Modeling and Verifying Randomized Fault-Tolerant Distributed Algorithms

Nathalie Bertrand

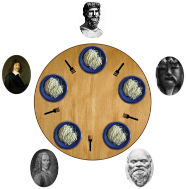
DisCoTec - June 17th 2020

Igor Konnov Marijana Lazić Josef Widder

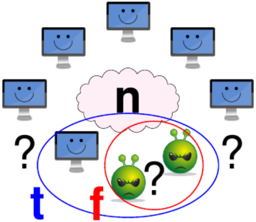


Randomization in distributed computing

- To prevent attacks
- To improve complexity
  - average complexity may only be better than worst-case
- To make impossible things possible!
  - impossibility of symmetric solution to **dining philosophers problem**
    - use randomness to break symmetry [Lehman Rabin'81]
  - impossibility of **consensus** in asynchronous setting as soon as one process can crash [Fischer Lynch Paterson'85]
    - use randomness to rule out non-terminating executions [BenOr'83]



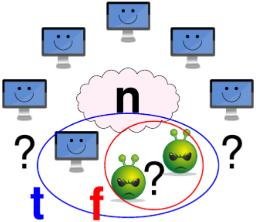
Fault-tolerant distributed algorithms



- n processes communicate by asynchronous message broadcast
- f processes are faulty in the current execution
- t is a known upper bound on f

Resilience condition constrains parameters n, f, t ensuring correctness

Fault-tolerant distributed algorithms



- n processes communicate by asynchronous message broadcast
- f processes are faulty in the current execution
- t is a known upper bound on f

Resilience condition constrains parameters n, f, t ensuring correctness

Ben Or's **randomized** algorithm solves consensus assuming $f \leq t < \frac{n}{5}$

The need for parameterized and automated verification

Need for **parameterized** verification

- correctness should hold **under all parameter valuations** that meet the resilience condition

$$\forall n, t, f \quad f \leq t < \frac{n}{5} \quad \Longrightarrow \quad C(n,t) || \cdots || C(n,t) || F || \cdots || F \models \varphi$$

The need for parameterized and automated verification

Need for **parameterized** verification

- correctness should hold **under all parameter valuations** that meet the resilience condition

$$\forall n, t, f \quad f \leq t < \frac{n}{5} \quad \Longrightarrow \quad C(n,t)||\cdots||C(n,t)||F||\cdots||F \models \varphi$$

Need for **automated** verification

- mostly hand-written proofs in the literature
- non-determinism combined with probabilities

Proofs of correctness for probabilistic distributed systems are extremely slippery

[Lehmann Rabin'81]

Threshold automata to the rescue

for non-randomized fault-tolerant distributed algorithms

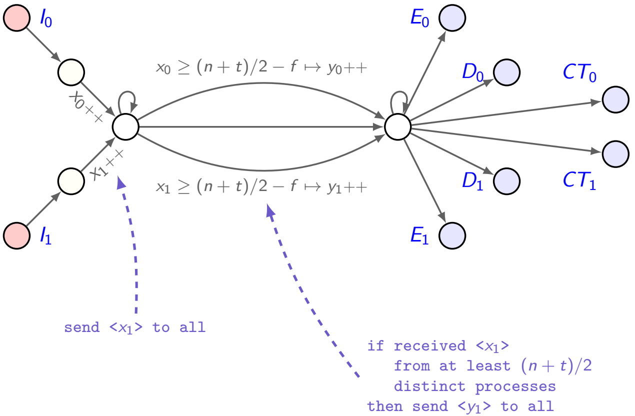
- locations represent algorithm control points
- shared variables count sent messages of each type
- guards as linear constraints on variables and parameters

[Konnov Veith Widder CAV'15, Konnov Lazić Veith Widder POPL'17]

Threshold automata to the rescue

for non-randomized fault-tolerant distributed algorithms

- locations represent algorithm control points
- shared variables count sent messages of each type
- guards as linear constraints on variables and parameters



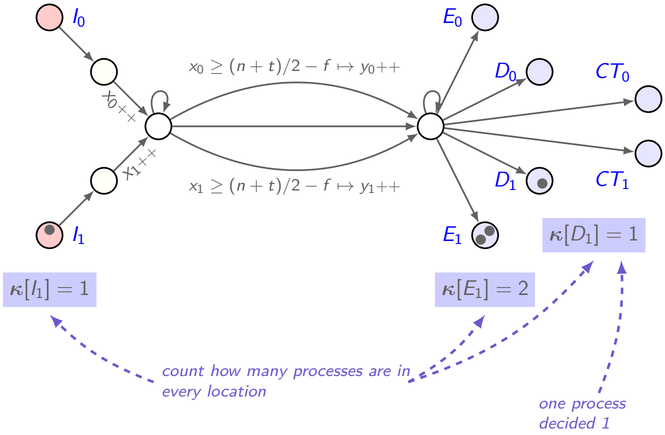
[Konnov Veith Widder CAV'15, Konnov Lazić Veith Widder POPL'17]

