Verification of Population Protocols

Javier Esparza
Technical University of Munich

Joint work with Pierre Ganty, Jérôme Leroux,
and Rupak Majumdar

Deaf Black Ninjas in the Dark

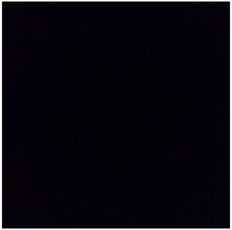
- Deaf Black Ninjas meet at a Zen garden in the dark

Deaf Black Ninjas in the Dark

- Deaf Black Ninjas meet at a Zen garden in the dark
- They must decide by majority to attack or not ("don't attack" if tie)

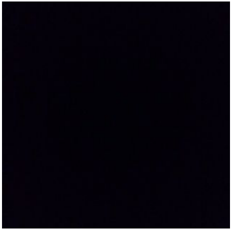
Deaf Black Ninjas in the Dark

- Deaf Black Ninjas meet at a Zen garden in the dark
- They must decide by majority to attack or not ("don't attack" if tie)



Deaf Black Ninjas in the Dark

- Deaf Black Ninjas meet at a Zen garden in the dark
- They must decide by majority to attack or not ("don't attack" if tie)
- How can they conduct the vote?



Deaf Black Ninjas in the Dark

- Ninjas randomly wander around the garden, interacting when they bump into each other

Deaf Black Ninjas in the Dark

- Ninjas randomly wander around the garden, interacting when they bump into each other
- Each Ninja stores their current estimation of the final outcome of the vote (Yes or No). Additionally, it is Active or Passive.

Deaf Black Ninjas in the Dark

- Ninjas randomly wander around the garden, interacting when they bump into each other
- Each Ninja stores their current estimation of the final outcome of the vote (Yes or No). Additionally, it is Active or Passive.
- Initially all Ninjas are Active, and their initial estimation is their own vote

Deaf Black Ninjas in the Dark

- Ninjas randomly wander around the garden, interacting when they bump into each other
- Each Ninja stores their current estimation of the final outcome of the vote (Yes or No). Additionally, it is Active or Passive.
- Initially all Ninjas are Active, and their initial estimation is their own vote
- Ninjas follow this protocol:

$$(YA, NA) \rightarrow (NP, NP) \quad \text{(opposite votes "cancel")}$$

$$\begin{array}{lcll} (YA, NP) & \rightarrow & (YA, YP) & \text{(active "survivors" tell} \\ (NA, YP) & \rightarrow & (NA, NP) & \quad \text{outcome to passive Ninjas)} \end{array}$$

$$(NP, YP) \rightarrow (NP, NP) \quad \text{(to deal with ties)}$$

Population protocols (PP)

Theoretical model for distributed computation

Proposed in 2004 by Angluin *et al.*

Designed to model collections of

identical, finite-state, and mobile agents

like

Population protocols (PP)

Theoretical model for distributed computation

Proposed in 2004 by Angluin *et al.*

Designed to model collections of

identical, finite-state, and mobile agents

like

Population protocols (PP)

Theoretical model for distributed computation

Proposed in 2004 by Angluin *et al.*

Designed to model collections of

identical, finite-state, and mobile agents

like

- ad-hoc networks of mobile sensors

Population protocols (PP)

Theoretical model for distributed computation

Proposed in 2004 by Angluin *et al.*

Designed to model collections of

identical, finite-state, and mobile agents

like

- ad-hoc networks of mobile sensors
- “soups” of interacting molecules

Population protocols (PP)

Theoretical model for distributed computation

Proposed in 2004 by Angluin *et al.*

Designed to model collections of

identical, finite-state, and mobile agents

like

- ad-hoc networks of mobile sensors
- “soups” of interacting molecules
- people in social networks

Population protocols (PP)

Theoretical model for distributed computation

Proposed in 2004 by Angluin *et al.*

Designed to model collections of

identical, finite-state, and mobile agents

like

- ad-hoc networks of mobile sensors
- "soups" of interacting molecules
- people in social networks
- ... and Ninjas

Syntax

A PP-scheme is a pair (Q, Δ), where

- Q is a finite set of states, and
- $\Delta \subseteq (Q \times Q) \times (Q \times Q)$ is a set of interactions.

