SMT and POR beat Counter Abstraction

Parameterized Model Checking of Threshold-Based Distributed Algorithms

Igor Konnov Helmut Veith Josef Widder

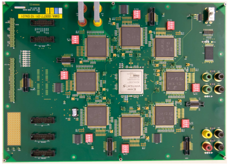
Alpine Verification Meeting

May 4-6, 2015

Why fault-tolerant (FT) distributed algorithms

faults not in the control of system designer

- bit-flips in memory
- power outage
- disconnection from the network
- intruders take control over some computers



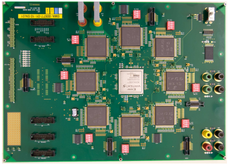
Why fault-tolerant (FT) distributed algorithms

faults not in the control of system designer

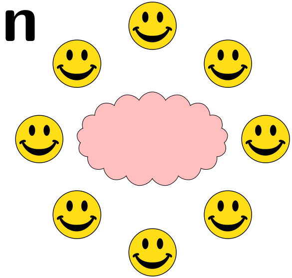
- bit-flips in memory
- power outage
- disconnection from the network
- intruders take control over some computers

distributed algorithms to make systems more reliable even in the presence of faults

- replicate processes
- exchange messages
- do coordinated computation
- goal: keep replicated processes in "good state"

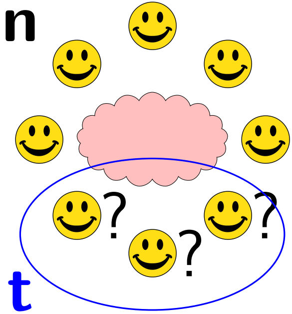


Fault-tolerant distributed algorithms



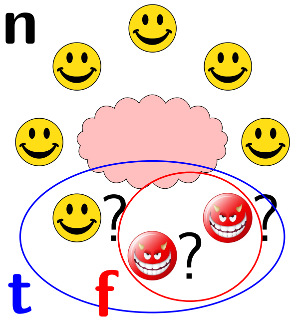
- n processes communicate by messages

Fault-tolerant distributed algorithms



- n processes communicate by messages
- all processes know that at most t of them might be faulty

Fault-tolerant distributed algorithms



- n processes communicate by messages
- all processes know that at most t of them might be faulty
- f are actually faulty, e.g., Byzantine
- resilience condition, e.g., $n > 3t \wedge t \geq f \geq 0$
- no masquerading: the processes know the origin of incoming messages

Distributed algorithms: computational model and faults

The classic model by [Fischer, Lynch, Paterson'85]

Environment:

- Asynchronous processes (no rounds, non-deterministic fair scheduler)
- Reliable asynchronous message passing (non-blocking send and receive)

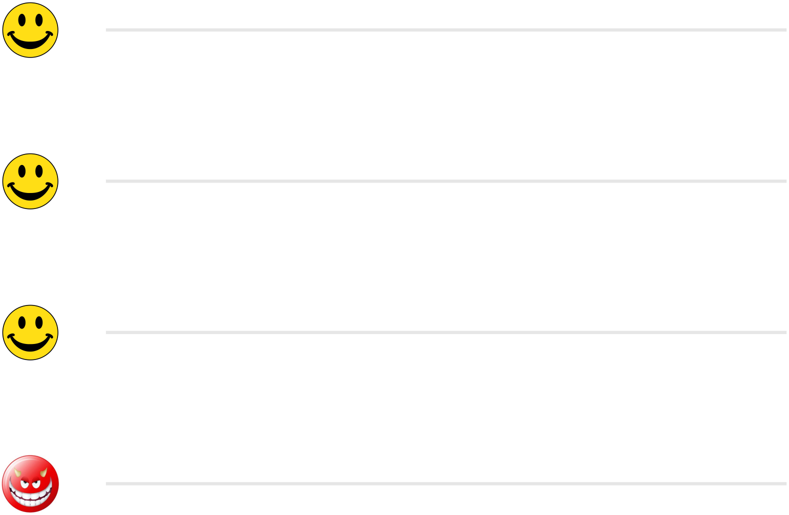
Faults:

- crashes and clean crashes,
- omission faults,
- symmetric faults,
- Byzantine faults

Reliable Broadcast by Srikanth & Toueg 85

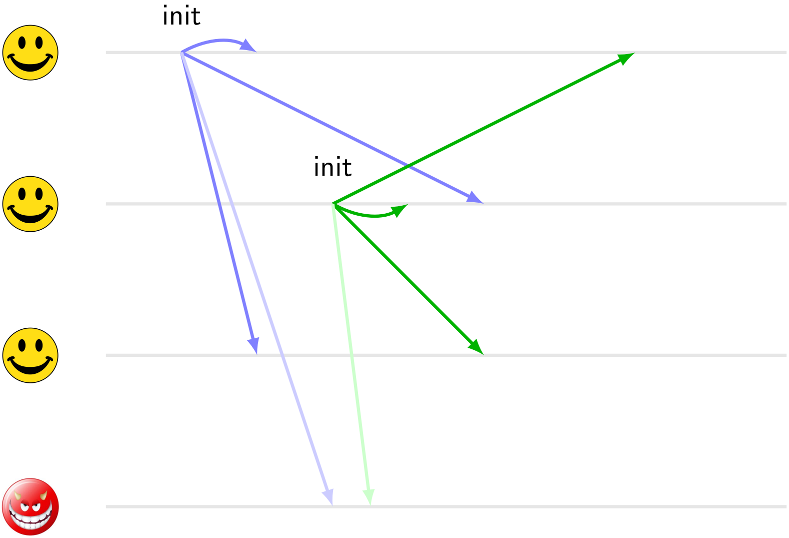
```
if initiator then send INIT to all;

while true do
 if received INIT from at least 1 distinct proc.
 then send ECHO to all;

 if received ECHO from at least t + 1 distinct proc.
    and not sent ECHO before
 then send ECHO to all;

 if received ECHO from at least n - t distinct proc.
 then accept;
od
```

