Verification of Randomized Distributed Algorithms under Round-Rigid Adversaries

Nathalie Bertrand, Igor Konnov, Marijana Lazic, Josef Widder

Verification of Randomized Distributed Algorithms under Round-Rigid Adversaries *

Nathalie Bertrand[1], Igor Konnov[2], Marijana Lazić[3], and Josef Widder[3]

[1] Inria Rennes, France
nathalie.bertrand@inria.fr
[2] University of Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
igor.konnov@inria.fr
[3] TU Wien (Vienna University of Technology), Vienna, Austria
{lazic,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. The combination of these challenges makes verification hard. Challenge (i) was recently addressed in the framework of threshold automata.
We extend threshold automata to model randomized 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 no process enters round r before all other 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 to 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.

1 Introduction

Fault-tolerant distributed algorithms is an active research area, and systems like Paxos and Blockchain receive much attention. These systems are out of reach with current automated verification techniques. One problem comes from the scale: these systems should be verified for a very large number of participants, and even ideally for an unbounded number. In addition, many systems (including Blockchain), provide probabilistic guarantees. To check their correctness, one has to reason about randomized distributed algorithms in the parameterized setting.

* Experiments presented in this paper were carried out using the Grid5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations, see grid5000.fr.

```
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
```

Fig. 1. Pseudo code of Ben-Or's algorithm for Byzantine faults

In this paper, we make first steps towards parameterized verification of fault-tolerant randomized distributed algorithms. We consider the algorithms that follow the ideas of Ben-Or [3]. Interestingly, these algorithms were analyzed in [14,12] where probabilistic reasoning was done using the probabilistic model checker PRISM [13] for systems of 10-20 processes, while only safety was verified in the parameterized setting using Cadence SMV. From a different perspective, these algorithms extend asynchronous threshold-guarded distributed algorithms from [10,9] with two features (i) a random choice (coin toss), and (ii) repeated executions of the same algorithm until it converges (with probability 1).

A prominent example of a consensus algorithm is [3]. It circumvents the impossibility of asynchronous consensus [7] by relaxing the termination requirement to almost-sure termination, *i.e.*, termination with probability 1. This is achieved by a multi-round fault-tolerant distributed algorithm given in Figure 1. Here processes execute an infinite sequence of asynchronous loop iterations, which are called rounds r. Each round consists of two stages where they first exchange messages tagged R, wait until the number of received messages reaches a certain threshold (given as expression over parameters in line 5) and then exchange messages tagged P. If n is the number of processes in the system, among which at most t are faulty, then all the thresholds, that is, $n-t$ and $(n+t)/2$ and $t+1$, should ensure that no two correct processes ever decide on different values, even if up to t Byzantine faulty processes send conflicting information. At the end of a round, if there is no "strong majority" for a value, that is, $(n+t)/2$ received messages, a process picks a new value randomly in line 16.

While these complex threshold expressions can be dealt with using the methods in [9], several challenges remain. Basically, the technique in [9] can be used to verify one iteration of the round from Figure 1 only. However, consensus algorithms should prevent that there are no two rounds r and r' such that a process decides 0 in r and another decides 1 in r'. This calls for a compositional approach that allows one to compose verification results for individual rounds. A challenge in the composition is that distributed algorithms implement "asynchronous rounds", that is, during the run processes may be in different rounds at the same time.

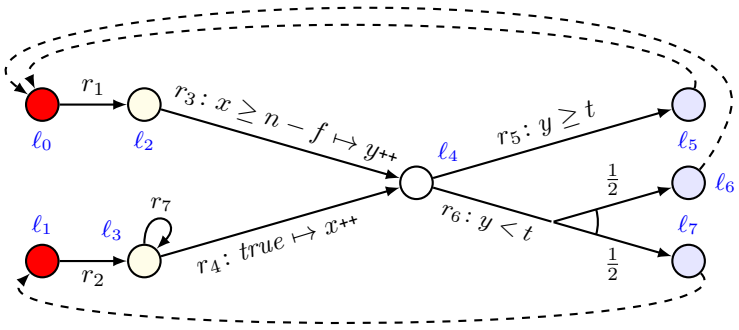

Fig. 2. Example of a probabilistic threshold automaton.

In addition, the combination of distributed aspects and probabilities makes reasoning difficult. Quoting Lehmann and Rabin [15], "proofs of correctness for probabilistic distributed systems are extremely slippery". This advocates the development of automated verification techniques for probabilistic properties of randomized distributed algorithms in the parameterized setting.

Contributions. We lift threshold automata to round-based algorithms with coin toss transitions. For the new framework we achieve the following:

1. For safety verification we introduce a method for compositional round-based reasoning. This allows us to invoke a reduction similar to the one in [6]. We highlight necessary fairness conditions on individual rounds. This provides us with specifications to be checked on a one-round automaton.
2. For probabilistic liveness verification, we explain how to reduce to proving termination with positive probability within a fixed number of rounds. To do so, we justify the restriction to round-rigid adversaries, that is, adversaries that respect the round ordering. In contrast to existing work that proves almost-sure termination for fixed number of participants, these are the first parameterized model checking results for probabilistic properties.
3. We checked the specifications that emerge from points 1. and 2. and thus verify challenging benchmarks in the parameterized setting. We verify Ben-Or's [3] and Bracha's [5] classic consensus algorithms, and the more recent algorithms 2-set agreement [18], and RS-Bosco [20].

2 Overview

We introduce probabilistic threshold automata to model randomized threshold-based algorithms. An example of such an automaton is given in Figure 2. Nodes represent local states (or locations) of processes, which move along the labeled edges or forks. Edges and forks are called rules. Labels have the form $\varphi \mapsto u$,

meaning that a process can move along the edge only if φ evaluates to true, and this is followed by the update u of shared variables. Additionally, each tine of a fork is labeled with a number in the $[0, 1]$ interval, representing the probability of a process moving along the fork to end up at the target location of the tine.

If we ignore the dashed arrows in Figure 2, a threshold automaton captures the behavior of a process in one round. The dashed edges, called round switch rules, encode how a process, after finishing a round, starts the next one. Since in the algorithm from Figure 1 processes iterate through the rounds r and send and receive messages for the round r, in the semantics of the automaton we will introduce a copy of all variables (counters for locations and shared variables) for each round. Because there are infinitely many rounds, this means we have infinitely many variables.

Threshold automata without probabilistic forks and round switching rules can be automatically checked for safety and liveness [9]. However, adding forks and round switches is required to adequately model randomized distributed algorithms.

In order to overcome the issue of infinitely many rounds, we prove in Section 5 and Section 6 that we can verify probabilistic threshold automata by analyzing a one-round automaton, that fits in the framework of [9]. We prove that we can reorder transitions of any fair execution such that their round numbers are in an increasing order. The obtained ordered execution is stutter equivalent with the original one, and thus, they satisfy the same $\mathsf{LTL}_{\setminus \mathsf{X}}$ properties over the atomic propositions describing only one round. In other words, our targeted concurrent systems can be transformed to a sequential composition of one-round systems.

The main problem with isolating a one-round system is that our specifications often talk about at least two different rounds. In this case we need to use round invariants that imply the specifications. For example, if we want to verify agreement, we have to check whether two processes decide different values, possibly in different rounds. We do this in two steps: (i) we check the round invariant that no process changes its decision from round to round, and (ii) we check that within a round no two processes disagree.

Finally, verifying almost-sure termination under round-rigid adversaries is in nature very different from proving safety specification which are properties that should hold on every path, and involve no probabilities. It thus calls for distinct arguments. Our methodology follows the lines of the manual proof of Ben Or's consensus algorithm by Aguilera and Toueg [1]. However, our arguments are not specific to Ben Or's algorithm, and we apply it to other randomized distributed algorithms (see Section 8). Compared to their paper-and-pencil proof, the threshold automata framework required us to provide a more formal setting and a more informative proof, also pinpointing the needed hypothesis. The crucial parts of our proof are automatically checked by the model checker ByMC. Hence the established correctness stands on less slippery ground, which addresses the mentioned concerns of Lehmann and Rabin.

3 The Probabilistic Threshold Automata Framework

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

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

A rule r is a tuple $(\mathit{from}, \delta_{to}, \varphi, \boldsymbol{u})$ where $\mathit{from} \in \mathcal{L}$ is the *source* location, $\delta_{to} \in \mathsf{Dist}(\mathcal{L})$ is a probability distribution over the *destination* locations, $\boldsymbol{u} \in \mathbb{N}_0^{|\Gamma|}$ is the *update vector*, and φ is a *guard* of the form $b \cdot x \geq \bar{a} \cdot \mathbf{p}^\intercal + a_0$ or $b \cdot x < \bar{a} \cdot \mathbf{p}^\intercal + a_0$, 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 vector or all paremeters. If $r.\delta_{to}$ is a Dirac distribution, i.e., there exists $\ell \in \mathcal{L}$ such that $\delta_{to}(\ell) = 1$, we call r a *Dirac rule*, and sometimes write it as $(\mathit{from}, \ell, \varphi, \boldsymbol{u})$. Destination locations of non-Dirac rules are final locations.

Probabilistic threshold automata allow one to model algorithms with successive identical rounds. Informally, a round happens between border locations and final locations, then round switch rules let processes move from final locations of a given round to border locations of the next round. From each border location there is exactly one Dirac rule to an initial location, and it has a form $(\ell, \ell', \mathtt{true}, \mathbf{0})$ where $\ell \in \mathcal{B}$ and $\ell' \in \mathcal{I}$. As $|\mathcal{B}| = |\mathcal{I}|$, one can think of border locations as copies of initial locations. It remains to model from which final locations to which border location (that is, initial for the next round) processes move. This is done by *round switch rules*. These rules are deterministic, and can be described by a function $\rho\colon \mathcal{F} \to \mathcal{B}$, or equivalently as Dirac rules $(\ell, \ell', \mathtt{true}, \mathbf{0})$ with $\ell \in \mathcal{F}$ and $\ell' \in \mathcal{B}$. The set of round switch rules is denoted by $\mathcal{S} \subseteq \mathcal{R}$.

We assume the following structure for probabilistic threshold automata. A location is a border location if and only if all the incoming edges are round switch edges. Similarly, a location is final if and only if there is only one outgoing edge and it is a round switch rule.

Example 1. In Figure 2 we have a PTA with border locations $\mathcal{B} = \{\ell_0, \ell_1\}$, initial locations $\mathcal{I} = \{\ell_2, \ell_3\}$, and final locations $\mathcal{F} = \{\ell_5, \ell_6, \ell_7\}$. The only rule that is not Dirac rule is r_6, and round switch rules are represented by dashed arrows.

3.1 Probabilistic Counter Systems

Given a probabilistic threshold automaton PTA, we define its semantics, called the *probabilistic counter system* $\mathsf{Sys}(\mathsf{PTA})$, to be the infinite-state MDP $(\Sigma, I, \mathsf{Act}, \Delta)$, where Σ is the set of configurations for PTA among which $I \subseteq \Sigma$ are initial, the set of actions is $\mathsf{Act} = \mathcal{R} \times \mathbb{N}_0$ and $\Delta\colon \Sigma \times \mathsf{Act} \to \mathsf{Dist}(\Sigma)$ is the probabilistic transition function.

Every resilience condition RC defines the set of *admissible parameters* $\mathbf{P}_{RC} = \{\mathbf{p} \in \mathbb{N}_0^{|\Pi|} \colon \mathbf{p} \models RC\}$. We introduce a function $N \colon \mathbf{P}_{RC} \to \mathbb{N}_0$ that maps a vector of admissible parameters to a number of modeled processes in the system.

Configurations. Every configuration $\sigma = (\boldsymbol{\kappa}, \boldsymbol{g}, \mathbf{p})$ is determined by a function $\sigma.\boldsymbol{\kappa} \colon \mathcal{L} \times \mathbb{N}_0 \to \mathbb{N}_0$ that defines values of local state counters per round, a function $\sigma.\boldsymbol{g} \colon \Gamma \times \mathbb{N}_0 \to \mathbb{N}_0$ defining shared variable values per round, and a vector $\sigma.\mathbf{p} \in \mathbb{N}_0^{|\Pi|}$ of parameter values. By $\boldsymbol{g}[k]$ we denote the vector $(\boldsymbol{g}[x,k])_{x \in \Gamma}$ of shared variables in a round k, and similarly by $\boldsymbol{\kappa}[k]$ we denote the vector $(\boldsymbol{\kappa}[\ell,k])_{\ell \in \mathcal{L}}$ of local state counters in a round k.

