Towards Efficient Verification of Population Protocols

Javier Esparza
Technical University of Munich

Joint work with
Michael Blondin, Stefan Jaax, and Philipp Meyer

Deaf Black Ninjas in the Dark

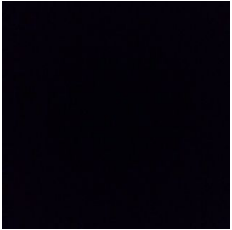
- Deaf Black Ninjas meet at a Zen garden in the dark

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)

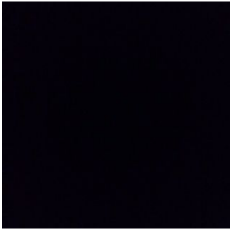
Deaf Black Ninjas in the Dark

- 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 his current estimation of the final outcome of the vote (Yes or No). Additionally, he 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 his current estimation of the final outcome of the vote (Yes or No). Additionally, he is Active or Passive.
- Initially all ninjas are Active, and their initial estimation is their own vote

Deaf Black Ninjas in the Dark

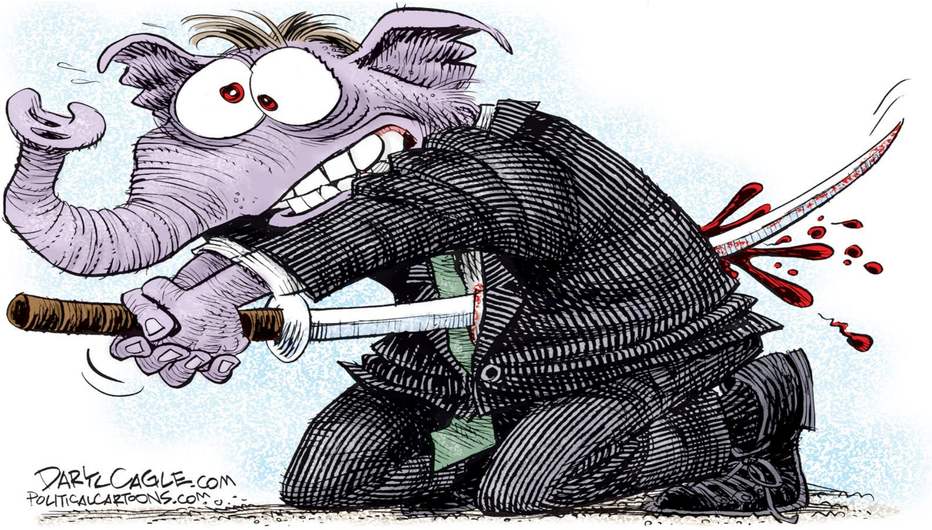
- Ninjas randomly wander around the garden, interacting when they bump into each other
- Each ninja stores his current estimation of the final outcome of the vote (Yes or No). Additionally, he is Active or Passive.
- Initially all ninjas are Active, and their initial estimation is their own vote
- Goal: eventually all ninjas reach the same estimation, and this estimation is the one corresponding to the majority vote

Deaf Black Ninjas in the Dark

- Ninjas randomly wander around the garden, interacting when they bump into each other
- Each ninja stores his current estimation of the final outcome of the vote (Yes or No). Additionally, he is Active or Passive.
- Initially all ninjas are Active, and their initial estimation is their own vote
- Goal: eventually all ninjas reach the same estimation, and this estimation is the one corresponding to the majority 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)

Deaf Black Ninjas in the Dark



Deaf Black Ninjas in the Dark: Corrected

The new Big Ninja added a rule in case there is a tie:

(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)

Deaf Black Ninjas in the Dark: Corrected

The new Big Ninja added a rule in case there is a tie:

(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)

Big Ninja's three questions:

- What is a protocol?

Deaf Black Ninjas in the Dark: Corrected

The new Big Ninja added a rule in case there is a tie:

(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)

Big Ninja's three questions:

- What is a protocol?
- When is a protocol correct?

