Parameterized Verification of Many Identical Probabilistic Timed Processes

Nathalie Bertrand[1] and Paulin Fournier[2]

1 Inria Rennes, France
nathalie.bertrand@inria.fr

2 ENS Cachan Antenne de Bretagne, France
paulin.fournier@eleves.bretagne.ens-cachan.fr

Abstract

Parameterized verification aims at validating a system's model irrespective of the value of a parameter. We introduce a model for networks of identical probabilistic timed processes, where the number of processes is a parameter. Each process is a probabilistic single-clock timed automaton and communicates with the others by broadcasting. The number of processes either is constant (static case), or evolves over time through random disappearances and creations (dynamic case). An example of relevant parameterized verification problem for these systems is whether, independently of the number of processes, a configuration where one process is in a target state is reached almost-surely under all scheduling policies. On the one hand, most parameterized verification problems turn out to be undecidable in the static case (even for untimed processes). On the other hand, we prove their decidability in the dynamic case.

1998 ACM Subject Classification D.2.4 Software/Program verification, F.3.1 Specifying and Verifying and Reasoning about Programs, G.3 Probabilities and Statistics

Keywords and phrases model checking, Markov decision processes, parameterized verification

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2013.501

1 Introduction

Automated verification of reactive programs started in the 80's with simple models, and specifically finite state automata. It was successfully applied to hardware validation. Later, the focus on software verification called the need for techniques tackling infinite-state systems. The infinite nature is mainly caused by two reasons: either the program handles data structures over infinite domains (*e.g.* counters, dense-time clocks, queues or stacks), or it runs on a network of an arbitrary number of processes. In the latter case, *parameterized verification* aims at verifying the system independently of its actual instantiation, that is, independently of the number of processes involved. Our contribution falls in these two categories: we investigate the parameterized verification of a class of programs over infinite domains.

Already in the 80's, Apt and Kozen showed the undecidability of very general parameterized model-checking problems [7]. For more specific models and properties, one can however achieve some decidability, see *e.g.* [12] where networks of identical finite-state automata are model-checked against regular properties. This framework was later extended to networks of identical timed automata for which safety properties are decidable if the timed automata have a single clock [5] and undecidable otherwise [4]. Broadcasts protocols form a model with an unbounded number of processes that communicate by rendez-vous or broadcast. For broadcast protocols, safety properties are decidable, and liveness properties are undecidable [15]. More recently, a series of papers, initiated with [14], investigates the parameterized

33rd Int'l Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2013).
Editors: Anil Seth and Nisheeth K. Vishnoi; pp. 501–513
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

verification of a model of networks, *e.g.* suitable for the representation of ad-hoc networks. The nodes in the network are modeled by finite automata that communicate through multiple broadcasts. The decidability status of reachability and coverability problems depends on the topology and its evolution [14]. Later, the positive results have been extended to networks of single-clock timed automata [3].

Simultaneous to the extension of verification techniques to infinite-state models, and with the common objective to verify more complex systems, the models and problems moved from boolean to quantitative. A prominent class of quantitative systems is the one of probabilistic models. Finite-state probabilistic models have been extensively studied and mature tools perform their verification (see [10, Chapter 10] and references therein). However, to the best of our knowledge, the parameterized verification of probabilistic systems hasn't been investigated yet. The model-checking of PCTL formulas against Markov chains with parametric coefficients was first investigated in [13], and the recent work [17] studies the verification of probabilistic programs with parameters. Both use parameters to model unknown probabilities and differ from our setting of parameterized verification where the parameter is the number of processes in a network.

We introduce a modeling formalism that combines infinite-state space, due to an unknown number of processes in the network as well as data structures with infinite domains, and probabilistic behaviors. *Probabilistic timed networks* are formed of many identical probabilistic timed automata [18] with a single clock, and interaction between the processes is modeled by message broadcasting. Moreover, in order to represent some mobility in the network, we further introduce *dynamic* probabilistic timed networks, where processes can disappear and be created, according to fixed probability laws.

A potential application domain for our model is the one of wireless sensor networks (WSN) that consist of a large number of nodes measuring and transmitting data. The number of nodes is a significant parameter while setting up the network, since it affects the performance by influencing the risks of collisions in communications. Most protocols for WSN include probabilistic choices and timing constraints. Also, in many cases the number of nodes in the network evolves over time due to nodes breaking down, or nodes refilling their battery using *e.g.* solar energy. Moreover, in some applications, the placement of sensors and their exact number is unknown. We therefore advocate that (dynamic) probabilistic timed networks with a parametric number of processes make a quite suitable model for WSN protocols. So far, the automated verification of such protocols has been performed for a fixed, and rather small, number of nodes, as in [16] where Prism [18] is used to verify the contention resolution protocol in IEEE standard 802.15.4. In comparison, the parameterized verification of probabilistic timed networks would provide answers for an arbitrary number of processes.

Given a probabilistic timed network, we consider relevant *parameterized verification problems*, such as the following. For a target state of the process model, can a configuration with some process in the target state be reached almost surely, whatever the initial number of processes and for every scheduling policy? Equivalently, the problem is whether independently of the number of processes, the minimum probability to reach a target configuration is 1. Beyond this particular problem, we consider all variants of qualitative questions, *i.e.* where the minimum or maximum probabilities are compared to the thresholds 0 or 1.

On the one hand we prove that most qualitative verification problems are undecidable when the topology is static, that is if the number of processes is constant (but unknown). These undecidability results already hold in the untimed case, *i.e.* when the individual processes are Markov decision processes (MDP) rather than probabilistic timed automata. To establish the undecidability results, we explain how a probabilistic network can simulate a 2-counter machine, using the processes to encode the counter values. On the other hand, in the dynamic case, we provide a decision procedure for all the parameterized verification problems of interest. In each case, the termination of the algorithm is ensured by a dedicated well-quasi-ordering on configurations of the network, and the correctness of the algorithm relies on a finite attractor property in our model. We also establish a complexity lower-bound by reducing the reachability problem in lossy channel systems: the qualitative parameterized verification problems in the dynamic case are non-primitive recursive.

The rest of the paper is organized as follows. We define the model of dynamic probabilistic timed networks in Section 2 together with the parameterized verification problems we consider. Section 3 is devoted to showing the undecidability in the static case. In Section 4 we establish the decidability results in the dynamic case. We conclude by mentioning open questions and future work.

2 Modeling probabilistic protocols

Given E, an at most countable set, we write $\mathsf{Dist}(E)$ for the set of discrete probability distributions over E, that is, functions $\delta : E \to [0,1]$ such that $\sum_{e\in E} \delta(e) = 1$.

