Contributions to the verification and control of timed and probabilistic models

Nathalie Bertrand

Contributions to the verification and control of timed and probabilistic models

Mémoire d'habilitation à diriger des recherches

Nathalie Bertrand

Defended on November 16th 2015, before the committee composed of

Christel Baier, TU Dresden	*examinator*
Paul Gastin, ENS Cachan	*examinator*
Thierry Jéron, Inria Rennes	*examinator*
Marta Kwiatkowska, Oxford University	*reviewer*
Catuscia Palamidessi, Inria Saclay	*examinator*
Prakash Panangaden, McGill University	*reviewer*
Sophie Pinchinat, Université Rennes 1	*president*
Jean-François Raskin, ULB	*reviewer*

Contents

Chapter 1

Introduction

In the last decade, softwares broke into our everyday lives, embedded in transportation systems with the example of cruise control system in cars, telecommunication means with the popular smart phones, energy services with remote meter readings and even health-related artefacts such as pacemakers, to cite just a few. The growing presence of embedded systems offered new capabilities, but it often comes with a threat, since software failures can have dramatic consequences, in terms of human lives or prohibitive costs. Our need to rely more and more on such critical complex systems, motivates the use of formal methods for their design, validation and supervision.

Model-based techniques

My work is centered on model-based techniques, that is, methodologies that are based on mathematical abstractions, and require models both of the system, and of the requirements it has to meet. Among these techniques, the importance of *model checking* was acknowledged when Ed Clarke, Allen Emerson and Joseph Sifakis received the Turing award in 2007. Model checking algorithms allow to automatically verify that the model of a system satisfies the model of the requirements. Assuming that the models (of both the system and the requirements) are accurate enough, this certifies that the system meets the requirements.

Testing is an alternative to model checking that can be applied either when the model of the system is unknown, or in case the model checking algorithms are prohibitive in terms of complexity. Compared to model checking, testing trades off exhaustiveness for efficiency. *Model-based testing* aims at automatically generating test cases from a model of the specification of the system. These test cases can then be run on the implementation itself in order to detect non conformance to the expected behaviour.

Another model-based technique that is complementary to model checking is *controller synthesis*. In case the model of a system does not satisfy the model of the requirements, one option is to synthesize a controller that restricts the behaviours of the system so as to meet the requirements. Controller synthesis can be used to fix bugs in existing systems, or in the design process to obtain correct-by-construction sofwares.

Rich models for complex systems

In order to faithfully represent real-life systems, models should incorporate several features including timing contraints, probabilities, unknown parameters and imperfect information.

Most systems not only have constraints on the order in which the events happen but also quantitative constraints on the delays between them. Examples of such timing constraints are time-outs in communication or security protocols. For these so-called *real-time* systems, several models were introduced such as time and timed Petri nets, and timed automata introduced by Alur and Dill [AD90, AD94], and for which they received in 2008 a CAV award. In these three models, timing requirements are expressed through constraints on dense-time variables that all evolve at the same speed, and represent either the time elapsed since a given action event, or since a transition is enabled.

Several unpredictable behaviours are conveniently modeled using *probabilities*. For example, processes going down in a network, and messages losses in communication protocols can naturally be abstracted by probabilistic events. This is also true for failure rates for physical components. Another example is the arrival of new jobs in a queue. Also, probabilities are sometimes inherent to programs, such as in randomized algorithms for routing in networks to guarantee good average performances, or to break symmetry in distributed protocols, with the famous example of Lehmann and Rabin's algorithm for the dining philosophers problem [LR81].

Parameters may also be used in a model, when a value is not known precisely, possibly also because the system has to be generic and adapt to various situations. Following the example of distributed protocols, typically the numbers of participants is variable, and the protocol should work independently of that parameterised value.

Last, users usually only have *partial information* on the system when they interact with it. This can be a design choice for security reasons when some data is kept secret to external users. It can also be a consequence of the system being too large or spatially distributed, so that it is not fully observable. A prominent example of models for such systems is the one of partially observable Markov decision processes (see *e.g.* [Son71, PT87]), that have a wide range of applications domains, such as machine maintenance, autonomous robot, and planning.

These four ingredients, time, probabilities, parameters and partial observation, are all features that one would naturally use to faithfully model systems. Of course, rich models come with a cost since they call for more advanced verification methodologies that are often computationally harder: tractability has to be traded off for expressiveness.

Contents of this thesis

My work in the last decade focused on complex systems that combine several of these aspects. Although I mostly contributed to the subfield of formal methods for probabilistic systems, I also had some contribution for (non probabilistic) timed systems. My work started with model checking, and gradually diversified to other formal methods techniques including model-based testing, diagnosis and control. In each of my contributions, I have been interested in decidability and complexity issues. The techniques I used are varied; they notably include abstractions, calculus, fixpoint computations, and well quasi orders. This habilitation thesis reports on a selection of my contributions after obtaining my PhD in 2006.

In **Chapter 2**, we give an overview of our contributions to the verification of non-probabilistic timed systems. Precisely, we developed increasingly efficient algorithms for the determiniza-

tion of timed automata, a well established model for real-time systems. Determinization is a key step in many validation methodology, so that our work on determinization of timed automata enabled to develop complementary techniques for the analysis of real-time sytems, such as model-based test generation.

This series of contributions started in 2008 in a collaboration with Christel Baier, Patricia Bouyer and Thomas Brihaye, and continued within the context of Amélie Stainer's PhD thesis under the joint supervision of Thierry Jéron. Chapter 2 is based on the papers [BBBB09, BSJK11, BJSK11, BJSK12, BSJK15].

In **Chapter 3**, we report on a series of contributions on the validation of stochastic timed automata, a model that we introduced to represent systems with timing contraints in which the delays are subject to randomization. Because they combine dense-time variables and probability distributions, even the model checking of such automata is not immediate. We first addressed questions such as whether an untimed property is satisfied almost surely, by providing a finite state abstraction which may modify the precise probability values but preserves positivity, so that it suffices for the qualitative model checking of stochastic timed automata. This coarse abstraction is not suitable for quantitative analysis, *i.e.* the estimation of the probability that a property is satisfied. Yet, we could exploit the structure of the stochastic process undelying some subclasses of stochastic timed automata to provide an approximation algorithm for such measures. In our latest work on the topic, we considered control problems for stochastic timed automata, established that optimal control strategies exist for simple objectives, and provided a decision procedure for the existence of limit-sure control strategies when optimal strategies do not necessarily exist.

The contents of Chapter 3 is based on the papers [BBB$^+$07, BBB$^+$08, BBBM08, BS12, BBH$^+$13, BBG14, BBB$^+$14]. This line of work was initiated while I was a post doc at TU Dresden, and continued when I moved to Inria Rennes. This research direction gave me the opportunity to work with many colleagues, including Sven Schewe from Liverpool University where I did a 9-month sabbatical in 2012. Recurrent and main collaborators on this topic are Patricia Bouyer and Thomas Brihaye, with whom we defined the model at first, and coined the name stochastic timed automata.

In **Chapter 4**, we review various contributions to the field of control of partially observable probabilistic systems. We start with probabilistic ω-automata, that can encode planning problems in a random environment, assuming the controller has no information on the state of the system, and yet must fulfill an infinitary property. Partially observable Markov decision processes (POMDP), a well-known model in Artificial Intelligence, generalize probabilistic ω-automata, since the controller is partially informed. We tackled the problem of minimizing the worst-case and average information needed for a controller to ensure a simple objective. In the even broader context of stochastic games under partial observation, we proved that for most objectives, the games are qualitatively determined, meaning that assuming the first player does not have an almost-surely winning strategy, then its opponent has a uniform positively winning counterstrategy. In a last contribution, we considered the more practical problem of fault diagnosis for probabilistic systems. Beyond the neat definition of the diagnosability problem and proving its decidability, we tackled the problem of controlling a system in order to ensure that faults are almost-surely diagnosed. This control problem can be rephrased into the existence of a strategy in a POMDP for an *ad hoc* objective.

The contributions on which Chapter 4 is based are all concerned with the control of partially observable probabilistic systems. However, they are the fruit of diverse collaborations and were

obtained in various contexts. Studying probabilistic ω-automata [BBG08, BBG12] during my post doc with Christel Baier and Marcus Größer, was my first contact with the control of probabilistic systems under partial observation. I later continued this line of work with Blaise Genest and Hugo Gimbert on POMDP and 2-player stochastic games with signals [BGG09, BG11]. Last, the application to fault diagnosis in probabilistic systems was initiated with Éric Fabre, Stefan Haar, Serge Haddad and Loïc Hélouët and continued in the context of Engel Lefaucheux's Master thesis, jointly supervised with Serge Haddad [BFH+14, BHL14].

In **Chapter 5** we report on our recent work on parameterised verification of probabilistic systems, in the context of Paulin Fournier's PhD thesis. We define adequate models for networks of identical probabilistic processes that communicate through selective broadcast, and look for verification techniques independently on the number of processes involved in the network. The semantics of such networks is an infinite state Markov decision process, in which configurations describe the state of each node in the network. Our parameterised verification algorithms exploit monotony properties and the underlying well structured transition system, in order to tackle qualitative properties such as almost-sure reachability of an unsure configuration.

Chapter 5 is based on papers [BF13, BFS14], and was the opportunity for me to discover parameterised verification, thanks to Arnaud Sangnier, who had already worked on that topic for non-probabilistic models.

Research directions on each topic are given at the end of the respective chapters and **Chapter 6** concludes this document by providing general perspectives.

Other recent contributions

This introduction ends by a short mention of some contributions that are not included in this thesis. I refer the reader to

http://people.rennes.inria.fr/Nathalie.Bertrand/publis.html

for a complete and up-to-date publication list.

Timed modal specifications Together with colleagues in Rennes, we studied how modal specifications could be extended to a timed setting. We came up with a model of event-clock timed automata with modalities, and showed that they could be the base of a complete interface theory with efficient conjunction and product operations, as well as consistency test, and quotient operation that enable incremental design [BPR09, BLPR12].

Frequency semantics In another work on purely timed systems, we defined and studied a frequency semantics for timed automata [BBBS11]. Frequency measures the proportion of time spent in specific states, and results in a quantitative language interpretation of timed automata. Although it is incomparable with the traditionnal Büchi semantics, we showed how to compute extremal frequencies for given subclasses of timed automata, and thus to decide the emptiness and universality of Boolean languages defined via a threshold on the frequency. The objective of the frequency semantics was to move towards a finer quantitative analysis of timed automata. It started during Amélie Stainer's PhD, and she later deepened our study by broadening the class of timed automata one could handle [Sta12].

Fixpoint termination in well structured systems Last but not least, let us mention a joint work with my PhD advisor Philippe Schnoebelen on the termination of fixpoint computation in infinite-state systems. We defined a generic symbolic model checking framework for well structured transition systems and provided syntactic sufficient conditions on fixpoint expressions for the iterative fixpoint computation to terminate in finite time [BS13a]. Such techniques can *e.g.* be used to compute winning regions in games on lossy channel systems with probabilistic losses [BS13b].

Chapter 2

Determinization of timed automata

Since their introduction in the 90's by Alur and Dill [AD90, AD94], timed automata have become a popular model for real-time systems. Informally, the model of timed automata extends the well known finite state automata with a set of continuous variables –called clocks– that evolve at the same speed, can be tested and reset to zero while taking a transition.

Similarly to classical finite-state automata, determinism ensures that two distinct runs cannot read the same (timed) word. The determinization, that is, the construction of an equivalent deterministic timed automaton, is used to address several problems, for which the underlying analyses depend on the observable behavior, such as implementability, diagnosis or test generation. For example, in the context of offline test generation, the specification has to be determinized in some sense, since the testing artefact needs to foresee the allowed outputs after a sequence of observations, thus the set of states after this sequence. Also, a deterministic timed automaton can easily be complemented, and it is for instance useful for automata-based model checking: given a deterministic timed automaton $\mathcal{A}_\varphi$ representing a property φ, one can easily decide whether another timed automaton satisfies the formula φ by performing the intersection with the complement of $\mathcal{A}_\varphi$, and then checking the emptiness of the language of the resulting timed automaton.

In general, restricting to the class of deterministic timed automata makes a lot of problems simpler. However considering deterministic timed automata is very restrictive: nondeterminism is often used in the modelling phase, to represent several implementation possibilities; also large or ditributed models composed of several components are nondeterministic in essence. This motivates the use of nondeterministic models, while at the same time deterministic timed automata are easier to handle.

Unfortunately, the determinization of timed automata is a real issue since, not all timed automata can be determinized [AD94] and their determinizability is undecidable [Fin06, Tri06]. Two alternatives appear to overcome this difficulty: either restricting to determinizable classes of timed automata, or building deterministic timed automata that only approximate the original language. Following the first approach, several subclasses of timed automata were defined: strongly non-Zeno timed automata [AMPS98], event-clock automata [AFH94], or timed automata with integer resets [SPKM08]. The second approach has been considered in the context of offline test generation for timed systems. Krichen and Tripakis proposed an algorithm that produces a deterministic over-approximation based on a simulation by a deterministic timed automaton with fixed number of clocks and maximal constant [KT09].

Outline of the contributions This chapter reports on our contributions to the determinization problem for timed automata. First, we proposed an exact determinization procedure, which is applicable to most determinizable timed automata [BBBB09]. We later improved that procedure using a game-based formulation and obtained an algorithm that always returns an overapproximate determinization, which is exact on all known determinizable classes of timed automata [BSJK11, BSJK15]. This latter approach can be tuned to generate test cases from nondeterministic real-time specifications offline [BJSK11, BJSK12].

2.1 Preliminaries on timed automata

Timed automata were introduced as a real-time extension of finite state automata [AD90, AD94]. In order to formally define the model, and introduce the problem of determinization, we start with notations and fix the terminology.

We denote by $X = \{x_1, \ldots, x_k\}$ a finite set of *clocks*. A *clock valuation* over X is a mapping $\nu : X \to \mathbb{R}_+$. For ν a clock valuation and $R \subseteq X$ a set of clocks, we let $\nu[R := 0]$ denote the valuation ν' defined by $\nu'(x) = 0$ for every $x \in R$ and $\nu'(x) = \nu(x)$ otherwise. A *guard* over X is a finite conjunction of expressions of the form $x \sim c$ where $x \in X$ is a clock, $c \in \mathbb{N}$ is a constant, and $\sim$ is one of the comparison symbols in $\{<, \leq, =, \geq, >\}$. We denote by $\mathcal{G}(X)$ the set of guards over X.

Definition 2.1 *A* timed automaton *(*TA*) over action alphabet Σ is a tuple $\mathcal{A} = (L, X, E$ such that: L is a finite set of locations, X is a finite set of clocks, $E \subseteq L \times \mathcal{G}(X) \times \Sigma \times 2^X \times L$ is a finite set of edges.*

States of a timed automaton $\mathcal{A} = (L, X, E$ are tuples $s = (\ell, \nu)$ consisting of a location $\ell \in L$ and a clock valuation $\nu \in \mathbb{R}_+^X$. Timed automata accept *timed words*, that is (finite or infinite) sequences of pairs consisting of an action label and a timestamp: $(a_i, t_i)_{i \in \mathbb{N}}$ with $a_i \in \Sigma$ and $(t_i)_{i \in \mathbb{N}} \in \mathbb{R}_+$ a nondecreasing sequence of non-negative reals. Given a timed word $w = (a_i, t_i)_{i \in \mathbb{N}}$, a run on w in a timed automaton $\mathcal{A} = (L, X, E$ is an alternating sequence of states $(\ell_i, \nu_i)_{i \in \mathbb{N}}$ of $\mathcal{A}$ and delays $(\tau_i)_{i \in \mathbb{N}} \in \mathbb{R}_+$ such that for every index $i \in \mathbb{N}$, there exists an edge $e_i \in E$ of $\mathcal{A}$ such that $e_i = (\ell_i, g_i, a_i, R_i, \ell_{i+1})$, $\nu_i + \tau_i \models g_i$ and $\nu_{i+1} = \nu_i[R_i := 0]$; moreover the delays in the run and timestamps of the timed word are related through the following equality: $t_i = \sum_{j \leq i} \tau_j$ for every index i. Given the notions of timed words and runs, we move to the definition of language of accepted timed words. In this document, we consider finite timed words only, and assume that a timed automaton $\mathcal{A}$ is equipped with sets of initial and final locations $L_I \subseteq L$ and $L_F \subseteq L$, respectively. The *language* accepted by a timed automaton $\mathcal{A}$ is denoted $\mathcal{L}(\mathcal{A})$ and is defined as the set of finite timed words that admit a run in $\mathcal{A}$ starting from an initial location (with null valuation) and ending in a final location.

The region abstraction Because the semantics of timed automata are infinite-state and infinitely-branching transition systems, their analysis has to be performed symbolically. The region abstraction, already presented in the seminal work by Alur and Dill [AD90, AD94], is a way to allow for the analysis of timed automata. The intuitive idea is to gather into finitely many equivalence classes, states that have equivalent behaviours with respect to untimed properties.

Given X a finite set of clocks, and $M \in \mathbb{N}$ an integer constant, the region abstraction derives from the equivalence relation $\equiv_{X,M}$ between valuations in $\mathbb{R}_+^X$ defined as follows: $\nu \equiv_{X,M} \nu'$

iff (i) for every clock $x \in X$, $\nu(x) \leq M$ iff $\nu'(x) \leq M$; (ii) for every clock $x \in X$, if $\nu(x) \leq M$, then $\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$, and (ii) for every pair of clocks $(x, y) \in X^2$ such that $\nu(x) \leq M$ and $\nu(y) \leq M$, $\{\nu(x)\} \leq \{\nu(y)\}$ iff $\{\nu'(x)\} \leq \{\nu'(y)\}$. In the latter definition, for $\alpha \in \mathbb{R}_+$, $\lfloor \alpha \rfloor$ denotes the integral part of α, and $\{\alpha\}$ its fractional part. The equivalence relation $\equiv_{X,M}$ is called the *region equivalence* for the set of clocks X w.r.t. M, and an equivalence class is called a *region*. We note Reg_M^X for the set of such regions. A region r' is a *time-successor* of a region r if there is $\nu \in r$ and $t \in \mathbb{R}_+$ such that $\nu + t \in r'$. If ν is a valuation, we will write $[\nu]$ for the region to which ν belongs.

The introduction of regions is motivated by the following classical result:

Proposition 2.1 ([AD94]) *Let $\mathcal{A}$ be a timed automaton with maximal constant M and set of clocks X. Then, the equivalence relation $\approx_{X,M}$ between configurations of $\mathcal{A}$ defined by $(\ell, \nu) \approx_{X,M} (\ell, \nu')$ iff $\nu \equiv_{X,M} \nu'$ is a time-abstract bisimulation.*

The region abstraction is a useful tool to decide untimed properties over timed automata. Yet, it has limitations and cannot be used when one considers refined timed properties, or to check for language inclusion. One way to test for language inclusion of finite state automata is to determinize one of them, complement it, and test for emptiness of the intersection with the second automaton. This approach fails for timed automata since, to the difference of finite automata, timed automata are not closed under complement, and not all timed automata can be determinized. Let us introduce deterministic and determinizable TA:

Definition 2.2 *A timed automaton $\mathcal{A} = (L, X, E$ is* deterministic *if for every pair of transitions $(\ell_1, g_1, a, Y_1, \ell_1')$ and $(\ell_2, g_2, a, Y_2, \ell_2')$, as soon as $\ell_1 = \ell_2$ and $g_1 \cap g_2 \neq \emptyset$, then $g_1 = g_2$, $Y_1 = Y_2$ and $\ell_1' = \ell_2'$.*
$\mathcal{A}$ is determinizable *if there exists a deterministic* TA *$\mathcal{B}$ with $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})$.*

As for finite automata, in a deterministic timed automaton, there is at most one run on a given input timed word. Unfortunately, not all timed automata can be determinized, and even 1-clock timed automata, that are computationally simpler than the general model for many problems, cannot always be determinized [AD94]. Also, determinizability is an undecidable problem, even when the number of clocks and maximal constant of the target automaton are fixed [Fin06, Tri06].

To circumvent the unfeasability of determinization for timed automata, two general approaches can be conducted. First, one can isolate subclasses (ideally syntactic subclasses) of timed automata that can be determinized, and provide *ad hoc* algorithms for them. This approach has *e.g.* been applied to event recording timed automata [AFH94], or integer reset timed automata [SPKM08, MK10]. The second option is to trade off unfeasability for approximation, *i.e.* to compute, for every timed automaton, a deterministic approximation. As an example, [KT09] proposes an algorithm that produces a deterministic overapproximation, with the objective to apply it to test generation for real-time systems.

2.2 Determinization procedure

In a first work [BBBB09], we proposed an exact abstract determinization procedure, in the line of the first approach, that can be turned into an effective procedure combining all benefits of *ad hoc* determinization algorithms, with no complexity overhead. The idea of this abstract

approach is to unfold the behaviour of the timed automaton into an infinite tree, adding a fresh clock at each step, that is never reset, and serves to encode the values of the original clocks. This infinite tree can then be determinized, and if some condition is met, the fresh clocks can then be reduced to a finite set, and the tree folded back into a timed automaton with finitely many locations.

In this document, we do not give all formal details of the successive steps, but rather illustrate the procedure over the example timed automaton from Figure 2.1. In the first step,

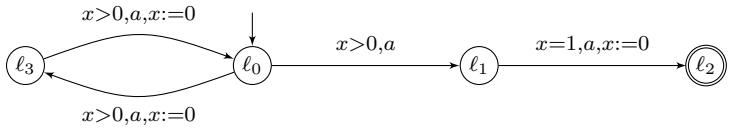

Figure 2.1: A timed automaton $\mathcal{A}$.

it is unfolded into an infinite tree, represented in Figure 2.2. Each node of the tree bears a

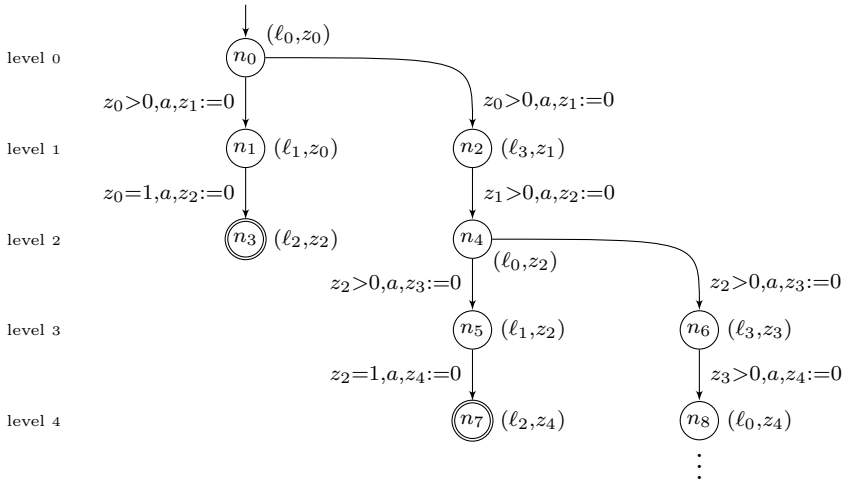

Figure 2.2: The infinite timed tree $\mathcal{A}^\infty$ associated with the timed automaton $\mathcal{A}$ of Fig. 2.1.

label that consists in a location of the original timed automaton, together with a mapping from clocks of the original TA to fresh clocks. In our illustrating example, the unique clock x is mapped in node n_1 to z_0 and in node n_2 to z_1. This reflects that clock x is not reset when entering ℓ_1, and is reset when entering ℓ_3. More generally, the mapping records which fresh clocks encode the clocks in the original timed automaton.

Note that in this infinite tree, resets are uniform by level: when entering level i, z_i and only z_i is reset. This is crucial since non uniform resets are the main obstacle to the determinization of timed automata. Without any prior assumption on the input timed automaton, this infinite tree can be determinized in a symbolic way, by first using the region abstraction. The symbolic determinization amounts to a determinization over the alphabet of pairs formed of a region and an action. More precisely, at level n, we consider the regions over fresh clocks $\{z_0, \cdots, z_n\}$ in order to split guards accordingly. As an example at level 1, the guard $0 < z_1 < z_0 = 1$ is a possible one. This process produces another infinite tree, that is now deterministic. On

our running example, it is represented on Figure 2.3, where for simplicity nodes with identical labels are merged, so that we depict a directed acyclic graph (DAG). Also, for z a clock, $z = \bot$ denotes that its value exceeds the maximal constant, thus the precise value is irrelevant.

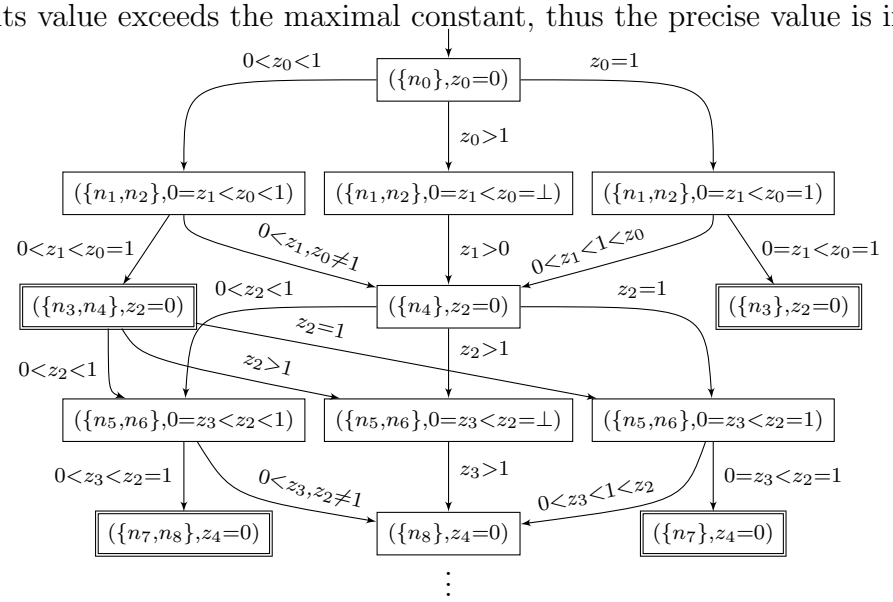

Figure 2.3: The DAG induced by the deterministic infinite tree.

In the last step of the procedure, assuming some finiteness constraint is met one can fold the infinite DAG back into a timed automaton with finitely many clocks and locations. The condition for the feasability of this refolding, is that there exists a bound γ on the number of clocks appearing at each level of the DAG. On our example, such a bound exists: one notices that at most two clocks are involved in the node labels as well as in the guards. The fresh clocks $z_0, z_1, \cdots$ can then be renamed into $x_1, \cdots, x_\gamma$, following a deterministic policy (*e.g.* the free clock with smaller index is used first). Then, identical nodes of the DAG are merged to obtain a timed automaton over clocks $x_1, \cdots, x_\gamma$. The correctness of this merging exploits that nodes of the DAG with identical labels are strong timed bisimilar. The result of this last step is illustrated on our example in Figure 2.4.

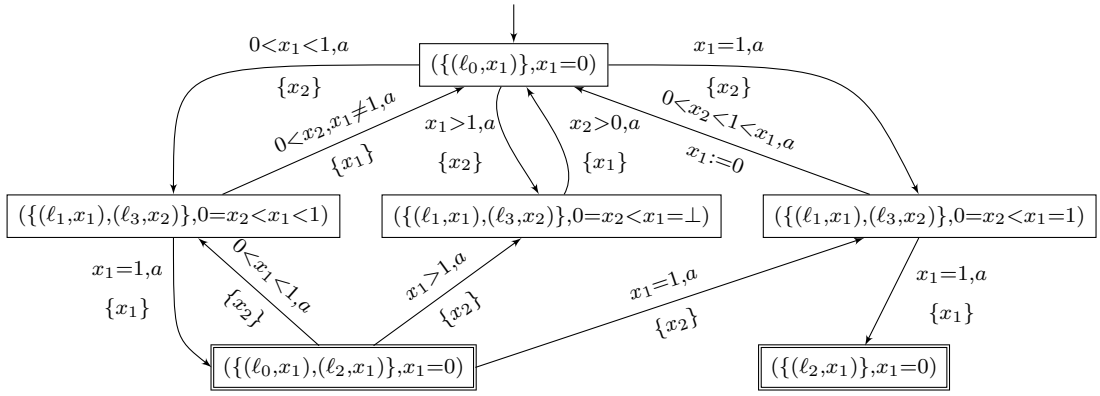

Figure 2.4: The deterministic version of the timed automaton from Figure 2.1.

The procedure described above is not effective, since it goes through the construction of infinite objects, with *a priori* no finite representation. However, we can abstract away

the complete construction, and exploit the fact that we know precisely how locations and transitions of the target timed automaton are derived. Hence, given a bound γ on the number of clocks needed, the resulting deterministic timed automaton can be computed on-the-fly by guessing new transitions. This way, the abstract procedure can be turned into an algorithm.

As announced earlier, our procedure unifies all *ad hoc* algorithms proposed so far. In particular, it allows to automatically generate a deterministic timed automaton for strongly non-Zeno automata, event-recording automata, and timed automata with integer resets.

Definition 2.3 (Subclasses of timed automata) *A timed automaton $\mathcal{A} = (L, X, E$ is*

- strongly non-Zeno *if there exists a bound $K \in \mathbb{N}$ such that any run of at least K transitions has cumulated delay greater than 1;*
- event-recording *if there is a bijection between Σ and X, and for every action $a \in \Sigma$, there is a clock $x_a \in X$ which is reset on every edge labelled with a;*
- with integer resets *if for every edge (ℓ, g, a, Y, ℓ') with $Y \neq \emptyset$, g contains an atomic constraint of the form $x = c$ for $x \in X$ and $c \in \mathbb{N}$.*

These subclasses of timed automata are denoted SnZTA*,* ERTA *and* IRTA*, respectively.*

Theorem 2.1 *For every timed automaton that belongs to* SnZTA*,* ERTA *or* IRTA*, one can effectively construct an equivalent deterministic timed automaton, of size at most doubly-exponential (and only exponential for* ERTA*).*

Note that beyond these known subclasses of determinizable timed automata, our approach proved applicable to other determinizable timed automata, such as the one of Figure 2.1, that is not strongly non-Zeno, nor event-recording or integer reset. We formalize a sufficient condition for applicability of our procedure as a clock-boundedness assumption on the determinized DAG.

As a by product of our approach, we obtain an EXPSPACE algorithm to check for the inclusion of timed languages, when they are described by a timed automaton satisfying our clock-boundedness assumption. We could also prove a matching lower bound for strongly non-Zeno timed automata, integer-reset timed automata, and generally timed automata satisfying the clock-boundedness assumption.

Theorem 2.2 *The inclusion problem is* EXPSPACE*-complete for timed automata belonging to* SnZTA*,* IRTA*, or satisfying the clock-boundedness assumption.*

Later, Wang *et al.* built on our unfolding technique to design a semi-algorithm for checking inclusion of timed languages [WSL$^+$14]. Their approach also incorporates simulation reduction based on antichains and LU-simulation, and has been implemented in the PAT model checker [SLDP09].

2.3 Game-approach to determinization

Intuitively, the subset construction, which successfully determinizes finite automata, fails for timed automata because of non-uniform resets. When performing a subset construction, it could thus be necessary to use an unbounded number of clocks to store information from all possible paths so far. This phenomenon on a particular timed automaton indicates that it is not determinizable by the approach of [BBBB09]. Our objective is to design a finer approach, yet the main problem remains to find a sufficient number of clocks and suitable resets to preserve all the timing information needed for the subset construction to be sound. One key feature of our approach lies in the use of relations between the clocks (*i.e.* conjunctions of atomic diagonal constraints) to encode the important timing information.

Our approach is partly based on [BCD05] which proposes, given a plant —modeled by a timed automaton— and fixed resources, a game where one player has a winning strategy if and only if occurrences of faults in the plant can be diagnosed by a timed automaton using the given resources. Inspired by this construction, given a timed automaton $\mathcal{A}$, and given fixed resources (k, M'), we derive a safety game between two players Spoiler and Determinizator, such that if Determinizator has a winning strategy, then a deterministic timed automaton $\mathcal{B}$, over resources (k, M'), with $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{A})$ can be effectively generated. Moreover, any strategy for Determinizator (winning or not) yields a deterministic overapproximation for $\mathcal{A}$.

We give here an informal definition of the game $\mathbf{G}_{(k,M')}(\mathcal{A})$, assuming X is the set of clocks of $\mathcal{A}$, and Y is a set of k clocks. All details and proofs can be found in [BSJK11, BSJK15]. The idea is to perform a subtle subset construction using relations to try to determinize $\mathcal{A}$. Using unions of regions (with a fixed maximal constant) instead of conjunctions of diagonal constraints allows to deal with a finite number of relations, in the same way as for regions. In the sequel, we often abuse notations and write conjunctions of constraints rather than unions of regions, for readability. As an example, $\bigwedge_{z,z' \in Z} z - z' = 0$ represents the union of all regions in which all clocks are equal, together with the unbounded region $(M, \infty)^Z$. In our game construction, relation updates use basic operations over zones such as projection and inverse projection, intersection, and time closure (*i.a.* union of all time-successors and time-predecessors).

- States of the game (future locations of the resulting deterministic timed automaton) are state estimates, symbolically represented using locations of $\mathcal{A}$ and regions over clocks in Y together with relations for the clocks in $X \cup Y$. The risk of overapproximation is marked and propagated thanks to Booleans.

- The initial state of the game is a state of Spoiler consisting of a single configuration with location ℓ_0 (initial location of $\mathcal{A}$), the simplest relation over $X \cup Y$: $\forall z, z' \in X \cup Y, z - z' = 0$, and the marking $\top$ (no overapproximation was done so far), together with the null region over Y.

- In each of its states, Spoiler challenges Determinizator by proposing an M'-bounded region r over Y, and an action $a \in \Sigma$, representing the fact that Spoiler chooses to read an a in the region r. Determinizator answers by deciding the set of clocks $Y' \subseteq Y$ it wishes to reset. The next state of Spoiler contains a region over Y $(r' = r_{[Y' \leftarrow 0]})$, and a finite set of configurations: triples formed of a location of $\mathcal{A}$, a relation on clocks in $X \cup Y$, and a boolean marking ($\top$ or $\bot$). A state of Spoiler thus constitutes a state estimate of $\mathcal{A}$, and the role of the markings is to indicate whether overapproximations

possibly happened. A state of Determinizator is a copy of the preceding state estimate of Spoiler together with the move of Spoiler.

- Bad states player Determinizator wants to avoid are, on the one hand states of the game where all configurations are marked $\perp$ and, on the other hand, states where all final configurations (if any) are marked $\perp$.

As an illustrating example, we give in Figure 2.6 the resulting game for resources $(1, 1)$, that is, a single clock, and maximal constant 1, for the timed automaton from Figure 2.5.

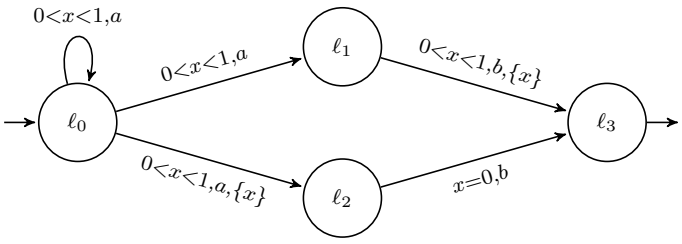

Figure 2.5: A second timed automaton example $\mathcal{A}'$.

It is a classical result that, for safety games, winning strategies can be chosen positional (the chosen move only depends on the current vertex) and they can be computed in linear time in the size of the arena [GTW02], so that we naturally restrict to positional strategies. A strategy for player Determinizator is described by a function assigning to each state of Determinizator a set $Y' \subseteq Y$ of clocks to reset. Symmetrically, a strategy for Spoiler is a mapping assigning to each state of Spoiler a region over Y and an action $a \in \Sigma$. A strategy profile (σ, σ') yields a path which is finite or has a lasso shape. A strategy σ for Determinizator is *winning* if whatever the strategy σ' for Spoiler, the path $\pi_{\sigma,\sigma'}$ does not visit any bad states.

With every strategy for Determinizator σ is associated the timed automaton $\mathsf{TA}(\sigma)$ obtained by merging a transition of Spoiler with the transition chosen by Determinizator just after, and setting as final locations those states of Spoiler containing at least one final location of $\mathcal{A}$.

Theorem 2.3 *Let $\mathcal{A}$ be a timed automaton, and (k, M') resources. For every strategy σ of Determinizator in $\mathbf{G}_{(k,M')}(\mathcal{A})$, $\mathsf{TA}(\sigma)$ is a deterministic timed automaton over resources (k, M') and satisfies $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathsf{TA}(\sigma))$. Moreover, if σ is a winning strategy, then $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathsf{TA}(\sigma))$.*

For the example from Figure 2.5, a winning strategy σ_{Win} for Determinizator is represented by the bold edges on Figure 2.6. This strategy yields the deterministic equivalent for $\mathsf{TA}(\sigma_{\mathsf{Win}})$ depicted in Figure 2.7.

