

Verification of Randomized Consensus Algorithms under Round-Rigid Adversaries

Nathalie Bertrand, Igor Konnov, Marijana Lazic, Josef Widder

► **To cite this version:**

Nathalie Bertrand, Igor Konnov, Marijana Lazic, Josef Widder. Verification of Randomized Consensus Algorithms under Round-Rigid Adversaries. CONCUR 2019 - 30th International Conference on Concurrency Theory, Aug 2019, Amsterdam, Netherlands. pp.1-16. hal-02191348

HAL Id: hal-02191348

<https://hal.inria.fr/hal-02191348>

Submitted on 23 Jul 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Verification of Randomized Consensus Algorithms under Round-Rigid Adversaries

Nathalie Bertrand 

Univ. Rennes, Inria, CNRS, IRISA, Rennes, France
nathalie.bertrand@inria.fr

Igor Konnov 

University of Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
igor.konnov@inria.fr

Marijana Lazić 

TU München, 85748 Garching b. München, Germany
lazic@in.tum.de

Josef Widder 

TU Wien, 1040 Vienna, Austria, and Interchain Foundation, 6340 Baar, Switzerland
widder@forsyte.at

Abstract

Randomized fault-tolerant distributed algorithms pose a number of challenges for automated verification: (i) parameterization in the number of processes and faults, (ii) randomized choices and probabilistic properties, and (iii) an unbounded number of asynchronous rounds. This combination makes verification hard. Challenge (i) was recently addressed in the framework of threshold automata.

We extend threshold automata to model randomized consensus algorithms that perform an unbounded number of asynchronous rounds. For non-probabilistic properties, we show that it is necessary and sufficient to verify these properties under round-rigid schedules, that is, schedules where processes enter round r only after all processes finished round $r - 1$. For almost-sure termination, we analyze these algorithms under round-rigid adversaries, that is, fair adversaries that only generate round-rigid schedules. This allows us to do compositional and inductive reasoning that reduces verification of the asynchronous multi-round algorithms to model checking of a one-round threshold automaton. We apply this framework and automatically verify the following classic algorithms: Ben-Or's and Bracha's seminal consensus algorithms for crashes and Byzantine faults, 2-set agreement for crash faults, and RS-Bosco for the Byzantine case.

2012 ACM Subject Classification Software and its engineering → Software verification; Software and its engineering → Software fault tolerance; Theory of computation → Logic and verification

Keywords and phrases threshold automata – counter systems – parameterized verification – randomized distributed algorithms – Byzantine faults

Related Version Full version with proofs is available at <https://hal.inria.fr/hal-01925533>.

Funding Supported by the Austrian Science Fund (FWF) via the National Research Network RiSE (S11403, S11405), project PRAVDA (P27722), and Doctoral College LogiCS (W1255-N23); by the Vienna Science and Technology Fund (WWTF) via project APALACHE (ICT15-103); and by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 787367 (PaVeS). Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER, and others, see www.grid5000.fr.

1 Introduction

Fault-tolerant distributed algorithms like Paxos and Blockchain recently receive much attention. Still, these systems are out of reach with current automated verification techniques.



© N. Bertrand, I. Konnov, M. Lazić, J. Widder;

licensed under Creative Commons License CC-BY

30th International Conference on Concurrency Theory (CONCUR 2019).

Editors: Wan Fokkink and Rob van Glabbeek; Article No. 29; pp. 29:1–29:16

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

```

1  bool v := input_value({0, 1});
2  int r := 1;
3  while (true) do
4    send (R,r,v) to all;
5    wait for n - t messages (R,r,*);
6    if received (n + t) / 2 messages (R,r,w)
7    then send (P,r,w,D) to all;
8    else send (P,r,?) to all;
9    wait for n - t messages (P,r,*);
10   if received at least t + 1
11     messages (P,r,w,D) then {
12     v := w;
13     if received at least (n + t) / 2
14       messages (P,r,w,D)
15     then decide w;
16   } else v := random({0, 1});
17   r := r + 1;
18   od

```

■ **Figure 1** Pseudo code of Ben-Or’s algorithm for Byzantine faults

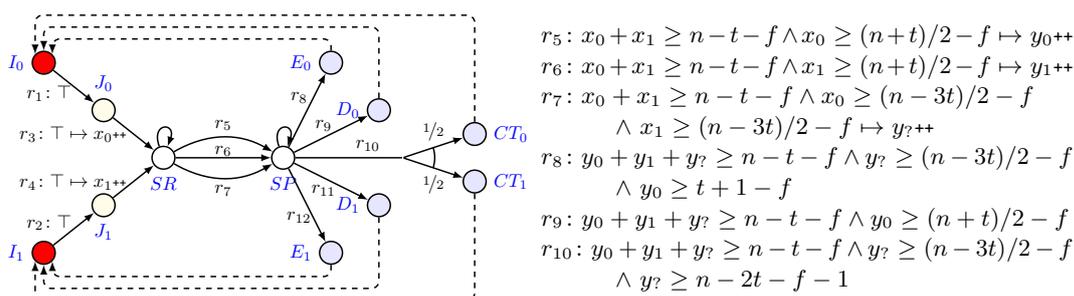
45 One problem comes from the scale: these systems should be verified for a very large (ideally even an unbounded) number of participants. In addition, many systems (including
46 Blockchain), provide probabilistic guarantees. To check their correctness, one has to reason
47 about randomized distributed algorithms in the parameterized setting.
48

49 In this paper, we make first steps towards parameterized verification of fault-tolerant
50 randomized distributed algorithms. We consider consensus algorithms that follow the ideas
51 of Ben-Or [3]. Interestingly, these algorithms were analyzed in [17, 15] where probabilistic
52 reasoning was done using the probabilistic model checker PRISM [16] for systems of 10-
53 20 processes, while only safety was verified in the parameterized setting using Cadence
54 SMV. From a different perspective, these algorithms extend asynchronous threshold-guarded
55 distributed algorithms from [12, 11] with two features (i) a random choice (coin toss), and
56 (ii) repeated executions of the same algorithm until it converges (with probability 1).

57 A prominent example is Ben-Or’s fault-tolerant consensus algorithm [3] given in Figure 1.
58 It circumvents the impossibility of asynchronous consensus [9] by relaxing the termination
59 requirement to almost-sure termination, *i.e.*, termination with probability 1. Here processes
60 execute an infinite sequence of asynchronous loop iterations, which are called rounds r . Each
61 round consists of two stages where they first exchange messages tagged R , wait until the
62 number of received messages reaches a certain threshold (given as expression over parameters
63 in line 5) and then exchange messages tagged P . In the code, n is the number of processes,
64 among which at most t are Byzantine faulty (which may send conflicting information). The
65 correctness of the algorithm should be verified for all values of the parameters n and t that
66 meet a so-called resilience condition, *e.g.*, $n > 3t$. Carefully chosen thresholds ($n - t$, $(n + t)/2$
67 and $t + 1$) on the number of received messages of a given type, ensure *agreement*, *i.e.*, that
68 two correct processes never decide on different values. At the end of a round, if there is no
69 “strong majority” for a value, *i.e.*, less than $(n + t)/2$ messages were received (cf. line 13), a
70 process picks a new value randomly in line 16.

71 While these non-trivial threshold expressions can be dealt with using the methods in [11],
72 several challenges remain. The technique in [11] can be used to verify one iteration of the
73 round from Figure 1 only. However, consensus algorithms should prevent that there are no two
74 rounds r and r' such that a process decides 0 in r and another decides 1 in r' . This calls for a
75 compositional approach that allows one to compose verification results for individual rounds.
76 A challenge in the composition is that distributed algorithms implement “asynchronous
77 rounds”, *i.e.*, during a run processes may be in different rounds at the same time.

78 In addition, the combination of distributed aspects and probabilities makes reasoning
79 difficult. Quoting Lehmann and Rabin [18], “proofs of correctness for probabilistic dis-
80 tributed systems are extremely slippery”. This advocates the development of automated



■ **Figure 2** Ben-Or's algorithm as PTA with resilience condition $n > 3t \wedge t > 0 \wedge t \geq f \geq 0$.

81 verification techniques for probabilistic properties of randomized distributed algorithms in
 82 the parameterized setting.

