Verification of Population Protocols

Javier Esparza
Technical University of Munich

Joint work with
Pierre Ganty, Jérôme Leroux, Rupak Majumdar,
and
Michael Blondin, Stefan Jaax, and Philipp Meyer

Deaf Black Ninjas in the Dark

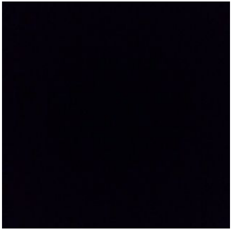
- Deaf Black Ninjas meet at a Zen garden in the dark

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)

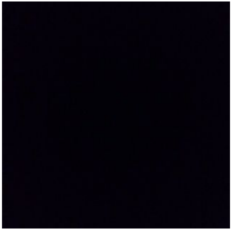
Deaf Black Ninjas in the Dark

- 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)	(opposite votes "cancel")
(YA , NP)	$\rightarrow$	(YA , YP)	(active "survivors" tell
(NA , YP)	$\rightarrow$	(NA , NP)	outcome to passive Ninjas)
(NP , YP)	$\rightarrow$	(NP , NP)	(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 $(q_1, q_2) \mapsto (q_1, q_2)$)

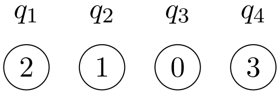
Semantics

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.



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

Semantics

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

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

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

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.

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

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

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

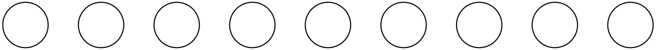
Execution: infinite sequence $C_0 \to C_1 \to C_2 \to \cdots$ of steps

Population protocols (PPs)

A population protocol (PP) consists of

- A PP-scheme (Q, Δ)

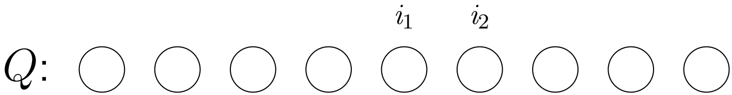
Q:



Population protocols (PPs)

A population protocol (PP) consists of

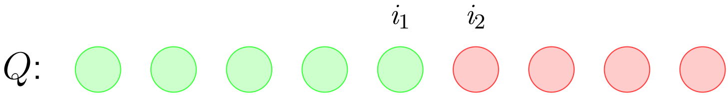
- A PP-scheme (Q, Δ)
- An ordered subset $(i_1, \ldots, i_k)$ of input states



Population protocols (PPs)

A population protocol (PP) consists of

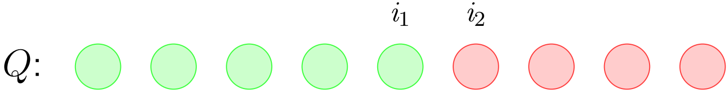
- A PP-scheme (Q, Δ)
- An ordered subset $(i_1, \ldots, i_k)$ of input states
- A partition of Q into 1-states (green) and 0-states (pink)



Population protocols (PPs)

A population protocol (PP) consists of

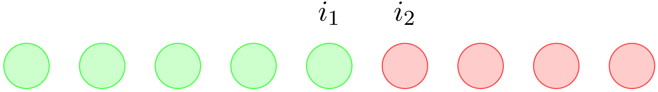
- A PP-scheme (Q, Δ)
- An ordered subset $(i_1, \dots, i_k)$ of input states
- A partition of Q into 1-states (green) and 0-states (pink)



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

Computing with PPs

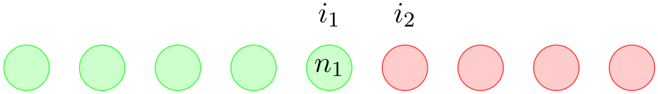
A PP computes the value b for input $(n_1, n_2, \ldots, n_k)$ if executions starting at the configuration



Computing with PPs

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

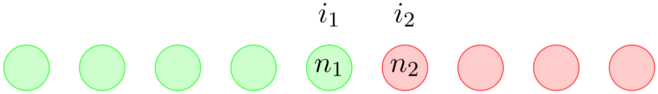
$n_1 \cdot \mathbf{i_1}$



Computing with PPs

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

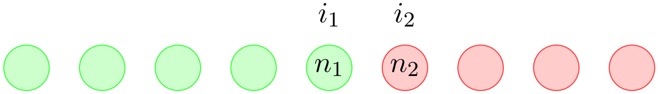
$$n_1 \cdot \mathbf{i_1} + n_2 \cdot \mathbf{i_2}$$



