Black Ninjas in the Dark: Formal Analysis of Population Protocols

Javier Esparza

Joint work with Michael Blondin, Martin Helfrich, Pierre Ganty, Stefan Jaax, Antonín Kučera, Jérôme Leroux, Rupak Majumdar, Philipp J. Meyer, Mikhail Raskin, and Chana Weil-Kennedy

Technical University of Munich

European Research Council
Established by the European Commission

Deaf Black Ninjas in the Dark

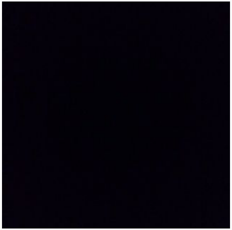
- Deaf Black Ninjas meet at a Zen garden in the dark

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 (no attack if tie)

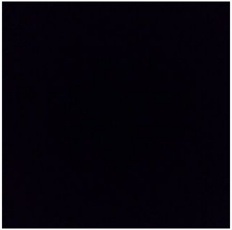
Deaf Black Ninjas in the Dark

- Deaf Black Ninjas meet at a Zen garden in the dark
- They must decide **by majority** to attack or not (no 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 (no attack if tie)
- How can they conduct the vote?



Deaf Black Ninjas in the Dark

- Ninjas wander randomly, interacting when they bump into each other.

Deaf Black Ninjas in the Dark

- Ninjas wander randomly, interacting when they bump into each other.
- Ninjas store their current estimation of the final outcome: **attack** or **don't attack**.

Deaf Black Ninjas in the Dark

- Ninjas wander randomly, interacting when they bump into each other.
- Ninjas store their current estimation of the final outcome: **attack** or **don't attack**.
- Additionally, they are active or passive .

attack
active

don't attack
active

attack
passive

don't attack
passive

Deaf Black Ninjas in the Dark

- Ninjas wander randomly, interacting when they bump into each other.
- Ninjas store their current estimation of the final outcome: **attack** or **don’t attack**.
- Additionally, they are active or passive .

attack
active

don’t attack
active

attack
passive

don’t attack
passive

- Initially: all ninjas active, estimation = own vote.

Goal of voting protocol:

- eventually all ninjas reach the same estimation, and
- this estimation corresponds to the majority.

Deaf Black Ninjas in the Dark

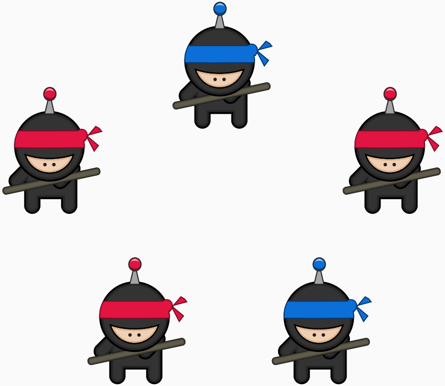
Goal of voting protocol:

- eventually all ninjas reach the same estimation, and
- this estimation corresponds to the majority.

Graphically:

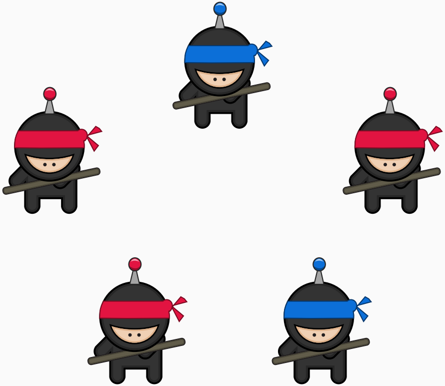
- Initially more red ninjas $\Longrightarrow$ eventually all ninjas red.
- Initially more blue ninjas or tie $\Longrightarrow$ eventually all ninjas blue.

Majority protocol: Are there more red ninjas than blue ninjas?



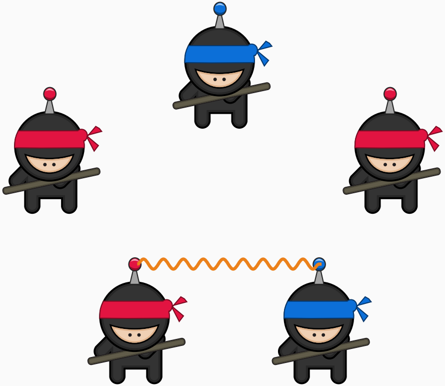
Majority protocol: Are there more red ninjas than blue ninjas?

- Active ninjas of opposite colors become passive and blue



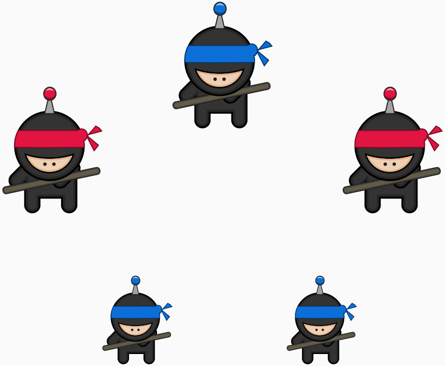
Majority protocol: Are there more red ninjas than blue ninjas?

- Active ninjas of opposite colors become passive and blue



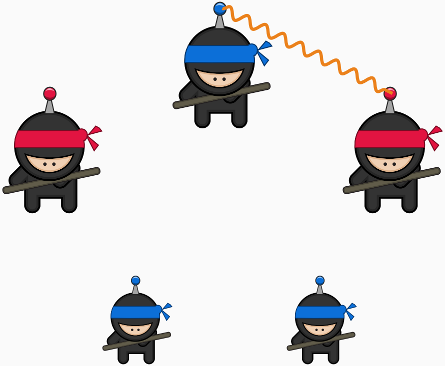
Majority protocol: Are there more red ninjas than blue ninjas?

- Active ninjas of opposite colors become passive and blue



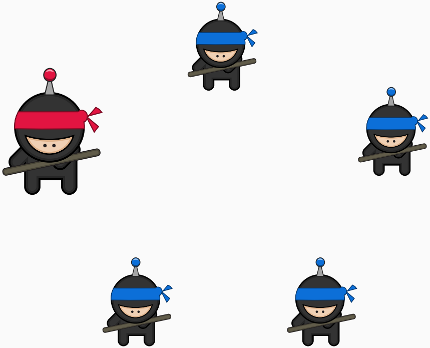
Majority protocol: Are there more red ninjas than blue ninjas?

- Active ninjas of opposite colors become passive and blue



Majority protocol: Are there more red ninjas than blue ninjas?

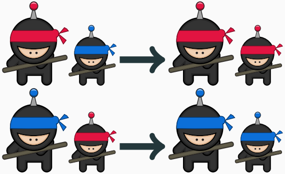
- Active ninjas of opposite colors become passive and blue

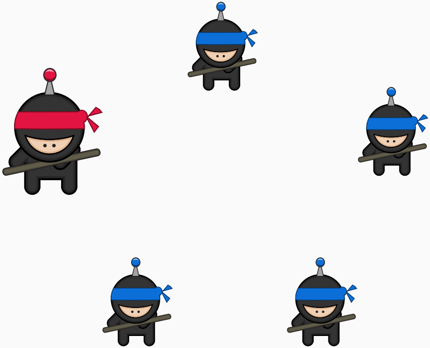


Majority protocol: Are there more red ninjas than blue ninjas?

- Active ninjas of opposite colors become passive and blue

- Active ninjas convert passive ninjas to their color

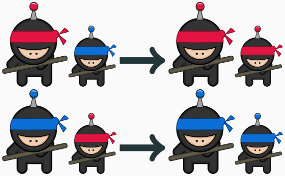


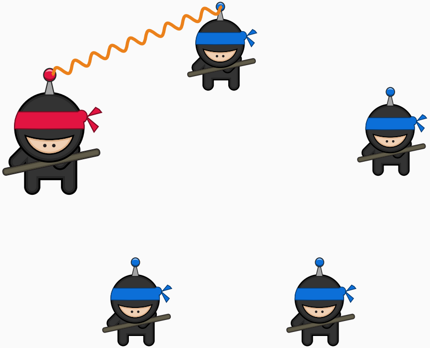


Majority protocol: Are there more red ninjas than blue ninjas?

- Active ninjas of opposite colors become passive and blue

- Active ninjas convert passive ninjas to their color

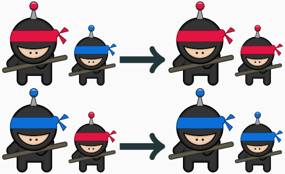


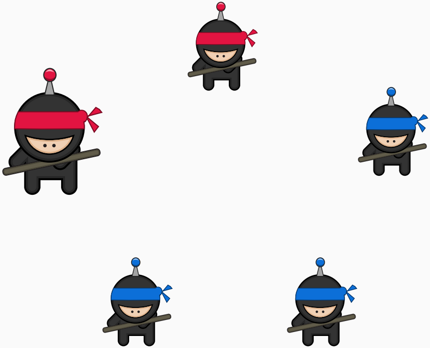


Majority protocol: Are there more red ninjas than blue ninjas?

- Active ninjas of opposite colors become passive and blue

- Active ninjas convert passive ninjas to their color

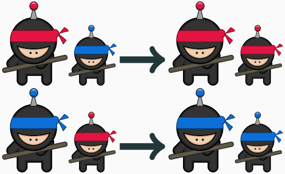


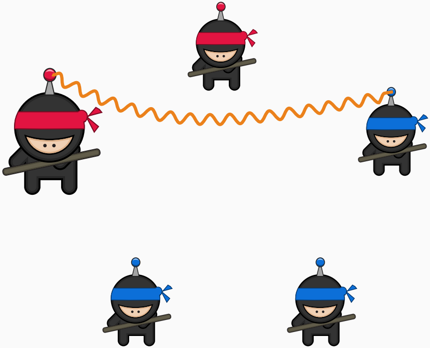


Majority protocol: Are there more red ninjas than blue ninjas?

- Active ninjas of opposite colors become passive and blue

- Active ninjas convert passive ninjas to their color

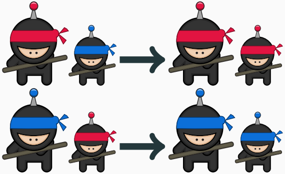


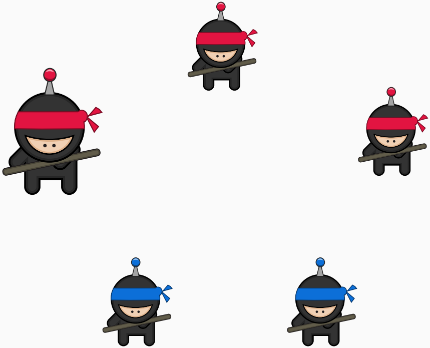


Majority protocol: Are there more red ninjas than blue ninjas?

- Active ninjas of opposite colors become passive and blue

- Active ninjas convert passive ninjas to their color

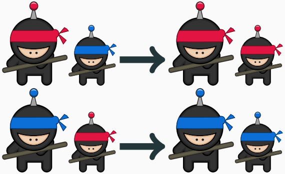


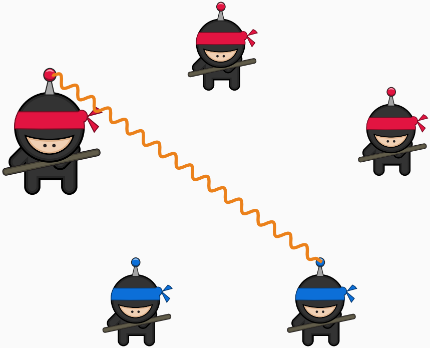


Majority protocol: Are there more red ninjas than blue ninjas?

- Active ninjas of opposite colors become passive and blue

- Active ninjas convert passive ninjas to their color

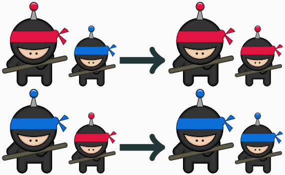


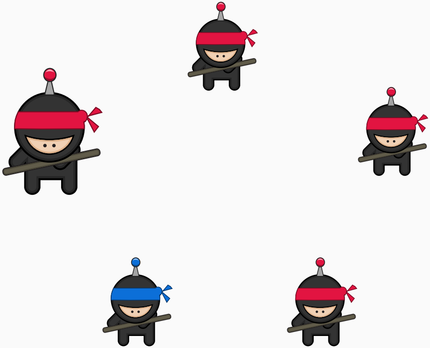


Majority protocol: Are there more red ninjas than blue ninjas?

- Active ninjas of opposite colors become passive and blue

- Active ninjas convert passive ninjas to their color

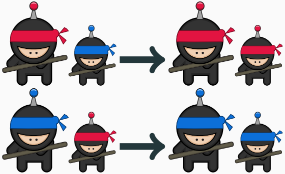


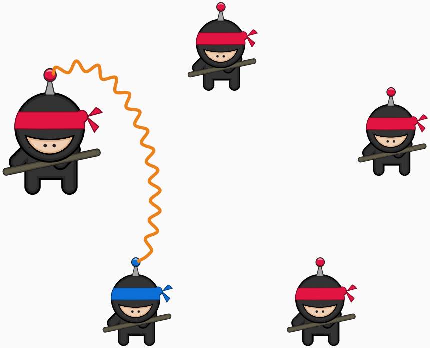


Majority protocol: Are there more red ninjas than blue ninjas?

- Active ninjas of opposite colors become passive and blue

- Active ninjas convert passive ninjas to their color

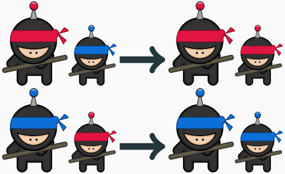


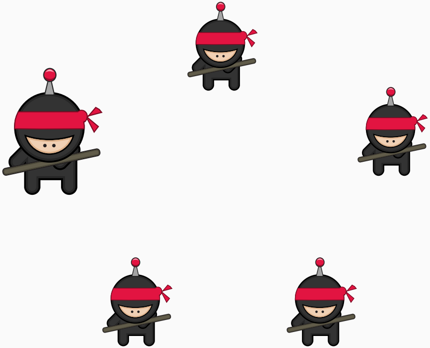


Majority protocol: Are there more red ninjas than blue ninjas?

- Active ninjas of opposite colors become passive and blue

- Active ninjas convert passive ninjas to their color

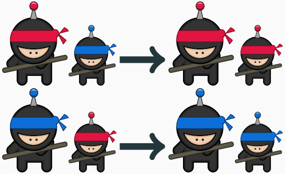




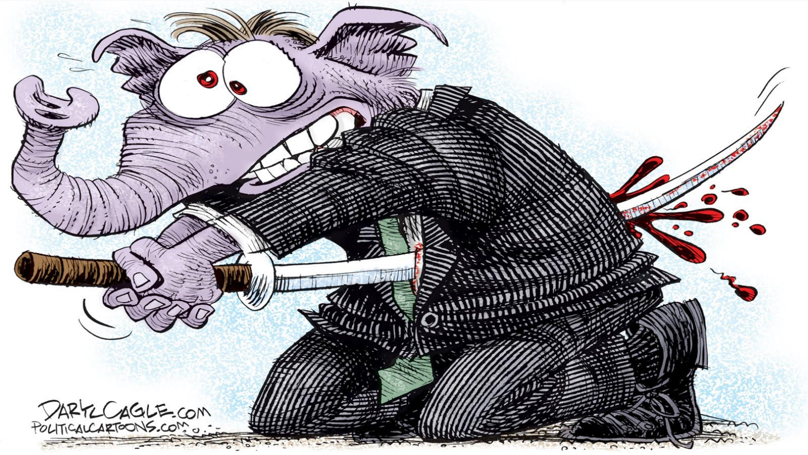
Majority protocol: Are there more red ninjas than blue ninjas?

- Active ninjas of opposite colors become passive and blue

- Active ninjas convert passive ninjas to their color



Sad story ...



Sensei II

Majority protocol: Why?

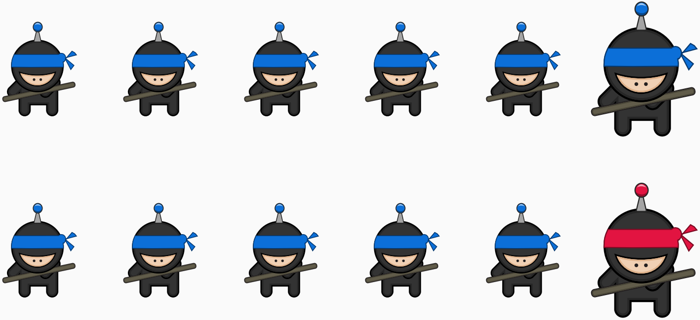
- The first rule has no priority over the other two.

Majority protocol: Why?

- The first rule has no priority over the other two.

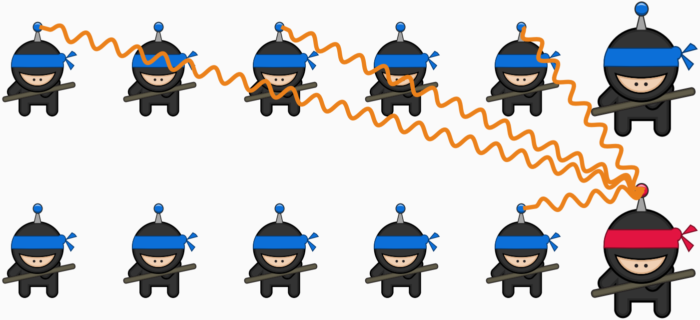
Majority protocol: Why?

- The first rule has no priority over the other two.



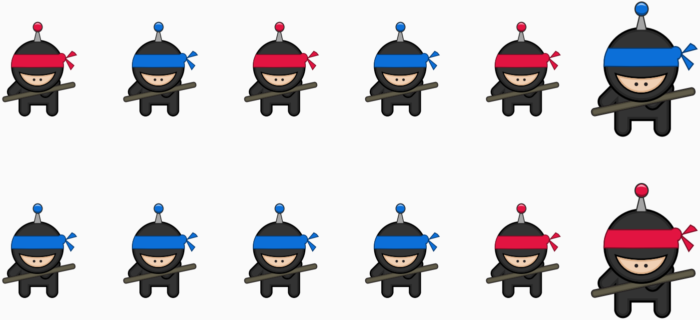
Majority protocol: Why?

- The first rule has no priority over the other two.