Semantics of threshold automata

- infinite counter system
  - finitely many counters: 1 per location of the TA
  - unbounded counter values because of parameters

Semantics of threshold automata

- infinite counter system
  - finitely many counters: 1 per location of the TA
  - unbounded counter values because of parameters



Specifying and verifying correctness

- $\mathsf{ELTL}_{\mathsf{FT}}$: LTL fragment without Next, and with counters
- atomic propositions: whether counter value is 0 or not

Specifying and verifying correctness

- $\mathsf{ELTL}_{\mathsf{FT}}$: LTL fragment without Next, and with counters
- atomic propositions: whether counter value is 0 or not

Agreement: No two correct processes decide differently (safety)

$$\mathbf{F}\ \boldsymbol{\kappa}[D_v] > 0 \quad \rightarrow \quad \mathbf{G}\ \boldsymbol{\kappa}[D_{1-v}] = 0$$

Termination: Eventually all correct processes decide (liveness)

$$\mathbf{F} \bigwedge_{\ell \in \mathcal{L} \setminus \{D_0, D_1\}} \boldsymbol{\kappa}[\ell] = 0$$

Given a threshold automaton TA, a specification φ in $\mathsf{ELTL}_{\mathsf{FT}}$, and a resilience condition RC, one can decide whether for all parameters satisfying RC, Sys(TA) $\models \varphi$

Tool support: ByMC at `forsyte.at/software/bymc/`

[Konnov Veith Widder CAV'15, Konnov Lazić Veith Widder POPL'17]

Outline

How to handle randomization?

Ben Or's randomized algorithm for consensus [Ben Or'83]

```
bool v := input_value({0, 1});
int r := 1 ;
while (true) do
 send (R,r,v) to all;
 wait for n − t messages (R,r,*);
 if received (n + t) / 2 messages (R,r,w)
 then send (P,r,w,D) to all;
 else send (P,r,?) to all;
 wait for n − t messages (P,r,*);
 if received at least t + 1
    messages (P,r,w,D) then {
  v := w;                   /* enough support -> update estimate */
  if received at least (n + t) / 2
   messages (P,r,w,D)
  then decide w;                    /* strong majority -> decide */
 } else v := random(0, 1) ;      /* unclear -> coin toss      */
 r := r + 1 ;
od
```

How to handle randomization?

Ben Or's randomized algorithm for consensus [Ben Or'83]

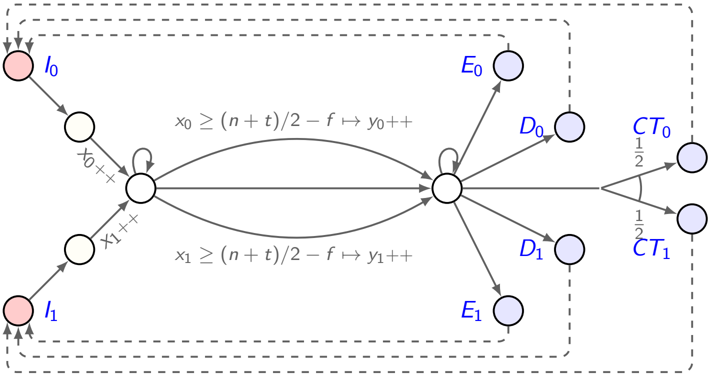
```
bool v := input_value({0, 1});
int r := 1 ;
while (true) do
 send (R,r,v) to all;
 wait for n − t messages (R,r,∗);
 if received (n + t) / 2 messages (R,r,w)
 then send (P,r,w,D) to all;
 else send (P,r,?) to all;
 wait for n − t messages (P,r,∗);
 if received at least t + 1
    messages (P,r,w,D) then {
  v := w;                  /* enough support −> update estimate */
  if received at least (n + t) / 2
   messages (P,r,w,D)
  then decide w;                 /* strong majority −> decide */
 } else v := random(0, 1) ;    /* unclear −> coin toss      */
 r := r + 1 ;
od
```

