Automatic Analysis of Population Protocols

Michael Blondin

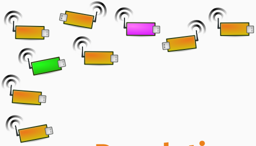
Joint work with Javier Esparza, Stefan Jaax, Antonín Kučera, Philipp J. Meyer

UNIVERSITÉ DE
SHERBROOKE

Overview

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

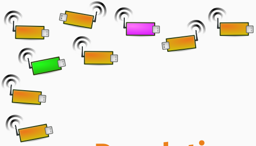
Overview



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

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

Overview



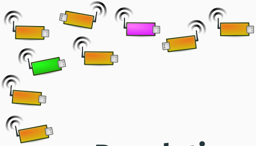
Population protocols: distributed computing model for massive networks of passively mobile finite-state agents

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

Protocols compute predicates of the form $\varphi\colon \mathbb{N}^d \to \{0, 1\}$

e.g. $\varphi(m, n)$ is computed by $m + n$ agents

Overview



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

This talk: automatic verification and expected termination time analysis

Population protocols

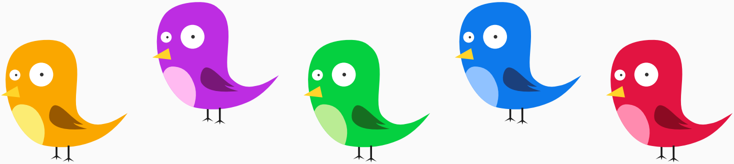
Angluin *et al.* PODC'04

- 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

Angluin *et al.* PODC'04

- 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

Angluin *et al.* PODC'04

- 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

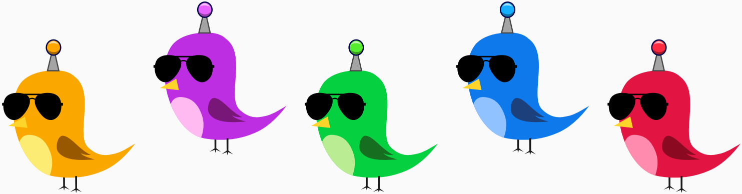
Angluin *et al.* PODC’04

- 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

Angluin *et al.* PODC'04

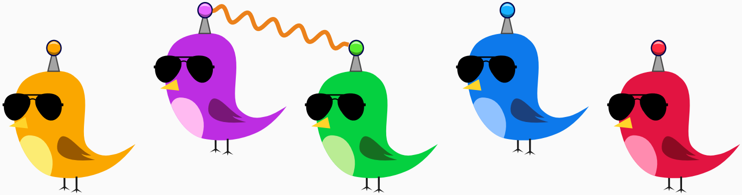
- anonymous mobile agents with very few resources
- agents change states via random pairwise interactions
- each agent has opinion true/false
- computes by stabilizing agents to some opinion



Population protocols

Angluin *et al.* PODC'04

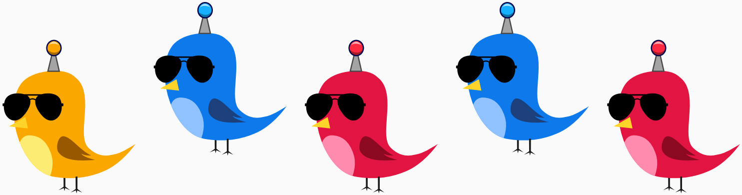
- anonymous mobile agents with very few resources
- agents change states via random pairwise interactions
- each agent has opinion true/false
- computes by stabilizing agents to some opinion



Population protocols

Angluin *et al.* PODC'04

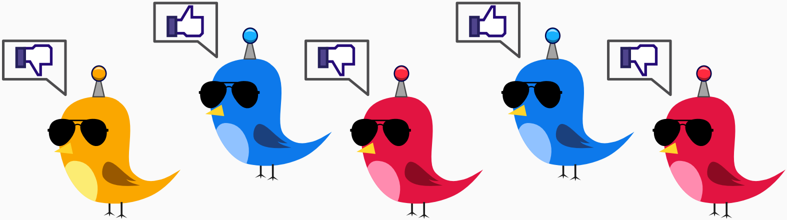
- anonymous mobile agents with very few resources
- agents change states via random pairwise interactions
- each agent has opinion true/false
- computes by stabilizing agents to some opinion



Population protocols

Angluin *et al.* PODC'04

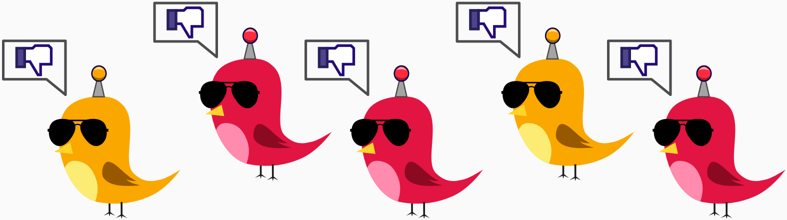
- anonymous mobile agents with very few resources
- agents change states via random pairwise interactions
- each agent has opinion true/false
- computes by stabilizing agents to some opinion



Population protocols

Angluin *et al.* PODC'04

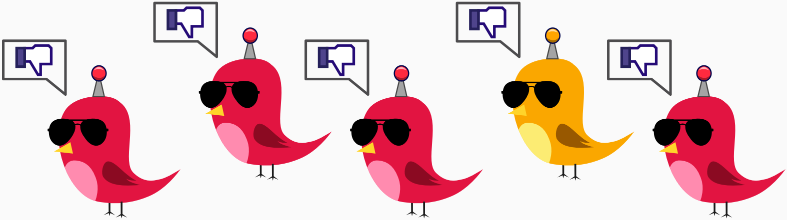
- anonymous mobile agents with very few resources
- agents change states via random pairwise interactions
- each agent has opinion true/false
- computes by stabilizing agents to some opinion



Population protocols

Angluin *et al.* PODC'04

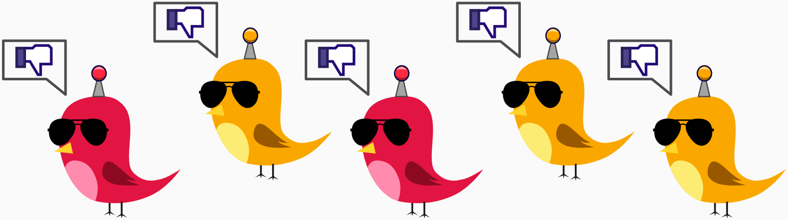
- anonymous mobile agents with very few resources
- agents change states via random pairwise interactions
- each agent has opinion true/false
- computes by stabilizing agents to some opinion



Population protocols

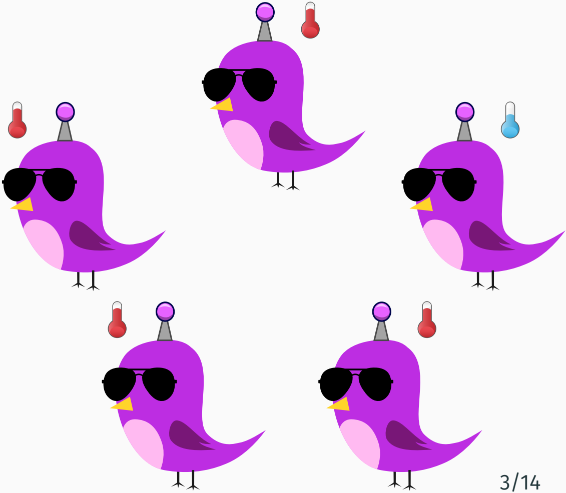
Angluin *et al.* PODC'04

- 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: threshold protocol

Are there at least 4 sick birds?

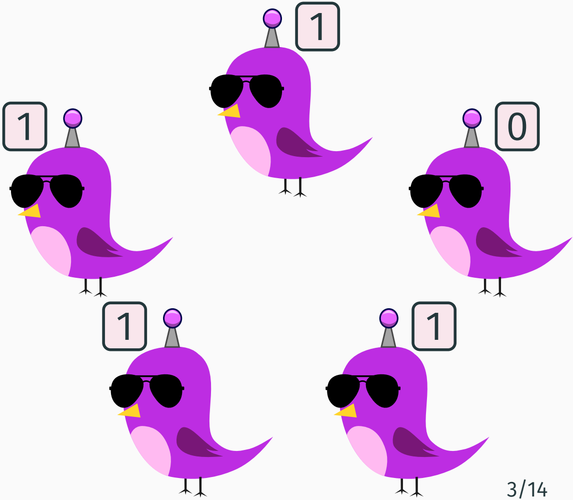


Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

- Each agent in a state of $\{0, 1, 2, 3, 4\}$
- $(m, n) \mapsto (m + n, 0)$
  if $m + n < 4$
- $(m, n) \mapsto (4, 4)$
  if $m + n \geq 4$

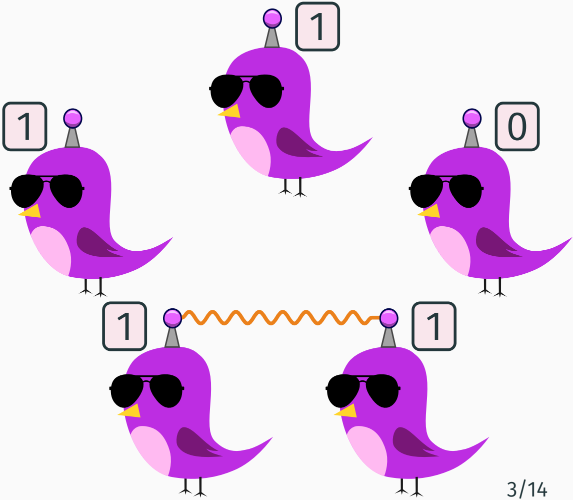


Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

- Each agent in a state of $\{0, 1, 2, 3, 4\}$
- $(m, n) \mapsto (m + n, 0)$
  if $m + n < 4$
- $(m, n) \mapsto (4, 4)$
  if $m + n \geq 4$

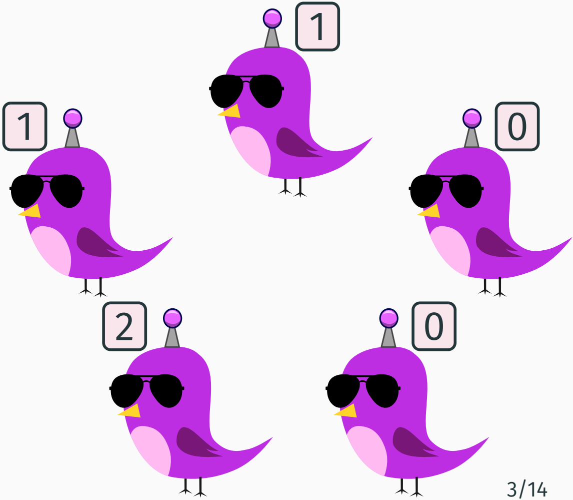


Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

- Each agent in a state of $\{0, 1, 2, 3, 4\}$
- $(m, n) \mapsto (m + n, 0)$
  if $m + n < 4$
- $(m, n) \mapsto (4, 4)$
  if $m + n \geq 4$

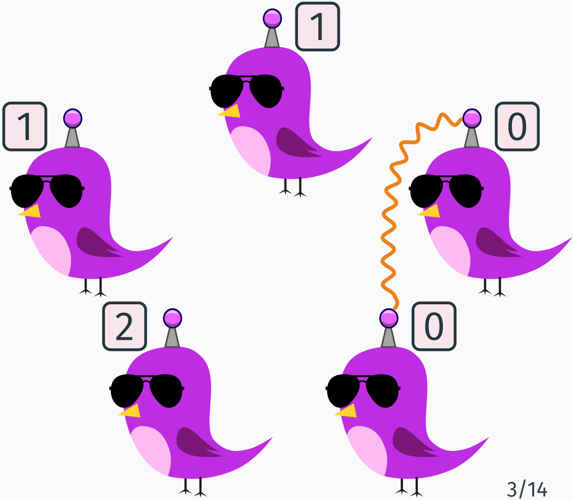


Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

- Each agent in a state of $\{0, 1, 2, 3, 4\}$
- $(m, n) \mapsto (m + n, 0)$ if $m + n < 4$
- $(m, n) \mapsto (4, 4)$ if $m + n \geq 4$

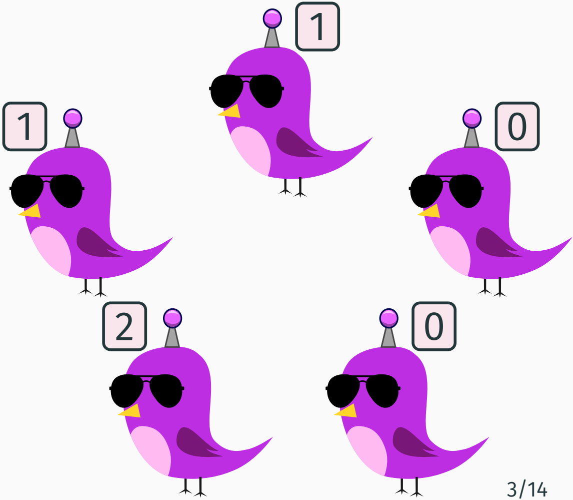


Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

- Each agent in a state of $\{0, 1, 2, 3, 4\}$
- $(m, n) \mapsto (m + n, 0)$
  if $m + n < 4$
- $(m, n) \mapsto (4, 4)$
  if $m + n \geq 4$

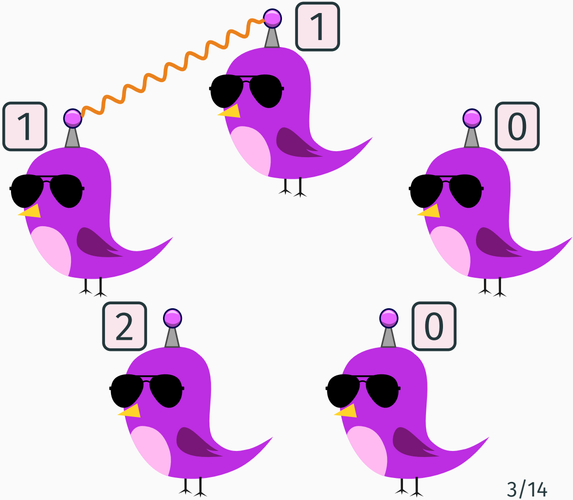


Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

- Each agent in a state of $\{0, 1, 2, 3, 4\}$
- $(m, n) \mapsto (m+n, 0)$
  if $m + n < 4$
- $(m, n) \mapsto (4, 4)$
  if $m + n \geq 4$

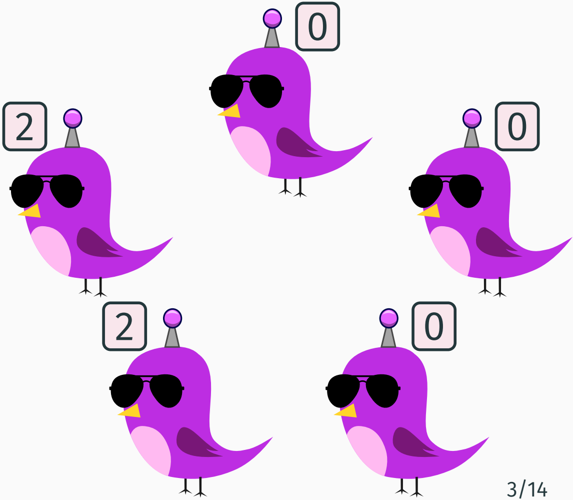


Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

- Each agent in a state of $\{0, 1, 2, 3, 4\}$
- $(m, n) \mapsto (m + n, 0)$ if $m + n < 4$
- $(m, n) \mapsto (4, 4)$ if $m + n \geq 4$

