Automatic Analysis of Expected Termination Time for Population Protocols

Michael Blondin Javier Esparza Antonín Kučera

UNIVERSITÉ DE SHERBROOKE

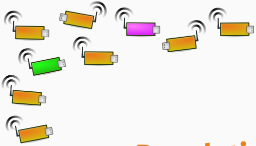
Technical University of Munich

FACULTAS ARTIS INFORMATICÆ UNIVERSITAS MASARYKIANA

Overview

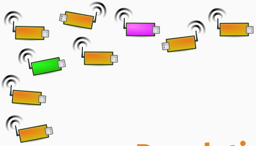
Population protocols: distributed computing model for massive networks of passively mobile finite-state agents

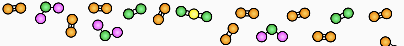
Overview

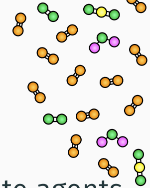


Population protocols: distributed computing model for massive networks of passively mobile finite-state agents

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







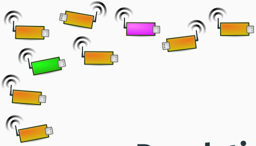
Population protocols: distributed computing model for massive networks of passively mobile finite-state agents

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

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

e.g. if φ 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: automatic derivation of upper bounds on the running time of protocols

Population protocols

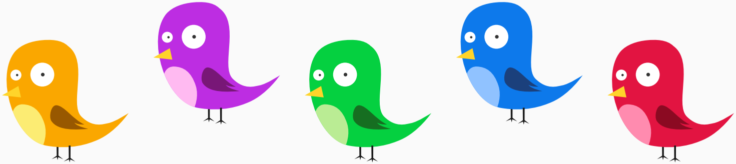
Angluin *et al.* PODC'04

- 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

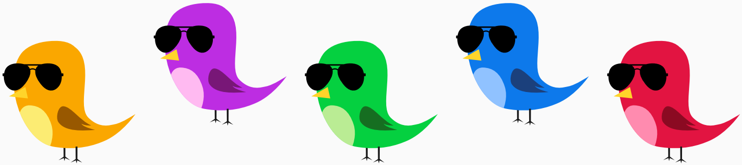
- anonymous mobile agents with very few resources
- agents change states via random pairwise interactions
- each agent has opinion true/false
- computes by stabilizing agents to some opinion



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

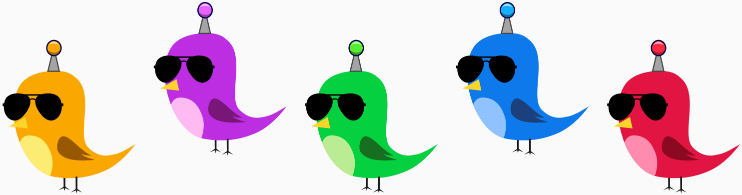


Population protocols

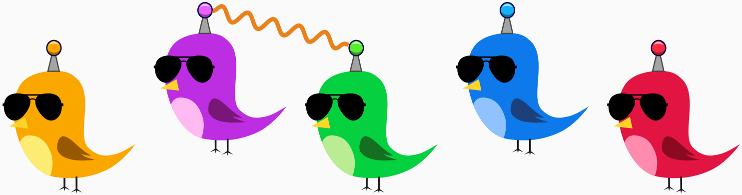
Angluin *et al.* PODC'04

- 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

- 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



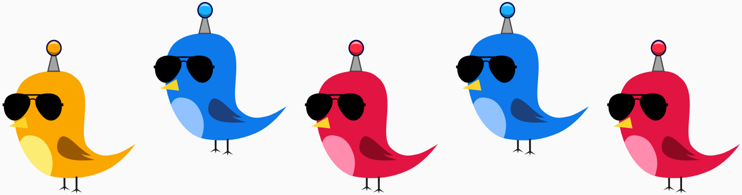
- anonymous mobile agents with very few resources
- agents change states via random pairwise interactions
- each agent has opinion true/false
- computes by stabilizing agents to some opinion



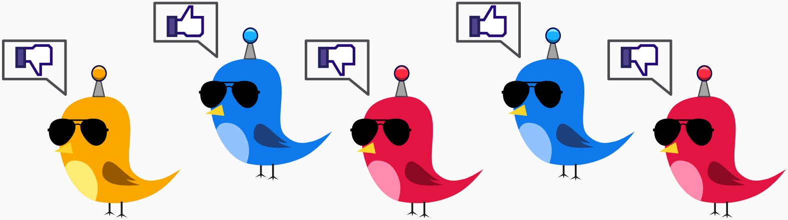
Population protocols

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



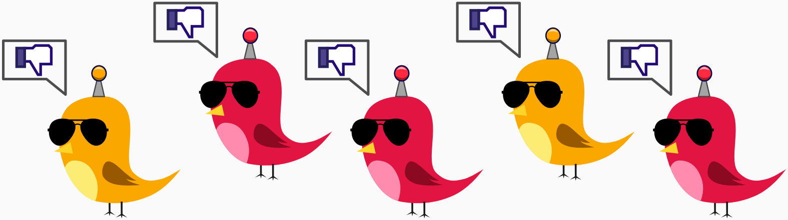
- anonymous mobile agents with very few resources
- agents change states via random pairwise interactions
- each agent has opinion true/false
- computes by stabilizing agents to some opinion



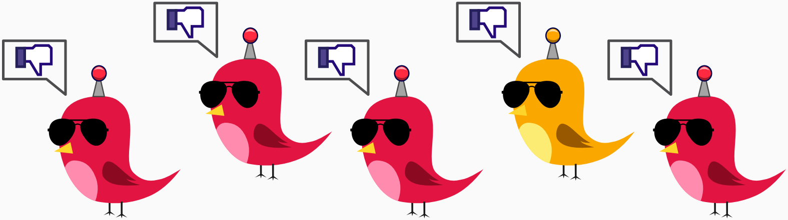
Population protocols

Angluin *et al.* PODC'04

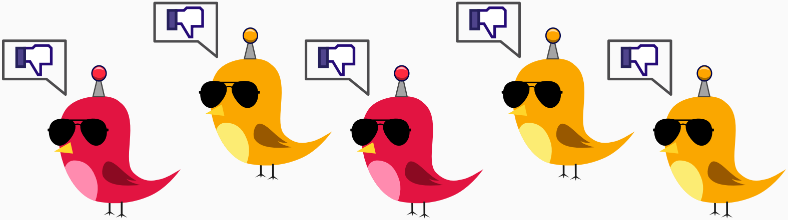
- anonymous mobile agents with very few resources
- agents change states via random pairwise interactions
- each agent has opinion true/false
- computes by stabilizing agents to some opinion



