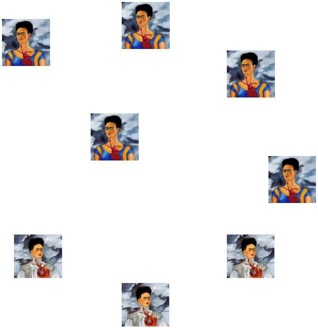
Parameterized verification of asynchronous round-based distributed algorithms reduced to nuXmv

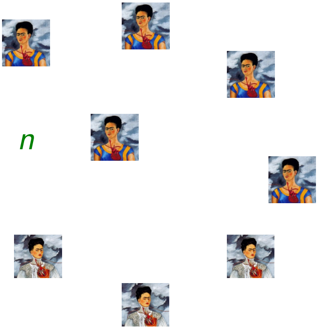
Nathalie Bertrand

joint work with Pranav Ghorpade and Sasha Rubin
University of Sydney

Fault-tolerant distributed algorithms

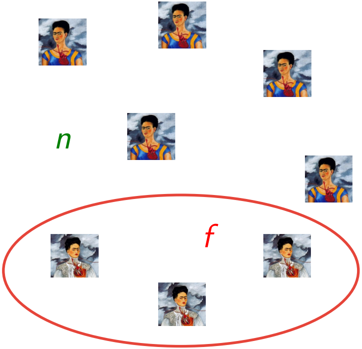


Fault-tolerant distributed algorithms



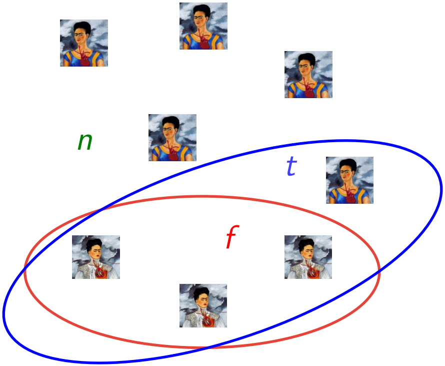
- n processes

Fault-tolerant distributed algorithms



- n processes
- f are faulty (e.g. crash or Byzantine failures)

Fault-tolerant distributed algorithms



- n processes
- f are faulty (e.g. crash or Byzantine failures)
- t known upper bound on f
- resilience condition between these parameters, e.g. $2t < n$

Asynchronous round-based distributed algorithms

Consensus or leader election protocols

- asynchronous communication by broadcast
- threshold guards on number of received messages
- finitely many local variables
- structured in rounds:
  - rounds are identical up to round index, used to tag messages
  - round increment not limited to $r := r + 1$

```
bool v := input_value({0,1});
int r := 1 ;
while (true) do
 broadcast (v,r) ;
 wait for n − t messages (∗,r);
 if received 2(n + t)/3 messages (w,r)
 then d := w; halt
 else if received (n + t)/2 messages (0,r)
 then v := 0; r:=r+2 ;
 else if received (n + t)/3 messages (1,r)
 then v := 1; r:=r+1 ;
od
```

Existing formal methods approaches

No fully automated techniques; Mostly human-guided methods

- interactiv theorem provers
  - TLA+ protocol formalization and verification for Paxos [Lamport, Merz, Doligez 2012], multi-Paxos [Chand, Liu, Stoller 2016] and DAG-based consensus in TLA+ [Bertrand, Ghorpade, Rubin, Scholz, Subotić 2025]
  - Rocq/VERDI specification and verification of Raft [Woos, Wilcox, Anton, Tatlock, Ernst, Anderson 2016]
- reduction to existing tools
  - restricted schedulers for randomized algorithms [Bertrand, Konnov, Lazić, Widder 2020]
- model checking with fixed number of processes
  - reduction theorem for finite instances to TLC [Chaouch-Saad, Charron-Bost, Merz 2009]
  - Paxos in SPIN [Delzanno, Tatarek, Traverso 2014]
  - agreement for asynchronous consensus algorithms [Noguchi, Tsuchiya, Kikuno 2012]