Syntax

A PP-scheme is a pair (Q, Δ), where

- Q is a finite set of states, and
- $\Delta \subseteq (Q \times Q) \times (Q \times Q)$ is a set of interactions.

Intuition:

if $(q_1, q_2) \mapsto (q_1', q_2') \in \Delta$ **and**
two agents in states q_1 and q_2 "meet",

then the agents can interact and
change their states to q_1', q_2'.

Assumption: at least one interaction for each pair of states (possibly $(q_1, q_2) \mapsto (q_1, q_2)$)

Semantics

Configuration: mapping $C\colon Q \to \mathbb{N}$, where $C(q)$ is the current number of agents in state q.

q_1	q_2	q_3	q_4
2	1	0	3

Semantics

Configuration: mapping $C\colon Q \to \mathbb{N}$, where $C(q)$ is the current number of agents in state q.

q_1	q_2	q_3	q_4
2	1	0	3

$$(q_1, q_2) \mapsto (q_3, q_4)$$

Semantics

Configuration: mapping $C\colon Q \to \mathbb{N}$, where $C(q)$ is the current number of agents in state q.

$$
\begin{array}{cccccccccc}
q_1 & q_2 & q_3 & q_4 & & q_1 & q_2 & q_3 & q_4 \\
2 & 1 & 0 & 3 & \longrightarrow & 1 & 0 & 1 & 4
\end{array}
$$

$$(q_1, q_2) \mapsto (q_3, q_4)$$

Semantics

Configuration: mapping $C\colon Q \to \mathbb{N}$, where $C(q)$ is the current number of agents in state q.

$$\begin{array}{cccccccccc} q_1 & q_2 & q_3 & q_4 & & q_1 & q_2 & q_3 & q_4 \\ 2 & 1 & 0 & 3 & \longrightarrow & 1 & 0 & 1 & 4 \end{array}$$

$$(q_1, q_2) \mapsto (q_3, q_4)$$

If several steps are possible, a random scheduler chooses one (fixed nonzero prob. for each pair)

Semantics

Configuration: mapping $C\colon Q \to \mathbb{N}$, where $C(q)$ is the current number of agents in state q.

$$\begin{array}{cccc} q_1 & q_2 & q_3 & q_4 \\ 2 & 1 & 0 & 3 \end{array} \longrightarrow \begin{array}{cccc} q_1 & q_2 & q_3 & q_4 \\ 1 & 0 & 1 & 4 \end{array}$$

$$(q_1, q_2) \mapsto (q_3, q_4)$$

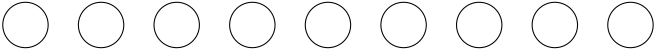
If several steps are possible, a random scheduler chooses one (fixed nonzero prob. for each pair)

Execution: infinite sequence $C_0 \to C_1 \to C_2 \to \cdots$ of steps

Population protocols (PPs)

A population protocol (PP) consists of

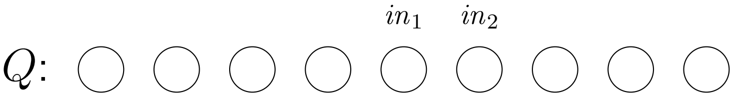
- A PP-scheme (Q, Δ)

Q: 

Population protocols (PPs)

A population protocol (PP) consists of

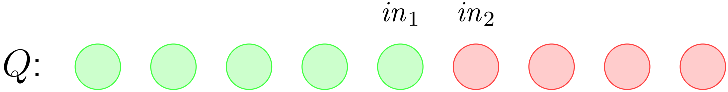
- A PP-scheme (Q, Δ)
- An ordered subset $(in_1, \ldots, in_k)$ of input states



Population protocols (PPs)

A population protocol (PP) consists of

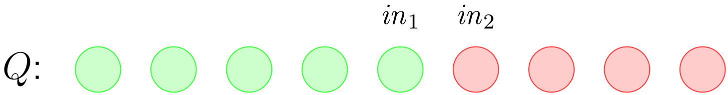
- A PP-scheme (Q, Δ)
- An ordered subset $(in_1, \ldots, in_k)$ of input states
- A partition of Q into 1-states (green) and 0-states (pink)



Population protocols (PPs)

A population protocol (PP) consists of

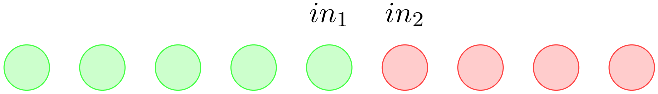
- A PP-scheme (Q, Δ)
- An ordered subset $(in_1, \ldots, in_k)$ of input states
- A partition of Q into 1-states (green) and 0-states (pink)



An execution reaches consensus $b \in \{0, 1\}$ if from some point on every agent stays within the b-states.