A configuration $\sigma = (\boldsymbol{\kappa}, \boldsymbol{g}, \mathbf{p})$ is *initial* if for every $x \in \Gamma$ and $k \in \mathbb{N}_0$ we have $\sigma.\boldsymbol{g}[x,k] = 0$, if $\sum_{\ell \in \mathcal{B}} \sigma.\boldsymbol{\kappa}[\ell,0] = N(\mathbf{p})$, and finally if $\sigma.\boldsymbol{\kappa}[\ell,k] = 0$, for every $(\ell,k) \in (\mathcal{L} \setminus \mathcal{B}) \times \{0\} \cup \mathcal{L} \times \mathbb{N}$.

We say that a threshold guard $\varphi : b \cdot x \geq \bar{a} \cdot \mathbf{p}^\intercal + a_0$ evaluates to true in a configuration σ for a round k, and write $\sigma, k \models \varphi$, if $b \cdot \sigma.\boldsymbol{g}[x,k] \geq \bar{a} \cdot \sigma.\mathbf{p}^\intercal + a_0$. Similarly we define when a guard of the other form, that is, $b \cdot x < \bar{a} \cdot \mathbf{p}^\intercal + a_0$, evaluates to true in σ for a round k.

Actions. Actions are induced by the rules of PTA. They are of the form $\alpha = (r,k) \in \mathcal{R} \times \mathbb{N}_0$ and stand for the application of rule r to some process[4] in round k. We use notation $\alpha.\mathit{from}$ for $r.\mathit{from}$, $\alpha.\varphi$ for $r.\varphi$, etc. If r is a Dirac rule, we say α is a Dirac action.

An action $\alpha = (r,k)$ is *unlocked* in configuration σ, if its guard evaluates to true in its round, that is $\sigma, k \models \varphi$. An action $\alpha = (r,k)$ is *applicable* to a configuration σ if α is unlocked in σ, and $\sigma.\boldsymbol{\kappa}[r.\mathit{from}, k] \geq 1$. Applicability thus represents the ability to apply r to σ in round k.

Definition 1. *We introduce a partial function apply*: $\mathsf{Act} \times \mathcal{L} \times \Sigma \nrightarrow \Sigma$ *such that given an action* $\alpha = (r,k) \in \mathsf{Act}$*, a location* $\ell \in \mathcal{L}$*, and a configuration* σ*, the result* $\mathit{apply}(\alpha, \ell, \sigma)$ *is defined if and only if* α *is applicable to* σ *and* $\alpha.\delta_{to}(\ell) > 0$*. We have that* $\mathit{apply}(\alpha, \ell, \sigma) = \sigma'$ *if and only if* $\mathit{apply}(\alpha, \ell, \sigma)$ *is defined and the following holds:*

- $\sigma'.\boldsymbol{g}[k] = \sigma.\boldsymbol{g}[k] + \alpha.\boldsymbol{u}$*, and* $\sigma'.\boldsymbol{g}[k'] = \sigma.\boldsymbol{g}[k']$*, for every round* $k' \neq k$*,*
- $\sigma'.\mathbf{p} = \sigma.\mathbf{p}$*,*
- *if* $r \in \mathcal{R} \setminus \mathcal{S}$ *and* $\alpha.\mathit{from} \neq \ell$*, then*
  - $\sigma'.\boldsymbol{\kappa}[\alpha.\mathit{from}, k] = \sigma.\boldsymbol{\kappa}[\alpha.\mathit{from}, k] - 1$*,*
  - $\sigma'.\boldsymbol{\kappa}[\ell, k] = \sigma.\boldsymbol{\kappa}[\ell, k] + 1$*,*
  - $\forall \ell \in \mathcal{L} \setminus \{\alpha.\mathit{from}, \ell\}$*,* $\sigma'.\boldsymbol{\kappa}[\ell, k] = \sigma.\boldsymbol{\kappa}[\ell, k]$*, and*
  - $\sigma'.\boldsymbol{\kappa}[k'] = \sigma.\boldsymbol{\kappa}[k']$*, for all rounds* $k' \neq k$

  if $r \in \mathcal{R} \setminus \mathcal{S}$ *and* $\alpha.\mathit{from} = \ell$*, then* $\sigma'.\boldsymbol{\kappa} = \sigma.\boldsymbol{\kappa}$*,*

  if $r \in \mathcal{S}$*, then*
  - $\sigma'.\boldsymbol{\kappa}[\alpha.\mathit{from}, k] = \sigma.\boldsymbol{\kappa}[\alpha.\mathit{from}, k] - 1$*,*
  - $\sigma'.\boldsymbol{\kappa}[\ell, k+1] = \sigma.\boldsymbol{\kappa}[\ell, k+1] + 1$*, and*
  - $\sigma'.\boldsymbol{\kappa}[\ell', k'] = \sigma.\boldsymbol{\kappa}[\ell', k']$*, for all* $(\ell', k') \in \mathcal{L} \times \mathbb{N}_0 \setminus \{(\alpha.\mathit{from}, k), (\ell, k+1)\}$.

[4] Note that actual id of the process is irrelevant.

A probabilistic transition function Δ is defined such that for every two configurations σ and σ' and for every action α applicable to σ, we have

$$\Delta(\sigma,\alpha)(\sigma') = \begin{cases} \alpha.\delta_{to}(\ell) > 0, & \text{if } \mathit{apply}(\alpha,\ell,\sigma) = \sigma', \\ 0, & \text{otherwise.} \end{cases}$$

3.2 Non-probabilistic Counter Systems

Non-probabilistic threshold automata were defined in [10], and they can be seen as probabilistic threshold automata where all rules are Dirac rules.

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

Definition 2. *Given a* $\mathsf{PTA} = (\mathcal{L}, \mathcal{V}, \mathcal{R}, RC)$*, its* underlying (non-probabilistic) threshold automaton *is* $\mathsf{TA}_{\mathsf{PTA}} = (\mathcal{L}, \mathcal{V}, \mathcal{R}_{np}, RC)$ *where the set of rules* $\mathcal{R}_{np}$ *is defined as* $\{r_\ell = (\mathit{from}, \ell, \varphi, \boldsymbol{u}) \colon r = (\mathit{from}, \delta_{to}, \varphi, \boldsymbol{u}) \in \mathcal{R} \wedge \ell \in \mathcal{L} \wedge \delta_{to}(\ell) > 0\}$.

We write TA instead of $\mathsf{TA}_{\mathsf{PTA}}$ when it is clear which PTA we refer to. Note that every rule from $\mathcal{R}_{np}$ corresponds to exactly one rule in $\mathcal{R}$, and for every rule in $\mathcal{R}$ there is at least one corresponding rule in $\mathcal{R}_{np}$ (and exactly one for Dirac rules).

If we understand a TA as a PTA where all rules are Dirac rules, we can define transitions using the partial function *apply* from Definition 1 in order to obtain an infinite (non-probabilistic) counter system, which we denote by $\mathsf{Sys}_\infty(\mathsf{TA})$. Moreover, since $\mathcal{R} = \mathcal{R}_{np}$, actions exactly match transitions. We obtain σ' by applying $t = (r, k)$ to σ, and write this as $\sigma' = t(\sigma)$, if and only if for the destination location ℓ of r holds that $\mathit{apply}(t, \ell, \sigma) = \sigma'$.

Equivalently, starting from a probabilistic counter system, one can also define a non-probabilistic counter part $\mathsf{Sys}_{np}(\mathsf{PTA})$. As the definitions are equivalent, we can use both interchangeably.

A (finite or infinite) sequence of transitions is called *schedule*, and it is often denoted by τ. A schedule $\tau = t_1, t_2, \ldots, t_{|\tau|}$ is applicable to a configuration σ if there exists a sequence of configurations $\sigma = \sigma_0, \sigma_1, \ldots, \sigma_{|\tau|}$ such that for every $1 \leq i \leq |\tau|$ we have that t_i is applicable to σ_{i-1} and $\sigma_i = t_i(\sigma_{i-1})$. A *path* is an alternating sequence of configurations and transitions, for example $\sigma_0, t_1, \sigma_1, \ldots, t_{|\tau|}, \sigma_{|\tau|}$, such that for every t_i, $1 \leq i \leq |\tau|$, in the 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 and a schedule $\tau = t_1, t_2, \ldots, t_{|\tau|}$, a path $\sigma_0, t_1, \sigma_1, \ldots, t_{|\tau|}, \sigma_{|\tau|}$ where $t_i(\sigma_{i-1}) = \sigma_i$, $1 \leq i \leq |\tau|$, we denote by $\mathsf{path}(\sigma_0, \tau)$. Similarly we define an infinite schedule $\tau = t_1, t_2, \ldots$, and an infinite path $\sigma_0, t_1, \sigma_1, \ldots$, also denoted by $\mathsf{path}(\sigma_0, \tau)$. An infinite path is *fair* if whenever some transition is applicable, it will eventually be performed.

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

3.3 Adversaries

Definition 3. *Let* Paths *be the set of all finite paths in* $\mathsf{Sys}(\mathsf{PTA})$*. An* adversary *is a function* $\mathtt{s} : \mathsf{Paths} \to \mathsf{Act}$*, that given a path* π *selects an action applicable to the last configuration of* π*.*

Given a configuration σ and an adversary $\mathtt{s}$, we generate a family of paths, depending of the outcomes of non-Dirac transitions. We denote this set by $\mathsf{paths}(\sigma, \mathtt{s})$. An adversary $\mathtt{s}$ is *fair* if all paths in $\mathsf{paths}(\sigma, \mathtt{s})$ are fair.

As usual, the MDP $\mathsf{Sys}(\mathsf{PTA})$ together with an initial configuration σ and an adversary $\mathtt{s}$ induce a Markov chain, written $\mathcal{M}_{\mathtt{s}}^{\sigma}$. In the sequel, we write $\mathbb{P}_{\mathtt{s}}^{\sigma}$ for the probability measure over infinite paths starting at σ in the latter Markov chain.

Definition 4. *An adversary* $\mathtt{s}$ *is* round-rigid *if it is fair, and if every sequence of actions it produces can be written as* $\mathtt{s}_1 \cdot \mathtt{s}_1^p \cdot \mathtt{s}_2 \cdot \mathtt{s}_2^p ...$*, where for every* $k \in \mathbb{N}_0$*, we have that* $\mathtt{s}_k$ *contains only Dirac transitions of round* k*, and* $\mathtt{s}_k^p$ *contains only non-Dirac transitions of round* k*.*
We denote the set of all round-rigid adversaries by $\mathcal{A}^R$*.*

3.4 Atomic Propositions and Stutter Equivalence

The atomic propositions we consider describe non-emptiness of the locations from $\mathcal{L} \setminus \mathcal{B}$ in a specific round. Formally, the set of all such propositions for a round $k \in \mathbb{N}_0$ is denoted by $\mathrm{AP}_k = \{\mathrm{p}(\ell, k) \colon \ell \in \mathcal{L} \setminus \mathcal{B}\}$. For every round k we define a labeling function $\lambda_k \colon \Sigma \to 2^{\mathrm{AP}_k}$ such that $\mathrm{p}(\ell, k) \in \lambda_k(\sigma)$ if and only if $\sigma.\boldsymbol{\kappa}[\ell, k] > 0$, i.e., if the location ℓ is nonempty in round k in σ.

For a path $\pi = \sigma_0, t_1, \sigma_1, \ldots, t_n, \sigma_n$, $n \in \mathbb{N}$, and a round k, a trace $\mathrm{trace}_k(\pi)$ w.r.t. the labeling function λ_k is the sequence $\lambda_k(\sigma_0)\lambda_k(\sigma_1)\ldots\lambda_k(\sigma_n)$. Similarly, if a path is infinite $\pi = \sigma_0, t_1, \sigma_1, t_2, \sigma_2, \ldots$, then $\mathrm{trace}_k(\pi) = \lambda_k(\sigma_0)\lambda_k(\sigma_1)\ldots$.