83 **Contributions.** We extend the threshold automata framework from [11] to round-based
 84 algorithms with coin toss transitions. For the new framework we achieve the following:

- 85 1. For safety verification we introduce a method for compositional round-based reasoning.
 86 This allows us to invoke a reduction similar to the one in [8, 6, 7]. We highlight necessary
 87 fairness conditions on individual rounds. This provides us with specifications to be
 88 checked on a one-round automaton.
- 89 2. We reduce probabilistic liveness verification to proving termination with positive prob-
 90 ability within a fixed number of rounds. To do so, we restrict ourselves to round-rigid
 91 adversaries, that is, adversaries that respect the round ordering. In contrast to existing
 92 work that proves almost-sure termination for fixed number of participants, these are the
 93 first parameterized model checking results for probabilistic properties.
- 94 3. We check the specifications that emerge from points 1. and 2. and thus verify challenging
 95 benchmarks in the parameterized setting. We verify Ben-Or's [3] and Bracha's [5] classic
 96 algorithms, and more recent algorithms such as 2-set agreement [21], and RS-Bosco [23].

97 2 Overview

98 We introduce probabilistic threshold automata to model randomized threshold-based algo-
 99 rithms. An example of such an automaton is given in Figure 2. Nodes represent local states
 100 (or locations) of processes, which move along the labeled edges or forks. Edges and forks
 101 are called rules. Labels have the form $\varphi \mapsto u$, meaning that a process can move along the
 102 edge only if φ evaluates to true, and this is followed by the update u of shared variables.
 103 Additionally, each tine of a fork is labeled with a number in the $[0, 1]$ interval, representing
 104 the probability of a process moving along the fork to end up at the target location of the tine.
 105 If we ignore the dashed arrows in Figure 2, a threshold automaton captures the behavior of
 106 a process in one round, that is, a loop iteration in Figure 1.

107 The code in Figure 1 refers to numbers of received messages and, as is typical for
 108 distributed algorithms, their relation to sent messages (that is the semantics of send and
 109 receive) is not explicit in the pseudo code. To formalize the behavior, the encoding in the
 110 threshold automaton directly refers to the numbers of sent messages, and they are encoded
 111 in the shared variables x_i and y_i . The algorithm is parameterized: n is the number of
 112 processes, t is the assumed number of faults and f is the actual number of faults. It should be
 113 demonstrated to work under the resilience condition $n > 3t \wedge t \geq f \wedge t > 0$. For instance, the

114 locations J_0 and J_1 capture that a loop is entered with v being 0 and 1, respectively. Sending
 115 an $(R, r, 0)$ and $(R, r, 1)$ message is captured by the increments on the shared variables x_0
 116 and x_1 in the rules r_3 and r_4 , respectively; e.g., a process that is in location J_0 uses rule
 117 r_3 to go to location SR (“sent R message”), and increments x_0 in doing so. Waiting for R
 118 and P messages in the lines 5 and 9, is captured by looping in the locations SR and SP .
 119 In line 7 a process sends, e.g., a $(P, r, 0, D)$ message if it has *received* $n - t$ messages out
 120 of which $(n + t)/2$ are $(R, r, 0)$ messages. This is captured in the guard of rule r_5 where
 121 $x_0 + x_1 \geq n - t - f$ checks the number of received messages in total, and $x_0 \geq (n + t)/2 - f$
 122 checks for the specific messages containing 0. The “ $-f$ ” term models that in the message
 123 passing semantics underlying Figure 1, f messages from Byzantine faults may be received *in*
 124 *addition* to the messages sent by correct processes (modeled by shared variables in Figure 2).
 125 The branching at the end of the loop from lines 10 to 18 is captured by the rules outgoing of
 126 SP . In particular rule r_{10} captures the coin toss in line 16. The non-determinism due to
 127 faults and asynchrony is captured by multiple rules being enabled in the same configuration.

128 Liveness properties of distributed algorithms typically require fairness constraints, e.g.,
 129 every message sent by a correct process to a correct process is eventually received. For
 130 instance, this implies in Figure 1 that if $n - t$ correct processes have sent messages of the
 131 form $(R, 1, *)$ and $(n + t)/2$ correct processes have sent messages of the form $(R, 1, 0)$ then
 132 every correct process should eventually execute line 7, and proceed to line 9. We capture
 133 this by the following fairness constraint: if $x_0 + x_1 \geq n - t \wedge x_0 \geq (n + t)/2$ —that is, rule
 134 r_5 is enabled without the help of the f faulty processes but by “correct processes alone”—
 135 then the source location of rule r_5 , namely SR should eventually be evacuated, that is, its
 136 corresponding counter should eventually be 0.

137 The dashed edges, called round switch rules, encode how a process, after finishing a round,
 138 starts the next one. The round number r serves as the loop iterator in Figure 1, and in each
 139 iteration, processes send messages that carry r . To capture this, our semantics will introduce
 140 fresh shared variables initialized with 0 for each round r . Because there are infinitely many
 141 rounds, this means a priori we have infinitely many variables.

142 As parameterized verification of threshold automata is in general undecidable [14], we
 143 consider the so-called “canonic” restrictions here, *i.e.*, only increments on shared variables,
 144 and no increments of the same variable within loops. These restrictions still allow us to
 145 model many threshold-based fault-tolerant distributed algorithms [11]. As a result, threshold
 146 automata without probabilistic forks and round switching rules can be automatically checked
 147 for safety and liveness [11]. Adding forks and round switches is required to adequately
 148 model randomized distributed algorithms. Here we will use a convenient restriction that
 149 requires that coin-toss transitions only appear at the end of a round, e.g., line 16 of Figure 1.
 150 Intuitively, as discussed in Section 1, a coin-toss is only necessary if there is no strong
 151 majority. Thus, all our benchmarks have this feature, and we exploit it in Section 7.

152 In order to overcome the issue of infinitely many rounds, we prove in Section 6 that we
 153 can verify probabilistic threshold automata by analyzing a one-round automaton that fits
 154 in the framework of [11]. We prove that we can reorder transitions of any fair execution
 155 such that their round numbers are in an increasing order. The obtained ordered execution is
 156 stutter equivalent with the original one, and thus, they satisfy the same LTL_X properties over
 157 the atomic propositions describing only one round. In other words, our targeted concurrent
 158 systems can be transformed to a sequential composition of one-round systems.

159 The main problem with isolating a one-round system is that consensus specifications
 160 often talk about at least two different rounds. In this case we need to use round invariants
 161 that imply the specifications. For example, if we want to verify agreement, we have to check

162 whether two processes decide different values, possibly in different rounds. We do this in two
 163 steps: (i) we check the round invariant that no process changes its decision from round to
 164 round, and (ii) we check that within a round no two processes disagree.

165 Finally, verifying almost-sure termination under round-rigid adversaries calls for distinct
 166 arguments. Our methodology follows the lines of the manual proof of Ben Or’s consensus
 167 algorithm by Aguilera and Toueg [1]. However, our arguments are not specific to Ben
 168 Or’s algorithm, and we apply it to other randomized distributed algorithms (see Section 8).
 169 Compared to their paper-and-pencil proof, the threshold automata framework required us to
 170 provide a more formal setting and a more informative proof, also pinpointing the needed
 171 hypothesis. The crucial parts of our proof are automatically checked by the model checker
 172 ByMC [13]. Hence the established correctness stands on less slippery ground, which addresses
 173 the mentioned concerns of Lehmann and Rabin.

174 **3 The Probabilistic Threshold Automata Framework**

175 A *probabilistic threshold automaton* PTA is a tuple $(\mathcal{L}, \mathcal{V}, \mathcal{R}, RC)$, where

- 176 ■ \mathcal{L} is a finite set of locations, that contains the following disjoint subsets: *initial locations*
 177 \mathcal{I} , *final locations* \mathcal{F} , and *border locations* \mathcal{B} , with $|\mathcal{B}| = |\mathcal{I}|$;
- 178 ■ \mathcal{V} is a set of variables. It is partitioned in two sets: Π contains *parameter variables*, and
 179 Γ contains *shared variables*;
- 180 ■ \mathcal{R} is a finite set of *rules*; and
- 181 ■ RC , the *resilience condition*, is a formula in linear integer arithmetic over parameter
 182 variables.

183 In the following we introduce rules in detail, and give syntactic restrictions on locations.
 184 The resilience condition RC only appears in the definition of the semantics in Section 3.1.

185 A rule r is a tuple $(from, \delta_{to}, \varphi, \vec{u})$ where $from \in \mathcal{L}$ is the *source* location, $\delta_{to} \in \text{Dist}(\mathcal{L})$ is
 186 a probability distribution over the *destination* locations, $\vec{u} \in \mathbb{N}_0^{|\Pi|}$ is the *update vector*, and φ
 187 is a guard, *i.e.*, a conjunction of expressions of the form $b \cdot x \geq \bar{a} \cdot \mathbf{p}^\top + a_0$ or $b \cdot x < \bar{a} \cdot \mathbf{p}^\top + a_0$,
 188 where $x \in \Gamma$ is a shared variable, $\bar{a} \in \mathbb{Z}^{|\Pi|}$ is a vector of integers, $a_0, b \in \mathbb{Z}$, and \mathbf{p} is the
 189 vector of all parameters. If $r.\delta_{to}$ is a Dirac distribution, *i.e.*, there exists $\ell \in \mathcal{L}$ such that
 190 $r.\delta_{to}(\ell) = 1$, we call r a *Dirac rule*, and write it as $(from, \ell, \varphi, \vec{u})$. Destination locations of
 191 non-Dirac rules are in \mathcal{F} (coin-toss transitions only happen at the end of a round).

