Fast and Succinct Population Protocols for Presburger Arithmetic

Philipp Czerner, Javier Esparza, Roland Guttenberg, Martin Helfrich

Technical University of Munich

September 12 2022

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
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,
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,
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**.
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 \geq 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 \( Q = \{0, 1, 2, 3\} \).
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 $\geq 3$.
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 $\geq 3$.
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 \\( \geq 3 \).
States \( Q = \{0, 1, 2, 3\} \).
Colors and numbers encode the same.
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 \(\geq 3\).
States \(Q = \{0, 1, 2, 3\}\).
Colors and numbers encode the same.
Population Protocols = model of computation
- anonymous \textit{finite-state} agents (birds),
- decide property of \textit{initial configuration},
- inputs = \textit{counts} of initial states,
- only \textit{pairwise} interactions,
- output by \textit{stable consensus}.

Example: Decide \#pink birds \geq 3.
States \( Q = \{0, 1, 2, 3\} \).

\textit{Colors} and \textit{numbers} encode the same.
Population Protocols = model of computation
- anonymous \textit{finite-state} agents (birds),
- decide property of \textit{initial configuration},
- inputs = \textit{counts} of initial states,
- only \textit{pairwise} interactions,
- output by \textit{stable consensus}.

**Example:** Decide \#\textit{pink} birds \(\geq 3\).
States \(Q = \{0, 1, 2, 3\}\).

\textbf{Colors and numbers} encode the same.
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 \( \geq 3 \).
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 \(\geq 3\).
States \(Q = \{0, 1, 2, 3\}\).
Colors and numbers encode the same.
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 \( \geq 3 \).
States \( Q = \{0, 1, 2, 3\} \).
Colors and numbers encode the same.
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 \(\geq 3\).
States \(Q = \{0, 1, 2, 3\}\).
Colors and numbers encode the same.
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 \( \geq 3 \).
States \( Q = \{0, 1, 2, 3\} \).

Colors and numbers encode the same.

Protocol has to be correct for all initial configurations.
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 \( \geq 3 \).
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.

\[ \text{HCl} + \text{NaOH} \rightarrow \text{NaCl} + \text{H}_2\text{O} \]
Applications of Population Protocols

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.

\[ \text{HCl} + \text{NaOH} \rightarrow \text{NaCl} + \text{H}_2\text{O} \]

State complexity: \# species.

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

Mobile sensor networks, ...
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.
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: $\#\text{pink birds} \geq 3$. 
Special classes of properties.

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

Example: #pink birds \( \geq 3 \).
Allowed initial states: 1, 0. Decide \( \geq 3 \).
Expressive Power

Special classes of properties.

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

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

Example: Majority.
Allowed initial states: 1, −1. Decide ≥ 0.
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 $\geq 0$. 

Glasses = Negative value
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.
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 $\equiv_m c$.

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

Regarding $|\varphi|$: Encode predicates.
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 $\geq 0$.
$4x - 3y \geq 0$.

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 $\geq 0$.

$4x - 3y \geq 0$.

$|\varphi|$ = 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 $\varphi \in QFPA$. 
Goal: Synthesis Procedure

Procedure for following problem:

**Input**: Formula $\varphi \in QFPA$.

**Output**: Population Protocol deciding $\varphi$. 
Procedure for following problem:

**Input:** Formula $\varphi \in QFPA$.

**Output:** Population Protocol deciding $\varphi$.

Synthesis procedures are compared via
Procedure for following problem:

**Input:** Formula $\varphi \in QFPA$.

**Output:** Population Protocol deciding $\varphi$.

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 $\varphi$.