- 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

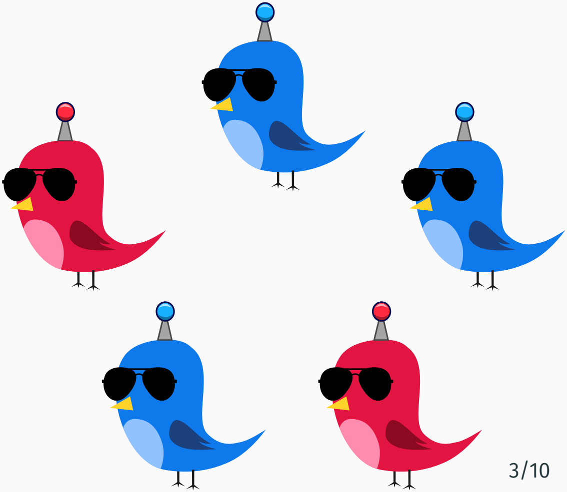


- anonymous mobile agents with very few resources
- agents change states via random pairwise interactions
- each agent has opinion true/false
- computes by stabilizing agents to some opinion



Example: majority protocol

At least as many blue birds than red birds?

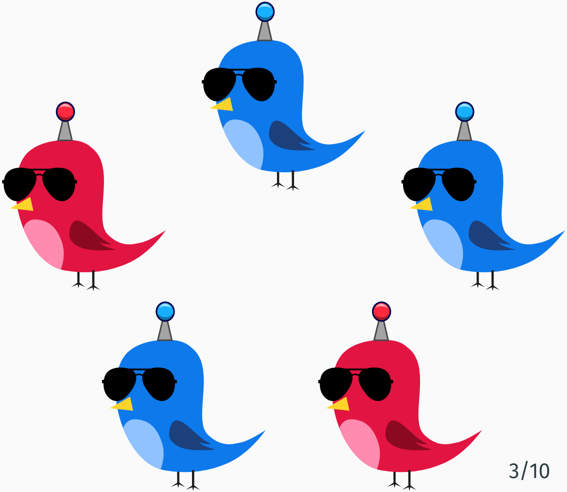


Example: majority protocol

At least as many blue birds than red birds?

Protocol:

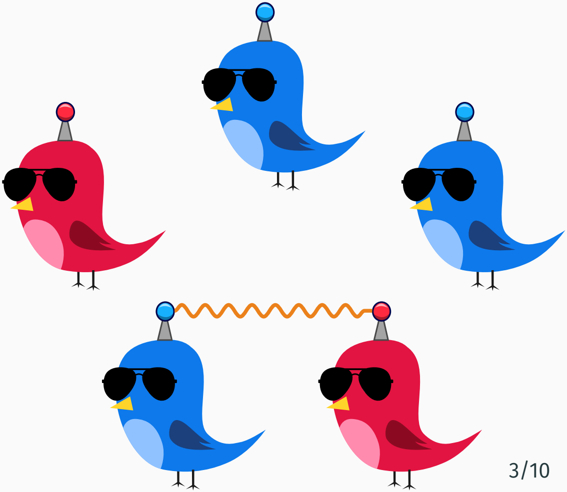
- Two large birds of different colors become small
- Large birds convert small birds to their color



At least as many blue birds than red birds?

Protocol:

- Two large birds of different colors become small
- Large birds convert small birds to their color

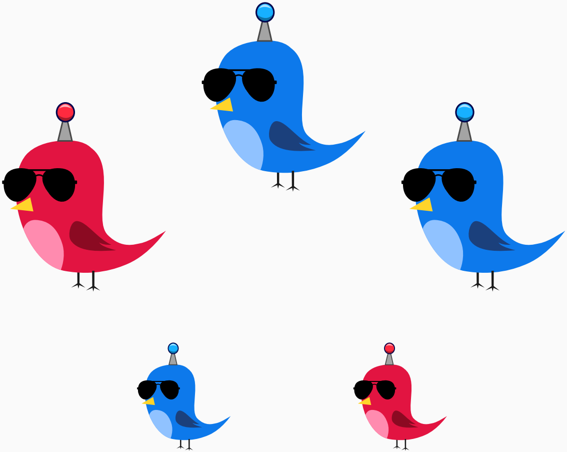


Example: majority protocol

At least as many blue birds than red birds?

Protocol:

- Two large birds of different colors become small
- Large birds convert small birds to their color

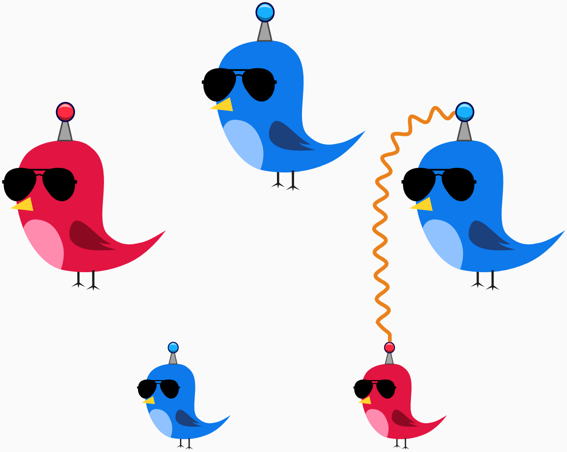


Example: majority protocol

At least as many blue birds than red birds?

Protocol:

- Two large birds of different colors become small
- Large birds convert small birds to their color

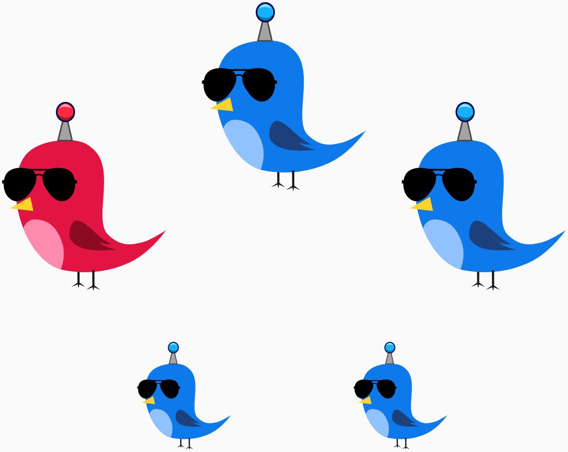


Example: majority protocol

At least as many blue birds than red birds?

