Peregrine – A Verification Tool for Population Protocols

Michael Blondin, Javier Esparza,
Stefan Jaax, Philipp Meyer

28.11. + 5.12.2018

TU München

Humble Beginnings

There is this neat computation model called 'Population Protocols'.

Maybe this could become your PhD topic, Stefan?

Simpol


file:///home/stev/simpol/index.html
Suchen
File: Default
Load
Reload
Draw
1
20
2
19
3
0
4
0
Rules
1 2 → 3 3
1 4 → 1 3
2 3 → 2 4
3 4 → 3 3
MAJORITY
1 1 1 1 2 2 2 2
1 1 1 1 2 2 2 2
1 1 1 1 2 2 2 2
1 1 1 1 2 2 2 2
1 1 1 1 2 2 2
Step >
Run >>

A Challenge

We proved the well-specification problem is decidable!

☺☺☺☺☺☺☺☺☺☺☺☺

A Challenge

We proved the well-specification problem is decidable!

☺☺☺☺☺☺☺☺☺☺☺☺

But the decision procedure is practically useless...

☹☹☹☹☹☹☹☹☹☹

A Challenge

...maybe there is a simple syntactic restriction that guarantees well-specification?

The majority of protocols I know are terminating in a peculiar manner...
...maybe we can exploit this?

A Challenge

Most protocols seem to progress in *phases.*

Consider Majority:

There is a 'cancellation' phase: A, B -> a, b

...and there is a 'domination' phase: A, b -> A, a
B, a -> B, b

...and a 'tie-breaker' phase:
a, b -> b, b

A Challenge

Maybe we can characterize phases syntactically and automatically derive Termination & Well-Specification?

A Challenge

Identify a natural class of protocols satisfying

- Termination:
  Every fair execution terminates.

- (Well-Specification | Termination):
  All terminal configurations reachable from any given initial configuration form the same consensus.

Class should be fully expressive + membership test should be feasible!

Three Shades of Termination

There are different shades of Termination:

- Termination irrespective of start configuration or fairness.
- Fair Termination irrespective of start configuration.
- Fair Termination starting in initial configuration.

Structural Termination/Strong Normalization

A protocol is structurally terminating if every computation step makes *progress*:

Structural Termination

There exists a well-founded partial order $(\preceq)$ such that for all configurations $C \neq C'$:

$C \to C' \implies C' \prec C$.

Structural Termination/Strong Normalization

A protocol is structurally terminating if every computation step makes *progress*:

Structural Termination

There exists a well-founded partial order $(\preceq)$ such that for all configurations $C \neq C'$:

$C \longrightarrow C' \Longrightarrow C' \prec C$.

Structural termination is a very strong assumption!
It implies that even unfair runs terminate, irrespective of where they started.

Bad News

Not even the simple Majority protocol satisfies Structural Termination:

The tie-breaker $a, b \longrightarrow b, b$ can reverse progress made by $A, b \longrightarrow A, a$.

Bad News

Not even the simple Majority protocol satisfies Structural Termination:

The tie-breaker $a, b \longrightarrow b, b$ can reverse progress made by $A, b \longrightarrow A, a$.

Maybe we can 'tame' Majority, so that it becomes structurally terminating? - Possible, and leads to faster Majority, but does not generalize well.

Structural Termination: Tractibility

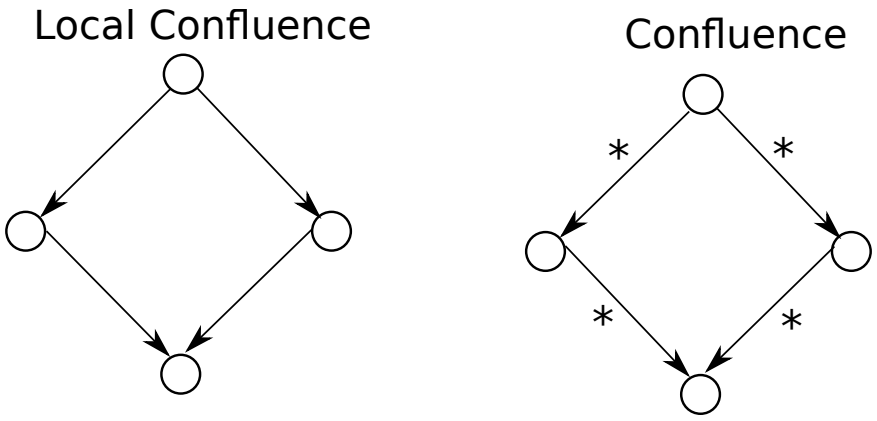
Structural Termination is decidable for Petri nets in PTIME.