For an arbitrary set E, we write $\mathbb{M}(E)$ for the set of multisets over E, *i.e.* the set of multiplicity functions $f : E \to \mathbb{N}$. We also write $f = \langle x, x, x, y\rangle$ for the multiset f defined as $f(x) = 3$, $f(y) = 1$ and $\forall z \in E \setminus \{x, y\}, f(z) = 0$. We now introduce simple operations on multisets. For $f \in \mathbb{M}(E)$ and $x \in E$ we write $f + x$ for the multiset f' defined by $f'(x) = f(x) + 1$ and $f'(y) = f(y)$ otherwise. Symmetrically, and assuming $f(x) > 0$, $f - x$ is a notation for f' such that $f' + x = f$.

Given a clock $\mathtt{x}$, we write $\mathcal{G}(\mathtt{x})$ for the set of guards over $\mathtt{x}$, *i.e.* conjunctions of atomic constraints of the form $\mathtt{x} \sim k$ for $\sim \in \{<, \leq, >, \geq, =\}$ and $k \in \mathbb{N}$. The trivial guard, consisting of no constraints, is written true. A clock is said to *satisfy* a guard, denoted $\mathtt{x} \models g$ if it's value satisfies all the constraints of the guard. The set of possible updates for clock $\mathtt{x}$ is $\mathsf{Up} = \{\mathtt{x} := 0, \emptyset\}$: $\mathtt{x}$ is either reset or left unchanged. We write $\mathsf{up}(\mathtt{x})$ the result on $\mathtt{x}$ of the update $\mathsf{up} \in \mathsf{Up}$.

2.1 Probabilistic timed networks

▶ **Definition 1** (Probabilistic one-clock timed protocol)**.** A *probabilistic one-clock timed protocol* (or for short *probabilistic timed protocol*) is a tuple $\mathcal{P} = (Q, q_0, \mathtt{x}, \Sigma, \Delta)$ where

- Q is a finite set of control states, and $q_0 \in Q$ is initial,
- $\mathtt{x}$ is a clock,
- Σ is a finite message alphabet with a subset Σ_ε of internal labels,
- Δ is the finite probabilistic edge relation, partitioned into
  - internal actions: $\Delta_\mathsf{i} \subseteq Q \times \mathcal{G}(\mathtt{x}) \times \Sigma_\varepsilon \times \mathsf{Dist}(\mathsf{Up} \times Q)$,
  - broadcasts: $\Delta_\mathsf{b} \subseteq Q \times \mathcal{G}(\mathtt{x}) \times !(\Sigma \setminus \Sigma_\varepsilon) \times \mathsf{Up} \times Q$,
  - receptions: $\Delta_\mathsf{r} \subseteq Q \times \mathcal{G}(\mathtt{x}) \times ?(\Sigma \setminus \Sigma_\varepsilon) \times \mathsf{Up} \times Q$.

A simple example of probabilistic timed protocol modelling mutual exclusion over two resources is given in Figure 1. Internal actions are $\{\alpha^i \mid i \in [\![0, 2]\!]\}$, and broadcast actions are $\{\beta^i, \beta^{i'} \mid i \in \{1, 2\}\}$. This example will be further developed in the sequel.

▶ Remark. In our model, a control state can be the source of several internal actions, each giving rise to a probability distribution for the successor state, whereas broadcasts and receptions are deterministic. This is not a real restriction, since systems with nondeterministic and probabilistic choices for broadcast and receptions can be encoded in our model by introducing intermediary states and additional internal actions.

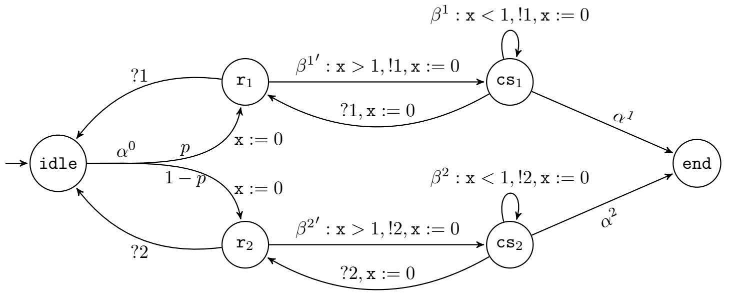

Figure 1 A probabilistic timed protocol modeling mutual exclusion over two resources.

Probabilistic timed protocols are probabilistic timed automata [18] with a single clock. We introduced a new terminology to highlight this particularity, and more importantly, to emphasize the communicative nature of probabilistic timed protocols.

▶ **Definition 2** (Probabilistic timed network). A probabilistic timed network $\mathcal{P}^N$, is composed of $N \in \mathbb{N}_{>0}$ copies, called *processes*, of a probabilistic timed protocol $\mathcal{P}$.

The intuitive interpretation of a probabilistic timed network $\mathcal{P}^N$ is that N processes arranged in a clique execute the probabilistic timed protocol $\mathcal{P}$ simultaneously.

▶ **Example 3.** On the simple example from Figure 1 modeling mutual exclusion over two resources, each process starts in state `idle` and can at any time request a resource. The choice of the allocated resource is probabilistic. The requesting process must then stay in the corresponding request state ($\mathtt{r}_i$) at least one time unit before moving to the critical section state ($\mathtt{cs}_i$) representing usage of resource i. If a requesting process receives a message indicating that the resource is already used, it moves back to the initial state `idle`. Hence, when granted the access to a resource, a process may use it for at least 1 time unit, and can inform the others by broadcast that he is still using it to be granted an other time unit.

Let us detail the semantics of the probabilistic timed network $\mathcal{P}^N$. A *configuration* is a finite multiset $\gamma \in \mathbb{M}(Q \times \mathbb{R}_+)$ over the set of pairs composed of a control state and a real value for the clock. Intuitively, if configuration γ contains exactly two occurrences of $(q, 0.5)$, written $\gamma(q, 0.5) = 2$, then two processes have current state q and current clock value $\mathtt{x} = 0.5$. Moreover, since N processes are involved in the network, $\sum_{(q,x)|\gamma(q,x)>0} \gamma(q, x) = N$. In the sequel, the set of configurations with N processes is written Conf^N.

The semantics of $\mathcal{P}^N$ is given in terms of a timed Markov decision process $[\![\mathcal{P}^N]\!] = (\mathsf{Conf}^N, \gamma_0^N, \mathsf{T}^N, \mathsf{Act} \cup \mathbb{R}_+)$, where γ_0^N is the initial configuration, defined by $\gamma_0^N(q_0, 0) = N$ and $\gamma_0^N(q, x) = 0$ otherwise; T^N is the set of transitions (defined in the sequel); and actions are partitioned into time elapsing $\mathbb{R}_+$, and discrete actions $\mathsf{Act} = Q \times \mathbb{R}_+ \times \Delta \cup \{\mathsf{unlock}\}$ where unlock is a special action. From any configuration $\gamma \in \mathsf{Conf}^N$, the set of possible discrete actions depends on the control state and clock value of the processes in γ, which explains the typing of discrete actions: $Q \times \mathbb{R}_+ \times \Delta$. Informally, when performing a discrete action from configuration γ, first a process is selected nondeterministically, and second, a (communication or internal) transition enabled for that process is performed. However, since we do not distinguish processes with same control state and same clock value, the intuitive semantics of discrete steps is rather to select a pair (q, x) with $\gamma(q, x) > 0$ and then to fire for some process in state (q, x) an enabled action. Messages are broadcast to all other processes, whereas internal actions only affect the chosen process. Moreover, in order to forbid finite runs, we use a special action unlock that is enabled only when no other discrete action is enabled, even after some time elapsing.

