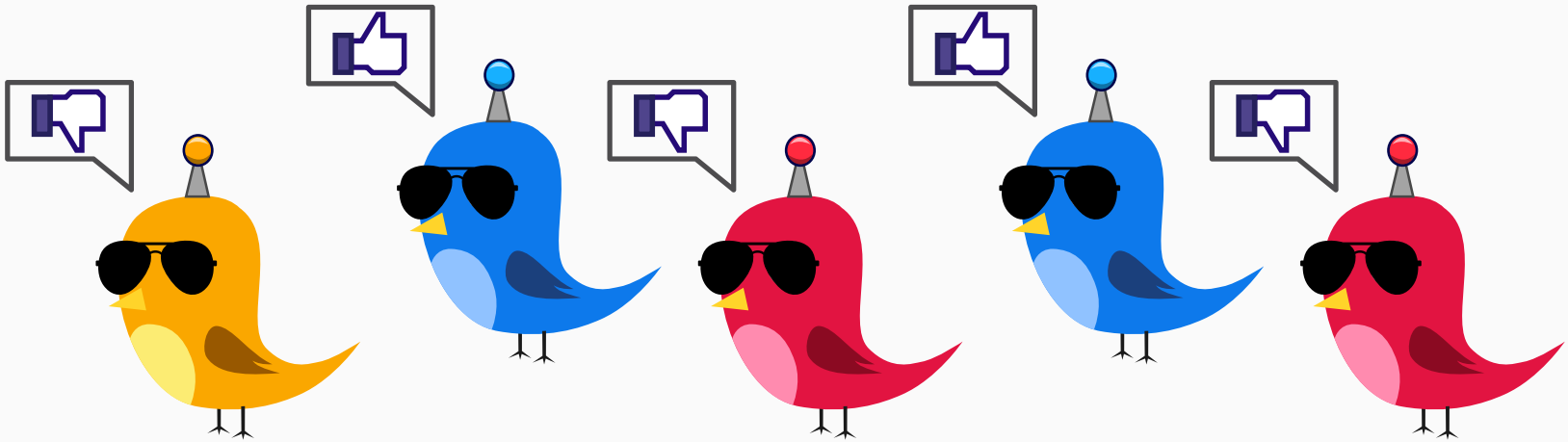
Population Protocols: Intro (birds by Michael Blondin)

- Anonymous mobile agents with very few resources
- Agents change states via random pairwise interactions



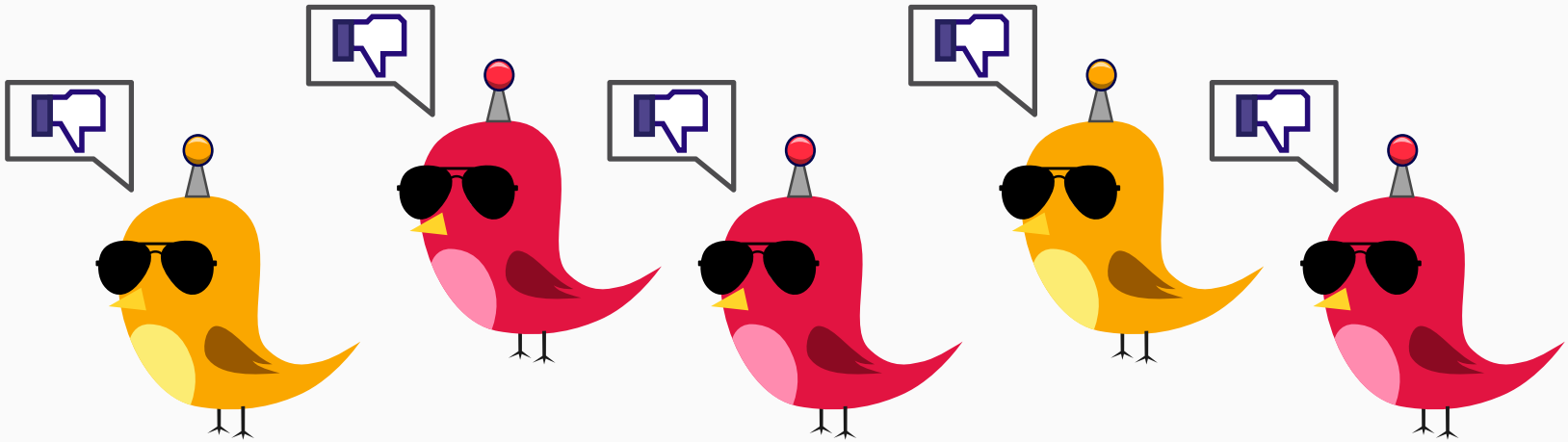
Population Protocols: Intro (birds by Michael Blondin)

- Anonymous mobile agents with very few resources
- Agents change states via random pairwise interactions
- Each agent has opinion true/false



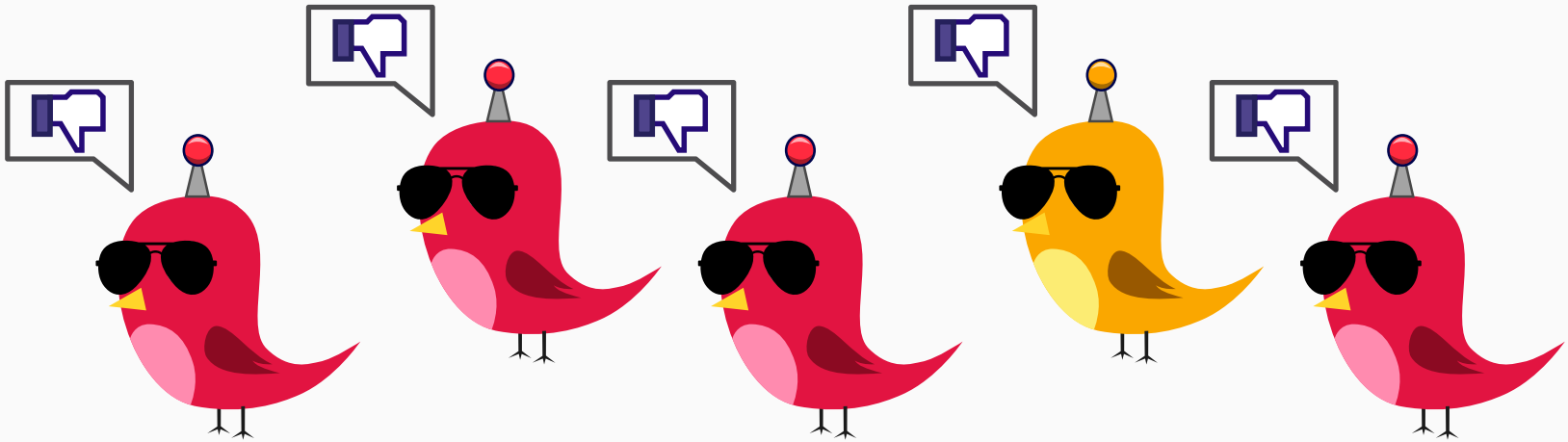
Population Protocols: Intro (birds by Michael Blondin)

- Anonymous mobile agents with very few resources
- Agents change states via random pairwise interactions
- Each agent has opinion true/false
- Computes by stabilizing agents to some opinion



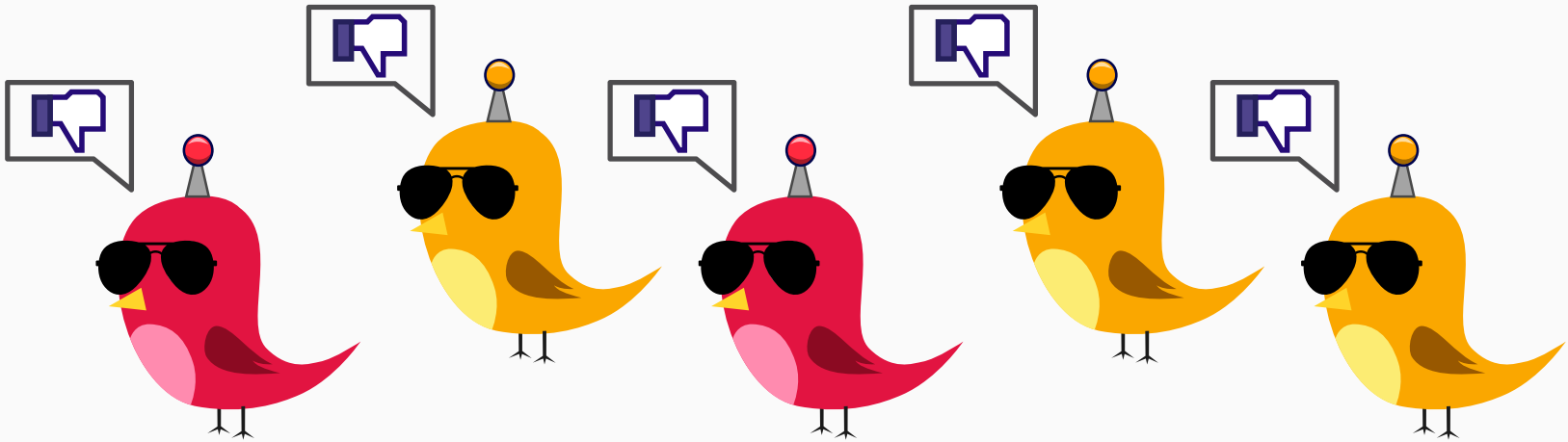
Population Protocols: Intro (birds by Michael Blondin)

- Anonymous mobile agents with very few resources
- Agents change states via random pairwise interactions
- Each agent has opinion true/false
- Computes by stabilizing agents to some opinion



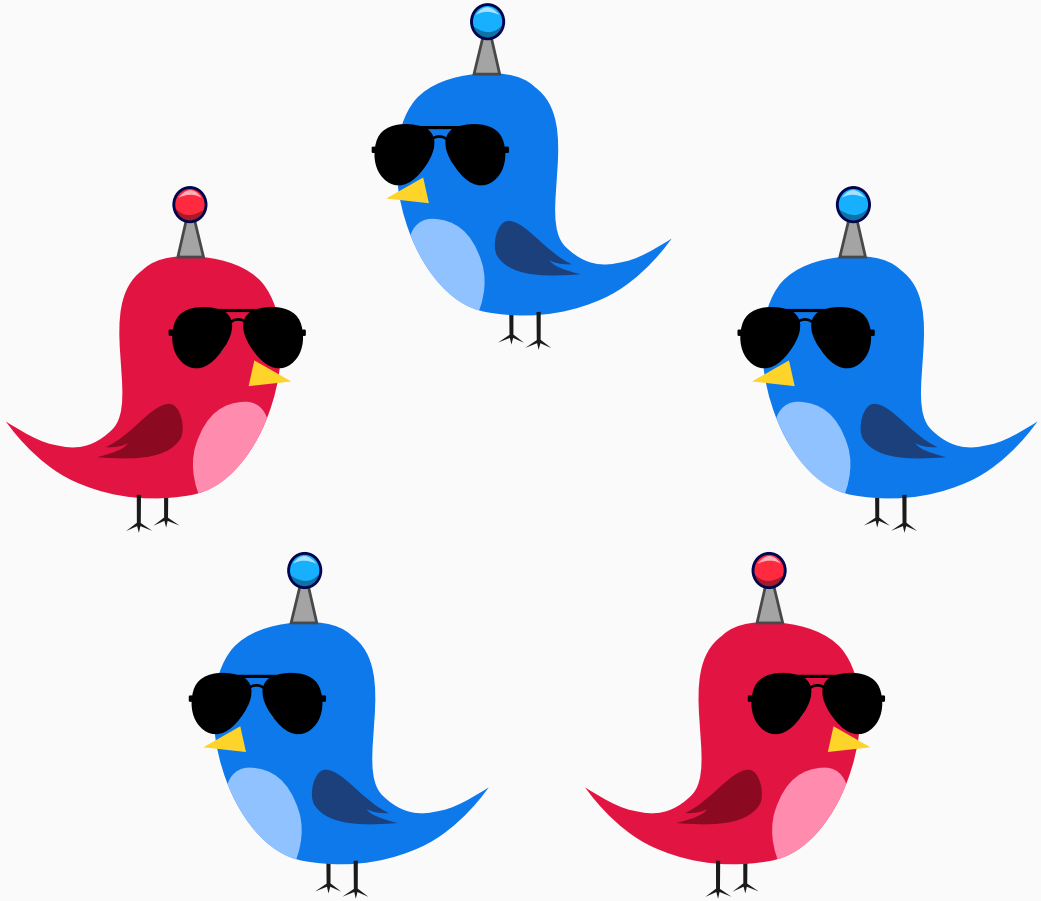
Population Protocols: Intro (birds by Michael Blondin)

- Anonymous mobile agents with very few resources
- Agents change states via random pairwise interactions
- Each agent has opinion true/false
- Computes by stabilizing agents to some opinion



Population Protocols: Example

At least as many blue birds as red birds?

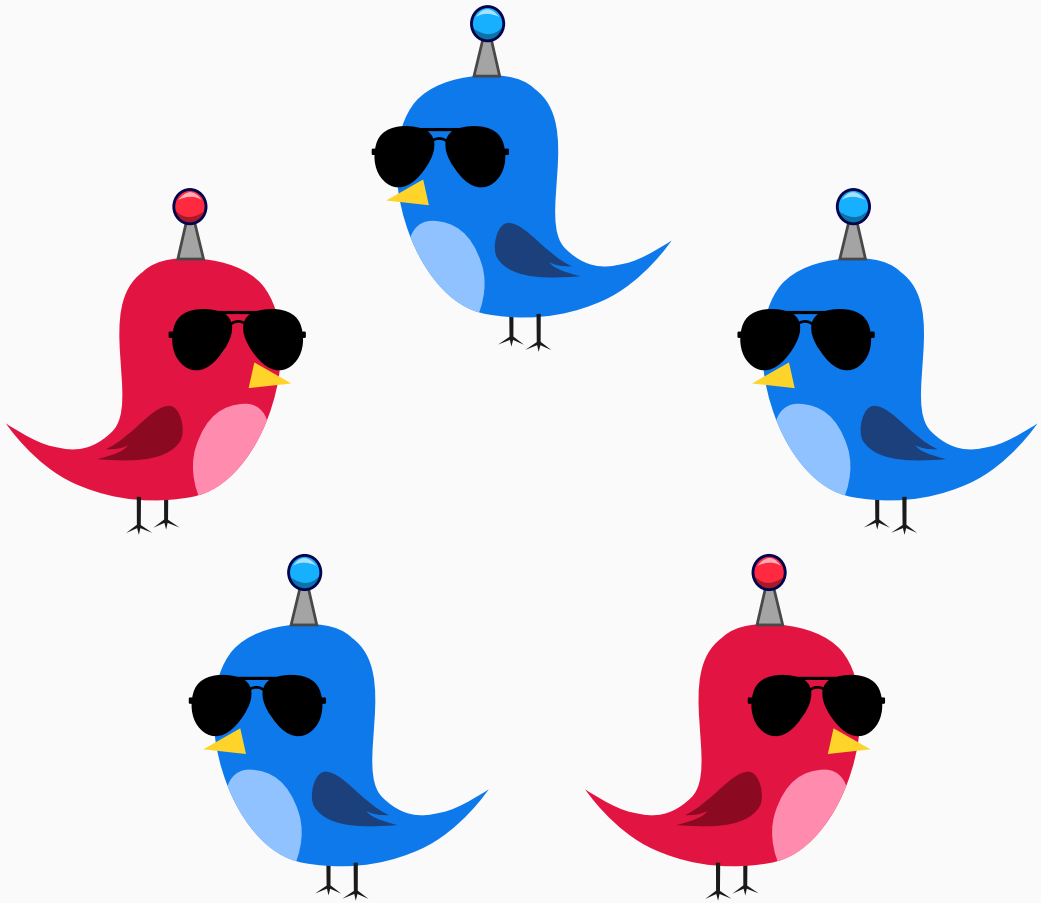


Population Protocols: Example

At least as many blue birds as red birds?

Protocol:

- 4 states: blue/red, large/small
- Two large birds of different colors become small
- Large birds convert small birds to their color
- Small blue birds convert small red birds

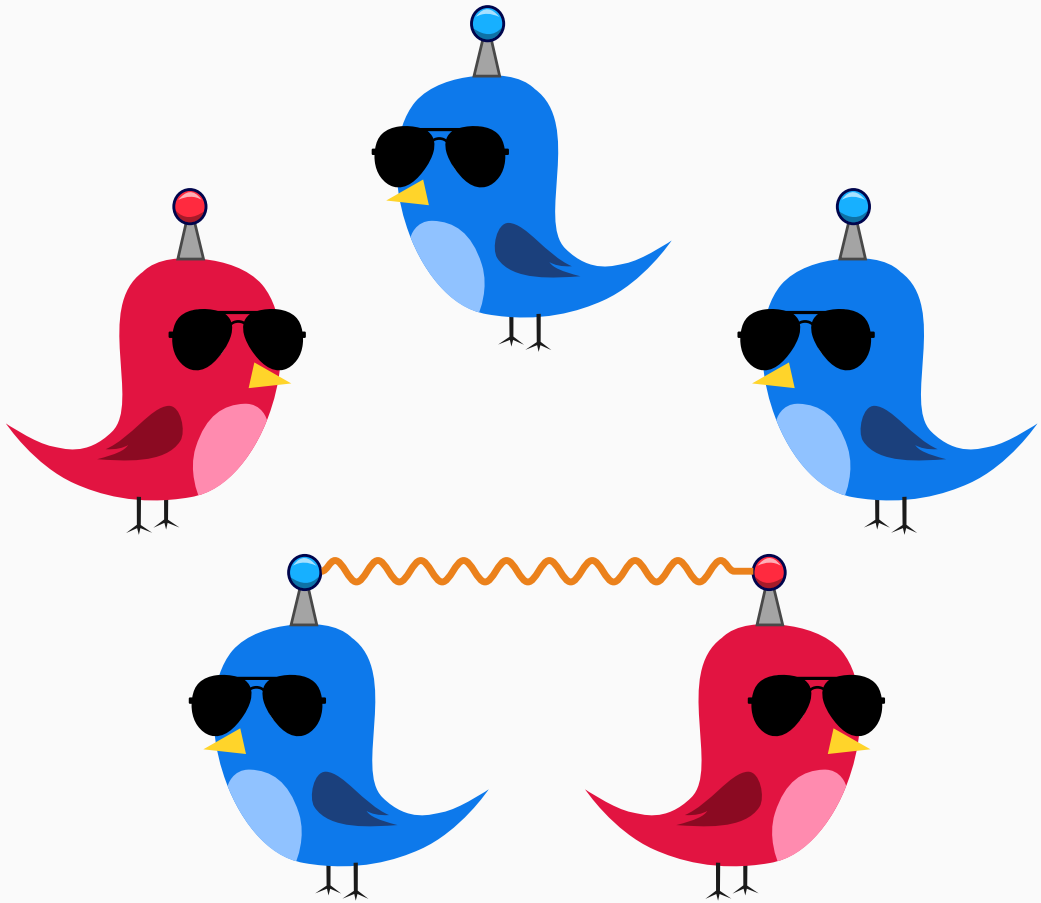


Population Protocols: Example

At least as many blue birds as red birds?

Protocol:

- 4 states: blue/red, large/small
- Two large birds of different colors become small
- Large birds convert small birds to their color
- Small blue birds convert small red birds

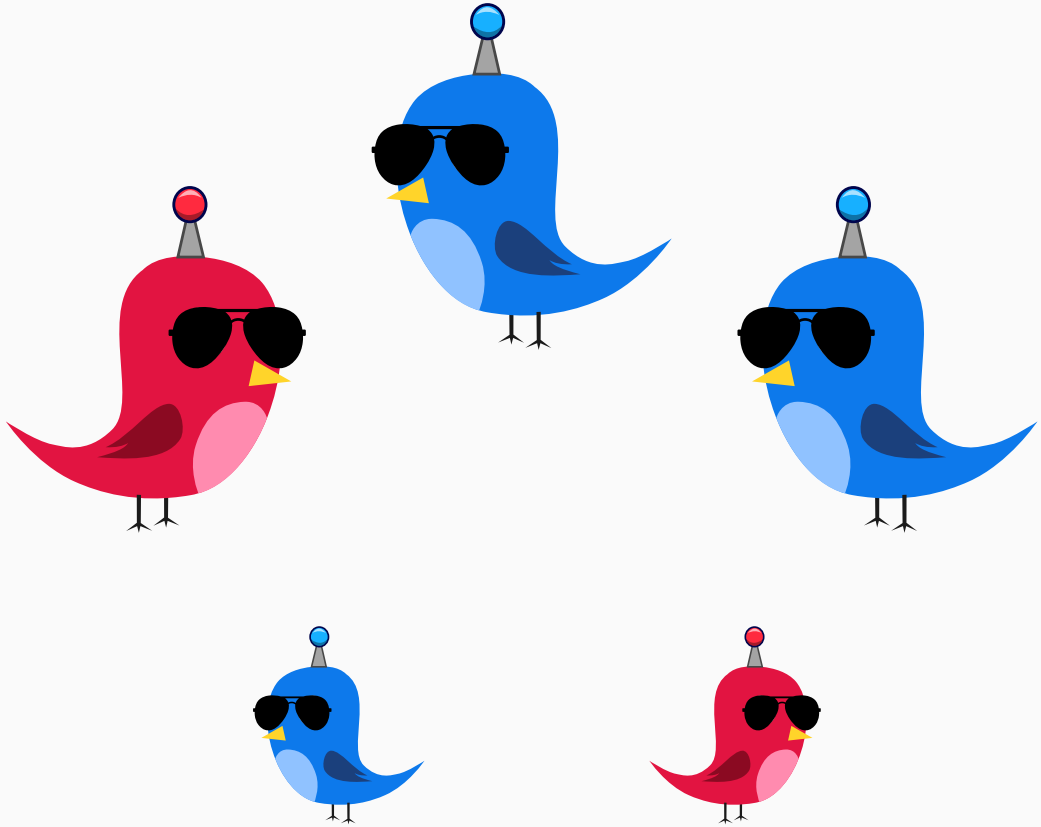


Population Protocols: Example

At least as many blue birds as red birds?