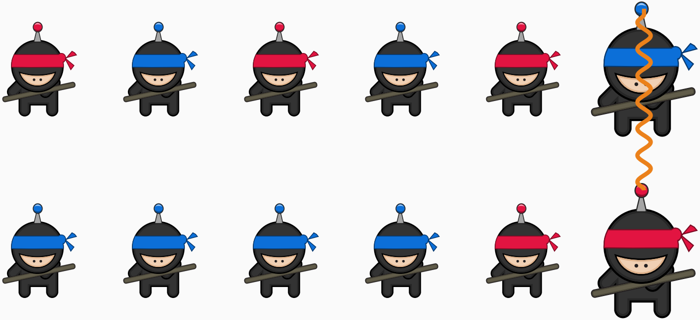
Majority protocol: Why?

- The first rule has no priority over the other two.



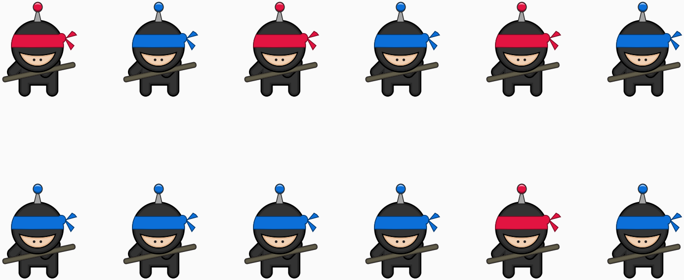
Majority protocol: Why?

- The first rule has no priority over the other two.



Majority protocol: Why?

- The first rule has no priority over the other two.



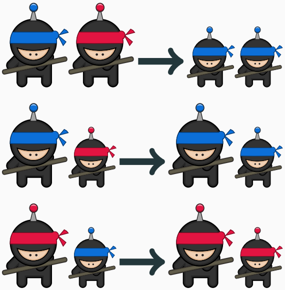
Majority protocol: Why?

- The first rule has no priority over the other two.

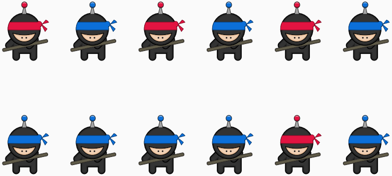
NO CONSENSUS!

Sensei II's protocol: Are there more red ninjas than blue ninjas?

Interaction rules:

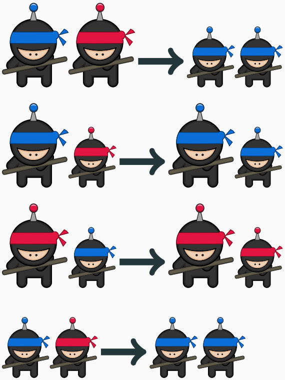


Sensei II



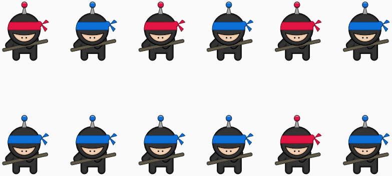
Sensei II's protocol: Are there more red ninjas than blue ninjas?

Interaction rules:



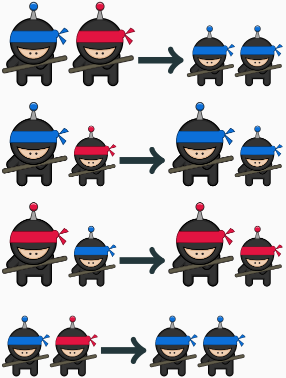
Passive blue ninjas convert passive red ninjas to their color

Sensei II



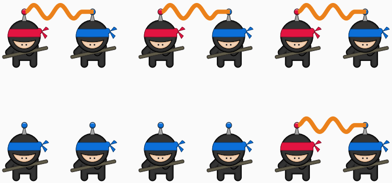
Sensei II's protocol: Are there more red ninjas than blue ninjas?

Interaction rules:



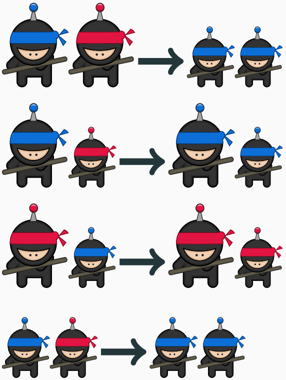
Passive blue ninjas convert passive red ninjas to their color

Sensei II



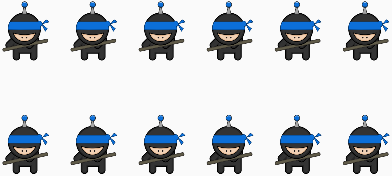
Sensei II's protocol: Are there more red ninjas than blue ninjas?

Interaction rules:



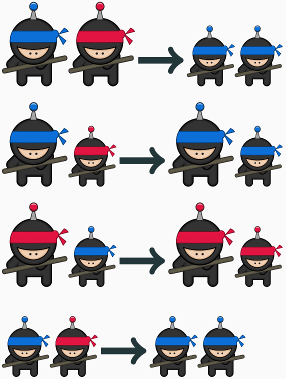
Passive blue ninjas convert passive red ninjas to their color

Sensei II



Sensei II's protocol: Are there more red ninjas than blue ninjas?

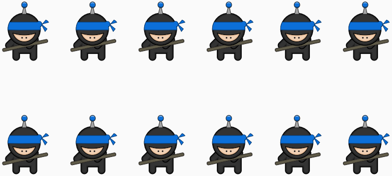
Interaction rules:



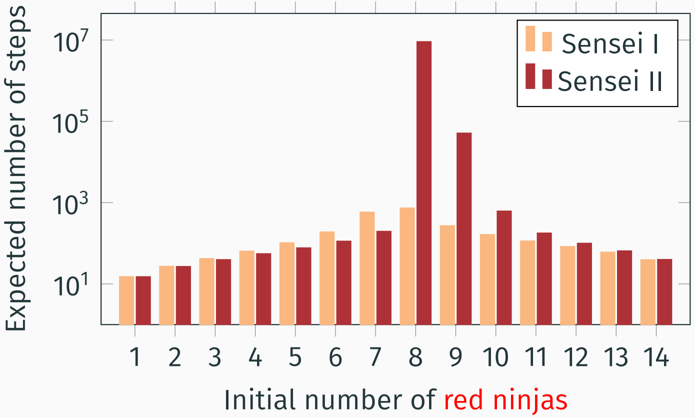
Passive blue ninjas convert passive red ninjas to their color

Sensei II

Go!



Sensei II's protocol: Are there more red ninjas than blue ninjas?



Expected number of steps to stable consensus for a population of 15 ninjas.

Very sad story ...

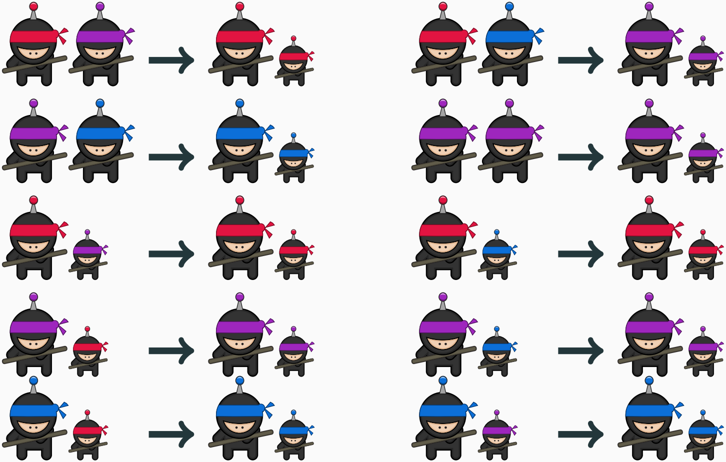
Sensei III

Sensei III's protocol

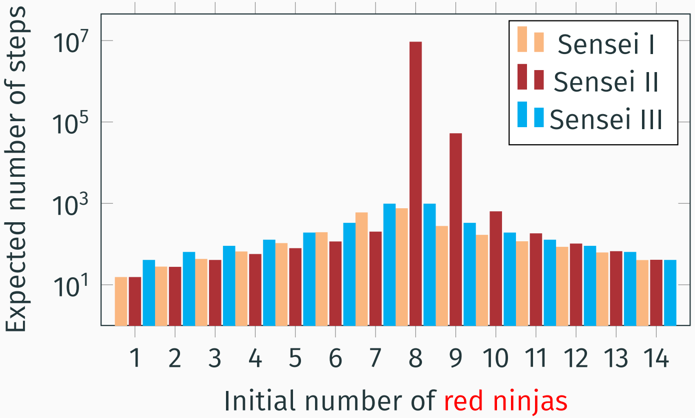
 = Attack majority  = Don't attack majority  = Tie

Interaction rules:

Go!



Sensei III's protocol



Expected number of steps to stable consensus for a population of 15 ninjas.

Sensei III's questions

Formalization questions:

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

Sensei III's questions

Verification questions:

- How do I check that my protocol is correct ?
- How do I check that my protocol is efficient ?

Sensei III's questions

Expressivity questions:

- Are there protocols for other problems?
- How large is the smallest protocol for a problem?
- And the smallest efficient protocol?

Population protocols

Angluin, Aspnes et al. PODC'04

Formal model of distributed computation by collections of

identical, finite-state, and mobile agents

like

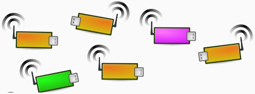
Population protocols

Angluin, Aspnes et al. PODC'04

Formal model of distributed computation by collections of

identical, finite-state, and mobile agents

like



ad-hoc networks of devices

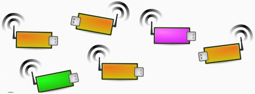
Population protocols

Angluin, Aspnes et al. PODC'04

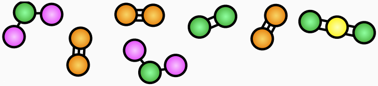
Formal model of distributed computation by collections of

identical, finite-state, and mobile agents

like



ad-hoc networks of devices



"soups" of molecules

(Chemical Reaction Networks)

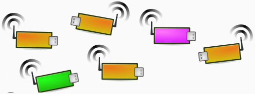
Population protocols

Angluin, Aspnes et al. PODC'04

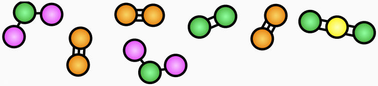
Formal model of distributed computation by collections of

identical, finite-state, and mobile agents

like

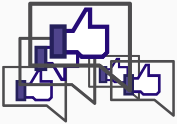


ad-hoc networks of devices



"soups" of molecules

(Chemical Reaction Networks)



people in social networks

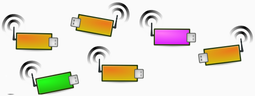
Population protocols

Angluin, Aspnes et al. PODC'04

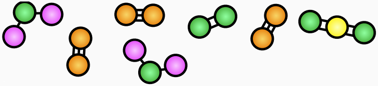
Formal model of distributed computation by collections of

identical, finite-state, and mobile agents

like

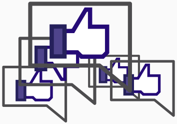


ad-hoc networks of devices

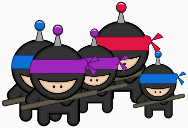


“soups” of molecules

(Chemical Reaction Networks)



people in social networks



...and ninjas!

Population protocols: formal model

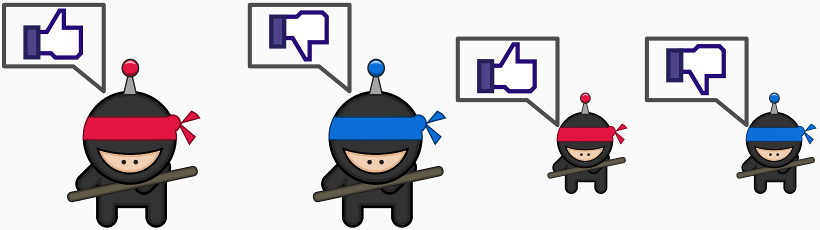
Angluin, Aspnes et al. PODC'04

- *States:* finite set Q
- *Opinions:* $O : Q \to \{0, 1\}$
- *Initial states:* $I \subseteq Q$
- *Transitions:* $T \subseteq Q^2 \times Q^2$

Population protocols: formal model

Angluin, Aspnes et al. PODC'04

- *States*: finite set Q
- *Opinions*: $O : Q \to \{0, 1\}$
- *Initial states*: $I \subseteq Q$
- *Transitions*: $T \subseteq Q^2 \times Q^2$



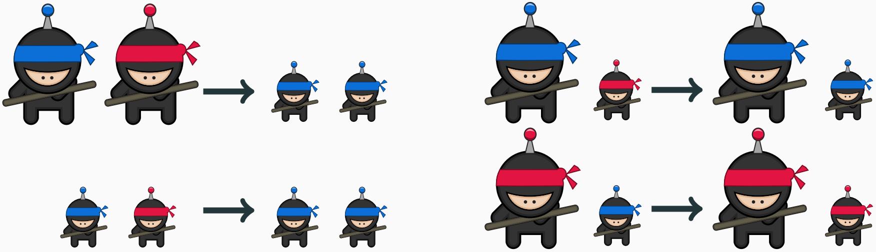
Population protocols: formal model

- *States*: finite set Q
- *Opinions*: $O : Q \to \{0, 1\}$
- *Initial states*: $I \subseteq Q$
- *Transitions*: $T \subseteq Q^2 \times Q^2$

Population protocols: formal model

Angluin, Aspnes et al. PODC'04

- *States:* finite set Q
- *Opinions:* $O : Q \to \{0, 1\}$
- *Initial states:* $I \subseteq Q$
- *Transitions:* $T \subseteq Q^2 \times Q^2$

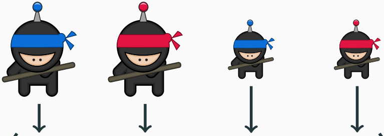


Population protocols: formal model

Angluin, Aspnes et al. PODC'04

- *States*: finite set Q
- *Opinions*: $O : Q \to \{0, 1\}$
- *Initial states*: $I \subseteq Q$
- *Transitions*: $T \subseteq Q^2 \times Q^2$

- *Configurations*: $Q \to \mathbb{N}$



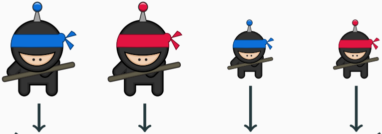
$(2 \quad 5 \quad 1 \quad 3)$

Population protocols: formal model

Angluin, Aspnes et al. PODC'04

- *States:* finite set Q
- *Opinions:* $O : Q \to \{0, 1\}$
- *Initial states:* $I \subseteq Q$
- *Transitions:* $T \subseteq Q^2 \times Q^2$

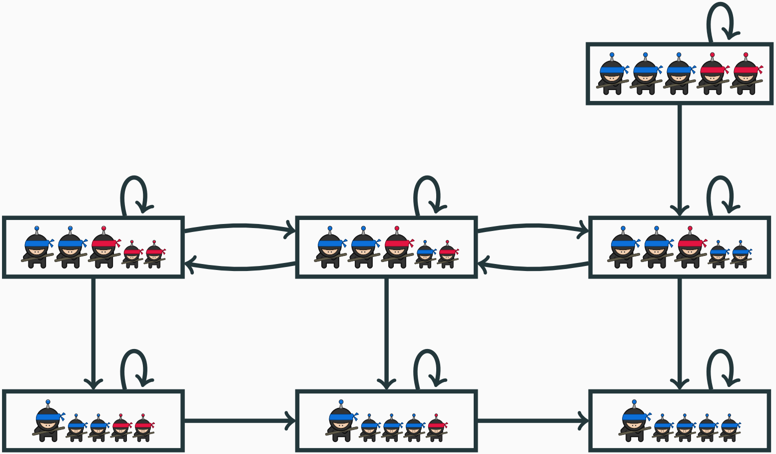
- *Configurations:* $Q \to \mathbb{N}$
- *Initial configurations:* $I \to \mathbb{N}$



$$(2 \quad 5 \quad 0 \quad 0)$$

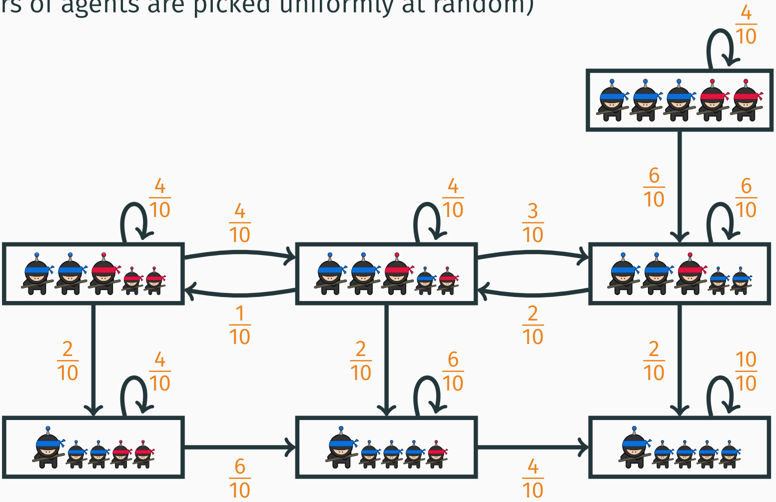
Population protocols: runs

Reachability graph for $(3, 2, 0, 0)$**:**



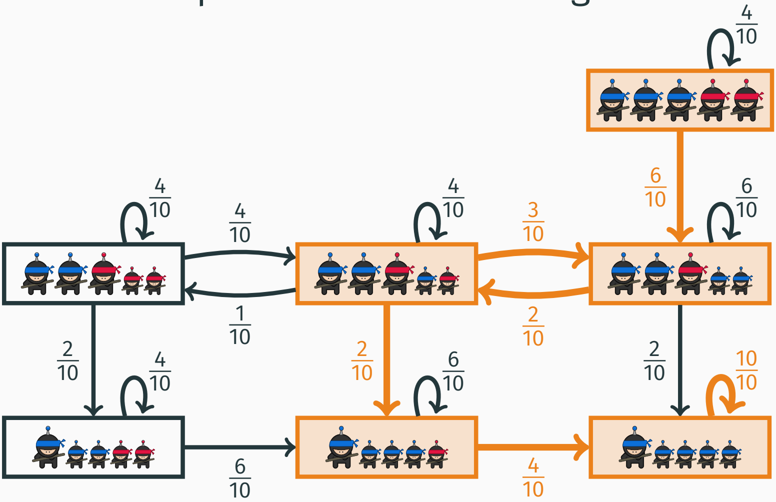
Underlying Markov chain:

(pairs of agents are picked uniformly at random)



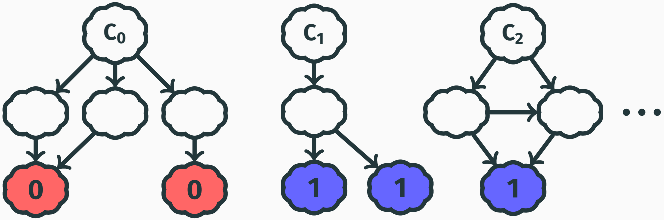
Population protocols: runs

Run : infinite path from initial configuration



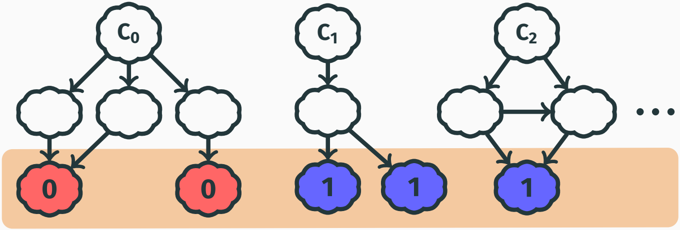
Population protocols: computing predicates

Protocol computes φ : InitC $\to \{0, 1\}$:
for every $C \in$ InitC, the runs starting at C reach **stable consensus** $\varphi(C)$ with probability 1.