We now define formally the transition function T^N and exemplify it on the example protocol from Figure 1 for the configuration $\gamma_{\mathsf{ex}} = \langle(\mathtt{idle}, 2), (\mathtt{r}_1, 0.3), (\mathtt{r}_1, 0.4), (\mathtt{cs}_1, 0.8)\rangle$. T^N is composed of the following transitions:

- time elapse: For every delay $\tau \in \mathbb{R}_+$ and every configuration $\gamma \in \mathsf{Conf}^N$, there exists a deterministic transition in the Markov decision process $[\![\mathcal{P}^N]\!]$ from γ to γ' defined by $\gamma'(q, x + \tau) = \gamma(q, x)$ (denoted $\gamma' = \gamma + \tau$), for all states $q \in Q$ and clock values $x \in \mathbb{R}_+$. In such a case we write $\gamma \xrightarrow{\tau} \gamma'$.
  E.g.: $\gamma_{\mathsf{ex}} \xrightarrow{0.1} \langle(\mathtt{idle}, 2.1), (\mathtt{r}_1, 0.4), (\mathtt{r}_1, 0.5), (\mathtt{cs}_1, 0.9)\rangle$
- internal: For every internal action $\alpha = (q, g, \varepsilon, \delta) \in \Delta_{\mathsf{i}}$, every $\gamma \in \mathsf{Conf}^N$ and every clock value $x \in \mathbb{R}_+$, such that $\gamma(q, x) > 0$ and $x \models g$, α_x is *enabled* from γ in the Markov decision process $[\![\mathcal{P}^N]\!]$ and yields a distribution μ defined by $\mu(\gamma'|\gamma, \alpha_x) = \delta(\mathsf{up}, q')$ where $\gamma' = \gamma - (q, x) + (q', \mathsf{up}(x))$. Note that several α actions can be available in γ, for various clock values x; this is why in $[\![\mathcal{P}^N]\!]$, we attach the subscript x to α. Here, we write $\gamma \xrightarrow{\alpha_x, \mu(\gamma'|\gamma, \alpha_x)} \gamma'$.
  E.g.: $\gamma_{\mathsf{ex}} \xrightarrow{\alpha_2^0, 1-p} \langle(\mathtt{r}_2, 0), (\mathtt{r}_1, 0.3), (\mathtt{r}_1, 0.4), (\mathtt{cs}_1, 0.8)\rangle$
- communication: For every broadcast $\beta = (q, g, !a, \mathsf{up}, q') \in \Delta_{\mathsf{b}}$, every $\gamma \in \mathsf{Conf}^N$ and every clock value $x \in \mathbb{R}_+$ such that $\gamma(q, x) > 0$ and $x \models g$, β_x is *enabled* from γ and is a deterministic transition from γ to γ' in the Markov decision process $[\![\mathcal{P}^N]\!]$, where γ' is obtained in three steps
  - $\gamma_1 = \gamma - (q, x)$ (the process responsible for the broadcast is treated separately)
  - $\gamma_2(r', y') = \sum\{\gamma_1(r, y) \mid \exists(r, g', ?a, \mathsf{up}', r') \in \Delta_{\mathsf{r}} \wedge y \models g' \wedge \mathsf{up}'(y) = y'\}$ (all other processes receive the message)
  - $\gamma' = \gamma_2 + (q', \mathsf{up}(x))$.

  For such communications, we write $\gamma \xrightarrow{\beta_x} \gamma'$.
  E.g.: $\gamma_{\mathsf{ex}} \xrightarrow{\beta_{0.8}^1} \langle(\mathtt{idle}, 2), (\mathtt{idle}, 0.3), (\mathtt{idle}, 0.4), (\mathtt{cs}_1, 0)\rangle$
- deadlock: Any configuration $\gamma \in \mathsf{Conf}^N$ such that for every $\tau \in \mathbb{R}_+$ there is no enabled actions from $\gamma + \tau$, is called a *deadlock* configuration. Action unlock is then *enabled* from γ, and does not change the configuration. Here, we write $\gamma \xrightarrow{\mathsf{unlock}} \gamma$.
  E.g.:$\langle(\mathtt{end}, 2), (\mathtt{end}, 0.3), (\mathtt{end}, 1.4)\rangle \xrightarrow{\mathsf{unlock}} \langle(\mathtt{end}, 2), (\mathtt{end}, 0.3), (\mathtt{end}, 1.4)\rangle$

2.2 Dynamic probabilistic timed networks

Probabilistic timed networks from Definition 2 are networks where the number of processes is constant. We now discuss a reasonable way to introduce dynamism in the network. In the application to wireless sensor networks, the number of nodes can change during the execution of the system, since nodes can break down or run out of battery, but also, can be newly inserted or refill their battery. We therefore propose a model of probabilistic timed networks, in which the number of processes evolves over time, and in which, disappearances and creations of processes are independent random events following given probability distributions. Abstracting dynamism by random events seems a good trade-off between simplicity and realism of the model.

▶ **Definition 4** (Dynamic probabilistic timed network)**.** A *dynamic probabilistic timed network*, written $\mathcal{P}_\Lambda^{N_0}$ is composed of initially N_0 copies of $\mathcal{P}$, together with a pair of disappearance and creation rates $\Lambda = (\lambda_-, \lambda_+) \in (0, 1)^2$.

The rates λ_+ and λ_- represent dynamic creation and disappearance of processes according to fixed probabilistic laws: after each discrete action , each process disappears with probability λ_-, followed by the creation of k processes (in control state q_0 with clock value 0) with probability $\lambda_+^k(1-\lambda_+)$, for every integer $k \in \mathbb{N}$. Obviously if $\lambda_+ = \lambda_- = 0$, the number of processes is constant and we recover the model of (static) probabilistic timed network from Definition 2.

