Towards Efficient Verification of Population Protocols

Michael Blondin
Technical University of Munich

Joint work with Javier Esparza, Stefan Jaax and Philipp J. Meyer (PODC'17)

Overview

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

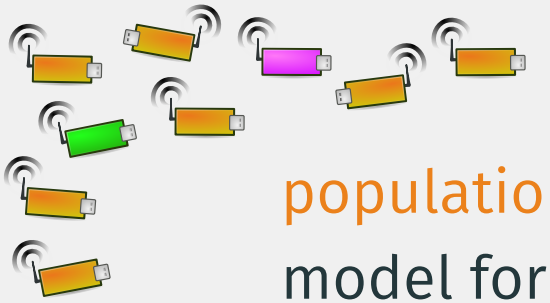
verifying whether a protocol is correct is difficult

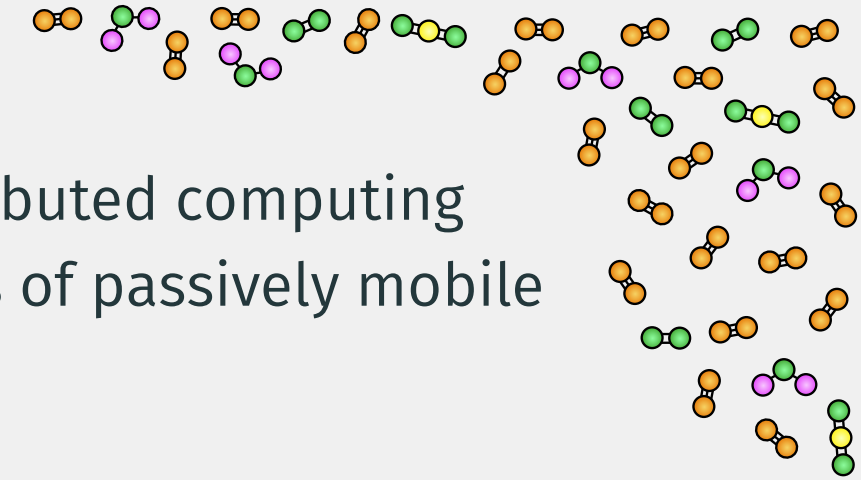
- bounded, but parametric
- EXSPACE-hard / at least as hard as Petri net reachability

new subclass suitable for automatic verification

- same expressiveness
- complexity around NP and coNP
- first tool handling "parametricity"

Overview





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

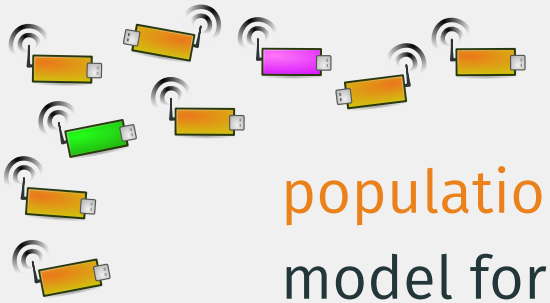
verifying whether a protocol is correct is difficult

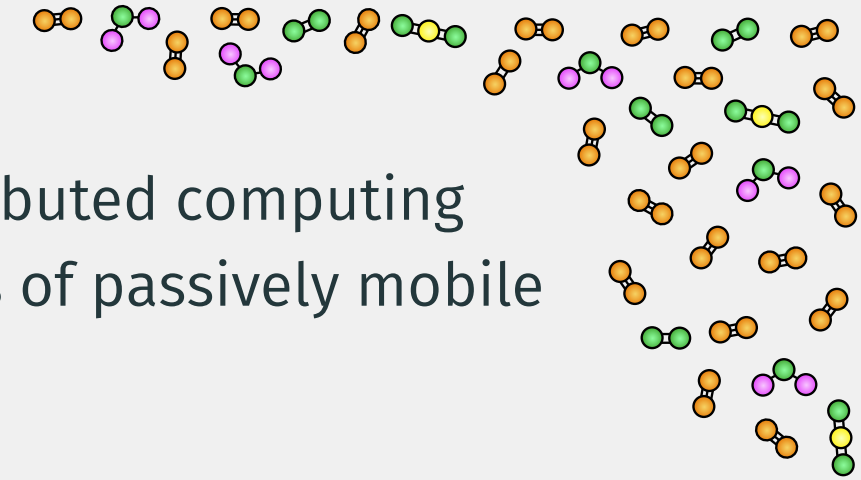
- bounded, but parametric
- EXSPACE-hard / at least as hard as Petri net reachability

new subclass suitable for automatic verification

- same expressiveness
- complexity around NP and coNP
- first tool handling "parametricity"

Overview





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

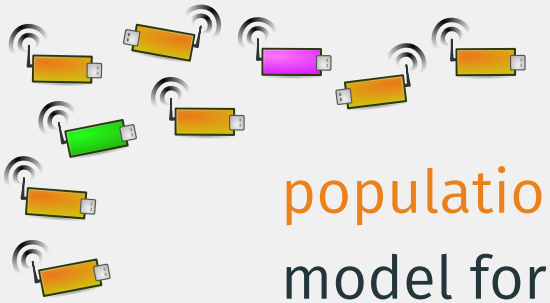
verifying whether a protocol is correct is difficult

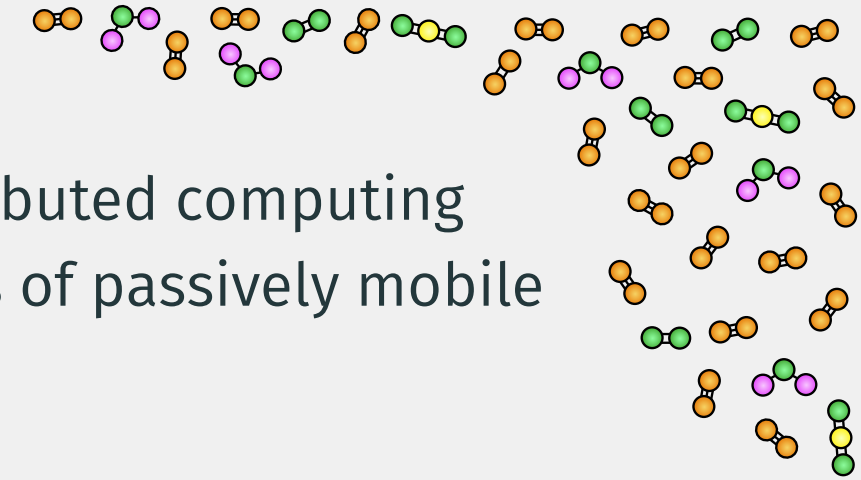
- bounded, but parametric
- EXSPACE-hard / at least as hard as Petri net reachability

new subclass suitable for automatic verification

- same expressiveness
- complexity around NP and coNP
- first tool handling "parametricity"

Overview





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

verifying whether a protocol is correct is difficult

- bounded, but parametric
- EXSPACE-hard / at least as hard as Petri net reachability

new subclass suitable for automatic verification

- same expressiveness
- complexity around NP and coNP
- first tool handling "parametricity"

Population protocols

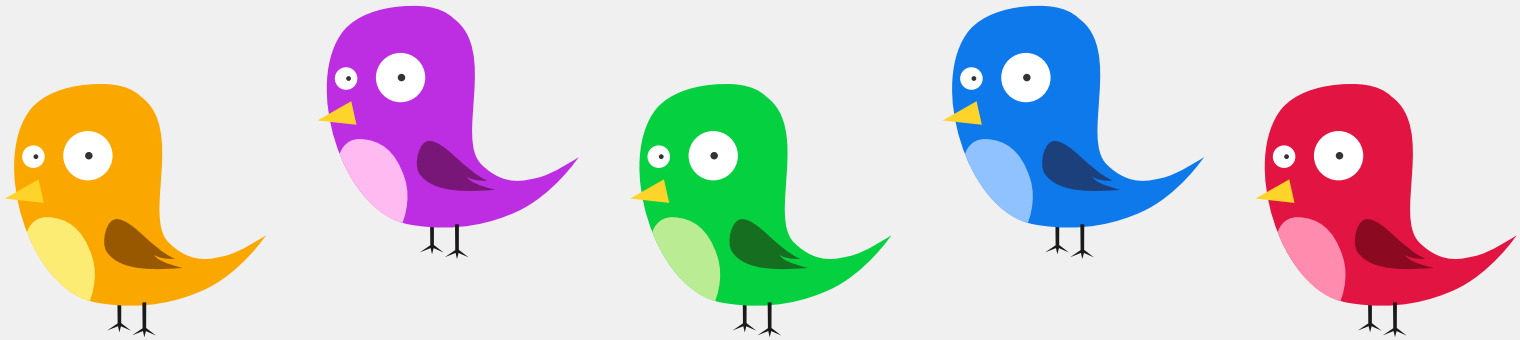
Angluin et al. PODC'04

- 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

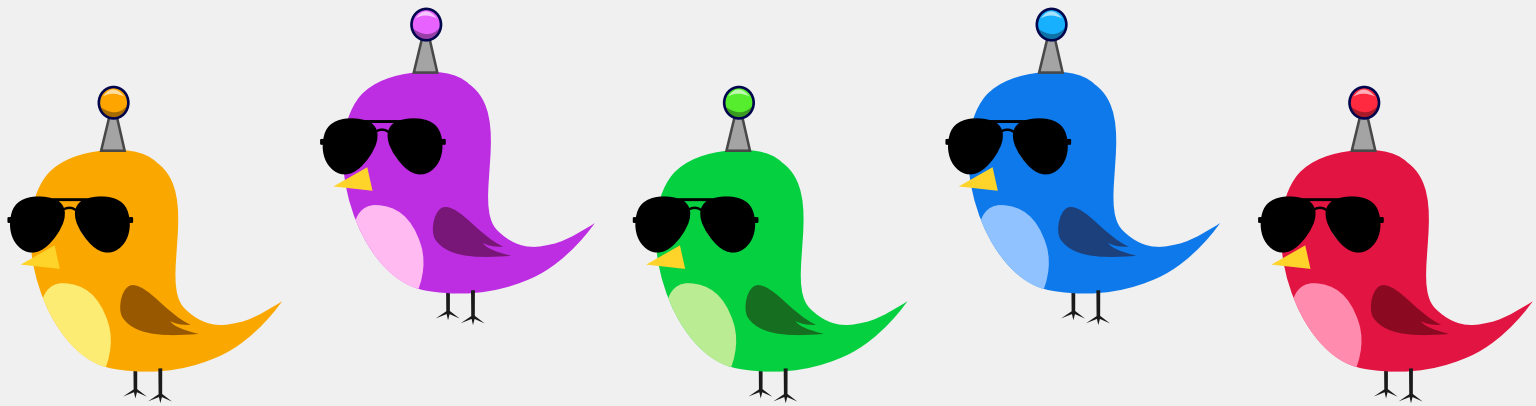
Angluin et al. PODC'04

- 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

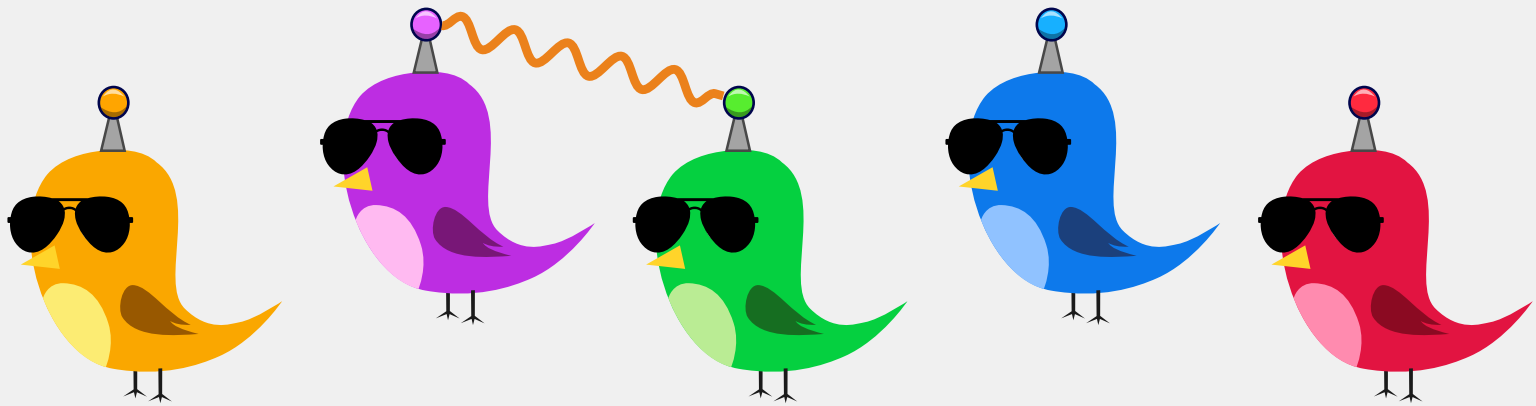
- anonymous mobile agents with very few resources
- agents change states via random pairwise interactions
- each agent has opinion true/false
- computes by stabilizing agents to some opinion



Population protocols

Angluin et al. PODC'04

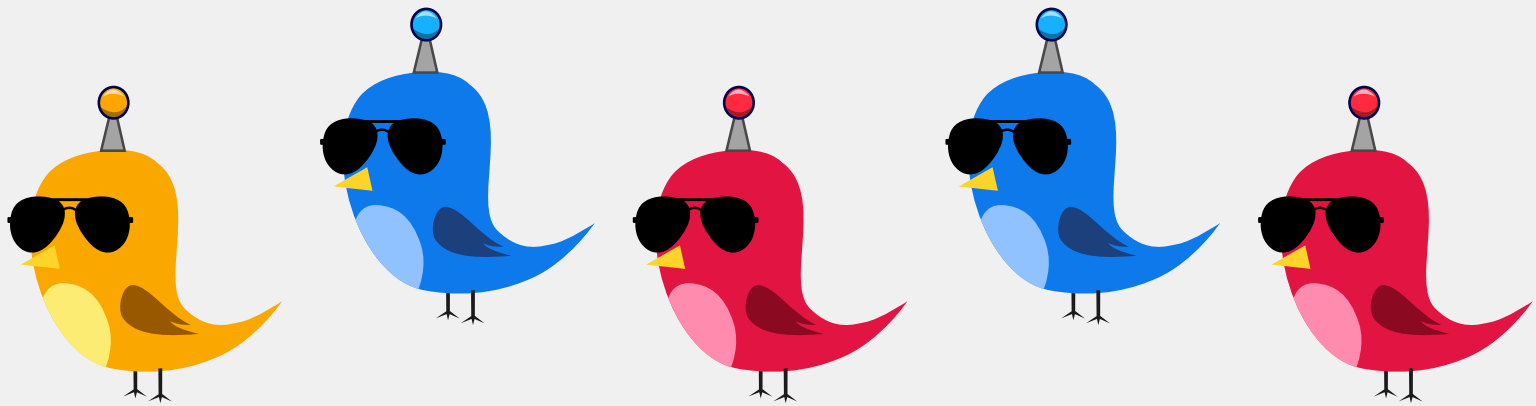
- anonymous mobile agents with very few resources
- agents change states via random pairwise interactions
- each agent has opinion true/false
- computes by stabilizing agents to some opinion



Population protocols

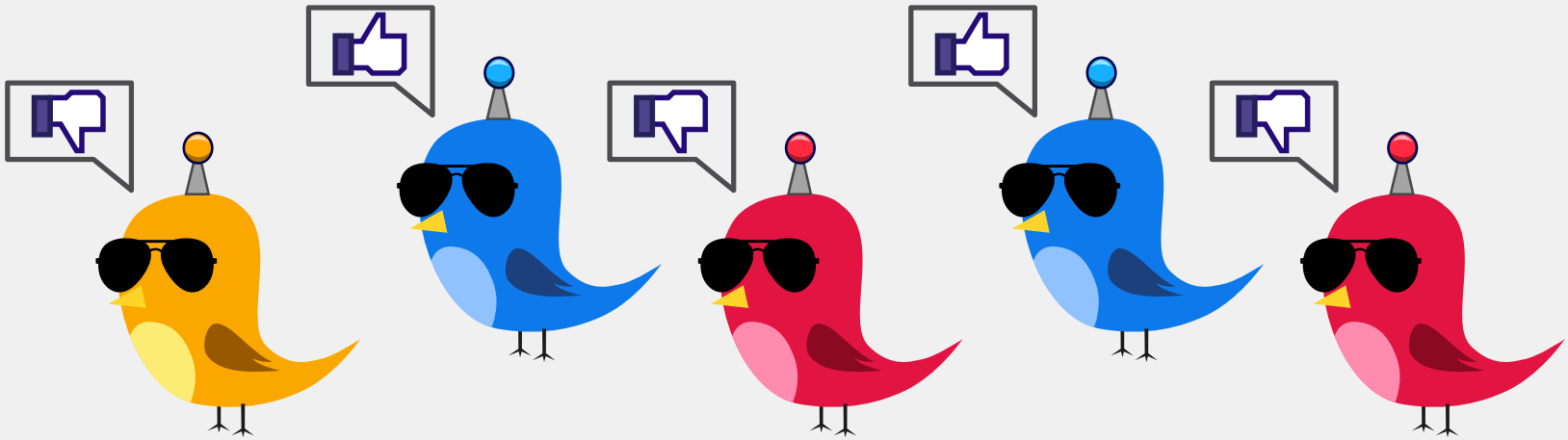
Angluin et al. PODC'04

- 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

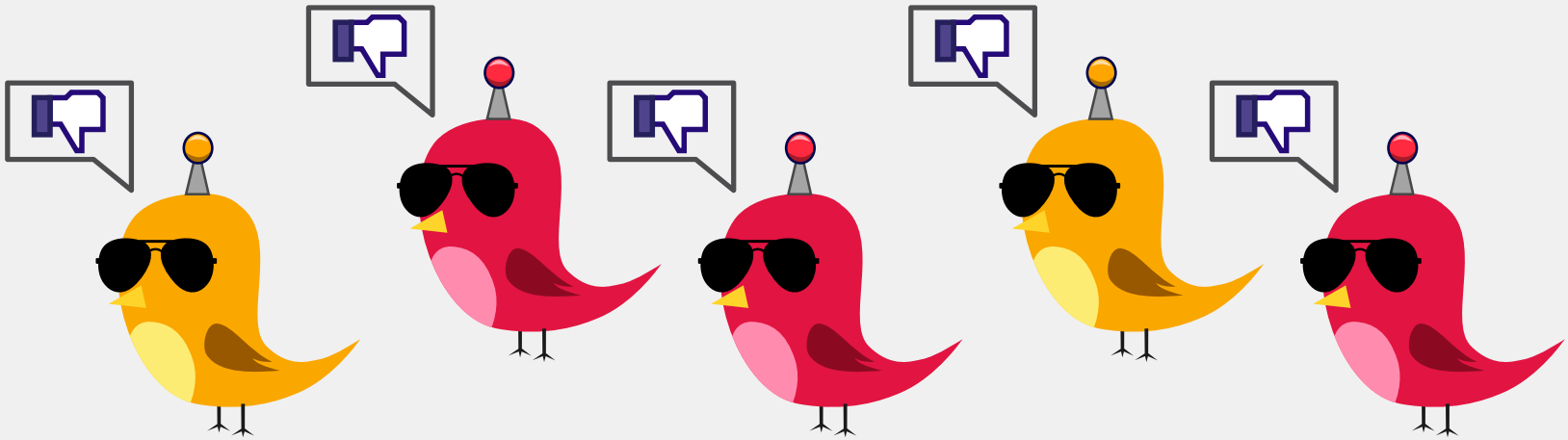
- anonymous mobile agents with very few resources
- agents change states via random pairwise interactions
- each agent has opinion true/false
- computes by stabilizing agents to some opinion