Modeling challenges

- unboundedly many rounds
- probabilistic choices for local/global coin tosses

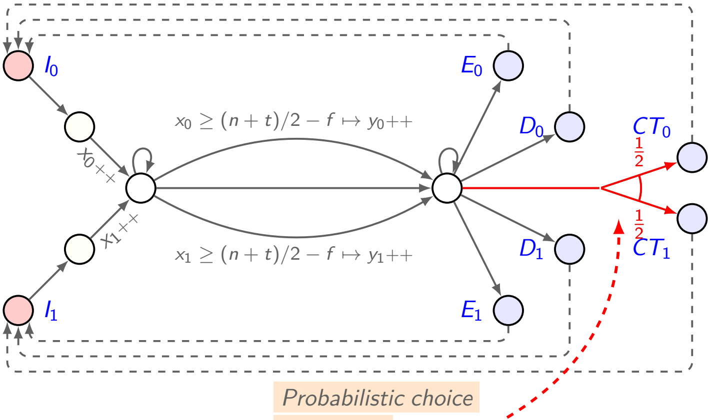
Probabilistic threshold automata

Illustration on Ben Or's algorithm



Probabilistic threshold automata

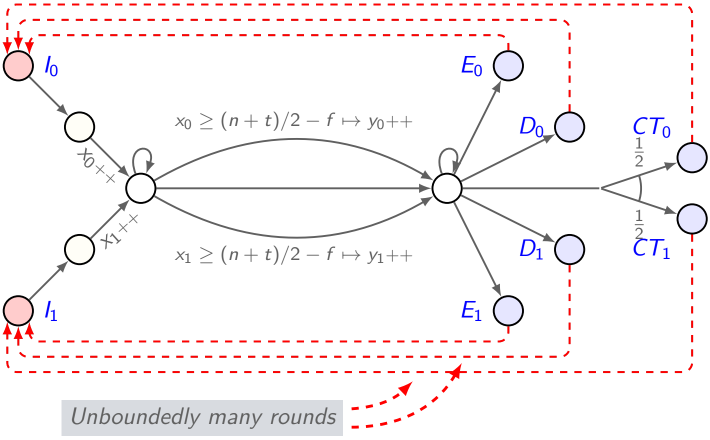
Illustration on Ben Or's algorithm



Probabilistic choice
(coin toss)

Probabilistic threshold automata

Illustration on Ben Or's algorithm



Correctness properties

Agreement: No two correct processes decide differently (safety)

$$(\forall k \in \mathbb{N}_0)\,(\forall k' \in \mathbb{N}_0)\;\; \mathbf{A}\;\Big(\mathbf{F}\;\boldsymbol{\kappa}[D_v, k] > 0 \quad \rightarrow \quad \mathbf{G}\;\boldsymbol{\kappa}[D_{1-v}, k'] = 0\Big)$$

Validity: Any decided value was proposed initially (safety)

$$(\forall k \in \mathbb{N}_0)\;\; \mathbf{A}\;\Big(\mathbf{F}\;\boldsymbol{\kappa}[I_{1-v}, 0] = 0 \quad \rightarrow \quad \mathbf{G}\;\boldsymbol{\kappa}[D_{1-v}, k] = 0\Big)$$

Almost sure termination: under every adversary, with probability 1 every correct process eventually decides (prob. liveness)

$$\mathbb{P}_{\mathtt{a}}\,\Big(\bigvee_{k \in \mathbb{N}_0} \bigvee_{v \in \{0,1\}} \mathbf{G} \bigwedge_{\ell \in \mathcal{L} \setminus \{D_v\}} \boldsymbol{\kappa}[\ell, k] = 0\Big) = 1$$

Correctness properties

Agreement: No two correct processes decide differently (safety)

$$(\forall k \in \mathbb{N}_0)\ (\forall k' \in \mathbb{N}_0)\ \mathbf{A}\ \big(\mathbf{F}\ \boldsymbol{\kappa}[D_v, k] > 0 \quad \rightarrow \quad \mathbf{G}\ \boldsymbol{\kappa}[D_{1-v}, k'] = 0\big)$$

Validity: Any decided value was proposed initially (safety)

$$(\forall k \in \mathbb{N}_0)\ \mathbf{A}\ \big(\mathbf{F}\ \boldsymbol{\kappa}[I_{1-v}, 0] = 0 \quad \rightarrow \quad \mathbf{G}\ \boldsymbol{\kappa}[D_{1-v}, k] = 0\big)$$

