Towards efficient Verification of Population Protocols

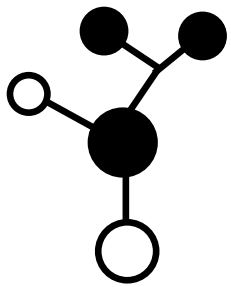
Michael Blondin, Javier Esparza, Philipp J. Meyer
Stefan Jaax

TU München

Population Protocols

Population protocols (Angluin et al., 2004) are a model of distributed computation of **anonymous finite-state** agents.

Can be used to model networks of passively mobile sensors and chemical reaction networks.



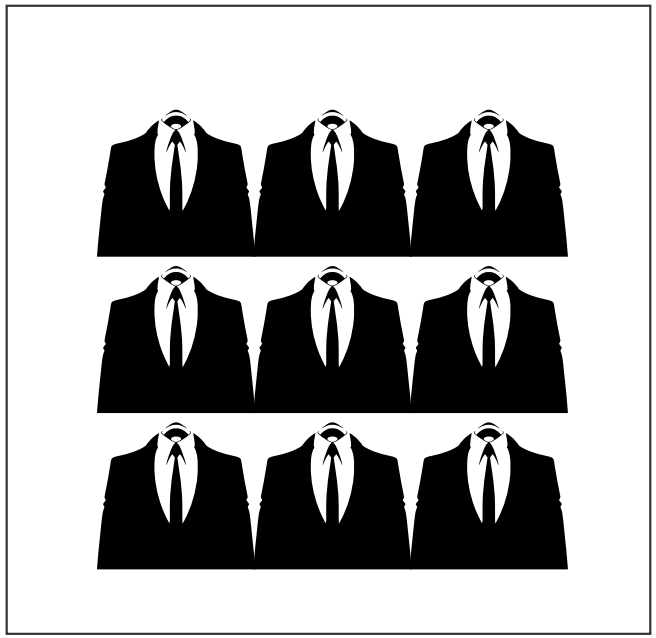
Our Contribution

Correct implementation of population protocols is non-trivial + automatic verification is **very hard**.

Our contribution: A new fully expressive subclass of population protocols suitable for automatic verification.

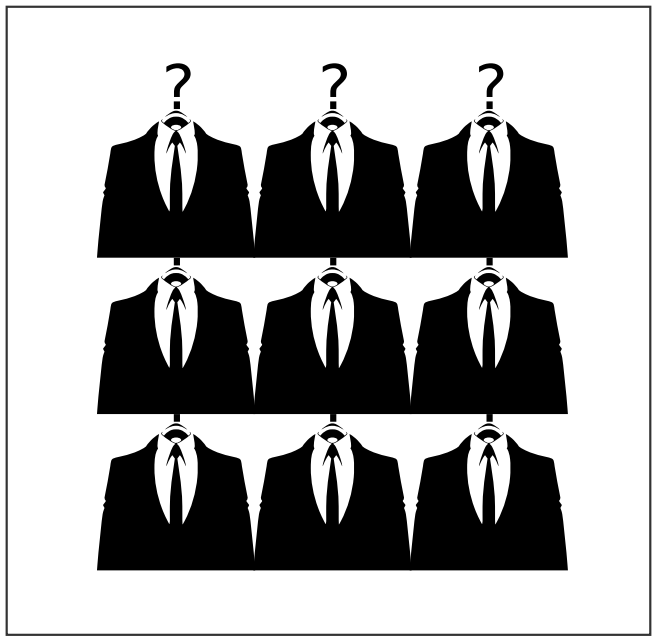
Overview

- Computation in a finite **population of identical mobile agents**.



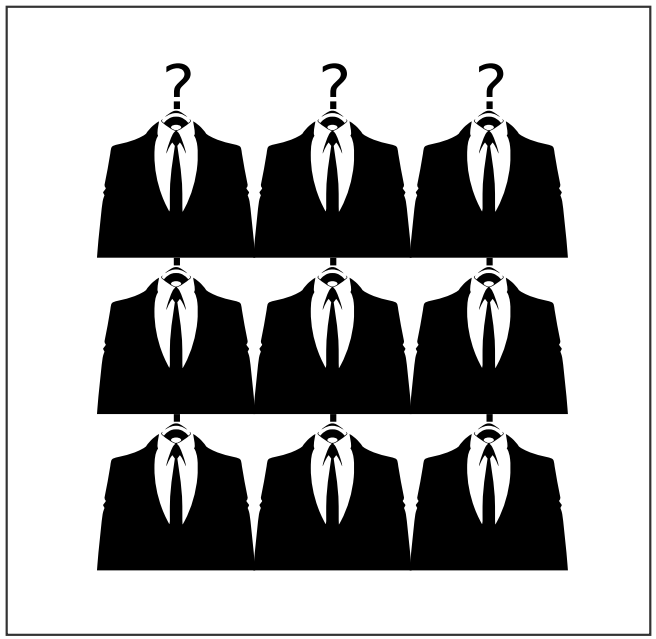
Overview

- Computation in a finite **population of identical mobile agents**.
- Agents are **anonymous**: they cannot identify each other.