Structural Termination: Diamond Lemma

Aren't population protocols
just
multi-set rewrite systems?

A structurally terminating multi-set rewrite system is confluent iff. it is locally confluent.

Confluence



Structural Termination +

Local Confluence +

(Terminal implies Consensus)

$\Longrightarrow$ Well-Specified

From Structural Termination to Layered Termination

Structural Termination is too demanding!

From Structural Termination to Layered Termination

Structural Termination is too demanding!

But maybe we can partition transitions into 'phases', where each phase is structurally terminating, and succeeding phases do not re-enable previous phases?

From Structural Termination to Layered Termination

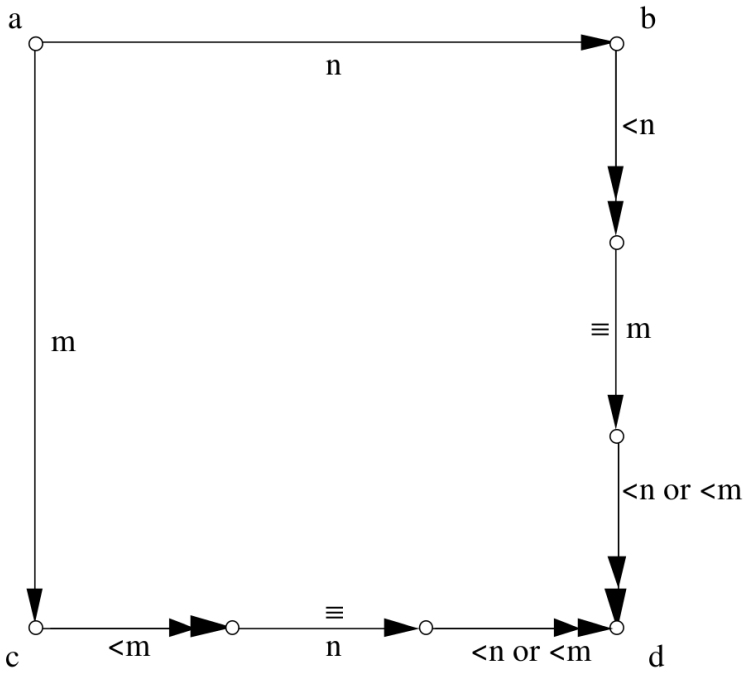
Structural Termination is too demanding!

But maybe we can partition transitions into 'phases', where each phase is structurally terminating, and succeeding phases do not re-enable previous phases?

We call the phases layers: Layered Termination.

Deciding Confluence becomes infeasible

If we abandon Structural Termination, then the diamond lemma is no longer applicable.



The general local criterion for confluence is quite involved!

A closer look at Well-Specification

(Well-Specification | Termination):

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

A closer look at Well-Specification

(Well-Specification | Termination):

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

Maybe we do not need exact reachability, but some sufficiently refined overapproximation will do the trick?

Overapproximating Reachability

Aren't population protocols *just* Petri nets?

Petrinizer to the rescue!

Overapproximating Reachability

[CAV'14] *Esparza, Ledesma-Garza, Majumdar, PJ Meyer*

An SMT-based Approach to Coverability Analysis

Petrinizer *over-approximates* reachability relation of Petri nets.
Approximation can be refined in refinement steps.

Peregrine

- [PODC'17] *Blondin, Esparza, Jaax, PJ Meyer* Towards efficient Verification of Population Protocols

- [CAV'18] *Blondin, Esparza, Jaax* Peregrine: A Tool for the Analysis of Population Protocols

Peregrine

Peregrine

peregrine.model.in.tum.de

Tool for

- Design
- Parameterized Verification
- Simulation

of Population Protocols.

Demo

DEMO

Towards a verifiable class

Search for subclass of class WS of well-specified protocols that:

- Has a membership test of reasonable complexity.
- Can compute all Presburger predicates.
- Naturally contains common protocols.
- Is suitable for verification of correctness.