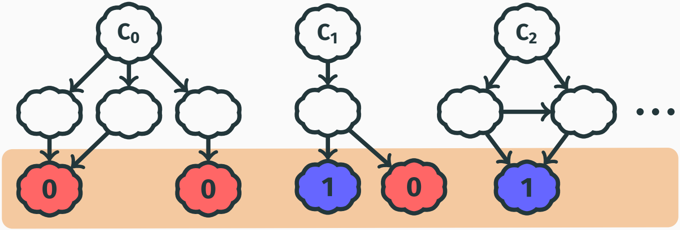
Population protocols: computing predicates

Protocol computes φ: InitC $\rightarrow \{0, 1\}$:
for every $C \in$ InitC, the runs starting at C
reach **stable consensus** $\varphi(C)$ with probability 1.



Protocol computes $\varphi(C_0) = 0, \varphi(C_1) = 1, \varphi(C_2) = 1, \ldots$

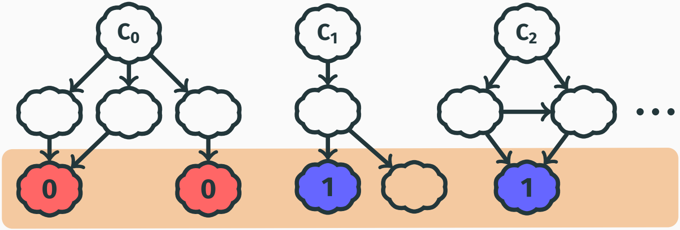
Protocol computes φ : InitC $\to \{0, 1\}$:
for every $C \in$ InitC, the runs starting at C
reach **stable consensus** $\varphi(C)$ with probability 1.



Protocol ill defined for C_1

Population protocols: computing predicates

Protocol computes φ : InitC $\to \{0,1\}$:
for every $C \in$ InitC, the runs starting at C
reach **stable consensus** $\varphi(C)$ with probability 1.



Protocol ill defined for C_1 (Sensei I's problem)

Population protocols: computing predicates

A protocol is well specified if it computes some predicate

A protocol is well specified if it computes some predicate

A protocol for a predicate φ is correct if it computes φ (in particular, correct protocols are well specified)

Sensei III's questions

What predicates can we compute?

How fast can we compute them?

How succinctly can we compute them?

How can I check correctness?

How can I check efficiency?

To conclude ...

Expressive power

Angluin, Aspnes, Eisenstat Dist. Comp.'07

Population protocols compute precisely the predicates definable in Presburger arithmetic

Expressive power

Angluin, Aspnes, Eisenstat Dist. Comp.'07

Population protocols compute precisely the predicates definable in Presburger arithmetic

Presburger arithmetic

- Atomic formulas: $a_1 x_1 + \cdots + a_m x_m < b$
- Formulas: Close under boolean operations and quantification
- Formula $F(x_1, \ldots, x_n)$ interpreted over $\mathbb{N}^n$
- Predicate $\varphi \colon \mathbb{N}^n \to \{0, 1\}$ definable in Presburger arithmetic if there is formula $F(x_1, \ldots, x_n)$ s.t. for every $\mathbf{v} \in \mathbb{N}^n$: $\varphi(\mathbf{v}) = 1$ iff $F(\mathbf{v})$ holds .

Expressive power

Angluin, Aspnes, Eisenstat Dist. Comp.'07

Population protocols compute precisely the predicates definable in Presburger arithmetic

Quantifier elimination

Every Presburger formula $F(x_1, \ldots, x_n)$ has an equivalent quantifier-free formula: A boolean combination of threshold and modulo predicates

$$a_1x_1 + \cdots + a_nx_n < b \quad a_1x_1 + \cdots + a_nx_n \equiv b \pmod{c}$$

with coefficients in $\mathbb{Z}$

Expressive power

Angluin, Aspnes, Eisenstat Dist. Comp.'07

Population protocols compute precisely the predicates definable in Presburger arithmetic

Proof:

1) PPs compute all Presburger predicates

Since Presburger arithmetic has quantifier elimination, it suffices to:

- Exhibit PPs for threshold and modulo predicates
- Prove that predicates computable by PPs are closed under negation and conjunction

Threshold predicates: A protocol for 2x $-$ 3y $<$ 5

A first protocol with infinitely many states

Threshold predicates: A protocol for $2x - 3y < 5$

A first protocol with infinitely many states

Initially: x agents, each with 2 euros, and
y agents, each with -3 euros (debt of 3 euros)

Agents must compute if total wealth less than 5 euros

Threshold predicates: A protocol for $2x - 3y < 5$

A first protocol with infinitely many states

Initially: x agents, each with 2 euros, and
y agents, each with -3 euros (debt of 3 euros)

Agents must compute if total wealth less than 5 euros

States: each agent has a number of euros
is active or passive (A or P)
has opinion on result (<5 or ≥ 5)

Threshold predicates: A protocol for $2x - 3y < 5$

A first protocol with infinitely many states

Initially: x agents, each with 2 euros, and
y agents, each with -3 euros (debt of 3 euros)

Agents must compute if total wealth less than 5 euros

States: each agent has a number of euros
is active or passive (A or P)
has opinion on result ($<$5 or $\geq$5)

Examples: (7, A, $\geq$5), (-2, P, $\geq$5)

Threshold predicates: A protocol for $2x - 3y < 5$

A first protocol with infinitely many states

Initially: x agents, each with 2 euros, and
y agents, each with -3 euros (debt of 3 euros)

Agents must compute if total wealth less than 5 euros

Interactions:

Threshold predicates: A protocol for $2x - 3y < 5$

A first protocol with infinitely many states

Initially: x agents, each with 2 euros, and
y agents, each with -3 euros (debt of 3 euros)

Agents must compute if total wealth less than 5 euros

Interactions:

- Two active agents: One agent transfers its money to the other and goes passive; new opinions given by wealth of active agent.

$$(7, A, \geq 5), (-4, A, <5) \mapsto (3, A, <5), (0, P, <5)$$
$$(2, A, <5), (4, A, <5) \mapsto (6, A, \geq 5), (0, P, \geq 5)$$

Threshold predicates: A protocol for $2x - 3y < 5$

A first protocol with infinitely many states

Initially: x agents, each with 2 euros, and
y agents, each with -3 euros (debt of 3 euros)

Agents must compute if total wealth less than 5 euros

Interactions:

- Two active agents: One agent transfers its money to the other and goes passive; new opinions given by wealth of active agent.

$$(7, A, \geq 5), (-4, A, <5) \mapsto (3, A, <5), (0, P, <5)$$
$$(2, A, <5), (4, A, <5) \mapsto (6, A, \geq 5), (0, P, \geq 5)$$

- One active, one passive agent: Same.
- Two passive agents: Nothing happens.

Threshold predicates: A protocol for $2x - 3y < 5$

A first protocol with infinitely many states

Initially: x agents, each with 2 euros, and
y agents, each with -3 euros (debt of 3 euros)

Agents must compute if total wealth less than 5 euros

Correctness:

Threshold predicates: A protocol for $2x - 3y < 5$

A first protocol with infinitely many states

Initially: x agents, each with 2 euros, and
y agents, each with -3 euros (debt of 3 euros)

Agents must compute if total wealth less than 5 euros

Correctness:

- Total wealth is an invariant

Threshold predicates: A protocol for $2x - 3y < 5$

A first protocol with infinitely many states

Initially: x agents, each with 2 euros, and
y agents, each with -3 euros (debt of 3 euros)

Agents must compute if total wealth less than 5 euros

Correctness:

- Total wealth is an invariant
- Eventually only one active agent left (leader).
  Leader has collected all wealth and has correct opinion

Threshold predicates: A protocol for $2x - 3y < 5$

A first protocol with infinitely many states

Initially: x agents, each with 2 euros, and
y agents, each with -3 euros (debt of 3 euros)

Agents must compute if total wealth less than 5 euros

Correctness:

- Total wealth is an invariant
- Eventually only one active agent left (leader).
  Leader has collected all wealth and has correct opinion
- Leader eventually changes opinions of all passive agents to correct one

Threshold predicates: A protocol for $2x - 3y < 5$

A first protocol with infinitely many states

Initially: x agents, each with 2 euros, and
y agents, each with -3 euros (debt of 3 euros)

Agents must compute if total wealth less than 5 euros

Final protocol with finitely many states

Initially: x agents, each with 2 euros, and
y agents, each with -3 euros (debt of 3 euros).

Agents must compute if total wealth less than 5 euros.

Threshold predicates: A protocol for $2x - 3y < 5$

Final protocol with finitely many states

Initially: x agents, each with 2 euros, and
y agents, each with -3 euros (debt of 3 euros).

Agents must compute if total wealth less than 5 euros.

States: agents now can only have -3,-2, ..., 4, 5 euros

Threshold predicates: A protocol for $2x - 3y < 5$

Final protocol with finitely many states

Initially: x agents, each with 2 euros, and
y agents, each with -3 euros (debt of 3 euros).

Agents must compute if total wealth less than 5 euros.

States: agents now can only have -3,-2, ..., 4, 5 euros

Interactions:

Threshold predicates: A protocol for $2x - 3y < 5$

Final protocol with finitely many states

Initially: x agents, each with 2 euros, and
y agents, each with -3 euros (debt of 3 euros).

Agents must compute if total wealth less than 5 euros.

States: agents now can only have -3,-2, ..., 4, 5 euros

Interactions:

- Two active agents: One agent transfers as much as possible money or debt to the other and goes passive; new opinions given by wealth of active agent.

$$(2, A, <5), (4, A, <5) \quad \mapsto \quad (5, A, \geq 5), (1, P, \geq 5)$$
$$(-2, A, <5), (-2, A, \geq 5) \quad \mapsto \quad (-3, A, <5), (-1, P, <5)$$

Threshold predicates: A protocol for $2x - 3y < 5$

Final protocol with finitely many states

Initially: x agents, each with 2 euros, and
y agents, each with -3 euros (debt of 3 euros).

Agents must compute if total wealth less than 5 euros.

States: agents now can only have -3,-2, ..., 4, 5 euros

Interactions:

- Two active agents: One agent transfers as much as possible money or debt to the other and goes passive; new opinions given by wealth of active agent.

$$(2, A, <5), (4, A, <5) \mapsto (5, A, \geq 5), (1, P, \geq 5)$$
$$(-2, A, <5), (-2, A, \geq 5) \mapsto (-3, A, <5), (-1, P, <5)$$

- One active, one passive or both passive: As before.

Threshold predicates: A protocol for $2x - 3y < 5$

Final protocol with finitely many states

Initially: x agents, each with 2 euros, and
y agents, each with -3 euros (debt of 3 euros).

Agents must compute if total wealth less than 5 euros.

States: agents now can only have -3,-2, ..., 4, 5 euros

Correctness:

Threshold predicates: A protocol for $2x - 3y < 5$

Final protocol with finitely many states

Initially: x agents, each with 2 euros, and
y agents, each with -3 euros (debt of 3 euros).

Agents must compute if total wealth less than 5 euros.

States: agents now can only have -3,-2, ..., 4, 5 euros

Correctness:

- Total wealth still invariant, and eventually one leader

Threshold predicates: A protocol for $2x - 3y < 5$

Final protocol with finitely many states

Initially: x agents, each with 2 euros, and
y agents, each with -3 euros (debt of 3 euros).

Agents must compute if total wealth less than 5 euros.

States: agents now can only have -3,-2, ..., 4, 5 euros

Correctness:

- Total wealth still invariant, and eventually one leader
- Eventually no agents in debt, or no agents with money.

Threshold predicates: A protocol for $2x - 3y < 5$

Final protocol with finitely many states

Initially: x agents, each with 2 euros, and
y agents, each with -3 euros (debt of 3 euros).

Agents must compute if total wealth less than 5 euros.

States: agents now can only have -3,-2, ..., 4, 5 euros

Correctness:

- Total wealth still invariant, and eventually one leader
- Eventually no agents in debt, or no agents with money.
- Eventually leader has $\max(-3, \min(5, \text{total-wealth}))$ euros and correct opinion

Threshold predicates: A protocol for $2x - 3y < 5$

Final protocol with finitely many states

Initially: x agents, each with 2 euros, and
y agents, each with -3 euros (debt of 3 euros).

Agents must compute if total wealth less than 5 euros.

States: agents now can only have -3,-2, ..., 4, 5 euros

Correctness:

- Total wealth still invariant, and eventually one leader
- Eventually no agents in debt, or no agents with money.
- Eventually leader has $\max(-3, \min(5, \text{total-wealth}))$ euros and correct opinion
- Eventually leader changes opinions of all passive agents to correct one

Remainder predicates: A protocol for $2x - 3y \equiv 1 \bmod 5$

States: each agent keeps a residue $0, 1, 2, 3, 4$
is active or passive (A or P)
has opinion on result ($\equiv$1, $\not\equiv$1)

Examples: (2, A, $\not\equiv$1), (0, P, $\equiv$1)

Remainder predicates: A protocol for $2x - 3y \equiv 1 \bmod 5$

States: each agent keeps a residue $0, 1, 2, 3, 4$
is active or passive (A or P)
has opinion on result ($\equiv$1, $\not\equiv$1)

Examples: (2, A, $\not\equiv$1), (0, P, $\equiv$1)

Initially:

x active agents, each with residue 2

y active agents, each with residue 2 as well ($-3 \equiv 2 \bmod 5$)

Agents compute total wealth modulo 5

Remainder predicates: A protocol for $2x - 3y \equiv 1 \bmod 5$

States: each agent keeps a residue $0, 1, 2, 3, 4$
is active or passive (A or P)
has opinion on result ($\equiv$1, $\not\equiv$1)

Examples: (2, A, $\not\equiv$1), (0, P, $\equiv$1)

Interactions:

Remainder predicates: A protocol for $2x - 3y \equiv 1 \bmod 5$

States: each agent keeps a residue $0, 1, 2, 3, 4$
is active or passive (A or P)
has opinion on result ($\equiv$1, $\not\equiv$1)

Examples: (2, A, $\not\equiv$1), (0, P, $\equiv$1)

Interactions:

- Two active agents: One agent "transfers its residue to the other" and goes passive; new opinions given by residue of active agent.

$$(2, A, \not\equiv 1)\,,(4, A, \not\equiv 1) \quad \mapsto \quad (1, A, \equiv 1)\,,(0, P, \equiv 1)$$

Remainder predicates: A protocol for $2x - 3y \equiv 1 \bmod 5$

States: each agent keeps a residue $0, 1, 2, 3, 4$
is active or passive (A or P)
has opinion on result ($\equiv 1$, $\not\equiv 1$)

Examples: $(2, A, \not\equiv 1)$, $(0, P, \equiv 1)$

Interactions:

- Two active agents: One agent "transfers its residue to the other" and goes passive; new opinions given by residue of active agent.

$$(2, A, \not\equiv 1), (4, A, \not\equiv 1) \quad \mapsto \quad (1, A, \equiv 1), (0, P, \equiv 1)$$

- One active, one passive or two passive: As before.

Remainder predicates: A protocol for $2x - 3y \equiv 1 \bmod 5$

States: each agent keeps a residue $0, 1, 2, 3, 4$
is active or passive (A or P)
has opinion on result ($\equiv 1$, $\not\equiv 1$)

Examples: (2, A, $\not\equiv 1$), (0, P, $\equiv 1$)

Correctness:

- Residue of total wealth is an invariant
- Eventually only one active agent left (leader).
  Leader has the correct residue and the correct opinion
- Leader eventually changes opinions of all passive agents to correct one

Closure under boolean operations

Computable predicates are closed under negation

Closure under boolean operations

Computable predicates are closed under negation

Invert the opinions of the states

Closure under boolean operations

Computable predicates are closed under negation

Invert the opinions of the states

Computable predicates are closed under conjunction

Let agents “multitask” to simultaneously compute the two predicates

Closure under boolean operations

Computable predicates are closed under negation

Invert the opinions of the states

Computable predicates are closed under conjunction

Let agents “multitask” to simultaneously compute the two predicates

States: pairs (q, r), where q and r states of the first and second protocols, resp.

Closure under boolean operations

Computable predicates are closed under negation

Invert the opinions of the states

Computable predicates are closed under conjunction

Let agents "multitask" to simultaneously compute the two predicates

States: pairs (q, r), where q and r states of the first and second protocols, resp.

Transitions:

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

$$r_1, r_2 \mapsto r_3, r_4$$

Closure under boolean operations

Computable predicates are closed under negation

Invert the opinions of the states

Computable predicates are closed under conjunction

Let agents “multitask” to simultaneously compute the two predicates

States: pairs (q, r), where q and r states of the first and second protocols, resp.

Transitions:

$$
\begin{aligned}
q_1, q_2 &\mapsto q_3, q_4 \\
r_1, r_2 &\mapsto r_3, r_4 \\
(q_1, r_1), (q_2, r_2) &\mapsto (q_3, r_3), (q_4, r_4)
\end{aligned}
$$

Closure under boolean operations

Computable predicates are closed under negation

Invert the opinions of the states

Computable predicates are closed under conjunction

Let agents "multitask" to simultaneously compute the two predicates

States: pairs (q, r), where q and r states of the first and second protocols, resp.