Deaf Black Ninjas in the Dark: Corrected

The new Big Ninja added a rule in case there is a tie:

(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)

Big Ninja's three questions:

- What is a protocol?
- When is a protocol correct?
- How can I decide if a protocol is correct?

Big Ninja's first question: What is a protocol?

Population protocols: Theoretical model for distributed computation proposed in 2004 by Yale group (Angluin, Fischer, Aspnes ...)

Designed to model collections of

identical, finite-state, and mobile agents

like

- ad-hoc networks of mobile sensors
- "soups" of interacting molecules (Chemical Reaction Networks)
- people in social networks

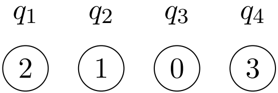
Syntax

PP-scheme: pair (Q, Δ), where Q is a finite set of states, and Δ is a set of interactions of the form $(q_1, q_2) \mapsto (q_3, q_4)$.

Syntax

PP-scheme: pair (Q, Δ), where Q is a finite set of states, and Δ is a set of interactions of the form $(q_1, q_2) \mapsto (q_3, q_4)$.

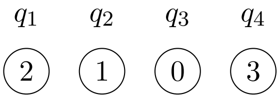
Configuration: mapping $C\colon Q \to \mathbb{N}$, where $C(q)$ is the current number of agents in q.



Syntax

PP-scheme: pair (Q, Δ), where Q is a finite set of states, and Δ is a set of interactions of the form $(q_1, q_2) \mapsto (q_3, q_4)$.

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

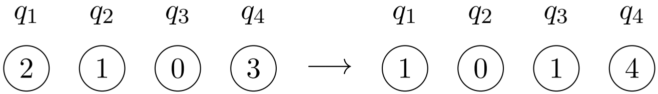


$$(q_1, q_2) \mapsto (q_3, q_4)$$

Syntax

PP-scheme: pair (Q, Δ), where Q is a finite set of states, and Δ is a set of interactions of the form $(q_1, q_2) \mapsto (q_3, q_4)$.

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



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

Syntax

PP-scheme: pair (Q, Δ), where Q is a finite set of states, and Δ is a set of interactions of the form $(q_1, q_2) \mapsto (q_3, q_4)$.

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

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

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

If several steps are possible, a random scheduler chooses one uniformly at random.

Syntax

PP-scheme: pair (Q, Δ), where Q is a finite set of states, and Δ is a set of interactions of the form $(q_1, q_2) \mapsto (q_3, q_4)$.

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

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

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

If several steps are possible, a random scheduler chooses one uniformly at random.

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

Semantics

A population protocol (PP) consists of

- A PP-scheme (Q, Δ)

Semantics

A population protocol (PP) consists of

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

Semantics

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)

Semantics

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)

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

Semantics

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)

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

A PP computes the value b for input $(n_1, n_2, \ldots, n_k)$ if the executions starting at the configuration with n_j agents in state i_j reach consensus b with probability 1.

Semantics

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)

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

A PP computes the value b for input $(n_1, n_2, \ldots, n_k)$ if the executions starting at the configuration with n_j agents in state i_j reach consensus b with probability 1.

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 $(n_1, \ldots, n_k)$

What predicates can PPs compute?

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

What predicates can PPs compute?

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

Presburger predicates: quantifier-free boolean combinations of

- Threshold predicates: $\sum_i \alpha_i x_i > c$
- Modulo predicates: $\sum_i \alpha_i x_i \bmod m = c$

What predicates can PPs compute?

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

Presburger predicates: quantifier-free boolean combinations of

- Threshold predicates: $\sum_i \alpha_i x_i > c$
- Modulo predicates: $\sum_i \alpha_i x_i \text{ mod } m = c$

To show that PPs compute all Presburger predicates:

- Give protocols for the threshold and remainder predicates.
- Show that computable predicates are closed under negation and conjunction.

Big Ninja's second question: When is a protocol correct?