Protocol:

- 4 states: blue/red, large/small
- Two large birds of different colors become small
- Large birds convert small birds to their color
- Small blue birds convert small red birds

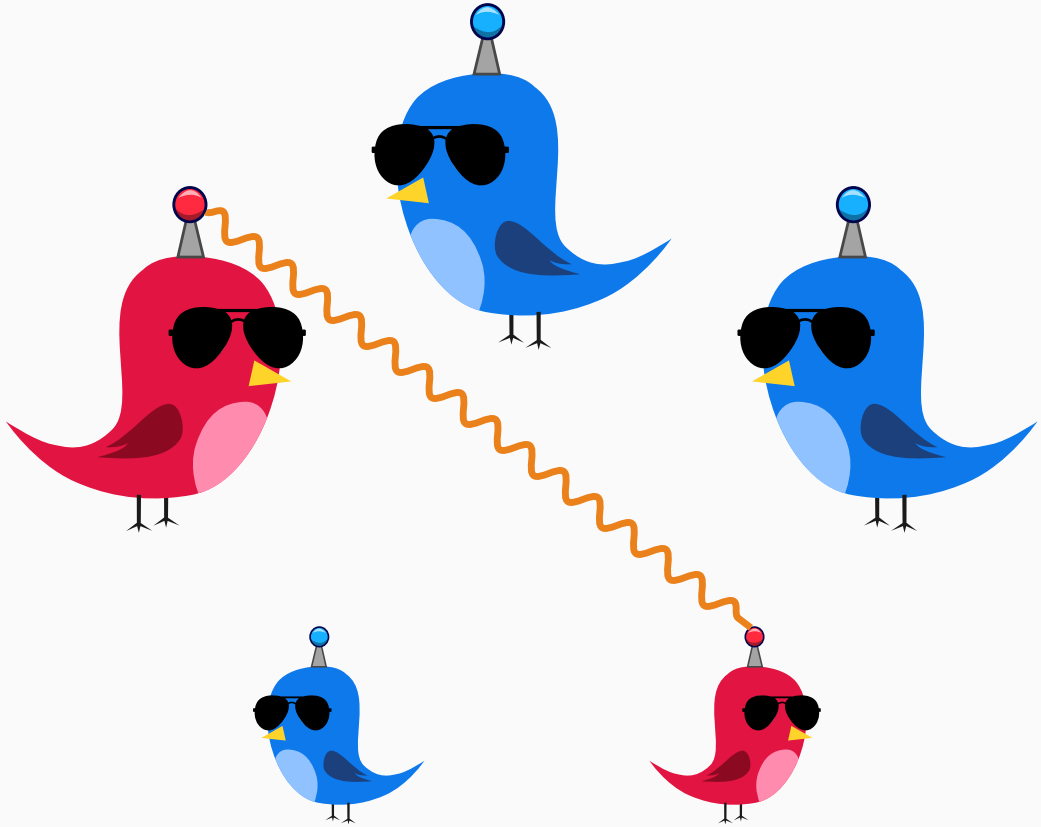


Population Protocols: Example

At least as many blue birds as red birds?

Protocol:

- 4 states: blue/red, large/small
- Two large birds of different colors become small
- Large birds convert small birds to their color
- Small blue birds convert small red birds

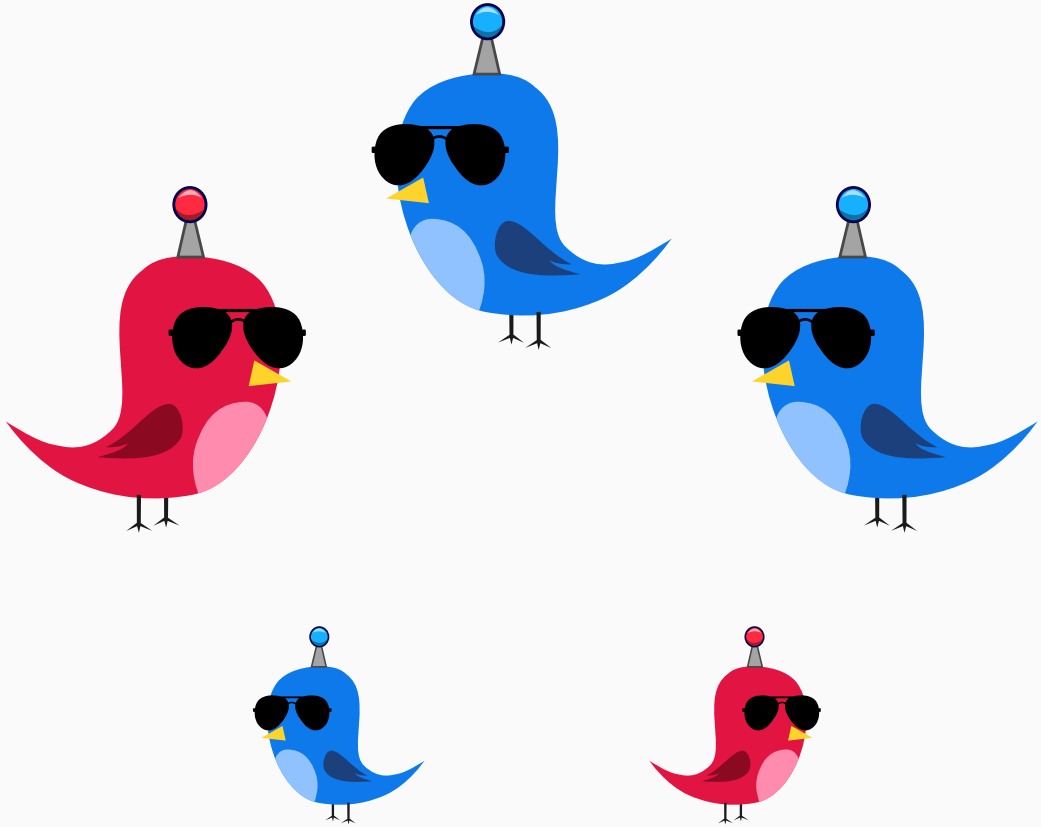


Population Protocols: Example

At least as many blue birds as red birds?

Protocol:

- 4 states: blue/red, large/small
- Two large birds of different colors become small
- Large birds convert small birds to their color
- Small blue birds convert small red birds

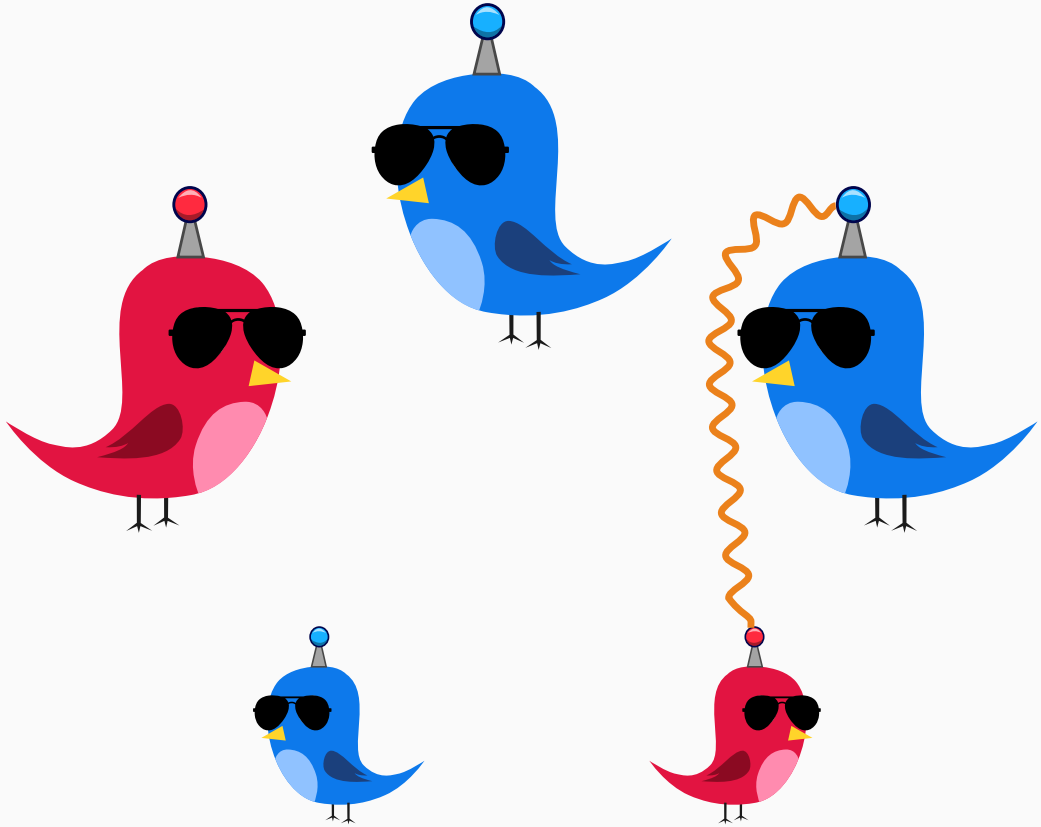


Population Protocols: Example

At least as many blue birds as red birds?

Protocol:

- 4 states: blue/red, large/small
- Two large birds of different colors become small
- Large birds convert small birds to their color
- Small blue birds convert small red birds

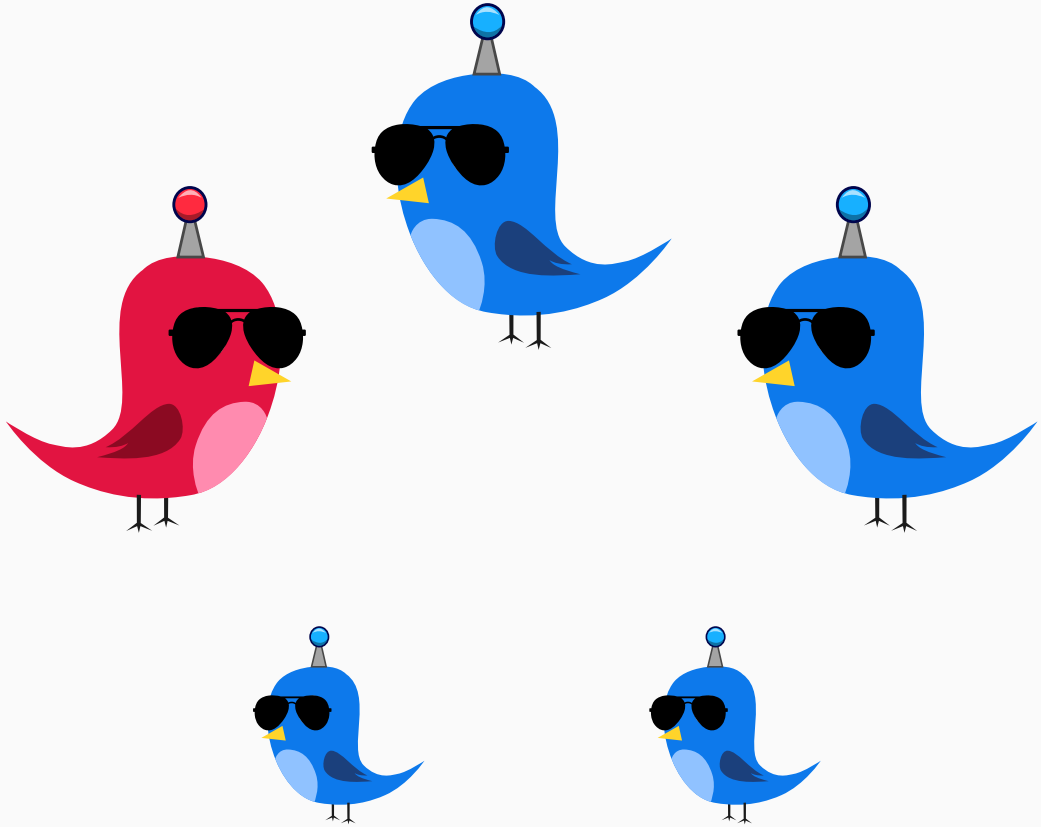


Population Protocols: Example

At least as many blue birds as red birds?

Protocol:

- 4 states: blue/red, large/small
- Two large birds of different colors become small
- Large birds convert small birds to their color
- Small blue birds convert small red birds

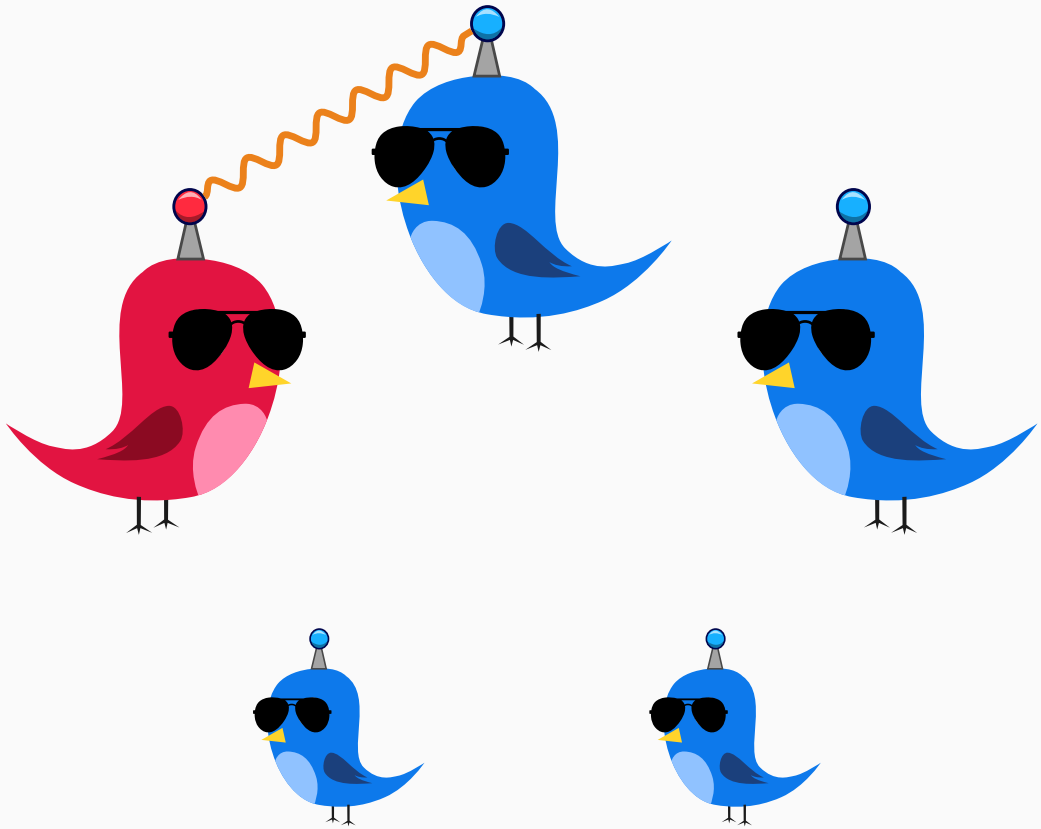


Population Protocols: Example

At least as many blue birds as red birds?

Protocol:

- 4 states: blue/red, large/small
- Two large birds of different colors become small
- Large birds convert small birds to their color
- Small blue birds convert small red birds

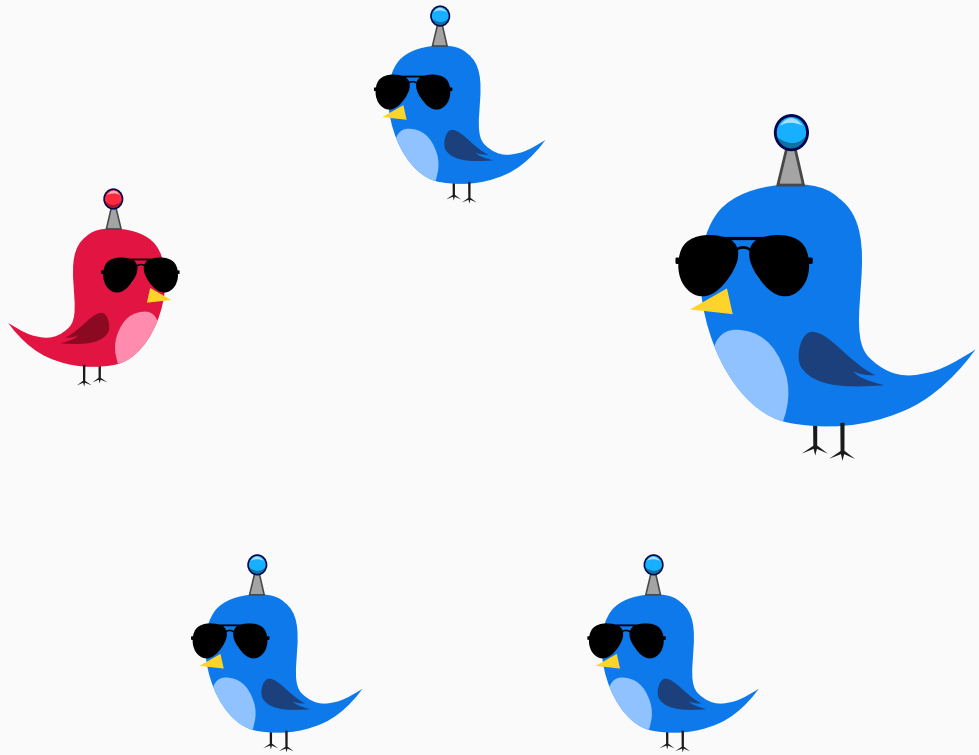


Population Protocols: Example

At least as many blue birds as red birds?

Protocol:

- 4 states: blue/red, large/small
- Two large birds of different colors become small
- Large birds convert small birds to their color
- Small blue birds convert small red birds

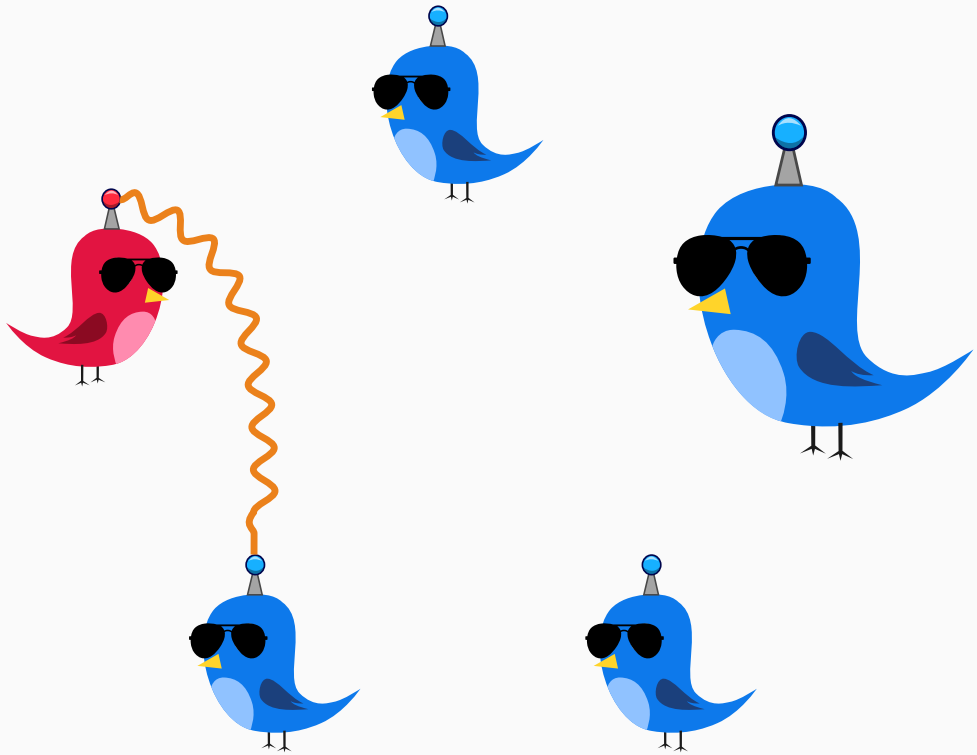


Population Protocols: Example

At least as many blue birds as red birds?

Protocol:

- 4 states: blue/red, large/small
- Two large birds of different colors become small
- Large birds convert small birds to their color
- Small blue birds convert small red birds

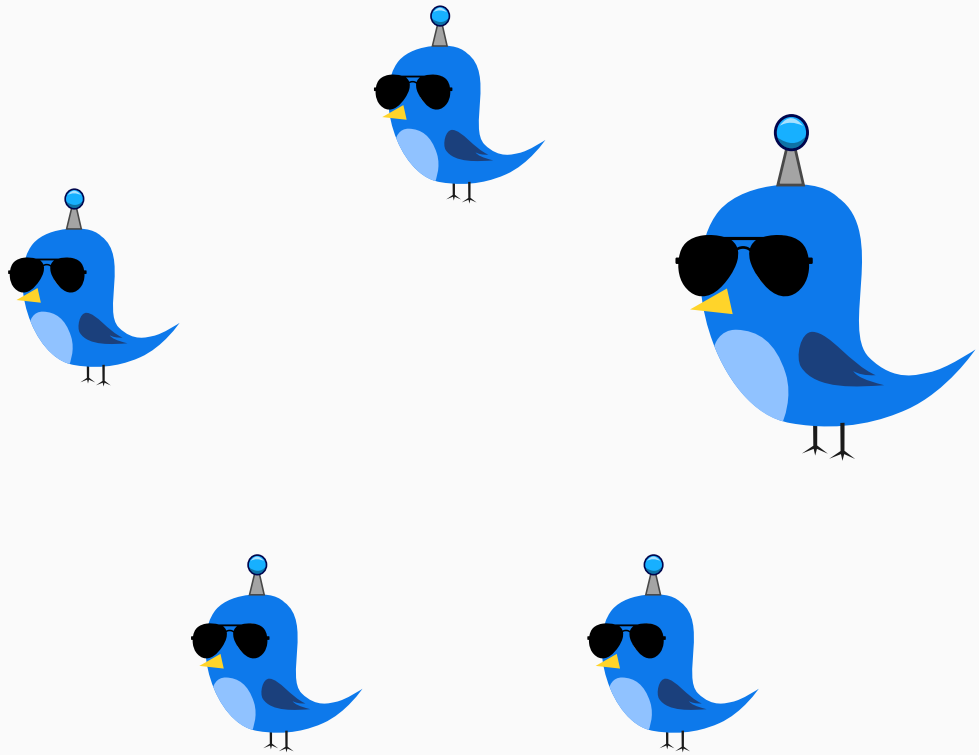


Population Protocols: Example

At least as many blue birds as red birds?