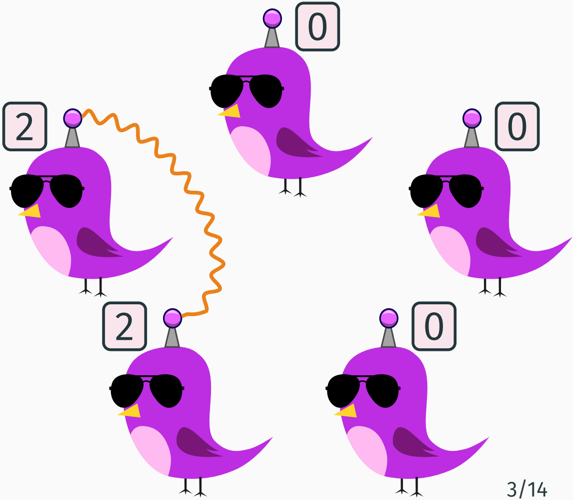


Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

- Each agent in a state of $\{0, 1, 2, 3, 4\}$
- $(m, n) \mapsto (m + n, 0)$ if $m + n < 4$
- $(m, n) \mapsto (4, 4)$ if $m + n \geq 4$

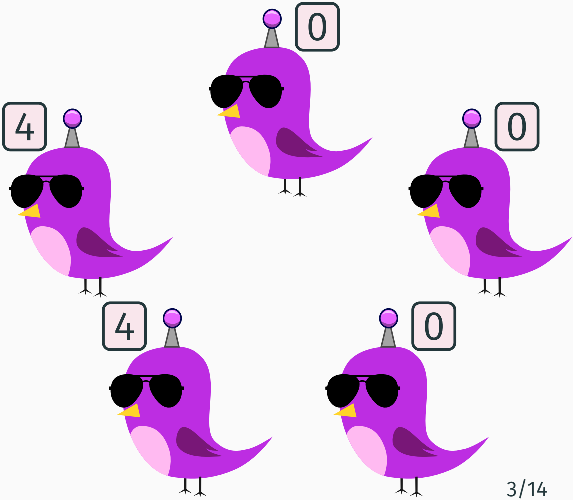


Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

- Each agent in a state of $\{0, 1, 2, 3, 4\}$
- $(m, n) \mapsto (m + n, 0)$
  if $m + n < 4$
- $(m, n) \mapsto (4, 4)$
  if $m + n \geq 4$

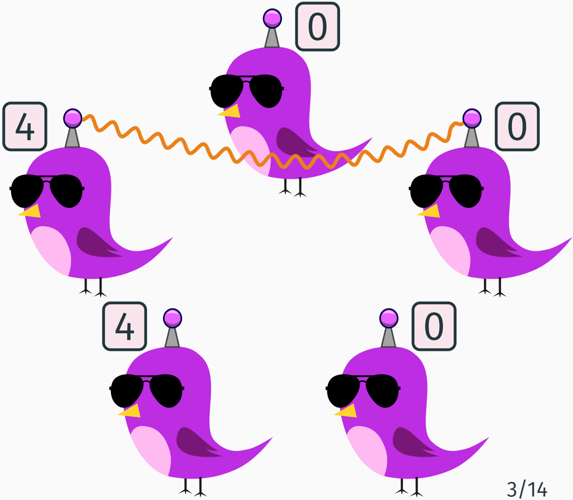


Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

- Each agent in a state of $\{0, 1, 2, 3, 4\}$
- $(m, n) \mapsto (m + n, 0)$
  if $m + n < 4$
- $(m, n) \mapsto (4, 4)$
  if $m + n \geq 4$

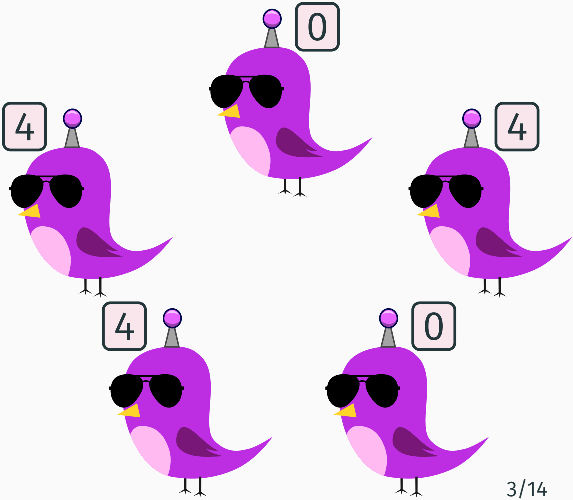


Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

- Each agent in a state of $\{0, 1, 2, 3, 4\}$
- $(m, n) \mapsto (m + n, 0)$
  if $m + n < 4$
- $(m, n) \mapsto (4, 4)$
  if $m + n \geq 4$

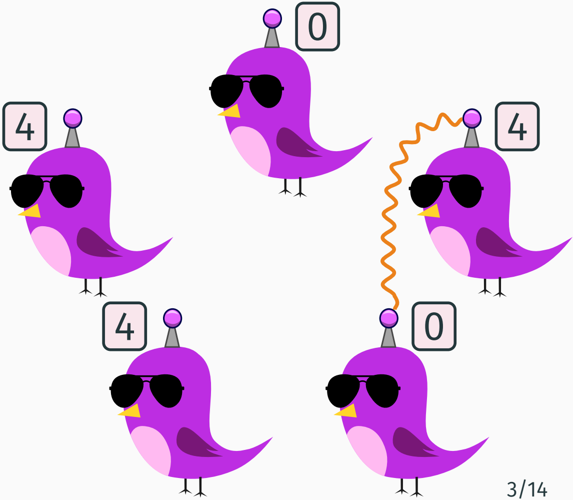


Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

- Each agent in a state of $\{0, 1, 2, 3, 4\}$
- $(m, n) \mapsto (m + n, 0)$
  if $m + n < 4$
- $(m, n) \mapsto (4, 4)$
  if $m + n \geq 4$

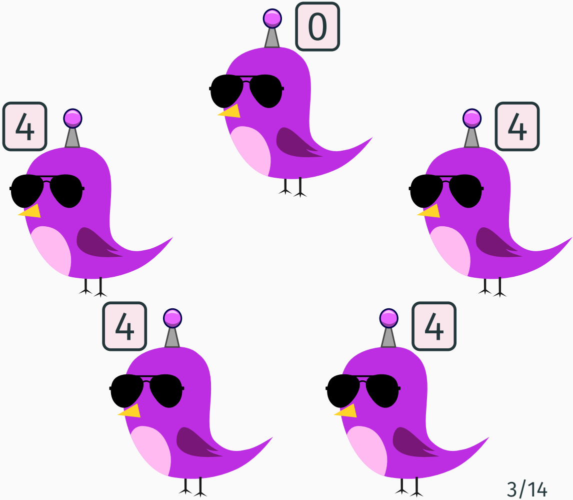


Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

- Each agent in a state of $\{0, 1, 2, 3, 4\}$
- $(m, n) \mapsto (m + n, 0)$ if $m + n < 4$
- $(m, n) \mapsto (4, 4)$ if $m + n \geq 4$

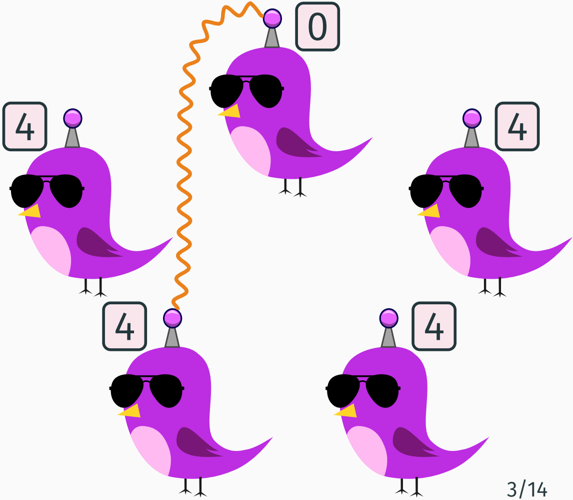


Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

- Each agent in a state of $\{0, 1, 2, 3, 4\}$
- $(m, n) \mapsto (m + n, 0)$
  if $m + n < 4$
- $(m, n) \mapsto (4, 4)$
  if $m + n \geq 4$

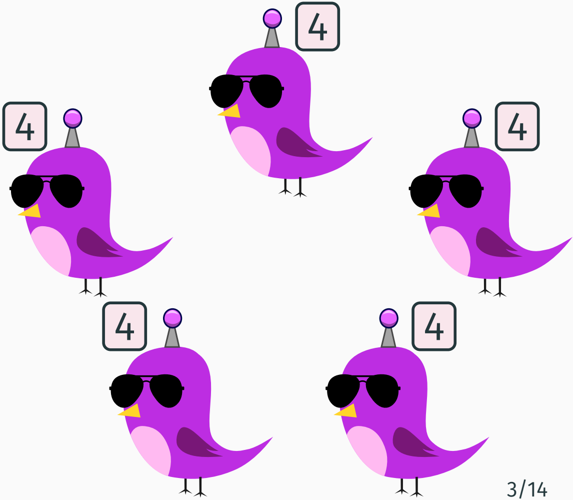


Example: threshold protocol

Are there at least 4 sick birds?

Protocol:

- Each agent in a state of $\{0, 1, 2, 3, 4\}$
- $(m, n) \mapsto (m + n, 0)$
  if $m + n < 4$
- $(m, n) \mapsto (4, 4)$
  if $m + n \geq 4$

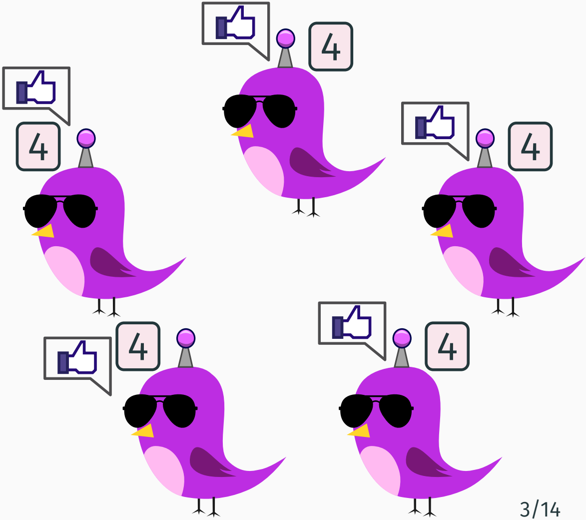


Example: threshold protocol

Are there at least 4 sick birds?

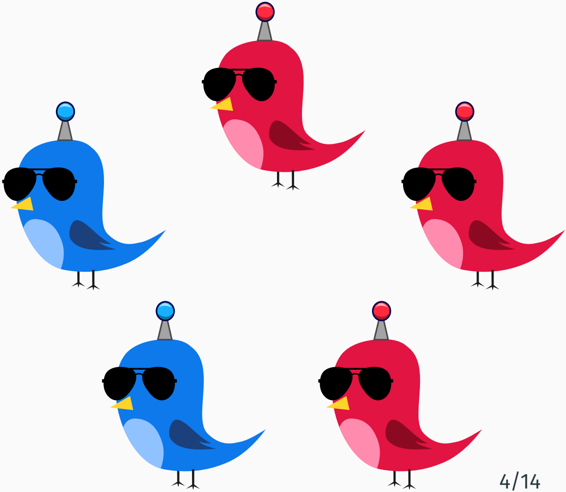
Protocol:

- Each agent in a state of $\{0, 1, 2, 3, 4\}$
- $(m, n) \mapsto (m + n, 0)$
  if $m + n < 4$
- $(m, n) \mapsto (4, 4)$
  if $m + n \geq 4$



Example: majority protocol

blue agents ≥ # red agents?

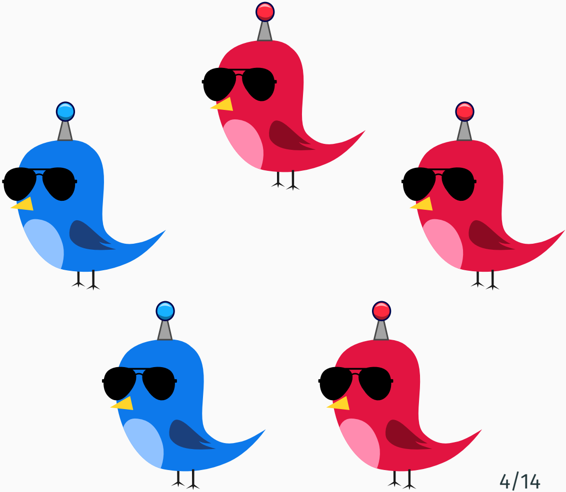


Example: majority protocol

blue agents ≥ # red agents?

Protocol:

- Two large agents become small blue agents
- Large agents convert small agents to their colour

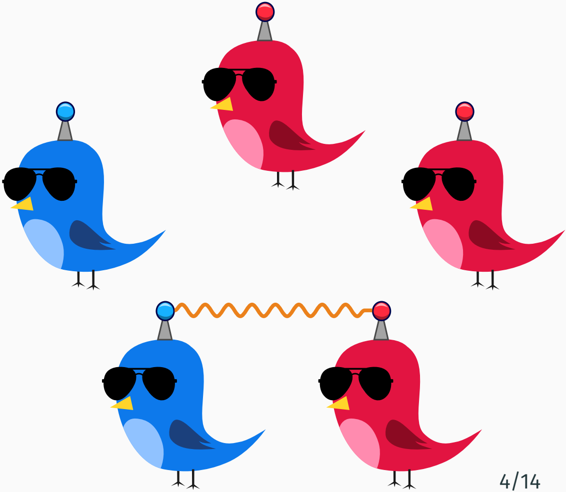


Example: majority protocol

blue agents ≥ # red agents?

Protocol:

- Two large agents become small blue agents
- Large agents convert small agents to their colour

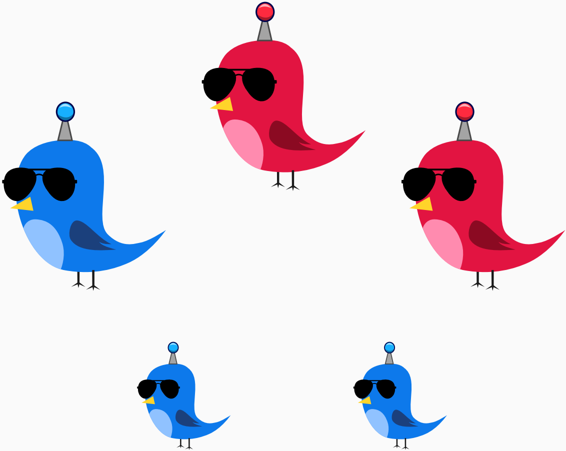


Example: majority protocol

blue agents ≥ # red agents?

Protocol:

- Two large agents become small blue agents
- Large agents convert small agents to their colour

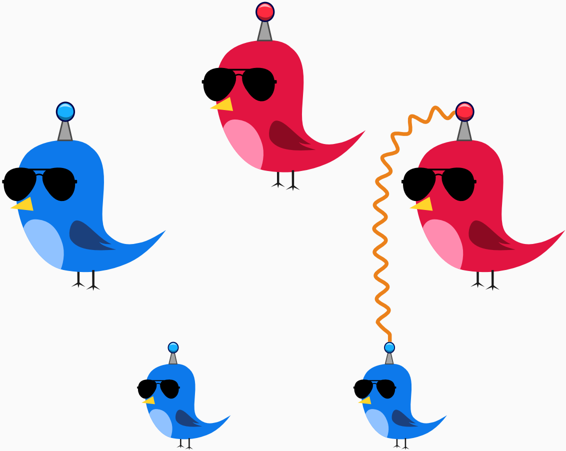


Example: majority protocol

blue agents ≥ # red agents?

Protocol:

- Two large agents become small blue agents
- Large agents convert small agents to their colour

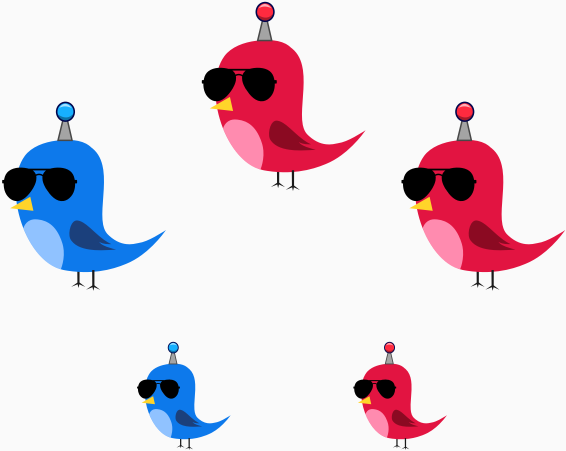


Example: majority protocol

blue agents ≥ # red agents?