Reliable Broadcast: Sample Execution

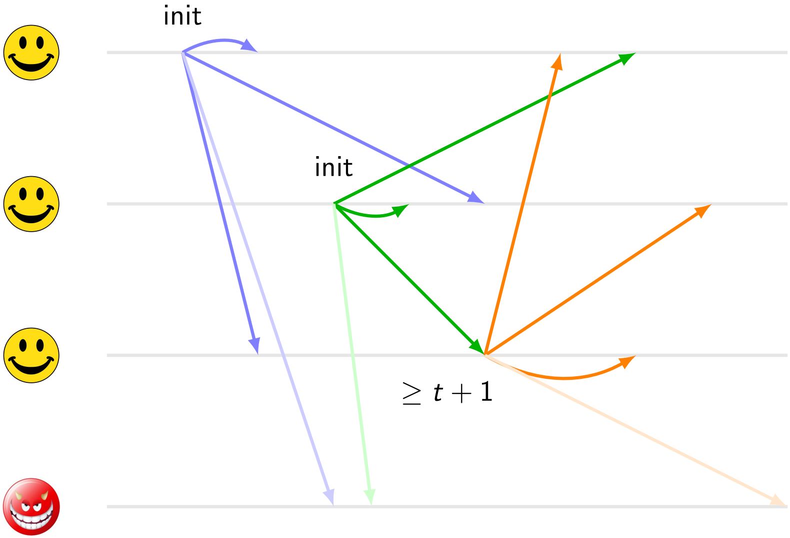


Reliable Broadcast: Sample Execution

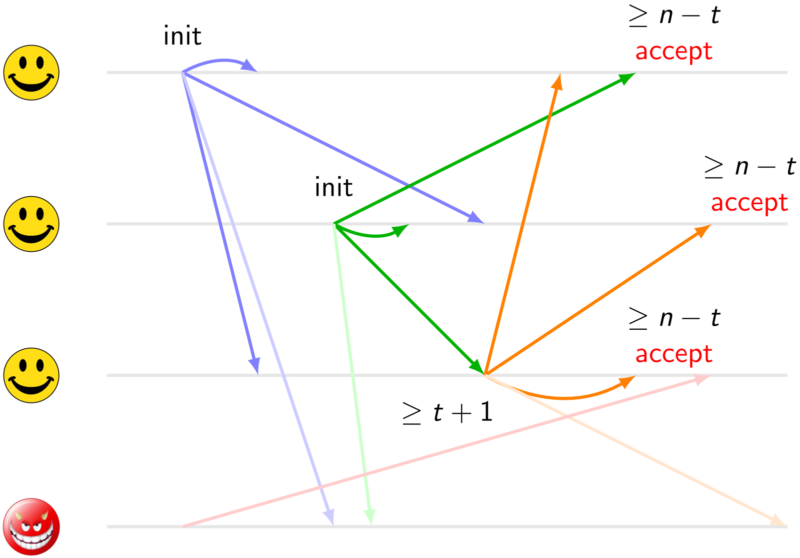
Reliable Broadcast: Sample Execution



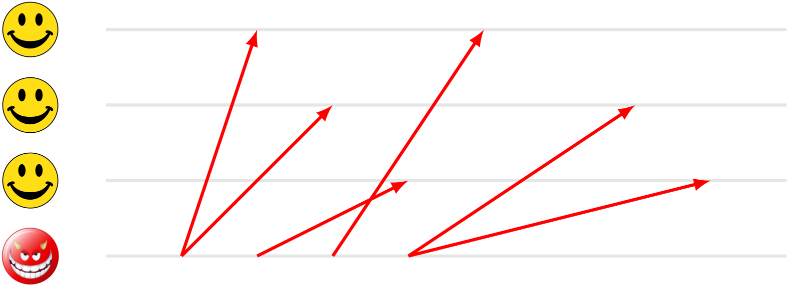
Reliable Broadcast: Sample Execution



Reliable Broadcast: Sample Execution



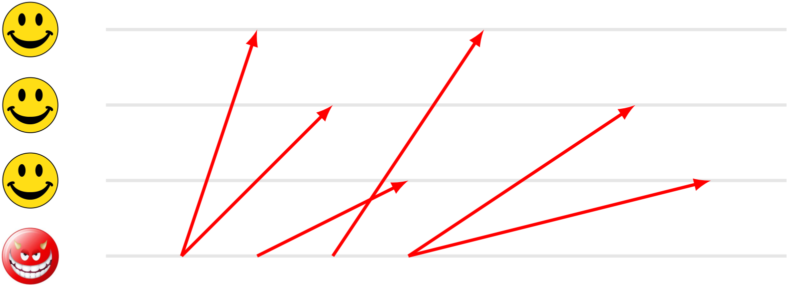
Reliable Broadcast: Sample Execution 2



Unforgeability: If no correct process sends <INIT> (broadcasts), then no correct process ever accepts.

Verification perspective: check, whether a bad state is reachable.

Reliable Broadcast: Sample Execution 2



Unforgeability: If no correct process sends <INIT> (broadcasts), then no correct process ever accepts.

Verification perspective: check, whether a bad state is reachable.