Protocol:

- 4 states: blue/red, large/small
- Two large birds of different colors become small
- Large birds convert small birds to their color
- Small blue birds convert small red birds

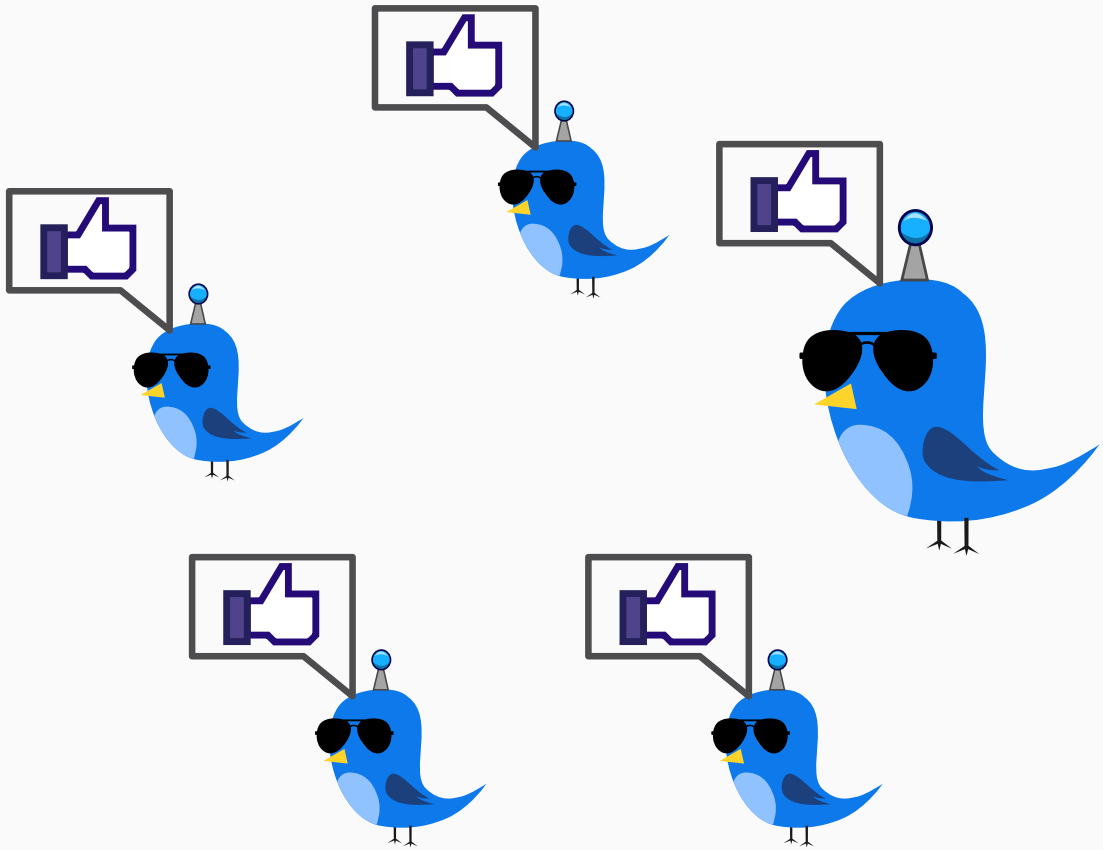


Population Protocols: Example

At least as many blue birds as red birds?

Protocol:

- 4 states: blue/red, large/small
- Two large birds of different colors become small
- Large birds convert small birds to their color
- Small blue birds convert small red birds



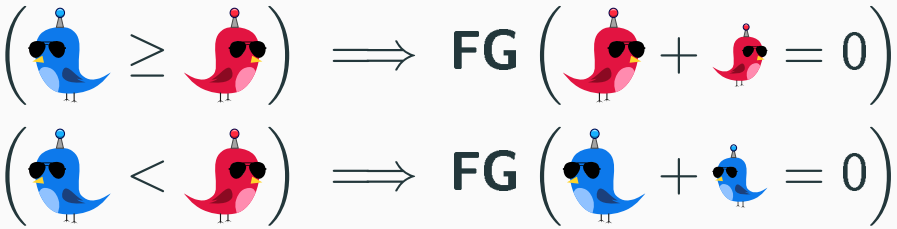
Population Protocols: Example

At least as many blue birds as red birds?

Protocol:

- 4 states: blue/red, large/small
- Two large birds of different colors become small
- Large birds convert small birds to their color
- Small blue birds convert small red birds

Correctness properties:

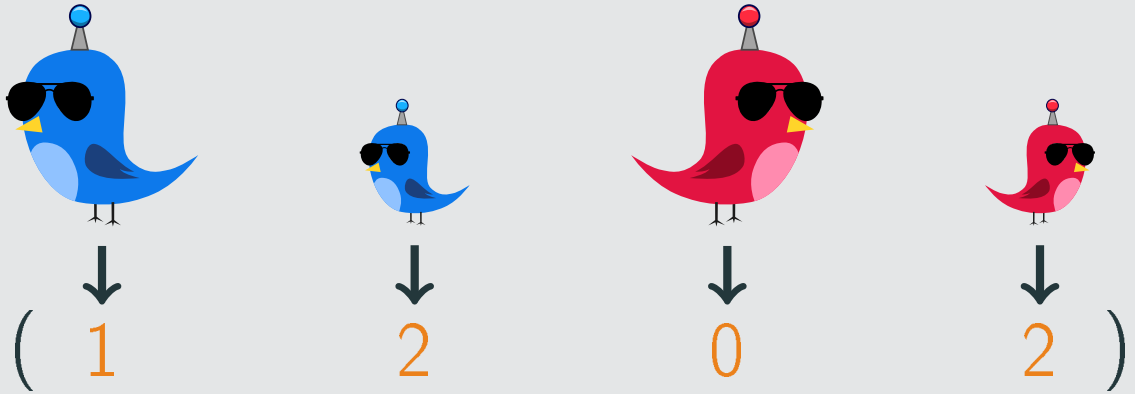


"Birds converge to color of majority."

Population Protocols: Verification Idea

Definition: Configuration

A vector describing the number of agents per state:

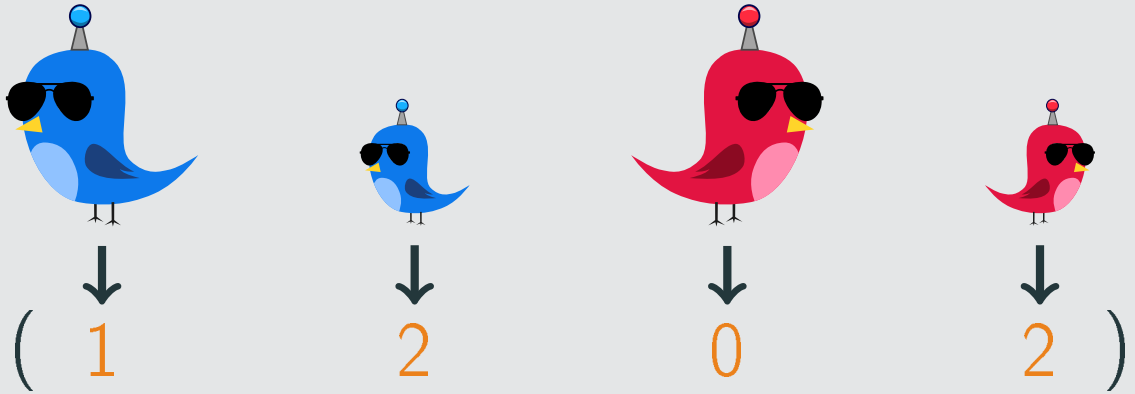


$\rightarrow$ Fully determines global state

Population Protocols: Verification Idea

Definition: Configuration

A vector describing the number of agents per state:



$\rightarrow$ Fully determines global state

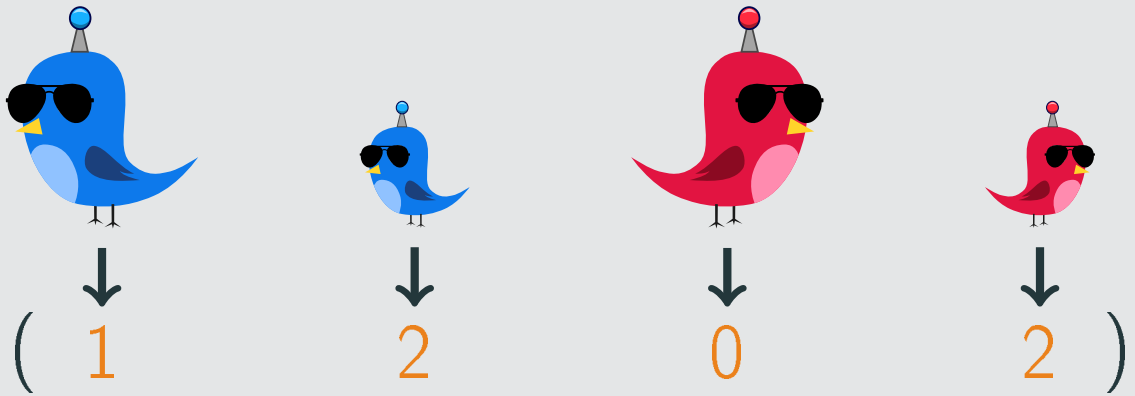
Observation:

Correctness proofs are typically structured in stages that trap the system in progressively more constrained subsets of configurations.

Population Protocols: Verification Idea

Definition: Configuration

A vector describing the number of agents per state:



→ Fully determines global state

Observation:

Correctness proofs are typically structured in stages that trap the system in progressively more constrained subsets of configurations.

→ **Idea:** Formalize structure as stage graph!

Population Protocols: Stage Graphs

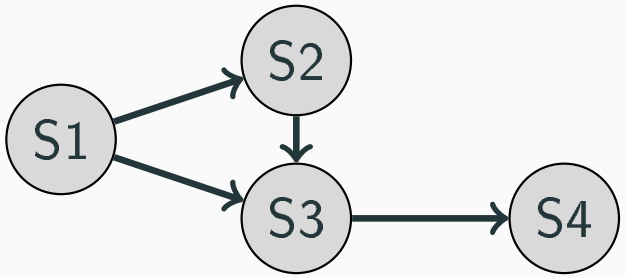
Definition: Stage Graph

A stage graph for $\varphi_{\text{pre}} \Rightarrow FG\varphi_{\text{post}}$ is a finite DAG satisfying:

1. Each node is an inductive set of configurations, called stage.

"It is impossible to leave stages."

Example:



Population Protocols: Stage Graphs

Definition: Stage Graph

A stage graph for $\varphi_{\text{pre}} \Rightarrow FG\varphi_{\text{post}}$ is a finite DAG satisfying:

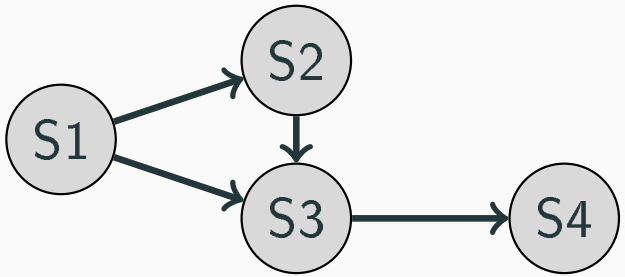
1. Each node is an inductive set of configurations, called stage.

   "It is impossible to leave stages."

2. Every configuration that satisfies φ_{pre} is in some stage.

   "The system starts in a stage."

Example:



Population Protocols: Stage Graphs

Definition: Stage Graph

A stage graph for $\varphi_{\text{pre}} \Rightarrow FG\varphi_{\text{post}}$ is a finite DAG satisfying:

1. Each node is an inductive set of configurations, called stage.

   "It is impossible to leave stages."

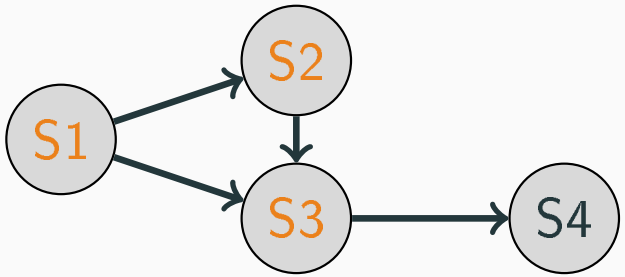
2. Every configuration that satisfies φ_{pre} is in some stage.

   "The system starts in a stage."

3. In stages with successors, executions enters a successor with prob. 1.

   "Stages lead to their successors."

Example:



Population Protocols: Stage Graphs

Definition: Stage Graph

A stage graph for $\varphi_{\text{pre}} \Rightarrow FG\varphi_{\text{post}}$ is a finite DAG satisfying:

1. Each node is an inductive set of configurations, called stage.

   "It is impossible to leave stages."

2. Every configuration that satisfies φ_{pre} is in some stage.

   "The system starts in a stage."

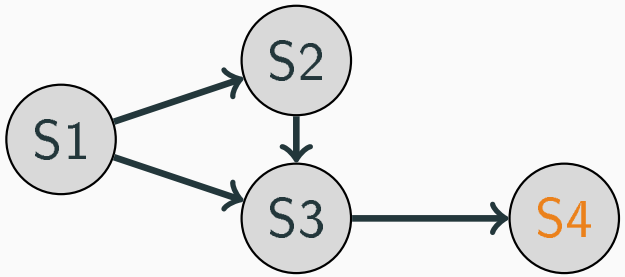
3. In stages with successors, executions enters a successor with prob. 1.

   "Stages lead to their successors."

4. In stages without successors, all configurations satisfy φ_{post}.

   "Finally, the postcondition holds forever."

Example:



Population Protocols: Results

Theory

A population protocol is correct if and only if there is a stage graph proving it.

Population Protocols: Results

Def: Presburger Stage Graph

A stage graph that is described using only Presburger formulas.

Theory

A population protocol is correct if and only if there is a Presburger stage graph proving it.

Population Protocols: Results

Def: Presburger Stage Graph

A stage graph that is described using only Presburger formulas.

Theory

A population protocol is correct if and only if there is a Presburger stage graph proving it.

$\rightarrow$ Decidable **but** stage graphs can be huge!

Population Protocols: Results

Def: Presburger Stage Graph

A stage graph that is described using only Presburger formulas.

Theory

A population protocol is correct if and only if there is a Presburger stage graph proving it.

→ Decidable **but** stage graphs can be huge!

Practice

- Most systems have small stage graphs
- Most systems make progress by "killing" transitions

→ Use heuristics to efficiently construct Presburger stage graphs

Population Protocols: Utility of Stage Graphs

Stage graphs:

- Can be efficiently constructed for most protocols
- Certify liveness properties
- Are independently checkable
- Explain how the protocol works
- Give speed guarantees
- Help to find counter examples

Chemical Reaction Networks

Chemical Reaction Networks

Example: Viral Infection

Species:	$\mathrm{RNA}, \mathrm{DNA}, \mathrm{V}, \mathrm{P}$	
Initial state:	$(1 \times \mathrm{RNA})$	
End time:	200s	
Reactions:	$\mathrm{DNA} + \mathrm{P} \xrightarrow{0.00001} \mathrm{V}$ $\mathrm{RNA} \xrightarrow{1000} \mathrm{RNA} + \mathrm{P}$ $\mathrm{DNA} \xrightarrow{0.025} \mathrm{DNA} + \mathrm{RNA}$ $\mathrm{RNA} \xrightarrow{1} \mathrm{DNA} + \mathrm{RNA}$	$\mathrm{RNA} \xrightarrow{0.25} \emptyset$ $\mathrm{P} \xrightarrow{2} \emptyset$

Chemical Reaction Networks

Example: Viral Infection

Species:	$\mathrm{RNA}, \mathrm{DNA}, \mathrm{V}, \mathrm{P}$	
Initial state:	$(1 \times \mathrm{RNA})$	
End time:	200s	
Reactions:	$\mathrm{DNA} + \mathrm{P} \xrightarrow{0.00001} \mathrm{V}$ $\mathrm{RNA} \xrightarrow{1000} \mathrm{RNA} + \mathrm{P}$ $\mathrm{DNA} \xrightarrow{0.025} \mathrm{DNA} + \mathrm{RNA}$ $\mathrm{RNA} \xrightarrow{1} \mathrm{DNA} + \mathrm{RNA}$	$\mathrm{RNA} \xrightarrow{0.25} \emptyset$ $\mathrm{P} \xrightarrow{2} \emptyset$

Similar to population protocols but:

- Variable number of molecules

Chemical Reaction Networks

Example: Viral Infection

Species:	$\mathrm{RNA}, \mathrm{DNA}, \mathrm{V}, \mathrm{P}$	
Initial state:	$(1 \times \mathrm{RNA})$	
End time:	200s	
Reactions:	$\mathrm{DNA} + \mathrm{P} \xrightarrow{0.00001} \mathrm{V}$ $\mathrm{RNA} \xrightarrow{1000} \mathrm{RNA} + \mathrm{P}$ $\mathrm{DNA} \xrightarrow{0.025} \mathrm{DNA} + \mathrm{RNA}$ $\mathrm{RNA} \xrightarrow{1} \mathrm{DNA} + \mathrm{RNA}$	$\mathrm{RNA} \xrightarrow{0.25} \emptyset$ $\mathrm{P} \xrightarrow{2} \emptyset$

Similar to population protocols but:

- Variable number of molecules
- Continuous time (,i.e., behaves like CTMC not DTMC)

Chemical Reaction Networks

Example: Viral Infection

Species:	$\mathrm{RNA}, \mathrm{DNA}, \mathrm{V}, \mathrm{P}$	
Initial state:	$(1 \times \mathrm{RNA})$	
End time:	200s	
Reactions:	$\mathrm{DNA} + \mathrm{P} \xrightarrow{0.00001} \mathrm{V}$ $\mathrm{RNA} \xrightarrow{1000} \mathrm{RNA} + \mathrm{P}$ $\mathrm{DNA} \xrightarrow{0.025} \mathrm{DNA} + \mathrm{RNA}$ $\mathrm{RNA} \xrightarrow{1} \mathrm{DNA} + \mathrm{RNA}$	$\mathrm{RNA} \xrightarrow{0.25} \emptyset$ $\mathrm{P} \xrightarrow{2} \emptyset$

Similar to population protocols but:

- Variable number of molecules
- Continuous time (,i.e., behaves like CTMC not DTMC)
- Focus on modeling systems (instead of designing them)

CRN: Transient Analysis

How does the system evolve?

CRN: Transient Analysis

How does the system evolve?

Transient analysis:

- Hard because of complex dynamics, state-space explosion, stochasticity, stiffness, multimodality, ...
- Two approaches:
  - Direct (numerical)
  - Indirect (using many trajectories)

CRN: Transient Analysis

How does the system evolve?

Transient analysis:

- Hard because of complex dynamics, state-space explosion, stochasticity, stiffness, multimodality, ...
- Two approaches:
  - Direct (numerical)
  - Indirect (using many trajectories)

This work

- Goal: Efficiently compute many simulations
- Idea: Use memoization!

CRN: Simulation

Gillespie's stochastic simulation algorithm (SSA) [9]

- Sample one reaction at a time

CRN: Simulation

Gillespie's stochastic simulation algorithm (SSA) [9]

- Sample one reaction at a time

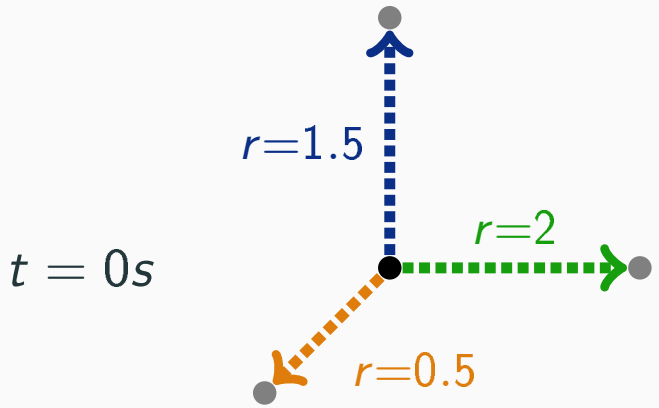
$t = 0s$ $\quad$ $\bullet$ s_{init}

Start in initial state.

CRN: Simulation

Gillespie's stochastic simulation algorithm (SSA) [9]

- Sample one reaction at a time

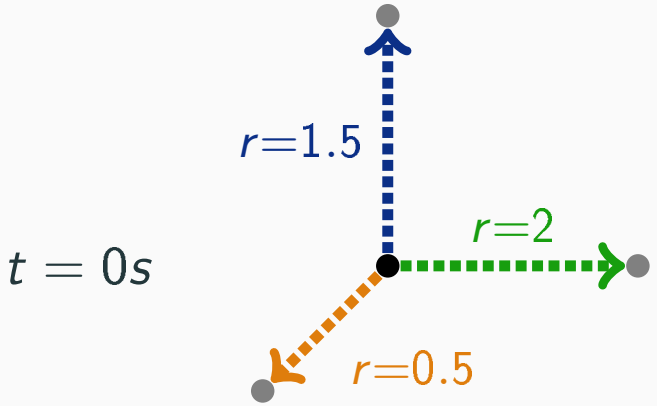


Compute rate of all reactions.

