Lower Bounds on the State Complexity of Population Protocols

Philipp Czerner, Javier Esparza
Department of Informatics, TU Munich

June 23, 2021

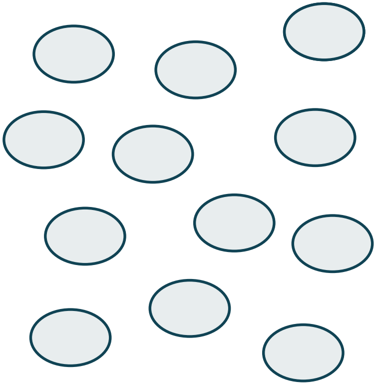
Introduction

The Setting

- population protocols

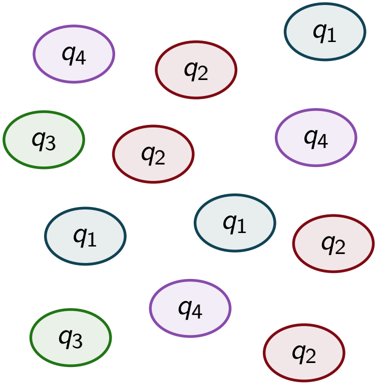
The Setting

- population protocols
- population of *agents*



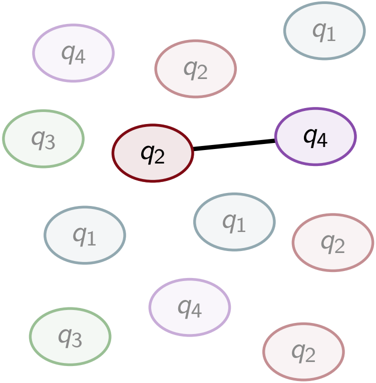
The Setting

- population protocols
- population of *agents*
- agents are finite-state machines



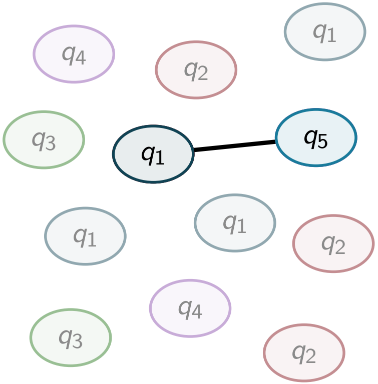
The Setting

- population protocols
- population of *agents*
- agents are finite-state machines
- random interactions



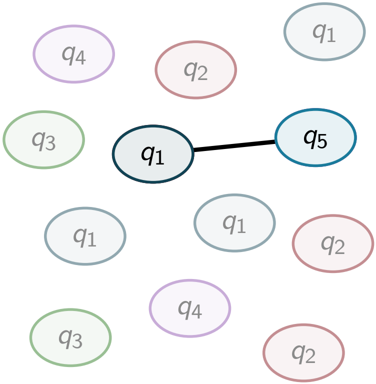
The Setting

- population protocols
- population of *agents*
- agents are finite-state machines
- random interactions



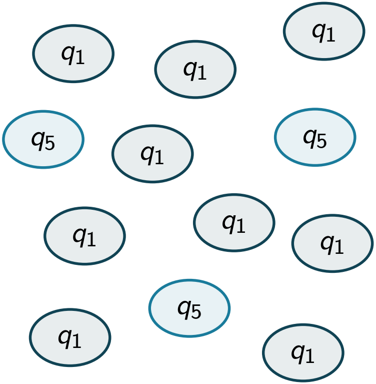
The Setting

- population protocols
- population of *agents*
- agents are finite-state machines
- random interactions
- want to decide if initial configuration satisfies a property



The Setting

- population protocols
- population of *agents*
- agents are finite-state machines
- random interactions
- want to decide if initial configuration satisfies a property
  - computation by *stable consensus*



Population Protocols

- finite set of states Q

Population Protocols

- finite set of states Q
- pairwise transitions $T : Q^2 \to Q^2$

Population Protocols

- finite set of states Q
- pairwise transitions $T : Q^2 \to Q^2$
- initial state $q_0 \in Q$

Population Protocols

- finite set of states Q
- pairwise transitions $T : Q^2 \to Q^2$
- initial state $q_0 \in Q$
- output function $O : Q \to \{0, 1\}$

Population Protocols

- finite set of states Q
- pairwise transitions $T : Q^2 \to Q^2$
- initial state $q_0 \in Q$
- output function $O : Q \to \{0, 1\}$
- compute exactly semi-linear (or Presburger) predicates

Population Protocols

- finite set of states Q
- pairwise transitions $T : Q^2 \to Q^2$
- initial state $q_0 \in Q$
- output function $O : Q \to \{0, 1\}$
- compute exactly semi-linear (or Presburger) predicates
- can compute threshold: $x \geq k$, for any $k \in \mathbb{N}$

The question:

The question:

How many states do we need for $x \geq k$,
if k grows large?