Almost sure termination: under every adversary, with probability 1 every correct process eventually decides (prob. liveness)

$$\mathbb{P}_{\mathsf{a}}\Big(\bigvee_{k \in \mathbb{N}_0} \bigvee_{v \in \{0,1\}} \mathbf{G} \bigwedge_{\ell \in \mathcal{L} \setminus \{D_v\}} \boldsymbol{\kappa}[\ell, k] = 0\Big) = 1$$

Verification challenges

- specifications over multiple rounds
- probabilistic guarantees

Outline

Motivations

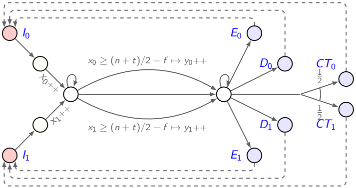
Probabilistic Threshold automata

Proving safety properties

Proving almost-sure termination
- Round-rigid adversaries
- Weak adversaries

Conclusions

Illustration on Ben Or's algorithm



Agreement

$$(\forall k \in \mathbb{N}_0)\ (\forall k' \in \mathbb{N}_0)$$

$$\mathbf{A}\ \big(\mathbf{F}\ \boldsymbol{\kappa}[D_v, k] > 0 \quad \rightarrow \quad \mathbf{G}\ \boldsymbol{\kappa}[D_{1-v}, k'] = 0\big)$$

must hold on **all** executions

$\rightarrow$ probabilistic choices can be transformed into non-determinism

Remaining challenges

- unboundedly many rounds
- specifications over multiple rounds

Solution

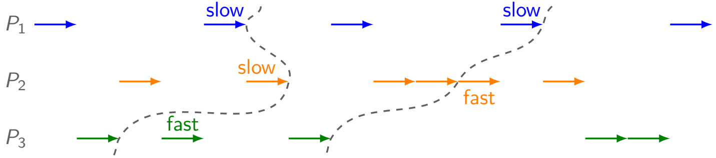
- reduce to one-round threshold automaton
- reduce to one-round specifications

Reduction to single-round TA

Communication-closure hyp.: only messages of current round impact
$\rightarrow$ reordering to analyze rounds in isolation

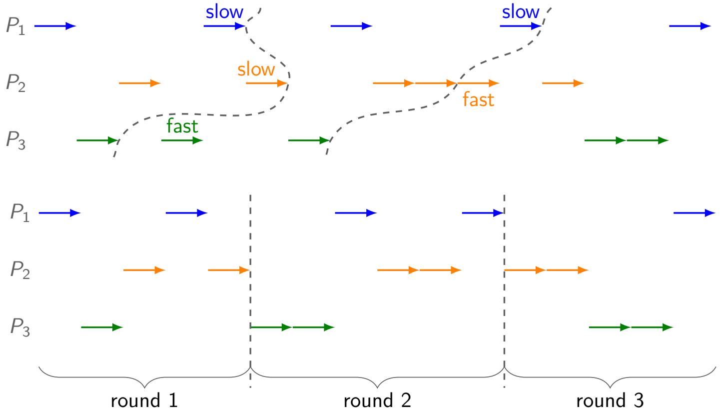
Reduction to single-round TA

Communication-closure hyp.: only messages of current round impact
$\rightarrow$ reordering to analyze rounds in isolation



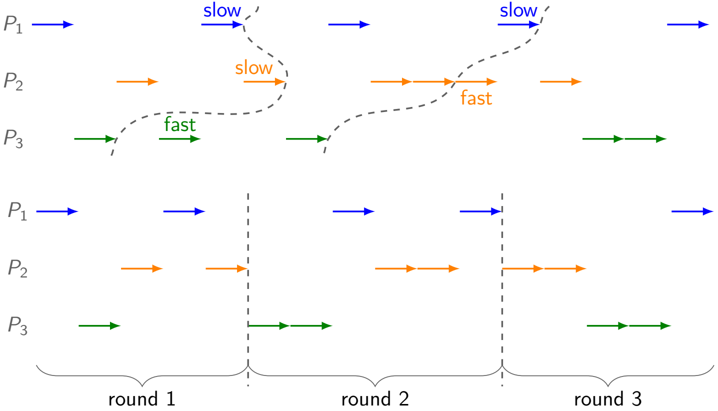
Reduction to single-round TA

Communication-closure hyp.: only messages of current round impact
$\rightarrow$ reordering to analyze rounds in isolation



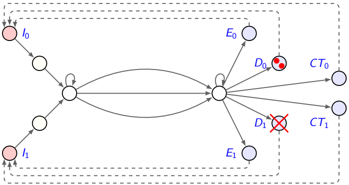
Reduction to single-round TA

Communication-closure hyp.: only messages of current round impact
$\rightarrow$ reordering to analyze rounds in isolation



Reordering preserves validity of $\mathsf{ELTL}_{\mathsf{FT}}$ specifications.