We say that two finite traces are stutter equivalent w.r.t. AP_k, denoted $\mathrm{trace}_k(\pi_1) \triangleq \mathrm{trace}_k(\pi_2)$, if there is a finite sequence $A_0 A_1 \ldots A_n \in (2^{\mathrm{AP}_k})^+$, $n \in \mathbb{N}_0$, such that both $\mathrm{trace}_k(\pi_1)$ and $\mathrm{trace}_k(\pi_2)$ are contained in the language given by the regular expression $A_0^+ A_1^+ \ldots A_n^+$. If traces are of π_1 and π_2 are infinite, then stutter equivalence $\mathrm{trace}_k(\pi_1) \triangleq \mathrm{trace}_k(\pi_2)$ is defined in the standard way [2]. To simplify notation, we say that paths π_1 and π_2 are stutter equivalent w.r.t. AP_k, and write $\pi_1 \triangleq_k \pi_2$, instead of referring to specific path traces.

Two counter systems C_0 and C_1 are stutter equivalent w.r.t. AP_k, written $C_0 \triangleq_k C_1$, if for every path π from C_i there is a path π' from C_{1-i} such that $\pi \triangleq_k \pi'$, for $i \in \{0, 1\}$.

4 Consensus Properties and their Verification

Probabilistic consensus consists of safety specifications and an almost-sure termination requirement. We discuss here the specifications of Ben-Or's algorithm shown in Figure 1. Every correct process has an initial value from $\{0, 1\}$, and must decide a value, either 0 or 1, such that:

Agreement: No two correct processes decide differently.
Validity: If all correct processes have v as the initial value, then no process decides $1 - v$.
Probabilistic wait-free termination: Under every round-rigid adversary, with probability 1 every correct process eventually decides.

Formalization. In order to formulate and analyze specifications, we partition sets $\mathcal{I}$, $\mathcal{B}$, and $\mathcal{F}$, each into two subsets, e.g., $\mathcal{I}_0$ and $\mathcal{I}_1$, and an analogous notation for the subsets of $\mathcal{B}$ and $\mathcal{F}$. Here are the restrictions for every $v \in \{0, 1\}$:

- Processes that are initially in a location $\ell \in \mathcal{I}_v$ have the initial value v.
- Rules connecting locations from $\mathcal{B}$ and $\mathcal{I}$ respect the partitioning, i.e., they connect $\mathcal{B}_v$ and $\mathcal{I}_v$.
- Similarly, rules connecting locations from $\mathcal{F}$ and $\mathcal{B}$ respect the partitioning.

We also 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 if and only if it decides v in that round.

Now we can express specifications for $\mathsf{Sys}(\mathsf{PTA})$ as follows:

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

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

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

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

Probabilistic wait-free termination: For every round-rigid adversary $\mathtt{s}$

$$\mathbb{P}_{\mathtt{s}}\big(\bigvee_{k \in \mathbb{N}_0} \bigvee_{v \in \{0,1\}} \mathbf{G} \bigwedge_{\ell \in \mathcal{F} \setminus \mathcal{D}_v} \boldsymbol{\kappa}[\ell, k] = 0\big) \quad = \quad 1 \quad (3)$$

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

In Section 5 we formalize safety specifications and reduce them to single-round specifications. In Section 6 we reduce verification of single-round specifications in the infinite counter system to their verification in a one-round counter system. In Section 7 we discuss our approach to probabilistic termination.

5 Reduction to Specifications with one Round Quantifier

While Agreement contains two round variables, Validity talks about round 0 and a round $k \in \mathbb{N}_0$. Thus, both of these specifications involve two round numbers.

Our goal is to reduce reasoning from unboundedly many rounds to one round. Therefore, properties are only allowed to talk about one round number. In this Section we show how to check formulas (1) and (2) by checking properties that describe one round. Namely, we introduce two properties, round invariants (4) and (5), and prove that they imply our two specifications.

Moreover, we define the Relay property, that is necessary for proving Probabilistic wait-free termination. Since this property also involves two round numbers, we introduce another round invariant (**??**), and prove that together with formula (5) it implies Relay.

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

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

The second round invariant claims that in every round in every path, if no process starts a round with a value v, then no process terminates that round with value v. Formally, the following formula holds:

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

The benefit of analyzing these two formulas instead of (1) and (2) lies in the fact that formulas (4) and (5) describe properties of only one round in a path, Next we want to prove that formulas (4) and (5) imply formulas (1) and (2).

Let us first give some useful properties of $\mathsf{Sys}_\infty(\mathsf{TA})$.

Lemma 1 (Round Switch). *For every $\mathsf{Sys}_\infty(\mathsf{TA})$ and every $v \in \{0, 1\}$:*

$$(\forall k \in \mathbb{N}_0)\ \mathbf{A}\,(\mathbf{G} \bigwedge_{\ell \in \mathcal{F}_v} \boldsymbol{\kappa}[\ell, k] = 0 \quad \rightarrow \quad \mathbf{G} \bigwedge_{\ell' \in \mathcal{I}_v} \boldsymbol{\kappa}[\ell', k+1] = 0). \tag{6}$$

Proof. By definitions of $\mathcal{F}_v$, $\mathcal{B}_v$ and $\mathcal{I}_v$, we have that

$$(\forall k \in \mathbb{N}_0)\ \mathbf{A}\,(\mathbf{G} \bigwedge_{\ell \in \mathcal{F}_v} \boldsymbol{\kappa}[\ell, k] = 0 \rightarrow \mathbf{G} \bigwedge_{\ell'' \in \mathcal{B}_v} \boldsymbol{\kappa}[\ell'', k+1] = 0), \text{ and}$$

$$(\forall k \in \mathbb{N}_0)\ \mathbf{A}\,(\mathbf{G} \bigwedge_{\ell'' \in \mathcal{B}_v} \boldsymbol{\kappa}[\ell'', k+1] = 0 \rightarrow \mathbf{G} \bigwedge_{\ell' \in \mathcal{I}_v} \boldsymbol{\kappa}[\ell', k+1] = 0).$$

The two formulas together yield the required one for both values of v. □

Lemma 2. *For every $\mathsf{Sys}_\infty(\mathsf{TA})$ such that $\mathsf{Sys}_\infty(\mathsf{TA}) \models (5)$, and for every $v \in \{0, 1\}$, the following holds:*

$$(\forall k \in \mathbb{N}_0)(\forall k' \in \mathbb{N}_0)\big(k \leq k' \rightarrow \mathbf{A}\,(\mathbf{G} \bigwedge_{\ell \in \mathcal{I}_v} \boldsymbol{\kappa}[\ell, k] = 0 \rightarrow \mathbf{G} \bigwedge_{\ell' \in \mathcal{I}_v} \boldsymbol{\kappa}[\ell', k'] = 0)\big), \tag{7}$$

$$(\forall k \in \mathbb{N}_0)(\forall k' \in \mathbb{N}_0)\big(k \leq k' \rightarrow \mathbf{A}\,(\mathbf{G} \bigwedge_{\ell \in \mathcal{F}_v} \boldsymbol{\kappa}[\ell, k] = 0 \rightarrow \mathbf{G} \bigwedge_{\ell' \in \mathcal{F}_v} \boldsymbol{\kappa}[\ell', k'] = 0)\big). \tag{8}$$

Proof. Assume formula (5) holds. Note that Lemma 1 together with formula (5) gives us that globally empty initial location is a round invariant. Formally, by transitivity we have that

$$(\forall k \in \mathbb{N}_0)\ \mathbf{A}\ (\mathbf{G} \bigwedge_{\ell \in \mathcal{I}_v} \boldsymbol{\kappa}[\ell, k] = 0\ \rightarrow\ \mathbf{G} \bigwedge_{\ell' \in \mathcal{I}_v} \boldsymbol{\kappa}[\ell', k+1] = 0). \tag{9}$$

By induction we obtain the required formula (7). Finally, by combining formulas (6), (5) and (7) we obtain formula (8). □

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

Proof. Assume $\mathsf{Sys}_\infty(\mathsf{TA}) \models (4) \wedge (5)$.

Let us first focus on formula (1), and prove that $\mathsf{Sys}_\infty(\mathsf{TA}) \models (1)$. Assume by contradiction that the formula does not hold on $\mathsf{Sys}_\infty(\mathsf{TA})$, that is, there exist rounds $k, k' \in \mathbb{N}_0$ and a path π such that:

$$\pi \models \mathbf{F} \bigvee_{\ell_0 \in \mathcal{D}_0} \boldsymbol{\kappa}[\ell_0, k] > 0\ \wedge\ \mathbf{F} \bigvee_{\ell_1 \in \mathcal{D}_1} \boldsymbol{\kappa}[\ell_1, k'] > 0. \tag{10}$$

Since by formula (10) we have $\pi \models \mathbf{F} \bigvee_{\ell_0 \in \mathcal{D}_0} \boldsymbol{\kappa}[\ell_0, k] > 0$, then from formula (4) with $v = 0$ we obtain that it also holds $\pi \models \mathbf{G} \bigwedge_{\ell \in \mathcal{F}_1} \boldsymbol{\kappa}[\ell, k] = 0$. As $\mathcal{D}_1 \subseteq \mathcal{F}_1$, we know that no process decides 1 in round k. Now formula (8) from Lemma 2 for $v = 1$ yields that $\pi \models \mathbf{G} \bigwedge_{\ell \in \mathcal{F}_1} \boldsymbol{\kappa}[\ell, k_1] = 0$ for every $k_1 \geq k$, i.e., in any round greater than k no process will ever decide 1. As by (10) we have that $\pi \models \mathbf{F} \bigvee_{\ell_1 \in \mathcal{D}_1} \boldsymbol{\kappa}[\ell_1, k'] > 0$, i.e., a process decides 1 in a round k', thus it must be that $k' < k$.

Now we consider the other part of formula (10), i.e., $\pi \models \mathbf{F} \bigvee_{\ell_1 \in \mathcal{D}_1} \boldsymbol{\kappa}[\ell_1, k'] > 0$. By following the analogous analysis we conclude that it must be that $k < k'$. This brings us to the contradiction with $k' < k$, which proves the first part of the statement, that violation of (4) and (5) implies violation of (1).

Next we focus on formula (2), and prove by contradiction that it must hold. We start by assuming that the formula does not hold, that is, there exists a round k and a path π such that no process starts the first round of π with value v and eventually in a round k a process decides v. Formally,

$$\pi \models \bigwedge_{\ell \in \mathcal{I}_v} \boldsymbol{\kappa}[\ell, 0] = 0\ \wedge\ \mathbf{F} \bigvee_{\ell' \in \mathcal{D}_v} \boldsymbol{\kappa}[\ell', k] > 0. \tag{11}$$

Note first that $\pi \models \bigwedge_{\ell \in \mathcal{I}_v} \boldsymbol{\kappa}[\ell, 0] = 0$ implies $\pi \models \mathbf{G} \bigwedge_{\ell \in \mathcal{I}_v} \boldsymbol{\kappa}[\ell, 0] = 0$. Then, since $\mathsf{Sys}_\infty(\mathsf{TA}) \models (5)$, that is, since formula (5) holds, we have that $\pi \models \mathbf{G} \bigwedge_{\ell' \in \mathcal{F}_v} \boldsymbol{\kappa}[\ell', 0] = 0$. Then by formula (8) we have that for every $k \in \mathbb{N}_0$ it holds $\pi \models \mathbf{G} \bigwedge_{\ell' \in \mathcal{F}_v} \boldsymbol{\kappa}[\ell', k] = 0$. Since $\mathcal{D}_v \subseteq \mathcal{F}_v$, we also have that $\pi \models \mathbf{G} \bigwedge_{\ell' \in \mathcal{D}_v} \boldsymbol{\kappa}[\ell', k] = 0$. As this contradicts our assumption from (11) that $\pi \models \mathbf{F} \bigvee_{\ell' \in \mathcal{D}_v} \boldsymbol{\kappa}[\ell', k] > 0$, it proves the second part of the statement, that violation of (4) and (5) implies violation of (2). □

6 Reduction to Single-Round Counter System

Given a property describing one round, our goal is to prove that there is a counterexample to the property in the infinite system if and only if there is a counterexample in a single-round system. This is formulated in Theorem 1, and it allows us to use the existing technique from [9] on a one-round system.

The proof idea contains two parts. Firstly, in Section 6.1 we prove that one can exchange an arbitrary finite schedule with a round-rigid one, while preserving atomic propositions of a fixed round. We show that swapping two neighboring transitions that do not respect the order in an execution, gives us a legal stutter equivalent execution, i.e., an execution satisfying the same $\mathsf{LTL}_{\setminus\mathsf{X}}$ properties.