Our approach

Challenges

- 2 sources of infinity: number of processes, number of rounds
- asynchronous communications: unbounded *drift* between processes

Our approach

Challenges

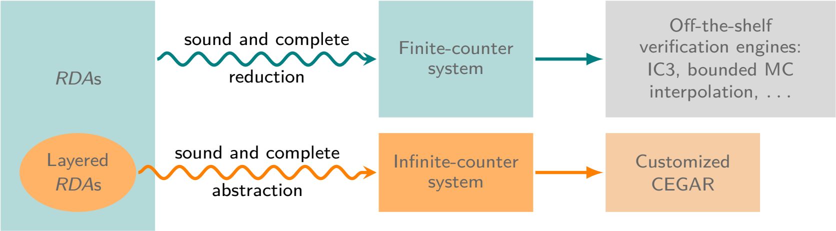
- 2 sources of infinity: number of processes, number of rounds
- asynchronous communications: unbounded *drift* between processes

Previous work [Bertrand, Thomas, Widder 2021] [Thomas, Sankur 2023]

Our approach

Challenges

- 2 sources of infinity: number of processes, number of rounds
- asynchronous communications: unbounded *drift* between processes



Previous work [Bertrand, Thomas, Widder 2021] [Thomas, Sankur 2023]

This work [Bertrand, Ghorpade, Rubin *under review*]

1. generalization of handled round-based distributed algorithms
2. reuse of mature model checkers e.g. nuXmv [Cavada *et al.* 2014]

Outline of the rest of the talk

1. Modelling formalism: process template and history state-count logic
2. Reduction steps
3. Experimental validation

Round-based process template

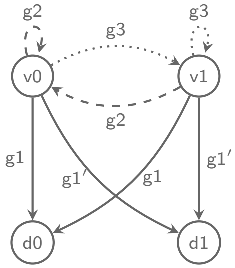
represents behaviour of a correct process

```
bool v := input_value({0,1});
int r := 1 ;
while (true) do
 broadcast (v,r) ;
 wait for n − t messages (∗,r);
 if received 2(n + t)/3 messages (w,r)
 then d := w; halt
 else if received (n + t)/2 messages (0,r)
 then v := 0; r:=r+2 ;
 else if received (n + t)/3 messages (1,r)
 then v := 1; r:=r+1 ;
od
```

Round-based process template

represents behaviour of a correct process

```
bool v := input_value({0,1});
int r := 1 ;
while (true) do
 broadcast (v,r) ;
 wait for n − t messages (∗,r);
 if received 2(n + t)/3 messages (w,r)
 then d := w; halt
 else if received (n + t)/2 messages (0,r)
 then v := 0; r:=r+2 ;
 else if received (n + t)/3 messages (1,r)
 then v := 1; r:=r+1 ;
od
```

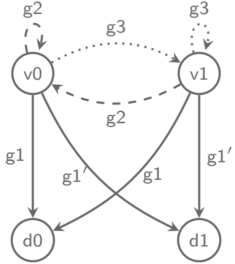


parameters $P = \{n, t\}$
resilience condition $rc = n > 2t$
locations $\mathcal{L} = \{v0, v1, d0, d1\}$
messages $\mathcal{M} = \{m0, m1\}$

Round-based process template

represents behaviour of a correct process

```
bool v := input_value({0,1});
int  r := 1 ;
while (true) do
 broadcast (v,r) ;
 wait for n − t messages (∗,r);
 if received 2(n + t)/3 messages (w,r)
 then d := w; halt
 else if received (n + t)/2 messages (0,r)
 then v := 0;  r:=r+2 ;
 else if received (n + t)/3 messages (1,r)
 then v := 1;  r:=r+1 ;
od
```



parameters $P = \{n, t\}$
resilience condition $rc = n > 2t$
locations $\mathcal{L} = \{v0, v1, d0, d1\}$
messages $\mathcal{M} = \{m0, m1\}$