CRN: Simulation

Gillespie's stochastic simulation algorithm (SSA) [9]

- Sample one reaction at a time



Time until the next reaction: $\Delta t \sim EXP(0.5{+}2{+}1.5)$

Probability of reactions: $\frac{0.5}{4}, \frac{2}{4}, \frac{1.5}{4}$

CRN: Simulation

Gillespie's stochastic simulation algorithm (SSA) [9]

- Sample one reaction at a time

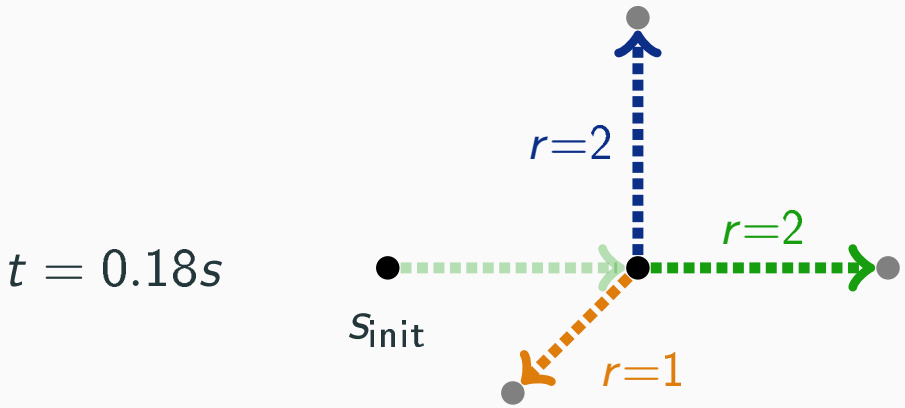
$t = 0.18s$

Apply change to both time and state, then repeat.

CRN: Simulation

Gillespie's stochastic simulation algorithm (SSA) [9]

- Sample one reaction at a time



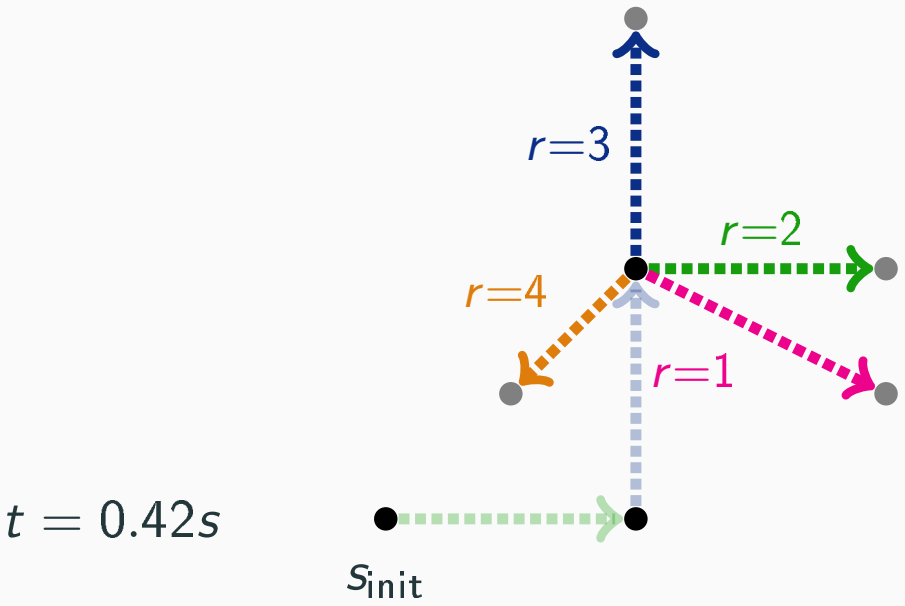
Time until the next reaction: $\Delta t \sim EXP(1{+}2{+}2)$

Probability of reactions: $\frac{1}{5}, \frac{2}{5}, \frac{2}{5}$

CRN: Simulation

Gillespie's stochastic simulation algorithm (SSA) [9]

- Sample one reaction at a time



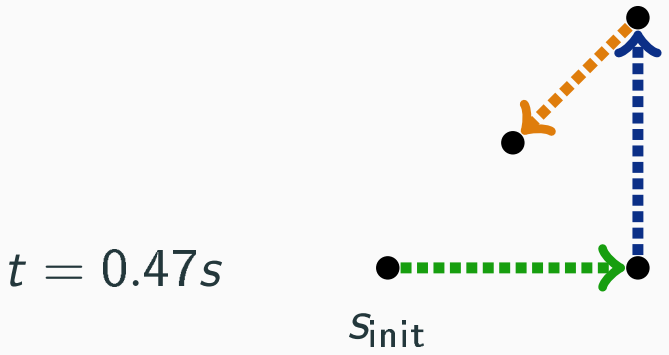
Time until the next reaction: $\Delta t \sim EXP(4+2+3+1)$

Probability of reactions: $\frac{4}{10}, \frac{2}{10}, \frac{3}{10}, \frac{1}{10}$

CRN: Simulation

Gillespie's stochastic simulation algorithm (SSA) [9]

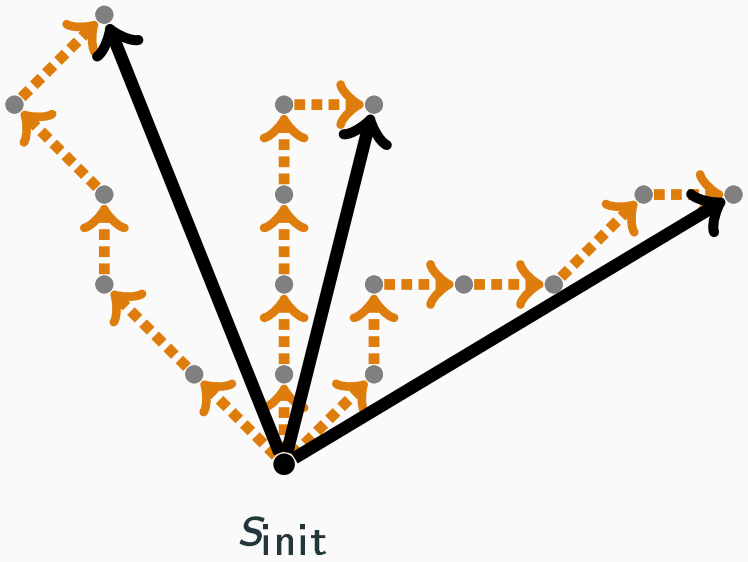
- Sample one reaction at a time
- May take a long time



CRN: Segmental Simulation

Precompute k short trajectories (called segments) for each state.

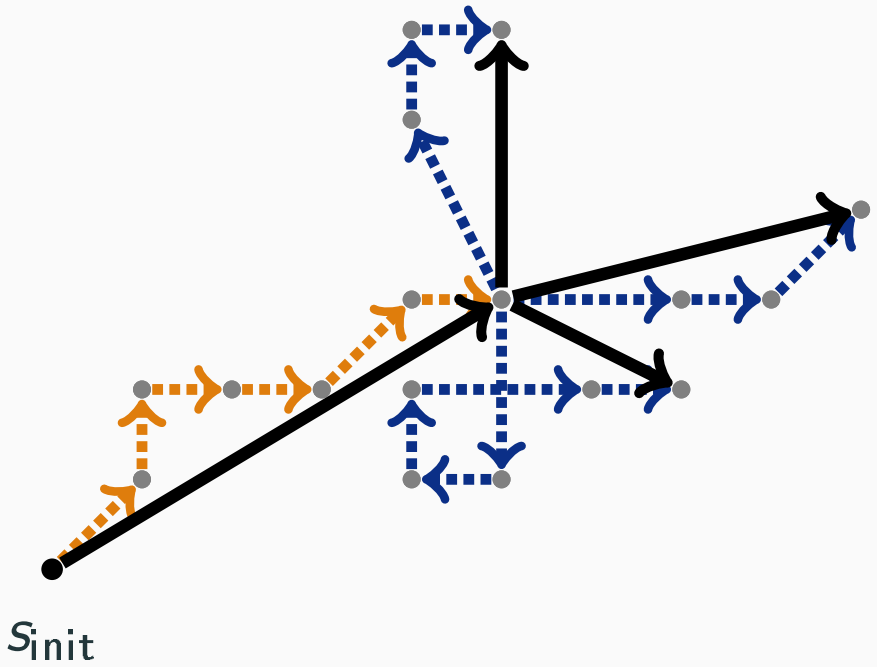
$\rightarrow$ Simulate by sampling segments instead of single reactions.



CRN: Segmental Simulation

Precompute k short trajectories (called segments) for each state.

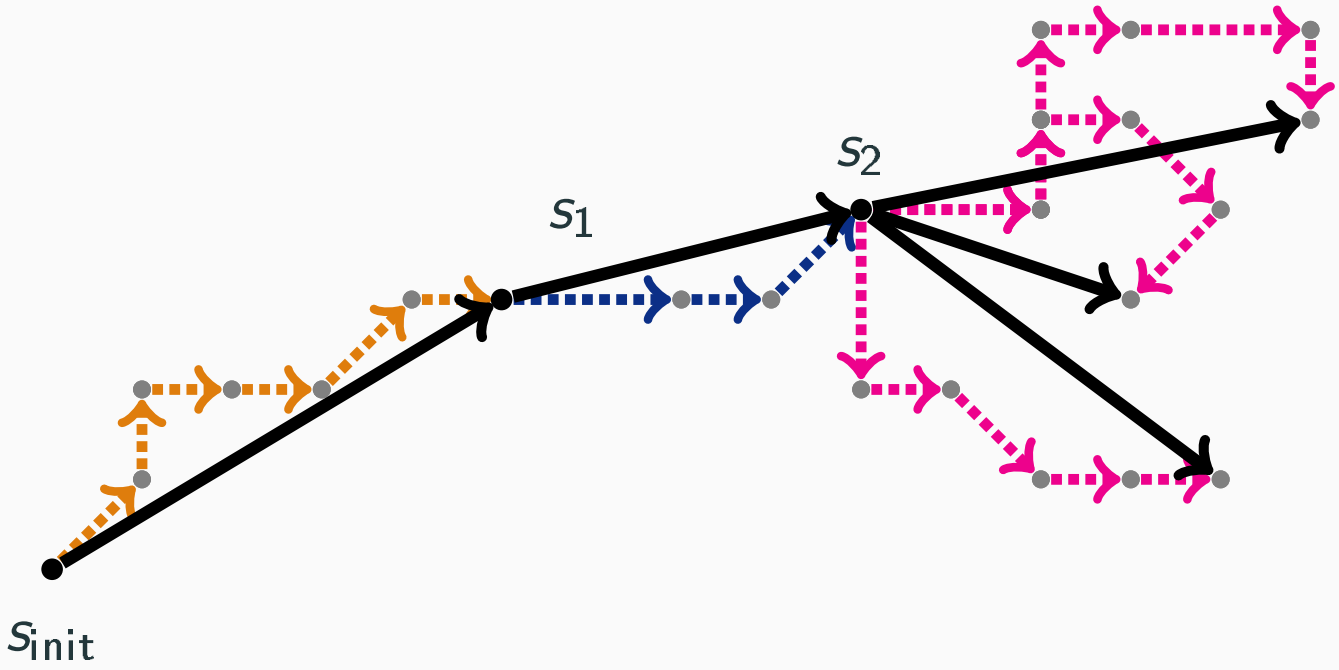
$\rightarrow$ Simulate by sampling segments instead of single reactions.



CRN: Segmental Simulation

Precompute k short trajectories (called segments) for each state.

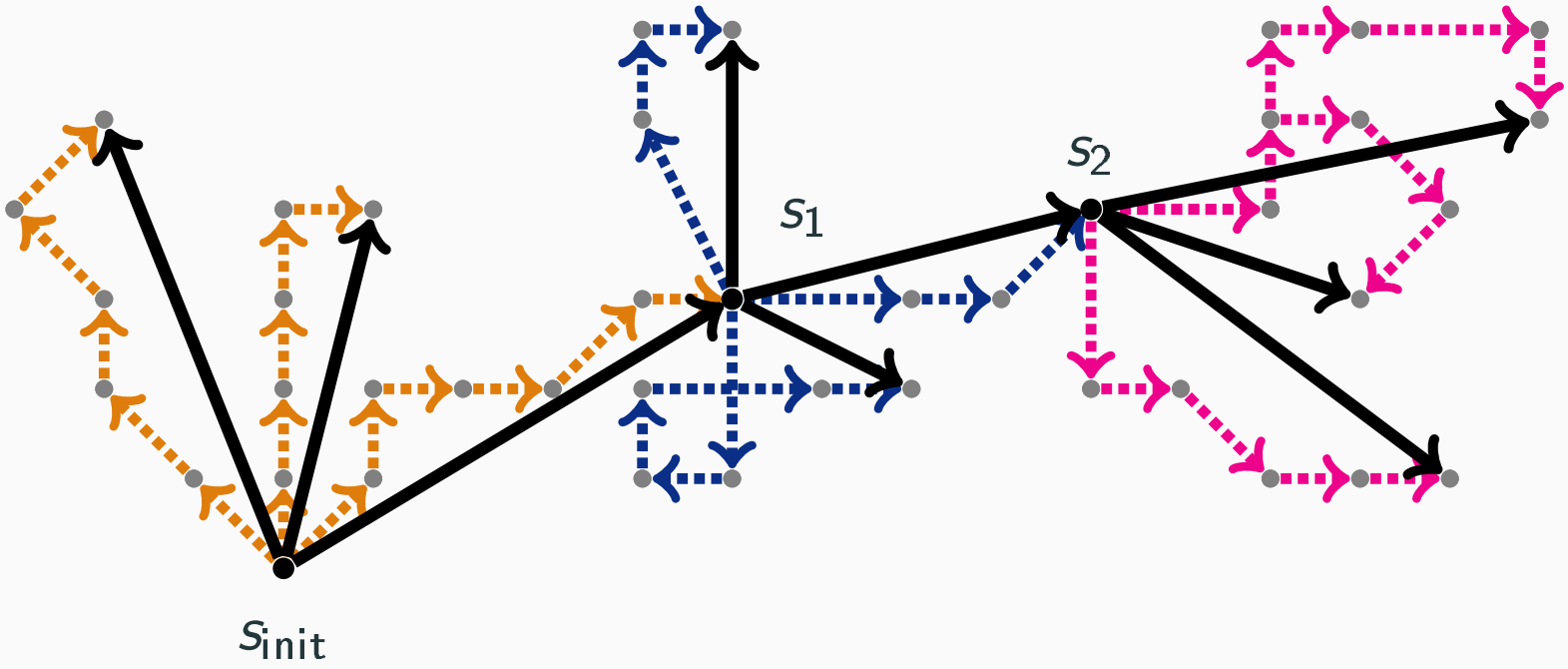
$\rightarrow$ Simulate by sampling segments instead of single reactions.



CRN: Segmental Simulation

Precompute k short trajectories (called segments) for each state.

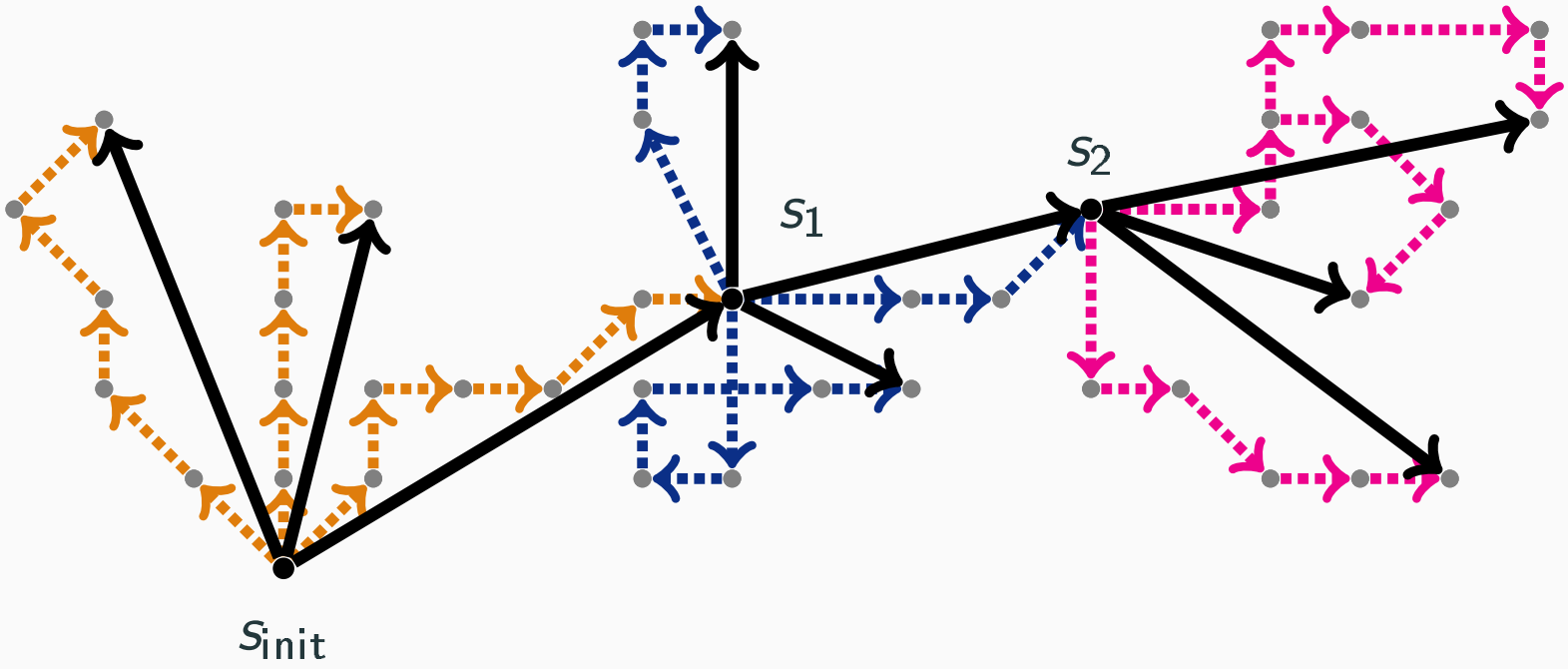
$\rightarrow$ Simulate by sampling segments instead of single reactions.



- much faster!

CRN: Segmental Simulation

Precompute k short trajectories (called segments) for each state.
$\rightarrow$ Simulate by sampling segments instead of single reactions.



- much faster!
- Problem: many states $\rightarrow$ too inefficient

CRN: Abstraction-Based Segmental Simulation

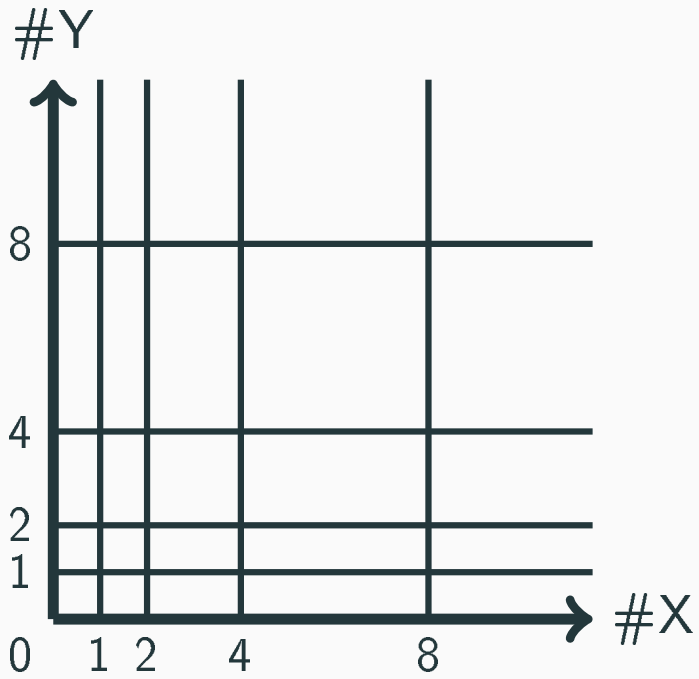
- Idea: Do not treat every state separately!