Towards a verifiable class: WS^2

- Many protocols are *silent*: fair executions always reach a terminal configuration where no state-changing transitions are enabled.

Towards a verifiable class: WS^2

- Many protocols are *silent*: fair executions always reach a terminal configuration where no state-changing transitions are enabled.
- Easier test if consensus is reached.
- Protocols have same expressiveness.

Towards a verifiable class: WS^2

- Many protocols are *silent*: fair executions always reach a terminal configuration where no state-changing transitions are enabled.
- Easier test if consensus is reached.
- Protocols have same expressiveness.
- Petri net reachability still reducable to membership problem for class WS^2 of well-specificed silent protocols.

Towards a verifiable class: WS^3

Original Property

Silent

For every initial C_0:

$$C_0 \xrightarrow{*} C' \implies$$

$$\exists \text{ terminal } C_\perp : C' \xrightarrow{*} C_\perp$$

Well-specified | Silent

All terminal configs reachable from init. C_0 form same consensus.

Approximation

Layered Termination

For every config C:

$\exists$ terminal $C_\perp : C \xrightarrow{*} C_\perp$

due to universal termination strategy (of a certain form).

Strong Consensus

All terminal configs *potentially reachable* from init. C_0 form same consensus.

Layered Termination

A protocol satisfies layered termination if there is a partition

$$T = T_1 \cup T_2 \cup \ldots \cup T_{n-1} \cup T_n$$

of T in n layers such that for every configuration C:

- each layer is structurally terminating, i.e. all executions from C using transitions from a single layer are finite.
- if all transitions of $T_1, \ldots, T_i$ are disabled at C, then they cannot be re-enabled by any transitions of $T_{i+1}, \ldots, T_n$.

Layered Termination

A protocol satisfies layered termination if there is a partition

$$T = T_1 \cup T_2 \cup \ldots \cup T_{n-1} \cup T_n$$

of T in n layers such that for every configuration C:

- each layer is structurally terminating, i.e. all executions from C using transitions from a single layer are finite.
- if all transitions of $T_1, \ldots, T_i$ are disabled at C, then they cannot be re-enabled by any transitions of $T_{i+1}, \ldots, T_n$.

If there is such a partition, then there is always a layered execution

$$C \xrightarrow{T_1^*} C_1 \xrightarrow{T_2^*} \ldots \xrightarrow{T_n^*} C_\perp$$

such that $C_\perp$ is terminal. Therefore the protocol is silent.

Deciding Structural Termination

A layer T_i is structurally terminating if there exists a partial order $\preceq$ such that for every configuration C and every $C \xrightarrow{t} C'$ with $t \in T_i$:

$$C' \prec C$$

Deciding Structural Termination

A layer T_i is structurally terminating if there exists a partial order $\preceq$ such that for every configuration C and every $C \xrightarrow{t} C'$ with $t \in T_i$:

$$C' \prec C$$

We have that $\preceq$ can be given by a linear ranking function r:

$C' \prec C \Leftrightarrow r(C') < r(C)$ where $r(C) \stackrel{\text{def}}{=} \sum_{q \in Q} r_q C(q)$ for some $\{r_q\}_{q \in Q} \subseteq \mathbb{Q}_{\geq 0}$

Deciding Structural Termination

A layer T_i is structurally terminating if there exists a partial order $\preceq$ such that for every configuration C and every $C \xrightarrow{t} C'$ with $t \in T_i$:

$$C' \prec C$$

We have that $\preceq$ can be given by a linear ranking function r:

$C' \prec C \Leftrightarrow r(C') < r(C)$ where $r(C) \stackrel{\text{def}}{=} \sum_{q \in Q} r_q C(q)$ for some $\{r_q\}_{q \in Q} \subseteq \mathbb{Q}_{\geq 0}$

Then T_i is structurally terminating if there exist $\{r_q\}_{q \in Q}$ such that for every transition $t \in T_i$:

$$r(\mathrm{post}(t)) < r(\mathrm{pre}(t))$$

Deciding Layered Termination

To decide layered termination:

- Guess partition of layers.
- Check that each layer is structurally terminating (compute coefficients of r in polynomial time)
- Test whether layers cannot re-enable previous layers (syntactic criterion given by the transition function)

$\Rightarrow$ **NP** decision procedure

Towards a verifiable class: WS^3

Original Property

Silent

For every init. C_0:

$$C_0 \xrightarrow{*} C' \implies$$

$$\exists \text{ terminal } C_\perp : C' \xrightarrow{*} C_\perp$$

Well-specified | Silent

All terminal configs reachable from init. C_0 form same consensus.