Synthesis procedures are compared via

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

Prior Work

\[ \text{time} \]

\[ 2^n \]

\[ \text{poly}(n) \]

\[ \text{poly}(|\varphi|) \]

\[ 2^{|\varphi|} \]

\[ \#\text{states} \]
Prior Work

\[
\begin{align*}
\text{time} & \quad 2^n \\
\text{poly}(n) & \quad \text{poly}(|\varphi|) \\
\text{#states} & \quad 2^{|\varphi|} \\
\end{align*}
\]

[2004]
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 $|\varphi|$. 

\begin{align*}
\text{time} & \quad 2^n \\
\text{poly}(n) & \quad \text{poly}(|\varphi|) \\
\text{poly}(|\varphi|) & \quad 2^{|\varphi|} \\
\text{#states} & \quad [2004] \\
\text{#states} & \quad [2020]
\end{align*}
$c + 1$ states for $x \geq c$ is exponential in $|\varphi|$. 
Prior Work

\[ n \leq 2^n \]

\[ \text{poly}(n) \leq 2^{\vert \varphi \vert} \]

\[ \text{poly}(\vert \varphi \vert) \leq \text{poly}(n) \]

\[ n \leq 2^{\vert \varphi \vert} \]

\[ c + 1 \text{ states for } x \geq c \text{ is exponential in } \vert \varphi \vert. \]
$c + 1$ states for $x \geq c$ is exponential in $|\varphi|$. 

\begin{align*}
\text{time} & \quad \text{poly}(n) \\
2^n & \quad \text{poly}(\varphi) \\
\text{this paper} & \quad [2018] \\
\text{[2020]} & \\
\text{[2004]} & \\
\text{[2016]} & \\
\#\text{states} & \quad 2^{|\varphi|}
\end{align*}
This paper:

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

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

- $2^n$ states for $|\varphi|^{1/4}$
- $n^2 \log n$ states for $|\varphi|$ poly($|\varphi|$)
- $n^2$ states for $2^{|\varphi|}$
To simplify protocol design, we introduce a more general model.
To simplify protocol design, we introduce a more general model.

Careful extension such that the conversion generates fast and succinct 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:
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.
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.
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.
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.
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

\[ \text{CH}_4 \]

\[ O_2 \]

\[ O_2 \]

\[ O_2 \]

\[ O_2 \]
Reminder: Chemical reaction networks.

\[ \text{CH}_4 \]
\[ \text{O}_2 \]
\[ \text{CH}_4 \]
Extension 1: Multiway interactions

Reminder: Chemical reaction networks.

\[ \text{CH}_4 + 2 \cdot \text{O}_2 \rightarrow \text{CO}_2 + 2 \cdot \text{H}_2\text{O} \]
\[ \text{CO}_2 + 6 \cdot \text{H}_2\text{O} \rightarrow 6 \cdot \text{O}_2 + \text{C}_6\text{H}_{12}\text{O}_6 \]
Reminder: Chemical reaction networks.

\[ CH_4 + 2 \cdot O_2 \rightarrow CO_2 + 2 \cdot H_2O \]
\[ CO_2 + 6 \cdot H_2O \rightarrow 6 \cdot O_2 + C_6H_{12}O_6 \]
Reminder: Chemical reaction networks.

\[ \text{CH}_4 + 2 \cdot \text{O}_2 \rightarrow \text{CO}_2 + 2 \cdot \text{H}_2\text{O} \]
\[ \text{CO}_2 + 6 \cdot \text{H}_2\text{O} \rightarrow 6 \cdot \text{O}_2 + \text{C}_6\text{H}_12\text{O}_6 \]
Reminder: Chemical reaction networks.

\[ \text{CH}_4 + 2 \cdot \text{O}_2 \rightarrow \text{CO}_2 + 2 \cdot \text{H}_2\text{O} \]
\[ \text{CO}_2 + 6 \cdot \text{H}_2\text{O} \rightarrow 6 \cdot \text{O}_2 + \text{C}_6\text{H}_{12}\text{O}_6 \]

Chemical reactions often have only few types of reactants.
Reminder: Chemical reaction networks.

