On the Analysis of Population Protocols

Michael Blondin
**Population protocols**: distributed computing model for massive networks of passively mobile finite-state agents
**Population protocols**: distributed computing model for massive networks of passively mobile finite-state agents

Can model *e.g.* networks of passively mobile sensors and chemical reaction networks
**Population protocols**: distributed computing model for massive networks of passively mobile finite-state agents

Can model *e.g.* networks of passively mobile sensors and chemical reaction networks

Protocols *compute predicates* of the form $\varphi : \mathbb{N}^d \rightarrow \{0, 1\}$

*e.g.* if $\varphi$ is unary, then $\varphi(n)$ is computed by $n$ agents
Population protocols: distributed computing model for massive networks of passively mobile finite-state agents

This talk:

• Automatic verification and testing
• Study of the minimal size of protocols
• 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

- 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
• **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

- 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
• 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
• 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
• 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

- 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
• 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
• 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
• 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
Example: majority protocol

More **blue birds** than **red birds**?
Example: majority protocol

More blue birds than red birds?

Protocol:

• Two large birds of different colors become small

• Large birds convert small birds to their color
More **blue birds** than **red birds**?

**Protocol:**

- Two large birds of different colors become small
- Large birds convert small birds to their color
Example: majority protocol

More blue birds than red birds?

Protocol:

- Two large birds of different colors become small
- Large birds convert small birds to their color
More **blue birds** than **red birds**?

**Protocol:**

- Two large birds of different colors become small
- Large birds convert small birds to their color
Example: majority protocol

More blue birds than red birds?

Protocol:

- Two large birds of different colors become small
- Large birds convert small birds to their color
Example: majority protocol

More blue birds than red birds?

Protocol:

• Two large birds of different colors become small

• Large birds convert small birds to their color
Example: majority protocol

More blue birds than red birds?

Protocol:

• Two large birds of different colors become small

• Large birds convert small birds to their color
More **blue birds** than **red birds**?

**Protocol:**

- Two large birds of different colors become small
- Large birds convert small birds to their color
Example: majority protocol

More **blue birds** than **red birds**?

**Protocol:**

- Two large birds of different colors become small
- Large birds convert small birds to their color
More **blue birds** than **red birds**?

**Protocol:**

- Two large birds of different colors become small
- Large birds convert small birds to their color
Example: majority protocol

More blue birds than red birds?

Protocol:

• Two large birds of different colors become small

• Large birds convert small birds to their color
Example: majority protocol

More blue birds than red birds?

Protocol:

• Two large birds of different colors become small

• Large birds convert small birds to their color
Example: majority protocol

More blue birds than red birds?

Protocol:

• Two large birds of different colors become small

• Large birds convert small birds to their color
Example: majority protocol

More **blue birds** than **red birds**?

**Protocol:**

- Two large birds of different colors become small
- Large birds convert small birds to their color
Example: threshold protocol

Are there at least 4 sick birds?
Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

• Each bird is in a state of \{0, 1, 2, 3, 4\}

• Sick birds initially in state 1 and healthy birds in state 0

• \((m, n) \mapsto (m + n, 0)\)
  if \(m + n < 4\)

• \((m, n) \mapsto (4, 4)\)
  if \(m + n \geq 4\)
Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

• Each bird is in a state of \{0, 1, 2, 3, 4\}

• Sick birds initially in state 1 and healthy birds in state 0

• \((m, n) \mapsto (m + n, 0)\)
  if \(m + n < 4\)

• \((m, n) \mapsto (4, 4)\)
  if \(m + n \geq 4\)
Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

- Each bird is in a state of \{0, 1, 2, 3, 4\}
- Sick birds initially in state 1 and healthy birds in state 0
- \((m, n) \mapsto (m + n, 0)\) if \(m + n < 4\)
- \((m, n) \mapsto (4, 4)\) if \(m + n \geq 4\)
Example: threshold protocol

Are there at least 4 sick birds?

**Protocol:**

- Each bird is in a state of \( \{0, 1, 2, 3, 4\} \)
- Sick birds initially in state 1 and healthy birds in state 0
- \((m, n) \mapsto (m + n, 0)\) if \(m + n < 4\)
- \((m, n) \mapsto (4, 4)\) if \(m + n \geq 4\)
Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

• Each bird is in a state of \{0, 1, 2, 3, 4\}

• Sick birds initially in state 1 and healthy birds in state 0

• \((m, n) \mapsto (m + n, 0)\) if \(m + n < 4\)

• \((m, n) \mapsto (4, 4)\) if \(m + n \geq 4\)
Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

- Each bird is in a state of \{0, 1, 2, 3, 4\}
- Sick birds initially in state 1 and healthy birds in state 0
- \((m, n) \mapsto (m + n, 0)\) if \(m + n < 4\)
- \((m, n) \mapsto (4, 4)\) if \(m + n \geq 4\)
Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

- Each bird is in a state of \( \{0, 1, 2, 3, 4\} \)
- Sick birds initially in state 1 and healthy birds in state 0
- \((m, n) \mapsto (m + n, 0)\) if \(m + n < 4\)
- \((m, n) \mapsto (4, 4)\) if \(m + n \geq 4\)
Are there at least 4 sick birds?