192 Probabilistic threshold automata model algorithms with successive identical rounds.
 193 Informally, a round happens between border locations and final locations. Then round switch
 194 rules let processes move from final locations of a given round to border locations of the next
 195 round. From each border location there is exactly one Dirac rule to an initial location, and
 196 it has a form $(\ell, \ell', \text{true}, \vec{0})$ where $\ell \in \mathcal{B}$ and $\ell' \in \mathcal{I}$. As $|\mathcal{B}| = |\mathcal{I}|$, one can think of border
 197 locations as copies of initial locations. It remains to model from which final locations to
 198 which border location (that is, initial for the next round) processes move. This is done by
 199 *round switch rules*. They can be described as Dirac rules $(\ell, \ell', \text{true}, \vec{0})$ with $\ell \in \mathcal{F}$ and
 200 $\ell' \in \mathcal{B}$. The set of round switch rules is denoted by $\mathcal{S} \subseteq \mathcal{R}$.

201 A location is in \mathcal{B} if and only if all the incoming edges are in \mathcal{S} . Similarly, a location is
 202 in \mathcal{F} if and only if there is only one outgoing edge and it is in \mathcal{S} .

203 Figure 2 depicts a PTA with border locations $\mathcal{B} = \{I_0, I_1\}$, initial locations $\mathcal{I} = \{J_0, J_1\}$,
 204 and final locations $\mathcal{F} = \{E_0, E_1, D_0, D_1, CT_0, CT_1\}$. The only rule that is not Dirac rule
 205 is r_{10} , and round switch rules are represented by dashed arrows.

206 3.1 Probabilistic Counter Systems

207 A resilience condition RC defines the set of *admissible parameters* $\mathbf{P}_{RC} = \{\mathbf{p} \in \mathbb{N}_0^{|\Pi|} : \mathbf{p} \models$
 208 $RC\}$. We introduce a function $N : \mathbf{P}_{RC} \rightarrow \mathbb{N}_0$ that maps a vector of admissible parameters
 209 to a number of modeled processes in the system. For instance, for the automaton in Figure 2,
 210 N is the function $(n, t, f) \mapsto n - f$, as we model only the $n - f$ correct processes explicitly,
 211 while the effect of faulty processes is captured in non-deterministic choices between different
 212 guards as discussed in Section 2. Given a PTA and a function N , we define the semantics,
 213 called *probabilistic counter system* $\text{Sys}(\text{PTA})$, to be the infinite-state MDP $(\Sigma, I, \text{Act}, \Delta)$,
 214 where Σ is the set of configurations for PTA among which $I \subseteq \Sigma$ are initial, the set of actions
 215 is $\text{Act} = \mathcal{R} \times \mathbb{N}_0$ and $\Delta : \Sigma \times \text{Act} \rightarrow \text{Dist}(\Sigma)$ is the probabilistic transition function.

216 **Configurations.** In a configuration $\sigma = (\vec{\kappa}, \vec{g}, \mathbf{p})$, a function $\sigma.\vec{\kappa} : \mathcal{L} \times \mathbb{N}_0 \rightarrow \mathbb{N}_0$ defines values
 217 of location counters per round, a function $\sigma.\vec{g} : \Gamma \times \mathbb{N}_0 \rightarrow \mathbb{N}_0$ defines shared variable values per
 218 round, and a vector $\sigma.\mathbf{p} \in \mathbb{N}_0^{|\Pi|}$ defines parameter values. We denote the vector $(\vec{g}[x, k])_{x \in \Gamma}$
 219 of shared variables in a round k by $\vec{g}[k]$, and by $\vec{\kappa}[k]$ we denote the vector $(\vec{\kappa}[\ell, k])_{\ell \in \mathcal{L}}$
 220 of local state counters in a round k .

221 A configuration $\sigma = (\vec{\kappa}, \vec{g}, \mathbf{p})$ is *initial* if all processes are in initial states of round 0, and
 222 all global variables evaluate to 0, that is, if for every $x \in \Gamma$ and $k \in \mathbb{N}_0$ we have $\sigma.\vec{g}[x, k] = 0$,
 223 if $\sum_{\ell \in \mathcal{B}} \sigma.\vec{\kappa}[\ell, 0] = N(\mathbf{p})$, and finally if $\sigma.\vec{\kappa}[\ell, k] = 0$, for every $(\ell, k) \in (\mathcal{L} \setminus \mathcal{B}) \times \{0\} \cup \mathcal{L} \times \mathbb{N}$.

224 A threshold guard evaluates to true in a configuration σ for a round k , written $\sigma, k \models \varphi$,
 225 if for all its conjuncts $b \cdot x \geq \bar{a} \cdot \mathbf{p}^\top + a_0$, it holds that $b \cdot \sigma.\vec{g}[x, k] \geq \bar{a} \cdot (\sigma.\mathbf{p}^\top) + a_0$ (and
 226 similarly for conjuncts of the other form, *i.e.*, $b \cdot x < \bar{a} \cdot \mathbf{p}^\top + a_0$).

227 **Actions.** An action $\alpha = (r, k) \in \text{Act}$ stands for the execution of a rule r in round k (by
 228 a single process). We write $\alpha.\text{from}$ for $r.\text{from}$, $\alpha.\delta_{to}$ for $r.\delta_{to}$, etc. An action $\alpha = (r, k)$ is
 229 *unlocked* in a configuration σ , if its guard evaluates to true in its round, that is $\sigma, k \models r.\varphi$.
 230 An action $\alpha = (r, k)$ is *applicable* to a configuration σ if α is unlocked in σ , and there is
 231 at least one process in the source location $r.\text{from}$, formally, $\sigma.\vec{\kappa}[r.\text{from}, k] \geq 1$. When an
 232 action α is applicable to σ , and when ℓ is a potential destination location for the probabilistic
 233 action α , we write $\text{apply}(\sigma, \alpha, \ell)$ for the resulting configuration: parameters are unchanged,
 234 shared variables are updated according to the update vector $r.\vec{u}$, and the values of counters
 235 are modified in a natural way: as a process moves from $r.\text{from}$ to ℓ in round k , counter
 236 $\vec{\kappa}[r.\text{from}, k]$ is decreased by 1 and $\vec{\kappa}[\ell, k]$ is increased by 1. The probabilistic transition
 237 function Δ is defined by: $\Delta(\sigma, \alpha)(\sigma') = \alpha.\delta_{to}(\ell)$ if $\text{apply}(\sigma, \alpha, \ell) = \sigma'$, and 0 otherwise.

238 3.2 Non-probabilistic Counter Systems

239 Non-probabilistic threshold automata are defined in [12], and they can be seen as a special
 240 case of probabilistic threshold automata where all rules are Dirac rules.

241 With a PTA, one can naturally associate a non-probabilistic threshold automaton, by
 242 replacing probabilities with non-determinism.

243 **► Definition 1.** Given a PTA $= (\mathcal{L}, \mathcal{V}, \mathcal{R}, RC)$, its (non-probabilistic) threshold automaton
 244 is $\text{TA}_{\text{PTA}} = (\mathcal{L}, \mathcal{V}, \mathcal{R}_{np}, RC)$ where the set of rules \mathcal{R}_{np} is defined as $\{r_\ell = (\text{from}, \ell, \varphi, \vec{u}) : r =$
 245 $(\text{from}, \delta_{to}, \varphi, \vec{u}) \in \mathcal{R} \wedge \ell \in \mathcal{L} \wedge \delta_{to}(\ell) > 0\}$.

246 We write TA instead of TA_{PTA} when the automaton PTA is clear from the context. Every
 247 rule from \mathcal{R}_{np} corresponds to exactly one rule in \mathcal{R} , and for every rule in \mathcal{R} there is at least
 248 one corresponding rule in \mathcal{R}_{np} (and exactly one for Dirac rules).

