Automatic Verification of non-silent Population Protocols
Master’s Thesis

Martin Helfrich

Technical University of Munich

September 2019
Population Protocols

Model of distributed computation
→ to study systems of identical and anonymous agents:
  - identical
  - anonymous
  - passively mobile
  - tiny computational resources
(e.g. sensor networks or chemical systems)
Population Protocols
Example

Flock of Birds:

Question: \( \geq 4 \)  
Goal: Lasting Consensus
Population Protocols

**Definition (Population Protocol)**

A *population protocol* is a tuple $\mathcal{P} = (Q, T, \Sigma, I, O)$ such that

- $Q$ is a finite set of *states*,
- $T \subseteq \bigcup_{2 \leq i \leq |Q|} Q^i \times Q^i$ is a set of *transitions*,
- $\Sigma$ is a non-empty finite input *alphabet*,
- $I : \Sigma \rightarrow Q$ is the *input function* and
- $O : Q \rightarrow \{0, 1\}$ is the *output function*.

**Definition (Configuration)**

A *configuration* of population protocol $\mathcal{P} = (Q, T, \Sigma, I, O)$ is a multiset $C \in \mathbb{N}^Q$ where $C(q)$ describes the number of agents in state $q \in Q$.

The *output* of configuration $C$ is

$$O(C) = \begin{cases} 
   b \in \{0, 1\} & \text{if for all states } C(q) > 0 \Rightarrow O(q) = b \\
   \bot & \text{otherwise}
\end{cases}$$
1. **input:**
   \[ x \in \mathbb{N}^\Sigma \]
   \[ \downarrow \text{input function } I \]

2. **initial configuration:**
   \[ C_0 \]
   \[ \downarrow \text{transitions } \mathcal{T} \]

3. **fair\(^1\) execution:**
   \[ \sigma \overset{\text{def}}{=} C_0 \xrightarrow{t_1} C_1 \xrightarrow{t_2} C_2 \rightarrow \cdots \]

**\(P\) computes** the predicate \(\varphi : \mathbb{N}^\Sigma \rightarrow \{0, 1\}\), if for all inputs \(x \in \mathbb{N}^\Sigma\) and corresponding fair executions \(C_0 \xrightarrow{t_1} C_1 \xrightarrow{t_2} C_2 \rightarrow \cdots\) we reach the correct **lasting consensus**:

\[ \exists i \in \mathbb{N} : \varphi(x) = \mathcal{O}(C_i) = \mathcal{O}(C_{i+1}) = \cdots \]

\(^1\)A fair execution cannot avoid configurations forever.
Population Protocols
Example

Flock of Birds:

\[ Q \overset{\text{def}}{=} \{0, 1, 2, 3, 4\} \]
\[ T \overset{\text{def}}{=} \{p, q \rightarrow \min(p+q, 4), 0 \mid p, q \in Q\} \]
\[ T = T \cup \{p, 4 \rightarrow 4, 4 \mid p \in Q\} \]
\[ \Sigma \overset{\text{def}}{=} \{\text{sick, healthy}\} \]
\[ I(x) \overset{\text{def}}{=} \begin{cases} 1 & \text{if } x = \text{sick} \\ 0 & \text{if } x = \text{healthy} \end{cases} \]
\[ O(q) \overset{\text{def}}{=} \begin{cases} 1 & \text{if } q = 4 \\ 0 & \text{otherwise} \end{cases} \]

Question: \( \text{\# sick birds} \geq 4 \)
Population Protocols
Correctness Problem

Question:
Is a given protocol correct?
→ TOWER-hard [1] [2]

Goal: Automatic Verification
→ need lower complexity!
→ Blondin et al. [3]:
  (incomplete) approach for silent protocols
→ Peregrine

Definition (Silent Population Protocol)
A population protocol is silent if for every fair execution $C_0 \rightarrow C_1 \rightarrow \cdots$ there is a $i \in \mathbb{N}$ such that:

$$C_i = C_{i+1} = C_{i+2} = \cdots$$
Automatic Verification of non-silent Population Protocols

Termination Behaviour

silent protocols
→ reach terminal configuration
→ all transitions disabled
→ easy description / test

vs

non-silent protocols
→ reach lasting consensus
BUT: How to describe "lasting"?
→ harder!

Idea: Group configurations into (infinite) sets
→ Describe all fair executions at once!
Directed Acyclic Graph (DAG) of stages such that:

1. Stages are inductive sets of configurations. *i.e. "can't leave"
2. Initial configurations are part of some stage.
3. Non-terminal stage: Executions will enter substage.
4. Terminal stage: correct consensus
Stage Graphs

Stage graphs are certificates for properties of the form:

$$\varphi_{pre} \Rightarrow FG \varphi_{post}$$

"If you start in a configuration that satisfies $\varphi_{pre}$, then you will eventually satisfy $\varphi_{post}$ forever."