Approximation

Layered Termination

For every config C:

$\exists$ terminal $C_\perp$: $C \xrightarrow{*} C_\perp$

NP

due to universal termination strategy (of a certain form).

Strong Consensus

All terminal configs *potentially reachable* from init. C_0 form same consensus.

Strong Consensus

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

$$\text{Reachability} \implies \text{Potential Reachability}$$

$$C \xrightarrow{*} C' \implies C \overset{*}{\dashrightarrow} C'$$

$$\text{Potential Reachability} \not\implies \text{Reachability}$$

Strong Consensus

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

$$\text{Reachability} \implies \text{Potential Reachability}$$

$$C \xrightarrow{*} C' \implies C \overset{*}{\dashrightarrow} C'$$

$$\text{Potential Reachability} \not\implies \text{Reachability}$$

Potential reachability should be easiliy checkable, e.g. by set of linear constraints, but still give a good approximation.

Potential Reachability: First Attempt

Let C, C' be configurations s.t. $C \xrightarrow{t_1 \dots t_m} C'$ for some $t_1, \dots, t_m$.

Let $x(t) =$ *#occurences* of t in $t_1 \dots t_m$.

C, C', x satisfy the *state equation*:

$$\boxed{C' = C + \Sigma_{t \in T} \left(\mathrm{post}(t) - \mathrm{pre}(t)\right) \cdot x(t)}$$

where

$$t : \underbrace{q_1, q_2}_{\mathrm{pre}(t)} \mapsto \underbrace{r_1, r_2}_{\mathrm{post}(t)} .$$

Potential Reachability: First Attempt

Let C, C' be configurations s.t. $C \xrightarrow{t_1 \dots t_m} C'$ for some $t_1, \dots, t_m$.

Let $x(t) = \#\text{occurences}$ of t in $t_1 \dots t_m$.

C, C', x satisfy the *state equation*:

$$\boxed{C' = C + \Sigma_{t \in T} \left(\text{post}(t) - \text{pre}(t)\right) \cdot x(t)}$$

where

$$t : \underbrace{q_1, q_2}_{\text{pre}(t)} \mapsto \underbrace{r_1, r_2}_{\text{post}(t)} .$$

The state equation disregards the order of execution. It only considers the *net effect* of transitions!

Potential Reachability: Example

$$C = \{A, B\} \xrightarrow{t_1} \{a, b\} \xrightarrow{t_4} \{b, b\} = C'$$

Majority

$t_1 : A, B \mapsto a, b$

$t_2 : A, b \mapsto A, a$

$t_3 : B, a \mapsto B, b$

$t_4 : a, b \mapsto b, b$

Potential Reachability: Example

$$C = \{A, B\} \xrightarrow{t_1} \{a, b\} \xrightarrow{t_4} \{b, b\} = C'$$

Majority

$t_1 : A, B \mapsto a, b$

$t_2 : A, b \mapsto A, a$

$t_3 : B, a \mapsto B, b$

$t_4 : a, b \mapsto b, b$

$$\begin{array}{c} \\ A \\ B \\ a \\ b \end{array} \begin{array}{c} C \\ \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \end{array} \xrightarrow{t_1 \begin{pmatrix} -1 \\ -1 \\ 1 \\ 1 \end{pmatrix}} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \xrightarrow{t_4 \begin{pmatrix} 0 \\ 0 \\ -1 \\ 1 \end{pmatrix}} \begin{array}{c} \\ A \\ B \\ a \\ b \end{array} \begin{array}{c} C' \\ \begin{pmatrix} 0 \\ 0 \\ 0 \\ 2 \end{pmatrix} \end{array}$$