Population protocols

Angluin et al. PODC'04

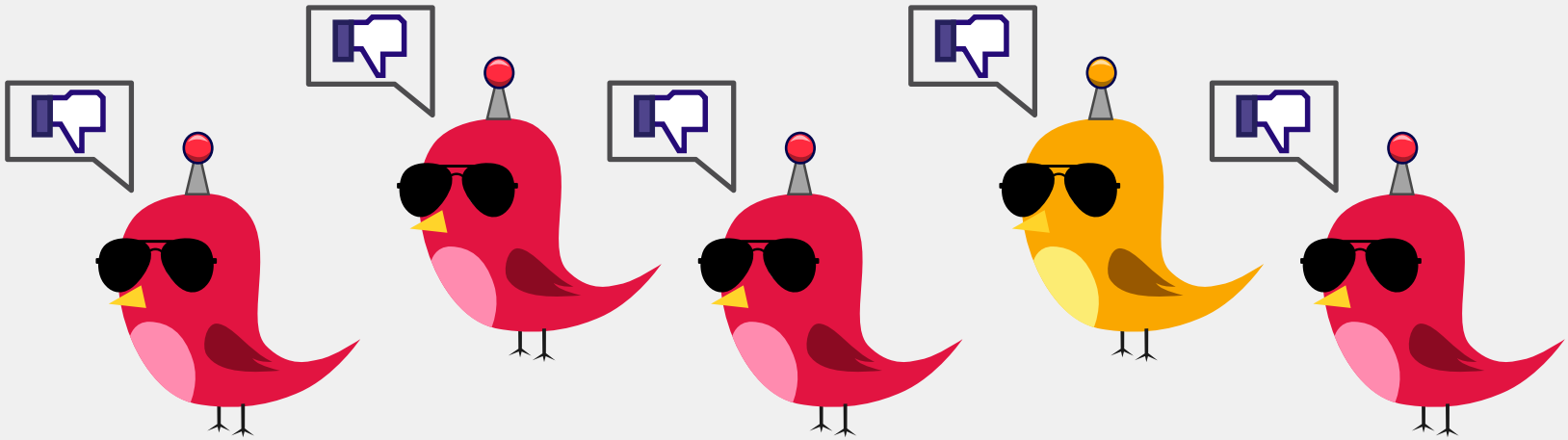
- anonymous mobile agents with very few resources
- agents change states via random pairwise interactions
- each agent has opinion true/false
- computes by stabilizing agents to some opinion



Population protocols

Angluin et al. PODC'04

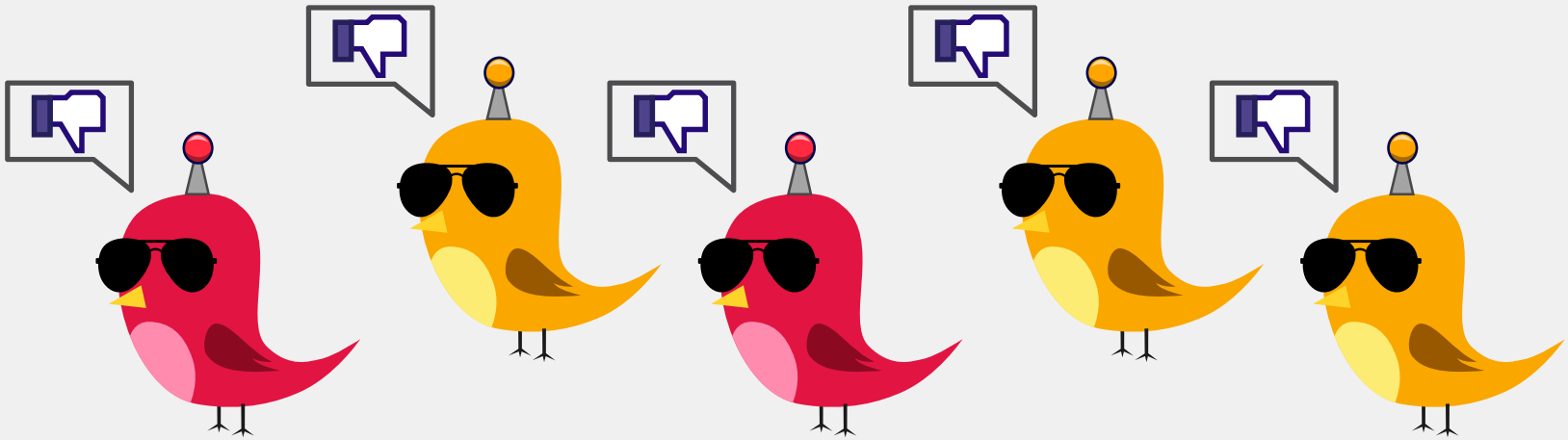
- anonymous mobile agents with very few resources
- agents change states via random pairwise interactions
- each agent has opinion true/false
- computes by stabilizing agents to some opinion



Population protocols

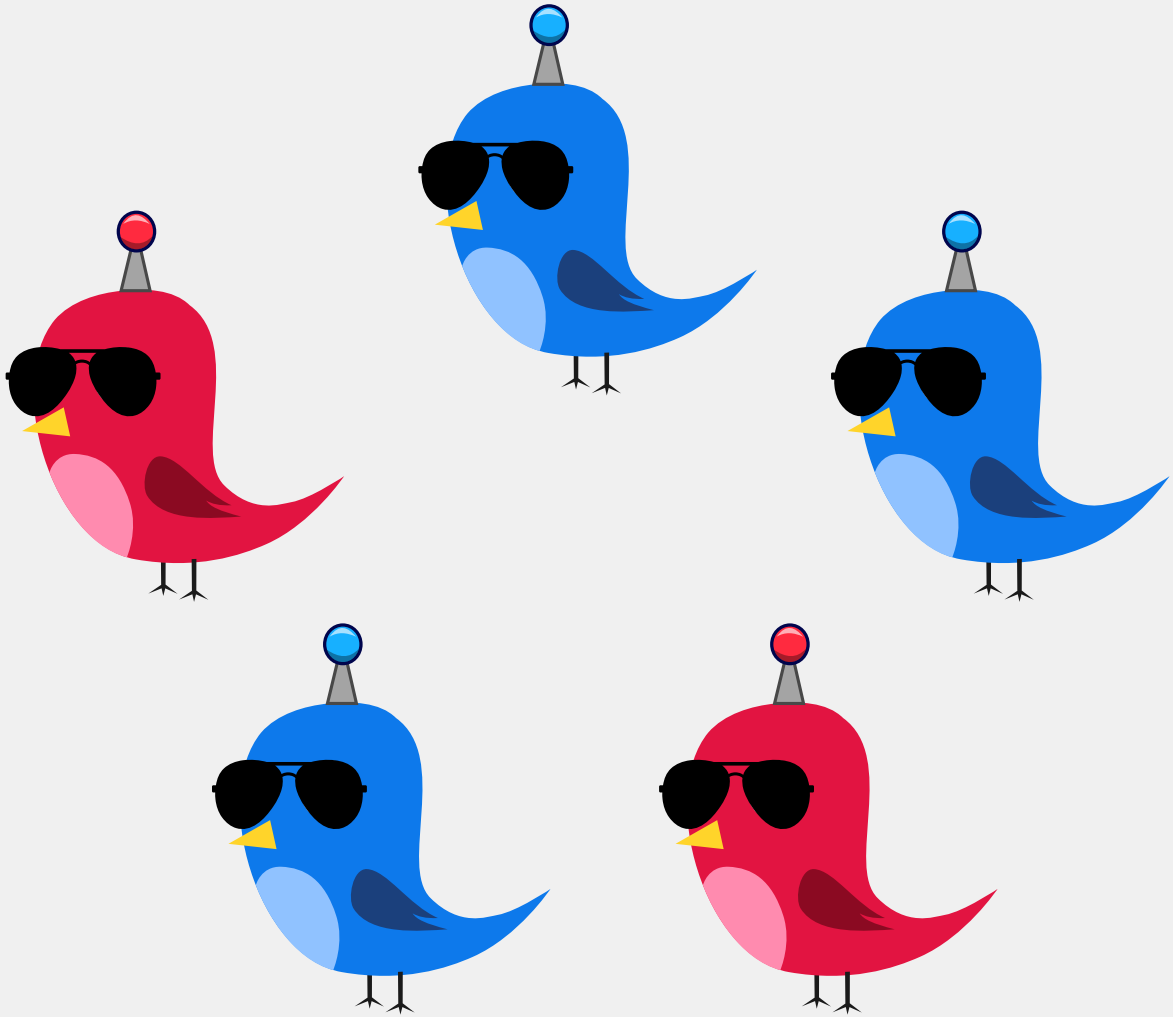
Angluin et al. PODC'04

- 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 : majority protocol

More blue birds than red birds ?

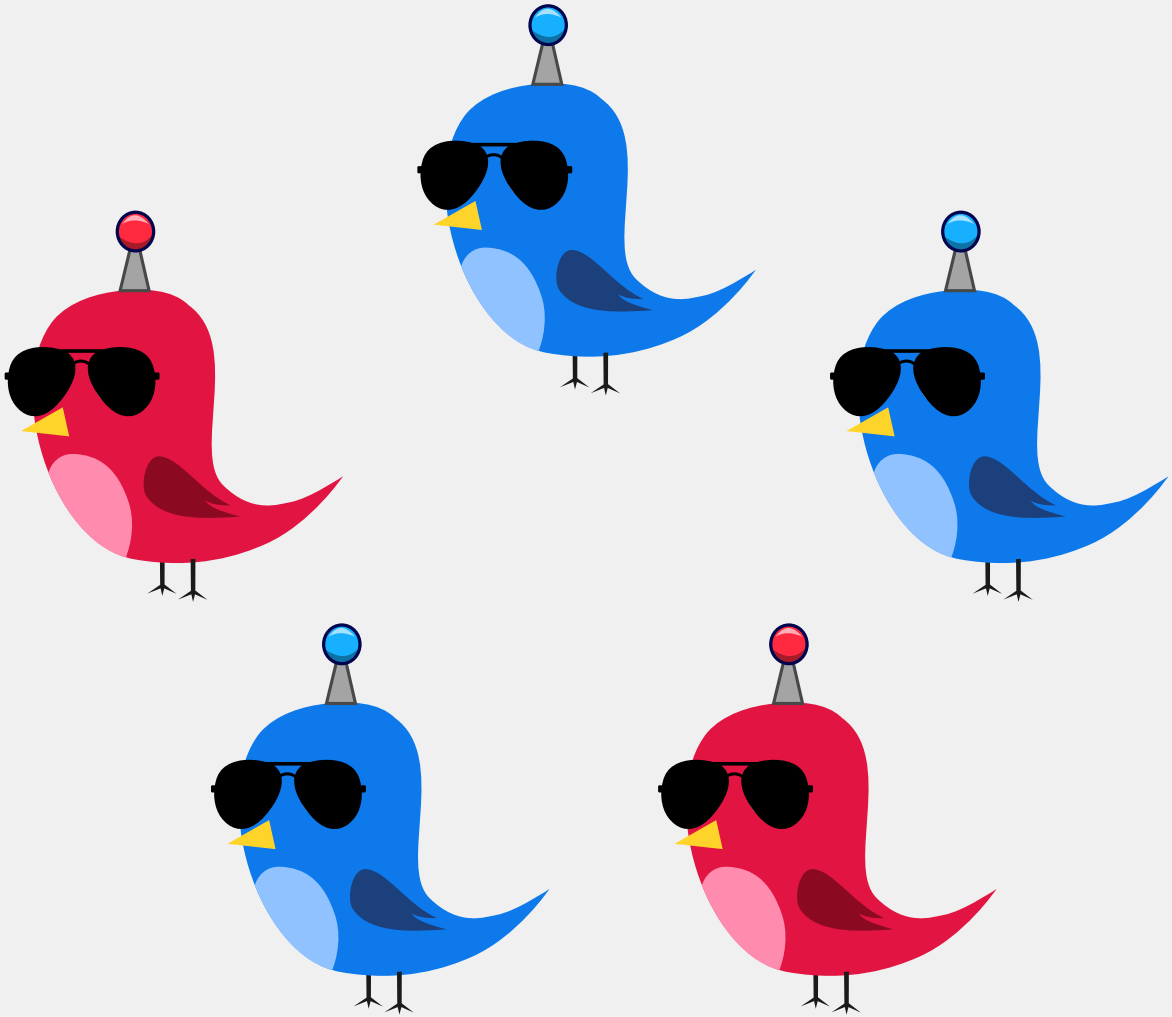


Example : majority protocol

More blue birds than red birds ?

Interactions :

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

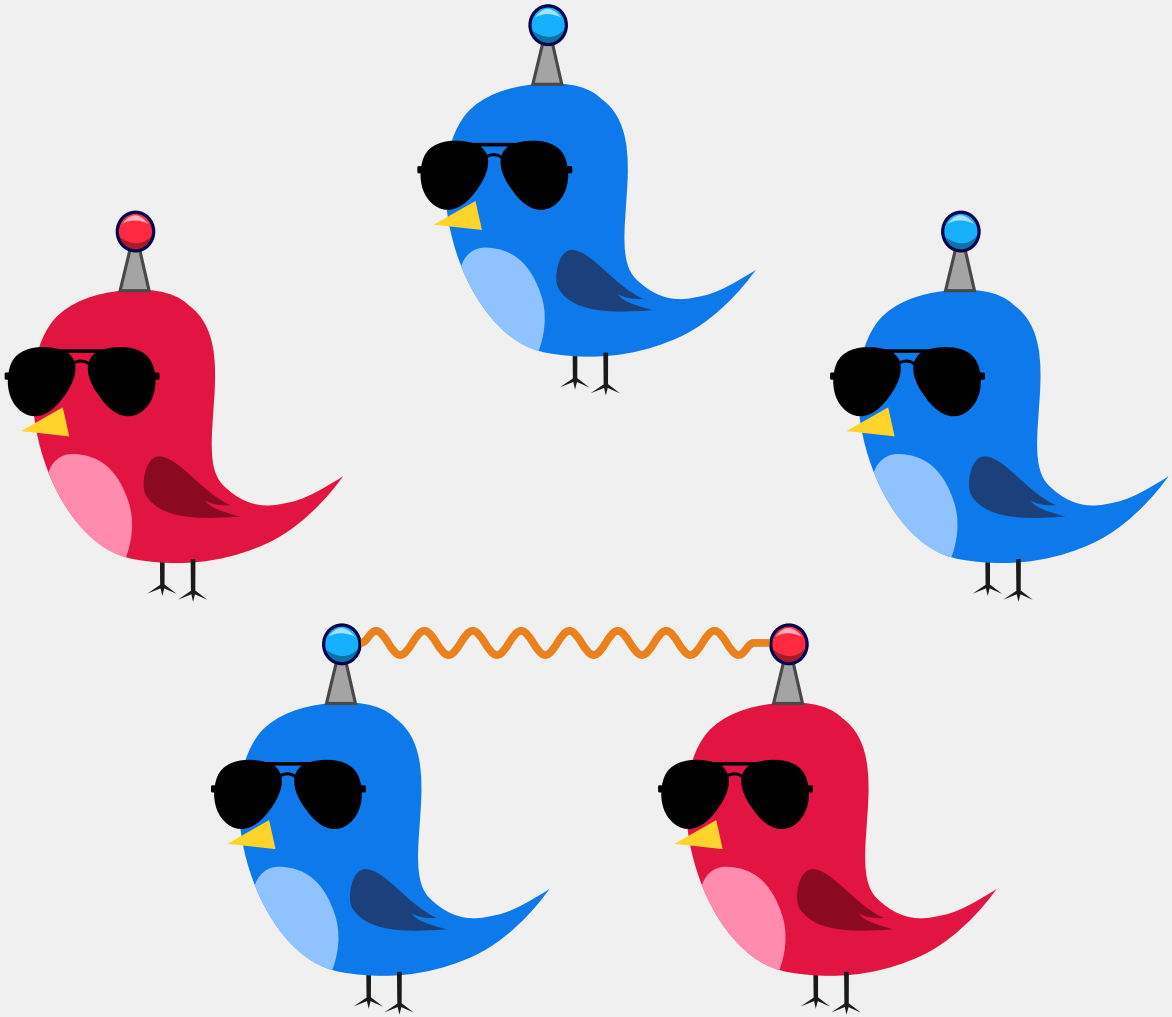


Example : majority protocol

More blue birds than red birds ?

Interactions :

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

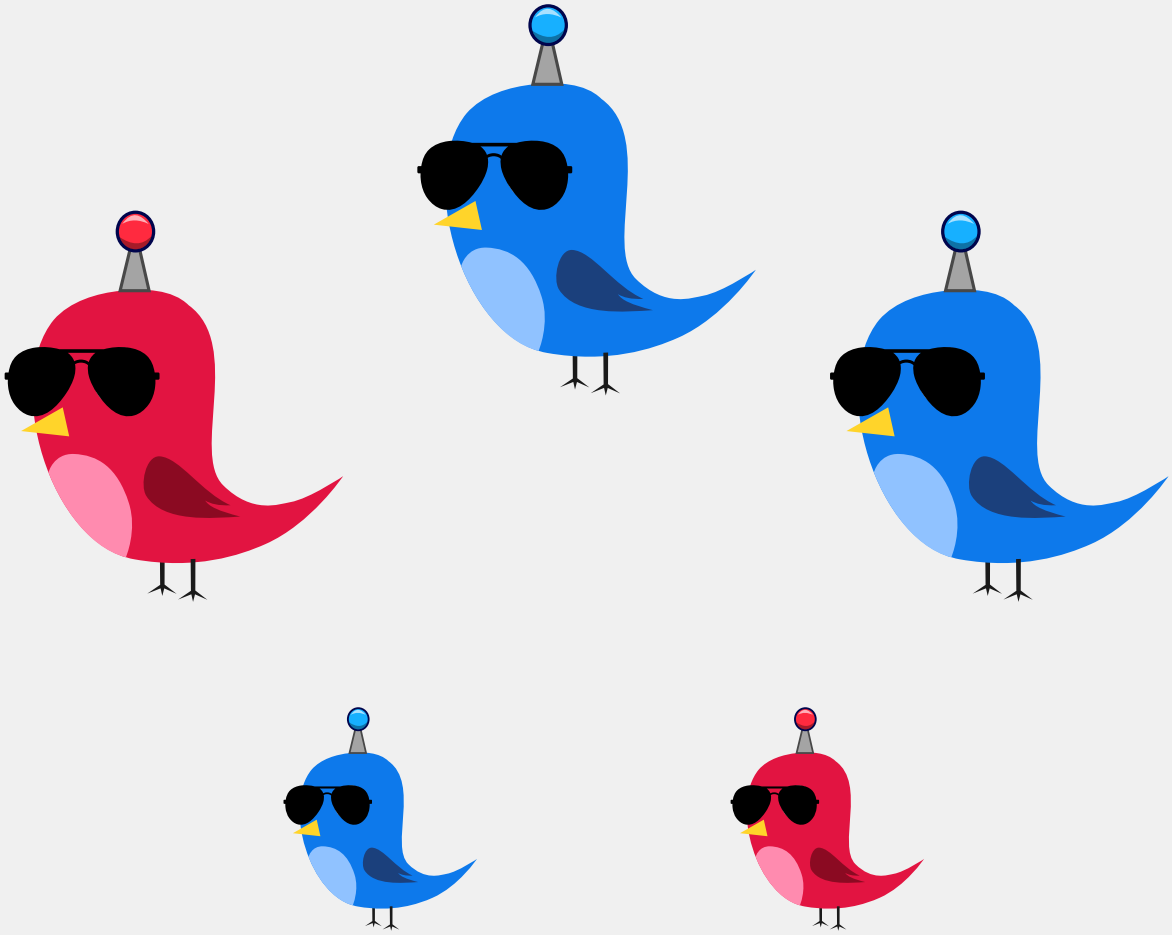


Example : majority protocol

More blue birds than red birds ?