**Protocol:**

- Each bird is in a state of {0, 1, 2, 3, 4}
- Sick birds initially in state 1 and healthy birds in state 0
- \((m, n) \mapsto (m + n, 0)\) if \(m + n < 4\)
- \((m, n) \mapsto (4, 4)\) if \(m + n \geq 4\)
Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

• Each bird is in a state of \{0, 1, 2, 3, 4\}

• Sick birds initially in state 1 and healthy birds in state 0

• \((m, n) \mapsto (m + n, 0)\)
  if \(m + n < 4\)

• \((m, n) \mapsto (4, 4)\)
  if \(m + n \geq 4\)
Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

- Each bird is in a state of \( \{0, 1, 2, 3, 4\} \)
- Sick birds initially in state 1 and healthy birds in state 0
- \((m, n) \rightarrow (m + n, 0)\)
  if \(m + n < 4\)
- \((m, n) \rightarrow (4, 4)\)
  if \(m + n \geq 4\)
Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

• Each bird is in a state of \{0, 1, 2, 3, 4\}

• Sick birds initially in state 1 and healthy birds in state 0

• \((m, n) \mapsto (m + n, 0)\) if \(m + n < 4\)

• \((m, n) \mapsto (4, 4)\) if \(m + n \geq 4\)
Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

- Each bird is in a state of \{0, 1, 2, 3, 4\}
- Sick birds initially in state 1 and healthy birds in state 0
- \((m, n) \mapsto (m + n, 0)\) if \(m + n < 4\)
- \((m, n) \mapsto (4, 4)\) if \(m + n \geq 4\)
Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

- Each bird is in a state of $\{0, 1, 2, 3, 4\}$
- Sick birds initially in state 1 and healthy birds in state 0
- $(m, n) \mapsto (m + n, 0)$ if $m + n < 4$
- $(m, n) \mapsto (4, 4)$ if $m + n \geq 4$
Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

• Each bird is in a state of \{0, 1, 2, 3, 4\}

• Sick birds initially in state 1 and healthy birds in state 0

• \((m, n) \maps (m + n, 0)\)
  if \(m + n < 4\)

• \((m, n) \maps (4, 4)\)
  if \(m + n \geq 4\)
Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

• Each bird is in a state of \{0, 1, 2, 3, 4\}

• Sick birds initially in state 1 and healthy birds in state 0

• \((m, n) \mapsto (m + n, 0)\) if \(m + n < 4\)

• \((m, n) \mapsto (4, 4)\) if \(m + n \geq 4\)
Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

- Each bird is in a state of \( \{0, 1, 2, 3, 4\} \)
- Sick birds initially in state 1 and healthy birds in state 0
- \((m, n) \mapsto (m + n, 0)\) if \(m + n < 4\)
- \((m, n) \mapsto (4, 4)\) if \(m + n \geq 4\)
Demonstration
Population protocols: formal model

- **States:** finite set $Q$
- **Opinions:** $O : Q \rightarrow \{0, 1\}$
- **Initial states:** $I \subseteq Q$
- **Transitions:** $T \subseteq Q^2 \times Q^2$
Population protocols: formal model

- **States:** finite set $Q$
- **Opinions:** $O : Q \rightarrow \{0, 1\}$
- **Initial states:** $I \subseteq Q$
- **Transitions:** $T \subseteq Q^2 \times Q^2$
Population protocols: formal model

- **States:** finite set $Q$
- **Opinions:** $O : Q \rightarrow \{0, 1\}$
- **Initial states:** $I \subseteq Q$
- **Transitions:** $T \subseteq Q^2 \times Q^2$
Population protocols: formal model

- **States:** finite set $Q$
- **Opinions:** $O : Q \rightarrow \{0, 1\}$
- **Initial states:** $I \subseteq Q$
- **Transitions:** $T \subseteq Q^2 \times Q^2$
Population protocols: formal model

Reachability graph:
Executions must be fair:
Executions must be fair:
A protocol computes a predicate $f: \mathbb{N}^l \rightarrow \{0, 1\}$ if fair executions reach common consensus.
Population protocols: formal model

A protocol computes a predicate $f: \mathbb{N}^I \rightarrow \{0, 1\}$ if fair executions reach common consensus.

Expressive power

<table>
<thead>
<tr>
<th>Expressive power</th>
<th>Angluin, Aspnes, Eisenstat PODC’06</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population protocols compute precisely predicates definable in Presburger arithmetic, i.e. $\text{FO}(\mathbb{N}, +, &lt;)$</td>
<td></td>
</tr>
</tbody>
</table>
Protocols can become complex, even for $B \geq R$:

**Fast and Exact Majority in Population Protocols**

Dan Alistarh  
Microsoft Research  
Rati Gelashvili*  
MIT  
Milan Vojnović  
Microsoft Research

1. $weight(x) = \begin{cases} |x| & \text{if } x \in \text{StrongStates} \text{ or } x \in \text{WeakStates}; \\ 1 & \text{if } x \in \text{IntermediateStates}. \end{cases}$

2. $sgn(x) = \begin{cases} 1 & \text{if } x \in \{+0, 1_d, \ldots, 1_1, 3, 5, \ldots, m\}; \\ -1 & \text{otherwise}. \end{cases}$

3. $value(x) = sgn(x) \cdot weight(x)$
   
   /* Functions for rounding state interactions */

4. $\phi(x) = -1_1$ if $x = -1_1; 1_1$ if $x = 1_1; x$, otherwise

5. $R_{1}(k) = \phi(k$ if $k$ odd integer, $k - 1$ if $k$ even)

