Fast and Succinct Population Protocols for Presburger Arithmetic

Philipp Czerner, Javier Esparza, Roland Guttenberg, Martin Helfrich

Technical University of Munich

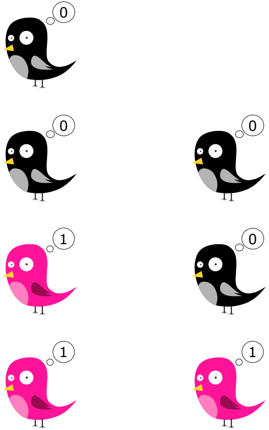
September 12 2022

European Research Council
Established by the European Commission

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 787367

Introduction to Population Protocols

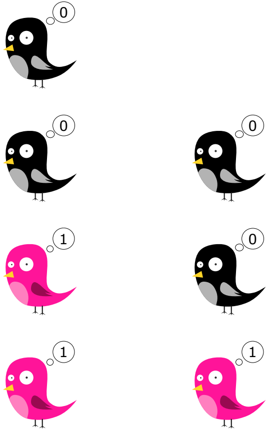
Population Protocols = model of computation



Introduction to Population Protocols

Population Protocols = model of computation

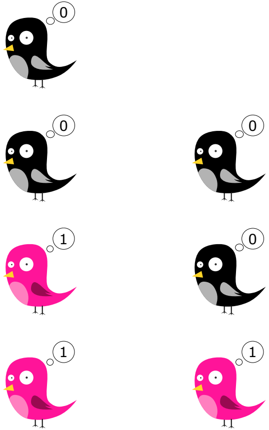
- anonymous finite-state agents (birds),



Introduction to Population Protocols

Population Protocols = model of computation

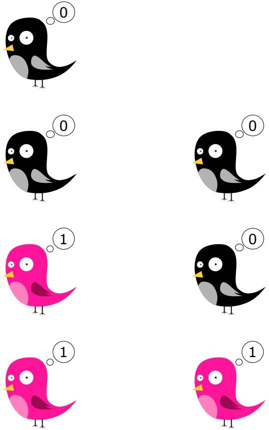
- anonymous finite-state agents (birds),
- decide property of initial configuration,



Introduction to Population Protocols

Population Protocols = model of computation

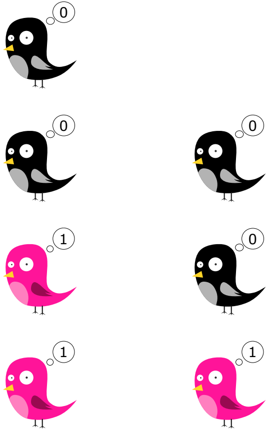
- anonymous finite-state agents (birds),
- decide property of initial configuration,
- inputs = counts of initial states,



Introduction to Population Protocols

Population Protocols = model of computation

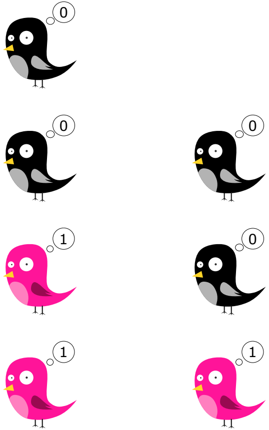
- anonymous finite-state agents (birds),
- decide property of initial configuration,
- inputs = counts of initial states,
- only pairwise interactions,



Introduction to Population Protocols

Population Protocols = model of computation

- anonymous finite-state agents (birds),
- decide property of initial configuration,
- inputs = counts of initial states,
- only pairwise interactions,
- output by stable consensus.

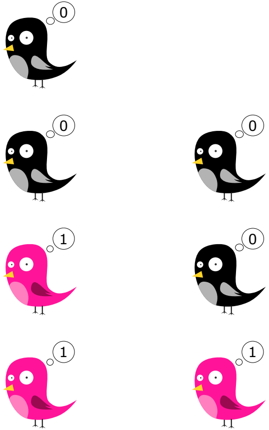


Introduction to Population Protocols

Population Protocols = model of computation

- anonymous finite-state agents (birds),
- decide property of initial configuration,
- inputs = counts of initial states,
- only pairwise interactions,
- output by stable consensus.

Example: Decide #pink birds ≥ 3.

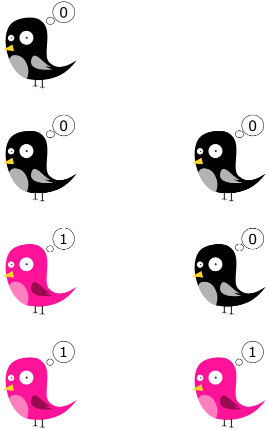


Introduction to Population Protocols

Population Protocols = model of computation

- anonymous finite-state agents (birds),
- decide property of initial configuration,
- inputs = counts of initial states,
- only pairwise interactions,
- output by stable consensus.

Example: Decide #pink birds ≥ 3.
States $Q = \{0, 1, 2, 3\}$.



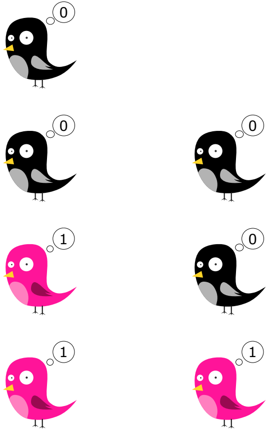
Introduction to Population Protocols

Population Protocols = model of computation

- anonymous finite-state agents (birds),
- decide property of initial configuration,
- inputs = counts of initial states,
- only pairwise interactions,
- output by stable consensus.

Example: Decide #pink birds ≥ 3.
States $Q = \{0, 1, 2, 3\}$.

Colors and numbers encode the same.



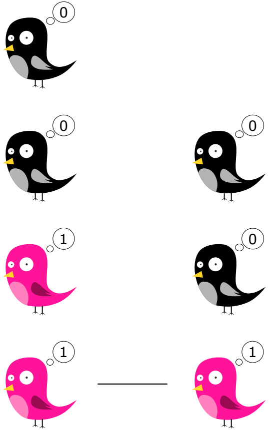
Introduction to Population Protocols

Population Protocols = model of computation

- anonymous finite-state agents (birds),
- decide property of initial configuration,
- inputs = counts of initial states,
- only pairwise interactions,
- output by stable consensus.

Example: Decide #pink birds ≥ 3.
States $Q = \{0, 1, 2, 3\}$.

Colors and numbers encode the same.



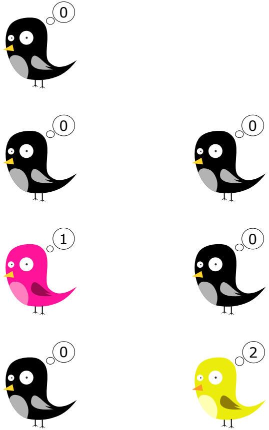
Introduction to Population Protocols

Population Protocols = model of computation

- anonymous finite-state agents (birds),
- decide property of initial configuration,
- inputs = counts of initial states,
- only pairwise interactions,
- output by stable consensus.

Example: Decide #pink birds ≥ 3.
States $Q = \{0, 1, 2, 3\}$.

Colors and numbers encode the same.



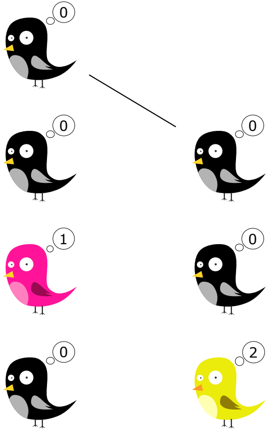
Introduction to Population Protocols

Population Protocols = model of computation

- anonymous finite-state agents (birds),
- decide property of initial configuration,
- inputs = counts of initial states,
- only pairwise interactions,
- output by stable consensus.

Example: Decide #pink birds ≥ 3.
States $Q = \{0, 1, 2, 3\}$.

Colors and numbers encode the same.



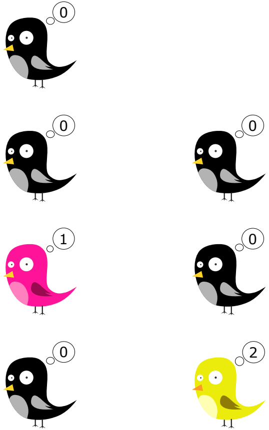
Introduction to Population Protocols

Population Protocols = model of computation

- anonymous finite-state agents (birds),
- decide property of initial configuration,
- inputs = counts of initial states,
- only pairwise interactions,
- output by stable consensus.

Example: Decide #pink birds ≥ 3.
States $Q = \{0, 1, 2, 3\}$.

Colors and numbers encode the same.



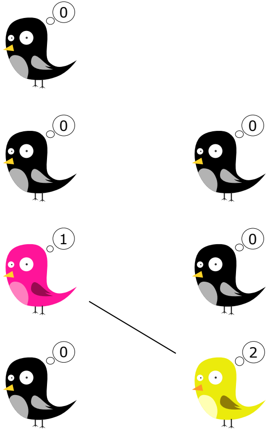
Introduction to Population Protocols

Population Protocols = model of computation

- anonymous finite-state agents (birds),
- decide property of initial configuration,
- inputs = counts of initial states,
- only pairwise interactions,
- output by stable consensus.