Protocol:

- Two large birds of different colors become small
- Large birds convert small birds to their color

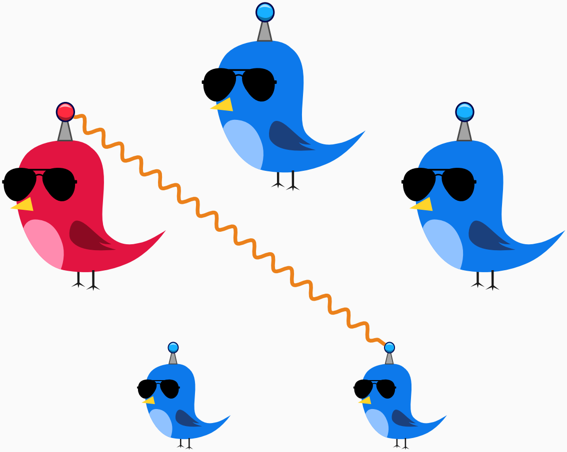


Example: majority protocol

At least as many blue birds than red birds?

Protocol:

- Two large birds of different colors become small
- Large birds convert small birds to their color



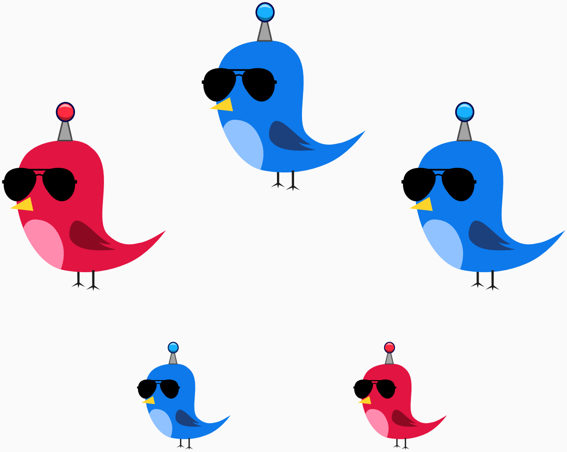
Example: majority protocol

At least as many blue birds than red birds?

Protocol:

- Two large birds of different colors become small

- Large birds convert small birds to their color

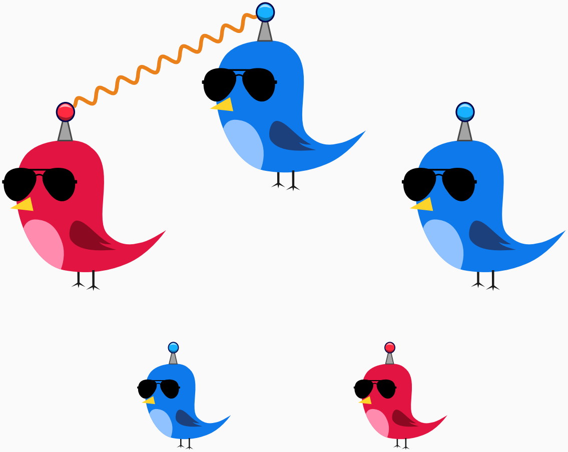


Example: majority protocol

At least as many blue birds than red birds?

Protocol:

- Two large birds of different colors become small
- Large birds convert small birds to their color

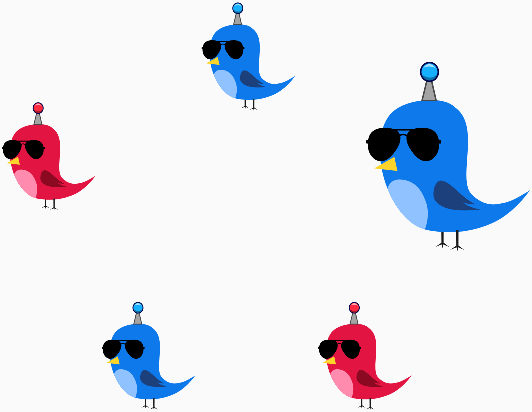


Example: majority protocol

At least as many blue birds than red birds?

Protocol:

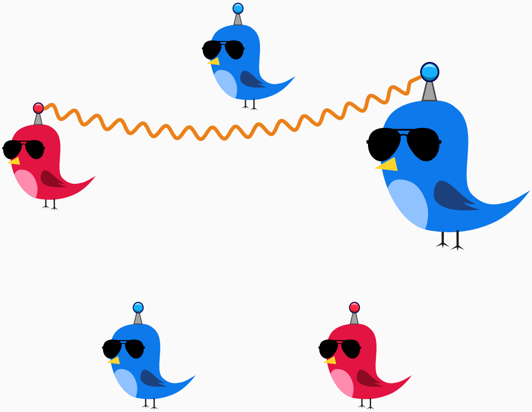
- Two large birds of different colors become small
- Large birds convert small birds to their color



At least as many blue birds than red birds?

Protocol:

- Two large birds of different colors become small
- Large birds convert small birds to their color

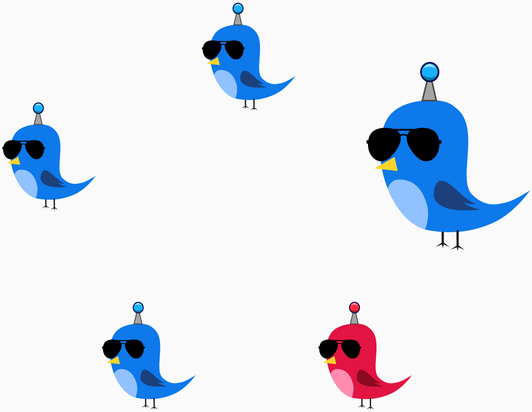


Example: majority protocol

At least as many blue birds than red birds?