6. $R_{1}(k) = \phi(k$ if $k$ odd integer, $k + 1$ if $k$ even)

7. Shift-to-Zero($x$) = \begin{cases} -1_{j+1} & \text{if } x = -1_j \text{ for some index } j < d \\ 1_{j+1} & \text{if } x = 1_j \text{ for some index } j < d \\ x & \text{otherwise.} \end{cases}$

8. Sign-to-Zero($x$) = \begin{cases} +0 & \text{if } sgn(x) > 0 \\ -0 & \text{otherwise.} \end{cases}$

9. procedure update($x$, $y$)

10. if $(weight(x) > 0 \text{ and } weight(y) > 1) \text{ or } (weight(y) > 0 \text{ and } weight(x) > 1)$ then

11. $x' \leftarrow R_{1} \left( \frac{value(x) + value(y)}{2} \right)$ and $y' \leftarrow R_{1} \left( \frac{value(x) + value(y)}{2} \right)$

12. else if $weight(x) \cdot weight(y) = 0 \text{ and } value(x) + value(y) > 0$ then

13. if $weight(x) \neq 0$ then $x' \leftarrow \text{Shift-to-Zero($x$)}$ and $y' \leftarrow \text{Sign-to-Zero($y$)}$

14. else $y' \leftarrow \text{Shift-to-Zero($y$)}$ and $x' \leftarrow \text{Sign-to-Zero($x$)}$

15. else if $(x \in \{-1_d, +1_d\} \text{ and } weight(y) = 1 \text{ and } sgn(x) \neq sgn(y))$ or

16. $(y \in \{-1_d, +1_d\} \text{ and } weight(x) = 1 \text{ and } sgn(y) \neq sgn(x))$ then

17. $x' \leftarrow -0 \text{ and } y' \leftarrow +0$

18. else

19. $x' \leftarrow \text{Shift-to-Zero($x$)}$ and $y' \leftarrow \text{Shift-to-Zero($y$)}$
Analysis of protocols

Protocols can become complex, even for $B \geq R$:

Fast and Exact Majority in Population Protocols

Dan Alistarh
Microsoft Research

Rati Gelashvili*
MIT

Milan Vojnović
Microsoft Research

1. $\text{weight}(x) = \begin{cases} |x| & \text{if } x \in \text{StrongStates or } x \in \text{WeakStates;} \\ 1 & \text{if } x \in \text{IntermediateStates.} \end{cases}$

2. $\text{sgn}(x) = \begin{cases} 1 & \text{if } x \in \{+0, 1_d, \ldots, 1_1, 3, 5, \ldots, m\}; \\ -1 & \text{otherwise.} \end{cases}$

3. $\text{value}(x) = \text{sgn}(x) \cdot \text{weight}(x)$

4. $\phi(x) = -1_1$ if $x = -1; 1_1$ if $x = 1; x$, otherwise

5. $R_1(k) = \phi(k$ if $k$ odd integer, $k-1$ if $k$ even)

6. $R_1(k) = \phi(k$ if $k$ odd integer, $k+1$ if $k$ even)

7. $\text{Shift-to-Zero}(x) = \begin{cases} -1_{j+1} & \text{if } x = -1_j \text{ for some index } j < d; \\ 1_{j+1} & \text{if } x = 1_j \text{ for some index } j < d; \\ x & \text{otherwise.} \end{cases}$

8. $\text{Sign-to-Zero}(x) = \begin{cases} +0 & \text{if } \text{sgn}(x) > 0; \\ -0 & \text{otherwise}. \end{cases}$

9. procedure $\text{update}(x, y)$

10. if (weight($x$) > 0 and weight($y$) > 1) or (weight($y$) > 0 and weight($x$) > 1) then

11. \[ x' \leftarrow R_1 \left( \frac{\text{value}(x) + \text{value}(y)}{2} \right) \text{ and } y' \leftarrow R_1 \left( \frac{\text{value}(x) + \text{value}(y)}{2} \right) \]

12. else if weight($x$) \cdot weight($y$) = 0 and value($x$) + value($y$) > 0 then

13. if weight($x$) \neq 0 then $x' \leftarrow \text{Shift-to-Zero}(x)$ and $y' \leftarrow \text{Sign-to-Zero}(x)$

14. else $y' \leftarrow \text{Shift-to-Zero}(y)$ and $x' \leftarrow \text{Sign-to-Zero}(y)$

15. else if $\{x \in \{-1_d, +1_d\}$ and weight($y$) = 1 and sgn($x$) \neq sgn($y$) \} or

16. $\{y \in \{-1_d, +1_d\}$ and weight($x$) = 1 and sgn($y$) \neq sgn($x$) \} then

17. $x' \leftarrow -0 \text{ and } y' \leftarrow +0$

18. else

19. $x' \leftarrow \text{Shift-to-Zero}(x)$ and $y' \leftarrow \text{Shift-to-Zero}(y)$

How to verify correctness automatically?
Analysis of protocols

Number of states corresponds to amount of memory, relevant to keep it minimal for embedded systems

- $B \geq R$ requires at least 4 states \hspace{1cm} (Mertzios et al. ICALP’14)
- $X \geq c$ requires at most $c + 1$ states
Number of states corresponds to amount of memory, relevant to keep it minimal for embedded systems