Example: Decide #pink birds ≥ 3.
States $Q = \{0, 1, 2, 3\}$.

Colors and numbers encode the same.



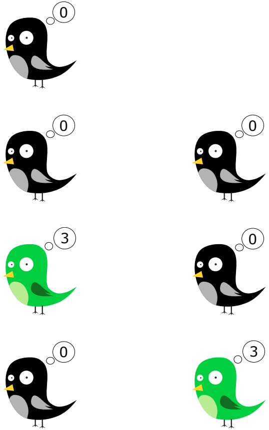
Introduction to Population Protocols

Population Protocols = model of computation

- anonymous finite-state agents (birds),
- decide property of initial configuration,
- inputs = counts of initial states,
- only pairwise interactions,
- output by stable consensus.

Example: Decide #pink birds ≥ 3.
States $Q = \{0, 1, 2, 3\}$.

Colors and numbers encode the same.



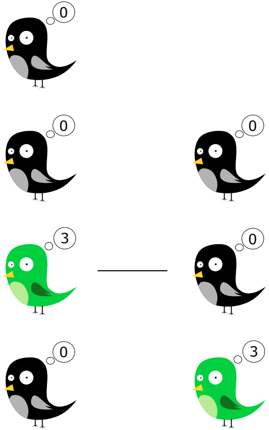
Introduction to Population Protocols

Population Protocols = model of computation

- anonymous finite-state agents (birds),
- decide property of initial configuration,
- inputs = counts of initial states,
- only pairwise interactions,
- output by stable consensus.

Example: Decide #pink birds ≥ 3.
States $Q = \{0, 1, 2, 3\}$.

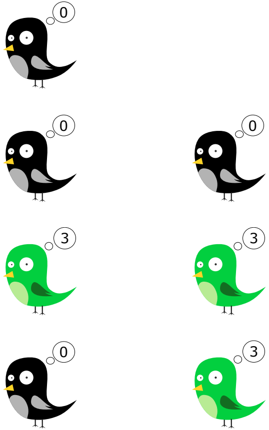
Colors and numbers encode the same.



Introduction to Population Protocols

Population Protocols = model of computation

- anonymous finite-state agents (birds),
- decide property of initial configuration,
- inputs = counts of initial states,
- only pairwise interactions,
- output by stable consensus.



Example: Decide #pink birds ≥ 3.
States $Q = \{0, 1, 2, 3\}$.

Colors and numbers encode the same.

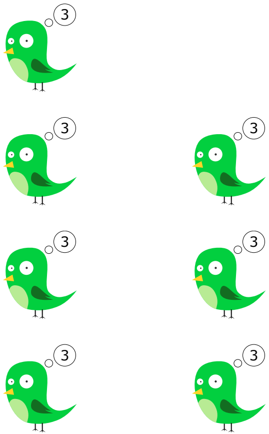
Introduction to Population Protocols

Population Protocols = model of computation

- anonymous finite-state agents (birds),
- decide property of initial configuration,
- inputs = counts of initial states,
- only pairwise interactions,
- output by stable consensus.

Example: Decide #pink birds ≥ 3.
States $Q = \{0, 1, 2, 3\}$.

Colors and numbers encode the same.



Introduction to Population Protocols

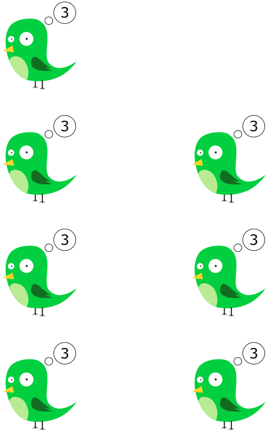
Population Protocols = model of computation

- anonymous finite-state agents (birds),
- decide property of initial configuration,
- inputs = counts of initial states,
- only pairwise interactions,
- output by stable consensus.

Example: Decide #pink birds ≥ 3.
States $Q = \{0, 1, 2, 3\}$.

Colors and numbers encode the same.

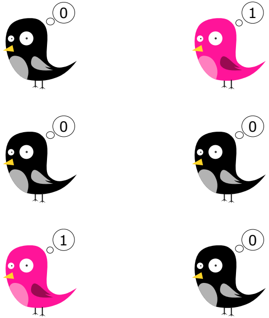
Protocol has to be correct for
all initial configurations.



Introduction to Population Protocols

Population Protocols = model of computation

- anonymous finite-state agents (birds),
- decide property of initial configuration,
- inputs = counts of initial states,
- only pairwise interactions,
- output by stable consensus.



Example: Decide #pink birds ≥ 3.
States $Q = \{0, 1, 2, 3\}$.

Colors and numbers encode the same.

Protocol has to be correct for
all initial configurations.

Applications of Population Protocols

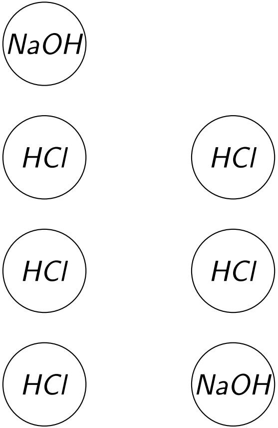
Applications of Population Protocols

Chemical Reaction Networks.

Applications of Population Protocols

Chemical Reaction Networks.

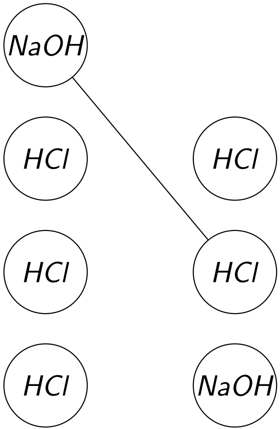
$HCl + NaOH \rightarrow NaCl + H_2O$



Applications of Population Protocols

Chemical Reaction Networks.

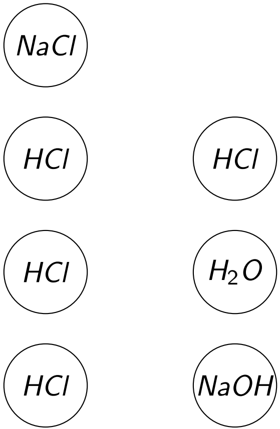
$$HCl + NaOH \rightarrow NaCl + H_2O$$



Chemical Reaction Networks.

$HCl + NaOH \rightarrow NaCl + H_2O$

State complexity: # species.



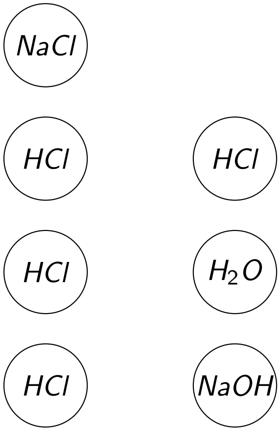
Applications of Population Protocols

Chemical Reaction Networks.

$HCl + NaOH \rightarrow NaCl + H_2O$

State complexity: # species.

Accordingly for protocols:
$|\{0, 1, 2, 3\}| = 4$.



Applications of Population Protocols

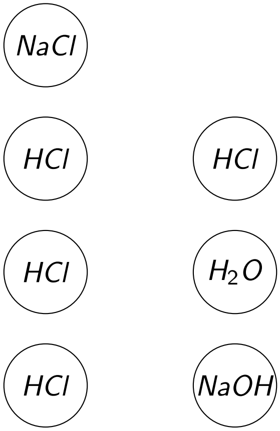
Chemical Reaction Networks.

$HCl + NaOH \to NaCl + H_2O$

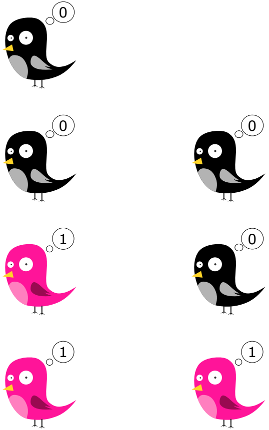
State complexity: # species.

Accordingly for protocols:
$|\{0, 1, 2, 3\}| = 4$.

Mobile sensor networks, ...

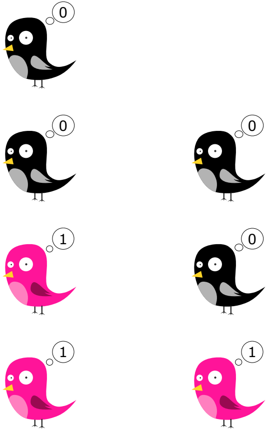


Speed of Population Protocols



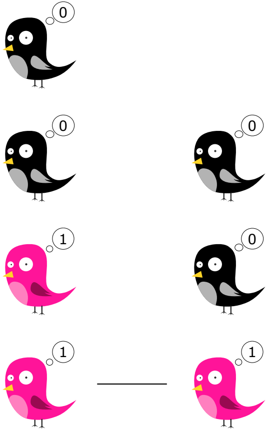
Speed of Population Protocols

In every step: choose pair of agents uniformly at random.