On the one hand, compared to the overapproximation algorithm of [KT09], our approach performs better on several aspects. First, intuitively, the construction of a deterministic overapproximation in [KT09] is guided by a *skeleton*, a finite automaton which governs the clock resets in the deterministic timed automaton in construction. The resets are thus defined by a regular untimed language. In comparison, strategies in our game approach can be seen as timed skeletons since resets also depend on the regions the actions are taken in. Moreover, the game allows us to choose a good strategy, contrary to the skeletons that are fixed *a*

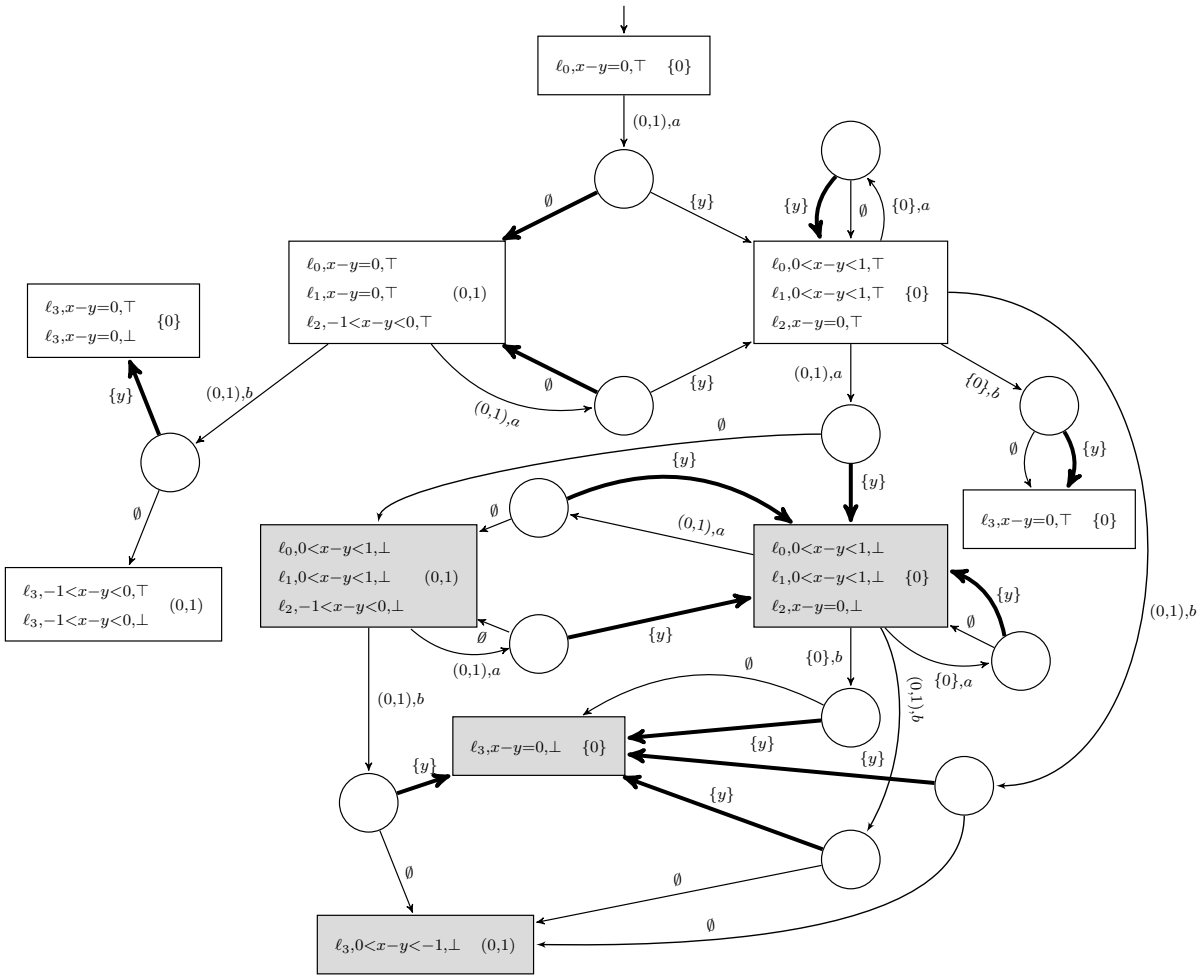

Figure 2.6: The game $\mathcal{G}_{\mathcal{A}',(1,1)}$ and an example of winning strategy σ for Determinizator.

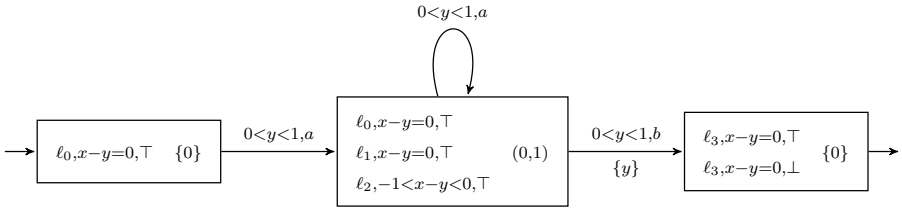

Figure 2.7: The deterministic TA $\mathsf{TA}(\sigma_{\mathsf{Win}})$ obtained by our construction.

priori. This is in particular helpful for example TA of Figure 2.6, for which no fixed skeleton would yield an exact determinization. Second, our approach improves the precision of the relations between clocks by taking the original guard into account when computing the updated relation. However, these two improvements do not guarantee that our methodology subsumes the existing overapproximation algorithm. Unfolding the game, to somehow add finite memory to the strategy, may lead to a tighter overapproximation. Untimed skeletons precisely allow one for such an unfolding, whereas our game construction doesn't. An easy way to stricly subsume the overapproximation approach from Krichen an Tripakis would thus be to incorporate a finite state skeleton in our game approach.

On the other hand, the game-based approach generalizes the determinization procedure from [BBBB09] we described ealier: for every timed automaton $\mathcal{A}$ such that the latter procedure yields an equivalent deterministic timed automaton with k clocks and maximal constant M', there is a winning strategy for Determinizator in $\mathbf{G}_{(k,M')}(\mathcal{A})$. Intuitively, this is a consequence of the fact that relations between clocks of $\mathcal{A}$ and clocks in the game generalize the mappings, since a mapping can be seen as a restricted relation, namely a conjunction of constraints of the form $x - y = 0$. Moreover, the game-approach strictly broadens the class of automata determinized by the previous procedure: It allows one to cope with some language inclusions because states are composed of several configurations. It should also be noted that the game-based approach applies to timed automata with invariants and ε-transitions, which was not the case of the determinization procedure. Beyond that, the game-approach performs better on some timed automata by providing a deterministic timed automaton with fewer resources. As an example, any timed automaton with integer resets can be determinized (thanks to the game approach) into a doubly exponential single-clock timed automaton with the same maximal constant, improving the doubly exponential deterministic timed automaton over $M + 1$ clocks obtained with the previous determinization procedure (where M is the maximal constant of the original timed automaton).

During a short stay at Aalborg University, Amélie Stainer implemented part of our game-based approach using Uppaal libraries for timed automata. So far, the prototype searches for a winning strategy in the game, and builds a deterministic equivalent when such a strategy exists. It relies on zones rather than regions for efficiency reasons in order to limit the size of the game in practice. This implementation was for instance used in the transformation of a logical specification in Metric Temporal Logic into a deterministic timed automaton [BDL+12].

2.4 Testing from timed automata specifications

The game approach we presented in the last section always produces an exact or overapproximated language. Overapproximations might not always be appropriate, and, depending on the context, underapproximations or different approximations might be more suitable. We therefore also proposed adaptations of the game-based determinization in order to generate deterministic underapproximations, and also combine over- and underapproximations. Such combinations of over and underapproximations are motivated by conformance testing for timed automata specifications.

Conformance testing aims at checking whether an implementation behaves correctly with respect to a specification [Tre96]. Implementations are considered as black boxes and only their interface with the environment is known and can be used to interact with the tester. The principles of offline model-based test generation is to design, from a specification, test-cases that are executed on the black box implementation in order to determine whether it conforms to the specification. The test generation is guided by test purposes, that express which specific behaviour one wants to test. The possible verdicts are then the following: Fail if a conformance error was detected, Pass if no conformance error was detected and the test purpose was satisfied, and Inconclusive when no conformance error was detected but the test prupose was not satisfied. The general framework is represented on Figure 2.8.

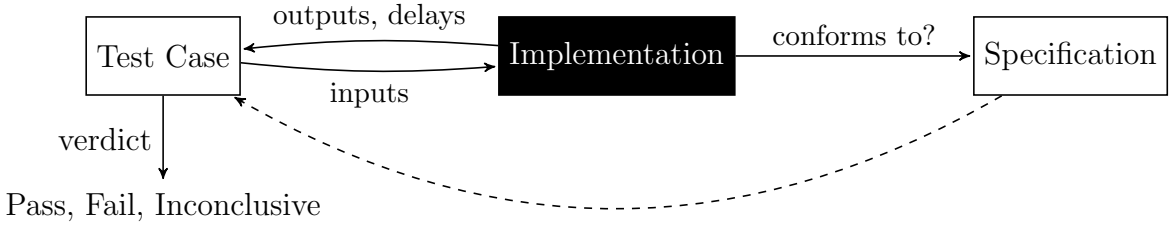

Figure 2.8: Principles of offline test generation.

For timed models, model-based conformance testing has been explored in the last decade, with different models and conformance relations (see *e.g.* [ST08] for a survey), and test generation algorithms (*e.g.* [BB05, KT09, NS03]). In this context, a very popular model is *timed automata with inputs and outputs* (TAIO), a variant of timed automata, in which the alphabet of actions is partitioned into input and output events.

We proposed a complete framework of offline test generation from TAIO [BJSK11, BJSK12]. Models are nondeterministic and incorporate invariants to represent urgency. Test cases are generated as deterministic timed automata and then executed on the implementation. One advantage is that test cases can be stored and further used *e.g.* for regression testing and serve for documentation. However, due to the undeterminizability of TAIO, the approach has often been limited in the litterature to deterministic or determinizable TAIO. We adapted our game-based approach to deal with inputs and outputs (rather than actions of unspecified type) and apply it to the offline test generation for timed systems. Intuitively, inputs should be underapproximated whereas outputs should be overapproximated. The resulting approximate determinization guarantees soundness of generated test cases by producing a deterministic io-abstraction of the input TAIO for a particular io-refinement relation, generalizing the existing io-refinement for deterministic TAIO of [DLL+10].

Chapter 3

Stochastic timed automata

Given the success of timed automata for the verification of real-time systems, several extensions have been proposed, with the objective to represent even more faithfully real-time systems. They include timed games [AMPS98] for modelling control problems, priced timed automata [ALP01, BFH+01, BFLM11] for modelling various quantities in timed systems, like energy consumption, and probabilistic extensions.

Probabilistic extensions of timed automata. Many applications like communication protocols require models which integrate both real-time constraints and randomized aspects (see *e.g.* [Sto03]). The development of such models and adequate verification algorithms is a challenging task, since it requires combining techniques from both real-time verification and probabilistic verification. In the literature two main modelling families can be distinguished.

A first approach consists in modelling the system as a purely stochastic process, and to express real-time constraints in the property that is checked. A model of choice for the system is that of continuous-time Markov chains (CTMC for short), while a rather wide spectrum of property formalisms has been considered, going from the logic CSL and extensions thereof [ASSB00, BHHK03, DHS09, ZJNH11] to (deterministic) timed automata [CHKM11]. In this context several exact and approximate model checking algorithms have been developed.

Another approach consists in integrating both features into a complex model (*e.g.* an extension of timed automata or Petri nets with stochastic evolution rules), and to analyse this model. In our work, we focus on automata-based models, and therefore only review related work on models based on timed automata. Such models include probabilistic timed automata [KNSS02] where discrete distributions are assigned to actions and for which the tool Prism [KNP11] has been developed. Delays or durations of events can also be made randomised. This is done for instance in [ACD91, ACD92] and later in [KNSS00], yielding either independent events and exact model checking algorithms (for a probabilistic and timed extension of CTL), or approximate model checking algorithms.

Stochastic timed automata fall in this second category. The semantics of a stochastic timed automaton is a stochastic process based on a timed automaton, in which both delays and discrete choices are chosen at random. We coined the term stochastic timed automata to emphasize that the underlying object we start with is a timed automaton while making at the same time explicit that the model yields a stochastic process.

A model which relaxes timed automata assumptions. One of the initial motivations for defining stochastic timed automata was linked to robustness of real-time systems. Indeed, timed automata form an idealized mathematical model, implying strong assumptions on the system, as for instance infinite precision of the clocks and the possibility of instantaneous events and communications. However, a real system has slightly different behaviours, *e.g.* time is measured with digital clocks. For a survey on robustness for timed automata, we refer to [BMS13].

Stochastic timed automata discard some disadvantages of timed automata implied by their mathematical definition. First, randomizing both delays and the choice of transitions rules out unlikely behaviours (like those requiring satisfaction of very precise clock constraints), and only important and meaningful sets of behaviours are then taken into account in the verification process. Then, the assumptions made in timed automata mentioned above lead to the existence of unreal(istic) behaviours of the model, such as Zeno behaviours that one would like to ignore. Unless the underlying timed automaton is inherently Zeno, the probability of Zeno behaviours will be 0 in the classes of models we have identified. We therefore convincingly claim that stochastic timed automata can be used as a possible solution for relaxing side-effects of mathematical assumptions made in timed automata.

Beyond the interesting purely Boolean property that stochastic timed automata permit to alleviate artefacts of the mathematical model, it goes without saying that equipping a timed automaton with random delays allows one to quantify over behaviors, *e.g.* by expressing that half of the runs will satisfy some specification. Alike many contributions to probabilistic model checking, in our work on stochastic timed automata however, we first concentrated on qualitative properties only (*i.e.* comparing probabilities with 0 and 1), and later considered quantitative properties.

Apart from purely probabilistic systems, we also considered variants in which decisions are made by agents. More precisely, we introduced extensions of stochastic timed automata with one or two players, yielding Markov decision processes and 2-player stochastic games extensions, respectively. These models generalize continuous-time Markov decision processes (CTMDP) and continuous-time Markov games (CTMG).

A potential application of the STA model As identified by Nicolas Basset, stochastic timed automata can serve in the random generation of runs for timed automata, paving the way to their statistical model checking. More precisely, the objective in [Bas13], is to find the "best" way to put probability distributions on delays and weights on edges of a timed automaton. Under some restrictions on the timed automaton, Basset solves this question and details how to define a stochastic process with maximal entropy, thus extending results on discrete time Markov chains by Shannon and Parry [Sha48, Par64]. Equipping a timed automaton with a maximal entropy stochastic process then allows to perform a uniform sampling over runs of the timed automaton.

"Other" stochastic timed automata The same terminology of stochastic timed automata is used by Pedro d'Argenio *et al.* to denote a more general model that extends probabilistic timed automata *à la* Prism with continuous distributions. The MoDeST toolset [DHKK01] supports the specification of rich systems with probabilities, time and nondeterminism, falling in the class of stochastic hybrid automata, via several possible input languages: probabilistic guarded commands, Uppaal models and a specific MoDeST modelling language. It pro-

vides several engines (roughly one per subclass of hybrid stochastic automata) to perform the analysis of such rich models. For what concerns stochastic timed automata, as far as we are aware, MoDeST performs simulation and statistical model checking. In comparison, our objective was to come up with exact model checking algorithms and decision procedures. Another difference is that we did not provide any tool interface, to the notable exception of networks of stochastic timed automata for which Marco Paolieri implemented an algorithm for the verification of timed-bounded until properties [BBH+13].

Outline of the contributions The rest of this chapter is organized as follows. We first define the model of stochastic timed automata and the corresponding model checking problems of interest. Then, we detail our contributions on their qualitative model checking. In particular, we show how a finite abstraction allows one to decide the almost sure model checking of stochastic timed automata for several subclasses. By refining this abstraction, one can even decide the quantitative model checking, here again for some subclasses of stochastic timed automata. Finally, we define games on stochastic timed automata and expose our results on optimization and qualitative model checking. This chapter is based on a series of papers, initiated in collaboration with Christel Baier, Patricia Bouyer, Thomas Brihaye and Marcus Grösser [BBB+07, BBB+08] and later continued mainly with Patricia Bouyer and Thomas Brihaye but also with various co-authors: Nicolas Markey [BBBM08], Sven Schewe [BS12], Paolo Ballarini, András Horváth, Marco Paolieri and Enrico Vicario [BBH+13], and Blaise Genest [BBG14]. Recently, we published an article to survey our results for stochastic timed automata, in their purely probabilistic interpretation [BBB+14].

3.1 Introducing the STA model

Rather than defining a probability measure over runs of a timed automaton, we give here an abstract definition, that can be instanciated by setting several parameters. This tuning should happen during the modelling process, when the designer wants to represent a given system by a stochastic timed automaton. Discussing how the parameters must be chosen is out of the scope of our work, yet we mention that Basset has come up with an elegant way to choose adequate probability distributions [Bas13].

A stochastic timed automaton consists in a timed automaton equipped with, for each state, a distribution over delays, and a discrete distribution over enabled edges. As a simplifying assumption, we assume that the latter distribution over edges is uniform by regions, and is given by weights on transitions, as it is classically done to turn nondeterminism into probabilities.

Definition 3.1 (Stochastic timed automaton) *A* stochastic timed automaton *(*STA*) is a tuple* $\langle \mathcal{A}, \mu, w \rangle$ *consisting of a timed automaton* $\mathcal{A} = (L, X, E$ *equipped with probability measures* $\mu = (\mu_s)_{s \in L \times \mathbb{R}_+^X}$*, and positive weights* $w = (w_e)_{e \in E}$*.*

Given s a state of $\mathcal{A}$, we let $I(s, e)$ for the set of delays from s after which edge e is enabled, and $I(s)$ for the set of delays from s that enable at least one edge. Formally $I(s, e) = \{\tau \in \mathbb{R} \mid s + \tau \xrightarrow{e}\}$ and $I(s) = \cup_{e \in E} I(s, e)$. For simplicity, we assume that timed automata have no deadlock; using the sets $I(\cdot)$ this can be rephrased as, for every state s of $\mathcal{A}$, $I(s) \neq \emptyset$.

In the sequel, we make the following assumptions about the probability distribution μ_s over delays from state s:

(H1) $\mu_s(I(s)) = \mu_s(\mathbb{R}_+) = 1$,

(H2) Writing λ^* for the standard Lebesgue measure, if $\lambda^*(I(s)) > 0$, then μ_s is equivalent to λ^* on $I(s)$; Otherwise, μ_s is equivalent to the uniform distribution over points of $I(s)$.

This last condition denotes some kind of fairness with respect to enabled transitions when only punctual delays are possible, in that we cannot disallow one transition by putting a probability 0 to delays enabling that transition.

In order to define a probability measure over the set of runs, we introduce constrained symbolic paths, to which we will assign a natural probability. If s is a state of $\mathcal{A}$, $(e_i)_{1\leq i\leq n}$ is a finite sequence of edges, and $\mathcal{C}$ is a constraint over n variables $(t_i)_{1\leq i\leq n}$, the *constrained symbolic path* $\pi_{\mathcal{C}}(s, e_1 \dots e_n)$ represents the following set of runs:

$$\pi_{\mathcal{C}}(s, e_1 \dots e_n) \;=\; \{\rho = s \xrightarrow{\tau_1, e_1} s_1 \dots \xrightarrow{\tau_n, e_n} s_n \in \mathsf{Runs}(\mathcal{A}, s) \mid (\tau_i)_{1\leq i\leq n} \models \mathcal{C}\} \;.$$

In order to equip STA with a probability measure over infinite runs, we first inductively associate with each (unconstrained) symbolic path a value as follows:

$$\mathbb{P}_{\mathcal{A}}(\pi(s, e_1 \dots e_n)) \;=\; \int_{t\in I(s,e_1)} p_{s+t}(e_1)\, \mathbb{P}_{\mathcal{A}}(\pi(s_t, e_2 \dots e_n))\, \mathrm{d}\mu_s(t)$$

where $s \xrightarrow{t} (s{+}t) \xrightarrow{e_1} s_t$, and we initialize with $\mathbb{P}_{\mathcal{A}}(\pi(s)) = 1$. The definition of $\mathbb{P}_{\mathcal{A}}$ over symbolic paths relies on the fact that the probability of taking transition e_1 at time t coincides with the probability of waiting t time units and then choosing e_1 among the enabled transitions, *i.e.*, $p_{s+t}(e_1)\mathrm{d}\mu_s(t)$. The value $\mathbb{P}_{\mathcal{A}}(\pi(s, e_1 \dots e_n))$ is the result of n successive one-dimensional integrals, but it can also be viewed as the result of an n-dimensional integral. Hence, we can easily extend the above definition to constrained symbolic paths $\pi_{\mathcal{C}}(s, e_1 \dots e_n)$ assuming $\mathcal{C}$ is Borel-measurable. So defined, $\mathbb{P}_{\mathcal{A}}$ is a probability measure over runs starting from a fixed initial state:

Proposition 3.1 *For every state s of $\mathcal{A}$, $\mathbb{P}_{\mathcal{A}}$ is a probability measure over $\mathsf{Runs}(\mathcal{A}, s)$.*

The complete proof (see [BBB+14]) is rather technical but not difficult: we first prove that $\mathbb{P}_{\mathcal{A}}$ is a probability measure on the set of constrained symbolic paths of length n (for all n), then extend this result to the ring generated by all constrained symbolic paths and finally use Caratheodory's extension theorem (a classical result from measure theory) to establish that $\mathbb{P}_{\mathcal{A}}$ is a probability measure on the set of all runs.

Now that $\mathbb{P}_{\mathcal{A}}$ is established to be a probability measure over runs, we observe that sets of runs defined by natural properties are measurable w.r.t to $\mathbb{P}_{\mathcal{A}}$.

Lemma 3.1 *ω-regular properties and properties given by* LTL *formulae are measurable. Timed properties given by specification Büchi or Muller timed automata are measurable.*

Beyond these property-defined sets of runs, the set of Zeno runs also is measurable since:

$$\mathsf{Zeno}(\mathcal{A}) = \bigcup_{M\in\mathbb{N}} \bigcap_{n\in\mathbb{N}} \bigcup_{(e_1,\dots,e_n)\in E^n} \pi_{\tau_1+\dots+\tau_n\leq M}(s, e_1 \dots e_n)$$

and each symbolic constrained path $\pi_{\tau_1+\dots+\tau_n\leq M}(s, e_1 \dots e_n)$ forms a cylinder.

We are finally in a position to define the model checking problems of interest for stochastic timed automata. If P is a measurable property over AP, we write $\mathbb{P}_{\mathcal{A}}(s \models P)$ for its measure, that is: $\mathbb{P}_{\mathcal{A}}(s \models P) = \mathbb{P}_{\mathcal{A}}\{\varrho \in \mathsf{Runs}(\mathcal{A}, s) \mid \varrho \models P\}$.

Definition 3.2 *Let s be a state of $\mathcal{A}$. Assume P is a measurable property over AP. Let $\lambda \in (0,1)$ be a threshold and $\varepsilon > 0$ an error.*

- **Almost sure model checking problem***: does $\mathbb{P}_{\mathcal{A}}(s \models P) = 1$?*
- **Quantitative model checking problem***: does $\mathbb{P}_{\mathcal{A}}(s \models P) > \lambda$?*
- **Approximate model checking problem***: compute p such that $|p - \mathbb{P}_{\mathcal{A}}(s \models P)| < \varepsilon$.*

Intuition on an example Consider $\mathcal{A}_{\mathsf{running}}$ from Figure 3.1, with initial state $s_0 = (\ell_0, 0)$. Assume for all states $s_t = (\ell_0, t)$ both uniform distributions over delays and discrete moves: $\mu_{s_0} = \lambda^*$ is the uniform distribution over $[0,1]$ and $\mu_{s_t} = \frac{\lambda^*}{1-t}$ is the uniform distribution over $[t,1]$; the weight of each edge is 1. Then, a simple calculation shows that $\mathbb{P}_{\mathcal{A}_{\mathsf{running}}}(\pi(s_0, {e_1}^n)) = \frac{1}{2^n}$, for $n \in \mathbb{N}$, and thus $\mathbb{P}_{\mathcal{A}_{\mathsf{running}}}(\pi(s_0, {e_1}^\omega)) = 0$.

Then, $\mathbb{P}_{\mathcal{A}_{\mathsf{running}}}(s_0 \models \Diamond(p_1 \wedge \Box(p_1 \Rightarrow \Diamond p_2)) = 1$. Indeed, in state (ℓ_0, ν) with $0 \leq \nu \leq 1$, the probability of firing e_2 (after some delay) is always $1/2$ (guards of e_1 and e_2 are the same, there is thus a uniform distribution over the two edges), the location ℓ_1 is eventually reached with probability 1. In ℓ_1, the transition e_3 will unlikely happen, because its guard $x = 1$ is much too "small" compared to the guard $x \geq 3$ of the transition e_4. The same phenomenon arises in location ℓ_2 between the transitions e_5 and e_6. In conclusion, the runs of the timed automaton $\mathcal{A}_{\mathsf{running}}$ (from s_0) almost surely follow sequences of transitions of the form ${e_1}^* e_2 (e_4 e_5)^\omega$. Hence, with probability 1, the formula $\Diamond(p_1 \wedge \Box(p_1 \Rightarrow \Diamond p_2))$ is satisfied.

Note that the latter formula is not satisfied in $\mathcal{A}_{\mathsf{running}}$ from s_0 (under the classical LTL semantics), since some runs violate it: '*staying in ℓ_0 forever*', '*reaching ℓ_3*', etc... All these counter-examples are somehow unlikely and vanish thanks to the probabilities that equip stochastic timed automata.

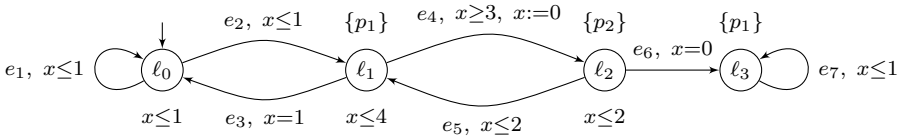

Figure 3.1: $\mathcal{A}_{\mathsf{running}}$, a single-clock running example timed automaton.

3.2 Almost sure model checking

We start with the simplest decision problem, namely the almost sure model checking problem. Early papers on timed automata and extensions thereof rely on the region automaton abstraction, briefly recalled in Chapter 2. A natural attempt in the context of stochastic timed automata, is thus to consider its region abstraction, here a finite Markov chain. This is promising since, for qualitative properties –such as, with probability one– the precise value of probabilities seems irrelevant. Moreover, in the region abstraction, negligible edges are exactly those with a guard of lower dimension (than the neighbouring edges). However, this simple finite abstraction fails in general. Yet, we can define two incomparable subclasses of stochastic timed automata for which this abstraction is correct with respect to the almost sure model checking problem: single-clock timed automata, and reactive timed automata.

3.2.1 The region Markov chain

Given an STA $\langle \mathcal{A}, \mu, w \rangle$ with $\mathcal{A}$ a TA with set of clocks X and maximal constant M, a natural finite abstraction is the following region Markov chain. The region Markov chain builds upon the standard region automaton, and uses a partition of its edges into thin and thick edges, reflecting whether the probability to fire them is negligible or not. Given e an edge of $\mathcal{A}$, we denote by $e_{R,R'}$ its copy in the region automaton when the initial region is R and the target region is R'.

Definition 3.3 (Region Markov chain) *Given $\langle \mathcal{A}, \mu_s, w_e \rangle$ a stochastic timed automaton, the* region Markov chain $\mathsf{MC}(\mathcal{A}) = (S_{\mathsf{MC}(\mathcal{A})}, \Delta_{\mathsf{MC}(\mathcal{A})})$ *has state space $S_{\mathsf{MC}(\mathcal{A})} = L \times \mathsf{Reg}_M^X$, and outgoing probabilities are uniform over* thick *edges of the region automaton,* i.e. $\Delta_{\mathsf{MC}(\mathcal{A})}((\ell, R) \xrightarrow{e} (\ell', R')) > 0$ *if* $\dim(I(s, e_{R,R'})) = \dim(I(s))$ *for every (or equivalently, for some) $s \in (\ell, R)$.*

It has to be stressed that the region Markov chain only depends on the structure of the timed automaton, not on the probability measures μ_s and weights w_e assigned to it, as long as the μ_s's satisfy the hypotheses (H_1) and (H_2). Therefore, the notation $\mathsf{MC}(\mathcal{A})$ does not need to incorporate μ_s and w_e. In our idea, the simpler, the better, and we shall examine whether this coarse abstraction is already helpful.

We first illustrate the construction on a simple 1-clock stochastic timed automaton. Figure 3.2 represents the classical region graph where thick (resp. thin) edges are depicted bold (resp. dashed). For example, starting from $(\ell_0, \{0\})$ the probability is null to sample delay exactly 0, or exactly 1, compared to picking a delay in $(0, 1)$, so that the edges to $(\ell_1, 0)$, $(\ell_0, 0)$ and $(\ell_0, 1)$ are thin, and the edge to $(\ell_0, (0, 1))$ is thick. From that region graph, by removing negligible edges, keeping only states reachable from the initial one, and assigning uniform distributions on outgoing edges, we obtain the region Markov chain, represented on Figure 3.3.

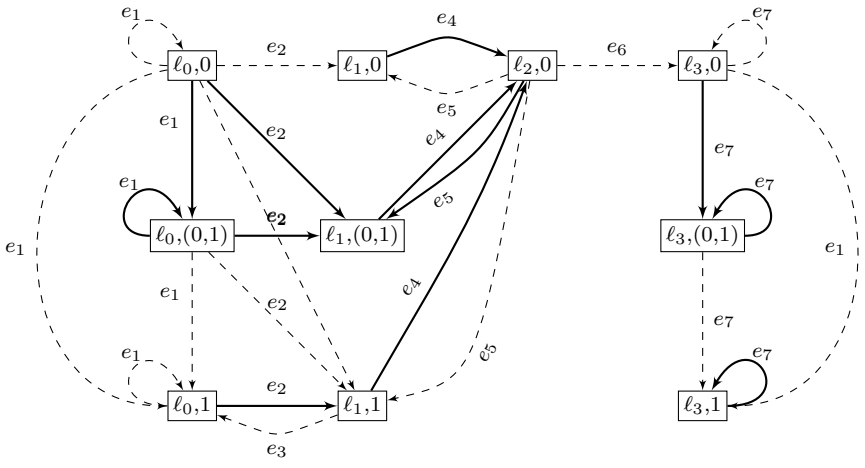

Figure 3.2: Constructing the region Markov chain on the running example.

Consider now the 2-clock TA $\mathcal{A}_{\mathsf{pacman}}$ of Figure 3.4. When equipped with uniform distributions, and starting at $s_t = (\ell_0, (0, t))$ for any value t, one can show that the probability of the symbolic path $\pi(s_t, (e_3 e_4 e_5)^\omega)$ is positive. Therefore, in this pacman-shaped stochastic timed

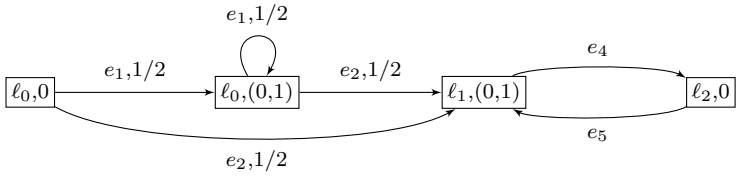

Figure 3.3: $\mathsf{MC}(\mathcal{A}_{\mathsf{running}})$, the region Markov chain for the running example.

automaton $\mathbb{P}_{\mathcal{A}_{\mathsf{pacman}}}(s_t \models \Diamond p_2) < 1$. Yet, in its region Markov chain, depicted on Figure 3.5, the probability to eventually reach the state fulfilling p_2 is one. Maybe even more surprisingly, $\mathbb{P}_{\mathcal{A}_{\mathsf{pacman}}}(s_t \models \Diamond\Box\neg p_2) = 1$, whereas $\mathbb{P}_{\mathsf{MC}(\mathcal{A}_{\mathsf{pacman}})}([s_t] \models \Box\Diamond p_2) = 1$. This example shows that STA may be less fair than their abstraction: in $\mathsf{MC}(\mathcal{A}_{\mathsf{pacman}})$, the state tagged with proposition p_2 is visited almost surely infinitely often, whereas this is not true in $\mathcal{A}_{\mathsf{pacman}}$.

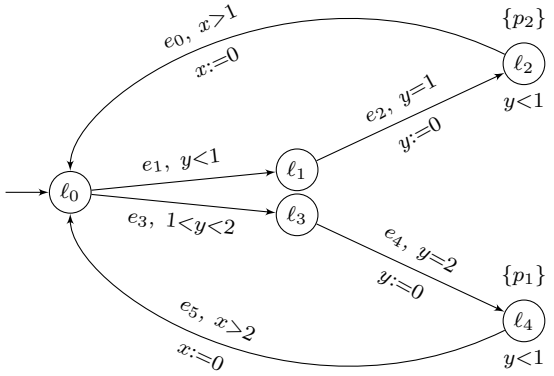

Figure 3.4: $\mathcal{A}_{\mathsf{pacman}}$, an intriguing two-clock automaton.

Since finite probabilistic models have some inherent fairness, it is natural to impose a fairness condition to stochastic timed automata: almost surely whenever an edge can be fired infinitely often, it is fired infinitely often. This rules out the above pacman example which is not almost surely fair. A general result is that under this strong fairness assumption, the region Markov chain is a correct abstraction for the almost sure model checking problem.

Theorem 3.1 *Let s be a state of $\mathcal{A}$, and P be a prefix-independent (untimed) property over AP. Assuming $\mathbb{P}_{\mathcal{A}}(s \models \mathsf{fair}) = 1$, the following holds:*

$$\mathbb{P}_{\mathcal{A}}(s \models P) = 1 \iff \mathbb{P}_{\mathsf{MC}(\mathcal{A})}([s] \models P) = 1 \ .$$

The pacman example does not leave much room for synctactic subclasses of stochastic timed automata for which Theorem 3.1 can hold. The two fundamental features in $\mathcal{A}_{\mathsf{pacman}}$ that create the convergence phenomenon are the 2 clocks and the fact that their values are always bounded. Let us thus examine stochastic timed automata in which either there is a single clock, or from any state all delays are possible.

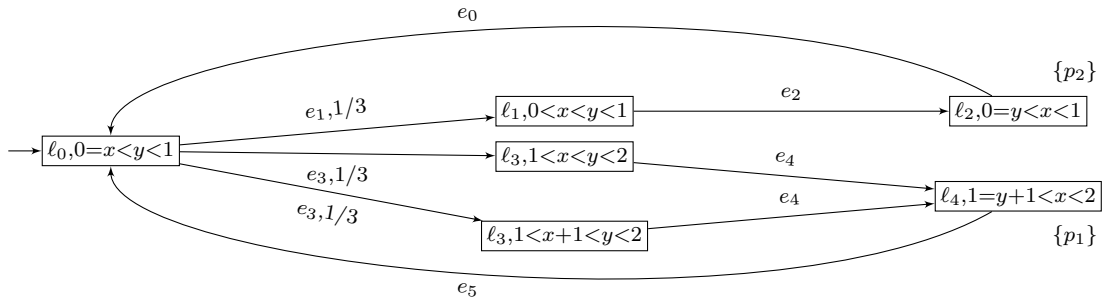

Figure 3.5: $\mathsf{MC}(\mathcal{A}_{\mathsf{pacman}})$ the region Markov chain of $\mathcal{A}_{\mathsf{pacman}}$.

3.2.2 Single-clock stochastic timed automata

We first focus on stochastic timed automata with a single clock, restricting this way the timing behaviour to simpler constraints.

Theorem 3.2 *Let $\mathcal{A}$ be a single-clock timed automaton. Then $\mathbb{P}_{\mathcal{A}}(s \models \mathsf{fair}) = 1$.*

In order to prove Theorem 3.2 the idea is to decompose the set of runs of $\mathcal{A}$ into three categories: 1) the ones with infinitely many resets, 2) the ones ultimately staying in the unbounded region $(M_{\mathcal{A}}, \infty)$ and 3) the ones ultimately staying in a bounded region of the form $(c, c+1)$ or $\{c\}$. It is rather easy to establish that, in all cases, the runs are almost surely fair. Indeed, assuming there are infinitely many resets, after each reset, the probability to take a non-negligible edge is positive and lower bounded so that almost surely it will be taken infinitely often. Also, once the unbounded region is reached, the precise clock value is irrelevant, and the STA behaves like a finite Markov chain, therefore in a fair way. In the last case, the runs end in a bottom strongly connected component (BSCC) where all guards are equal to the same bounded region; therefore here again the STA behaves like a standard Markov chain, and is almost surely fair.

Zeno runs form a measurable set, but can obviously not be expressed by an untimed property, and therefore Theorem 3.1 won't apply for checking almost sure non-Zenoness. Yet, we can prove that the region abstraction is correct for checking non-Zenoness on single-clock stochastic timed automata. More precisely, almost all runs that end in the unbounded region $x > M$ or for which the clock is reset infinitely often are non-Zeno. As a consequence, Zeno runs necessarily end in a bounded region with no reset. A BSCC of $\mathsf{MC}(\mathcal{A})$ is said Zeno if it is bounded and does not contain a reset. In the example from Figure 3.3, the unique BSCC is bounded, yet it contains a reset edge, namely e_4. With this notion, we formulate a characterization of almost-sure non-Zenoness:

Proposition 3.2 $\mathbb{P}_{\mathcal{A}}(s \models \mathsf{nonZeno}) = 1 \iff \sum_{B \text{ Zeno BSCC of } \mathsf{MC}(\mathcal{A})} \mathbb{P}([s] \models \Diamond B) = 0.$

Checking that no Zeno BSCC can be reached from the region $[s]$ is a simple reachability question in $\mathsf{MC}(\mathcal{A})$. Since $\mathsf{MC}(\mathcal{A})$ is polynomial in the size of $\mathcal{A}$ (recall that $\mathcal{A}$ has a single clock), and reachability can be checked in $\mathsf{NLOGSPACE}$, we obtain the following coplexity result:

Theorem 3.3 *Almost sure non-Zenoness is decidable in* NLOGSPACE *for single-clock stochastic timed automata.*

3.2.3 Reactive stochastic timed automata

Another class of stochastic timed automata for which the region Markov chain abstraction is correct is the one of reactive stochastic timed automata, introduced in [BBJM12]. Stochastic timed automata are called reactive whenever for every state, after any delay, some transition is immediately enabled. This assumption is rather natural from a modeling perspective, and the name suggests that the system can react at any time.

Notice that reactive stochastic timed automata generalize continuous-time Markov chains (CTMC). A CTMC is nothing else than a single-clock reactive stochastic timed automaton in which (i) on all transitions, the guard is trivial, and the clock is reset, and (ii) each location is assigned an exponential distribution over delays.

Theorem 3.4 *Let $\mathcal{A}$ be a reactive timed automaton. Then $\mathbb{P}_{\mathcal{A}}(s \models \mathsf{fair}) = 1$.*

The key point in the proof of Theorem 3.4 is that a subset of regions, called *memoryless* are visited infinitely often with probability 1. Moreover, memoryless regions are particularly simple: for every clock either its value is 0 or it is above the maximal constant. When entering a memoryless region, future behaviours and their probabilities are independent of both the past and the precise clock valuations, hence their name. Memoryless regions thus somehow form a finite attractor (see *e.g.* [BBS06, ABM07]) in the infinite Markov chain defined by STA. Visiting infinitely often memoryless regions prevents converging phenomena such as the one observed in the example of Figure 3.4.

A side-result of the proof of Theorem 3.4 is that Zeno runs have negligible measure in reactive stochastic timed automata.

Theorem 3.5 *Let $\mathcal{A}$ be a reactive timed automaton. Then $\mathbb{P}_{\mathcal{A}}(s \models \mathsf{nonZeno}) = 1$.*

3.3 Quantitative model checking

3.3.1 Quantitative LTL model checking for single-clock STA

Beyond the decidability results for the qualitative model checking of STA, we investigated the *quantitative* probabilistic model checking problem, which aims at approximating the probability of a given ω-regular property in a timed automaton. The region Markov chain abstraction is no longer correct when one is interested in quantitative properties. For single-clock timed automata, and under some further restrictions on their structure, we are able to *(i)* compute a closed-form expression for the probability that $\mathcal{A}$ satisfies φ, *(ii)* approximate this probability, and *(iii)* compare the probability to a given threshold, in order to answer the quantitative model checking problem.

The difficulty when trying to build a finite abstraction of the semantics of an STA is that the probability to fire a given edge depends on the precise clock value. Observe that this dependency no longer exist in memoryless regions, that is in the unbounded region or right after a reset. For single-clock STA, memoryless regions are all $(\ell, 0)$ where the clock has

just been reset, and all unbounded regions $(\ell, (M, \infty))$ in which the clock value is irrelevant. We propose to construct a finite Markov chain whose state-space consists in the memoryless regions, and in which the discrete probabilities are exactly the probabilities of all runs from one memoryless region to another. In order to have only finitely many paths to consider for each edge in the abstraction (called macro edge), we assume that the STA does not contain bounded cycles with no reset.