Computing with PPs

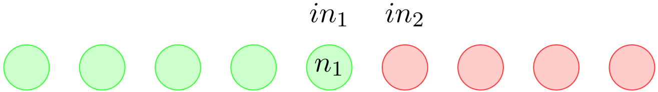
A PP computes the value b for input $(n_1, n_2, \ldots, n_k)$ if executions starting at the configuration



Computing with PPs

A PP computes the value b for input $(n_1, n_2, \ldots, n_k)$ if executions starting at the configuration

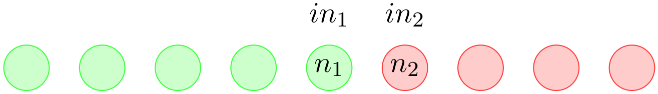
$$n_1 \cdot \mathbf{in_1}$$



Computing with PPs

A PP computes the value b for input $(n_1, n_2, \ldots, n_k)$ if executions starting at the configuration

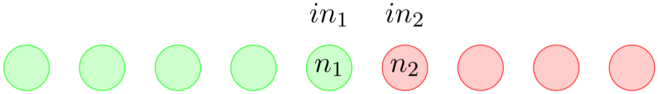
$$n_1 \cdot \mathbf{in_1} + n_2 \cdot \mathbf{in_2}$$



Computing with PPs

A PP computes the value b for input $(n_1, n_2, \ldots, n_k)$ if executions starting at the configuration

$$n_1 \cdot \mathbf{in_1} + n_2 \cdot \mathbf{in_2} + \cdots + n_k \cdot \mathbf{in_k}$$

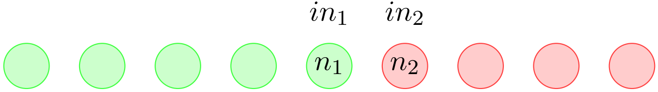


Computing with PPs

A PP computes the value b for input $(n_1, n_2, \ldots, n_k)$ if executions starting at the configuration

$$n_1 \cdot \mathbf{in_1} + n_2 \cdot \mathbf{in_2} + \cdots + n_k \cdot \mathbf{in_k}$$

reach consensus b with probability 1.

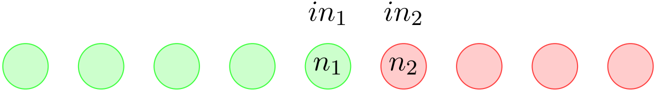


Computing with PPs

A PP computes the value b for input $(n_1, n_2, \ldots, n_k)$ if executions starting at the configuration

$$n_1 \cdot \mathbf{in_1} + n_2 \cdot \mathbf{in_2} + \cdots + n_k \cdot \mathbf{in_k}$$

reach consensus b with probability 1.



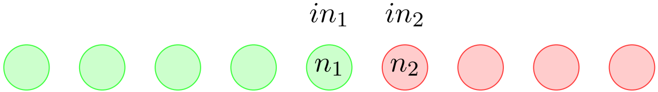
Equivalently: executions that do not reach consensus or reach consensus $1 - b$ have probability 0

Computing with PPs

A PP computes the value b for input $(n_1, n_2, \ldots, n_k)$ if executions starting at the configuration

$$n_1 \cdot \mathbf{in_1} + n_2 \cdot \mathbf{in_2} + \cdots + n_k \cdot \mathbf{in_k}$$

reach consensus b with probability 1.



