

Automatic Verification of non-silent Population Protocols

Master's Thesis

Martin Helfrich

Technical University of Munich

September 2019



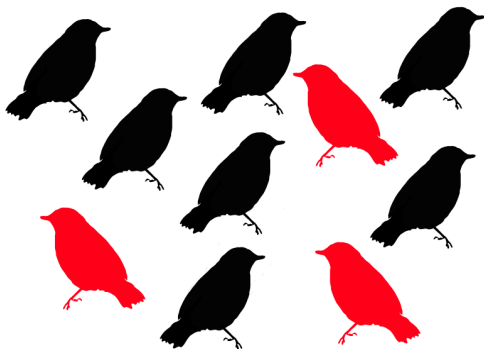
Model of distributed computation

→ to study systems of identical and anonymous *agents*:

- identical
- anonymous
- passively mobile
- tiny computational resources

(e.g. sensor networks or chemical systems)

Flock of Birds:



Question:  ≥ 4

Goal: Lasting Consensus

Definition (Population Protocol)

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

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

Definition (Configuration)

A *configuration* of population protocol $\mathcal{P} = (Q, \mathcal{T}, \Sigma, \mathcal{I}, \mathcal{O})$ 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

$$\mathcal{O}(C) = \begin{cases} b \in \{0, 1\} & \text{if for all states } C(q) > 0 \Rightarrow \mathcal{O}(q) = b \\ \perp & \text{otherwise} \end{cases}$$

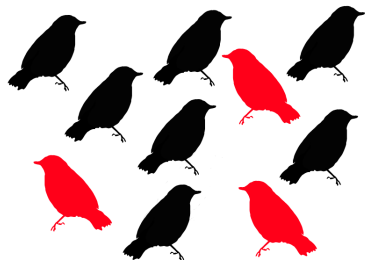
- | | |
|--------------------------------|---|
| ① input: | $x \in \mathbb{N}^\Sigma$ |
| ↓ input function \mathcal{I} | ↓ |
| ② initial configuration: | C_0 |
| ↓ transitions \mathcal{T} | ↓ |
| ③ fair ¹ execution: | $\sigma \stackrel{\text{def}}{=} C_0 \xrightarrow{t_1} C_1 \xrightarrow{t_2} C_2 \rightarrow \dots$ |

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

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

¹A fair execution cannot avoid configurations forever.

Flock of Birds:



$$Q \stackrel{\text{def}}{=} \{0, 1, 2, 3, 4\}$$

$$\mathcal{T} \stackrel{\text{def}}{=} \{p, q \rightarrow \min(p+q, 4), 0 \mid p, q \in Q\} \\ \cup \{p, 4 \rightarrow 4, 4 \mid p \in Q\}$$

$$\Sigma \stackrel{\text{def}}{=} \{sick, healthy\}$$

$$\mathcal{I}(x) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } x = sick \\ 0 & \text{if } x = healthy \end{cases}$$

$$\mathcal{O}(q) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } q = 4 \\ 0 & \text{otherwise} \end{cases}$$

Question:  ≥ 4

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 $C_0 \rightarrow C_1 \rightarrow \dots$ there is a $i \in \mathbb{N}$ such that:

$$C_i = C_{i+1} = C_{i+2} = \dots$$

Automatic Verification of non-silent Population Protocols

Termination Behaviour

silent protocols

- reach terminal configuration
- all transitions disabled
- easy description / test

vs

non-silent protocols

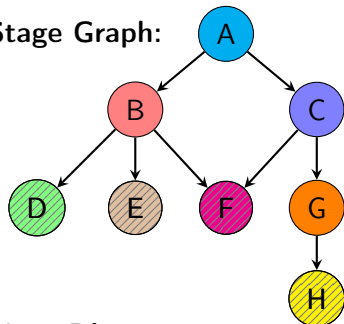
- reach lasting consensus
- BUT: How to describe "lasting"?
- harder!

Idea: Group configurations into (infinite) sets

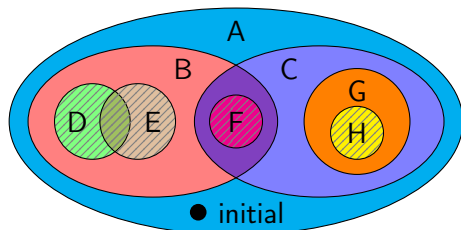
- Describe all fair executions at once!

Stage Graphs

Stage Graph:



Venn-Diagram:



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 φ_{pre} , then you will eventually satisfy φ_{post} forever."

Theorem

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

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

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

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 Λ .

\Rightarrow sound and complete

Proof.