249 If we understand a TA as a PTA where all rules are Dirac rules, we can define transitions
 250 using the partial function *apply* in order to obtain an infinite (non-probabilistic) counter
 251 system, which we denote by $\text{Sys}_\infty(\text{TA})$. Moreover, since in this case $\mathcal{R} = \mathcal{R}_{np}$, actions of
 252 the PTA exactly match transitions of its TA. We obtain σ' by applying $t = (r, k)$ to σ ,
 253 and write this as $\sigma' = t(\sigma)$, if and only if for the destination location ℓ of r holds that
 254 $\text{apply}(\sigma, t, \ell) = \sigma'$.

255 Also, starting from a PTA, one can define the counter system $\text{Sys}(\text{PTA})$, and consequently
 256 its non-probabilistic counterpart $\text{Sys}_{np}(\text{PTA})$. As the definitions of $\text{Sys}_{np}(\text{PTA})$ and $\text{Sys}_\infty(\text{TA})$
 257 are equivalent for a given PTA, we are free to choose one, and always use $\text{Sys}_\infty(\text{TA})$.

258 A (finite or infinite) sequence of transitions is called *schedule*, and it is often denoted
 259 by τ . A schedule $\tau = t_1, t_2, \dots, t_{|\tau|}$ is applicable to a configuration σ if there is a sequence
 260 of configurations $\sigma = \sigma_0, \sigma_1, \dots, \sigma_{|\tau|}$ such that for every $1 \leq i \leq |\tau|$ we have that t_i
 261 is applicable to σ_{i-1} and $\sigma_i = t_i(\sigma_{i-1})$. A *path* is an alternating sequence of configurations
 262 and transitions, for example $\sigma_0, t_1, \sigma_1, \dots, t_{|\tau|}, \sigma_{|\tau|}$, such that for every t_i , $1 \leq i \leq |\tau|$, in the
 263 sequence, we have that t_i is applicable to σ_{i-1} and $\sigma_i = t_i(\sigma_{i-1})$. Given a configuration σ_0
 264 and a schedule $\tau = t_1, t_2, \dots, t_{|\tau|}$, we denote by $\text{path}(\sigma_0, \tau)$ a path $\sigma_0, t_1, \sigma_1, \dots, t_{|\tau|}, \sigma_{|\tau|}$
 265 where $t_i(\sigma_{i-1}) = \sigma_i$, $1 \leq i \leq |\tau|$. Similarly we define an infinite schedule $\tau = t_1, t_2, \dots$,
 266 and an infinite path $\sigma_0, t_1, \sigma_1, \dots$, also denoted by $\text{path}(\sigma_0, \tau)$. An infinite path is *fair* if
 267 no transition is applicable forever from some point on. Equivalently, when a transition is
 268 applicable, eventually either its guard becomes false, or all processes leave its source location.

269 Since every transition in $\text{Sys}_\infty(\text{TA})$ comes from an action in $\text{Sys}(\text{PTA})$, note that every
 270 path in $\text{Sys}_\infty(\text{TA})$ is a valid path in $\text{Sys}(\text{PTA})$.

271 3.3 Adversaries

272 As usual, the non-determinism is resolved by a so-called adversary. Let Paths be the set of
 273 all finite paths in $\text{Sys}(\text{PTA})$. An *adversary* is a function $\mathbf{a} : \text{Paths} \rightarrow \text{Act}$, that given a finite
 274 path π selects an action applicable to the last configuration of π . Given a configuration σ
 275 and an adversary \mathbf{a} , we generate a family of paths, depending on the outcomes of non-Dirac
 276 transitions. We denote this set by $\text{paths}(\sigma, \mathbf{a})$. An adversary \mathbf{a} is *fair* if all paths in $\text{paths}(\sigma, \mathbf{a})$
 277 are fair. As usual, the Markov Decision Process (MDP) $\text{Sys}(\text{PTA})$ together with an initial
 278 configuration σ and an adversary \mathbf{a} induce a Markov chain, written $\mathcal{M}_\mathbf{a}^\sigma$. We write $\mathbb{P}_\mathbf{a}^\sigma$ for
 279 the probability measure over infinite paths starting at σ in the latter Markov chain.

280 We call an adversary \mathbf{a} *round-rigid* if it is fair, and if every sequence of actions it produces
 281 can be decomposed to a concatenation of sequences of action of the form $s_1 \cdot s_1^p \cdot s_2 \cdot s_2^p \dots$,
 282 where for all $k \in \mathbb{N}$, we have that s_k contains only Dirac actions of round k , and s_k^p contains
 283 only non-Dirac actions of round k . We denote the set of all round-rigid adversaries by \mathcal{A}^R .

284 3.4 Atomic Propositions and Stutter Equivalence

285 The atomic propositions we consider describe non-emptiness of a location $\ell \in \mathcal{L} \setminus \mathcal{B}$ in a
 286 round k , *i.e.*, whether there is at least one process in location ℓ in round k . Formally, the set of
 287 all such propositions for a round $k \in \mathbb{N}_0$ is denoted by $\text{AP}_k = \{p(\ell, k) : \ell \in \mathcal{L} \setminus \mathcal{B}\}$. For every k
 288 we define a labeling function $\lambda_k : \Sigma \rightarrow 2^{\text{AP}_k}$ such that $p(\ell, k) \in \lambda_k(\sigma)$ iff $\sigma.\vec{\mathfrak{r}}[\ell, k] > 0$. By
 289 abusing notation, we write “ $\vec{\mathfrak{r}}[\ell, k] > 0$ ” and “ $\vec{\mathfrak{r}}[\ell, k] = 0$ ” instead of $p(\ell, k)$ and $\neg p(\ell, k)$, resp.

290 We denote by $\pi_1 \stackrel{\Delta}{\sim}_k \pi_2$ that the paths π_1 and π_2 are stutter equivalent [2] w.r.t. AP_k .
 291 Two counter systems C_0 and C_1 are stutter equivalent w.r.t. AP_k , written $C_0 \stackrel{\Delta}{\sim}_k C_1$, if for
 292 every path π from C_i there is a path π' from C_{1-i} such that $\pi \stackrel{\Delta}{\sim}_k \pi'$, for $i \in \{0, 1\}$.

293 4 Consensus Properties and their Verification

294 In probabilistic (binary) consensus every correct process has an initial value from $\{0, 1\}$.
 295 It consists of safety specifications and an almost-sure termination requirement, which we
 296 consider in its round-rigid variant:

297 **Agreement:** No two correct processes decide differently.

298 **Validity:** If all correct processes have v as the initial value, then no process decides $1 - v$.

299 **Probabilistic wait-free termination:** Under every round-rigid adversary, with probability 1
 300 every correct process eventually decides.

301 We now discuss the formalization of these specifications in the context of Ben-Or's
 302 algorithm whose threshold automaton is given in Figure 2.

303 **Formalization.** In order to formulate and analyze the specifications, we partition every
 304 set \mathcal{I} , \mathcal{B} , and \mathcal{F} , into two subsets $\mathcal{I}_0 \uplus \mathcal{I}_1$, $\mathcal{B}_0 \uplus \mathcal{B}_1$, and $\mathcal{F}_0 \uplus \mathcal{F}_1$, respectively. For every
 305 $v \in \{0, 1\}$, the partitions satisfy the following:

306 (R1) The processes that are initially in a location $\ell \in \mathcal{I}_v$ have the initial value v .

307 (R2) Rules connecting locations from \mathcal{B} and \mathcal{I} respect the partitioning, *i.e.*, they connect \mathcal{B}_v
 308 and \mathcal{I}_v . Similarly, rules connecting locations from \mathcal{F} and \mathcal{B} respect the partitioning.

309 We introduce two subsets $\mathcal{D}_v \subseteq \mathcal{F}_v$, for $v \in \{0, 1\}$. Intuitively, a process is in \mathcal{D}_v in a round k
 310 if and only if it decides v in that round. Now we can express the specifications as follows:

311 **Agreement:** For both values $v \in \{0, 1\}$ the following holds:

$$312 \quad \forall k \in \mathbb{N}_0, \forall k' \in \mathbb{N}_0. \mathbf{A} \left(\mathbf{F} \bigvee_{\ell \in \mathcal{D}_v} \vec{\kappa}[\ell, k] > 0 \rightarrow \mathbf{G} \bigwedge_{\ell' \in \mathcal{D}_{1-v}} \vec{\kappa}[\ell', k'] = 0 \right) \quad (1)$$

313 **Validity:** For both $v \in \{0, 1\}$ it holds

$$314 \quad \forall k \in \mathbb{N}_0. \mathbf{A} \left(\bigwedge_{\ell \in \mathcal{I}_v} \vec{\kappa}[\ell, 0] = 0 \rightarrow \mathbf{G} \bigwedge_{\ell' \in \mathcal{D}_v} \vec{\kappa}[\ell', k] = 0 \right) \quad (2)$$

315 **Probabilistic wait-free termination:** For every round-rigid adversary \mathbf{a}

$$316 \quad \mathbb{P}_{\mathbf{a}} \left(\bigvee_{k \in \mathbb{N}_0} \bigvee_{v \in \{0, 1\}} \mathbf{G} \bigwedge_{\ell \in \mathcal{F} \setminus \mathcal{D}_v} \vec{\kappa}[\ell, k] = 0 \right) = 1 \quad (3)$$

317 Agreement and validity are non-probabilistic properties, and thus can be analyzed on the
 318 non-probabilistic counter system $\text{Sys}_{\infty}(\text{TA})$. For verifying probabilistic wait-free termination,
 319 we make explicit the following assumption that is present in all our benchmarks: all non-Dirac
 320 transitions have non-zero probability to lead to an \mathcal{F}_v location, for both values $v \in \{0, 1\}$.