The formal semantics of a dynamic probabilistic timed network $\mathcal{P}_\Lambda^{N_0}$ is obtained from the ones of $\mathcal{P}^N$ for each $N \in \mathbb{N}$. More precisely $[\![\mathcal{P}_\Lambda^{N_0}]\!]$ is a timed Markov decision process $(\mathsf{Conf}, \gamma_0^{N_0}, \mathsf{T}, \mathsf{Act} \cup \mathbb{R}_+)$, where the transition function T is defined based on the T^N's and the rates Λ ((see below)). The set of configurations is $\mathsf{Conf} = \bigcup_N \mathsf{Conf}^N$, and the set of discrete actions in the MDP is still $\mathsf{Act} = Q \times \mathbb{R}_+ \times \Delta \cup \{\mathsf{unlock}\}$. The initial configuration is still $\gamma_0^{N_0}$ defined by $\gamma_0(q_0, 0) = N_0$ and $\gamma_0(q, x) = 0$ otherwise. Note that the only difference between two instances of the dynamic probabilistic timed network, $[\![\mathcal{P}_\Lambda^{N_0}]\!]$ and $[\![\mathcal{P}_\Lambda^{N_0'}]\!]$, lies in their respective initial state.

We now detail the transition function T. For every $\gamma \in \mathsf{Conf}$, $\alpha \in \mathsf{Act} \cup \mathbb{R}_+$ and every transition $\gamma \xrightarrow{\alpha,p} \gamma'$ (in T^N, for N the size of γ), there are transitions $\gamma \xrightarrow{\alpha,p\cdot p'} \gamma''$ in T for every configuration γ'' that can be obtained from γ' through disappearance followed by creation of processes, with the appropriate probability. Most likely, γ'' contains a different number of processes than γ and γ'. Let us explain how the probability p' is defined. We write $\mathbf{p}_-(\gamma_1, \gamma_2)$ for the probability to obtain γ_2 from γ_1 when processes disappear; recall that each process can disappear with probability λ_-. Similarly, $\mathbf{p}_+(\gamma_1, \gamma_2)$ is the probability to obtain γ_2 from γ_1 when processes are created; recall that for every integer k, k processes are created (in $(q_0, 0)$) with probability $\lambda_+^k(1-\lambda_+)$. Using these notations, the probability to move from γ' to γ'' by disappearance and creation of processes is $p' = \sum_{\gamma_1} \mathbf{p}_-(\gamma', \gamma_1) \cdot \mathbf{p}_+(\gamma_1, \gamma'')$.

An *execution* in $[\![\mathcal{P}_\Lambda^{N_0}]\!]$ is a finite or infinite sequence $\rho = \gamma_0 \to \gamma_1 \to \gamma_2 \cdots$, where the transitions correspond to time elapsing, internal actions, communications or the special action unlock.

2.3 Schedulers

Schedulers (also called strategies or policies) resolve the non-determinism in Markov decision processes. We restrict to stationary deterministic schedulers (sometimes also called pure memoryless schedulers), that make their decision based on the current configuration only.

▶ **Definition 5** (Scheduler). A *scheduler* for the timed Markov decision process $[\![\mathcal{P}_\Lambda^{N_0}]\!] = (\mathsf{Conf}, \gamma_0^{N_0}, \mathsf{T}, \mathsf{Act} \cup \mathbb{R}_+)$ is a function $\sigma : \mathsf{Conf} \to \mathbb{R}_+ \times \mathsf{Act}$, specifying in each configuration the delay and discrete action to perform, and such that

(i) whenever $\sigma(\gamma) = (\tau, (q, x, \alpha))$, then $(\gamma + \tau)(q, x) > 0$ and α is enabled in (q, x), and

(ii) whenever $\sigma(\gamma) = (\tau, \mathsf{unlock})$, γ is a deadlock configuration.

Recall that $\mathsf{Act} = (Q \times \mathbb{R}_+ \times \Delta) \cup \{\mathsf{unlock}\}$. In words, a scheduler resolves the non-determinism by choosing in each configuration a delay τ, followed by some discrete action. Condition (i) ensures that σ only chooses enabled discrete actions; moreover, thanks to condition (ii), if the system is deadlock, σ chooses the special action unlock.

The dynamic probabilistic timed network $\mathcal{P}_\Lambda^{N_0}$ together with a fixed scheduler σ give rise to a Markov chain with state space Conf, and in which the probability measure over executions of $\mathcal{P}_\Lambda^{N_0}$, defined in a standard way, is written $\mathbb{P}_\sigma$.

2.4 Problem formulation

We are now in a position to introduce relevant verification questions for dynamic probabilistic timed networks. In this paper, we focus on qualitative reachability problems.

Let $q_f \in Q$, and ρ an execution of $\mathcal{P}_\Lambda^{N_0}$. Execution ρ satisfies $\Diamond q_f$, denoted $\rho \models \Diamond q_f$, if there exists a configuration γ along ρ with $\gamma(q_f, x) > 0$ for some arbitrary clock value x. Given a scheduler σ for $\mathcal{P}_\Lambda^{N_0}$, we write $\mathbb{P}_\sigma(\mathcal{P}_\Lambda^{N_0} \models \Diamond q_f)$ for the probability under σ of the set of executions ρ with $\rho \models \Diamond q_f$. Further, $\min_\sigma \mathbb{P}_\sigma(\mathcal{P}_\Lambda^{N_0} \models \Diamond q_f)$ (resp. $\max_\sigma \mathbb{P}_\sigma(\mathcal{P}_\Lambda^{N_0} \models \Diamond q_f)$) denotes the minimum (resp. maximum) of these values among all possible schedulers.

We define the following family of decision problems, for $\texttt{opt} \in \{\min, \max\}$, $\texttt{b} \in \{0, 1\}$ and $\sim \in \{<, =, >\}$

$\textsc{Reach}_{\texttt{opt}}^{\sim\texttt{b}}$
Input: A probabilistic timed protocol $\mathcal{P}$, a rate pair Λ, and a control state $q_f \in Q$.
Question: Does there exist $N_0 \in \mathbb{N}_{>0}$ such that $\texttt{opt}_\sigma\ \mathbb{P}_\sigma(\mathcal{P}_\Lambda^{N_0} \models \Diamond q_f) \sim \texttt{b}$?

Note that we state the decision problems in an existential way; however negating the condition $\sim \texttt{b}$, we equivalently deal with universal quantification over network sizes.

3 Decidability status in the static case

In this section, we consider the static case, i.e., when $\lambda_- = \lambda_+ = 0$, and start by establishing simple decidability results.