Transitions:

$$\begin{aligned} q_1, q_2 &\mapsto q_3, q_4 \\ r_1, r_2 &\mapsto r_3, r_4 \\ (q_1, r_1), (q_2, r_2) &\mapsto (q_3, r_3), (q_4, r_4) \end{aligned}$$

Opinion of (q, r): $O(q) \wedge O(r)$

Expressive power

Angluin, Aspnes, Eisenstat Dist. Comp.'07

Population protocols compute precisely the predicates definable in Presburger arithmetic

Expressive power

Angluin, Aspnes, Eisenstat Dist. Comp.'07

Population protocols compute precisely the predicates definable in Presburger arithmetic

Proof:

2) PPs only compute Presburger predicates

Expressive power

Angluin, Aspnes, Eisenstat Dist. Comp.'07

Population protocols compute precisely the predicates definable in Presburger arithmetic

Proof:

2) PPs only compute Presburger predicates

- Much harder!

Expressive power

Angluin, Aspnes, Eisenstat Dist. Comp.'07

Population protocols compute precisely the predicates definable in Presburger arithmetic

Proof:

2) PPs only compute Presburger predicates

- Much harder!
- Dist. Comp. '07 proof is "non-constructive"

Expressive power

Angluin, Aspnes, Eisenstat Dist. Comp.'07

Population protocols compute precisely the predicates definable in Presburger arithmetic

Proof:

2) PPs only compute Presburger predicates

- Much harder!
- Dist. Comp. '07 proof is "non-constructive"
- "Constructive" proof by E., Ganty, Leroux, Majumdar Acta Inf.'17
- More on this later!

Beyond Presburger predicates

- PPs can only compute predicates in DSPACE($\log \log n$)

  Meaning: if n agents can compute $\varphi(n)$ then there is a deterministic TM that on input the unary encoding of n computes $\varphi(n)$ using $\log \log n$ space

  Proof: Presburger languages are regular

Beyond Presburger predicates

- PPs can only compute predicates in DSPACE($\log \log n$)
- Several ways to increase power to NSPACE($\log n$):

Beyond Presburger predicates

- PPs can only compute predicates in DSPACE($\log \log n$)
- Several ways to increase power to NSPACE($\log n$):
  - Absence detectors — Michail, Spirakis JPDC'16

    Agents detect absence of agents in states.

Beyond Presburger predicates

- PPs can only compute predicates in DSPACE($\log \log n$)
- Several ways to increase power to NSPACE($\log n$):
  - Absence detectors — Michail, Spirakis JPDC'16

    Agents detect absence of agents in states.
  - Clocked protocols — Aspnes PODC'17

    Agents detect if stable consensus has been reached.

Beyond Presburger predicates

- PPs can only compute predicates in DSPACE($\log \log n$)
- Several ways to increase power to NSPACE($\log n$):
  - Absence detectors — Michail, Spirakis JPDC'16
    Agents detect absence of agents in states.
  - Clocked protocols — Aspnes PODC'17
    Agents detect if stable consensus has been reached.
  - Broadcast protocols — Blondin, Esparza, Jaax, CONCUR'19
    Agents can broadcast a signal to all agents.

Beyond Presburger predicates

- PPs can only compute predicates in DSPACE($\log \log n$)
- Several ways to increase power to NSPACE($\log n$):
  - Absence detectors — Michail, Spirakis JPDC'16
    Agents detect absence of agents in states.
  - Clocked protocols — Aspnes PODC'17
    Agents detect if stable consensus has been reached.
  - Broadcast protocols — Blondin, Esparza, Jaax, CONCUR'19
    Agents can broadcast a signal to all agents.

For all three: n agents can simulate a NCM with counters bounded by n^c, and so an NTM using logspace in n

Beyond Presburger predicates

Increasing the expressive power to NSPACE($n \log n$) or NSPACE(n^2):

- Community protocols — Guerraoui, Ruppert ICALP'09
    - Agents have unique identities (integers)
    - Agents can store a fixed number of identities in registers
    - Agents can only compare identities according to $<$
    - New states depends on old states and register contents

Beyond Presburger predicates

Increasing the expressive power to NSPACE($n \log n$) or NSPACE(n^2):

- Community protocols — Guerraoui, Ruppert ICALP'09
  - Agents have unique identities (integers)
  - Agents can store a fixed number of identities in registers
  - Agents can only compare identities according to $<$
  - New states depends on old states and register contents
- Mediated protocols — Michail et al. ICALP'09, TCS'11

  No identities, but channels have state.

Sensei III's questions

What predicates can we compute?

How fast can we compute them?

How succinctly can we compute them?

How can I check correctness?

How can I check efficiency?

To conclude ...

Efficiency

Efficiency measured by the expected number of interactions until stable consensus: $Inter(n)$

Efficiency

Efficiency measured by the expected number of interactions until stable consensus: *Inter*(n)

Expected parallel time to consensus depends on the concurrency model

Efficiency

Efficiency measured by the expected number of interactions until stable consensus: $\mathit{Inter}(n)$

Expected parallel time to consensus depends on the concurrency model

Most popular model:

- A communication channel for every pair of agents
- For each pair, number of communications follows a Poisson distribution with given rate
- Important advantage of the model: expected parallel time $\mathit{Time}(n)$ satisfies

$$\mathit{Time}(n) = \mathit{Inter}(n)/n$$

Efficiency

Angluin, Aspnes *et al.* **, PODC'04**

Every Presburger predicate is computable by a protocol in $O(n \log n)$ time

Efficiency

Angluin, Aspnes *et al.* , PODC'04

Every Presburger predicate is computable by a protocol in $O(n \log n)$ time

Angluin, Aspnes, Eisenstat Dist.Comp.'08

Every Presburger predicate is computable by a protocol with a leader in $O(\log^{O(1)}(n))$ time

Efficiency

Angluin, Aspnes *et al.* , PODC'04

Every Presburger predicate is computable by a protocol in $O(n \log n)$ time

Angluin, Aspnes, Eisenstat Dist.Comp.'08

Every Presburger predicate is computable by a protocol with a leader in $O(\log^{O(1)}(n))$ time

Can a leader be elected in $O(\log^{O(1)}(n))$ time ?

Efficiency

Angluin, Aspnes *et al.* **, PODC'04**

Every Presburger predicate is computable by a protocol in $O(n \log n)$ time

Angluin, Aspnes, Eisenstat Dist.Comp.'08

Every Presburger predicate is computable by a protocol with a leader in $O(\log^{O(1)}(n))$ time

Can a leader be elected in $O(\log^{O(1)}(n))$ time ?

Doty and Soloveichik, DISC '15 and Dist. Comp. '18

Electing a leader takes $\Omega(n)$ time.

Efficiency

Alistarh *et al.* consider families $\{\mathcal{P}_n\}_{n=1}^{\infty}$ of protocols, where $\mathcal{P}_n$ is the protocol used for inputs with n agents.

Alistarh et al. PODC '15

There is a uniform family of protocols with $O(n)$ states that computes majority (without ties) in $O(\log^{O(1)}(n))$ time.

Efficiency

Alistarh *et al.* consider families $\{\mathcal{P}_n\}_{n=1}^{\infty}$ of protocols, where $\mathcal{P}_n$ is the protocol used for inputs with n agents.

Alistarh et al. PODC '15

There is a uniform family of protocols with $O(n)$ states that computes majority (without ties) in $O(\log^{O(1)}(n))$ time.

Alistarh et al. SODA '17

Every (uniform or nonuniform) family of protocols with $O(\log \log n)$ states that correctly elects a leader or computes majority takes $\Omega(n/\text{polylog } n)$ time.

Efficiency

Alistarh *et al.* consider families $\{\mathcal{P}_n\}_{n=1}^{\infty}$ of protocols, where $\mathcal{P}_n$ is the protocol used for inputs with n agents.

Alistarh et al. PODC ’15

There is a uniform family of protocols with $O(n)$ states that computes majority (without ties) in $O(\log^{O(1)}(n))$ time.

Alistarh et al. SODA ’17

Every (uniform or nonuniform) family of protocols with $O(\log \log n)$ states that correctly elects a leader or computes majority takes $\Omega(n/\text{polylog } n)$ time.

Alistarh et al. SODA ’18

There exists a uniform family of protocols with $O(\log^2 n)$ states that computes majority in $\mathcal{O}(\log^{O(1)}(n))$ time.

Sensei III's questions

What predicates can we compute?

How fast can we compute them?

How succinctly can we compute them?

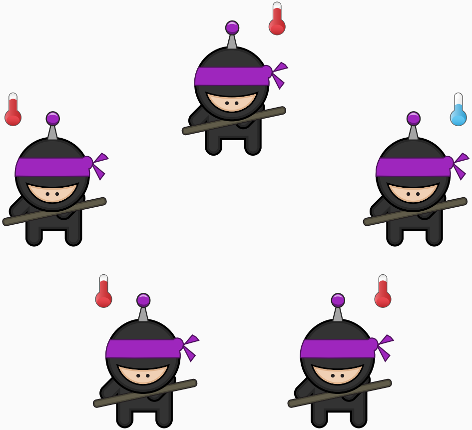
How can I check correctness?

How can I check efficiency?

To conclude ...

Succinctness–An Example

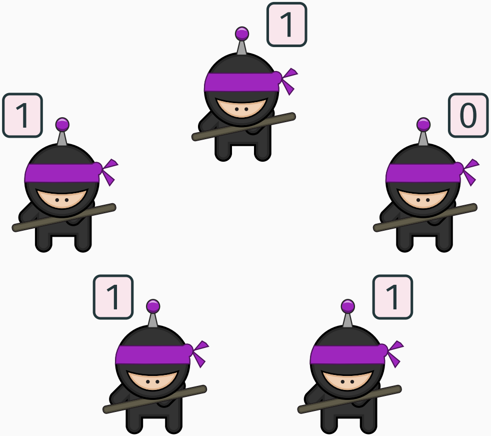
Protocol for: Are there at least 4 sick ninjas?



Succinctness–An Example

Protocol for: Are there at least 4 sick ninjas?

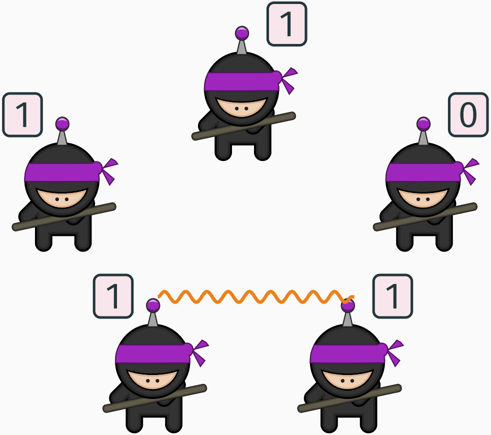
- Each ninja is in a state of $\{0, 1, 2, 3, 4\}$
- Initially, sick ninjas in state 1, healthy ninjas in state 0
- $(m, n) \mapsto (m + n, 0)$
  if $m + n < 4$
- $(m, n) \mapsto (4, 4)$
  if $m + n \geq 4$



Succinctness–An Example

Protocol for: Are there at least 4 sick ninjas?

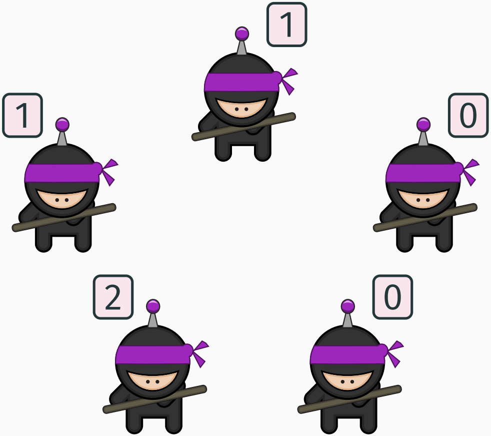
- Each ninja is in a state of $\{0, 1, 2, 3, 4\}$
- Initially, sick ninjas in state 1, healthy ninjas in state 0
- $(m, n) \mapsto (m + n, 0)$
  if $m + n < 4$
- $(m, n) \mapsto (4, 4)$
  if $m + n \geq 4$



Succinctness–An Example

Protocol for: Are there at least 4 sick ninjas?

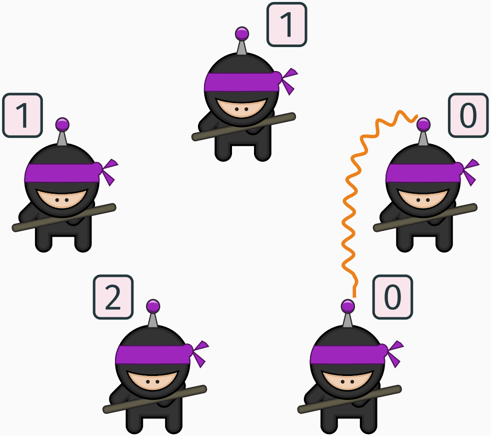
- Each ninja is in a state of $\{0, 1, 2, 3, 4\}$
- Initially, sick ninjas in state 1, healthy ninjas in state 0
- $(m, n) \mapsto (m + n, 0)$ if $m + n < 4$
- $(m, n) \mapsto (4, 4)$ if $m + n \geq 4$



Succinctness–An Example

Protocol for: Are there at least 4 sick ninjas?

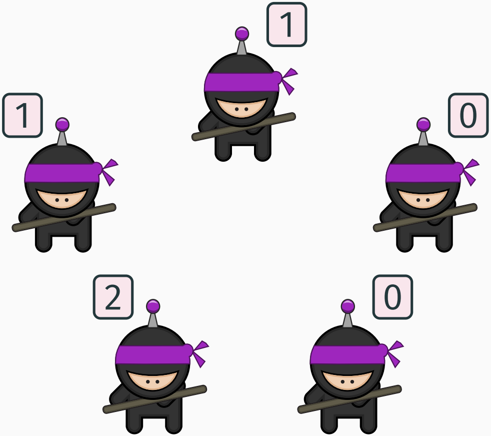
- Each ninja is in a state of $\{0, 1, 2, 3, 4\}$
- Initially, sick ninjas in state 1, healthy ninjas in state 0
- $(m, n) \mapsto (m + n, 0)$ if $m + n < 4$
- $(m, n) \mapsto (4, 4)$ if $m + n \geq 4$



Succinctness–An Example

Protocol for: Are there at least 4 sick ninjas?

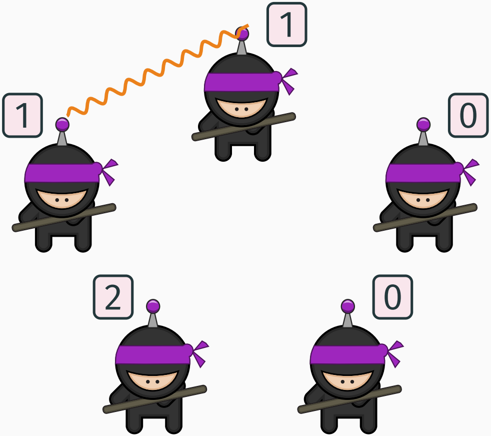
- Each ninja is in a state of $\{0, 1, 2, 3, 4\}$
- Initially, sick ninjas in state 1, healthy ninjas in state 0
- $(m, n) \mapsto (m+n, 0)$
  if $m + n < 4$
- $(m, n) \mapsto (4, 4)$
  if $m + n \geq 4$



Succinctness–An Example

Protocol for: Are there at least 4 sick ninjas?

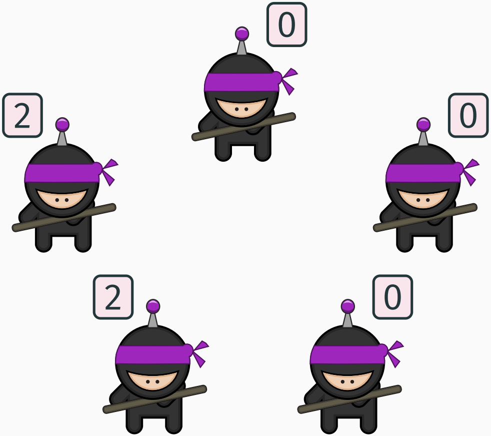
- Each ninja is in a state of $\{0, 1, 2, 3, 4\}$
- Initially, sick ninjas in state 1, healthy ninjas in state 0
- $(m, n) \mapsto (m + n, 0)$
  if $m + n < 4$
- $(m, n) \mapsto (4, 4)$
  if $m + n \geq 4$



Succinctness–An Example

Protocol for: Are there at least 4 sick ninjas?

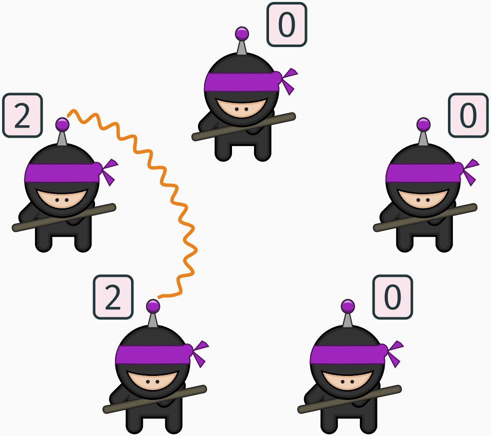
- Each ninja is in a state of $\{0, 1, 2, 3, 4\}$
- Initially, sick ninjas in state 1, healthy ninjas in state 0
- $(m, n) \mapsto (m + n, 0)$ if $m + n < 4$
- $(m, n) \mapsto (4, 4)$ if $m + n \geq 4$



Succinctness–An Example

Protocol for: Are there at least 4 sick ninjas?

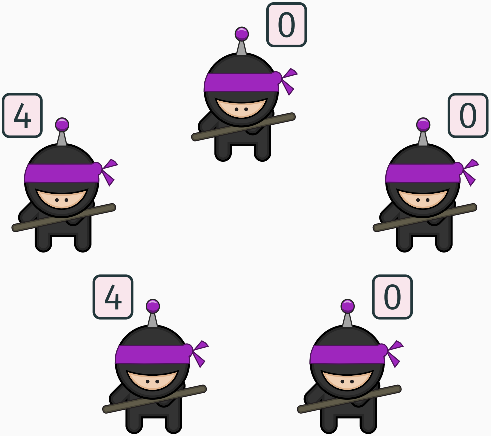
- Each ninja is in a state of $\{0, 1, 2, 3, 4\}$
- Initially, sick ninjas in state 1, healthy ninjas in state 0
- $(m, n) \mapsto (m + n, 0)$
  if $m + n < 4$
- $(m, n) \mapsto (4, 4)$
  if $m + n \geq 4$



Succinctness–An Example

Protocol for: Are there at least 4 sick ninjas?