A protocol is well specified if it computes some predicate:

- for every input $(x_1, \ldots, x_n)$, the executions reach the same consensus (which depends on $(x_1, \ldots, x_n)$) with probability one.

A protocol is correct for a given predicate P if it is well specified and computes P.

Big Ninja's second question: When is a protocol correct?

A protocol is well specified if it computes some predicate:

- for every input $(x_1, \ldots, x_n)$, the executions reach the same consensus (which depends on $(x_1, \ldots, x_n)$) with probability one.

A protocol is correct for a given predicate P if it is well specified and computes P.

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.

Big Ninja's third question: How can I decide correctness?

Theorem [E., Ganty, Leroux, Majumdar '15]: The well-specification and correctness problems can be reduced to the reachability problem for Petri nets, and are thus decidable.

But . . .

Theorem: The reachability problem for Petri nets is polynomially reducible to the well-specification problem.

The reachability problem for Petri nets is

- EXPSPACE-hard
- All known algorithms have non-primitive recursive complexity

Fighting complexity

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

- has a membership problem of reasonable complexity,
- still can compute all Presburger predicates, and
- contains many of the protocols in the literature.

Fighting complexity II: The class WS^2

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

Fighting complexity II: The class WS^2

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

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

Fighting complexity II: The class WS^2

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

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

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

Fighting complexity III: 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 III: 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 weakly 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

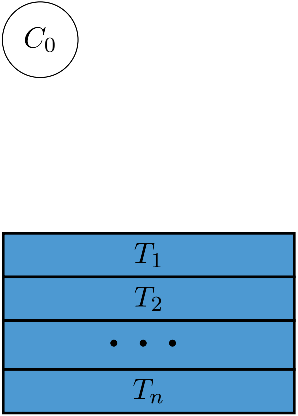
A protocol is layered if there is a partition of the set T of transitions into layers $T_1, \ldots T_n$ s.t. for every configuration C (reachable or not):

- 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 $T_{i+1}, \ldots, T_n$.

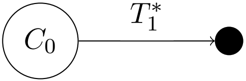
An execution is layered if it "respects the layers", i.e., if it belongs to $T_1^* T_2^* \ldots T_n^*$.

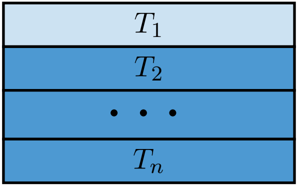
Fact: For every configuration C (reachable or not) there exists a layered execution leading from C to a terminal configuration $C_\perp$.

Layered Termination

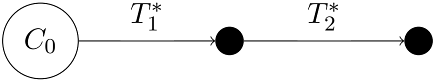


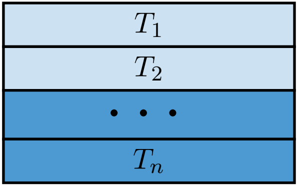
Layered Termination



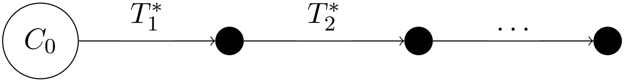


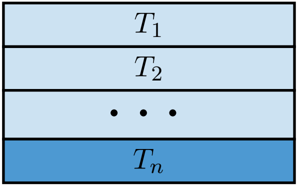
Layered Termination



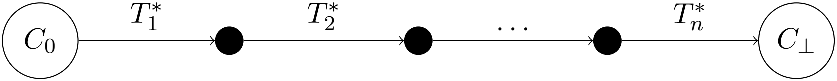


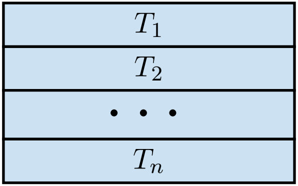
Layered Termination





Layered Termination





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:

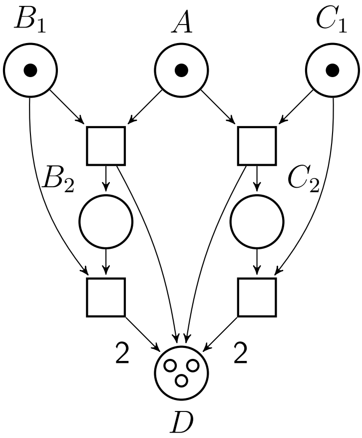
- Guess layers.
- Test that each individual layer terminates.
  Reducible to a Linear Programming Problem.
- Test that lower layers cannot re-enable higher layers.
  Simple syntactic check.