Secondly, in Section 6.2 we extend this reasoning to infinite schedules, and lift it from schedules to transition systems. The main idea is to do inductive and compositional reasoning over the rounds. In order to do so, we require that round boundaries are well-defined, which is the case if every round that is started is also finished; a property we can automatically check for fair schedules. In more detail, regarding propositions for one round, we show in Lemma 11 that the multi-round transition system is stutter equivalent to a single-round transition system. This holds under the assumption that all fair executions of a one-round transition system terminate, and this can be checked using the technique from [9]. As stutter equivalence of systems implies preserving $\mathsf{LTL}_{\setminus\mathsf{X}}$ properties, this is enough to prove the main goal of the section.

6.1 Reduction from arbitrary schedules to round-rigid schedules

Definition 5. *A schedule $\tau = (r_1, k_1) \cdot (r_2, k_2) \cdot \ldots \cdot (r_m, k_m)$, $m \in \mathbb{N}_0$, is called* round-rigid *if for every $1 \leq i < j \leq m$, we have $k_i \leq k_j$.*

The following lemma follows directly from the definitions of transitions, and it gives us the most important transition invariants.

Lemma 3. *Let σ be a configuration and let $t = (r, k)$ be a transition. If $\sigma' = t(\sigma)$ then the following holds:*

(a) $\sigma'.\boldsymbol{g}[k'] = \sigma.\boldsymbol{g}[k']$, *for every round* $k' \neq k$,
(b) $\sigma'.\boldsymbol{\kappa}[k'] = \sigma.\boldsymbol{\kappa}[k']$, *for every round* $k' \in \mathbb{N}_0 \setminus \{k, k+1\}$,
(c) $\sigma'.\boldsymbol{\kappa}[\ell, k'] = \sigma.\boldsymbol{\kappa}[\ell, k']$, *for every round* $k' \neq k$ *and every location* $\ell \in \mathcal{L} \setminus \mathcal{B}$,
(d) $\sigma'.\boldsymbol{\kappa}[k+1] \geq \sigma.\boldsymbol{\kappa}[k+1]$,

The following lemma establishes a central argument for inductive round-based reasoning: a transition belonging to a smaller round can always be moved before a transition of the larger round.

Lemma 4. *Let σ be a configuration, and let $t_1 = (r_1, k_1)$ and $t_2 = (r_2, k_2)$ be transitions, such that $k_1 > k_2$. If $t_1 \cdot t_2$ is applicable to σ, then $t_2 \cdot t_1$ is also applicable to σ.*

Proof. Let us denote $t_1(\sigma)$ by σ_1. As $t_1 \cdot t_2$ is applicable to σ, this means that t_1 is applicable to σ and t_2 is applicable to σ_1. By definition of applicability, this means that

$$\sigma.\boldsymbol{\kappa}[r_1.\mathit{from}, k_1] \geq 1 \quad \text{and} \quad \sigma_1.\boldsymbol{\kappa}[r_2.\mathit{from}, k_2] \geq 1, \tag{12}$$

and additionally we have that $\sigma, k_1 \models t_1.\varphi$ and $\sigma_1, k_2 \models t_2.\varphi$.

We show that $t_2 \cdot t_1$ is applicable to σ by showing that: (i) t_2 is applicable to σ, and (ii) t_1 is applicable to $t_2(\sigma)$.

(i) First we need to show that $\sigma.\boldsymbol{\kappa}[r_2.\mathit{from}, k_2] \geq 1$ and $\sigma, k_2 \models t_2.\varphi$.

As $\sigma_1 = t_1(\sigma)$ and $k_2 < k_1$, by Lemma 3(b) we have $\sigma_1.\boldsymbol{\kappa}[r_2.\mathit{from}, k_2] = \sigma.\boldsymbol{\kappa}[r_2.\mathit{from}, k_2]$. From this and (12) we get that $\sigma.\boldsymbol{\kappa}[r_2.\mathit{from}, k_2] \geq 1$.

Note that evaluation of the guard $t_2.\varphi$ depends only on the values of shared variables $\sigma.\boldsymbol{g}[k_2]$ in round k_2 and parameter values $\sigma.\mathbf{p}$. As $\sigma_1 = t_1(\sigma)$ and $k_1 > k_2$, from Lemma 3(a) we have that $\sigma.\boldsymbol{g}[k_2] = \sigma_1.\boldsymbol{g}[k_2]$. Recall that $\sigma_1, k_2 \models t_2.\varphi$, and thus it must be the case that also $\sigma, k_2 \models t_2.\varphi$. This shows that t_2 is applicable to σ.

(ii) Let $\sigma_2 = t_2(\sigma)$. Next we show that t_1 is applicable to σ_2. Using the same reasoning as in (i), we prove that $\sigma_2.\boldsymbol{\kappa}[r_1.\mathit{from}, k_1] \geq 1$ and that $\sigma_2, k_1 \models t_1.\varphi$.

As $\sigma_2 = t_2(\sigma)$ and $k_2 < k_1$, Lemma 3(b) and 3(d) yield $\sigma_2.\boldsymbol{\kappa}[r_1.\mathit{from}, k_1] \geq \sigma.\boldsymbol{\kappa}[r_1.\mathit{from}, k_1]$. Together with (12) we obtain that $\sigma_2.\boldsymbol{\kappa}[r_1.\mathit{from}, k_1] \geq 1$.

To this end, we show that $\sigma_2, k_1 \models t_1.\varphi$. Since $\sigma_2 = t_2(\sigma)$ and $k_1 > k_2$, by Lemma 3(a) we know that $\sigma.\boldsymbol{g}[k_1] = \sigma_2.\boldsymbol{g}[k_1]$. Since by the initial assumption we have $\sigma, k_1 \models t_1.\varphi$, and evaluation of the guard only depends on shared variable values and parameter values, then it also holds $\sigma_2, k_1 \models t_1.\varphi$. □

Lemma 5. *Let σ be a configuration, let $t_1 = (r_1, k_1)$ and $t_2 = (r_2, k_2)$ be transitions such that $k_1 > k_2$. If $t_1 \cdot t_2$ is applicable to σ, then the following holds:*

(a) Both $t_1 \cdot t_2$ and $t_2 \cdot t_1$ reach the same configuration, i.e., $t_1 \cdot t_2(\sigma) = t_2 \cdot t_1(\sigma)$.
(b) For every $k \in \mathbb{N}_0$ we have $\mathsf{path}(\sigma, t_1 \cdot t_2) \triangleq_k \mathsf{path}(\sigma, t_2 \cdot t_1)$.

Proof. Note that since $t_1 \cdot t_2$ is applicable to σ, we also have that $t_2 \cdot t_1$ is applicable to σ by Lemma 4, since $k_1 > k_2$.

(a) When a transition is applied to a configuration, the obtained configuration has the same parameter values, and counters and global variables are incremented or decremented depending on the transition (and independently of the initial configuration). For any configuration $(\boldsymbol{\kappa}, \boldsymbol{g}, \mathbf{p})$, we can write $t_i(\boldsymbol{\kappa}, \boldsymbol{g}, \mathbf{p}) = (\boldsymbol{\kappa} + \boldsymbol{u}_i, \boldsymbol{g} + \boldsymbol{v}_i, \mathbf{p})$ for $i \in \{1, 2\}$, and some vectors $\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{v}_1, \boldsymbol{v}_2$ of integers. By only using commutativity of addition and subtraction, we obtain $t_1 \cdot t_2(\sigma) = (\boldsymbol{\kappa} + \boldsymbol{u}_1 + \boldsymbol{u}_2, \boldsymbol{g} + \boldsymbol{v}_1 + \boldsymbol{v}_2, \mathbf{p}) = (\boldsymbol{\kappa} + \boldsymbol{u}_2 + \boldsymbol{u}_1, \boldsymbol{g} + \boldsymbol{v}_2 + \boldsymbol{v}_1, \mathbf{p}) = t_2 \cdot t_1(\sigma)$.

(b) Let $\sigma_1 = t_1(\sigma)$, let $\sigma_2 = t_2(\sigma)$, and $\sigma_3 = t_1 \cdot t_2(\sigma)$. Then $\mathrm{trace}_k(\mathsf{path}(\sigma, t_1 \cdot t_2)) = \lambda_k(\sigma)\lambda_k(\sigma_1)\lambda_k(\sigma_3)$, and $\mathrm{trace}_k(\mathsf{path}(\sigma, t_2 \cdot t_1)) = \lambda_k(\sigma)\lambda_k(\sigma_2)\lambda_k(\sigma_3)$. We consider three cases: (i) $k \neq k_1$ and $k \neq k_2$, (ii) $k = k_1$, and (iii) $k = k_2$.
(i) In this case, by Lemma 3(c), we have $\lambda_k(\sigma) = \lambda_k(\sigma_1) = \lambda_k(\sigma_2) = \lambda_k(\sigma_3)$. Thus, both traces are $\lambda_k(\sigma)\lambda_k(\sigma)\lambda_k(\sigma)$, and they are clearly stutter equivalent.

(ii) Since $k = k_1 > k_2$, then again by Lemma 3(c) we have that $\lambda_k(\sigma_1) = \lambda_k(\sigma_3)$ and $\lambda_k(\sigma) = \lambda_k(\sigma_2)$. Thus, $\text{trace}_k(\mathsf{path}(\sigma, t_1 \cdot t_2)) = \lambda_k(\sigma)\lambda_k(\sigma_3)\lambda_k(\sigma_3)$, and $\text{trace}_k(\mathsf{path}(\sigma, t_2 \cdot t_1)) = \lambda_k(\sigma)\lambda_k(\sigma)\lambda_k(\sigma_3)$, and the traces are stutter equivalent.
(iii) The last case is analogous to the previous one. □

The following lemma tells us that adding or removing transitions of a round different from k results in a k-stutter equivalent path.

Lemma 6. *Let σ be a configuration and let $t_1 = (r_1, k_1)$ and $t_2 = (r_2, k_2)$ be transitions such that $t_1 t_2$ is appllicable to σ. Then the following holds:*

(a) $\mathsf{path}(\sigma, t_1 t_2) \triangleq_k \mathsf{path}(\sigma, t_1)$*, for every $k \neq k_2$, and*
(b) $\mathsf{path}(\sigma, t_1 t_2) \triangleq_k \mathsf{path}(t_1(\sigma), t_2)$*, for every $k \neq k_1$.*

Proof. It follows directly from Lemma 3 (c). □

The following reduction theorem shows that every schedule can be re-ordered into a round-rigid schedule that for all rounds k is stutter equivalent regarding $\mathsf{LTL}_{\text{-X}}$ formulas over proposition over k.

Proposition 2. *For every configuration σ and every finite schedule τ applicable to it, there is a round-rigid schedule τ' such that the following holds:*

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

Proof. Since τ is finite, the claim (a) follows from Lemma 4, the second claim follows from Lemma 5(a), and the last one from Lemma 5(b). □

The following proposition follows from the well-known result that stutter equivalent traces satisfy the same $\mathsf{LTL}_{\text{-X}}$ specifications [2, Thm. 7.92].

Proposition 3. *Fix a $k \in \mathbb{N}_0$. If π_1 and π_2 are paths such that $\pi_1 \triangleq_k \pi_2$, then for every formula φ from $\mathsf{LTL}_{\text{-X}}$ over AP_k we have $\pi_1 \models \varphi$ if and only if $\pi_2 \models \varphi$.*

We conclude that instead of reasoning about all schedules of $\mathsf{Sys}_\infty(\mathsf{TA})$, it is thus sufficient to reason about its round-rigid schedules. In the following section we will use this to simplify the verification further, namely to a single-round counter system.

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

For each PTA, we define a *round* threshold automaton that can be analyzed with the tools of [10] and [9]. Roughly speaking, we focus on one round, but also keep the border locations of the next round, where we add self-loops. We show that for specific fairness constraints, this automaton shows the same behavior as a round in $\mathsf{Sys}_\infty(\mathsf{TA})$. In Theorem 1 we prove that we can use it for analysis of non-probabilistic properties of $\mathsf{Sys}_\infty(\mathsf{TA})$.