Equivalently: executions that do not reach consensus or reach consensus $1 - b$ have probability 0

A PP computes $P(x_1, \ldots, x_n) \colon \mathbb{N}^n \to \{0, 1\}$ if it computes $P(n_1, \ldots, n_k)$ for every input $(n_1, \ldots, n_k)$

Previous work

Expressive power thoroughly studied:

- PPs compute exactly the Presburger predicates
  (Angluin *et al.* 2007)

Previous work

Expressive power thoroughly studied:

- PPs compute exactly the Presburger predicates (Angluin *et al.* 2007)
- Probabilistic PPs (Angluin *et al.* 2004-2006, Chatzigiannakis and Spirakis, 2008)
- Fault-tolerant PPs (Delporte-Gallet *et al.* 2006)
- Private computation in PPs (Delporte-Gallet *et al.* 2007)
- PPs with identifiers (Guerraoui *et al.* 2007)
- PPs with a leader (Angluin *et al.* 2008)
- Mediated PPs (Michail *et al.*, 2011)
- Trustful PPs (Bournez *et al.*, 2013)

Well-specified protocols

Q: And if the processes only reach consensus with probability < 1?

Well-specified protocols

Q: And if the processes only reach consensus with probability < 1?

A: *Then your protocol is not well-specified. Repair it!*

Well-specified protocols

Q: And if the processes only reach consensus with probability < 1?

A: *Then your protocol is not well-specified. Repair it!*

Q: And if the processes may reach consensus 0 and 1 for the same input, both with positive probability?

Well-specified protocols

Q: And if the processes only reach consensus with probability < 1?

A: *Then your protocol is not well-specified. Repair it!*

Q: And if the processes may reach consensus 0 and 1 for the same input, both with positive probability?

A: *Then your protocol is not well-specified. Repair it!*

Well-specified protocols

Q: And if the processes only reach consensus with probability < 1?

A: *Then your protocol is not well-specified. Repair it!*

Q: And if the processes may reach consensus 0 and 1 for the same input, both with positive probability?

A: *Then your protocol is not well-specified. Repair it!*

Q: And how do I know if my protocol is well-specified?

Well-specified protocols

Q: And if the processes only reach consensus with probability < 1?

A: *Then your protocol is not well-specified. Repair it!*

Q: And if the processes may reach consensus 0 and 1 for the same input, both with positive probability?

A: *Then your protocol is not well-specified. Repair it!*

Q: And how do I know if my protocol is well-specified?

A: *That's your problem . . .*

Well-specified protocols

Q: And if the processes only reach consensus with probability < 1?

A: *Then your protocol is not well-specified. Repair it!*

Q: And if the processes may reach consensus 0 and 1 for the same input, both with positive probability?

A: *Then your protocol is not well-specified. Repair it!*

Q: And how do I know if my protocol is well-specified?

A: *That's your problem ...*

Well-specification problem: Given a protocol, decide if it is well-specified.

Correctness problem: Given a protocol and a Presburger predicate, decide if the protocol is well-specified and computes the predicate.

Verifying population protocols: Previous work

Verifying population protocols: Previous work

- For each input, the semantics of the protocol is a finite-state Markov chain
- The semantics for all inputs is an infinite collection of finite-state Markov chain

Verifying population protocols: Previous work

- For each input, the semantics of the protocol is a finite-state Markov chain
- The semantics for all inputs is an infinite collection of finite-state Markov chain
- Use model-checkers (SPIN, PRISM , . . .) to verify correctness for some inputs
  Pang *et al.*, 2008 ; Sun *et al.*, 2009
  Chatzigiannakis *et al.*, 2010 ; Clément *et al.*, 2011

Verifying population protocols: Previous work

- For each input, the semantics of the protocol is a finite-state Markov chain
- The semantics for all inputs is an infinite collection of finite-state Markov chain
- Use model-checkers (SPIN, PRISM , . . .) to verify correctness for some inputs
  Pang *et al.*, 2008 ; Sun *et al.*, 2009
  Chatzigiannakis *et al.*, 2010 ; Clément *et al.*, 2011
- Use interactive theorem provers (Coq) to prove correctness of a specific protocol
  Deng *et al.*, 2009 and 2011

Verifying population protocols: Previous work

- For each input, the semantics of the protocol is a finite-state Markov chain
- The semantics for all inputs is an infinite collection of finite-state Markov chain
- Use model-checkers (SPIN, PRISM , . . .) to verify correctness for some inputs
  Pang *et al.*, 2008 ; Sun *et al.*, 2009
  Chatzigiannakis *et al.*, 2010 ; Clément *et al.*, 2011
- Use interactive theorem provers (Coq) to prove correctness of a specific protocol
  Deng *et al.*, 2009 and 2011

Not complete or not automatic.