Protocol:

- Two large agents become small blue agents
- Large agents convert small agents to their colour

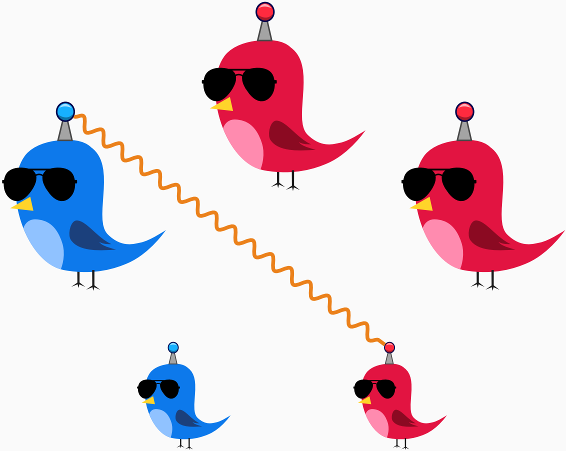


Example: majority protocol

blue agents ≥ # red agents?

Protocol:

- Two large agents become small blue agents
- Large agents convert small agents to their colour

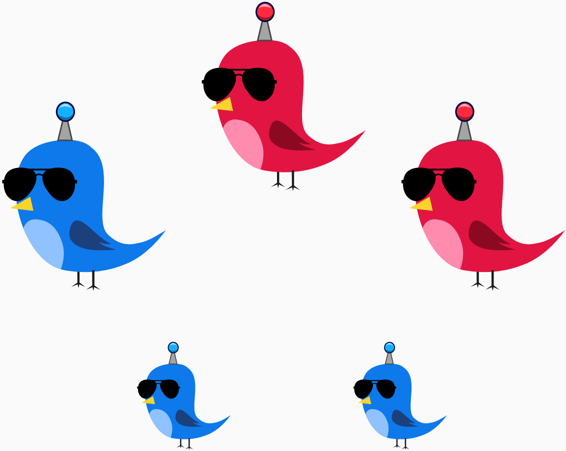


Example: majority protocol

blue agents ≥ # red agents?

Protocol:

- Two large agents become small blue agents
- Large agents convert small agents to their colour

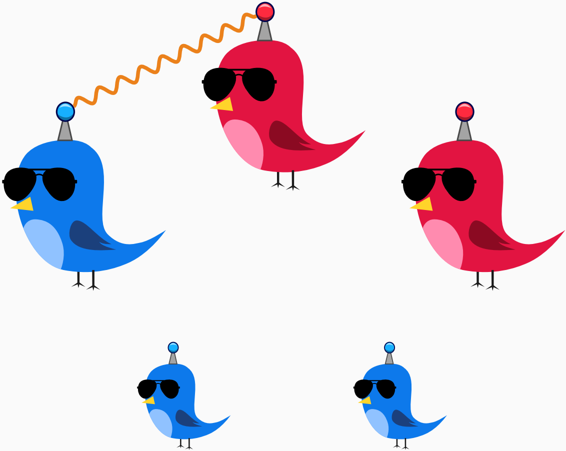


Example: majority protocol

blue agents ≥ # red agents?

Protocol:

- Two large agents become small blue agents
- Large agents convert small agents to their colour

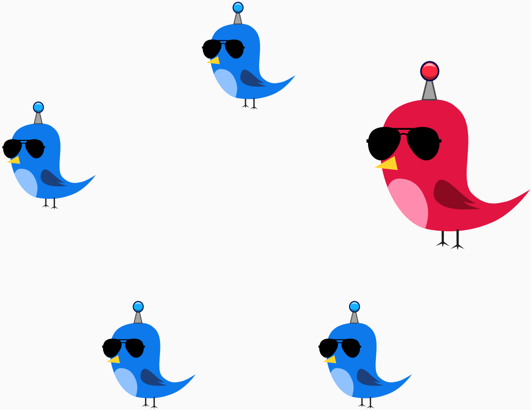


Example: majority protocol

blue agents ≥ # red agents?

Protocol:

- Two large agents become small blue agents
- Large agents convert small agents to their colour



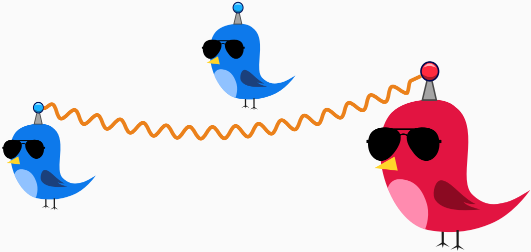
Example: majority protocol

blue agents ≥ # red agents?

Protocol:

- Two large agents become small blue agents

- Large agents convert small agents to their colour



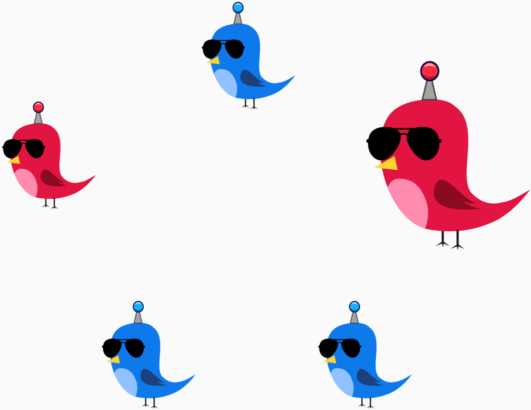
Example: majority protocol

blue agents ≥ # red agents?

Protocol:

- Two large agents become small blue agents

- Large agents convert small agents to their colour



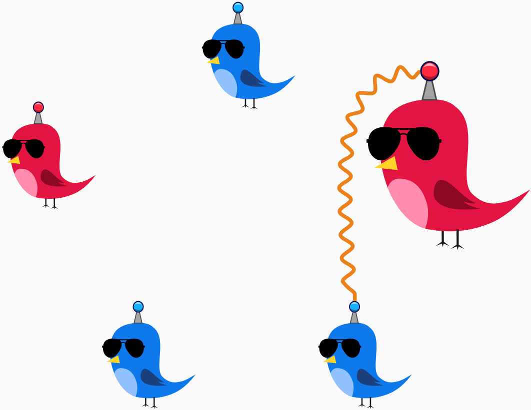
Example: majority protocol

blue agents ≥ # red agents?

Protocol:

- Two large agents become small blue agents

- Large agents convert small agents to their colour

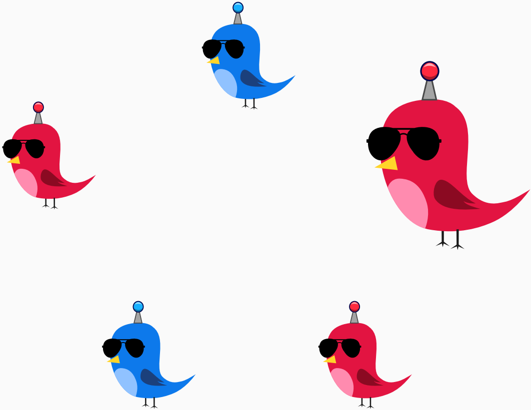


Example: majority protocol

 # blue agents ≥ # red agents?

Protocol:

- Two large agents become small blue agents
- Large agents convert small agents to their colour

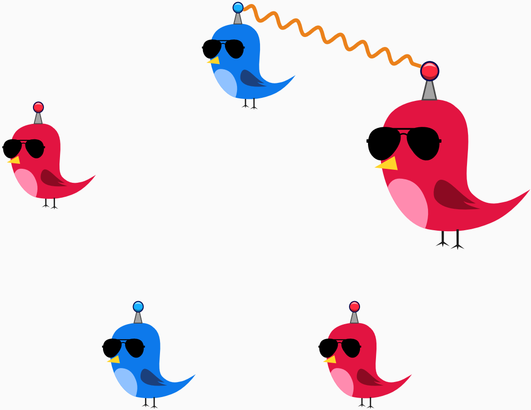


Example: majority protocol

blue agents ≥ # red agents?

Protocol:

- Two large agents become small blue agents
- Large agents convert small agents to their colour



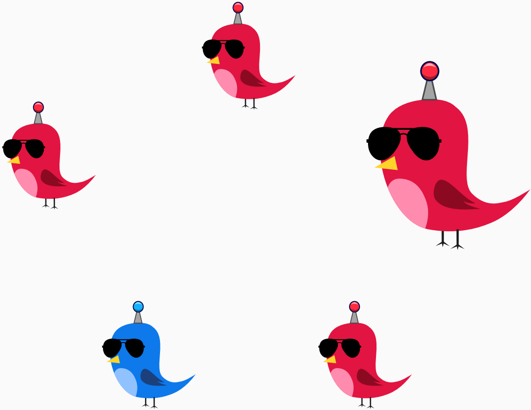
Example: majority protocol

blue agents ≥ # red agents?

Protocol:

- Two large agents become small blue agents

- Large agents convert small agents to their colour

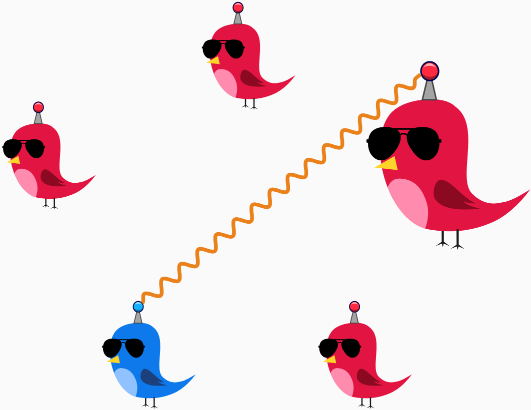


Example: majority protocol

blue agents ≥ # red agents?

Protocol:

- Two large agents become small blue agents
- Large agents convert small agents to their colour

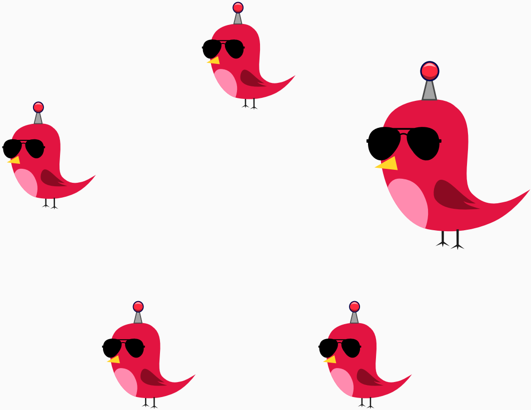


Example: majority protocol

blue agents ≥ # red agents?

Protocol:

- Two large agents become small blue agents
- Large agents convert small agents to their colour

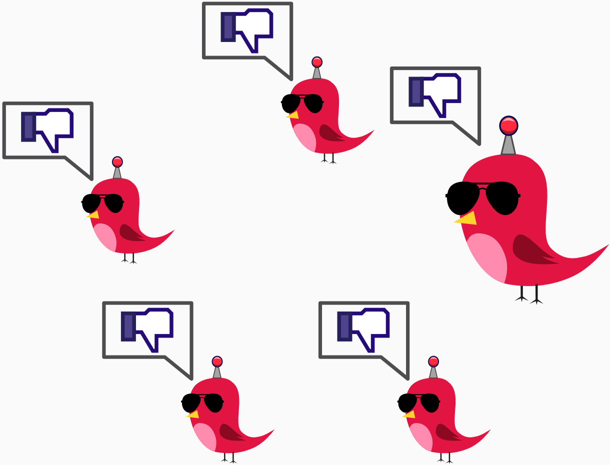


Example: majority protocol

blue agents ≥ # red agents?

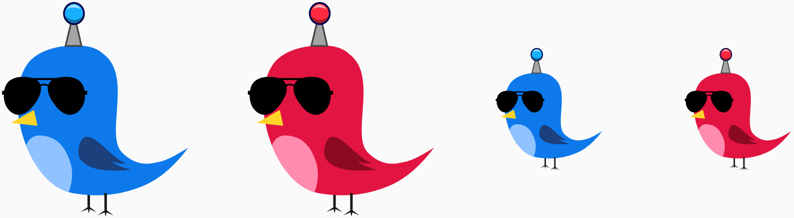
Protocol:

- Two large agents become small blue agents
- Large agents convert small agents to their colour



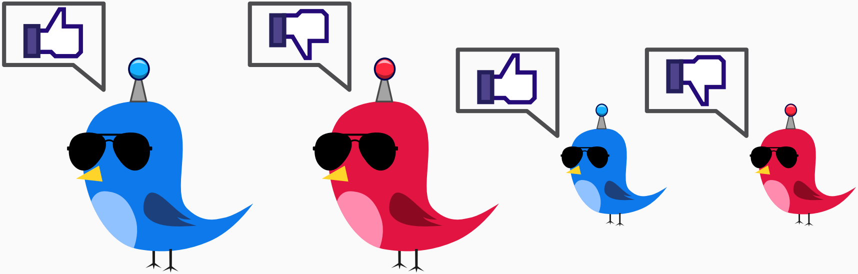
Population protocols: formal model

- *States*: finite set Q
- *Opinions*: $O\colon Q \to \{\texttt{false}, \texttt{true}\}$
- *Initial states*: $I \subseteq Q$
- *Transitions*: $T \subseteq Q^2 \times Q^2$



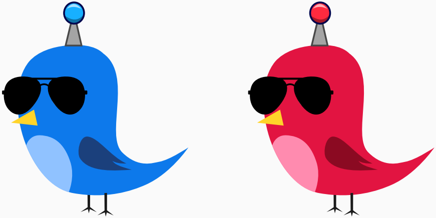
Population protocols: formal model

- *States*: finite set Q
- *Opinions*: $O\colon Q \to \{\texttt{false}, \texttt{true}\}$
- *Initial states*: $I \subseteq Q$
- *Transitions*: $T \subseteq Q^2 \times Q^2$



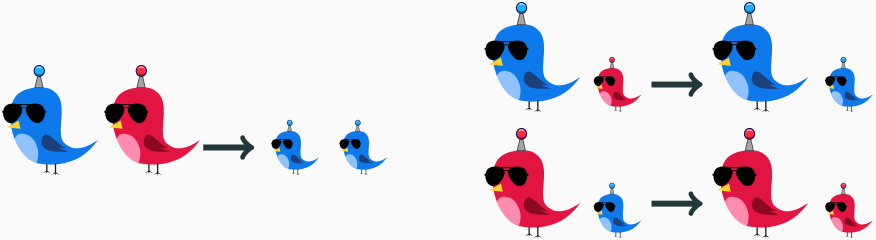
Population protocols: formal model

- *States*: finite set Q
- *Opinions*: $O\colon Q \to \{\texttt{false}, \texttt{true}\}$
- *Initial states*: $I \subseteq Q$
- *Transitions*: $T \subseteq Q^2 \times Q^2$



Population protocols: formal model

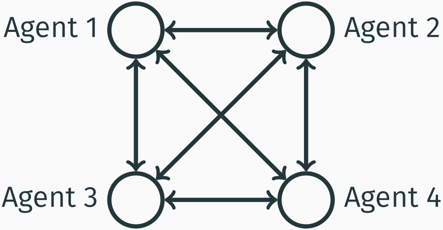
- *States*: finite set Q
- *Opinions*: $O\colon Q \to \{\texttt{false}, \texttt{true}\}$
- *Initial states*: $I \subseteq Q$
- *Transitions*: $T \subseteq Q^2 \times Q^2$



Population protocols: interactions

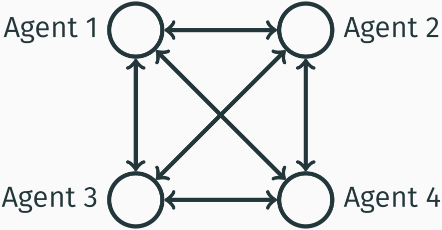
All agents can interact pairwise

(complete topology)