Protocol:

- Two large birds of different colors become small
- Large birds convert small birds to their color

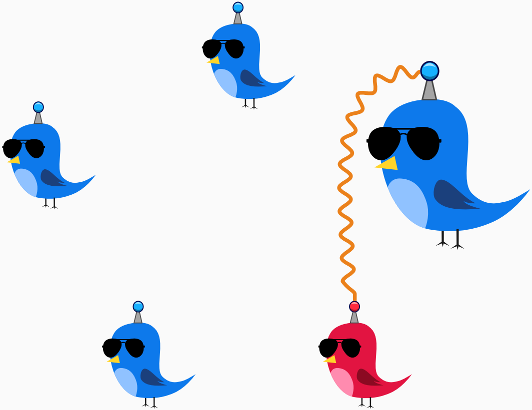


Example: majority protocol

At least as many blue birds than red birds?

Protocol:

- Two large birds of different colors become small
- Large birds convert small birds to their color

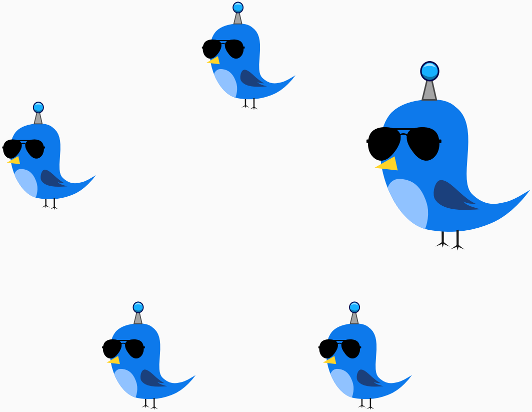


Example: majority protocol

At least as many blue birds than red birds?

Protocol:

- Two large birds of different colors become small
- Large birds convert small birds to their color

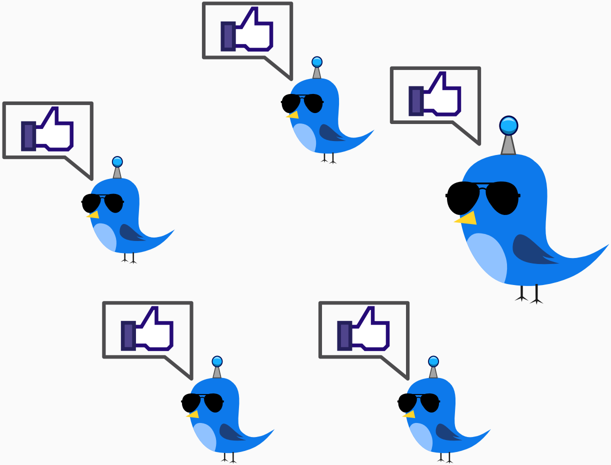


Example: majority protocol

At least as many blue birds than red birds?

Protocol:

- Two large birds of different colors become small
- Large birds convert small birds to their color

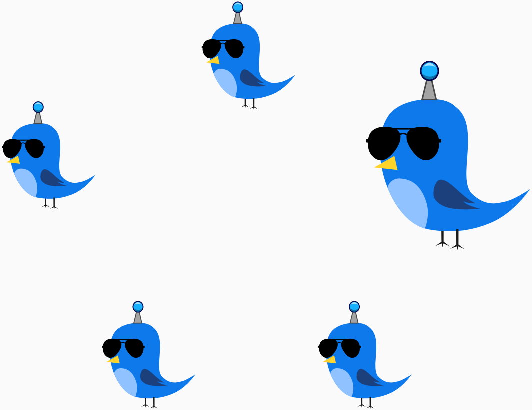


Example: majority protocol

At least as many blue birds than red birds?

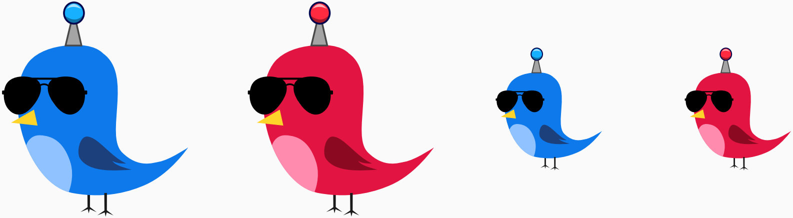
Protocol:

- Two large birds of different colors become small
- Large birds convert small birds to their color
- **To break ties:** small blue birds convert small red birds



