Formal Analysis of Population Protocols

Michael Blondin
Joint work with Javier Esparza, Stefan Jaax,
Antonín Kučera and Philipp J. Meyer
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 $\varphi : \mathbb{N}^d \to \{0, 1\}$

e.g. if $\varphi$ is unary, then $\varphi(n)$ is computed by $n$ agents


**Overview**

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

**This talk:** overview of recent advances on the formal analysis of 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
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
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
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**
Example: majority protocol

At least as many **blue birds** than **red birds**?
Example: majority protocol

At least as many **blue** birds than **red** birds?

**Protocol:**

- Two large birds of different colors become small and blue
- Large birds convert small birds to their color
At least as many blue birds than red birds?

Protocol:

• Two large birds of different colors become small and blue

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

At least as many blue birds than red birds?

Protocol:

• Two large birds of different colors become small and blue

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

At least as many blue birds than red birds?

Protocol:

• Two large birds of different colors become small and blue

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

At least as many blue birds than red birds?

Protocol:

• Two large birds of different colors become small and blue

• Large birds convert small birds to their color
At least as many blue birds than red birds?

Protocol:

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

At least as many blue birds than red birds?

Protocol:

• Two large birds of different colors become small and blue
• Large birds convert small birds to their color
At least as many **blue birds** than **red birds**?

**Protocol:**

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

At least as many **blue birds** than **red birds**?

**Protocol:**

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

At least as many blue birds than red birds?

Protocol:

• Two large birds of different colors become small and blue

• Large birds convert small birds to their color
At least as many blue birds than red birds?

Protocol:

- Two large birds of different colors become small and blue
- Large birds convert small birds to their color
- **To break ties:** small blue birds convert small red birds
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
- \((m, n) \rightarrow (m + n, 0)\) 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 \(m + n \geq 4\)
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

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

• \((m, n) \leftrightarrow (4, 4)\) 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

• \((m, n) \mapsto (m + n, 0)\)
  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 \(m + n \geq 4\)
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**: $T \subseteq Q^2 \times Q^2$
Interaction graph:

Agent 1 \quad Agent 2

Agent 3 \quad Agent 4
Population protocols: computations

Reachability graph:
Underlying Markov chain:
(pairs of agents are picked uniformly at random)
A run is an infinite path:
A protocol computes a predicate $\varphi : \mathbb{N}^l \rightarrow \{0, 1\}$ if runs reach common stable consensus with probability 1.
A protocol computes a predicate $\varphi : \mathbb{N}^I \rightarrow \{0, 1\}$ if runs reach **common stable consensus** with probability 1

**Expressive power**

Angluin, Aspnes, Eisenstat PODC’06

Population protocols compute precisely predicates definable in Presburger arithmetic, *i.e.* $\text{FO}(\mathbb{N}, +, <)$
Population protocols: computations

Other variants considered:

- Approximate protocols  
  e.g. Angluin, Aspnes, Eisenstat DISC’07
- Protocols with leaders  
  Angluin, Aspnes, Eisenstat Dist. Comput.’08
- Protocols with failures  
  Delporte-Gallet et al. DCOSS’06
- Trustful protocols  
  Bournez, Lefevre, Rabie DISC’13
- Mediated protocols, etc.  
  Michail, Chatzigiannakis, Spirakis TCS’11

Expressive power

Population protocols compute precisely predicates definable in Presburger arithmetic, i.e. $\text{FO}(\mathbb{N}, +, <)$  

Angluin, Aspnes, Eisenstat PODC’06
Formal 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. \( \text{weight}(x) = \begin{cases} |x| & \text{if } x \in \text{StrongStates} \text{ or } x \in \text{WeakStates}; \\ 1 & \text{if } x \in \text{IntermediateStates}. \end{cases} \)

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

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

4. \( \phi(x) = -1_j \text{ if } x = -1; 1_j \text{ if } x = 1; \text{ otherwise} \)

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

6. \( R_1(k) = \phi(k \text{ if } k \text{ odd integer}, k - 1 \text{ if } k \text{ even}) \)

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

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

9. **procedure** update\((x, y)\)

10. **if** \((\text{weight}(x) > 0 \text{ and } \text{weight}(y) > 1) \text{ or } (\text{weight}(y) > 0 \text{ and } \text{weight}(x) > 1)\) **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** \((\text{weight}(x) \cdot \text{weight}(y) = 0 \text{ and } \text{value}(x) + \text{value}(y) > 0)\) **then**