Reduction to single-round specifications

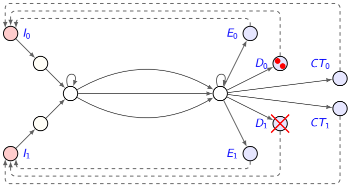


Agreement

if **F** decision v in k

then **G** no decision $1-v$ in k'

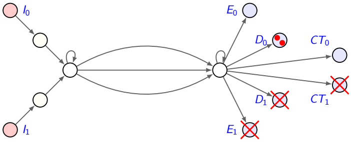
Reduction to single-round specifications



Agreement

if **F** decision v in k

then **G** no decision $1-v$ in k'

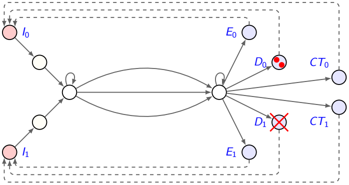


2 sufficient conditions on 1-round TA

if **F** decision v in k

then **G** empty final locs with $1-v$ in k

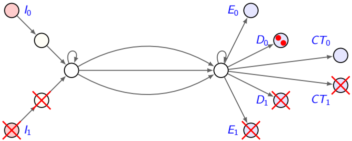
Reduction to single-round specifications



Agreement

if **F** decision v in k

then **G** no decision $1-v$ in k'



2 sufficient conditions on 1-round TA

if **F** decision v in k

then **G** empty final locs with $1-v$ in k

if **G** empty initial with $1-v$ in k

then **G** empty final with $1-v$ in k

Outline

Motivations

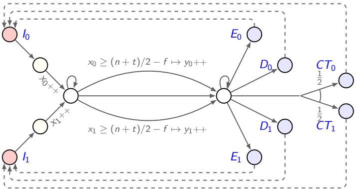
Probabilistic Threshold automata

Proving safety properties

Proving almost-sure termination
- Round-rigid adversaries
- Weak adversaries

Conclusions

Illustration on Ben Or's algorithm



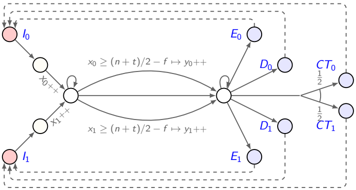
Almost sure termination

$$\forall \mathtt{a}\ \mathbb{P}_{\mathtt{a}}\ \Big(\bigvee_{k \in \mathbb{N}_0} \bigvee_{v \in \{0,1\}} \mathbf{G} \bigwedge_{\ell \in \mathcal{L} \setminus \{D_v\}} \boldsymbol{\kappa}[\ell, k] = 0 \Big) = 1$$

must hold for **all** adversaries, on **almost all** executions

$\rightarrow$ probabilities matter!

Illustration on Ben Or's algorithm



Almost sure termination

$$\forall \mathsf{a}\ \mathbb{P}_{\mathsf{a}}\ \Big(\bigvee_{k\in\mathbb{N}_0}\ \bigvee_{v\in\{0,1\}}\ \mathbf{G} \bigwedge_{\ell\in\mathcal{L}\setminus\{D_v\}} \boldsymbol{\kappa}[\ell,k]=0\Big)=1$$

must hold for **all** adversaries, on **almost all** executions

$\rightarrow$ probabilities matter!

Extra challenge: reordering in the presence of probabilistic branching

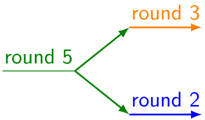
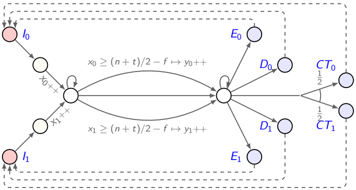


Illustration on Ben Or's algorithm



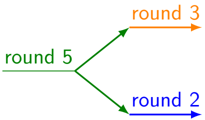
Almost sure termination

$$\forall \mathtt{a}\ \mathbb{P}_{\mathtt{a}}\ \Big(\bigvee_{k\in\mathbb{N}_0}\ \bigvee_{v\in\{0,1\}}\ \mathbf{G} \bigwedge_{\ell\in\mathcal{L}\setminus\{D_v\}} \boldsymbol{\kappa}[\ell,k]=0\Big)=1$$

must hold for **all** adversaries, on **almost all** executions

$\rightarrow$ probabilities matter!