State Complexity
of Population Protocols

- Two natural questions: time complexity and space complexity

- Two natural questions: time complexity and space complexity
    - time: how long until consensus is achieved?

- Two natural questions: time complexity and space complexity
  - time: how long until consensus is achieved?
  - space: how much memory is needed?

- Two natural questions: time complexity and space complexity
  - time: how long until consensus is achieved?
  - space: how much memory is needed?
- Time complexity is well studied

- Two natural questions: time complexity and space complexity
  - time: how long until consensus is achieved?
  - space: how much memory is needed?
- Time complexity is well studied
  - upper bounds [Angluin, Aspnes, Eisenstat 2008], [Draief, Vojnović 2012], [Kosowski, Uznański 2018] ...

- Two natural questions: time complexity and space complexity
  - time: how long until consensus is achieved?
  - space: how much memory is needed?
- Time complexity is well studied
  - upper bounds [Angluin, Aspnes, Eisenstat 2008], [Draief, Vojnović 2012], [Kosowski, Uznański 2018] ...
  - lower bounds [Doty, Soloveichik 2015], [Alistarh *et al.* 2017], [Alistarh, Aspnes, Gelashvili 2018]

- Two natural questions: time complexity and space complexity
  - time: how long until consensus is achieved?
  - space: how much memory is needed?
- Time complexity is well studied
  - upper bounds [Angluin, Aspnes, Eisenstat 2008], [Draief, Vojnović 2012], [Kosowski, Uznański 2018] ...
  - lower bounds [Doty, Soloveichik 2015], [Alistarh *et al.* 2017], [Alistarh, Aspnes, Gelashvili 2018]
- Only little is known about space

- Two natural questions: time complexity and space complexity
  - time: how long until consensus is achieved?
  - space: how much memory is needed?
- Time complexity is well studied
  - upper bounds [Angluin, Aspnes, Eisenstat 2008], [Draief, Vojnović 2012], [Kosowski, Uznański 2018] ...
  - lower bounds [Doty, Soloveichik 2015], [Alistarh *et al.* 2017], [Alistarh, Aspnes, Gelashvili 2018]
- Only little is known about space
  - upper bounds [Blondin, Esparza, Jaax 2018], [Blondin *et al.* 2020]

- Two natural questions: time complexity and space complexity
  - time: how long until consensus is achieved?
  - space: how much memory is needed?
- Time complexity is well studied
  - upper bounds [Angluin, Aspnes, Eisenstat 2008], [Draief, Vojnović 2012], [Kosowski, Uznański 2018] ...
  - lower bounds [Doty, Soloveichik 2015], [Alistarh *et al.* 2017], [Alistarh, Aspnes, Gelashvili 2018]
- Only little is known about space
  - upper bounds [Blondin, Esparza, Jaax 2018], [Blondin *et al.* 2020]
  - no lower bounds!

Remark

- A different model allows states Q to grow with number of agents n

Remark

- A different model allows states Q to grow with number of agents n
- Lots of research into time-space tradeoffs in that model

Remark

- A different model allows states Q to grow with number of agents n
- Lots of research into time-space tradeoffs in that model
- This considers growth of Q w.r.t. n, whereas we consider growth of Q w.r.t. the size of the predicate

Remark

- A different model allows states Q to grow with number of agents n
- Lots of research into time-space tradeoffs in that model
- This considers growth of Q w.r.t. n, whereas we consider growth of Q w.r.t. the size of the predicate
- For us, Q remains fixed independent of n

Measures

- To simplify, consider only threshold predicates $x \geq k$

Measures

- To simplify, consider only threshold predicates $x \geq k$
- We investigate lower bounds on state complexity:

"How many states for $x \geq k$?"