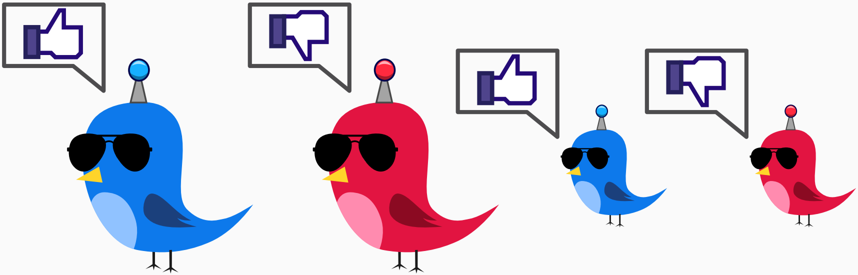
Population protocols: formal model

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



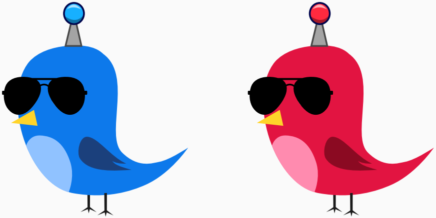
Population protocols: formal model

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



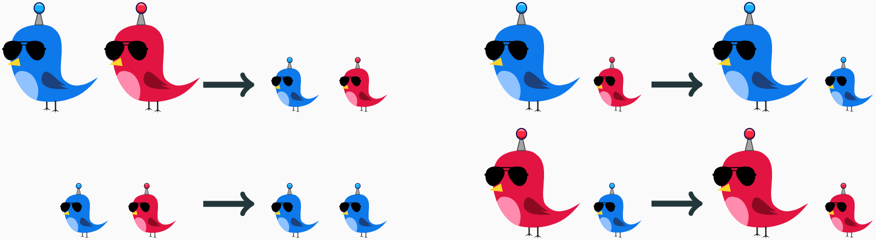
Population protocols: formal model

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

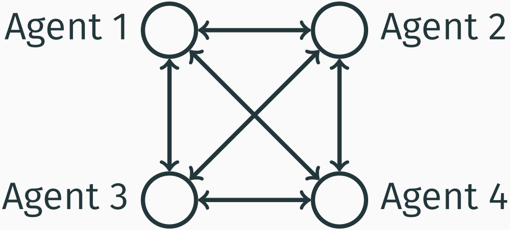


Population protocols: formal model

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

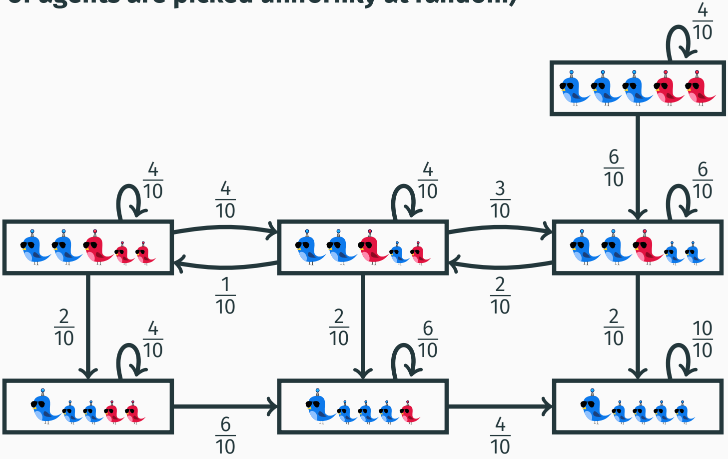


Interaction graph:

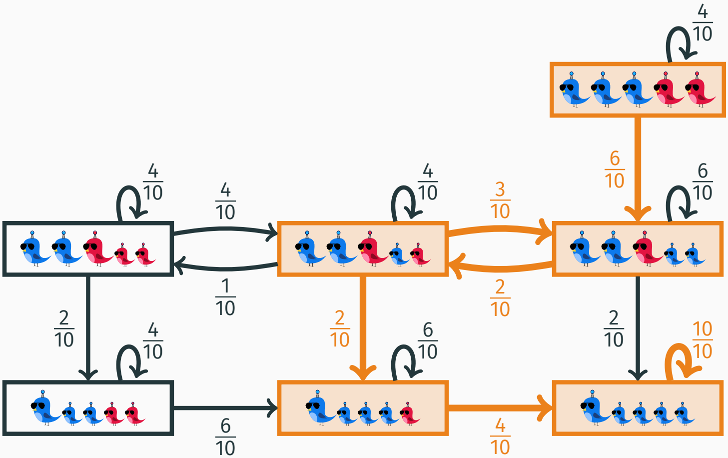


Underlying Markov chain:

(pairs of agents are picked uniformly at random)



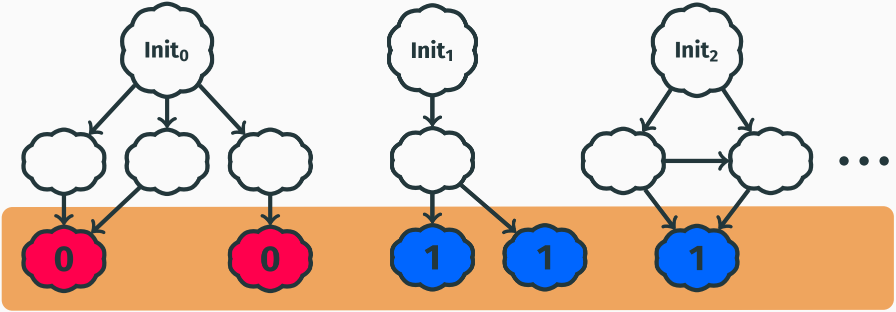
A *run* is an infinite path:



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

if runs reach **common stable consensus**

with probability 1



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

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.* $\mathrm{FO}(\mathbb{N}, +, <)$

Protocols speed

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

Computes correctly predicate $\#\mathbf{B} \geq \#\mathbf{R}$

...but how fast?

Protocols speed

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

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

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

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

Computes correctly predicate $\#\mathbf{B} \geq \#\mathbf{R}$

...but how fast?

- **Natural to want protocols to be fast**
- **Upper bounds on number of steps useful since generally not possible to know whether a protocol has stabilized**

Protocols speed

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

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}

Protocols speed

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

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

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

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

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

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

Alternative protocol

Protocols speed

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

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

Protocols speed

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

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

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

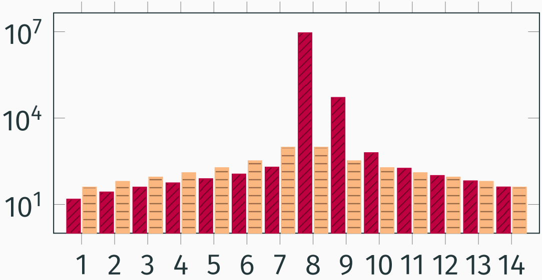
$\mathbf{T}, \mathbf{T} \mapsto \mathbf{T}, \mathbf{t}$

$X, y \mapsto X, x$ for $x, y \in \{\mathbf{b}, \mathbf{r}, \mathbf{t}\}$

Is it faster?

Yes, for size 15...

expected number of steps to stable consensus



number of agents initially in state **R**

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

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

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

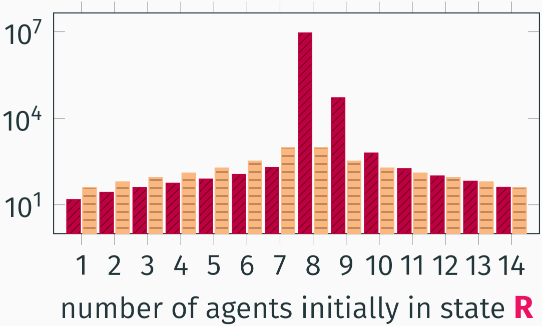
$\mathbf{T}, \mathbf{T} \mapsto \mathbf{T}, \mathbf{t}$

$X, y \mapsto X, x$ for $x, y \in \{\mathbf{b}, \mathbf{r}, \mathbf{t}\}$

Obtained using PRISM

Clément et al. ICDCS'11, Offtermatt '17

expected number of steps to stable consensus



Protocols speed

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

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

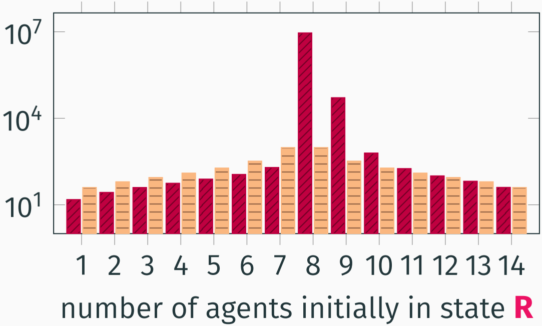
$\mathbf{R}, \mathbf{T} \mapsto \mathbf{R}, \mathbf{r}$

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

$X, y \mapsto X, x$ for $x, y \in \{\mathbf{b}, \mathbf{r}, \mathbf{t}\}$

Our goal: analyze speed for all sizes

expected number of steps to stable consensus



Protocols speed: related work

- Any Presburger-definable predicate is computable in time $\mathcal{O}(n^2 \log n)$ Angluin *et al.* (PODC'04)

- Upper/lower bounds for majority and leader election

- Study of trade-offs between speed and number of states

  e.g.