321 In Section 5 we formalize safety specifications and reduce them to single-round specifica-
 322 tions. In Section 6 we reduce verification of multi-round counter systems to verification of
 323 single-round systems. In Section 7 we discuss our approach to probabilistic termination.

324 5 Reduction to Specifications with one Round Quantifier

325 Agreement contains two round variables k and k' , and Validity considers rounds 0 and k .
 326 Thus, both involve two round numbers. As ByMC can only analyze one round systems [11],
 327 the properties are only allowed to use one round number. In this section we show how to
 328 check formulas (1) and (2) by checking properties that refer to one round. Namely, we
 329 introduce round invariants (4) and (5), and prove that they imply Agreement and Validity.

330 The first round invariant claims that in every round and in every path, once a process
331 decides v in a round, no process ever enters a location from \mathcal{F}_{1-v} in that round. Formally:

$$332 \quad \forall k \in \mathbb{N}_0. \mathbf{A} \left(\mathbf{F} \bigvee_{\ell \in \mathcal{D}_v} \vec{\kappa}[\ell, k] > 0 \rightarrow \mathbf{G} \bigwedge_{\ell' \in \mathcal{F}_{1-v}} \vec{\kappa}[\ell', k] = 0 \right) \quad (4)$$

333 The second round invariant claims that in every round in every path, if no process starts
334 a round with a value v , then no process terminates that round with value v . Formally:

$$335 \quad \forall k \in \mathbb{N}_0. \mathbf{A} \left(\mathbf{G} \bigwedge_{\ell \in \mathcal{I}_v} \vec{\kappa}[\ell, k] = 0 \rightarrow \mathbf{G} \bigwedge_{\ell' \in \mathcal{F}_v} \vec{\kappa}[\ell', k] = 0 \right) \quad (5)$$

336 The following proposition is proved using restriction (R2) and by an inductive argument
337 over the round number.

338 ► **Proposition 2.** *If $\text{Sys}_\infty(TA) \models (4) \wedge (5)$, then $\text{Sys}_\infty(TA) \models (1) \wedge (2)$.*

339 6 Reduction to Single-Round Counter System

340 Given a property of one round, our goal is to prove that there is a counterexample to the
341 property in the multi-round system iff there is a counterexample in a single-round system.
342 This is formulated in Theorem 6, and it allows us to use ByMC on a single-round system.

343 The proof idea contains two parts. First, in Section 6.1 we prove that one can replace an
344 arbitrary finite schedule with a round-rigid one, while preserving atomic propositions of a
345 fixed round. We show that swapping two adjacent transitions that do not respect the order
346 over round numbers in an execution, gives us a legal stutter equivalent execution, *i.e.*, an
347 execution satisfying the same LTL_X properties.

348 Second, in Section 6.2 we extend this reasoning to infinite schedules, and lift it from
349 schedules to transition systems. The main idea is to do inductive and compositional reasoning
350 over the rounds. To do so, we need well-defined round boundaries, which is the case if
351 every round that is started is also finished; a property we can automatically check for fair
352 schedules. In more detail, regarding propositions for one round, we show that the multi-round
353 transition system is stutter equivalent to a single-round transition system. This holds under
354 the assumption that all fair executions of a single-round transition system terminate, and
355 this can be checked using the technique from [11]. As stutter equivalence of systems implies
356 preservation of LTL_X properties, this is sufficient to prove the main goal of the section.

357 6.1 Reduction from arbitrary schedules to round-rigid schedules

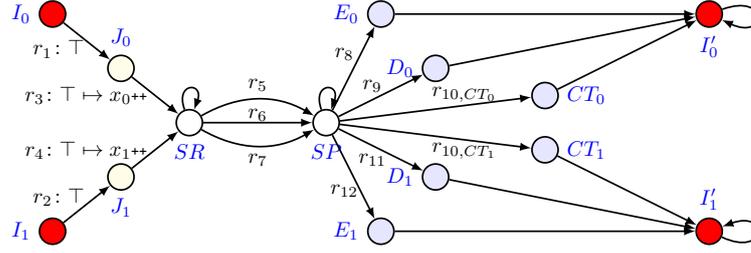
358 ► **Definition 3.** *A schedule $\tau = (r_1, k_1) \cdot (r_2, k_2) \cdot \dots \cdot (r_m, k_m)$, $m \in \mathbb{N}_0$, is called round-rigid
359 if for every $1 \leq i < j \leq m$, we have $k_i \leq k_j$.*

360 The following proposition shows that any finite schedule can be re-ordered into a round-
361 rigid one that is stutter equivalent regarding LTL_X formulas over proposition from AP_k , for
362 all rounds k . It is proved using arguments on the commutativity of transitions, similar to [8].

363 ► **Proposition 4.** *For every configuration σ and every finite schedule τ applicable to σ , there
364 is a round-rigid schedule τ' such that the following holds:*

- 365 (a) *Schedule τ' is applicable to configuration σ .*
- 366 (b) *τ' and τ reach the same configuration when applied to σ , *i.e.*, $\tau'(\sigma) = \tau(\sigma)$.*
- 367 (c) *For every $k \in \mathbb{N}_0$ we have $\text{path}(\sigma, \tau) \stackrel{\Delta}{=} \text{path}(\sigma, \tau')$.*

368 Thus, instead of reasoning about all finite schedules of $\text{Sys}_\infty(TA)$, it is sufficient to reason
369 about its round-rigid schedules. In the following section we use this to simplify the verification
370 further, namely to a single-round counter system.



■ **Figure 3** The single-round threshold automaton TA^{rd} obtained from PTA in Figure 2

371 6.2 From round-rigid schedules to single-round counter system

372 For each PTA, we define a *single-round threshold automaton* that can be analyzed with the
 373 tools of [12] and [11]. Roughly speaking, we focus on one round, but also keep the border
 374 locations of the next round, where we add self-loops. Figure 3 represents the single-round
 375 threshold automaton associated with the PTA from Figure 2. We can prove that for specific
 376 fairness constraints, this automaton shows the same behavior as a round in $\text{Sys}_\infty(\text{TA})$.

377 We restrict ourselves to fair schedules, that is, those where no transition is applicable
 378 forever. We also assume that every fair schedule of a single-round system terminates, *i.e.*,
 379 eventually every process reaches a state from \mathcal{B}' . Under the fairness assumption we check
 380 the latter assumption with ByMC [13]. Moreover, we restrict ourselves to non-blocking
 381 threshold automata, that is, we require that in each configuration each location has at least
 382 one outgoing rule unlocked. As we use TAs to model distributed algorithms, this is no
 383 restriction: locations in which no progress should be made unless certain thresholds are
 384 reached, typically have self-loops that are guarded with `true` (e.g. *SR* and *SP*). Thus for
 385 our benchmarks one can easily check whether they are non-blocking using SMT (we have to
 386 check that there is no evaluation of the variables such that all outgoing rules are disabled).