Interactions :

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

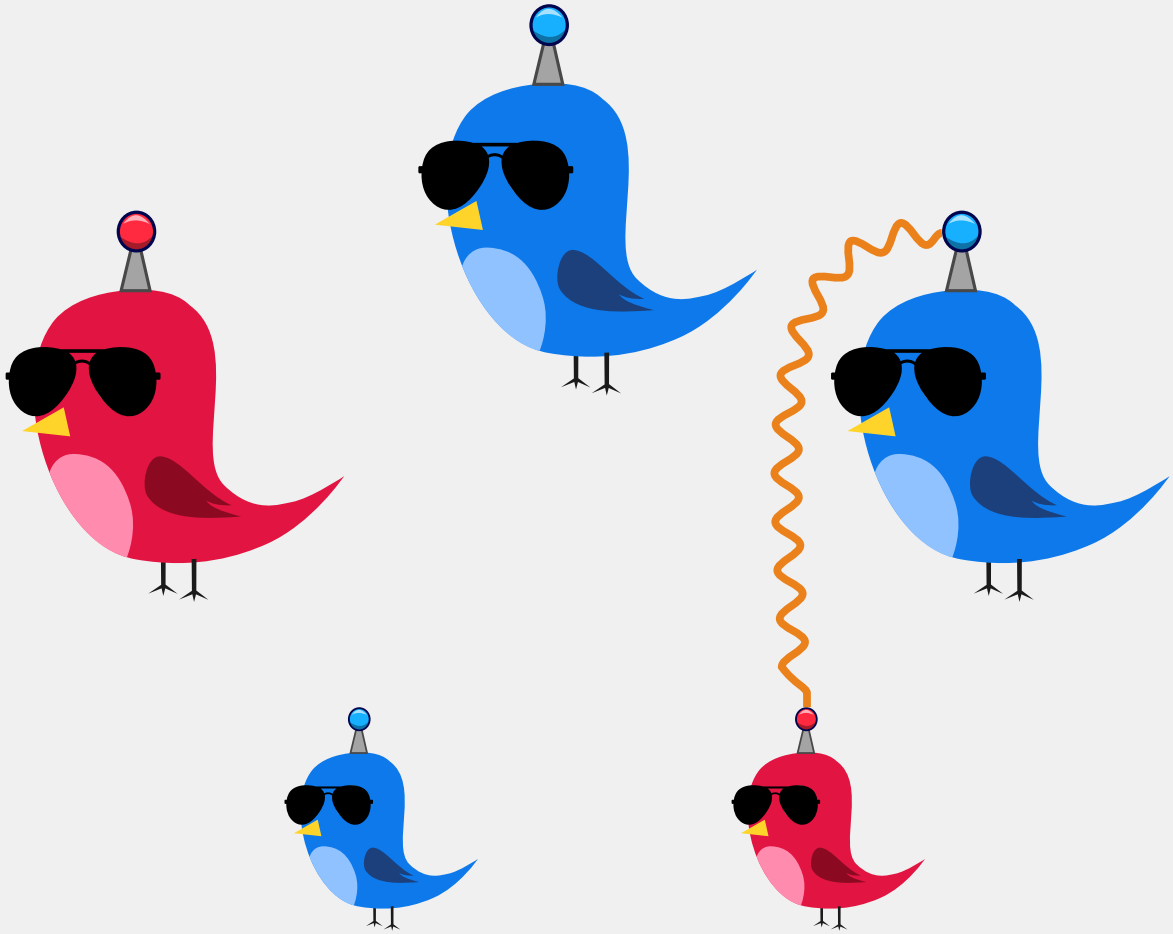


Example : majority protocol

More blue birds than red birds ?

Interactions :

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

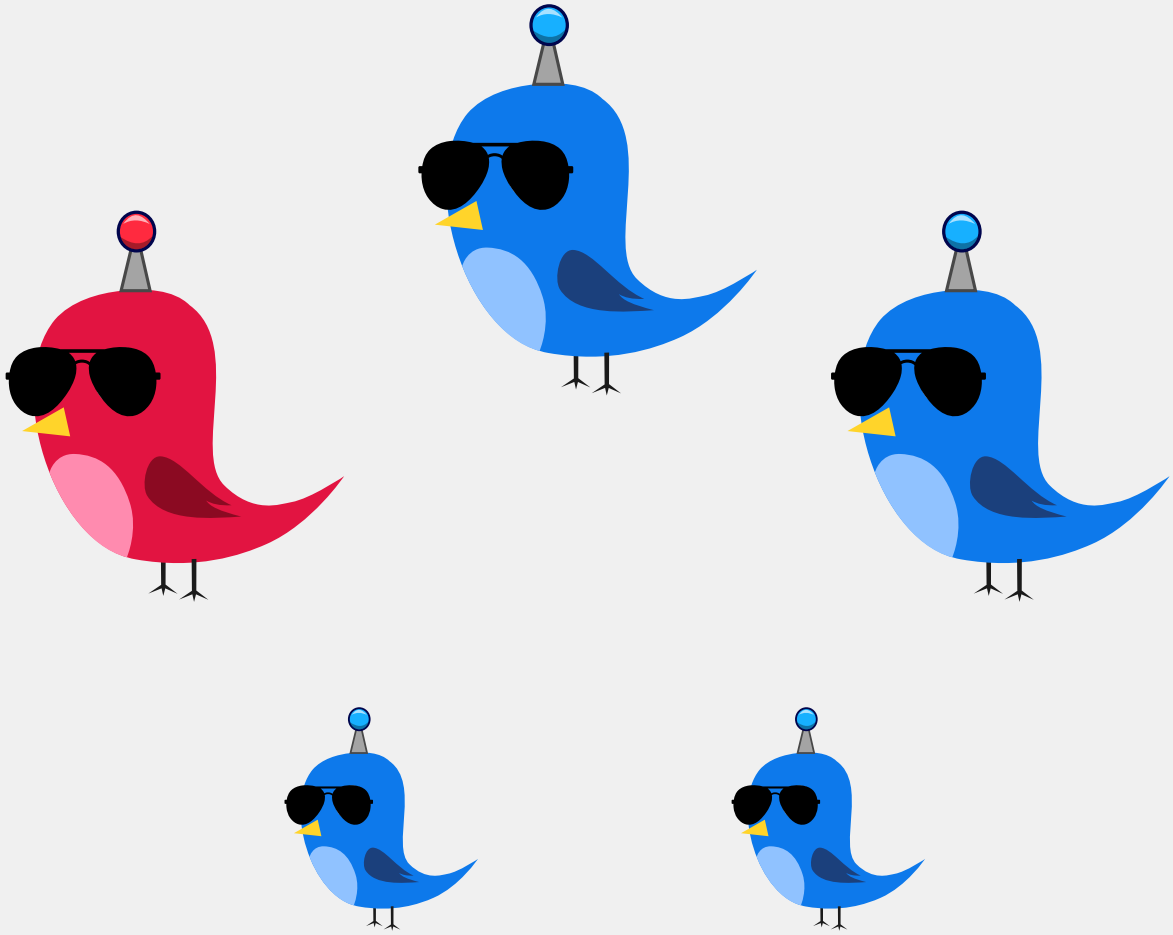


Example : majority protocol

More blue birds than red birds ?

Interactions :

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

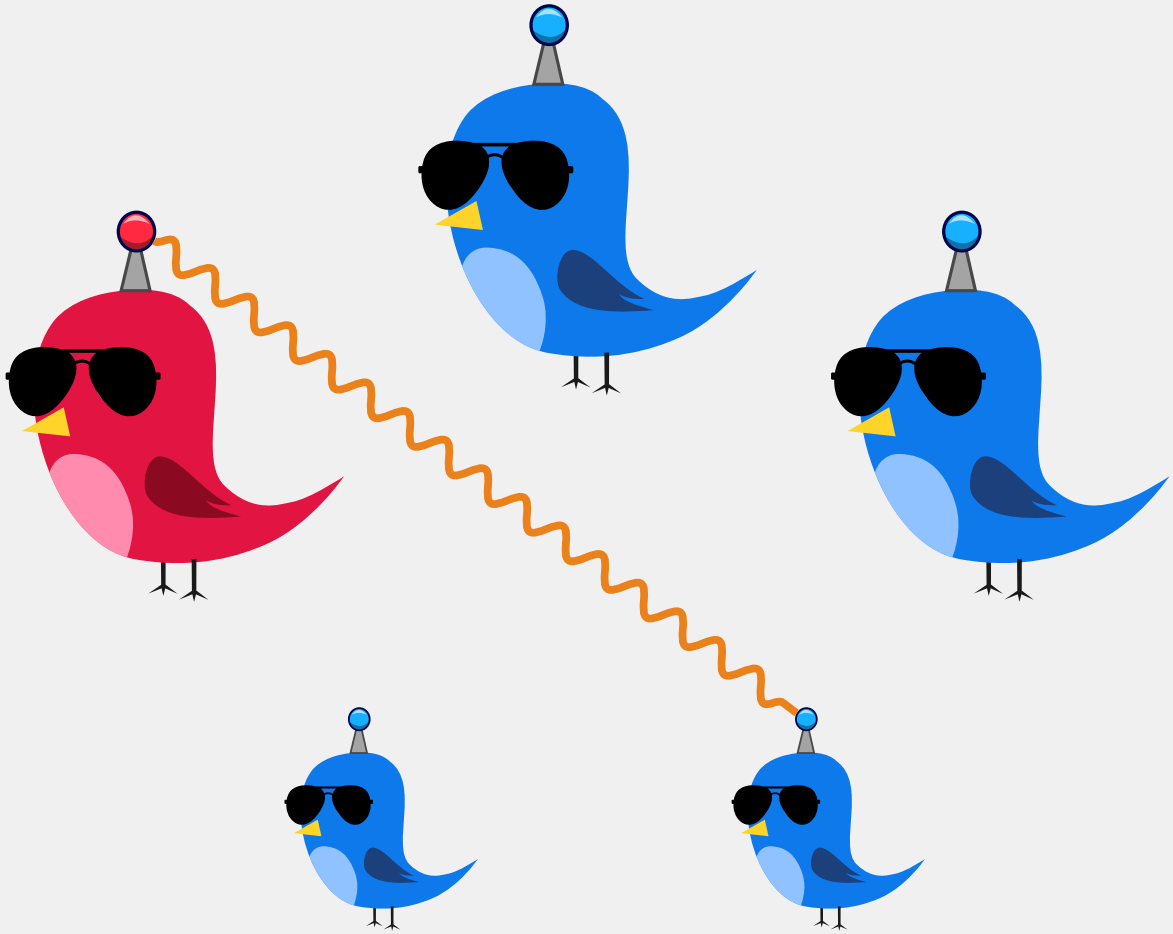


Example : majority protocol

More blue birds than red birds ?

Interactions :

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

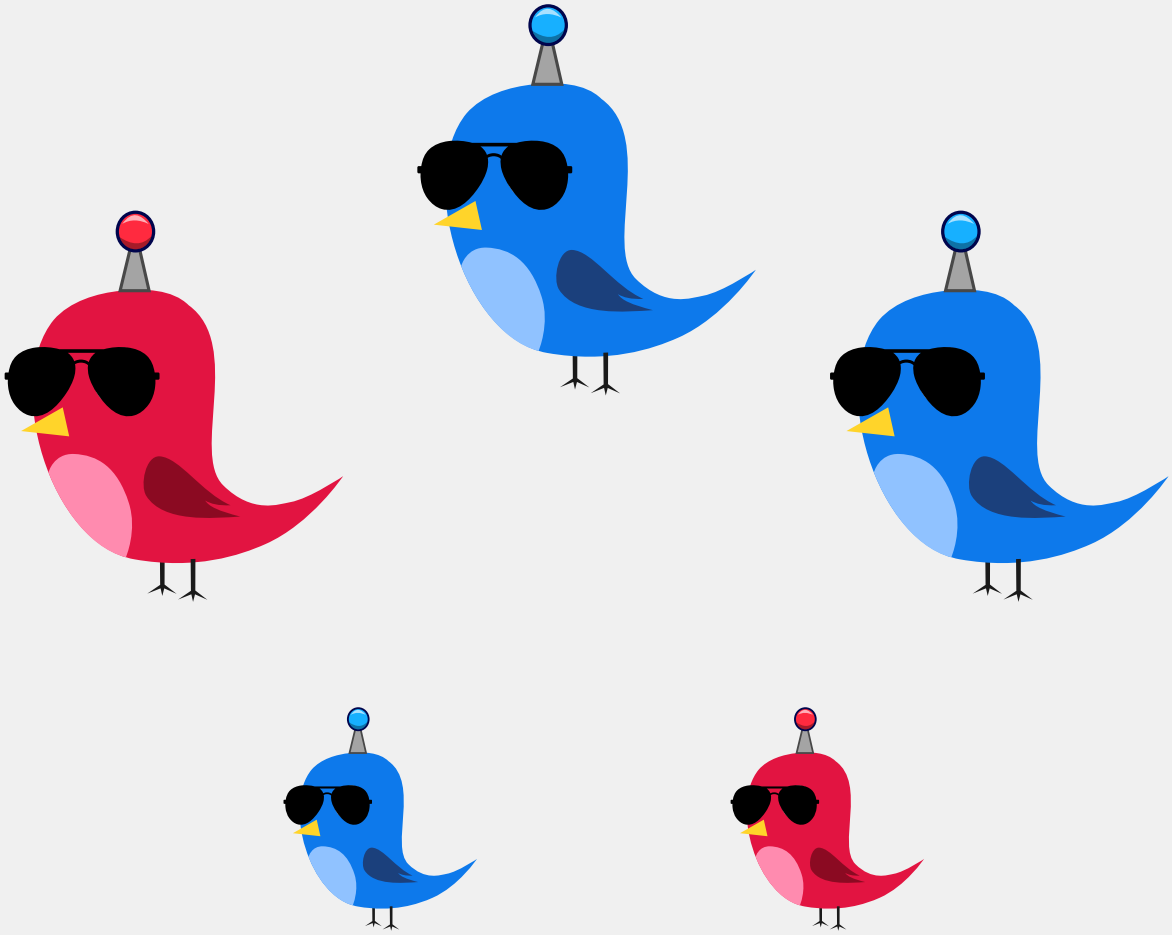


Example : majority protocol

More blue birds than red birds ?

Interactions :

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

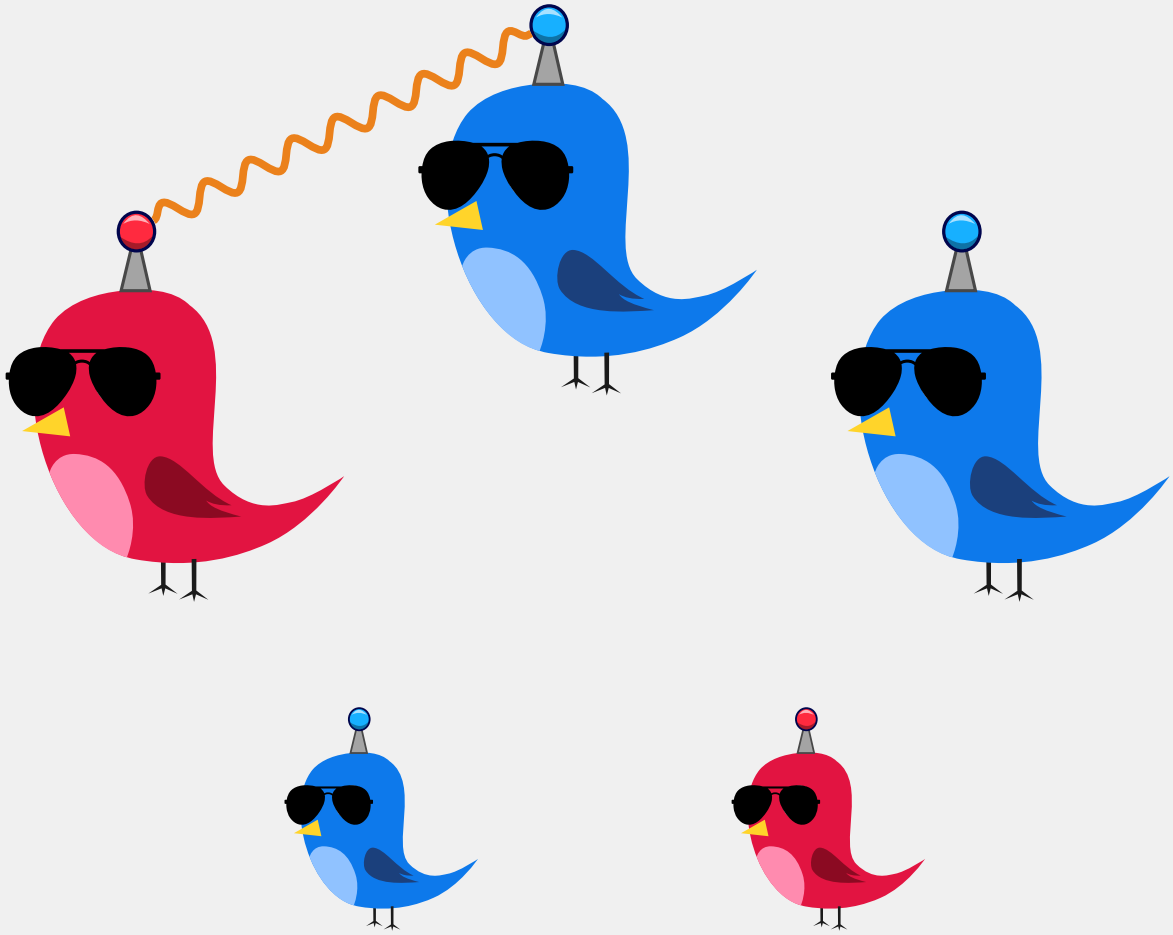


Example : majority protocol

More blue birds than red birds ?