Definition 3.4 *Let $\mathcal{A}$ be a single-clock* STA*, in which all bounded cycles contain a reset. The* memoryless region Markov chain *associated with $\mathcal{A}$ is $\mathsf{MMC}(\mathcal{A}) = (S_{\mathsf{MMC}}, \Delta_{\mathsf{MMC}})$ with $S_{\mathsf{MMC}} = L \times (\{0\} \cup (M, \infty))$ is the set of memoryless regions, and*

$$\Delta_{\mathsf{MMC}}(s, s') = \sum_{\pi \in \mathsf{SPaths}(s,s')} \mathbb{P}_{\mathcal{A}}(\pi)$$

where $\mathsf{SPaths}(s, s')$ denotes the set of simple symbolic paths $\pi(s, e_1 \cdots e_n)$, that end in s' and do not visit another memoryless region in between.

We illustrate the construction of the memoryless region Markov chain on the example of Figure 3.6. For instance, from memoryless region $(\ell_0, 0)$, one can reach the memoryless regions

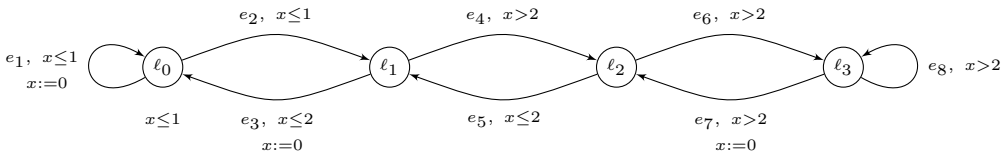

Figure 3.6: An example single-clock STA for the quantitative analysis.

$(\ell_2, > 2)$ via the sequence of edges $e_2 \cdot e_4$, and $(\ell_0, 0)$ itself via two possible paths: e_1 or $e_2 \cdot e_3$. Assuming in state $(\ell_0, 0)$ a uniform distribution over $[0, 1]$, and exponential distributions with rate 1 in all other locations, the probability of these symbolic paths starting (and ending) in memoryless regions can be computed. The resulting finite-state Markov chain is depicted in Figure 3.7, where we indicate on macro edges the sequence of edges it corresponds to and its probability.

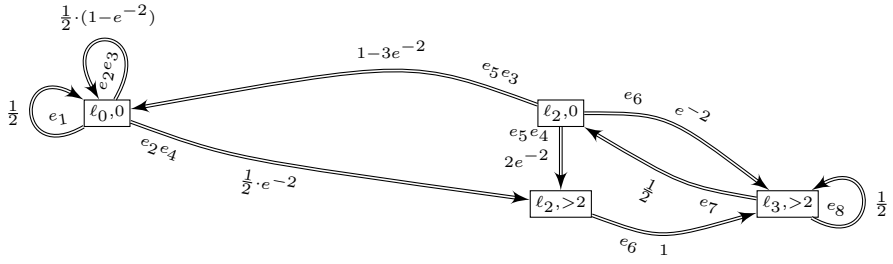

Figure 3.7: The memoryless region Markov chain for the STA from Figure 3.6.

Theorem 3.6 *Let $\mathcal{A}$ be a single-clock* STA *with no bounded cycles without reset, and P a prefix independent property. Then*

$$\mathbb{P}_{\mathcal{A}}(s \models P) = \mathbb{P}_{\mathsf{MMC}(\mathcal{A})}([s] \models P) \ .$$

Theorem 3.6 expresses that the memoryless region Markov chain is a correct abstraction for the quantitative model checking. Yet, it does not immediately provide a decision algorithm for it, since the coefficients in $\mathsf{MMC}(\mathcal{A})$ might not be computable. In particular, when one considers a sequence of edges through bounded locations, with successive guards $x < 1$, $x < 2$ $\cdots$ $x < n$, there does not seem to be a closed form for the probability of firing such a symbolic path. In order to obtain expressions for the probabilities in $\mathsf{MMC}(\mathcal{A})$, we therefore additionally require that $\mathcal{A}$ is reactive, and all locations are equipped with exponential distributions.

Proposition 3.3 *Let $\mathcal{A}$ be a reactive single-clock* STA *with no bounded cycle without reset, and P a prefix independent property. Then, there exist $q \in \mathbb{N}_{>0}$ a positive integer and $f \in \mathbb{Q}(X)$ a rational function such that $\mathbb{P}_{\mathcal{A}}(s \models P) = f(e^{-\frac{1}{q}})$.*

Building on Proposition 3.3, one can decide whether $\mathbb{P}_{\mathcal{A}}(s \models P)$ is rational, compute it if it is rational, and if not approximate it within the desired error bound. More precisely, one starts with lower- and upper-approximating sequences of rationals for the transcendental number $e^{-\frac{1}{q}}$: $l_0 \leq l_1 \cdots \leq l_n \cdots \leq e^{-\frac{1}{q}} \cdots \leq u_n \cdots \leq u_1 \leq u_0$, with $\lim l_n = \lim u_n = e^{-\frac{1}{q}}$. Now, we observe that any rationnal function $f \in \mathbb{Q}(X)$ (and thus the one from Proposition 3.3) is monotonic on a sufficiently small interval around $e^{-\frac{1}{q}}$. Moreover, one can determine elements of the above approximating sequences such that f is monotonic on (l_i, u_i), for some index i. Given an approximation factor ε, to obtain an ε-approximation of $f(e^{-\frac{1}{q}})$ it thus suffices to evaluate f at points l_n and u_n for values of n greater than i and large enough. This provides an algorithm to compute the probability of any prefix independent property up to any desired error.

Theorem 3.7 *The approximate model checking problem is feasible for reactive single-clock* STA *with no bounded cycle without reset.*

3.3.2 Approximate reachability probability for reactive STA

We now turn to another subclass of STA, this time with multiple clocks. Reactive STA are essentially Markov chains with a finite attractor. Indeed, we already mentionned that the so-called memoryless regions, *i.e.* regions in which every clock value is either 0 or above the maximal constant, are visited almost surely infinitely often. Note that, as such, the set of memoryless regions consists in infinitely many states, and, although it is an attractor, it is not stricly speaking a finite attractor. Yet, the interesting feature of memoryless regions is precisely that when entering such a region, the outgoing probabilities do not depend on the precise valuation, but only on the region. All states of a memoryless region can therefore be collapsed, yielding in a true finite attractor for reactive stochastic timed automata.

Theorem 3.8 *Let $\mathcal{A}$ be a reactive* STA*, and $F \subseteq L \times R$ a goal region. Given $\varepsilon > 0$, one can compute p^- and p^+ such that $p^- \leq \mathbb{P}_{\mathcal{A}}(s \models \Diamond F) \leq p^+$ and $p^+ - p^- < \varepsilon$.*

We provide in the sequel two ways to obtain Theorem 3.8. They both exploit the finite attractor property, but at different levels. As an illustrating example, we consider the reactive STA depicted in Figure 3.8, in which we aim at computing the probability to reach location ℓ_4 from the initial state $(\ell_0, 0, 0)$.

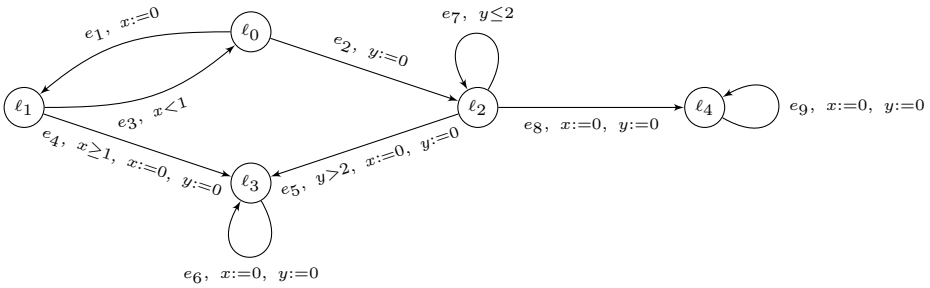

Figure 3.8: A reactive STA example for the quantitative analysis.

Applying the approximation scheme directly The approximation scheme described for probabilistic lossy channels in [PN97, Rab03], and generalized to decisive Markov chains in [ABM07] applies to approximate reachability probabilities in reactive STA, since they have a finite attractor.

The idea of the approximation scheme is to compute under- and over-approximations of the value $\mathbb{P}_{\mathcal{A}}(s \models \Diamond F)$. To do so, p_n^- is defined as the probability to reach F within n transitions. This probability is expressed as a sum over symbolic paths of length n. Similarly, p_n^+ is the probability to either reach F in no more than n steps, or reach at the n-th step, a state from which F is still reachable. This way, p_n^- and p_n^+ clearly are lower and upper bounds for $\mathbb{P}_{\mathcal{A}}(s \models \Diamond F)$. Now, the finite attractor property (or more generally decisiveness) ensures convergence of this procedure, since one can show that $\lim_{n\to\infty} p_+ - p_n^- = 0$. Taking as stopping criterion $p_n^+ - p_n^- < \varepsilon$ gives an ε-approximation of the desired probability.

Computing finer and finer abstractions Rather than applying the approximation at the STA level, another approach is possible. Pursuing our general methodology, the reactive STA can be abstracted into a family of finite Markov chains parameterised by a tolerance factor δ. To do so, one exploits again the memoryless regions, that serve as states of these Markov chains. The probability of a transition between memoryless regions in the memoryless Markov chain $\mathsf{MMC}(\mathcal{A})$ is defined as the infinite sum of individual probabilities of paths between the two regions that do not visit any memoryless region in between. This is illustrated on the reactive STA example of Figure 3.8: we represent its memoryless Markov chain on Figure 3.9. The transitions bear the label of the possible sequences of edges leading from one memoryless region to the next one. We use rational expressions to denote these infinite sets.

In this second approximation scheme, the idea is to approximate these transitions probabilities defined by (potentially) infinite sums, up to a desired error, still relying on the finite attractor property. This yields a family of finite sub-Markov chains (and finite sup-Markov chains), for which one can compute the probability to reach F, say p_δ^- and p_δ^+. By continuity of the probability measures when δ converges to 0, both p_δ^- and p_δ^+ converge towards

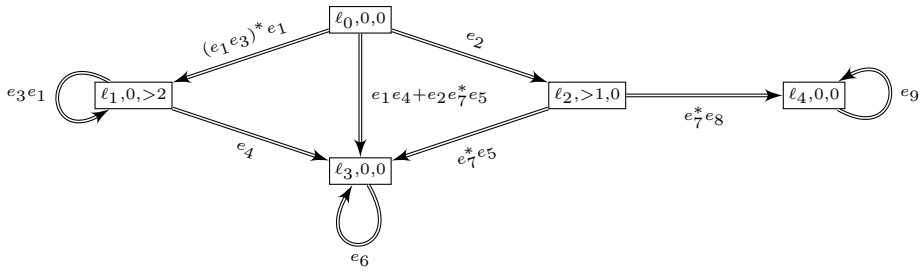

Figure 3.9: The abstract Markov chain on the example of Figure 3.8.

$\mathbb{P}_{\mathcal{A}}(s \models \Diamond F)$. An approximation of this probability up to any desired error can therefore be computed.

Comparison with Section 3.3.1 It is worth commenting on the results presented for reactive STA compared to the ones for 1-clock STA. One could think, at first, that the latter are subsumed by the former, although it is not the case, at least for three reasons. First of all, in the case of reactive single-clock STA, we showed that one can compute an expression of the probability, which we do not know how to do for general reactive STA. Second, in the context of single-clock STA with no bounded cycle without reset, we provided a finite abstraction that is correct for the quantitative model checking, which again, we do not have for multi-clock reactive STA. This brings us to our last, but not least, difference: while we can deal with arbitrary omega-regular formulae for single-clock STA, we are only able to treat reachability properties for reactive STA. In conclusion, the results we obtained so far for single-clock STA are much stronger than the ones for the reactive class.

3.3.3 Transient analysis of reactive STA

We now consider the transient analysis of reactive STA, that is the behaviour of the stochastic process underlying a reactive STA before steady state. This can, in particular, be applied to the approximate computation of the probability for bounded until formulae, for the same class of reactive STA. To do so, we exploit the method of stochastic state classes [BPSV05], first developed in the context of stochastic time Petri nets to perform their steady-state analysis, transient analysis, and probabilistic model checking. A similar approach was used for the bounded model checking of generalized semi-Markov processes models for stochastic real-time systems [AB06].

Stochastic state classes are particularly useful when so-called regeneration points, referred to as memoryless regions in what precedes, are not guaranteed to appear infinitely often. Note that even for reactive STA, the memoryless regions are bound to be visited infinitely often almost surely, not surely.

To explain how stochastic state classes will be used in the context of STA, we take an alternative view: states of STA are "observed" just before a discrete transition (rather than right after a discrete transition in what precedes). Therefore, from a given state, first a discrete transition is chosen, according to a discrete probability distribution, and then a delay (or sojourn time, in analogy to Petri nets) is randomly chosen. These two steps yield a probability distribution over clock valuations and sojourn times. The objective of stochastic

state classes is precisely to represent these probability distributions. More precisely, stochastic state classes characterize the stochastic process by representing explicitely, after each discrete transition and random delay, the resulting location and probability density function of the clock values and remaining sojourn time.

Definition 3.5 *A* stochastic state class *for the* STA $\langle \mathcal{A}, \mu, w \rangle$ *is a tuple* $\langle \ell, D, f \rangle$ *where* ℓ *is a location of* $\mathcal{A}$*,* $D \subseteq \mathbb{R}_{\leq 0}^{|X|} \times \mathbb{R}_{\geq 0}$ *is the support of* $f : D \to [0, 1]$*, the probability density function over clocks and sojourn time.*

Note that in the latter definition, the support of sojourn time is $\mathbb{R}_{\geq 0}$, while that of clocks is $\mathbb{R}_{\leq 0}$. The reason is that sojourn time decreases when time elapses while clocks values increase. The support D of the probability density function can therefore be encoded by a Difference Bounds Matrix (DBM). DBM offer the advantage to provide a compact representation and are preserved by all operations needed in the computation of successors.

Given a stochastic state class, depending on which edge is selected at random, one can compute the set of all possible next distributions, together with transition probabilities from the source state class. Iterating this process, one can build incrementally a tree in which each node is a stochastic state class, and edges represent the successor relation and are labelled with probabilities. Of course, this tree is a priori infinite for two reasons: first, it is well known that the symbolic forward computation of the reachability set does not necessarily terminate for timed automata, and extrapolation operators are neeeded to ensure convergence; second, and more specifically related to our context, there are uncountably many continuous distributions over a fixed DBM. These two difficulties are dealt with by imposing some hypotheses on the stochastic timed automaton under consideration (or rather its induced stochastic process).

The exact evaluation of probabilities up to a given time bound requires the enumeration of all stochastic state classes that can be reached within T. The termination of the enumeration of relevant stochastic state classes relies on the guarantee of time progression. To ensure termination of the scheme, we gave a necessary condition that is essentially equivalent to the strongly non-Zeno hypothesis, that we reformulate here in terms of minimum sojourn time.

Theorem 3.9 *For a fixed timed bound* T*, the construction of the tree of stochastic state classes for which the support of the sojourn time is within* $[0, T]$ *terminates provided that every cycle of* $\mathcal{A}$ *contains at least one location from which the minimum sojourn time is lower bounded by a positive constant.*

As a consequence of Theorem 3.9, under the termination condition, one can precisely compute the probability of time-bounded until properties for stochastic timed automata. If an error bound ε is allowed, the construction of the tree can be stopped as soon as the total probability of reaching any leaf node before the time-bound is bounded by ε. In this case, the necessary condition can be relaxed by requiring time progression almost surely (rather than surely).

Note that in this document, we described the latter contribution for stochastic timed automata. It was originally presented for networks of such models that can synchronize (through broadcast communications). More details, examples, and reports on an implementation for the transient analysis of networks of stochastic timed automata based on stochastic state classes can be found in [BBH+13].

3.4 Games on STA

So far, we considered purely stochastic processes induced by STA. In order to model interaction of the system with an environment, or between adversarial agents, we now move to a game framework in which: the delays are random – as they were in the case of STA – and the choice between enabled transitions is left to a decisioner. More precisely, we focused on the framework where locations of the timed automaton are split between two players, that have antogonist objectives: one player aims at reaching a target set of locations, and his or her opponent has the opposite goal.

We first concentrate on such reachability objectives with a time bound, and then move to time-unbounded reachability.

3.4.1 Optimizing time-bounded reachability

Let us stick to the rather smooth subclass of STA formed of reactive STA. Moreover, we assume that delays are exponentially distributed in all states, and that the rate is location-uniform: each location $\ell \in L$ is associated with a rate $\Lambda(\ell)$. Also, we assume the set of locations is partitionned into $L_1 \sqcup L_2$, corresponding to locations of one player or the other.

The decisions of each player in games on STA is formalized through the notion of strategy. We write Runs_i for the set of finite runs in $\mathcal{A}$ ending in a location of Player i. A strategy for Player i is a function $\sigma_i : \mathsf{Runs}_i \times \mathbb{R}_{\geq 0} \to \mathsf{Dist}(E)$ such that for every e_n with $\sigma_i(s_0 \xrightarrow{t_0,e_0} s_1 \cdots s_n, t_n)(e_n) > 0$, there exists a transition $s_n \xrightarrow{t_n,e_n}$.

As pointed out in [WJ06], not all strategies are meaningful, already in the context of CTMDP, because of measurability considerations. We therefore focus on so-called measurable strategies, that can, *e.g.* be obtained from cylindrical strategies by completion (see [RS11] for details).

Given a game on STA, a strategy profile (σ_1, σ_2) composed of measurable strategies yields a probability measure, denoted $\mathbb{P}_{(\sigma_1,\sigma_2)}$ over $\mathsf{Runs}(\mathcal{A})$.

We proved that the optimal probability can be attained by a measurable strategy profile, therefore showing that optimal strategies exist [BS12]. Formally:

Theorem 3.10 *For every game on* STA *with reachability objective $G \subseteq L$ and time-bound T, there exists a measurable strategy profile (σ_1, σ_2) such that*

$$\mathbb{P}_{\sigma_1,\sigma_2}(\Diamond^{\leq T} G) = \sup_{\sigma_1} \inf_{\sigma_2} \mathbb{P}_{\sigma_1,\sigma_2}(\Diamond^{\leq T} G) \ .$$

Moreover, the proof shows that positional strategies, *i.e.* that depend only on the current state, the sampled delay and the remaining time, are sufficient.

Limits As explained above, memoryless and deterministic strategies are sufficient for the optimal time-bounded reachability probability. A question of interest is whether one can restrict to even simpler strategies. Clearly, location-based strategies are not powerful enough for games on STA, but a slight generalization would be to consider region-based strategies, or even polyhedral strategies that base their decision on a partition of valuations into finitely many polyhedra. Unfortunately, one can show that polyhedral strategies are not sufficient for time-bounded reachability properties. Consider the example of Figure 3.10, in which the objective is to maximize the probability to reach the goal G within 1 time-unit. The only

non-trivial decision the strategy has to make in that example is in location ℓ_0, where it has to choose between looping back to ℓ_0 or progressing towards ℓ_1. In order to determine an optimal strategy, one can show that looping is optimal iff $(t+1)e^{-t} \leq e^{x-1}$, in which x denotes the clock value and t the remaining time until the time-bound. Obviously, this set cannot be represented by a finite union of polyhedra.

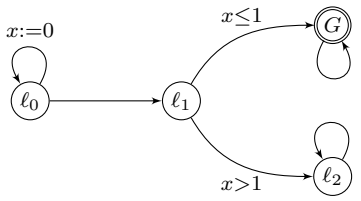

Figure 3.10: A simple 1-player game over a single-clock STA.

Note also that the restriction to time-bounded reachability is necessary in order to obtain our optimality result. Consider again the example of Figure 3.10. It is easy to see that the chances of reaching G from ℓ_1 are 0 if the value of the clock x is greater or equal to 1, and $e^{-\varepsilon} - e^{-1}$ for a clock value $\varepsilon \in [0, 1]$. This implies an upper bound on the time-unbounded reachability of $1 - e^{-1}$. This value attained (up to any desired error) by a scheduling policy that guarantees a time-unbounded reachability $> 1 - e^{-1} - \varepsilon$ by progressing to ℓ_1 iff the clock value of x is smaller than ε. While this defines the value of time-unbounded reachability, it does not provide a strategy that realises this value. Consider the strategy that, for any $\varepsilon \in]0, 1]$, provides a positive probability p_ε to progress to ℓ_1 with a clock valuation $\geq \varepsilon$, then the likelihood of reaching G is bounded by $1 - e^{-1} - p_\varepsilon(1 - e^{-\varepsilon})$. At the same time, this chance being 0 for all $\varepsilon > 0$ implies that we almost surely never progress to ℓ_1. Consequently, no optimal strategy exists for the time-unbounded reachability of G.

3.4.2 Limit-sure reachability

Optimal strategies do not necessarily exist for games on STA, even for a unique player, and assuming the underlying automaton has a single clock. A natural question is whether the value (*i.e.* the supremum probability over all strategies), can be computed or at least approximated. We first tackled the simpler problem of deciding whether the value is 1. Formally, given a single-clock STA, for every tolerance $\varepsilon > 0$, does there exists a strategy that ensures reaching the objective with probability greater than $1 - \varepsilon$?

To decide the value 1 problem, and following our usual approach for stochastic timed automata, we build a finite abstraction based on a refinement of regions, and consider the almost sure reachability problem in this abstraction. Let us start with an example, that will illustrate the whole construction. The 1-player STA represented on Figure 3.11 has value 1, when the objective is to reach the ☺-location. Intuitively, an ε-optimal strategy from $(\ell_0, 0)$ to reach ☺ is the following: stay in ℓ_0 until a large clock value is sampled, then move to ℓ_1; if then the sampled clock value is above 1, move back to ℓ_0 and iterate the same process, otherwise, proceed to ℓ_2; finally, reach ☺ or ☹ depending on the sampled clock value in ℓ_2.

Performing a simple region construction on this example would not help. On the example above, we saw that the family of ε-optimal strategies behave non-uniformly inside regions: roughly the decision can be different in the left part and in the right part of an open region.

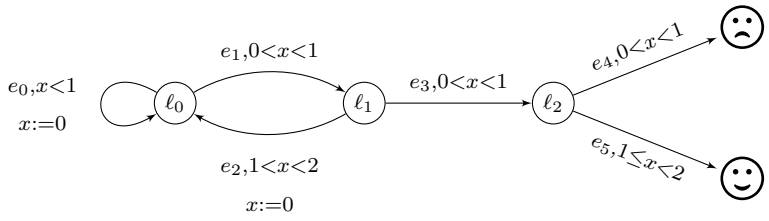

Figure 3.11: A 1-player game on STA with value 1.

This motivates the use of pointed regions, initially introduced in the corner-point abstraction to solve an optimal scheduling problem [BBL08]. Bounded open regions are duplicated in a left copy and a right copy, with the intuitive interpretation of being close to the infinimum value or the supremum value of the region.

Also, contrary to purely qualitative questions, when interested in the value 1 problem, some leaking probability is acceptable (as soon as it is bounded by the tolerance), and it should somehow be represented in the abstraction. To model this, we allow players to "cheat" by selecting an action, unfeasible in the concrete STA that from a right corner jumps to the next open region. These two observations lead us to the limit corner-point abstraction, that we first illustrate on our example. To the exception of the limit edges e_1^{limit} and e_3^{limit}, we rely

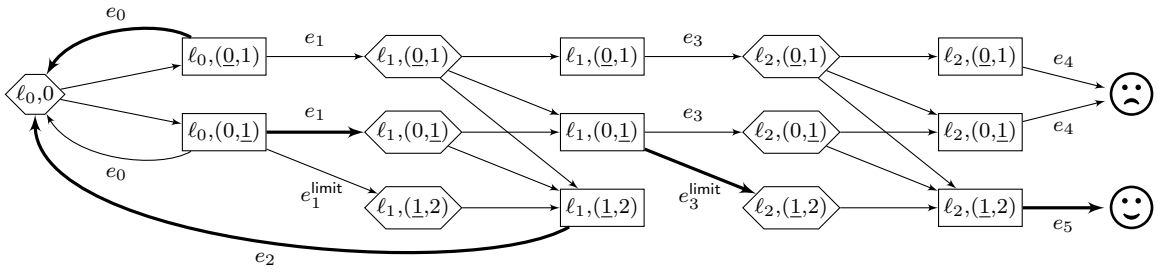

Figure 3.12: The limit corner-point abstraction for the STA of Figure 3.11.

on the standard corner point abstraction: open regions are simply duplicated into a left and a right copy, indicated by underlined corner, *e.g.* $(0, \underline{1})$. Now, intuitively, limit edges allow the player to cheat by taking a transition in the next open region although it is not allowed by the guard. As an example, e_1^{limit} can be seen as a copy of e_1 from region $1 < x < 2$. Limit edges are *ad hoc* to the value 1 problem, and reflect that we want to reduce to an almost-sure question.

To formally define the limit corner-point abstraction, we introduce some notations. For a bounded open region $(c, c+1) \in \mathsf{Reg}_M^{\{x\}}$, we write $\mathsf{left}(R)$ for its left version in the corner-point, *i.e.* $(\underline{c}, c+1)$. We generalize the notation to arbitrary regions $R \in \mathsf{Reg}_M^{\{x\}}$: punctual regions and the unbounded region are unchanged.

Definition 3.6 *Let $\mathcal{A}$ be a 1-player game on a single-clock* STA*. The* limit corner-point abstraction *associated with $\mathcal{A}$ is the MDP $\mathcal{M}_{\mathcal{R}}(\mathcal{A}) = (Q^{(n)} \sqcup Q^{(p)}, q_0, \Sigma_{\mathcal{M}}, \Delta_{\mathcal{M}})$ with*

- *$Q^{(n)} = L \times \mathcal{R} \times \{\mathsf{left}, \mathsf{right}\} \times \{(n)\}$ is the set of nondeterministic states, of the form $(\ell, \mathsf{left}(R))^{(n)}$ or $(\ell, \mathsf{right}(R))^{(n)}$;*

- $Q^{(p)} = L \times \mathcal{R} \times \{\mathsf{left}, \mathsf{right}\} \times \{(p)\}$ *is the set of probabilistic states, of the form* $(\ell, \mathsf{left}(R))^{(p)}$ *or* $(\ell, \mathsf{right}(R))^{(p)}$*;*
- $q_0 = (\ell_0, 0)^{(p)}$ *is the initial state;*
- $\Sigma_{\mathcal{M}} = E \cup E^{\mathsf{limit}}$ *is the set of actions (available from nondeterministic states);*
- $\Delta_{\mathcal{M}} = \Delta_{\mathcal{M}}^{(n)} \sqcup \Delta_{\mathcal{M}}^{(p)}$ *with*
  - $\Delta_{\mathcal{M}}^{(n)} \subseteq Q^{(n)} \times \Sigma_{\mathcal{M}} \times Q^{(p)}$ *is such that* $\Delta_{\mathcal{M}}^{(n)}((\ell, \mathsf{side}(R))^{(n)}, e) = (\ell', \mathsf{side}(R'))^{(p)}$ *iff* $e \in E$ *and* $(\ell, R) \xrightarrow{e} (\ell', R')$*, moreover* $\Delta_{\mathcal{M}}^{(n)}((\ell, \mathsf{right}(R))^{(n)}, e^{\mathsf{limit}}) = (\ell', \mathsf{left}(R'))^{(p)}$ *iff* $(\ell, R) \xrightarrow{e} (\ell', R'')$ *and* R' *is the first non-negligible time-successor of* R''*; and*
  - $\Delta_{\mathcal{M}}^{(p)} : Q^{(p)} \to \mathsf{Dist}(Q^{(n)})$ *is such that* $\Delta_{\mathcal{M}}^{(p)}((\ell, \mathsf{side}(R))^{(p)})((\ell', \mathsf{side}'(R'))^{(n)}) > 0$ *iff* $\ell' = \ell$ *and* R' *is a non-negligible time-successor of* R*, and if* $\mathsf{side} = \mathsf{right}$*, then* $\mathsf{side}' = \mathsf{right}$.

Note that for convenience, states in the limit corner-point MDP are partitionned into nondeterministic states and probabilistic ones. Let us illustrate the construction of the limit corner-point abstraction represented in Figure 3.12. In this figure, nondeterministic states are boxes and probabilistic states are diamond-shaped. The possible time-sucessors of $(\ell_0, 0)^{(r)}$ are $(\ell_0, 0)^{(n)}$, $(\ell_0, (\underline{0}, 1))^{(n)}$ and $(\ell_0, (0, \underline{1}))^{(n)}$ since from ℓ_0 the union of guards on outgoing edges is the interval $[0, 1)$. Now, among these successors, $(\ell_0, 0)^{(n)}$ is negligible; this explains that it is not represented in the figure. From nondeterministic state $(\ell_0, (\underline{0}, 1))^{(n)}$, which is a left region, only normal edges ($e \in E$) can be fired, in this case, e_1, and the successor is the one of the traditional corner-point abstraction. Now, from the right region $(\ell_0, (0, \underline{1}))^{(n)}$, edge e_1 can be fired, but also its limit copy e_1^{limit}. The intuitive reason for this is that from the next location ℓ_1, clock values beyond $x = 1$ are allowed to then fire ℓ_3. We therefore allow the player to "cheat" and take a limit edge to represent that the clock value was so close to 1, that it could as well be 1. More detailed justifications on this contruction can be found in [BBG14].

Proposition 3.4 *Let* $\mathcal{A}$ *be a 1-player* STA *and* s *a state of* $\mathcal{A}$*, and* F *a set of goal locations. Then:*

$$\mathsf{val}_{\mathcal{A}}(s) = 1 \iff \max_{\sigma} \mathbb{P}^{\sigma}_{\mathcal{M}_{\mathcal{R}}(\mathcal{A})}(\mathsf{left}([s]) \models \Diamond F) = 1 \ .$$

The intuition of turning a value 1 problem into a probability 1 problem might seem clear. Yet, the proof of Proposition 3.4 is non-trivial.

A first easy observation is that ε-strategies cannot be uniform by region. Quite obviously, on the example of Figure 3.11, in order to achieve a high probability to reach the target ☺-state, the strategy should only decide to progress towards ℓ_1 when the sampled value for clock x in ℓ_1 is close enough to 1. Indeed, otherwise, the chance would be too high to be forced in the next step to move to ℓ_2 and lose the step after. On the contrary the probability when reaching ℓ_2 should be high to sample a clock value in $[1, 2[$ rather than smaller than 1. More generally, we prove that ε-optimal strategies can be taken positional and of the following form: with each pair consisting of a location and a region is associated a cutpoint τ in the region, and the choice made by the strategy is uniform to the left of the cutpoint, and to the right of it. Such strategies reflect the high level strategies given at the corner-point MDP level. On our example, the best decision in the left and right part of $(\ell_0, (0, 1))$ are different, as the limit

corner-point shows. More precisely, one can compute a value $\tau \in (0,1)$ such that from any state (ℓ_0, t) with $0 < t \leq \tau$, the strategy is to loop on ℓ_0 (and reset the clock), and from any state (ℓ_0, t) with $\tau < t < 1$, the strategy is to progress to ℓ_1.

Quite surprisingly, the choice of the cutpoint must depend on the pair location and region. This is illustrated on the example of Figure 3.13, where the implicit probability distributions over delays are all uniformly distributed. Decisions can only be taken in locations ℓ_0 and

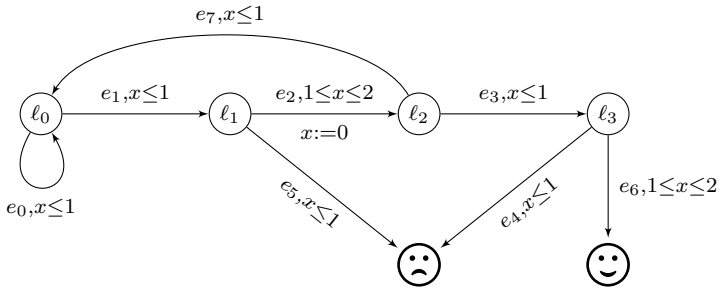

Figure 3.13: An example where non-uniform cutpoints are needed.

ℓ_2, where transitions with overlapping guards are possible. Intuitively, from ℓ_0, to reach ☺, transition e_1 needs to be taken, with a risk that once ℓ_1 is reached, transition e_5 is triggered. Let t_0 be the cutpoint in ℓ_0 such that if $t_0 < t < 1$, the player decides to take e_1 from (ℓ_0, t). In the same way, let $t_2 \in (0,1)$ be the cutpoint in ℓ_2 such that if $t_2 < t < 1$, the player decides to take e_3 from (ℓ_2, t). To reach a contradiction, we assume $t_0 = t_2$, and write τ for this value. From $[\ell_2, t]$ with $1 - \tau < t < 1$, a simple calculation shows that the probability to lose is $p^T_{\mathsf{lose}}(\ell_2, t) = (1-t)/(2-t)$. Also, from $[\ell_0, t]$, the losing probability is lower bounded by the probability to lose in two steps, directly from ℓ_1, hence $p^T_{\mathsf{lose}}(\ell_0, t) \geq (1-t)/(2-t)$. Moreover, $p^T_{\mathsf{win}}(\ell_0, t) \leq p^T_{\mathsf{win}}(\ell_1, t) \leq (1-t)p^T_{\mathsf{win}}(\ell_3, t) \leq (1-t)$. Hence, $p^T_{\mathsf{lose}}(\ell_0, t) > p^T_{\mathsf{win}}(\ell_0, t)/2$ for all $t > 1 - \tau$, that is $\mathbb{P}_T(\langle \ell_0, t \rangle \models \Diamond F) < 2/3$. This shows that F is not limit-surely reachable under simple strategies, defined by uniform cutpoints. Yet, the ☺-state is limit-surely reachable from $[\ell_0, 0]$. However, to achieve this, t_0 needs to be set to a much lower value than t_2, *e.g.* $t_2 = \tau$ and $t_0 = \tau^2$.

From Proposition 3.4 we deduce that whether s has value 1 in a 1-player STA $\mathcal{A}$ can be decided in polynomial time. To establish this complexity bound, it suffices to recall that for single-clock (stochastic) timed automata, the number of regions is linear [LMS04], thus the corner-point abstraction is linear in the size of the STA; moreover, checking almost sure reachability properties for MDP is doable in PTIME.

Theorem 3.11 *The value 1 problem is decidable in* PTIME *for 1-player* STA*.*

The solution we propose to the value 1 problem for 1-player STA is already non-trivial, and we proved that ε-optimal strategies, although positionnal, may already need to be involved. The general scheme to achieve a value close to 1 is to "push" the clock value to the right of a region before risking a progressive move. This is very particular to the value 1 problem, and only little connected with the asymmetry of time elapsing. In particular, if one wants to optimize the value –in case it is not 1–, it may be better to loop until the clock value is close enough to a left border of a region before daring a risky move. This is illustrated on the example of Figure 3.14 whose value from state $(\ell_0, 0)$ to reach ☺ is 0.5. In that STA, in order

to maximize the probability to reach the target location, one should progress from ℓ_0 to ℓ_2 when the clock value is as close as 0 as possible.

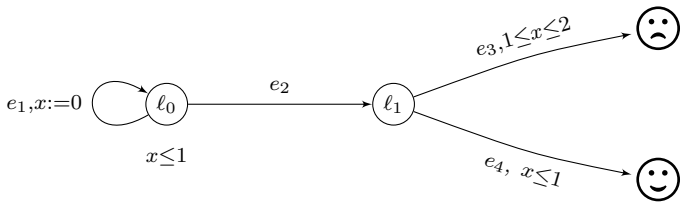

Figure 3.14: An example of 1-player STA with value 0.5.

3.5 Perspectives

Reactive stochastic timed automata Together with Patricia Bouyer and Thomas Brihaye, we are currently extending the latter decidability result for the value 1 problem to reactive stochastic timed automata, with arbitrary many clocks. It seems that a construction similar to the limit corner-point abstraction is feasible for reactive STA, allowing one to decide the value 1 problem.

Moreover, this subclass of stochastic timed automata share *e.g.* with probabilistic lossy channel systems, the nice property of having a finite attractor [BBS06]. Essentially, although their underlying stochastic process has infinitely many states, it behaves almost like a finite-state Markov chain, in that regeneration points (from a finite set) are encountered almost surely infinitely often. Given the results we obtained for probabilistic lossy channel systems, it is worth investigating whether our generic fixpoint termination theorem [BS13a] may be used in the context of reactive stochastic timed automata. In particular, we aim at developing approximation schemes for the quantitative analysis of MDP or games semantics for reactive stochastic timed automata. Such approximation algorithms for infinite state MDP are currently missing, even for channel systems with probabilistic losses. New ideas are needed to extend the approximation scheme for reachability in decisive Markov chains [PN97, ABM07] to MDP, with the hope that it then applies to MDP generated by STA.

Qualitative model checking Qualitative model checking has, till now, only been proven decidable for subclasses of timed automata (single-clock, reactive), and the question remains open for general stochastic timed automata. As explained earlier, the weird pacman example of Figure 3.4, cannot be analysed through its region Markov chain abstraction, due to a convergence of some clock value towards a constant when entering a given location. Such convergence phenoma, that generalizes Zeno behaviours, are typical of so-called non-forgetful timed automata [BA11, Pur00]. On the contrary, forgetful timed automata do not exhibit convergent behaviours, and have better properties than general timed automata. For example, forgetful automata have a non-degenerated entropy [BA11]. Also, the existence of a forgetful cycle in a timed automaton ensures its robust controllability [SBMR13]. We propose to explore whether the region Markov chain abstraction is correct for the almost sure model checking of forgetful stochastic timed automata.

For general timed automata, the decidability status of the qualitative model checking problem is wide open. Here again, we will investigate whether the analysis of non-forgetful cycles helps to design a decision algorithm. More precisely, analysing such cycles with convergence behaviour could well lead to a refinement of the region Markov chain abstraction that is correct to decide the almost sure model checking for general stochastic timed automata.

From value 1 problem to qualitative verification We have shown the value 1 problem to be decidable for 1-player games on single-clock stochastic timed automata. Maybe suprisingly, this problem is simpler than –formally, polytime reducible to– the qualitative model checking of 1-player games on STA, and more precisely to the almost sure model checking of co-Büchi properties. Indeed, one can rely on reduction similar to the one presented in the context of probabilistic automata (see Figure 4.3, page 50). Therefore a solution to the qualitative model checking problem for games on stochastic timed automata should incorporate all technicalities we presented for the value 1 problem, and this already for single-clock timed automata. This motivates *a posteriori* that we solved the value 1 problem before providing a solution to qualitative objectives in games on STA.

Chapter 4

Partially observable probabilistic systems

Partial observation stochastic games are a convenient framework to represent at the same time systems with failures or uncertainties, and in which only a partial information about the current state is available, due *e.g.* to limited sensing capacities.

A variety of applications with high societal impact can be recast in the framework of partial observation stochastic games. As an example, train traffic management rely on signalling equipment and sensors. In order to maintain low operational costs, the positionning of the sensors may be rather sparse so that the supervisor only has partial knowledge of the system's current state. Moreover, unexpected errors such as disfunctionning of equipment, as well as imprecision in figures, such as instantaneous speed compared to averaged one, are conveniently abstracted using probability distributions. The objective of the supervisor is then to send speed commands to each of the trains to avoid at any cost collisions, while being on schedule as much as possible.