CRN: Abstraction-Based Segmental Simulation

- Idea: Do not treat every state separately!
- States with similar species counts have similar propensities
  $\rightarrow$ their behave similarly

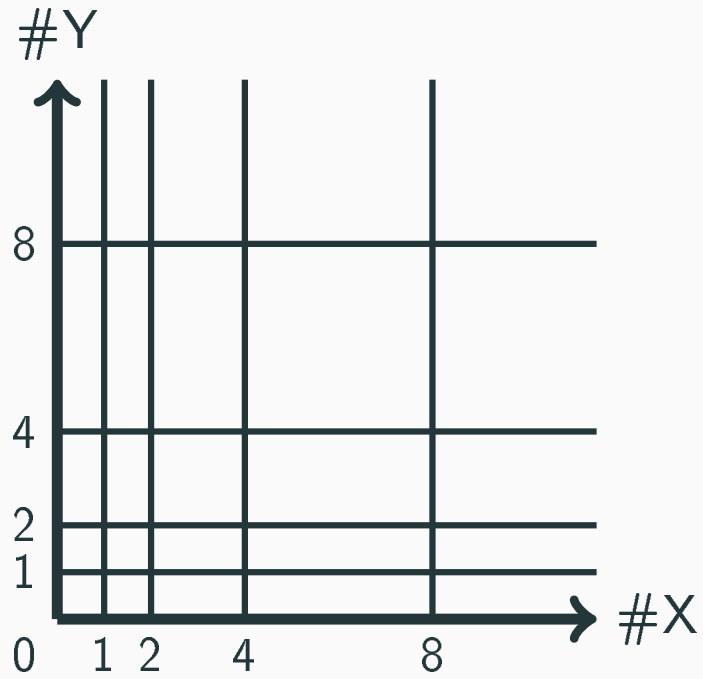
CRN: Abstraction-Based Segmental Simulation

- Idea: Do not treat every state separately!
- States with similar species counts have similar propensities
  $\rightarrow$ their behave similarly
- Population-level abstraction: split state-space into regions (called abstract states)



CRN: Abstraction-Based Segmental Simulation

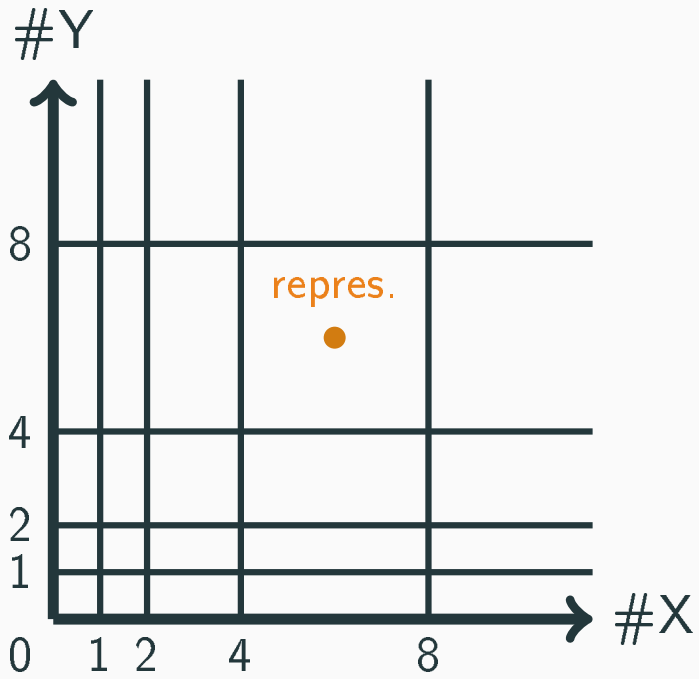
- Idea: Do not treat every state separately!
- States with similar species counts have similar propensities
  $\rightarrow$ their behave similarly
- Population-level abstraction: split state-space into regions (called abstract states)



- Choose representative for each abstract state (usually center)

CRN: Abstraction-Based Segmental Simulation

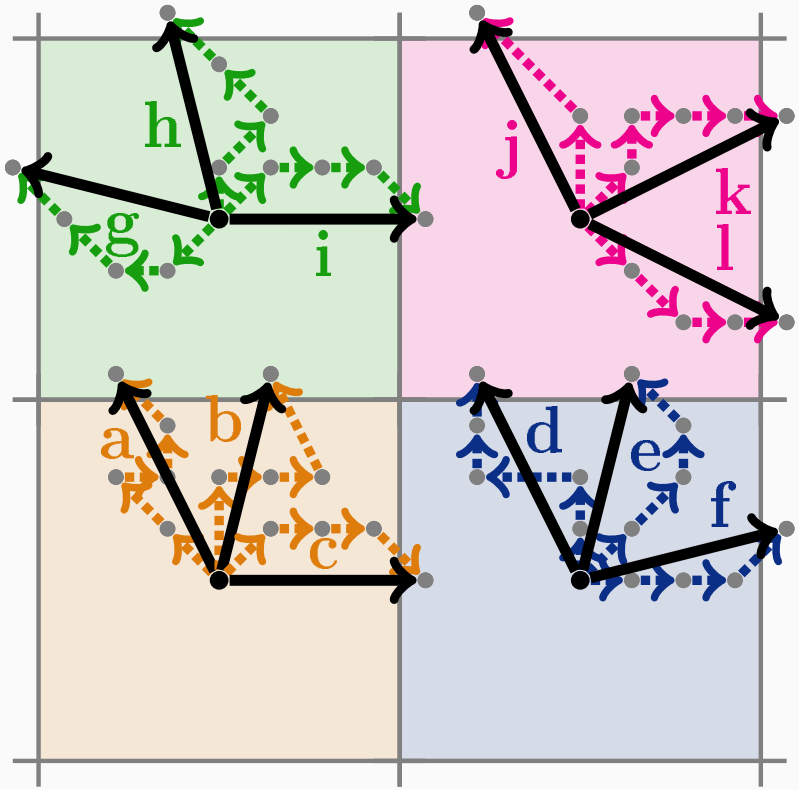
- Idea: Do not treat every state separately!
- States with similar species counts have similar propensities
  $\rightarrow$ their behave similarly
- Population-level abstraction: split state-space into regions (called abstract states)



- Choose representative for each abstract state (usually center)

CRN: Abstraction-Based Segmental Simulation

Only precompute k segments for each representative.

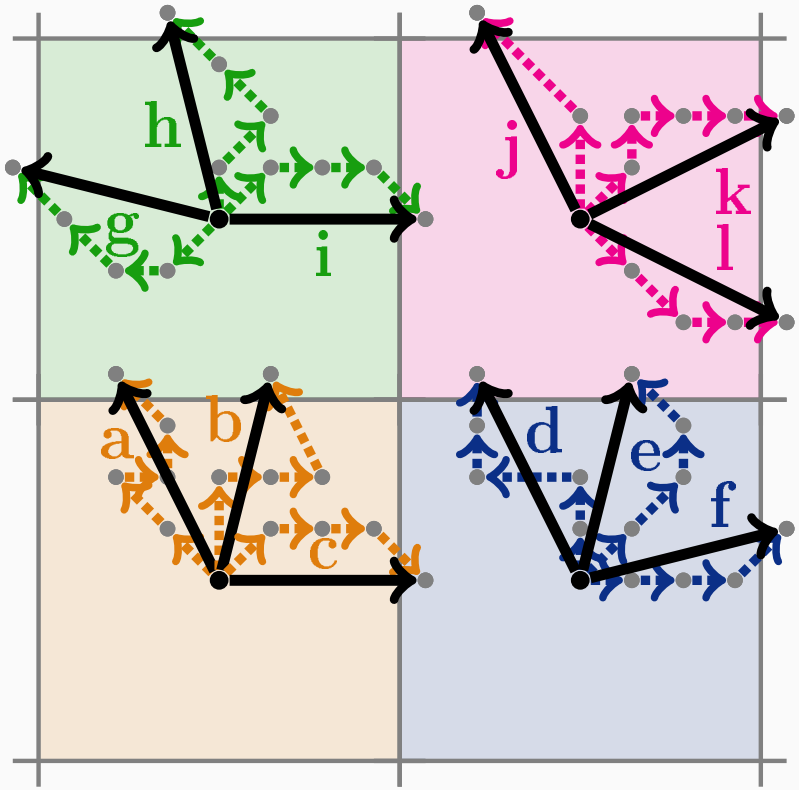


Segments end when they leave the abstract state.

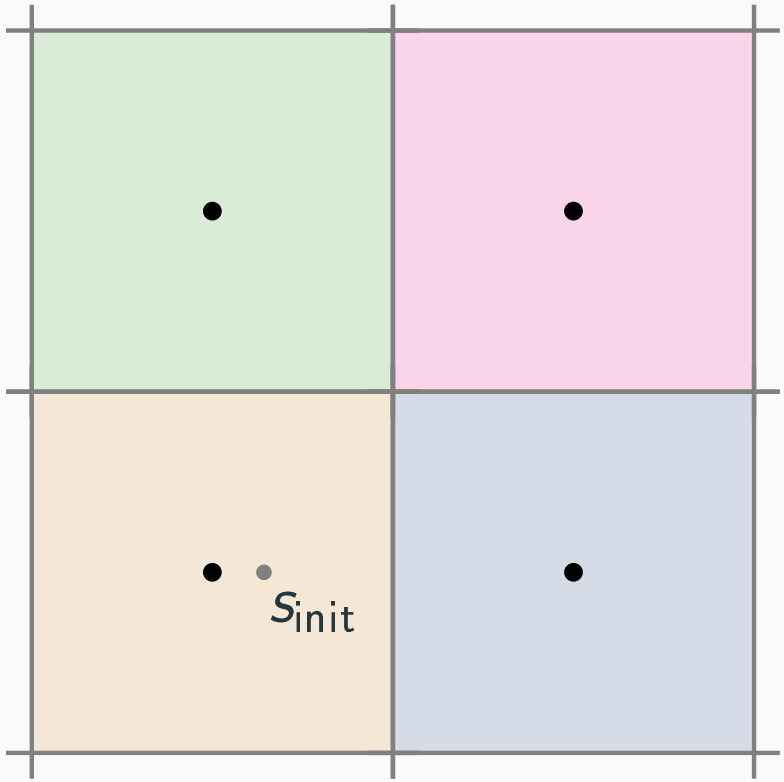
→ Intuition: "*significant change*"

CRN: Abstraction-Based Segmental Simulation

Only precompute k segments for each representative.



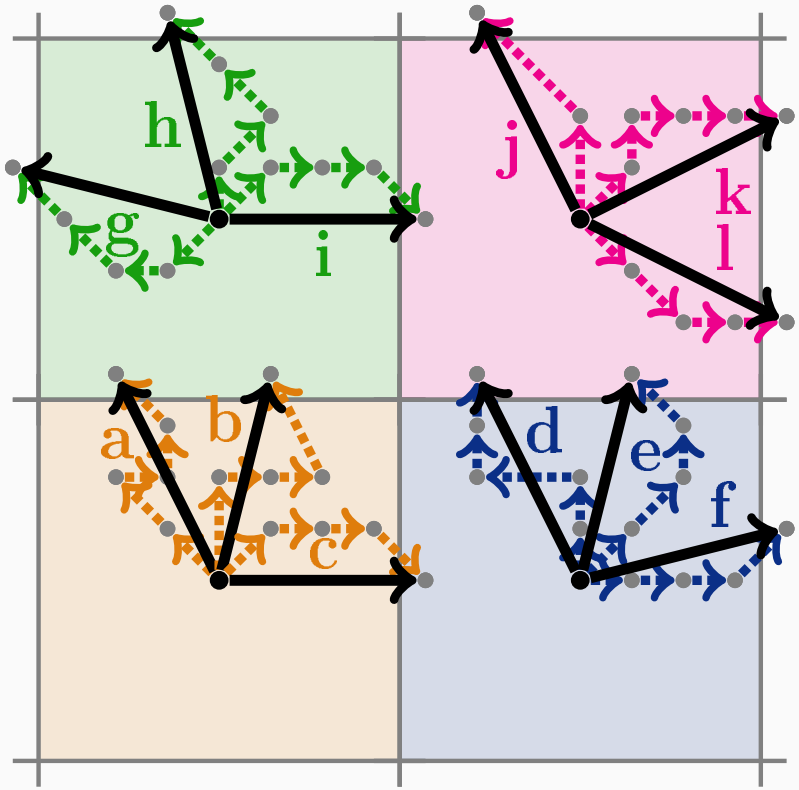
MEMORY
$(k = 3)$



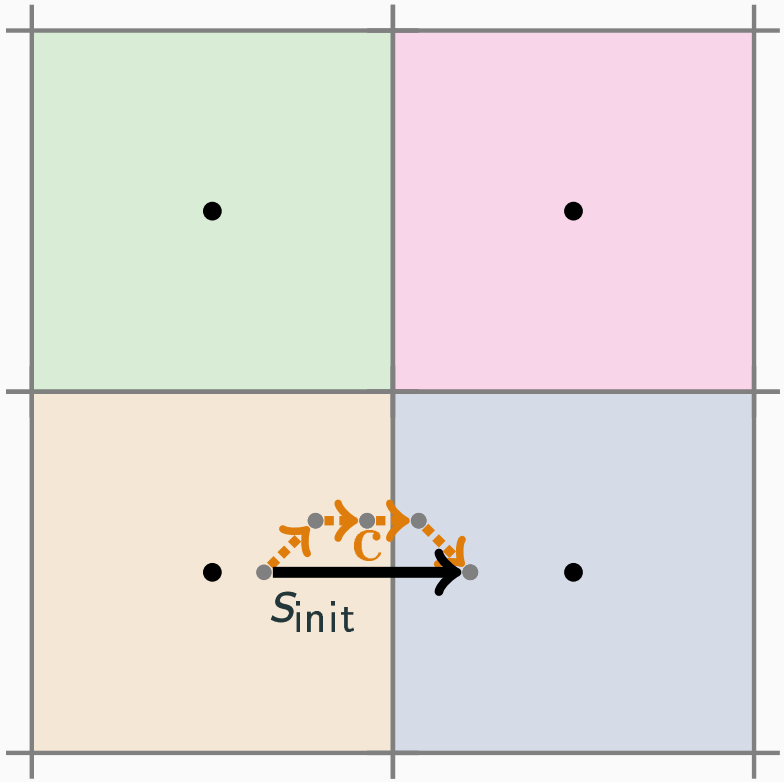
SIMULATION

CRN: Abstraction-Based Segmental Simulation

Only precompute k segments for each representative.



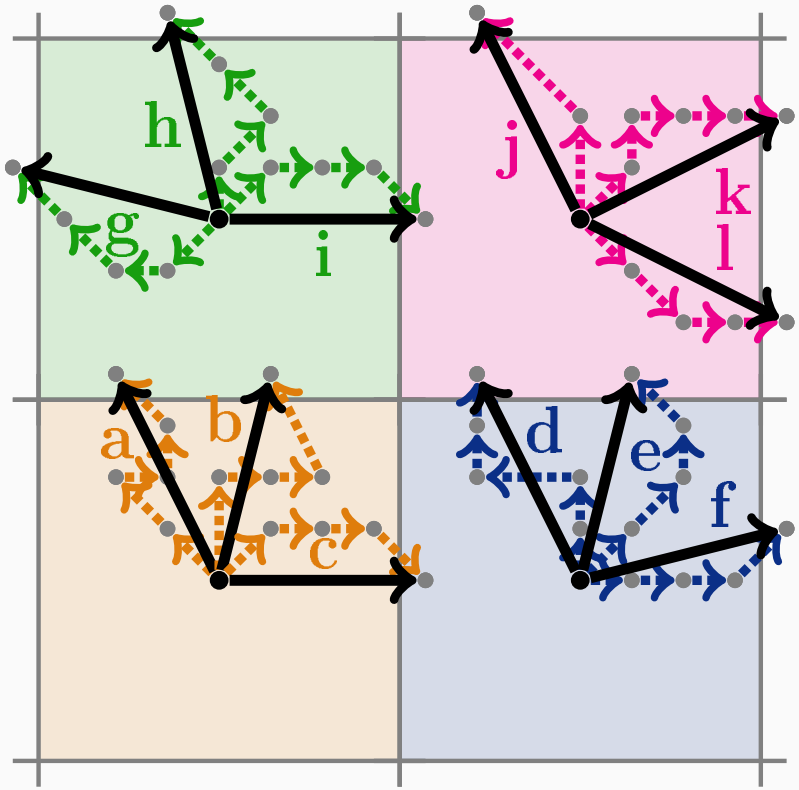
MEMORY
$(k = 3)$



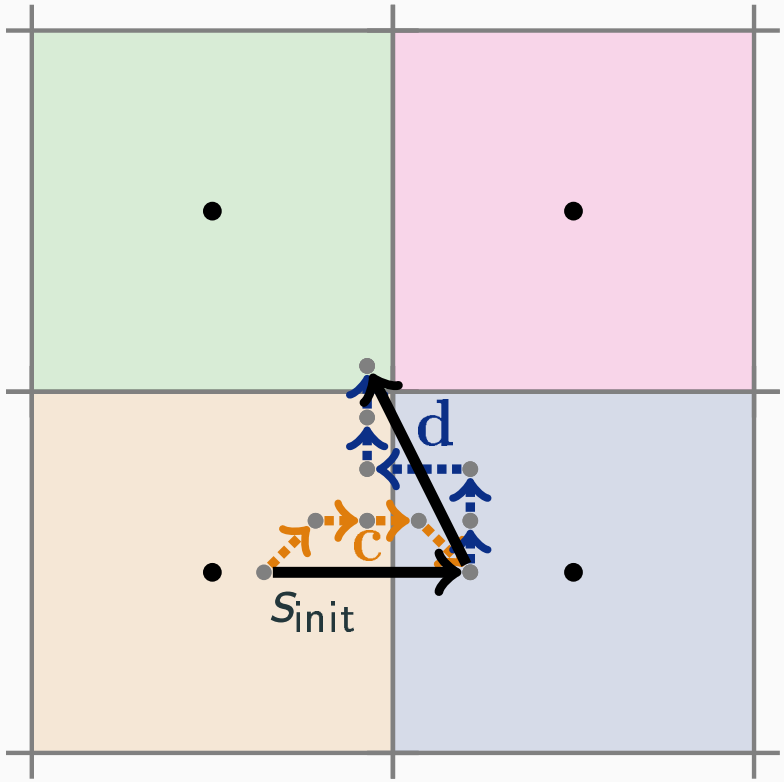
SIMULATION

CRN: Abstraction-Based Segmental Simulation

Only precompute k segments for each representative.



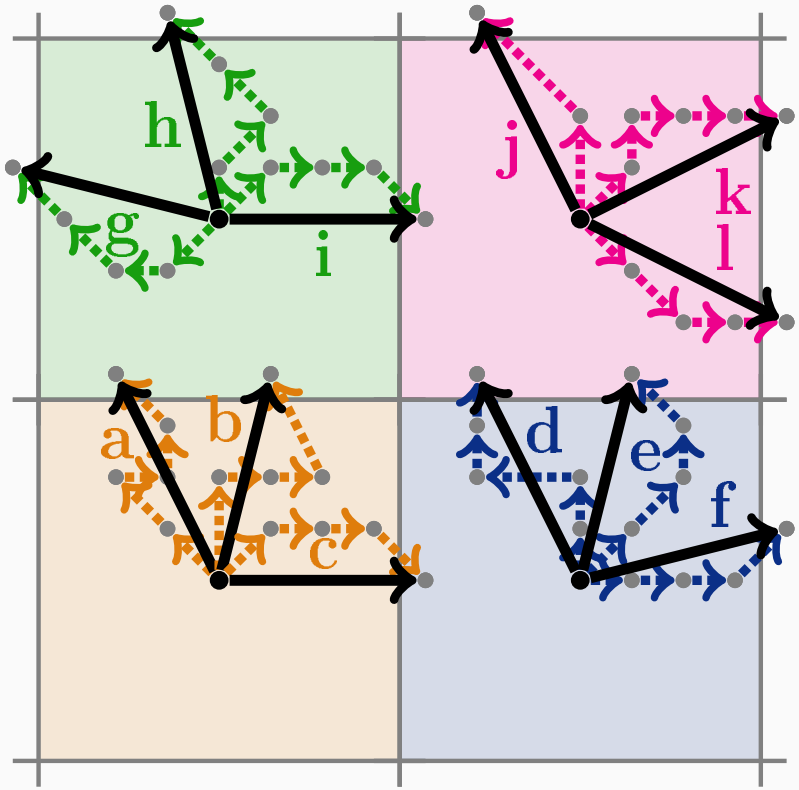
MEMORY
($k = 3$)



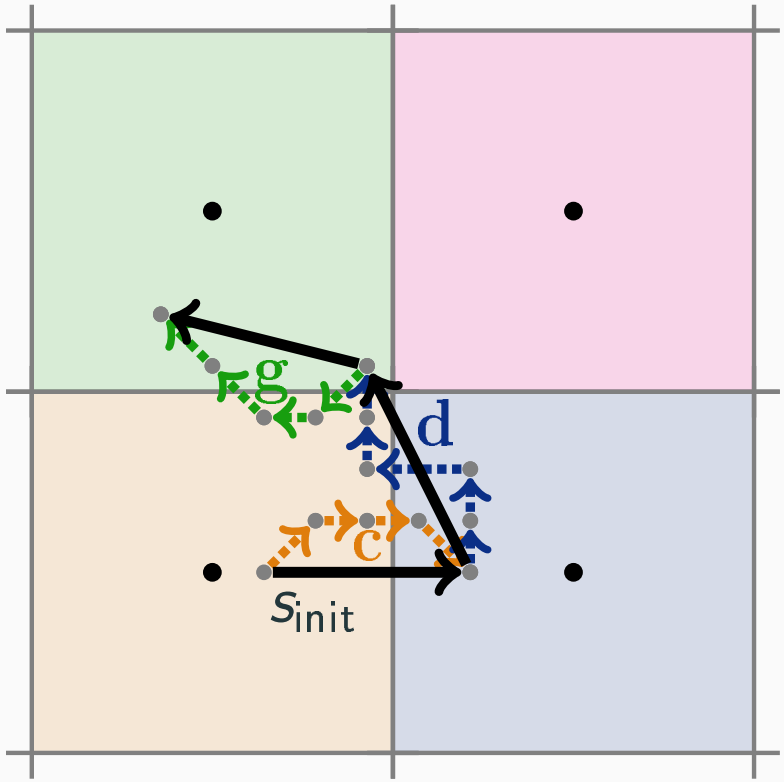
SIMULATION

CRN: Abstraction-Based Segmental Simulation

Only precompute k segments for each representative.