Computing with PPs

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

$$n_1 \cdot \mathbf{i_1} + n_2 \cdot \mathbf{i_2} + \cdots + n_k \cdot \mathbf{i_k}$$

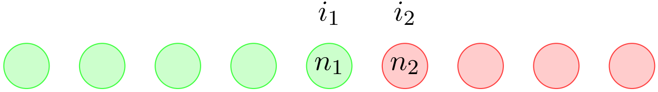


Computing with PPs

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

$$n_1 \cdot \mathbf{i_1} + n_2 \cdot \mathbf{i_2} + \cdots + n_k \cdot \mathbf{i_k}$$

reach consensus b with probability 1.

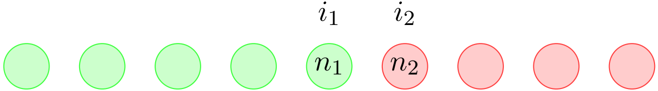


Computing with PPs

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

$$n_1 \cdot \mathbf{i_1} + n_2 \cdot \mathbf{i_2} + \cdots + n_k \cdot \mathbf{i_k}$$

reach consensus b with probability 1.



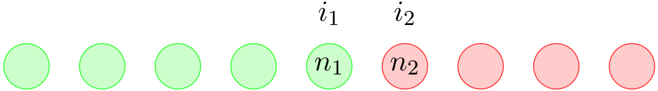
Equivalently: executions that do not reach consensus or reach consensus $1 - b$ have probability 0

Computing with PPs

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

$$n_1 \cdot \mathbf{i_1} + n_2 \cdot \mathbf{i_2} + \cdots + n_k \cdot \mathbf{i_k}$$

reach consensus b with probability 1.



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

A PP computes $P(x_1, \ldots, x_n) \colon \mathbb{N}^n \rightarrow \{0, 1\}$ 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 chains

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 chains
- 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 chains
- 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 chains
- 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

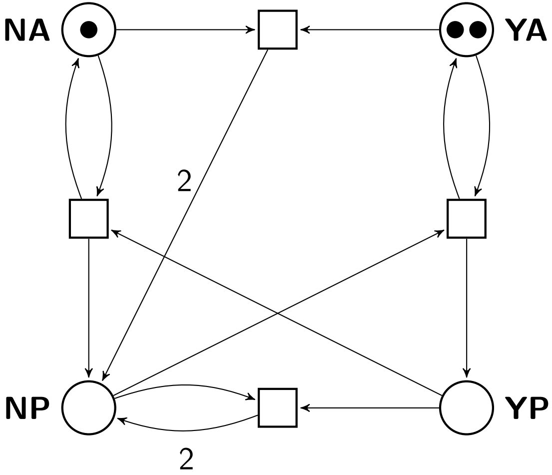
From PPs to Petri nets

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 $+$ infinite family of initial markings

Petri net of the majority protocol



Reducing well-specification to a reachability problem

Configuration graph of a PP:

- Nodes: all (infinitely many) posible configurations
- Edges: steps

Reducing well-specification to a reachability problem

Configuration graph of a PP:

- Nodes: all (infinitely many) posible configurations
- Edges: steps

Fact: Infinite, but every node has only finitely many successors.

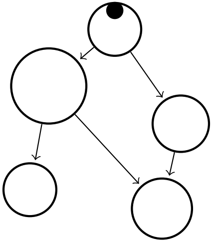
Reducing well-specification to a reachability problem

Configuration graph of a PP:

- Nodes: all (infinitely many) posible configurations
- Edges: steps

Fact: Infinite, but every node has only finitely many successors.

Fact: Every execution of a PP gets eventually trapped in a bottom SCC of its configuration graph w.p.1, and visits all configurations of the SCC infinitely often w.p.1.



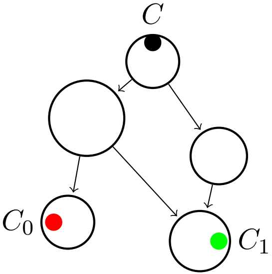
Reducing well-specification to a reachability problem

Bottom configuration: configuration of a bottom SCC.

Fact: A PP is ill-specified iff there is

- an initial configuration C, and
- two bottom configurations C_0 and C_1, reachable from C

such that C_0 has at least one agent in a 0-state, and C_1 has at least one agent in a 1-state.