Threshold-based fault-tolerant distributed algorithms

- The parameters (n, t, f) are fixed in each run
- Main loop with the body executed atomically
- Processes are anonymous (no identifiers)

- Receiving messages, counting them and comparing to thresholds, e.g.,

  `if received <ECHO> from` $t+1$ `distinct processes`

  `then ...`

- Sending messages to all processes, e.g.,

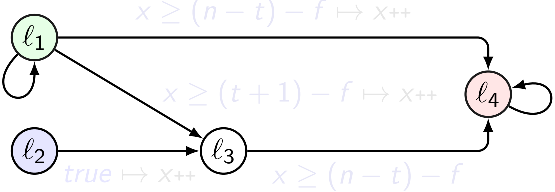
  ```
  send <ECHO> to all
  ```

Outline

1. **Threshold automata (TA)**:
   formalization of process code using shared variables
2. **Counter systems with acceleration**:
   computational model for parameterized systems of TA
3. **Parameterized reachability**:
   safety properties stated formally
4. **Counter abstraction and acceleration:**
   other approaches
5. **Representatives and schemas:**
   parameterized bounded model checking with SMT

Preliminaries

Every correct process follows the control flow graph (L, E):



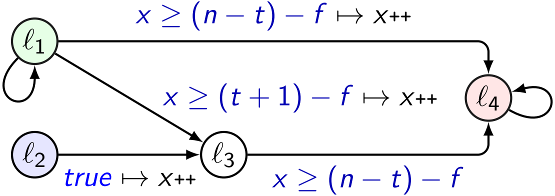
Processes move from one location to another along the edges labeled with:

- **Threshold guards**, e.g., $x \geq (t+1) - f$
  compare a shared variable to a linear combination of parameters.
- **Updates**, e.g., $x{+}{+}$
  increment shared variables (or do nothing).

(multiple guards and increments are allowed)

Threshold automata (TA)

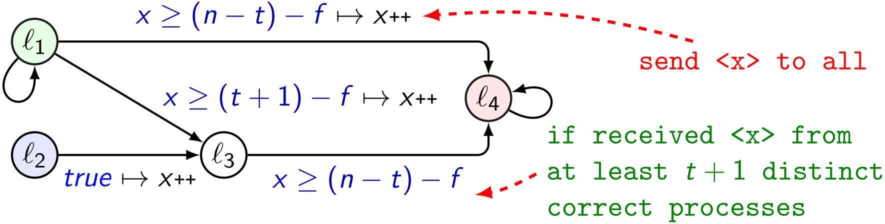
Every correct process follows the control flow graph (L, E):



Processes move from one location to another along the edges labeled with:

- **Threshold guards**, e.g., $x \geq (t+1) - f$
  compare a shared variable to a linear combination of parameters.
- **Updates**, e.g., $x{+}{+}$
  increment shared variables (or do nothing).

(multiple guards and increments are allowed)



Crash faults:

run n processes,

Byzantine faults:

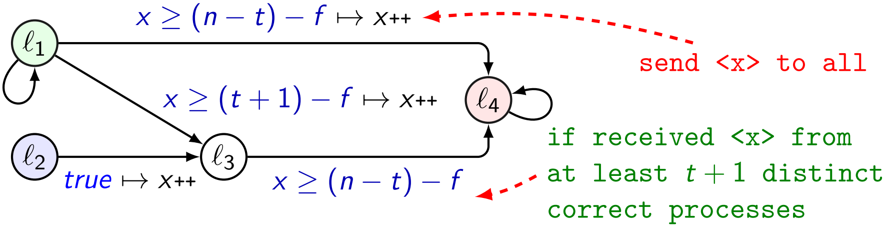
run $n - f$ processes,

count messages modulo Byzantine processes, e.g., $x + f \geq (t+1)$

Warning:

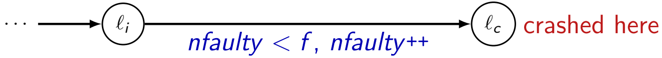
This requires preliminary abstraction of message counters [FMCAD'13]

Intuition: threshold automata and threshold-based DAs?



Crash faults:

run n processes,



Byzantine faults:

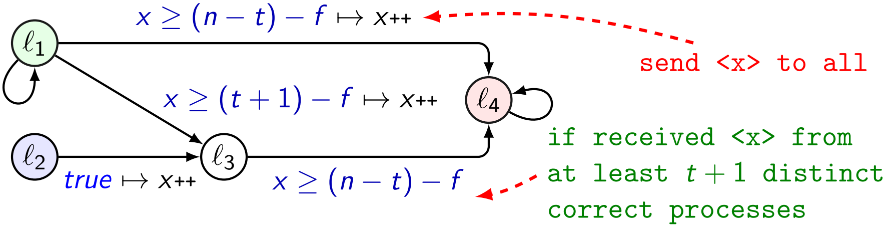
run $n - f$ processes,

count messages modulo Byzantine processes, e.g., $x + f \geq (t+1)$

Warning:

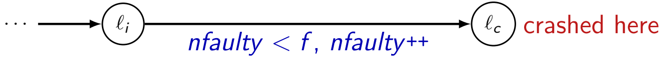
This requires preliminary abstraction of message counters [FMCAD'13]

Intuition: threshold automata and threshold-based DAs?



Crash faults:

run n processes,



Byzantine faults:

run $n - f$ processes,

count messages modulo Byzantine processes, e.g., $x + f \geq (t+1)$

Warning:

This requires preliminary abstraction of message counters [FMCAD'13]

Natural Restrictions of TA

Recall how processes count messages:

`if received <ECHO> from` $t+1$ **distinct** `processes`

The case studies lead us to the natural restrictions on threshold automata:

Restriction 1: Every process changes a shared variable at most once

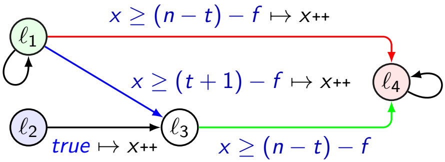
Restriction 2: The edges in cycles do not change the shared variables

Counter system with acceleration!

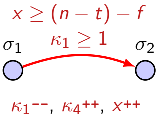
Counter system is a transition system simulating every system $P(\mathbf{p})^{N(\mathbf{p})}$.