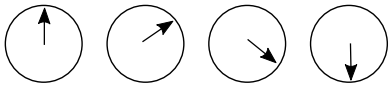
Overview

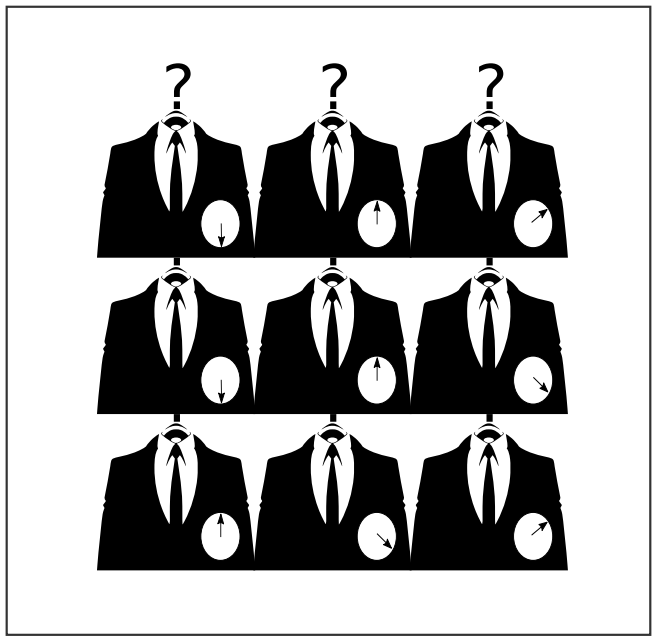
- Computation in a finite **population of identical mobile agents**.
- Agents are **anonymous**: they cannot identify each other.
- Number of agents is arbitrary, but fixed.



Overview

- Computation in a finite **population of identical mobile agents**.
- Agents are **anonymous**: they cannot identify each other.
- Number of agents is arbitrary, but fixed.
- Very **few resources**! Number of states is finite:

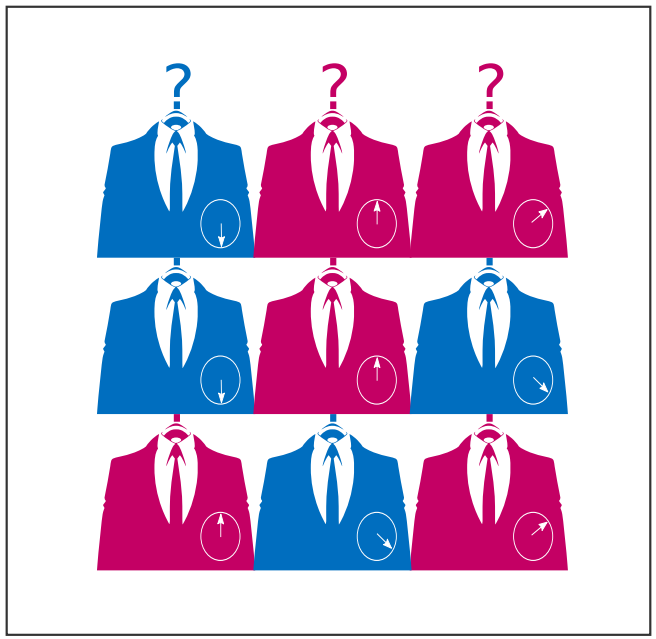




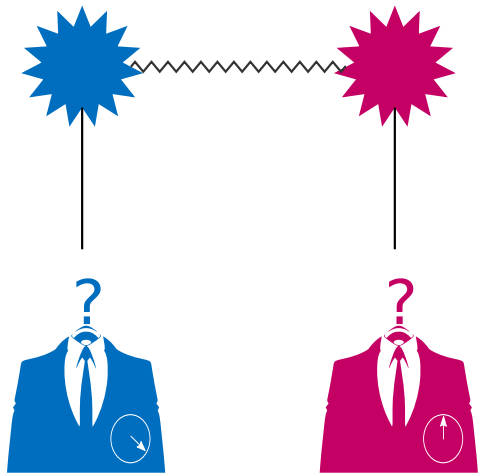
Overview

- Computation in a finite **population of identical mobile agents**.
- Agents are **anonymous**: they cannot identify each other.
- Number of agents is arbitrary, but fixed.
- Very **few resources**! Number of states is finite:

- States map to opinions (true/false).