Interactions :

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

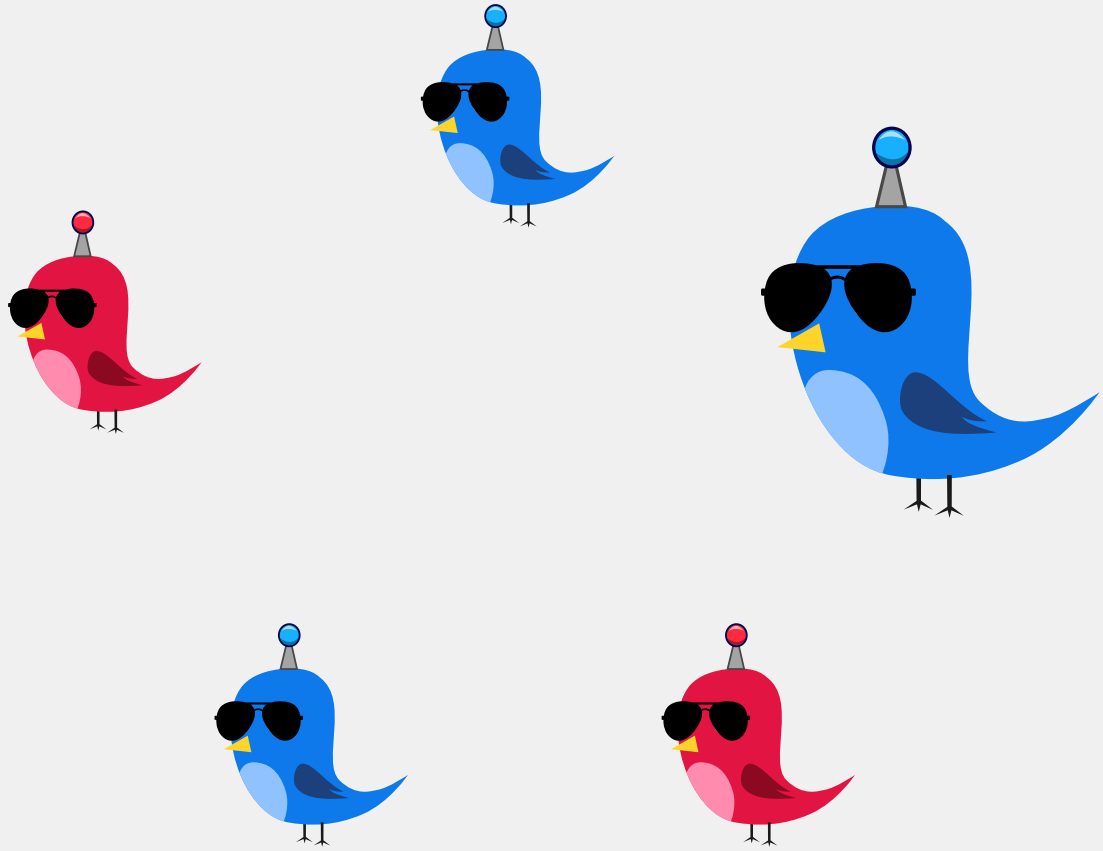


Example : majority protocol

More blue birds than red birds ?

Interactions :

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

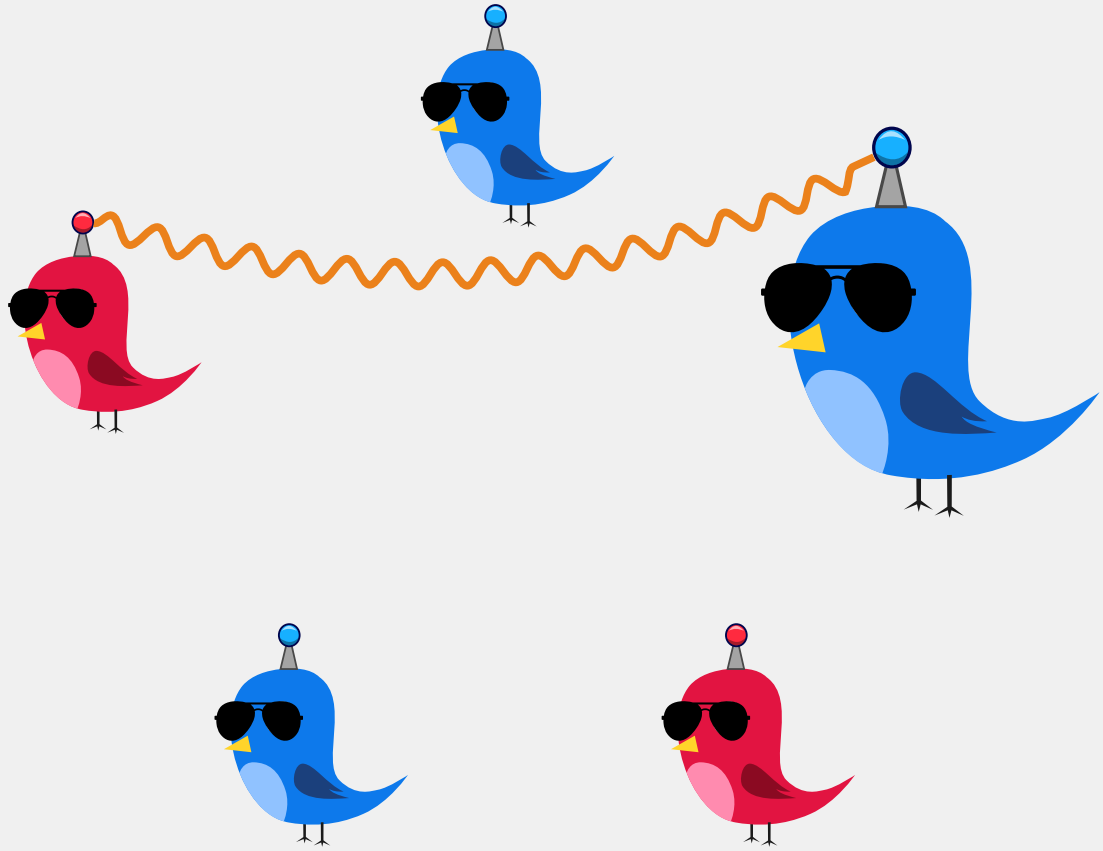


Example : majority protocol

More blue birds than red birds ?

Interactions :

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

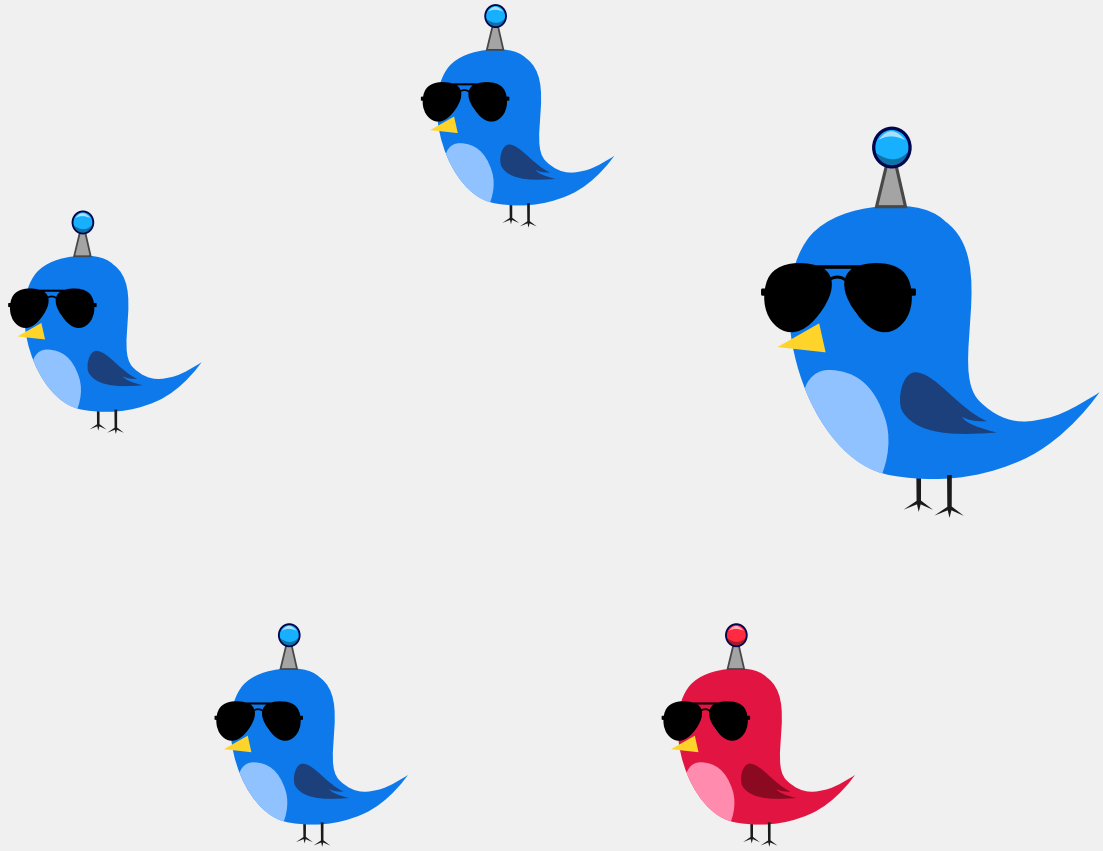


Example : majority protocol

More blue birds than red birds ?

Interactions :

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

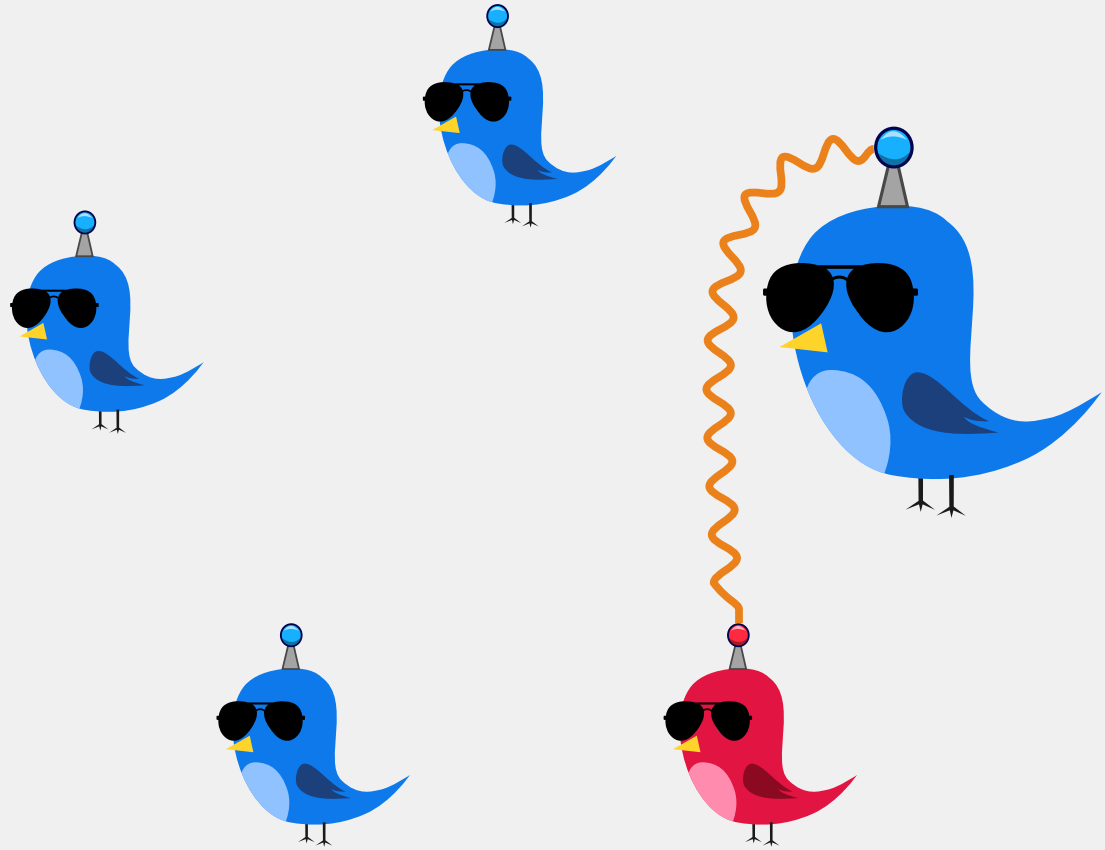


Example : majority protocol

More blue birds than red birds ?

Interactions :

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

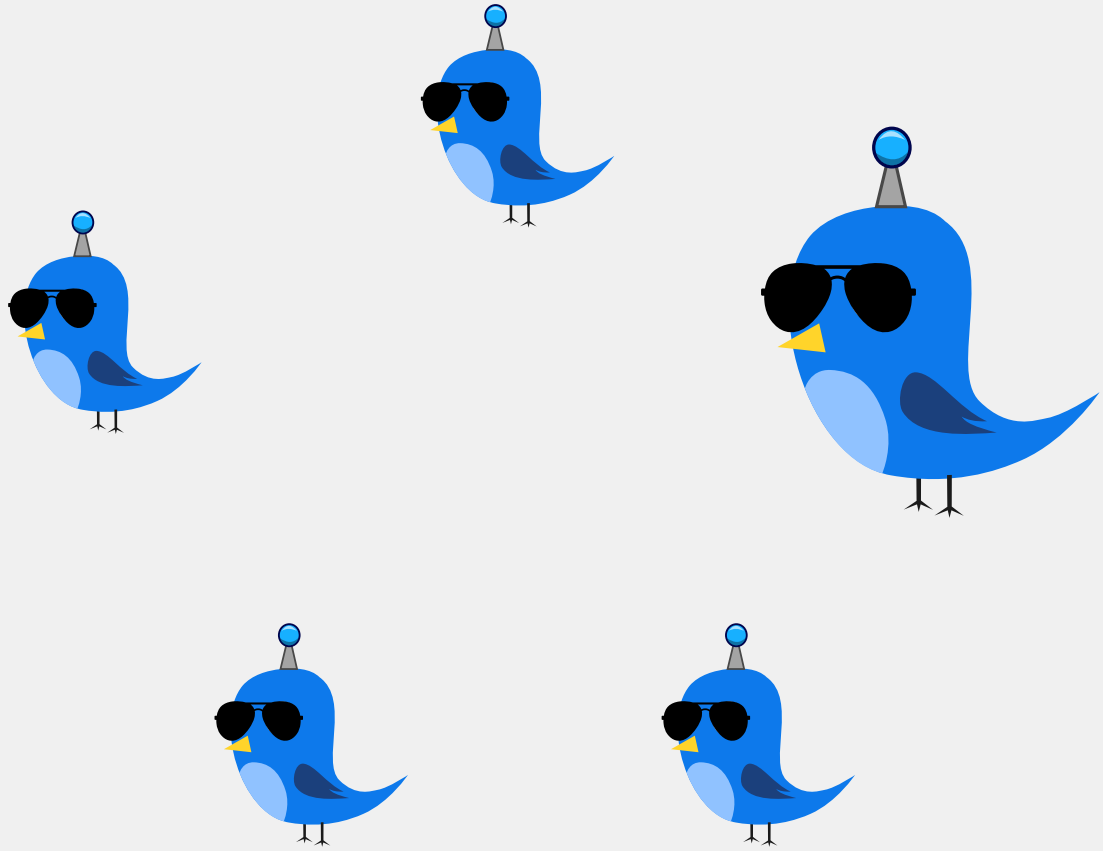


Example : majority protocol

More blue birds than red birds ?

Interactions :

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

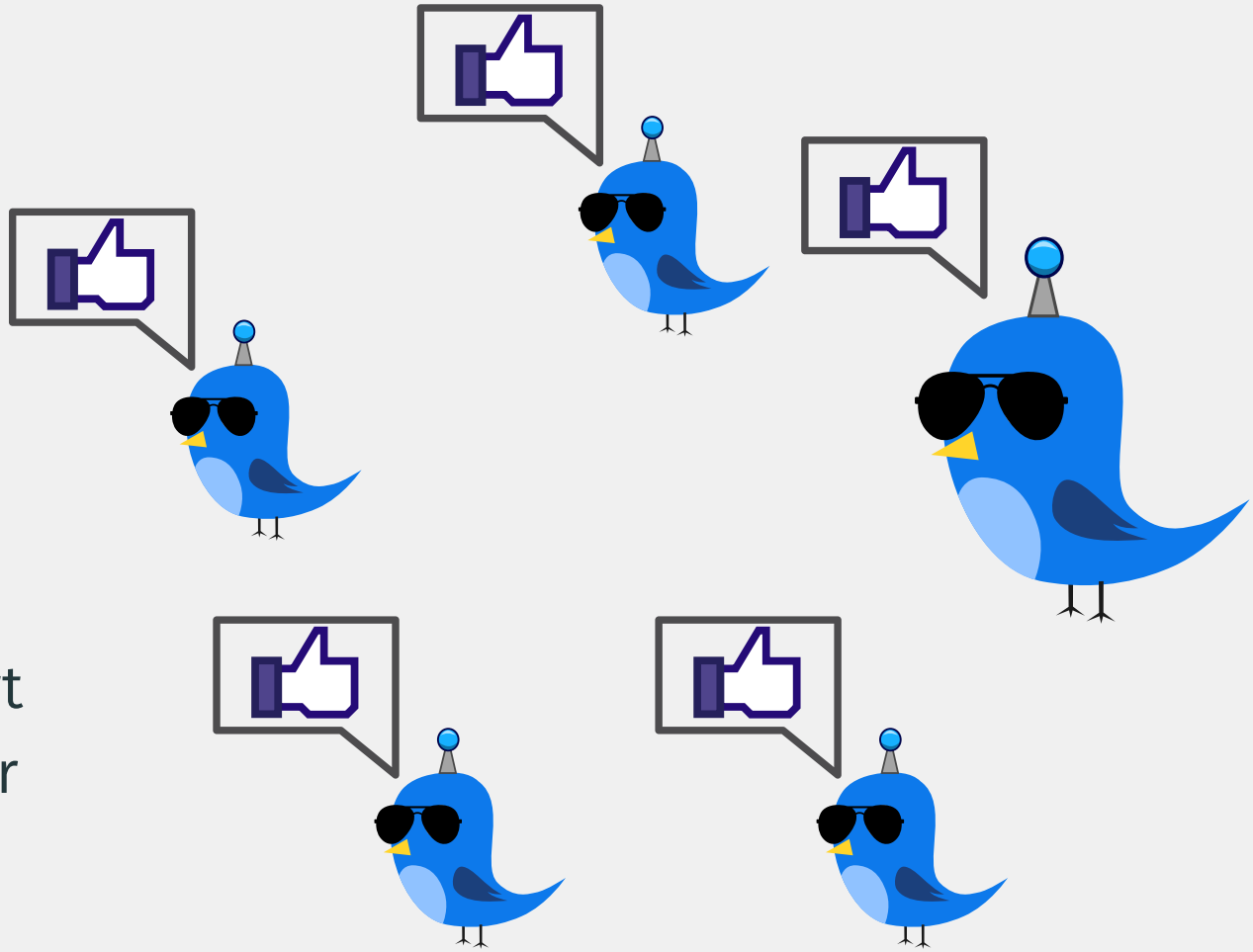


Example : majority protocol

More blue birds than red birds ?