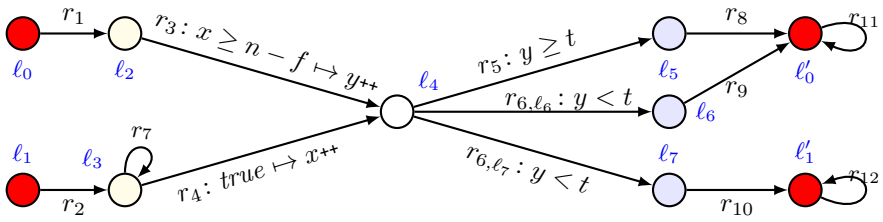

Fig. 3. The $\mathsf{TA}^{\mathrm{rd}}$ obtained from PTA in Figure 2

In the proof we restrict ourselves to fair schedules, that is, those where every transition that is applicable will eventually be performed. We also assume that every fair schedule of a single-round algorithm terminates. Under the fairness assumption we check the latter assumption with ByMC [11]. Moreover, we restrict ourselves to deadlock-free threshold automata, that is, we require that in each configuration each location has at least one outgoing edge unlocked. As we use TAs to model distributed algorithms, this is no restriction: locations in which no progress should be made unless certain thresholds are reached, typically have self-loops that are guarded with `true`. Thus for our benchmarks one can easily check whether they are deadlock-free using SMT.

Definition 6. *Given a* $\mathsf{PTA} = (\mathcal{L}, \mathcal{V}, \mathcal{R}, RC)$ *or its* $\mathsf{TA} = (\mathcal{L}, \mathcal{V}, \mathcal{R}_{np}, RC)$*, we define a* round threshold automaton TA^{rd} *to be the tuple* $(\mathcal{L} \cup \mathcal{B}', \mathcal{V}, \mathcal{R}^{rd}, RC)$*, where* $\mathcal{B}' = \{\ell' \colon \ell \in \mathcal{B}\}$ *are copies of border locations, and* $\mathcal{R}^{rd}$ *is defined as follows. We have* $\mathcal{R}^{rd} = (\mathcal{R}_{np} \setminus \mathcal{S}) \cup \mathcal{S}' \cup \mathcal{R}^{loop}$*, where modifications* $\mathcal{S}'$ *of round switch rules are*

$$\mathcal{S}' = \{(from, \ell', \mathtt{true}, \mathbf{0}) \colon (from, \ell, \mathtt{true}, \mathbf{0}) \in \mathcal{S} \text{ with } \ell' \in \mathcal{B}'\},$$

and $\mathcal{R}^{loop} = \{(\ell', \ell', \mathtt{true}, \mathbf{0}) \colon \ell' \in \mathcal{B}'\}$ *are self-looping rules at locations from* $\mathcal{B}'$*. Initial locations of* TA^{rd} *are locations from* $\mathcal{B} \subseteq \mathcal{L}$*.*

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

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

We define $\Sigma^k_{\mathcal{B}} \subseteq \Sigma^k$, for a $k \in \mathbb{N}_0$, to be the set of all configurations σ such that $\sigma.\boldsymbol{g}[x,k] = 0$ for all $x \in \Gamma$, then $\sum_{\ell \in \mathcal{B}} \sigma.\boldsymbol{\kappa}[\ell,k] = N(\mathbf{p})$, and $\sigma.\boldsymbol{\kappa}[\ell,i] = 0$ for all $(\ell,i) \in \mathcal{D} \setminus (\mathcal{B} \times \{k\})$. We call these configurations *border configurations for the round* k. The set of initial states I^k is a subset of $\Sigma^k_{\mathcal{B}}$.

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

If we do not restrict initial configurations, all these systems are isomorphic, and this is formalized in the following Lemma.

Lemma 7. *All systems $\mathsf{Sys}^k(\mathsf{TA}^{rd})$, $k \in \mathbb{N}_0$, are isomorphic to each other w.r.t. $\Sigma^k_{\mathcal{B}}$, i.e., for every $k \in \mathbb{N}_0$, if $I^k = \Sigma^k_{\mathcal{B}}$, then we have $\mathsf{Sys}^0(\mathsf{TA}^{rd}) \cong \mathsf{Sys}^k(\mathsf{TA}^{rd})$.*

We restrict our attention to *fair paths*, that is, those paths π such that for every configuration σ in π the following holds: if a transition is applicable, then it will eventually be fired in π. Moreover, we assume that all such paths in $\mathsf{Sys}^0(\mathsf{TA}^{\mathrm{rd}})$ terminate, that is, they reach a configuration with all processes in $\mathcal{B}'$. Formally, we assume that for every fair path π in $\mathsf{Sys}^0(\mathsf{TA}^{\mathrm{rd}})$ it holds that $\pi \models \mathbf{F} \bigwedge_{\ell \in \mathcal{L}} \boldsymbol{\kappa}[\ell, 0] = 0$. This can easily be checked with ByMC [11].

Lemma 8. *If all fair executions in $\mathsf{Sys}^0(\mathsf{TA}^{rd})$ terminate w.r.t. $\Sigma^0_{\mathcal{B}}$, then the same holds for $\mathsf{Sys}^k(\mathsf{TA}^{rd})$ w.r.t. $\Sigma^k_{\mathcal{B}}$, for every $k \in \mathbb{N}_0$.*

Proof. It follows directly from Lemma 7. □

We assume that in TA (and thus also in $\mathsf{TA}^{\mathrm{rd}}$) holds that for every configuration, every location, and every round, there is an unlocked outgoing edge from ℓ in that configuration and that round. Formally, for every σ, ℓ, and k, there is a rule r such that $r.\mathit{from} = \ell$ and $\sigma, k \models r.\varphi$. This property assures that systems $\mathsf{Sys}_\infty(\mathsf{TA})$ and $\mathsf{Sys}^k(\mathsf{TA}^{\mathrm{rd}})$, $k \in \mathbb{N}_0$, are deadlock-free.

In order to relate $\mathsf{Sys}_\infty(\mathsf{TA})$ and $\mathsf{Sys}^k(\mathsf{TA}^{\mathrm{rd}})$, $k \in \mathbb{N}_0$, we define the set of initial states I^0 of $\mathsf{Sys}^0(\mathsf{TA}^{\mathrm{rd}})$ to be the set I of initial states of $\mathsf{Sys}_\infty(\mathsf{TA})$, and then inductively we define I^{k+1} to be the set of final configurations of $\mathsf{Sys}^k(\mathsf{TA}^{\mathrm{rd}})$.

From now on, we fix a TA and a $\mathsf{TA}^{\mathrm{rd}}$, and if not specified differently, for every $\mathsf{Sys}^k(\mathsf{TA}^{\mathrm{rd}})$ we assume the above definition of I^k.

Lemma 9. *If all fair executions of $\mathsf{Sys}^0(\mathsf{TA}^{rd})$ w.r.t. $\Sigma^0_{\mathcal{B}}$ terminate, then for every $k \in \mathbb{N}_0$ we have that the set I^k is well-defined and all fair executions of $\mathsf{Sys}^k(\mathsf{TA}^{rd})$ terminate (w.r.t. I^k).*

Proof. We prove this claim by induction on $k \in \mathbb{N}_0$. The set $I^0 = I$ is clearly well-defined, and since $I^0 \subseteq \Sigma^0_{\mathcal{B}}$, by our assumption we have that all fair executions of $\mathsf{Sys}^0(\mathsf{TA}^{\mathrm{rd}})$ terminate. Since for every $k \in \mathbb{N}_0$ we have $I^k \subseteq \Sigma^k_{\mathcal{B}}$, by Lemma 8 we have that every fair execution of $\mathsf{Sys}^k(\mathsf{TA}^{\mathrm{rd}})$ terminates and therefore I^{k+1} is well-defined. □

Let us make here a short digression by giving a property of every $\mathsf{Sys}_\infty(\mathsf{TA})$, which is necessary for proving Lemma 11.

Lemma 10. *Let $\mathsf{Sys}_\infty(\mathsf{TA})$ be deadlock-free, fix a $k \in \mathbb{N}_0$ and let σ be a configuration in $\mathsf{Sys}_\infty(\mathsf{TA})$ with a non-empty border location in round $k+1$, i.e., $\bigvee_{\ell \in \mathcal{B}} \sigma.\boldsymbol{\kappa}[\ell, k+1] \geq 1$. Then for every configuration σ' reachable from σ, there is a transition $t = (r, f, k_1)$ with $k_1 > k$ that is applicable to σ'.*

Proof. Let σ be a configuration with a non-empty border location in round $k+1$, and let σ' be a configuration reachable from σ. Assume by contradiction that there is no transition $t = (r, f, k_1)$ with $k_1 > k$ that is applicable to σ'. Recall that every location has a non-guarded outgoing rule. Thus, it must hold that for every location ℓ we have that $\sigma'.\boldsymbol{\kappa}[\ell, k_1] = 0$, for every $k_1 > k$. This is a contradiction with the assumption that σ' is reachable from σ and $\bigvee_{\ell \in \mathcal{B}} \sigma.\boldsymbol{\kappa}[\ell, k+1] \geq 1$. □

Lemma 11. *If $\mathsf{Sys}_\infty(\mathsf{TA})$ is deadlock-free, and if all fair executions of $\mathsf{Sys}^0(\mathsf{TA}^{rd})$ w.r.t. $\Sigma^0_{\mathcal{B}}$ terminate, then for every $k \in \mathbb{N}_0$ we have $\mathsf{Sys}^k(\mathsf{TA}^{rd}) \triangleq_k \mathsf{Sys}_\infty(\mathsf{TA})$, i.e., the two systems are stutter equivalent w.r.t. AP_k.*

Proof. We prove the statement by induction on $k \in \mathbb{N}_0$.

Base case. Let us first show that $\mathsf{Sys}^0(\mathsf{TA}^{\mathrm{rd}}) \triangleq_0 \mathsf{Sys}_\infty(\mathsf{TA})$

($\Rightarrow$) Let $\pi = \mathsf{path}(\sigma, \tau)$ be a path in $\mathsf{Sys}^0(\mathsf{TA}^{\mathrm{rd}})$. We need to find a path π' from $\mathsf{Sys}_\infty(\mathsf{TA})$, such that $\pi \triangleq_k \pi'$.

If $\tau = t_1 t_2 \ldots$, then every transition t_i either exists also in TA, or it is a self-loop at the copy of a border location. Using this, we construct a schedule $\tau' = t'_1 t'_2 \ldots$ in the following way.

For every $i \in \mathbb{N}$, if t_i exists in TA, then we define t'_i to be exactly t_i, and if t'_i is a self-loop at an $\ell' \in \mathcal{B}'$, then Lemma 10 gives us that there exists a transition $\tilde{t_i}$ from a round greater than 0 that is applicable to the current configuration, and we define $t'_i = \tilde{t_i}$. Thus, $\tau' = t'_1 t'_2 \ldots$ is obtained from τ by removing certain self-looping transitions and adding transitions of rounds greater than 0. By Lemma 6 we have $\mathsf{path}(\sigma, \tau') \triangleq_0 \mathsf{path}(\sigma, \tau)$.

Now we have that $\pi' = \mathsf{path}(\sigma, \tau') \triangleq_0 \mathsf{path}(\sigma, \tau) = \pi$.

($\Leftarrow$) Let now $\pi = \mathsf{path}(\sigma, \tau)$ be a path in $\mathsf{Sys}_\infty(\mathsf{TA})$. We construct a path $\pi' = \mathsf{path}(\sigma', \tau')$ from $\mathsf{Sys}^k(\mathsf{TA}^{\mathrm{rd}})$ such that $\pi \triangleq_k \pi'$. Since $I = I^0$, we define $\sigma' = \sigma$.

Let τ_0 be the projection of τ to round 0. There are two cases to consider. First, if τ and τ_0 are either both infinite or both finite schedules, then by Lemma 6 they yield stutter equivalent paths starting in σ. Observe that by Lemma 3 counters $\boldsymbol{\kappa}[\ell, 0]$ only change due to transitions for round 0, so that the applicability of τ_0 to σ follows from the applicability of τ.. Thus, in these cases we define τ' to be τ_0.