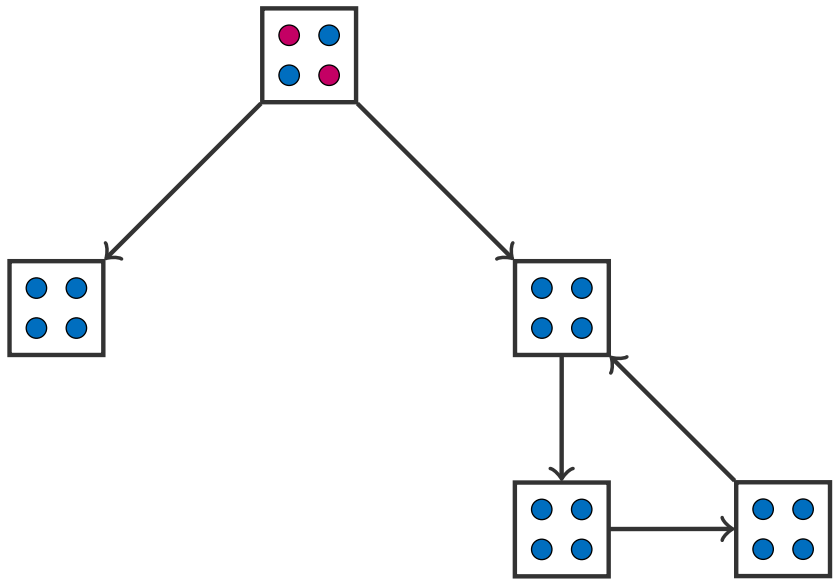
Computations in Population Protocols



Pairwise asynchronous interactions lead to state changes.

Effect of interaction is specified by a transition relation.

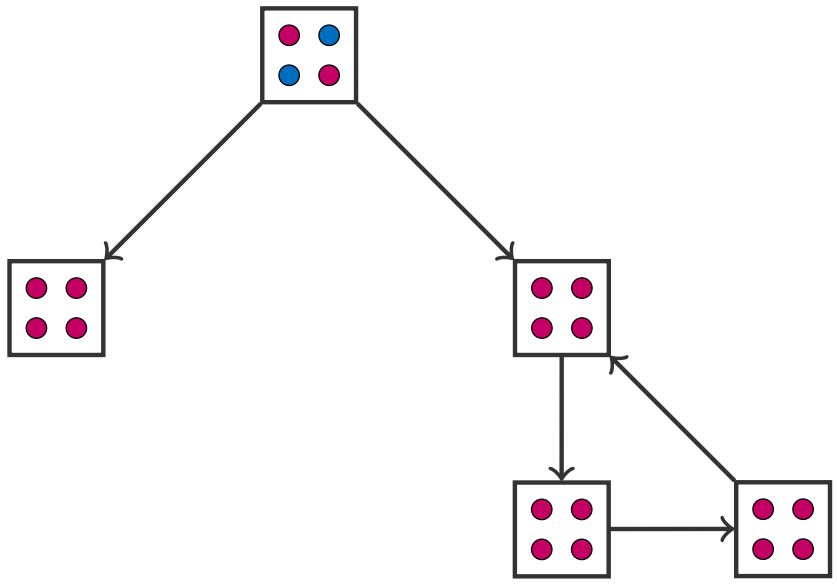
Well-Specified Population Protocols



Goal: stabilize to lasting consensus.

Final opinion must be unique for every initial configuration.

Well-Specified Population Protocols



Goal: stabilize to lasting consensus.

Final opinion must be unique for every initial configuration.

Why well-specification matters

Well-specified protocols compute predicates: Every initial configuration can be mapped to the value of the unique consensus.

Well-Specification

Checking Well-specification for fixed population size:

- PAT: LTL model checker with fairness

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

- bp-ver: graph exploration algorithms + parallelism

  (Chatzigiannakis, Michail and Spirakis SSS'10)

- Protocols to counter machines verified with PRISM/Spin

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

Well-Specification

Checking Well-specification for fixed population size:

- PAT: LTL model checker with fairness

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

- bp-ver: graph exploration algorithms + parallelism

  (Chatzigiannakis, Michail and Spirakis SSS'10)

- Protocols to counter machines verified with PRISM/Spin

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

$\leq$ 9 states, 28 transitions

Well-Specification

Checking Well-specification for fixed population size:

- PAT: LTL model checker with fairness

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

- bp-ver: graph exploration algorithms + parallelism

  (Chatzigiannakis, Michail and Spirakis SSS'10)

- Protocols to counter machines verified with PRISM/Spin

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

Possible to verify all sizes?

Well-Specification Problem

Well-Specification Problem

Given a protocol as input, answer whether it is well-specified.