Configuration $\sigma = (\kappa, \mathbf{g}, \mathbf{p})$:

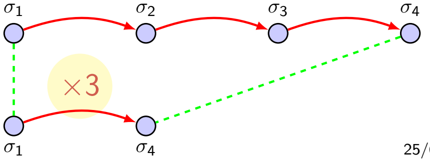
- κ_i counts processes at location ℓ_i with $\kappa_1 + \cdots + \kappa_{|L|} = N(\mathbf{p})$,
- g_j is the value of the shared variable x_j,
- $\mathbf{p}$ are the values of the parameters.



one transition r^1 (interleaving):



accelerated transition r^3:



Reachability and parameterized reachability

Reachability (fixed parameters):

Fix the parameters, e.g., $n = 4$, $t = 1$, $f = 1$, $N = n - f = 3$.

Fix configurations σ and σ' of P^N.

Question: is σ' reachable from σ in P^N?

Parameterized reachability:

Fix properties S and S' on configurations,

e.g., $S : \kappa_1 = N(\mathbf{p}) = n - f$ and $S' : \kappa_4 \neq 0$.

Question: are there parameter values $\mathbf{p}$ and configurations σ, σ' of $P^{N(\mathbf{p})}$:

- parameters $\mathbf{p}$ satisfy the resilience condition $RC(\mathbf{p})$,
- $\sigma \models S$ and $\sigma' \models S'$,
- σ' is reachable from σ in $P^{N(\mathbf{p})}$.

Reachability and parameterized reachability

Reachability (fixed parameters):

Fix the parameters, e.g., $n = 4$, $t = 1$, $f = 1$, $N = n - f = 3$.

Fix configurations σ and σ' of P^N.

Question: is σ' reachable from σ in P^N?

Parameterized reachability:

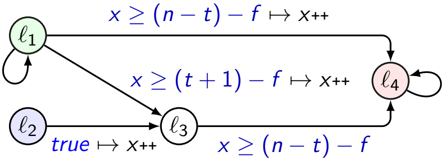
Fix properties S and S' on configurations,

e.g., $S : \kappa_1 = N(\mathbf{p}) = n - f$ and $S' : \kappa_4 \neq 0$.

Question: are there parameter values $\mathbf{p}$ and configurations σ, σ' of $P^{N(\mathbf{p})}$:

- parameters $\mathbf{p}$ satisfy the resilience condition $RC(\mathbf{p})$,
- $\sigma \models S$ and $\sigma' \models S'$,
- σ' is reachable from σ in $P^{N(\mathbf{p})}$.

Parameterized reachability: Example 1



Resilience condition 1: $n > 3t$ and $t \geq f \geq 0$.

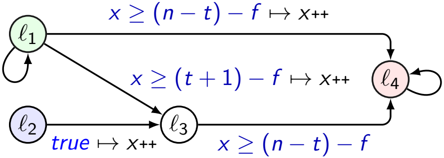
Can the faulty processes forge the broadcast by a correct process?

that is, can correct processes reach ℓ_4, if they start at ℓ_1? NO

$$(t+1) - f > 0 = x$$

$$(n-t) - f \geq n - t - t > t \geq 0 = x$$

Parameterized reachability: Example 1



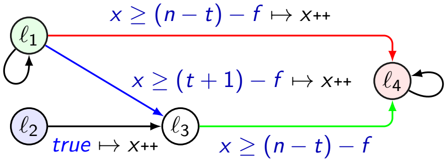
Resilience condition 1: $n > 3t$ and $t \geq f \geq 0$.

Can the faulty processes forge the broadcast by a correct process?

that is, can correct processes reach ℓ_4, if they start at ℓ_1? NO

$$(t+1) - f > 0 = x$$

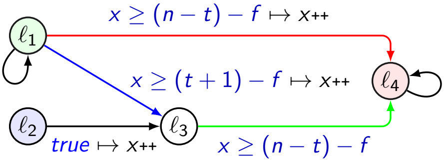
$$(n-t) - f \geq n - t - t > t \geq 0 = x$$



Resilience condition 2: $n > 3t$ and $t + 1 \geq f \geq 0$.

Can the faulty processes forge the broadcast by a correct process?

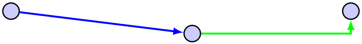
that is, can correct processes reach ℓ_4, if they start at ℓ_1? YES



Resilience condition 2: $n > 3t$ and $t + 1 \geq f \geq 0$.

Can the faulty processes forge the broadcast by a correct process?

that is, can correct processes reach ℓ_4, if they start at ℓ_1? YES



$\kappa_1 = 3$	$\kappa_1 = 0$	$\kappa_1 = 0$
$\kappa_2 = 0$	$\kappa_2 = 0$	$\kappa_2 = 0$
$\kappa_3 = 0$	$\kappa_3 = 3$	$\kappa_3 = 0$
$\kappa_4 = 0$	$\kappa_4 = 0$	$\kappa_4 = 3$
$x = 0$	$x = 3$	$x = 3$

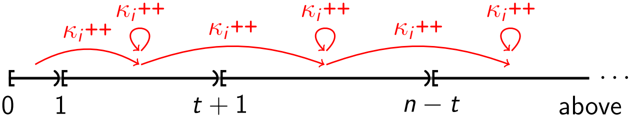
Parameterized reachability:

counter abstraction and acceleration

Use counter abstraction to get a finite system $\mathcal{A}$.

Counters κ_i are mapped to a finite domain $\widehat{D}$, e.g.,

- $\{0, 1, \infty\}$ by [Pnueli, Xu, Zuck'02].
- Domain of parametric intervals extracted from thresholds,
  e.g., $\{[0,1), [1, t+1), [t+1, n-t), [n-t, \infty)\}$, see [FMCAD'13].