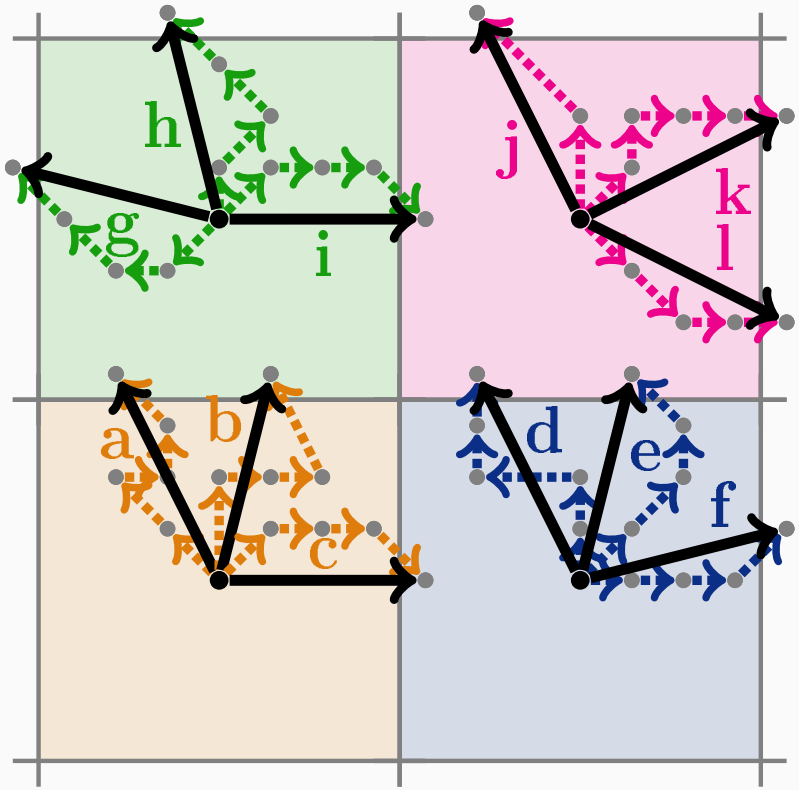
MEMORY
($k = 3$)



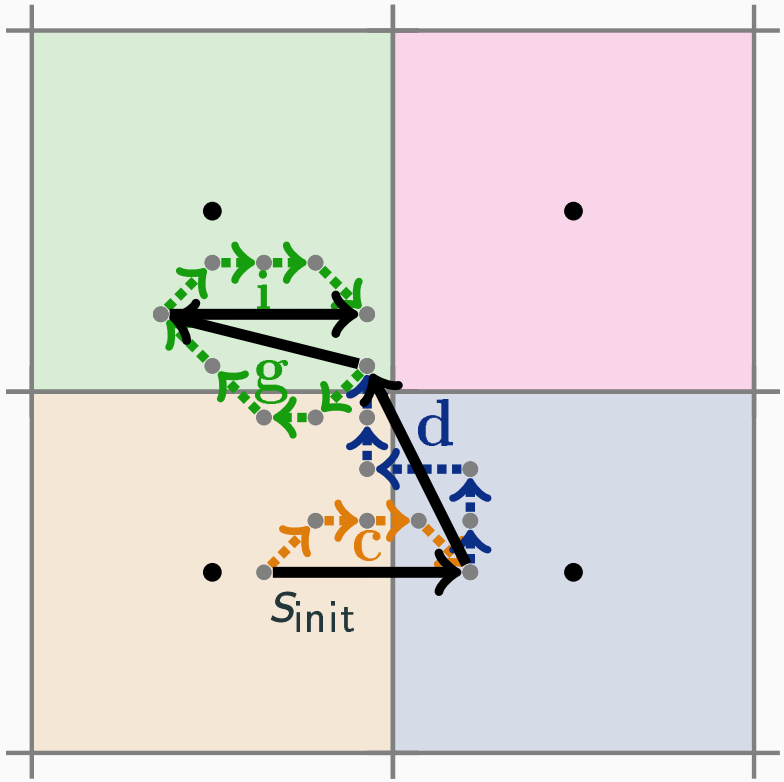
SIMULATION

CRN: Abstraction-Based Segmental Simulation

Only precompute k segments for each representative.



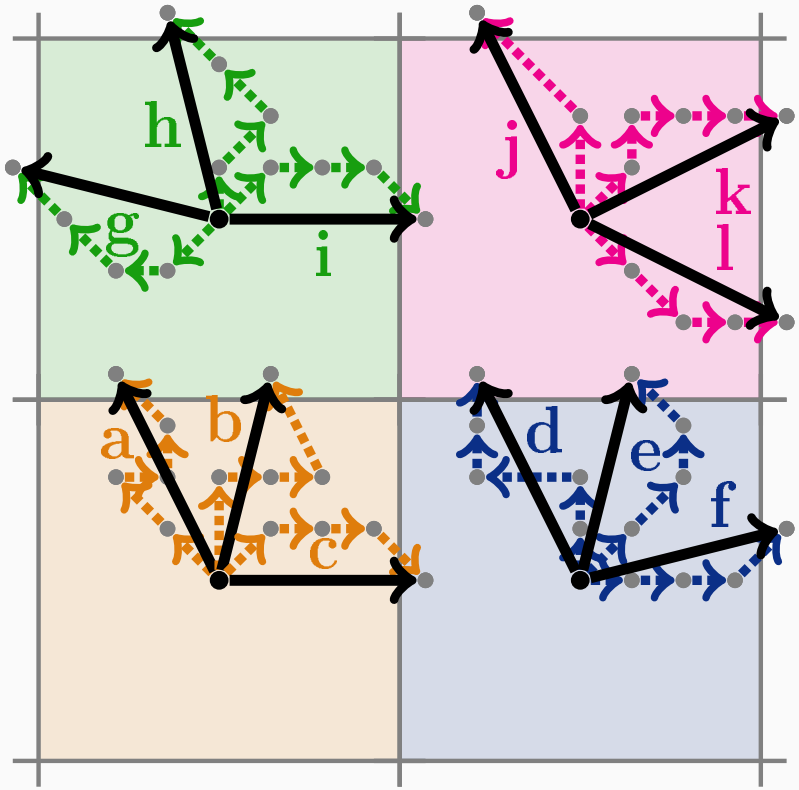
MEMORY
($k = 3$)



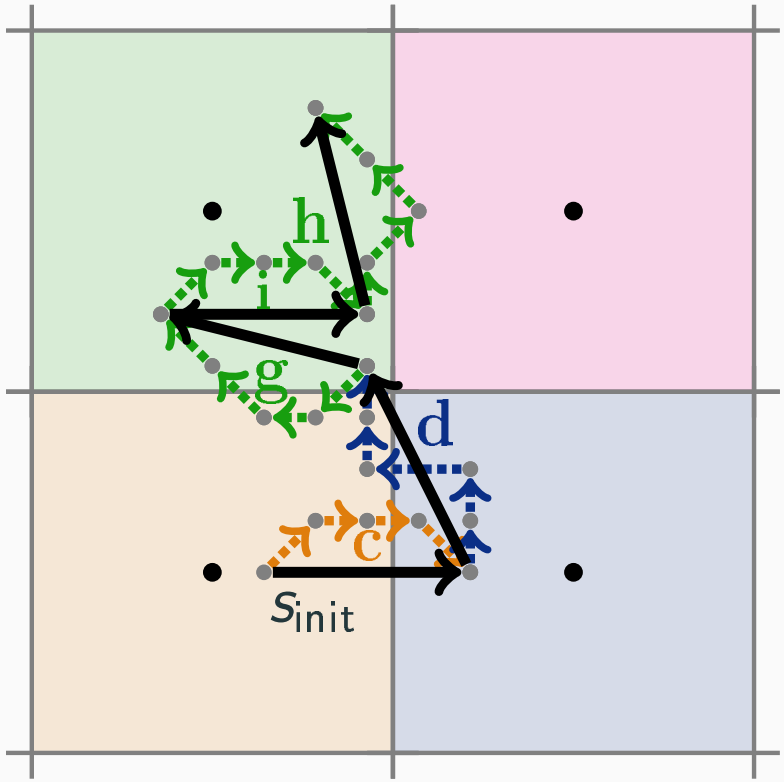
SIMULATION

CRN: Abstraction-Based Segmental Simulation

Only precompute k segments for each representative.



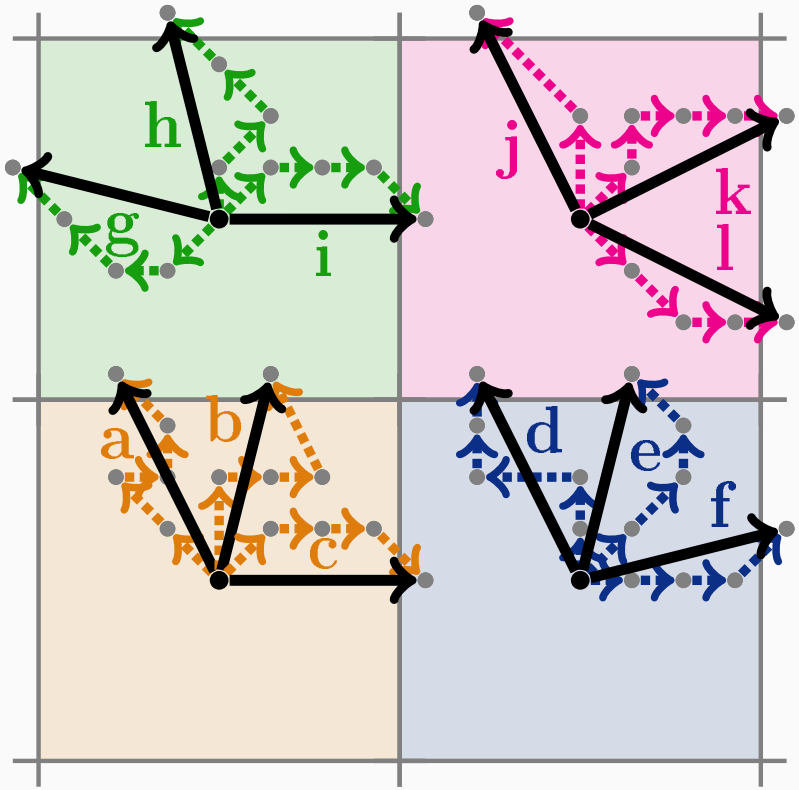
MEMORY
$(k = 3)$



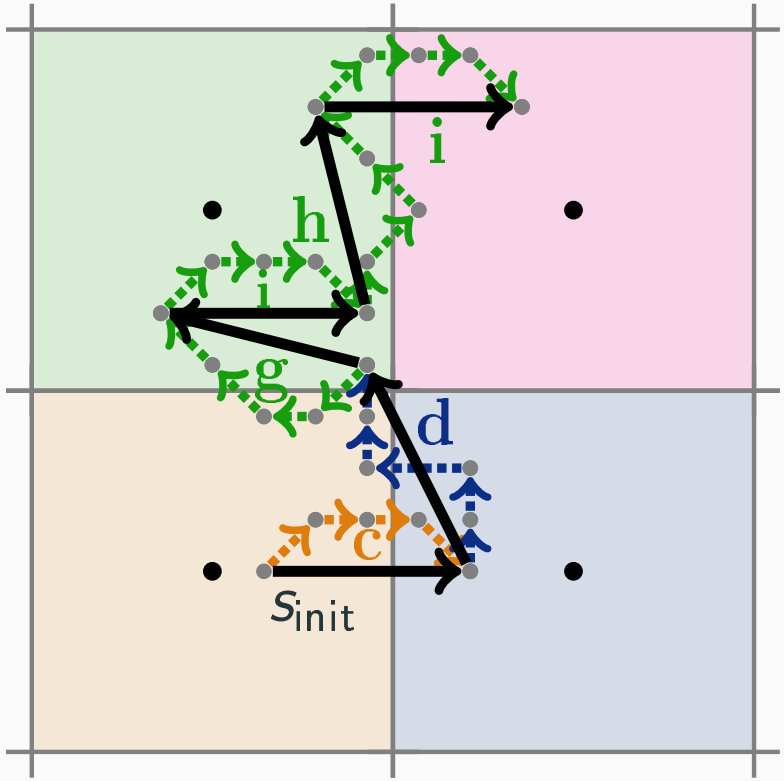
SIMULATION

CRN: Abstraction-Based Segmental Simulation

Only precompute k segments for each representative.



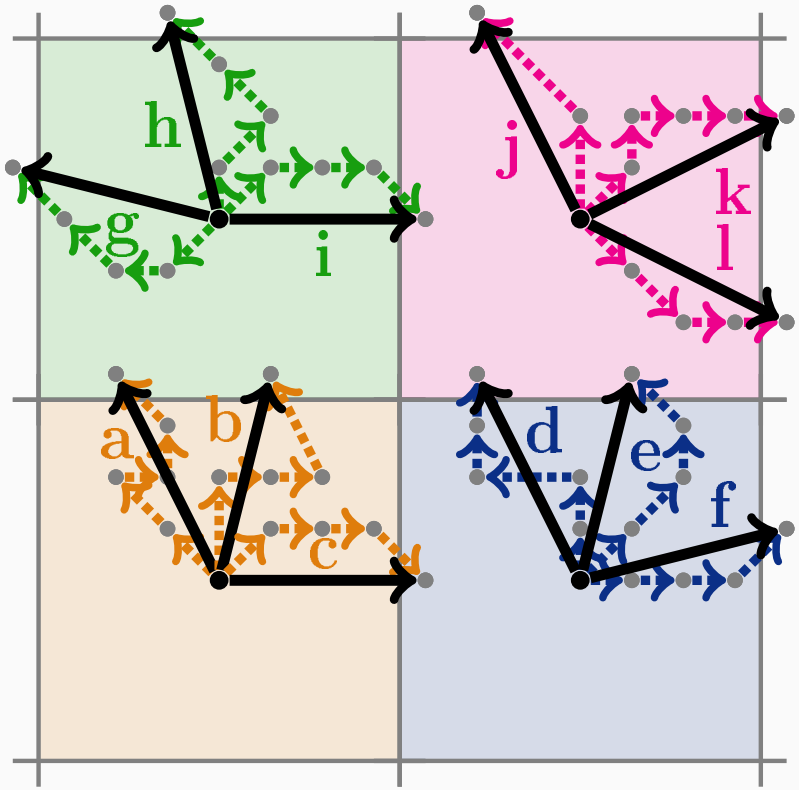
MEMORY
($k = 3$)



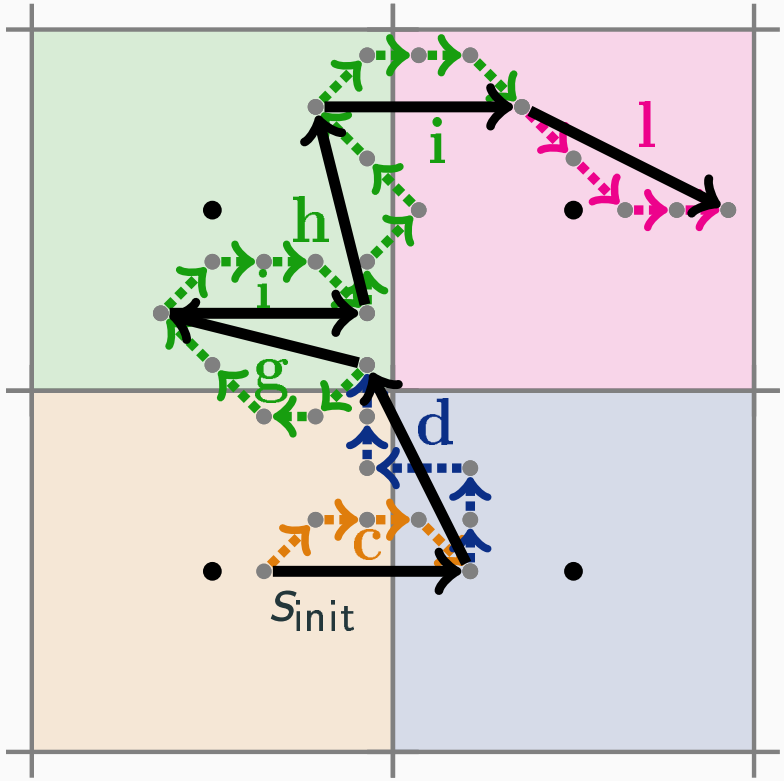
SIMULATION

CRN: Abstraction-Based Segmental Simulation

Only precompute k segments for each representative.



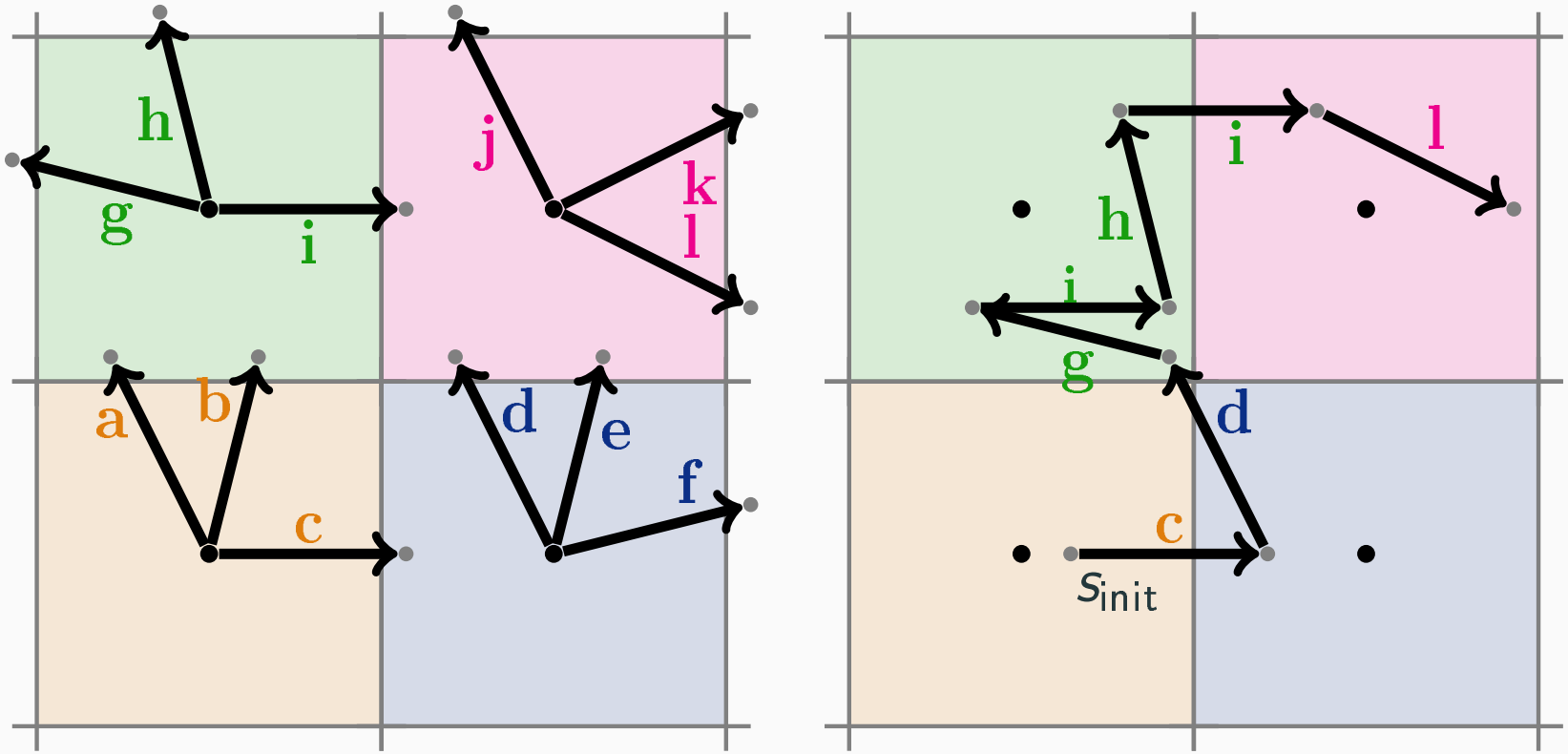
MEMORY
($k = 3$)



SIMULATION

CRN: Abstraction-Based Segmental Simulation

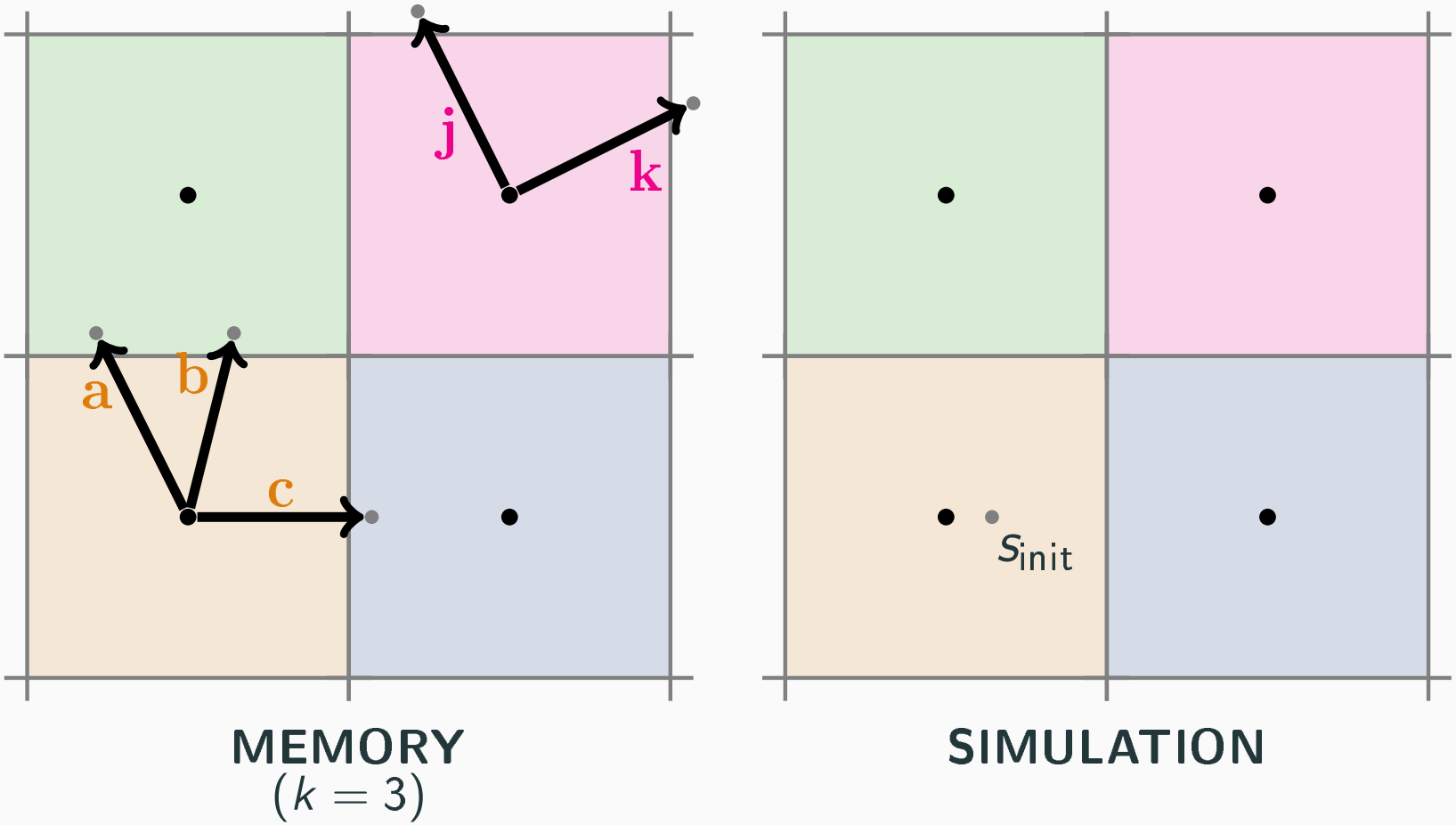
Only precompute k segments for each representative.