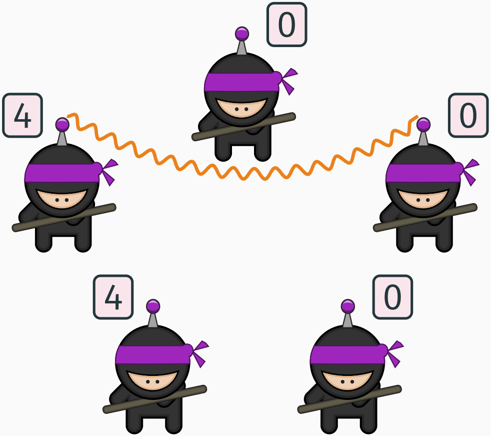
- Each ninja is in a state of $\{0, 1, 2, 3, 4\}$
- Initially, sick ninjas in state 1, healthy ninjas in state 0
- $(m, n) \mapsto (m + n, 0)$
  if $m + n < 4$
- $(m, n) \mapsto (4, 4)$
  if $m + n \geq 4$



Succinctness–An Example

Protocol for: Are there at least 4 sick ninjas?

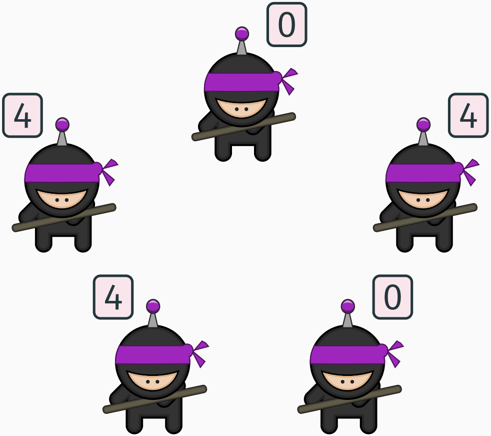
- Each ninja is in a state of $\{0, 1, 2, 3, 4\}$
- Initially, sick ninjas in state 1, healthy ninjas in state 0
- $(m, n) \mapsto (m + n, 0)$
  if $m + n < 4$
- $(m, n) \mapsto (4, 4)$
  if $m + n \geq 4$



Succinctness–An Example

Protocol for: Are there at least 4 sick ninjas?

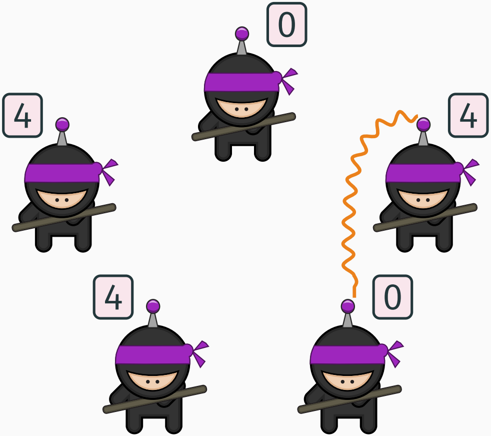
- Each ninja is in a state of $\{0, 1, 2, 3, 4\}$
- Initially, sick ninjas in state 1, healthy ninjas in state 0
- $(m, n) \mapsto (m + n, 0)$
  if $m + n < 4$
- $(m, n) \mapsto (4, 4)$
  if $m + n \geq 4$



Succinctness–An Example

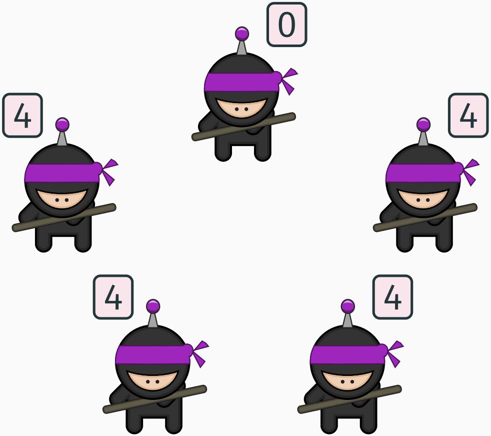
Protocol for: Are there at least 4 sick ninjas?

- Each ninja is in a state of $\{0, 1, 2, 3, 4\}$
- Initially, sick ninjas in state 1, healthy ninjas in state 0
- $(m, n) \mapsto (m + n, 0)$
  if $m + n < 4$
- $(m, n) \mapsto (4, 4)$
  if $m + n \geq 4$



Protocol for: Are there at least 4 sick ninjas?

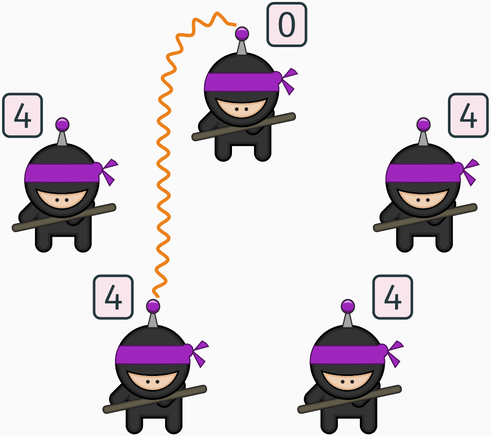
- Each ninja is in a state of $\{0, 1, 2, 3, 4\}$
- Initially, sick ninjas in state 1, healthy ninjas in state 0
- $(m, n) \mapsto (m + n, 0)$ if $m + n < 4$
- $(m, n) \mapsto (4, 4)$ if $m + n \geq 4$



Succinctness–An Example

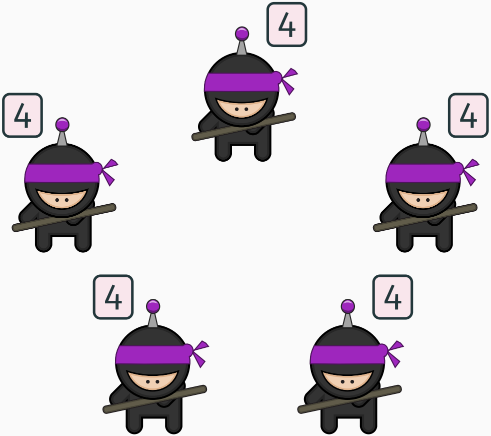
Protocol for: Are there at least 4 sick ninjas?

- Each ninja is in a state of $\{0, 1, 2, 3, 4\}$
- Initially, sick ninjas in state 1, healthy ninjas in state 0
- $(m, n) \mapsto (m + n, 0)$
  if $m + n < 4$
- $(m, n) \mapsto (4, 4)$
  if $m + n \geq 4$



Protocol for: Are there at least 4 sick ninjas?

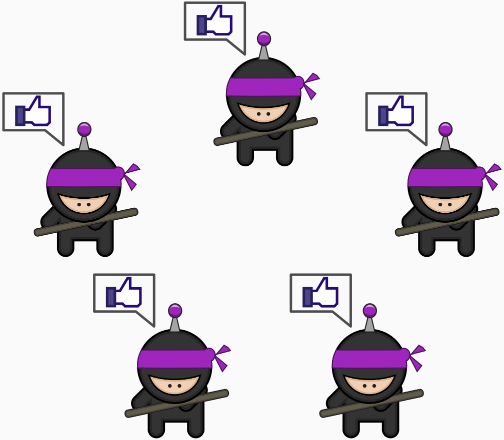
- Each ninja is in a state of $\{0, 1, 2, 3, 4\}$
- Initially, sick ninjas in state 1, healthy ninjas in state 0
- $(m, n) \mapsto (m + n, 0)$ if $m + n < 4$
- $(m, n) \mapsto (4, 4)$ if $m + n \geq 4$



Succinctness–An Example

Protocol for: Are there at least 4 sick ninjas?

- Each ninja is in a state of $\{0, 1, 2, 3, 4\}$
- Initially, sick ninjas in state 1, healthy ninjas in state 0
- $(m, n) \mapsto (m + n, 0)$
  if $m + n < 4$
- $(m, n) \mapsto (4, 4)$
  if $m + n \geq 4$



Sensei III's questions: Succinctness–An Example

Protocol for: Are there at least 2^ℓ sick ninjas?

- Each ninja is in a state of $\{0, 1, \ldots, 2^\ell - 1, 2^\ell\}$
- Initially, sick ninjas in state 1, healthy ninjas in state 0
- $(m, n) \mapsto (m + n, 0)$ if $m + n < 2^\ell$
- $(m, n) \mapsto (2^\ell, 2^\ell)$ if $m + n \geq 2^\ell$

Sensei III's questions: Succinctness–An Example

Protocol for: Are there at least 2^ℓ sick ninjas?

- Each ninja is in a state of $\{0, 1, \ldots, 2^\ell - 1, 2^\ell\}$
- Initially, sick ninjas in state 1, healthy ninjas in state 0
- $(m, n) \mapsto (m + n, 0)$ if $m + n < 2^\ell$
- $(m, n) \mapsto (2^\ell, 2^\ell)$ if $m + n \geq 2^\ell$

- Each ninja is in a state of $\{0, 2^0, \ldots, 2^{\ell-1}, 2^\ell\}$
- Initially, sick ninjas in state 2^0, healthy ninjas in state 0
- $(2^m, 2^m) \mapsto (2^{m+1}, 0)$ if $m + 1 \leq \ell$
- $(2^\ell, n) \mapsto (2^\ell, 2^\ell)$

Sensei III's questions: Succinctness–An Example

Protocol for: Are there at least 2^ℓ sick ninjas?

- Each ninja is in a state of $\{0, 1, \ldots, 2^\ell - 1, 2^\ell\}$
- Initially, sick ninjas in state 1, healthy ninjas in state 0
- $(m, n) \mapsto (m + n, 0)$
  if $m + n < 2^\ell$
- $(m, n) \mapsto (2^\ell, 2^\ell)$
  if $m + n \geq 2^\ell$

- Each ninja is in a state of $\{0, 2^0, \ldots, 2^{\ell-1}, 2^\ell\}$
- Initially, sick ninjas in state 2^0, healthy ninjas in state 0
- $(2^m, 2^m) \mapsto (2^{m+1}, 0)$
  if $m + 1 \leq \ell$
- $(2^\ell, n) \mapsto (2^\ell, 2^\ell)$
- Can be generalized to non-powers of 2

Succinctness

Just gave a protocol for **X ≥ c** with $\mathcal{O}(\log c)$ states.

Succinctness

Just gave a protocol for **X ≥ c** with $\mathcal{O}(\log c)$ states.

Is $\mathcal{O}(\log \log c)$ possible?

Succinctness

Just gave a protocol for **X ≥ c** with $\mathcal{O}(\log c)$ states.

Is $\mathcal{O}(\log \log c)$ possible?

Not for every **c** ...

Blondin, E., Jaax STACS'18

There exist infinitely many **c** such that every protocol for **X ≥ c** has at least $(\log \mathbf{c})^{1/4}$ states

Succinctness

Just gave a protocol for $\mathbf{X \geq c}$ with $\mathcal{O}(\log c)$ states.

Is $\mathcal{O}(\log \log c)$ possible?

Not for every **c** ...

Blondin, E., Jaax STACS'18

There exist infinitely many **c** such that every protocol for $\mathbf{X \geq c}$ has at least $(\log \mathbf{c})^{1/4}$ states

...but for some **c**, if we allow leaders:

Blondin, E., Jaax STACS'18

For infinitely many **c** there is a protocol with two leaders and $\mathcal{O}(\log \log \mathbf{c})$ states that computes $\mathbf{X \geq c}$

Succinctness

Blondin, E., Jaax STACS'18

For infinitely many **c** there is a protocol with two leaders and $\mathcal{O}(\log \log \mathbf{c})$ states that computes $\mathbf{X} \geq \mathbf{c}$

Proof:

Succinctness

Blondin, E., Jaax STACS'18

For infinitely many **c** there is a protocol with two leaders and $\mathcal{O}(\log \log \mathbf{c})$ states that computes $\mathbf{X} \geq \mathbf{c}$

Proof:

- **Mayr and Meyer '82**: For every n there is a commutative semigroup presentation and two elements s, t such that the shortest word α leading from s to t (i.e., $t = s\,\alpha$) has length $|\alpha| \geq 2^{2^n}$

Succinctness

Blondin, E., Jaax STACS'18

For infinitely many **c** there is a protocol with two leaders and $\mathcal{O}(\log\log \mathbf{c})$ states that computes **X ≥ c**

Proof:

- **Mayr and Meyer '82**: For every n there is a commutative semigroup presentation and two elements s, t such that the shortest word α leading from s to t (i.e., $t = s\,\alpha$) has length $|\alpha| \geq 2^{2^n}$
- Construct a protocol that "simulates" derivations in the semigroup

Succinctness

$\mathcal{O}(\log \log c)$ without leaders?

Succinctness

$\mathcal{O}(\log \log c)$ without leaders? Open

Succinctness

$\mathcal{O}(\log \log c)$ without leaders? Open

And $\mathcal{O}(\log \log \log c)$?

Succinctness

$\mathcal{O}(\log\log c)$ without leaders? Open

And $\mathcal{O}(\log\log\log c)$? Open

Succinctness

$\mathcal{O}(\log\log c)$ without leaders? Open

And $\mathcal{O}(\log\log\log c)$? Open

$\mathcal{O}(|\varphi|)$ states for all φ?

Succinctness

$\mathcal{O}(\log\log c)$ without leaders? Open

And $\mathcal{O}(\log\log\log c)$? Open

$\mathcal{O}(|\varphi|)$ states for all φ? Yes

Blondin et al., in preparation

Sensei III's questions

What predicates can we compute?

How fast can we compute them?

How succinctly can we compute them?

How can I check correctness?

How can I check efficiency?

To conclude ...

Checking correctness

Protocols can become complex, even for B ≥ R:

Fast and Exact Majority in Population Protocols

Dan Alistarh
Microsoft Research

Rati Gelashvili*
MIT

Milan Vojnović
Microsoft Research

$$weight(x) = \begin{cases} |x| & \text{if } x \in StrongStates \text{ or } x \in WeakStates; \\ 1 & \text{if } x \in IntermediateStates. \end{cases}$$

$$sgn(x) = \begin{cases} 1 & \text{if } x \in \{+0, 1_d, \ldots, 1_1, 3, 5, \ldots, m\}; \\ -1 & \text{otherwise.} \end{cases}$$

$value(x) = sgn(x) \cdot weight(x)$

```
/* Functions for rounding state interactions */
```

$\phi(x) = -1_1$ if $x = -1$; 1_1 if $x = 1$; x, otherwise

$R_\downarrow(k) = \phi(k$ if k odd integer, $k - 1$ if k even)

$R_\uparrow(k) = \phi(k$ if k odd integer, $k + 1$ if k even)

$$Shift\text{-}to\text{-}Zero(x) = \begin{cases} -1_{j+1} & \text{if } x = -1_j \text{ for some index } j < d \\ 1_{j+1} & \text{if } x = 1_j \text{ for some index } j < d \\ x & \text{otherwise.} \end{cases}$$

$$Sign\text{-}to\text{-}Zero(x) = \begin{cases} +0 & \text{if } sgn(x) > 0 \\ -0 & \text{oherwise.} \end{cases}$$

procedure update$\langle x, y \rangle$
    if $(weight(x) > 0$ *and* $weight(y) > 1)$ ***or*** $(weight(y) > 0$ *and* $weight(x) > 1)$ **then**
        $x' \leftarrow R_\downarrow\left(\frac{value(x)+value(y)}{2}\right)$ and $y' \leftarrow R_\uparrow\left(\frac{value(x)+value(y)}{2}\right)$
    else if $weight(x) \cdot weight(y) = 0$ *and* $value(x) + value(y) > 0$ **then**
        if $weight(x) \neq 0$ **then** $x' \leftarrow Shift\text{-}to\text{-}Zero(x)$ and $y' \leftarrow Sign\text{-}to\text{-}Zero(x)$
        else $y' \leftarrow Shift\text{-}to\text{-}Zero(y)$ and $x' \leftarrow Sign\text{-}to\text{-}Zero(y)$
    else if $(x \in \{-1_d, +1_d\}$ *and* $weight(y) = 1$ *and* $sgn(x) \neq sgn(y))$ ***or***
            $(y \in \{-1_d, +1_d\}$ *and* $weight(x) = 1$ *and* $sgn(y) \neq sgn(x))$ **then**
        $x' \leftarrow -0$ and $y' \leftarrow +0$
    else
        $x' \leftarrow Shift\text{-}to\text{-}Zero(x)$ and $y' \leftarrow Shift\text{-}to\text{-}Zero(y)$

Checking correctness

Protocols can become complex, even for **B** ≥ **R**:

Fast and Exact Majority in Population Protocols

Dan Alistarh
Microsoft Research

Rati Gelashvili*
MIT

Milan Vojnović
Microsoft Research

$$weight(x) = \begin{cases} |x| & \text{if } x \in StrongStates \text{ or } x \in WeakStates; \\ 1 & \text{if } x \in IntermediateStates. \end{cases}$$

$$sgn(x) = \begin{cases} 1 & \text{if } x \in \{+0, 1_d, \ldots, 1_1, 3, 5, \ldots, m\}; \\ -1 & \text{otherwise.} \end{cases}$$

$value(x) = sgn(x) \cdot weight(x)$

`/* Functions for rounding state interactions */`

$\phi(x) = -1_1$ if $x = -1$; 1_1 if $x = 1$; x, otherwise

$R_\downarrow(k) = \phi(k$ if k odd integer, $k - 1$ if k even)

$R_\uparrow(k) = \phi(k$ if k odd integer, $k + 1$ if k even)

$$Shift\text{-}to\text{-}Zero(x) = \begin{cases} -1_{j+1} & \text{if } x = -1_j \text{ for some index } j < d \\ 1_{j+1} & \text{if } x = 1_j \text{ for some index } j < d \\ x & \text{otherwise.} \end{cases}$$

$$Sign\text{-}to\text{-}Zero(x) = \begin{cases} +0 & \text{if } sgn(x) > 0 \\ -0 & \text{oherwise.} \end{cases}$$

procedure update$\langle x, y \rangle$
  if $(weight(x) > 0$ *and* $weight(y) > 1)$ ***or*** $(weight(y) > 0$ *and* $weight(x) > 1)$ **then**
    $x' \leftarrow R_\downarrow\left(\frac{value(x)+value(y)}{2}\right)$ and $y' \leftarrow R_\uparrow\left(\frac{value(x)+value(y)}{2}\right)$
  else if $weight(x) \cdot weight(y) = 0$ *and* $value(x) + value(y) > 0$ **then**
    if $weight(x) \neq 0$ **then** $x' \leftarrow Shift\text{-}to\text{-}Zero(x)$ and $y' \leftarrow Sign\text{-}to\text{-}Zero(x)$
    else $y' \leftarrow Shift\text{-}to\text{-}Zero(y)$ and $x' \leftarrow Sign\text{-}to\text{-}Zero(y)$
  else if $(x \in \{-1_d, +1_d\}$ *and* $weight(y) = 1$ *and* $sgn(x) \neq sgn(y))$ ***or***
      $(y \in \{-1_d, +1_d\}$ *and* $weight(x) = 1$ *and* $sgn(y) \neq sgn(x))$ **then**
    $x' \leftarrow -0$ and $y' \leftarrow +0$
  else
    $x' \leftarrow Shift\text{-}to\text{-}Zero(x)$ and $y' \leftarrow Shift\text{-}to\text{-}Zero(y)$

How can we verify correctness automatically?