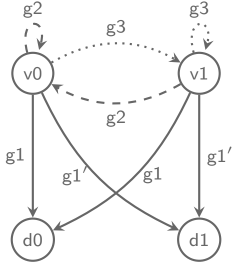
edges w. guard and round update

broadcast associated with locations
`Bcast:` w0$\mapsto$ m0
w1$\mapsto$ m1
d0 $\mapsto \perp$
d1$\mapsto \perp$

Round-based process template

represents behaviour of a correct process

```
bool v := input_value({0,1});
int  r := 1 ;
while (true) do
 broadcast (v,r) ;
 wait for n − t messages (∗,r);
 if received 2(n + t)/3 messages (w,r)
 then d := w; halt
 else if received (n + t)/2 messages (0,r)
 then v := 0; r:=r+2 ;
 else if received (n + t)/3 messages (1,r)
 then v := 1; r:=r+1 ;
od
```



parameters $P = \{n, t\}$
resilience condition $rc = n > 2t$
locations $\mathcal{L} = \{v0, v1, d0, d1\}$
messages $\mathcal{M} = \{m0, m1\}$

edges w. guard and round update

broadcast associated with locations
$\texttt{Bcast}$: $w0 \mapsto m0$
$w1 \mapsto m1$
$d0 \mapsto \perp$
$d1 \mapsto \perp$

$g1 = \texttt{Quorum} \wedge m0 > 2(n+t)/3$
$g1' = \texttt{Quorum} \wedge m1 > 2(n+t)/3$
$g2 = \texttt{Quorum} \wedge m0 > n + t/2$
$g3 = \texttt{Quorum} \wedge m0 \leq n + t/2 \wedge m1 \geq n + t/3$
where $\texttt{Quorum} = m0 + m1 \geq n - t$

$\longrightarrow$: no round increment
$\dashrightarrow$: round increment of 1
$\cdots\rightarrow$: round increment of 2

Semantics

Fixed-instance semantics $\mathcal{S}(\mathcal{T}, \nu)$

- ν: fixed values for parameters (n and t)
- n processes execute the same template
- configurations
  - process state: current location and round index, multiset of received messages
  - network state: multiset of broadcast messages
- actions
  - reception of a message by a process
  - process update according to a template rule (if guard permits; updates location and round index)

$\rightarrow$ infinitely many finite-valued variables

(counting the processes in each location and round)

Parameterized semantics $\mathcal{S}(\mathcal{T}) = \sqcup_{\nu \models \mathrm{RC}} \mathcal{S}(\mathcal{T}, \nu)$

$\rightarrow$ infinitely many unbounded variables

History State-Count Logic

$$\psi ::= \forall r.\ \alpha_r \mid \beta \mid \neg\psi \mid \psi \wedge \psi$$

round-local atom $\alpha_r ::= \sum_{\ell \in \mathcal{L}} c_\ell \cdot \kappa(\ell, r) \leq \varphi(n, t)$

cumulative atom $\beta ::= \sum_{\ell \in \mathcal{X}} c_\ell \cdot \sum_{r \in \mathbb{N}} \kappa(\ell, r) \leq \varphi(n, t)$

$\varphi(n, t)$ is a linear term with variables n and t

$\kappa(\ell, r)$ counts the number of process visits to location ℓ in round r

History State-Count Logic

$$\psi ::= \forall r.\ \alpha_r \mid \beta \mid \neg\psi \mid \psi \wedge \psi$$

round-local atom α_r $::=$ $\sum_{\ell \in \mathcal{L}} c_\ell \cdot \kappa(\ell, r) \ \leq \ \varphi(n, t)$
cumulative atom β $::=$ $\sum_{\ell \in \mathcal{X}} c_\ell \cdot \sum_{r \in \mathbb{N}} \kappa(\ell, r) \ \leq \ \varphi(n, t)$