Well-specification is decidable

Theorem 1: Given two (possibly infinite!) Presburger sets of configurations $\mathcal{C}_1$, $\mathcal{C}_2$ of a Petri net, it is decidable if some configuration of $\mathcal{C}_2$ is reachable from some configuration of $\mathcal{C}_1$.

Easy reduction to the reachability problem for Petri nets.

Well-specification is decidable

Theorem 1: Given two (possibly infinite!) Presburger sets of configurations $\mathcal{C}_1$, $\mathcal{C}_2$ of a Petri net, it is decidable if some configuration of $\mathcal{C}_2$ is reachable from some configuration of $\mathcal{C}_1$.

Easy reduction to the reachability problem for Petri nets.

Theorem 2: The set of all configurations belonging to all bottom SCCs from all configurations is an effectively Presburger set.

Well-specification is decidable

Theorem 1: Given two (possibly infinite!) Presburger sets of configurations $\mathcal{C}_1$, $\mathcal{C}_2$ of a Petri net, it is decidable if some configuration of $\mathcal{C}_2$ is reachable from some configuration of $\mathcal{C}_1$.

Easy reduction to the reachability problem for Petri nets.

Theorem 2: The set of all configurations belonging to all bottom SCCs from all configurations is an effectively Presburger set.

"Presburgerness" follows immediately from a classical result by Eilenberg and Schützenberger (1969) on rational sets in commutative monoids.

Well-specification is decidable

Theorem 1: Given two (possibly infinite!) Presburger sets of configurations $\mathcal{C}_1$, $\mathcal{C}_2$ of a Petri net, it is decidable if some configuration of $\mathcal{C}_2$ is reachable from some configuration of $\mathcal{C}_1$.

Easy reduction to the reachability problem for Petri nets.

Theorem 2: The set of all configurations belonging to all bottom SCCs from all configurations is an effectively Presburger set.

"Presburgerness" follows immediately from a classical result by Eilenberg and Schützenberger (1969) on rational sets in commutative monoids.

Effectiveness follows from a bound on the size of the Presburger representation due to Leroux (2011).

Well-specification is decidable

- $\mathcal{S}$: protocol scheme
- $\mathcal{I}$: set of all initial configurations
- $\mathcal{B}_b$ (where $b \in \{0, 1\}$): set of all bottom configurations with at least one agent in a b-state

Well-specification is decidable

- $\mathcal{S}$: protocol scheme
- $\mathcal{I}$: set of all initial configurations
- $\mathcal{B}_b$ (where $b \in \{0, 1\}$): set of all bottom configurations with at least one agent in a b-state

Decision procedure:

Well-specification is decidable

- $\mathcal{S}$: protocol scheme
- $\mathcal{I}$: set of all initial configurations
- $\mathcal{B}_b$ (where $b \in \{0, 1\}$): set of all bottom configurations with at least one agent in a b-state

Decision procedure:

- Construct the net $\mathcal{S} \parallel \mathcal{S}$ ("two copies of $\mathcal{S}$ side by side").

Well-specification is decidable

- $\mathcal{S}$: protocol scheme
- $\mathcal{I}$: set of all initial configurations
- $\mathcal{B}_b$ (where $b \in \{0, 1\}$): set of all bottom configurations with at least one agent in a b-state

Decision procedure:

- Construct the net $\mathcal{S} \parallel \mathcal{S}$ ("two copies of $\mathcal{S}$ side by side").
- Construct the set $\mathcal{I}_2 = \{(C, C) \mid C \in \mathcal{I}\}$ of configurations of $\mathcal{S} \parallel \mathcal{S}$.
  Presburger, because $\mathcal{I}$ is Presburger.

Well-specification is decidable

- $\mathcal{S}$: protocol scheme
- $\mathcal{I}$: set of all initial configurations
- $\mathcal{B}_b$ (where $b \in \{0,1\}$): set of all bottom configurations with at least one agent in a b-state

Decision procedure:

- Construct the net $\mathcal{S} \parallel \mathcal{S}$ ("two copies of $\mathcal{S}$ side by side").
- Construct the set $\mathcal{I}_2 = \{(C, C) \mid C \in \mathcal{I}\}$ of configurations of $\mathcal{S} \parallel \mathcal{S}$.
  Presburger, because $\mathcal{I}$ is Presburger.
- Check if $\mathcal{B}_0 \times \mathcal{B}_1$ is reachable from $\mathcal{I}_2$
  $\mathcal{B}_0 \times \mathcal{B}_1$ is effectively Presburger by Theorem 2, the check is effective by Theorem 1.

More decidable problems

Given: A PP P, a Presburger predicate Π
Decide: Does P compute Π?