- $B \geq R$ requires at least 4 states \cite{Mertzios et al. ICALP’14}

- $X \geq c$ requires at most $c + 1$ states

What is the state complexity of common predicates?
Analysis of protocols

Convergence speed may vary wildly, challenging to establish bounds

![Graph showing the relationship between initial amount of R's and average number of steps to convergence for different protocols. The x-axis represents the initial amount of R's ranging from 0 to 20, and the y-axis represents the average number of steps to convergence ranging from $10^6$ to $10^{20}$. The graph includes lines for AVC, 3-state, 4-state, and 4-state tiebreaker protocols.]
Analysis of protocols

Convergence speed may vary wildly, challenging to establish bounds

How to derive asymptotic bounds automatically?
Analysis of protocols

1. **Automatic verification of correctness**
   - PODC’17 with Javier, Stefan and Philipp
   - Submission to CAV’18 with Javier and Stefan
   - Interns: Philip Offtermatt and Amrita Suresh

2. **State complexity of common predicates**
   - STACS’18 with Javier and Stefan

3. **Automatic analysis of convergence speed**
   - Ongoing work with Javier and Antonín Kučera
1. **Automatic verification of correctness**
   - PODC’17 with Javier, Stefan and Philipp
   - Submission to CAV’18 with Javier and Stefan

2. **State complexity of common predicates**
   - STACS’18 with Javier and Stefan

This talk
Verification: state of the art

Existing verification tools:

- **PAT**: model checker with global fairness
  (Sun, Liu, Song Dong and Pang CAV’09)

- **bp-ver**: graph exploration
  (Chatzigiannakis, Michail and Spirakis SSS’10)

- Conversion to counter machines + PRISM/Spin
  (Clément, Delporte-Gallet, Fauconnier and Sighireanu ICDCS’11)
Verification: state of the art

Existing verification tools:

• PAT: model checker with global fairness
  (Sun, Liu, Song Dong and Pang CAV’09)

• bp-ver: graph exploration
  (Chatzigiannakis, Michail and Spirakis SSS’10)

• Conversion to counter machines + PRISM/Spin
  (Clément, Delporte-Gallet, Fauconnier and Sighireanu ICDCS’11)

Only for populations of fixed size!
Sometimes possible to verify all sizes:

- Verification with the interactive theorem prover Coq
  
  (Deng and Monin TASE’09)

- \textit{bp-ver}: graph exploration
  
  (Chatzigiannakis, Michail and Spirakis SSS’10)

- Conversion to counter machines + PRISM/Spin
  
  (Clément, Delporte-Gallet, Fauconnier and Sighireanu ICDCS’11)

Only for populations of fixed size!

Challenge: verifying automatically all sizes
Sometimes possible to verify all sizes:

- Verification with the interactive theorem prover Coq
  (Deng and Monin TASE’09)

Not automatic!
Sometimes possible to verify all sizes:

- Verification with the interactive theorem prover Coq
  (Deng and Monin TASE’09)

Challenge: verifying automatically all sizes
Testing whether a protocol computes $\varphi$ amounts to testing:

$$\neg \exists C, D : \ C \xrightarrow{*} D \land$$

$C$ is initial $\land$

$D$ is in a BSCC $\land$

opinion$(D) \neq \varphi(C)$
Testing whether a protocol computes $\varphi$ amounts to testing:

$$\neg \exists C, D : \quad C \xrightarrow{*} D \land$$

$C$ is initial $\land$

$D$ is in a BSCC $\land$

$\text{opinion}(D) \neq \varphi(C)$

As difficult as verification
Testing whether a protocol computes $\varphi$ amounts to testing:

$$\neg \exists C, D : \quad C \xrightarrow{*} D \land$$

$C$ is initial $\land$

$D$ is in a BSCC $\land$

opinion($D$) $\neq \varphi(C)$

Relaxed with Presburger-definable overapproximation!
Testing whether a protocol computes \( \varphi \) amounts to testing:

\[ \neg \exists C, D : \quad C \rightarrow^* D \land \]

- C is initial \( \land \)
- D is in a BSCC \( \land \)
- opinion(D) \( \neq \varphi(C) \)

\textbf{Difficult to express}
Testing whether a protocol computes $\phi$ amounts to testing:

$$\neg \exists C, D : \quad C \xrightarrow{\ast} D \land$$

- $C$ is initial $\land$
- $D$ is terminal $\land$
- opinion$(D) \neq \phi(C)$

**BSCCs are of size 1 for most protocols!**
Testing whether a protocol computes $\varphi$ amounts to testing:

$$\neg \exists C, D : \quad C \xrightarrow{*} D \land$$

$C$ is initial $\land$
$D$ is terminal $\land$
$\text{opinion}(D) \neq \varphi(C)$

Testable with an SMT solver
Testing whether a protocol computes $\varphi$ amounts to testing:

$$\neg \exists C, D : \quad C \xrightarrow{*} D \land$$

- $C$ is initial $\land$
- $D$ is terminal $\land$
- opinion($D$) $\neq \varphi(C)$

But how to know whether all BSCCs are of size 1?
Protocol is *silent* if fair executions reach terminal configurations.