Probabilistic models under partial observation The simplest model combining probabilities and partial observation is the one of probabilistic automata, dating back to the sixties and due to Rabin [Rab63] (see also [Paz71]). Compared to nondeterministic finite automata, probabilistic automata resolve nondeterminism by probability distributions, and thus allow one to consider quantitative versions of languages by assigning an acceptance probability to each word, rather than a Boolean. One can also define classical (non-quantitative) languages by considering the set of words with probability greater than a threshold. Unfortunately, for such threshold languages, unless if the threshold is 0 or 1, the emptiness problem is undecidable.

Probabilistic automata, that were first studied from a language-theoretic point of view, can be seen as a particular case of partially observable Markov decision processes (POMDP) [Ast65]. POMDP generalize Markov decision processes by revealing to the decisionner (or player) only partial information about the current state of the system. In probabilistic automata, the player is blind: it has no prior information on the current state, and may only derive a belief from the model and the sequence of letters played so far. A word in a probabilistic automaton thus corresponds to an *a priori* fixed deterministic strategy. On the contrary, in POMDP, a strategy can be represented by a tree, and the player can choose a different action depending not only on the past sequence of actions, but also on the observations received so far. In rela-

tion to verification, POMDP are mostly considered under infinite horizon semantics, that is for strategies corresponding to infinite trees. In contrast, probabilistic automata were first studied for finite words. A last difference is that strategies in POMDP can be randomized, although is has been shown that for reasonable objectives –precisely Borel objectives– deterministic strategies suffice to optimize the probability [CDGH10].

The richer model combining probabilities and partial observation that we will consider in this document is the one of stochastic 2-player games, in which both players are partially informed. More precisely, we consider stochastic games with signals [Sor02, RSV03, Ren07], in which players cannot observe the actual state of the game and are only informed by private signals they receive throughout the play. Stochastic games with signals in particular subsume concurrent games [dAH00] and deterministic games with imperfect information on one side [Rei84, CDHR07].

Outline of the contributions This chapter gathers our contributions to the control of probabilistic systems under partial observation. It starts with probabilistic Büchi automata, a variant of probabilistic automata to recognize Boolean languages of infinite words. Different to probabilistic automata, their emptiness even for a threshold 0 is non trivial, and even undecidable. After a deep study of properties of probabilistic Büchi automata in Section 4.1, we move to a contribution for POMDP, in which we tackle the problem of minimal information requirement to achieve a reachability objective almost surely (see Section 4.2). Then we present in Section 4.3 a determinacy result for the general model of stochastic games with signals, together with optimal memory constraints and complexity bounds. Last, we review our contributions to the diagnosis problem for probabilistic systems in Section 4.4.

The control of probabilistic systems under partial observation is at the same time a challenging and attractive research avenue. Together with Serge Haddad, we presented some foundation results at a young researchers school, and published lecture notes for students [BH15].

4.1 Probabilistic Büchi automata

Probabilistic Büchi automata have been introduced by Baier and Größer in [BG05] as probabilistic acceptors for languages of infinite words. They extend probabilistic finite automata (PFA) *à la* Rabin by considering infinite words. Similarly to PFA, given an infinite word, one can consider the probability of the set of runs on that word that are accepted. Yet, contrary to PFA, the emptiness problem for probabilistic Büchi automata with threshold 0 is not trivially reducible to the emptiness of the underlying non-probabilistic automaton. We will even see that this decision problem is undecidable. Also, the language recognized by probabilistic Büchi automata with threshold 0 may depend on the precise probability distributions. We also studied probabilistic Büchi automata under an almost sure semantics, by considering as language the set of words accepted almost surely. In the next two subsections, we detail the main results we obtained for probabilistic Büchi automata under the positive semantics (probability > 0) and almost sure semantics (probability $= 1$). Before that, let us formally define probabilistic Büchi automata.

Definition 4.1 *A* probabilistic Büchi automaton *(*PBA*) over alphabet Σ is a tuple $\mathcal{B} = (Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of states, with $q_0 \in Q$ the initial state; F is the set*

of accepting states; and $\delta : Q \times \Sigma \times Q \to [0,1]$ *is the probabilistic transition function such that: for every* $q \in Q$ *and* $a \in \Sigma$, $\sum_{q' \in Q} \delta(q,a,q') \in \{0,1\}$.

Examples of PBA are given in Figure 4.1, in which final states are doubly circled. In these examples and in the sequel, we only depict transitions with positive probabilities.

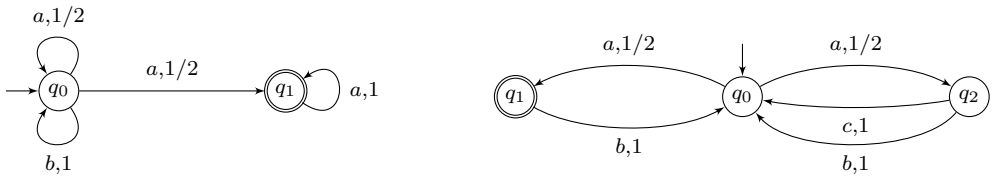

Figure 4.1: Examples of probabilistic Büchi automata.

4.1.1 Positive semantics

Let us first consider probabilistic Büchi automata under the positive semantics. Given a PBA $\mathcal{B}$ and $w \in \Sigma^\omega$ an infinite input word, the behaviour of $\mathcal{B}$ is described by the infinite-state Markov chain obtained by unfolding $\mathcal{B}$ and applying the strategy w. In this induced Markov chain, we write $\mathbb{P}(w)$ for the acceptance probability of w in $\mathcal{B}$, that is, the probability measure of the set of runs over w that satisfy the Büchi acceptance condition of $\mathcal{B}$. The language recognized by $\mathcal{B}$ is then easily defined:

Definition 4.2 *Let* $\mathcal{B}$ *be a* PBA. *Its* language under the positive semantics *is defined as*

$$\mathcal{L}_{>0}(\mathcal{B}) = \{w \in \Sigma^\omega \mid \mathbb{P}(w) > 0\}.$$

Consider the example PBA depicted on the left of Figure 4.1. Ignoring the probabilities, it can be seen as a nondeterministic Büchi automaton (NBA) with language $(a+b)^*a^\omega$. This ω-regular language is also the language accepted by the PBA under positive semantics. Indeed, in general, the language of PBA is included in the language of its underlying NBA. On that example the two languages coincide: any word in $(a+b)^*a^\omega$ is accepted with positive probability, namely 2^{-k} where k is the number of a's before the last b.

However, most likely, the NBA and PBA languages differ, as illustrated by the PBA on the right of Figure 4.1. Its underlying NBA accepts the language $((ac)^*ab)^\omega$, whereas the PBA accepts $(ab+ac)^*(ab)^\omega$ under the positive semantics. Intuitively, any word in $(ab+ac)^\omega$ with infinitely many c's is rejected, because before reading a c, the probability to move with a to q_1 is half, and it will thus happen almost surely that some c cannot be consumed.

Language dependency on precise probabilities

For many probabilistic models, qualitative questions do not depend on the probability distributions, but only on their support. For example, the set of recurrent states in a finite-state Markov chain is independent of the precise probabilities, and these values are only relevant when one wants to compute the steady state distribution. Probabilistic Büchi automata contrast with this general observation, and one can show that the set of words accepted with positive probability may depend on the precise probability distributions, not only on their support.

Proposition 4.1 *There exists a family of* PBA $(\mathcal{B}_\lambda)_{\lambda\in(0,1)}$ *parameterised by* $\lambda \in (0,1)$ *such that for* $\mu \neq \lambda$, $\mathcal{L}_{>0}(\mathcal{B}_\mu) \neq \mathcal{L}_{>0}(\mathcal{B}_\lambda)$.

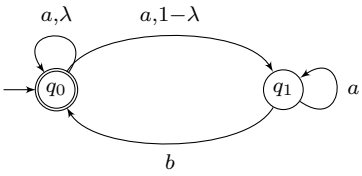

Figure 4.2: PBA $\mathcal{B}_\lambda$ with $\lambda \in (0,1)$.

A family of PBA witnessing Proposition 4.1 is depicted in Figure 4.2. The automata can be completed by adding a sink state that can be reached from state q_0 on reading b. We start by justifying the language accepted by $\mathcal{B}_\lambda$. The only accepting state is q_0, and, clearly enough, in order to visit it infinitely often with positive probability, a word should contain infinitely many b letters. Also reading two consecutive b's surely leads to the sink state. Let us therefore consider a word $w = a^{k_1}ba^{k_2}b\cdots$, with $k_i \in \mathbb{N}$ for $i > 0$, of the only candidate shape for accepted words. After reading the first k_1 a-letters, the probability to still be in state q_0, and therefore to move to the sink when reading b, is λ^{k_1}. A simple induction thus shows that the acceptance probability of w is $\mathbb{P}(w) = \prod_{i>0}(1-\lambda^{k_i})$. We deduce the language of words accepted with positive probability by $\mathcal{B}_\lambda$:

$$\mathcal{L}_{>0}(\mathcal{B}_\lambda) = \{a^{k_1}ba^{k_2}b\cdots \mid \prod_{i>0}(1-\lambda^{k_i}) > 0\}.$$

Notice already that this language is not ω-regular. The nonregular convergence condition for the words accepted by $\mathcal{B}_\lambda$ can be explained by the observation that there are finite input words that $\mathcal{B}_\lambda$ rejects with arbitrary small probability. More precisely, when k tends to infinity, the word a^kb is rejected with probability λ^k, which tends to 0.

Moreover, the language $\mathcal{L}_{>0}(\mathcal{B}_\lambda)$ depends on the parameter λ. Intuitively, the smaller λ, the bigger the product $\prod_{i>0}(1-\lambda^{k_i})$, so that $\mathcal{L}_{>0}(\mathcal{B}_\lambda) \subseteq \mathcal{L}_{>0}(\mathcal{B}_\mu)$ whenever $\lambda > \mu$. Let us now show that the inclusion is strict. For $\lambda > \mu$, we consider two integers $n, m \in \mathbb{N}$ such that $\mu < \frac{n}{m} < \lambda$. Let us pick the nondecreasing sequence $(k_i)_{i>0}$ such that $\lfloor(\frac{m}{n})^j\rfloor$ elements are equal to j, and show that the word $w = a^{k_1}ba^{k_2}b\cdots$ belongs to $\mathcal{L}_{>0}(\mathcal{B}_\mu)$ but not to $\mathcal{L}_{>0}(\mathcal{B}_\lambda)$. Observe first that $\prod_{i\geq 1}(1-x^{k_i}) > 0$ if and only if the series $\sum_{i\geq 1}\log(1-x^{k_i})$ converges. Since $\log(1-\varepsilon) \sim_{\varepsilon\mapsto 0} -\varepsilon$, the latter series behaves as $\sum_{i\geq 1} x^{k_j}$ (*i.e.*, either both converge, or both diverge). For the sequence $(k_i)_{i\in\mathbb{N}}$ we picked, it holds that $\sum_{i\geq 1} x^{k_i} = \sum_{j\geq 1}\lfloor(\frac{m}{n})^j\rfloor \cdot x^j$, which converges if and only if $\sum_{j\geq 1}(\frac{m}{n})^j \cdot x^j$ converges, and the latter is equivalent to $x < \frac{n}{m}$. Therefore, the word we exhibited belongs to $\mathcal{L}_{>0}(\mathcal{B}_\mu) \setminus \mathcal{L}_{>0}(\mathcal{B}_\lambda)$, and thus $\mathcal{L}_{>0}(\mathcal{B}_\lambda) \subsetneq \mathcal{L}_{>0}(\mathcal{B}_\mu)$.

Closure under complementation

As for ω-regular languages, it is not difficult to show that PBA are closed under union and intersection. Closure under complementation however, is harder to establish, but actually holds for the positive semantics.

Theorem 4.1 *Probabilistic Büchi automata (under the positive semantics) are closed under complementation: for each* PBA $\mathcal{B}$*, one can effetively construct a* PBA $\mathcal{B}'$ *such that* $\mathcal{L}_{>0}(\mathcal{B}') = \Sigma^\omega \setminus \mathcal{L}_{>0}(\mathcal{B})$.

In order to complement a PBA under positive semantics, the idea is to provide a series of transformations. First, from $\mathcal{B}$ we build an equivalent probabilistic automaton with Rabin acceptance condition (PRA) $\mathcal{R}$ and such that every input word is accepted either with probability 0 or with probability 1. This very particular PRA is then easily complemented into a probabilistic Streett automaton (PSA) $\mathcal{S}$ with the same property. Last, the PSA is transformed into an equivalent PBA $\mathcal{B}'$. The resulting PBA $\mathcal{B}'$ we construct is exponential in the size of the original PBA $\mathcal{B}$.

Full details about the series of transformations can be found in the article [BBG12]. In this document, we only provide high-level ideas on how each transformation works.

Let us start with the transformation of a PBA $\mathcal{B}$ into an equivalent PRA $\mathcal{R}$ that accepts words in a binary way: either with probability 0 or with probability 1. This transformation has similarities with Safra's determinization algorithm for NBA [Saf88] and also relies on some kind of powerset construction. However, we argue that the probabilistic setting is slightly simpler. Instead of organizing the potential accepting runs in Safra trees, we may deal with up to n independent sample runs (where $n = |Q|$ is the number of states in $\mathcal{B}$) that are representative for all potential accepting runs. The idea is to represent the current states of the sample runs by tuples $\langle q_1, \cdots, q_k \rangle$ of pairwise distinct states in $\mathcal{B}$. Whenever two sample runs meet at some point, say the next states q_1' and q_2' in the first two sample runs agree, then they are merged, which requires a shift operation for the other sample runs and yields a tuple of the form $\langle q_1', q_3', \cdots q_k', \cdots q \cdots \rangle$ where q_i' is a successor of q_i in the i-th sample run. Additionally, new sample runs are generated in case the original PBA $\mathcal{B}$ can be in an accepting state $f \in \{q_1', \cdots, q_k'\}$. The Rabin condition serves to express that at least one of the sample runs enters the set F of accepting states in $\mathcal{B}$ infinitely often and is a proper run in $\mathcal{B}$ (*i.e.*, is affected by the shift operations only finitely many times). Intuitively, the resulting automaton $\mathcal{R}$ simulates $\mathcal{B}$; moreover each time $\mathcal{B}$ could be in an accepting state, $\mathcal{R}$ starts a new sample run. Let $w \in \mathcal{L}_{>0}(\mathcal{B})$, thus with positive probability $\mathcal{B}$ can be in an accepting state infinitely often. But then $\mathcal{R}$ almost surely either already is in a corresponding sample run or starts a new sample run infinitely often and from there on accepts the remaining suffix with positive and lower-bounded probability, as it simulates $\mathcal{B}$ and $\mathbb{P}_{\mathcal{B}}(w) > 0$. This implies that the automaton $\mathcal{R}$ accepts w with probability 1.

The second step in the complementation of PBA is an easy consequence of the duality of Rabin and Streett acceptance conditions. The third step is due to Baier and Größer [BG05], and transforms a PSA $\mathcal{S}$ (accepting with probability 0 or 1) into an equivalent PBA $\mathcal{B}'$. Note that the transformation even holds when starting with a PSA without the special property of binary acceptation. Different to the case of nondeterministic automata [SV89], the transformation does not imply an exponential blowup, and one can effectively construct an equivalent PBA, polynomial in the size of the PSA.

Altogether, these three steps provide a complementation algorithm for PBA under the positive semantics.

Undecidability of the emptiness problem

We now turn to the emptiness problem, that asks given $\mathcal{B}$ a PBA whether $\mathcal{L}_{>0}(\mathcal{B})$ is empty or not. Recall that this problem is trivial for PFA with threshold 0 since it amounts to checking for the reachability of the set of final states. We will see that the situation is very different when one considers infinite words.

Theorem 4.2 *The emptiness problem is undecidable for probabilistic Büchi automata (under the positive semantics).*

Theorem 4.2 was originally proved in [BBG08, BBG12] by reducing a variant of the emptiness problem for probabilistic automata with thresholds. The proof builds on the family of PBA represented in Figure 4.2, for which the language depends on the parameter λ. However, for this family, the (non-)emptiness of the language does not depend on the precise probabilities. Yet, this is not generally true: there are families of PBA for which the language emptiness depends on the precise probabilities. Intuitively, combining by intersection two PBA with different convergence phenomena yields such a strange phenomenon, and is the base of our undecidability proof.

As an alternative proof, we explain here how to reduce the value 1 problem for probabilistic automata. Gimbert and Oualhadj established that the isolation problem for threshold 1 is undecidable in PFA [GO10]. They indeed showed the undecidability of the following so-called value 1 problem: given a probabilistic automaton, does there exist for every $\varepsilon > 0$ a word w_ε such that $\mathbb{P}(w_\varepsilon) > 1 - \varepsilon$? In their undecidability proof, they use a similar gadget as we do: two probabilistic automata with conflicting convergence phenomena. We do not detail their elegant proof here, but reuse their undecidability result.

From a probabilistic automata $\mathcal{A}$ we construct a probabilistic Büchi automata $\mathcal{B}$, as illustrated on Figure 4.3. $\mathcal{B}$ extends $\mathcal{A}$ with an extra state $f_\sharp$ that is the unique final state for $\mathcal{B}$, an additional action $\sharp$ and transitions $\delta(f, \sharp, f_\sharp) = 1$ as soon as $f \in F$ is final in $\mathcal{A}$ and $\delta(f_\sharp, \sharp, i) = 1$, where i is the initial state of $\mathcal{A}$.

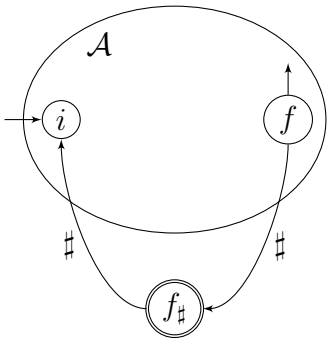

Figure 4.3: Construction of a PBA $\mathcal{B}$ from a PFA $\mathcal{A}$.

Writing $\mathsf{val}(\mathcal{A})$ for the supremum over finite words w of the acceptance probability of w in $\mathcal{A}$, the above construction ensures the following equivalence:

$$\mathsf{val}(\mathcal{A}) = 1 \iff \mathcal{L}_{>0}(\mathcal{B}) \neq \emptyset \ .$$

Indeed, assume first that $\mathsf{val}(\mathcal{A}) = 1$. Then, for every $k \in \mathbb{N}$ there exists $w_k \in \Sigma^*$ such that $\mathbb{P}_\mathcal{A}(w) \geq 1 - \frac{1}{2^k}$. We define the infinite word w as the concatenation of all w_k's separated by

two consecutive $\sharp$-symbols: $w = w_1\sharp\sharp w_2\sharp\sharp w_3 \cdots$. From $\mathbb{P}_{\mathcal{B}}(w) \geq \prod_{k\in\mathbb{N}} 1 - \frac{1}{2^k} > 0$, we derive that $w \in \mathcal{L}_{>0}(\mathcal{B})$.

Assume now that $\mathsf{val}(\mathcal{A}) < 1$. Then, there exists $\varepsilon > 0$ such that, for every word $v \in \Sigma^*$, $\mathbb{P}_{\mathcal{A}}(v) < 1 - \varepsilon$. Observe that, in order to belong to $\mathcal{L}_{>0}(\mathcal{B})$, an infinite word w necessarily contains infinitely many factors $\sharp\sharp$: $w = v_1\sharp\sharp v_2\sharp\sharp v_3 \cdots$ with $v_k \in \Sigma^*$ for every $k \in \mathbb{N}$. Now, $\mathbb{P}_{\mathcal{B}}(w) = \prod_{k\in\mathbb{N}} \mathbb{P}_{\mathcal{A}}(v_k) < \prod_{k\in\mathbb{N}}(1-\varepsilon) = 0$, and thus $\mathcal{L}_{>0}(\mathcal{B}) = \emptyset$.

Let us take a detour, and introduce the well-established model of partially observable Markov decision processes [Ast65]. We consider here simple models with no rewards, and in which the observation is deterministic and only depends on the state of the system.

Definition 4.3 *A* partially observable Markov decision process *(*POMDP*) is a tuple* $\mathcal{M} = (Q, q_0, \mathcal{O}, \mathsf{Obs}, \mathsf{Act}, \Delta)$ *such that*

- Q *is a finite set of states, and* q_0 *the initial state;*
- $\mathcal{O}$ *is a finite set of observations and* $\mathsf{Obs} : Q \to \mathcal{O}$ *assigns an observation to each state;*
- Act *is a finite set of actions; and*
- $\Delta : Q \times \mathsf{Act} \to \mathsf{Dist}(Q)$ *is a partial transition function such that*
  - *for every* $q \in Q$*, there exists* $\alpha \in \mathsf{Act}$ *such that* $\Delta(q, \alpha)$ *is defined; and*
  - *whenever* $\mathsf{Obs}(q) = \mathsf{Obs}(q')$*, for every* $\alpha \in \mathsf{Act}$*,* $\Delta(q, \alpha)$ *is defined if and only if* $\Delta(q', \alpha)$ *is defined.*

In order to obtain a stochastic process from a POMDP, one has to define a strategy. A *strategy* for the POMDP $\mathcal{M} = (Q, q_0, \mathcal{O}, \mathsf{Obs}, \mathsf{Act}, \Delta)$ is a mapping $\pi : (\mathsf{Act}\ \mathcal{O})^* \to \mathsf{Dist}(\mathsf{Act})$ assigning to each alternating sequence of past actions and observations a distribution over the next action. Note that strategies in POMDP are observation-based: the decisioner does not have access to the precise state of the system, but only to its observation. POMDP generalize probabilistic automata, that can be seen as blind POMDP. Moreover, strategies in probabilistic automata are words, and thus correspond to pure (*i.e.* non randomized) strategies in POMDP.

A consequence of the undecidability result from Theorem 4.2, is that the following problem for POMDP is also undecidable: given a POMDP and a subset F of its states, does there exist a pure strategy that ensures visiting F infinitely often with positive probability. This undecidability result extends to the class of more general randomized strategies, thanks to [CDGH10], where it is shown that pure strategies are as powerful as randomized ones in POMDP for any Borel objective.

The undecidability of the emptiness problem for PBA (under positive semantics) can be considered as bad news, since it strongly limitates the applicability of PBA for verification purposes. Several attempts have been made to recover decidability. First of all, Christel Baier and Marcus Größer in their seminal paper identified a subclass of PBA, called uniform PBA, for which the emptiness is decidable [BG05]. Later, Mathieu Tracol proposed an alternative acceptance constraint that strengthens the Büchi condition: not only final states must be visited infinitely often, but they should be visited with positive frequency. Under this strong recurrence property, the emptiness problem becomes decidable [Tra11]. Last, in their work on randomized monitors, Rohit Chadha, A. Prasad Sistla and Mahesh Viswanathan exhibited subclasses of PBA with decidable emptiness problem [CSV09a, CSV09b].

4.1.2 Almost sure semantics

Considering the threshold 1, and thus the set of words such that almost all runs are accepted is not very interesting for PFA since it is equivalent to asking whether some word has all its computations accepted, which is sometimes referred to as the non emptiness of the universal language, and is PSPACE-complete. For PBA, since we consider infinite runs, probability 1 does not coincide with universal acceptance. We therefore define now the almost-sure semantics.

Definition 4.4 *Let $\mathcal{B}$ be a* PBA*. Its* language under the almost sure semantics *is defined as*

$$\mathcal{L}_{=1}(\mathcal{B}) = \{w \in \Sigma^\omega \mid \mathbb{P}(w) = 1\}.$$

We illustrate the definition of PBA language under the almost sure semantics on the example probabilistic Büchi automata from Figure 4.1, page 47. The PBA on the left accepts b^*a^ω, whereas the one on the right recognizes $(ab)^\omega$. In the two cases, these languages differ with both the language accepted by the underlying NBA, and by the PBA under positive semantics.

Language dependency on the precise probabilities

Similarly to PBA under positive semantics, the language defined by PBA under almost sure semantics depends on the precise probability distributions.

Proposition 4.2 *There exists a family of* PBA $(\mathcal{B}'_\lambda)_{\lambda\in(0,1)}$ *parameterised by $\lambda \in (0,1)$ such that for $\mu \neq \lambda$, $\mathcal{L}_{=1}(\mathcal{B}'_\mu) \neq \mathcal{L}_{=1}(\mathcal{B}'_\lambda)$.*

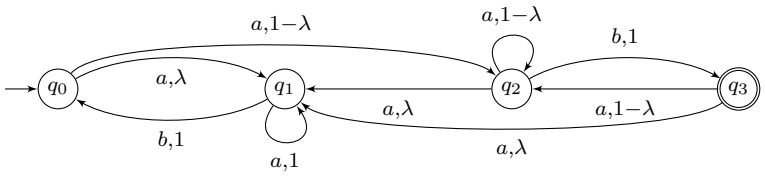

Figure 4.4: PBA $\mathcal{B}'_\lambda$ with $\lambda \in (0,1)$.

Figure 4.4 depicts such a family $(\mathcal{B}'_\lambda)_{\lambda\in(0,1)}$. Let us show that the language accepted by $\mathcal{B}'_\lambda$ under the almost sure semantics is

$$\mathcal{L}_{=1}(\mathcal{B}'_\lambda) = \{a^{k_1}ba^{k_2}b\cdots \mid k_i > 0 \text{ and } \prod_{i>0}(1-(1-\lambda)^{k_i}) = 0\}.$$

Indeed, starting in q_0 (or in q_3), $(1-\lambda)^{k_i}$ is the probability to be in q_2 after reading a^{k_i}, so that the complement probability $1-(1-\lambda)^{k_i}$ is the probability to be in state q_1 after the same factor a^{k_i}. The product $\prod_{i>0}(1-(1-\lambda)^{k_i})$ thus represents the probability to avoid the final state q_3 forever. Intuitively any finite prefix would not change the limit behaviour of this product: $\prod_{i>0}(1-(1-\lambda)^{k_i}) = 0$ if and only if $\prod_{i>j}(1-(1-\lambda)^{k_i}) = 0$ for some $j > 0$. This allows us to prove that $\mathcal{L}_{=1}(\mathcal{B}'_\lambda) = \{a^{k_1}ba^{k_2}b\cdots \mid k_i > 0 \text{ and } \prod_{i>0}(1-(1-\lambda)^{k_i}) = 0\}$.

Therefore, the almost sure semantics of $\mathcal{B}'_\lambda$ is very much related to the positive semantics of $\mathcal{B}_\lambda$, where $\mathcal{B}_\lambda$ is the PBA from Figure 4.2, we presented earlier. More precisely, $\mathcal{L}_{=1}(\mathcal{B}'_\lambda) =$

$(a^+b)^\omega \setminus \mathcal{L}_{>0}(\mathcal{B}_{1-\lambda})$. From this equality and Proposition 4.1, we deduce that the probabilistic Büchi automata $(\mathcal{B}'_\lambda)_{\lambda\in(0,1)}$ indeed witness Proposition 4.2.

Note that the family $(\mathcal{B}'_\lambda)_{\lambda\in(0,1)}$ has the nice property that any input word is accepted either with probability 0, or with probability 1, similarly to the PRA and PSA we built in the proof of closure under complementation of the PBA for the positive semantics.

Non-closure under complementation

Similarly to ω-regular languages, and to languages of PBA under the positive semantics, one can easily show that PBA under the almost sure semantics are closed under union and intersection. Yet, they behave differently with respect to complementation.

Proposition 4.3 *Probabilistic Büchi automata (under the almost sure semantics) are not closed under complementation: there exists a* PBA $\mathcal{B}$*, such that no* PBA $\mathcal{B}'$ *satisfies* $\mathcal{L}_{=1}(\mathcal{B}') = \Sigma^\omega \setminus \mathcal{L}_{=1}(\mathcal{B})$.

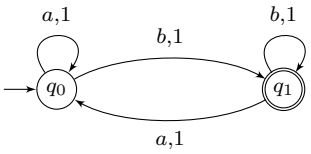

Figure 4.5: A PBA $\mathcal{B}$ such that there is no PBA $\mathcal{B}'$ with $\mathcal{L}_{=1}(\mathcal{B}') = (a+b)^\omega \setminus \mathcal{L}_{=1}(\mathcal{B})$.

To establish Proposition 4.3, consider the ω-regular language $(a^*b)^\omega$. It can be recognized by a deterministic Büchi automaton (DBA), and therefore by a PBA under the almost sure semantics. Indeed, any DBA can be turned into an equivalent PBA for the almost sure semantics by assigning probability 1 to every transition. For the sake of completeness, we represented the corresponding DBA/PBA in Figure 4.5. However, the complement of this language over alphabet $\{a, b\}$, that is $(a+b)^*a^\omega$, cannot be recognized by any PBA under almost sure semantics. Assume, towards a contradiction, that $\mathcal{B}'$ is a PBA such that $\mathcal{L}_{=1}(\mathcal{B}') = (a+b)^*a^\omega$. For every (reachable) state q in $\mathcal{B}'$, a^ω must be accepted with probability 1, so that we let n_q be the length of a path from q to a final state $f \in F$, and we further define $n = \max_{q\in Q} n_q$. Consider now the input word $w = (a^n b)^\omega \notin (a+b)^*a^\omega$ which is not accepted by $\mathcal{B}'$ under the almost sure semantics. By definition of n, from each state $q \in Q$, there is a positive probability $p_q > 0$ of visiting at least one accepting state when reading a^n. The positive uniform bound $p = \min_{q\in Q} p_q$ ensures that almost surely, reading $(a^n b)^\omega$ will imply visiting accepting states infinitely often. This contradicts the assumption that $\mathcal{L}_{=1}(\mathcal{B}') = (a+b)^*a^\omega$, and shows that the class of languages recognized by PBA under the almost sure semantics is not closed under complementation.

As a by-product, remark that the examples PBA from Figures 4.5 and 4.4 together show that the class of languages recognized by PBA under the almost sure semantics is uncomparable with the one of ω-regular languages.

Decidability of the emptiness problem

Another difference for PBA between the positive and the almost sure semantics is the decidability status for the emptiness problem.

Theorem 4.3 *The emptiness problem is decidable for probabilistic Büchi automata (under the almost sure semantics).*

Theorem 4.3 even holds in the more general framework of POMDP, and we stated in [BBG12] the decidability of the following problem: given a POMDP and a subset F of states, does there exists a pure observation-based strategy achieving the Büchi objective $\Box\Diamond F$ almost surely? Unfortunately, the construction in the proof of Theorem 8.5 in [BBG12] is erroneous, as noticed independently by Serge Haddad [Had13] and Tali Sznajder [Szn13]. To fix the proof, we give here an EXPTIME algorithm based on the belief construction, for POMDP with possibly randomized strategies. Using then the result of [CDGH10] that pure strategies are sufficient in POMDP, we will obtain the desired result for POMDP, and thus the decidabilty of the emptiness problem for PBA under the almost sure semantics.

Given a POMDP $\mathcal{M} = (Q, q_0, \mathcal{O}, \mathsf{Obs}, \mathsf{Act}, \Delta)$, we define its *belief automaton*. A *belief* B for $\mathcal{M}$ is a non empty subset of states included in some observation $\mathsf{o} \in \mathcal{O}(Q)$. We write Bel for the set of all beliefs, and we define the deterministic belief automaton $\mathcal{M}_{\mathsf{Bel}} = (\mathsf{Bel}, \{q_0\}, \mathsf{Act}\times\mathcal{O}, \delta)$ such that: for $B \in \mathsf{Bel}$, $\alpha \in \mathsf{Act}$ and $\mathsf{o} \in \mathcal{O}(Q)$, $\delta(B, (\alpha, \mathsf{o})) = \bigcup_{q\in B} \mathsf{Supp}(\Delta(q, \alpha)) \cap \mathsf{o}$. In words, $\delta(B, (\alpha, \mathsf{o}))$ updates the set of possible states the system is in, given the action that has been triggered and the observation that was made. For a sequence of actions and observations $(\alpha_1, \mathsf{o}_1)\cdots(\alpha_n, \mathsf{o}_n)$, we write $\delta(B, (\alpha_1, \mathsf{o}_1)\cdots(\alpha_n, \mathsf{o}_n))$ for $\delta(\cdots\delta(B, (\alpha_1, \mathsf{o}_1)), \cdots), (\alpha_n, \mathsf{o}_n))$.

Building on the belief automaton, the set Win of beliefs from which there exists a winning strategy (to ensure the Büchi condition almost surely), can be computed as a greatest fixpoint. Let $\mathsf{Bel}_F = \{B \in \mathsf{Bel} \mid B \subseteq F\}$. Then, Win is the limit of the non-increasing sequence that starts with $\mathsf{Win}_0 = \mathsf{Bel}$ and is defined inductively by:

$$\mathsf{Win}_{n+1} = \{B \in \mathsf{Win}_n \mid \exists(\alpha_1, \mathsf{o}_{i_1})\cdots(\alpha_n, \mathsf{o}_{i_n}),\ \delta\big(B, (\alpha_1, \mathsf{o}_{i_1})\cdots(\alpha_n, \mathsf{o}_{i_n})\big) \in \mathsf{Bel}_F$$
$$\wedge\ \forall k,\ \forall \mathsf{o}_{j_k},\ \delta\big(B, (\alpha_1, \mathsf{o}_1)\cdots(\alpha_k, \mathsf{o}_{j_k})\big) \neq \emptyset \Rightarrow \delta\big(B, (\alpha_1, \mathsf{o}_1)\cdots(\alpha_k, \mathsf{o}_{x_k})\big) \in \mathsf{Win}_n\}.$$

Then, there exists a strategy to ensure the Büchi objective $\Box\Diamond F$ with probability 1, if and only if $\{q_0\} \in \mathsf{Win}$. The fixpoint Win can be computed in polynomial time in the size of the belief automaton, that is exponential in the original POMDP, thus yielding an EXPTIME algorithm. Our proof shows the decidability in EXPTIME of the emptiness problem for PBA under the almost sure semantics, and we conjectured that the problem was EXPTIME-complete in [BBG08]. However, Rohit Chadha, A. Prasad Sistla and Mahesh Viswanathan established later the problem to be PSPACE-complete, and they also proved that, surprisingly, the language inclusion for PBA under the almost sure semantics is undecidable [CSV09b].

The decidability of the emptiness problem for PBA under the almost sure semantics might be surprising at a first glance, since we explained earlier that the language of a PBA (under the almost sure semantics) depends on the exact probability distributions. However, and contrary to PBA under the positive semantics, whether the language is empty or not does not depend on these precise probabilities. This fact can be seen as a consequence of the decidability proof for almost sure reachability in POMDP that provides a graph-based algorithm and thus does not consider the values of the probabilities, but only whether they are zero or not.

As we shall see in Section 4.3, we later generalized the above decision algorithm to 2-player stochastic games under partial observation [BGG09].

4.2 Minimal disclosure in POMDP

As mentioned earlier, POMDP generalize probabilistic automata, in the sense that the strategy or controller may access partial information about the system's state. This model thus somehow lies between (fully observable) Markov decision processes and (blind) probabilistic automata. In a joint work with Blaise Genest, we studied a model of POMDP in which the observation given to the player is not rigidly fixed, with the objective to select the optimal observation scheme [BG11]. Motivations for this problem find their roots in practical applications. For example, train traffic management relies on signalling equipment and their positioning must be designed to optimize a threshold between operational cost that should be low, and covering of tracks that should be sufficient to minimize risks of accidents. In other applications, the actual use of sensors might be costly in terms of energy so that optimizing the duration they are turned on while ensuring a given objective is natural. In order to keep the framework simple and general, we propose a POMDP model with only two possible information sets: by default, the controller accesses a fixed partial information, as in standard POMDP; and upon request it can also obtain full information about the system's state. In this framework, each execution is assigned a cost corresponding to the number of requests made by the controller. The problem we investigated is then to design a strategy for the controller that ensures a reachability objective almost surely, and minimizes the cost. Now, there are at least two natural ways to associate a cost with a strategy: either the worst case cost, or the average cost. In both cases, we are interested in computing the optimal cost among all possible strategies, and to synthesize optimal strategies that achieve this value.

We illustrate our developments on the simple POMDP example presented in Figure 4.6. The initial state is q_1, states q_1, q_2 and q_3 have the same observation, and the objective of the controller is to reach q_4 almost surely. We assume that the controller can perform a special **req**-action, whose effect is to disclose the precise state of the system. The set of possible observations in that example is therefore: $\mathcal{O} = \{\{q_1, q_2, q_3\}, \{q_4\}, \{q_5\}\} \cup \{\{q\} \mid q \in Q\}$. Without the request action, the controller has no way to almost surely visit q_4 on that example. Intuitively, after a finite sequence of a's, b and c are both bad options as they would yield a high losing probability. In the fully observable MDP however, the controller has a strategy to ensure visiting q_4 with probability 1: play a in q_1, b in q_2 and c in q_3.

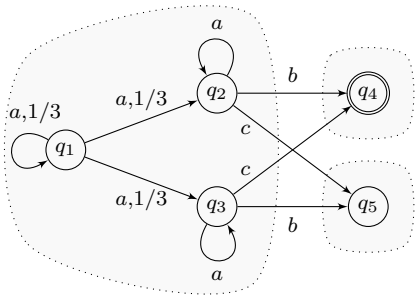

Figure 4.6: An example POMDP.

4.2.1 Worst-case cost

Theorem 4.4 *The optimal worst-case cost can be computed in* EXPTIME*, together with a finite-memory optimal strategy.*

In order to compute the worst-case cost, and an associated optimal strategy, we rely on the belief automaton, see definition on page 54. Recall that the belief automaton $\mathcal{M}_{\mathsf{Bel}}$ of a POMDP $\mathcal{M}$ is an exponential size finite automaton. Its states are subsets of states of $\mathcal{M}$ included in an observation class, and transitions are labelled with actions of $\mathcal{M}$. There is a transition from B to B' labelled by a if starting with belief B and performing action a, some observation yields the new belief B'. For our example POMDP of Figure 4.6 the reachable part of the belief automaton from initial belief $\{q_1\}$ is depicted in Figure 4.7.

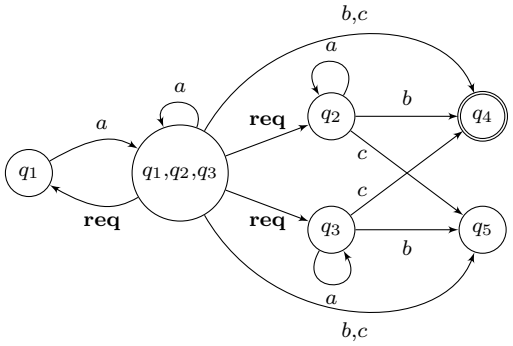

Figure 4.7: Belief automaton for the POMDP of Figure 4.6.