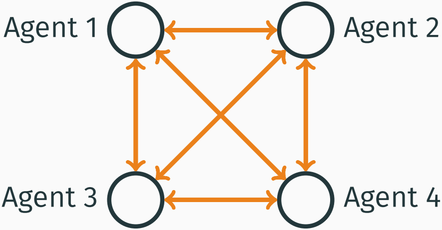
Population protocols: interactions

$$\mathbb{P}[\text{fire } p, q \mapsto p', q' \text{ in } C] = \begin{cases} \dfrac{2 \cdot C(p) \cdot C(q)}{n^2 - n} & \text{if } p \neq q \\[2ex] \dfrac{C(p) \cdot (C(p) - 1)}{n^2 - n} & \text{if } p = q \end{cases}$$



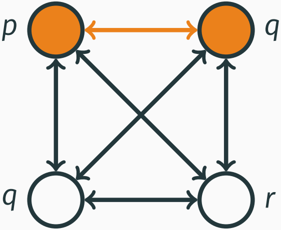
Population protocols: interactions

$$\mathbb{P}[\text{fire } p, q \mapsto p', q' \text{ in } C] = \begin{cases} \dfrac{2 \cdot C(p) \cdot C(q)}{n^2 - n} & \text{if } p \neq q \\ \dfrac{C(p) \cdot (C(p) - 1)}{n^2 - n} & \text{if } p = q \end{cases}$$



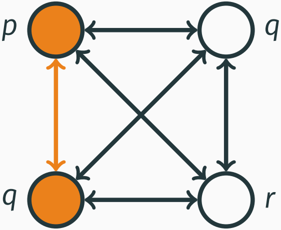
Population protocols: interactions

$$\mathbb{P}[\text{fire } p, q \mapsto p', q' \text{ in } C] = \begin{cases} \dfrac{2 \cdot C(p) \cdot C(q)}{n^2 - n} & \text{if } p \neq q \\[2ex] \dfrac{C(p) \cdot (C(p) - 1)}{n^2 - n} & \text{if } p = q \end{cases}$$



Population protocols: interactions

$$\mathbb{P}[\text{fire } p, q \mapsto p', q' \text{ in } C] = \begin{cases} \frac{2 \cdot C(p) \cdot C(q)}{n^2 - n} & \text{if } p \neq q \\ \frac{C(p) \cdot (C(p) - 1)}{n^2 - n} & \text{if } p = q \end{cases}$$



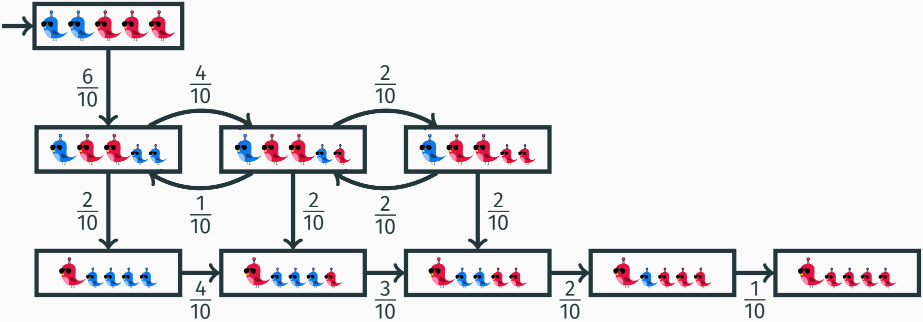
Population protocols: interactions

$$\mathbb{P}[\text{fire } p, q \mapsto p', q' \text{ in } C] \;=\; \begin{cases} \dfrac{2 \cdot C(p) \cdot C(q)}{n^2 - n} & \text{if } p \neq q \\[2ex] \dfrac{C(p) \cdot (C(p) - 1)}{n^2 - n} & \text{if } p = q \end{cases}$$

$$\mathbb{P}[C \to C'] \;=\; \sum_{t \text{ s.t. } C \xrightarrow{t} C'} \mathbb{P}[\text{fire } t \text{ in } C]$$

Population protocols: computations

Underlying Markov chain:

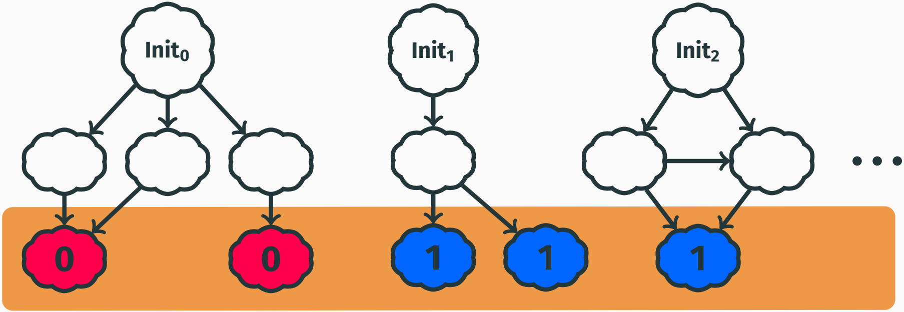


Population protocols: computations

A protocol computes a predicate $f\colon \mathbb{N}^I \to \{0,1\}$

if runs reach **common stable consensus**

with probability 1



Population protocols: computations

A protocol computes a predicate $f\colon \mathbb{N}^I \rightarrow \{0,1\}$

if runs reach **common stable consensus**

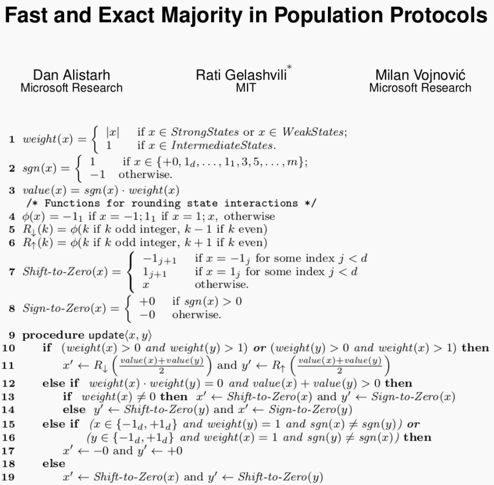
with probability 1

Expressive power **Angluin, Aspnes, Eisenstat PODC'06**

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

Verifying correctness

Protocols can become complex, even for B ≥ R:



Fast and Exact Majority in Population Protocols

Dan Alistarh
Microsoft Research

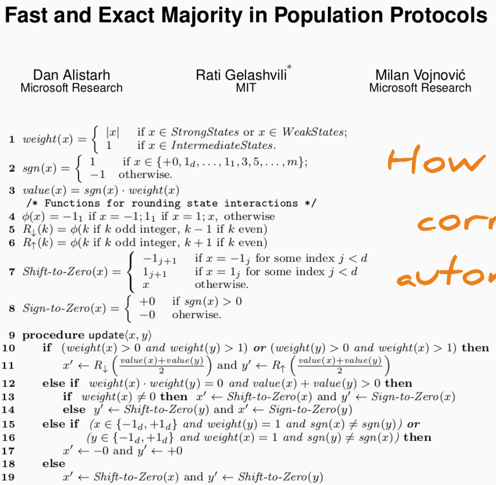
Rati Gelashvili*
MIT

Milan Vojnović
Microsoft Research

```
$weight(x) = \begin{cases} |x| & \text{if } x \in StrongStates \text{ or } x \in WeakStates; \\ 1 & \text{if } x \in IntermediateStates. \end{cases}$
$sgn(x) = \begin{cases} 1 & \text{if } x \in \{+0, 1_d, \dots, 1_1, 3, 5, \dots, m\}; \\ -1 & \text{otherwise.} \end{cases}$
$value(x) = sgn(x) \cdot weight(x)$
 /* Functions for rounding state interactions */
$\phi(x) = -1_1$ if $x = -1$; $1_1$ if $x = 1$; $x$, otherwise
$R_\downarrow(k) = \phi(k$ if $k$ odd integer, $k - 1$ if $k$ even$)$
$R_\uparrow(k) = \phi(k$ if $k$ odd integer, $k + 1$ if $k$ even$)$
$Shift\text{-}to\text{-}Zero(x) = \begin{cases} -1_{j+1} & \text{if } x = -1_j \text{ for some index } j < d \\ 1_{j+1} & \text{if } x = 1_j \text{ for some index } j < d \\ x & \text{otherwise.} \end{cases}$
$Sign\text{-}to\text{-}Zero(x) = \begin{cases} +0 & \text{if } sgn(x) > 0 \\ -0 & \text{oherwise.} \end{cases}$
procedure update$\langle x, y \rangle$
    if $(weight(x) > 0$ and $weight(y) > 1)$ or $(weight(y) > 0$ and $weight(x) > 1)$ then
        $x' \leftarrow R_\downarrow\left(\frac{value(x)+value(y)}{2}\right)$ and $y' \leftarrow R_\uparrow\left(\frac{value(x)+value(y)}{2}\right)$
    else if $weight(x) \cdot weight(y) = 0$ and $value(x) + value(y) > 0$ then
        if $weight(x) \neq 0$ then $x' \leftarrow Shift\text{-}to\text{-}Zero(x)$ and $y' \leftarrow Sign\text{-}to\text{-}Zero(x)$
        else $y' \leftarrow Shift\text{-}to\text{-}Zero(y)$ and $x' \leftarrow Sign\text{-}to\text{-}Zero(y)$
    else if $(x \in \{-1_d, +1_d\}$ and $weight(y) = 1$ and $sgn(x) \neq sgn(y))$ or
            $(y \in \{-1_d, +1_d\}$ and $weight(x) = 1$ and $sgn(y) \neq sgn(x))$ then
        $x' \leftarrow -0$ and $y' \leftarrow +0$
    else
        $x' \leftarrow Shift\text{-}to\text{-}Zero(x)$ and $y' \leftarrow Shift\text{-}to\text{-}Zero(y)$
```

Verifying correctness

Protocols can become complex, even for B ≥ R:



Fast and Exact Majority in Population Protocols

Dan Alistarh
Microsoft Research

Rati Gelashvili*
MIT

Milan Vojnović
Microsoft Research

```
$weight(x) = \begin{cases} |x| & \text{if } x \in StrongStates \text{ or } x \in WeakStates; \\ 1 & \text{if } x \in IntermediateStates. \end{cases}$
$sgn(x) = \begin{cases} 1 & \text{if } x \in \{+0, 1_d, \ldots, 1_1, 3, 5, \ldots, m\}; \\ -1 & \text{otherwise.} \end{cases}$
$value(x) = sgn(x) \cdot weight(x)$
 /* Functions for rounding state interactions */
$\phi(x) = -1_1 \text{ if } x = -1; 1_1 \text{ if } x = 1; x, \text{ otherwise}$
$R_\downarrow(k) = \phi(k \text{ if } k \text{ odd integer, } k - 1 \text{ if } k \text{ even})$
$R_\uparrow(k) = \phi(k \text{ if } k \text{ odd integer, } k + 1 \text{ if } k \text{ even})$
$Shift\text{-}to\text{-}Zero(x) = \begin{cases} -1_{j+1} & \text{if } x = -1_j \text{ for some index } j < d \\ 1_{j+1} & \text{if } x = 1_j \text{ for some index } j < d \\ x & \text{otherwise.} \end{cases}$
$Sign\text{-}to\text{-}Zero(x) = \begin{cases} +0 & \text{if } sgn(x) > 0 \\ -0 & \text{oherwise.} \end{cases}$
procedure update$\langle x, y \rangle$
    if $(weight(x) > 0$ and $weight(y) > 1)$ or $(weight(y) > 0$ and $weight(x) > 1)$ then
        $x' \leftarrow R_\downarrow\left(\frac{value(x)+value(y)}{2}\right)$ and $y' \leftarrow R_\uparrow\left(\frac{value(x)+value(y)}{2}\right)$
    else if $weight(x) \cdot weight(y) = 0$ and $value(x) + value(y) > 0$ then
        if $weight(x) \neq 0$ then $x' \leftarrow Shift\text{-}to\text{-}Zero(x)$ and $y' \leftarrow Sign\text{-}to\text{-}Zero(x)$
        else $y' \leftarrow Shift\text{-}to\text{-}Zero(y)$ and $x' \leftarrow Sign\text{-}to\text{-}Zero(y)$
    else if $(x \in \{-1_d, +1_d\}$ and $weight(y) = 1$ and $sgn(x) \neq sgn(y))$ or
            $(y \in \{-1_d, +1_d\}$ and $weight(x) = 1$ and $sgn(y) \neq sgn(x))$ then
        $x' \leftarrow -0$ and $y' \leftarrow +0$
    else
        $x' \leftarrow Shift\text{-}to\text{-}Zero(x)$ and $y' \leftarrow Shift\text{-}to\text{-}Zero(y)$
```

How to verify correctness automatically?

Verifying correctness

Testing whether a protocol computes φ amounts to testing:

$$\begin{aligned} \neg\exists C, D : \quad & C \xrightarrow{*} D \wedge \\ & C \text{ is initial} \wedge \\ & D \text{ is in a BSCC} \wedge \\ & \text{opinion}(D) \neq \varphi(C) \end{aligned}$$

Verifying correctness

Testing whether a protocol computes φ amounts to testing:

$$\begin{aligned} \neg\exists C, D : \quad & C \xrightarrow{*} D \wedge \\ & C \text{ is initial} \wedge \\ & D \text{ is in a BSCC} \wedge \\ & \text{opinion}(D) \neq \varphi(C) \end{aligned}$$

Theorem **Esparza *et al.* CONCUR'15**

Verification is decidable

Verifying correctness

Testing whether a protocol computes φ amounts to testing:

$$\begin{aligned} \neg\exists C, D:\ & C \xrightarrow{*} D \land \\ & C \text{ is initial} \land \\ & D \text{ is in a BSCC} \land \\ & \text{opinion}(D) \neq \varphi(C) \end{aligned}$$

As difficult as verification

TOWER-hard (Czerwinski et al. STOC'19,
Esparza et al. CONCUR'15)

Verifying correctness

Testing whether a protocol computes φ amounts to testing:

$$\begin{aligned} \neg\exists C, D:\quad & C \overset{*}{\dashrightarrow} D \wedge \\ & C \text{ is initial} \wedge \\ & D \text{ is in a BSCC} \wedge \\ & \text{opinion}(D) \neq \varphi(C) \end{aligned}$$

Relaxed with Presburger-definable overapproximation!

Verifying correctness

Testing whether a protocol computes φ amounts to testing:

$$
\begin{aligned}
\neg\exists C, D:\quad & C \overset{*}{\dashrightarrow} D \land \\
& C \text{ is initial} \land \\
& D \text{ is in a BSCC} \land \\
& \text{opinion}(D) \neq \varphi(C)
\end{aligned}
$$

Difficult to express

Verifying correctness

Testing whether a protocol computes φ amounts to testing:

$$\begin{aligned} \neg\exists C, D:\quad & C \overset{*}{\dashrightarrow} D \land \\ & C \text{ is initial} \land \\ & D \text{ is terminal} \land \\ & \text{opinion}(D) \neq \varphi(C) \end{aligned}$$

BSCCs are of size 1 for most protocols!

Verifying correctness

Testing whether a protocol computes φ amounts to testing:

$$\begin{aligned}\neg\exists C, D:\quad & C \overset{*}{\dashrightarrow} D \wedge \\ & C \text{ is initial} \wedge \\ & D \text{ is terminal} \wedge \\ & \text{opinion}(D) \neq \varphi(C)\end{aligned}$$

Testable with an SMT solver

Verifying correctness

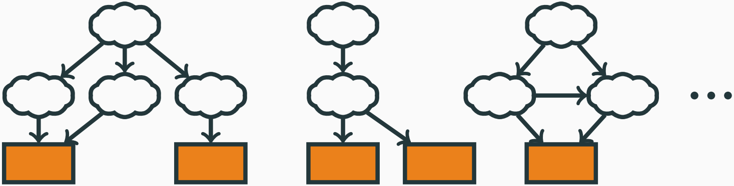
Testing whether a protocol computes φ amounts to testing:

$$\begin{aligned} \neg\exists C, D\colon\quad & C \overset{*}{\dashrightarrow} D \land \\ & C \text{ is initial} \land \\ & D \text{ is terminal } \land \\ & \text{opinion}(D) \neq \varphi(C) \end{aligned}$$

But how to know whether all BSCCs are of size 1?