Use a finite-state model checker, e.g., NuSMV or Spin

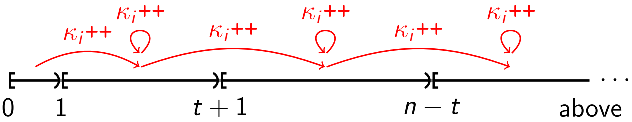
Warning:

Sometimes, abstraction refinement is needed [FMCAD'13]

Use counter abstraction to get a finite system $\mathcal{A}$.

Counters κ_i are mapped to a finite domain $\widehat{D}$, e.g.,

- $\{0, 1, \infty\}$ by [Pnueli, Xu, Zuck'02].
- Domain of parametric intervals extracted from thresholds, e.g., $\{[0, 1), [1, t+1), [t+1, n-t), [n-t, \infty)\}$, see [FMCAD'13].



Use a finite-state model checker, e.g., NuSMV or Spin

Warning:

Sometimes, abstraction refinement is needed [FMCAD'13]

Bounded diameter

Fix a threshold automaton TA and a size function N.

Theorem [CONCUR'14]

For each $\mathbf{p}$ with $RC(\mathbf{p})$, the diameter of an accelerated counter system is independent of parameters and is less than or equal to $|E| \cdot (|\mathcal{C}| + 1) + |\mathcal{C}|$:

- $|E|$ is the number of edges in TA (self-loops excluded).
- $|\mathcal{C}|$ is the number of edge conditions in TA that can be unlocked (locked) by an edge appearing later (resp. earlier) in the control flow, or by a parallel edge.

In our example:

$|E| = 4$, $|\mathcal{C}| = 1$.

Thus, $d \leq 9$.

Fix a threshold automaton TA and a size function N.

Theorem [CONCUR'14]

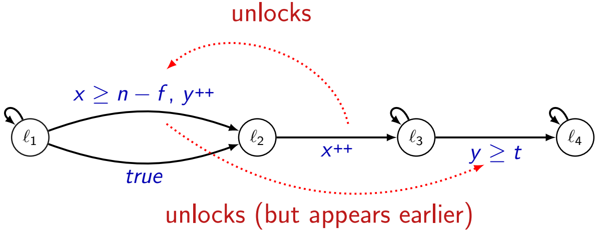
For each $\mathbf{p}$ with $RC(\mathbf{p})$, the diameter of an accelerated counter system is independent of parameters and is less than or equal to $|E| \cdot (|\mathcal{C}| + 1) + |\mathcal{C}|$:

- $|E|$ is the number of edges in TA (self-loops excluded).
- $|\mathcal{C}|$ is the number of edge conditions in TA that can be unlocked (locked) by an edge appearing later (resp. earlier) in the control flow, or by a parallel edge.

In our example:

$|E| = 4$, $|\mathcal{C}| = 1$.

Thus, $d \leq 9$.

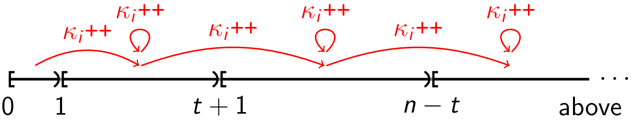


Way 2: Complete parameterized bounded model checking

Use counter abstraction to get a finite system $\mathcal{A}$.

Counters κ_i are mapped to a finite domain $\widehat{D}$, e.g.,

- $\{0, 1, \infty\}$ by [Pnueli, Xu, Zuck'02].
- Domain of parametric intervals extracted from thresholds, e.g., $\{[0,1), [1, t+1), [t+1, n-t), [n-t, \infty)\}$, see [FMCAD'13].



Once we know the diameter d of the accelerated counter system,

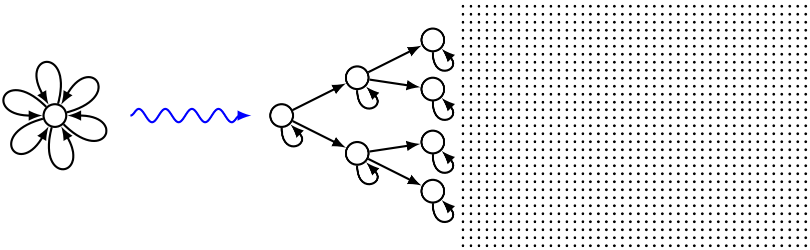
we know the diameter of the abstract system:

$$diam(\mathcal{A}) \leq d \cdot (|\widehat{D}| - 1)$$

Threshold automata are a special case of counter automata.

Apply symbolic acceleration techniques for counter automata, e.g., FAST [Bardin, Finkel, Leroux et al.'08].

The diameter bound implies that the threshold automata are flattable

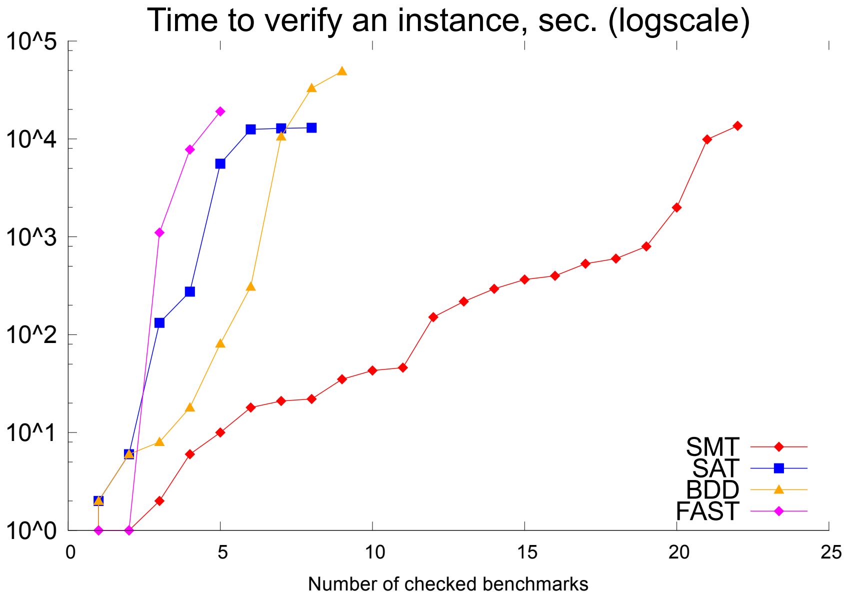


Thus, FAST always terminates on threshold automata (in theory)