Using the belief automaton, one can iteratively compute the set of beliefs from which the optimal worst-case cost is k. Let us explain the computation on the example. Belief $\{q_4\}$ is clearly winning, with worst-case cost 0, since all its states (here only q_4) are included in the target set. Singleton beliefs $\{q_2\}$ and $\{q_3\}$ also have worst-case cost 0, since from each of them, some action (a and b, respectively) surely leads to a worst-case cost 0 belief, namely $\{q_4\}$. From belief $\{q_1, q_2, q_3\}$ however, actions b and c are risky as they may lead to a losing belief q_5. Action a is safe, but does not help progressing towards the target q_4. The special request action **req** permits to reach a set of beliefs, two of them with worst-case cost 0, and the last one $\{q_1\}$ with undefined worst-case cost. Therefore, $\{q_1, q_2, q_3\}$ and $\{q_1\}$ are beliefs with worst-case cost ∞.

In the general case, in order to compute the set of beliefs Win_{k+1} from which $k+1$ requests are needed and sufficient, one first computes $W_{\mathbf{req},\leq k}$ the set of beliefs for which every singleton belongs to $\bigcup_{j<k+1} \mathsf{Win}_j$, the union of all previously computed sets of beliefs for smaller values of worst-case cost. It then suffices to compute the attractor of that set in the standard POMDP, that is the set of beliefs from which almost surely $W_{\mathbf{req},\leq k}$ can be reached without performing any **req**-action. Last, Win_∞ gathers all beliefs that do not belong to any Win_j for $j \in \mathbb{N}$. So defined, the sets Win_k are optimal in the following sense.

Lemma 4.1 *For any strategy σ reaching the target almost surely*

- *for every belief $B \in \mathsf{Win}_{k+1} \setminus \mathsf{Win}_k$, there is a σ-path from an initial distribution with support B that contains at least $k+1$ **req**-actions;*

- *for every belief $B \in \mathsf{Win}_\infty$ and every constant $N \in \mathbb{N}$ there is a σ-path from an initial distribution with support B that contains at least N **req**-actions.*

The above partition of beliefs leads us to the definition of a family of canonical strategies for the worst-case cost, parameterised by n. When the belief belongs to $W_{\mathbf{req},\leq k}$ for some k, the controller must play **req** in order to decrease the current worst-case cost. From a belief in W_{k+1} the controller plays standard actions so as to reach $W_{\mathbf{req},\leq k}$ almost surely. Last, from Win_∞ the controller plays **req** with probability $1/n$ and the remaining probabilities to stay within Win_∞. On our example POMDP, the canonical randomized strategy σ^n for parameter value n is thus defined by $\sigma^n(\{q_1\}) = a$, $\sigma^n(\{q_2\}) = b$, $\sigma^n(\{q_3\}) = c$ and $\sigma^n(\{q_1, q_2, q_3\}) = 1/n\ \mathbf{req} + (1 - 1/n)\ a$.

4.2.2 Average cost

On the example POMDP of Figure 4.6, the family of canonical strategies $(\sigma^n)_{n>0}$ we defined happens to be optimal as well for the average-cost. Let us detail this in order to emphasize that computing the average cost in a POMDP under fixed strategy is not easy, since it manipulates the set of stochastic belief states (precise distributions over states in the discrete belief states) which is potentially infinite.

The game starts in state q_1, and the first decision of σ^n is to play a. After this action, the belief state is $\{q_1, q_2, q_3\}$ with uniform probabilities over the three states. In the sequel, we denote by E_k the expected number of requests under σ^n from $\{q_1, q_2, q_3\}$ when the probability is $1/3^k$ to be in state q_1, and equally distributed between q_2 and q_3 for the remaining probability. Our objective is thus to compute E_1, which is equal to the optimal average cost from $\{q_1\}$. Assume now that the current belief state is $\{q_1, q_2, q_3\}$ and the probability to be in state q_1 is $1/3^k$. With probability $1/n$, a request is performed, which discloses state q_1 with probability $1/3^k$; with probability $(n-1)/n$, a is played, and the resulting state is $\{q_1, q_2, q_3\}$ with probability mass $1/3^{k+1}$ in state q_1. As a consequence, $E_k = \frac{1}{n}\big(1 + \frac{1}{3^k}E_1\big) + \frac{n-1}{n}E_{k+1}$. From there, we derive: $E_1 = 1 + \frac{1}{2n}$. Hence, the average number of **req** actions performed by σ^n is smaller than $1 + \frac{1}{2n}$. In fact, we can prove that this family of strategies is optimal in the sense that no almost surely winning strategy can achieve an average number of request of 1 or less on this particular example.

However, the optimality result for both costs in this simple example is far from being generally true, and the situation for average cost is quite different than the one for worst-case cost. Indeed, first of all, we establish that computing optimal average cost for a POMDP under a reachability objective is unfeasible. Even worse, it is undecidable to even approximate it, whatever the approximation factor. Last, we give non approximability factors exponential in the size of the system, and prove that –assuming PTIME $\neq$ NP– there is no PTIME algorithm (in the number of belief states) to approximate the optimal average cost within that factor.

Theorem 4.5 *Let $K > 0$. The problem whether, given a POMDP $\mathcal{M}$ and a set F of its states, the optimal average cost in $\mathcal{M}$ to almost surely reach F is smaller than K, is undecidable.*

Moreover, the optimal average cost cannot be approximated (neither for approximation factor, nor for absolute approximation error) by a PTIME algorithm.

The impossibility of optimal average cost computation is not very surprising given that quantitative questions on partially observable probabilistic models, such as emptiness of

threshold languages for PFA, or optimizing cost functions for POMDP are known to be undecidable. We briefly sketch how one can encode a variant of the emptiness problem for probabilistic finite automata, which is known to be undecidable. Take a PFA $\mathcal{A}$ and a threshold $\varepsilon \in (0, 1/2)$ such that either there exists a word accepted with probability at least $1 - \varepsilon$ or all words are accepted with probability less than ε. By [MHC03], it is undecidable to know which case holds. From $\mathcal{A}$, we build a POMDP $\mathcal{M}$, as illustrated on Figure 4.8. The

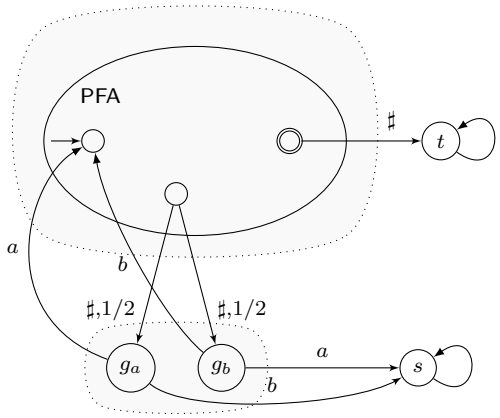

Figure 4.8: Reduction for the undecidability of the average cost evaluation.

resulting POMDP contains four additional states, including the target t and a gadget ensuring that whenever the set $\{g_a, g_b\}$ is reached via a $\sharp$ action, the player must play **req** not to risk to move to the sink s (from which t is no longer reachable). This reduction ensures the following: $\mathcal{A}$ accepts a word with probability greater than $1 - \varepsilon$ if and only if the optimal average cost in $\mathcal{M}$ to reach almost surely t is less than $\frac{\varepsilon}{1-\varepsilon}$. Indeed, assuming $\mathcal{A}$ accepts a word w with probability at least $1 - \varepsilon$, the strategy playing iteratively $w\sharp$, and then, if t has not been reached, **req** followed by the appropriate guess a or b, achieves a cost of at most $\sum_{n\in\mathbb{N}} n\varepsilon^n(1-\varepsilon) = \varepsilon/(1-\varepsilon)$. Now, if all words are accepted with probability less than ε, any strategy has cost at least $\sum_{n\in\mathbb{N}} n(1-\varepsilon)^n\varepsilon = (1-\varepsilon)/\varepsilon$, the latter being greater than $\varepsilon/(1-\varepsilon)$ since $\varepsilon \in (0, 1/2)$.

For the inapproximability result, we show the following: assuming PTIME $\neq$ NP, no PTIME algorithm can approximate the optimal average cost within any polynomial factor in the number of belief states of the POMDP. The proof is by reduction from the NP-complete problem 3-SAT. From a SAT φ formula with m clauses and k variables, we derive a POMDP $\mathcal{M}$ such that: if φ is satisfiable then the optimal average cost in $\mathcal{M}$ is less than $3n2^{-n}$, where $n = mk$, otherwise, it is larger than $1/n - 2^{-n}$. More details can be found in [BG11]. We emphasize that compared to other results on non approximability of optimal cost in POMDP [LGM01], our reduction does not rely on the (at worse exponentially many) discrete belief states to encode the problem, but uses the actual probability to be in a state.

4.3 Stochastic games with signals

To generalize the model of POMDP presented in the previous sections, with Blaise Genest and Hugo Gimbert, we considered in [BGG09] a standard model from game theory for 2-player stochastic games with partial observation, namely stochastic games with signals [Sor02,

RSV03, Ren07]. When playing a stochastic game with signals, players cannot observe the actual state of the game, nor the actions played by themselves or their opponent: their only source of information are private signals they receive throughout the play. Stochastic games with signals in particular subsume concurrent games [dAH00] and deterministic games with imperfect information on one side [Rei84, CDHR07]. Formally:

Definition 4.5 *A* 2-player stochastic game with signals *is a tuple* $(Q, q_0, I, J, C, D, \Delta)$ *where* Q *is a finite set of states with* $q_0 \in Q$ *an initial state,* I *(resp.* J*) is the set of actions for Player 1 (resp. for Player 2),* C *(resp.* D*) is the set of signals for Player 1 (resp. for Player 2), and* $\Delta : Q \times I \times J \to \mathsf{Dist}(Q \times C \times D)$ *is the transition function.*

A 2-player stochastic game with signals behaves as follows: at each step $n \in \mathbb{N}$, each player chooses an action $i_n \in I$ and $j_n \in J$. They respectively receive signals $c_n \in C$ and $d_n \in D$, and the game moves to a new state q_{n+1}. This happens with probability $\Delta(q_n, i_n, j_n)(q_{n+1}, c_n, d_n)$ given by the transition function Δ. A *play* is then a finite or infinite sequence $(q_0, i_0, j_0, c_1, d_1, q_1, \cdots, c_n, d_n, q_n \cdots)$ such that $\Delta(q_m, i_m, j_m)(q_{m+1}, c_{m+1}, d_{m+1}) > 0$ for every $0 \leq m$.

Players make their decisions based upon the sequence of signals they receive: a strategy is a mapping from finite sequences of private signals to probability distributions over actions. Formally, a (behavioral) *strategy* of Player 1 is a mapping $\sigma : C^* \to \mathsf{Dist}(I)$. If Player 1 has seen signals $c_1, \ldots, c_n$ then he plays action i with probability $\sigma(c_1, \ldots, c_n)(i)$. Strategies for Player 2 are defined symmetrically, and denoted τ.

Note that the choice of a strategy determines which lotteries over actions are played, and players do know the lotteries they choose, but we emphasize that we do not assume that players can observe the actions they have actually played. This contrasts with the model of strategies used in most other theoretical computer science papers about stochastic games with partial information [CDHR07, GS09, CD12, CDH13].

A 2-player stochastic game with signals and a strategy profile (σ, τ), that is a strategy for each player, induces a probability measure $\mathbb{P}^{\sigma,\tau}$ over the set of infinite plays.

We considered subsets of plays that are commonly used for applications in logic and controller synthesis (expressed by reachability, safety, Büchi, and co-Büchi constraints) in combination with a qualitative winning condition (positively, or almost surely). Given a subset Win of plays, Player 1 aims at maximizing the probability of winning plays, whereas Player 2 has the opposite goal. A strategy σ for Player 1 is said *almost surely winning* if for every strategy τ for Player 2, $\mathbb{P}^{\sigma,\tau}(\mathsf{Win}) = 1$. A less favorable situation is when σ is only *positively winning*, that is, for every strategy τ for Player 2, $\mathbb{P}^{\sigma,\tau}(\mathsf{Win}) > 0$.

It is important to notice that, whether σ is almost surely or positively winning from some distribution only depends on its support, not on the precise distribution.

Figure 4.9 provides an example of 1-player stochastic game with signals. Here, $I = \{a, g_1, g_2\}$ and $C = \{\alpha, \beta, \bot\}$. The labels on transitions should be understood as follows: transition $(q_1, (a, \alpha, 1/2), q_1)$ means that, when the game is in state q_1, and action a is played, with probability $1/2$, the player receives signal α and the game stays in state q_1. Actions g_1 and g_2 are guesses: when playing g_i from state j, the player wins if and only if $i = j$. The $\star$-symbol stands for any action. We assume that initially, the game is in states q_1 or q_2 with equal probability. In this 1-player game, there exists an almost surely winning strategy to reach state t. The player should play a until it receives a signal α or β, which almost surely

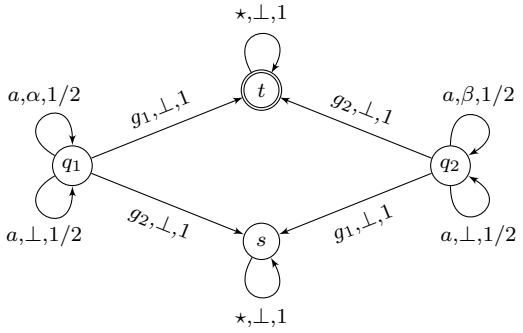

Figure 4.9: A 1-player stochastic game with signals.

happens. Then, it can guess at no risk whether the game started in state q_1 or q_2, and reach the target state t.

Figure 4.10 shows an example of 2-player stochastic game with signals. Again, transitions are labelled with triplets: the first component corresponds to the actions chosen by each player, the second component describes the signals they receive, and the last component is the probability of that particular transition. The game starts in state q_0, and the players get

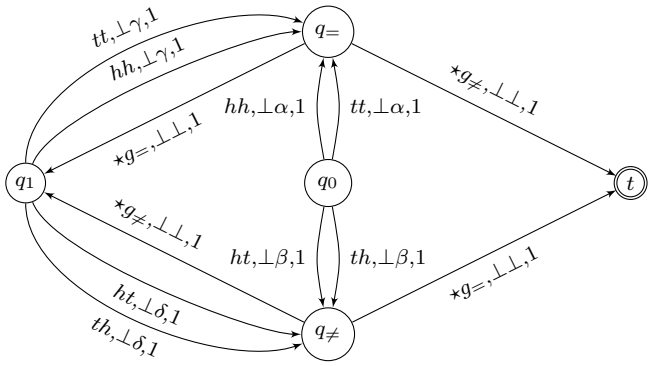

Figure 4.10: An example of 2-player stochastic game with signals.

to choose either heads (h) or tails (t). If their choices agree, the game moves to state $q_=$, otherwise to state $q_\neq$. The behaviour is similar from state q_1, expect that the signals received by Player 2 differ a priori. Player 1 is blind (the signals it receives bear no information) and can only count the number of steps. The objective for Player 1 is to reach the target state t, and it succeeds if Player 2 makes a wrong guess: either it plays $g_\neq$ from state $q_=$ or it plays $g_=$ from state $q_\neq$. Depending on the set of signals received by Player 2, the game will be almost surely winning, positively winning, or winning with probability zero for Player 1. We now examine the existence of almost-sure winning or positive winning strategies in this game, depending on the signals α, β, γ and δ.

Assume first that all signals α, β, γ and δ are parwise distinct. Then, Player 2 always knows when the play enters states $q_=$ and $q_\neq$ and can play accordingly, in order to avoid t. Therefore Player 2 has a surely winning strategy for its safety objective, and Player 1 wins with probability 0.

Assume now that $\alpha = \beta$, but γ and δ are distinct. Informally, after the first move, Player 2 cannot distinguish if the play is in state $q_=$ or $q_\neq$. The best choice is then to play uniformly at random $g_=$ and $g_\neq$. Later, if the game reaches state q_1, since $\gamma \neq \delta$, Player 2 will be able to avoid t, whatever Player 1 does. For both players, in the first move, the best choice is to play uniformly at random heads or tails, so that in this case, Player 1 wins positively, more precisely with probability 1/2.
Last, assume that $\alpha = \beta$ and $\gamma = \delta$, so that Player 2 can never distinguish between states $q_=$ or $q_\neq$. The best strategy for Player 1 is to always choose uniformly at random heads or tails. Under this purely random strategy, and whatever Player 2 does, every other move, the probability is half to move to the target state t, so that Player 1 wins almost surely.

4.3.1 Qualitative determinacy

Qualitative determinacy can be seen as a non-quantitative version of value determinacy and is quite appropriate for stochastic games with qualitative winning conditions. If an initial state is positively winning for Player 1 then by definition it is not almost surely winning for its opponent. A natural question is whether the converse implication holds.

Definition 4.6 (Qualitative determinacy) *A winning condition* Win *is* qualitatively determined *if for every stochastic game with signals equipped with* Win*, every initial state is either almost surely winning for Player 1 or positively winning for Player 2.*

Qualitative determinacy is similar to but different from the usual notion of determinacy which refers to the existence of a value. Actually both qualitative determinacy and value determinacy are formally expressed by a quantifier inversion. On one hand, qualitative determinacy rewrites as:

$$(\forall\sigma\ \exists\tau\ \mathbb{P}^{\sigma,\tau}(\mathsf{Win}) < 1) \implies (\exists\tau\ \forall\sigma\ \mathbb{P}^{\sigma,\tau}(\mathsf{Win}) < 1)\,.$$

On the other hand, the game has a value if:

$$\sup_\sigma \inf_\tau \mathbb{P}^{\sigma,\tau}(\mathsf{Win}) \geq \inf_\tau \sup_\sigma \mathbb{P}^{\sigma,\tau}(\mathsf{Win}).$$

Both the converse implication of the first equation and the converse inequality of the second equation are obvious.

The existence of an almost surely winning strategy ensures that the value of the game is 1, but the converse is not true. Actually it can even hold that Player 2 has a positively winning strategy while at the same time the value of the game is 1.

One of the major results of the paper [BGG09] is the qualitative determinacy of stochastic games with signals. Recall that we always give the winning condition in terms of objective for the first player: *e.g.* in a reachability game, the objective of Player 1 is to reach a target set whereas Player 2 has the opposite objective of avoiding this set.

Theorem 4.6 *Reachability, safety and Büchi winning conditions are qualitatively determined in 2-player stochastic games with signals.*

In order to prove this determinacy result, we characterize the set of winning belief states using fixpoint expressions, and show that on the complement set the adversary has a winning strategy. More details on these fixpoints computations are given in Section 4.3.2.

Since reachability and safety games are dual, a consequence of Theorem 4.6, is that in a reachability game, every initial state is either almost surely winning for Player 1, almost surely winning for Player 2, or positively winning for both players. If Player 2 wins almost surely a reachability game, it trivially implies that its safety condition is satisfied by all consistent plays, in other words Player 2 wins surely.

Büchi games do not share this nice feature because co-Büchi games are not qualitatively determined. A counter-example is represented on Figure 4.11. Similar examples can be used to prove that stochastic Büchi games with signals do not have a value. In this game, Player 1 observes everything, Player 2 is blind, and Player 1's objective is to avoid the target state t from some moment on. The initial state is t. One can show that Player 1 does not have an

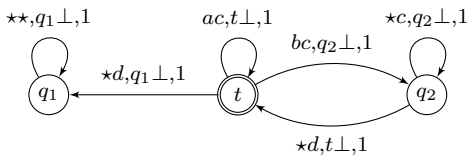

Figure 4.11: Co-Büchi winning conditions are not qualitatively determined.

almost surely winning strategy for the co-Büchi objective $\Diamond\Box(q_1 \vee q_2)$, neither does Player 2 have a positively winning strategy for the complement objective $\Box\Diamond t$.

First assume by contradiction that Player 1 has an almost surely winning strategy σ for the co-Büchi objective. To win against the strategy where Player 2 plays consistently c, almost surely, strategy σ must play action b eventually so that the play is not stuck in state t. Since σ is fixed, for any arbitrarily high probability $p < 1$, there exists a number of rounds n after which Player 1 has played b at least once with probability greater than p. Recall that Player 2 is blind, and consider its counting strategy which plays c for the first $n-1$ actions, and then plays d. After $n-1$ steps, with probability greater than p, the game is in state q_2, thus playing d puts the game back in the initial state t. Combining such strategies of Player 2 for greater and greater values of p yields a counterstrategy to σ. Precisely, given $\varepsilon > 0$, define the strategy τ for Player 2 that plays c until the probability under σ to be in state q_2 is $1 - \varepsilon/2$, and then plays d; and in the second round, assuming q_1 was not reached so far, τ plays c until the probability to be in state q_2 is $1 - \varepsilon/4$, and plays d, etc. In the i-th round, Player 2 plays c to reach state q_2 with probability at least $1 - \varepsilon/2^i$ and then plays d. The probability under strategy profile (σ, τ) to visit infinitely often state t is bounded by below by $\prod_{i=1}^{\infty}(1 - \varepsilon/2^i)$, a positive value. This contradicts that σ is almost surely winning.

Now assume by contradiction that Player 2 has a positively winning strategy τ for its Büchi objective $\Box\Diamond t$. Since Player 2 is blind, the probability under strategy τ that some action is played at a given step is independent of the strategy chosen by Player 1. Consider first the case that under τ, almost surely action d occurs at least once. In this case, the strategy for Player 1 that always plays a ensures to reach the sink state q_1 with probability 1, so that τ is not positively winning. Consider now that $p < 1$ is the probability that d is eventually played under strategy τ. In this case, for every precision $\varepsilon > 0$, there exists a number of steps n such that the probability under τ that d was played in the first n steps or will never be played is greater than $1 - \varepsilon$. We define the strategy σ for Player 1 that plays a for n steps and then plays b consistently. Under the strategy profile (σ, τ), the probability that t is never visited after the n first steps is greater than $1 - \varepsilon$. In the unlikely option that d is played

again after these n steps (which happens with probability less than ε), Player 1 notices it by the signal it receives and decides to iterate its counterstrategy, that is, play a for n steps and then play b. This way, the probability to stay in states q_1 and q_2 after $2n$ steps is greater than $(1-\varepsilon)^2$. Iterating this reasoning, one obtains a strategy σ for Player 1 such that under strategy profile (σ, τ) the probability to visit t infinitely often is 0. This contradicts that τ is positively winning.

4.3.2 Complexity and memory bounds

Positively winning a reachability game

Proposition 4.4 *In a stochastic game with signals and reachability winning condition, the partition of beliefs into positively winning ones for Player 1 and surely winning ones for Player 2 can be computed in* EXPTIME *in the size of the arena.*
The algorithm computes at the same time finite-memory winning strategies for each player.

The set of supports $\mathcal{L} \subseteq 2^Q$ that are surely winning for Player 2 is characterized as the largest fixpoint of some monotonic operator $\Phi : 2^{2^Q} \to 2^{2^Q}$. Operator Φ associates with $\mathcal{L} \subseteq 2^Q$ the set of supports $L \in \mathcal{L}$ that do not intersect target states and from which Player 2 has an action that ensures its next belief to be in $\mathcal{L}$ as well, whatever the action chosen by Player 1 and the signal received by Player 2. For $\mathcal{L} \subseteq 2^Q$, the value of $\Phi(\mathcal{L})$ is computable in time linear in $\mathcal{L}$ and in the description of the game, yielding the exponential complexity bound.

Almost surely winning a Büchi game

To decide whether Player 1 wins almost surely a Büchi game, we provide an algorithm which runs in doubly-exponential time and uses the algorithm of Proposition 4.4 as sub-procedure.

Theorem 4.7 *In a stochastic game with signals and Büchi winning condition, the partition of beliefs into almost surely winning ones for Player 1 and positively winning ones for Player 2 can be computed in 2-*EXPTIME *in the size of the arena.*
The algorithm computes at the same time finite-memory winning strategies for each player.

The proof of Theorem 4.7 is based on the following ideas.
First, suppose that from every initial support Player 1 can win the reachability objective with positive probability. Then, repeating the same strategy, Player 1 can guarantee the Büchi condition to hold with probability 1. Otherwise, according to the determinacy of reachability games (see Theorem 4.6), there would exist a support L that is surely winning for Player 2 for the complementary co-Büchi objective.
In fact, in case Player 2 can force the belief of Player 1 to be L someday with positive probability from another support L', then L' is positively winning as well for Player 2. This is not completely obvious because in general Player 2 cannot know exactly when the belief of Player 1 is L. For winning positively from L', Player 2 plays totally randomly until it guesses randomly that the belief of Player 1 is L, and at that moment switches to a strategy surely winning from L. Such a strategy is far from being optimal, because Player 2 plays randomly, and in most cases will make a wrong guess about the belief of Player 1. However it suffices for Player 2 to win positively.

As a consequence, Player 1 should surely avoid its belief to be L or L' if it wants to win almost surely. However, doing so Player 1 may prevent the play from reaching target states, which may create another positively winning support for Player 2, etc...
Building on these observations, we define the set $\mathcal{L}_\infty \subseteq 2^Q$ of supports almost surely winning for Player 1 for the Büchi objective as the largest set of initial supports from which Player 1 has a strategy for winning positively the reachability game while ensuring at the same time its belief to stay in $\mathcal{L}_\infty$. This property can be reformulated as a reachability condition in a new game whose states are states of the original game augmented with beliefs of Player 1, kept hidden to Player 2.
The fixpoint characterization suggests the following algorithm to compute the set of supports positively winning for Player 2: $2^Q \setminus \mathcal{L}_\infty$ is the limit of the sequence $\emptyset = \mathcal{L}'_0 \subsetneq \mathcal{L}'_0 \cup \mathcal{L}''_1 \subsetneq \mathcal{L}'_0 \cup \mathcal{L}'_1 \subsetneq \mathcal{L}'_0 \cup \mathcal{L}'_1 \cup \mathcal{L}''_2 \subsetneq \ldots \subsetneq \mathcal{L}'_0 \cup \cdots \cup \mathcal{L}'_m = 2^Q \setminus \mathcal{L}_\infty$, where

(a) from supports in $\mathcal{L}''_{i+1}$ Player 2 can surely guarantee the safety objective, under the hypothesis that Player 1's beliefs stay outside $\mathcal{L}'_i$,

(b) from supports in $\mathcal{L}'_{i+1}$ Player 2 can ensure with positive probability the belief of Player 1 to eventually be in $\mathcal{L}''_{i+1}$, under the same hypothesis.

The overall strategy of Player 2 that is positively winning for the co-Büchi objective consists in playing randomly for some time until it decides to pick up randomly a belief L of Player 1 in $\mathcal{L}''_i$ for some i. It then forgets the signals it has received so far and switches definitively to a strategy which guarantees (a). With positive probability, Player 2 is lucky enough to guess correctly the belief of Player 1 at the right moment, and future beliefs of Player 1 will stay in $\mathcal{L}'_i$, in which case the co-Büchi condition holds.

Memory requirements

To end this section on stochastic games with signals, we report on results on the memory sufficient and needed for the players to win from their winning regions.

Theorem 4.8
- *In case Player 1 has a positively winning strategy for a reachability game, then playing randomly any action is also positively winning.*
  Otherwise, Player 2 has a surely winning belief-based strategy.

- *In case Player 1 has an almost surely winning strategy for a Büchi game, then it has an almost surely winning belief-based strategy.*
  Otherwise, Player 2 has a positively winning strategy with finite memory $2^{2^Q \times Q}$.

A surprising fact in 2-player stochastic games with signals is that doubly exponential memory is necessary to win positively a safety game. As briefly explained above, Player 2 uses beliefs on beliefs of Player 1. We showed that this amount of memory was indeed needed (see [BGG09] for details):

Proposition 4.5 *There exists a family of arenas of size n with reachability winning condition such that Player 2 needs doubly exponential memory in n to win positively.*

4.4 Diagnosis of probabilistic systems

We now move to a theoretical problem with more practical roots, namely the diagnosis of faults for partially observable probabilistic models. Within computer science, diagnosis may refer to different kinds of activities. For instance, in artificial intelligence it can describe the process of identifying a desease from its symptoms, as performed by the expert system MYCIN [BS84]. In control theory, diagnosis applies to partially observable systems prone to faults, and intuitively the task of diagnosis is to detect the occurrence of faults. A sequence of observations of a partially observable system is said to be surely correct (respectively surely faulty) if all possible runs corresponding to this sequence are correct (respectively faulty); otherwise the observed sequence is ambiguous. While monitoring the system, the *diagnoser* should rule out ambiguities, and in particular detect that a fault occurred; and the problem of existence of such a diagnoser is referred to as *diagnosability* [SSL+95]. Diagnosability was first defined and studied in the framework of finite discrete event systems modelled by labelled transition systems, and the problem was shown to be solvable in PTIME (see [JHCK01]). Despite this polynomial time complexity, for diagnosable systems, the size of the diagnoser may be exponential [HHMS13]. Diagnosers must satisfy two requirements: *correctness*, meaning that the information provided by the diagnoser/predictor is accurate, and *reactivity*, ensuring that a fault will eventually be detected.

Building on the work for non-probabilistic systems, the notion of diagnosability was later extended to Markov chains with labels on transitions, also called probabilitic labelled transition systems [TT05]. In a probabilistic context, the reactivity requirement now asks that faults will be *almost surely* eventually detected. Regarding correctness, two specifications have been proposed: either one sticks to the original definition and requires that the provided information is accurate; or one weakens the correctness by admitting errors in the provided information that should, however, have an arbitrary small probability when the delay before the diagnostic is long enough. From a computational viewpoint, PTIME algorithms have been proposed to solve these two specifications of probabilistic diagnosability [CK13]. Predictability with arbitrary small probability of erroneous information has also been studied in [CK14].

In the recent work [BHL14], we revisited the diagnosability problem for probabilistic systems. With the objective to come up with a thorough semantical classification, we defined four notions of diagnosability, depending on two criteria: first whether the information provided by the diagnoser is related to faulty runs only or to all runs, and second whether the ambiguity is defined for infinite runs or for their longer and longer finite prefixes. All details can be found in [BHL14] and its companion research report. In this document, we concentrate on a single notion of diagnosability, refered to as almost sure diagnosability, which requires that the set of sequences whose observation is ambiguous has null measure.

The model we consider is the one of probabilistic labelled transition system, or equivalently Markov chains with actions labels on transitions.

Definition 4.7 *A* probabilistic labelled transition system *is a tuple* $\mathcal{A} = \langle Q, q_0, \Sigma, \mathbf{P} \rangle$ *where:*

- Q *is an at most countable set of states with* $q_0 \in Q$ *the initial state;*
- Σ *is a finite set of events;*
- $\mathbf{P} : Q \times \Sigma \times Q \to \mathbb{Q}_{\geq 0}$ *is the probabilistic transition function satisfying for every* $q \in Q$, $\sum_{a \in \Sigma} \sum_{q' \in Q} \mathbf{P}(q, a, q') = 1$.

A run in $\mathcal{A} = \langle Q, q_0, \Sigma, \mathbf{P}\rangle$ is an infinite sequence of transitions $q_0 a_0 q_1 a_1 \cdots$ starting from the initial state and such that $\mathbf{P}(q_i, a_i, q_{i+1}) > 0$ for every index $i \in \mathbb{N}$. Given a probabilistic labelled transition system $\mathcal{A}$, we write $\mathcal{L}^\omega(\mathcal{A})$ for the set of infinite sequences of events that label runs of $\mathcal{A}$. Since a pLTS $\mathcal{A}$ is a discrete-time Markov chain with labels on transitions, its set of infinite runs is naturally equipped with a probability measure, that we denote $\mathbb{P}$ (assuming $\mathcal{A}$ is clear from the context).

4.4.1 Solving almost sure diagnosability

Let us introduce the diagnostic problem in the context of probabilistic systems and explain how to solve it. We assume the alphabet Σ to be partitionned into observable and unobservable events Σ_o and Σ_u, with a particular event $\mathbf{f} \in \Sigma_u$, and denote by $\mathcal{P}$ the projection morphism from sequences in Σ^ω to observed sequences in Σ_o^ω.

Definition 4.8 (Ambiguity and almost sure diagnosability) *An observed sequence $\sigma \in \Sigma_o^\omega$ is* ambiguous *if $\mathcal{P}^{-1}(\sigma) \cap \mathcal{L}^\omega(\mathcal{A}) \cap (\Sigma \setminus \mathbf{f})^\omega \neq \emptyset$ and $\mathcal{P}^{-1}(\sigma) \cap \mathcal{L}^\omega(\mathcal{A}) \cap \Sigma^* \mathbf{f} \Sigma^\omega \neq \emptyset$.*

A probabilistic labelled transition system $\mathcal{A}$ is almost surely diagnosable *if*

$$\mathbb{P}(\{\rho \in \mathcal{L}^\omega(\mathcal{A}) \mid \rho \textit{ is ambiguous}\}) = 0.$$

Ambiguous observed sequences are thus sequences of observables that can be explained both by non-faulty sequences and by faulty ones. Figure 4.12 presents a probabilistic labelled

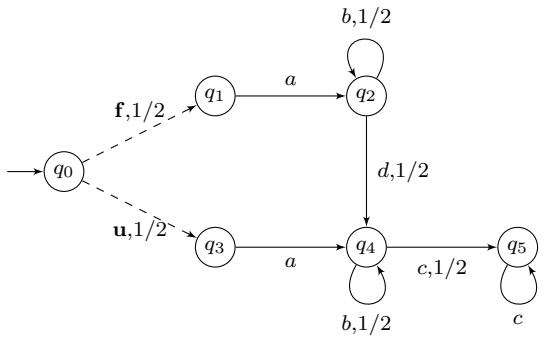

Figure 4.12: An almost-surely diagnosable labelled Markov chain.

transition system, that is a Markov chain with labels on transitions, in which both $\mathbf{u}$ and $\mathbf{f}$ are unobservable labels, as depicted by the dashed transitions. Its underlying non-probabilistic labelled transition system is not diagnosable. Indeed, ab^ω is an ambiguous observed sequence, as it corresponds to a faulty sequence $q_0 \xrightarrow{\mathbf{f}} q_1 \xrightarrow{a} q_2 \xrightarrow{b} q_2 \cdots$ as well as to a correct sequence $q_0 \xrightarrow{\mathbf{u}} q_3 \xrightarrow{a} q_4 \xrightarrow{b} q_4 \cdots$. Yet, this system is almost surely diagnosable, since to the unique ambiguous infinite observed sequence ab^ω corresponds a set of sequences of measure zero.

To simplify further developments, and without loss of generality, we assume the state spaces of the probabilistic labelled transition systems we consider are partitionned into correct states (reached only by correct sequences) and faulty states (reached only by faulty sequences). This is, for example the case of the probabilistic labelled transition system from Figure 4.13, for which q_0 and q_3 are correct states, whereas q_1 and q_2 are faulty states. The only ambiguous

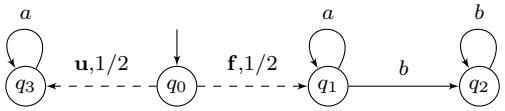

Figure 4.13: A labelled Markov chain which is not almost surely diagnosable.

infinite observed sequence is a^ω, and the runs that yield this observation are $q_0 \xrightarrow{\mathbf{f}} q_1 \xrightarrow{a} q_1 \cdots$ and $q_0 \xrightarrow{\mathbf{u}} q_3 \xrightarrow{a} q_3 \cdots$. They gather a probability of $1/2$, therefore, the system is not almost surely diagnosable. Yet, after a fault, almost surely a b-event will eventually happen, so that faults are almost surely diagnosed. We thus emphasize that almost sure diagnosability is more demanding than the almost sure detection of faults. This also motivates alternative definitions of the diagnosability in a probabilistic context. In this document we focus on almost sure diagnosability, but we performed a thorough study of other relevant semantical definitions of diagnosability for probabilistic systems [BHL14].

For finite state probabilistic systems, almost sure diagnosability (as well as the other variants) can be characterized based on deterministic (finite or Büchi) automata acting as monitors, and synchronized with the probabilistic system. These characterizations allowed us to establish the following complexity result, therefore contradicting the polynomial time algorithm from [CK13].

Proposition 4.6 *The almost sure diagnosability is* PSPACE*-complete.*

Let us illustrate the diagnoser construction, as well as the characterisation of almost sure diagnosability on the example from Figure 4.12. Its diagnoser is the deterministic Büchi automaton over Σ_o introduced in [HHMS13] whose states are triples of disjoint subsets of states (U, V, W) where given some observed sequence, U is the set of possible correct states and V and W are possible faulty states. The decomposition between V and W reflects the fact that the diagnoser tries to resolve the ambiguity between U and W (when both are non empty), while V corresponds to a waiting room of states reached by faulty runs that will be examined when the current ambiguity is resolved. In this Büchi automaton, the set F of accepting states consists of all triples (U, V, W) with $U = \emptyset$ or $W = \emptyset$. When $U = \emptyset$, the runs corresponding to the observation so far are surely faulty. When $W = \emptyset$ the current run may still be ambiguous (if $V \neq \emptyset$) but the "oldest" possible faulty runs have been discarded. Hence, any infinite observed sequence of $\mathcal{A}$ passing infinitely often through F is not ambiguous (ambiguities are resolved one after another).

Figure 4.14 shows the diagnoser of the probabilistic labelled transition system depicted on Figure 4.12. Observe that the unique ambiguous observed sequence ab^ω is not accepted by this deterministic Büchi automaton, which recognizes unambiguous sequences.

To illustrate further the construction, and in particular the utility of triplets (U, V, W), we provide another example of pLTS and associated diagnoser on Figure 4.15. Observe that, despite the fact that all observed sequences a^n are ambiguous as witnessed by the possible faulty state f_2, a^ω, which is indeed unambiguous, is accepted by the diagnoser since its execution infinitely often visits state $(\{q_1, q_2\}, \{f_2\}, \emptyset)$.

Building on the diagnoser that accepts exactly the unambiguous sequences, almost sure diagnosability can be decided on the product of the pLTS with its diagnoser. Since the diagnoser is deterministic, this product is still a pLTS, and we proved the following characterization:

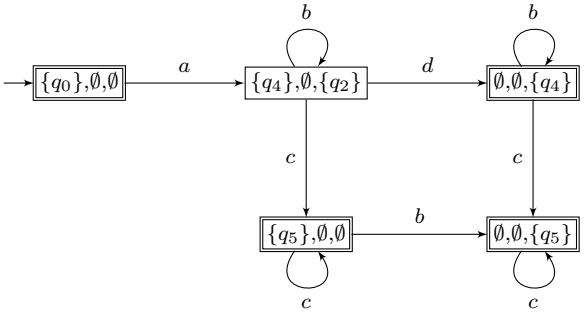

Figure 4.14: The Büchi automaton serving as diagnoser for the pLTS depicted on Figure 4.12.