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

Chemical reactions often have only few types of reactants.
We only allow multiways with two types of reacting states.
Extension 2: Output Function
Reminder: Example \#pink birds $\geq 3$. 
Extension 2: Output Function

**Reminder:** Example #pink birds $\geq 3$.

\[
i, j \mapsto i + j, 0 \quad \text{if } i + j < 3, \\
i, j \mapsto 3, 3 \quad \text{if } i + j \geq 3.
\]
Extension 2: Output Function

Reminder: Example #pink birds $\geq 3$.

\[ i, j \mapsto i + j, 0 \quad \text{if } i + j < 3, \]
\[ i, j \mapsto 3, 3 \quad \text{if } i + j \geq 3. \]
Extension 2: Output Function

**Reminder:** Example #pink birds $\geq 3$.

$$i, j \mapsto i + j, 0 \quad \text{if } i + j < 3,$$

$$i, j \mapsto 3, 3 \quad \text{if } i + j \geq 3.$$
Reminder: Example #pink birds $\geq 3$.

\[i, j \mapsto i + j, 0 \quad \text{if } i + j < 3,\]

\[i, j \mapsto 3, 3 \quad \text{if } i + j \geq 3.\]
Reminder: Example #pink birds $\geq 3$.

\[ i, j \mapsto i + j, 0 \quad \text{if } i + j < 3, \]
\[ i, j \mapsto 3, 3 \quad \text{if } i + j \geq 3. \]
Reminder: Example #pink birds $\geq 3$.

\[ i, j \mapsto i + j, 0 \quad \text{if } i + j < 3, \]
\[ i, j \mapsto 3, 3 \quad \text{if } i + j \geq 3. \]
Extension 2: Output Function

**Reminder:** Example #pink birds \( \geq 3 \).

\[
\begin{align*}
  i, j & \mapsto i + j, 0 \quad \text{if } i + j < 3, \\
  i, j & \mapsto 3, 3 \quad \text{if } i + j \geq 3.
\end{align*}
\]

Output broadcast has little in common with rest of the protocol.
Reminder: Example #pink birds ≥ 3.

\[ i, j \mapsto i + j, 0 \quad \text{if } i + j < 3, \]
\[ i, j \mapsto 3, 3 \quad \text{if } i + j \geq 3. \]

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

Split these two parts.
Extension 2: Output Function

Reminder: Example #pink birds $\geq 3$.

\[ i, j \mapsto i + j, 0 \quad \text{if } i + j < 3, \]
\[ i, j \mapsto 3, 3 \quad \text{if } i + j \geq 3. \]

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

Split these two parts.

More general output function.
Auxiliary agents which do not count towards the input.
Auxiliary agents which do not count towards the input.

Caution: Count is not known, only minimum is.
Auxiliary agents which do not count towards the input.

Caution: Count is not known, only minimum is.

Idea: Computations often require auxiliary variables/gadgets.
Conversion of Population Computers/Main Theorems
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.
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.
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.

Determining boundedness does not require a complicated analysis.
Blondin et al. [2020]: Remove input restriction at cost of $O(\text{poly}(\vert \varphi \vert))$ states.
Conversion of Population Computers/Main Theorems

Population Computer

Blondin et al. [2020]: Remove input restriction at cost of \( O(poly(|\phi|)) \) states.
Population Computer
State complexity $O(|\varphi|)$
Conversion of Population Computers/Main Theorems

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

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

Bounded
Conversion of Population Computers/Main Theorems

Population Computer
State complexity $\mathcal{O}(|\phi|)$
Bounded

Population Protocol

Blondin et. al. [2020]: Remove input restriction at cost of $O(poly(|\phi|))$ states.
<table>
<thead>
<tr>
<th>Population Computer</th>
<th>Population Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>State complexity $\mathcal{O}(</td>
<td>\varphi</td>
</tr>
<tr>
<td>Bounded</td>
<td></td>
</tr>
<tr>
<td>Population Computer</td>
<td>Population Protocol</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>State complexity $\mathcal{O}(</td>
<td>\varphi</td>
</tr>
<tr>
<td>Bounded</td>
<td>Speed $\mathcal{O}(n^3)$</td>
</tr>
</tbody>
</table>
### Conversion of Population Computers/Main Theorems

<table>
<thead>
<tr>
<th>Population Computer</th>
<th>Population Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>State complexity $O(</td>
<td>\varphi</td>
</tr>
<tr>
<td>Bounded</td>
<td>Speed $O(n^3)$</td>
</tr>
<tr>
<td></td>
<td>Inputs fulfilling $n \in \Omega(</td>
</tr>
</tbody>
</table>