Speed of Population Protocols

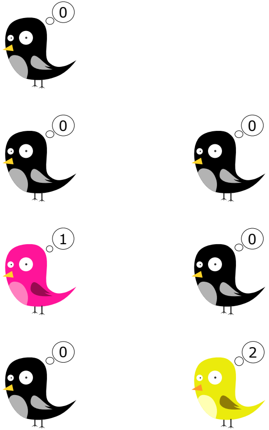
In every step: choose pair of agents uniformly at random.
These agents interact in this step.



Speed of Population Protocols

In every step: choose pair of agents uniformly at random.
These agents interact in this step.

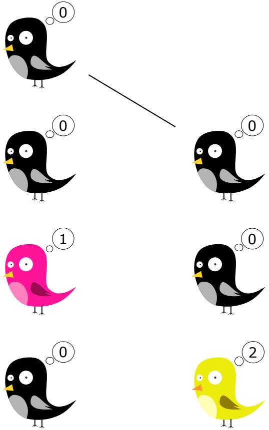
Speed = expected number of steps until reaching stable consensus.



Speed of Population Protocols

In every step: choose pair of agents uniformly at random.
These agents interact in this step.

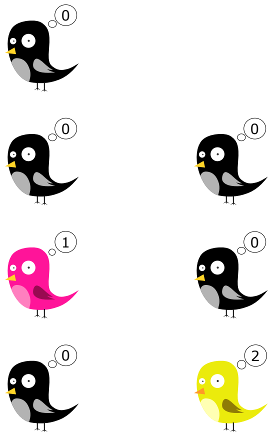
Speed = expected number of steps until reaching stable consensus.



Speed of Population Protocols

In every step: choose pair of agents uniformly at random.
These agents interact in this step.

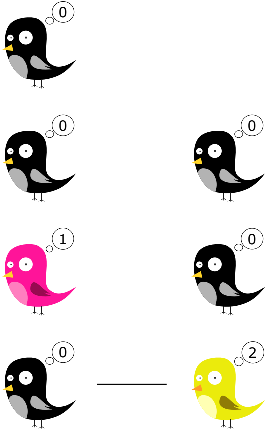
Speed = expected number of steps until reaching stable consensus.



Speed of Population Protocols

In every step: choose pair of agents uniformly at random.
These agents interact in this step.

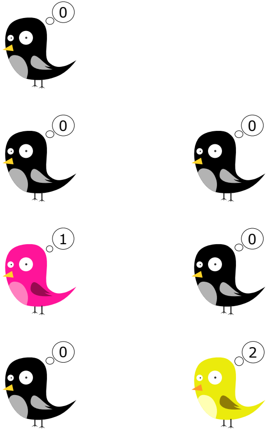
Speed = expected number of steps until reaching stable consensus.



Speed of Population Protocols

In every step: choose pair of agents uniformly at random.
These agents interact in this step.

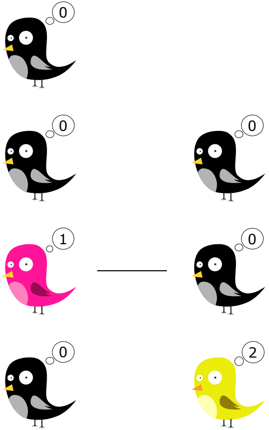
Speed = expected number of steps until reaching stable consensus.



Speed of Population Protocols

In every step: choose pair of agents uniformly at random.
These agents interact in this step.

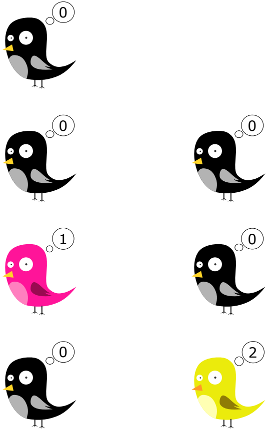
Speed = expected number of steps until reaching stable consensus.



Speed of Population Protocols

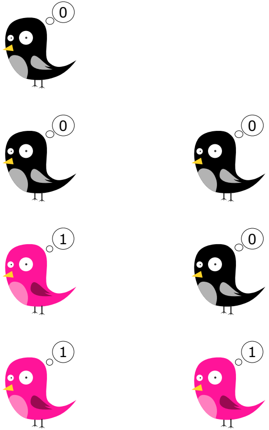
In every step: choose pair of agents uniformly at random.
These agents interact in this step.

Speed = expected number of steps until reaching stable consensus.



Expressive Power

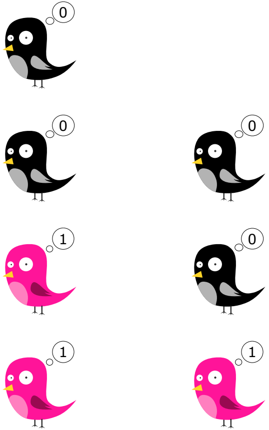
Special classes of properties.



Expressive Power

Special classes of properties.

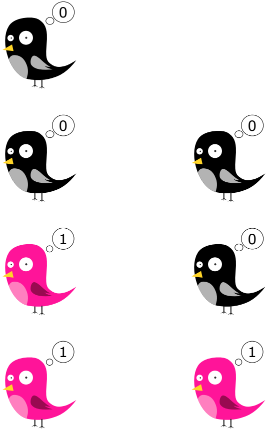
Class 1 (Threshold):



Expressive Power

Special classes of properties.

Class 1 (Threshold):
Every bird: Initially integer value
Decide total sum $\geq c$.

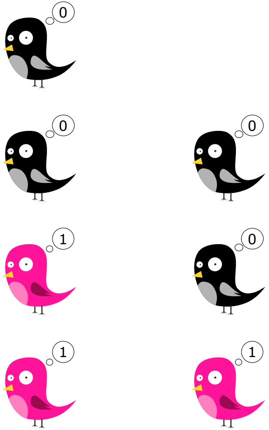


Expressive Power

Special classes of properties.

Class 1 (Threshold):
Every bird: Initially integer value
Decide total sum $\geq c$.

Example: #pink birds ≥ 3.

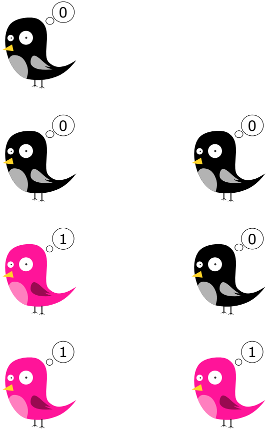


Expressive Power

Special classes of properties.

Class 1 (Threshold):
Every bird: Initially integer value
Decide total sum $\geq c$.

Example: #pink birds ≥ 3.
Allowed initial states: $1, 0$. Decide ≥ 3.

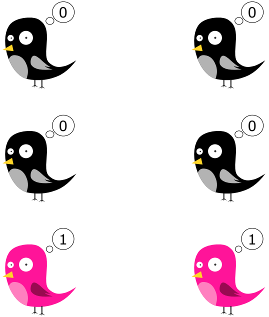


Expressive Power

Special classes of properties.

Class 1 (Threshold):
Every bird: Initially integer value
Decide total sum $\geq c$.

Example: #pink birds ≥ 3.
Allowed initial states: $1, 0$. Decide ≥ 3.



Expressive Power

Special classes of properties.

Class 1 (Threshold):
Every bird: Initially integer value
Decide total sum $\geq c$.

Example: Majority.

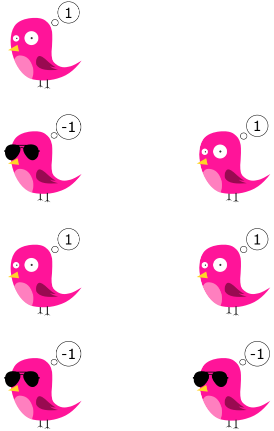
Expressive Power

Special classes of properties.

Class 1 (Threshold):
Every bird: Initially integer value
Decide total sum $\geq c$.

Example: Majority.
Allowed initial states: $1, -1$. Decide ≥ 0.

Glasses $=$ Negative value



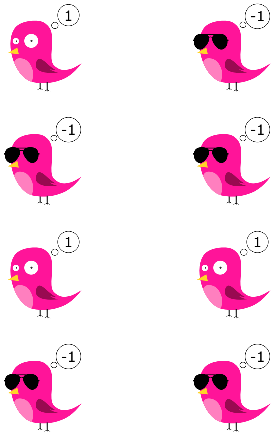
Expressive Power

Special classes of properties.

Class 1 (Threshold):
Every bird: Initially integer value
Decide total sum $\geq c$.

Example: Majority.
Allowed initial states: $1, -1$. Decide ≥ 0.

Glasses = Negative value



Expressive Power

Special classes of properties.

Class 1 (Threshold):
Every bird: Initially integer value
Decide total sum $\geq c$.

Class 2 (Modulo):

Expressive Power

Special classes of properties.

Class 1 (Threshold):
Every bird: Initially integer value
Decide total sum $\geq c$.

Class 2 (Modulo):
Every bird: Initially integer value
Decide total sum $\equiv_m c$

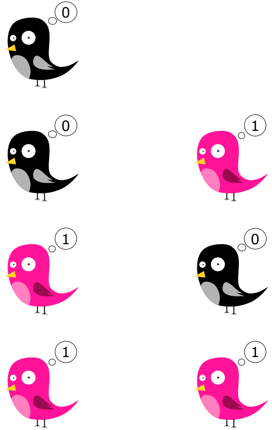
Expressive Power

Special classes of properties.