▶ **Proposition 6.** *The problems* $\textsc{Reach}_{\max}^{=0}$, $\textsc{Reach}_{\max}^{<1}$, *and* $\textsc{Reach}_{\max}^{>0}$ *are decidable for static probabilistic timed networks.*

Proof. First, remark that $\max_\sigma \mathbb{P}_\sigma(\mathcal{P}^N \models \Diamond q_f) \leq \max_\sigma \mathbb{P}_\sigma(\mathcal{P}^{N+1} \models \Diamond q_f)$. Indeed in the network $\mathcal{P}^{N+1}$, some schedulers only involve N processes. Hence, there exists N such that $\max_\sigma \mathbb{P}_\sigma(\mathcal{P}^N \models \Diamond q_f) = 0$ if and only if $\max_\sigma \mathbb{P}_\sigma(\mathcal{P}^1 \models \Diamond q_f) = 0$. And the same holds for the case < 1. The decidability of $\textsc{Reach}_{\max}^{=0}$, and $\textsc{Reach}_{\max}^{<1}$ thus derive from the decidability of $\max_\sigma \mathbb{P}_\sigma(M \models \Diamond q_f) = 0$ (resp. < 1) for M a finite-state Markov decision process [18, 10].

For $\textsc{Reach}_{\max}^{>0}$, observe that, for $N \in \mathbb{N}$, $\max_\sigma \mathbb{P}_\sigma(\mathcal{P}^N \models \Diamond q_f) > 0$ if and only if there exists a scheduler σ with $\mathbb{P}_\sigma(\mathcal{P}^N \models \Diamond q_f) > 0$ if and only if there exists an execution ρ in $\mathcal{P}^N$ with $\rho \models \Diamond q_f$. The decidability of $\textsc{Reach}_{\max}^{>0}$ is therefore a consequence of the decidability of the parameterized reachability problem in the non-probabilistic case, established in [3]. ◀

We now consider the remaining cases, and prove their undecidability, already for the restricted class of untimed probabilistic networks.

▶ **Theorem 7.** *The problems* $\textsc{Reach}_{\max}^{=1}$, $\textsc{Reach}_{\min}^{=0}$, $\textsc{Reach}_{\min}^{>0}$, $\textsc{Reach}_{\min}^{<1}$ *and* $\textsc{Reach}_{\min}^{=1}$ *are undecidable for static probabilistic (timed) networks.*

Proof sketch. Let us explain the key ideas of the undecidability proof for $\textsc{Reach}_{\min}^{<1}$, which is inspired by techniques from [1]. We reduce from the halting problem of a deterministic infinitely testing 2-counter machine $\mathcal{M}$, which is known to be undecidable [19]. From $\mathcal{M}$, we build a probabilistic protocol $\mathcal{P}$, such that, for every $N \in \mathbb{N}_{>0}$, the network $\mathcal{P}^N$ weakly simulates $\mathcal{M}$: each execution either faithfully simulates $\mathcal{M}$ or a simulation error is detected, and some process is in an error state.

First, one process is selected to play the role of the *controller*, that keeps track of the control state in $\mathcal{M}$. The other processes will serve to encode the values of the counters, and are grouped in state $\texttt{idle}$. The increment of counter $\texttt{c}_i$ is represented by moving a process from `idle` to state $\mathtt{c}_i$ where the counter value is encoded. This can be done by two communications: one from the controller to the processes in `idle`, followed by one from one "counter" process to the controller. For the *test to zero decrement* operation of counter $\mathtt{c}_i$, the controller randomly guesses whether the counter value is zero or not. If it guesses zero while $\mathtt{c}_i > 0$, all processes in $\mathtt{c}_i$ move to an error state $\mathtt{err}_z$. Symmetrically, if it guesses non-zero while $\mathtt{c}_i = 0$, the infeasibility of the decrement will force the controller to move to an other error state `err`. If the guess is correct, the simulation continues, and no process are in error states. The only way to avoid error states is thus to faithfully simulate $\mathcal{M}$. Indeed, in the probabilistic protocol $\mathcal{P}$, the only blocking state for the controller is $\mathtt{k}_{acc}$, representing the accepting state of $\mathcal{M}$, and from all other states it can reach state `err`. As a consequence, all maximal executions reach `err` except for the ones which faithfully simulate $\mathcal{M}$ and end in $\mathtt{k}_{acc}$. Moreover, for N large enough there exists an execution ρ in $\mathcal{P}^N$ simulating π the unique maximal execution of $\mathcal{M}$. Assuming $\mathcal{M}$ terminates, π is finite, and thus ρ has a positive probability under some scheduler σ. We derive that $\min_\sigma \mathbb{P}_\sigma(\mathcal{P}^N \models \Diamond(\mathtt{err} \cup \mathtt{err}_z)) < 1$.

Assume now that $\mathcal{M}$ does not terminate, and thus its unique execution π contains infinitely many zero tests, with a non-zero counter value. Under any scheduler σ, the probability of executions simulating faithfully π is then zero. As a consequence reaching an error state is almost-sure, for all schedulers.

The undecidability of $\textsc{Reach}^{<1}_{\min}$ derives from the observation that this construction ensures: $\exists N \in \mathbb{N}_{>0}\ \min_\sigma \mathbb{P}_\sigma(\mathcal{P}^N \models \Diamond(\mathtt{err}_z \cup \mathtt{err})) < 1 \iff \mathcal{M}$ terminates. ◀

4 Decidability status in the dynamic case

We now turn to dynamic probabilistic timed networks, and will see that the decidability of the qualitative parameterized verification problems is recovered thanks to probabilistic disappearance and creation of processes. To establish this result, we first abstract the Markov decision process $[\![\mathcal{P}^{N_0}_\Lambda]\!]$ into a discrete MDP, using an ad-hoc region abstraction which preserves the extremal probabilities. Then, we prove that the so-called region MDP enjoys the finite attractor property, and that it can be equipped with a well-quasi-ordering on its configurations. These two properties entail the decidability of the qualitative verification questions in the region MDP that are equivalent to the initial parameterized verification problems in the network.

4.1 Region abstraction

The classical region abstraction for timed automata, presented in the seminal paper [6], is based on the observation that the relevant information in clock valuations consists of the integer part of each clock (up to the maximal constant appearing in guards) and the ordering of their fractional parts. In our context, since the number of processes is unbounded (hence the number of clocks is unbounded), the region abstraction cannot be used directly. Still, based on classical regions, we present an equivalence relation over configurations.

For $x \in \mathbb{R}_+$ a non-negative real, we denote $\lfloor x \rfloor$ its integer part and $\{x\}$ its fractional part. Note that $x = \lfloor x \rfloor + \{x\}$.

▶ **Definition 8** (Region equivalence).
Let $b \in \mathbb{N}$. Two configurations $\gamma_1 = \langle (q_1, x_1), \ldots, (q_n, x_n) \rangle$ and $\gamma_2 = \langle (p_1, y_1), \ldots, (p_m, y_m) \rangle$ are *region equivalent*, denoted $\gamma_1 \approx_b \gamma_2$ if there exists a bijection $h : [\![1, n]\!] \to [\![1, m]\!]$ such that the following conditions hold, $\forall i, j \in [\![1, n]\!]$:

(i) $q_i = p_{h(i)}$: states of processes agree,

(ii) $(\lfloor x_i \rfloor \leq b) \vee (\lfloor y_{h(i)} \rfloor \leq b) \Rightarrow \lfloor y_{h(i)} \rfloor = \lfloor x_i \rfloor$: integer part of clocks agree up to b,

(iii) $(\{x_i\} = 0) \Leftrightarrow (\{y_{h(i)}\} = 0)$: clocks with integer value agree,

(iv) for $\sim \in \{<, =, >\}$, $(\{x_i\} \sim \{x_j\}) \Leftrightarrow (\{y_{h(i)}\} \sim \{y_{h(j)}\})$: the two orderings of fractional parts coincide.