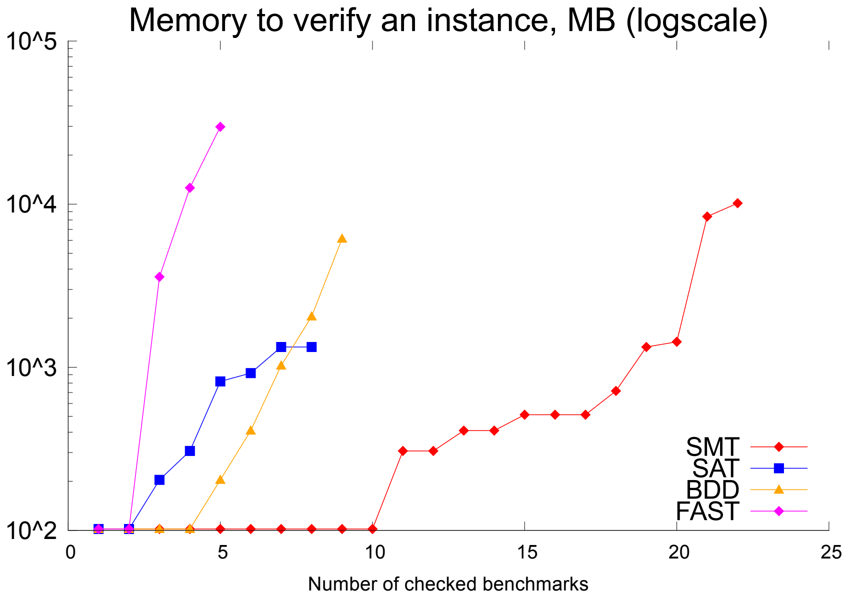
Accelerated systems:

partial order reduction and SMT

Partial orders and SMT beat counter abstraction



Partial orders and SMT beat counter abstraction (2)



Our new solution

Our new solution consists of the key ingredients:

Contexts: In every execution, evaluation of a guard changes at most once

e.g., $x \geq t + 1 - f$ is initially false and later turns to true.
A *context* keeps track of all unlocked guards.

Representatives: As before, transform every execution to a representative by reordering and accelerating the rules with the same context.

the schedule $r_1^1 r_2^1 r_1^1 r_2^1 r_2^1$ becomes $r_1^2 r_2^3$.

Schemas: Representatives are generated by schemas.

e.g., $r_1 r_2$ generates schedule $r_1^2 r_2^3$ by picking acceleration factors 2 and 3.

offline partial order reduction

Our new solution

Our new solution consists of the key ingredients:

Contexts: In every execution, evaluation of a guard changes at most once

e.g., $x \geq t + 1 - f$ is initially false and later turns to true.

A *context* keeps track of all unlocked guards.

Representatives: As before, transform every execution to a representative by reordering and accelerating the rules with the same context.

the schedule $r_1^1 r_2^1 r_1^1 r_2^1 r_2^1$ becomes $r_1^2 r_2^3$.

Schemas: Representatives are generated by schemas.

e.g., $r_1 r_2$ generates schedule $r_1^2 r_2^3$ by picking acceleration factors 2 and 3.

offline partial order reduction

Our new solution

Our new solution consists of the key ingredients:

Contexts: In every execution, evaluation of a guard changes at most once

e.g., $x \geq t + 1 - f$ is initially false and later turns to true.
A *context* keeps track of all unlocked guards.

Representatives: As before, transform every execution to a representative by reordering and accelerating the rules with the same context.

the schedule $r_1^1 r_2^1 r_1^1 r_2^1 r_2^1$ becomes $r_1^2 r_2^3$.

Schemas: Representatives are generated by schemas.

e.g., $r_1 r_2$ generates schedule $r_1^2 r_2^3$ by picking acceleration factors 2 and 3.

offline partial order reduction

Our new solution

Our new solution consists of the key ingredients:

Contexts: In every execution, evaluation of a guard changes at most once

e.g., $x \geq t + 1 - f$ is initially false and later turns to true.
A *context* keeps track of all unlocked guards.

Representatives: As before, transform every execution to a representative by reordering and accelerating the rules with the same context.

the schedule $r_1^1 r_2^1 r_1^1 r_2^1 r_2^1$ becomes $r_1^2 r_2^3$.

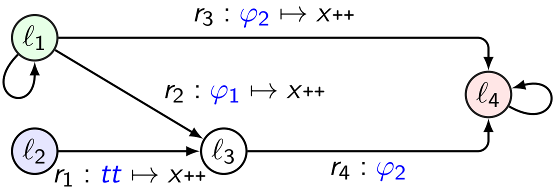
Schemas: Representatives are generated by schemas.

e.g., $r_1 r_2$ generates schedule $r_1^2 r_2^3$ by picking acceleration factors 2 and 3.

offline partial order reduction

Contexts and representatives

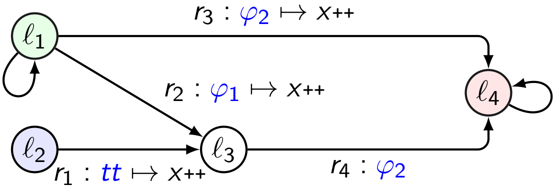
Contexts



Φ is the set of all threshold guards of TA,
e.g., $\Phi = \{\varphi_1, \varphi_2\}$