Extra challenge: reordering in the presence of probabilistic branching



Solution

- restrict to **round-rigid** or **weak** adversaries

Round-rigid adversaries

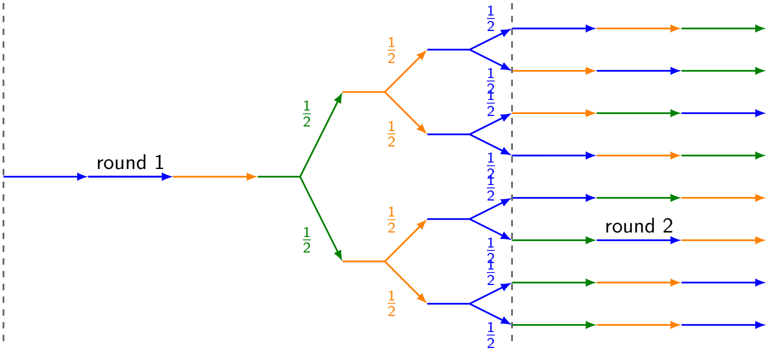
Round-rigid adversary

- respects round order
- schedules probabilistic choices at the end of rounds

Round-rigid adversaries

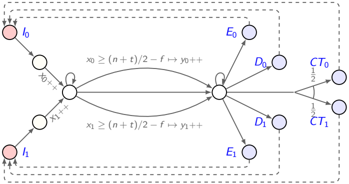
Round-rigid adversary

- respects round order
- schedules probabilistic choices at the end of rounds



Reduction to single-round specifications

From almost-sure to being lucky



Almost-sure termination

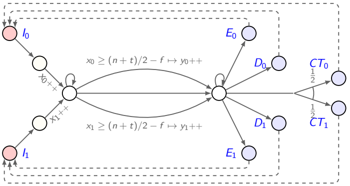
for every **round-rigid** adv. almost surely

F all decide in k

$\forall \mathsf{a},\ \mathbb{P}_{\mathsf{a}}\big(\bigvee_k \bigvee_v \mathbf{G} \bigwedge_{\ell \neq D_v} \boldsymbol{\kappa}[\ell, k] = 0\big) = 1$

Reduction to single-round specifications

From almost-sure to being lucky

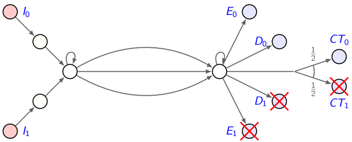


Almost-sure termination

for every **round-rigid** adv. almost surely

F all decide in k

$\forall \mathsf{a},\ \mathbb{P}_{\mathsf{a}}\big(\bigvee_k \bigvee_v \mathbf{G} \bigwedge_{\ell \neq D_v} \boldsymbol{\kappa}[\ell, k] = 0\big) = 1$



2 sufficient conditions on 1-round PTA

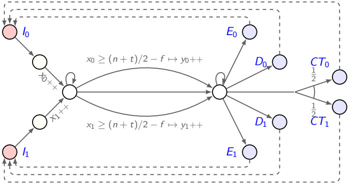
One may be lucky

for every adv. with bounded probability

G empty final with $1-v$ in k

Reduction to single-round specifications

From almost-sure to being lucky

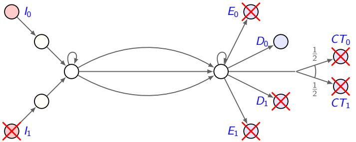


Almost-sure termination

for every **round-rigid** adv. almost surely

F all decide in k

$\forall \mathtt{a},\ \mathbb{P}_{\mathtt{a}}\left(\bigvee_k \bigvee_v \mathbf{G} \bigwedge_{\ell \neq D_v} \boldsymbol{\kappa}[\ell, k] = 0 \right) = 1$