To save memory: Work with summaries instead of segments.

CRN: Abstraction-Based Segmental Simulation

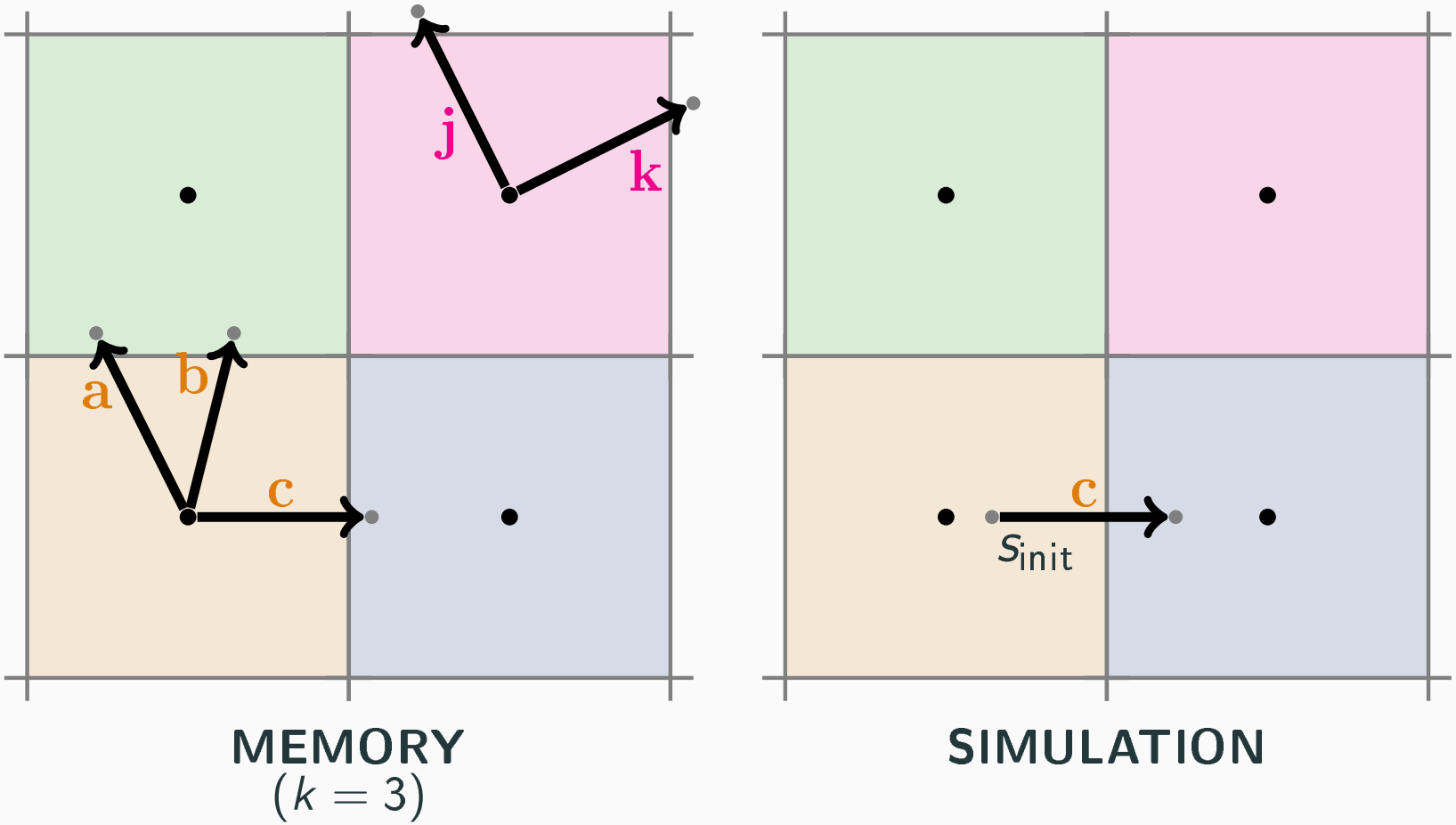
Only precompute k segments for each representative.



Lazy: Do not precompute but fill memory on-the-fly!

CRN: Abstraction-Based Segmental Simulation

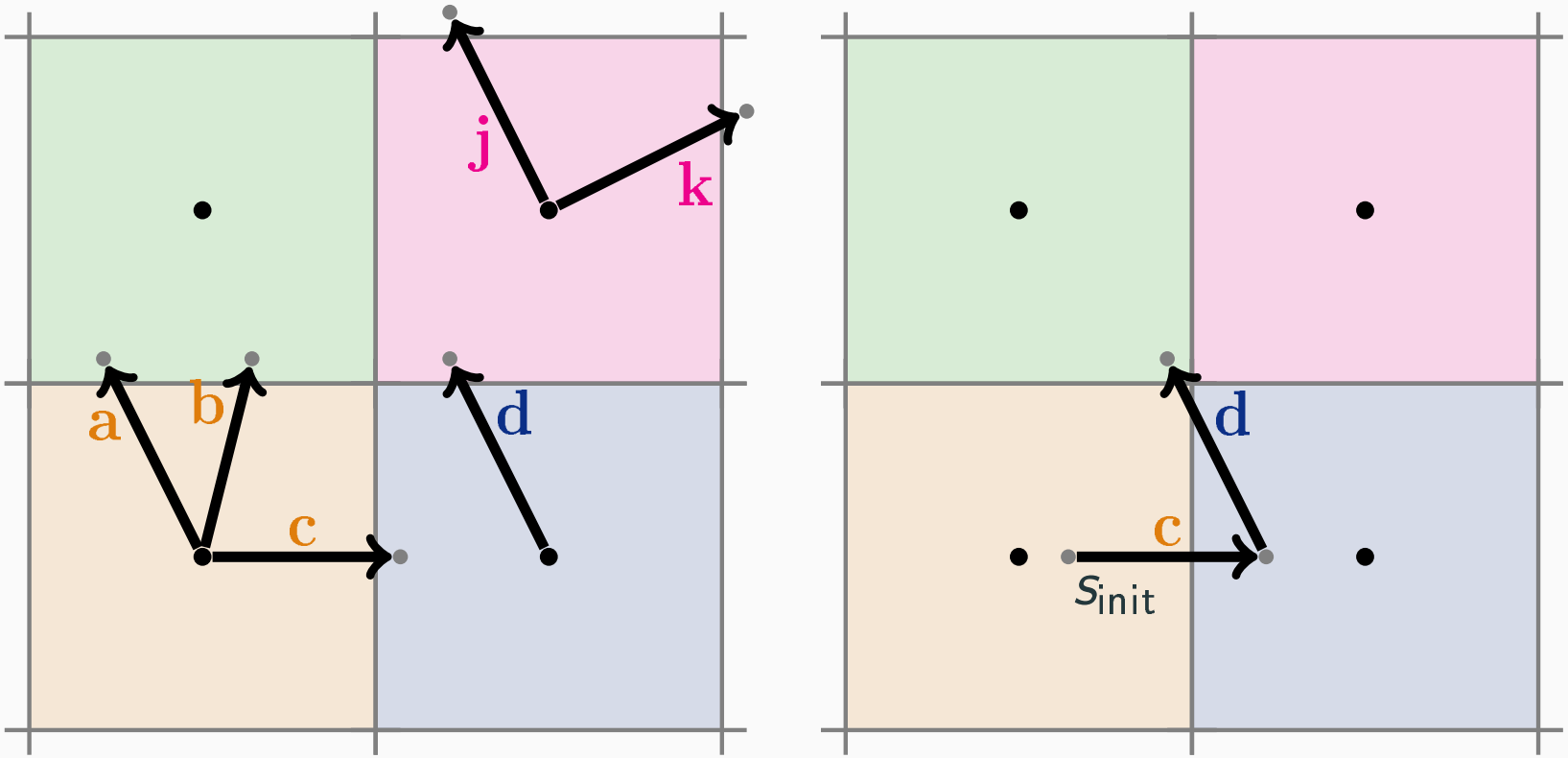
Only precompute k segments for each representative.



Lazy: Do not precompute but fill memory on-the-fly!

CRN: Abstraction-Based Segmental Simulation

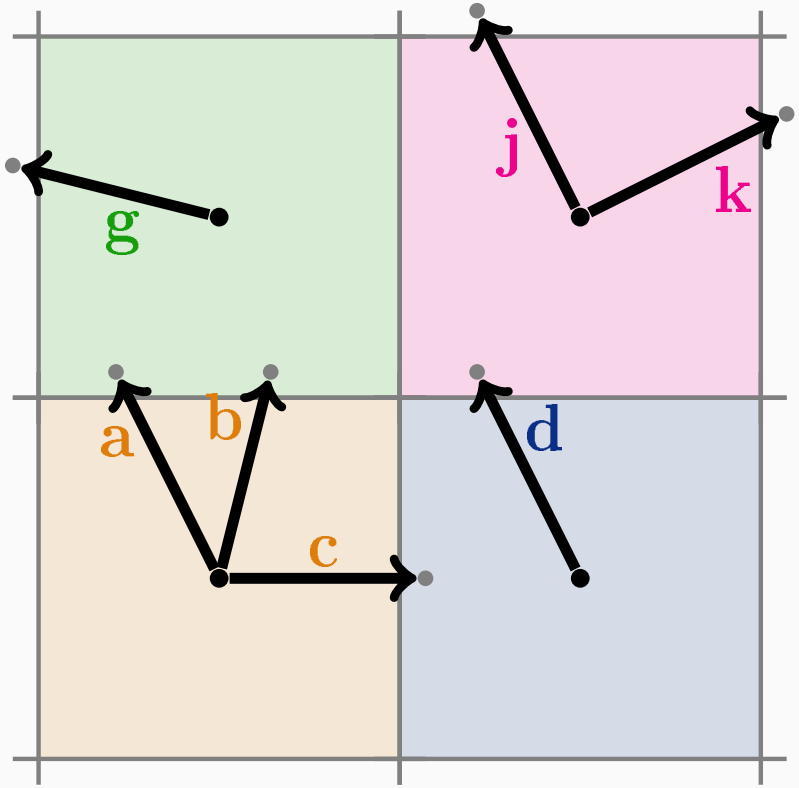
Only precompute k segments for each representative.



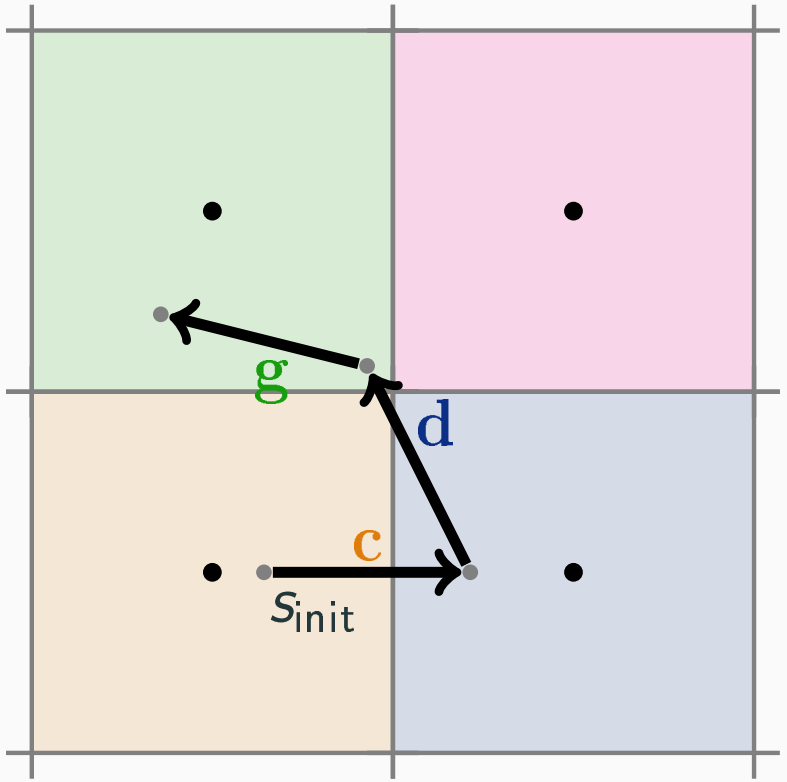
Lazy: Do not precompute but fill memory on-the-fly!

CRN: Abstraction-Based Segmental Simulation

Only precompute k segments for each representative.



MEMORY
($k = 3$)

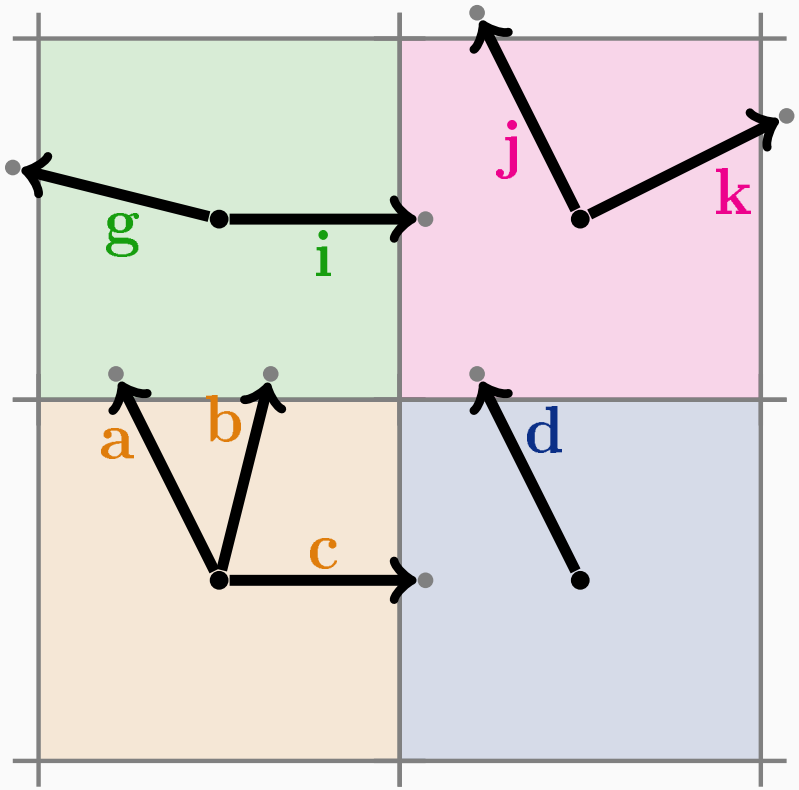


SIMULATION

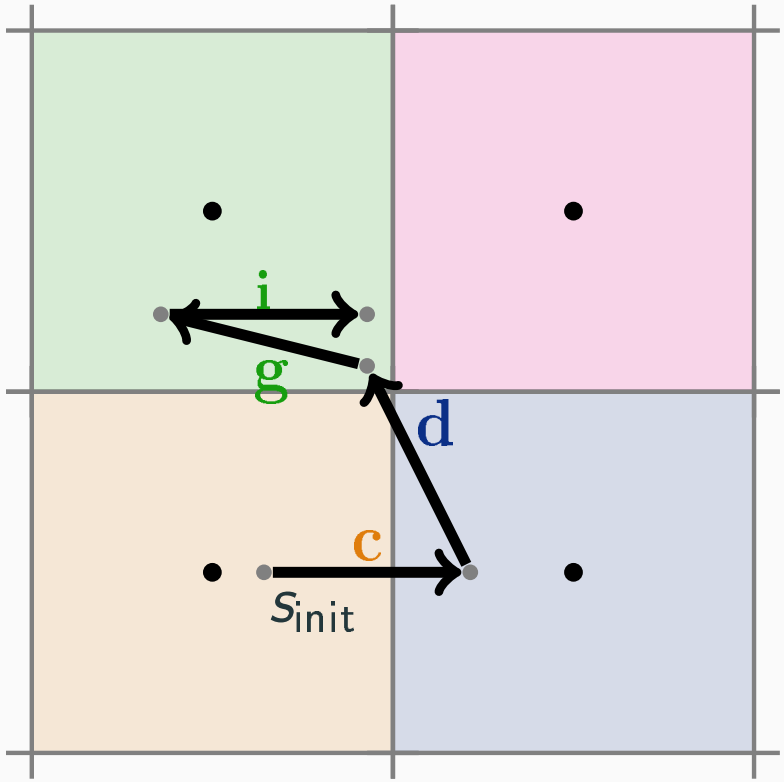
Lazy: Do not precompute but fill memory on-the-fly!

CRN: Abstraction-Based Segmental Simulation

Only precompute k segments for each representative.



MEMORY
($k = 3$)

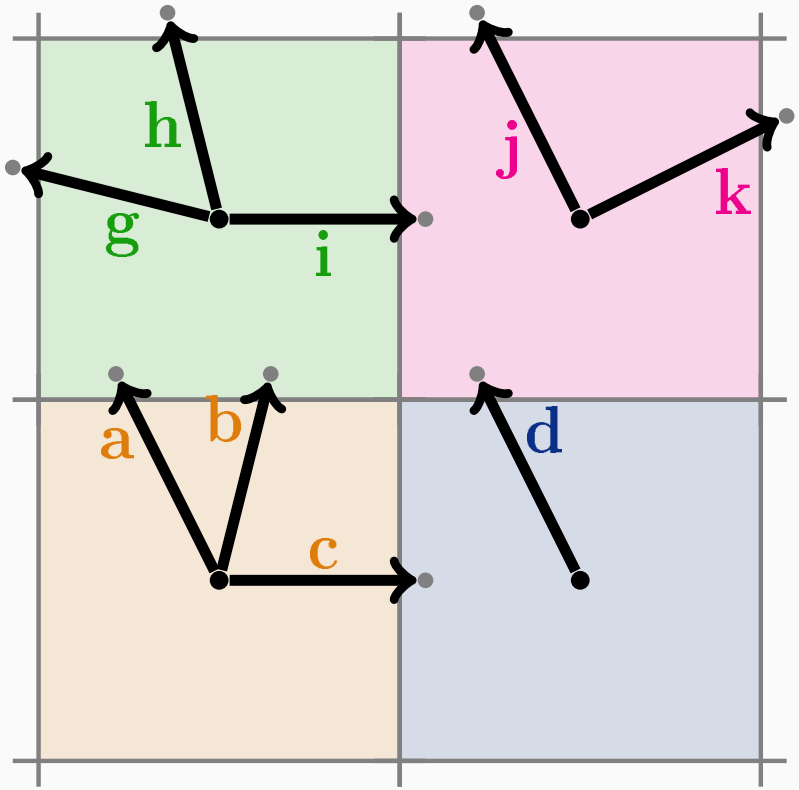


SIMULATION

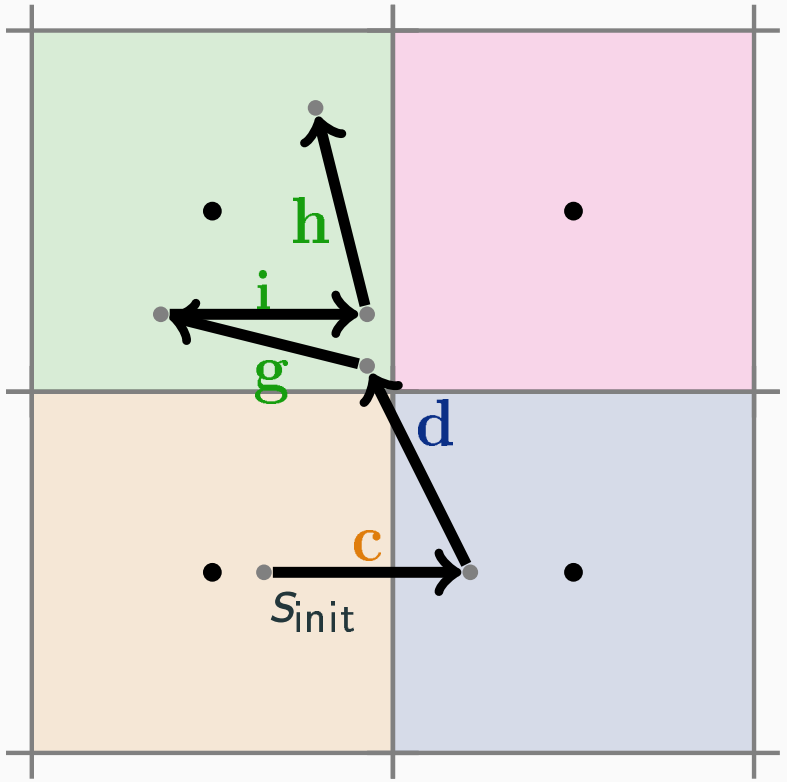
Lazy: Do not precompute but fill memory on-the-fly!

CRN: Abstraction-Based Segmental Simulation

Only precompute k segments for each representative.