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

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

16. \(x' \leftarrow -0 \text{ and } y' \leftarrow +0\)

17. **else**

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

```latex
\begin{Verbatim}
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}
\end{Verbatim}
```

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

```latex
\begin{Verbatim}
value(x) = \text{sign}(x) \cdot \text{weight}(x)
\end{Verbatim}
```

```latex
\begin{Verbatim}
ϕ(x) = -1_1 \text{ if } x = -1_1; 1, \text{ if } x = 1_1; x, \text{ otherwise}
\end{Verbatim}
```

```latex
\begin{Verbatim}
R_1(k) = ϕ(k \text{ if } k \text{ odd integer, } k - 1 \text{ if } k \text{ even})
\end{Verbatim}
```

```latex
\begin{Verbatim}
R_1^+(k) = ϕ(k \text{ if } k \text{ odd integer, } k + 1 \text{ if } k \text{ even})
\end{Verbatim}
```

```latex
\begin{Verbatim}
\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}
\end{Verbatim}
```

```latex
\begin{Verbatim}
\text{Sign-to-Zero}(x) = \begin{cases} 
+0 & \text{if } \text{sign}(x) > 0 \\
-0 & \text{otherwise}.
\end{cases}
\end{Verbatim}
```

```latex
\begin{Verbatim}
\text{update}(x, y) = \begin{cases} 
\text{if } \text{weight}(x) > 0 \text{ and } \text{weight}(y) > 1 \text{ or } \text{weight}(y) > 0 \text{ and } \text{weight}(x) > 1 \text{ then} \\
x' = R_1\left(\frac{\text{value}(x) + \text{value}(y)}{2}\right) \quad \text{and} \quad y' = R_1^+\left(\frac{\text{value}(x) + \text{value}(y)}{2}\right)
\end{cases}
\end{Verbatim}
```

```latex
\begin{Verbatim}
\text{else if } \text{weight}(x) \cdot \text{weight}(y) = 0 \text{ and } \text{value}(x) + \text{value}(y) > 0 \text{ then} \\
\text{if } \text{weight}(x) \neq 0 \text{ then } x' = \text{Shift-to-Zero}(x) \text{ and } y' = \text{Sign-to-Zero}(x) \\
\text{else } y' = \text{Shift-to-Zero}(y) \text{ and } x' = \text{Sign-to-Zero}(y)
\end{Verbatim}
```

```latex
\begin{Verbatim}
\text{else if } \{x \in \{-1_d, +1_d\} \text{ and } \text{weight}(y) = 1 \text{ and } \text{sign}(x) \neq \text{sign}(y)\} \text{ or} \\
\{y \in \{-1_d, +1_d\} \text{ and } \text{weight}(x) = 1 \text{ and } \text{sign}(y) \neq \text{sign}(x)\} \text{ then} \\
x' = -0 \text{ and } y' = +0
\end{Verbatim}
```

```latex
\begin{Verbatim}
\text{else} \\
x' = \text{Shift-to-Zero}(x) \text{ and } y' = \text{Shift-to-Zero}(y)
\end{Verbatim}
```
Convergence speed may vary wildly, challenging to establish bounds

The expected number of steps to stable consensus varies significantly with the number of agents initially in state $\mathcal{R}$. The bar chart shows the distribution of expected steps for different numbers of agents, with the y-axis representing the expected number of steps and the x-axis indicating the number of agents in state $\mathcal{R}$. The expected steps are measured on a logarithmic scale, indicating the diversity in convergence times.
Formal analysis of protocols

Convergence speed may vary wildly, challenging to establish bounds

How to derive asymptotic bounds automatically?
Formal 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 (Mertzios et al. ICALP’14)

- $X \geq c$ requires at most $c + 1$ states
Formal 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 (Mertzios et al. ICALP’14)

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

What is the state complexity of common predicates?
Formal analysis of protocols

1. **Automatic verification of correctness**
   - Decidability
     - Esparza, Ganty, Leroux, Majumdar CONCUR’15, FSTTCS’16
   - Towards efficient verification
     - B., Esparza, Jaax, Meyer PODC’17
   - Complete tool
     - B., Esparza, Jaax CAV’18