387 ► **Definition 5.** *Given a PTA $(\mathcal{L}, \mathcal{V}, \mathcal{R}, RC)$ or its TA $(\mathcal{L}, \mathcal{V}, \mathcal{R}_{np}, RC)$, we define a single-*
 388 *round threshold automaton $\text{TA}^{\text{rd}} = (\mathcal{L} \cup \mathcal{B}', \mathcal{V}, \mathcal{R}^{\text{rd}}, RC)$, where $\mathcal{B}' = \{\ell' : \ell \in \mathcal{B}\}$ are copies*
 389 *of border locations, and $\mathcal{R}^{\text{rd}} = (\mathcal{R}_{np} \setminus \mathcal{S}) \cup \mathcal{S}' \cup \mathcal{R}^{\text{loop}}$, where $\mathcal{R}^{\text{loop}} = \{(\ell', \ell', \text{true}, \vec{0}) : \ell' \in \mathcal{B}'\}$*
 390 *are self-loop rules at locations from \mathcal{B}' and $\mathcal{S}' = \{(from, \ell', \text{true}, \vec{0}) : (from, \ell, \text{true}, \vec{0}) \in$*
 391 *\mathcal{S} with $\ell' \in \mathcal{B}'\}$ consists of modifications of round switch rules. Initial locations of TA^{rd} are*
 392 *the locations from $\mathcal{B} \subseteq \mathcal{L}$.*

393 For a TA^{rd} and a $k \in \mathbb{N}_0$ we define a counter system $\text{Sys}^k(\text{TA}^{\text{rd}})$ as the tuple (Σ^k, I^k, R^k) .
 394 A configuration is a tuple $\sigma = (\vec{\kappa}, \vec{g}, \mathbf{p}) \in \Sigma^k$, where $\sigma.\vec{\kappa}: \mathcal{D} \rightarrow \mathbb{N}_0$ defines values of the
 395 counters, for $\mathcal{D} = (\mathcal{L} \times \{k\}) \cup (\mathcal{B}' \times \{k+1\})$; and $\sigma.\vec{g}: \Gamma \times \{k\} \rightarrow \mathbb{N}_0$ defines shared variable
 396 values; and $\sigma.\mathbf{p} \in \mathbb{N}_0^{|\Pi|}$ is a vector of parameter values.

397 Note that by using \mathcal{D} in the definition of $\sigma.\vec{\kappa}$ above, every configuration $\sigma \in \text{Sys}^k(\text{TA}^{\text{rd}})$
 398 can be extended to a valid configuration of $\text{Sys}_\infty(\text{TA})$, by assigning zero to all other counters
 399 and global variables. In the following, we identify a configuration in $\text{Sys}^k(\text{TA}^{\text{rd}})$ with its
 400 extension in $\text{Sys}_\infty(\text{TA})$, since they have the same labeling function λ_k , for every $k \in \mathbb{N}_0$.

401 We define $\Sigma_{\mathcal{B}}^k \subseteq \Sigma^k$, for a $k \in \mathbb{N}_0$, to be the set of all configurations σ where every process
 402 is in a location from \mathcal{B} , and all shared variables are set to zero in k , formally, $\sigma.\vec{g}[x, k] = 0$ for
 403 all $x \in \Gamma$, and $\sum_{\ell \in \mathcal{B}} \sigma.\vec{\kappa}[\ell, k] = N(\mathbf{p})$, and $\sigma.\vec{\kappa}[\ell, i] = 0$ for all $(\ell, i) \in \mathcal{D} \setminus (\mathcal{B} \times \{k\})$. We call
 404 these configurations *border configurations for k* . The set of initial states I^k is a subset of $\Sigma_{\mathcal{B}}^k$.

405 We define the transition relation R as in $\text{Sys}_\infty(\text{TA})$, *i.e.*, two configurations are in the
 406 relation R^k if and only if they (or more precisely, their above described extensions) are in R .

407 For every $k \in \mathbb{N}_0$ and every $\sigma \in \Sigma_{\mathcal{B}}^k$, there is a corresponding configuration $\sigma' \in \Sigma_{\mathcal{B}}^0$
 408 obtained from σ by renaming the round k to round 0. Let f_k be the renaming function, *i.e.*,
 409 $\sigma' = f_k(\sigma)$. Let us define $\Sigma^u \subseteq \Sigma_{\mathcal{B}}^0$ to be the union of all renamed initial configurations from
 410 all rounds, that is, $\{f_k(\sigma) : k \in \mathbb{N}_0, \sigma \in I^k\}$.

411 ► **Theorem 6.** *Let TA be non-blocking, and let all fair executions of $\text{Sys}^0(\text{TA}^{rd})$ w.r.t. $\Sigma_{\mathcal{B}}^0$*
 412 *terminate. Given a formula $\varphi[i]$ from $\text{LTL}_{\mathcal{X}}$ over AP_i , for a round variable i , we have*
 413 *(A) $\text{Sys}^0(\text{TA}^{rd}) \models \mathbf{E}\varphi[0]$ w.r.t. initial configurations Σ^u if and only if*
 414 *(B) there exists $k \in \mathbb{N}_0$ such that $\text{Sys}_{\infty}(\text{TA}) \models \mathbf{E}\varphi[k]$.*

415 **Proof sketch.** The theorem is proved using the following arguments. In statement (B), the
 416 existential quantification over k corresponds to the definition of Σ^u as union, over all rounds,
 417 of projections of all reachable initial configurations of that round.

418 Implication (B) \rightarrow (A) exploits the fact that all rounds are equivalent up to renaming of
 419 round numbers (with the exception of possible initial configurations).

420 Implication (A) \rightarrow (B), note that an initial configuration in Σ^u is not necessarily initial in
 421 round 0, so that one cannot *a priori* take $k = 0$. Let us explain how to extend an execution
 422 of round k into an infinite execution in $\text{Sys}_{\infty}(\text{TA})$. By termination, all rounds up to $k-1$
 423 terminate, so that there is execution that reaches a configuration where all processes are in
 424 initial locations of round k . The executions of round k mimick the ones of round 0 (modulo
 425 the round number). Finally, the non-blocking assumption is required to be always able to
 426 extend to infinite executions after round k is terminated. ◀

427 In Section 4 we showed how to reduce our specifications to formulas of the form $\forall k \in$
 428 $\mathbb{N}_0. \mathbf{A}\psi[k]$. Theorem 6 deals with negations of such formulas, namely with existence of
 429 a round k such that formula $\mathbf{E}\varphi[k]$ holds. Therefore, we can check specifications on the
 430 single-round system.

431 7 Probabilistic Wait-Free Termination

432 We start by defining two conditions that are sufficient to establish Probabilistic Wait-Free
 433 Termination under round-rigid adversaries. Condition (C1) states the existence of a positive
 434 probability lower-bound for all processes ending round k with equal final values. Condition
 435 (C2) states that if all correct processes start round k with the same value, then they all will
 436 decide on that value in that round.

437 (C1) There is a bound $p \in (0, 1]$, such that for every round-rigid adversary \mathbf{a} , and every $k \in \mathbb{N}_0$,
 438 and every configuration σ_k with parameters \mathbf{p} that is initial for round k , it holds that

$$439 \mathbb{P}_{\mathbf{a}}^{\sigma_k} \left(\bigvee_{v \in \{0,1\}} \mathbf{G} \left(\bigwedge_{\ell \in \mathcal{F}_v} \vec{\kappa}[\ell, k] = 0 \right) \right) > p.$$

440 (C2) For every $v \in \{0, 1\}$, $\forall k \in \mathbb{N}_0. \mathbf{A} \left(\mathbf{G} \bigwedge_{\ell \in \mathcal{I}_{1-v}} \vec{\kappa}[\ell, k] = 0 \rightarrow \mathbf{G} \bigwedge_{\ell' \in \mathcal{F} \setminus \mathcal{D}_v} \vec{\kappa}[\ell', k] = 0 \right)$.

441 Combining (C1) and (C2), under every round-rigid adversary, from any initial configuration
 442 of round k , the probability that all correct processes decide before end of round $k+1$ is at
 443 least p . Thus the probability not to decide within $2n$ rounds is at most $(1-p)^n$, which tends
 444 to 0 when n tends to infinity. This reasoning follows the arguments of [1].

445 ► **Proposition 7.** *If $\text{Sys}_{\infty}(\text{PTA}) \models (C1) \wedge (C2)$, then $\text{Sys}_{\infty}(\text{PTA}) \models (3)$.*

446 Observe (C2) is a non-probabilistic property of the same form as (5), so that we can
 447 check (C2) using the method of Section 6.

448 In the rest of this section, we detail how to reduce the verification of (C1), to a verification
 449 task that can be handled by ByMC. First observe that (C1) contains a single round variable,
 450 and recall that we restrict to round-rigid adversaries, so that it is sufficient to check them
 451 (omitting the round variables) on the single-round system. We introduce analogous objects
 452 as in the non-probabilistic case: PTA^{rd} (analogously to Definition 5), and its counter
 453 system $\text{Sys}(\text{PTA}^{\text{rd}})$.