  - Alistarh, Aspnes, Eisenstat, Gelashvili and Rivest (SODA'17)
  - Belleville, Doty and Soloveichik (ICALP'17)
  - Doty and Soloveichik (DISC'15), etc.

Definitions: a simple temporal logic

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

Definitions: a simple temporal logic

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

Definitions: a simple temporal logic

$$C \models q \iff C(q) \geq 1$$

$$C \models q! \iff C(q) = 1$$

$$C \models Out_b \iff O(q) = b \text{ for every } q \models C$$

$$C \models \neg\varphi \iff C \not\models \varphi$$

$$C \models \varphi \wedge \psi \iff C \models \varphi \wedge \psi$$

$$C \models \Box\varphi \iff \mathbb{P}_C(\{\sigma \in Runs(C) : \sigma_i \models \varphi \text{ for every } i\}) = 1$$

$$C \models \Diamond\varphi \iff \mathbb{P}_C(\{\sigma \in Runs(C) : \sigma_i \models \varphi \text{ for some } i\}) = 1$$

Definitions: expected termination time

Random variable Steps_{φ}:

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

Definitions: expected termination time

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

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

Definitions: expected termination time

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

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

Definitions: expected termination time

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

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

Definitions: expected termination time

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

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

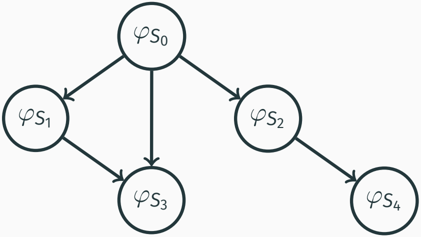
Our approach:

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

Stage graphs

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

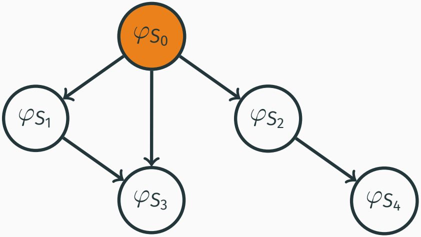
- every node $S \in \mathbb{S}$ is associated to a formula φ_S



Stage graphs

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

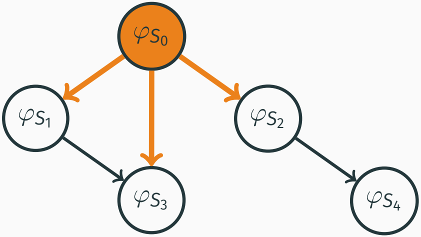
- every node $S \in \mathbb{S}$ is associated to a formula φ_S
- for every $C \in \mathrm{Init}$, there exists $S \in \mathbb{S}$ such that $C \models \varphi_S$



Stage graphs

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

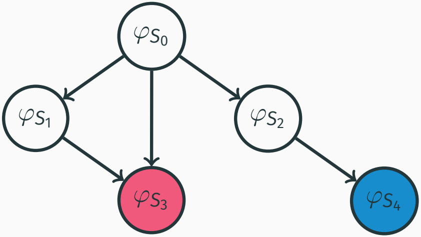
- every node $S \in \mathbb{S}$ is associated to a formula φ_S
- for every $C \in \mathrm{Init}$, there exists $S \in \mathbb{S}$ such that $C \models \varphi_S$
- $C \models \Diamond \bigvee_{S \to S'} \varphi_{S'}$ for every $S \in \mathbb{S}$ and $C \models \varphi_S$



Stage graphs

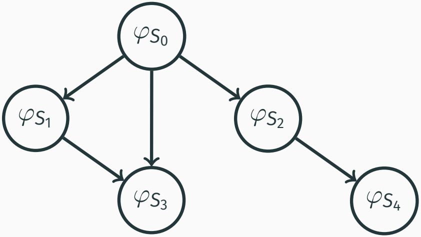
A *stage graph* is a directed acyclic graph $(\mathbb{S}, \rightarrow)$ such that

- 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

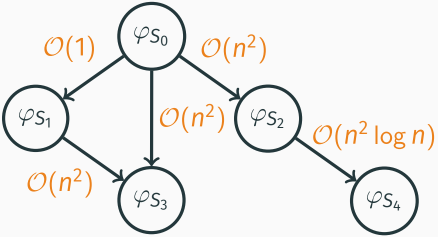
time(n) is bounded by the maximal expected number of steps to move from a stage to a successor



Stage graphs

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

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



A procedure for computing stage graphs

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

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

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

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

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

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

A procedure for computing stage graphs

$$
\begin{array}{ccc}
\mathbf{B}, \mathbf{R} & \mapsto & \mathbf{T}, \mathbf{t} \\
\mathbf{B}, \mathbf{T} & \mapsto & \mathbf{B}, \mathbf{b} \\
\mathbf{R}, \mathbf{T} & \mapsto & \mathbf{R}, \mathbf{r} \\
\mathbf{T}, \mathbf{T} & \mapsto & \mathbf{T}, \mathbf{t} \\
X, y & \mapsto & X, x
\end{array}
$$

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

A procedure for computing stage graphs

$$\begin{array}{lcl}
\mathbf{B}, \mathbf{R} & \mapsto & \mathbf{T}, \mathbf{t} \\
\mathbf{B}, \mathbf{T} & \mapsto & \mathbf{B}, \mathbf{b} \\
\mathbf{R}, \mathbf{T} & \mapsto & \mathbf{R}, \mathbf{r} \\
\mathbf{T}, \mathbf{T} & \mapsto & \mathbf{T}, \mathbf{t} \\
X, y & \mapsto & X, x
\end{array}$$