Class 1 (Threshold):
Every bird: Initially integer value
Decide total sum $\geq c$.

Class 2 (Modulo):
Every bird: Initially integer value
Decide total sum $\equiv_m c$

Example: # pink birds is even.

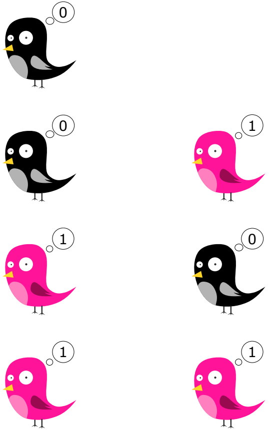


Special classes of properties.

Class 1 (Threshold):
Every bird: Initially integer value
Decide total sum $\geq c$.

Class 2 (Modulo):
Every bird: Initially integer value
Decide total sum $\equiv_m c$

Example: # pink birds is even.
Allowed initial states: $1, 0$. Decide $\equiv_2 0$.



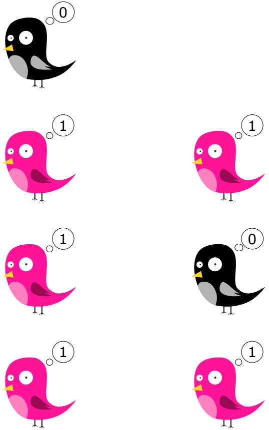
Expressive Power

Special classes of properties.

Class 1 (Threshold):
Every bird: Initially integer value
Decide total sum $\geq c$.

Class 2 (Modulo):
Every bird: Initially integer value
Decide total sum $\equiv_m c$

Example: # pink birds is even.
Allowed initial states: $1, 0$. Decide $\equiv_2 0$.



Expressive Power

Special classes of properties.

Class 1 (Threshold):
Every bird: Initially integer value
Decide total sum $\geq c$.

Class 2 (Modulo):
Every bird: Initially integer value
Decide total sum $\equiv_m c$

Angluin et. al. [2006]: Expressive power: Exactly all boolean combinations of threshold and modulo. This class is called Quantifier Free Presburger Arithmetic (*QFPA*).

Expressive Power

Special classes of properties.

Regarding $|\varphi|$: Encode predicates.

Class 1 (Threshold):
Every bird: Initially integer value
Decide total sum $\geq c$.

Class 2 (Modulo):
Every bird: Initially integer value
Decide total sum $\equiv_m c$

Angluin et. al. [2006]: Expressive power: Exactly all boolean combinations of threshold and modulo. This class is called Quantifier Free Presburger Arithmetic (*QFPA*).

Expressive Power

Special classes of properties.

Class 1 (Threshold):
Every bird: Initially integer value
Decide total sum $\geq c$.

Class 2 (Modulo):
Every bird: Initially integer value
Decide total sum $\equiv_m c$

Regarding $|\varphi|$: Encode predicates.

Allowed initial states: $4, -3$. Decide ≥ 0.
$4x - 3y \geq 0$.

Angluin et. al. [2006]: Expressive power: Exactly all boolean combinations of threshold and modulo. This class is called Quantifier Free Presburger Arithmetic (*QFPA*).

Expressive Power

Special classes of properties.

Class 1 (Threshold):
Every bird: Initially integer value
Decide total sum $\geq c$.

Class 2 (Modulo):
Every bird: Initially integer value
Decide total sum $\equiv_m c$

Regarding $|\varphi|$: Encode predicates.

Allowed initial states: $4, -3$. Decide ≥ 0.
$4x - 3y \geq 0$.

$|\varphi|=$ length of string with numbers in binary.

Angluin et. al. [2006]: Expressive power: Exactly all boolean combinations of threshold and modulo. This class is called Quantifier Free Presburger Arithmetic (*QFPA*).

Goal: Synthesis Procedure

Goal: Synthesis Procedure

Procedure for following problem:

Goal: Synthesis Procedure

Procedure for following problem:

Input: Formula $\varphi \in \mathit{QFPA}$.

Goal: Synthesis Procedure

Procedure for following problem:

Input: Formula $\varphi \in \mathit{QFPA}$.

Output: Population Protocol deciding φ.

Goal: Synthesis Procedure

Procedure for following problem:

Input: Formula $\varphi \in \mathit{QFPA}$.

Output: Population Protocol deciding φ.

Synthesis procedures are compared via

Goal: Synthesis Procedure

Procedure for following problem:

Input: Formula $\varphi \in \mathit{QFPA}$.

Output: Population Protocol deciding φ.

Synthesis procedures are compared via

- state complexity of protocols in $|\varphi|$,

Goal: Synthesis Procedure

Procedure for following problem:

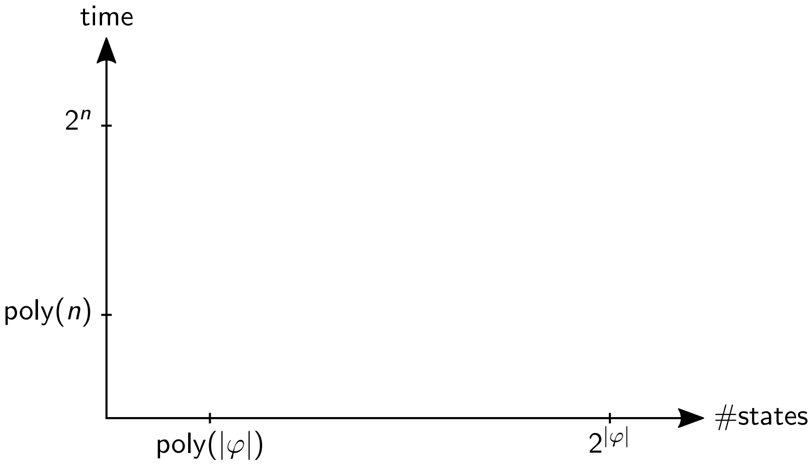
Input: Formula $\varphi \in QFPA$.

Output: Population Protocol deciding φ.

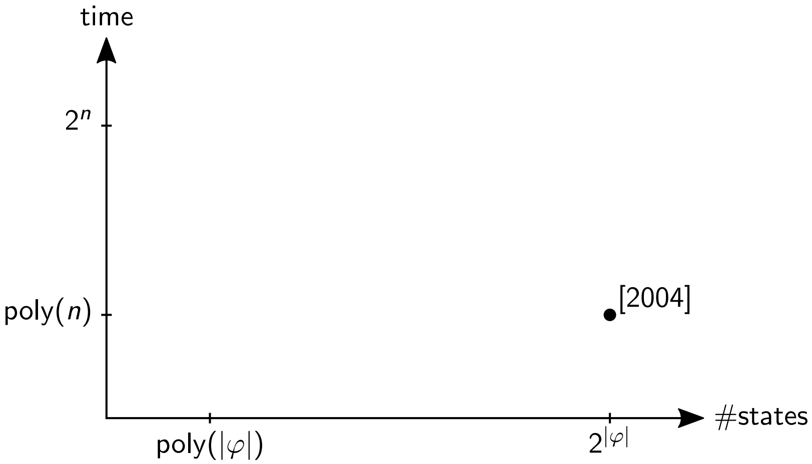
Synthesis procedures are compared via

- state complexity of protocols in $|\varphi|$,
- speed in $n := \#$agents participating.