Blondin et al. [2020]: Remove input restriction at cost of $O(poly(|\varphi|))$ states.
Population Computer
State complexity $O(|\varphi|)$
Bounded

→→→

Population Protocol
State complexity $O(|\varphi|^2)$
Speed $O(n^3)$
Inputs fulfilling $n \in \Omega(|\varphi|)$

Population Computer

Blondin et. al. [2020]: Remove input restriction at cost of $O(\text{poly}(|\varphi|))$ states.
Conversion of Population Computers/Main Theorems

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

Population Protocol
State complexity $\mathcal{O}(|\varphi|^2)$
Speed $\mathcal{O}(n^3)$
Inputs fulfilling $n \in \Omega(|\varphi|)$
<table>
<thead>
<tr>
<th><strong>Population Computer</strong></th>
<th><strong>Population Protocol</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>State complexity $\mathcal{O}(</td>
<td>\varphi</td>
</tr>
<tr>
<td>Bounded</td>
<td>Speed $\mathcal{O}(n^3)$</td>
</tr>
<tr>
<td>Rapid</td>
<td>Inputs fulfilling $n \in \Omega(</td>
</tr>
</tbody>
</table>
**Conversion of Population Computers/Main Theorems**

<table>
<thead>
<tr>
<th>Population Computer</th>
<th>Population Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>State complexity $O(</td>
<td>\varphi</td>
</tr>
<tr>
<td>Bounded</td>
<td>Speed $O(n^3)$</td>
</tr>
<tr>
<td>Inputs fulfilling $n \in \Omega(</td>
<td>\varphi</td>
</tr>
</tbody>
</table>

Blondin et. al. [2020]: Remove input restriction at cost of $O(\text{poly}(|\varphi|))$ states.
Conversion of Population Computers/Main Theorems

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

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

Population Protocol
Conversion of Population Computers/Main Theorems

Population Computer
State complexity $O(|\varphi|)$
Bounded

Population Protocol
State complexity $O(|\varphi|^2)$
Speed $O(n^3)$
Inputs fulfilling $n \in \Omega(|\varphi|)$

Population Computer
State complexity $O(|\varphi|)$
Rapid

Population Protocol
State complexity $O(|\varphi|)$

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

13 / 14
### Conversion of Population Computers/Main Theorems

<table>
<thead>
<tr>
<th>Population Computer</th>
<th>Population Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>State complexity</strong> $\mathcal{O}(</td>
<td>\varphi</td>
</tr>
<tr>
<td><strong>Bounded</strong></td>
<td><strong>Speed</strong> $\mathcal{O}(n^3)$</td>
</tr>
<tr>
<td></td>
<td><strong>Inputs fulfilling</strong> $n \in \Omega(</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Population Computer</th>
<th>Population Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>State complexity</strong> $\mathcal{O}(</td>
<td>\varphi</td>
</tr>
<tr>
<td><strong>Rapid</strong></td>
<td><strong>Speed</strong> $\mathcal{O}(n^2)$</td>
</tr>
</tbody>
</table>

Blondin et. al. [2020]: Remove input restriction at cost of $\mathcal{O}(\text{poly}(|\varphi|))$ states.
Conversion of Population Computers/Main Theorems

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

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

Population Protocol
State complexity $\mathcal{O}(|\varphi|)$
Speed $\mathcal{O}(n^2)$
Inputs fulfilling $n \in \Omega(|\varphi|)$
Blondin et. al. [2020]: Remove input restriction at cost of $O(\text{poly}(|\varphi|))$ states.
Thank you for your attention!

- This paper

\[
\begin{array}{c|c|c}
\text{time} & \phi & \text{#states} \\
\hline
2^n & \phi^{1/4} & \text{poly}(\phi) \\
\hline
n^2 \log n & \phi & 2\phi \\
n^2 & \text{this paper} & [2016] \\
\end{array}
\]

- [2016]
- [2008]
- [2004]
- [2020]