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

Transformation graph

B T R

b t r

A procedure for computing stage graphs

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

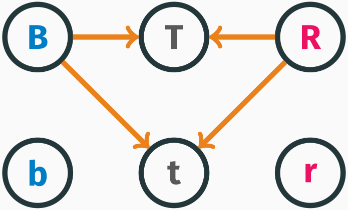
$\mathbf{B}, \mathbf{T} \quad \mapsto \quad \mathbf{B}, \mathbf{b}$

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

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

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

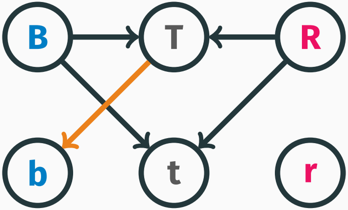
$$\begin{tikzcd}[column sep=small]
& S_0 : (\mathbf{B} \vee \mathbf{R}) \wedge \bigwedge_{q \notin \{\mathbf{B}, \mathbf{R}\}} \neg q \arrow[dl, bend right, "\mathcal{O}(1)"'] \arrow[d, "\mathcal{O}(1)"'] \\
S_1 : \Box \left(\mathbf{B} \wedge \bigwedge_{q \neq \mathbf{B}} \neg q \right) & S_2 : \Box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



A procedure for computing stage graphs

$$\begin{aligned}
\mathbf{B}, \mathbf{R} &\mapsto \mathbf{T}, \mathbf{t} \\
&\boxed{\mathbf{B}, \mathbf{T} \mapsto \mathbf{B}, \mathbf{b}} \\
\mathbf{R}, \mathbf{T} &\mapsto \mathbf{R}, \mathbf{r} \\
\mathbf{T}, \mathbf{T} &\mapsto \mathbf{T}, \mathbf{t} \\
X, y &\mapsto X, x
\end{aligned}$$

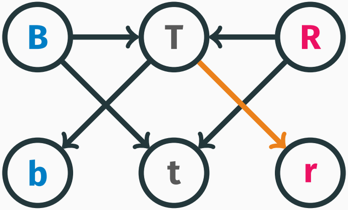
$$\begin{tikzcd}
& S_0 : (\mathbf{B} \lor \mathbf{R}) \land \bigwedge\limits_{q \notin \{\mathbf{B}, \mathbf{R}\}} \neg q \arrow[dl, bend right, "\mathcal{O}(1)"'] \arrow[d, "\mathcal{O}(1)"'] \\
S_1 : \Box \left(\mathbf{B} \land \bigwedge\limits_{q \neq \mathbf{B}} \neg q \right) & S_2 : \Box \left(\mathbf{R}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



A procedure for computing stage graphs

$$\begin{array}{lcl}
\mathbf{B}, \mathbf{R} & \mapsto & \mathbf{T}, \mathbf{t} \\
\mathbf{B}, \mathbf{T} & \mapsto & \mathbf{B}, \mathbf{b} \\
\boxed{\mathbf{R}, \mathbf{T} \ \mapsto \ \mathbf{R}, \mathbf{r}} \\
\mathbf{T}, \mathbf{T} & \mapsto & \mathbf{T}, \mathbf{t} \\
X, y & \mapsto & X, x
\end{array}$$

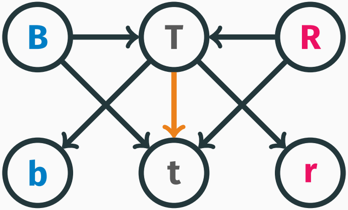
$$\begin{tikzcd}
 & S_0 : (\mathbf{B} \vee \mathbf{R}) \wedge \bigwedge_{q \notin \{\mathbf{B}, \mathbf{R}\}} \neg q \arrow[dl, bend right, "\mathcal{O}(1)"'] \arrow[d, "\mathcal{O}(1)"'] \\
S_1 : \Box \left(\mathbf{B} \wedge \bigwedge_{q \neq \mathbf{B}} \neg q \right) & S_2 : \Box \left(\mathbf{R}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



A procedure for computing stage graphs

$$\begin{array}{lcl}
\mathbf{B}, \mathbf{R} & \mapsto & \mathbf{T}, \mathbf{t} \\
\mathbf{B}, \mathbf{T} & \mapsto & \mathbf{B}, \mathbf{b} \\
\mathbf{R}, \mathbf{T} & \mapsto & \mathbf{R}, \mathbf{r} \\
\boxed{\mathbf{T}, \mathbf{T} \mapsto \mathbf{T}, \mathbf{t}} & & \\
X, y & \mapsto & X, x
\end{array}$$

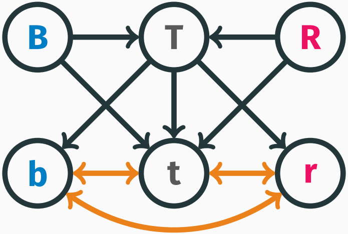
$$\begin{tikzcd}
 & S_0 : (\mathbf{B} \vee \mathbf{R}) \wedge \bigwedge_{q \notin \{\mathbf{B}, \mathbf{R}\}} \neg q \arrow[dl, bend right, "\mathcal{O}(1)"'] \arrow[d, "\mathcal{O}(1)"'] \\
S_1 : \Box \left(\mathbf{B} \wedge \bigwedge_{q \neq \mathbf{B}} \neg q \right) & S_2 : \Box \left(\mathbf{R}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



A procedure for computing stage graphs

$$\mathbf{B}, \mathbf{R} \mapsto \mathbf{T}, \mathbf{t}$$
$$\mathbf{B}, \mathbf{T} \mapsto \mathbf{B}, \mathbf{b}$$
$$\mathbf{R}, \mathbf{T} \mapsto \mathbf{R}, \mathbf{r}$$
$$\mathbf{T}, \mathbf{T} \mapsto \mathbf{T}, \mathbf{t}$$
$$\boxed{X, y \mapsto X, x}$$