Silent protocols

Protocol is *silent* if fair executions reach terminal configurations

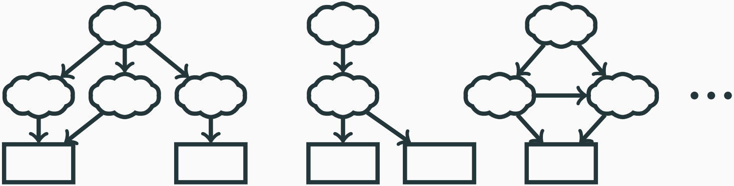


BSCCs of size 1

Silent protocols

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

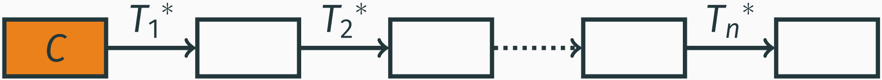


BSCCs of size 1

Silent protocols: layered termination

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

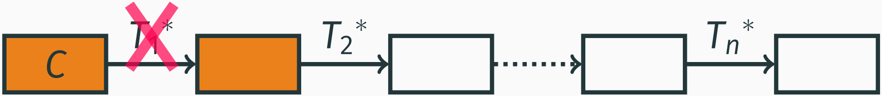
- all executions restricted to T_i terminate
- if $T_1 \cup \cdots \cup T_{i-1}$ disabled in C and $C \xrightarrow{T_i{}^*} D$, then $T_1 \cup \cdots \cup T_{i-1}$ also disabled in D



Silent protocols: layered termination

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



Silent protocols: layered termination

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

Silent protocols: layered termination

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

Silent protocols: layered termination

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

Silent protocols: layered termination

T_1

$$\begin{aligned} B\,R &\to b\,b \\ B\,r &\to B\,b \\ R\,b &\to R\,r \\ b\,r &\to b\,b \end{aligned}$$

Silent protocols: layered termination

T_1

$$\begin{aligned} B\ R &\rightarrow b\ b \\ B\ r &\rightarrow B\ b \\ R\ b &\rightarrow R\ r \\ b\ r &\rightarrow b\ b \end{aligned}$$

Bad partition: not all executions over T_1 terminate

Silent protocols: layered termination

T_1

$$\begin{array}{l} \mathbf{B}\ \mathbf{R} \to \mathbf{b}\ \mathbf{b} \\ \mathbf{B}\ \mathbf{r} \to \mathbf{B}\ \mathbf{b} \\ \mathbf{R}\ \mathbf{b} \to \mathbf{R}\ \mathbf{r} \\ \mathbf{b}\ \mathbf{r} \to \mathbf{b}\ \mathbf{b} \end{array}$$

Bad partition: not all executions over T_1 terminate

$$\{\mathbf{B}, \mathbf{B}, \mathbf{R}, \mathbf{R}\} \to \{\mathbf{B}, \mathbf{b}, \mathbf{b}, \mathbf{R}\} \to \{\mathbf{B}, \mathbf{b}, \mathbf{r}, \mathbf{R}\} \to \{\mathbf{B}, \mathbf{b}, \mathbf{b}, \mathbf{R}\} \to \{\mathbf{B}, \mathbf{b}, \mathbf{r}, \mathbf{R}\} \to \cdots$$

Silent protocols: layered termination

T_1

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

T_2

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

T_3

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

$$b\ r \rightarrow b\ b$$

Silent protocols: layered termination

T_1

$$B\ R \to b\ b$$

T_2

$$R\ b \to R\ r$$

T_3

$$B\ r \to B\ b$$

$$b\ r \to b\ b$$

$\#B \geq \#R$:

$$\{B^*,\ R^*\}$$

Silent protocols: layered termination

T_1 ✗

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

T_2

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

T_3

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

$\#B \geq \#R$:

$$\{B^*,\ R^*\} \xrightarrow{*} \{B^*,\ b^*\}$$

Silent protocols: layered termination

T_1 ✗

$$B\ R \to b\ b$$

T_2 ✗

$$R\ b \to R\ r$$

T_3

$$B\ r \to B\ b$$

$$b\ r \to b\ b$$

$\#B \geq \#R$:

$$\{B^*,\ R^*\} \xrightarrow{*} \{B^*,\ b^*\}$$

Silent protocols: layered termination

T_1 ✗

$$B\,R \rightarrow b\,b$$

T_2 ✗

$$R\,b \rightarrow R\,r$$

T_3 ✗

$$B\,r \rightarrow B\,b$$

$$b\,r \rightarrow b\,b$$

$\#B \geq \#R$:

$$\{B^*,\ R^*\} \xrightarrow{*} \{B^*,\ b^*\} \xrightarrow{*} \{B^*,\ b^*\}$$

Silent protocols: layered termination

T_1

$$\boldsymbol{B}\ \boldsymbol{R} \rightarrow \boldsymbol{b}\ \boldsymbol{b}$$

T_2

$$\boldsymbol{R}\ \boldsymbol{b} \rightarrow \boldsymbol{R}\ \boldsymbol{r}$$

T_3

$$\boldsymbol{B}\ \boldsymbol{r} \rightarrow \boldsymbol{B}\ \boldsymbol{b}$$

$$\boldsymbol{b}\ \boldsymbol{r} \rightarrow \boldsymbol{b}\ \boldsymbol{b}$$

$\# \boldsymbol{B} \geq \# \boldsymbol{R}$:

$$\{\boldsymbol{B}^*,\ \boldsymbol{R}^*\} \xrightarrow{*} \{\boldsymbol{B}^*,\ \boldsymbol{b}^*\} \xrightarrow{*} \{\boldsymbol{B}^*,\ \boldsymbol{b}^*\}$$

$\# \boldsymbol{R} > \# \boldsymbol{B}$:

$$\{\boldsymbol{R}^+,\ \boldsymbol{B}^*\}$$

Silent protocols: layered termination

T_1 ✗

$$B\ R \to b\ b$$

T_2

$$R\ b \to R\ r$$

T_3

$$B\ r \to B\ b$$

$$b\ r \to b\ b$$

$\#B \geq \#R$:

$$\{B^*,\ R^*\} \xrightarrow{*} \{B^*,\ b^*\} \xrightarrow{*} \{B^*,\ b^*\}$$

$\#R > \#B$:

$$\{R^+,\ B^*\} \xrightarrow{*} \{R^+,\ b^*\}$$

Silent protocols: layered termination

T_1 ✗

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

T_2 ✗

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

T_3

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

$\#B \geq \#R$:

$$\{B^*,\ R^*\} \xrightarrow{*} \{B^*,\ b^*\} \xrightarrow{*} \{B^*,\ b^*\}$$

$\#R > \#B$:

$$\{R^+,\ B^*\} \xrightarrow{*} \{R^+,\ b^*\} \xrightarrow{*} \{R^+,\ r^*\}$$

Silent protocols: layered termination

T_1 ✗

$B\;R \to b\;b$

T_2 ✗

$R\;b \to R\;r$

T_3 ✗

$B\;r \to B\;b$

$b\;r \to b\;b$

$\#B \geq \#R$:

$$\{B^*,\ R^*\} \xrightarrow{*} \{B^*,\ b^*\} \xrightarrow{*} \{B^*,\ b^*\}$$

$\#R > \#B$:

$$\{R^+,\ B^*\} \xrightarrow{*} \{R^+,\ b^*\} \xrightarrow{*} \{R^+,\ r^*\}$$

Silent protocols: layered termination

Theorem

Deciding whether a protocol is strongly silent $\in$ NP

Proof sketch

Guess partition $T = T_1 \cup T_2 \cup \cdots \cup T_n$ and test whether it is correct by verifying

- Petri net structural termination
- Additional simple structural properties

Peregrine: a tool for population protocols

CAV'18, PODC'17

`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!

`Peregrine`: a tool for population protocols

Protocol	Predicate	# states	# trans.	Time (secs.)
Majority [*a*]	$x \geq y$	4	4	0.1
Broadcast [*b*]	$x_1 \vee \cdots \vee x_n$	2	1	0.1
Lin. ineq. [*c*]	$\sum a_i x_i \geq 9$	75	2148	2376
Modulo [*c*]	$\sum a_i x_i = 0 \bmod 70$	72	2555	3177
Threshold [*d*]	$x \geq 50$	51	1275	182
Threshold [*b*]	$x \geq 325$	326	649	3471
Threshold [*e*]	$x \geq 10^7$	37	155	19

[*a*] Draief *et al.* 2012

[*b*] Clément *et al.* 2011

[*c*] Angluin *et al.* 2006

[*d*] Chatzigiannakis *et al.* 2010

[*e*] Offtermatt 2017

Peregrine: a tool for population protocols CAV'18, PODC'17

For example, if population size = 1000:

PRISM takes 1 hour to verify a single configuration

Protocol	Predicate	# states	# trans.	Time (secs.)
Majority [*a*]	$x \geq y$	4	4	0.1
Broadcast [*b*]	$x_1 \vee \cdots \vee x_n$	2	1	0.1
Lin. ineq. [*c*]	$\sum a_i x_i \geq 9$	75	2148	2376
Modulo [*c*]	$\sum a_i x_i = 0 \bmod 70$	72	2555	3177
Threshold [*d*]	$x \geq 50$	51	1275	182
Threshold [*b*]	$x \geq 325$	326	649	3471
Threshold [*e*]	$x \geq 10^7$	37	155	19

[*a*] Draief *et al.* 2012
[*b*] Clément *et al.* 2011
[*c*] Angluin *et al.* 2006
[*d*] Chatzigiannakis *et al.* 2010
[*e*] Offtermatt 2017

Demonstration

Expected termination time

$$\begin{aligned} \mathbf{B}, \mathbf{R} &\mapsto \mathbf{b}, \mathbf{b} \\ \mathbf{B}, \mathbf{r} &\mapsto \mathbf{B}, \mathbf{b} \\ \mathbf{R}, \mathbf{b} &\mapsto \mathbf{R}, \mathbf{r} \\ \mathbf{b}, \mathbf{r} &\mapsto \mathbf{b}, \mathbf{b} \end{aligned}$$

Correctly computes predicate $\#\mathbf{B} \geq \#\mathbf{R}$

...but how fast?

Expected termination time

$$\mathbf{B}, \mathbf{R} \mapsto \mathbf{b}, \mathbf{b}$$

$$\mathbf{B}, \mathbf{r} \mapsto \mathbf{B}, \mathbf{b}$$

$$\mathbf{R}, \mathbf{b} \mapsto \mathbf{R}, \mathbf{r}$$

$$\mathbf{b}, \mathbf{r} \mapsto \mathbf{b}, \mathbf{b}$$

Correctly computes predicate #**B** ≥ #**R**

...but how fast?

- **Natural to look for fast protocols**
- **Bounds on expected termination time useful since generally not possible to know whether a protocol has stabilized**

Expected termination time

$\mathbf{B}, \mathbf{R} \mapsto \mathbf{b}, \mathbf{b}$

$\mathbf{B}, \mathbf{r} \mapsto \mathbf{B}, \mathbf{b}$

$\mathbf{R}, \mathbf{b} \mapsto \mathbf{R}, \mathbf{r}$

$\mathbf{b}, \mathbf{r} \mapsto \mathbf{b}, \mathbf{b}$

Correctly computes predicate $\#\mathbf{B} \geq \#\mathbf{R}$

...but how fast?

Theorem **Angluin *et al.* PODC'04**

Every Presburger-definable predicate is computable by a protocol with expected termination time $\in \mathcal{O}(n^2 \log n)$

Expected termination time

$$\mathbf{B}, \mathbf{R} \mapsto \mathbf{b}, \mathbf{b}$$

$$\mathbf{B}, \mathbf{r} \mapsto \mathbf{B}, \mathbf{b}$$

$$\mathbf{R}, \mathbf{b} \mapsto \mathbf{R}, \mathbf{r}$$

$$\mathbf{b}, \mathbf{r} \mapsto \mathbf{b}, \mathbf{b}$$

Simulations show that it is slow when **R** has slight majority:

	Steps	Initial configuration
🟧	100000	{B: 7, R: 8}
🟩	7	{B: 3, R: 12}
🟩	27	{B: 4, R: 11}
🟧	100000	{B: 7, R: 8}
🟩	3	{B: 13, R: 2}

Expected termination time

$$\begin{aligned}
\mathbf{B}, \mathbf{R} &\mapsto \mathbf{T}, \mathbf{t} & \qquad X, y &\mapsto X, x \quad \text{for } x, y \in \{\mathbf{b}, \mathbf{r}, \mathbf{t}\} \\
\mathbf{B}, \mathbf{T} &\mapsto \mathbf{B}, \mathbf{b} \\
\mathbf{R}, \mathbf{T} &\mapsto \mathbf{R}, \mathbf{r} \\
\mathbf{T}, \mathbf{T} &\mapsto \mathbf{T}, \mathbf{t}
\end{aligned}$$

$$O(\mathbf{B}) = O(\mathbf{b}) = O(\mathbf{T}) = O(\mathbf{t}) = 1$$

$$O(\mathbf{R}) = O(\mathbf{r}) = 0$$

Alternative protocol

Expected termination time

$$\begin{aligned} \mathbf{B}, \mathbf{R} &\mapsto \mathbf{T}, \mathbf{t} \qquad\qquad X, y \mapsto X, x \quad \text{for } x, y \in \{\mathbf{b}, \mathbf{r}, \mathbf{t}\} \\ \mathbf{B}, \mathbf{T} &\mapsto \mathbf{B}, \mathbf{b} \\ \mathbf{R}, \mathbf{T} &\mapsto \mathbf{R}, \mathbf{r} \\ \mathbf{T}, \mathbf{T} &\mapsto \mathbf{T}, \mathbf{t} \end{aligned}$$

Is it faster?

$$O(\mathbf{B}) = O(\mathbf{b}) = O(\mathbf{T}) = O(\mathbf{t}) = 1$$
$$O(\mathbf{R}) = O(\mathbf{r}) = 0$$

Alternative protocol

Expected termination time

$$\mathbf{B}, \mathbf{R} \mapsto \mathbf{T}, \mathbf{t} \qquad X, y \mapsto X, x \quad \text{for } x, y \in \{\mathbf{b}, \mathbf{r}, \mathbf{t}\}$$

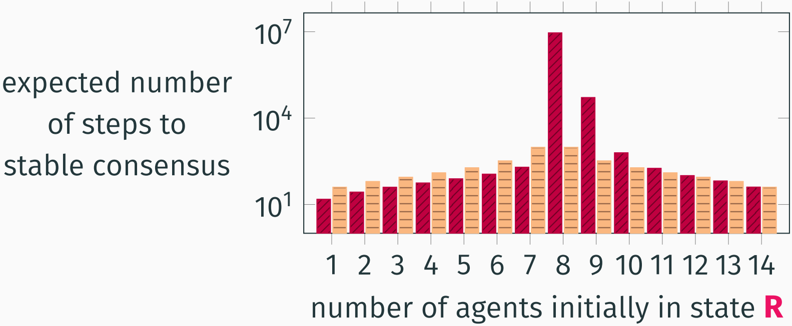
$$\mathbf{B}, \mathbf{T} \mapsto \mathbf{B}, \mathbf{b}$$

$$\mathbf{R}, \mathbf{T} \mapsto \mathbf{R}, \mathbf{r}$$

$$\mathbf{T}, \mathbf{T} \mapsto \mathbf{T}, \mathbf{t}$$

Is it faster?

Yes, for size 15...