Given: A well-specified PP P
Compute: A Presburger formula (or semilinear representation of) the predicate computed by P.

Fighting complexity I: The class WS^2

Search for a subclass of the class WS of well-specified protocols that

- Has a membership problem of reasonable complexity.
- Can compute all Presburger predicates.

Fighting complexity I: The class WS^2

Search for a subclass of the class WS of well-specified protocols that

- Has a membership problem of reasonable complexity.
- Can compute all Presburger predicates.

Many protocols from the literature are silent: Executions end w.p.1 in terminal configurations that enable no transitions.

Equivalent definition: bottom SCCs contain only one configuration.

Fighting complexity I: The class WS^2

Search for a subclass of the class WS of well-specified protocols that

- Has a membership problem of reasonable complexity.
- Can compute all Presburger predicates.

Many protocols from the literature are silent: Executions end w.p.1 in terminal configurations that enable no transitions.

Equivalent definition: bottom SCCs contain only one configuration.

Proposition: WS^2 protocols (well specified and silent) can compute all Presburger predicates.

Fighting complexity I: The class WS^2

Search for a subclass of the class WS of well-specified protocols that

- Has a membership problem of reasonable complexity.
- Can compute all Presburger predicates.

Many protocols from the literature are silent: Executions end w.p.1 in terminal configurations that enable no transitions.

Equivalent definition: bottom SCCs contain only one configuration.

Proposition: WS^2 protocols (well specified and silent) can compute all Presburger predicates.

Proposition : Petri net reachability is reducible to the membership problem for WS^2.

Fighting complexity II: The class WS^3

WS^2: Well-sp. silent

Termination

For every reachable configuration C there exists an execution leading from C to a terminal conf. $C_\perp$

Consensus

All terminal configurations reachable from a given initial configuration form the same consensus.

Fighting complexity II: The class WS^3

WS^2: Well-sp. silent

Termination

For every reachable configuration C there exists an execution leading from C to a terminal conf. $C_\perp$

Consensus

All terminal configurations reachable from a given initial configuration form the same consensus.

WS^3: Well-sp. strongly silent

Layered Termination

For every configuration C there exists a layered execution leading from C to a terminal configuration $C_\perp$

Strong Consensus

All terminal configurations potentially reachable from a given initial configuration form the same consensus.

Layered Termination

A protocol is layered if there is a partition of the set T of transitions into layers $T_1, \dots T_n$ s.t. for every configuration C (reachable or not):

- all executions from C containing only transitions of a single layer are finite.
- if all transitions of T_i are disabled at C, then they cannot be re-enabled by any sequence of transitions of $T_{i+1}, \dots, T_n$.

An execution is layered if it "respects the layers", i.e., if it belongs to $T_1^* T_2^* \dots T_n^*$.

Layered Termination

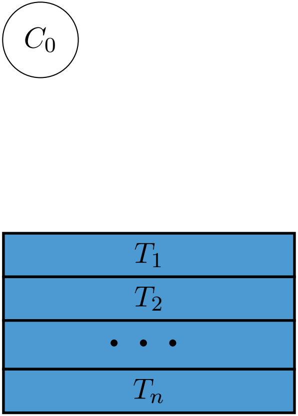
A protocol is layered if there is a partition of the set T of transitions into layers $T_1, \ldots T_n$ s.t. for every configuration C (reachable or not):

- all executions from C containing only transitions of a single layer are finite.
- if all transitions of T_i are disabled at C, then they cannot be re-enabled by any sequence of transitions of $T_{i+1}, \ldots, T_n$.

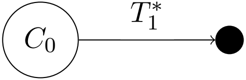
An execution is layered if it "respects the layers", i.e., if it belongs to $T_1^* T_2^* \ldots T_n^*$.

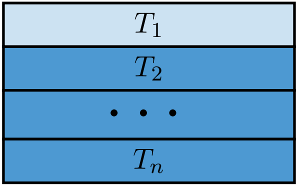
Fact: For every configuration C (reachable or not) there exists a layered execution leading from C to a terminal configuration $C_\perp$.