BSCCs of size 1
Protocol is *silent* if fair executions reach terminal configurations

- Testing silentness is as hard as verification of correctness
- But most protocols satisfy a common design

BSCCs of size 1
Partition $T = T_1 \cup T_2 \cup \cdots \cup T_n$ s.t. for every $i$

- all executions restricted to $T_i$ terminate
- if $T_1 \cup \cdots \cup T_{i-1}$ disabled in $C$ and $C \xrightarrow{T_i^*} D$, then $T_1 \cup \cdots \cup T_{i-1}$ also disabled in $D$
Partition $T = T_1 \cup T_2 \cup \cdots \cup T_n$ s.t. for every $i$

- all executions restricted to $T_i$ terminate
- if $T_1 \cup \cdots \cup T_{i-1}$ disabled in $C$ and $C \xrightarrow{T_i^*} D$, then $T_1 \cup \cdots \cup T_{i-1}$ also disabled in $D$
Partition $T = T_1 \cup T_2 \cup \cdots \cup T_n$ s.t. for every $i$

- all executions restricted to $T_i$ terminate
- if $T_1 \cup \cdots \cup T_{i-1}$ disabled in $C$ and $C \xrightarrow{T_i^*} D$, then $T_1 \cup \cdots \cup T_{i-1}$ also disabled in $D$
Partition $T = T_1 \cup T_2 \cup \cdots \cup T_n$ s.t. for every $i$

- all executions restricted to $T_i$ terminate

- if $T_1 \cup \cdots \cup T_{i-1}$ disabled in $C$ and $C \xrightarrow{T_i^*} D$, then $T_1 \cup \cdots \cup T_{i-1}$ also disabled in $D$
Partition $T = T_1 \cup T_2 \cup \cdots \cup T_n$ s.t. for every $i$

- all executions restricted to $T_i$ terminate
- if $T_1 \cup \cdots \cup T_{i-1}$ disabled in $C$ and $C \xrightarrow{T_i^*} D$, then $T_1 \cup \cdots \cup T_{i-1}$ also disabled in $D$
Common design: layered termination

\[ T_1 \]

\[
\begin{align*}
B R & \rightarrow b r \\
R b & \rightarrow R r \\
B r & \rightarrow B b \\
b r & \rightarrow b b 
\end{align*}
\]
Common design: layered termination

$$T_1$$

$$B R \rightarrow b r$$
$$R b \rightarrow R r$$
$$B r \rightarrow B b$$
$$b r \rightarrow b b$$

Bad partition: not all executions over $$T_1$$ terminate
Common design: layered termination

Bad partition: not all executions over $T_1$ terminate

\[
\{B, B, R, R\} \rightarrow \{B, b, r, R\} \rightarrow \{B, b, b, R\} \rightarrow \\
\{B, b, r, R\} \rightarrow \{B, b, b, R\} \rightarrow \cdots
\]
Common design: layered termination

\[ T_1 \]

\[ B R \rightarrow b r \]

\[ T_2 \]

\[ R b \rightarrow R r \]

\[ T_3 \]

\[ B r \rightarrow B b \]

\[ b r \rightarrow b b \]
Common design: layered termination

\[ T_1 \]

\[ B R \rightarrow b r \]

\[ T_2 \]

\[ R b \rightarrow R r \]

\[ T_3 \]

\[ B r \rightarrow B b \]

\[ b r \rightarrow b b \]

\[ \#B \geq \#R: \]

\[ \{ B^*, R^* \} \]
Common design: layered termination

\[ T_1 \quad \times \quad T_2 \quad \quad T_3 \]

\[ B R \rightarrow b r \quad R b \rightarrow R r \quad B r \rightarrow B b \]

\[ b r \rightarrow b b \]

\[ \#B \geq \#R: \]

\[ \{B^*, R^*\} \xrightarrow{*} \{B^*, b^*, r^*\} \]
Common design: layered termination

\[ T_1 \quad \times \quad T_2 \quad \times \quad T_3 \]

\[ B \ R \rightarrow \ b \ r \quad \quad R \ b \rightarrow \ R \ r \quad \quad B \ r \rightarrow \ B \ b \]

\[ b \ r \rightarrow \ b \ b \]

\[ \#B \geq \#R: \]

\[ \{B^*, \ R^*\} \quad \rightarrow \quad \{B^*, \ b^*, \ r^*\} \]
Common design: layered termination

\[ T_1 \quad \times \quad T_2 \quad \times \quad T_3 \]

\[
\begin{align*}
B R &\rightarrow b r \\
R b &\rightarrow R r \\
B r &\rightarrow B b \\
b r &\rightarrow b b
\end{align*}
\]

#B ≥ #R:

\[
\{ B^*, R^* \} \xrightarrow{*} \{ B^*, b^*, r^* \} \xrightarrow{*} \{ B^*, b^* \}
\]
Common design: layered termination

\[ T_1 \]

\[ B R \rightarrow b r \]

\[ T_2 \]

\[ R b \rightarrow R r \]

\[ T_3 \]

\[ B r \rightarrow B b \]

\[ \begin{align*}
\#B \geq \#R: \\
\{B^*, \ R^*\} \rightarrow^* \{B^*, \ b^*, \ r^*\} \rightarrow^* \{B^*, \ b^*\}
\end{align*} \]

\[ \#R > \#B: \]

\[ \{R^+, \ B^*\} \]
Common design: layered termination

\[ T_1 \times \]

\[ B R \rightarrow b r \quad | \quad R b \rightarrow R r \quad | \quad B r \rightarrow B b \]

\#B \geq \#R:

\{B^*, R^*\} \rightarrow \{B^*, b^*, r^*\} \rightarrow \{B^*, b^*\}

\#R > \#B:

\{R^+, B^*\} \rightarrow \{R^+, r^*, b^*\}
Common design: layered termination

\[ T_1 \times \quad T_2 \times \quad T_3 \]

\[
\begin{align*}
B & \rightarrow b \quad R & \rightarrow \quad b \quad r \\
R & \rightarrow b \quad R & \rightarrow R \quad r \\
B & \rightarrow B \quad b \quad r & \rightarrow b \quad b
\end{align*}
\]

\[ \#B \geq \#R: \]
\[
\{ B^*, R^* \} \rightarrow \{ B^*, b^*, r^* \} \rightarrow \{ B^*, b^* \}
\]

\[ \#R > \#B: \]
\[
\{ R^+, B^* \} \rightarrow \{ R^+, r^*, b^* \} \rightarrow \{ R^+, r^* \}
\]
**Common design: layered termination**

\[ T_1 \times \quad T_2 \times \quad T_3 \times \]

\[
\begin{align*}
B & \rightarrow b & r & \\
R & \rightarrow b & R & \rightarrow R & r \\
B & \rightarrow b & b & \\
B & \rightarrow B & b & \\
\end{align*}
\]

**#B ≥ #R:**

\[
\{B^*, R^*\} \rightarrow \{B^*, b^*, r^*\} \rightarrow \{B^*, b^*\}
\]

**#R > #B:**

\[
\{R^+, B^*\} \rightarrow \{R^+, r^*, b^*\} \rightarrow \{R^+, r^*\}
\]
Common design: layered termination

<table>
<thead>
<tr>
<th>Theorem</th>
<th>PODC’17</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deciding whether a protocol is strongly silent $\in \text{NP}$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Proof sketch</th>
<th>PODC’17</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guess partition $T = T_1 \cup T_2 \cup \cdots \cup T_n$ and test whether it is correct by verifying</td>
<td></td>
</tr>
<tr>
<td>• Petri net structural termination</td>
<td></td>
</tr>
<tr>
<td>• Additional simple structural properties</td>
<td></td>
</tr>
</tbody>
</table>
**Theorem**

Strongly silent protocols as expressive as general protocols

**Proof sketch**

- Protocols for
  \[ a_1x_1 + \ldots + a_nx_n \geq b \]
  \[ a_1x_1 + \ldots + a_nx_n \equiv b \pmod{m} \]
  have layered termination partitions

- Conjunction and negation preserve layered termination
## A new tool: Peregrine

PODC’17 / CAV’18 submission

Peregrine: Haskell + Z3 + JavaScript (front end)

gitlab.lrz.de/i7/peregrine

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Predicate</th>
<th># states</th>
<th># trans.</th>
<th>Time (secs.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority [a]</td>
<td>$x \geq y$</td>
<td>4</td>
<td>4</td>
<td>0.1</td>
</tr>
<tr>
<td>Broadcast [b]</td>
<td>$x_1 \lor \cdots \lor x_n$</td>
<td>2</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>Linear ineq. [c]</td>
<td>$\sum a_i x_i \geq 9$</td>
<td>75</td>
<td>2148</td>
<td>2376</td>
</tr>
<tr>
<td>Modulo [c]</td>
<td>$\sum a_i x_i = 0 \mod 70$</td>
<td>72</td>
<td>2555</td>
<td>3177</td>
</tr>
<tr>
<td>Threshold [d]</td>
<td>$x \geq 50$</td>
<td>51</td>
<td>1275</td>
<td>182</td>
</tr>
<tr>
<td>Threshold [b]</td>
<td>$x \geq 325$</td>
<td>326</td>
<td>649</td>
<td>3471</td>
</tr>
<tr>
<td>Threshold [e]</td>
<td>$x \geq 10^7$</td>
<td>37</td>
<td>155</td>
<td>19</td>
</tr>
</tbody>
</table>

[b] Clément et al. 2011  [d] Chatzigiannakis et al. 2010
Demonstration
Threshold state complexity: logarithmic bounds

**Given:** Presburger-definable predicate $\varphi$

**Question:** Smallest number of states necessary to compute $\varphi$?
Threshold state complexity: logarithmic bounds

Given: Presburger-definable predicate $\varphi$

Question: Smallest number of states necessary to compute $\varphi$?

Difficult problem...

What about basic predicates?
Threshold state complexity: logarithmic bounds

**Given:** \( c \in \mathbb{N} \)

**Question:** Smallest number of states necessary to compute \( x \geq c \)?
Threshold state complexity: logarithmic bounds

Given: $c \in \mathbb{N}$

Question: Smallest number of states necessary to compute $x \geq c$?

Upper bound: $c + 1$

Lower bound: 2
### Threshold state complexity: logarithmic bounds

**Given:** \( c \in \mathbb{N} \)

**Upper bound:** \( c + 1 \)

**Question:** Smallest number of states necessary to compute \( x \geq c \)?

**Lower bound:** \( 2 \)

<table>
<thead>
<tr>
<th>Theorem</th>
<th>STACS’18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computable with ( O(\log c) ) states, if ( c = 2^n ).</td>
<td></td>
</tr>
</tbody>
</table>

**Proof sketch**

\[
(1, 1) \quad \mapsto \quad (2, 0) \\
(2, 2) \quad \mapsto \quad (4, 0) \\
\vdots \quad \vdots \\
(2^{n-1}, 2^{n-1}) \quad \mapsto \quad (2^n, 0) \\
(2^n, m) \quad \mapsto \quad (2^n, 2^n)
\]
## Threshold state complexity: logarithmic bounds