Interactions :

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



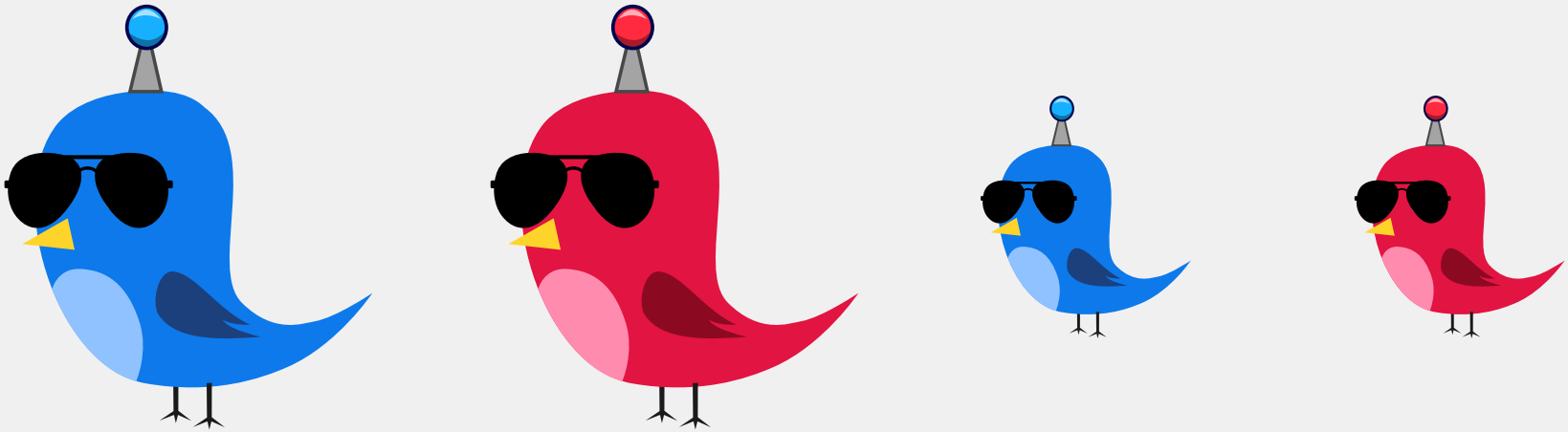
Demonstration

Population protocols : definitions

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

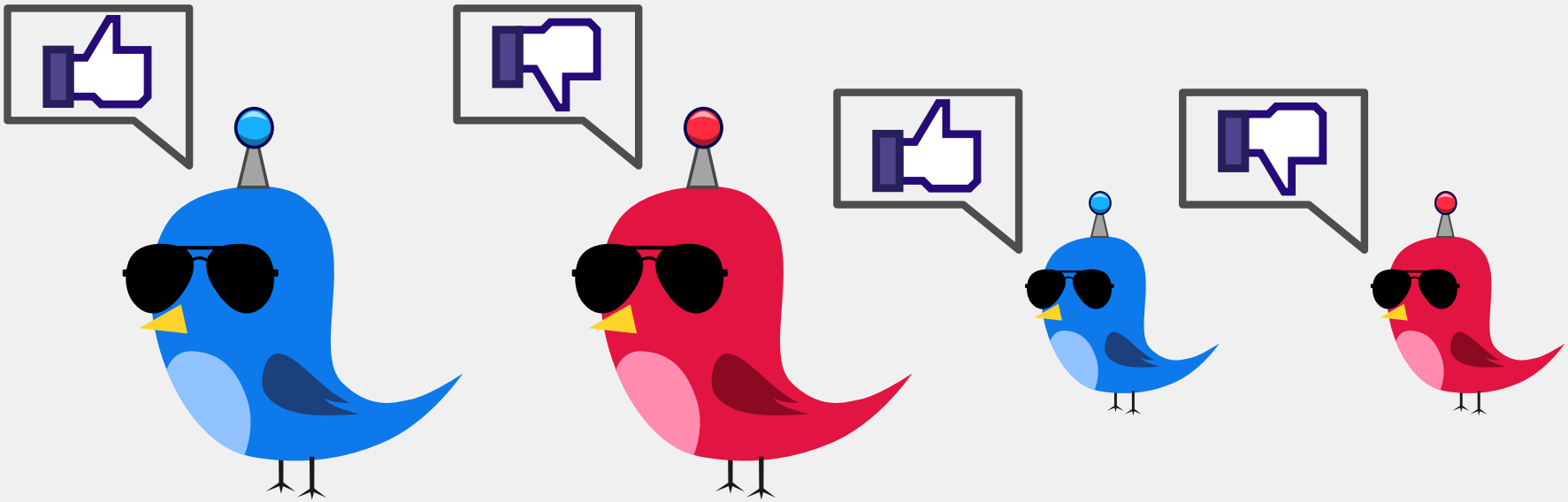
Population protocols : definitions

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



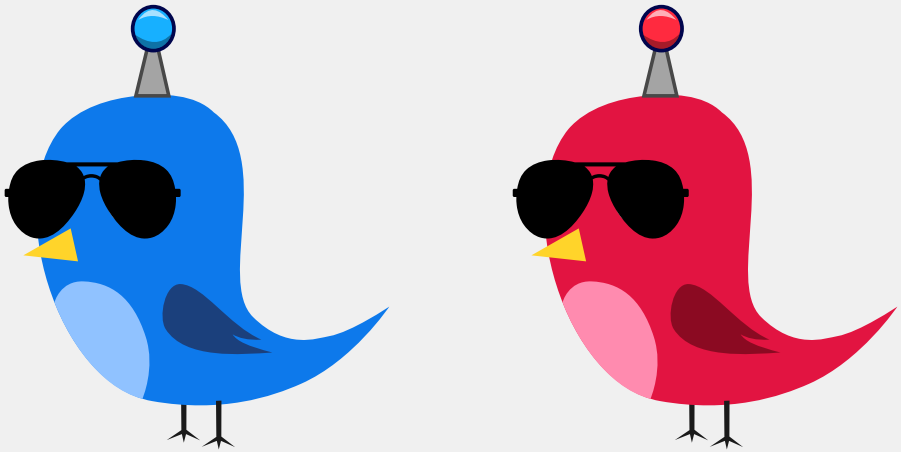
Population protocols : definitions

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



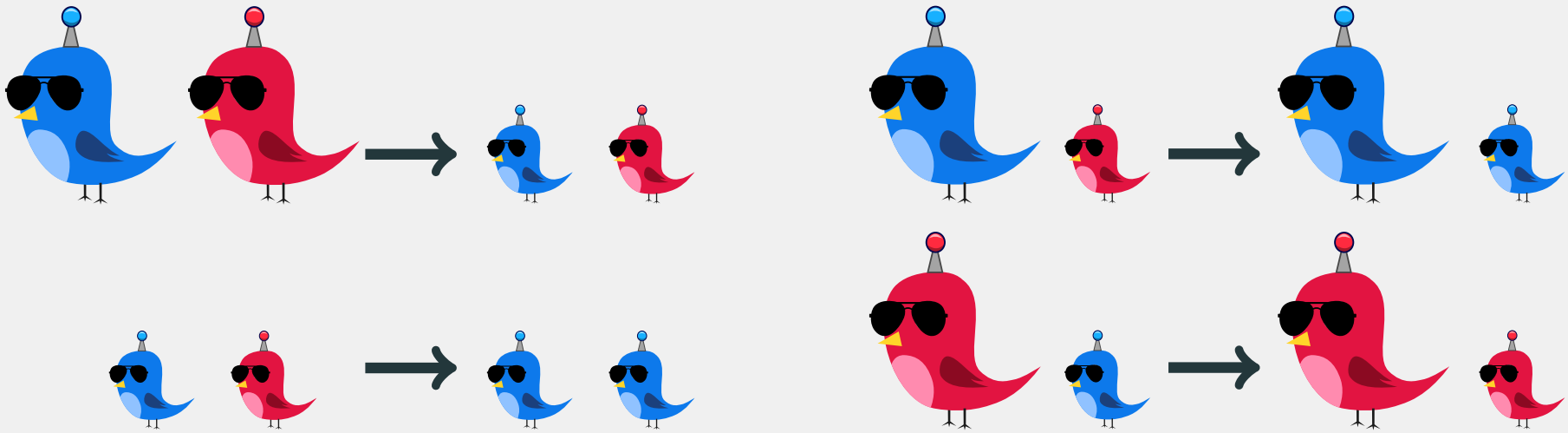
Population protocols : definitions

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



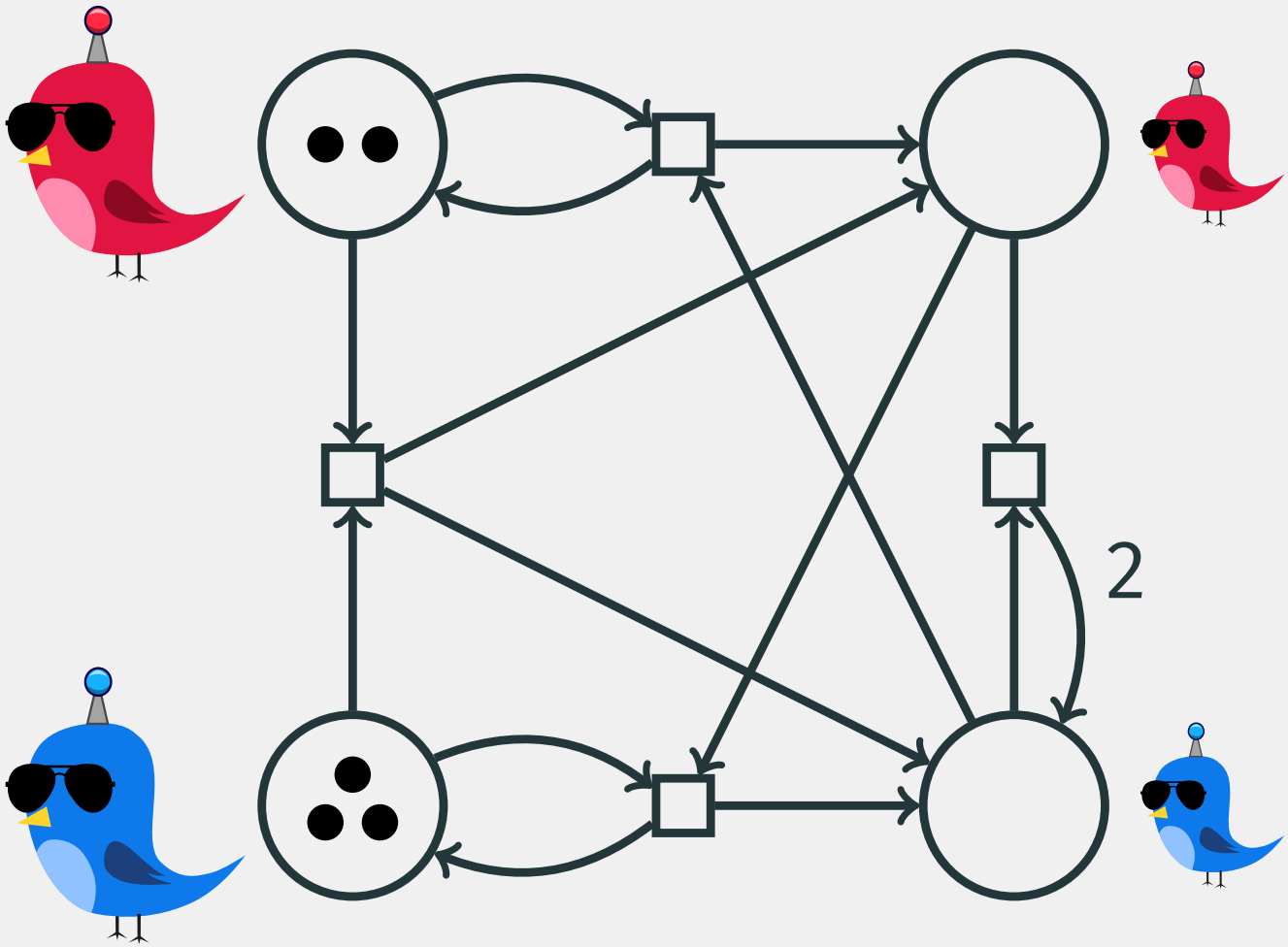
Population protocols : definitions

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



Population protocols : definitions

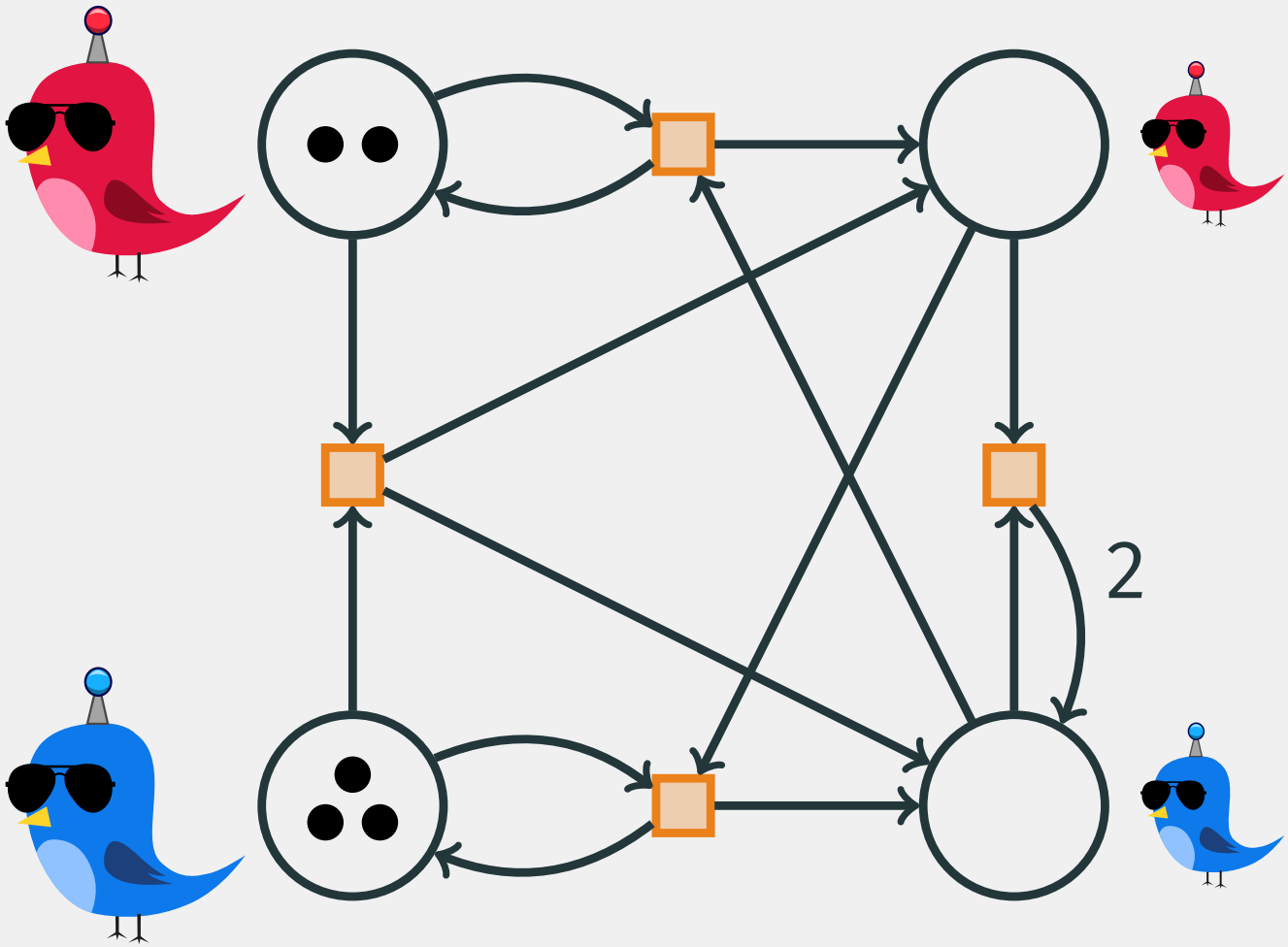
Protocols can be translated into Petri nets



Population protocols : definitions

Protocols can be translated into Petri nets

conservative / bounded



Population protocols : definitions

- Transition $(p, q) \mapsto (p', q')$ ...

  ... is *enabled* if $C = \{p, q, \ldots\}$

  ... *leads to* C' if $C' = C - \{p, q\} + \{p', q'\}$

  ... is *silent* if $\{p, q\} = \{p', q'\}$

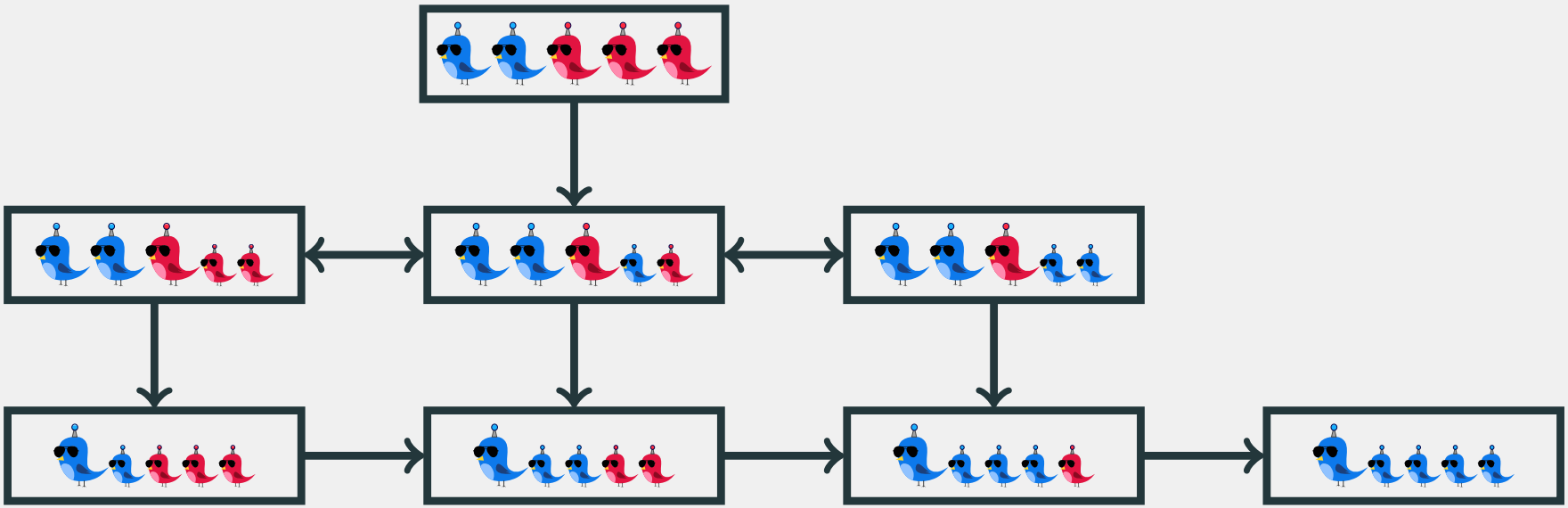
- *Configuration* $C \in Q^{\mathbb{N}}$ ...