Theorem

Let $\Lambda$ be a predicate. For $b \in \{0, 1\}$ let

$$\varphi_{init,b}(C) \overset{\text{def}}{=} \exists X \in \mathbb{N}^\Sigma : (\Lambda(X) = b) \land (I(X) = C)$$

$$\varphi_{out,b}(C) \overset{\text{def}}{=} (O(C) = b).$$

A population protocol $\mathcal{P}$ has a $(\varphi_{init,0}, \varphi_{out,0})$-stage-graph and a $(\varphi_{init,1}, \varphi_{out,1})$-stage-graph if and only if it computes the predicate $\Lambda$.

$\Rightarrow$ sound and complete
Proof.

"⇒":
1. Executions can’t leave stages.
2. All executions start some stage.
3. Non-terminal & Fairness ⇒ "enter" substage
4. Terminal ⇒ correct consensus

"⇐": As protocol computes Λ, there are the needed stage graphs, each with 2 stages:
- Initial stage: all reachable configurations
- Terminal stage: all configurations with the correct lasting consensus
Idea: Protocols designed to work in stages → correspond to non-reversible change in configuration:

- "death" of a transition
  Example: $t$ and $u$ are dead
  i.e. "$t$ and $u$ can’t be enabled anymore."

- a state becomes "deserted"
  Example: $q$ is deserted
  i.e. "$q$ can’t be populated anymore."

→ automatically find such stages
Stage \( S = (T_{\text{dead}}, Q_{\text{deserted}}) \) where

- \( T_{\text{dead}} \subseteq T \) is the set of dead transitions.
- \( Q_{\text{deserted}} \subseteq Q \) is the set of deserted states.

Configuration \( C \) is in stage \( S \) if

1. there is a configuration \( C_0 \models \varphi_{\text{pre}} \) such that \( C_0 \xrightarrow{*} C \), and
2. \( T_{\text{dead}} \) are dead, and
3. \( Q_{\text{deserted}} \) are deserted.
Computing Stage Graphs

Algorithm

\begin{algorithm}
\begin{algorithmic}
\State \textbf{input}: protocol \( \mathcal{P} = (Q, T, \Sigma, I, O) \)
\State Presburger predicate \( \varphi_{\text{pre}} \)
\State Presburger predicate \( \varphi_{\text{post}} \)
\State \( S_0 := (\emptyset, \emptyset) \)
\State \textbf{Unprocessed} := \{S_0\}
\While{\text{\textbf{|Unprocessed|} > 0}}
\State \( S := \text{\textbf{Unprocessed}}.\text{pop}() \)
\If{\text{Substages}(\mathcal{P}, \varphi_{\text{pre}}, \varphi_{\text{post}}, S) \text{ \textbf{fails}}}
\State then \text{\textbf{abort}}
\Else\quad \textbf{Unprocessed} := \text{\textbf{Unprocessed}} \cup \text{Substages}(\mathcal{P}, \varphi_{\text{pre}}, \varphi_{\text{post}}, S)\end{algorithmic}
\end{algorithm}
Computing Stage Graphs
Algorithm: Find new substages

**input:** protocol \( P = (Q, \mathcal{T}, \Sigma, \mathcal{I}, \mathcal{O}) \)

Presburger predicate \( \varphi_{pre} \)

Presburger predicate \( \varphi_{post} \)

stage \( S = (T_{\text{dead}}, Q_{\text{deserted}}) \)

**if** Terminal\((P, \varphi_{pre}, S, \varphi_{post})\)  
**return** \( \emptyset \)

\( T'_{\text{dead}} := \text{EventuallyDead}(P, \varphi_{pre}, S) \)

**if** \( T'_{\text{dead}} \supset T_{\text{dead}} \)  
**return** \( \{(T'_{\text{dead}}, Q_{\text{deserted}})\} \)

**if** Split\((P, \varphi_{pre}, S)\) **fails**  
**then** abort  
**else** return Split\((P, \varphi_{pre}, S)\)

**Parametric in 3 auxiliary functions**

**Terminal:**
Try to prove: \( S \) is terminal

**EventuallyDead:**
Find "eventually dead" transitions

**Split:**
Split \( S \) in substages with more deserted states.
Need to decide: $C \in S$.

**Problem:** "reachable", "dead" and "deserted" are non-trivial

**Idea:** Overapproximate!

1. "reachable": use potential reachability [3]
   - flow equation & siphons & traps
2. "dead": use "disabled"\(^2\)
3. "deserted": use "empty"

**Implementation:** Use Z3 to check

$$\forall C : C \models \neg \text{PotInStage}(P, \varphi_{pre}, S) \lor \varphi_{post}$$

\(^2\)We also use tighter approximations using the backwards coverability algorithm.
Goal: Find transitions that will eventually become dead from every configuration $C \in S$.