2. **Automatic analysis of convergence speed**
   - First procedure
     - B., Esparza, Kučera (submitted to CONCUR’18)

3. **State complexity of protocols w.r.t. predicates**
   - Study of linear inequalities
     - B., Esparza, Jaax STACS’18
Formal analysis of protocols

1. Automatic verification of correctness
   - Decidability
     Esparza, Ganty, Leroux, Majumdar CONCUR’15, FSTTCS’16
   - Towards efficient verification
     B., Esparza, Jaax, Meyer PODC’17
   - Complete tool
     B., Esparza, Jaax CAV’18

2. Automatic analysis of convergence speed
   - First procedure
     B., Esparza, Kučera (submitted to CONCUR’18)

3. State complexity of protocols w.r.t. predicates
   - Study of linear inequalities
     B., Esparza, Jaax STACS’18
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)

- 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:

\[
\neg \exists C, D : \quad C \xrightarrow{\ast} D \land C \text{ is initial} \land D \text{ is bottom} \land \text{opinion}(D) \neq \varphi(C)
\]
Testing whether a protocol computes $\varphi$ amounts to testing:

$\neg \exists C, D : \begin{array}{c}
C \xrightarrow{*} D \\
C \text{ is initial} \\
D \text{ is bottom} \\
\text{opinion}(D) \neq \varphi(C)
\end{array}$

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

$$\neg \exists C, D : \quad C \overset{*}{\longrightarrow} D \land$$
$$\quad C \text{ is initial} \land$$
$$\quad D \text{ is bottom} \land$$
$$\quad \text{opinion}(D) \neq \varphi(C)$$

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

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

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

Difficult to express
Testing whether a protocol computes $\varphi$ amounts to testing:

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

$C$ is initial $\land$

$D$ is terminal $\land$

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

Most protocols are terminating!
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) $\neq \varphi(C)$

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

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

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

Protocol termination tested by structural analysis + SMT solving
Random variable $Steps$: 

assigns to each run $\sigma$ the smallest $k$ s.t. $\sigma_k$ in stable consensus

**Maximal expected termination time**

We are interested in $time: \mathbb{N} \rightarrow \mathbb{N}$ where

$$time(n) = \max\{\mathbb{E}_C[Steps] : C \text{ is initial and } |C| = n\}$$
Our approach:

• Most protocols are naturally designed in stages

• Construct these stages automatically

• Derive upper bounds on $\text{time}(n)$ from stages structure
Analysis of termination time

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

\[ \begin{align*}
\square (B \land \bigwedge_{q \neq B} \neg q) & \rightarrow \Theta(1) \\
\square (R \land \bigwedge_{q \neq R} \neg q) & \rightarrow \Theta(1) \\
\square (\neg B \lor \neg R) \land b \land \neg b! & \rightarrow \Theta(n^2 \log n)
\end{align*} \]

\[ \begin{align*}
\square (\neg B \land \neg R \land b \land \neg r) & \rightarrow \Theta(n^2 \log n) \\
\square (B \land \neg R \land b \land \neg r) & \rightarrow \Theta(n^2 \log n) \\
\square (\neg B \land R \land \neg b \land r) & \rightarrow \Theta(exp(n))
\end{align*} \]
• Prototype implemented in **Python** + Microsoft Z3

• Can report: $O(1), O(n^2), O(n^2 \log n), O(n^3), O(\text{poly}(n))$ or $O(\text{exp}(n))$

• Tested on various protocols from the literature
Peregrine: Haskell + Microsoft Z3 + JavaScript

peregrine.model.in.tum.de

- Design of protocols
- Manual and automatic simulation
- Statistics of properties such as termination time
- Automatic verification of correctness
- More to come!
Demonstration
Population protocols can be formally analyzed automatically:

- Verification of correctness
- Analysis of expected termination time
- Tool support!

Ongoing investigation of state complexity
Conclusion: future work (seeking for PhD students/Postdocs)

ERC Advanced Grant —

PaVeS: Parameterized Verification and Synthesis

• Goal: Develop proof and synthesis techniques for distributed algorithms working correctly for an arbitrary number of processes

• PI: Javier Esparza (esparza@in.tum.de), TU Munich

• Start of the project: Sept. 1, 2018

• Start of the PhDs/Postdocs: flexible, from Sept. 1, 2018 to about Sept. 1, 2019
Thank you!
Vielen Dank!