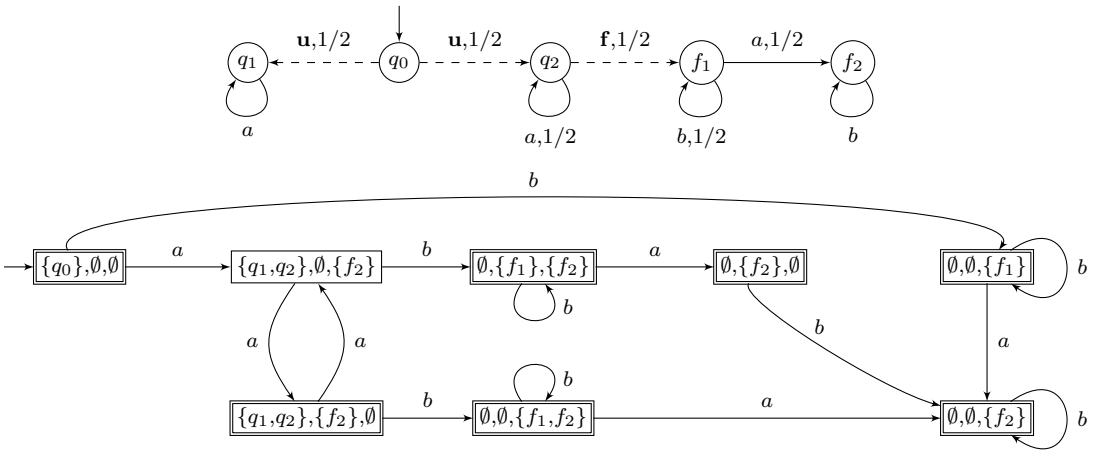

Figure 4.15: A pLTS and its deterministic Büchi automaton serving as diagnoser.

Lemma 4.2 *A pLTS $\mathcal{A}$ is almost surely diagnosable if and only if, the product of $\mathcal{A}$ with its diagnoser has no bottom strongly connected component such that*

- *either all its states (q, U, V, W) satisfy q is faulty and $U \neq \emptyset$*
- *or all its states (q, U, V, W) satisfy q is correct and $W \neq \emptyset$.*

Using Lemma 4.2, we can prove the decidability in PSPACE of the almost sure diagnosability. Reducing a variant of universality, we also established the PSPACE-hardness of that decision problem, thus showing Proposition 4.6.

Our definition of almost sure diagnosability is semantical: it gives a condition on the set of ambiguous observed sequences. In contrast, in the discrete event system community, diagnosability is most often defined from a monitoring view-point as the existence of a mapping associating with each observed sequence a verdict that expresses whether the observed sequence is surely correct, surely faulty, or ambiguous. We formalized the notion of almost sure diagnoser as follows. A diagnoser is a function $D : \Sigma_o^* \to \{?, \top, \bot\}$ assigning to every finite observation sequence a verdict. Informally when a diagnoser outputs ? it does not provide any information, while $\top$ ensures that a fault is certain and $\bot$ means that some information about correctness has been provided. We consider the natural partial order $\prec$ on these values defined by $? \prec \top$ and $? \prec \bot$. Moreover, we restrict to diagnosers that, once they output $\top$, never change their verdict in the future, which is natural since faults are permanent (there are no repair events). In the following definition, for $w \in \Sigma_o^*$ an observed sequence and $n \leq |w|$, $w_{\leq n}$ is the prefix of w of length n; also, for ρ a run, $|\rho|_{\downarrow k}$ denotes the prefix of ρ up to the k-th observable event.

Definition 4.9 *An almost sure diagnoser for $\mathcal{A}$ is a function $D : \Sigma_o^* \to \{\top, \bot, ?\}$ such that*

soundness *For all $w \in \Sigma_o^*$*

- *if $D(w) = \top$, then w is surely faulty;*
- *if $D(w) = \bot$, letting $|D(w)|_\bot = |\{0 < n \leq |w| \mid D(w_{\leq n}) = \bot\}|$, then for every run ρ such that $\mathcal{P}(\rho) = w$, $\rho_{\downarrow |D(w)|_\bot}$ is correct.*

reactivity $\mathbb{P}(\{\rho \in \mathsf{Runs} \mid D_{\sup}(\mathcal{P}(\rho)) = ?\}) = 0$ *where for $w \in \Sigma_o^\omega$, $D_{\sup}(w) = \limsup_{n\to\infty} D(w_{\leq n})$.*

Soundness ensures that the information provided is accurate and reactivity specifies which pieces of information the diagnoser must provide. Intuitively, almost sure diagnosers may resolve an ambiguity late, while another one has already been produced. For all the variants of diagnosability for probabilistic systems we studied, we also provided corresponding notions of diagnosers, by adapting the soundness and reactivity conditions.

With the above definition of almost sure diagnosers, one can show the equivalence of almost sure diagnosability and the existence of an almost sure diagnoser. Moreover, based on the deterministic automaton that characterizes the set of unambiguous observed sequences, a diagnoser with exponential memory (in the size of the input probabilistic system) can be synthesized. Notice that the exponential bound is optimal, as shown by the following family of almost surely diagnosable labelled Markov chains, which require diagnosers of exponential size. The probability distributions are not explicited and are assumed to be uniform. Intuitively, the diagnoser must remember a window of n events in case a c event occurs, to be able to tell whether the sequence is faulty or correct.

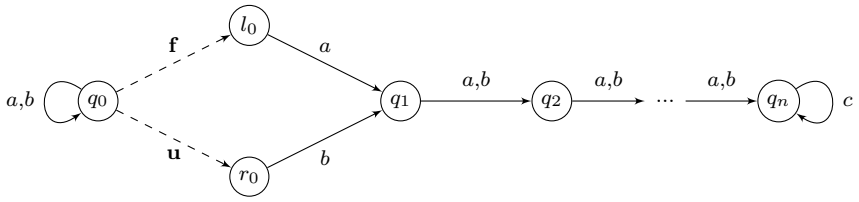

Figure 4.16: Diagnosable labelled Markov chain requiring exponential-memory diagnoser.

4.4.2 From diagnosis to active diagnosis.

In case a system is not diagnosable, one may be able to control it, by forbidding some controllable actions, so that it becomes diagnosable. Pursuing the work of [SLT98, HHMS13] for discrete-event systems, we studied in [BFH+14] the active diagnosability problem for probabilistic systems. We considered decidability and complexity issues, and tackled the problem of synthesizing optimal size diagnosers.

In order to specify a control problem, in the spirit of Ramadge and Wonham [RW87], and orthogonally to the observable/unobservable duality, the event alphabet Σ is partitionned into controllable and uncontrollable events. As the terminology suggests, controllable actions can be disabled by the controller, whereas uncontrollable ones cannot. Based on its observation, that is the observed sequence so far, the controller decides upon the set of enabled controllable actions, keeping in mind that uncontrollable events may also happen. The active diagnosis problem for probabilistic systems amounts to designing such a controller that enforces the almost sure diagnosability of the system, while preserving its liveness, that is, without introducing deadlocks.

Starting from a pLTS describing the system, the active diagnosability problem thus takes the form of a 1-player stochastic game under partial observation. This motivates the use of partially observable Markov decision processes to formalize the semantics of the controllable models we consider. Because controllers will call for a normalization of the probability distributions over enabled events, we preferred to start with weighted transition systems rather than probabilistic transition systems. A *controllable weighted labelled transition system* (cLTS) is a tuple $\mathcal{C} = \langle Q, q_0, \Sigma, \Delta \rangle$ where the event alphabet Σ is partitionned into observable Σ_o and unobservable Σ_u events, and also partitionned into controllable Σ_c and uncontrollable Σ_e (e for environment) events; $\Sigma_u = \{\mathbf{f}, \mathbf{u}\}$ contains a faulty event, and a non-faulty one; and $\Delta : S \times \Sigma \times S \to \mathbb{N}$ is the transition function, labelling transitions with integer weights. We assume the cLTS to be live: in every state at least one event is enabled.

Figure 4.17 represents an example of controllable LTS, in which observable events are also controllable ones. For simplicity we assume the weights to be all equal to 1. Note that, seen as a probabilistic labelled transition system (with uniform probability distributions), this example cLTS is not almost surely diagnosable. Indeed, the observed sequence $aadcb^\omega$ is ambiguous (it corresponds to a faulty sequence and a correct sequence that have respectively for prefix $q_0 \xrightarrow{\mathbf{f}} q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_2$ and $q_0 \xrightarrow{\mathbf{u}} q_3 \xrightarrow{a} q_4 \xrightarrow{a} q_2$) and its set of corresponding runs has non null probability.

Now, let us present strategies for the controller and their induced pLTS. A *strategy* for a cLTS $\mathcal{C}$ is a mapping $\sigma : \Sigma_o^* \to \mathsf{Dist}(2^\Sigma)$ such that for every observed sequence $w \in \Sigma_o^*$, for every $\Sigma' \in \mathsf{Supp}(\sigma(w))$, $\Sigma' \supseteq \Sigma_e$. A strategy consists in, given some observation, randomly choosing

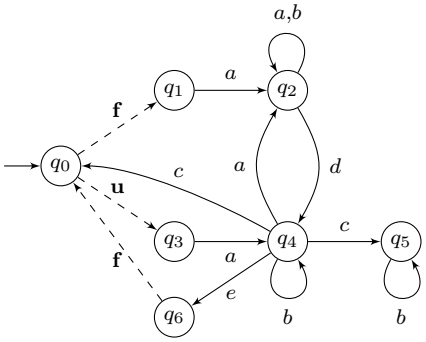

Figure 4.17: Example of a controllable labelled transition system.

a subset of allowed events that must include the uncontrollable events. Given a cLTS $\mathcal{C}$, a strategy σ yields a probabilistic labelled transition system obtained by unfolding the cLTS, enabling only events as dictated by the strategy, and normalizing in each state the weigths of enabled events into a probability distribution. We denote by $\mathcal{C}_\sigma$ the (possibly infinite state) pLTS obtained from cLTS $\mathcal{C}$ and strategy σ.

Intuitively, for the cLTS depicted on Figure 4.17, disabling the a-loop at q_2 rules out the ambiguous infinite observed sequence mentionned above $aadcb^\omega$. More precisely, the strategy will forbid controllable event a to occur after the observations of the form ab^*. The only remaining ambiguous observed sequence is ab^ω, but this observation happens with probability 0. Therefore defining σ as $\sigma(ab^*) = \Sigma \setminus \{a\}$ and $\sigma(w) = \Sigma$ for any other observation $w \in \Sigma_o^*$, yields an almost surely diagnosable and live pLTS.

Definition 4.10 *The* active probabilistic diagnosability problem *asks, given $\mathcal{C}$ a cLTS, whether there exists a strategy σ such that $\mathcal{C}_\sigma$ is almost surely diagnosable and live.*
If such a strategy exists, the cLTS $\mathcal{C}$ is said to be actively diagnosable.

To solve the active probabilistic diagnosability problem, we reduce it to a decidable problem for POMDP, namely, the existence of a strategy ensuring a Büchi objective almost surely. To do so, we need to rephrase the semantics of a cLTS $\mathcal{C}$ into a POMDP $\mathcal{M}$, and face several issues. The first difficulty is harmless and can be tackled by a traditionnal shift from transitions to states: the observations in cLTS are related to transitions, whereas they label states of the POMDP. Second, in the POMDP we derive, the information about ambiguity of runs must be incorporated into the states. Similarly to what we did for almost sure diagnosability in the previous section, we reuse the deterministic Büchi automaton that characterizes the infinite unambiguous observed sequences from [HHMS13]. As a consequence, the states of the POMDP comprise of a state q of the cLTS, together with a state l of the deterministic Büchi automaton. Actions in POMDP $\mathcal{M}_\mathcal{C}$ are subsets of events $\Sigma_e \subseteq \Sigma'$ specifying which controlled event is allowed. Given some control Σ', to define the transition probability in $\mathcal{M}_\mathcal{C}$ from (l, q) to (l', q'), one must consider all paths in $\mathcal{C}$ labelled by events of Σ' from q to q' that end with an observable event b. The probability of any such path is obtained by the product of the individual steps probabilities. The latter are then defined by the normalization of weights with respect to Σ'. The POMDP $\mathcal{M}_\mathcal{C}$ is exponential in the size of $\mathcal{C}$ and can be constructed in exponential time. All details of this construction can be found in [BFH+14].

A direct consequence of the definition of POMDP $\mathcal{M}_{\mathcal{C}}$ from $\mathcal{C}$ is the following, where we use LTL-notations on the components of the Büchi automaton.

Proposition 4.7 *$\mathcal{C}$ is actively diagnosable if and only if there exists a strategy π in $\mathcal{M}_{\mathcal{C}}$ such that $\mathbb{P}_{\pi}(\mathcal{M}_{\mathcal{C}} \models \Box\Diamond(W = \emptyset \vee U = \emptyset)) = 1$.*

As announced earlier, the active diagnosis problem for controllable labelled transition systems thus reduces to the existence of an almost sure winning strategy for a Büchi objective on some exponential size POMDP. Recall that the latter question for POMDP is decidable (see Theorem 4.3, page 54), so that we obtain the decidability of the active almost sure diagnosis problem:

Theorem 4.9 *The active almost sure diagnosis problem is* EXPTIME*-complete.*

The EXPTIME upper bound may seem surprising, since $\mathcal{M}_{\mathcal{C}}$ is exponential in the size of $\mathcal{C}$, and the procedure to decide whether there exists a strategy in a POMDP to ensure a Büchi objective with probability 1 is in EXPTIME, due to the use of beliefs. However, in the POMDP $\mathcal{M}_{\mathcal{C}}$ we consider, the information on the belief is already contained in the state $((U, V, W), q)$, as $U \cup V \cup W$. Therefore, the second exponential blowup due to the beliefs is avoided and the active probabilistic diagnosis problem remains in EXPTIME. For the lower-bound, we adapt the EXPTIME-hardness proof for active diagnosis of non-probabilistic systems [HHMS13] that reduces from safety games with imperfect information [BD08].

Beyond showing that the active probabilistic diagnosis problem is EXPTIME-complete, we can also prove that strategies with exponentially many memory states are needed.

Proposition 4.8 *There exists a family $(\mathcal{C}_n)_{n\geq 1}$ of actively diagnosable cLTS such that $|\mathcal{C}_n| = \mathcal{O}(n)$ such that any winning strategy has at least 2^n different memory states.*

Such a family is represented in Figure 4.18. Intuitively, in order to obtain a diagnosable pLTS,

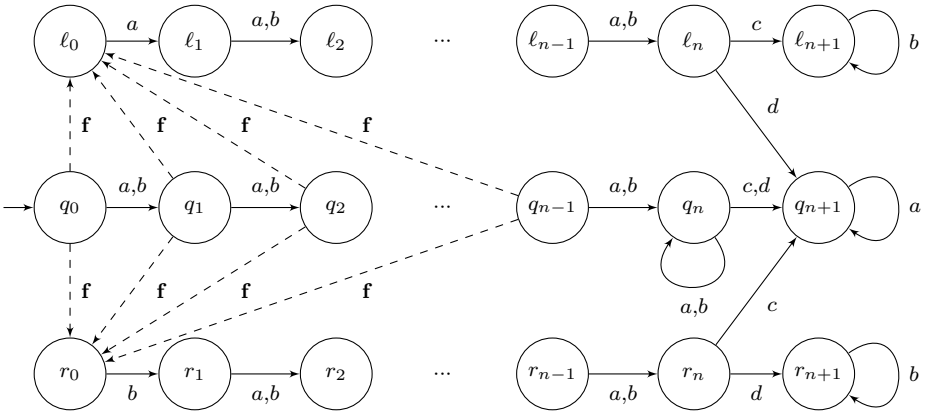

Figure 4.18: A cLTS $\mathcal{C}_n$ with $\Sigma_o = \{a, b, c, d\}$, $\Sigma_c = \{c, d\}$ and weights are all equal.

the controller should disable c or d, depending on the letter it observed n steps before. Indeed, if that letter was an a, the system can now be in state ℓ_n, after a faulty sequence, or in state q_n

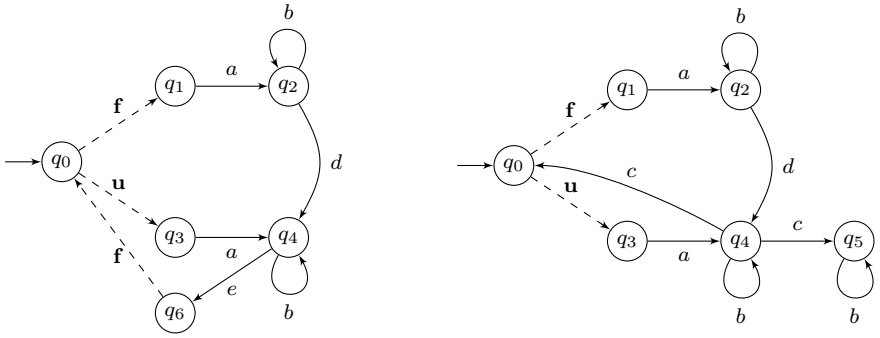

Figure 4.19: Two ways to control the cLTS from Figure 4.17 into a diagnosable pLTS.

after a correct one. To be able to distinguish the two cases, the controller should thus forbid d, so that in the next step either b (from ℓ_{n+1}) or a (from q_{n+1}) is observed. The situation is symmetrical if the letter observed n-steps before was b, and the controller should disable c. This explains that the controller needs exponential memory to remember a window of the last n observed events. The formal proof that this family $\mathcal{C}_n$ of cLTS needs exponential memory controllers is an easy adaptation of the same result from [HHMS13] in a non-probabilistic context.

4.4.3 Safe active diagnosis

Forcing a system to commit a fault is definitely a way to ensure that it can be diagnosed. Yet, everyone should agree that it is not a desirable way to attain diagnosability. In order to avoid fault provocative controllers, we refine the active diagnosis problem by imposing a positive probability to correct runs.

Definition 4.11 *The* safe active probabilistic diagnosability problem *asks, given $\mathcal{C}$ a cLTS, whether there exists a strategy σ such that $\mathcal{C}_\sigma$ is almost surely diagnosable, live, and $\mathbb{P}_\sigma(\{\rho \in \mathcal{L}^\omega(\mathcal{C}_\sigma) \mid \rho \text{ is correct}\}) > 0$.*
If such a strategy exists, the cLTS $\mathcal{C}$ is said to be safely actively diagnosable.

Back to the example of Figure 4.17, if we define a strategy that always forbids c, and forbids a after any observation ending with ab^*, we obtain an almost surely diagnosable pLTS depicted on Figure 4.19. Yet, the set of correct runs in that labelled Markov chain (with uniform distributions) is negligible. However, another strategy can be defined that shows the system to be safely actively diagnosable. Indeed, defining the controller that always forbids e and forbids a after any observation ending with ab^* yields the pLTS represented right of Figure 4.19 which is almost surely diagnosable, and has a positive measure of correct runs.

Although safe active diagnosability seems a reasonable and natural extension of active diagnosability, they are quite different problems from a computability perspective. A first difference between the two problems is that, contrary to the active diagnosis, the safe active diagnosis may require infinite-memory strategies.

To warm up, let us first show that strategies based on beliefs only, that is strategies whose memory set is the set of beliefs, are not sufficient. Consider the cLTS represented in Figure 4.20 with $\Sigma_c = \{a, b\}$, and all weights are equal. The set of beliefs reachable from initial belief $\{q_0\}$

is exactly composed of $\{q_0\}$, $\{q_1, q_2\}$, $\{q_1, q_2, q_3\}$ and $\{q_4\}$. The real choices any belief-based strategy has, is when the belief is $B = \{q_1, q_2, q_3\}$, to allow or disallow a or b, or allow both with a fixed probability. In all situations, a simple calculation shows that the probability mass in q_1 and q_2 tends to 0 implying that a fault almost surely happens. However, a much simpler strategy, not based on beliefs, answers our problem: after the first observed a, disable a and b alternatively. This yields an almost surely diagnosable pLTS with half probability of correct runs. The above example shows that belief-based strategies are not sufficient to ensure a safe diagnosis. It is not surprising since already ensuring a safety objective with positive probability in POMDP requires non belief-based strategies. Indeed, the very same idea as in Figure 4.20 applies: a POMDP where the unique unsafe state is q_4, and states q_1, q_2 and q_3 are observationally equivalent gives an example for which non belief-based strategies are needed.

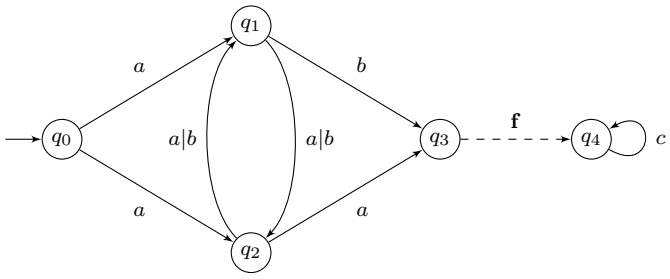

Figure 4.20: A cLTS with no belief-based strategy for safe probabilistic diagnosability.

For the safe active diagnosis problem however, the situation is even worse, and the example cLTS of Figure 4.21 shows that finite-memory strategies are not enough. Let us spend some

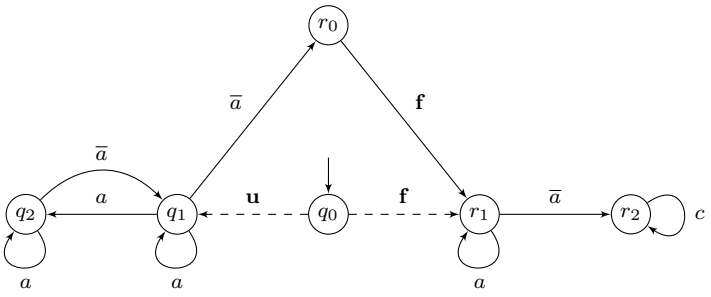

Figure 4.21: A cLTS requiring infinite-memory strategies for the safe diagnosability.

time to explain why this cLTS is actively safely diagnosable, but requires infinite-memory strategies to that aim.

To show that it admits an infinite-memory strategy ensuring the safe diagnosability, pick any sequence of positive integers $\{\alpha_i\}_{i\geq 1}$ such that $\prod_{i\geq 1} 1 - 2^{-\alpha_i} > 0$ and define the strategy σ that consists in selecting, after n observations, the n^{th} subset in the following sequence $A^{\alpha_1}\overline{A}A^{\alpha_2}\overline{A}\ldots$, where $A = \{a\} \cup \Sigma_e$ and $\overline{A} = \{\overline{a}\} \cup \Sigma_e$. We claim that strategy σ is winning, *i.e.* it yields an almost surely diagnosable pLTS, that has a positive probability of correct runs. Observe that after an observable sequence of length $i \leq \alpha_1$, the system is either after a faulty sequence in r_1 with probability $\frac{1}{2}$, or after a correct sequence in q_1 with probability 2^{-i-1}, or after a correct sequence in q_2 with probability $\frac{1}{2}(1 - 2^{-i})$. So, after an observable sequence of

length $\alpha_1 + 1$, the system is either after a faulty sequence in r_2 with probability $\frac{1}{2}$, or after a faulty sequence in r_1 (*via* r_0) with probability $2^{-\alpha_1-1}$, or after a correct sequence in q_1 with probability $\frac{1}{2}(1 - 2^{-\alpha_1})$. At the next step, the faulty sequence in r_2 is then detected by the occurrence of c. Iterating this reasoning we conclude that:

- any fault that may occur after σ is applied up to $A^{\alpha_1}\overline{A}A^{\alpha_2}\overline{A}\dots A^{\alpha_i}\overline{A}$, is detected after σ is applied up to $A^{\alpha_1}\overline{A}A^{\alpha_2}\overline{A}\dots A^{\alpha_{i+1}}\overline{A}A$. So strategy σ *surely* detects faults.

- the probability of infinite correct sequences under σ is equal to $\frac{1}{2}\prod_{i\geq 1} 1-2^{-\alpha_i}$, a positive value due to our choice of the α_i's.

Hence, the strategy we proposed indeed solves the safe active diagnosis.

Consider now any finite-memory strategy σ for the same cLTS, and let us show that σ cannot ensure the safe diagnosability. Without loss of generality, we assume that the memory states M of σ contain the belief information and that the belief part of the memory is updated as usual for beliefs. We emphasize that, in the context of cLTS, beliefs represent the possible states after the last observed event. For instance, when the belief is $\{q_0\}$, the current state may either be q_0, or q_1 after event $\mathbf{u}$, or r_1 after fault $\mathbf{f}$. For strategy σ, there are three possible subsets of allowed events: A, $\overline{A}$ and the whole alphabet Σ. The decision rule associated with the initial memory state m_0 (containing belief $\{q_0\}$), must allow a in order to get the possibility of a correct sequence. In case a occurs, this leads to a memory state m with belief-component $\{q_1, q_2, r_1\}$. Consider the (randomized) decision rule of σ associated with memory state m: $\vec{p}^m = p^m_A \cdot A + p^m_{\overline{A}} \cdot \overline{A} + p^m_\Sigma \cdot \Sigma$. Recall that the belief associated with m is $\{q_1, q_2, r_1\}$, meaning that with positive probability (when in r_1), a fault occured before the first occurrence of a. If $p^m_A = 1$, then this first fault remains undetected, and σ is losing because the diagnosability condition is not satisfied. So $p^m_a < 1$, and therefore, $\overline{a}$ may occur. If $\overline{a}$ occurs, this leads to a memory state m' with associated belief $\{q_1, r_0, r_2\}$. Let $\vec{p}^{m'} = p^{m'}_A \cdot A + p^{m'}_{\overline{A}} \cdot \overline{A} + p^{m'}_\Sigma \cdot \Sigma$ be the decision rule of σ associated with memory state m' (with belief $\{q_1, r_0, r_2\}$)). If $p^{m'}_{\overline{A}} = 1$, then at the next step, there is no possible correct sequence, and σ is losing, because the positivity of correct sequences is not ensured. Therefore, $p^{m'}_{\overline{A}} < 1$. The above reasoning is independent of the precise memory state, and applies to any memory states m, m' with respective beliefs $\{q_1, q_2, r_1\}$ and $\{q_1, r_0, r_2\}$. Hence, for any such memory states, $p^m_A < 1$ and $p^{m'}_{\overline{A}} < 1$. We let p_A and $p_{\overline{A}}$ be the maximum values of p^m_A and $p^{m'}_{\overline{A}}$ when m, m' range over possible memory states. Since M, the set of memory states of σ, is finite, $p_A < 1$ and $p_{\overline{A}} < 1$. Let us pick m' a memory state with belief $\{q_1, r_0, r_2\}$, and let us consider what happens from m' until the next occurrence of event $\overline{a}$. Assume that the current distribution over states is $\alpha q_1 + \beta r_0 + (1-\alpha-\beta)r_2$ (which is consistent with the belief of m'). The distribution after the next occurrence of $\overline{a}$ is defined by $\alpha_\sigma \alpha q_1 + (1-\alpha_\sigma)\alpha r_0 + (1-\alpha)r_2$, where α_σ only depends on the strategy σ. Moreover, from $p_A < 1$ and $p_{\overline{A}} < 1$, we can deduce that $\alpha_\sigma < 1$. To conclude, it remains to observe that any correct sequence contains an infinite number of $\overline{a}$. Since after n occurrences of $\overline{a}$, the probability of correct sequences is bounded by α^n_σ, the probability of all infinite correct sequences is null. Therefore σ is losing.

Not only finite-memory strategies are not enough for the safe active diagnosis problem, but the decision problem itself is not solvable.

Theorem 4.10 *The safe active almost sure diagnosis problem is undecidable.*

To prove the undecidability of the safe active diagnosability, we perform a reduction from the following undecidable problem: given a blind POMDP and a set F of states, does there exist a strategy that ensures the Büchi objective $\Box\Diamond F$ with positive probability. In Section 4.1, we explained that the latter problem is undecidable for probabilistic Büchi automata (therefore for non randomized strategies), and the undecidability also holds for POMDP thanks to the general result of [CDGH10]. The structure of the cLTS we construct is similar to the one of the example from Figure 4.21, except that the states q_1 and q_2 are replaced with two copies of the POMDP. Consistently a and $\overline{a}$ are replaced by two copies of the alphabet of the POMDP with one of them overlined. From F states in the first copy, with a non bared event one moves to the second copy, and from any state, with bared events, one moves back from the second copy to the first copy, or moves from the first copy to r_0. The reduction is represented on Figure 4.22. It ensures that there exists a blind strategy σ in $\mathcal{M}$ such that $\mathbb{P}_\sigma(\mathcal{M} \models \Box\Diamond F) > 0$

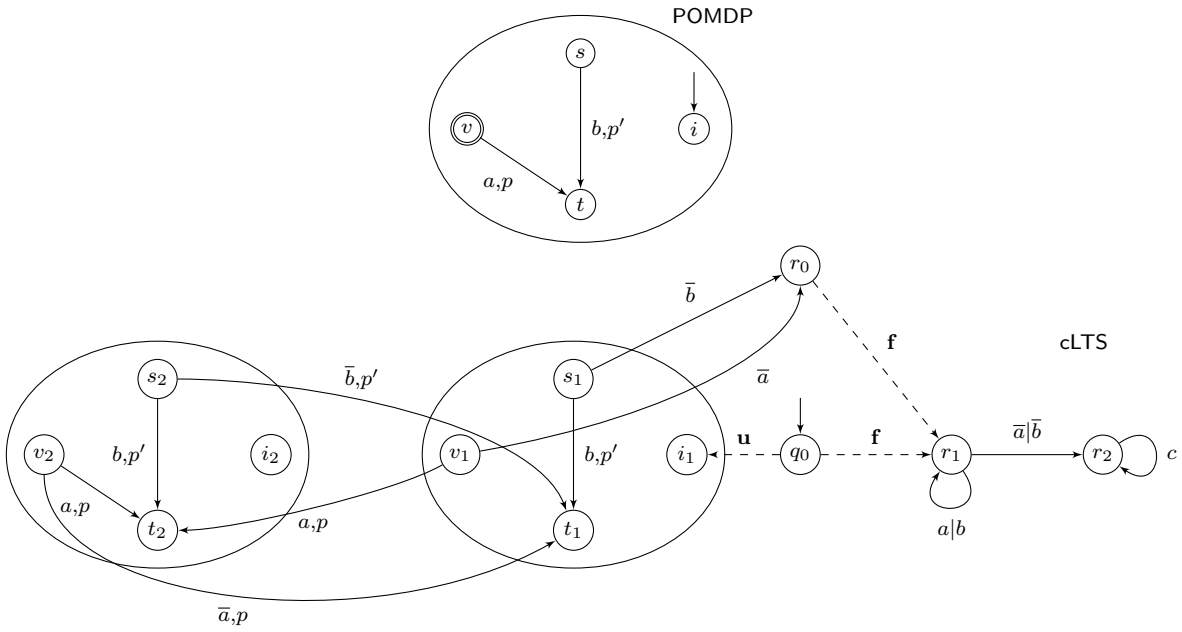

Figure 4.22: A POMDP $\mathcal{M}$ and the derived cLTS $\mathcal{C}$ for the undecidability proof.

if and only if the cLTS $\mathcal{C}$ is safely active diagnosable. The proof generalizes the ideas we gave for the example of Figure 4.20.

Safe active diagnosability of cLTS can be formalized as the existence of a strategy for a POMDP that ensures a Büchi goal almost surely, and at the same time a safety goal positively. As a corollary of Theorem 4.10, we obtain the undecidability of this more general problem for POMDP.

Corollary 4.1 *The problem, given a* POMDP $\mathcal{M}$ *with subsets of states* F *and* I*, of the existence of a strategy* σ *with* $\mathbb{P}_\sigma(\mathcal{M} \models \Box\Diamond F) = 1$ *and* $\mathbb{P}_\sigma(\mathcal{M} \models \Box I) > 0$*, is undecidable.*

This undecidability result for POMDP is interesting on its own. At first, it can seem surprising since both problems (Büchi almost surely and safety with positive probability), taken independently, are decidable for POMDP. Yet, combining the two, one can quite naturally encode a Büchi condition with positive probability.

One way to circumvent this impossibility result is to restrict to finite-memory strategies. Note that, similarly to any decision problem on games, the safe active diagnosis problem does not become trivially decidable when restricted to finite-memory strategies, since even if the memory is finite, it is not necessarily bounded.

Theorem 4.11 *The problem, given a* POMDP $\mathcal{M}$ *with subsets of states* F *and* I*, of the existence of a finite-memory strategy* σ *with* $\mathbb{P}_\sigma(\mathcal{M} \models \Box\Diamond F) = 1$ *and* $\mathbb{P}_\sigma(\mathcal{M} \models \Box I) > 0$*, is* EXPTIME*-complete.*

Clearly, belief-based strategies are not sufficient for the above problem, since they already do not suffice for positively winning in a POMDP with safety objective (see Figure 4.20). However, we can show that among finite-memory strategies, strategies with memory $2^Q \times Q$ are sufficient. An EXPTIME algorithm to compute the set of winning beliefs Win is the following. We use the belief automaton $\mathcal{M}_{\mathsf{Bel}} = (\mathsf{Bel}, \{q_0\}, \mathsf{Act} \times \mathcal{O}, \delta)$ to maintain candidate pairs of beliefs $(B', B) \in \mathsf{Bel}^2$ with $B' \subseteq B$, with the intuitive meaning that there exists a strategy ensuring from B (and therefore from B') to almost surely satisfy the Büchi objective, and such that from B' the safety objective is surely satisfied. Initially, $\mathsf{Win}'_0 = \{(B', B) \in \mathsf{Bel} \times \mathsf{Bel} \mid B' \subseteq B \text{ and } B' \subseteq I\}$. We also update the set of safe actions, initialized as $\mathsf{Act}_0((B', B)) = \{\alpha \mid \forall \mathsf{o} \in \mathcal{O},\ \delta(B, (\alpha, \mathsf{o})) \neq \emptyset \implies \delta(B', (\alpha, \mathsf{o})) \subseteq I\}$. The sequences $(\mathsf{Win}'_i)_{i\in\mathbb{N}}$ and $(\mathsf{Act}_i)_{i\in\mathbb{N}}$ are then iteratively computed by:

$$\mathsf{Win}'_{i+1} = \{(B', B) \in \mathsf{Win}'_i \mid \forall (C', C) \subseteq (B', B),\ \exists (\alpha_1, \mathsf{o}_1) \cdots (\alpha_n, \mathsf{o}_n) \in (\mathsf{Act}_i \times \mathcal{O})^n : \\ \delta((C', C), (\alpha_1, \mathsf{o}_1) \cdots (\alpha_n, \mathsf{o}_n)) \in \mathsf{Bel} \times \mathsf{Bel}_F\}$$

where $\mathsf{Bel}_F = \{B \in \mathsf{Bel} \mid B \subseteq F\}$.

$$\mathsf{Act}_{i+1}((B', B)) = \{\alpha \in \mathsf{Act}_i((B', B)) \mid \forall \mathsf{o} \in \mathcal{O}, \\ \delta(B, (\alpha, \mathsf{o})) \neq \emptyset \implies \delta((B', B), (\alpha, \mathsf{o})) \in \mathsf{Win}'_{i+1}\}.$$

In words, Win'_{i+1} is the set of pairs of beliefs such that there is a path from the largest belief to Bel_F, using only actions in Act_i, and Act_{i+1} is the set of actions that allow to stay in Win'_{i+1}.

The limit Win' of the non-increasing sequence $(\mathsf{Win}'_i)_{i\in\mathbb{N}}$ satisfies: for every $(B, B') \in \mathsf{Win}'$ there exists a finite memory strategy σ' such that $\mathbb{P}_{\sigma'}(B \models \Box\Diamond F) = 1$ and $\mathbb{P}_{\sigma'}(B' \models \Box I) = 1$.

Last, we define Win as the set of beliefs from which beliefs in the second component of Win' are almost surely reachable. More precisely

$$\mathsf{Win} = \{B_0 \in \mathsf{Bel} \mid \exists \sigma,\ \mathbb{P}_\sigma(B_0 \models \Diamond\{B \mid \exists B' \neq \emptyset :\ (B', B) \in \mathsf{Win}'\}) = 1\}.$$

Both Win' and Win can be computed in exponential time from the POMDP $\mathcal{M}$. To justify their definition, let us explain how to build a winning strategy for beliefs in Win. From $B_0 \in \mathsf{Win}$, the winning strategy σ first aims at reaching $\{B \mid \exists B' \neq \emptyset :\ (B', B) \in \mathsf{Win}'\}$, and this happens almost surely by definition of Win. Once such a belief B is reached, it guesses a non-empty belief $B' \subseteq B$ such that $(B', B) \in \mathsf{Win}'$. The player here bets that the system might be in B' and applies a finite memory strategy σ' that ensures visiting infinitely often F from B, and surely staying in I from B'. All in all, this provides a finite memory strategy that has two modes and ensures at the same time the Büchi condition with probability 1 and the safety condition positively.

Let us now come back to our safe active diagnosis problem. As we did for the active diagnosability problem, we can reduce the safe active diagnosability problem to a decidable problem for POMDP, using the deterministic Büchi automaton that characterizes the set of unambiguous observed sequences. Here again, in order to establish the EXPTIME complexity, and avoid a doubly exponential blowup, we observe that the beliefs and sub-beliefs that are computed during the resolution on the POMDP only concern the cLTS part of the product, and not the Büchi component. This implies that the safe active probabilistic diagnosis problem is in EXPTIME when restricted to finite memory strategies. The proof of the matching lower bound is similar to the one for active probabilistic diagnosability.

Corollary 4.2 *When restricted to finite-memory strategies, the safe active probabilistic diagnosis is* EXPTIME*-complete.*

4.5 Perspectives

Our contributions to the control of probabilistic systems under partial observation are varied in terms of models and problems, but mostly concern the qualitative analysis. This is natural since quantitative questions very often turn out to be undecidable for such systems, given the undecidability of the emptiness problem for threshold languages defined by probabilistic automata. However, we believe that, to some extent, it is worth investigating further quantative questions for probabilistic sytems under partial observation.

Quantifying diagnosis In particular, in the context of diagnosis, we envision the following developments. First, we will consider optimization questions, that one can classify into spatial and temporal optimization. A typical setting for spatial optimization consists in minimizing the use of sensors while maintaining diagnosability. This can be formalized by a flexible alphabet of observed events, and the controller may choose dynamically which events to observe with the objective to minimize the alphabet size. Increasing this observation cost can increase the probability of diagnosability, so that the controller needs to resolve a trade-off between these two measures. For what concerns temporal optimization, the controller should optimize when to observe an event, or when to observe the state of the system, while preserving the diagnosability of the system. As a first concrete problem, given a labelled Markov chain, one could be interested in minimizing the number of observations before a given state is observed. All these problems in which costs and diagnosability should be traded off, beyond being a natural problem for practitionners, would unify and further develop our contributions on minimal disclosure for POMDP on the one hand, and on qualitative diagnosability on the other hand.

Another research direction in the context of diagnosis is the definition of approximate notions of diagnosability for probabilistic systems. So far, we considered the set of ambiguous sequences, and asked that is has null probability. Thorsley and Teneketzis introduced a relaxed definition, called AA-diagnosability, in which they restrict to almost-surely ambiguous sequences. To the best of our knowledge, the exact complexity of the approximate diagnosis is still open, and even its decidability status. Contrary to the notions we studied so far, AA-diagnosability calls for more quantitative techniques since the diagnoser may have to perform a frequency analysis, and to compute likelihood ratios in order to output the correct verdict. Therefore, although the definition itself is still qualitative, AA-diagnosability strongly depends

on the values of probabilities in the model, so that the decision techniques need to be quantitative. This contrasts with most of the problems we tackled so far, to the notable exception of probabilistic Büchi automata, for which the precise values of probabilities matter. A general comment is that decidability results in the context of probabilistic systems under partial observation are obtained either relying on the fact that the precise values are not important, or because of an argument of bounded horizon. Going beyond this duality (or proving such a duality theorem) is a long term research objective.