  ... is *initial* if $C \in I^{\mathbb{N}}$

  ... is *terminal* if only silent transitions enabled

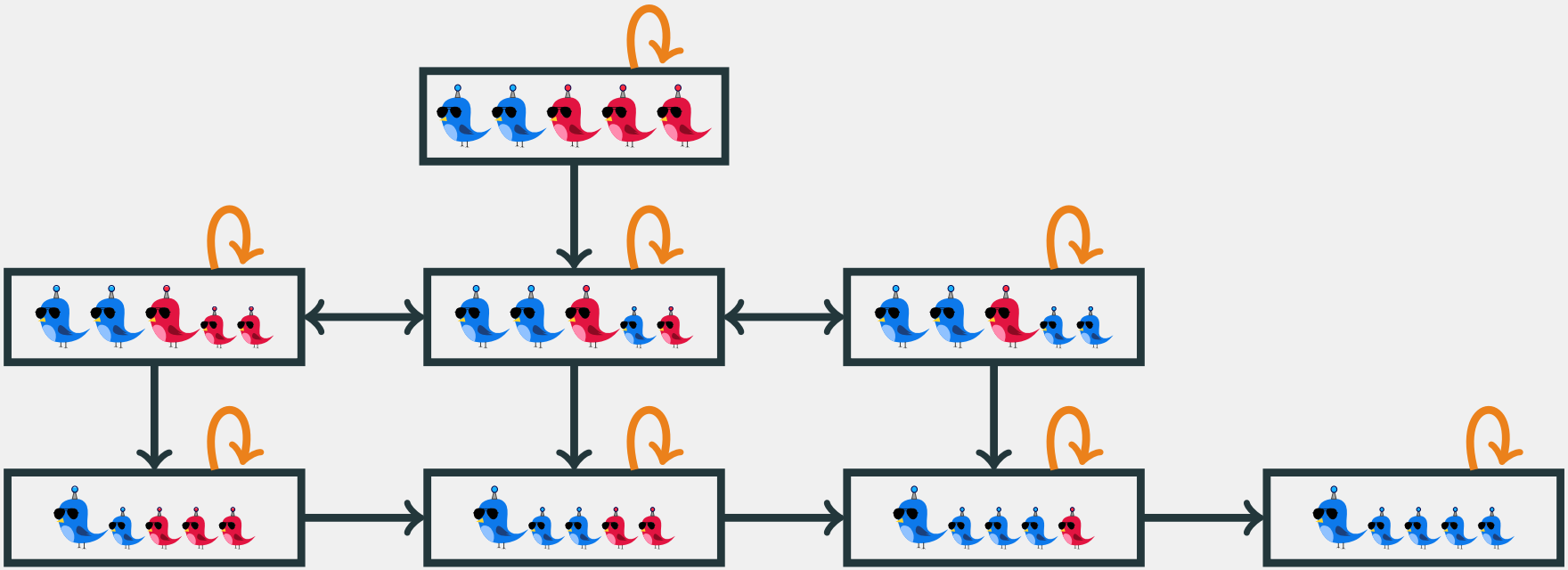
Computing with population protocols

Reachability graph from an initial configuration C :



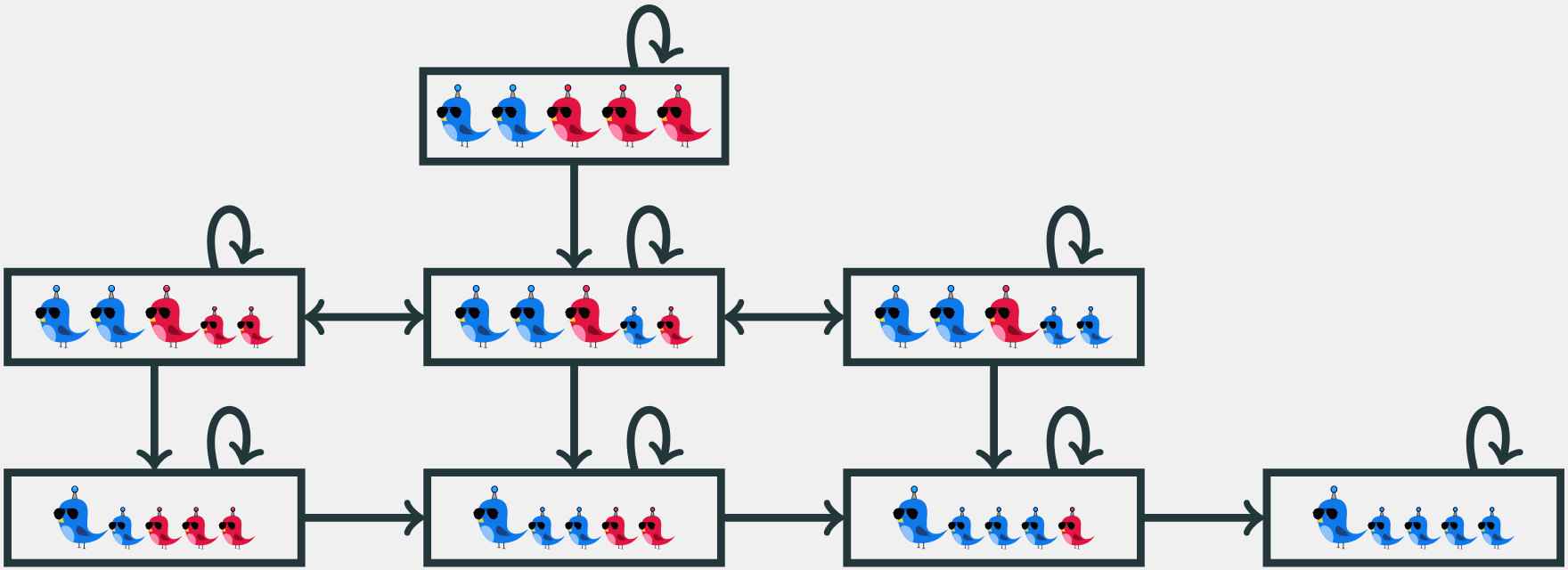
Computing with population protocols

Reachability graph from an initial configuration C :



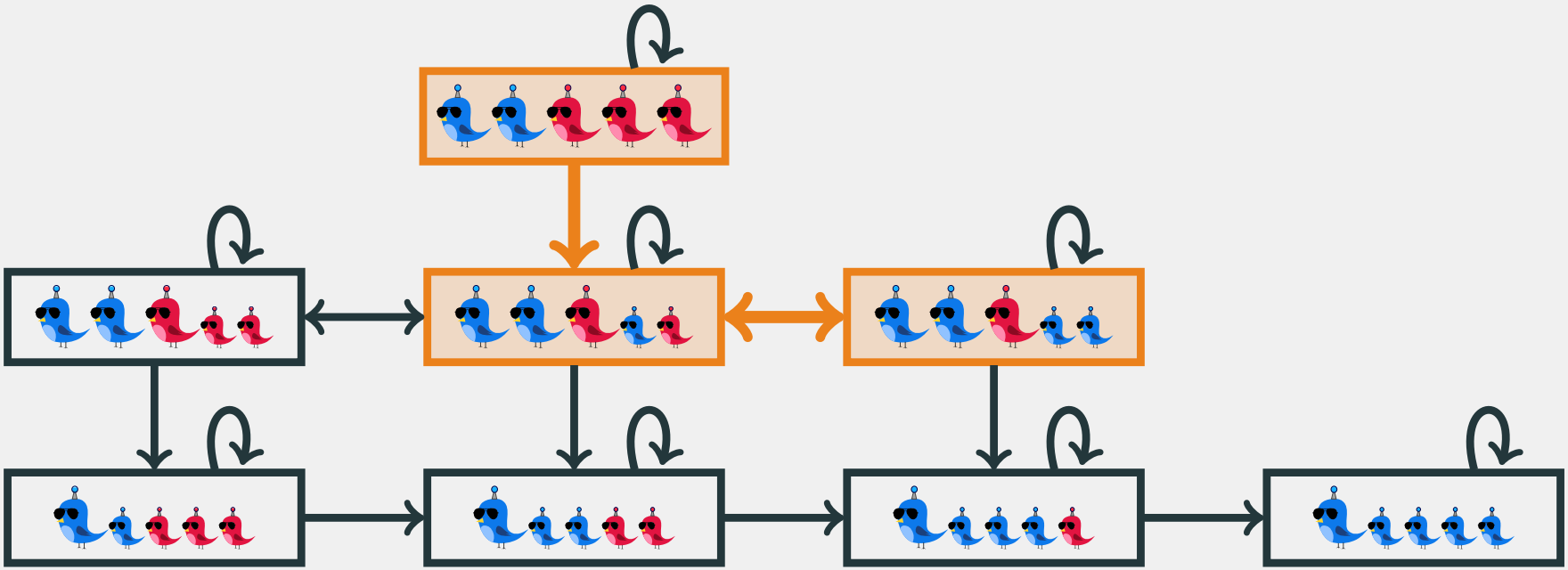
Computing with population protocols

An *execution* is an infinite path from C



Computing with population protocols

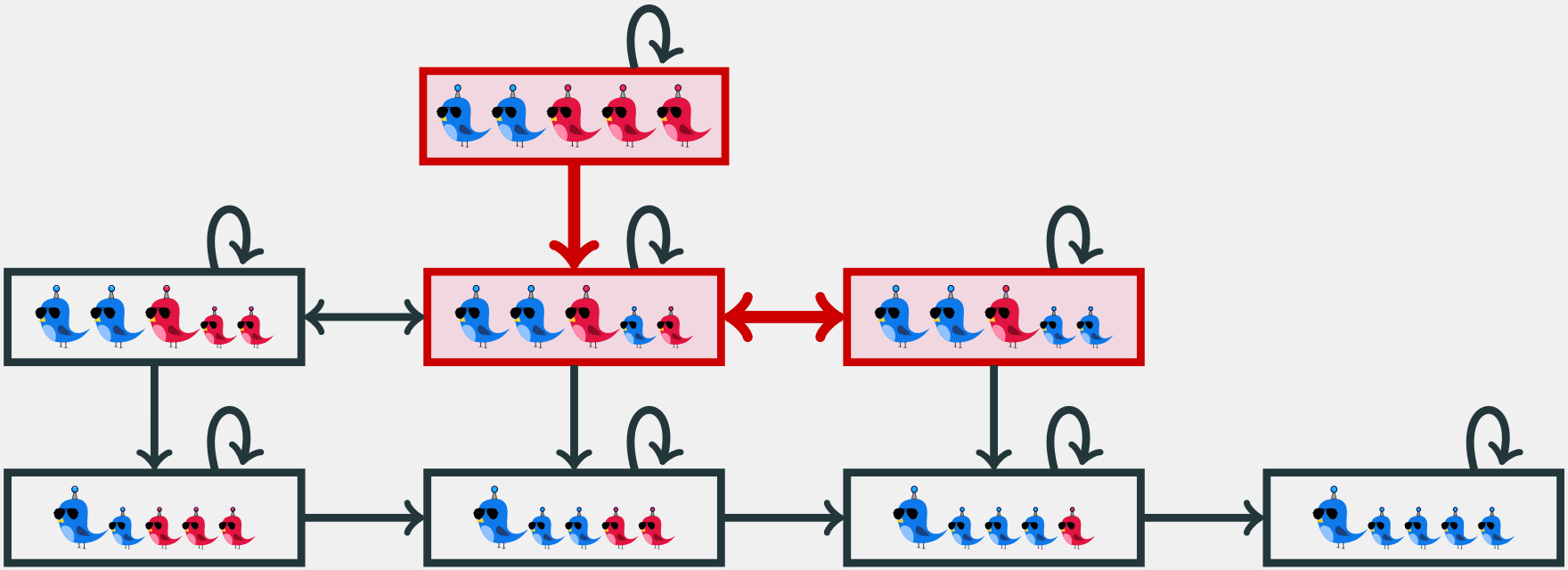
An *execution* is an infinite path from C



Computing with population protocols

Fair execution :

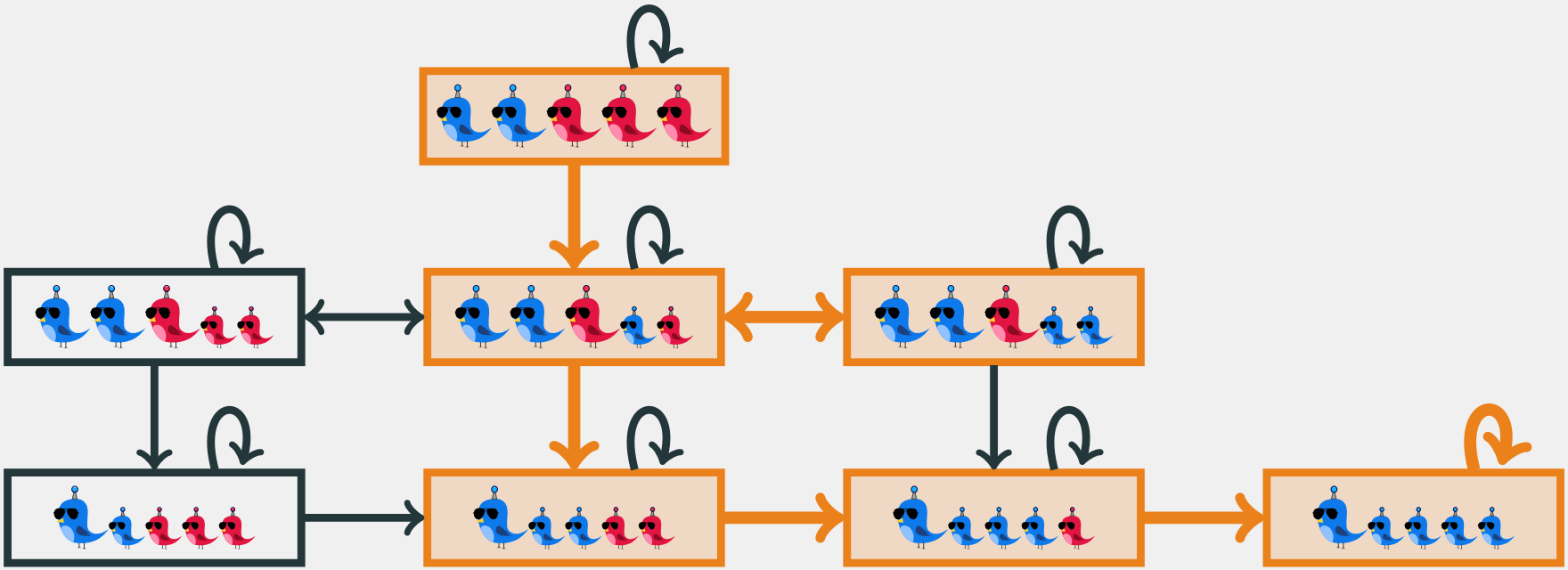
D appears inf. often, and $D \to D'$ $\implies$ D' appears inf. often



Computing with population protocols

Fair execution :

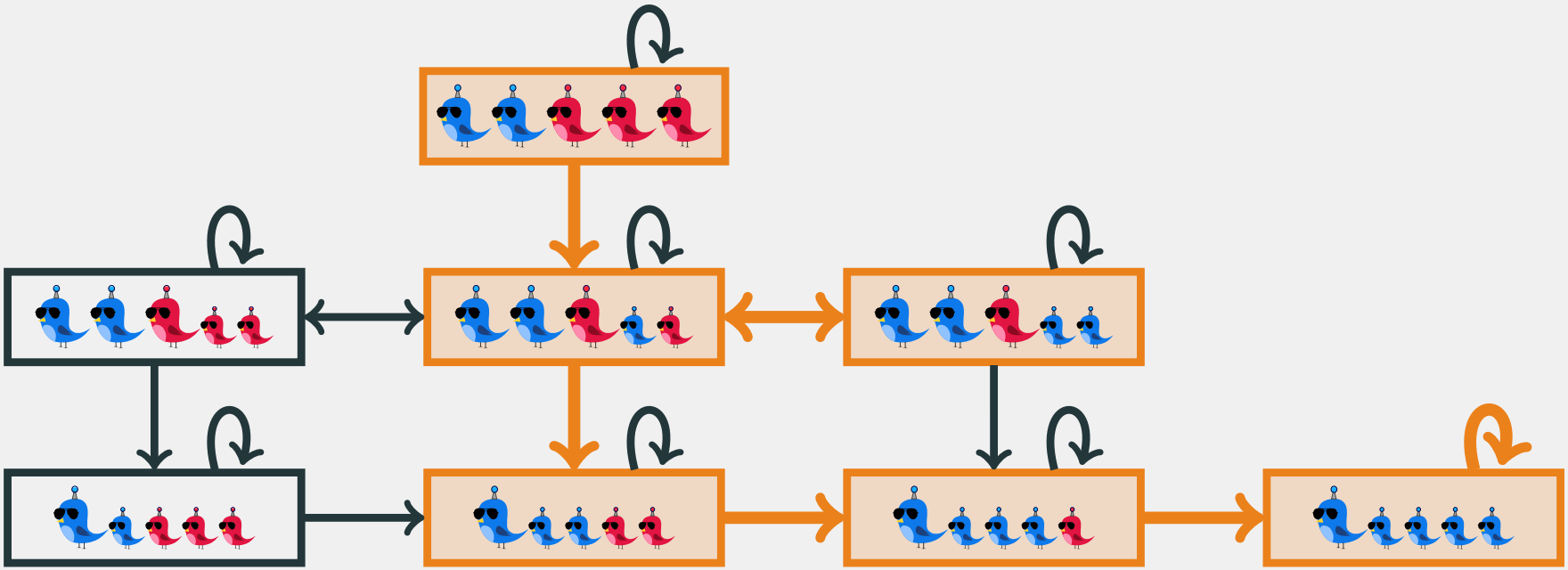
D appears inf. often, and $D \to D'$ $\implies$ D' appears inf. often