" \Rightarrow ":

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

" \Leftarrow ": As protocol computes Λ , there are the needed stage graphs, each with 2 stages:

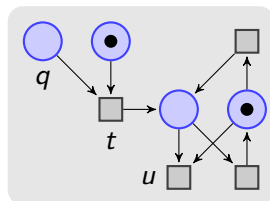
- Initial stage: all reachable configurations
- Terminal stage: all configurations with the correct lasting consensus



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 $S = (T_{dead}, Q_{deserted})$ where

- $T_{dead} \subseteq \mathcal{T}$ is the set of dead transitions.
- $Q_{deserted} \subseteq \mathcal{Q}$ is the set of deserted states.

Configuration C is in stage S if

- 1 there is a configuration $C_0 \models \varphi_{pre}$ such that $C_0 \xrightarrow{*} C$, and
- 2 T_{dead} are dead, and
- 3 $Q_{deserted}$ are deserted.

Computing Stage Graphs

Algorithm

```
input: protocol  $\mathcal{P} = (Q, \mathcal{T}, \Sigma, \mathcal{I}, \mathcal{O})$   
       Presburger predicate  $\varphi_{pre}$   
       Presburger predicate  $\varphi_{post}$ 
```

```
 $S_0 := (\emptyset, \emptyset)$ 
```

```
 $Unprocessed := \{S_0\}$ 
```

```
while  $|Unprocessed| > 0$ 
```

```
     $S := Unprocessed.pop()$ 
```

```
    if  $Substages(\mathcal{P}, \varphi_{pre}, \varphi_{post}, S)$  fails  
        then abort
```

```
    else
```

```
         $Unprocessed := Unprocessed \cup Substages(\mathcal{P}, \varphi_{pre}, \varphi_{post}, S)$ 
```

Computing Stage Graphs

Algorithm: Find new substages

```
input: protocol  $\mathcal{P} = (Q, \mathcal{T}, \Sigma, \mathcal{I}, \mathcal{O})$   
Presburger predicate  $\varphi_{pre}$   
Presburger predicate  $\varphi_{post}$   
stage  $S = (T_{dead}, Q_{deserted})$ 
```

```
if Terminal( $\mathcal{P}, \varphi_{pre}, S, \varphi_{post}$ )  
    return  $\emptyset$ 
```

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

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

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

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

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

²We 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:**
→ 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
 - $Disabled(L) \Rightarrow Dead(L)$
- **Combined:**
use ranking functions and layered termination

Goal: Split stage into substages with more deserted states.

(i.e. "case distinction")

Idea: empty siphon \Rightarrow deserted

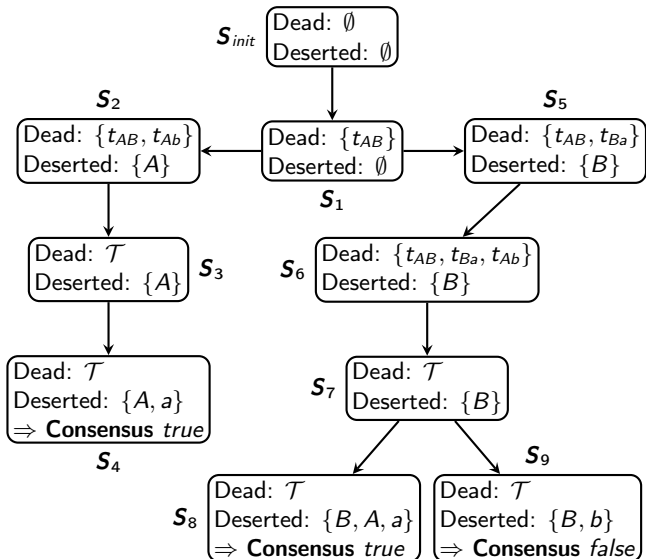
\rightarrow find set of siphons R such that

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

Implementation: Guess siphons using Z3.

Computing Stage Graphs

Example



Majority Protocol

" $A \leq B$ "

$t_{AB} : AB \rightarrow ab$

$t_{Ab} : Ab \rightarrow Aa$

$t_{Ba} : Ba \rightarrow Bb$

$t_{ab} : ab \rightarrow bb$

Computing Stage Graphs

Results

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

Table: Automatic verification of silent and non-silent protocols using stage graphs.