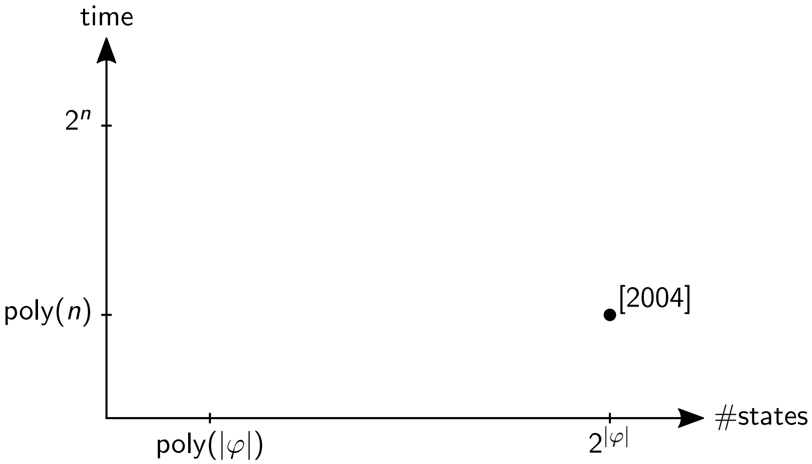
Prior Work



Prior Work

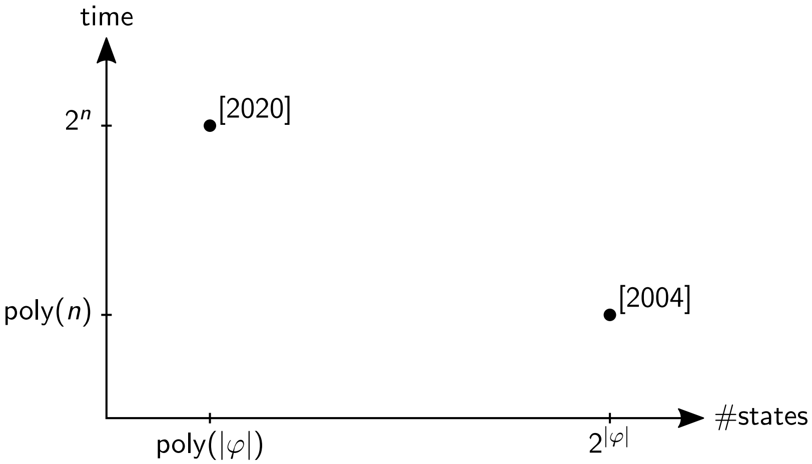


Prior Work



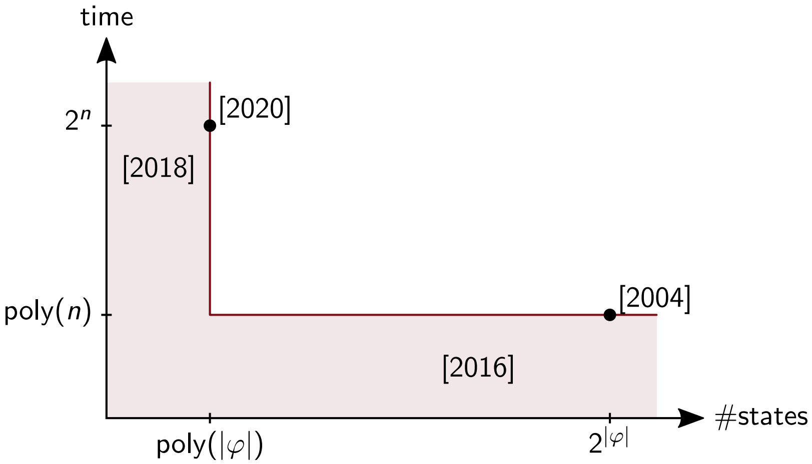
$c + 1$ states for $x \geq c$ is exponential in $|\varphi|$.

Prior Work



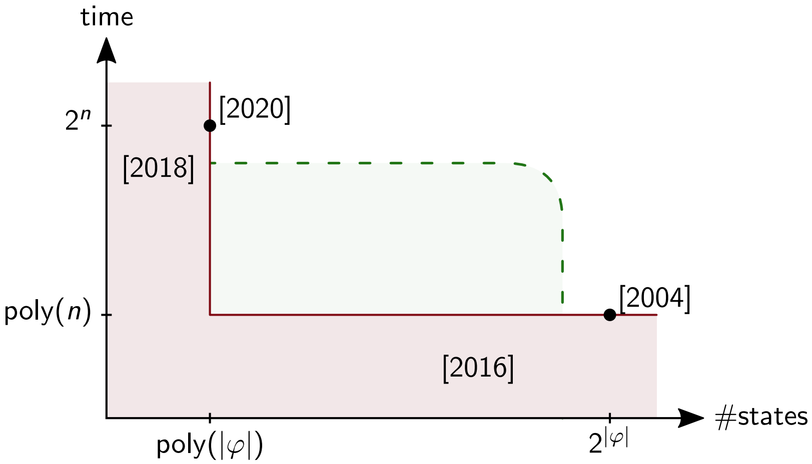
$c + 1$ states for $x \geq c$ is exponential in $|\varphi|$.

Prior Work



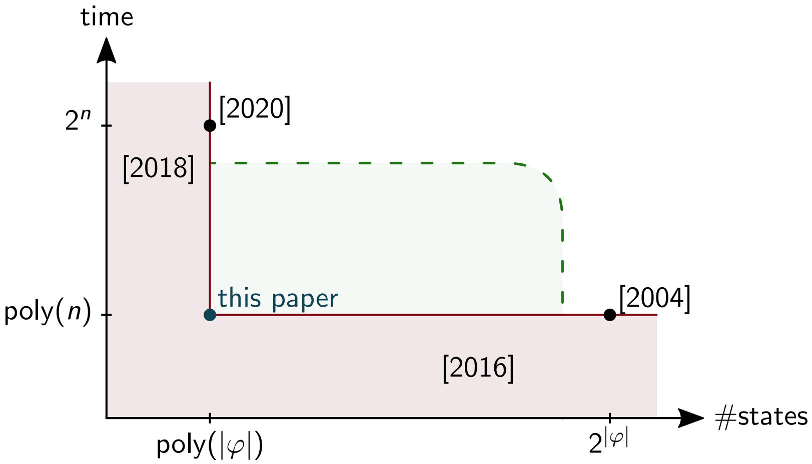
$c + 1$ states for $x \geq c$ is exponential in $|\varphi|$.

Prior Work



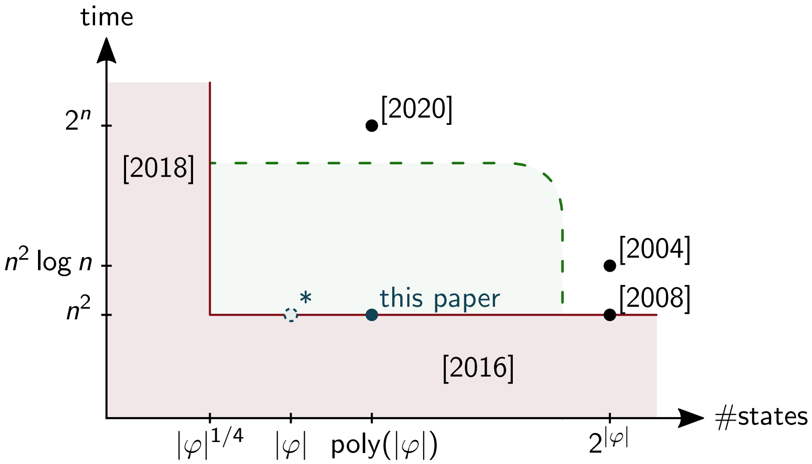
$c + 1$ states for $x \geq c$ is exponential in $|\varphi|$.

Overview



$c+1$ states for $x \geq c$ is exponential in $|\varphi|$.

Overview



$c + 1$ states for $x \geq c$ is exponential in $|\varphi|$.

$*$: $n \in \Omega(|\varphi|)$

Roadmap towards Fast and Succinct Population Protocols

Roadmap towards Fast and Succinct Population Protocols

- To simplify protocol design, we introduce a more general model.

Roadmap towards Fast and Succinct Population Protocols

- To simplify protocol design, we introduce a more general model.
- Careful extension such that the conversion generates fast and succinct protocols.

Roadmap towards Fast and Succinct Population Protocols

- To simplify protocol design, we introduce a more general model.
- Careful extension such that the conversion generates fast and succinct protocols.
- Population Computers (PC) extension:
    1.
    2.
    3.

Roadmap towards Fast and Succinct Population Protocols

- To simplify protocol design, we introduce a more general model.
- Careful extension such that the conversion generates fast and succinct protocols.
- Population Computers (PC) extension:
  1. Multiway interactions.
  2.
  3.

Roadmap towards Fast and Succinct Population Protocols

- To simplify protocol design, we introduce a more general model.
- Careful extension such that the conversion generates fast and succinct protocols.
- Population Computers (PC) extension:
  1. Multiway interactions.
  2. Output function.
  3.

Roadmap towards Fast and Succinct Population Protocols

- To simplify protocol design, we introduce a more general model.
- Careful extension such that the conversion generates fast and succinct protocols.
- Population Computers (PC) extension:
  1. Multiway interactions.
  2. Output function.
  3. Helpers.

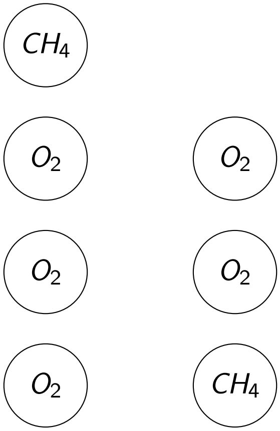
Roadmap towards Fast and Succinct Population Protocols

- To simplify protocol design, we introduce a more general model.
- Careful extension such that the conversion generates fast and succinct protocols.
- Population Computers (PC) extension:
    1. Multiway interactions.
    2. Output function.
    3. Helpers.
- Design succinct PCs satisfying a simple property.

Roadmap towards Fast and Succinct Population Protocols

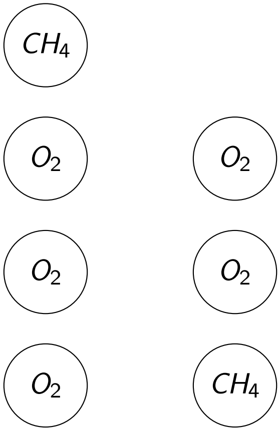
- To simplify protocol design, we introduce a more general model.
- Careful extension such that the conversion generates fast and succinct protocols.
- Population Computers (PC) extension:
  1. Multiway interactions.
  2. Output function.
  3. Helpers.
- Design succinct PCs satisfying a simple property.
- Convert them to population protocols.