Implementations:

- **Ranking function:**
  $\rightarrow$ imply eventual death of some transition

- **Layered termination:** [3]
  find "layer" $L \subseteq \mathcal{T}$ and ranking function such that
  - $L$ will eventually be disabled, and
  - $\text{Disabled}(L) \Rightarrow \text{Dead}(L)$

- **Combined:**
  use ranking functions and layered termination
Computing Stage Graphs

Split

Goal: Split stage into substages with more deserted states. (i.e. "case distinction")

Idea: empty siphon $\Rightarrow$ deserted

$\Rightarrow$ find set of siphons $R$ such that

$$\forall C : C \models \neg \text{PotInStage}(P, \varphi_{pre}, S) \lor \bigvee_{R_i \in R} \text{empty}(R_i)$$

Implementation: Guess siphons using Z3.
Computing Stage Graphs

Example

**Majority Protocol**

```
“A ≤ B”
```

- \( t_{AB} : AB \rightarrow ab \)
- \( t_{Ab} : Ab \rightarrow Aa \)
- \( t_{Ba} : Ba \rightarrow Bb \)
- \( t_{ab} : ab \rightarrow bb \)

**Consensus**

- \( \Rightarrow \text{Consensus true} \)
- \( \Rightarrow \text{Consensus false} \)
Computing Stage Graphs
Results

<table>
<thead>
<tr>
<th>protocol</th>
<th>predicate</th>
<th>silent</th>
<th></th>
<th>Q</th>
<th></th>
<th></th>
<th>T</th>
<th></th>
<th>proven</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority</td>
<td>( A \leq B )</td>
<td>yes</td>
<td>4</td>
<td>4</td>
<td>yes</td>
<td>&lt; 1s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A&amp;C(11,9)</td>
<td>( A \leq B )</td>
<td>no</td>
<td>28</td>
<td>406</td>
<td>yes</td>
<td>700s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flock-of-Birds</td>
<td>( X \geq 60 )</td>
<td>yes</td>
<td>61</td>
<td>1891</td>
<td>yes</td>
<td>328s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>succinct FoB.</td>
<td>( X \geq 2^{35} - 1 )</td>
<td>yes</td>
<td>70</td>
<td>1294</td>
<td>yes</td>
<td>334s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>suc. rev. FoB.</td>
<td>( X \geq 63 )</td>
<td>no</td>
<td>12</td>
<td>31</td>
<td>yes</td>
<td>40s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remainder</td>
<td>( \sum_{1 \leq i &lt; 20} i \cdot x_i \equiv 0 \mod 20 )</td>
<td>yes</td>
<td>22</td>
<td>250</td>
<td>yes</td>
<td>565s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>succinct Rem.</td>
<td>( \sum_{1 \leq i &lt; 63} i \cdot x_i \equiv 0 \mod 63 )</td>
<td>no</td>
<td>16</td>
<td>41</td>
<td>yes</td>
<td>75s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Threshold</td>
<td>(-2a - b + c + 2d &lt; 3)</td>
<td>yes</td>
<td>36</td>
<td>495</td>
<td>yes</td>
<td>32s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>succinct Thr.</td>
<td>(-2a - b + c + 2d &lt; 63)</td>
<td>yes</td>
<td>20</td>
<td>66</td>
<td>yes</td>
<td>100s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table: Automatic verification of silent and non-silent protocols using stage graphs.
We can even verify leader election! (i.e. via postcondition)

| protocol            | n    | silent | |Q| | |T| | proven | time     |
|---------------------|------|--------|-----|----|------|--------|---------|
| simple              | ∞    | yes    | 2   | 1  | yes  | < 1s   |
| Israeli-Jalfon      | 70   | no     | 140 | 280| yes  | 2537s  |
| Herman              | 91   | no     | 182 | 182| no   | 203s   |
| Herman modified     | 91   | no     | 182 | 182| yes  | 2785s  |

**Table:** Automatic verification of leader election protocols for $n$ agents.
Important questions in practise:

- Correctness: ✓
- Fast: ?

But what does "fast" mean?
→ expected number of interactions
→ probabilistic model
  (i.e. "random" instead of fairness)

Apply idea of Blondin et al. [4]!
Let $n = |C_0|$.