$\varphi(n, t)$ is a linear term with variables n and t

$\kappa(\ell, r)$ counts the number of process visits to location ℓ in round r

Expressivity of HSCL

- Agreement $:= \forall r.\kappa(\mathrm{d0}, r) \leq 0 \ \vee \forall_r.\kappa(\mathrm{d1}, r) \leq 0$
- `Validity` $:= \forall r.\kappa(\mathrm{d0}, r) \leq 0$ (assuming all start with $v = 1$)
- `Termination` $:= \neg(\sum_r \kappa(\mathrm{d0}, r) + \kappa(\mathrm{d1}, r) \leq \mathtt{N_c} - 1)$
- `RestrictedTermination` $:= \neg(\sum_r \kappa(\mathrm{d0}, r) + \kappa(\mathrm{d1}, r) \leq 0) \longrightarrow$ `Term`
- `LeaderUniqueness` $:= \forall r.\kappa(\mathrm{ldr}, r) \leq 1$

Overview of reductions (1)

- Step 1: **received message abstraction**
  - only sent messages are kept in the network state
  - local counters for received messages are abstracted away
  - similar in spirit to e.g. [Stoilkovska, Konnov, Widder, Zuleger 2020]
  - always sound, and also complete for *common* templates
    within a round subsequent guards are monotone
    e.g. $m0 \geq n/3$ cannot follow $m0 \geq n/2$

Overview of reductions (1)

- Step 1: **received message abstraction**
  - only sent messages are kept in the network state
  - local counters for received messages are abstracted away
  - similar in spirit to e.g. [Stoilkovska, Konnov, Widder, Zuleger 2020]
  - always sound, and also complete for *common* templates
    within a round subsequent guards are monotone
    e.g. $m0 \geq n/3$ cannot follow $m0 \geq n/2$

- Step 2: **process identity abstraction**
  - process ids are irrelevant, only number of processes in each location and round matter
  - classical counting abstraction from parameterized verification of systems composed of identical anonymous processes [German, Sistla 1992]

Overview of reductions (2)

- Step 3: **synchronous restriction**
  - re-ordering to focus on "semi-synchronous" executions
    the sequence of target round indices is non-decreasing
  - commutativity arguments [Chaouch-Saad, Charron-Bost, Merz 2009]

Overview of reductions (2)

- Step 3: **synchronous restriction**
  - re-ordering to focus on "semi-synchronous" executions
    the sequence of target round indices is non-decreasing
  - commutativity arguments [Chaouch-Saad, Charron-Bost, Merz 2009]

- Step 4: **bounded-window abstraction**
  - counters that no longer influence updates can be forgotten
  - window of size $b + 1$ suffices, with b bound on round increment in template
  - sliding window can be encoded with finitely many counters

Overview of reductions (2)

- Step 3: **synchronous restriction**
  - re-ordering to focus on "semi-synchronous" executions
    the sequence of target round indices is non-decreasing
  - commutativity arguments [Chaouch-Saad, Charron-Bost, Merz 2009]

- Step 4: **bounded-window abstraction**
  - counters that no longer influence updates can be forgotten
  - window of size $b+1$ suffices, with b bound on round increment in template
  - sliding window can be encoded with finitely many counters

For common templates, these four steps are **sound and complete** for history state-count properties.

From HSCL to LTL

Two more transformation steps on models: history-record extension and round identify abstraction

$\rightarrow$ allows to reduce HSCL verification to LTL model checking

From HSCL to LTL

Two more transformation steps on models: history-record extension and round identify abstraction