Second, we show the construction of τ' in the case when τ is an infinite schedule and τ_0 is finite. In this case we construct τ' as infinite extension of τ_0 as follows: Note that, since TA is deadlock-free, there must exist at least one location $\ell \in \mathcal{B}_1$ that is nonempty after executing τ_0 from σ, i.e., $\tau_0(\sigma).\boldsymbol{\kappa}[\ell, 1] \geq 1$. This must also be the case in $\mathsf{Sys}^0(\mathsf{TA}^{\mathrm{rd}})$, with a difference that the nonempty location belongs to $\mathcal{B}'$, since $\mathcal{B}'$ plays the role of $\mathcal{B}_1$. If r is the self-looping rule at ℓ, then we obtain τ' by concatenating infinitely many transitions $(r, 1)$ to τ_0, i.e., $\tau' = \tau_0 (r, 1)^\omega$. Transition $(r, 1)$ does not affect atomic propositions of round 0, and thus we have stutter equivalence by Lemma 6.

Induction step. Assume that $\mathsf{Sys}^i(\mathsf{TA}^{\mathrm{rd}}) \triangleq_i \mathsf{Sys}_\infty(\mathsf{TA})$ for every $0 \leq i < k$, and let us prove that the claim holds for k.

($\Rightarrow$) Let $\pi = \mathsf{path}(\sigma, \tau)$ be a path in $\mathsf{Sys}^k(\mathsf{TA}^{\mathrm{rd}})$. We need to find a path π' from $\mathsf{Sys}_\infty(\mathsf{TA})$, such that $\pi \triangleq_k \pi'$.

First note that $\sigma \in I^k$. By definition of I^k, there exist a configuration $\sigma_0 \in I^0$ and schedules $\tau_1, \tau_2, \ldots, \tau_{k-1}$, such that every τ_i contains only transitions from round i, and $\tau_1\tau_2 \ldots \tau_{k-1}(\sigma_0) = \sigma$. Since no transition here is from round k, by Lemma 6 we have that $\mathsf{path}(\sigma_0, \tau_1\tau_2 \ldots \tau_{k-1}) \triangleq_k \mathsf{path}(\sigma, \varepsilon)$, where ε is the empty schedule. This path will be a prefix of π'.

If $\tau = t_1 t_2 \ldots$, then we use the same strategy as in the base case to define $\tau' = t'_1 t'_2 \ldots$ such that $\mathsf{path}(\sigma, \tau') \triangleq_k \mathsf{path}(\sigma, \tau)$.

Now we have that $\pi' = \mathsf{path}(\sigma_0, \tau_1\tau_2 \ldots \tau_{k-1}\tau') \triangleq_k \mathsf{path}(\sigma, \varepsilon\tau) = \pi$.

($\Leftarrow$) Let now $\pi = \mathsf{path}(\sigma, \tau)$ be a path in $\mathsf{Sys}_\infty(\mathsf{TA})$. We construct a path π' from $\mathsf{Sys}^k(\mathsf{TA}^{\mathrm{rd}})$ such that $\pi \triangleq_k \pi'$.

Since we assume that all fair executions of $\mathsf{Sys}^0(\mathsf{TA}^{\mathrm{rd}})$ terminate w.r.t. $\Sigma^0_{\mathcal{B}}$, then by Lemma 9 for every $0 \leq i < k$ the set I^i is well-defined and all fair executions of $\mathsf{Sys}^i(\mathsf{TA}^{\mathrm{rd}})$ terminate. By the induction hypothesis, we know that $\mathsf{Sys}^i(\mathsf{TA}^{\mathrm{rd}}) \triangleq_i \mathsf{Sys}_\infty(\mathsf{TA})$. Together, this gives us that all rounds i, with $0 \leq i < k$, terminate in $\mathsf{Sys}_\infty(\mathsf{TA})$. In other words, every execution of $\mathsf{Sys}_\infty(\mathsf{TA})$ has a finite prefix that contains all its transitions of rounds less than k.

Let τ_{pre} be such a prefix of $\tau = \tau_{\mathrm{pre}}\tau_{\mathrm{suf}}$. Because τ_{pre} is finite, we may invoke Proposition 2, from which follows that there exist schedules $\tau_0, \tau_1, \ldots, \tau_{k-1}, \tau_{\geq k}$ such that every τ_i, $0 \leq i < k$ contains only round i transitions, $\tau_{\geq k}$ contains transitions of rounds at least k, the schedule $\tau_0\tau_1 \ldots \tau_{k-1}\tau_{\geq k}$ is applicable to σ, leads to $\tau_{\mathrm{pre}}(\sigma)$ when applied to σ, and

$$\mathsf{path}(\sigma, \tau_0\tau_1 \ldots \tau_{k-1}\tau_{\geq k}\tau_{\mathrm{suf}}) \triangleq_k \mathsf{path}(\sigma, \tau_{\mathrm{pre}}\tau_{\mathrm{suf}}). \tag{13}$$

Since $\sigma \in I = I^0$, the existence of schedules $\tau_0, \tau_1, \ldots, \tau_{k-1}$ confirms that $\sigma' = \tau_0\tau_1 \ldots \tau_{k-1}(\sigma)$ is in I^k. Next we apply the strategy from the base case to construct τ' from $\tau_{\geq k}\tau_{\mathrm{suf}}$, by projecting it to round k, such that

$$\mathsf{path}(\sigma', \tau_{\geq k}\tau_{\mathrm{suf}}) \triangleq_k \mathsf{path}(\sigma', \tau'). \tag{14}$$

By (13) and (14) we get $\pi' = \mathsf{path}(\sigma, \tau_0\tau_1 \ldots \tau_{k-1}\tau') \triangleq_k \mathsf{path}(\sigma, \tau_{\mathrm{pre}}\tau_{\mathrm{suf}}) = \pi$. □

By Lemma 7, for every $k \in \mathbb{N}_0$ and every $\sigma \in \Sigma^k_{\mathcal{B}}$, there is a corresponding configuration $\sigma' \in \Sigma^0_{\mathcal{B}}$ obtained from σ by renaming the round k to 0. Let f_k be the renaming function, i.e., $\sigma' = f_k(\sigma)$. Let us define $\Sigma^u \subseteq \Sigma^0_{\mathcal{B}}$ to be the union of all renamed initial configurations $\{f_k(\sigma) \colon k \in \mathbb{N}_0, \sigma \in I^k\}$.

Theorem 1. *Let system $\mathsf{Sys}_\infty(\mathsf{TA})$ be deadlock-free, and let all fair executions of $\mathsf{Sys}^0(\mathsf{TA}^{rd})$ w.r.t. $\Sigma^0_{\mathcal{B}}$ terminate. Given a formula $\varphi[i]$ from $\mathsf{LTL}_{\setminus\mathsf{X}}$ over AP_i, for a round variable i, we have $\mathsf{Sys}^0(\mathsf{TA}^{rd}) \models \mathbf{E}\,\varphi[0]$ w.r.t. initial configurations Σ^u if and only if there exists a $k \in \mathbb{N}_0$ such that $\mathsf{Sys}_\infty(\mathsf{TA}) \models \mathbf{E}\,\varphi[k]$.*

Proof. Let us first assume that $\mathsf{Sys}^0(\mathsf{TA}^{\mathrm{rd}}) \models \mathbf{E}\,\varphi_0$ w.r.t. initial configurations Σ^u. This means there is a path $\pi = \mathsf{path}(\sigma, \tau)$ such that $\sigma \in \Sigma^u$ and $\pi \models \varphi_0$. Since $\sigma \in \Sigma^u$, there is a $k \in \mathbb{N}_0$ and a $\sigma_k \in I^k$ such that $\sigma = f_k(\sigma_k)$. From Lemma 7 we know that $\mathsf{Sys}^0(\mathsf{TA}^{\mathrm{rd}}) \cong \mathsf{Sys}^k(\mathsf{TA}^{\mathrm{rd}})$, and thus there is a

schedule τ_k in $\mathsf{Sys}^k(\mathsf{TA}^{\mathrm{rd}})$ such that $\mathsf{path}(\sigma_k, \tau_k) \models \varphi_k$. Now Lemma 11 tells us that there must be a path π' from $\mathsf{Sys}_\infty(\mathsf{TA})$ such that $\mathsf{path}(\sigma_k, \tau_k) \triangleq_k \pi'$. By Proposition 3 we know that $\pi' \models \varphi_k$, and thus $\mathsf{Sys}_\infty(\mathsf{TA}) \models \mathbf{E}\,\varphi_k$. This proves one direction of the statement.

Assume now that there is a $k \in \mathbb{N}_0$ such that $\mathsf{Sys}_\infty(\mathsf{TA}) \models \mathbf{E}\,\varphi_k$. Thus, there is a path $\pi = \mathsf{path}(\sigma, \tau)$ in $\mathsf{Sys}_\infty(\mathsf{TA})$ such that $\pi \models \varphi_k$. By Lemma 11 we know that there is a path $\pi' = \mathsf{path}(\sigma', \tau')$ in $\mathsf{Sys}^k(\mathsf{TA}^{\mathrm{rd}})$ with $\pi \triangleq_k \pi'$, and then by Proposition 3 also $\pi' \models \varphi_k$. Finally, by Lemma 7 there is an equivalent path π_0 in $\mathsf{Sys}^0(\mathsf{TA}^{\mathrm{rd}})$ starting in $f_k(\sigma')$. Then we have that $\pi_0 \models \varphi_0$, and since $f_k(\sigma') \in \Sigma^u$, we know that $\mathsf{Sys}^0(\mathsf{TA}^{\mathrm{rd}}) \models \mathbf{E}\,\varphi_0$ w.r.t. initial configurations Σ^u. This concludes the other direction of the proof. □

In Section 4 we show how to reduce our specifications to formulas of the form $(\forall k \in \mathbb{N}_0)\mathbf{A}\,\psi[k]$. Theorem 1 deals with negations of such forms, namely with existence of a round k such that formula $\mathbf{E}\,\varphi[k]$ holds. Therefore, the theorem allows us to check on the single-round system instead of on the infinite one, if there is a counterexample to formulas we want to check.

7 Probabilistic Wait-Free Termination

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

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

$$\mathbb{P}^{\sigma_k}_{\mathbf{s}}\Big(\bigvee_{v\in\{0,1\}} \mathbf{G}\,\big(\bigwedge_{\ell\in\mathcal{F}_v} \boldsymbol{\kappa}[\ell, k] = 0\big)\Big) > p.$$

(C2) For every $v \in \{0, 1\}$,

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

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

Proposition 4. *If* $\mathsf{Sys}_\infty(\mathsf{PTA}) \models (C1) \wedge (C2)$*, then* $\mathsf{Sys}_\infty(\mathsf{PTA}) \models (3)$.

Proof. Fix a $\mathbf{p} \in \mathbf{P}_{RC}$, an initial configuration σ_0, and a round-rigid adversary $\mathtt{s}$.

Two options may occur along a path $\pi \in \mathsf{paths}(\sigma_0, \mathtt{s})$: (i) either round 0 ends with a final configuration in which all processes have the same value, say v, or (ii) round 0 ends with a final configuration with both values present.

(i) In this case we have that $\pi \models \mathbf{G}\,(\bigwedge_{\ell \in \mathcal{F}_{1-v}} \boldsymbol{\kappa}[\ell, 0] = 0)$, and by (C1), for $k = 0$, the probability that this case happens is at least p. Then, by Lemma 1 we also have $\pi \models \mathbf{G}\,(\bigwedge_{\ell \in \mathcal{I}_{1-v}} \boldsymbol{\kappa}[\ell, 1] = 0)$. Using (C2), in this case all processes decide value v in round 1.

(ii) The probability that the second case happens is at most $1 - p$. In this case, round 1 starts with an initial configuration σ_1 with both initial values 0 and 1. From σ_1 under $\mathtt{s}$, by the same reasoning as from σ_0, at the end on the round 1 we have the analogous two cases, and all processes decide in round 2 with probability at least p.

Iterating this reasoning, almost surely all processes eventually decide. Let us formally explain this iteration. Let σ_0 be an initial configuration, and let $\mathtt{s}$ be a round-rigid adversary. For a $k \in \mathbb{N}$, consider the event $\mathcal{E}_k$: from σ_0 and under $\mathtt{s}$, not every process decides in the first k rounds. In particular, at the end of every round $i < k$ it is not the case that everyone decides. By the reasoning above, namely case (ii) for round i, this happens with probability at most $(1 - p)$. Therefore, for k rounds we have $\mathbb{P}^{\sigma}_{\mathtt{s}}(\mathcal{E}_k) \leq (1 - p)^k$. The limit when k tends to infinity yields that the probability for not having Probabilistic Wait-Free Termination is 0. This is equivalent to the required formula (3). □

Notice that the non-zero probability p in condition (C1) depends only on $\mathbf{p}$, but does not depend on the precise initial configuration, nor on the adversary.