Computing with population protocols

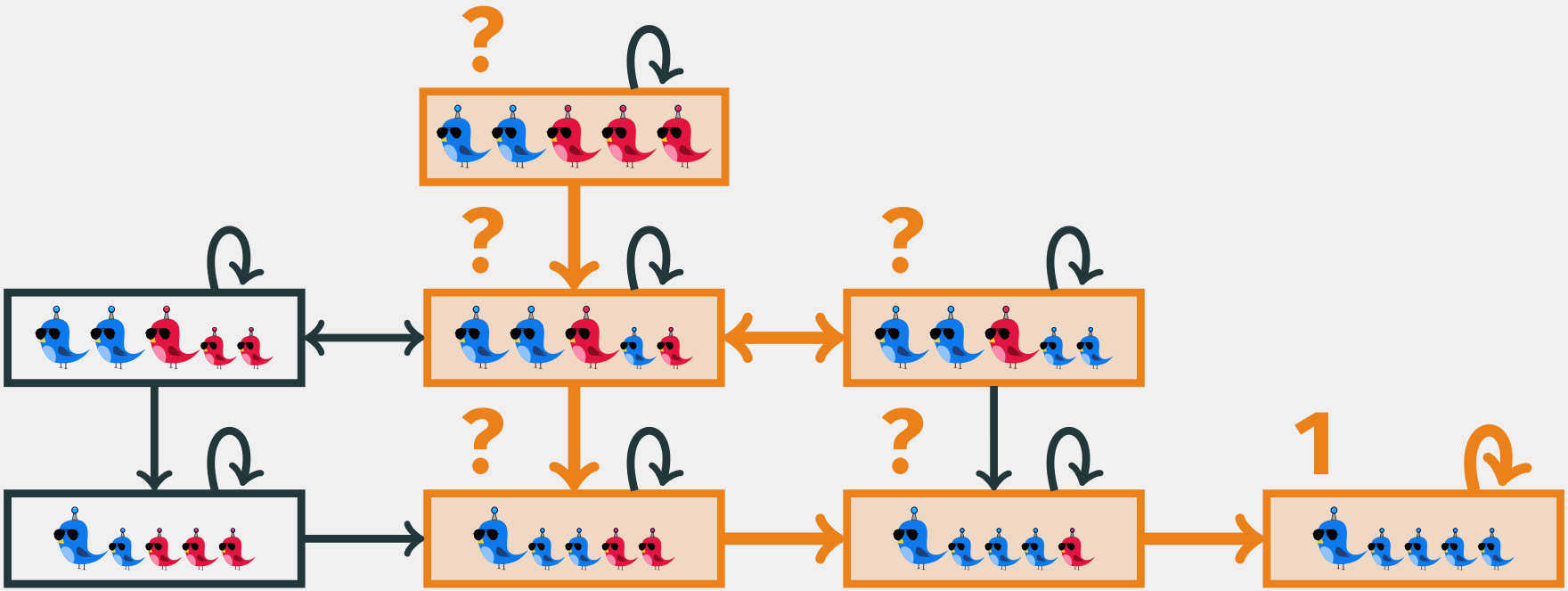
Fair execution :

ends up in a bottom strongly connected component (BSCC)



Computing with population protocols

Execution has *output b* if agents' outputs stabilize to *b*



Computing with population protocols

A protocol is *well-specified* if

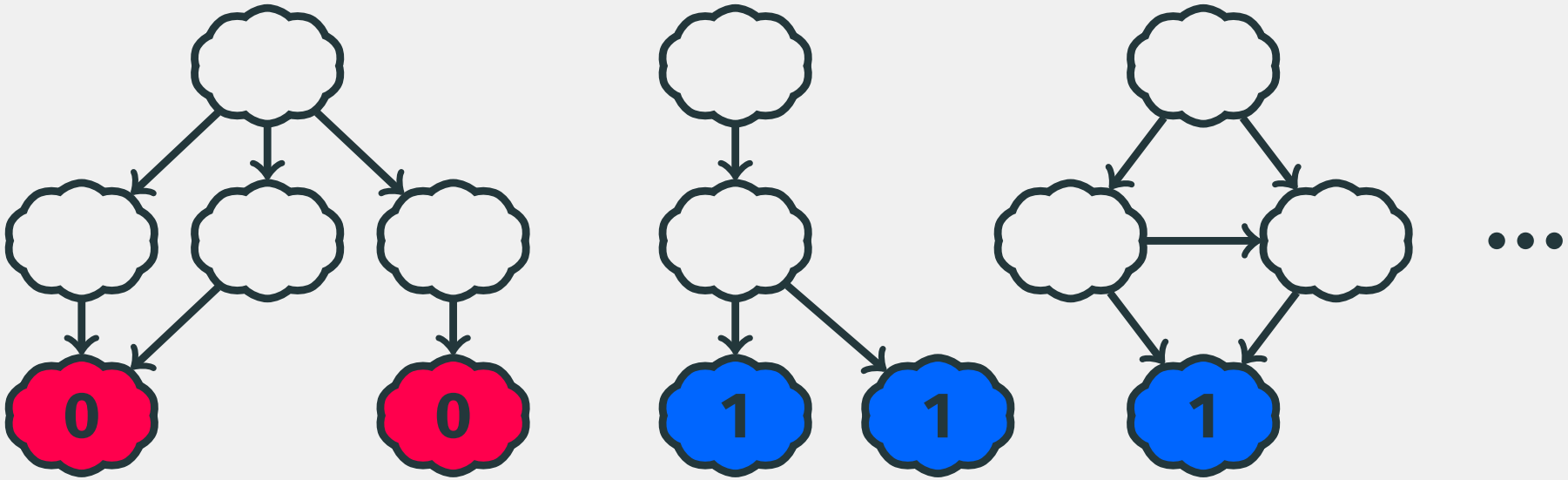
for every initial C, *all* fair executions from C have same output

Computing with population protocols

A protocol is *well-specified* if

for every initial C, *all* fair executions from C have same output

BSCCs

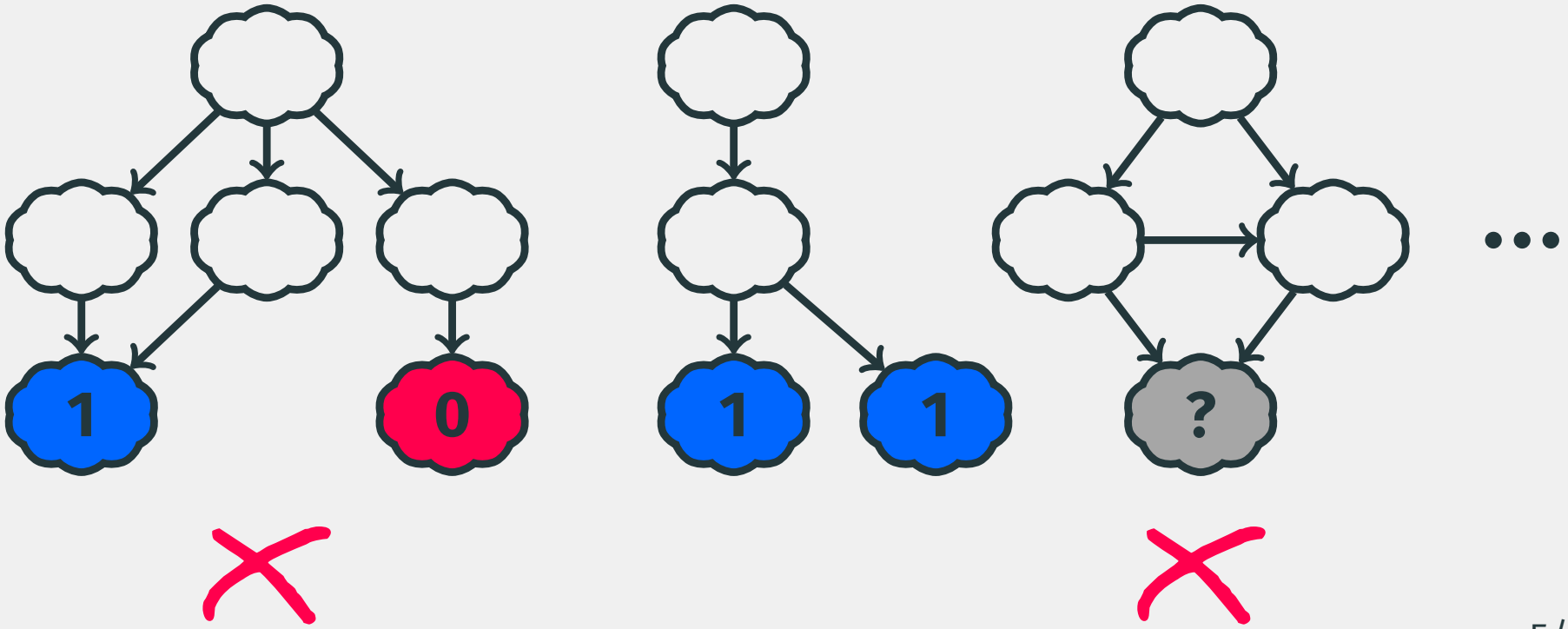


Computing with population protocols

A protocol is *ill-specified* if

- for every initial C, *all* fair executions from C have same output

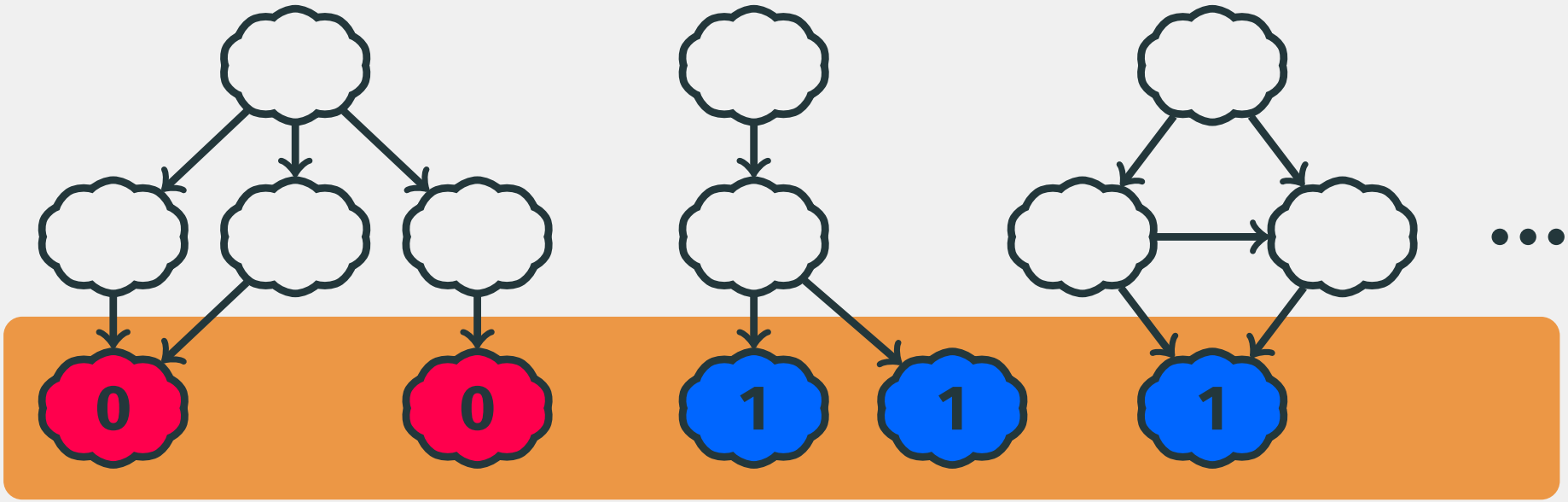
BSCCs



Computing with population protocols

A well-specified protocol computes a predicate

$$\varphi : I^{\mathbb{N}} \to \{0, 1\}$$



Computing with population protocols

Population protocols compute precisely $\mathrm{FO}(\mathbb{N}, +, <)$-definable predicates

(Angluin, Aspnes et al. PODC'04/PODC'06)

Computing with population protocols

Population protocols compute precisely FO($\mathbb{N}$, +, <)-definable predicates

(Angluin, Aspnes et al. PODC'04/PODC'06)

Some other extensions/results :

- Fast protocols (e.g. Alistarh, Gelashvili, Vojnovic PODC'15)
- Approximate protocols (e.g. Angluin, Aspnes, Eisenstat DISC'07)
- Protocols with leaders (Angluin, Aspnes, Eisenstat Dist. Comput.'08)
- Protocols with failures (Delporte-Gallet et al. DCOSS'06)
- Trustful protocols (Bournez, Lefevre, Rabie DISC'13)
- Mediated protocols (Michail, Chatzigiannakis, Spirakis TCS'11), etc.

Verifying protocols

Verifying correctness for fixed population size :

- PAT : LTL model checker with fairness

  (Sun, Liu, Song Dong and Pang CAV'09)

- `bp-ver` : graph exploration algorithms + parallelism

  (Chatzigiannakis, Michail and Spirakis SSS'10)

- Protocols to counter machines verified with `PRISM/Spin`

  (Clément, Delporte-Gallet, Fauconnier and Sighireanu ICDCS'11)

Verifying protocols

Verifying correctness for fixed population size :

- PAT : LTL model checker with fairness

  (Sun, Liu, Song Dong and Pang CAV'09)

- `bp-ver` : graph exploration algorithms + parallelism

  (Chatzigiannakis, Michail and Spirakis SSS'10)

- Protocols to counter machines verified with `PRISM`/`Spin`

  (Clément, Delporte-Gallet, Fauconnier and Sighireanu ICDCS'11)

≤ 9 states, 28 trans., pop. size 1750...

Verifying protocols

Verifying correctness for fixed population size :

- PAT : LTL model checker with fairness

  (Sun, Liu, Song Dong and Pang CAV'09)

- `bp-ver` : graph exploration algorithms + parallelism

  (Chatzigiannakis, Michail and Spirakis SSS'10)

- Protocols to counter machines verified with `PRISM/Spin`

  (Clément, Delporte-Gallet, Fauconnier and Sighireanu ICDCS'11)

Possible to verify all sizes ?

Verifying protocols

With an interactive theorem prover... (Deng and Monin TASE'09)

Possible to verify all sizes ?

Verifying protocols

With an interactive theorem prover... (Deng and Monin TASE'09)

Possible to verify all sizes ?

Automatically ?

- Well-specification and correctness are decidable
- Protocol's predicate is computable

Verifying protocols

Esparza, Ganty, Leroux and Majumdar CONCUR'15/FSTTCS'16

- Well-specification and correctness are decidable
- Protocol's predicate is computable

Petri net mutual reachability effectively Presburger-definable

(Leroux CONCUR'11)

+

Protocols compute Presburger-definable predicates

Verifying protocols

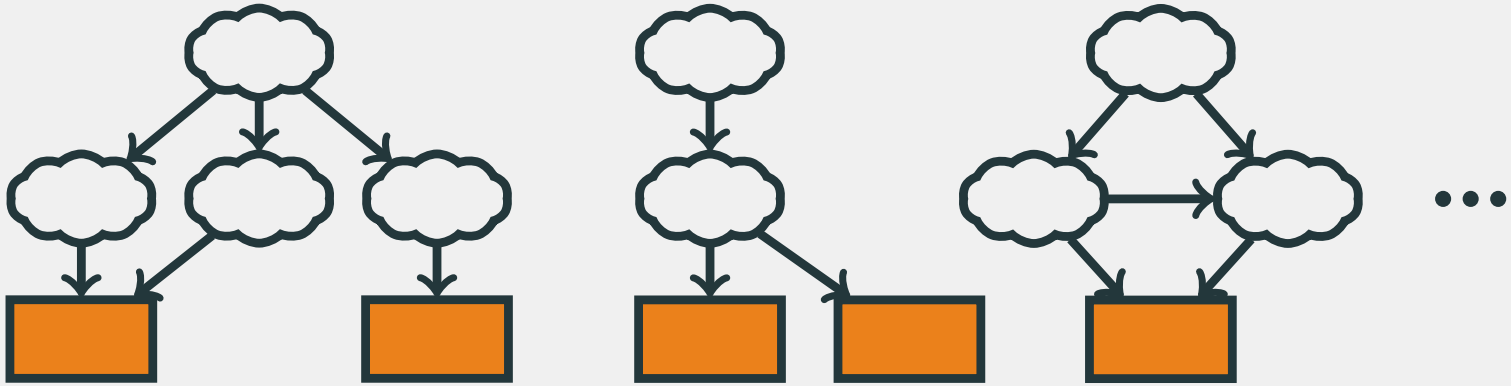
Esparza, Ganty, Leroux and Majumdar CONCUR'15/FSTTCS'16

- Well-specification and correctness are decidable
- Protocol's predicate is computable
- Qualitative probabilistic LTL model checking is decidable but quantitative variant is undecidable
- Petri net reachability reduces to all problems
- complexity between EXPSPACE and cubic-Ackermannian

Towards a verifiable class

- Most protocols are *silent* :

  fair executions reach terminal configurations



BSCCs of size 1

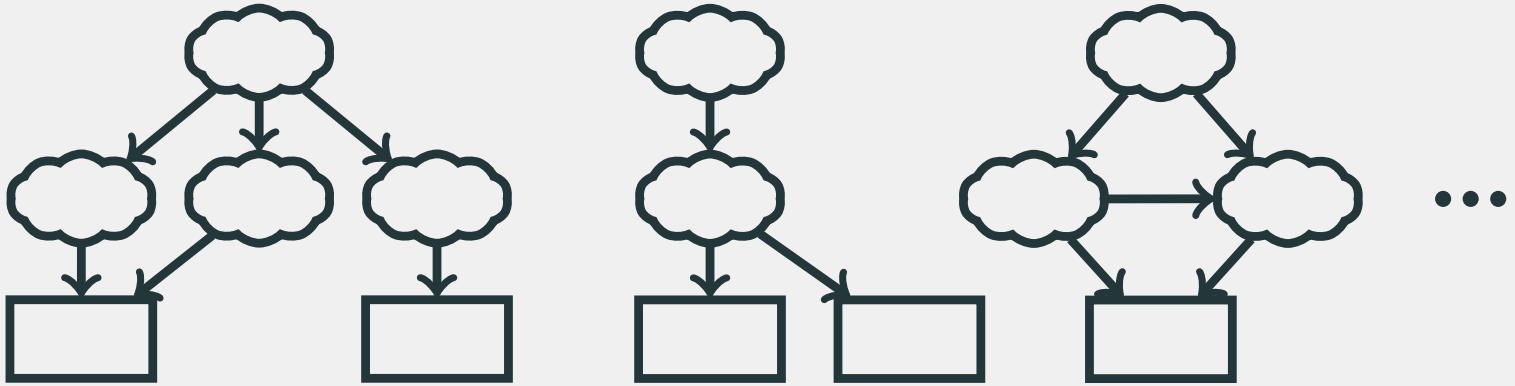
Towards a verifiable class

- Most protocols are *silent* :

  fair executions reach terminal configurations

- Much easier to test "is terminal ?" than "in BSCC ?"

- Silent protocols have same expressiveness



BSCCs of size 1

Towards a verifiable class

- Most protocols are *silent* :

  fair executions reach terminal configurations

- Much easier to test "is terminal ?" than "in BSCC ?"

- Silent protocols have same expressiveness

- Unfortunately, Petri net reachability reduces to...

  ...testing whether a protocol is silent

  ...well-specification, assuming protocols are silent

Towards a verifiable class

- Most protocols are *silent* :

  fair executions reach terminal configurations

- Much easier to test "is terminal ?" than "in BSCC ?"

- Silent protocols have same expressiveness

- Unfortunately, Petri net reachability reduces to...

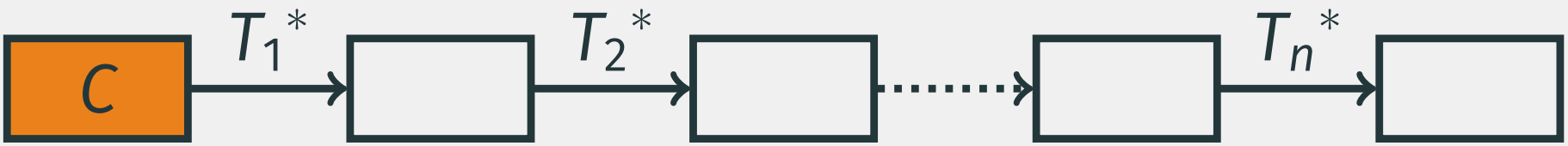
  ...testing whether a protocol is silent

  ...well-specification, assuming protocols are silent

But most silent protocols have a common design !