Expected termination time

$\mathbf{B}, \mathbf{R} \quad \mapsto \quad \mathbf{T}, \mathbf{t} \qquad X, y \quad \mapsto \quad X, x \quad \text{for } x, y \in \{\mathbf{b}, \mathbf{r}, \mathbf{t}\}$

$\mathbf{B}, \mathbf{T} \quad \mapsto \quad \mathbf{B}, \mathbf{b}$

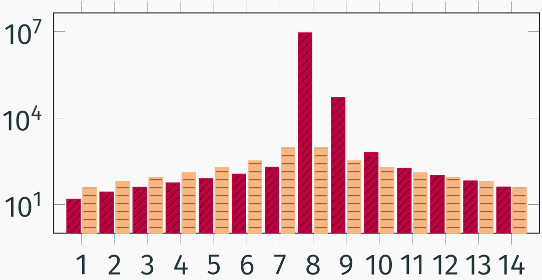
$\mathbf{R}, \mathbf{T} \quad \mapsto \quad \mathbf{R}, \mathbf{r}$

$\mathbf{T}, \mathbf{T} \quad \mapsto \quad \mathbf{T}, \mathbf{t}$

Obtained using PRISM

Clément et al. ICDCS'11, Offtermatt '17

expected number of steps to stable consensus



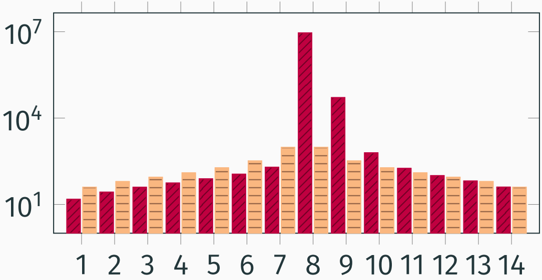
number of agents initially in state **R**

Expected termination time

B, **R**	$\mapsto$	**T**, **t**	$X, y \mapsto X, x$	for $x, y \in \{$**b**, **r**, **t**$\}$
B, **T**	$\mapsto$	**B**, **b**		
R, **T**	$\mapsto$	**R**, **r**		
T, **T**	$\mapsto$	**T**, **t**		

Our goal: analyze time for all sizes

expected number of steps to stable consensus



number of agents initially in state **R**

Expected termination time: a simple temporal logic

$$\begin{aligned}
C &\models q &&\iff && C(q) \geq 1 \\
C &\models q! &&\iff && C(q) = 1 \\
C &\models Out_b &&\iff && O(q) = b \text{ for every } C \models q \\
C &\models \neg\varphi &&\iff && C \not\models \varphi \\
C &\models \varphi \wedge \psi &&\iff && C \models \varphi \wedge C \models \psi \\
C &\models \Box\varphi &&\iff && \mathbb{P}_C(\{\sigma \in Runs(C) : \sigma_i \models \varphi \ \text{ for every } i\ \}) = 1 \\
C &\models \Diamond\varphi &&\iff && \mathbb{P}_C(\{\sigma \in Runs(C) : \sigma_i \models \varphi \ \text{ for some } i\ \}) = 1
\end{aligned}$$

Expected termination time: a simple temporal logic

$$
\begin{array}{lcl}
C \models q & \iff & C(q) \geq 1 \\
C \models q! & \iff & C(q) = 1 \\
C \models Out_b & \iff & O(q) = b \text{ for every } C \models q \\
C \models \neg\varphi & \iff & C \not\models \varphi \\
C \models \varphi \wedge \psi & \iff & C \models \varphi \wedge C \models \psi \\
C \models \Box\varphi & \iff & \mathbb{P}_C(\{\sigma \in Runs(C) : \sigma_i \models \varphi \ \text{ for every } i\ \} = 1 \\
C \models \Diamond\varphi & \iff & \mathbb{P}_C(\{\sigma \in Runs(C) : \sigma_i \models \varphi \ \text{ for some } i\ \} = 1
\end{array}
$$

Expected termination time: a simple temporal logic

$$
\begin{array}{lcl}
C \models q & \iff & C(q) \geq 1 \\
C \models q! & \iff & C(q) = 1 \\
C \models Out_b & \iff & O(q) = b \text{ for every } C \models q \\
C \models \neg\varphi & \iff & C \not\models \varphi \\
C \models \varphi \wedge \psi & \iff & C \models \varphi \wedge C \models \psi \\
C \models \Box\varphi & \iff & \mathbb{P}_C(\{\sigma \in Runs(C) : \sigma_i \models \varphi \ \text{ for every } i\ \}) = 1 \\
C \models \Diamond\varphi & \iff & \mathbb{P}_C(\{\sigma \in Runs(C) : \sigma_i \models \varphi \ \text{ for some } i\ \}) = 1
\end{array}
$$

Expected termination time: formal definition

Random variable Steps_{φ}:

assigns to each run σ the smallest k s.t. $\sigma_k \models \varphi$, otherwise ∞

Expected termination time: formal definition

Random variable Steps_{φ}:

assigns to each run σ the smallest k s.t. $\sigma_k \models \varphi$, otherwise ∞

Maximal expected termination time

We are interested in $\mathit{time}\colon \mathbb{N} \to \mathbb{N}$ where

$$\mathit{time}(n) = \max\{\mathbb{E}_C[\mathit{Steps}_{\Box \mathit{Out}_0 \vee \Box \mathit{Out}_1}] : C \text{ is initial and } |C| = n\}$$

Expected termination time: formal definition

Random variable $Steps_{\varphi}$:

assigns to each run σ the smallest k s.t. $\sigma_k \models \varphi$, otherwise ∞

Maximal expected termination time

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

$$time(n) = \max\{\mathbb{E}_C[Steps_{\Box Out_0 \vee \Box Out_1}] : C \text{ is initial and } |C| = n\}$$

Expected termination time: formal definition

Random variable Steps_{φ}:

assigns to each run σ the smallest k s.t. $\sigma_k \models \varphi$, otherwise ∞

Maximal expected termination time

We are interested in $\mathit{time} \colon \mathbb{N} \to \mathbb{N}$ where

$$\mathit{time}(n) = \max\{\mathbb{E}_C[\mathit{Steps}_{\Box \mathit{Out}_0 \vee \Box \mathit{Out}_1}] : C \text{ is initial and } |C| = n\}$$

Expected termination time: formal definition

Random variable Steps_φ:

assigns to each run σ the smallest k s.t. $\sigma_k \models \varphi$, otherwise ∞

Maximal expected termination time

We are interested in $\mathit{time} \colon \mathbb{N} \to \mathbb{N}$ where

$$\mathit{time}(n) = \max\{\mathbb{E}_C[\mathit{Steps}_{\Box \mathit{Out}_0 \vee \Box \mathit{Out}_1}] : C \text{ is initial and } |C| = n\}$$

Stage graphs

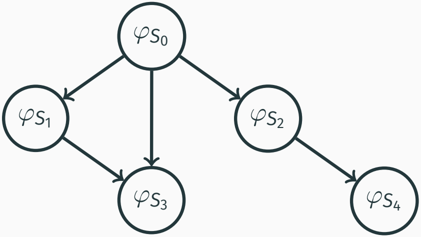
Our approach:

- Most protocols are naturally designed in stages
- Construct these stages automatically
- Derive bounds on expected termination time from stages structure

Stage graphs

A *stage graph* is a directed acyclic graph $(\mathbb{S}, \rightarrow)$ such that

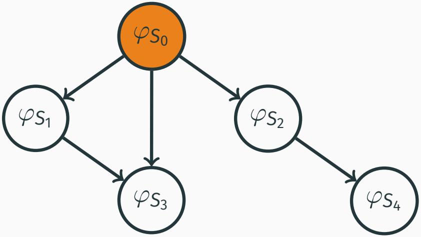
- every node $S \in \mathbb{S}$ is associated to a formula φ_S



Stage graphs

A *stage graph* is a directed acyclic graph $(\mathbb{S}, \to)$ such that

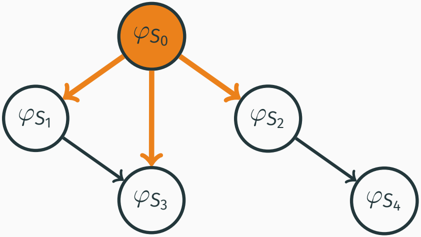
- every node $S \in \mathbb{S}$ is associated to a formula φ_S
- for every $C \in \mathrm{Init}$, there exists $S \in \mathbb{S}$ such that $C \models \varphi_S$



Stage graphs

A *stage graph* is a directed acyclic graph $(\mathbb{S}, \to)$ such that

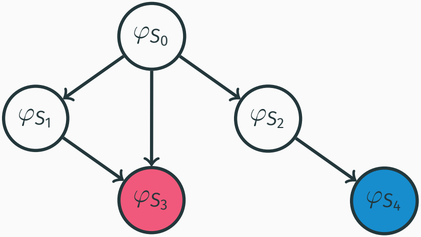
- every node $S \in \mathbb{S}$ is associated to a formula φ_S
- for every $C \in \mathrm{Init}$, there exists $S \in \mathbb{S}$ such that $C \models \varphi_S$
- $C \models \Diamond \bigvee_{S \to S'} \varphi_{S'}$ for every $S \in \mathbb{S}$ and $C \models \varphi_S$



Stage graphs

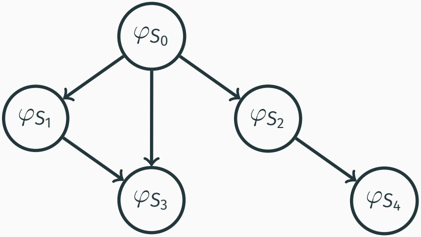
A *stage graph* is a directed acyclic graph $(\mathbb{S}, \rightarrow)$ such that

- every node $S \in \mathbb{S}$ is associated to a formula φ_S
- for every $C \in \mathrm{Init}$, there exists $S \in \mathbb{S}$ such that $C \models \varphi_S$
- $C \models \Diamond \bigvee_{S \rightarrow S'} \varphi_{S'}$ for every $S \in \mathbb{S}$ and $C \models \varphi_S$
- $C \models \varphi_S$ implies $C \models \Box Out_0 \vee \Box Out_1$ for every bottom $S \in \mathbb{S}$



Stage graphs

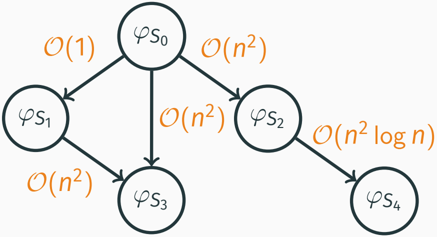
time(n) is bounded by the maximal expected number of steps to move from a stage to a successor



Stage graphs

time(n) is bounded by the maximal expected number of steps to move from a stage to a successor

For example, $time(n) \in \mathcal{O}(n^2 \log n)$ if:



A procedure for computing stage graphs

$\mathbf{B}, \mathbf{R} \quad \mapsto \quad \mathbf{T}, \mathbf{t}$

$\mathbf{B}, \mathbf{T} \quad \mapsto \quad \mathbf{B}, \mathbf{b}$

$\mathbf{R}, \mathbf{T} \quad \mapsto \quad \mathbf{R}, \mathbf{r}$

$\mathbf{T}, \mathbf{T} \quad \mapsto \quad \mathbf{T}, \mathbf{t}$

$X, y \quad \mapsto \quad X, x$

$$S_0 : (\mathbf{B} \vee \mathbf{R}) \wedge \bigwedge_{q \notin \{\mathbf{B}, \mathbf{R}\}} \neg q$$

A procedure for computing stage graphs

$\mathbf{B}, \mathbf{R} \quad \mapsto \quad \mathbf{T}, \mathbf{t}$

$\mathbf{B}, \mathbf{T} \quad \mapsto \quad \mathbf{B}, \mathbf{b}$

$\mathbf{R}, \mathbf{T} \quad \mapsto \quad \mathbf{R}, \mathbf{r}$

$\mathbf{T}, \mathbf{T} \quad \mapsto \quad \mathbf{T}, \mathbf{t}$

$X, y \quad \mapsto \quad X, x$

$$\begin{tikzcd}
& S_0 : (\mathbf{B} \vee \mathbf{R}) \wedge \bigwedge_{q \notin \{\mathbf{B}, \mathbf{R}\}} \neg q \arrow[dl, bend right, "\mathcal{O}(1)"'] \arrow[d, "\mathcal{O}(1)"'] \\
S_1 : \Box \left(\mathbf{B} \wedge \bigwedge_{q \neq \mathbf{B}} \neg q \right) & S_2 : \Box \left(\mathbf{R} \wedge \bigwedge_{q \neq \mathbf{R}} \neg q \right)
\end{tikzcd}$$

A procedure for computing stage graphs

$\mathbf{B}, \mathbf{R} \quad \mapsto \quad \mathbf{T}, \mathbf{t}$

$\mathbf{B}, \mathbf{T} \quad \mapsto \quad \mathbf{B}, \mathbf{b}$

$\mathbf{R}, \mathbf{T} \quad \mapsto \quad \mathbf{R}, \mathbf{r}$

$\mathbf{T}, \mathbf{T} \quad \mapsto \quad \mathbf{T}, \mathbf{t}$

$X, y \quad \mapsto \quad X, x$

$$\begin{tikzcd}
 & S_0 : (\mathbf{B} \lor \mathbf{R}) \land \bigwedge_{q \notin \{\mathbf{B},\mathbf{R}\}} \neg q \arrow[dl, bend right, "\mathcal{O}(1)"'] \arrow[d, "\mathcal{O}(1)"'] \\
S_1 : \Box \left(\mathbf{B} \land \bigwedge_{q \neq \mathbf{B}} \neg q \right) & S_2 : \Box \left(\mathbf{R} \land \bigwedge_{q \neq \mathbf{R}} \neg q \right)
\end{tikzcd}$$

Transformation graph

A procedure for computing stage graphs

$\boxed{\mathbf{B}, \mathbf{R} \quad \mapsto \quad \mathbf{T}, \mathbf{t}}$

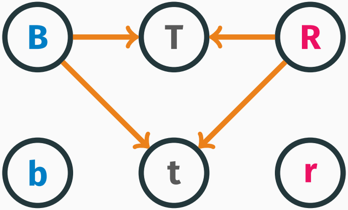
$\mathbf{B}, \mathbf{T} \quad \mapsto \quad \mathbf{B}, \mathbf{b}$

$\mathbf{R}, \mathbf{T} \quad \mapsto \quad \mathbf{R}, \mathbf{r}$

$\mathbf{T}, \mathbf{T} \quad \mapsto \quad \mathbf{T}, \mathbf{t}$

$X, y \quad \mapsto \quad X, x$

$$\begin{tikzcd}
& S_0 : (\mathbf{B} \lor \mathbf{R}) \land \bigwedge_{q \notin \{\mathbf{B},\mathbf{R}\}} \neg q \arrow[dl, bend right, "\mathcal{O}(1)"'] \arrow[d, "\mathcal{O}(1)"'] \\
S_1 : \Box \left(\mathbf{B} \land \bigwedge_{q \neq \mathbf{B}} \neg q \right) & S_2 : \Box \left(\mathbf{R} \land \bigwedge_{q \neq \mathbf{R}} \neg q \right)
\end{tikzcd}$$



A procedure for computing stage graphs

$\mathbf{B}, \mathbf{R} \mapsto \mathbf{T}, \mathbf{t}$

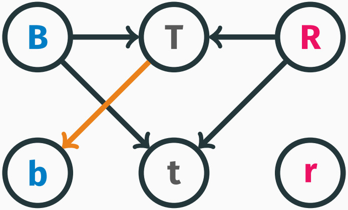
$\boxed{\mathbf{B}, \mathbf{T} \mapsto \mathbf{B}, \mathbf{b}}$

$\mathbf{R}, \mathbf{T} \mapsto \mathbf{R}, \mathbf{r}$

$\mathbf{T}, \mathbf{T} \mapsto \mathbf{T}, \mathbf{t}$

$X, y \mapsto X, x$

$$\begin{tikzcd}
 & S_0 : (\mathbf{B} \vee \mathbf{R}) \wedge \bigwedge_{q \notin \{\mathbf{B},\mathbf{R}\}} \neg q \arrow[dl, bend right, "\mathcal{O}(1)"'] \arrow[d, "\mathcal{O}(1)"'] \\
S_1 : \Box \left(\mathbf{B} \wedge \bigwedge_{q \neq \mathbf{B}} \neg q \right) & S_2 : \Box \left(\mathbf{R} \wedge \bigwedge_{q \neq \mathbf{R}} \neg q \right)
\end{tikzcd}$$