- **Terminal:** $O(1)$
- **Split:** $O(1)$
- **EventuallyDead:**
  - layered: $O(n^n)$
  - ranking: $O(n^c)$ for some constant $c$
  - layered + ranking: $O(n^3)$
  - layered + ranking + "fast": $O(n^2 \log n)$
Termination Time

Example

Majority Protocol

“\( A \leq B \)”

\( t_{AB} : AB \rightarrow ab \)
\( t_{Ab} : Ab \rightarrow Aa \)
\( t_{Ba} : Ba \rightarrow Bb \)
\( t_{ab} : ab \rightarrow bb \)

\( S_1 \)

Dead: \{ \( t_{AB} \) \}
Deserted: \{ \( B \) \}

\( S_2 \)

Dead: \{ \( t_{AB}, t_{Ab} \) \}
Deserted: \{ \( A \) \}

\( S_3 \)

Dead: \( T \)
Deserted: \{ \( A \), \( A \) \}

\( S_4 \)

Dead: \( T \)
Deserted: \{ \( A \), \( a \) \}

\( S_5 \)

Dead: \{ \( t_{AB}, t_{Ba} \) \}
Deserted: \{ \( B \) \}

\( S_6 \)

Dead: \{ \( t_{AB}, t_{Ba}, t_{Ab} \) \}
Deserted: \{ \( B \) \}

\( S_7 \)

Dead: \( T \)
Deserted: \{ \( B \) \}

\( S_8 \)

Dead: \{ \( B, A, a \) \}
Deserted: \{ \( B, A, a \) \}

\( \Rightarrow \) Consensus \( true \)

\( S_9 \)

Dead: \{ \( B, b \) \}
Deserted: \{ \( B, b \) \}

\( \Rightarrow \) Consensus \( false \)
## Termination Time

### Results

| protocol                    | $|Q|$ | $|T|$ | bound   | time   |
|-----------------------------|-----|-----|---------|--------|
| Majority                    | 4   | 4   | $O(n^n)$ | < 1s   |
| simple leader election      | 2   | 1   | $O(n^2 \log n)$ | < 1s   |
| Flock-of-Birds (45)         | 46  | 2026| $O(n^3)$ | 307s   |
| succinct FoB (511)          | 18  | 97  | $O(n^3)$ | 2.5s   |
| suc. rev. FoB (63)          | 12  | 31  | $O(n^c)$ | 307s   |
| Remainder ($\equiv 4$)      | 6   | 18  | $O(n^2 \log n)$ | 2.8s   |
| Threshold ($< 2$)           | 28  | 301 | $O(n^3)$ | 62s    |
| A&C (7,1)                   | 10  | 55  | $O(n^2 \log n)$ | 8.3s   |
| A&C (11,10)                 | 32  | 528 | $O(n^3)$ | 550s   |

**Table:** Automatically found and proven speed bounds.
Question: Can we verify liveness?
E.g. will process 1 enter its critical section infinitely often?

Answer: No!
→ property does not have form $\varphi_{pre} \Rightarrow FG \varphi_{post}$
Overview

- \( \phi \) is a liveness property (LTL)
- \( GF(\text{enter}_1) \)
- \( FG(\neg \text{enter}_1) \)

Transform:
\[
\neg \phi \\
\neg \neg \phi \\
\text{transform} \\
LDBA \ B \\
\times \\
\text{product} \\
\mathcal{P}' \\
\text{verify:} \\
B \text{ can’t accept} \\
\checkmark \\
\text{population protocol} \\
\mathcal{P} \\
\text{product protocol}
We can verify liveness of a single process in mutex algorithms!

<table>
<thead>
<tr>
<th>Mutex algorithm</th>
<th>processes</th>
<th>proven</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple</td>
<td>400</td>
<td>yes</td>
<td>2049s</td>
</tr>
<tr>
<td>Array</td>
<td>11</td>
<td>yes</td>
<td>2284s</td>
</tr>
<tr>
<td>Burns</td>
<td>6</td>
<td>yes</td>
<td>1074s</td>
</tr>
<tr>
<td>Peterson</td>
<td>2</td>
<td>yes</td>
<td>&lt; 1s</td>
</tr>
<tr>
<td>Dijkstra</td>
<td>4</td>
<td>yes</td>
<td>3221s</td>
</tr>
<tr>
<td>Szymanski</td>
<td>3</td>
<td>yes</td>
<td>38s</td>
</tr>
<tr>
<td>Lehmann Rabin</td>
<td>10</td>
<td>yes</td>
<td>3141s</td>
</tr>
</tbody>
</table>

Table: Automatic verification of liveness of a single process in mutex algorithms.
Future Work

Verify more expressive models?

- Petri nets with inhibitor arcs
- Population protocols with broadcast
- ... 

→ automatically?

Other fairness assumptions?
References I