Layered Termination

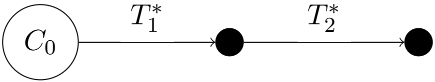


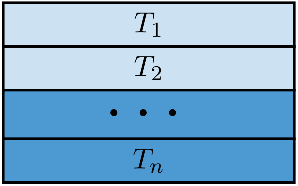
Layered Termination



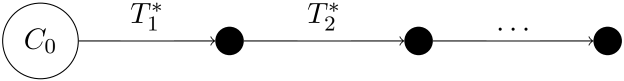


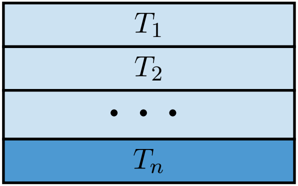
Layered Termination





Layered Termination





Layered Termination

$$C_0 \xrightarrow{T_1^*} \bullet \xrightarrow{T_2^*} \bullet \xrightarrow{\ \dots\ } \bullet \xrightarrow{T_n^*} C_\perp$$

T_1
T_2
$\bullet\ \bullet\ \bullet$
T_n

Complexity of checking Layered Termination

Lemma: Deciding Layered Termination is in NP.

Complexity of checking Layered Termination

Lemma: Deciding Layered Termination is in NP.

Proof sketch:

- Guess layers.
- Test that each individual layer terminates.
  Reducible to a Linear Programming Problem
- Test that lower layers cannot re-enable higher layers.
  Simple syntactic check.

Strong Consensus

Replace reachability by *cruder* relation called *potential reachability*:

$$\begin{aligned} \text{Reachability} &\Longrightarrow \text{Potential Reachability} \\ \text{Potential Reachability} &\not\Longrightarrow \text{Reachability} \end{aligned}$$

Potential reachability is defined in terms of a class of linear invariants derivable from the Petri net of the protocol by syntactic means (place invariants, siphons, traps).

A configuration C' is potentially reachable from C if both C and C' satisfy the same invariants of the class.

Lemma: Deciding Strong Consensus is in co-NP.

Completeness

Lemma: All well-specified population protocols can be represented by an equivalent population protocol satisfying Layered Termination and Strong Consensus.

By quantifier elimination, all predicates computable by Population Protocols can be defined as boolean combinations of

- Threshold: Is the weighted sum of the input values larger than a given threshold?

$$\Sigma_i \alpha_i x_i > c$$

- Remainder: Is the sum of the input values modulo a given m equal to a given c?

$$\Sigma_i \alpha_i x_i \bmod m = c$$

- Give WS^3 protocols for Threshold and Remainder predicates
- Prove that WS^3 protocols are closed under conjunction and negation.

Implementation

On top of the SMT-solver **Z3**.

Our tool 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.

Protocols from the literature for Majority, Threshold, Remainder, etc. belong to WS^3.

Experimental Results

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

Threshold			
ℓ_{max}	$\lvert Q\rvert$	$\lvert T\rvert$	Time[s]
3	28	288	8.0
4	36	478	26.5
5	44	716	97.6
6	52	1002	243.4
7	60	1336	565.0
8	68	1718	1019.7
9	76	2148	2375.9
10	84	2626	timeout

Remainder			
m	$\lvert Q\rvert$	$\lvert T\rvert$	Time[s]
10	12	65	0.4
20	22	230	2.8
30	32	495	15.9
40	42	860	79.3
50	52	1325	440.3
60	62	1890	3055.4
70	72	2555	3176.5
80	82	3320	timeout

Experimental Results

Flock of birds[1]			
c	$\|Q\|$	$\|T\|$	Time[s]
20	21	210	1.5
25	26	325	3.3
30	31	465	7.7
35	36	630	20.8
40	41	820	106.9
45	46	1035	295.6
50	51	1275	181.6
55	56	1540	timeout

[1] Chatzigiannakis et al., 2010

Flock of birds[2]			
c	$\|Q\|$	$\|T\|$	Time[s]
50	51	99	11.8
100	101	199	44.8
150	151	299	369.1
200	201	399	778.8
250	251	499	1554.2
300	301	599	2782.5
325	326	649	3470.8
350	351	699	timeout

[2] Clement et al., 2011

Conclusions

- The natural verification problems for population protocols are decidable.
- Efficient verification algorithms for the class WS^3.
- Implementation on top of SMT-solvers.

Conclusions

- The natural verification problems for population protocols are decidable.
- Efficient verification algorithms for the class WS^3.
- Implementation on top of SMT-solvers.
- Many open questions:
  - Complexity for immediate observation and immediate transmission protocols
  - Continuous Petri nets as abstractions
  - Expressive power of PP in non-uniform computational models
  - Applications to theoretical chemistry and systems biology
  - Correctness problem and convergence speed for WS^3 protocols.
  - Fault localization and repair.
  - Synthesis of WS^3 protocols.

Thank You