Measures

- To simplify, consider only threshold predicates $x \geq k$
- We investigate lower bounds on state complexity:

  "How many states for $x \geq k$?"
- Conversely: upper bounds on

  "Largest k for given number of states?"

Measures

- To simplify, consider only threshold predicates $x \geq k$
- We investigate lower bounds on state complexity:

  "How many states for $x \geq k$?"
- Conversely: upper bounds on

  "Largest k for given number of states?"
- Busy beaver function for population protocols!

Results

Results

Prior results due to [Blondin, Esparza, Jaax 2018].

Leaderless:

- $k \in \Omega(2^{|Q|})$

Results

Prior results due to [Blondin, Esparza, Jaax 2018].

Leaderless:

- $k \in \Omega(2^{|Q|})$

With Leader:

- $k \in \Omega(2^{2^{|Q|}})$

Results

Prior results due to [Blondin, Esparza, Jaax 2018].

Leaderless:

- $k \in \Omega(2^{|Q|})$

our result:

- $k \in \mathcal{O}(2^{2^{2^{|Q|}}})$

With Leader:

- $k \in \Omega(2^{2^{|Q|}})$

Results

Prior results due to [Blondin, Esparza, Jaax 2018].

Leaderless:

- $k \in \Omega(2^{|Q|})$

our result:

- $k \in \mathcal{O}(2^{2^{2^{|Q|}}})$

With Leader:

- $k \in \Omega(2^{2^{|Q|}})$

our result:

- $k \in \mathcal{O}(\mathrm{ack}(|Q|))$

Results

Prior results due to [Blondin, Esparza, Jaax 2018].

Leaderless:

- $k \in \Omega(2^{|Q|})$

our result:

- $k \in \mathcal{O}(2^{2^{2^{|Q|}}})$

↑
this talk

With Leader:

- $k \in \Omega(2^{2^{|Q|}})$

our result:

- $k \in \mathcal{O}(\mathrm{ack}(|Q|))$

Results

Prior results due to [Blondin, Esparza, Jaax 2018].

Leaderless:

- $k \in \Omega(2^{|Q|})$

our result:

- $k \in \mathcal{O}(2^{2^{2^{|Q|}}})$

$\uparrow$
this talk

With Leader:

- $k \in \Omega(2^{2^{|Q|}})$

our result:

- $k \in \mathcal{O}(\mathsf{ack}(|Q|))$

Terms in this talk are only correct up to the number of exponents.

A bound for leaderless protocols

Overview

Techniques used:

Overview

Techniques used:

1. "Pumping": adding additional agents to a rejecting run, s.t. it remains rejecting

Overview

Techniques used:

1. "Pumping": adding additional agents to a rejecting run, s.t. it remains rejecting
   - Uses results from the theory of Vector Addition Systems

Overview

Techniques used:

1. "Pumping": adding additional agents to a rejecting run, s.t. it remains rejecting
   - Uses results from the theory of Vector Addition Systems
2. Allowing the protocol to temporarily borrow agents, s.t. the reachability relation is an integer linear program

Overview

Techniques used:

1. "Pumping": adding additional agents to a rejecting run, s.t. it remains rejecting
   - Uses results from the theory of Vector Addition Systems
2. Allowing the protocol to temporarily borrow agents, s.t. the reachability relation is an integer linear program
3. Showing that long sequences of runs admit linear combinations with certain properties

Overview

Techniques used:

1. “Pumping”: adding additional agents to a rejecting run, s.t. it remains rejecting
   - Uses results from the theory of Vector Addition Systems
2. Allowing the protocol to temporarily borrow agents, s.t. the reachability relation is an integer linear program
3. Showing that long sequences of runs admit linear combinations with certain properties
   - Purely mathematical result, based on linear algebra

Stable Consensuses

Configuration $C : Q \to \mathbb{N}$.

Stable Consensuses

Configuration $C : Q \to \mathbb{N}$.

- C is a 0-*consensus*, if only rejecting states occur

Stable Consensuses

Configuration $C : Q \to \mathbb{N}$.

- C is a 0-*consensus*, if only rejecting states occur
- C is a *stable* 0-*consensus*, if C reaches only 0-consensuses

Stable Consensuses

Configuration $C : Q \to \mathbb{N}$.

- C is a 0-*consensus*, if only rejecting states occur
- C is a *stable* 0-*consensus*, if C reaches only 0-consensuses

The protocol rejects **iff** it reaches a stable 0-consensus.