A procedure for computing stage graphs

$\mathbf{B}, \mathbf{R} \mapsto \mathbf{T}, \mathbf{t}$

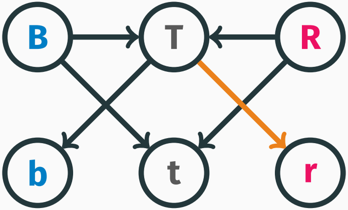
$\mathbf{B}, \mathbf{T} \mapsto \mathbf{B}, \mathbf{b}$

$\mathbf{R}, \mathbf{T} \mapsto \mathbf{R}, \mathbf{r}$

$\mathbf{T}, \mathbf{T} \mapsto \mathbf{T}, \mathbf{t}$

$X, y \mapsto X, x$

$$\begin{tikzcd}
 & S_0 : (\mathbf{B} \vee \mathbf{R}) \wedge \bigwedge_{q \notin \{\mathbf{B},\mathbf{R}\}} \neg q \arrow[dl, bend right, "\mathcal{O}(1)"'] \arrow[d, "\mathcal{O}(1)"'] \\
S_1 : \Box \left(\mathbf{B} \wedge \bigwedge_{q \neq \mathbf{B}} \neg q \right) & S_2 : \Box \left(\mathbf{R} \wedge \bigwedge_{q \neq \mathbf{R}} \neg q \right)
\end{tikzcd}$$



A procedure for computing stage graphs

$\mathbf{B}, \mathbf{R} \mapsto \mathbf{T}, \mathbf{t}$

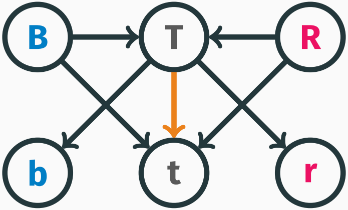
$\mathbf{B}, \mathbf{T} \mapsto \mathbf{B}, \mathbf{b}$

$\mathbf{R}, \mathbf{T} \mapsto \mathbf{R}, \mathbf{r}$

$\boxed{\mathbf{T}, \mathbf{T} \mapsto \mathbf{T}, \mathbf{t}}$

$X, y \mapsto X, x$

$$\begin{tikzcd}
 & S_0 : (\mathbf{B} \vee \mathbf{R}) \wedge \bigwedge_{q \notin \{\mathbf{B},\mathbf{R}\}} \neg q \arrow[dl, bend right, "\mathcal{O}(1)"'] \arrow[d, "\mathcal{O}(1)"'] \\
S_1 : \Box \left(\mathbf{B} \wedge \bigwedge_{q \neq \mathbf{B}} \neg q \right) & S_2 : \Box \left(\mathbf{R} \wedge \bigwedge_{q \neq \mathbf{R}} \neg q \right)
\end{tikzcd}$$



A procedure for computing stage graphs

$\mathbf{B}, \mathbf{R} \quad \mapsto \quad \mathbf{T}, \mathbf{t}$

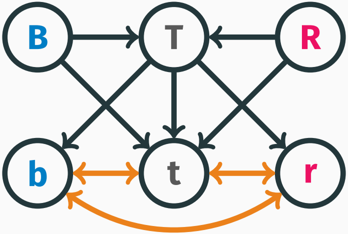
$\mathbf{B}, \mathbf{T} \quad \mapsto \quad \mathbf{B}, \mathbf{b}$

$\mathbf{R}, \mathbf{T} \quad \mapsto \quad \mathbf{R}, \mathbf{r}$

$\mathbf{T}, \mathbf{T} \quad \mapsto \quad \mathbf{T}, \mathbf{t}$

$\boxed{X, y \quad \mapsto \quad X, x}$

$$\begin{tikzcd} & S_0 : (\mathbf{B} \lor \mathbf{R}) \land \bigwedge_{q \notin \{\mathbf{B},\mathbf{R}\}} \neg q \arrow[dl, bend right, "\mathcal{O}(1)"'] \arrow[d, "\mathcal{O}(1)"'] \\ S_1 : \Box \left(\mathbf{B} \land \bigwedge_{q \neq \mathbf{B}} \neg q \right) & S_2 : \Box \left(\mathbf{R} \land \bigwedge_{q \neq \mathbf{R}} \neg q \right) \end{tikzcd}$$



A procedure for computing stage graphs

$\mathbf{B}, \mathbf{R} \mapsto \mathbf{T}, \mathbf{t}$

$\mathbf{B}, \mathbf{T} \mapsto \mathbf{B}, \mathbf{b}$

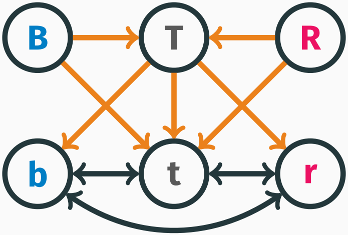
$\mathbf{R}, \mathbf{T} \mapsto \mathbf{R}, \mathbf{r}$

$\mathbf{T}, \mathbf{T} \mapsto \mathbf{T}, \mathbf{t}$

$X, y \mapsto X, x$

$$\begin{tikzcd} & S_0 : (\mathbf{B} \lor \mathbf{R}) \land \bigwedge_{q \notin \{\mathbf{B},\mathbf{R}\}} \neg q \arrow[dl, bend right, "\mathcal{O}(1)"'] \arrow[d, "\mathcal{O}(1)"'] \\ S_1 : \Box\left(\mathbf{B} \land \bigwedge_{q \neq \mathbf{B}} \neg q\right) & S_2 : \Box\left(\mathbf{R} \land \bigwedge_{q \neq \mathbf{R}} \neg q\right) \end{tikzcd}$$

Will become permanently disabled almost surely



A procedure for computing stage graphs

$\mathbf{B}, \mathbf{R} \mapsto \mathbf{T}, \mathbf{t}$

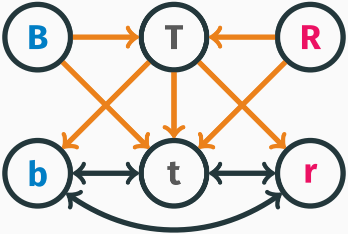
$\mathbf{B}, \mathbf{T} \mapsto \mathbf{B}, \mathbf{b}$

$\mathbf{R}, \mathbf{T} \mapsto \mathbf{R}, \mathbf{r}$

$\mathbf{T}, \mathbf{T} \mapsto \mathbf{T}, \mathbf{t}$

$X, y \mapsto X, x$

$$\begin{tikzcd}
 & S_0 : (\mathbf{B} \vee \mathbf{R}) \wedge \bigwedge_{q \notin \{\mathbf{B}, \mathbf{R}\}} \neg q \arrow[dl, bend right, "\mathcal{O}(1)"'] \arrow[d, "\mathcal{O}(1)"'] \arrow[ddr, bend left] & \\
S_1 : \Box \left(\mathbf{B} \wedge \bigwedge_{q \neq \mathbf{B}} \neg q \right) & S_2 : \Box \left(\mathbf{R} \wedge \bigwedge_{q \neq \mathbf{R}} \neg q \right) & \\
 & & S_3 : \Box \left[(\neg \mathbf{B} \vee \neg \mathbf{R}) \wedge (\neg \mathbf{B} \vee \neg \mathbf{T}) \wedge (\neg \mathbf{R} \vee \neg \mathbf{T}) \wedge (\neg \mathbf{T} \vee \mathbf{T}!) \right] \wedge ((\mathbf{B} \wedge \mathbf{b}) \vee (\mathbf{R} \wedge \mathbf{r}) \vee (\mathbf{T} \wedge \mathbf{t}))
\end{tikzcd}$$



A procedure for computing stage graphs

$\mathbf{B}, \mathbf{R} \mapsto \mathbf{T}, \mathbf{t}$

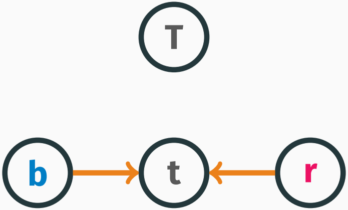
$\mathbf{B}, \mathbf{T} \mapsto \mathbf{B}, \mathbf{b}$

$\mathbf{R}, \mathbf{T} \mapsto \mathbf{R}, \mathbf{r}$

$\mathbf{T}, \mathbf{T} \mapsto \mathbf{T}, \mathbf{t}$

$X, y \mapsto X, x$

$$\begin{tikzcd}
& S_0 : (\mathbf{B} \vee \mathbf{R}) \wedge \bigwedge_{q \notin \{\mathbf{B},\mathbf{R}\}} \neg q \arrow[dl, bend right, "\mathcal{O}(1)"'] \arrow[d, "\mathcal{O}(1)"'] \arrow[ddr, bend left] & \\
S_1 : \Box \left(\mathbf{B} \wedge \bigwedge_{q \neq \mathbf{B}} \neg q \right) & S_2 : \Box \left(\mathbf{R} \wedge \bigwedge_{q \neq \mathbf{R}} \neg q \right) & \\
& & S_3 : \Box \left[(\neg \mathbf{B} \vee \neg \mathbf{R}) \wedge (\neg \mathbf{B} \vee \neg \mathbf{T}) \wedge (\neg \mathbf{R} \vee \neg \mathbf{T}) \wedge (\neg \mathbf{T} \vee \mathbf{T}!) \right] \wedge ((\mathbf{B} \wedge \mathbf{b}) \vee (\mathbf{R} \wedge \mathbf{r}) \vee (\mathbf{T} \wedge \mathbf{t}))
\end{tikzcd}$$



A procedure for computing stage graphs

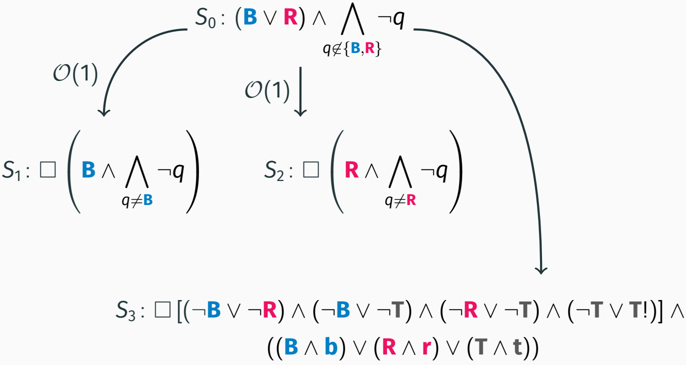
$\mathbf{B}, \mathbf{R} \mapsto \mathbf{T}, \mathbf{t}$

$\mathbf{B}, \mathbf{T} \mapsto \mathbf{B}, \mathbf{b}$

$\mathbf{R}, \mathbf{T} \mapsto \mathbf{R}, \mathbf{r}$

$\mathbf{T}, \mathbf{T} \mapsto \mathbf{T}, \mathbf{t}$

$X, y \mapsto X, x$



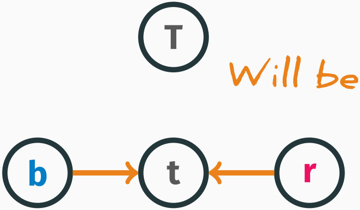
$$S_0 : (\mathbf{B} \vee \mathbf{R}) \wedge \bigwedge_{q \notin \{\mathbf{B},\mathbf{R}\}} \neg q$$

$\mathcal{O}(1)$

$\mathcal{O}(1)$

$$S_1 : \Box \left(\mathbf{B} \wedge \bigwedge_{q \neq \mathbf{B}} \neg q \right) \qquad S_2 : \Box \left(\mathbf{R} \wedge \bigwedge_{q \neq \mathbf{R}} \neg q \right)$$

$$S_3 : \Box \left[(\neg \mathbf{B} \vee \neg \mathbf{R}) \wedge (\neg \mathbf{B} \vee \neg \mathbf{T}) \wedge (\neg \mathbf{R} \vee \neg \mathbf{T}) \wedge (\neg \mathbf{T} \vee \mathbf{T}!) \right] \wedge ((\mathbf{B} \wedge \mathbf{b}) \vee (\mathbf{R} \wedge \mathbf{r}) \vee (\mathbf{T} \wedge \mathbf{t}))$$



Will become permanently disabled almost surely

A procedure for computing stage graphs

B, **R** $\mapsto$ **T**, **t**

B, **T** $\mapsto$ **B**, **b**

R, **T** $\mapsto$ **R**, **r**

T, **T** $\mapsto$ **T**, **t**

$X, y \mapsto X, x$

$$\begin{tikzcd}
& S_0 : (\mathbf{B} \vee \mathbf{R}) \wedge \bigwedge_{q \notin \{\mathbf{B},\mathbf{R}\}} \neg q \arrow[dl, bend right, "\mathcal{O}(1)"'] \arrow[d, "\mathcal{O}(1)"'] \arrow[ddr, bend left] & \\
S_1 : \Box \left(\mathbf{B} \wedge \bigwedge_{q \neq \mathbf{B}} \neg q \right) & S_2 : \Box \left(\mathbf{R} \wedge \bigwedge_{q \neq \mathbf{R}} \neg q \right) & \\
& & S_3 : \Box \left[(\neg \mathbf{B} \vee \neg \mathbf{R}) \wedge (\neg \mathbf{B} \vee \neg \mathbf{T}) \wedge (\neg \mathbf{R} \vee \neg \mathbf{T}) \wedge (\neg \mathbf{T} \vee \mathbf{T}!) \right] \wedge ((\mathbf{B} \wedge \mathbf{b}) \vee (\mathbf{R} \wedge \mathbf{r}) \vee (\mathbf{T} \wedge \mathbf{t})) \arrow[d, bend right] \\
& & S_6 : \Box \left(\mathbf{T}! \wedge \mathbf{t} \wedge \bigwedge_{q \notin \{\mathbf{T},\mathbf{t}\}} \neg q \right)
\end{tikzcd}$$

T

t

A procedure for computing stage graphs

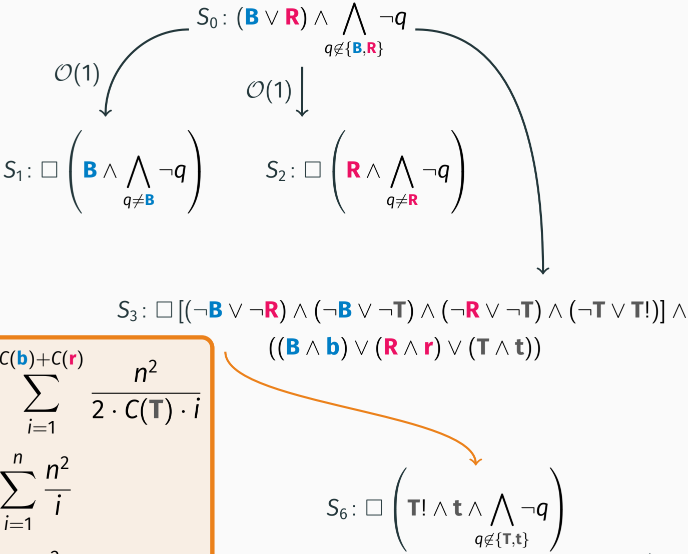
$\mathbf{B}, \mathbf{R} \mapsto \mathbf{T}, \mathbf{t}$

$\mathbf{B}, \mathbf{T} \mapsto \mathbf{B}, \mathbf{b}$

$\mathbf{R}, \mathbf{T} \mapsto \mathbf{R}, \mathbf{r}$

$\mathbf{T}, \mathbf{T} \mapsto \mathbf{T}, \mathbf{t}$

$X, y \mapsto X, x$



$$S_0 : (\mathbf{B} \vee \mathbf{R}) \wedge \bigwedge_{q \notin \{\mathbf{B},\mathbf{R}\}} \neg q$$