$\rightarrow$ allows to reduce HSCL verification to LTL model checking

- Agreement $:= \mathbf{G}\ \big(\mathsf{local}(\mathsf{d0}) \leq 0 \vee \mathsf{local}(\mathsf{d1}) \leq 0\big)$
- `Validity` $:= \mathbf{G}\big(\mathsf{local}(\mathsf{d0}) \leq 0\big)$
- `Termination` $:= \neg\mathbf{G}\big(\mathsf{cumul}(\mathsf{d0}) + \mathsf{cumul}(\mathsf{d1}) \leq \mathtt{N_c}\big)$
- `RestrictedTermination` $:= \mathbf{G}\big(\mathsf{cumul}(\mathsf{d0}) + \mathsf{cumul}(\mathsf{d1}) \leq 0\big) \longrightarrow$ `Termination`
- `LeaderUniqueness` $:= \mathbf{G}\big(\mathsf{local}(\mathsf{ldr}) \leq 1\big)$

From HSCL to LTL

Two more transformation steps on models: history-record extension and round identify abstraction

→ allows to reduce HSCL verification to LTL model checking

- Agreement $:= \mathbf{G}\ \big(\mathsf{local}(\mathsf{d0}) \leq 0 \vee \mathsf{local}(\mathsf{d1}) \leq 0\big)$
- `Validity` $:= \mathbf{G}\big(\mathsf{local}(\mathsf{d0}) \leq 0\big)$
- `Termination` $:= \neg\mathbf{G}\big(\mathsf{cumul}(\mathsf{d0}) + \mathsf{cumul}(\mathsf{d1}) \leq \mathtt{N_c}\big)$
- `RestrictedTermination` $:= \mathbf{G}\big(\mathsf{cumul}(\mathsf{d0}) + \mathsf{cumul}(\mathsf{d1}) \leq 0\big) \longrightarrow$ `Termination`
- `LeaderUniqueness` $:= \mathbf{G}\big(\mathsf{local}(\mathsf{ldr}) \leq 1\big)$

Parameterized verification of HSCL on round-based distributed algorithms reduces to LTL model checking on finite-counter systems

Rk: for fixed parameters, reduction to LTL over finite-state systems

Experimental validation

Case studies to demonstrate applicability of the approach

- `.smv` file with counter system and LTL properties
- IC3 engine of nuXmv: `check_ltlspec_ic3`

3 consensus algorithms with round increment of at most 1

Protocol	loc.	rules	rc	Agree.	Valid.	Term.	R. Term.
Ben-Or (crash)	9	26	$n > 2t$	1.4s (13)	0.4s (9)	0.5 (3)	3.1s (8)
Ben-Or (Byz.)	10	27	$n > 5t$	7.0s (11)	1.2s (7)	0.6 (3)	4.3s (7)
Bracha (Byz.)	12	31	$n > 3t$	14.0s (14)	1.8s (8)	0.7 (3)	6.5s (11)

1 leader election protocol with round increment of at most 2

Protocol	b	loc.	rules	rc	Leader U.
Raft leader election	2	11	25	$n > 2t$	1.8s (8)

Additional tests: bugged variants (altered guards or resilience condition) detected within seconds; also verification of fixed parameter valuations (thus finite-state model checking)

Conclusion and future work

Contribution

- verification of correctness properties of round-based distributed algorithms
- reduction theorems to encode it into LTL verification over finite-counter system
- experiments conducted with symbolic model checker nuXmv

Conclusion and future work

Contribution

- verification of correctness properties of round-based distributed algorithms
- reduction theorems to encode it into LTL verification over finite-counter system
- experiments conducted with symbolic model checker nuXmv

Remaining challenges

- how to model and verify probabilistic behaviors?
- how to deal with unbounded round jumps?
- how to deal with algorithms in which the number of locations per round grows with round index?

Conclusion and future work

Contribution

- verification of correctness properties of round-based distributed algorithms
- reduction theorems to encode it into LTL verification over finite-counter system
- experiments conducted with symbolic model checker nuXmv

Remaining challenges

- how to model and verify probabilistic behaviors?
- how to deal with unbounded round jumps?
- how to deal with algorithms in which the number of locations per round grows with round index?

Vielen Dank für Ihre Aufmerksamkeit!