454 7.1 Reducing probabilistic to non-probabilistic specifications

455 Since probabilistic transitions end in final locations, they cannot appear on a cycle in PTA^{rd} .
 456 Therefore, for fixed parameter valuation \mathbf{p} , any path contains at most $N(\mathbf{p})$ probabilistic
 457 transitions, and its probability is therefore uniformly lower-bounded. As a consequence:

458 **► Lemma 8.** *Let $\mathbf{p} \in \mathbf{P}_{RC}$ be a parameter valuation. In $\text{Sys}(\text{PTA}^{\text{rd}})$, for every LTL formula φ
 459 over atomic proposition AP, the following two statements are equivalent:*

- 460 (a) $\exists p > 0, \forall \sigma \in I_{\mathbf{p}}, \forall \mathbf{a} \in \mathcal{A}^R. \mathbb{P}_{\mathbf{a}}^{\sigma}(\varphi) > p,$
 461 (b) $\forall \sigma \in I_{\mathbf{p}}, \forall \mathbf{a} \in \mathcal{A}^R, \exists \pi \in \text{paths}(\sigma, \mathbf{a}). \pi \models \varphi.$

462 7.2 Verifying (C1) on a non-probabilistic TA

463 Applying Lemma 8, proving (C1) is equivalent to proving the following property on $\text{Sys}(\text{PTA}^{\text{rd}})$
 464

$$465 \quad \forall \sigma \in I_{\mathbf{p}}, \forall \mathbf{a} \in \mathcal{A}^R, \exists \pi \in \text{paths}(\sigma, \mathbf{a}). \quad \pi \models \bigvee_{v \in \{0,1\}} \mathbf{G} \left(\bigwedge_{\ell \in \mathcal{F}_v} \vec{\kappa}[\ell] = 0 \right). \quad (6)$$

466 In the sequel, we explain how to reduce the verification of (6) to checking the simpler formula
 467 $\mathbf{A} \bigvee_{v \in \{0,1\}} \mathbf{G} \left(\bigwedge_{\ell \in \mathcal{F}_v} \vec{\kappa}[\ell] = 0 \right)$ on a single-round non-probabilistic TA obtained from PTA^{rd} .

468 As in Section 6, it is possible to modify PTA^{rd} into a non-probabilistic TA, by replacing
 469 probabilistic choices by non-determinism. Still, the quantifier alternation of (6) (universal
 470 over initial configurations and adversaries vs. existential on paths) is not in the fragment
 471 handled by ByMC [13]. Once an initial configuration σ and an adversary \mathbf{a} are fixed, the
 472 remaining branching is induced by non-Dirac transitions. By assumption, these transitions
 473 lead to final locations only, to both \mathcal{F}_0 and \mathcal{F}_1 , and under round-rigid adversaries, they are
 474 the last transitions to be fired. To prove (6), it is sufficient to prove that all processes that
 475 fire only Dirac transitions will reach final locations of the same type (\mathcal{F}_0 or \mathcal{F}_1). If this is the
 476 case, then the existence of a path corresponds to all non-Dirac transitions being resolved in
 477 the same way. This allows us to remove the non-Dirac transitions from the model as follows.
 478 Given a PTA^{rd} , we define a threshold automaton TA^{m} with locations \mathcal{L} (without \mathcal{B}') such
 479 that for every non-Dirac rule $r = (\text{from}, \delta_{to}, \varphi, \vec{u})$ in PTA, all locations ℓ with $\delta_{to}(\ell) > 0$ are
 480 merged into a new location ℓ^{mrg} in TA^{m} . Note that this location must belong to \mathcal{F} . Naturally,
 481 instead of a non-Dirac rule r we obtain a Dirac rule $(\text{from}, \ell^{\text{mrg}}, \varphi, \vec{u})$. Also we add self-loops
 482 at all final locations. Paths in $\text{Sys}(\text{TA}^{\text{m}})$ correspond to prefixes of paths in $\text{Sys}(\text{PTA}^{\text{rd}})$. In
 483 $\text{Sys}(\text{TA}^{\text{m}})$, from a configuration σ , an adversary \mathbf{a} yields a unique path, that is, $\text{paths}(\sigma, \mathbf{a})$
 484 is a singleton set. Thus, the existential quantifier from (6) can be replaced by the universal one.

485 By construction, property (6) on PTA^{rd} is equivalent to $\mathbf{A} \bigvee_{v \in \{0,1\}} \mathbf{G} \left(\bigwedge_{\ell \in \mathcal{F}_v} \vec{\kappa}[\ell] = 0 \right)$
 486 on $\text{Sys}(\text{TA}^{\text{m}})$. The latter can be checked automatically by ByMC, allowing us to prove (C1).

Label	Name	Automaton	Formula
S1	agreement_0	N	$\mathbf{A G}(\neg\text{EX}\{D0\}) \vee \mathbf{G}(\neg\text{EX}\{D1, E1\})$
S2	validity_0	N	$\mathbf{A ALL}\{V0\} \rightarrow \mathbf{G}(\neg\text{EX}\{D1, E1\})$
S3	completeness_0	N	$\mathbf{A ALL}\{V0\} \rightarrow \mathbf{G}(\neg\text{EX}\{D1, E1\})$
S4	round-term	N	$\mathbf{A fair} \rightarrow \mathbf{F ALL}\{D0, D1, E0, E1, CT\}$
S5	decide-or-flip	P	$\mathbf{A fair} \rightarrow \mathbf{F}(\text{ALL}\{D0, E0, CT\} \vee \text{ALL}\{D1, E1, CT\})$
S1'	sim-agreement	N	$\mathbf{A G}(\neg\text{EX}\{D0, E0\} \vee \neg\text{EX}\{D1, E1\})$
S1''	2-agreement	N	$\mathbf{A G}(\neg\text{EX}\{D0, E0\} \vee \neg\text{EX}\{D1, E1\} \vee \neg\text{EX}\{D2, E2\})$

■ **Table 1** Temporal properties verified in our experiments for value 0.

8 Experiments

487

488 We have applied the approach presented in Sections 4–7 to five randomized fault-tolerant
 489 distributed algorithms. (The benchmarks and the instructions on running the experiments
 490 are available from: <https://forsyte.at/software/bymc/artifact42/>)

- 491 1. Protocol 1 for randomized consensus by Ben-Or [3], with two kinds of crashes: clean
 492 crashes (*ben-or-cc*), for which a process either sends to all processes or none, and dirty
 493 crashes (*ben-or-dc*), for which a process may send to a subset of processes. This algorithm
 494 works correctly when $n > 2t$.
- 495 2. Protocol 2 for randomized Byzantine consensus (*ben-or-byz*) by Ben-Or [3]. This algorithm
 496 tolerates Byzantine faults when $n > 5t$.
- 497 3. Protocol 2 for randomized consensus (*rabc-c*) by Bracha [5]. It runs as a high-level
 498 algorithm together with a low-level broadcast that turns Byzantine faults into “little
 499 more than fail-stop (faults)”. We check only the high-level algorithm for clean crashes.
- 500 4. k -set agreement for crash faults (*kset*) by Raynal [21], for $k = 2$. This algorithm works in
 501 presence of clean crashes when $n > 3t$.
- 502 5. Randomized Byzantine one-step consensus (*rs-bosco*) by Song and van Renesse [23]. This
 503 algorithm tolerates Byzantine faults when $n > 3t$, and it terminates fast when $n > 7t$ or
 504 $n > 5t$ and $f = 0$.

505 Following the reduction approach of Sections 4–7, for each benchmark, we have encoded
 506 two versions of one-round threshold automata: an N-automaton that models a coin toss
 507 by a non-deterministic choice in a coin-toss location, and is used for the non-probabilistic
 508 reasoning, and a P-automaton that never leaves the coin-toss location and which is used to
 509 prove probabilistic wait-free termination. Both automata are given as the input to Byzantine
 510 Model Checker (ByMC) [13], which implements the parameterized model checking techniques
 511 for safety [10] and liveness [11] of counter systems of threshold automata.

512 Both automata follow the pattern shown in Figure 2: They start in one of the initial
 513 locations (e.g., V_0 or V_1), progress by switching locations and incrementing shared variables
 514 and end up in a location that corresponds to a decision (e.g., D_0 or D_1), an estimate of a
 515 decision (e.g., E_0 or E_1), or a coin toss (CT).

516 Table 1 summarizes the temporal properties that were verified in our experiments. Given
 517 the set of all possible locations \mathcal{L} , a set $Y = \{\ell_1, \dots, \ell_m\} \subseteq \mathcal{L}$ of locations, and the designated
 518 crashed location $\text{CR} \in \mathcal{L}$, we use the shorthand notation: $\text{EX}\{\ell_1, \dots, \ell_m\}$ for $\bigvee_{\ell \in Y} \vec{\kappa}[\ell] \neq 0$
 519 and $\text{ALL}\{\ell_1, \dots, \ell_m\}$ for $\bigwedge_{\ell \in \mathcal{L} \setminus Y} (\vec{\kappa}[\ell] = 0 \vee \ell = \text{CR})$. For *rs-bosco* and *kset*, instead of
 520 checking S1, we check S1' and S1''.