**Given:** $c \in \mathbb{N}$

**Question:** Smallest number of states necessary to compute $x \geq c$?

### Theorem

<table>
<thead>
<tr>
<th>STACS’18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computable with $O(\log c)$ states, if $c = 2^n$.</td>
</tr>
</tbody>
</table>

### Proof sketch

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$c \in \mathbb{N}$</td>
<td></td>
</tr>
<tr>
<td>Upper bound: $c + 1$</td>
<td>Lower bound: 2</td>
</tr>
</tbody>
</table>

1. $(1, 1) \mapsto (2, 0)$
2. $(2, 2) \mapsto (4, 0)$
3. $\vdots$
4. $(2^{n-1}, 2^{n-1}) \mapsto (2^n, 0)$
5. $(2^n, m) \mapsto (2^n, 2^n)$

+ extra states and transitions
Threshold state complexity: logarithmic bounds

**Given:** \( c \in \mathbb{N} \)

**Question:** Smallest number of states necessary to compute \( x \geq c \)?

**Upper bound:** \( O(\log c) \)

**Lower bound:** 2
Threshold state complexity: logarithmic bounds

Given: \( c \in \mathbb{N} \)

Question: Smallest number of states necessary to compute \( x \geq c \)?

Upper bound: \( O(\log c) \)

Lower bound: 2

Theorem

Let \( P_0, P_1, \ldots \) be protocols such that \( P_c \) computes \( x \geq c \).

There are infinitely many \( c \) s.t. \( P_c \) has \( \geq (\log c)^{1/4} \) states.

Proof sketch

Counting argument on \# unary predicates vs. \# protocols.
Threshold state complexity: logarithmic bounds

Given: \( c \in \mathbb{N} \)

Question: Smallest number of states necessary to compute \( x \geq c \)?

Upper bound: \( O(\log c) \)

Lower bound: \( O(\log^{1/4} c) \) for inf. many \( c \)
Threshold state complexity: logarithmic bounds

Given: \( c \in \mathbb{N} \)

Question: Smallest number of states necessary to compute \( x \geq c \)?

Upper bound: \( O(\log c) \)

Lower bound: \( O(\log^{1/4} c) \) for inf. many \( c \)

Possible to go below \( \log c \) for some \( c \)?
Threshold state complexity: logarithmic bounds

Given: \( c \in \mathbb{N} \)

Question: Smallest number of states necessary to compute \( x \geq c \)?

Upper bound: \( O(\log c) \)

Lower bound: \( O(\log^{1/4} c) \) for inf. many \( c \)

Possible to go below \( \log c \) for some \( c \)?

Yes!
There exist protocols $P_0, P_1, \ldots$ and numbers $c_0 < c_1 < \cdots$ such that $P_i$ computes $x \geq c_i$ and has $O(\log \log c_i)$ states.
Threshold state complexity: sublogarithmic bounds

<table>
<thead>
<tr>
<th>Theorem</th>
<th>STACS’18</th>
</tr>
</thead>
<tbody>
<tr>
<td>There exist protocols $P_0, P_1, \ldots$ and numbers $c_0 &lt; c_1 &lt; \cdots$ such that $P_i$ computes $x \geq c_i$ and has $O(\log \log c_i)$ states.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lemma</th>
<th>Mayr and Meyer ’82</th>
</tr>
</thead>
<tbody>
<tr>
<td>For every $c \in \mathbb{N}$, there exists a reversible multiset rewriting system $\mathcal{R}_c$ over alphabet $\Sigma \supseteq {x, y, z, w}$ of size $O(c)$ with rewriting rules $T \subseteq \Sigma^{\leq 5} \times \Sigma^{\leq 5}$ such that</td>
<td></td>
</tr>
</tbody>
</table>
| \[
\{x, y\} \overset{*}{\rightarrow} M \text{ and } w \in M \iff M = \{y, z^{2^c}, w\}
\] |
Threshold state complexity: sublogarithmic bounds

<table>
<thead>
<tr>
<th>Theorem</th>
<th>STACS’18</th>
</tr>
</thead>
<tbody>
<tr>
<td>There exist protocols $P_0, P_1, \ldots$ and numbers $c_0 &lt; c_1 &lt; \cdots$ such that $P_i$ computes $x \geq c_i$ and has $O(\log \log c_i)$ states.</td>
<td></td>
</tr>
</tbody>
</table>

Proof sketch

- $\mathcal{R}_c$ can be simulated by adding a padding symbol ⊥
Threshold state complexity: sublogarithmic bounds

**Theorem**

STACS’18

There exist protocols $P_0, P_1, \ldots$ and numbers $c_0 < c_1 < \cdots$ such that $P_i$ computes $x \geq c_i$ and has $O(\log \log c_i)$ states.

**Proof sketch**

- $\mathcal{R}_c$ can be simulated by adding a padding symbol $\bot$

<table>
<thead>
<tr>
<th>Rewriting system $\mathcal{R}_c$</th>
<th>5-way population protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(e, f, g) \mapsto (h, i)$</td>
<td>$(e, f, g, \bot, \bot) \mapsto (h, i, \bot, \bot, \bot)$</td>
</tr>
<tr>
<td>$(e, f) \mapsto (g, h, i)$</td>
<td>$(e, f, \bot, \bot, \bot) \mapsto (g, h, i, \bot, \bot)$</td>
</tr>
</tbody>
</table>
Threshold state complexity: sublogarithmic bounds

**Theorem**

There exist protocols $P_0, P_1, \ldots$ and numbers $c_0 < c_1 < \cdots$ such that $P_i$ computes $x \geq c_i$ and has $O(\log \log c_i)$ states.