Main results

Are the well-specification and correctness problems decidable?

Main results

Are the well-specification and correctness problems decidable?

Open for about 10 years.

Main results

Are the well-specification and correctness problems decidable?

Open for about 10 years.

Theorem: The well-specification and correctness problems can be reduced to the reachability problem for Petri nets, and are thus decidable.

Main results

Are the well-specification and correctness problems decidable?

Open for about 10 years.

Theorem: The well-specification and correctness problems can be reduced to the reachability problem for Petri nets, and are thus decidable.

Theorem: The reachability problem for Petri nets can be reduced to the well-specification and correctness problems for PPs.

From PPs to Petri nets

Population protocols	Petri nets
State	Place

From PPs to Petri nets

Population protocols	Petri nets
State	Place
Interaction $(q_1, q_2) \mapsto (q'_1, q'_2)$	Transition with input places q_1, q_2 output places q'_1, q'_2

From PPs to Petri nets

Population protocols	Petri nets
State	Place
Interaction $(q_1, q_2) \mapsto (q_1', q_2')$	Transition with input places q_1, q_2 output places q_1', q_2'
PP-scheme	Net without marking

From PPs to Petri nets

Population protocols	Petri nets
State	Place
Interaction $(q_1, q_2) \mapsto (q'_1, q'_2)$	Transition with input places q_1, q_2 output places q'_1, q'_2
PP-scheme	Net without marking
Configuration	Marking

From PPs to Petri nets

Population protocols	Petri nets
State	Place
Interaction $(q_1, q_2) \mapsto (q_1', q_2')$	Transition with input places q_1, q_2 output places q_1', q_2'
PP-scheme	Net without marking
Configuration	Marking
Configuration graph	Reachability graph

From PPs to Petri nets

Population protocols	Petri nets
State	Place
Interaction $(q_1, q_2) \mapsto (q_1', q_2')$	Transition with input places q_1, q_2 output places q_1', q_2'
PP-scheme	Net without marking
Configuration	Marking
Configuration graph	Reachability graph
PP	Net $+$ family of initial markings

Some results from Petri net theory

Theorem [Mayr, Kosaraju, Lambert, Leroux]: The reachability problem for Petri nets is decidable.

Best known algorithms are non-primitive recursive.

Some results from Petri net theory

Theorem [Mayr, Kosaraju, Lambert, Leroux]: The reachability problem for Petri nets is decidable.

Best known algorithms are non-primitive recursive.

A Presburger set of markings is a set of markings expressible in Presburger arithmetic.

$$\mathcal{M}(x, y) = 3x + y \geq 6 \wedge y - x \leq 2$$

Some results from Petri net theory

Theorem [Mayr, Kosaraju, Lambert, Leroux]: The reachability problem for Petri nets is decidable.

Best known algorithms are non-primitive recursive.

A Presburger set of markings is a set of markings expressible in Presburger arithmetic.

$$\mathcal{M}(x, y) = 3x + y \geq 6 \wedge y - x \leq 2$$

Theorem [Easy generalization]: Given two Presburger sets of markings $\mathcal{M}_1$, $\mathcal{M}_2$, it is decidable if some marking of $\mathcal{M}_2$ is reachable from some marking of $\mathcal{M}_1$

Well-specification is decidable

Fact: Every execution of a PP gets eventually trapped in a bottom SCC of its configuration graph with probability 1.

Well-specification is decidable

Fact: Every execution of a PP gets eventually trapped in a bottom SCC of its configuration graph with probability 1.

Our main technical result: The set of all configurations belonging to all bottom SCCs of a PP, for all initial configurations, is effectively Presburger.

Well-specification is decidable

Fact: Every execution of a PP gets eventually trapped in a bottom SCC of its configuration graph with probability 1.

Our main technical result: The set of all configurations belonging to all bottom SCCs of a PP, for all initial configurations, is effectively Presburger.

Fact: A PP is ill-specified iff there is an initial configuration C and

- a bottom SCC reachable from C with agents in both 0- and 1-states; or
- two bottom SCCs, one with only 0-states and the other with only 1-states, both reachable from C.