Extension 1: Multiway interactions



Extension 1: Multiway interactions

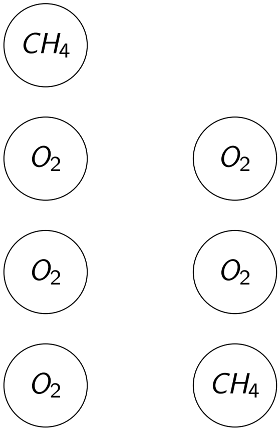
Reminder: Chemical reaction networks.



Extension 1: Multiway interactions

Reminder: Chemical reaction networks.

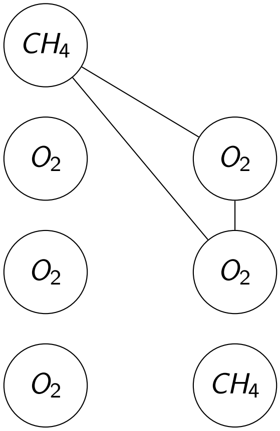
$$CH_4 + 2 \cdot O_2 \rightarrow CO_2 + 2 \cdot H_2O$$
$$CO_2 + 6 \cdot H_2O \rightarrow 6 \cdot O_2 + C_6H_{12}O_6$$



Extension 1: Multiway interactions

Reminder: Chemical reaction networks.

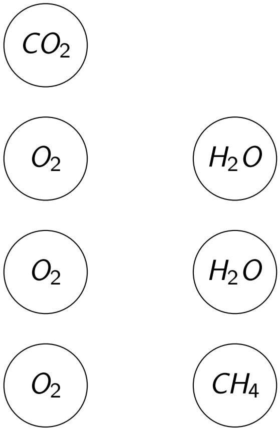
$$CH_4 + 2 \cdot O_2 \rightarrow CO_2 + 2 \cdot H_2O$$
$$CO_2 + 6 \cdot H_2O \rightarrow 6 \cdot O_2 + C_6H_{12}O_6$$



Extension 1: Multiway interactions

Reminder: Chemical reaction networks.

$$CH_4 + 2 \cdot O_2 \rightarrow CO_2 + 2 \cdot H_2O$$
$$CO_2 + 6 \cdot H_2O \rightarrow 6 \cdot O_2 + C_6H_{12}O_6$$

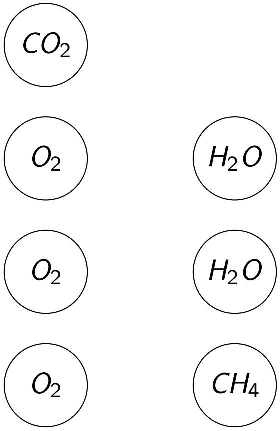


Extension 1: Multiway interactions

Reminder: Chemical reaction networks.

$$
\begin{aligned}
CH_4 + 2 \cdot O_2 &\to CO_2 + 2 \cdot H_2O \\
CO_2 + 6 \cdot H_2O &\to 6 \cdot O_2 + C_6H_{12}O_6
\end{aligned}
$$

Chemical reactions often have only few types of reactants.

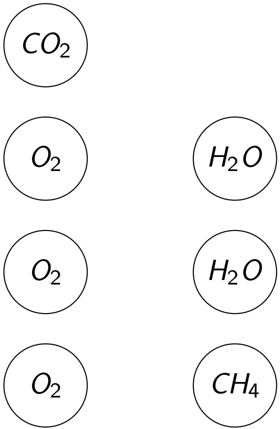


Extension 1: Multiway interactions

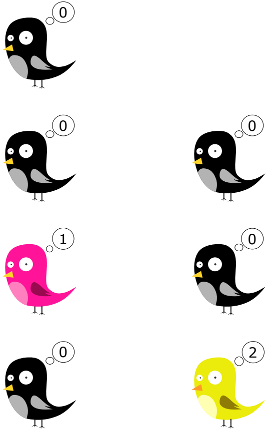
Reminder: Chemical reaction networks.

$$
\begin{aligned}
CH_4 + 2 \cdot O_2 &\rightarrow CO_2 + 2 \cdot H_2O \\
CO_2 + 6 \cdot H_2O &\rightarrow 6 \cdot O_2 + C_6H_{12}O_6
\end{aligned}
$$

Chemical reactions often have only few types of reactants.
We only allow multiways with two types of reacting states.

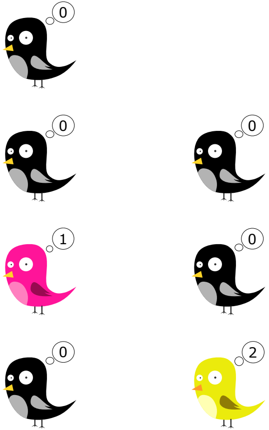


Extension 2: Output Function



Extension 2: Output Function

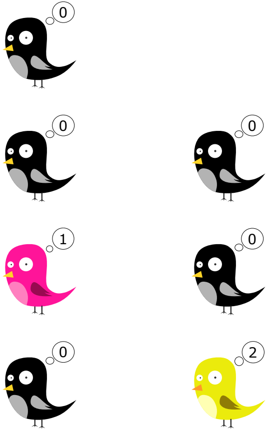
Reminder: Example #pink birds ≥ 3.



Extension 2: Output Function

Reminder: Example #pink birds ≥ 3.

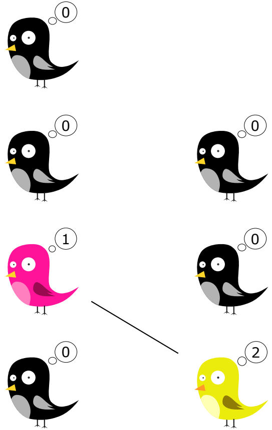
$$
\begin{aligned}
i, j &\mapsto i + j, 0 && \text{if } i + j < 3, \\
i, j &\mapsto 3, 3 && \text{if } i + j \geq 3.
\end{aligned}
$$



Extension 2: Output Function

Reminder: Example #pink birds ≥ 3.

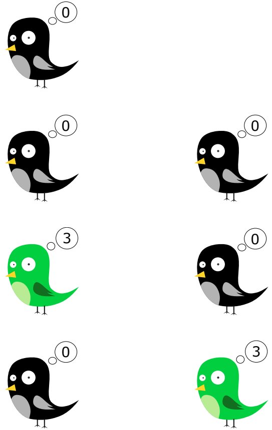
$$
\begin{aligned}
i, j &\mapsto i + j, 0 && \text{if } i + j < 3, \\
i, j &\mapsto 3, 3 && \text{if } i + j \geq 3.
\end{aligned}
$$



Extension 2: Output Function

Reminder: Example #pink birds ≥ 3.

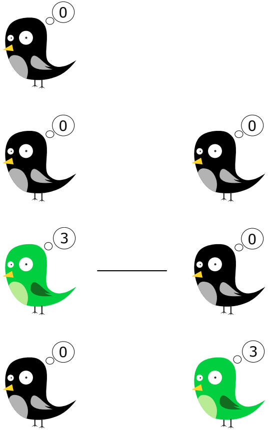
$$i, j \mapsto i + j, 0 \qquad \text{if } i + j < 3,$$
$$i, j \mapsto 3, 3 \qquad \text{if } i + j \geq 3.$$



Extension 2: Output Function

Reminder: Example #pink birds ≥ 3.

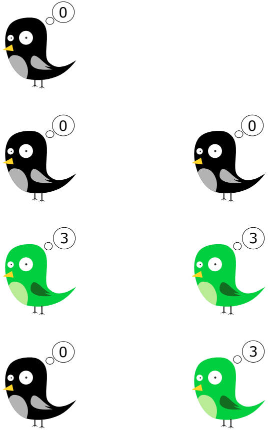
$$i, j \mapsto i + j, 0 \qquad \text{if } i + j < 3,$$
$$i, j \mapsto 3, 3 \qquad \text{if } i + j \geq 3.$$



Extension 2: Output Function

Reminder: Example #pink birds ≥ 3.

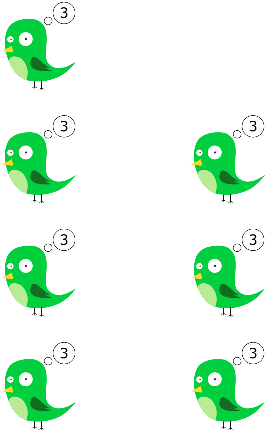
$$
\begin{array}{ll}
i, j \mapsto i + j, 0 & \text{if } i + j < 3, \\
i, j \mapsto 3, 3 & \text{if } i + j \geq 3.
\end{array}
$$



Extension 2: Output Function

Reminder: Example #pink birds ≥ 3.

$$
\begin{aligned}
i, j &\mapsto i + j, 0 && \text{if } i + j < 3, \\
i, j &\mapsto 3, 3 && \text{if } i + j \geq 3.
\end{aligned}
$$

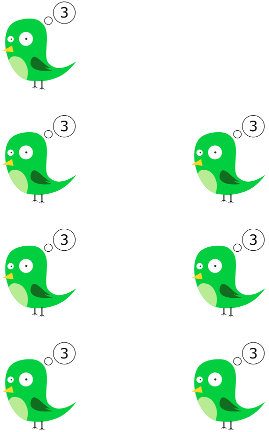


Extension 2: Output Function

Reminder: Example #pink birds ≥ 3.

$$i, j \mapsto i + j, 0 \qquad \text{if } i + j < 3,$$
$$i, j \mapsto 3, 3 \qquad \text{if } i + j \geq 3.$$

Output broadcast has little in common with rest of the protocol.