■ **Table 2** The experiments for first 5 rows were run on a single computer (Apple MacBook Pro 2018, 16GB). The experiments for last row (*rs-bosco*) were run in Grid5000 on 32 nodes (2 CPUs Intel Xeon Gold 6130, 16 cores/CPU, 192GB). Wall times are given.

Automaton			S1/S1'/S1''		S2		S3		S4		S5	
Name	$ \mathcal{L} $	$ \mathcal{R} $	$ \mathcal{S} $	Time								
ben-or-cc	10	27	9	1	5	0	5	0	5	0	5	0
ben-or-dc	10	32	9	1	5	1	5	0	5	0	5	1
ben-or-byz	9	18	3	1	2	0	2	0	2	0	2	1
rabc-cr	11	31	9	0	5	1	5	1	5	0	5	0
kset	13	58	65	3	65	17	65	12	65	39	65	40
rs-bosco	19	48	156M	3:21	156M	3:02	156M	3:21	TO	TO	156M	3:43

521 Table 2 presents the computational results of our experiments: column $|\mathcal{L}|$ shows the
 522 number of automata locations, column $|\mathcal{R}|$ shows the number of automata rules, column $|\mathcal{S}|$
 523 shows the number of SMT queries (which depends on the structure of the automaton and
 524 the specification), column *time* shows the computation times — either in seconds or in the
 525 format HH:MM. As the N-automata have more rules than the P-automata, column $|\mathcal{R}|$ shows
 526 the figures for N-automata. To save space, we omit the figures for memory use from the
 527 table: Benchmarks 1–5 need 30–170 MB, whereas *rs-bosco* needs up to 1.5 GB per CPU.

528 The benchmark *rs-bosco* is a challenge for the technique of [11]: Its threshold automaton
 529 has 12 threshold guards that can change their values almost in any order. Additional
 530 combinations are produced by the temporal formulas. Although ByMC reduces the number
 531 of combinations by analyzing dependencies between the guards, it still produces between
 532 11! and 14! SMT queries. Hence, we ran the experiments for *rs-bosco* on 1024 CPU cores
 533 of Grid5000 and gave the wall time results in Table 2. (To find the total computing time,
 534 multiply wall time by 1024.) ByMC timed out on the property *S4* after 1 day (shown as TO).

535 For all the benchmarks in Table 2, ByMC has reported that the specifications hold. By
 536 changing $n > 3t$ to $n > 2t$, we found that *rabc-cr* can handle more faults (the original $n > 3t$
 537 was needed to implement the underlying communication structure which we assume given in
 538 the experiments). In other cases, whenever we changed the parameters, that is, increased
 539 the number of faults beyond the known bound, the tool reported a counterexample.

540 9 Conclusions

541 Our proof methodology for almost sure termination applies to round-rigid adversaries only.
 542 As future work we shall prove that verifying almost-sure termination under round-rigid
 543 adversaries is sufficient to prove it for more general adversaries. Transforming an adversary
 544 into a round-rigid one while preserving the probabilistic properties over the induced paths,
 545 comes up against the fact that, depending on the outcome of a coin toss in some step at
 546 round k , different rules may be triggered later for processes in rounds less than k .

547 A few contributions address automated verification of probabilistic parameterized sys-
 548 tems [22, 4, 20, 19]. In contrast to these, our processes are not finite-state, due to the round
 549 numbers and parameterized guards. The seminal work by Pnueli and Zuck [22] requires
 550 shared variables to be bounded and cannot use arithmetic thresholds different from 1 and n .
 551 Algorithms for well-structured transition systems [4] do not directly apply to multi-parameter
 552 systems produced by probabilistic threshold automata. Regular model checking [20, 19]
 553 cannot handle arithmetic resilience conditions such as $n > 3t$, nor unbounded shared variables.

554 — **References** —

- 555 1 Marcos Aguilera and Sam Toueg. The correctness proof of Ben-Or’s randomized consensus
556 algorithm. *Distributed Computing*, pages 1–11, 2012. online first.
- 557 2 Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.
- 558 3 Michael Ben-Or. Another advantage of free choice: Completely asynchronous agreement
559 protocols (extended abstract). In *PODC*, pages 27–30, 1983.
- 560 4 Nathalie Bertrand and Paulin Fournier. Parameterized verification of many identical proba-
561 bilistic timed processes. In *FSTTCS*, volume 24 of *LIPICs*, pages 501–513, 2013.
- 562 5 Gabriel Bracha. Asynchronous Byzantine agreement protocols. *Inf. Comput.*, 75(2):130–143,
563 1987.
- 564 6 Mouna Chaouch-Saad, Bernadette Charron-Bost, and Stephan Merz. A reduction theorem for
565 the verification of round-based distributed algorithms. In *RP*, volume 5797 of *LNCS*, pages
566 93–106, 2009.
- 567 7 Andrei Damian, Cezara Drăgoi, Alexandru Militaru, and Josef Widder. Communication-closed
568 asynchronous protocols. In *CAV*, 2019. (to appear).
- 569 8 Tzilla Elrad and Nissim Francez. Decomposition of distributed programs into communication-
570 closed layers. *Sci. Comput. Program.*, 2(3):155–173, 1982.
- 571 9 Michael J. Fischer, Nancy A. Lynch, and M. S. Paterson. Impossibility of distributed consensus
572 with one faulty process. *J. ACM*, 32(2):374–382, 1985.
- 573 10 Igor Konnov, Marijana Lazic, Helmut Veith, and Josef Widder. Para²: Parameterized path
574 reduction, acceleration, and SMT for reachability in threshold-guarded distributed algorithms.
575 *Formal Methods in System Design*, 51(2):270–307, 2017.
- 576 11 Igor Konnov, Marijana Lazić, Helmut Veith, and Josef Widder. A short counterexample
577 property for safety and liveness verification of fault-tolerant distributed algorithms. In *POPL*,
578 pages 719–734, 2017.
- 579 12 Igor Konnov, Helmut Veith, and Josef Widder. On the completeness of bounded model checking
580 for threshold-based distributed algorithms: Reachability. *Information and Computation*, 252:95–
581 109, 2017.
- 582 13 Igor Konnov and Josef Widder. ByMC: Byzantine model checker. In *ISoLA (3)*, volume 11246
583 of *LNCS*, pages 327–342. Springer, 2018.
- 584 14 Jure Kukovec, Igor Konnov, and Josef Widder. Reachability in parameterized systems: All
585 flavors of threshold automata. In *CONCUR*, pages 19:1–19:17, 2018.
- 586 15 Marta Z. Kwiatkowska and Gethin Norman. Verifying randomized byzantine agreement. In
587 *FORTE*, pages 194–209, 2002.
- 588 16 Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of
589 probabilistic real-time systems. In *CAV*, pages 585–591, 2011.
- 590 17 Marta Z. Kwiatkowska, Gethin Norman, and Roberto Segala. Automated verification of a
591 randomized distributed consensus protocol using Cadence SMV and PRISM. In *CAV*, pages
592 194–206, 2001.
- 593 18 Daniel J. Lehmann and Michael O. Rabin. On the advantages of free choice: A symmetric and
594 fully distributed solution to the dining philosophers problem. In *POPL*, pages 133–138, 1981.
- 595 19 Ondrej Lengál, Anthony Widjaja Lin, Rupak Majumdar, and Philipp Rümmer. Fair termination
596 for parameterized probabilistic concurrent systems. In *TACAS*, volume 10205 of *LNCS*, pages
597 499–517, 2017. doi:10.1007/978-3-662-54577-5_29.
- 598 20 Anthony Widjaja Lin and Philipp Rümmer. Liveness of randomised parameterised systems
599 under arbitrary schedulers. In *CAV*, volume 9780 of *LNCS*, pages 112–133. Springer, 2016.
600 doi:10.1007/978-3-319-41540-6_7.
- 601 21 Achour Mostéfaoui, Hamouma Moumen, and Michel Raynal. Randomized k-set agreement in
602 crash-prone and Byzantine asynchronous systems. *Theor. Comput. Sci.*, 709:80–97, 2018.
- 603 22 Amir Pnueli and Lenore D. Zuck. Verification of multiprocess probabilistic protocols. *Dis-
604 tributed Computing*, 1(1):53–72, 1986. doi:10.1007/BF01843570.

29:16 Verification of Randomized Consensus Algorithms under Round-Rigid Adversaries

- 605 23 Yee Jiun Song and Robbert van Renesse. Bosco: One-step Byzantine asynchronous consensus.
606 In *DISC*, volume 5218 of *LNCS*, pages 438–450, 2008.