Extending Stable Consensuses

Known:[1] If C is a stable 0-consensus, and $C(q) \geq 2^{2^{|Q|}}$, then $C + q$ is a stable 0-consensus.

[1] Follows from Rackoff's Theorem [Rackoff 1978]

Extending Stable Consensuses

Known:[1] If C is a stable 0-consensus, and $C(q) \geq 2^{2^{|Q|}}$, then $C + q$ is a stable 0-consensus.

($C + q$ is the configuration C with an additional agent in state q)

[1] Follows from Rackoff's Theorem [Rackoff 1978]

Extending Stable Consensuses

Known:[1] If C is a stable 0-consensus, and $C(q) \geq 2^{2^{|Q|}}$, then $C + q$ is a stable 0-consensus.

($C + q$ is the configuration C with an additional agent in state q)

Our goal: modify a rejecting run and "smuggle" additional agents from the initial state q_0 to a q with $C(q) > 2^{2^{|Q|}}$

[1] Follows from Rackoff's Theorem [Rackoff 1978]

Pumping

Goal:

- Find a run ending in stable 0-consensus C

Pumping

Goal:

- Find a run ending in stable 0-consensus C
- Let $S := \{q : C(q) \geq 2^{2^{|Q|}}\}$ denote large components of C; we call S *colour* of C

Pumping

Goal:

- Find a run ending in stable 0-consensus C
- Let $S := \{q : C(q) \geq 2^{2^{|Q|}}\}$ denote large components of C; we call S *colour* of C
- Find a way to move agents from initial state q_0 to S ("extension")

Pumping

Goal:

- Find a run ending in stable 0-consensus C
- Let $S := \{q : C(q) \geq 2^{2^{|Q|}}\}$ denote large components of C; we call S *colour* of C
- Find a way to move agents from initial state q_0 to S ("extension")
  - i.e. a sequence of transitions

Pumping

Goal:

- Find a run ending in stable 0-consensus C
- Let $S := \{q : C(q) \geq 2^{2^{|Q|}}\}$ denote large components of C; we call S *colour* of C
- Find a way to move agents from initial state q_0 to S (”extension“)
  - i.e. a sequence of transitions
- Too hard!

Borrowing

- To find an extension, allow “borrowing”

Borrowing

- To find an extension, allow "borrowing"
  - Transitions may cause the number of agents in a state to go negative, temporarily

Borrowing

- To find an extension, allow “borrowing”
  - Transitions may cause the number of agents in a state to go negative, temporarily
- Finding an extension is now a linear program

Borrowing

- To find an extension, allow "borrowing"
  - Transitions may cause the number of agents in a state to go negative, temporarily
- Finding an extension is now a linear program
- We show: 2^{2^j} runs partitioned into j colours yield a solution to the LP for some colour

Borrowing

- To find an extension, allow "borrowing"
  - Transitions may cause the number of agents in a state to go negative, temporarily
- Finding an extension is now a linear program
- We show: 2^{2^j} runs partitioned into j colours yield a solution to the LP for some colour
  - Only works for leaderless protocols; linear combination of runs

Borrowing

- To find an extension, allow "borrowing"
  - Transitions may cause the number of agents in a state to go negative, temporarily
- Finding an extension is now a linear program
- We show: 2^{2^j} runs partitioned into j colours yield a solution to the LP for some colour
  - Only works for leaderless protocols; linear combination of runs

Only $2^{|Q|}$ colours

$\Rightarrow$ Every $x \geq k$ protocol with $k \geq 2^{2^{2^{|Q|}}}$ has a "borrowing extension"

Eliminating Borrowing with Liquidity

- Every reachable state can be reached from $2^{|Q|}$ agents in initial state

Eliminating Borrowing with Liquidity

- Every reachable state can be reached from $2^{|Q|}$ agents in initial state (in leaderless protocols!)