A subset $\Omega \subseteq \Phi$ is a **context**,
e.g., $\emptyset$, $\{\varphi_1\}$, and $\{\varphi_1, \varphi_2\}$ are contexts

Contexts and executions



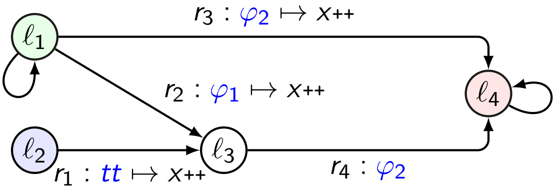
Every execution defines a **monotonically increasing** sequence of contexts: e.g., for a configuration σ with $n = 5, t = 1, f = 1$ and $\kappa_1 = 1$, $\kappa_2 = 3$

Transitions $r_1^1, r_1^1, r_2^1, r_1^1, r_4^1$ applied to σ define the sequence of contexts

$$\emptyset \subset \{\varphi_1\} \subset \{\varphi_1, \varphi_2\}.$$

Or, annotated, $\{\}\ r_1^1\ \{\varphi_1\}\ r_1^1, r_2^1, r_1^1\ \{\varphi_1, \varphi_2\}\ r_4^1\ \{\varphi_1, \varphi_2\}$

Constructing short representatives



$$\varphi_1 \equiv x \geq t+1, \quad \varphi_2 \equiv x \geq n-t$$

$$\{\}\ r_1^1\ \{\varphi_1\}\ r_1^1, r_2^1, r_1^1\ \{\varphi_1, \varphi_2\}\ r_4^1\ \{\varphi_1, \varphi_2\}$$

the transitions with the same context are **sorted**, e.g., if $r_1 \preceq^{lin} r_2 \preceq^{lin} r_4$:

$$\{\}\ r_1^1\ \{\varphi_1\}\ r_1^1, r_1^1, r_2^1\ \{\varphi_1, \varphi_2\}\ r_4^1\ \{\varphi_1, \varphi_2\}$$

and the instances of the same rule are **accelerated**:

$$\{\}\ r_1^1\ \{\varphi_1\}\ r_1^2, r_2^1\ \{\varphi_1, \varphi_2\}\ r_4^1\ \{\varphi_1, \varphi_2\}$$

Formal result on representatives

By applying sorting and acceleration, we prove:

Proposition 9 [CAV'15]

Given a threshold automaton, a configuration σ, and schedule τ applicable to σ, there exists a schedule $\mathsf{rep}[\sigma, \tau]$ with the following properties:

1. $\mathsf{rep}[\sigma, \tau]$ is applicable to σ, and $\mathsf{rep}[\sigma, \tau](\sigma) = \tau(\sigma)$,
2. $|\mathsf{rep}[\sigma, \tau]| \leq 2 \cdot |\mathcal{R}| \cdot (|\Phi| + 1) + |\Phi|$.

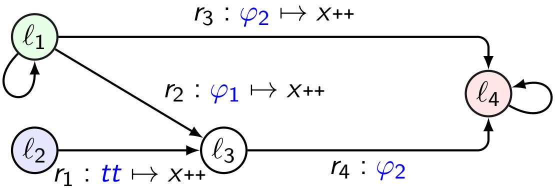
where

- $\mathcal{R}$ is the set of rules (edges of TA),
- Φ is the set of all threshold guards used in $\mathcal{R}$.

Schemas

(the new ingredient)

What can we do with the representatives?



To check reachability, we have to explore all the representatives.

For a monotonically increasing sequence of contexts,

e.g., $\emptyset$, $\{\varphi_1\}$, $\{\varphi_1, \varphi_2\}$

all representatives follow the same pattern:

$\{\}\ r_1\ \{\varphi_1\}\ r_1, r_2\ \{\varphi_1, \varphi_2\}\ r_1, r_2, r_3, r_4\ \{\varphi_1, \varphi_2\}$

Schemas

A **schema** is a sequence of contexts and rule sequences:

$$\mathcal{S} = \{\Omega_0\}\rho_1\{\Omega_1\}\dots\{\Omega_{m-1}\}\rho_m\{\Omega_m\}$$

A schema **generates** paths (including the representatives):

e.g., $\{\}\ r_1\ \{\varphi_1\}\ r_1, r_3, r_4\ \{\varphi_1, \varphi_2\}$

generates

$\{\}\ r_1^2\ \{\varphi_1\}\ r_1^1, r_3^3, r_4^3\ \{\varphi_1, \varphi_2\}$

$\{\}\ r_1^2\ \{\varphi_1\}\ r_1^0, r_3^0, r_4^2\ \{\varphi_1, \varphi_2\}$

How to find a feasible path that reaches a bad state?

Schemas

A **schema** is a sequence of contexts and rule sequences:

$$\mathcal{S} = \{\Omega_0\}\rho_1\{\Omega_1\} \dots \{\Omega_{m-1}\}\rho_m\{\Omega_m\}$$

A schema **generates** paths (including the representatives):

e.g., $\{\}\ r_1\ \{\varphi_1\}\ r_1, r_3, r_4\ \{\varphi_1, \varphi_2\}$

generates

$\{\}\ r_1^2\ \{\varphi_1\}\ r_1^1, r_3^3, r_4^3\ \{\varphi_1, \varphi_2\}$

$\{\}\ r_1^2\ \{\varphi_1\}\ r_1^0, r_3^0, r_4^2\ \{\varphi_1, \varphi_2\}$

How to find a feasible path that reaches a bad state?