Checking correctness—Early days

Model checkers:

- `PAT`: model checker with global fairness

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

- `bp-ver`: graph exploration

  (Chatzigiannakis, Michail and Spirakis SSS'10)

- Conversion to counter machines + `PRISM`/`Spin`

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

Checking correctness—Early days

Model checkers:

- **PAT**: model checker with global fairness

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

- **bp-ver**: graph exploration

  (Chatzigiannakis, Michail and Spirakis SSS'10)

- Conversion to counter machines + **PRISM/Spin**

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

Only for populations of fixed size!

Checking correctness—Early days

Theorem provers:

- Verification with the interactive theorem prover **Coq**

(Deng and Monin TASE'09)

Checking correctness—Early days

Theorem provers:

- Verification with the interactive theorem prover **Coq**

(Deng and Monin TASE'09)

Not automatic!

Checking correctness—Early days

Theorem provers:

- Verification with the interactive theorem prover **Coq**

(Deng and Monin TASE'09)

Challenge: verifying automatically

<u>all</u> sizes

Checking correctness—Decidability

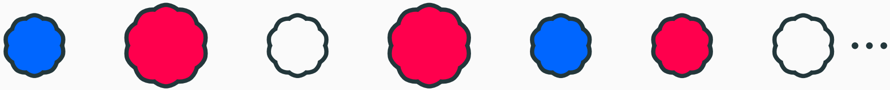
E., Ganty, Leroux, Majumdar Acta Inf.'17

It is decidable if a population protocol computes a given (Presburger) predicate.

Checking correctness—Decidability

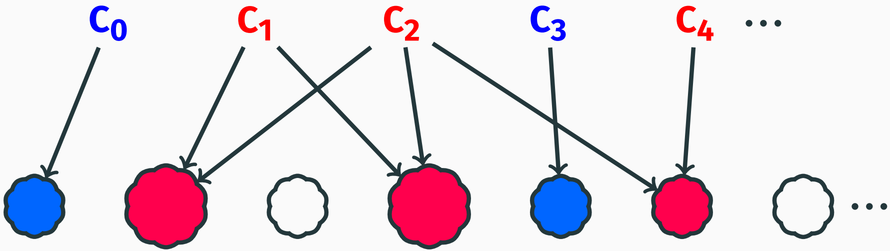
Initial configurations, colored with intended result

$C_0 \quad C_1 \quad C_2 \quad C_3 \quad C_4 \quad \cdots$



Bottom configurations, colored if consensus

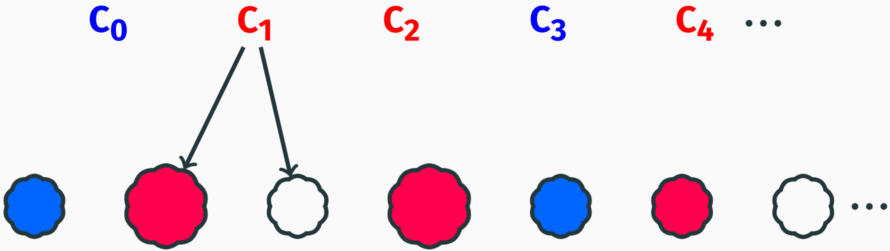
Initial configurations, colored with intended result



Bottom configurations, colored if consensus

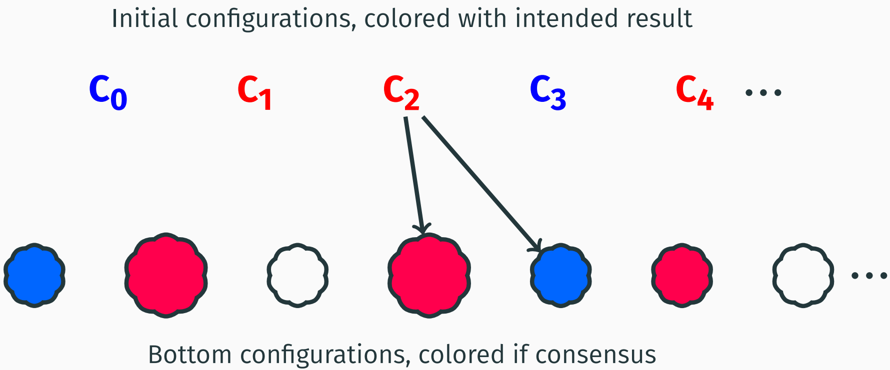
Correct protocol

Initial configurations, colored with intended result



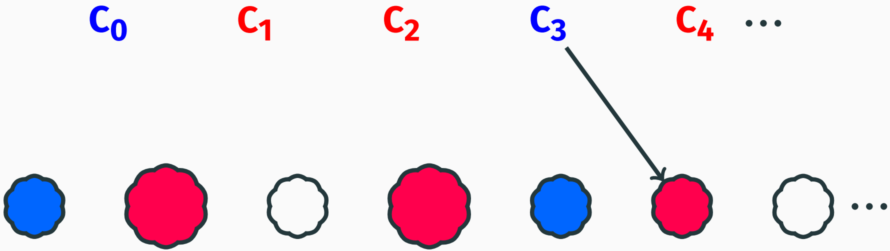
Bottom configurations, colored if consensus

Incorrect protocol: sometimes no result for C_1



Incorrect protocol: sometimes wrong for C_2

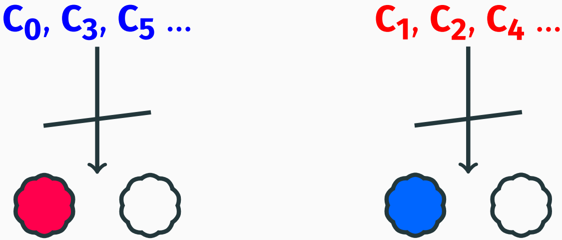
Initial configurations, colored with intended result



Bottom configurations, colored if consensus

Incorrect protocol: always wrong result for C_3

Checking correctness—Decidability

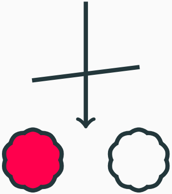


A protocol correctly computes the given predicate iff:

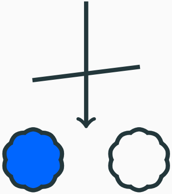
- no white or red SCCs reachable from blue initial configurations
- no white or blue SCCs reachable from red initial configurations

Correctness reduced to reachability question between infinite sets of configurations

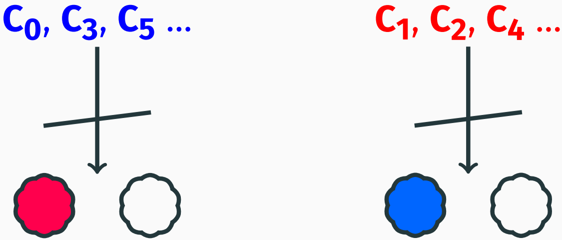
$C_0, C_3, C_5 \ldots$



$C_1, C_2, C_4 \ldots$



Define: sets $\mathbf{I}$, $\mathbf{I_1}$ and $\mathbf{I_0}$ of initial configurations
sets $\mathbf{B}$, $\mathbf{B_1}$ and $\mathbf{B_0}$ of bottom configurations



Define: sets **I**, **I_1** and **I_0** of initial configurations
sets **B**, **B_1** and **B_0** of bottom configurations

We study the shape of these infinite sets

Basic notions: Presburger arithmetic

Presburger arithmetic

- Atomic formulas: $a_1x_1 + \cdots + a_nx_n < b$
- Formulas: Close under boolean operations and quantification
- Formula $F(x_1, \ldots, x_m)$ with free variables $x_1, \ldots, x_m$ interpreted over $\mathbb{N}^m$

Init, Init_1 and Init_0 are Presburger definable

- **Init** is the set of configurations having
  - arbitrarily many agents in initial states, and
  - zero agents in non-initial states

$\Rightarrow$ **Init** is Presburger definable

Init, Init_1 and Init_0 are Presburger definable

- **Init** is the set of configurations having
  - arbitrarily many agents in initial states, and
  - zero agents in non-initial states

  $\Rightarrow$ **Init** is Presburger definable

- **Init_1** is the set of initial configurations satisfying the given predicate φ, and φ is Presburger definable
  $\Rightarrow$ **Init_1** is Presburger definable

Init, Init_1 and Init_0 are Presburger definable

- **Init** is the set of configurations having
    - arbitrarily many agents in initial states, and
    - zero agents in non-initial states

  $\Rightarrow$ **Init** is Presburger definable

- **Init_1** is the set of initial configurations satisfying the given predicate φ, and φ is Presburger definable

  $\Rightarrow$ **Init_1** is Presburger definable

- **Init_0** is the set of initial configurations that do not satisfy φ, and φ is Presburger definable

  $\Rightarrow$ **Init_0** is Presburger definable

Bottom, Bottom_1 and Bottom_0 are Presburger definable

The mutual reachability relation $\overset{*}{\longleftrightarrow}$ on the configurations of a given protocol is defined by:

$$C \overset{*}{\longleftrightarrow} C' \;\text{ iff }\; C \overset{*}{\longrightarrow} C' \text{ and } C' \overset{*}{\longrightarrow} C$$

Bottom, Bottom_1 and Bottom_0 are Presburger definable

The mutual reachability relation $\overset{*}{\longleftrightarrow}$ on the configurations of a given protocol is defined by:

$$C \overset{*}{\longleftrightarrow} C' \;\text{ iff }\; C \overset{*}{\longrightarrow} C' \text{ and } C' \overset{*}{\longrightarrow} C$$

$\overset{*}{\longleftrightarrow}$ is a congruence:

- $\overset{*}{\longleftrightarrow}$ is an equivalence relation
- $C_1 \overset{*}{\longleftrightarrow} C_1' \;\wedge\; C_2 \overset{*}{\longleftrightarrow} C_2' \;\Rightarrow\; C_1 + C_2 \overset{*}{\longleftrightarrow} C_1' + C_2'$

Bottom, Bottom_1 and Bottom_0 are Presburger definable

The mutual reachability relation $\overset{*}{\longleftrightarrow}$ on the configurations of a given protocol is defined by:

$$C \overset{*}{\longleftrightarrow} C' \;\text{ iff }\; C \overset{*}{\longrightarrow} C' \text{ and } C' \overset{*}{\longrightarrow} C$$

$\overset{*}{\longleftrightarrow}$ is a congruence:

- $\overset{*}{\longleftrightarrow}$ is an equivalence relation
- $C_1 \overset{*}{\longleftrightarrow} C_1' \;\wedge\; C_2 \overset{*}{\longleftrightarrow} C_2' \;\Rightarrow\; C_1 + C_2 \overset{*}{\longleftrightarrow} C_1' + C_2'$

Eilenberg and Schützenberger '69, Hirshfeld '94

Congruences over $\mathbb{N}^n$ are Presburger definable subsets of $\mathbb{N}^{2n}$

Bottom, Bottom_1 and Bottom_0 are Presburger definable

The mutual reachability relation $\overset{*}{\longleftrightarrow}$ on the configurations of a given protocol is defined by:

$$C \overset{*}{\longleftrightarrow} C' \;\text{ iff }\; C \overset{*}{\longrightarrow} C' \text{ and } C' \overset{*}{\longrightarrow} C$$

$\overset{*}{\longleftrightarrow}$ is a congruence:

- $\overset{*}{\longleftrightarrow}$ is an equivalence relation
- $C_1 \overset{*}{\longleftrightarrow} C_1' \;\wedge\; C_2 \overset{*}{\longleftrightarrow} C_2' \;\Rightarrow\; C_1 + C_2 \overset{*}{\longleftrightarrow} C_1' + C_2'$

Eilenberg and Schützenberger '69, Hirshfeld '94

Congruences over $\mathbb{N}^n$ are Presburger definable subsets of $\mathbb{N}^{2n}$

Corollary

$\overset{*}{\longleftrightarrow}$ is definable in Presburger arithmetic

Bottom, Bottom_1 and Bottom_0 are Presburger definable

A configuration C is a bottom configuration iff for every configuration C', C'':

$$\left(C \overset{*}{\longleftrightarrow} C' \;\wedge\; C' \longrightarrow C'' \right) \;\Rightarrow\; C \overset{*}{\longleftrightarrow} C''$$

Bottom, Bottom_1 and Bottom_0 are Presburger definable

A configuration C is a bottom configuration iff for every configuration C', C'':

$$\left(C \xleftrightarrow{*} C' \ \wedge \ C' \longrightarrow C'' \right) \ \Rightarrow \ C \xleftrightarrow{*} C''$$

So the predicate $Bottom(C)$ is captured by the formula:

$$Bottom(C) := \ \forall C', C'' \colon \left(C \xleftrightarrow{*} C' \ \wedge \ C' \longrightarrow C'' \right) \ \Rightarrow \ C \xleftrightarrow{*} C''$$

Bottom, Bottom_1 and Bottom_0 are Presburger definable

A configuration C is a bottom configuration iff for every configuration C', C'':

$$\left(C \xleftrightarrow{*} C' \ \wedge \ C' \longrightarrow C'' \right) \ \Rightarrow \ C \xleftrightarrow{*} C''$$

So the predicate $Bottom(C)$ is captured by the formula:

$$Bottom(C) := \ \forall C', C'' \colon \left(C \xleftrightarrow{*} C' \ \wedge \ C' \longrightarrow C'' \right) \ \Rightarrow \ C \xleftrightarrow{*} C''$$

Since both $\xleftrightarrow{*}$ and $\longrightarrow$ (one step!) are Presburger definable:

Proposition

Bottom, $\mathbf{Bottom_1}$ and $\mathbf{Bottom_0}$ are Presburger definable

Bottom, Bottom_1 and Bottom_0 are effectively Presburger definable

Leroux '11, Acta.Inf. '17

The mutual reachability relation of a population protocol is effectively Presburger definable

Proof:

1) Prove it first for globally cyclic protocols in which mutual reachability and reachability coincide

2) Show: for every protocol there is a globally cyclic protocol with the same mutual reachability relation

Bottom, Bottom_1 and Bottom_0 are effectively Presburger definable

Leroux '11, Acta.Inf. '17

The mutual reachability relation of a population protocol is effectively Presburger definable

Proof:

1) Prove it first for globally cyclic protocols in which mutual reachability and reachability coincide

2) Show: for every protocol there is a globally cyclic protocol with the same mutual reachability relation

Corollary

Bottom, $\mathbf{Bottom_1}$ and $\mathbf{Bottom_0}$ are effectively Presburger definable

From correctness to Petri net reachability

Recall that a protocol correctly computes a predicate iff:

$\textbf{Bottom} \setminus \textbf{Bottom}_0$ is not reachable from $\textbf{Init}_1$, and
$\textbf{Bottom} \setminus \textbf{Bottom}_1$ is not reachable from $\textbf{Init}_0$.

From correctness to Petri net reachability

Recall that a protocol correctly computes a predicate iff:

> $\textbf{Bottom} \setminus \textbf{Bottom}_\mathbf{0}$ is not reachable from $\textbf{Init}_\mathbf{1}$, and
> $\textbf{Bottom} \setminus \textbf{Bottom}_\mathbf{1}$ is not reachable from $\textbf{Init}_\mathbf{0}$.

We have reduced the correctness problem to:

Given: A population protocol $\mathcal{P}$
(Eff.) Presburger sets $\mathcal{C}, \mathcal{C}'$ of configurations of $\mathcal{P}$

Decide: Is some configuration of $\mathcal{C}'$ reachable from some configuration of $\mathcal{C}$?

From correctness to Petri net reachability

Recall that a protocol correctly computes a predicate iff:

> **Bottom**$\setminus$**Bottom$_0$** is not reachable from **Init$_1$**, and
> **Bottom**$\setminus$**Bottom$_1$** is not reachable from **Init$_0$**.

We have reduced the correctness problem to:

Given: A population protocol $\mathcal{P}$
(Eff.) Presburger sets $\mathcal{C}, \mathcal{C}'$ of configurations of $\mathcal{P}$

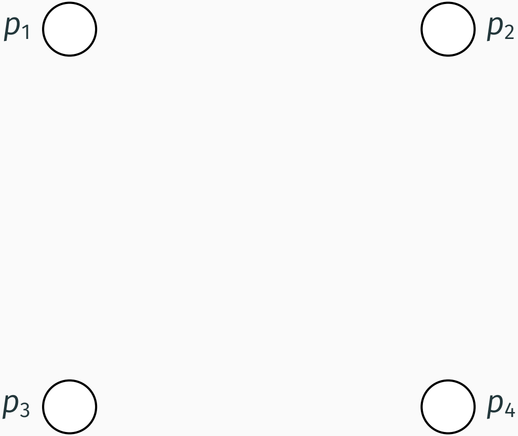
Decide: Is some configuration of $\mathcal{C}'$ reachable from some configuration of $\mathcal{C}$?

Now we reduce this to the reachability problem for Petri nets:

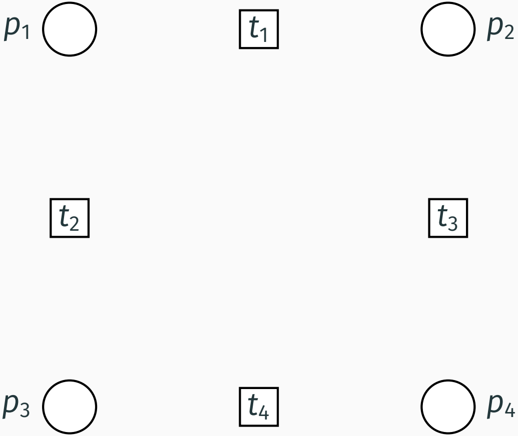
Given: Two markings M, M' of a Petri net

Decide: Is M' reachable from M ?