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

In the rest of this section, we detail how to reduce the verification of (C1), to a verification task that can be handled by ByMC. First observe that (C1) contains a single round variable, and recall that we restrict to round-rigid adversaries, so that it is sufficient to check them (omitting the round variables) on the single-round system. We introduce analogous objects as in the non-probabilistic case: $\mathsf{PTA}^{\mathrm{rd}}$ (analogously to Definition 6), and its counter system $\mathsf{Sys}(\mathsf{PTA}^{\mathrm{rd}})$.

7.1 Reducing probabilistic to non-probabilistic specifications

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

Lemma 12. *Let $\mathbf{p} \in \mathbf{P}_{RC}$ be a parameter valuation. In $\mathsf{Sys}(\mathsf{PTA}^{rd})$, for every LTL formula φ over atomic proposition* AP*, the following two statements are equivalent:*
(a) $\exists p > 0,\ \forall \sigma \in I_{\mathbf{p}},\ \forall \mathtt{s} \in \mathcal{A}^R.\quad \mathbb{P}^{\sigma}_{\mathtt{s}}(\varphi) > p$,
(b) $\forall \sigma \in I_{\mathbf{p}},\ \forall \mathtt{s} \in \mathcal{A}^R,\ \exists \pi \in \mathsf{paths}(\sigma, \mathtt{s}).\quad \pi \models \varphi$.

Proof. Fix parameters $\mathbf{p} \in \mathbf{P}_{RC}$.

The implication from top to bottom is trivial: if a probability is lower bounded by a positive constant, then there must be at least a path satisfying that property. It is thus sufficient to prove the bottom to top implication.

Assume that from every initial configuration σ with parameter values $\mathbf{p}$, and under all round-rigid adversaries $\mathtt{s}$, there exists a path $\pi \in \mathsf{paths}(\sigma, \mathtt{s})$ in $\mathsf{Sys}(\mathsf{PTA}^{\mathrm{rd}})$ such that $\pi \models \varphi$. Independently of σ and $\mathtt{s}$, our assumption that non-Dirac transitions may only happen at the end of PTA, we have that any path contains has at most $N(\mathbf{p})$ non-Dirac transitions. If δ is the smallest probability value appearing on such transitions, the probability of any path in $\mathsf{Sys}(\mathsf{PTA}^{\mathrm{rd}})$ is therefore lower-bounded by $\delta^{N(\mathbf{p})}$. Therefore, we can set $p = \delta^{N(\mathbf{p})}$, which only depends on PTA and $\mathbf{p}$. □

7.2 Verifying (C1) on a non-probabilistic TA

According to Lemma 12, in order to prove (C1), we only need to prove that in the system $\mathsf{Sys}(\mathsf{PTA}^{\mathrm{rd}})$ it holds that

$$(\forall \sigma \in I^k_{\mathbf{p}})\ (\forall \mathtt{s} \in \mathcal{A}^{\mathrm{R}})\ (\exists \pi \in \mathsf{paths}(\sigma, \mathtt{s}))\quad \pi \models \bigvee_{v \in \{0,1\}} \mathbf{G}\,\Big(\bigwedge_{\ell \in \mathcal{F}_v} \boldsymbol{\kappa}[\ell, k] = 0\Big). \quad (15)$$

As in Section 6, it is possible to modify $\mathsf{PTA}^{\mathrm{rd}}$ into a non-probabilistic TA, by replacing probabilistic choices by non-determinism. Still, the quantifier alternation of (15) (universal over initial configurations and adversaries vs. existential on paths) is not in the fragment handled by ByMC [11]. Once an initial configuration σ and an adversary $\mathtt{s}$ are fixed, the remaining branching is induced by non-Dirac transitions. By assumption, these transitions lead to final locations only, to both $\mathcal{F}_0$ and $\mathcal{F}_1$, and under round-rigid adversaries, they are the last transitions to be fired. To prove (15), it is sufficient to prove that all processes that fire only Dirac transitions will reach final locations of the same type ($\mathcal{F}_0$ or $\mathcal{F}_1$). If this is the case, then the existence of a path corresponds to all non-Dirac transitions being resolved in the same way. This allows us to remove the non-Dirac transitions from the model as follows. Given a $\mathsf{PTA}^{\mathrm{rd}}$, we define a threshold automaton TA^{m} with locations $\mathcal{L}$ (without $\mathcal{B}'$) such that for every non-Dirac rule $r = (\mathit{from}, \delta_{to}, \varphi, \boldsymbol{u})$ in PTA, all locations ℓ with $\delta_{to}(\ell) > 0$ are merged into a new location ℓ^{mrg} in TA^{m}. Note that this location must belong to $\mathcal{F}$. Naturally, instead of a non-Dirac rule r we obtain a Dirac rule $(\mathit{from}, \ell^{\mathrm{mrg}}, \varphi, \boldsymbol{u})$. Also we add self-loops at all final locations. Paths in $\mathsf{Sys}(\mathsf{TA}^{\mathrm{m}})$ correspond to prefixes of paths in $\mathsf{Sys}(\mathsf{PTA}^{\mathrm{rd}})$. In $\mathsf{Sys}(\mathsf{TA}^{\mathrm{m}})$, from a configuration σ, an adversary $\mathtt{s}$ yields a unique path, that is, $\mathsf{paths}(\sigma, \mathtt{s})$ is a singleton set. Thus, the existential quantifier from (15) can be replaced by the universal one.

Figure 4 illustrates the transformation on our running example from Figure 2. The new final location ℓ^{mrg} represents a coin toss taking place; it belongs neither to $\mathcal{F}_0$ nor $\mathcal{F}_1$.

Initial configurations in $\mathsf{Sys}^k(\mathsf{PTA}^{\mathrm{rd}})$ coincide with initial configurations in $\mathsf{Sys}^k(\mathsf{TA}^{\mathrm{m}})$. This exploits our definition of round-rigid adversaries, where all non-Dirac transitions are gathered at the end of a round.

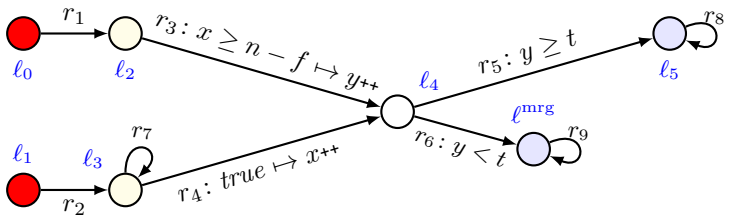

Fig. 4. A one-round non-probabilistic threshold automaton TA^{m} obtained from the PTA from Figure 2.

Lemma 13. *Fix $k \in \mathbb{N}_0$, an initial configuration σ from $\mathsf{Sys}^k(\mathsf{PTA}^{rd})$, and a round-rigid adversary $\mathtt{s}$. For every LTL formula $\varphi[k]$, the statements are equivalent:*

(a) there exists $\pi \in \mathsf{paths}(\sigma, \mathtt{s})$ such that $\pi \models \varphi[k]$ in $\mathsf{Sys}^k(\mathsf{PTA}^{rd})$,
(b) for every $\pi \in \mathsf{paths}(\sigma, \mathtt{s})$ holds $\pi \models \varphi[k]$ in $\mathsf{Sys}^k(\mathsf{TA}^m)$.

Proof. Paths in $\mathsf{Sys}^k(\mathsf{TA}^{\mathrm{m}})$ are prefixes of paths in $\mathsf{Sys}^k(\mathsf{PTA}^{\mathrm{rd}})$. Moreover, since every set of paths $\mathsf{paths}(\sigma, \mathtt{s})$ in $\mathsf{Sys}^k(\mathsf{TA}^{\mathrm{m}})$ is a singleton, then existential and universal quantification coincide. □

Finally, proving (15) on $\mathsf{PTA}^{\mathrm{rd}}$ reduces to proving $\mathbf{A} \bigvee_{v \in \{0,1\}} \mathbf{G}\,(\bigwedge_{\ell \in \mathcal{F}_v} \boldsymbol{\kappa}[\ell] = 0)$ holds on $\mathsf{Sys}(\mathsf{TA}^{\mathrm{m}})$, which can be checked automatically by ByMC.

8 Experiments

We have applied the approach presented in Sections 4–7 to five randomized fault-tolerant distributed algorithms [5]:

1. Protocol 1 for randomized consensus by Ben-Or [3]. We consider two kinds of crashes: clean crashes (ben-or-cc) and dirty crashes (ben-or-dc). During a dirty crash a process can send to a subset of processes, while in clean crashes a process is either sending to all proceses or none. This algorithm works correctly when $n > 2t$.
2. Protocol 2 for randomized Byzantine consensus (ben-or-byz) by Ben-Or [3]. This algorithm tolerates Byzantine faults when $n > 5t$.
3. Protocol 2 for randomized consensus (rabc-c) by Bracha [5]. It runs as a high-level algorithm together with a low-level broadcast that turns Byzantine faults into "little more than fail-stop (faults)". We check only the high-level algorithm for clean crashes. Our model checker produces counterexamples when Byzantine or Byzantine-symmetric faults are introduced in rabc-c. The multi-layered protocol is designed for $f < n/3$ faults. However, our tool shows that rabc-c itself tolerates $f < n/2$ clean crashes.

[5] The benchmarks and the instructions on running the experiments are available from: `https://forsyte.at/software/bymc/artifact42/`

4. k-set agreement for crash faults (kset) by Raynal [18], for $k = 2$. This algorithm works in presense of clean crashes when $n > 3t$.
5. Randomized Byzantine one-step consensus (rs-bosco) by Song and van Renesse [20]. This algorithm tolerates Byzantine faults when $n > 3t$, and it terminates fast when $n > 7t$ or $n > 5t$ and $f = 0$.

Following the reduction approach of Sections 4–7, for each benchmark, we have encoded two versions of one-round threshold automata: an N-automaton that models a coin toss by a non-deterministic choice, and a P-automaton that never leaves the coin-toss location, once it entered this location. The N-automaton is used to support the non-probabilistic reasoning, while the P-automaton is used to prove probabilistic wait-free termination. Both automata are given as the input to Byzantine Model Checker (ByMC) [11], which implements the parameterized model checking techniques for safety [8] and liveness [9] of counter systems of threshold-automata (for a bounded number of rounds and no randomization).

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

Label	Name	Automaton	Formula
S1	`agreement_0`	N	$\mathbf{A}\,\mathbf{G}\,(\neg\textsc{Ex}\{\mathsf{D0}\}) \vee \mathbf{G}\,(\neg\textsc{Ex}\{\mathsf{D1},\mathsf{E1}\})$
S2	`validity_0`	N	$\mathbf{A}\,\textsc{All}\{\mathsf{V0}\} \rightarrow \mathbf{G}\,(\neg\textsc{Ex}\{\mathsf{D1},\mathsf{E1}\})$
S3	`completeness_0`	N	$\mathbf{A}\,\textsc{All}\{\mathsf{V0}\} \rightarrow \mathbf{G}\,(\neg\textsc{Ex}\{\mathsf{D1},\mathsf{E1}\})$
S4	`round-term`	N	$\mathbf{A}\,\mathit{fair} \rightarrow \mathbf{F}\,\textsc{All}\{\mathsf{D0},\mathsf{D1},\mathsf{E0},\mathsf{E1},\mathsf{CT}\}$
S5	`decide-or-flip`	P	$\mathbf{A}\,\mathit{fair} \rightarrow \mathbf{F}\,(\textsc{All}\{\mathsf{D0},\mathsf{E0},\mathsf{CT}\} \vee \textsc{All}\{\mathsf{D1},\mathsf{E1},\mathsf{CT}\})$
S1′	`sim-agreement`	N	$\mathbf{A}\,\mathbf{G}\,(\neg\textsc{Ex}\{\mathsf{D0},\mathsf{E0}\} \vee \neg\textsc{Ex}\{\mathsf{D1},\mathsf{E1}\})$
S1″	`2-agreement`	N	$\mathbf{A}\,\mathbf{G}\,(\neg\textsc{Ex}\{\mathsf{D0},\mathsf{E0}\} \vee \neg\textsc{Ex}\{\mathsf{D1},\mathsf{E1}\} \vee \neg\textsc{Ex}\{\mathsf{D2},\mathsf{E2}\})$

Table 1. Temporal properties verified in our experiments for value 0 (the properties for value 1 can be obtained by swapping 0 and 1). We write fairness constraints as *fair* to save space.