Strong Consensus: The Liquid Approximation

Strong Consensus: The Liquid Approximation

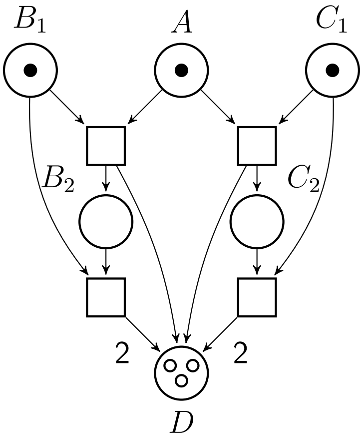
Fluid agents in action

$$(A, B_1) \rightarrow (D, B_2)$$
$$(A, C_1) \rightarrow (D, C_2)$$
$$(B_1, B_2) \rightarrow (D, D)$$
$$(C_1, C_2) \rightarrow (D, D)$$



Fluid agents in action

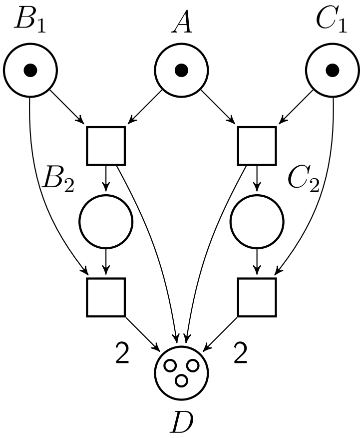
$$(A\ ,\ B_1) \quad \to \quad (D\ ,\ B_2)$$
$$(A\ ,\ C_1) \quad \to \quad (D\ ,\ C_2)$$
$$(B_1\ ,\ B_2) \quad \to \quad (D\ ,\ D)$$
$$(C_1\ ,\ C_2) \quad \to \quad (D\ ,\ D)$$



Theorem (Fraca, Haddad '15): Liquid reachability is in NP (P).

Fluid agents in action

$$(A\ ,\ B_1) \quad \rightarrow \quad (D\ ,\ B_2)$$
$$(A\ ,\ C_1) \quad \rightarrow \quad (D\ ,\ C_2)$$
$$(B_1\ ,\ B_2) \quad \rightarrow \quad (D\ ,\ D)$$
$$(C_1\ ,\ C_2) \quad \rightarrow \quad (D\ ,\ D)$$



Theorem (Fraca, Haddad '15): Liquid reachability is in NP (P).

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.

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

Completeness

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

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

Fact: Protocols from the literature for Majority, Threshold, Modulo, etc. belong to WS^3.

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

Intel Core i7-4810MQ CPU and 16 GB of RAM.

Protocol	Predicate	$\|Q\|$	$\|T\|$	Time[s]
Majority [1]	$x \geq y$	4	4	0.1
Approx. Majority [2]	Not well-specified	3	4	0.1
Broadcast [3]	$x \geq 1$	2	1	0.1
Threshold [4]	$\Sigma_i \alpha_i x_i \geq c$	76	2148	2375.9
Modulo [5]	$\Sigma_i \alpha_i x_i \bmod 70 = 1$	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
Poly-log flock of birds	$x \geq 8 \cdot 10^4$	66	244	12.79

[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

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.
  - Correctness problem and convergence speed for WS^3 protocols.
  - Minimal population protocols for given predicates.
  - Fault localization and repair.
  - Automatic synthesis of WS^3 protocols.
  - Theoretical and practical power of the liquid abstraction.
  - Expressive power of PPs in non-uniform computational models.
  - Applications to theoretical chemistry and systems biology.

Thank You