$\mathcal{O}(1)$

$\mathcal{O}(1)$

$$S_1 : \Box \left(\mathbf{B} \wedge \bigwedge_{q \neq \mathbf{B}} \neg q \right) \qquad S_2 : \Box \left(\mathbf{R} \wedge \bigwedge_{q \neq \mathbf{R}} \neg q \right)$$

$$S_3 : \Box \, [(\neg \mathbf{B} \vee \neg \mathbf{R}) \wedge (\neg \mathbf{B} \vee \neg \mathbf{T}) \wedge (\neg \mathbf{R} \vee \neg \mathbf{T}) \wedge (\neg \mathbf{T} \vee \mathbf{T}!)] \wedge ((\mathbf{B} \wedge \mathbf{b}) \vee (\mathbf{R} \wedge \mathbf{r}) \vee (\mathbf{T} \wedge \mathbf{t}))$$

$$S_6 : \Box \left(\mathbf{T}! \wedge \mathbf{t} \wedge \bigwedge_{q \notin \{\mathbf{T},\mathbf{t}\}} \neg q \right)$$

$$\begin{aligned} \mathbb{E}_C[Steps_{\neg \mathbf{b} \wedge \neg \mathbf{r}}] &\leq \sum_{i=1}^{C(\mathbf{b})+C(\mathbf{r})} \frac{n^2}{2 \cdot C(\mathbf{T}) \cdot i} \\ &\leq \sum_{i=1}^{n} \frac{n^2}{i} \\ &\leq \alpha \cdot n^2 \cdot \log n \end{aligned}$$

A procedure for computing stage graphs

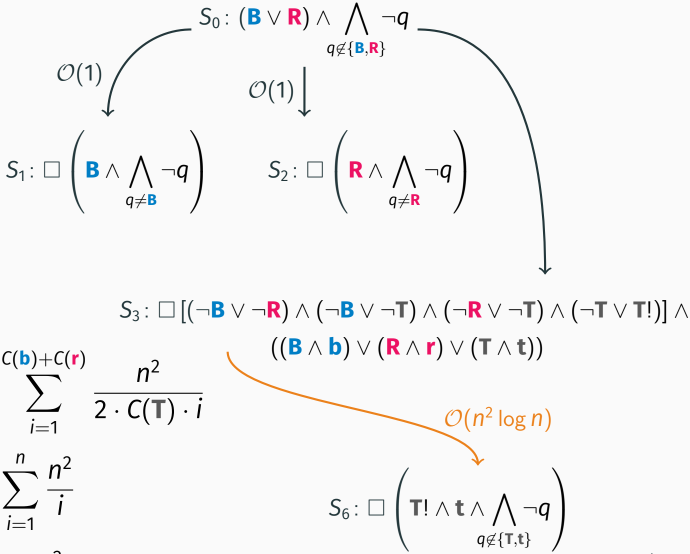
$$\mathbf{B}, \mathbf{R} \mapsto \mathbf{T}, \mathbf{t}$$

$$\mathbf{B}, \mathbf{T} \mapsto \mathbf{B}, \mathbf{b}$$

$$\mathbf{R}, \mathbf{T} \mapsto \mathbf{R}, \mathbf{r}$$

$$\mathbf{T}, \mathbf{T} \mapsto \mathbf{T}, \mathbf{t}$$

$$X, y \mapsto X, x$$



$$S_0 : (\mathbf{B} \vee \mathbf{R}) \wedge \bigwedge_{q \notin \{\mathbf{B},\mathbf{R}\}} \neg q$$

$\mathcal{O}(1)$

$\mathcal{O}(1)$

$$S_1 : \Box \left(\mathbf{B} \wedge \bigwedge_{q \neq \mathbf{B}} \neg q \right)$$

$$S_2 : \Box \left(\mathbf{R} \wedge \bigwedge_{q \neq \mathbf{R}} \neg q \right)$$

$$S_3 : \Box \left[(\neg \mathbf{B} \vee \neg \mathbf{R}) \wedge (\neg \mathbf{B} \vee \neg \mathbf{T}) \wedge (\neg \mathbf{R} \vee \neg \mathbf{T}) \wedge (\neg \mathbf{T} \vee \mathbf{T}!) \right] \wedge ((\mathbf{B} \wedge \mathbf{b}) \vee (\mathbf{R} \wedge \mathbf{r}) \vee (\mathbf{T} \wedge \mathbf{t}))$$

$\mathcal{O}(n^2 \log n)$

$$S_6 : \Box \left(\mathbf{T}! \wedge \mathbf{t} \wedge \bigwedge_{q \notin \{\mathbf{T},\mathbf{t}\}} \neg q \right)$$

$$\begin{aligned} \mathbb{E}_C[Steps_{\neg \mathbf{b} \wedge \neg \mathbf{r}}] &\leq \sum_{i=1}^{C(\mathbf{b})+C(\mathbf{r})} \frac{n^2}{2 \cdot C(\mathbf{T}) \cdot i} \\ &\leq \sum_{i=1}^{n} \frac{n^2}{i} \\ &\leq \alpha \cdot n^2 \cdot \log n \end{aligned}$$

A procedure for computing stage graphs

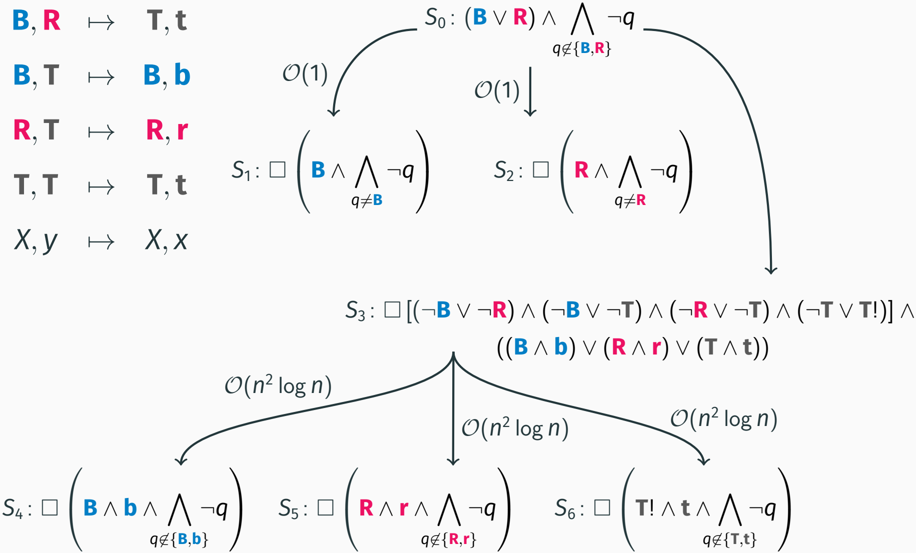
$\mathbf{B}, \mathbf{R} \mapsto \mathbf{T}, \mathbf{t}$

$\mathbf{B}, \mathbf{T} \mapsto \mathbf{B}, \mathbf{b}$

$\mathbf{R}, \mathbf{T} \mapsto \mathbf{R}, \mathbf{r}$

$\mathbf{T}, \mathbf{T} \mapsto \mathbf{T}, \mathbf{t}$

$X, y \mapsto X, x$



$S_0 : (\mathbf{B} \vee \mathbf{R}) \wedge \bigwedge_{q \notin \{\mathbf{B},\mathbf{R}\}} \neg q$

$\mathcal{O}(1)$

$\mathcal{O}(1)$

$S_1 : \Box \left(\mathbf{B} \wedge \bigwedge_{q \neq \mathbf{B}} \neg q \right)$

$S_2 : \Box \left(\mathbf{R} \wedge \bigwedge_{q \neq \mathbf{R}} \neg q \right)$

$S_3 : \Box [(\neg \mathbf{B} \vee \neg \mathbf{R}) \wedge (\neg \mathbf{B} \vee \neg \mathbf{T}) \wedge (\neg \mathbf{R} \vee \neg \mathbf{T}) \wedge (\neg \mathbf{T} \vee \mathbf{T!})] \wedge$
$((\mathbf{B} \wedge \mathbf{b}) \vee (\mathbf{R} \wedge \mathbf{r}) \vee (\mathbf{T} \wedge \mathbf{t}))$

$\mathcal{O}(n^2 \log n)$

$\mathcal{O}(n^2 \log n)$

$\mathcal{O}(n^2 \log n)$

$S_4 : \Box \left(\mathbf{B} \wedge \mathbf{b} \wedge \bigwedge_{q \notin \{\mathbf{B},\mathbf{b}\}} \neg q \right)$

$S_5 : \Box \left(\mathbf{R} \wedge \mathbf{r} \wedge \bigwedge_{q \notin \{\mathbf{R},\mathbf{r}\}} \neg q \right)$

$S_6 : \Box \left(\mathbf{T!} \wedge \mathbf{t} \wedge \bigwedge_{q \notin \{\mathbf{T},\mathbf{t}\}} \neg q \right)$

A procedure for computing stage graphs

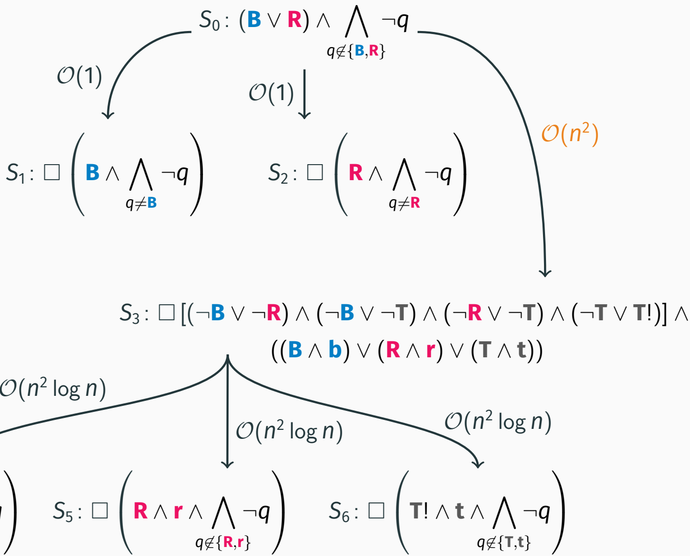
$\mathbf{B}, \mathbf{R} \mapsto \mathbf{T}, \mathbf{t}$

$\mathbf{B}, \mathbf{T} \mapsto \mathbf{B}, \mathbf{b}$

$\mathbf{R}, \mathbf{T} \mapsto \mathbf{R}, \mathbf{r}$

$\mathbf{T}, \mathbf{T} \mapsto \mathbf{T}, \mathbf{t}$

$X, y \mapsto X, x$



$S_0 : (\mathbf{B} \vee \mathbf{R}) \wedge \bigwedge_{q \notin \{\mathbf{B},\mathbf{R}\}} \neg q$

$\mathcal{O}(1)$

$\mathcal{O}(1)$

$\mathcal{O}(n^2)$

$S_1 : \Box \left(\mathbf{B} \wedge \bigwedge_{q \neq \mathbf{B}} \neg q \right)$

$S_2 : \Box \left(\mathbf{R} \wedge \bigwedge_{q \neq \mathbf{R}} \neg q \right)$

$$S_3 : \Box\, [(\neg \mathbf{B} \vee \neg \mathbf{R}) \wedge (\neg \mathbf{B} \vee \neg \mathbf{T}) \wedge (\neg \mathbf{R} \vee \neg \mathbf{T}) \wedge (\neg \mathbf{T} \vee \mathbf{T}!)] \wedge ((\mathbf{B} \wedge \mathbf{b}) \vee (\mathbf{R} \wedge \mathbf{r}) \vee (\mathbf{T} \wedge \mathbf{t}))$$

$\mathcal{O}(n^2 \log n)$

$\mathcal{O}(n^2 \log n)$

$\mathcal{O}(n^2 \log n)$

$S_4 : \Box \left(\mathbf{B} \wedge \mathbf{b} \wedge \bigwedge_{q \notin \{\mathbf{B},\mathbf{b}\}} \neg q \right)$

$S_5 : \Box \left(\mathbf{R} \wedge \mathbf{r} \wedge \bigwedge_{q \notin \{\mathbf{R},\mathbf{r}\}} \neg q \right)$

$S_6 : \Box \left(\mathbf{T}! \wedge \mathbf{t} \wedge \bigwedge_{q \notin \{\mathbf{T},\mathbf{t}\}} \neg q \right)$

A procedure for computing stage graphs

Φ: propositional formula describing current configurations

π: set of permanently present/absent states

$\mathcal{T}$: set of permanently disabled transitions

Successors computed by enriching
π through trap/siphon-like analysis and
$\mathcal{T}$ and Φ from transformation graph

A procedure for computing stage graphs

Φ: propositional formula describing current configurations
π: set of permanently present/absent states
$\mathcal{T}$: set of permanently disabled transitions

Successors computed by enriching
π through trap/siphon-like analysis and
$\mathcal{T}$ and Φ from transformation graph

Experimental results

- Prototype implemented in python™ + Microsoft Z3
- Can report: $\mathcal{O}(1), \mathcal{O}(n^2), \mathcal{O}(n^2 \log n), \mathcal{O}(n^3), \mathcal{O}(\text{poly}(n))$ or $\mathcal{O}(\exp(n))$
- Tested on various protocols from the literature

Experimental results

Protocol			**Stages**	**Bound**	**Time**
φ / params.	$\lvert Q\rvert$	$\lvert T\rvert$			
$x_1 \vee \ldots \vee x_n$ [b]	2	1	5	$n^2 \log n$	0.1
$x \geq y$ [a]	6	10	23	$n^2 \log n$	0.9
$x \geq y$ [c]	4	3	9	$n^2 \log n$	0.2
$x \geq y$ [c]	4	4	11	$\exp(n)$	0.3
Threshold [a]: $x \geq c$					
$c = 5$	6	21	26	n^3	0.8
$c = 15$	16	136	66	n^3	12.1
$c = 25$	26	351	106	n^3	58.0
$c = 35$	36	666	146	n^3	222.3
$c = 45$	46	1081	186	n^3	495.3
$c = 55$	56	1596	—	—	T/O
Logarithmic threshold: $x \geq c$					
$c = 7$	6	14	34	n^3	1.9
$c = 31$	10	34	130	n^3	6.1
$c = 127$	14	62	514	n^3	39.4
$c = 1023$	20	119	4098	n^3	395.7
$c = 4095$	24	167	—	—	T/O

[a] Angluin *et al.* 2006

[b] Clément *et al.* 2011

[c] Draief *et al.* 2012

[d] Alistarh *et al.* 2015

Protocol			**Stages**	**Bound**	**Time**
φ / params.	$\lvert Q\rvert$	$\lvert T\rvert$			
Threshold [b]: $x \geq c$					
$c = 5$	6	9	54	n^3	2.5
$c = 7$	8	13	198	n^3	11.3
$c = 10$	11	19	1542	n^3	83.9
$c = 13$	14	25	12294	n^3	816.4
$c = 15$	16	29	—	—	T/O
Average-and-conquer [d]: $x \geq y$ (param. m, d)					
$m = 3, d = 1$	6	21	41	$n^2 \log n$	2.0
$m = 3, d = 2$	8	36	1948	$n^2 \log n$	98.7
$m = 5, d = 1$	8	36	1870	n^3	80.1
$m = 5, d = 2$	10	55	—	—	T/O
Remainder [a]: $\sum_{1 \leq i < m} i \cdot x_i \equiv 0 \pmod{c}$					
$c = 5$	7	25	225	$n^2 \log n$	12.5
$c = 7$	9	42	1351	$n^2 \log n$	88.9
$c = 9$	11	63	7035	$n^2 \log n$	544.0
$c = 10$	12	75	—	—	T/O
Linear inequalities [a]					
$-x_1 + x_2 < 0$	12	57	21	n^3	3.0
$-x_1 + x_2 < 1$	20	155	131	n^3	30.3
$-x_1 + x_2 < 2$	28	301	—	—	T/O

Conclusion: summary

Population protocols analyzable automatically:

- Formal verification of correctness
- Bounds on expected termination time
- Tool support

Conclusion: future work

- Combining verification and expected termination time analysis?
- Asymptotic *lower* bounds on expected termination time?
- Interesting class of protocols with decidable quantitative model checking?

Thank you!

Merci!