Eliminating Borrowing with Liquidity

- Every reachable state can be reached from $2^{|Q|}$ agents in initial state (in leaderless protocols!)
- Thus: we can reach a configuration C_{plenty} with $2^{|Q|}$ agents in *every* state

Eliminating Borrowing with Liquidity

- Every reachable state can be reached from $2^{|Q|}$ agents in initial state (in leaderless protocols!)
- Thus: we can reach a configuration C_{plenty} with $2^{|Q|}$ agents in *every* state
- Also: Every LP has a small ($\leq 2^{|Q|}$) solution

Eliminating Borrowing with Liquidity

- Every reachable state can be reached from $2^{|Q|}$ agents in initial state (in leaderless protocols!)
- Thus: we can reach a configuration C_{plenty} with $2^{|Q|}$ agents in *every* state
- Also: Every LP has a small ($\leq 2^{|Q|}$) solution

Remove borrowing by executing the solution on top of C_{plenty} !

Summary

Summary

1. Generate $2^{2^{2^{|Q|}}}$ runs starting in C_{plenty} with additional agents in initial state

Summary

1. Generate $2^{2^{2^{|Q|}}}$ runs starting in C_{plenty} with additional agents in initial state
2. Assume all are rejecting

Summary

1. Generate $2^{2^{2^{|Q|}}}$ runs starting in C_{plenty} with additional agents in initial state
2. Assume all are rejecting
3. Solve LP to find borrowing extension

Summary

1. Generate $2^{2^{2^{|Q|}}}$ runs starting in C_{plenty} with additional agents in initial state
2. Assume all are rejecting
3. Solve LP to find borrowing extension
   - sequence of transition σ moving agents from initial state to states S

Summary

1. Generate $2^{2^{2^{|Q|}}}$ runs starting in C_{plenty} with additional agents in initial state
2. Assume all are rejecting
3. Solve LP to find borrowing extension
   - sequence of transition σ moving agents from initial state to states S
4. Among the rejecting runs, pick one ending in 0-stable consensus C with colour S

Summary

1. Generate $2^{2^{2^{|Q|}}}$ runs starting in C_{plenty} with additional agents in initial state
2. Assume all are rejecting
3. Solve LP to find borrowing extension
   - sequence of transition σ moving agents from initial state to states S
4. Among the rejecting runs, pick one ending in 0-stable consensus C with colour S
   - i.e. $2^{2^{|Q|}}$ agents in every state in S

Summary

1. Generate $2^{2^{2^{|Q|}}}$ runs starting in C_{plenty} with additional agents in initial state
2. Assume all are rejecting
3. Solve LP to find borrowing extension
   - sequence of transition σ moving agents from initial state to states S
4. Among the rejecting runs, pick one ending in 0-stable consensus C with colour S
   - i.e. $2^{2^{|Q|}}$ agents in every state in S
5. Add new agents and execute σ at the beginning, ignore these agents afterwards

Summary

1. Generate $2^{2^{2^{|Q|}}}$ runs starting in C_{plenty} with additional agents in initial state
2. Assume all are rejecting
3. Solve LP to find borrowing extension
   - sequence of transition σ moving agents from initial state to states S
4. Among the rejecting runs, pick one ending in 0-stable consensus C with colour S
   - i.e. $2^{2^{|Q|}}$ agents in every state in S
5. Add new agents and execute σ at the beginning, ignore these agents afterwards
   - At the end we have C plus some agents in S: still rejecting!

Summary

1. Generate $2^{2^{2^{|Q|}}}$ runs starting in C_{plenty} with additional agents in initial state
2. Assume all are rejecting
3. Solve LP to find borrowing extension
   - sequence of transition σ moving agents from initial state to states S
4. Among the rejecting runs, pick one ending in 0-stable consensus C with colour S
   - i.e. $2^{2^{|Q|}}$ agents in every state in S
5. Add new agents and execute σ at the beginning, ignore these agents afterwards
   - At the end we have C plus some agents in S: still rejecting!
6. Repeat 5 to reject arbitrarily high inputs: Contradiction!

Conclusion

Conclusion

- Space complexity of population protocols is interesting

Conclusion

- Space complexity of population protocols is interesting
- Still a gap for leaderless protocols, and large gap for protocols with a leader

Conclusion

- Space complexity of population protocols is interesting
- Still a gap for leaderless protocols, and large gap for protocols with a leader
- Protocols with a leader might be exponentially more succinct that without (or more!)

Conclusion

- Space complexity of population protocols is interesting
- Still a gap for leaderless protocols, and large gap for protocols with a leader
- Protocols with a leader might be exponentially more succinct that without (or more!)
- Conjecture: both known lower bounds on k are tight

Thank you for
your attention!