In Definition 8, the region equivalence is indexed by a bound b. For a given protocol $\mathcal{P}$ this bound is set as the maximal constant appearing in guards. For simplicity, we omit it in what follows and simply write $\approx$. Given $\gamma \in \mathsf{Conf}$ a configuration, $[\gamma]$ denotes its equivalence class for $\approx$, and is called a *region-configuration*. As an example with $b = 1$, $\langle (q_1, 0), (q_2, 2.1) \rangle \approx \langle (q_1, 0), (q_2, 4.5) \rangle \not\approx \langle (q_1, 0.8), (q_2, 4.5) \rangle$.

Similarly to classical regions in timed automata, two region-equivalent configurations exhibit similar future behaviors in $\mathcal{P}_\Lambda^{N_0}$. This is formalized in the following proposition:

▶ **Proposition 9.** *Let γ_1 and γ_2 be two configurations. If $\gamma_1 \approx \gamma_2$, then γ_1 and γ_2 are time-abstract bisimilar,* i.e.*:*

- $\forall \tau_1 \in \mathbb{R}_+$, $\forall \gamma_1 \xrightarrow{\tau_1} \gamma_1'$, $\exists \tau_2 \in \mathbb{R}_+$, $\exists \gamma_2' \in [\gamma_1']$ *such that* $\gamma_2 \xrightarrow{\tau_2} \gamma_2'$*;*
- $\forall \alpha \in \mathsf{Act}$, $\forall \gamma_1 \xrightarrow{\alpha,p} \gamma_1'$, $\exists \gamma_2' \in [\gamma_1']$ *such that* $\gamma_2 \xrightarrow{\alpha,p} \gamma_2'$.

Thanks to Proposition 9, the MDP $[\![\mathcal{P}_\Lambda^{N_0}]\!]$ can be abstracted into its quotient by $\approx$, a countable MDP, formally defined as follows:

▶ **Definition 10** (Region MDP)**.** The *region MDP* of a dynamic network $\mathcal{P}_\Lambda^{N_0}$ is $\mathcal{R}([\![\mathcal{P}_\Lambda^{N_0}]\!]) = (\mathsf{Conf}_\approx, [\gamma_0^{N_0}], \mathsf{T}_\approx, \mathsf{Act}_\approx)$ defined from $[\![\mathcal{P}_\Lambda^{N_0}]\!] = (\mathsf{Conf}, \gamma_0^{N_0}, \mathsf{T}_\Lambda, \mathsf{Act})$ by $\mathsf{Conf}_\approx = \{[\gamma] \mid \gamma \in \mathsf{Conf}\}$, $\mathsf{Act}_\approx = \mathsf{Act} \cup \{\mathtt{TSucc}\}$, and $\mathsf{T}_\approx \subseteq \mathsf{Conf}_\approx \times \mathsf{Act}_\approx \times \mathsf{Dist}(\mathsf{Conf}_\approx)$ is such that

- $[\gamma_1] \xrightarrow{\mathtt{TSucc}} [\gamma_2] \in \mathsf{T}_\approx$ as soon as $\exists \tau \in \mathbb{R}_+$, $\exists \gamma_1 \xrightarrow{\tau} \gamma_2 \in \mathsf{T}_\Lambda$ such that $\gamma_1 \not\approx \gamma_2$ and $\forall \tau' \leq \tau$, $\gamma_1 + \tau' \in [\gamma_1] \cup [\gamma_2]$;
- $[\gamma_1] \xrightarrow{\alpha,p} [\gamma_2] \in \mathsf{T}_\approx$ as soon as $\exists x$, $\exists \gamma_1 \xrightarrow{\alpha_x,p'} \gamma_2 \in \mathsf{T}_\Lambda$, and $p = \sum_{\gamma_2' \approx \gamma_2} \{p' \mid \gamma_1 \xrightarrow{\alpha_x,p'} \gamma_2'\}$,

Remark that the region MDP $\mathcal{R}([\![\mathcal{P}_\Lambda^{N_0}]\!])$ is well-defined, thanks to Proposition 9. First, the successor by $\mathtt{TSucc}$ is well-defined, since the next time-successor is uniform inside an equivalence class for $\approx$. Second, the existence of successors by discrete actions are also uniform inside an equivalence class, and the sum $\sum_{\gamma_2' \approx \gamma_2} \{p' \mid \gamma_1 \xrightarrow{\alpha_x,p'} \gamma_2'\}$ does not depend on γ_1 but only on $[\gamma_1]$. Last, the above sum is well-defined since there are only finitely many γ_2' such that $\gamma_2' \approx \gamma_2$ and $\gamma_1 \xrightarrow{\alpha_x,p'} \gamma_2'$.

A *scheduler* for $\mathcal{R}([\![\mathcal{P}_\Lambda^{N_0}]\!])$ is a mapping $\mathcal{U} : \mathsf{Conf}_\approx \to \mathsf{Act}_\approx$ resolving the non-determinism and such that $\mathcal{U}([\gamma])$ is enabled in $[\gamma]$. Given $q_f \in Q$ a state of the protocol $\mathcal{P}$, we write $\mathcal{R}([\![\mathcal{P}_\Lambda^{N_0}]\!]) \models \Diamond q_f$ for the set of executions in $\mathcal{R}([\![\mathcal{P}_\Lambda^{N_0}]\!])$ that eventually visit some $[\gamma]$ with $\gamma(q_f, x) > 0$ for some $x \in \mathbb{R}_+$. When a scheduler $\mathcal{U}$ is fixed, $\mathbb{P}_\mathcal{U}(\mathcal{R}([\![\mathcal{P}_\Lambda^{N_0}]\!]) \models \Diamond q_f)$ is the probability in $\mathcal{R}([\![\mathcal{P}_\Lambda^{N_0}]\!])$ under $\mathcal{U}$ of reaching q_f. As intended, the region MDP $\mathcal{R}([\![\mathcal{P}_\Lambda^{N_0}]\!])$ is equivalent to $[\![\mathcal{P}_\Lambda^{N_0}]\!]$ in the following sense:

▶ **Proposition 11.** *Let $\sim \in \{<, =, >\}$ and $\mathtt{b} \in \{0, 1\}$.*

$$\exists \sigma \ \mathbb{P}_\sigma(\mathcal{P}_\Lambda^{N_0} \models \Diamond q_f) \sim \mathtt{b} \iff \exists \mathcal{U} \ \mathbb{P}_\mathcal{U}(\mathcal{R}([\![\mathcal{P}_\Lambda^{N_0}]\!]) \models \Diamond q_f) \sim \mathtt{b} \ .$$

The right-to-left implication of the equivalence is the easiest: From a scheduler $\mathcal{U}$ in the region MDP, one can define a scheduler σ for the original network by mimicking the discrete actions, and transforming the abstract delays ($\mathtt{TSucc}$) into concrete ones. The other implication is more subtle since a scheduler in the network may well take different decisions for equivalent configurations. Yet, we can prove that for qualitative properties, one can always consider schedulers that are region-uniform.