**Proof sketch**

- $\mathcal{R}_c$ can be simulated by adding a padding symbol $\perp$

  Each 5-way transition is converted to a “gadget” of 2-way transitions
Theorem

There exist protocols $P_0, P_1, \ldots$ and numbers $c_0 < c_1 < \cdots$ such that $P_i$ computes $x \geq c_i$ and has $O(\log \log c_i)$ states.

Proof sketch

- $R_c$ can be simulated by adding a padding symbol $\bot$
- New rule: agents in state $w$ can convert others to $w$
Threshold state complexity: sublogarithmic bounds

**Theorem**

There exist protocols $P_0, P_1, \ldots$ and numbers $c_0 < c_1 < \cdots$ such that $P_i$ computes $x \geq c_i$ and has $O(\log \log c_i)$ states.

**Proof sketch**

- $R_c$ can be simulated by adding a padding symbol $\perp$
- New rule: agents in state $w$ can convert others to $w$
- Simulate $R_c$ from $\{x, y, \perp, \perp, \ldots, \perp\}$
Threshold state complexity: sublogarithmic bounds

<table>
<thead>
<tr>
<th>Theorem</th>
<th>STACS’18</th>
</tr>
</thead>
<tbody>
<tr>
<td>There exist protocols $P_0, P_1, \ldots$ and numbers $c_0 &lt; c_1 &lt; \cdots$ such that $P_i$ computes $x \geq c_i$ and has $O(\log \log c_i)$ states.</td>
<td></td>
</tr>
</tbody>
</table>

Proof sketch

- $\mathcal{R}_c$ can be simulated by adding a padding symbol $\perp$
- New rule: agents in state $w$ can convert others to $w$
- Simulate $\mathcal{R}_c$ from $\{x, y, \perp, \perp, \ldots, \perp\}$
- $\{w, w, \ldots, w\}$ reachable $\iff$ initially $\geq 2^{2^c}$ agents in $\perp$
## Threshold state complexity: sublogarithmic bounds

<table>
<thead>
<tr>
<th>Theorem</th>
<th>STACS’18</th>
</tr>
</thead>
<tbody>
<tr>
<td>There exist protocols $P_0, P_1, \ldots$ and numbers $c_0 &lt; c_1 &lt; \cdots$ such that $P_i$ computes $x \geq c_i$ and has $O(\log \log c_i)$ states.</td>
<td></td>
</tr>
</tbody>
</table>

### Proof sketch

- $\mathcal{R}_c$ can be simulated by adding a padding symbol $\bot$
- New rule: agents in state $w$ can convert others to $w$
- Simulate $\mathcal{R}_c$ from $\{x, y, \bot, \bot, \ldots, \bot\}$
- $\{w, w, \ldots, w\}$ reachable $\iff$ initially $\geq 2^{2^c}$ agents in $\bot$
- By reversibility and fairness, cannot avoid $\{w, w, \ldots, w\}$
Let \( A \in \mathbb{Z}^{m \times k} \), let \( c \in \mathbb{Z}^m \) and let \( n \) be the largest absolute value of numbers occurring in \( A \) and \( c \).

**Observation**

Classical protocol computing \( Ax + c > 0 \) has \( O(n^m) \) states.
Let $A \in \mathbb{Z}^{m \times k}$, let $c \in \mathbb{Z}^{m}$ and let $n$ be the largest absolute value of numbers occurring in $A$ and $c$.

**Observation**

Classical protocol computing $Ax + c > 0$ has $O(n^m)$ states.

**Theorem**

There exists a protocol that computes $Ax + c > 0$ and has

- at most $O((m + k) \cdot \log mn)$ states
- at most $O(m \cdot \log mn)$ leaders
**Conclusion**

**Peregrine:**
- Graphical and command-line tool for designing, simulating and verifying population protocols
- Can verify silent protocols

**Future work:**
- Verification of non silent protocols (ongoing with Amrita)
- Convergence speed analysis (ongoing with Javier and Tony)
- Failure ratio analysis
- LTL model checking
Conclusion

Peregrine:

- Graphical and command-line tool for designing, simulating and verifying population protocols
- Can verify silent protocols

Future work:

- Verification of non silent protocols (ongoing with Amrita)
- Convergence speed analysis (ongoing with Javier and Tony)
- Failure ratio analysis
- LTL model checking
Conclusion

State complexity:

- Complexity of $x \geq c$ can be decreased from $O(c)$ to $O(\log c)$ and sometimes $O(\log \log c)$
- Similar results for systems of linear inequalities

Future work:

- Is $O(\log \log \log c)$ sometimes possible? (not for the class of 1-aware protocols)
- State complexity of Presburger-definable predicates
- Study of the trade-off between size and speed
Conclusion

State complexity:

• Complexity of $x \geq c$ can be decreased from $O(c)$ to $O(\log c)$ and sometimes $O(\log \log c)$

• Similar results for systems of linear inequalities

Future work:

• Is $O(\log \log \log c)$ sometimes possible? (not for the class of 1-aware protocols)

• State complexity of Presburger-definable predicates

• Study of the trade-off between size and speed
Thank you! Vielen Dank!