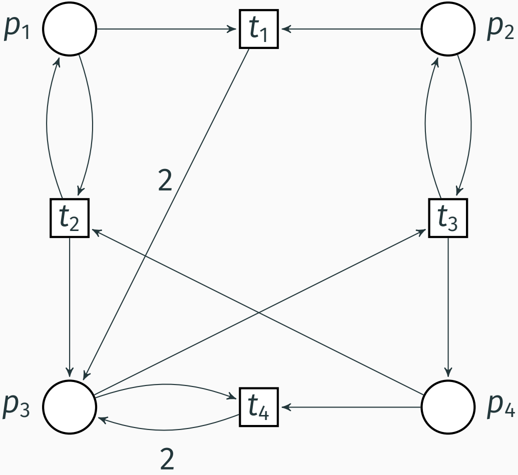
Petri nets



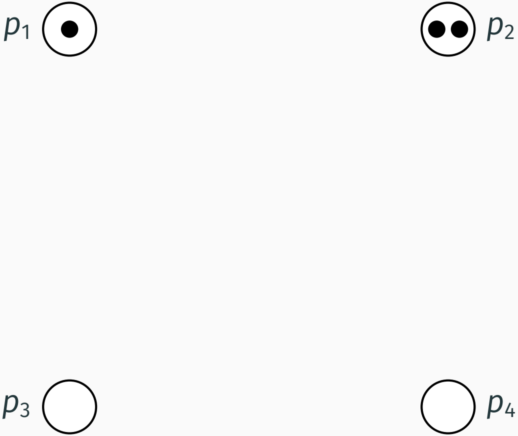
Petri nets



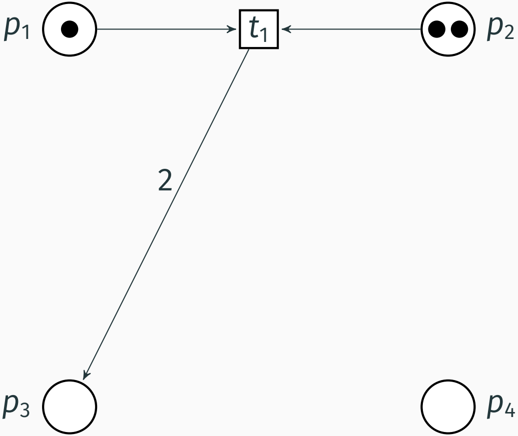
Petri nets



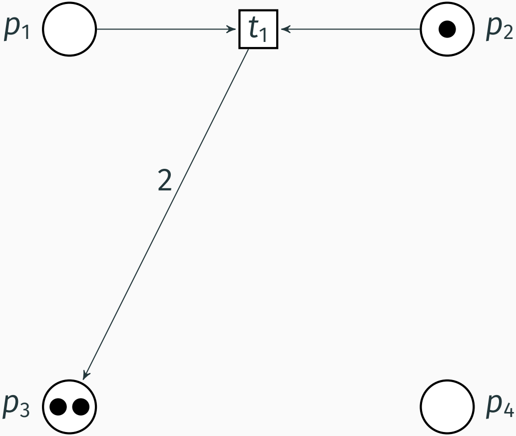
Markings



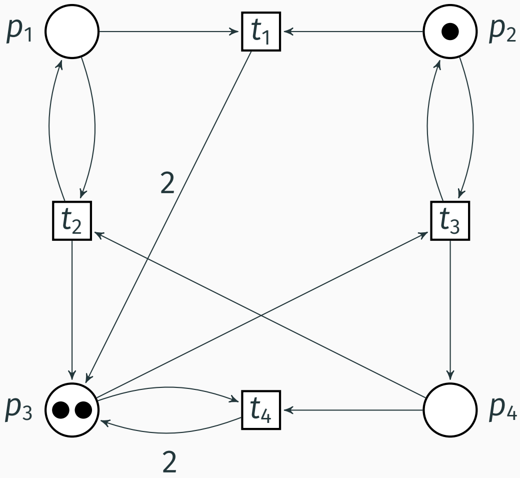
Firing rule



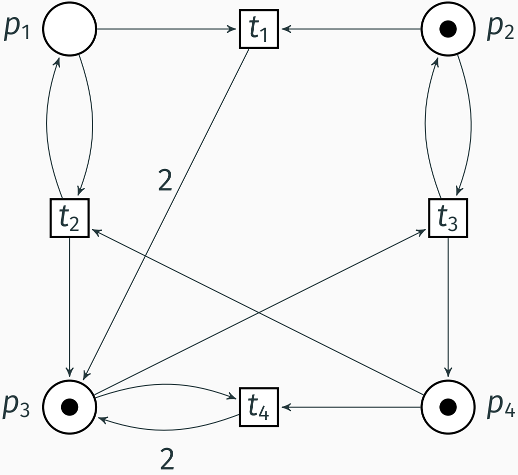
Firing rule



Petri nets



Petri nets



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

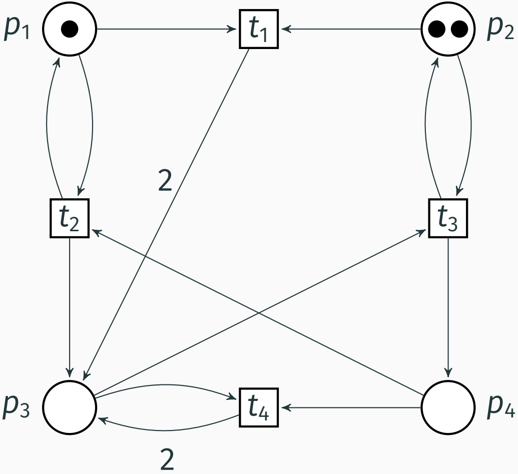
From PPs to Petri nets

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

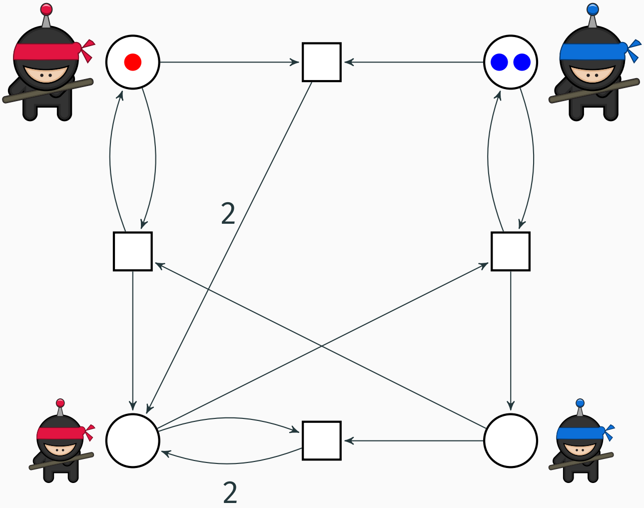
From PPs to Petri nets

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
PP	Net + infinite family of initial markings

Petri net of the slow majority protocol



Petri net of the majority protocol



From PPs to Petri nets

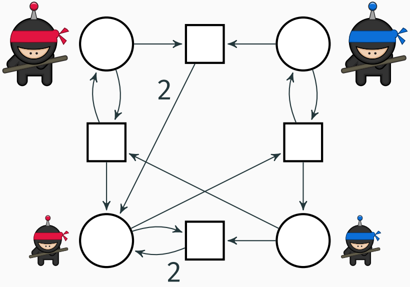
Every population protocol yields a net (without marking)

Not every net corresponds to a protocol!

- Protocol transitions neither create nor destroy tokens
- In particular, Petri nets derived from protocols are bounded for every initial marking

Petri nets “more general” than population protocols in this sense

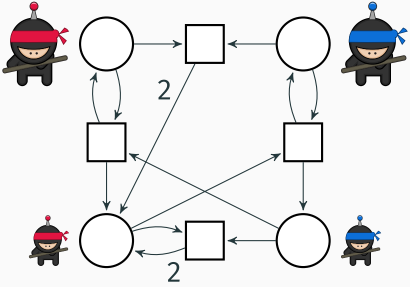
Additional power of Petri nets

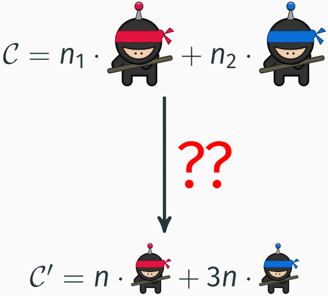


$\mathcal{C} = n_1 \cdot$  $+ n_2 \cdot$ 

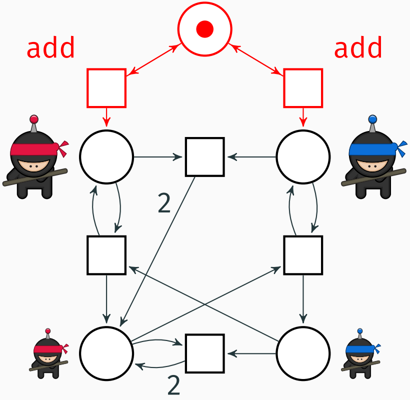
$\mathcal{C}' = n \cdot$  $+ 3n \cdot$ 

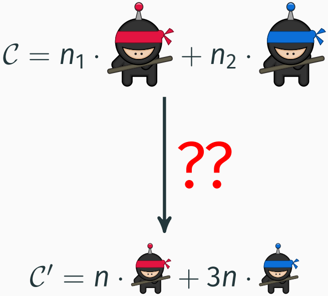
Additional power of Petri nets



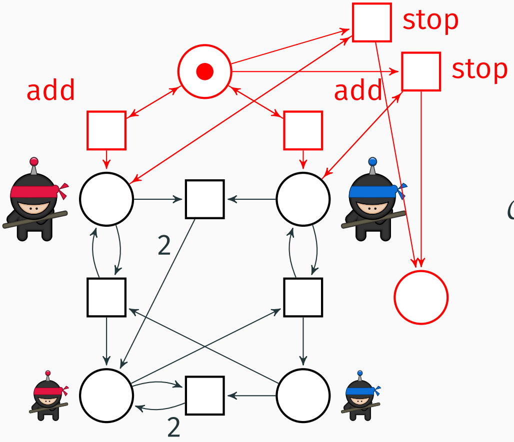


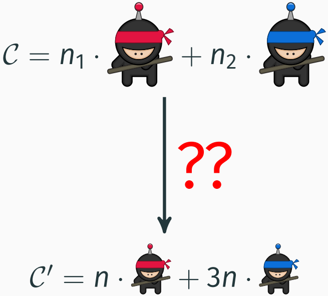
Additional power of Petri nets



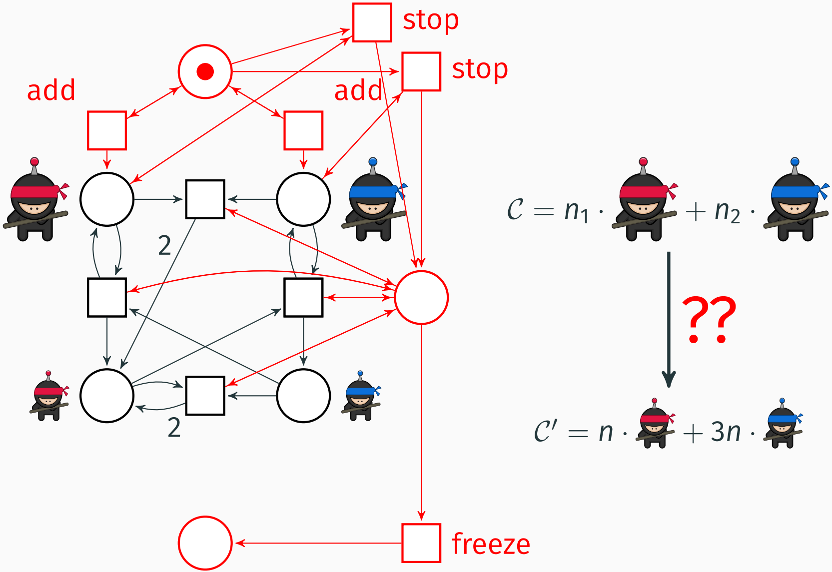


Additional power of Petri nets

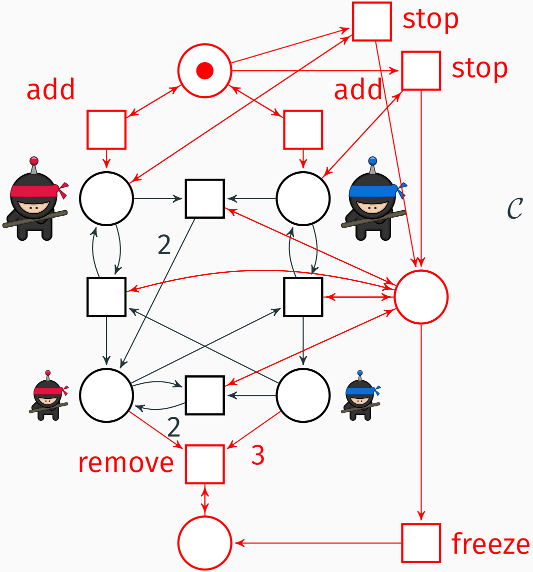


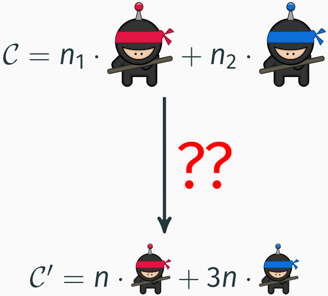


Additional power of Petri nets



Additional power of Petri nets





Checking correctness—Feasibility

Our approach:

- Most protocols are naturally analyzed in “stages”: “milestones” until the protocol reaches consensus 0 or 1, depending on the input

Checking correctness—Feasibility

Our approach:

- Most protocols are naturally analyzed in “stages”: “milestones” until the protocol reaches consensus 0 or 1, depending on the input
- Sound and complete methodology reflecting this idea:

Stage Graphs

Checking correctness—Feasibility

Our approach:

- Most protocols are naturally analyzed in “stages”: “milestones” until the protocol reaches consensus 0 or 1, depending on the input
- Sound and complete methodology reflecting this idea:

Stage Graphs

- Stage graphs for $b = 0$ and $b = 1$, describing “milestones” from the initial configurations for which the output should be b

Checking correctness—Feasibility

Our approach:

- Most protocols are naturally analyzed in "stages": "milestones" until the protocol reaches consensus 0 or 1, depending on the input
- Sound and complete methodology reflecting this idea:

Stage Graphs

- Stage graphs for $b = 0$ and $b = 1$, describing "milestones" from the initial configurations for which the output should be b
- SMT-based semi-algorithm for the automatic construction of stage graphs

Preliminaries: Transition-based consensus

- Split set T of transitions into $T_0, T_1, T_\perp$.
- Run reaches stable consensus b if from some moment on it only executes transitions of T_b.
- Equivalent to state-based consensus

Preliminaries: Transition-based consensus

- Split set T of transitions into $T_0, T_1, T_\perp$.
- Run reaches stable consensus b if from some moment on it only executes transitions of T_b.
- Equivalent to state-based consensus

Milestones: "killing" transitions

- A transition is dead at a configuration C if no configuration reachable from C enables it
- Intuition: protocols make progress towards consensus by "killing" transitions, until all "survivors" in T_0 or all in T_1.

Preliminaries: Progress certificates

- Let $\mathcal{C}, \mathcal{C}'$ be sets of configurations (think $\mathcal{C}'$ has more dead transitions than $\mathcal{C}$)
- $\mathcal{C} \rightsquigarrow \mathcal{C}'$: runs starting at $\mathcal{C}$ visit $\mathcal{C}'$ with probability 1
- Certificate for $\mathcal{C} \rightsquigarrow \mathcal{C}'$: mapping $f \colon \mathcal{C} \to \mathbb{N}$ such that for every $C \in \mathcal{C} \setminus \mathcal{C}'$ there exists $C \xrightarrow{*} C'$ such that $f(C) > f(C')$.

Preliminaries: Progress certificates

- Let $\mathcal{C}, \mathcal{C}'$ be sets of configurations (think $\mathcal{C}'$ has more dead transitions than $\mathcal{C}$)
- $\mathcal{C} \rightsquigarrow \mathcal{C}'$: runs starting at $\mathcal{C}$ visit $\mathcal{C}'$ with probability 1
- Certificate for $\mathcal{C} \rightsquigarrow \mathcal{C}'$: mapping $f \colon \mathcal{C} \to \mathbb{N}$ such that for every $C \in \mathcal{C} \setminus \mathcal{C}'$ there exists $C \xrightarrow{*} C'$ such that $f(C) > f(C')$.
- One-step certificate for $\mathcal{C} \rightsquigarrow \mathcal{C}'$: $C \longrightarrow C'$ instead of $C \xrightarrow{*} C'$.