MEMORY
($k = 3$)

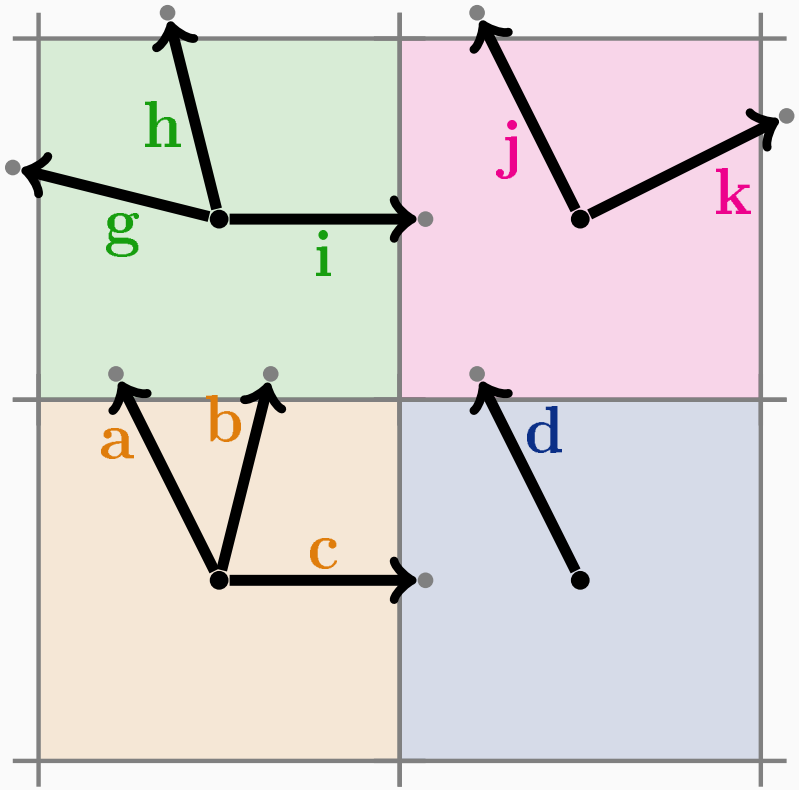


SIMULATION

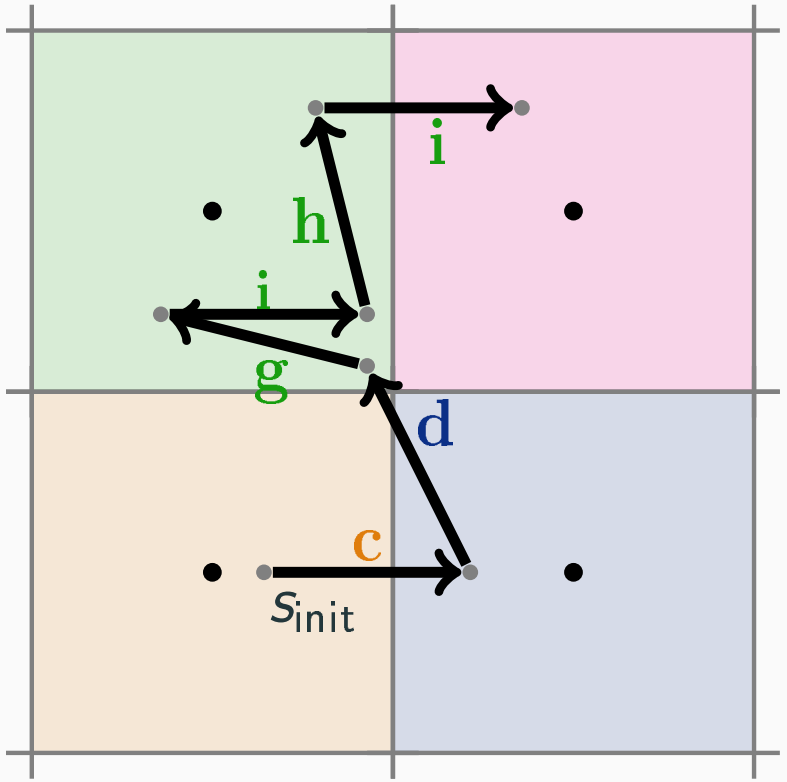
Lazy: Do not precompute but fill memory on-the-fly!

CRN: Abstraction-Based Segmental Simulation

Only precompute k segments for each representative.



MEMORY
($k = 3$)

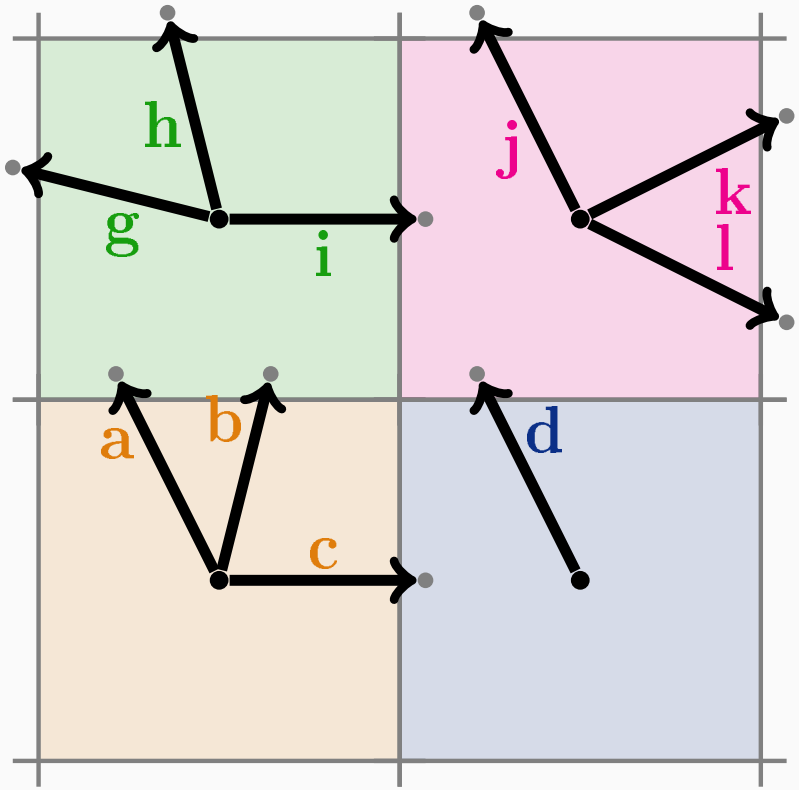


SIMULATION

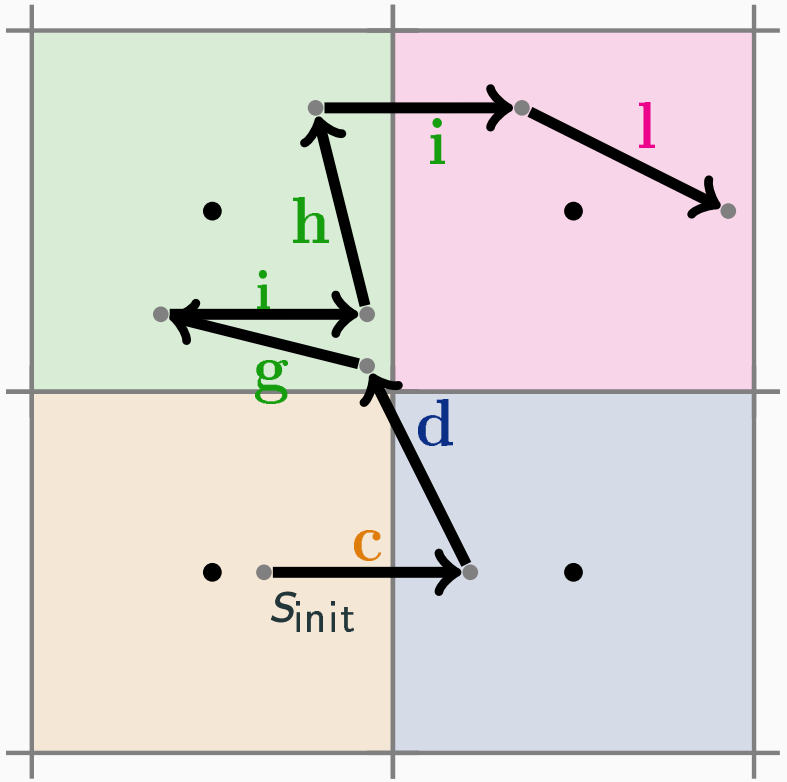
Lazy: Do not precompute but fill memory on-the-fly!

CRN: Abstraction-Based Segmental Simulation

Only precompute k segments for each representative.



MEMORY
($k = 3$)



SIMULATION

Lazy: Do not precompute but fill memory on-the-fly!

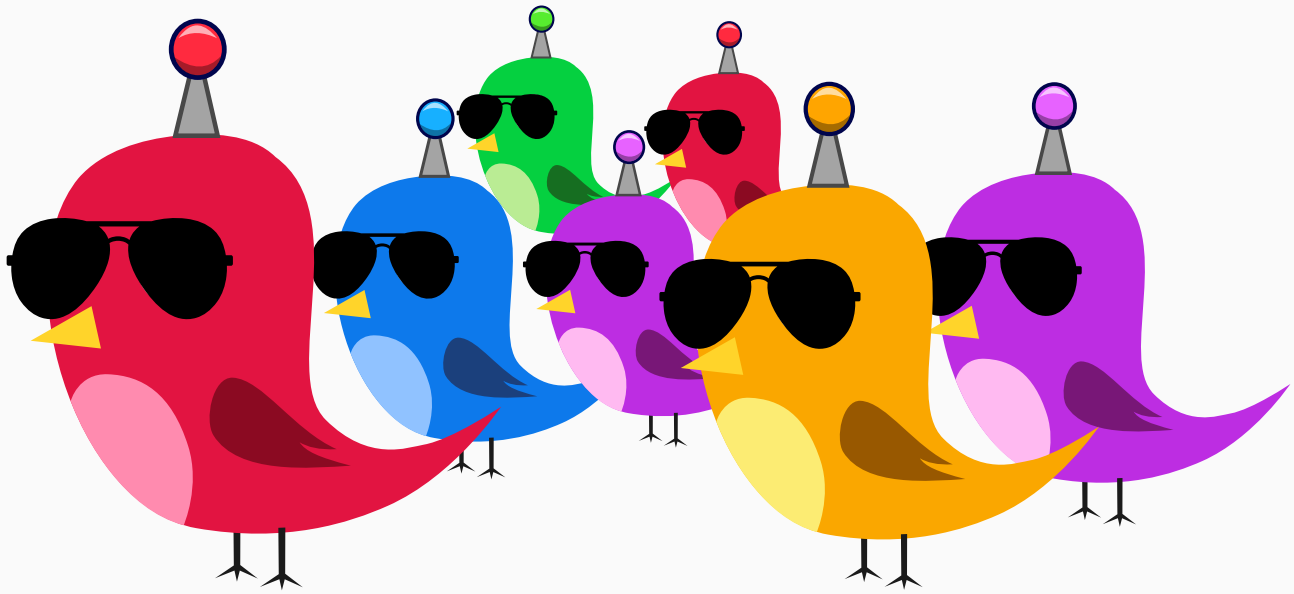
CRN: Conclusion

Segmental simulation:

- Reuses previous simulations to generate new ones
- Is an approximate simulation technique
- Accurately captures dynamics of most systems
- Speeds up transient analysis (up to 4000x faster)
- Efficiently predict the behavior of CRNs

Try it in SAQuaiA!

Thank you



THANK YOU!

My Publications i

First Author:

- Checking Qualitative Liveness Properties of Replicated Systems with Stochastic Scheduling (CAV'20) [3]
- Peregrine 2.0: Explaining Correctness of Population Protocols Through Stage Graphs (ATVA'20) [8]
- Abstraction-Based Segmental Simulation of Chemical Reaction Networks (CMSB'22) [10]

Non-first Author:

- Succinct Population Protocols for Presburger Arithmetic (STACS'20) [2]
- Fast and Succinct Population Protocols for Presburger Arithmetic (SAND'22) [5]

My Publications ii

Others (not part of thesis):

- Automata Tutor v3 (CAV20) [7]
- Decision Power of Weak Asynchronous Models of Distributed Computing (PODC'21) [4]
- Fast and succinct population protocols for Presburger arithmetic (Journal of Comp. and Sys. Sciences 2023) [6]

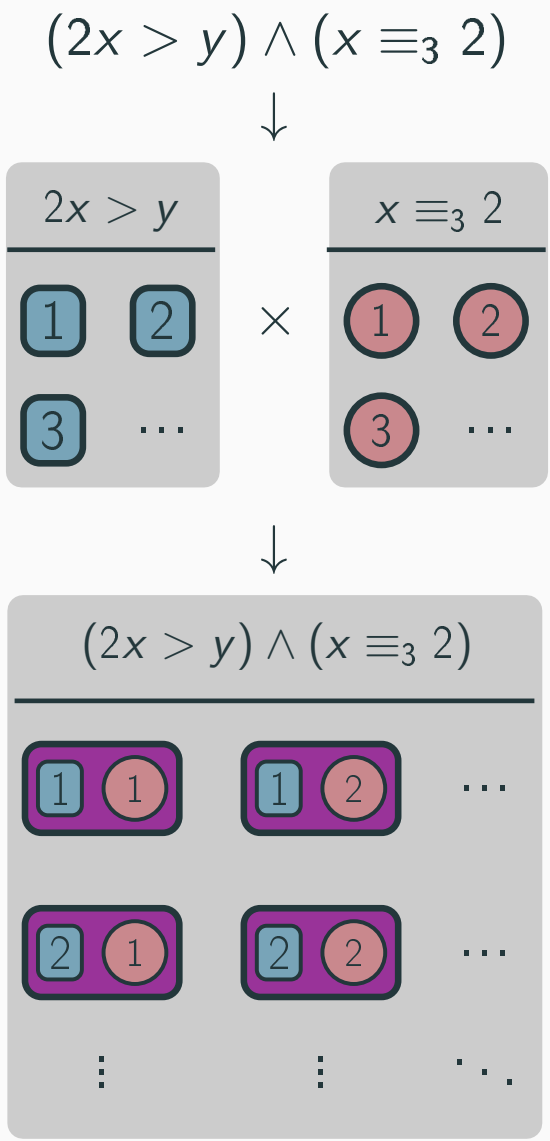
Population Protocol: Synthesis

Our result

For every quantifier-free Presburger formula φ there is an population protocol which

- has $\mathcal{O}(\mathrm{POLY}(|\varphi|))$ states,
  $\rightarrow$ succinct
- for n agents stabilizes in $\mathcal{O}(n^2 \log n)$ expected interactions, and
  $\rightarrow$ fast
- can be constructed efficiently.

Population Protocol: Synthesis (Fast) (by Angluin *et al.* [1])



Population Protocol: Synthesis Idea

Idea:

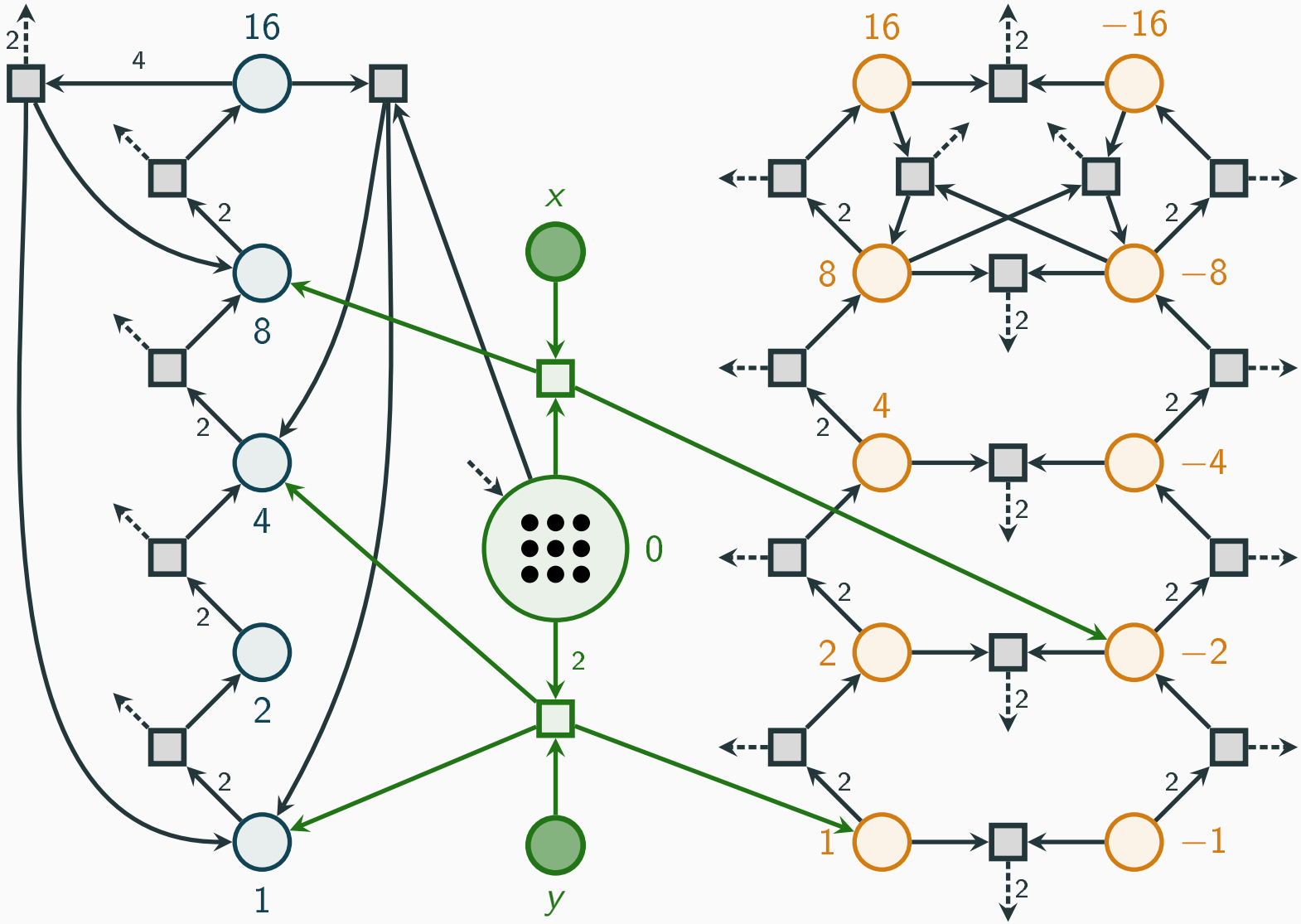
- Binary value representation
  (instead of unary)
- Agents participate in computation of only one atomic formula
  (instead of in all)

Population Computer

Like population protocol but easier to design because:

- k-way transitions
- Helper agents
- More flexible output definition

Population Protocol: Synthesis (Fast and Succinct)



$$8x + 5y \equiv_{11} 4 \quad \vee \quad y - 2x \geq 5$$

References i

[1] Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M.J., Peralta, R.: Computation in networks of passively mobile finite-state sensors. p. 290–299. PODC '04, Association for Computing Machinery, New York, NY, USA (2004). https://doi.org/10.1145/1011767.1011810

[2] Blondin, M., Esparza, J., Genest, B., Helfrich, M., Jaax, S.: Succinct Population Protocols for Presburger Arithmetic. In: Paul, C., Bläser, M. (eds.) 37th International Symposium on Theoretical Aspects of Computer Science (STACS 2020). Leibniz International Proceedings in Informatics (LIPIcs), vol. 154, pp. 40:1–40:15. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2020). https://doi.org/10.4230/LIPIcs.STACS.2020.40

[3] Blondin, M., Esparza, J., Helfrich, M., Kučera, A., Meyer, P.J.: Checking qualitative liveness properties of replicated systems with stochastic scheduling. In: Lahiri, S.K., Wang, C. (eds.) Computer Aided Verification. pp. 372–397. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-53291-8_20

References ii

[4] Czerner, P., Guttenberg, R., Helfrich, M., Esparza, J.: Decision power of weak asynchronous models of distributed computing. p. 115–125. PODC'21, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3465084.3467918, https://doi.org/10.1145/3465084.3467918

[5] Czerner, P., Guttenberg, R., Helfrich, M., Esparza, J.: Fast and Succinct Population Protocols for Presburger Arithmetic. In: Aspnes, J., Michail, O. (eds.) 1st Symposium on Algorithmic Foundations of Dynamic Networks (SAND 2022). Leibniz International Proceedings in Informatics (LIPIcs), vol. 221, pp. 11:1–11:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2022). https://doi.org/10.4230/LIPIcs.SAND.2022.11

[6] Czerner, P., Guttenberg, R., Helfrich, M., Esparza, J.: Fast and succinct population protocols for presburger arithmetic. Journal of Computer and System Sciences **140**, 103481 (2024). https://doi.org/https://doi.org/10.1016/j.jcss.2023.103481, https://www.sciencedirect.com/science/article/pii/S0022000023000867

References iii

[7] D'Antoni, L., Helfrich, M., Kretinsky, J., Ramneantu, E., Weininger, M.: Automata tutor v3. In: Lahiri, S.K., Wang, C. (eds.) Computer Aided Verification. pp. 3–14. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-53291-8_1

[8] Esparza, J., Helfrich, M., Jaax, S., Meyer, P.J.: Peregrine 2.0: Explaining correctness of population protocols through stage graphs. In: Hung, D.V., Sokolsky, O. (eds.) Automated Technology for Verification and Analysis. pp. 550–556. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-59152-6_32

[9] Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. The Journal of Physical Chemistry **81**(25), 2340–2361 (12 1977). https://doi.org/10.1021/j100540a008

[10] Helfrich, M., Češka, M., Křetínský, J., Martiček, Š.: Abstraction-based segmental simulation of chemical reaction networks. In: Petre, I., Păun, A. (eds.) Computational Methods in Systems Biology. pp. 41–60. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-15034-0_3