2 sufficient conditions on 1-round PTA

One may be lucky

for every adv. with bounded probability

G empty final with $1-v$ in k

Luckiness implies termination in next round

if **G** empty initial with $1-v$ in k

then **F** all decide v in k

Reduction to single-round specifications

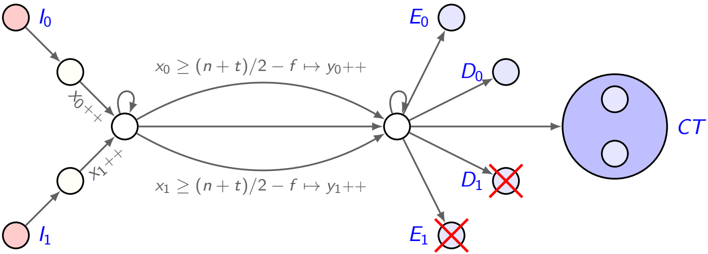
Checking that one may be lucky

for every adv. with bounded probability **G** empty final with $1-v$ in k

- the possibility of being lucky implies a bounded probability (for fixed parameter values)

  E G empty final with $1-v$ in k $\implies \mathbb{P}_{\mathsf{a}}($**G** empty final with $1-v$ in k $) \geq p > 0$

- probabilistic choices can be abstracted to obtain a TA



Experimental evaluation

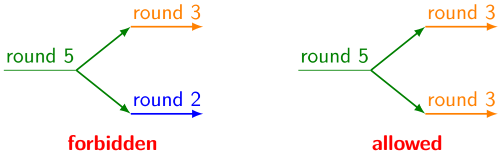
- 6 randomized consensus algorithms
- several one-round safety and liveness properties for each
- tool support: `forsyte.at/software/bymc/`

Algorithm	Verif time per property
- Ben-Or's Byzantine random. consensus	≤ 1 sec
- Ben-Or's crash random. consensus	≤ 1 sec
- Ben-Or's clean crash random. consensus	≤ 1 sec
- Bracha's randomized consensus	≤ 1 sec
- Raynal's k-set agreement	3–40 sec
- Song's and van Renesse's BOSCO	3 hours on a cluster

Weak adversaries

Weak adversary

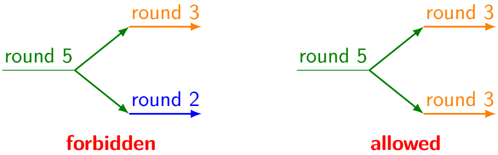
- does not see outcome of random choices
- sees sender and type of messages, not contents
- tags messages with IDs "deliver message 42 to P1"



Weak adversaries

Weak adversary

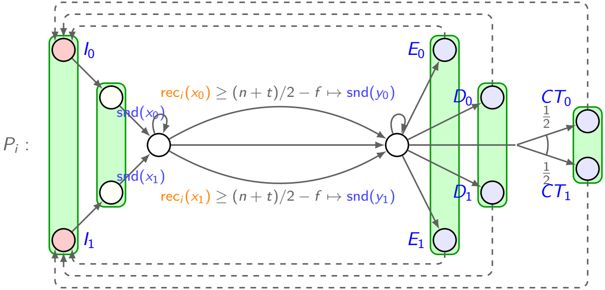
- does not see outcome of random choices
- sees sender and type of messages, not contents
- tags messages with IDs "deliver message 42 to P1"



Need for refined model for probabilistic threshold automaton with message IDs, process IDs, etc.

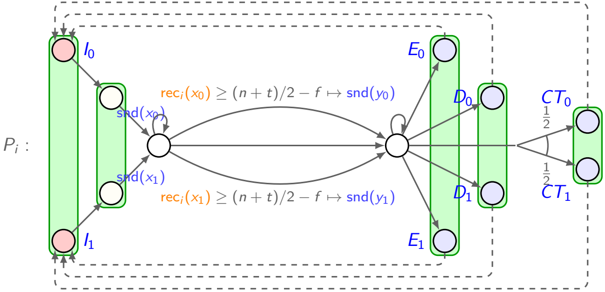
Probabilistic threshold automata with IDs

Illustration on Ben Or's algorithm



Probabilistic threshold automata with IDs

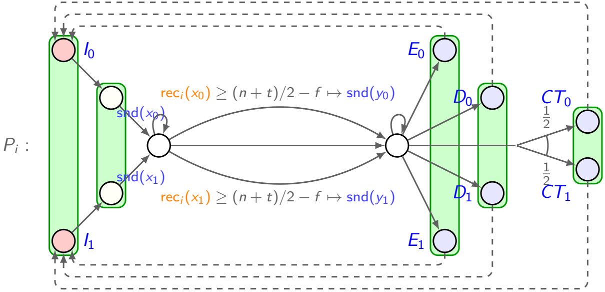
Illustration on Ben Or's algorithm



- local variables count received messages of each type
- global set of sent messages
- equivalence on locations that **weak** adversaries do not distinguish

Probabilistic threshold automata with IDs

Illustration on Ben Or's algorithm



- local variables count received messages of each type
- global set of sent messages
- equivalence on locations that **weak** adversaries do not distinguish

Specifications still in $\mathsf{ELTL}_{\mathsf{FT}}$
atomic propositions: whether some/no process is in ℓ at round k

Reduction to round-rigid adversaries

For every **weak** adversary a, there is a **round-rigid weak** adversary a′ such that for every specification φ in $\mathsf{ELTL_{FT}}$, $\mathbb{P}_{\mathsf{a}}(\varphi) = \mathbb{P}_{\mathsf{a'}}(\varphi)$.