A consequence of Proposition 11 is that for $\mathtt{opt} \in \{\min, \max\}$,

$$\mathtt{opt}_\sigma \mathbb{P}_\sigma(\mathcal{P}_\Lambda^{N_0} \models \Diamond q_f) \sim \mathtt{b} \iff \mathtt{opt}_\mathcal{U} \mathbb{P}_\mathcal{U}(\mathcal{R}(\llbracket \mathcal{P}_\Lambda^{N_0} \rrbracket) \models \Diamond q_f) \sim \mathtt{b} \ .$$

Therefore the parameterized verification problems are equivalent in the dynamic probabilistic timed network and its region MDP.

4.2 Deciding parameterized problems on the region MDP

Using Proposition 11, the qualitative reachability problems are equivalent in $\mathcal{P}_\Lambda^{N_0}$ and in $\mathcal{R}(\llbracket \mathcal{P}_\Lambda^{N_0} \rrbracket)$. We now expose how to decide them in the region MDP. The decidability relies on two key arguments: first, $\mathcal{R}(\llbracket \mathcal{P}_\Lambda^{N_0} \rrbracket)$ admits a finite attractor, and second, in $\mathcal{R}(\llbracket \mathcal{P}_\Lambda^{N_0} \rrbracket)$, the predecessor operator is effective and preserves upward-closure for some well-quasi-ordering. The decidability can then be derived by applying similar techniques as for nondeterministic and probabilistic lossy channel systems [9, 11].

The finite attractor property was introduced originally for probabilistic lossy channel systems (pLCS) and states that Markov chains induced by pLCS admit a finite recurrent set [2, 8]. Roughly said, some results for finite Markov chains extend to infinite Markov chains with a finite attractor. A Markov chain is said to have a *finite attractor*, if there exists a finite set of states that is visited infinitely often almost-surely (*i.e.*, with probability 1). Further, a Markov decision process has a finite attractor if there exists a finite set of states which is an attractor under all possible schedulers.

▶ **Proposition 12.** *The set* $\{[\emptyset]\} \subseteq 2^{\mathsf{Conf}_\approx}$ *is an attractor for* $\mathcal{R}(\llbracket \mathcal{P}_\Lambda^{N_0} \rrbracket)$.

▶ Remark. In the region MDP, the empty region-configuration $[\emptyset]$, with no process, forms an attractor. Since any initial region-configuration $[\gamma_0^{N_0}] = [\langle (q_0, 0)^{N_0} \rangle]$ can be reached, with positive probability, from the empty region-configuration, this entails that along infinite executions, almost-surely, all possible initial region-configurations $[\gamma_0^{N_0}]$ are visited infinitely often. A consequence is that the validity of a qualitative property does not depend on the initial number of processes N_0. Moreover the extremal probabilities of reaching a configuration with some process in q_f is either 0 or 1.

These observations can be thought as drawbacks of the dynamic probabilistic timed network model. Yet, we argue that the setting and the decidability results to come can easily be adapted to the case where a fixed number of processes cannot disappear. This would lead to a realistic model which, for example, can represent a system with fixed antennas and a parametric number of wireless devices that disappear and are created following probabilistic laws. More importantly, it would yield a system still with a finite attractor (the finite set of possible states for the antennas), but in which the reachability probabilities can be different from 0 and 1. The fact that the empty region-configuration forms a finite attractor, should thus not be seen as an unrealistic feature.

Now that the existence of a finite attractor has been established, we define an appropriate partial order $\preceq$ on $\mathsf{Conf}_\approx$. Intuitively, region-configuration c is smaller than c', if we can remove some processes of a configuration of $\gamma' \in c'$ and obtain a configuration $\gamma \in c$. We also add a side-condition on the clocks with integer value, which is necessary to obtain a good property on the predecessor operator. Formally:

▶ **Definition 13** (Ordering on region-configurations)**.** The order $\preceq \subseteq \mathsf{Conf}_\approx \times \mathsf{Conf}_\approx$ is defined as follows: for $c, c' \in \mathsf{Conf}_\approx$, $c \preceq c'$ if there exists $\gamma \in c$ and $\gamma' \in c'$ such that:

(i) $\forall q \in Q,\ \forall x \in \mathbb{R}_+,\ \gamma(q,x) \le \gamma'(q,x)$; and

(ii) $\sum_{q \in Q} \sum_{n \in \mathbb{N}} \gamma(q,n) = 0 \implies \sum_{q \in Q} \sum_{n \in \mathbb{N}} \gamma'(q,n) = 0$.

Adapting the proof of Higman's lemma, one obtains:

▶ **Proposition 14.** *The partial order $\preceq$ is a well-quasi-ordering.*

We now consider the upward-closure operator $\uparrow$ w.r.t. $\preceq$: given $C \subseteq \mathsf{Conf}_{\approx}$,

$$\uparrow C = \{c' \in \mathsf{Conf}_{\approx} \mid \exists c \in C \text{ such that } c \preceq c'\} \ .$$

A set $C \subseteq \mathsf{Conf}_{\approx}$ is said to be *upward-closed* whenever $C = \uparrow C$. Since $\preceq$ is a well-quasi-ordering, any non-decreasing sequence of upward-closed sets eventually stabilizes. This property will be useful in the sequel.

Last we define, in the region MDP, the predecessor operator: for $c \in \mathsf{Conf}_{\approx}$, $\mathsf{Pre}(c)$ denotes the set of region-configurations $c' \in \mathsf{Conf}_{\approx}$ that can reach c in one step. It happens that for any upward-closed set $C \subseteq \mathsf{Conf}_{\approx}$, the set $\mathsf{Pre}(C)$ can be computed, and is itself upward closed.

▶ **Proposition 15.** Pre *preserves upward-closure, is effective and satisfies* $\mathsf{Pre}(C) = \mathsf{Pre}(\uparrow C)$.