Schemas

A **schema** is a sequence of contexts and rule sequences:

$$\mathcal{S} = \{\Omega_0\}\rho_1\{\Omega_1\}\dots\{\Omega_{m-1}\}\rho_m\{\Omega_m\}$$

A schema **generates** paths (including the representatives):

e.g., $\{\}\ r_1\ \{\varphi_1\}\ r_1, r_3, r_4\ \{\varphi_1, \varphi_2\}$

generates

$\{\}\ r_1^2\ \{\varphi_1\}\ r_1^1, r_3^3, r_4^3\ \{\varphi_1, \varphi_2\}$

$\{\}\ r_1^2\ \{\varphi_1\}\ r_1^0, r_3^0, r_4^2\ \{\varphi_1, \varphi_2\}$

How to find a feasible path that reaches a bad state?

Checking feasibility with SMT

It is easy to check with SMT, whether a schema generates a feasible path:

e.g., $\{\}\ r_1\ \{\varphi_1\}\ r_2\ \{\varphi_1, \varphi_2\}\ r_4\ \{\varphi_1, \varphi_2\}$

κ_1	$\kappa_1^0 = n - f$		$\kappa_1^2 = \kappa_1^0 - \delta_2$	
κ_2	$\kappa_2^0 = 0$	$\kappa_2^1 = \kappa_2^0 - \delta_1$		
κ_3	$\kappa_3^0 = 0$	$\kappa_3^1 = \kappa_3^0 + \delta_1$	$\kappa_3^2 = \kappa_3^1 + \delta_2$	$\kappa_3^3 = \kappa_3^2 - \delta_3$
κ_4	$\kappa_4^0 = 0$			$\kappa_4^3 = \kappa_4^0 + \delta_3$
x	$x^0 = 0$	$x^1 = x^0 + \delta_1$	$x^2 = x^2 + \delta_2$	
		$x^1 \geq (t+1) - f$	$x^2 \geq (n-t) - f$	$\kappa_4^3 = n - f$

Complete parameterized reachability checking

Sound and complete algorithm for parameterized reachability in TA:

For each monotonically increasing sequence Ω of contexts:
construct a schema S for Ω
if there is a path π generated by S that reaches a bad state,
then report π as a counterexample

Theorem 1 [CAV'15]

For a threshold automaton, there is a complete schema set of cardinality at most $|\Phi|!$, where the length of each schema does not exceed $(3 \cdot |\Phi| + 2) \cdot |\mathcal{R}|$.

Note:

This result also holds for the guards like *nfaulty* $< f$

Complete parameterized reachability checking

Sound and complete algorithm for parameterized reachability in TA:

For each monotonically increasing sequence Ω of contexts:
construct a schema S for Ω
if there is a path π generated by S that reaches a bad state,
then report π as a counterexample

Theorem 1 [CAV'15]

For a threshold automaton, there is a complete schema set of cardinality at most $|\Phi|!$, where the length of each schema does not exceed $(3 \cdot |\Phi| + 2) \cdot |\mathcal{R}|$.

Note:

This result also holds for the guards like *nfaulty* $< f$

Complete parameterized reachability checking

Sound and complete algorithm for parameterized reachability in TA:

For each monotonically increasing sequence Ω of contexts:
construct a schema S for Ω
if there is a path π generated by S that reaches a bad state,
then report π as a counterexample

Theorem 1 [CAV'15]

For a threshold automaton, there is a complete schema set of cardinality at most $|\Phi|!$, where the length of each schema does not exceed $(3 \cdot |\Phi| + 2) \cdot |\mathcal{R}|$.

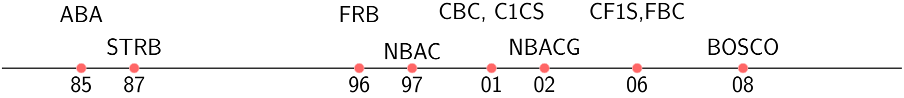
Note:

This result also holds for the guards like $nfaulty < f$

Results

Now we can verify **safety** of the parameterized algorithms:

- Reliable broadcast (FRB, STRB, ABA)
- Non-blocking atomic commit with failure detectors (NBAC, NBACG)
- Condition-based consensus (CBC)
- One-step consensus (CF1S, C1CS, BOSCO)



Liveness?

"...when looking for errors, most of your effort should be devoted to examining the safety part." Leslie Lamport. Specifying Systems (2002)

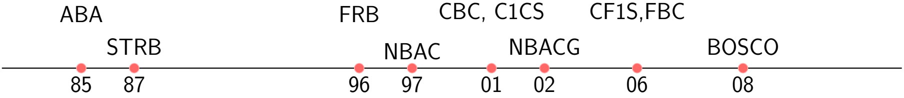
"Liveness is whatever prevents an empty system from being correct."

Orna Kupferman. Beyond Safety Workshop (2004)

Results

Now we can verify **safety** of the parameterized algorithms:

- Reliable broadcast (FRB, STRB, ABA)
- Non-blocking atomic commit with failure detectors (NBAC, NBACG)
- Condition-based consensus (CBC)
- One-step consensus (CF1S, C1CS, BOSCO)



Liveness?

"...when looking for errors, most of your effort should be devoted to examining the safety part." Leslie Lamport. Specifying Systems (2002)

"Liveness is whatever prevents an empty system from being correct."

Orna Kupferman. Beyond Safety Workshop (2004)

Conclusions

Standard model checkers are not tuned to the computational models of fault-tolerant distributed algorithms

Computational primitives in FTDAs are simpler than the standard ones

This and parameterization helped us to develop efficient techniques

check FTDAs used in the cloud:
variations of Paxos, RAFT, etc.?

Conclusions

Standard model checkers are not tuned to the computational models of fault-tolerant distributed algorithms

Computational primitives in FTDAs are simpler than the standard ones

This and parameterization helped us to develop efficient techniques

check FTDAs used in the cloud:
variations of Paxos, RAFT, etc.?

Thank you!

[http://forsyte.at/software/bymc]

SMT and POR beat Counter Abstraction:

Parameterized Model Checking of Threshold-Based Distributed Algorithms.

To appear at CAV'15.