Reduction to round-rigid adversaries

For every **weak** adversary a, there is a **round-rigid weak** adversary a' such that for every specification φ in $\mathsf{ELTL_{FT}}$, $\mathbb{P}_{\mathsf{a}}(\varphi) = \mathbb{P}_{\mathsf{a}'}(\varphi)$.

Key ideas

- transform into a communication-closed adversary
  by postponing delivery of messages from future rounds

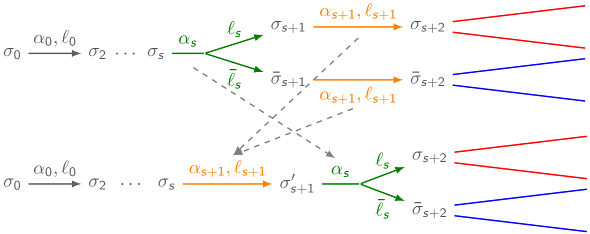
Reduction to round-rigid adversaries

For every **weak** adversary a, there is a **round-rigid weak** adversary a′ such that for every specification φ in $\mathsf{ELTL}_{\mathsf{FT}}$, $\mathbb{P}_{\mathsf{a}}(\varphi) = \mathbb{P}_{\mathsf{a'}}(\varphi)$.

Key ideas

- transform into a communication-closed adversary by postponing delivery of messages from future rounds

- further transform into a round-rigid adversary by re-ordering swapped transitions



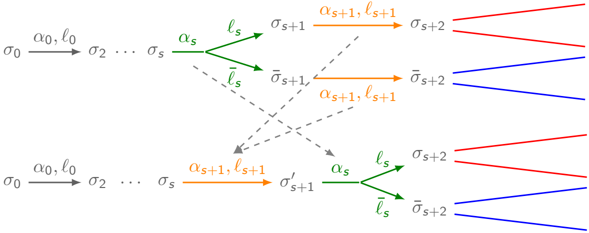
Reduction to round-rigid adversaries

For every **weak** adversary a, there is a **round-rigid weak** adversary a$'$ such that for every specification φ in $\mathsf{ELTL_{FT}}$, $\mathbb{P}_{\mathsf{a}}(\varphi) = \mathbb{P}_{\mathsf{a'}}(\varphi)$.

Key ideas

- transform into a communication-closed adversary by postponing delivery of messages from future rounds

- further transform into a round-rigid adversary by re-ordering swapped transitions

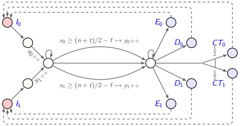


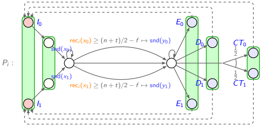
establish correspondence between models with and without IDs

Outline

Contributions

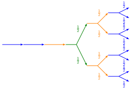
- Modeling of randomized fault-tolerant distributed algorithms



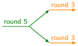


probabilistic threshold automata (PTA) probabilistic threshold automata with IDs (PTA-ID)

- Efficient verification techniques for PTA to prove
    - non-probabilistic specs
    - prob. specs under **round-rigid** adversaries



- Experimental validation on randomized consensus algorithms
- Verification framework for PTA-ID to prove
    - non-prob. and prob. specs under **weak** adversaries



On-going and future work

- Formalisation of correspondence between PTA and PTA-ID

- Structural conditions to enable reordering for strong(er) adversaries
- Quantitative verification techniques for performance evaluation

  average number of rounds before termination

- Models and verification techniques for other classes of randomized distributed algorithms

  global coin tosses

  randomized adversary

On-going and future work

- Formalisation of correspondence between PTA and PTA-ID

- Structural conditions to enable reordering for strong(er) adversaries
- Quantitative verification techniques for performance evaluation

  average number of rounds before termination

- Models and verification techniques for other classes of randomized distributed algorithms

  global coin tosses

  randomized adversary

Nirringrazzjak