Well-Specification Problem

Well-Specification Problem

Given a protocol as input, answer whether it is well-specified.

- The Well-Specification Problem was shown by Esparza et. al in 2015 to be decidable, but EXPSPACE-hard.
- Reachability Problem of Petri nets is polynomially reducible to Well-Specification Problem.
- It is unknown whether the Reachability Problem is primitive recursive!

Our Approach

Find subclass of well-specified protocols that

- Has an automatic membership test of reasonable complexity.
- Captures the entire expressive power of population protocols.

Our class

Our Class = Layered Termination + Strong Consensus

Layered Termination

A terminal configuration is always reachable due to universal termination strategy (of a certain form).

Strong Consensus

Terminal configurations *pseudo-reachable* from a given initial configuration form unique consensus.

Our class

Our Class = Layered Termination + Strong Consensus

Layered Termination

A terminal configuration is always reachable due to universal termination strategy (of a certain form).

NP

Strong Consensus

Terminal configurations *pseudo-reachable* from a given initial configuration form unique consensus.

co-NP

Peregrine

- Peregrine: Haskell + SMT solver Z3

  gitlab.lrz.de/i7/peregrine

- Peregrine reads a protocol and constructs two sets of constraints:
  - The first is satisfiable iff. Layered Termination holds.
  - The second is unsatisfiable iff. Strong Consensus holds.

Experimental Results

Experiments were performed on a machine equipped with an Intel Core i7-4810MQ CPU and 16 GB of RAM.

- For parameterized families of protocols, we gradually increased the parameter value until we reached a timeout.
- The timeout was set to 1 hour.

Experimental Results

Protocol	Predicate	$\lvert Q\rvert$	$\lvert T\rvert$	Time[s]
Majority [1]	$x \geq y$	4	4	0.1
Approx. Majority [2]	Not well-specified	3	4	0.1
Broadcast [3]	$x_1 \vee \ldots \vee x_n$	2	1	0.1
Threshold [4]	$\Sigma_i \alpha_i x_i < c \colon \lvert\alpha_i\rvert \leq 9$	76	2148	2375.9
Remainder [5]	$\Sigma_i \alpha_i x_i \bmod 70 = 0$	72	2555	3176.5
Flock of birds [6]	$x \geq 50$	51	1275	181.6
Flock of birds [7]	$x \geq 325$	326	649	3470.8
Prime-Flock of birds	$x \geq 10^7$	37	155	18.91
Log-Flock of birds	$x \geq 10^{34}$	155	2693	1918.67

[1] Draief et al., 2012 [2] Angluin et al., 2007 [3] Clément et al., 2011
[4][5] Angluin et al., 2006 [6] Chatzigiannakis et al., 2010 [7] Clément et al., 2011

Experimental Results

Protocol	Predicate	$\lvert Q\rvert$	$\lvert T\rvert$	Time[s]
Majority [1]	$x \geq y$	4	4	0.1
Approx. Majority [2]	Not well-specified	3	4	0.1
Broadcast [3]	$x_1 \vee \ldots \vee x_n$	2	1	0.1
Threshold [4]	$\Sigma_i \alpha_i x_i < c : \lvert\alpha_i\rvert \leq 9$	76	2148	2375.9
Remainder [5]	$\Sigma_i \alpha_i x_i \bmod 70 = 0$	72	2555	3176.5
Flock of birds [6]	$x \geq 50$	51	1275	181.6
Flock of birds [7]	$x \geq 325$	326	649	3470.8
Prime-Flock of birds	$x \geq 10^7$	37	155	18.91
Log-Flock of birds	$x \geq 10^{34}$	155	2693	1918.67

[1] Draief et al., 2012 [2] Angluin et al., 2007 [3] Clément et al., 2011
[4][5] Angluin et al., 2006 [6] Chatzigiannakis et al., 2010 [7] Clément et al., 2011

Peregrine and Correctness

Check correctness: add additional constraint in SMT-solver to check whether consensus always has the right value.

Peregrine and Correctness

Check correctness: add additional constraint in SMT-solver to check whether consensus always has the right value.

Peregrine successfully verified all protocols in our benchmark!

Peregrine and Correctness

Check correctness: add additional constraint in SMT-solver to check whether consensus always has the right value.

Peregrine successfully verified all protocols in our benchmark!

Verification at least as fast as test for well-specification in most protocols.

Concluding Remarks

- We introduced a class of population protocols with tractable verification problem.
- No loss in expressive power!
- Our approach is automatic and completely parametric. Other automatic approaches only consider populations up to a fixed size!

Future Research

In the future:

- Transform any well-specified protocol into our class.
- Can we achieve automatic verification for *parameterized families of protocols*?

Thank you

Thank you for your attention!

gitlab.lrz.de/i7/peregrine