Blindness and partial obsevation Speaking of general results, one observation is that, up to my knowledge, probabilistic automata and POMDP do not distinguish any decision problem in terms of decidability. For example, the existence of a strategy ensuring a Büchi objective with positive probability is undecidable for POMDP and the undecidability already holds for probabilistic automata: the existence of infinite word accepted with positive probability is undecidable. Conversely, the decidability result on the existence of an infinite word accepted almost surely transfers to POMDP: one can decide the existence of a strategy ensuring a Büchi objective almost surely. These examples seem to indicate that partial observation makes problems as easy (from a complexity viewpoint) than no observation at all. A natural question is whether this is a general fact, and it would be interesting to establish a meta theorem stating that problems decidable for blind POMDP are also decidable for POMDP in general. For the moment, one problem is proven decidable for probabilistic automata, and still open for POMDP, namely, considering as objective a refinement of Büchi condition in which the proportion of visits to final states (also called the frequency) has to be positive. The existence of an infinite word that yields a positive probability to positive frequency runs is decidable for probabilistic automata [Tra11]. Extending this decidability result to general POMDP (or proving its undecidability!) is on our research agenda.

Uniform control under complete information Controlling probabilistic systems under partial observation raises difficulties and yields to many negative results, in terms of decidability. To circumvent this unfeasability, the usual workarounds are either to restrict to subclasses of models, *e.g.* structurally defined, that are more tractable, or to focus on given properties for which efficient decision algorithms can be developed. Apart from these classical solutions, we also would like to study a new model that combines probabilities and a weak form of partial observation. Precisely, motivated by an application in systems biology, we wish to consider families of identical MDP that are controlled uniformely: the decisions made by the perfectly informed scheduler apply to all MDP simultaneously. When the population size tends to infinity, this model approximates the behaviour of the MDP under blind schedulers. However, for any fixed finite number of MDP, decision and even optimization problems can be trivially solved by building the product MDP. We believe that such approximate discrete models for probabilistic automata are worth being studied, and want to address parameterised qualitative and quantitative problems. Therefore, this research objective is also related to the contributions developed next, and will be further discussed at the end of Chapter 5.

Chapter 5

Parameterised probabilistic systems

Parameterised models are conviently used to represent systems in which some parameter is unknown. For example, when modeling an unreliable channel, it can be useful to use a parameter for the transmission failure rate. Also, when designing a scheduling algorithm, it might be that certain deadlines or execution times are not yet fixed. Last, when modeling networks, the number of components is most often variable, so that it is natural to abstract it by a parameter.

Parameterised verification then aims at verifying all instances of a parameterised model at once, without enumerating all possible values for the parameter. Very related, the goal of parameter synthesis, is to compute parameter values that ensure the model to satisfy a given specification.

In that direction, our work concentrated on parameterised models of networks, for which the parameter is the number of nodes (and possibly the communication topology). Following the line of work initiated by Giorgio Delzanno, Arnaud Sangnier and Gianluigi Zavattaro [DSZ10, DSZ11a, DSZ11b, DSZ12], we considered networks of arbitrarily many identical processes communicating by broadcast with their neighbours.

Quite often, distributed protocols use probabilities in their description. Randomization can be used to break symmetry, as in the dining philosophers problem; it is also very frequent in applications related to cryptography, *e.g.* for keys and nonces generation. The well established tools for probabilistic model checking (see *e.g.* Prism [KNPS08] and MRMC [KZH+09]) do not apply to parameterised systems, and even for fixed number of components, the state space explosion problem limits the automatic verification to networks of a handful of processes [Fru06]. Parameterised verification of probabilistic systems is thus a challenging research avenue.

Outline of the contributions Our contributions focused on networks of identical Markov decision processes. Together with Paulin Fournier we studied networks of arbitrary size, but fixed clique topology [BF13]. A natural question is then whether there exists a network size for which a given qualitative reachability property holds. Most of these problems happen to be undecidable. We also considered a dynamic variant of such networks in which the number of processes evolves over time, according to fixed disappearance and creation rates. In this case, we show all qualitative problems to be decidable using a well quasi order on network configurations and relying on a finite attractor property.

In another work with Paulin Fournier and Arnaud Sangnier [BFS14], we considered recon-

figurable networks – *i.e.* networks in which the communication topology changes nondeterministically– with a fixed number of processes. We showed the decidability of all variants of parameterised qualitative reachability questions via the analysis of parameterised networks of 2-player (non-stochastic) games.

5.1 Introducing probabilistic broadcast networks

Definition 5.1 *A* probabilistic broadcast network $\mathcal{P}^N$ *consists of* $N \in \mathbb{N}$ *copies of the same* probabilistic protocol $\mathcal{P} = (Q, q_0, \Sigma, \Delta)$ *where* Q *is a finite set of control states with* $q_0 \in Q$ *the initial state,* Σ *is a finite message alphabet, and* $\Delta \subseteq (Q \times \{!!, ??\} \times \Sigma \times Q) \cup (Q \times \{\varepsilon\} \times \mathsf{Dist}(Q))$ *is the edge function.*

Probabilistic protocols thus are Markov decision processes in which actions are either internal actions labelled with ε, message broadcasts labelled by $!!m$, or message receptions labelled by $??m$, for $m \in \Sigma$. Figure 5.1 represents an example of probabilistic protocol.

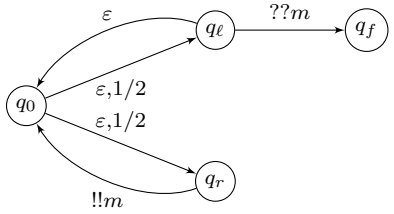

Figure 5.1: An example of probabilistic protocol.

The intuitive interpretation of a probabilistic broadcast network is that a number of processes execute the probabilistic protocol $\mathcal{P}$ simultaneously. The precise semantics however depends on several criteria: first whether the topology is fixed or reconfigurable, and second whether the number of nodes is fixed (yet unknown) or may vary during an execution.

Generally, the semantics of a network can be defined as an infinite-state MDP. A scheduler in such an MDP is responsible for choosing: (1) which process will execute an active action (internal or broadcast), (2) which action it will actually perform, (3) possibly which other processes will be the recepients of a broadcast and (4) which reception edge the receivers will fire. Note that the third step only applies to the case of networks with reconfigurable communication topology. The probabilities are induced by (a) probabilities inherent to the probabilistic protocol description and (b) possibly network size evolution by random disappearances and creations of nodes. Note that the second probabilistic feature only applies to networks with variable size. For each case we consider, we will detail the semantics and use illustrative examples.

Once an MDP semantics $\mathcal{M}_{\mathcal{P},N}$ is given for a probabilistic broadcast network $\mathcal{P}^N$, we focus on qualitative reachability questions. An execution ρ in $\mathcal{M}_{\mathcal{P},N}$ satisfies the property $\Diamond q_f$ for $q_f \in Q$ a state of $\mathcal{P}$ if along ρ, there exists a configuration for which at least one process is in state q_f. Note that not all processes should reach state q_f, and a fortiori not all at the same time, and a single process visiting q_f is sufficient. This particular reachability notion can thus also be seen as a coverability property. We are then interested in the set of executions that satify $\Diamond q_f$.

More precisely, we consider the following parameterised verification problems. For every (initial) network size N, is it the case that for every scheduler σ, $\mathbb{P}_\sigma(\mathcal{M}_{\mathcal{P},N} \models \Diamond q_f) = 1$? The latter question can be reformalized as the existence of $N \in \mathbb{N}_{>0}$ such that $\min_\sigma \mathbb{P}_\sigma(\mathcal{M}_{\mathcal{P},N} \models \Diamond q_f) = 1$. In general, all qualitative questions of interest can be unified as follows: for $\mathtt{opt} \in \{\min, \max\}$, $\mathtt{b} \in \{0, 1\}$ and $\sim \in \{<, =, >\}$ the decision problem $\textsc{Reach}^{\sim \mathtt{b}}_{\mathtt{opt}}$ asks, given a probabilistic protocol $\mathcal{P}$ and a control state $q_f \in Q$, whether there exists $N \in \mathbb{N}_{>0}$ such that $\mathtt{opt}_\sigma\ \mathbb{P}_\sigma(\mathcal{M}_{\mathcal{P},N} \models \Diamond q_f) \sim \mathtt{b}$.

5.2 Probabilistic clique networks

In this section we detail our contributions for probabilistic broadcast networks in which the communication topology is a clique. We considered two frameworks: on the one hand static networks with a fixed number of nodes, and on the other hand dynamic networks of varying size.

5.2.1 Fixed size clique networks

We first consider a semantics in which the communication topology is a clique, and the number of processes is fixed. This semantics is appropriate for networks of processes that are always connected, such as sensors deployed in a room or a small building. Let us first illustrate the semantics with an example execution of $\mathcal{P}^4$ for the protocol $\mathcal{P}$ from Figure 5.1. Processes are drawn as diamonds labelled by the state of $\mathcal{P}$ they are in. The (undirected) edges between processes represent the communication topology, in this case a complete graph. In the first

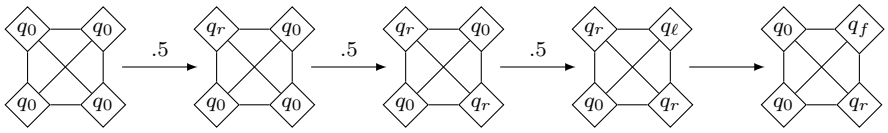

Figure 5.2: A fixed size clique execution of the probabilistic protocol from Figure 5.1.

step of this example clique execution, the top left process performs an internal action from q_0, resulting –with half probability– in moving to state q_r; the same happens for the bottom right process in the second step; in the third step, the same internal action performed by the top right process leads –still with probability .5– to q_ℓ; finally in the last step, the top left process broadcasts message m to its neighbours (here all other processes), and the top right one is the only one able to receive it, and moves to q_f.

Since the topology is bound to be a clique, and the processes do not have identifiers, a configuration can be encoded by a multiset over Q, and the example execution from Figure 5.2 can alternatively be represented by the following sequence

$$4q_0 \xrightarrow{.5} 3q_0 + 1q_r \xrightarrow{.5} 2q_0 + 2q_r \xrightarrow{.5} 1q_0 + 1q_\ell + 2q_r \to 2q_0 + 1q_r + 1q_f.$$

Assuming the sequence of actions (as well as the process responsible for each action) is fixed, the probability of this execution is $\frac{1}{8}$. More generally, the choice of active process and action is decided upon by a *strategy*. From a given configuration, a strategy chooses 1) a process to

perform an action, 2) the action it fires and 3) the reception edge for each other process, in case the chosen action in the second step is a broadcast. To formally define the transition relation between configurations, we assume an order on processes, and write $\gamma[i]$ for the state of process i in configuration γ. Moreover, the size of a configuration γ is defined as the number of processes involved in the network: $|\gamma| = \sum_{q \in Q} \gamma(q)$. Using these notations, there is a transition from γ to γ' if $|\gamma| = |\gamma'|$ and one of the following conditions holds:

- there exists an index $i \leq |\gamma|$ such that $(\gamma[i], \varepsilon, \delta) \in \Delta$ is an edge of $\mathcal{P}$, $\delta(\gamma'[i]) > 0$ and for every $j \neq i$, $\gamma'[j] = \gamma[j]$.
- there exists an index $i \leq |\gamma|$ such that $(\gamma[i], !!m, \gamma'[i]) \in \Delta$ is an edge of $\mathcal{P}$, and for every $j \neq i$, $(\gamma[j], ??m, \gamma'[j]) \in \Delta$ is an edge of $\mathcal{P}$.

In the first case, the probability of the transition from γ to γ' is exactly $\delta(\gamma'[i])$. In the second case, the transition has probability one. Note that we implicitely assume that broadcast protocols are complete with respect to receptions of messages: from every state q and for every message type m, there exists a transition $(q, ??m, q')$.

Some simple qualitative reachability questions are decidable for networks of probabilistic protocols under clique topology and fixed number of processes.

Proposition 5.1 *The problems* $\text{REACH}^{=0}_{\max}$, $\text{REACH}^{<1}_{\max}$, *and* $\text{REACH}^{>0}_{\max}$ *are decidable for static probabilistic networks.*

Recall, that *e.g.*, $\text{REACH}^{=0}_{\max}$ asks whether the maximum probability over all strategies to reach a given control state is null. Equivalently, it asks whether for all strategies, the probability is 0 to reach a configuration where some process is in a given final state.

To establish the decidability of $\text{REACH}^{=0}_{\max}$ and $\text{REACH}^{<1}_{\max}$, we remark that networks of probabilistic protocols enjoy a monotonicity property. In the network $\mathcal{P}^{N+1}$, some schedulers only involve N processes, so that the maximal probability among schedulers of eventually reaching a configuration with q_f is non-decreasing with N. With this observation, $\text{REACH}^{=0}_{\max}$ and $\text{REACH}^{<1}_{\max}$ can be reduced to the non-parameterised question on $\mathcal{P}$ (that is for a single participant). The decidability of $\text{REACH}^{=0}_{\max}$ and $\text{REACH}^{<1}_{\max}$ thus derives from the decidability of $\max_\sigma \mathbb{P}_\sigma(\Diamond q_f) = 0$ (resp. < 1) in finite-state MDP (see *e.g.* [BK08]).
Also $\text{REACH}^{>0}_{\max}$ is equivalent to the existence of an execution reaching a configuration with q_f in the network $\mathcal{P}^N$ for some $N \in \mathbb{N}$. The decidability of $\text{REACH}^{>0}_{\max}$ is therefore a consequence of decidability of the parameterised reachability problem in the non-probabilistic case, established in [DSZ11a].

Although the three above cases are easily decidable, the situation is quite different for the remaining parameterised verification questions.

Theorem 5.1 *The problems* $\text{REACH}^{=1}_{\max}$, $\text{REACH}^{=0}_{\min}$, $\text{REACH}^{>0}_{\min}$, $\text{REACH}^{<1}_{\min}$ *and* $\text{REACH}^{=1}_{\min}$ *are undecidable for static probabilistic networks.*

The undecidability results in Theorem 5.1 are shown by reduction from the halting and boundedness problems of a 2-counter machine which are known to be undecidable [Min67]. For $\text{REACH}^{<1}_{\min}$ and, $\text{REACH}^{=1}_{\min}$, the proofs are more elaborate and are inspired by techniques from [ABM07]. Without loss of generality, we consider a deterministic *infinitely testing* 2-counter machine $\mathcal{M}$, in which if the execution is infinite, then infinitely often some counter has a positive value and is tested to zero. 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. Let us explain the key ideas of the reduction for $\text{REACH}^{<1}_{\min}$.

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 `idle`. The increment of counter $\mathtt{c}_i$ 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 guesses at random 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 processes 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 $\min_\sigma \mathbb{P}_\sigma(\mathcal{M}_{\mathcal{P},N} \models \Diamond(\mathtt{err} \cup \mathtt{err}_z)) < 1$.

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

5.2.2 Variable size clique networks

We now turn to dynamic probabilistic networks, and will see that the decidability of the qualitative parameterised verification problems is recovered thanks to random evolution of the network size. With an application like wireless sensor networks in mind, the number of nodes can change during the execution of the system, since nodes can break down or run out of battery, but also, nodes can be newly inserted or refill their battery. We therefore proposed a model of probabilistic 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. A dynamic probabilistic network is thus defined by its underlying probabilistic protocol $\mathcal{P}$, an initial number of nodes N, together with a pair $\Lambda = (\lambda_-, \lambda_+) \in (0,1)^2$ of disappearance and creation rates.

The rates λ_+ and λ_- represent dynamic creation and disappearance of processes according to fixed probabilistic laws. More precisely, after every discrete action, each process disappears with probability λ_-, followed by the creation of k processes (in control state q_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 network from Subsection 5.2.1.

Let us illustrate the semantics of dynamic probabilistic networks by giving an example of

execution. In the first step of that execution, the top left process performs an internal action and moves to q_r (with probability .5) followed by the disappearance of the two right processes (with probability $\lambda_-^2(1-\lambda_-)^2$ since the left processes were unaffected), and no creation of processes (with probability $(1-\lambda_+)$). During the second and third steps however, a new process is created top right and bottom right respectively. In the last step the top right process reaches q_f before all other processes disappear. Note that the very same execution can be obtained by different disappearances and creations: *e.g.* in the first step, all but one process (the one in q_r after the transition) disappear, and one process is created, which would all in all happen with probability $\lambda_-^3(1-\lambda_-)\lambda_+(1-\lambda_+)$. For simplicity, since many such options lead to the same execution, we chose not to indicate the precise value of the probability on the transitions.

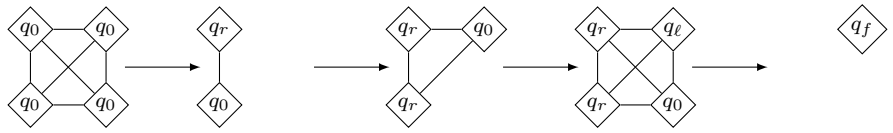

Figure 5.3: A dynamic clique execution of the probabilistic protocol from Figure 5.1.

Here also, even if the number of processes evolves over time, since the topology is always a clique and because the processes do not have identifiers, a configuration can be encoded by a multiset over Q, and the example execution from Figure 5.3 can alternatively be represented by the following sequence

$$4q_0 \to 1q_0 + 1q_r \to 1q_0 + 2q_r \to 1q_0 + 1q_\ell + 2q_r \to 1q_f.$$

Adding random creations and disappearances of nodes in the network makes parameterised verification easier:

Theorem 5.2 *The problems* $\textsc{Reach}^{\sim \mathsf{b}}_{\mathsf{opt}}$ *are decidable for dynamic probabilistic networks, and are non-primitive recursive.*

To establish the decidability of all parameterised qualitative reachability problems, we show that the finite-state MDP $\mathcal{M}^{\Lambda}_{\mathcal{P},N_0}$ (for fixed N_0) enjoys a finite attractor property, and that it can be equipped with a well quasi order on its configurations. We then rely on fixpoint characterizations of the set of initial configurations that satisfy a given property, and reuse techniques from [BS13a] to show the convergence in finite time of the iterative computation of these fixpoints.

Let us give some details on these different steps. In the case of clique networks, configurations are multisets of states of the probabilistic protocol $\mathcal{P}$, that is $\gamma : Q \to \mathbb{N}$. The set Conf of configurations is naturally equipped with the partial order $\preceq$ defined as follows: $\gamma \preceq \gamma'$ as soon as for every $q \in Q$, $\gamma(q) \leq \gamma'(q)$. By Dickson's lemma, $\preceq$ is a well quasi order: every infinite sequence of configurations $(\gamma_i)_{i\in\mathbb{N}}$ contains two indices $i < j$ such that $\gamma_i \preceq \gamma_j$.

Given any set $\Gamma \subseteq \mathsf{Conf}$, we write $\uparrow\Gamma$ for the upward closure of Γ with respect to the well quasi order $\preceq$. Formally, $\uparrow\Gamma = \{\gamma' \in \mathsf{Conf} \mid \exists \gamma \in \Gamma :\ \gamma \preceq \gamma'\}$. Moreover, Γ is *upward-closed* if $\uparrow\Gamma = \Gamma$. We can show that the MDP induced by a probabilistic network is compatible with $\preceq$, and that given an upward-closed set Γ, the set of its predecessors is also upward-closed, and can be computed effectively.

The second step is to show that the MDP $\mathcal{M}^{\Lambda}_{\mathcal{P},N_0}$ has a finite attractor. In a (possibly infinite-state) Markov chain, a finite attractor, is a finite set of states that is guaranteed to be visited infinitely often almost-surely. An MDP has a *finite attractor* if there exists a finite set of states which is an attractor under all possible schedulers. Typical examples of Markov chains and Markov decision processes with a finite attractor are the ones induced by probabilistic lossy channel systems (pLCS), a model we studied with Philippe Schnoebelen [BS03, BS13b]. Here, similarly to pLCS, one can show that the singleton configuration with no process is a finite attractor for $\mathcal{M}^{\Lambda}_{\mathcal{P},N_0}$. Intuitively, because the creation rate is fixed, whereas the disappearance rate increases with the number of processes, for large size networks, at each step, it is more likely that the size decreases than that it increases. Applying the general criterion from [BBS06] on left oriented Markov chains allows to conclude that $\mathcal{M}^{\Lambda}_{\mathcal{P},N_0}$ admits a finite attractor.

The fact that the empty configuration is a finite attractor can seem to be an undesirable property of the model. Let us argue that it is not totally unrealistic. A consequence is that the validity of a qualitative property does not depend on the initial number of processes N. Moreover the extremal probabilities of reaching a configuration with some process in q_f is either 0 or 1. At first, these observations are drawbacks of the dynamic probabilistic network model. Yet, the setting and the decidability results 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.

In a joint work with Philippe Schnoebelen [BS13a], we developped a general framework to prove the convergence of fixpoint computations in infinite-state systems. Thanks to the finite attractor property, and because the predecessor operator preserves upward closure, we can reuse fixpoint characterizations from [BS13a] and guarantee that their computation will terminate.

Consider as an example the decision problem $\textsc{Reach}^{>0}_{\max}$, that is whether there exists an initial network size N and a scheduler σ such that $\mathbb{P}_\sigma(\mathcal{M}^{\Lambda}_{\mathcal{P},N} \models \Diamond q_f) > 0$. We have the following equivalences:

$$\begin{aligned} \exists N,\ \exists \sigma\ \mathbb{P}_\sigma(\mathcal{M}^{\Lambda}_{\mathcal{P},N} \models \Diamond q_f) > 0 \quad &\iff \quad \exists \sigma\ \mathbb{P}_\sigma(\mathcal{M}^{\Lambda}_{\mathcal{P},0} \models \Diamond q_f) > 0 \\ &\iff \quad \gamma_\emptyset \in \mu X.(\uparrow q_f \vee \mathsf{Pre}(X)) \ . \end{aligned}$$

The first equivalence comes from the remark above, and more specifically that the qualitative reachability does not depend on the initial network size N because of the finite attractor property. Now, the fixpoint can be computed effectively (and the computation terminates) because Pre preserves upward-closure and $\preceq$ is a well quasi order. It then suffices to test whether the empty configuration $\gamma_\emptyset$ belongs to that set to decide $\textsc{Reach}^{>0}_{\max}$. Of course, this case is the simplest one, but for the other cases, we could also provide fixpoint expressions, and guarantee their effective computability by [BS13a].

To conclude this section, for the lower-bound, we performed a reduction from the reachability problem in lossy channel systems, which is known to be non-primitive recursive [Sch02] and more precisely $\mathbf{F}_{\omega^\omega}$-complete [SS12].

5.3 Reconfigurable probabilistic networks

Different to the model of clique networks studied in the last section, we turn now to probabilistic networks with reconfigurable communication links. Since the topology is not fixed, the parameterised reachability problems we consider incorporate the initial network (number of nodes and communication topology). Also, configurations can no longer be seen as multisets over the probabilistic protocol state space Q, but we need to consider graphs labelled by Q.

A configuration γ is thus a labelled graph (N, E, L) where N is a set of nodes, $E \subseteq (N \times N) \setminus \{(n, n) \mid n \in N\}$ is a set of edges and $L : N \to Q$ is a labelling function. There is a transition from $\gamma = (N, E, L)$ to $\gamma' = (N', E', L')$ in the reconfigurable network if one of the following conditions holds:

- there exists a node $n \in N$ and $(L(n), \varepsilon, \delta) \in \Delta$ an edge of $\mathcal{P}$, such that $\delta(L'(n)) > 0$, and for every $n' \neq n$, $L'(n') = L(n')$; or
- there exists a node $n \in N$ such that $(L(n), !!m, L'(n)) \in \Delta$ is an edge of $\mathcal{P}$, and for every $n' \neq n$, $(L(n'), ??m, L'(n')) \in \Delta$ is an edge of $\mathcal{P}$.

In the first case, there is an internal transition outgoing γ and leading to γ' with probability $\delta(L'(n)) > 0$. In the latter case, there is a communication transition, with probability 1, from γ to γ'.

This semantics is illustrated in Figure 5.4 by an execution of the network with 4 processes executing the probabilistic protocol from Figure 5.1. Note that the communication topology may change at each step. For example, in the last step, the message broadcast by the bottom right node only reached the top left process.

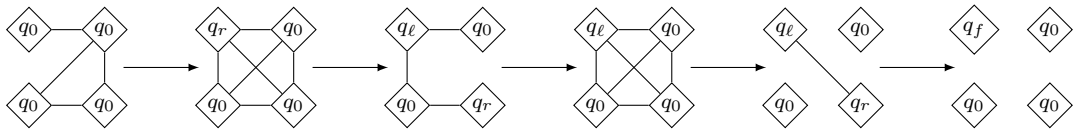

Figure 5.4: A reconfigurable execution of the probabilistic protocol from Figure 5.1.

Let us first explain how to solve the easiest decision problems for reconfigurable probabilistic networks. First of all, $\text{REACH}^{>0}_{\max}$ is interreducible to the reachability problem in non-probabilistic reconfigurable broadcast networks, which is known to be PTIME-complete [DSTZ12]. For $\text{REACH}^{=0}_{\max}$, $\text{REACH}^{<1}_{\max}$, $\text{REACH}^{=1}_{\min}$ and $\text{REACH}^{>0}_{\min}$ we use a monotonicity property. Intuitively, in reconfigurable probabilisitic networks, the more nodes, the higher the probability to reach the target. The problems cited above therefore reduce to qualitative reachability questions in the finite state MDP $\mathcal{P}$ (the network with a single process) and are thus solvable in PTIME.

Proposition 5.2 $\text{REACH}^{>0}_{\max}$, $\text{REACH}^{=0}_{\max}$, $\text{REACH}^{<1}_{\max}$, $\text{REACH}^{=1}_{\min}$ *and* $\text{REACH}^{>0}_{\min}$ *are decidable in* PTIME *for reconfigurable probabilistic networks.*

The resolution of the other qualitative reachability problems for probabilistic reconfigurable broadcast networks is not direct and goes through another model of parameterised broadcast networks in which the local behaviour of processes is described by a 2-player game, rather than by an MDP. Markov decision processes and 2-player games are often related, and in

the monolithic case, one can think of an MDP as a 2-player game in which the opponent is somehow fair. Indeed, probabilities rule out the possibility to always choose the same transition. Our translation is inspired by a similar construction for finite state systems (see for instance [CdAFL09]). However when moving to parameterised systems, several problems raise. First, it is unclear whether the translation proposed for finite-state systems is correct for parameterised –hence infinite-state– systems. More importantly, our aim is to design such a transformation at the protocol level, rather than at the network level, *i.e.* avoiding to build the product, since the number of nodes is unknown. This induces another issue, which is to define the winning condition locally, and not globally. All these reasons make the finite-state tranformation unapplicable in our context, and justify an *ad hoc* construction.

Before providing a reduction from probabilistic broadcast networks to networks of 2-player games, let us introduce the latter model.

Definition 5.2 *A* 2-player broadcast network $\mathcal{G}^N$ *consists of* $N \in \mathbb{N}$ *copies of the same 2-player game protocol* $\mathcal{G} = (Q^{(1)}, Q^{(2)}, \Sigma, \Delta, \mathsf{col})$ *where* $Q = Q^{(1)} \sqcup Q^{(2)}$ *is the set of states partitionned into player* 1 *and player* 2 *states, with* $q_0 \in Q^{(1)}$ *the initial state,* Σ *is a finite message alphabet,* $\Delta \subseteq (Q^{(1)} \times (\{!!m, ??m \mid m \in \Sigma\} \cup \{\varepsilon\}) \times Q) \cup (Q^{(2)} \times \{\varepsilon\} \times Q)$ *is the edge function, and* $\mathsf{col} : \Delta \to \mathbb{N}$ *is the coloring function.*

The semantics of a 2-player broadcast network is given in term of a 2-player game whose definition is similar to the MDP associated with a probabilistic network. Here also player 1 has the ability to choose a communication topology and a node which will perform an action. The only difference is that, according to the control state labelling this node, either player 1 or player 2 will then perform the next move. Note that the roles of player 1 and player 2 are not symmetric: only player 1 can initiate a communication, and player 2 performs only internal actions. Colors labelling the protocol edges serve to express a parity winning condition. An infinite execution in the 2-player game describing the network is winning for player 1 if the maximal color seen infinitely often along the execution is even.

The natural parameterised verification question associated with a 2-player broadcast network is then whether there exists a network size N such that player 1 has a winning strategy in the 2-player game $\mathcal{G}^N$. To solve this question for 2-player broadcast networks, we first show that we can restrict the strategies of player 2 to strategies that always choose from a given control state the same successor, independently of the configuration, or the history in the game. Such a strategy is called local, and amounts to selecting, at the protocol level, one successor per state of player 2. Then, we solve the problem of the existence of a network size and a strategy for player 1 winning against a fixed local strategy for player 2. When the strategy of player 2 is fixed, we are left with a network of identical automata, and it suffices to decide whether there exists an infinite execution of this network satisfying the parity condition on colors. Using a counting abstraction which translates the network into a Vector Addition System with States (VASS) and then by looking in this VASS for a cycle whose effect on each of the manipulated values is null, we deduce a PTIME algorithm for the latter problem thanks to [KS88]. Altogether this provides a co-NP algorithm for solving parity conditions on 2-player broadcast networks.

To conclude for the original problem of interest on probabilistic reconfiguration networks, let us sketch the idea of the reduction for $\textsc{Reach}^{=1}_{\max}$ to 2-player broadcast networks. Intuitively, all random choices corresponding to internal actions in $\mathcal{P}$ are replaced in $\mathcal{G}_{\mathcal{P},p}$ (p for parity) with choices for player 2, where either it decides the outcome of the probabilistic choice,

or lets player 1 choose. Only transitions where player 2 makes the decision corresponding to a probabilistic choice and the self loop on the state q_f have parity 2. Figure 5.5 illustrates this reduction on the example probabilistic protocol from Figure 5.1.

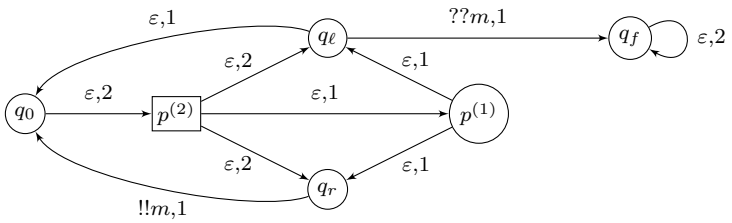

Figure 5.5: 2-player parity protocol for the probabilistic protocol from Figure 5.1.

The intuition of the reduction, is that random choices are not controlled by the strategy, and should therefore be decided in the game by player 2. Yet, player 2 should behave in a fair way, and it may not deliberately avoid some transition forever. This explains that it may choose the successor, but with a parity 2 transition (which is good for player 1). Of course, the other option for player 1 to win is to reach q_f. Under this construction, we obtain a reduction from $\textsc{Reach}^{=1}_{\max}$ to networks of 2-player games with parity winning condition.

Proposition 5.3

$$\exists N,\ \exists \sigma\ \mathbb{P}_\sigma(\mathcal{M}_{\mathcal{P},N} \models \Diamond q_f) < 1 \quad \Longleftrightarrow \quad \exists N,\ \exists \sigma^1,\ \forall \sigma^2\ (\mathcal{G}^N_{\mathcal{P},p}, \sigma^1, \sigma^2) \models \mathsf{Parity}\ .$$

Theorem 5.3 $\textsc{Reach}^{=1}_{\max}$ *is* co-NP*-complete for reconfigurable probabilistic networks.*

We explained above how to obtain a co-NP algorithm for $\textsc{Reach}^{=1}_{\max}$ via a parity question on 2-player broadcast networks. To establish the lower bound, we reduce the unsatisfiability problem to $\textsc{Reach}^{=1}_{\max}$. From φ a formula in conjunctive normal form, we define a probabilistic protocol $\mathcal{P}_\varphi$ and a control state q_f such that φ is unsatisfiable if and only if there exists an initial configuration $\gamma_0 \in \mathsf{Conf}_0$ and a stratey σ such that $\mathbb{P}_\sigma(\gamma_0 \models \Diamond q_f) = 1$. Figure 5.6 presents the construction for an example formula. For readability, the initial state q_0 of the probabilistic protocol is duplicated in the picture: it appears once in the formula part, and once per variable. The idea, if φ is unsatisfiable, is to generate a random assignment of the variables (using the gadgets represented bottom of the figure), which will necessarily violate some clause of φ. Choosing then this clause in the above part of the protocol allows one to reach state r_1, and from there to reach q_f with probability half. Iterating this process, the target can be almost-surely reached. The converse implication relies on the fact that if φ is satisfiable, there is a positive probability to generate a valuation satisfying it, and then not to be able to reach r_1, a necessary condition to reach q_f. Therefore, the maximum probability to reach the target is smaller than 1 in this case.

For the two remaining problems $\textsc{Reach}^{=0}_{\min}$ and $\textsc{Reach}^{<1}_{\min}$, we also reduce to a decision problem on networks of 2-player games. However, the winning condition in the game network cannot be a simple parity condition: we need to consider either a safety condition or safety and parity conditions simultaneously. It is important to notice that, in classical 2-player games, any safety condition (even combined with a parity one) can be encoded into an equivalent

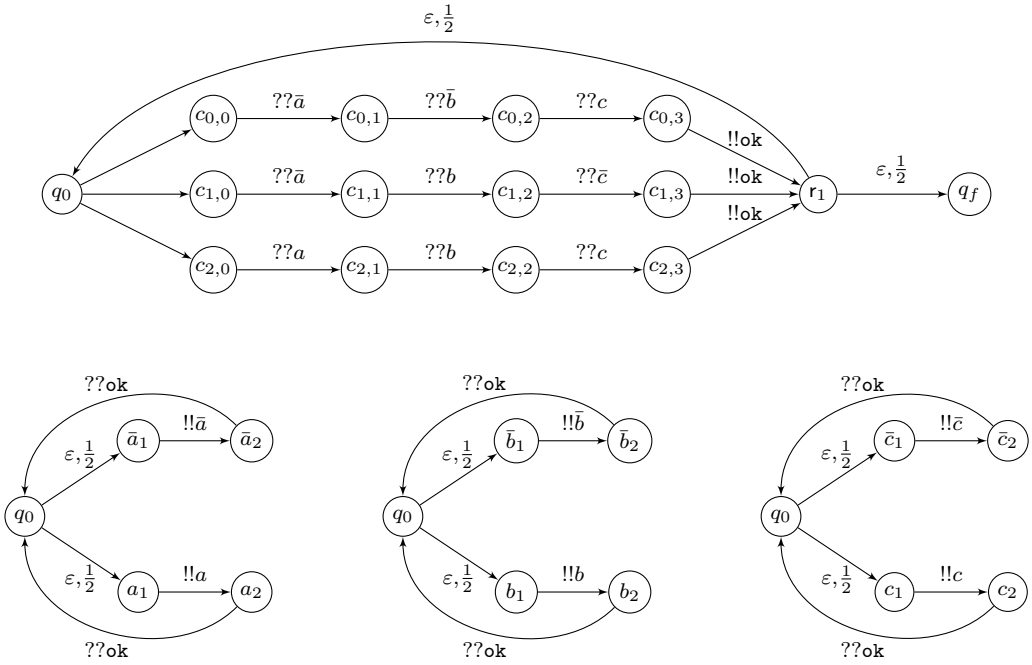

Figure 5.6: Probabilistic protocol for the formula $\varphi = (a \vee b \vee \bar{c}) \wedge (a \vee \bar{b} \vee c) \wedge (\bar{a} \vee \bar{b} \vee \bar{c})$.

parity condition. We couldn't find such an encoding for networks of 2-player games. The intuition is that in a network semantics, the safety condition can be violated by one process being in an unsafe state. Yet, if this process does not play any role in the remaining of the execution, the safety violation cannot be expressed as a parity condition on the edges that are fired infinitely often. We proved in [BFS14] that solving networks of 2-player games with a safety, or a safety-parity condition is in $\mathsf{co\text{-}NP}^{\mathsf{NP}}$. Recall that this complexity class gathers decision problems that can be solved by a $\mathsf{co\text{-}NP}$ algorithm with access to an NP oracle. Yet, we later improved that bound and showed that these decision problems are in fact in $\mathsf{co\text{-}NP}$.

Back to our qualitative reachability questions on reconfigurable probabilistic networks, $\textsc{Reach}^{=0}_{\min}$ and $\textsc{Reach}^{<1}_{\min}$ can be reduced to a game problem for networks of 2-player games, with a safety and a safety/parity winning condition, respectively. From a probabilistic protocol $\mathcal{P}$, for $\textsc{Reach}^{=0}_{\min}$, we build a parity protocol $\mathcal{G}_{\mathcal{P},s}$ (s for safety) where all random choices in $\mathcal{P}$ are replaced in $\mathcal{G}_{\mathcal{P},s}$ with choices for player 2. The transitions with target q_f are the only ones that violate the safety objective. This simple transformation is illustrated on Figure 5.7 for our example probabilistic protocol from Figure 5.1, and unsafe transitions are depicted by dashed arrows. Intuitively, if some strategy ensures to almost surely avoid q_f, it in fact avoids q_f surely. Therefore, random choices can legally be replaced by choices of the opponent. This transformation thus yields the following equivalence:

Proposition 5.4

$$\exists N,\ \exists \sigma\ \mathbb{P}_\sigma(\mathcal{P}^N \models \Diamond q_f) = 0 \quad \Longleftrightarrow \quad \exists N,\ \exists \sigma^1,\ \forall \sigma^2\ (\mathcal{G}^N_{\mathcal{P},s}, \sigma^1, \sigma^2) \models \mathsf{Safety}\ .$$

For $\textsc{Reach}^{<1}_{\min}$, $\mathcal{G}_{\mathcal{P},sp}$ (sp for safety-parity) consists of two copies of $\mathcal{P}$. In the first copy, all random choices are replaced with choices of player 1, whereas in the second copy they are replaced with choices of player 2. Also, at any time, one can move from the first to the second

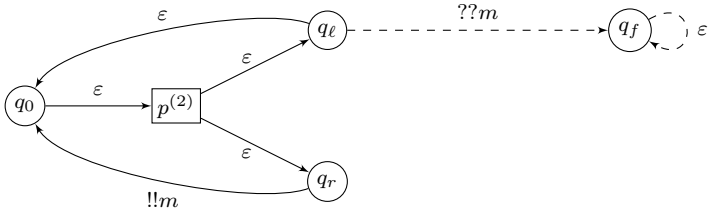

Figure 5.7: 2-player safety protocol for the probabilistic protocol from Figure 5.1.

copy. The parity of transitions with target in the second copy is 0, and otherwise it is 1. Moreover, the only unsafe transitions are those with target q_f. Again, this transformation is illustrated on Figure 5.8 for our example probabilistic protocol from Figure 5.1.