Well-specification is decidable

Given a PP, let

- $\mathcal{N}$: Petri net for the PP
- $\mathcal{I}$: markings corresponding to initial configurations
- $\mathcal{B}$: markings corresponding to bottom configurations

Well-specification is decidable

Given a PP, let

- $\mathcal{N}$: Petri net for the PP
- $\mathcal{I}$: markings corresponding to initial configurations
- $\mathcal{B}$: markings corresponding to bottom configurations

Decision procedure:

Well-specification is decidable

Given a PP, let

- $\mathcal{N}$: Petri net for the PP
- $\mathcal{I}$: markings corresponding to initial configurations
- $\mathcal{B}$: markings corresponding to bottom configurations

Decision procedure:

- Partition $\mathcal{B}$ into $\mathcal{B}_{\text{true}}$, $\mathcal{B}_{\text{false}}$, $\mathcal{B}_{\text{neither}}$

Well-specification is decidable

Given a PP, let

- $\mathcal{N}$: Petri net for the PP
- $\mathcal{I}$: markings corresponding to initial configurations
- $\mathcal{B}$: markings corresponding to bottom configurations

Decision procedure:

- Partition $\mathcal{B}$ into $\mathcal{B}_{\text{true}}$, $\mathcal{B}_{\text{false}}$, $\mathcal{B}_{\text{neither}}$
- Check if $\mathcal{B}_{\text{neither}}$ is reachable from $\mathcal{I}$
  (using reachability in Petri nets)

Well-specification is decidable

Given a PP, let

- $\mathcal{N}$: Petri net for the PP
- $\mathcal{I}$: markings corresponding to initial configurations
- $\mathcal{B}$: markings corresponding to bottom configurations

Decision procedure:

- Partition $\mathcal{B}$ into $\mathcal{B}_{\text{true}}$, $\mathcal{B}_{\text{false}}$, $\mathcal{B}_{\text{neither}}$
- Check if $\mathcal{B}_{\text{neither}}$ is reachable from $\mathcal{I}$
  (using reachability in Petri nets)
- Construct the net $N \parallel N$ (two copies of N side by side).

Well-specification is decidable

Given a PP, let

- $\mathcal{N}$: Petri net for the PP
- $\mathcal{I}$: markings corresponding to initial configurations
- $\mathcal{B}$: markings corresponding to bottom configurations

Decision procedure:

- Partition $\mathcal{B}$ into $\mathcal{B}_{\text{true}}$, $\mathcal{B}_{\text{false}}$, $\mathcal{B}_{\text{neither}}$
- Check if $\mathcal{B}_{\text{neither}}$ is reachable from $\mathcal{I}$ (using reachability in Petri nets)
- Construct the net $N \parallel N$ (two copies of N side by side).
- Construct the set $\mathcal{I}_2 = \{(M, M) \mid M \in \mathcal{I}\}$.

Well-specification is decidable

Given a PP, let

- $\mathcal{N}$: Petri net for the PP
- $\mathcal{I}$: markings corresponding to initial configurations
- $\mathcal{B}$: markings corresponding to bottom configurations

Decision procedure:

- Partition $\mathcal{B}$ into $\mathcal{B}_{\mathrm{true}}$, $\mathcal{B}_{\mathrm{false}}$, $\mathcal{B}_{\mathrm{neither}}$
- Check if $\mathcal{B}_{\mathrm{neither}}$ is reachable from $\mathcal{I}$ (using reachability in Petri nets)
- Construct the net $N \parallel N$ (two copies of N side by side).
- Construct the set $\mathcal{I}_2 = \{(M, M) \mid M \in \mathcal{I}\}$.
- Check if $\mathcal{B}_{\mathrm{true}} \times \mathcal{B}_{\mathrm{false}}$ is reachable from $\mathcal{I}_2$ (using reachability in Petri nets)

Further results

- Elementary decision procedure for immediate observation protocols

Further results

- Elementary decision procedure for immediate observation protocols
- Decidability extends to all properties expressible in LTL

Further results

- Elementary decision procedure for immediate observation protocols
- Decidability extends to all properties expressible in LTL
- Quantitative verification (probability exceeds given threshold) is undecidable

Further results

- Elementary decision procedure for immediate observation protocols
- Decidability extends to all properties expressible in LTL
- Quantitative verification (probability exceeds given threshold) is undecidable

Thank You