Table 1 summarizes the temporal properties that were verified in our experiments. Given the set of all possible locations $\mathcal{L}$, a set $Y = \{\ell_1, \ldots, \ell_m\} \subseteq \mathcal{L}$ of locations, and the designated crashed location $\mathsf{CR} \in \mathcal{L}$, we use the shorthand notation: $\textsc{Ex}\{\ell_1, \ldots, \ell_m\}$ for $\bigvee_{\ell \in Y} \boldsymbol{\kappa}[\ell] \neq 0$ and $\textsc{All}\{\ell_1, \ldots, \ell_m\}$ for $\bigwedge_{\ell \in \mathcal{L} \setminus Y} (\boldsymbol{\kappa}[\ell] = 0 \vee \ell = \mathsf{CR})$. For rs-bosco and kset, instead of checking S1, we check S1′ and S1″.

Table 2 presents the computational results of our experiments. The meaning of the columns is as follows: column $|\mathcal{L}|$ shows the number of automata locations, column $|\mathcal{R}|$ shows the number of automata rules, column $|\mathcal{S}|$ shows the number of enumerated schemas (which depends on the structure of the automaton and the specification), column *time* show the computation times — either in seconds or in the format `HH:MM:SS`. For $|\mathcal{R}|$, we give the figures for the N-automata, since

Table 2. The experiments for rows 1-5 were run on a single computer (Apple MacBook Pro 2018, 16GB). The experiments for row 6 (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	$\|\mathcal{S}\|$	Time	$\|\mathcal{S}\|$	Time	$\|\mathcal{S}\|$	Time	$\|\mathcal{S}\|$	Time
1 ben-or-cc	10	27	9	1	5	0	5	0	5	0	5	0
2 ben-or-dc	10	32	9	1	5	1	5	0	5	0	5	1
3 ben-or-byz	9	18	3	1	2	0	2	0	2	0	2	1
4 rabc-cr	11	31	9	0	5	1	5	1	5	0	5	0
5 kset	13	58	65	3	65	17	65	12	65	39	65	40
6 rs-bosco	19	48	156M	3:21	156M	3:02	156M	3:21	n/a	n/a	156M	3:43

they have more rules in addition to the rules in P-automata. To save space, we omit the figures for memory use from the table: Benchmarks 1–5 need 30–170 MB RAM, whereas rs-bosco needs up to 1.5 GB RAM per cluster node.

The benchmark rs-bosco presents a challenge for the schema enumeration technique of [9]: Its threshold automaton contains 12 threshold guards that can change their values almost in any order. Additional combinations are produced by the temporal formulas. ByMC reduces the number of combinations by analyzing dependencies between the guards. However, this benchmark requires us to enumerate between 11! and 14! schemas. To this end, we have run the verification experiments for rs-bosco on 1024 CPU cores of the computing cluster Grid5000. Table 2 presents the wall time results for rs-bosco, that is, the actual number of computation hours on all the cores is the wall time multiplied by 1024.

For all the benchmarks in Table 2, ByMC has reported that the specifications hold. By varying the relations between the parameters (e.g., by changing $n > 3t$ to $n > 2t$), we have found that rabc-cr can handle more faults, that is, $t < n/2$ in contrast to the original $t < n/3$ (the original was needed to implement the underlying communication structure which we assume given in the experiments). In other cases, whenever we changed the parameters, that is, increased the number of faults beyond the known bound, the tool reported a counterexample.

9 Conclusions

We lifted the threshold automata framework to multi-round randomized algorithms. We proved a reduction that allows us to check $\mathsf{LTL}_{-\mathsf{X}}$ specifications over propositions for one round in a single-round automaton so that the verification results transfer directly to the multi-round counter system. Using round-based compositional reasoning, we have shown that this is sufficient to check specifications that span multiple rounds, *e.g.*, agreement. Almost-sure termination under round-rigid adversaries relies on a distinct reduction argument.

By experimental evaluation we showed that the verification conditions that came out of our reduction can be automatically verified for several challenging randomized consensus algorithms in the parameterized setting.

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

There are few contributions that address the automated verification of probabilistic parameterized systems [19,4,17,16]. In contrast with these, our processes are not finite-state, due to the round numbers and parameterized guard expressions. The seminal work of Pnueli and Zuck restricts to bounded shared variables and cannot use arithmetic thresholds (different from 1 and n) [19]. Algorithms based on well-structured transition systems [4] do not directly apply to systems with several parameters, such as the ones generated by probabilistic threshold automata. Regular model checking techniques [17,16] cannot handle arithmetic resilience conditions such as $n > 3t$, nor unbounded shared variables.

References

1. Aguilera, M., Toueg, S.: The correctness proof of Ben-Or's randomized consensus algorithm. Distributed Computing pp. 1–11 (2012), online first
2. Baier, C., Katoen, J.P.: Principles of model checking. MIT Press (2008)
3. Ben-Or, M.: Another advantage of free choice: Completely asynchronous agreement protocols (extended abstract). In: PODC. pp. 27–30 (1983)
4. Bertrand, N., Fournier, P.: Parameterized verification of many identical probabilistic timed processes. In: FSTTCS. LIPIcs, vol. 24, pp. 501–513 (2013)
5. Bracha, G.: Asynchronous byzantine agreement protocols. Inf. Comput. 75(2), 130–143 (1987)
6. Elrad, T., Francez, N.: Decomposition of distributed programs into communication-closed layers. Sci. Comput. Program. 2(3), 155–173 (1982)
7. Fischer, M.J., Lynch, N.A., Paterson, M.S.: Impossibility of distributed consensus with one faulty process. J. ACM 32(2), 374–382 (1985)
8. Konnov, I., Lazic, M., Veith, H., Widder, J.: Para^2: Parameterized path reduction, acceleration, and SMT for reachability in threshold-guarded distributed algorithms. Formal Methods in System Design 51(2), 270–307 (2017)
9. Konnov, I., Lazić, M., Veith, H., Widder, J.: A short counterexample property for safety and liveness verification of fault-tolerant distributed algorithms. In: POPL. pp. 719–734 (2017)
10. Konnov, I., Veith, H., Widder, J.: On the completeness of bounded model checking for threshold-based distributed algorithms: Reachability. Information and Computation 252, 95–109 (2017)
11. Konnov, I., Widder, J.: Bymc: Byzantine model checker. In: ISoLA (3). LNCS, vol. 11246, pp. 327–342. Springer (2018)
12. Kwiatkowska, M.Z., Norman, G.: Verifying randomized byzantine agreement. In: FORTE. pp. 194–209 (2002)
13. Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: CAV. pp. 585–591 (2011)

14. Kwiatkowska, M.Z., Norman, G., Segala, R.: Automated verification of a randomized distributed consensus protocol using cadence SMV and PRISM. In: CAV. pp. 194–206 (2001)
15. Lehmann, D.J., Rabin, M.O.: On the advantages of free choice: A symmetric and fully distributed solution to the dining philosophers problem. In: POPL. pp. 133–138 (1981)
16. Lengál, O., Lin, A., Majumdar, R., Rümmer, P.: Fair termination for parameterized probabilistic concurrent systems. In: TACAS. LNCS, vol. 10205, pp. 499–517 (2017)
17. Lin, A., Rümmer, P.: Liveness of randomised parameterised systems under arbitrary schedulers. In: CAV. LNCS, vol. 9780, pp. 112–133. Springer (2016)
18. Mostéfaoui, A., Moumen, H., Raynal, M.: Randomized k-set agreement in crash-prone and byzantine asynchronous systems. Theor. Comput. Sci. 709, 80–97 (2018)
19. Pnueli, A., Zuck, L.: Verification of multiprocess probabilistic protocols. Distributed Computing 1(1), 53–72 (1986)
20. Song, Y.J., van Renesse, R.: Bosco: One-step Byzantine asynchronous consensus. In: DISC. LNCS, vol. 5218, pp. 438–450 (2008)

APPENDIX

A Detailed Definition of the Underlying Counter Systems

Definition 7. *Given a probabilistic counter system* $\mathsf{Sys}(\mathsf{PTA}) = (\Sigma, I, \mathsf{Act}, \Delta)$*, we define its non-probabilistic version* $\mathsf{Sys}_{np}(\mathsf{PTA})$ *to be the tuple* (Σ, I, R)*, where* R *is a transition relation defined below.*

If $\mathsf{Act} = \mathcal{R} \times \mathbb{N}_0$ *and if* $\mathcal{R}_{np}$ *is defined from* $\mathcal{R}$ *as in Definition 2, then transitions are tuples* $t = (r_\ell, k) \in \mathcal{R}_{np} \times \mathbb{N}_0$ *such that* $\alpha = (r, k)$ *is an action from* Act *and for* $\ell \in \mathcal{L}$ *holds that* $\alpha.\delta_{to}(\ell) > 0$*. Transition* t *is unlocked in a configuration* σ *from* $\mathsf{Sys}_{np}(\mathsf{PTA})$ *if* α *is unlocked in* σ *in* $\mathsf{Sys}(\mathsf{PTA})$*. Similarly we define when* t *is applicable to* σ*. We obtain* σ' *by applying an applicable transition* t *to* σ*, written* $t(\sigma) = \sigma'$*, if and only if there exists a location* $\ell \in \mathcal{L}$ *such that* $\mathrm{apply}(\alpha, \ell, \sigma) = \sigma'$*.*

Two configurations σ *and* σ' *are in the transition relation* R*, i.e.,* $(\sigma.\sigma') \in R$*, if and only if there exists a transition* t *such that* $\sigma' = t(\sigma)$*.*

Definition 8. *Given an arbitrary* $\mathsf{TA} = (\mathcal{L}, \mathcal{V}, \mathcal{R}_{np}, RC)$*, with border, initial, and final location sets* $\mathcal{B}$*,* $\mathcal{I}$*, and* $\mathcal{F}$*, respectively, we define its* infinite counter system $\mathsf{Sys}_\infty(\mathsf{TA})$ *to be the tuple* (Σ, I, R)*. Configurations from* Σ *and* I *are defined as in Section 3.1. A transition* t *is a tuple* $(r_\ell, k) \in \mathcal{R}_{np} \times \mathbb{N}_0$*. Since it coincides with Dirac actions, we define when a transition is unlocked in a configuration and when it is applicable to a configuration, in the same way as for a Dirac action in Section 3.1. A configuration* σ' *is obtained by applying an applicable transition* $t = (r_\ell, k)$ *to* σ*, written* $\sigma' = t(\sigma)$*, if and only if* $\mathrm{apply}(\alpha, \ell, \sigma) = \sigma'$*, for a Dirac action* $\alpha = (r_\ell, k)$ *and the destination location* ℓ *of* r*.*

Now we have $(\sigma, \sigma') \in R$*, if and only if there exists a transition* t *such that* $\sigma' = t(\sigma)$*.*

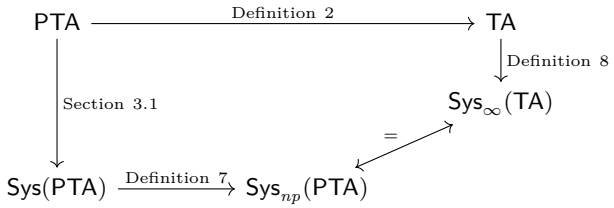

Fig. 5. Diagram following Proposition 5

Proposition 5. *Given a* PTA*, the non-probabilistic version* $\mathsf{Sys}_{np}(\mathsf{PTA})$ *of its counter system coincides with the infinite counter system* $\mathsf{Sys}_\infty(\mathsf{TA})$ *of its threshold automaton.*

It is easy to see that the diagram from Figure 5 commutes, and thus every PTA yields the unique non-probabilistic counter system. The two constructions give us possibility to remove probabilistic reasoning either on the level of a PTA (using Definition 2) or on the level of a counter system Sys(PTA) (using Definition 7). Therefore, for different reasoning we will refer to the construction that is more convenient.