Potential Reachability: Example

$$C = \{A, B\} \xrightarrow{t_1} \{a, b\} \xrightarrow{t_4} \{b, b\} = C'$$

Majority

$t_1 : A, B \mapsto a, b$

$t_2 : A, b \mapsto A, a$

$t_3 : B, a \mapsto B, b$

$t_4 : a, b \mapsto b, b$

$$\begin{array}{c} \\ A \\ B \\ a \\ b \end{array} \begin{array}{c} C \\ \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \end{array} + \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{pmatrix} -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & 1 \end{pmatrix} \end{array} \cdot \begin{array}{c} x \\ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \end{array} = \begin{array}{c} \\ A \\ B \\ a \\ b \end{array} \begin{array}{c} C' \\ \begin{pmatrix} 0 \\ 0 \\ 0 \\ 2 \end{pmatrix} \end{array}$$

Potential Reachability

First Approximation to Reachability

C' is potentially reachable from C if there exists a counting vector $x : T \mapsto \mathbb{N}$ s.t. C, C', x is a solution to the state equation:

$$\boxed{C' = C + \Sigma_{t \in T} \left(\mathrm{post}(t) - \mathrm{pre}(t)\right) \cdot x(t)}$$

Potential Reachability

First Approximation to Reachability

C' is potentially reachable from C if there exists a counting vector $x : T \mapsto \mathbb{N}$ s.t. C, C', x is a solution to the state equation:

$$\boxed{C' = C + \Sigma_{t \in T} \left(\mathrm{post}(t) - \mathrm{pre}(t)\right) \cdot x(t)}$$

Satisfaction of the *state equation* is too crude, even for simple protocols: Too many false negatives for the well-specification problem.

Potential Reachability: Example

$$C = \{A, B\} \xrightarrow{t_1} \{a, b\} \nrightarrow \{a, a\} = C'$$

Majority

$t_1 : A, B \mapsto a, b$

$t_2 : A, b \mapsto A, a$

$t_3 : B, a \mapsto B, b$

$t_4 : a, b \mapsto b, b$

Potential Reachability: Example

$$C = \{A, B\} \xrightarrow{t_1} \{a, b\} \nrightarrow \{a, a\} = C'$$

Majority

$t_1 : A, B \mapsto a, b$

$t_2 : A, b \mapsto A, a$

$t_3 : B, a \mapsto B, b$

$t_4 : a, b \mapsto b, b$

$$\begin{array}{c} \\ A \\ B \\ a \\ b \end{array} \begin{array}{c} C \\ \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \end{array} + \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{pmatrix} -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & 1 \end{pmatrix} \end{array} \cdot \begin{array}{c} x \\ \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \end{array} = \begin{array}{c} \\ A \\ B \\ a \\ b \end{array} \begin{array}{c} C' \\ \begin{pmatrix} 0 \\ 0 \\ 2 \\ 0 \end{pmatrix} \end{array}$$

Potential Reachability: Example

$$C = \{A, B\} \xrightarrow{t_1} \{a, b\} \nrightarrow \{a, a\} = C'$$

Majority

$t_1 : A, B \mapsto a, b$

$t_2 : A, b \mapsto A, a$

$t_3 : B, a \mapsto B, b$

$t_4 : a, b \mapsto b, b$

Once an agent is in one of the states $\{A, b\}$, there will always be an agent in one of the the states $\{A, b\}$.

Potential Reachability: Example

$$C = \{A, B\} \xrightarrow{t_1} \{a, b\} \nrightarrow \{a, a\} = C'$$

Majority

$t_1 : A, B \mapsto a, b$

$t_2 : A, b \mapsto A, a$

$t_3 : B, a \mapsto B, b$

$t_4 : a, b \mapsto b, b$

Once an agent is in one of the states $\{A, b\}$, there will always be an agent in one of the the states $\{A, b\}$.

$\{A, b\}$ is called a *trap* of the protocol.

Potential Reachability: Example

$$C = \{A, B\} \xrightarrow{t_1} \{a, b\} \nrightarrow \{a, a\} = C'$$

Majority

$t_1 : A, B \mapsto a, b$

$t_2 : A, b \mapsto A, a$

$t_3 : B, a \mapsto B, b$

$t_4 : a, b \mapsto b, b$

Once an agent is in one of the states $\{A, b\}$, there will always be an agent in one of the the states $\{A, b\}$.

$\{A, b\}$ is called a *trap* of the protocol.

Dually, sets of states that stay empty, once they are emptied, are called *siphons*. Here, $\{A, B\}$ is a siphon.

Solution: Add constraints for traps and siphons!