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

- all executions restricted to T_i are silent
- if $C \xrightarrow{T_i^*} C'$ and C is $(T_1 \cup \cdots \cup T_{i-1})$-terminal, then C' as well



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

- all executions restricted to T_i are silent
- if $C \xrightarrow{T_i^*} C'$ and C is $(T_1 \cup \cdots \cup T_{i-1})$-terminal, then C' as well

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

- all executions restricted to T_i are silent
- if $C \xrightarrow{T_i^*} C'$ and C is $(T_1 \cup \cdots \cup T_{i-1})$-terminal, then C' as well

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

- all executions restricted to T_i are silent
- if $C \xrightarrow{T_i^*} C'$ and C is $(T_1 \cup \cdots \cup T_{i-1})$-terminal, then C' as well

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

- all executions restricted to T_i are silent
- if $C \xrightarrow{T_i^*} C'$ and C is $(T_1 \cup \cdots \cup T_{i-1})$-terminal, then C' as well

Layered termination : majority protocol

T_1

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

Layered termination : majority protocol

T_1

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

Bad partition :

not all executions over T_1 are silent !

Layered termination : majority protocol

T_1

$$B\,R \to b\,r$$
$$R\,b \to R\,r$$
$$B\,r \to B\,b$$
$$b\,r \to b\,b$$

Bad partition :

not all executions over T_1 are silent !

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

Layered termination : majority protocol

T_1

$$B\,R \to b\,r$$

T_2

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

T_3

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

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

Layered termination : majority protocol

T_1

$$B\,R \to b\,r$$

T_2

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

T_3

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

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

$\#B \geq \#R$:

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

Layered termination : majority protocol

T_1 

$$B\,R \to b\,r$$

T_2

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

T_3

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

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

$\#B \geq \#R$:

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

Layered termination : majority protocol

T_1 ✗ $B\,R \to b\,r$ | T_2 ✗ $R\,b \to R\,r$ | T_3 $B\,r \to B\,b$, $b\,r \to b\,b$

$\#B \geq \#R$:

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

Layered termination : majority protocol

T_1 ✗ $B\,R \to b\,r$

T_2 ✗ $R\,b \to R\,r$

T_3 ✗ $B\,r \to B\,b$, $b\,r \to b\,b$

$\#B \geq \#R$:

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

Layered termination : majority protocol

T_1

$$B\,R \to b\,r$$

T_2

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

T_3

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

$\#B \geq \#R$:

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

$\#R > \#B$:

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

Layered termination : majority protocol

T_1 ✗

$$B\,R \to b\,r$$

T_2

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

T_3

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

$\#B \geq \#R$:

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

$\#R > \#B$:

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

Layered termination : majority protocol

T_1 ✗ $B\,R \to b\,r$ | T_2 ✗ $R\,b \to R\,r$ | T_3 $B\,r \to B\,b$, $b\,r \to b\,b$

$\#B \geq \#R$:

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

$\#R > \#B$:

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

Layered termination : majority protocol

T_1 ✗ $B\,R \to b\,r$ | T_2 ✗ $R\,b \to R\,r$ | T_3 ✗ $B\,r \to B\,b$, $b\,r \to b\,b$

$\#B \geq \#R$:

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

$\#R > \#B$:

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

Strongly silent protocols

A protocol is *strongly silent* if it satisfies layered termination

Theorem **PODC'17**

- Every strongly silent protocol is silent
- Strongly silent protocols are as expressive as general protocols
- Deciding whether a protocol is strongly silent $\in$ NP

Strongly silent protocols

Theorem **PODC'17**

Every strongly silent protocol is silent

Proof sketch

Layered termination $\Longrightarrow$ every configuration can reach a terminal configuration

$\Longrightarrow$ BSCCs are of size 1

$\Longrightarrow$ fair executions are silent

Strongly silent protocols

Theorem **PODC'17**

Strongly silent protocols as expressive as general protocols

Proof sketch

- Protocols for

$$a_1 x_1 + \ldots + a_n x_n \geq b$$

$$a_1 x_1 + \ldots + a_n x_n \equiv b \pmod{m}$$

  have layered termination strategies

- Conjunction and negation preserve layered termination

Strongly silent protocols

Theorem **PODC'17**

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 in poly. time :

Strongly silent protocols

Theorem **PODC'17**

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 in poly. time :

1) all executions restricted to T_i are silent

Strongly silent protocols

Theorem **PODC'17**

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 in poly. time :

1) all executions restricted to T_i are silent

Test for Petri net structural termination with

$$\neg\, \exists \boldsymbol{x} \in \mathbb{Q}^Q \quad \text{Incid} \cdot \boldsymbol{x} \geq \mathbf{0} \wedge \boldsymbol{x} > \mathbf{0}$$

Strongly silent protocols

Theorem **PODC'17**

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 in poly. time :

2) if $C \xrightarrow{T_i^*} C'$ and C is $(T_1 \cup \cdots \cup T_{i-1})$-terminal, then C' as well

Strongly silent protocols

Theorem **PODC'17**

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 in poly. time :

2) if $C \xrightarrow{T_i^*} C'$ and C is $(T_1 \cup \cdots \cup T_{i-1})$-terminal, then C' as well

Test $\forall t \in T_i \quad \forall$ non silent $u \in T_1 \cup \cdots \cup T_{i-1}$

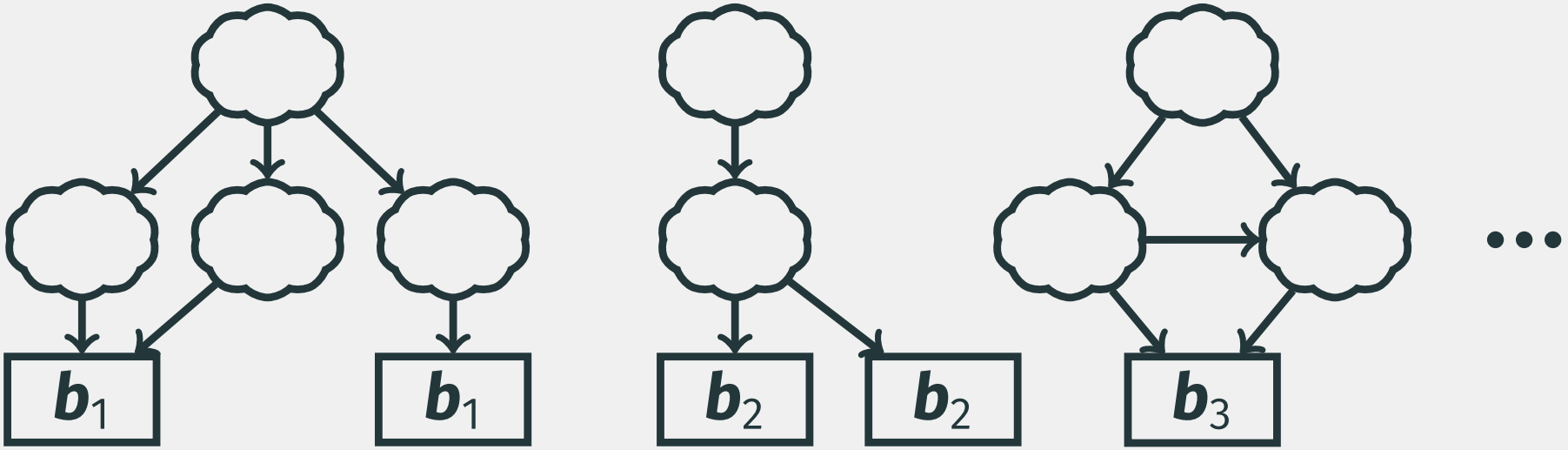
$\exists$ non silent $u' \in T_1 \cup \cdots \cup T_{i-1}$ s.t.

$$\mathrm{pre}(u') \leq \mathrm{pre}(t) + (\mathrm{pre}(u) \ominus \mathrm{post}(t))$$

Silent protocols : verifying well-specification

It suffices to test consensus :

$$\forall \text{ init. } C \quad \exists b \quad \forall \text{ terminal } C' \qquad C \xrightarrow{*} C' \implies O(C') = b$$

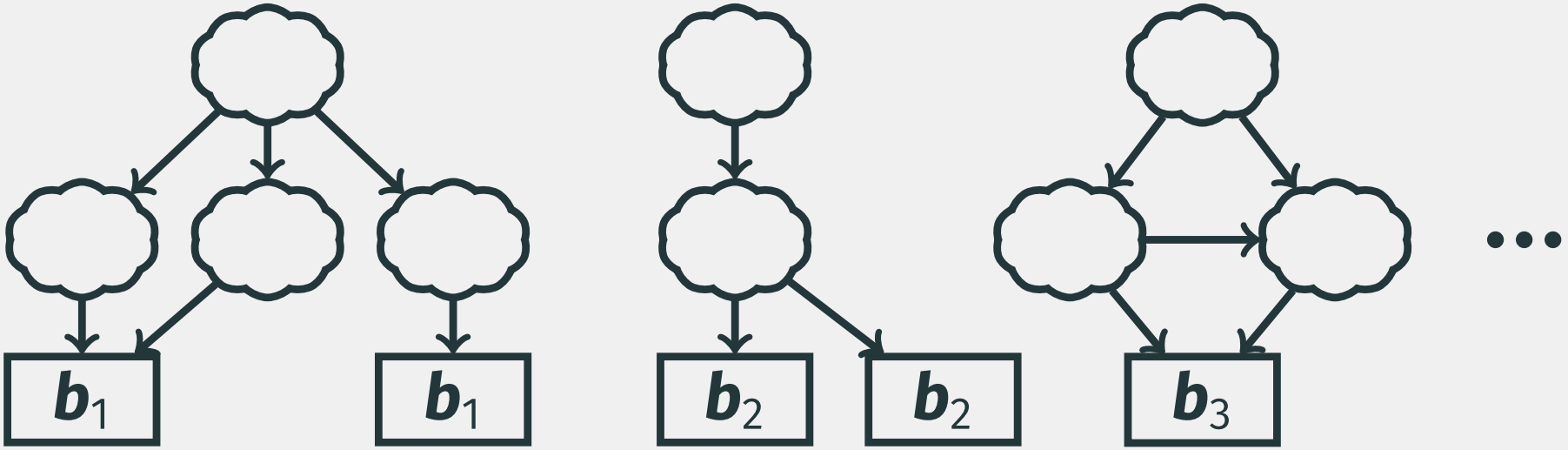


Silent protocols : verifying well-specification

It suffices to test consensus :

$$\forall \text{ init. } C \quad \exists b \quad \forall \text{ terminal } C' \qquad C \xrightarrow{*} C' \implies O(C') = b$$

reachability...

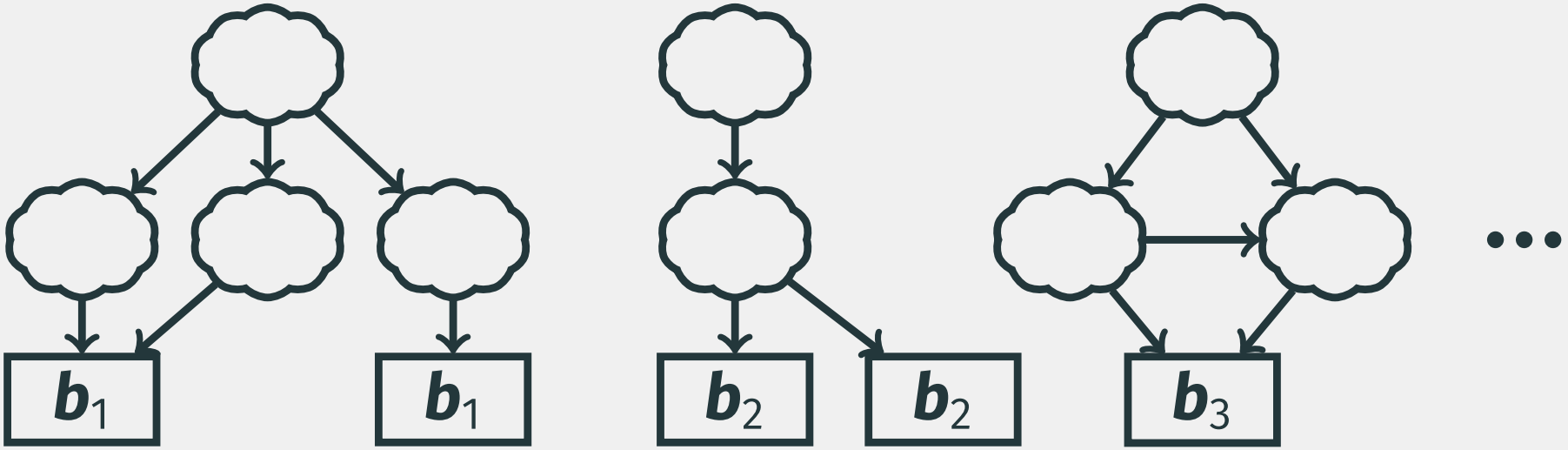


Silent protocols : verifying well-specification

It suffices to test consensus :

$$\forall \text{ init. } C \quad \exists b \quad \forall \text{ terminal } C' \qquad C \xrightarrow{*} C' \implies O(C') = b$$

over-approximation!



Silent protocols : verifying well-specification

It suffices to test strong consensus :

$$\forall \text{ init. } C \quad \exists b \quad \forall \text{ terminal } C' \qquad C \xrightarrow{*} C' \implies O(C') = b$$

- Reachable when ignoring guards
- Non empty traps are never emptied
- Empty siphons remain empty

(e.g. Esparza, Ledesma-Garza, Majumdar, Meyer and Nikšić CAV'14)

Silent protocols : verifying well-specification

It suffices to test strong consensus :

$$\forall \text{ init. } C \quad \exists b \quad \forall \text{ terminal } C' \qquad C \xrightarrow{*} C' \implies O(C') = b$$

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

$$\mathbf{B\ r} \to \mathbf{B\ b}$$

$$\mathbf{R\ b} \to \mathbf{R\ r}$$

$$\mathbf{b\ r} \to \mathbf{b\ b}$$

Silent protocols : verifying well-specification

It suffices to test strong consensus :

$$\forall \text{ init. } C \quad \exists b \quad \forall \text{ terminal } C' \qquad C \xrightarrow{*} C' \implies O(C') = b$$

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

$$\{\boldsymbol{B}, \boldsymbol{R}\} \to \{\boldsymbol{b}, \boldsymbol{r}\} \xrightarrow{\mathbb{Z}} \{\boldsymbol{r}, \boldsymbol{r}\}$$

Silent protocols : verifying well-specification

It suffices to test strong consensus :

$$\forall \text{ init. } C \quad \exists b \quad \forall \text{ terminal } C' \qquad C \xrightarrow{*} C' \implies O(C') = b$$

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

$$\{B,\ R\} \to \{b,\ r\} \not\to \{r,\ r\}$$

Trap $\{R, b\}$ cannot be emptied !

Silent protocols : verifying well-specification

It suffices to test strong consensus :

$$\forall \text{ init. } C \quad \exists b \quad \forall \text{ terminal } C' \qquad C \xrightarrow{*} C' \implies O(C') = b$$

Theorem **PODC'17**

- Testing strong consensus $\in$ coNP
- Strongly silent protocols with strong consensus have same expressiveness as general protocols

Experimental results

- `peregrine` : Haskell + SMT solver Z3

  `gitlab.lrz.de/i7/peregrine`

- Tests whether protocol is strongly silent
  - Structural termination constraints + Farkas' lemma
  - Tries ordered partitions of size $1, 2, \ldots, |T|$

- Tests well-specification
  - Checks if two conflicting terminal config. are $\mathbb{Z}$-reachable
  - If so, adds traps/siphons constraints until `unsat`

Experimental results

- `peregrine` : Haskell + SMT solver Z3

  `gitlab.lrz.de/i7/peregrine`

- Tests whether protocol is strongly silent
  - Structural termination constraints + Farkas' lemma
  - Tries ordered partitions of size $1, 2, \ldots, |T|$
- Tests well-specification and correctness!
  - Checks if two conflicting terminal config. are $\mathbb{Z}$-reachable
  - If so, adds traps/siphons constraints until `unsat`

Experimental results

Majority

$\#B \geq \#R$?

$\lvert Q\rvert$	$\lvert T\rvert$	time (secs.)
4	4	0.1

Broadcast

$x_1 \vee \ldots \vee x_n = 1$?

$\lvert Q\rvert$	$\lvert T\rvert$	time (secs.)
2	1	0.1

Experimental results

Majority

$\#B \geq \#R$?

$\lvert Q\rvert$	$\lvert T\rvert$	time (secs.)
4	4	0.1

Broadcast

$x_1 \vee \ldots \vee x_n = 1$?

$\lvert Q\rvert$	$\lvert T\rvert$	time (secs.)
2	1	0.1

With PRISM : 1 hour
for single config. of size 1000

(250 000 reachable configs.)

Experimental results

Majority

$\#B \geq \#R$?

$\lvert Q\rvert$	$\lvert T\rvert$	time (secs.)
4	4	0.1

Broadcast

$x_1 \vee \ldots \vee x_n = 1$?

$\lvert Q\rvert$	$\lvert T\rvert$	time (secs.)
2	1	0.1

With PRISM : ? years
for all config. of size 1000

(83 millions reachable configs.)

Experimental results

Threshold

$a_1x_1 + \ldots + a_nx_n \geq b$?

max coeff.	$\lvert Q\rvert$	$\lvert T\rvert$	time (secs.)
3	28	288	8.0
4	36	478	26.5
5	44	716	97.6
6	52	1002	243.4
7	60	1336	565.0
8	68	1718	1019.7
9	76	2148	2375.9
10	84	2626	timeout

Modulo

$a_1x_1 + \ldots + a_nx_n \equiv b \pmod{m}$?

m	$\lvert Q\rvert$	$\lvert T\rvert$	time (secs.)
10	12	65	0.4
20	22	230	2.8
30	32	495	15.9
40	42	860	79.3
50	52	1325	440.3
60	62	1890	3055.4
70	72	2555	3176.5
80	82	3320	timeout

Experimental results

Flock of birds (variant 1)

$\#B \geq n$?

n	$\lvert Q\rvert$	$\lvert T\rvert$	time (secs.)
20	21	210	1.5
25	26	325	3.3
30	31	465	7.7
35	36	630	20.8
40	41	820	106.9
45	46	1035	295.6
50	51	1275	181.6
55	56	1540	timeout

Flock of birds (variant 2)

$\#B \geq n$?

n	$\lvert Q\rvert$	$\lvert T\rvert$	time (secs.)
50	51	99	11.8
100	101	199	44.8
150	151	299	369.1
200	201	399	778.8
250	251	499	1554.2
300	301	599	2782.5
325	326	649	3470.8
350	351	699	timeout

Conclusion : summary

- New subclass encompassing most existing protocols, with same expressiveness power, and verifiable
- Our approach is automatic and entirely parametric. Other automatic approaches consider fixed size populations !
- New tool that can verify existing protocols

Conclusion : future work

- Verifying non silent protocols
- Strengthening strong consensus
- Diagnosis : return explanations when protocol not strongly silent/well-specified
- Verification of approximate protocols
- LTL model checking ?

Thank you !