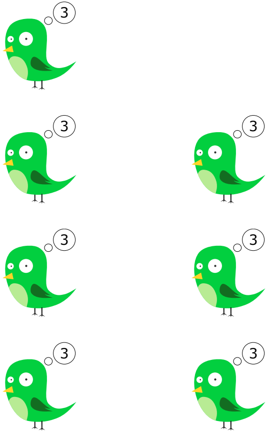
Extension 2: Output Function

Reminder: Example #pink birds ≥ 3.

$$
\begin{aligned}
i, j &\mapsto i+j, 0 && \text{if } i+j < 3, \\
i, j &\mapsto 3, 3 && \text{if } i+j \geq 3.
\end{aligned}
$$

Output broadcast has little in common with rest of the protocol.

Split these two parts.



Extension 2: Output Function

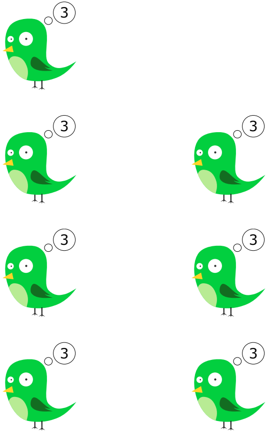
Reminder: Example #pink birds ≥ 3.

$$
\begin{aligned}
i, j &\mapsto i + j, 0 && \text{if } i + j < 3, \\
i, j &\mapsto 3, 3 && \text{if } i + j \geq 3.
\end{aligned}
$$

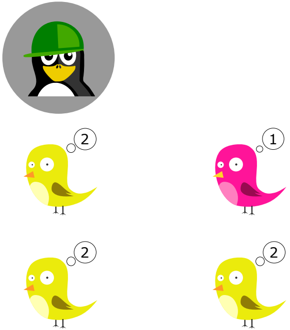
Output broadcast has little in common with rest of the protocol.

Split these two parts.

More general output function.

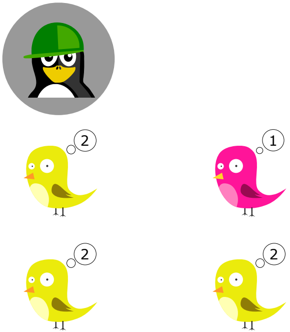


Extension 3: Helpers



Extension 3: Helpers

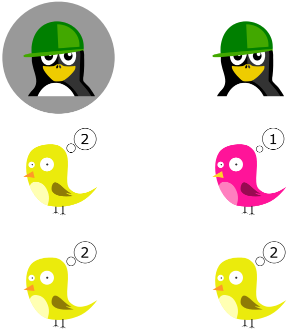
Auxiliary agents which do not count towards the input.



Extension 3: Helpers

Auxiliary agents which do not count towards the input.

Caution: Count is not known, only minimum is.

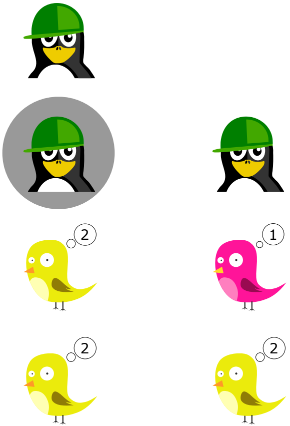


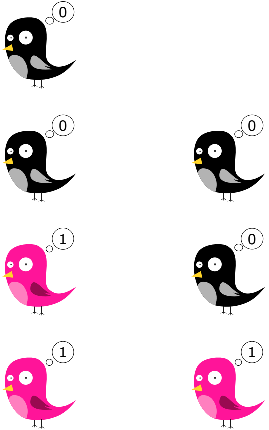
Extension 3: Helpers

Auxiliary agents which do not count towards the input.

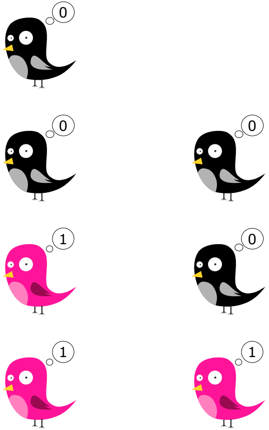
Caution: Count is not known, only minimum is.

Idea: Computations often require auxiliary variables/gadgets.

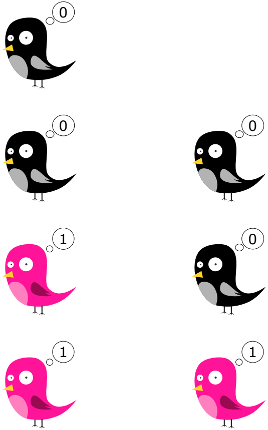



0
0
0
1
0
1
1

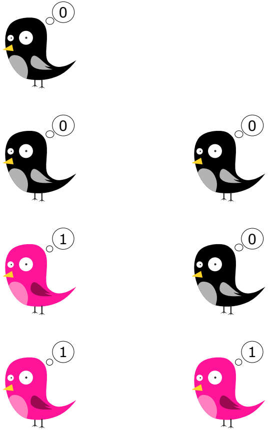
To ensure speed, we need bounded computers.



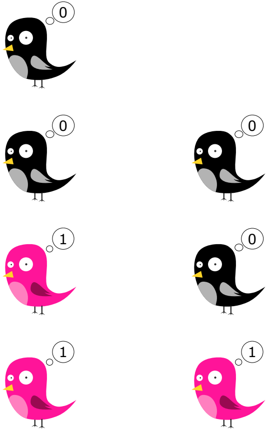
To ensure speed, we need bounded computers.
A computer is bounded if,



To ensure speed, we need bounded computers.
A computer is bounded if,
only counting transitions with an effect,

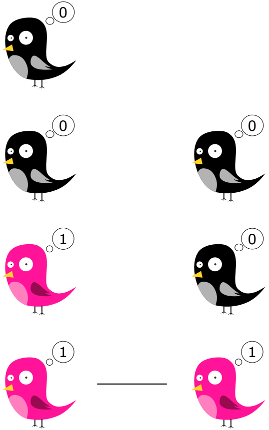


To ensure speed, we need bounded computers.
A computer is bounded if,
only counting transitions with an effect,
every execution is finite.

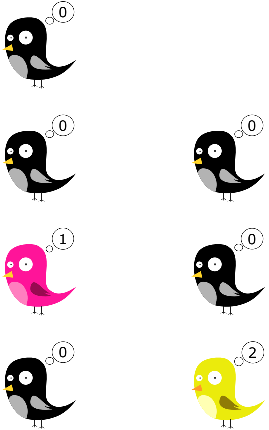


Conversion of Population Computers/Main Theorems

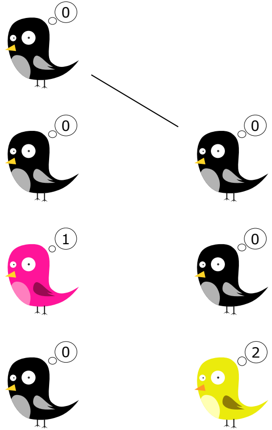
To ensure speed, we need bounded computers.
A computer is bounded if,
only counting transitions with an effect,
every execution is finite.



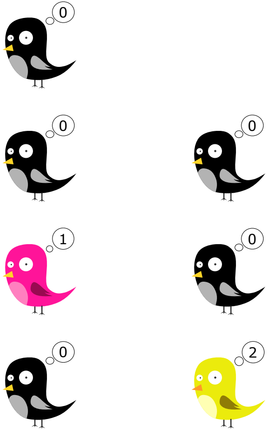
To ensure speed, we need bounded computers.
A computer is bounded if,
only counting transitions with an effect,
every execution is finite.



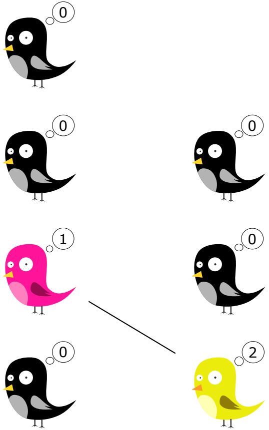
To ensure speed, we need bounded computers.
A computer is bounded if,
only counting transitions with an effect,
every execution is finite.



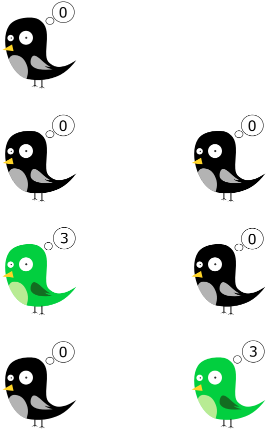
To ensure speed, we need bounded computers.
A computer is bounded if,
only counting transitions with an effect,
every execution is finite.



To ensure speed, we need bounded computers.
A computer is bounded if,
only counting transitions with an effect,
every execution is finite.