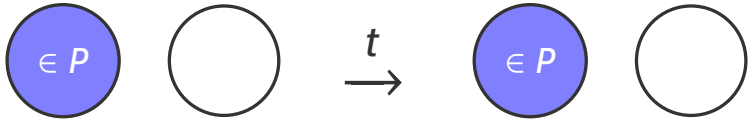
Trap/Siphon Constraints

Let $P \subseteq Q$ be a set of states

Definition (Trap)

P is a trap: Every transition that removes an agent from P also moves an agent into P:

$$\forall t \in T : \mathrm{pre}(t) \cap P \neq \emptyset \implies \mathrm{post}(t) \cap P \neq \emptyset$$



Trap/Siphon Constraints

Let $P \subseteq Q$ be a set of states

Definition (Siphon)

P is a siphon: Every transition that moves an agent into P also removes an agent from P:

$$\forall t \in T : \mathrm{post}(t) \cap P \neq \emptyset \implies \mathrm{pre}(t) \cap P \neq \emptyset$$

Potential Reachability

C' is *potentially reachable* from C if there is a counting vector x s.t. C, C', x is a solution to the state equation:

$$\boxed{C' = C + \Sigma_{t \in T} \left(\mathrm{post}(t) - \mathrm{pre}(t)\right) \cdot x(t)}$$

Moreover, the trap/siphon constraints must be satisfied for all $P \subseteq Q$:

1. If P is a trap then we have:
   $C \cap P \neq \emptyset$ implies $C' \cap P \neq \emptyset$.
2. If P is a siphon then we have:
   $C \cap P = \emptyset$ implies $C' \cap P = \emptyset$.

Potential Reachability: Finding traps and siphons

- Enumeration of all traps and siphons can be avoided:
  Only check largest unmarked trap/siphon at C'/C
  $\Rightarrow$ can be found in polynomial time given C and C'.

Potential Reachability: Finding traps and siphons

- Enumeration of all traps and siphons can be avoided:
  Only check largest unmarked trap/siphon at C'/C
  $\Rightarrow$ can be found in polynomial time given C and C'.
- However, implementation adds traps/siphons iteratively in counter-example guided abstraction refinement loop.

Potential Reachability: Finding traps and siphons

- Enumeration of all traps and siphons can be avoided:
  Only check largest unmarked trap/siphon at C'/C
  $\Longrightarrow$ can be found in polynomial time given C and C'.
- However, implementation adds traps/siphons iteratively in counter-example guided abstraction refinement loop.
- Additionally, finer refinements called U-traps and U-siphons are used, depending on set $U \subseteq T$ and support of x.

Deciding Strong Consensus

A protocol satisfies strong consensus iff there exist no configurations C_0, C_1, C_2 and $q_1, q_2 \in Q$ such that

$$C_0 \xrightarrow{*} C_1 \wedge C_0 \xrightarrow{*} C_2 \qquad \text{(reachability approximation)}$$

$$\mathrm{init}(C_0) \wedge \mathrm{term}(C_1) \wedge \mathrm{term}(C_2) \qquad \text{(configuration constraints)}$$

$$O(q_1) \neq O(q_2) \wedge C_1(q_1) \geq 1 \wedge C_2(q_2) \geq 1 \qquad \text{(different consensus)}$$

Deciding Strong Consensus

A protocol satisfies strong consensus iff there exist no configurations C_0, C_1, C_2 and $q_1, q_2 \in Q$ such that

$$C_0 \xrightarrow{*} C_1 \wedge C_0 \xrightarrow{*} C_2 \qquad \text{(reachability approximation)}$$

$$\mathrm{init}(C_0) \wedge \mathrm{term}(C_1) \wedge \mathrm{term}(C_2) \qquad \text{(configuration constraints)}$$

$$O(q_1) \neq O(q_2) \wedge C_1(q_1) \geq 1 \wedge C_2(q_2) \geq 1 \qquad \text{(different consensus)}$$

Can be encoded as an SMT problem over linear integer arithmetic:

$\Rightarrow$ **co-NP** decision procedure

Towards a verifiable class: WS^3

Original Property

Silent

For every init. C_0:

$$C_0 \xrightarrow{*} C' \implies$$

$$\exists \text{ terminal } C_\perp : C' \xrightarrow{*} C_\perp$$

Well-specified | Silent

All terminal configs reachable from init. C_0 form same consensus.