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



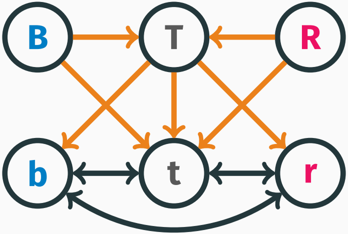
A procedure for computing stage graphs

$$\begin{array}{ccc} \mathbf{B}, \mathbf{R} & \mapsto & \mathbf{T}, \mathbf{t} \\ \mathbf{B}, \mathbf{T} & \mapsto & \mathbf{B}, \mathbf{b} \\ \mathbf{R}, \mathbf{T} & \mapsto & \mathbf{R}, \mathbf{r} \\ \mathbf{T}, \mathbf{T} & \mapsto & \mathbf{T}, \mathbf{t} \\ X, y & \mapsto & X, x \end{array}$$

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

$$S_0 \xrightarrow{\mathcal{O}(1)} S_1 : \Box \left(\mathbf{B} \wedge \bigwedge_{q \neq \mathbf{B}} \neg q \right) \qquad S_0 \xrightarrow{\mathcal{O}(1)} S_2 : \Box \left(\mathbf{R} \wedge \bigwedge_{q \neq \mathbf{R}} \neg q \right)$$

Will become permanently disabled almost surely



A procedure for computing stage graphs

$$\begin{array}{ccc} \mathbf{B}, \mathbf{R} & \mapsto & \mathbf{T}, \mathbf{t} \\ \mathbf{B}, \mathbf{T} & \mapsto & \mathbf{B}, \mathbf{b} \\ \mathbf{R}, \mathbf{T} & \mapsto & \mathbf{R}, \mathbf{r} \\ \mathbf{T}, \mathbf{T} & \mapsto & \mathbf{T}, \mathbf{t} \\ X, y & \mapsto & X, x \end{array}$$

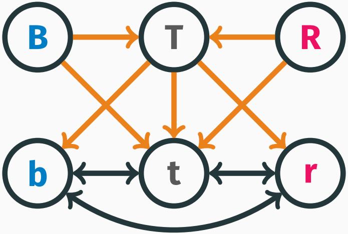
$$S_0 : (\mathbf{B} \vee \mathbf{R}) \wedge \bigwedge_{q \notin \{\mathbf{B}, \mathbf{R}\}} \neg q$$

$\mathcal{O}(1)$

$\mathcal{O}(1)$

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

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



A procedure for computing stage graphs

$$\mathbf{B}, \mathbf{R} \mapsto \mathbf{T}, \mathbf{t}$$
$$\mathbf{B}, \mathbf{T} \mapsto \mathbf{B}, \mathbf{b}$$
$$\mathbf{R}, \mathbf{T} \mapsto \mathbf{R}, \mathbf{r}$$
$$\mathbf{T}, \mathbf{T} \mapsto \mathbf{T}, \mathbf{t}$$
$$X, y \mapsto X, x$$

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

$\mathcal{O}(1)$

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

$\mathcal{O}(1)$

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

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

$$S_4 : \Box \left(\mathbf{B} \wedge \mathbf{b} \wedge \bigwedge_{q \notin \{\mathbf{B},\mathbf{b}\}} \neg q \right) \qquad S_5 : \Box \left(\mathbf{R} \wedge \mathbf{r} \wedge \bigwedge_{q \notin \{\mathbf{R},\mathbf{r}\}} \neg q \right) \qquad S_6 : \Box \left(\mathbf{T!} \wedge \mathbf{t} \wedge \bigwedge_{q \notin \{\mathbf{T},\mathbf{t}\}} \neg q \right)$$

A procedure for computing stage graphs

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

$$
\begin{tikzcd}
 & S_0 : (\mathbf{B} \vee \mathbf{R}) \wedge \bigwedge_{q \notin \{\mathbf{B},\mathbf{R}\}} \neg q \arrow[dl, bend right, "\mathcal{O}(1)"'] \arrow[d, "\mathcal{O}(1)"'] \arrow[dd, bend left, "\mathcal{O}(n^2)"] & \\
S_1 : \Box \left(\mathbf{B} \wedge \bigwedge_{q \neq \mathbf{B}} \neg q \right) & S_2 : \Box \left(\mathbf{R} \wedge \bigwedge_{q \neq \mathbf{R}} \neg q \right) & \\
 & S_3 : \Box \left[(\neg \mathbf{B} \vee \neg \mathbf{R}) \wedge (\neg \mathbf{B} \vee \neg \mathbf{T}) \wedge (\neg \mathbf{R} \vee \neg \mathbf{T}) \wedge (\neg \mathbf{T} \vee \mathbf{T}!) \right] \wedge ((\mathbf{B} \wedge \mathbf{b}) \vee (\mathbf{R} \wedge \mathbf{r}) \vee (\mathbf{T} \wedge \mathbf{t})) \arrow[dl, bend right, "\mathcal{O}(n^2 \log n)"'] \arrow[d, "\mathcal{O}(n^2 \log n)"] \arrow[dr, bend left, "\mathcal{O}(n^2 \log n)"] & \\
S_4 : \Box \left(\mathbf{B} \wedge \mathbf{b} \wedge \bigwedge_{q \notin \{\mathbf{B},\mathbf{b}\}} \neg q \right) & S_5 : \Box \left(\mathbf{R} \wedge \mathbf{r} \wedge \bigwedge_{q \notin \{\mathbf{R},\mathbf{r}\}} \neg q \right) & S_6 : \Box \left(\mathbf{T}! \wedge \mathbf{t} \wedge \bigwedge_{q \notin \{\mathbf{T},\mathbf{t}\}} \neg q \right)
\end{tikzcd}
$$

Experimental results

- Prototype implemented in python™ + Microsoft Z3
- Can report: $\mathcal{O}(1), \mathcal{O}(n^2), \mathcal{O}(n^2 \log n), \mathcal{O}(n^3), \mathcal{O}(\mathrm{poly}(n))$ or $\mathcal{O}(\exp(n))$
- Tested on various protocols from the literature
- Available @ github.com/blondimi/pp-time-analysis

Experimental results

Protocol φ / params.	$\lvert Q \rvert$	$\lvert T \rvert$	Stages	Bound	Time
$x_1 \vee \ldots \vee x_n$ [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 φ / params.	$\lvert Q \rvert$	$\lvert T \rvert$	Stages	Bound	Time
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

- First procedure providing *asymptotic* upper bounds on expected termination time
- Approach promising in practice
- New crucial notions: stage graphs and transformation graphs

- Is our procedure "weakly complete"? *i.e.* for every φ, is there a protocol for φ analyzable by our procedure?
- Approach can be used for verification?
- How to compute lower bounds?

Thank you!