Computing Stage Graphs

Results: Leader election

We can even verify leader election! (i.e. via postcondition)

protocol	n	silent	$ Q $	$ \mathcal{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)

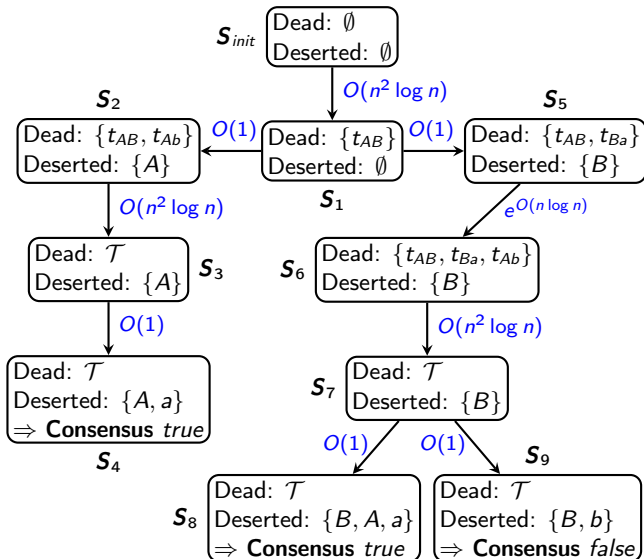
Apply idea of Blondin *et al.* [4]!

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

" $A \leq B$ "

$t_{AB} : AB \rightarrow ab$

$t_{Ab} : Ab \rightarrow Aa$

$t_{Ba} : Ba \rightarrow Bb$

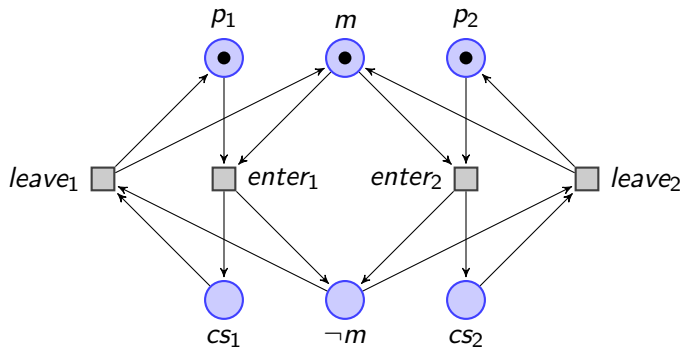
$t_{ab} : ab \rightarrow bb$

Termination Time

Results

protocol	$ Q $	$ \mathcal{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.

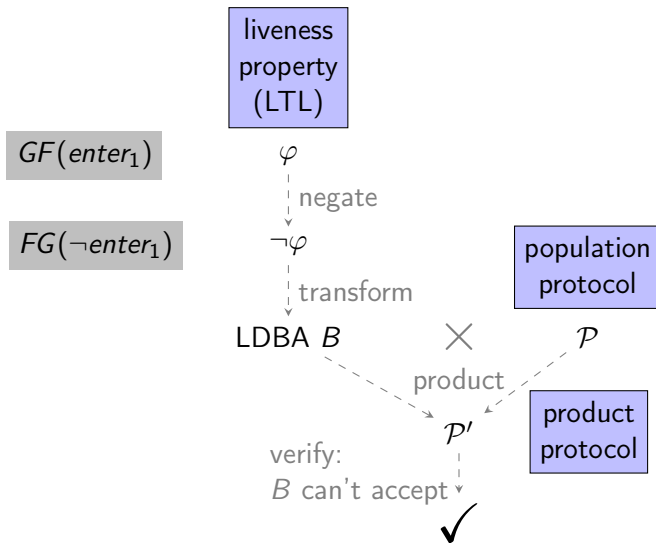


Question: Can we verify liveness?

E.g. will process 1 enter its critical section infinitely often?

Answer: No!

→ property does not have form $\varphi_{pre} \Rightarrow FG\varphi_{post}$



We can verify liveness of a single process in mutex algorithms!

Mutex algorithm	processes	proven	time
Simple	400	yes	2049s
Array	11	yes	2284s
Burns	6	yes	1074s
Peterson	2	yes	< 1s
Dijkstra	4	yes	3221s
Szymanski	3	yes	38s
Lehmann Rabin	10	yes	3141s

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

Verify more expressive models?

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

→ automatically?

Other fairness assumptions?

- [1] Javier Esparza, Pierre Ganty, Jérôme Leroux, and Rupak Majumdar.
Verification of population protocols.
Acta Informatica, 54(2):191–215, 03 2017.
- [2] Wojciech Czerwiński, Sławomir Lasota, Ranko Lazić, Jérôme Leroux, and Filip Mazowiecki.
The reachability problem for petri nets is not elementary.
In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 24–33. ACM, 2019.
- [3] Michael Blondin, Javier Esparza, Stefan Jaax, and Philipp J Meyer.
Towards efficient verification of population protocols.
In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 423–430. ACM, 2017.

- [4] Michael Blondin, Javier Esparza, and Antonín Kucera.
Automatic analysis of expected termination time for population protocols.
In *Proc. 29th International Conference on Concurrency Theory (CONCUR)*, pages 33:1–33:16, 2018.