Approximation

Layered Termination

For every config C:

$\exists$ terminal $C_\perp$: $C \xrightarrow{*} C_\perp$

NP

due to universal termination strategy (of a certain form).

Strong Consensus

All terminal configs *potentially reachable* from init. C_0 form same consensus.

co-NP

Deciding correctness within WS^3

Given a Presburger predicate φ, a protocol in WS^3 is correct w.r.t. to φ iff there exist no configurations C_0, C_1 such that

$$C_0 \xrightarrow{*} C_1 \qquad \text{(reachability approximation)}$$

$$\mathrm{init}(C_0) \wedge \mathrm{term}(C_1) \qquad \text{(configuration constraints)}$$

$$\varphi(C_0) \neq O(C_1) \qquad \text{(incorrect consensus)}$$

Deciding correctness within WS^3

Given a Presburger predicate φ, a protocol in WS^3 is correct w.r.t. to φ iff there exist no configurations C_0, C_1 such that

$$C_0 \xrightarrow{*} C_1 \qquad \text{(reachability approximation)}$$

$$\mathrm{init}(C_0) \wedge \mathrm{term}(C_1) \qquad \text{(configuration constraints)}$$

$$\varphi(C_0) \neq O(C_1) \qquad \text{(incorrect consensus)}$$

Complexity depends on φ, but still in **co-NP** for quantifier-free Presburger predicates with threshold and remainder.

Can be combined with constraints for strong consensus into a single shared set of constraints.

Towards a verifiable class: WS^3

- Class WS^3 of well-specified strongly silent protocols.
- Membership test in **DP** $= \{L_1 \cap L_2 \mid L_1 \in \textbf{NP} \cap L_2 \in \textbf{co-NP}\}$.
- Correctness test for WS^3 protocols in **co-NP**.
- Can express all Presburger predicates
  (by combination of threshold and remainder protocol).
- Protocols from the literature for Majority, Threshold,
  Remainder, Flock of Birds etc. already belong to WS^3.

Counter-example generation

If either check for strong consensus or correctness fails, a *potential* counter-example given by C_0, C_1, C_2 is returned.

Actual reachability $C_0 \xrightarrow{*} C_1$ and $C_0 \xrightarrow{*} C_2$ can be checked by a model checker:

- Reachable: genuine counter-example found.
- Unreachable: configurations can be excluded by additional constraints and check repeated.

Peregrine: A Verification Tool for Population Protocols

- Features of Peregrine:
  - Allows specification of (parameterized) population protocols.
  - Visualizations of randomly generated executions.
  - Statistics generation of convergence speed until consensus.
  - Automatic verification of well-specification and correctness in the class WS^3.

Peregrine: A Verification Tool for Population Protocols

- Features of Peregrine:
  - Allows specification of (parameterized) population protocols.
  - Visualizations of randomly generated executions.
  - Statistics generation of convergence speed until consensus.
  - Automatic verification of well-specification and correctness in the class WS^3.
- Backend implemented in Haskell
  - Call SMT solver Z3 to check well-specification and correctness.
  - Call model checker LoLA to find counter-examples.
  - Handles simulations for quantitative analysis.

Peregrine: A Verification Tool for Population Protocols

- Features of Peregrine:
  - Allows specification of (parameterized) population protocols.
  - Visualizations of randomly generated executions.
  - Statistics generation of convergence speed until consensus.
  - Automatic verification of well-specification and correctness in the class WS^3.
- Backend implemented in Haskell
  - Call SMT solver Z3 to check well-specification and correctness.
  - Call model checker LoLA to find counter-examples.
  - Handles simulations for quantitative analysis.
- Web frontend using JavaScript
  - Specification of protocols using graphical editor or python.
  - Visualisation of simulations and simulation results.

Conclusions

- Use of new class suitable for verification of correctness
- First tool for automatic and entirely parametric verification of population protocols (for all initial population sizes!)
- Successful verification of existing protocols
- Interface for design and analysis of protocols

Conclusions

- Use of new class suitable for verification of correctness
- First tool for automatic and entirely parametric verification of population protocols (for all initial population sizes!)
- Successful verification of existing protocols
- Interface for design and analysis of protocols
- Possible future work:
  - Verification of non-silent protocols
  - Strengthening of reachability approximation
  - Diagnosis information when protocol is not strongly silent
  - Synthesis of small protocols
  - LTL model checking


Thank you!