Easy but important

For $\mathcal{C}$ inductive (closed under reachability):

$$\mathcal{C} \rightsquigarrow \mathcal{C}' \text{ iff there is a certificate for it}$$

Stage graphs

A stage graph for a given protocol, a given predicate, and a given $b \in \{0, 1\}$ is a finite DAG satisfying:

Stage graphs

A stage graph for a given protocol, a given predicate, and a given $b \in \{0, 1\}$ is a finite DAG satisfying:

1. The nodes of the DAG, called stages, are inductive sets of configurations (closed under reachability)

Stage graphs

A stage graph for a given protocol, a given predicate, and a given $b \in \{0, 1\}$ is a finite DAG satisfying:

1. The nodes of the DAG, called stages, are inductive sets of configurations (closed under reachability)
2. Every b-initial configuration of the protocol belongs to some initial stage

Stage graphs

A stage graph for a given protocol, a given predicate, and a given $b \in \{0, 1\}$ is a finite DAG satisfying:

1. The nodes of the DAG, called stages, are inductive sets of configurations (closed under reachability)
2. Every b-initial configuration of the protocol belongs to some initial stage
3. For every non-terminal stage $\mathcal{C}$ with children $\mathcal{C}_1, \ldots, \mathcal{C}_n$ there is a certificate for $\mathcal{C} \rightsquigarrow \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_n$

Stage graphs

A stage graph for a given protocol, a given predicate, and a given $b \in \{0, 1\}$ is a finite DAG satisfying:

1. The nodes of the DAG, called stages, are inductive sets of configurations (closed under reachability)
2. Every b-initial configuration of the protocol belongs to some initial stage
3. For every non-terminal stage $\mathcal{C}$ with children $\mathcal{C}_1, \ldots, \mathcal{C}_n$ there is a certificate for $\mathcal{C} \rightsquigarrow \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_n$
4. For every terminal stage $\mathcal{C}$: every $C \in \mathcal{C}$ enables only transitions of T_b

An example

Majority protocol $(\mathbf{R} \overset{?}{>} \mathbf{B})$

$t_1: \; \mathbf{B}, \mathbf{R} \; \mapsto \; \mathbf{b}, \mathbf{b}$ $\qquad$ $t_3: \; \mathbf{R}, \mathbf{b} \; \mapsto \; \mathbf{R}, \mathbf{r}$

$t_2: \; \mathbf{B}, \mathbf{r} \; \mapsto \; \mathbf{B}, \mathbf{b}$ $\qquad$ $t_4: \; \mathbf{b}, \mathbf{r} \; \mapsto \; \mathbf{b}, \mathbf{b}$

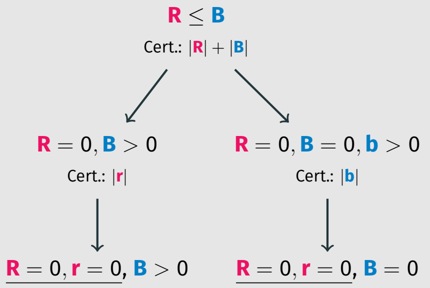
An example

Majority protocol $(\mathbf{R} \overset{?}{>} \mathbf{B})$

$$t_1 : \mathbf{B}, \mathbf{R} \mapsto \mathbf{b}, \mathbf{b} \qquad t_3 : \mathbf{R}, \mathbf{b} \mapsto \mathbf{R}, \mathbf{r}$$

$$t_2 : \mathbf{B}, \mathbf{r} \mapsto \mathbf{B}, \mathbf{b} \qquad t_4 : \mathbf{b}, \mathbf{r} \mapsto \mathbf{b}, \mathbf{b}$$

Stage graph for $b = 0$



An example

Majority protocol $(\mathbf{R} \overset{?}{>} \mathbf{B})$

$$t_1: \mathbf{B}, \mathbf{R} \mapsto \mathbf{b}, \mathbf{b} \qquad t_3: \mathbf{R}, \mathbf{b} \mapsto \mathbf{R}, \mathbf{r}$$

$$t_2: \mathbf{B}, \mathbf{r} \mapsto \mathbf{B}, \mathbf{b} \qquad t_4: \mathbf{b}, \mathbf{r} \mapsto \mathbf{b}, \mathbf{b}$$

Stage graph for $b = 0$

$\mathbf{R} \leq \mathbf{B}$

Cert.: $|\mathbf{R}| + |\mathbf{B}|$

↙ ↘

$\mathbf{R} = 0, \mathbf{B} > 0$

Cert.: $|\mathbf{r}|$

↓

$\underline{\mathbf{R} = 0, \mathbf{r} = 0}, \mathbf{B} > 0$

$\mathbf{R} = 0, \mathbf{B} = 0, \mathbf{b} > 0$

Cert.: $|\mathbf{b}|$

↓

$\underline{\mathbf{R} = 0, \mathbf{r} = 0}, \mathbf{B} = 0$

Stage graph for $b = 1$

$\mathbf{R} > \mathbf{B}$

Cert.: $|\mathbf{R}| + |\mathbf{B}|$

↓

$\mathbf{R} > 0, \mathbf{B} = 0$

Cert.: $|\mathbf{b}|$

↓

$\mathbf{R} > 0, \underline{\mathbf{B} = 0, \mathbf{b} = 0}$

Stage graphs: soundness, completeness, decidabilty

Soundness

If a protocol has stage graphs for a predicate φ and both 0 and 1, then the protocol computes φ.

Proof.

Easy.
Show that executions “go down” the stage graph w.p.1 till they get “trapped” in a bottom stage. $\square$

Stage graphs: soundness, completeness, decidabilty

A Presburger stage graph is a stage graph whose nodes are Presburger sets and whose certificates are one-step Presburger certificates ($f(C) = a$ iff $\psi(C, a) \equiv$ **true**)

Completeness — **Acta Inf. 2017**

If a protocol computes φ, then it has Presburger stage graphs for φ and both 0 and 1.

Proof.

Very hard.

Initial stage: Inductive Presburger "envelope" of the b-initial configurations.

Final stage: set of all b-stable consensuses. □

Stage graphs: soundness, completeness, decidabilty

Decidability **Acta Inf. 2017**

It is decidable if a given DAG of Presburger sets and Presburger functions is a Presburger stage graph for a given b.

Proof.

Follows from properties of Presburger sets:

Inductivity of Presburger sets is decidable (in NP for existential fragment)

Whether a Presburger function is a one-step Presburger certificate is decidable (in NP for existential fragment) □

Stage graphs: soundness, completeness, decidabilty

Alternative algorithm for decidability of correctness:

- Two semi-decision algorithms
- For non-correctness: enumerate all initial configurations and check convergence to the right value
- For correctness: enumerate all DAGs of Presburger sets and functions, and check if they are Presburger stage graphs for 0 or for 1

Computing the children of a stage $\mathcal{S}$

Input: $\mathcal{S}$

$U := ASDead(\mathcal{S})$

if $U \neq \emptyset$ **then**

  output $DeadAt(U, \mathcal{S})$

else

  output $Split(\mathcal{S})$

Computing the children of a stage $\mathcal{S}$

Input: $\mathcal{S}$

$U := \mathit{ASDead}(\mathcal{S})$

Returns a set of transitions that will die almost surely from **any** configuration of $\mathcal{S}$

if $U \neq \emptyset$ **then**

output $\mathit{DeadAt}(U, \mathcal{S})$

else

output $\mathit{Split}(\mathcal{S})$

Computing the children of a stage $\mathcal{S}$

Input: $\mathcal{S}$

$U := \mathit{ASDead}(\mathcal{S})$

if $U \neq \emptyset$ **then**

  output $\mathit{DeadAt}(U, \mathcal{S})$

else

  output $\mathit{Split}(\mathcal{S})$

Overapproximates the configurations reachable from $\mathcal{S}$ at which U is dead

Computing the children of a stage $\mathcal{S}$

Input: $\mathcal{S}$

$U := \mathit{ASDead}(\mathcal{S})$

if $U \neq \emptyset$ **then**

  output $\mathit{DeadAt}(U, \mathcal{S})$

else

  output $\mathit{Split}(\mathcal{S})$

Attempts to split $\mathcal{S}$ into stages with more dead transitions

Computing the children of a stage $\mathcal{S}$

Input: $\mathcal{S}$

$U := ASDead(\mathcal{S})$

if $U \neq \emptyset$ **then**

  output $DeadAt(U, \mathcal{S})$

else

  output $Split(\mathcal{S})$

Implementing AsDead($\mathcal{S}$): Kirchoff's equations

Transition $t \implies$ offset $\Delta(t)$

Examples: $t\colon\ q_1, q_2 \mapsto q_2, q_3 \quad\implies\quad \Delta(t) = (-1, 0, 1)$

$t\colon\ q_1, q_2 \mapsto q_3, q_3 \quad\implies\quad \Delta(t) = (-1, -1, 2)$

We have: if $C \xrightarrow{t} C'$ then $C' = C + \Delta(t)$

Implementing AsDead($\mathcal{S}$): Kirchoff's equations

Transition $t \implies$ offset $\Delta(t)$

Examples: $t\colon\ q_1, q_2 \mapsto q_2, q_3 \implies \Delta(t) = (-1, 0, 1)$

$t\colon\ q_1, q_2 \mapsto q_3, q_3 \implies \Delta(t) = (-1, -1, 2)$

We have: if $C \xrightarrow{t} C'$ then $C' = C + \Delta(t)$

Transition sequence $w = t_1 \dots t_n$

$$\implies \text{offset } \Delta(w) = \sum_{i=1}^{n} \Delta(t_i) = \sum_{t \in T} \#(t, w) \cdot \Delta(t)$$

Example: if $C \xrightarrow{t_1 t_2 t_1} C'$ then $C' = C + 2 \cdot \Delta(t_1) + \Delta(t_2)$

We have: if $C \xrightarrow{w} C'$ then $C' = C + \Delta(w)$

Implementing AsDead($\mathcal{S}$): Kirchoff's equations

If transition u can occur infinitely often

$\Longrightarrow$ there is $C \xrightarrow{w} C$ with $\#(u, w) \geq 1$

$\Longrightarrow$ there is w with $\Delta(w) = 0$ and $\#(u, w) \geq 1$

Implementing AsDead($\mathcal{S}$): Kirchoff's equations

If transition u can occur infinitely often

$\implies$ there is $C \xrightarrow{w} C$ with $\#(u, w) \geq 1$

$\implies$ there is w with $\Delta(w) = 0$ and $\#(u, w) \geq 1$

$\implies$ there is w with $\sum_{t \in T} \#(t, w) \cdot \Delta(t) = 0$ and $\#(u, w) \geq 1$

Implementing AsDead($\mathcal{S}$): Kirchoff's equations

If transition u can occur infinitely often

$\Longrightarrow$ there is $C \xrightarrow{w} C$ with $\#(u, w) \geq 1$

$\Longrightarrow$ there is w with $\Delta(w) = 0$ and $\#(u, w) \geq 1$

$\Longrightarrow$ there is w with $\sum_{t \in T} \#(t, w) \cdot \Delta(t) = 0$ and $\#(u, w) \geq 1$

Implementing AsDead($\mathcal{S}$): Kirchoff's equations

If transition u can occur infinitely often

$\implies$ there is $C \xrightarrow{w} C$ with $\#(u, w) \geq 1$

$\implies$ there is w with $\Delta(w) = 0$ and $\#(u, w) \geq 1$

$\implies$ there is w with $\sum_{t \in T} \#(t, w) \cdot \Delta(t) = 0$ and $\#(u, w) \geq 1$

Implementing AsDead($\mathcal{S}$): Kirchoff's equations

If transition u can occur infinitely often

$\Longrightarrow$ there is $C \xrightarrow{w} C$ with $\#(u, w) \geq 1$

$\Longrightarrow$ there is w with $\Delta(w) = 0$ and $\#(u, w) \geq 1$

$\Longrightarrow$ there is w with $\sum_{t \in T} \#(t, w) \cdot \Delta(t) = 0$ and $\#(u, w) \geq 1$

$\Longrightarrow$ there are coefficients n_t for every $t \in T$ with

$$\sum_{t \in T} n_t \cdot \Delta(t) = 0 \quad \text{and} \quad n_u \geq 1$$

Implementing AsDead($\mathcal{S}$): Kirchoff's equations

If transition u can occur infinitely often

$\Longrightarrow$ there is $C \xrightarrow{w} C$ with $\#(u, w) \geq 1$

$\Longrightarrow$ there is w with $\Delta(w) = 0$ and $\#(u, w) \geq 1$

$\Longrightarrow$ there is w with $\sum_{t \in T} \#(t, w) \cdot \Delta(t) = 0$ and $\#(u, w) \geq 1$

$\Longrightarrow$ there are coefficients n_t for every $t \in T$ with

$$\underbrace{\sum_{t \in T} n_t \cdot \Delta(t) = 0 \quad \text{and} \quad n_u \geq 1}_{\text{Kirchoff's equations (unknowns: } \{n_t \mid t \in T\})}$$

Implementing AsDead($\mathcal{S}$): Kirchoff's equations

If transition u can occur infinitely often

$\Longrightarrow$ there is $C \xrightarrow{w} C$ with $\#(u, w) \geq 1$

$\Longrightarrow$ there is w with $\Delta(w) = 0$ and $\#(u, w) \geq 1$

$\Longrightarrow$ there is w with $\sum_{t \in T} \#(t, w) \cdot \Delta(t) = 0$ and $\#(u, w) \geq 1$

$\Longrightarrow$ there are coefficients n_t for every $t \in T$ with

$$\underbrace{\sum_{t \in T} n_t \cdot \Delta(t) = 0 \quad \text{and} \quad n_u \geq 1}_{\text{Kirchoff's equations (unknowns: } \{n_t \mid t \in T\})}$$

Kirchoff's equations unsatisfiable

$\Longrightarrow$ t cannot occur infinitely often from any configuration

Implementing AsDead($\mathcal{S}$): Layers

A layer of a protocol is a set L of transitions such that for every configuration C (reachable or not):

- all executions from C containing only transitions of L are finite
- if all transitions of L are disabled at C, then they cannot be re-enabled by any sequence $w \in (T \setminus L)^*$.

If L is a layer, then from any configuration all transitions of L eventually die

Implementing AsDead($\mathcal{S}$): Layers

A layer of a protocol is a set L of transitions such that for every configuration C (reachable or not):

- all executions from C containing only transitions of L are finite
- if all transitions of L are disabled at C, then they cannot be re-enabled by any sequence $w \in (T \setminus L)^*$.

If L is a layer, then from any configuration all transitions of L eventually die

There exists a set of integer linear constraints whose solutions correspond to the possible layers of the protocol $\longrightarrow$ finding a layer is in NP

Implementing DeadAt$(U, \mathcal{S})$

Recall: $\mathit{DeadAt}(U, \mathcal{S})$ overapproximates the configurations reachable from $\mathcal{S}$ at which U is dead

Computable as intersection of:

- overapproximation of the configurations reachable from $\mathcal{S}$
(overapproximation is necessary)
- overapproximation of the configurations at which U is dead
(overapproximation is optional)

Implementing DeadAt$(U, \mathcal{S})$

Set of configurations reachable from $\mathcal{S}$

Overapproximated by set of configurations satisfying automatically computed linear invariants of the form

$$\sum_{q \in Q} a_q \cdot C(q) \geq b$$

Implementing DeadAt$(U, \mathcal{S})$

Set of configurations at which U is dead

$Dead(U)$: configurations at which U is dead

$En(U)$: configurations enabling some transition of U.

$$Dead(U) = \overline{\mathrm{pre}^*(En(U))}$$

Implementing DeadAt$(U, \mathcal{S})$

Set of configurations at which U is dead

Dead(U): configurations at which U is dead

En(U): configurations enabling some transition of U.

$$\mathit{Dead}(U) = \overline{\mathrm{pre}^*(\mathit{En}(U))}$$

Observation: $\mathrm{pre}^*(\mathit{En}(U))$ is upward-closed

Dead(U) is downward-closed

Implementing DeadAt$(U, \mathcal{S})$

Set of configurations at which U is dead

$\mathit{Dead}(U)$: configurations at which U is dead

$\mathit{En}(U)$: configurations enabling some transition of U.

$$\mathit{Dead}(U) = \overline{\mathrm{pre}^*(\mathit{En}(U))}$$

Observation: $\mathrm{pre}^*(\mathit{En}(U))$ is upward-closed

$\mathit{Dead}(U)$ is downward-closed

$\Rightarrow$ both are Presburger

Implementing DeadAt$(U, \mathcal{S})$

Set of configurations at which U is dead

$Dead(U)$: configurations at which U is dead

$En(U)$: configurations enabling some transition of U.

$$Dead(U) = \overline{\mathrm{pre}^*(En(U))}$$

Proposition

$\mathrm{pre}^*(En(U))$ has finitely many minimal elements, and they can be computed using a symbolic backward reachability algorithm.

Implementing DeadAt$(U, \mathcal{S})$

Set of configurations at which U is dead

$\mathit{Dead}(U)$: configurations at which U is dead

$\mathit{En}(U)$: configurations enabling some transition of U.

$$\mathit{Dead}(U) = \overline{\mathrm{pre}^*(\mathit{En}(U))}$$

Proposition

$\mathrm{pre}^*(\mathit{En}(U))$ has finitely many minimal elements, and they can be computed using a symbolic backward reachability algorithm.

$\Rightarrow$ both are effectively Presburger

Some experimental results (a bit outdated ...)

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]	$x_1 \vee \ldots \vee x_n$	2	1	0.1
Threshold[4]	$\Sigma_i \alpha_i x_i < c$	76	2148	2375.9
Remainder[5]	$\Sigma_i \alpha_i x_i \bmod 70 = 1$	72	2555	3176.5
Sick ninjas[6]	$x \geq 50$	51	1275	181.6
Sick ninjas[7]	$x \geq 325$	326	649	3470.8
Poly-log sick ninjas	$x \geq 8 \cdot 10^4$	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

Sensei III's questions

What predicates can we compute?

How fast can we compute them?

How succinctly can we compute them?

How can I check correctness?

How can I check efficiency?

To conclude ...

Peregrine: a tool for population protocols Blondin, E., Jaax CAV'18

Peregrine: Haskell + Microsoft Z3 + JavaScript

peregrine.model.in.tum.de

- Design of protocols
- Manual and automatic simulation
- Statistics of properties such as termination time
- Automatic verification of correctness
- More to come!

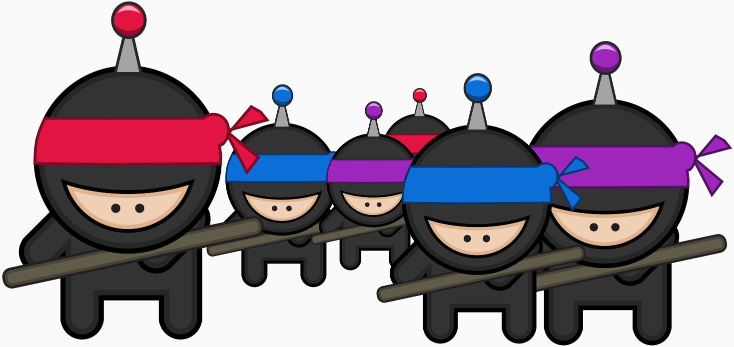
Population protocols are a great model to study fundamental questions of distributed computation:

- Power of anonymous computation
- Network-independent algorithms
- Role of leaders
- Emergent behaviour and its limits

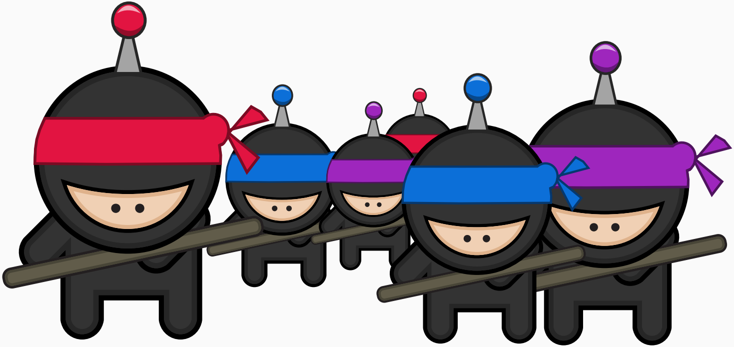
Conclusion

...and of formal verification:

- Verification of stochastic parameterized systems (parameterization, liveness under fairness)
- Automatic synthesis of parameterized systems



THANK YOU!



Go!

THANK YOU!