Propositions 12, 14 and 15 are decisive to obtain the decidability of parameterized verification problems in the region MDP $\mathcal{R}([\![\mathcal{P}_{\Lambda}^{N_0}]\!])$. Combined with Proposition 11 we obtain the main contribution of this paper:

▶ **Theorem 16.** *The problems* $\textsc{Reach}_{\mathtt{opt}}^{\sim\mathtt{b}}$ *are decidable for dynamic probabilistic timed networks, and are non-primitive recursive.*

Proof sketch. As an example, we explain how decidability is obtained for $\textsc{Reach}_{\max}^{>0}$. As explained above, by Proposition 11, we can consider the same decision problem in the region MDP. It thus suffices to show the decidability of whether there exists $N_0 \in \mathbb{N}$ and a scheduler $\mathcal{U}$ such that $\mathbb{P}_{\mathcal{U}}(\mathcal{R}([\![\mathcal{P}_{\Lambda}^{N_0}]\!]) \models \Diamond q_f) > 0$. We have the series of equivalences:

$$\begin{aligned} \exists N_0,\ \exists \mathcal{U}\ \mathbb{P}_{\mathcal{U}}(\mathcal{R}([\![\mathcal{P}_{\Lambda}^{N_0}]\!]) \models \Diamond q_f) > 0 \quad &\Longleftrightarrow \quad \exists \mathcal{U}\ \mathbb{P}_{\mathcal{U}}(\mathcal{R}([\![\mathcal{P}_{\Lambda}^{0}]\!]) \models \Diamond q_f) > 0 \\ &\Longleftrightarrow \quad [\emptyset] \in \mathsf{Pre}^*(q_f) \ . \end{aligned}$$

The first equivalence uses Remark 4.2, and more specifically that the qualitative reachability does not depend on N_0 because of the finite attractor property. Now, recall that q_f represents the set where some process is in state q_f, and is thus upward-closed. Since the Pre operator preserves upward-closure and is effective, and because $\preceq$ is a well-quasi-ordering, the set $\mathsf{Pre}^*(q_f)$ can be computed effectively by successive iterations of Pre. It then suffices to test whether the region-configuration $[\emptyset]$ belongs to $\mathsf{Pre}^*(q_f)$ to decide $\textsc{Reach}_{\max}^{>0}$.

For the lower-bound, we perform a reduction from the reachability problem in lossy channel systems, which is known to be non-primitive recursive. ◀

5 Conclusion

We studied qualitative parameterized verification problems for a model of network of many identical probabilistic timed processes. Interesting qualitative questions turn out to be undecidable in the static case, and become decidable under the assumption that processes can be created and disappear. The complexity of the decision algorithms for parameterized reachability questions in dynamic probabilistic timed networks is theoretically high, but it would be worth implementing our decision algorithms into a prototype to demonstrate whether they can be used in practice on academic case studies.

Interesting research directions for future work, are to consider distributed schedulers, where the choice of each process is independent of the state of the other processes, and to tackle quantitative verification problems. For the latter, adapting existing techniques for Markov chains to Markov decision processes could allow one to approximate maximum expected time to reachability, or to compute optimal values of the parameter.

References

1 P. A. Abdulla, N. Ben Henda, and R. Mayr. Decisive Markov chains. *Logical Methods in Computer Science*, 3(4), 2007.

2 P. A. Abdulla, N. Bertrand, A. Rabinovich, and Ph. Schnoebelen. Verification of probabilistic systems with faulty communication. *Information and Computation*, 202(2):141–165, 2005.

3 P. A. Abdulla, G. Delzanno, O. Rezine, A. Sangnier, and R. Traverso. On the verification of timed ad hoc networks. In *Proc. 9th Int. Conference on Formal Modeling and Analysis of Timed Systems (ForMATS'11)*, volume 6919 of *LNCS*, pages 256–270. Springer, 2011.

4 P. A. Abdulla, J. Deneux, and P. Mahata. Multi-clock timed networks. In *Proc. 19th IEEE Symposium on Logic in Computer Science (LICS'04)*, pp. 345–354. IEEE CS, 2004.

5 P. A. Abdulla and B. Jonsson. Model checking of systems with many identical timed processes. *Theoretical Computer Science*, 290(1):241–263, 2003.

6 R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

7 K. Apt and D. Kozen. Limits for automatic verification of finite-state concurrent systems. *Information Processing Letters*, 22:307–309, 1986.

8 C. Baier, N. Bertrand, and Ph. Schnoebelen. A note on the attractor-property of infinite-state Markov chains. *Information Processing Letters*, 97(2):58–63, 2006.

9 C. Baier, N. Bertrand, and Ph. Schnoebelen. Verifying nondeterministic probabilistic channel systems against ω-regular linear-time properties. *ACM Transactions on Computational Logic*, 9(1), 2007.

10 C. Baier and J.-P. Katoen. *Principles of model checking.* MIT Press, 2008.

11 N. Bertrand and Ph. Schnoebelen. Computable fixpoints in well-structured symbolic model checking. *Formal Methods in System Design*, 2013. To appear.

12 E. Clarke, O. Grumberg, and S. Jha. Verifying parameterized networks using abstraction and regular languages. In *Proc. 6th Int. Conference on Concurrency Theory (CONCUR'95)*, volume 962 of *LNCS*, pages 395–407. Springer, 1995.

13 C. Daws. Symbolic and parametric model checking of discrete-time Markov chains. In *Proc. of 1st International Colloquium on Theoretical Aspects of Computing (ICTAC'04)*, volume 3407 of *Lecture Notes in Computer Science*, pages 280–294. Springer, 2004.

14 G. Delzanno, A. Sangnier, and G. Zavattaro. Parameterized verification of ad hoc networks. In *Proc. 21th Int. Conference on Concurrency Theory (CONCUR'10)*, volume 6269 of *LNCS*, pages 313–327. Springer, 2010.

15 J. Esparza, A. Finkel, and R. Mayr. On the verification of broadcast protocols. In *Proc. of 14th Annual IEEE Symposium on Logic in Computer Science (LICS'99)*, pages 352–359. IEEE Computer Society, 1999.

16 M. Fruth. Probabilistic model checking of contention resolution in the IEEE 802.15.4 low-rate wireless personal area network protocol. In *Proc. 2nd Int. Symposium on Leveraging Applications of Formal Methods (IsoLA'06)*, pages 290–297. IEEE, 2006.

17 F. Gretz, J.-P. Katoen, and A. McIver. Operational versus weakest precondition semantics for the probabilistic guarded command language. In *Proc. 9th Int. Conference on Quantitative Evaluation of SysTems (QEST'12)*, 2012.

18 M. Kwiatkowska, G. Norman, D. Parker, and J. Sproston. *Modeling and Verification of Real-Time Systems: Formalisms and Software Tools*, chapter Verification of Real-Time Probabilistic Systems, pages 249–288. John Wiley & Sons, 2008.

19 M. Minsky. *Computation: Finite and Infinite Machines*. Prentice Hall International, 1967.