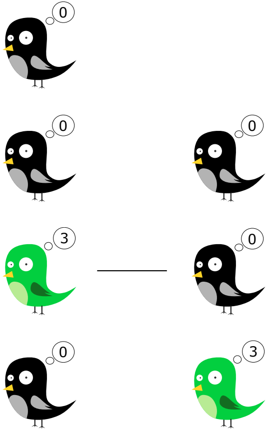


To ensure speed, we need bounded computers.
A computer is bounded if,
only counting transitions with an effect,
every execution is finite.

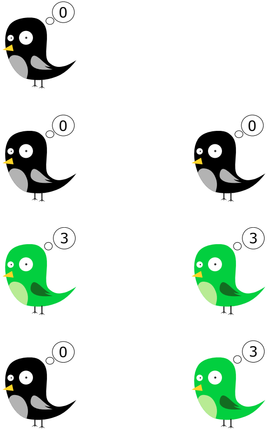


Conversion of Population Computers/Main Theorems

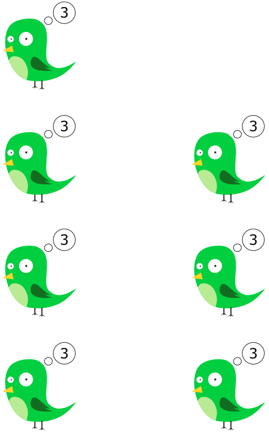
To ensure speed, we need bounded computers.
A computer is bounded if,
only counting transitions with an effect,
every execution is finite.



To ensure speed, we need bounded computers.
A computer is bounded if,
only counting transitions with an effect,
every execution is finite.



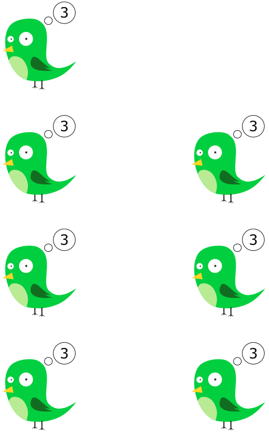
To ensure speed, we need bounded computers.
A computer is bounded if,
only counting transitions with an effect,
every execution is finite.



Conversion of Population Computers/Main Theorems

To ensure speed, we need bounded computers.
A computer is bounded if,
only counting transitions with an effect,
every execution is finite.

Determining boundedness does not require
a complicated analysis.



Population Computer

Population Computer

State complexity $\mathcal{O}(|\varphi|)$

Population Computer

State complexity $\mathcal{O}(|\varphi|)$

Bounded

Conversion of Population Computers/Main Theorems

Population Computer

State complexity $\mathcal{O}(|\varphi|)$

Bounded

$\rightarrow\rightarrow\rightarrow$

Population Computer		Population Protocol
State complexity $\mathcal{O}(\lvert\varphi\rvert)$	$\to\to\to$	
Bounded		

Conversion of Population Computers/Main Theorems

Population Computer		Population Protocol
State complexity $\mathcal{O}(\vert\varphi\vert)$	$\to\to\to$	State complexity $\mathcal{O}(\vert\varphi\vert^2)$
Bounded		

Conversion of Population Computers/Main Theorems

Population Computer		Population Protocol
State complexity $\mathcal{O}(\|\varphi\|)$	$\rightarrow\rightarrow\rightarrow$	State complexity $\mathcal{O}(\|\varphi\|^2)$
Bounded		Speed $\mathcal{O}(n^3)$

Conversion of Population Computers/Main Theorems

Population Computer		Population Protocol
State complexity $\mathcal{O}(\lvert\varphi\rvert)$	$\rightarrow\rightarrow\rightarrow$	State complexity $\mathcal{O}(\lvert\varphi\rvert^2)$
Bounded		Speed $\mathcal{O}(n^3)$
		Inputs fulfilling $n \in \Omega(\lvert\varphi\rvert)$

Conversion of Population Computers/Main Theorems

Population Computer

State complexity $\mathcal{O}(|\varphi|)$

Bounded

$\rightarrow\rightarrow\rightarrow$

Population Protocol

State complexity $\mathcal{O}(|\varphi|^2)$

Speed $\mathcal{O}(n^3)$

Inputs fulfilling $n \in \Omega(|\varphi|)$

Population Computer

Conversion of Population Computers/Main Theorems

Population Computer

State complexity $\mathcal{O}(|\varphi|)$

Bounded

$\rightarrow\rightarrow\rightarrow$

Population Protocol

State complexity $\mathcal{O}(|\varphi|^2)$

Speed $\mathcal{O}(n^3)$

Inputs fulfilling $n \in \Omega(|\varphi|)$

Population Computer

State complexity $\mathcal{O}(|\varphi|)$

Conversion of Population Computers/Main Theorems

Population Computer

State complexity $\mathcal{O}(|\varphi|)$

Bounded

$\rightarrow\rightarrow\rightarrow$

Population Protocol

State complexity $\mathcal{O}(|\varphi|^2)$

Speed $\mathcal{O}(n^3)$

Inputs fulfilling $n \in \Omega(|\varphi|)$

Population Computer

State complexity $\mathcal{O}(|\varphi|)$

Rapid

Conversion of Population Computers/Main Theorems

Population Computer

State complexity $\mathcal{O}(|\varphi|)$

Bounded

$\rightarrow\rightarrow\rightarrow$

Population Protocol

State complexity $\mathcal{O}(|\varphi|^2)$

Speed $\mathcal{O}(n^3)$

Inputs fulfilling $n \in \Omega(|\varphi|)$

Population Computer

State complexity $\mathcal{O}(|\varphi|)$

Rapid

$\rightarrow\rightarrow\rightarrow$

Conversion of Population Computers/Main Theorems

Population Computer		Population Protocol
State complexity $\mathcal{O}(\lvert\varphi\rvert)$	$\longrightarrow\!\longrightarrow\!\longrightarrow$	State complexity $\mathcal{O}(\lvert\varphi\rvert^2)$
Bounded		Speed $\mathcal{O}(n^3)$
		Inputs fulfilling $n \in \Omega(\lvert\varphi\rvert)$

Population Computer		Population Protocol
State complexity $\mathcal{O}(\lvert\varphi\rvert)$	$\longrightarrow\!\longrightarrow\!\longrightarrow$	
Rapid		

Conversion of Population Computers/Main Theorems

Population Computer

State complexity $\mathcal{O}(|\varphi|)$

Bounded

$\rightarrow\rightarrow\rightarrow$

Population Protocol

State complexity $\mathcal{O}(|\varphi|^2)$

Speed $\mathcal{O}(n^3)$

Inputs fulfilling $n \in \Omega(|\varphi|)$

Population Computer

State complexity $\mathcal{O}(|\varphi|)$

Rapid

$\rightarrow\rightarrow\rightarrow$

Population Protocol

State complexity $\mathcal{O}(|\varphi|)$

Conversion of Population Computers/Main Theorems

Population Computer		Population Protocol
State complexity $\mathcal{O}(\lvert\varphi\rvert)$	$\to\to\to$	State complexity $\mathcal{O}(\lvert\varphi\rvert^2)$
Bounded		Speed $\mathcal{O}(n^3)$
		Inputs fulfilling $n \in \Omega(\lvert\varphi\rvert)$

Population Computer		Population Protocol
State complexity $\mathcal{O}(\lvert\varphi\rvert)$	$\to\to\to$	State complexity $\mathcal{O}(\lvert\varphi\rvert)$
Rapid		Speed $\mathcal{O}(n^2)$

Conversion of Population Computers/Main Theorems

Population Computer		Population Protocol
State complexity $\mathcal{O}(\lvert\varphi\rvert)$	$\to\to\to$	State complexity $\mathcal{O}(\lvert\varphi\rvert^2)$
Bounded		Speed $\mathcal{O}(n^3)$
		Inputs fulfilling $n \in \Omega(\lvert\varphi\rvert)$

Population Computer		Population Protocol
State complexity $\mathcal{O}(\lvert\varphi\rvert)$	$\to\to\to$	State complexity $\mathcal{O}(\lvert\varphi\rvert)$
Rapid		Speed $\mathcal{O}(n^2)$
		Inputs fulfilling $n \in \Omega(\lvert\varphi\rvert)$

Conversion of Population Computers/Main Theorems

Population Computer		Population Protocol
State complexity $\mathcal{O}(\lvert\varphi\rvert)$	$\rightarrow\rightarrow\rightarrow$	State complexity $\mathcal{O}(\lvert\varphi\rvert^2)$
Bounded		Speed $\mathcal{O}(n^3)$
		Inputs fulfilling $n \in \Omega(\lvert\varphi\rvert)$

Population Computer		Population Protocol
State complexity $\mathcal{O}(\lvert\varphi\rvert)$	$\rightarrow\rightarrow\rightarrow$	State complexity $\mathcal{O}(\lvert\varphi\rvert)$
Rapid		Speed $\mathcal{O}(n^2)$
		Inputs fulfilling $n \in \Omega(\lvert\varphi\rvert)$

Blondin et. al. [2020]: Remove input restriction at cost of $\mathcal{O}(\text{poly}(\lvert\varphi\rvert))$ states.

Thank you for your attention!