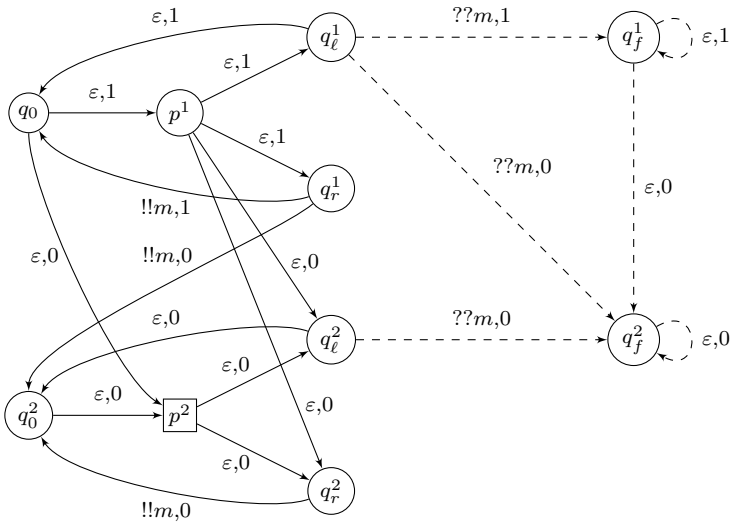

Figure 5.8: 2-player safety-parity protocol for the probabilistic protocol from Figure 5.1.

Intuitively, if some strategy can reach q_f with probability less than 1, necessarily it first reaches a configuration γ from which q_f is avoided surely. This explains the separation into two parts of the game protocol. More precisely, the first part corresponds to choosing a prefix leading to γ; this prefix has to be finite otherwise the parity condition is not met. Then, in the second part, probabilistic choices are left to the opponent, since q_f must be surely avoided.

Proposition 5.5

$$\exists N,\ \exists \sigma\ \mathbb{P}_\sigma(\mathcal{P}^N \models \Diamond q_f) < 1 \quad \Longleftrightarrow \quad \exists N,\ \exists \sigma^1,\ \forall \sigma^2\ (\mathcal{G}^N_{\mathcal{P},sp}, \sigma^1, \sigma^2) \models \mathsf{SafetyParity}\ .$$

These transformations from probabilistic protocols to 2-player game protocols, together with our resolution of networks of 2-player games, yield the following decidability result:

Theorem 5.4 $\textsc{Reach}^{=0}_{\min}$ *and* $\textsc{Reach}^{<1}_{\min}$ *are* co-NP*-complete for reconfigurable probabilistic networks.*

5.4 Perspectives

Local strategies So far, we considered strategies that take decisions based on the configuration of the whole network, that is given the current state of each of the processes composing the network. In order to represent more faithfully distributed systems, we aim at investigating *local* strategies, that would base their decisions only on the knowledge of the configuration each process has. More precisely, given an execution of the network, it seems reasonable to assume a process decides which action to perform based only on its view of the execution, that is on the actions it fired so far. A global scheduler is still needed to decide which process will perform the next action, yet, this second decision is local and should be resolved the same way by two processes sharing the same view. Such local strategies already make sense for non-probabilistic broadcast protocols, and we established the precise complexity of the reachability and synchronization problems[1] in some recent work with Fournier and Sangnier [BFS15]. Our decidability results contrast with the well-known unfeasibility of distributed synthesis as studied in [PR90] since the local strategies we restrict to can be seen as local controllers for the processes executing the protocol and whose role is to resolve the nondeterministic choices. Pursuing this line of work, we aim at extending to probabilistic networks the restriction to local strategies, and at developing verification techniques for parameterised networks that are more realistic models for distributed systems.

Towards distributed protocols More generally, one of the initial motivations for considering probabilistic broadcast protocols, was the validation of distributed protocols since many distributed protocols incorporate probabilities (*e.g.* as a convenient way to break symmetry between the participants) and should be validated for any number of participants. The synthesis of local strategies goes in the direction of tackling the design of distributed protocols from nondeterministic specifications. Beyond the reachability and synchronization problems, in order to answer needs from the distributed systems community it seems unavoidable to be able to formalize and solve richer properties such as the notion of consensus. Expressing consensus in broadcast networks calls for the need to enrich protocols with data. In general, our mid-term objective will be to demonstrate on case studies that the parameterised verification of probabilistic broadcast protocols can impact the design and validation of distributed protocols.

Parameterised probabilistic models for biology The verification of parameterised networks of probabilistic processes may also have application outside of the domain of distributed algorithms. Recently in the context of the french research project ANR STOCH-MC (on efficient techniques for model checking of large stochastic models), with Blaise Genest and Hugo Gimbert, we came accross a biological case study that would conveniently be modelled into a parameterised network of identical Markov decision processes. The evolution of the size of a yeast cell after an osmotic stress is not deterministic and rather follows a fixed probability distribution. The reaction of cells composing a population when biologists apply some drug may differ from cell to cell because of randomness, yet the drug dosage is applied uniformly to all the cells. The objective is to control the drug dosage at each time step so that globally

[1] The synchronization problem asks for the existence of an execution ending with a configuration in which all processes are in a state belonging to some target set.

in the population, the size of each cell is within a given range, and this based on the observation of the cells sizes in real time through fluorescence levels. The control is thus global (the same amount of drug for all the cells) but the response is different from cell to cell because of stochasticity, and because all cells are not in the same configuration before the osmotic stress. This biological example case study fits in a model of parameterised network of identical Markov decision processes. Different to the probabilistic broadcast networks we studied so far, here the processes (that is, the cells) do not communicate one with another, moreover, the strategy uniformly affects all cells. Typical questions of interest that we propose to tackle on such systems in a near future are, *a priori* by increasing complexity order: can one control the cell population to ensure that they all stay within a given size range? can one guarantee with high probability that a large proportion of cells has the desired size? Interestingly, such a model of parameterised population MDP can be seen as a discrete approximation of the MDP viewed as probability mass transformer.

Chapter 6

Conclusion and perspectives

This thesis summarises my recent contributions to the verification and control of complex systems with at least one of these features: time, probabilities, partial observation and parameters. More precisely, Chapter 2 focused on determinization of timed automata, Chapter 3 touched the model checking and control problems for stochastic timed automata, Chapter 4 gave an overview of various contributions for partially observable probabilistic systems, and Chapter 5 presented our results on parameterised verification of probabilistic models. Ongoing work and perspectives were given at the end on each chapter, I recall the main ones here and more generally discuss directions I envision for my research in the coming years.

More theoretical work

As a general objective, I would like to carry on investigating foundations of computer science and envision the following theoretical contributions.

Blindness and partial observation Since no problem so far separates POMDP from MDP under blind schedulers, I would like to better understand the connection between the two models. At the end of Chapter 4, I mentioned two concrete related research directions. First of all, while a lot of attention has been put recently on decidability results for probabilistic automata, I would like to investigate to what extent these positive results extend to MDP under partial observation. In particular, I will study the positive frequency objective for POMDP. I will also work on structural restrictions of POMDP so as to obtain decidability, thus trying to generalise *e.g.* the decidability results obtained for hierarchical probabilistic automata [CSVB15]. When lifting decidability results from probabilistic automata to partially observable MDP, one somehow needs to adapt arguments for words to strategies, hence trees. To fulfill this objective, I will thus have to learn more techniques related to trees, possibly including tree automata. More generally, one could look for a meta-theorem establishing that any decision problem decidable for the class of blind POMDP (or equivalently probabilistic automata) is also decidable for the whole class of POMDP. Alternatively, we might come accross a decision problem that separates probabilistic automata from POMDP.

Qualitative analysis of stochastic timed automata The decidability of qualitative model checking has only been established for restricted classes of stochastic timed automata, including single-clock STA and reactive STA with arbitrary many clocks. As explained in

Chapter 3 and illustrated on an example, the difficulty to obtain a decision algorithm for general STA lies in designing a correct finite-state abstraction when timed automata exhibit some convergence phenomena. In recent years, the characterization of these converging behaviours on the orbit graph, and namely forgetfulness, was used to perform a finer analysis of timed automata [BA11, Sta12, Bas13, SBMR13]. Also in the context of stochastic timed automata, we believe that analysing non-forgetful cycles could be the key to design a decision algorithm for the qualitative model checking problem. As a first step, we will explore whether the finite Markov chain abstraction presented in Chapter 3 is correct for the subclass of forgetful stochastic timed automata.

A mid-term objective will then be to tackle qualitative questions in games over STA. Given that the value 1 problem is polytime reducible to that latter problem, a decision algorithm for the qualitative analysis of games on STA would have to incorporate the rather precise analysis we conducted already for the value 1 problem in 1-player games on single-clock STA. Our general result on the verification of well structured transitions systems [BS13a] might be useful here, as it happened to be the case for qualitative games on probabilistic lossy channel systems [BS13b].

More quantitative properties

Although the models I consider are in essence quantitative, so far I mostly focused on their qualitative properties. Not only they are simpler to handle, but also qualitative properties can be justified in several ways. First, rather than asking that the probability is above a threshold and having to decide upon that threshold (0.9 or 0.99?), it is convenient to impose probability 1. Second, qualitative probabilities are more robust than quantitative ones in that numerical perturbations in the model barely alter the qualitative behaviour, so that modelisation imprecisions have little impact. Yet, in the future, I would like to move to quantitative questions, following the tendency in verification in the last decade.

Quantitative model checking of reactive stochastic timed automata A first quantitative analysis we propose to perform is the model checking of reactive stochastic timed automata. As explained in this document, reactivity is a nice property for STA since it induces the existence of regeneration points that are almost surely visited infinitely often. This allowed us to define approximation schemes for the probability of simple properties such as reachability or repeated reachability. On our agenda are now control problems to optimise these probabilities. More specifically, for 1-player games over STA, we aim at deciding the existence of a strategy that yields at least a given probability of successful runs. More generally, approximation schemes for infinite-state MDP under certain structural conditions is a topic we would like to investigate. A first attempt would be to try to generalise the approximation scheme for reachability probability in Markov chains with a finite attractor [PN97], possibly imposing more constraints on the MDP than the mere existence of such an attractor under all strategies.

Quantifying diagnosis As explained in Chapter 4, I would like to investigate further, diagnosis and more generally supervision problems for probabilistic systems under partial observation. A first immediate research topic will be to establish the precise complexity for the approximate diagnosability [TT05]. Although the definition of so-called AA-diagnosability

is qualitative, and contrary to the other notions I have studied so far, whether a system is approximately diagnosable or not, does depend on the probability values in the model. In that sense, solving approximate diagnosability and the related diagnoser synthesis problem would be a first step towards quantitative fault diagnosis in a probabilistic setting. As a mid-term objective, I envision optimization questions related to probabilistic diagnosis. On the one hand, defining diagnosers that dynamically optimize observed events is quite natural. Of course decreasing the set of observations may alter the probability of diagnosed faults, so that the controller should optimize a trade-off between observation cost and diagnosability level. Another such trade-off appears when the cost is linked to the observation frequency (*i.e.* to when the diagnoser requests an observation). A first concrete problem is to consider a labeled Markov chain and approximate the expected number of state observations before a given target set is observed. These optimization problems in which observation costs and diagnosability levels have to be balanced are natural for practitionners, and would at the same time unify and develop further our contributions on minimal disclosure for POMDP and on probabilistic diagnosis.

Robustness for probabilistic systems Related to the above quantification of diagnosability, I would like to consider robustness issues in probabilistic systems. Robustness expresses that the behavior of a system does not change drastically when the model is subject to small perturbations. Strategy synthesis for MDP in which the environment is unknown recently received attention with a setting in which the controller does not know *a priori* against which environment it plays within a finite set of them [RS14]. This problem is related to the approximate diagnosability, and its resolution involves learning phases in which the controller tries to increase its confidence on the environment identity, and control phases to optimize the probability of a given global objective. A similar methodology could be used to control a probabilistic system so as to diagnose it while at the same time ensuring a global property. Given the link with diagnosis, and because robustness is a crucial property, robustness questions for probabilistic systems are on my agenda.

More applications

In the forthcoming years, although I am naturally more attracted to theoretical work, I would like to dedicate some of my time to applications, starting with two objectives.

Uniform control under complete information First, thanks to the project STOCH-MC led by Blaise Genest on the model checking of large stochastic systems, I had the opportunity to discover problems steming from biology, for which computer science techniques could help. In particular, predicting the response of a cell population to osmotic stress is one of the project case studies. The reaction of individual cells to drug injection is varied and can be modeled by a probability distribution. Biologists can control the dosage, but the drug is applied to the whole population. The objective is thus to control uniformly a population of cells, based on the observation of each of them. We propose to model this case study by a parameterised family of identical Markov decision processes, and to design efficient decision algorithms ranging from qualitative to quantitative, to answer concrete questions steered by the application: for any population size, does there exist a control strategy that ensures all cells to remain healthy? can one guarantee with high probability that a large proportion of the cells will remain healthy?

Our contributions on parameterised verification and on control of probabilistic systems under partial observation will help solving such questions.

Parameterised verification of distributed protocols In some on-going work, as touched upon in the perspectives of Chapter 5, we get closer to distributed strategies in parameterised networks of identical protocols. Indeed, we define local strategies that are only based on the view of the processes. The motivation to do so, is to be able to verify distributed protocols with an arbitrary number of agents. Apart from local strategies, to achieve this objective, we will have to tackle verification questions beyond the reachability and synchronization problems, for example to model consensus, or leader election. The automated verification of a distributed algorithm modeled by networks of probabilistic protocols would validate on a case study our theoretical developments on parameterised verification of probabilistic systems.

Bibliography

[AB06] Rajeev Alur and Mikhail Bernadsky. Bounded model checking for GSMP models of stochastic real-time systems. In *Proceedings of the 9th International Workshop on Hybrid Systems: Computation and Control (HSCC'06)*, volume 3927 of *Lecture Notes in Computer Science*, pages 19–33. Springer, 2006.

[ABM07] Parosh Aziz Abdulla, Noomene Ben Henda, and Richard Mayr. Decisive Markov chains. *Logical Methods in Computer Science*, 3(4), 2007.

[ACD91] Rajeev Alur, Costas Courcoubetis, and David L. Dill. Model-checking for probabilistic real-time systems. In *Proceedings of the 18th International Colloquium on Automata, Languages and Programming (ICALP'91)*, volume 510 of *Lecture Notes in Computer Science*, pages 115–126. Springer, 1991.

[ACD92] Rajeev Alur, Costas Courcoubetis, and David L. Dill. Verifying automata specifications of probabilistic real-time systems. In *Proceedings of the REX Workshop on Real-Time: Theory in Practice*, volume 600 of *Lecture Notes in Computer Science*, pages 28–44. Springer, 1992.

[AD90] Rajeev Alur and David L. Dill. Automata for modeling real-time systems. In *Proceedings of the 17th International Colloquium on Automata, Languages and Programming (ICALP'90)*, volume 443 of *Lecture Notes in Computer Science*, pages 322–335. Springer, 1990.

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[AFH94] Rajeev Alur, Limor Fix, and Thomas A. Henzinger. A determinizable class of timed automata. In *Proceedings of the 6th International Conference on Computer Aided Verification (CAV'94)*, volume 818 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 1994.

[ALP01] Rajeev Alur, Salvatore La Torre, and George J. Pappas. Optimal paths in weighted timed automata. In *Proceedings of the 4th International Workshop on Hybrid Systems: Computation and Control (HSCC'01)*, volume 2034 of *Lecture Notes in Computer Science*, pages 49–62. Springer, 2001.

[AMPS98] Eugene Asarin, Oded Maler, Amir Pnueli, and Joseph Sifakis. Controller synthesis for timed automata. In *Proceedings of IFAC Symposium on System Structure and Control*, pages 469–474. Elsevier Science, 1998.

[ASSB00] Adnan Aziz, Kumud Sanwal, Vigyan Singhal, and Robert K. Brayton. Model-checking continuous-time Markov chains. *ACM Transactions on Computational Logic*, 1(1):162–170, 2000.

[Ast65] Karl J. Aström. Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 10:174–205, 1965.

[BA11] Nicolas Basset and Eugene Asarin. Thin and thick timed regular languages. In *Proceedings of the 9th International Colloquium on Formal Modeling and Analysis of Timed Systems (FORMATS'11)*, volume 6919 of *Lecture Notes in Computer Science*, pages 113–128. Springer, 2011.

[Bas13] Nicolas Basset. A maximal entropy stochastic process for a timed automaton. In *Proceedings of the 40th International Colloquium on Automata, Languages and Programming (ICALP'13)*, volume 7966 of *Lecture Notes in Computer Science*, pages 61–73. Springer, 2013.

[BB05] Laura Bernán Briones and Ed Brinksma. A test generation framework for quiescent real-time systems. In *Proceedings of the 4th International Workshop on Formal Approaches to Software Testing (FATES'04)*, volume 3395 of *Lecture Notes in Computer Science*, pages 64–78, 2005.

[BBB+07] Christel Baier, Nathalie Bertrand, Patricia Bouyer, Thomas Brihaye, and Marcus Größer. Probabilistic and topological semantics for timed automata. In *Proceedings of the 27th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'07)*, volume 4855 of *Lecture Notes in Computer Science*, pages 179–191. Springer, 2007.

[BBB+08] Christel Baier, Nathalie Bertrand, Patricia Bouyer, Thomas Brihaye, and Marcus Größer. Almost-sure model checking of infinite paths in one-clock timed automata. In *Proceedings of the 23rd Annual IEEE Symposium on Logic in Computer Science (LICS'08)*, pages 217–226. IEEE Computer Society Press, 2008.

[BBB+14] Nathalie Bertrand, Patricia Bouyer, Thomas Brihaye, Quentin Menet, Christel Baier, Marcus Größer, and Marcin Jurdziński. Stochastic timed automata. *Logical Methods in Computer Science*, 10(4), 2014.

[BBBB09] Christel Baier, Nathalie Bertrand, Patricia Bouyer, and Thomas Brihaye. When are timed automata determinizable? In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP'09)*, volume 5556 of *Lecture Notes in Computer Science*, pages 43–54. Springer, 2009.

[BBBM08] Nathalie Bertrand, Patricia Bouyer, Thomas Brihaye, and Nicolas Markey. Quantitative model-checking of one-clock timed automata under probabilistic semantics. In *Proceedings of the 5th International Conference on the Quantitative Evaluation of SysTems (QEST'08)*, pages 55–64, Saint Malo, France, 2008. IEEE Computer Society Press.

[BBBS11] Nathalie Bertrand, Patricia Bouyer, Thomas Brihaye, and Amélie Stainer. Emptiness and universality problems in timed automata with positive frequency. In *Proceedings of the 38th International Colloquium on Automata, Languages and Programming (ICALP'11)*, volume 6756 of *Lecture Notes in Computer Science*, pages 246–257. Springer, 2011.

[BBG08] Christel Baier, Nathalie Bertrand, and Marcus Größer. On decision problems for probabilistic Büchi automata. In *Proceedings of the 11th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'08)*, volume 4972 of *Lecture Notes in Computer Science*, pages 287–301. Springer, 2008.

[BBG12] Christel Baier, Nathalie Bertrand, and Marcus Grösser. Probabilistic ω-automata. *Journal of the ACM*, 59(1):1, 2012.

[BBG14] Nathalie Bertrand, Thomas Brihaye, and Blaise Genest. The steady-state control problem for Markov decision processes. In *Proceedings of the 11th International Conference on Quantitative Evaluation of Systems (QEST'14)*, volume 8657 of *Lecture Notes in Computer Science*, pages 313–328. Springer, 2014.

[BBH+13] Paolo Ballarini, Nathalie Bertrand, András Horváth, Marco Paolieri, and Enrico Vicario. Transient analysis of networks of stochastic timed automata using stochastic state classes. In *Proceedings of the 10th International Conference on Quantitative Evaluation of Systems (QEST'13)*, volume 8054 of *Lecture Notes in Computer Science*, pages 355–371. Springer, 2013.

[BBJM12] Patricia Bouyer, Thomas Brihaye, Marcin Jurdzinski, and Quentin Menet. Almost-sure model-checking of reactive timed automata. In *Proceedings of the 9th International Conference on Quantitative Evaluation of Systems (QEST'12)*, pages 138–147. IEEE Computer Society Press, 2012.

[BBL08] Patricia Bouyer, Ed Brinksma, and Kim G. Larsen. Optimal infinite scheduling for multi-priced timed automata. *Formal Methods in System Design*, 32(1):2–23, 2008.

[BBS06] Christel Baier, Nathalie Bertrand, and Philippe Schnoebelen. A note on the attractor-property of infinite-state Markov chains. *Information Processing Letters*, 97(2):58–63, 2006.

[BCD05] Patricia Bouyer, Fabrice Chevalier, and Deepak D'Souza. Fault diagnosis using timed automata. In *Proceedings of the 8th International Conference on Foundations of Software Science and Computational Structures (FOSSACS'05)*, volume 3441 of *Lecture Notes in Computer Science*, pages 219–233. Springer, 2005.

[BD08] Dietmar Berwanger and Laurent Doyen. On the power of imperfect information. In *Proceedings of 28th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'08)*, volume 2 of *Leibniz International Proceedings in Informatics*, pages 73–82. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2008.

[BDL+12] Peter E. Bulychev, Alexandre David, Kim Guldstrand Larsen, Axel Legay, Guangyuan Li, Danny Bøgsted Poulsen, and Amélie Stainer. Monitor-based statistical model checking for weighted metric temporal logic. In *18th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'18)*, volume 7180 of *Lecture Notes in Computer Science*, pages 168–182. Springer, 2012.

[BF13] Nathalie Bertrand and Paulin Fournier. Parameterized verification of many identical probabilistic timed processes. In *Proceedings of the 33rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'13)*, volume 24 of *Leibniz International Proceedings in Informatics*, pages 501–513. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.

[BFH+01] Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim G. Larsen, Paul Pettersson, Judi Romijn, and Frits Vaandrager. Minimum-cost reachability for priced timed automata. In *Proceedings of the 4th International Workshop on Hybrid Systems: Computation and Control (HSCC'01)*, volume 2034 of *Lecture Notes in Computer Science*, pages 147–161. Springer, 2001.

[BFH+14] Nathalie Bertrand, Éric Fabre, Stefan Haar, Serge Haddad, and Loïc Hélouët. Active diagnosis for probabilistic systems. In *Proceedings of the 17th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'14)*, volume 8412 of *Lecture Notes in Computer Science*, pages 29–42. Springer, 2014.

[BFLM11] Patricia Bouyer, Uli Fahrenberg, Kim G. Larsen, and Nicolas Markey. Quantitative analysis of real-time systems using priced timed automata. *Communication of the ACM*, 54(9):78–87, 2011.

[BFS14] Nathalie Bertrand, Paulin Fournier, and Arnaud Sangnier. Playing with probabilities in reconfigurable broadcast networks. In *Proceedings of the 17th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'14)*, volume 8412 of *Lecture Notes in Computer Science*, pages 134–148. Springer, 2014.

[BFS15] Nathalie Bertrand, Paulin Fournier, and Arnaud Sangnier. Distributed local strategies in broadcast networks. In *Proceedings of the 26th International Conference on Concurrency Theory (CONCUR'15)*, volume 42 of *Leibniz International Proceedings in Informatics*, pages 44–57. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[BG05] Christel Baier and Marcus Größer. Recognizing omega-regular languages with probabilistic automata. In *Proceedings of the 20th IEEE Symposium on Logic in Computer Science (LICS'05)*, pages 137–146. IEEE Computer Society, 2005.

[BG11] Nathalie Bertrand and Blaise Genest. Minimal disclosure in partially observable Markov decision processes. In *Proceedings of the 31st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'11)*, volume 13 of *Leibniz International Proceedings in Informatics*, pages 411–422. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

[BGG09] Nathalie Bertrand, Blaise Genest, and Hugo Gimbert. Qualitative determinacy and decidability of stochastic games with signals. In *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science (LICS'09)*, pages 319–328. IEEE Computer Society Press, 2009.

[BH15] Nathalie Bertrand and Serge Haddad. Contrôle, probabilités et observation partielle. In Nicolas Ollinger, editor, *Informatique Mathématique : Une photographie en 2015*, pages 177–227. CNRS éditions, 2015.

[BHHK03] Christel Baier, Boudewijn Haverkort, Holger Hermanns, and Joost-Pieter Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE Transactions on Software Engineering*, 29(7):524–541, 2003.

[BHL14] Nathalie Bertrand, Serge Haddad, and Engel Lefaucheux. Foundation of diagnosis and predictability in probabilistic systems. In *Proceedings of the 34th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'14)*, volume 29 of *Leibniz International Proceedings in Informatics*, pages 417–429. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014. Long version available at `http://www.lsv.ens-cachan.fr/Publis/RAPPORTS_LSV/PDF/rr-lsv-2014-09.pdf`.

[BJSK11] Nathalie Bertrand, Thierry Jéron, Amélie Stainer, and Moez Krichen. Off-line test selection with test purposes for non-deterministic timed automata. In *Proceedings of the 17th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'11)*, volume 6605 of *Lecture Notes in Computer Science*, pages 96–111. Springer, 2011.

[BJSK12] Nathalie Bertrand, Thierry Jéron, Amélie Stainer, and Moez Krichen. Off-line test selection with test purposes for non-deterministic timed automata. *Logical Methods in Computer Science*, 8(4:8), 2012.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of model checking.* MIT Press, 2008.

[BLPR12] Nathalie Bertrand, Axel Legay, Sophie Pinchinat, and Jean-Baptiste Raclet. Modal event-clock specifications for timed component-based design. *Science of Computer Programming*, 77:1212–1234, 2012.

[BMS13] Patricia Bouyer, Nicolas Markey, and Ocan Sankur. Robustness in timed automata. In *Proceedings of the 7th Workshop on Reachability Problems in Computational Models (RP'13)*, volume 8169 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2013.

[BPR09] Nathalie Bertrand, Sophie Pinchinat, and Jean-Baptiste Raclet. Refinement and consistency of timed modal specifications. In *Proceedings of the 3rd International Conference on Language and Automata Theory and Applications (LATA'09)*, volume 5457 of *Lecture Notes in Computer Science*, pages 152–163. Springer, 2009.

[BPSV05] Giacomo Bucci, Riccardo Piovosi, Luigi Sassoli, and Enrico Vicario. Introducing probability within state class analysis of dense-time-dependent systems. In *Proceedings of the Second International Conference on the Quantitative Evaluation of Systems (QEST'05)*, pages 13–22. IEEE Computer Society, 2005.

[BS84] Bruce G. Buchanan and Edward H. Shortliffe. *Rule Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project.* Addison-Wesley, 1984.

[BS03] Nathalie Bertrand and Philippe Schnoebelen. Model checking lossy channels systems is probably decidable. In *Proceedings of the 6th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'03)*, volume 2620 of *Lecture Notes in Computer Science*, pages 120–135. Springer, 2003.

[BS12] Nathalie Bertrand and Sven Schewe. Playing optimally on timed automata with random delays. In *Proceedings of the 10th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'12)*, volume 7595 of *Lecture Notes in Computer Science*, pages 43–58. Springer, 2012.

[BS13a] Nathalie Bertrand and Philippe Schnoebelen. Computable fixpoints in well-structured symbolic model checking. *Formal Methods in System Design*, 43(2):233–267, 2013.

[BS13b] Nathalie Bertrand and Philippe Schnoebelen. Solving stochastic büchi games on infinite arenas with a finite attractor. In *Proceedings of the 11th International Workshop on Quantitative Aspects of Programming Languages and Systems (QAPL'13)*, volume 117 of *Electronic Proceedings in Theoretical Computer Science*, pages 116–131, 2013.

[BSJK11] Nathalie Bertrand, Amélie Stainer, Thierry Jéron, and Moez Krichen. A game approach to determinize timed automata. In *Proceedings of the 14th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'11)*, volume 6604 of *Lecture Notes in Computer Science*, pages 245–259. Springer, 2011.

[BSJK15] Nathalie Bertrand, Amélie Stainer, Thierry Jéron, and Moez Krichen. A game approach to determinize timed automata. *Formal Methods in System Design*, 46(1):42–80, 2015.

[CD12] Krishnendu Chatterjee and Laurent Doyen. Partial-observation stochastic games: How to win when belief fails. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science (LICS'12)*, pages 175–184. IEEE, 2012.

[CdAFL09] Krishnendu Chatterjee, Luca de Alfaro, Marco Faella, and Axel Legay. Qualitative logics and equivalences for probabilistic systems. *Logical Methods in Computer Science*, 5(2), 2009.

[CDGH10] Krishnendu Chatterjee, Laurent Doyen, Hugo Gimbert, and Thomas A. Henzinger. Randomness for free. In *Proceedings of the 35th International Symposium on Mathematical Foundations of Computer Science 2010 (MFCS'10)*, volume 6281 of *Lecture Notes in Computer Science*, pages 246–257. Springer, 2010.

[CDH13] Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. A survey of partial-observation stochastic parity games. *Formal Methods in System Design*, 43(2):268–284, 2013.

[CDHR07] Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, and Jean-François Raskin. Algorithms for omega-regular games with imperfect information. *Logical Methods in Computer Science*, 3(3:4):1–23, 2007.

[CHKM11] Taolue Chen, Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre. Model-checking of continuous-time Markov chains against timed automata specifications. *Logical Methods in Computer Science*, 7(1:12):1–34, 2011.

[CK13] Jun Chen and Ratnesh Kumar. Polynomial test for stochastic diagnosability of discrete-event systems. *IEEE Transactions on Automation Science and Engineering*, 10(4):969–979, 2013.

[CK14] Jun Chen and Ratnesh Kumar. Failure prognosability of stochastic discrete event systems. In *Proceedings of the American Control Conference (ACC'14)*, pages 2041–2046. IEEE, 2014.

[CSV09a] Rohit Chadha, A. Prasad Sistla, and Mahesh Viswanathan. On the expressiveness and complexity of randomization in finite state monitors. *Journal of the ACM*, 56(5), 2009.

[CSV09b] Rohit Chadha, A. Prasad Sistla, and Mahesh Viswanathan. Power of randomization in automata on infinite strings. In *Proceedings of the 20th International Conference on Concurrency Theory (CONCUR'09)*, volume 5710 of *Lecture Notes in Computer Science*, pages 229–243. Springer, 2009.

[CSVB15] Rohit Chadha, A. Prasad Sistla, Mahesh Viswanathan, and Yue Ben. Decidable and expressive classes of probabilistic automata. In *Proceedings of the 18th International Conference onFoundations of Software Science and Computation Structures (FoSSaCS'15)*, volume 9034 of *Lecture Notes in Computer Science*, pages 200–214. Springer, 2015.

[dAH00] Luca de Alfaro and Thomas A. Henzinger. Concurrent omega-regular games. In *Proceedings of the 15th Annual IEEE Symposium on Logic in Computer Science (LICS'00)*, pages 141–154. IEEE Computer Society, 2000.

[DHKK01] Pedro D'Argenio, Holger Hermanns, Joost-Pieter Katoen, and Ric Klaren. MoDeST - A modelling and description language for stochastic timed systems. In *Joint International Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification (PAPM-PROBMIV'01)*, volume 2165 of *Lecture Notes in Computer Science*, pages 87–104. Springer, 2001.

[DHS09] Susanna Donatelli, Serge Haddad, and Jeremy Sproston. Model checking timed and stochastic properties with CSL^{TA}. *IEEE Transactions on Software Engineering*, 35(2):224–240, 2009.

[DLL+10] Alexandre David, Kim G. Larsen, Axel Legay, Ulrik Nyman, and Andrzej Wasowski. Timed I/O automata: a complete specification theory for real-time systems. In *Proceedings 13th ACM international conference on Hybrid systems: computation and control (HSCC'10)*, pages 91–100. ACM, 2010.

[DSTZ12] Giorgio Delzanno, Arnaud Sangnier, Riccardo Traverso, and Gianluigi Zavattaro. On the complexity of parameterized reachability in reconfigurable broadcast networks. In *Proceedings of the 32nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'12)*, volume 18 of *Leibniz International Proceedings in Informatics*, pages 289–300. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

[DSZ10] Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. Parameterized verification of ad hoc networks. In *Proceedings of the 21st International Conference on Concurrency Theory (CONCUR'10)*, volume 6269 of *Lecture Notes in Computer Science*, pages 313–327. Springer, 2010.

[DSZ11a] Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. On the power of cliques in the parameterized verification of ad hoc networks. In *Proceedings of the 14th International Conference on Foundations of Software Science and Computational Structures (FoSSaCS'11)*, volume 6604 of *Lecture Notes in Computer Science*, pages 441–455. Springer, 2011.

[DSZ11b] Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. Parameterized verification of safety properties in ad hoc network protocols. In *Proceedings of the 1st International Workshop on Process Algebra and Coordination (PACO'11)*, volume 60 of *Electronic Proceedings in Theoretical Computer Science*, pages 56–65, 2011.

[DSZ12] Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. Verification of ad hoc networks with node and communication failures. In *Proceedings of the joint 14th IFIP WG 6.1 International Conference and 32nd IFIP WG 6.1 International Conference on Formal Techniques for Distributed Systems (FMOODS/FORTE'12)*, volume 7273 of *Lecture Notes in Computer Science*, pages 235–250. Springer, 2012.

[Fin06] Olivier Finkel. Undecidable problems about timed automata. In *Proceedings of the 4th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'06)*, volume 4202 of *Lecture Notes in Computer Science*, pages 187–199. Springer, 2006.

[Fru06] Matthias Fruth. Probabilistic model checking of contention resolution in the IEEE 802.15.4 low-rate wireless personal area network protocol. In *Proceedings of the 2nd International Symposium on Leveraging Applications of Formal Methods (IsoLA'06)*, pages 290–297. IEEE, 2006.

[GO10] Hugo Gimbert and Youssouf Oualhadj. Probabilistic automata on finite words: Decidable and undecidable problems. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP'10)*, volume 6199 of *Lecture Notes in Computer Science*, pages 527–538. Springer, 2010.

[GS09] Vincent Gripon and Olivier Serre. Qualitative concurrent stochastic games with imperfect information. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP'09)*, volume 5556 of *Lecture Notes in Computer Science*, pages 200–211. Springer, 2009.

[GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.

[Had13] Serge Haddad, April 2013. Personal communication.

[HHMS13] Stefan Haar, Serge Haddad, Tarek Melliti, and Stefan Schwoon. Optimal constructions for active diagnosis. In *Proceedings of the 33rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'13)*, volume 24 of *Leibniz International Proceedings in Informatics*, pages 527–539. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.

[JHCK01] Shengbing Jiang, Zhongdong Huang, Vigyan Chandra, and Ratnesh Kumar. A polynomial algorithm for testing diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 46(8):1318–1321, 2001.

[KNP11] Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: verification of probabilistic real-time systems. In *Proceedings of the 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *Lecture Notes in Computer Science*, pages 585–591. Springer, 2011.

[KNPS08] Marta Kwiatkowska, Gethin Norman, David Parker, and Jeremy 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.

[KNSS00] Marta Kwiatkowska, Gethin Norman, Roberto Segala, and Jeremy Sproston. Verifying quantitative properties of continuous probabilistic timed automata. In *Proceedings of the 11th International Conference on Concurrency Theory (CONCUR'00)*, volume 1877 of *Lecture Notes in Computer Science*, pages 123–137. Springer, 2000.

[KNSS02] Marta Kwiatkowska, Gethin Norman, Roberto Segala, and Jeremy Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theoretical Computer Science*, 282(1):101–150, 2002.

[KS88] S. Rao Kosaraju and Gregory F. Sullivan. Detecting cycles in dynamic graphs in polynomial time (preliminary version). In *STOC'88*, pages 398–406. ACM, 1988.

[KT09] Moez Krichen and Stavros Tripakis. Conformance testing for real-time systems. *Formal Methods in System Design*, 34(3):238–304, 2009.

[KZH+09] Joost-Pieter Katoen, Ivan S. Zapreev, Ernst Moritz Hahn, Holger Hermanns, and David N. Jansen. The Ins and Outs of The Probabilistic Model Checker MRMC. In *Proceedings of the 6th International Conference on the Quantitative Evaluation of Systems Quantitative (QEST'09)*, pages 167–176. IEEE Computer Society, 2009.

[LGM01] Christopher Lusena, Judy Goldsmith, and Martin Mundhenk. Nonapproximability results for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 14, 2001.

[LMS04] François Laroussinie, Nicolas Markey, and Philippe Schnoebelen. Model checking timed automata with one or two clocks. In *Proceedings of the 15th International Conference on Concurrency Theory (CONCUR'04)*, volume 3170 of *Lecture Notes in Computer Science*, pages 387–401. Springer, 2004.

[LR81] Daniel J. Lehmann and Michael O. Rabin. On the advantages of free choice: a symmetric and fully distributed solution to the dining philosophers problem. In *Proceedings of the 8th ACM Symposium on Principles of Programming Languages (POPL'81)*, pages 133–138. ACM Press, 1981.

[MHC03] Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence*, 147(1-2):5–34, 2003.

[Min67] Marvin Minsky. *Computation: Finite and Infinite Machines*. Prentice Hall International, 1967.

[MK10] Lakshmi Manasa and Shankara Narayanan Krishna. Integer reset timed automata: Clock reduction and determinizability. CoRR arXiv:1001.1215v1, 2010.

[NS03] Brian Nielsen and Arne Skou. Automated test generation from timed automata. *International Journal on Software Tools for Technology Transfer*, 5:59–77, 2003.

[Par64] William Parry. Intrinsic Markov chains. *Transactions of the American Mathematical Society*, 112:55–66, 1964.

[Paz71] Azaria Paz. *Introduction to probabilistic automata*. Academic Press, 1971.

[PN97] S. Purushothaman Iyer and M. Narasimha. Probabilistic lossy channel systems. In *Proceedings of the 7th International Joint Conference on Theory and Practice of Software Development (TAPSOFT'97)*, volume 1214 of *Lecture Notes in Computer Science*, pages 667–681. Springer, 1997.

[PR90] Amir Pnueli and Roni Rosner. Distributed reactive systems are hard to synthesize. In *31st Annual Symposium on Foundations of Computer Science (FOCS'90)*, pages 746–757. IEEE Computer Society, 1990.

[PT87] C. Papadimitriou and J. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[Pur00] Anuj Puri. Dynamical properties of timed automata. *Discrete Event Dynamic Systems*, 10(1-2):87–113, 2000.

[Rab63] Michael O. Rabin. Probabilistic automata. *Information and Control*, 6(3):230–245, 1963.

[Rab03] Alexander Rabinovich. Quantitative analysis of probabilistic lossy channel systems. In *Proceedings of the 30th International Colloquium on Automata, Languages, and Programming (ICALP'03)*, volume 2719 of *Lecture Notes in Computer Science*, pages 1008–1021. Springer, 2003.

[Rei84] John H. Reif. The complexity of two-player games of incomplete information. *Journal of Computer and System Sciences*, 29(2):274–301, 1984.

[Ren07] Jérôme Renault. The value of repeated games with an informed controller. Technical report, CEREMADE, Paris, January 2007.

[RS11] Markus N. Rabe and Sven Schewe. Finite optimal control for time-bounded reachability in CTMDPs and continuous-time Markov games. *Acta Informatica*, 48(5-6):291–315, 2011.

[RS14] Jean-François Raskin and Ocan Sankur. Multiple-environment Markov decision processes. In *Proceedings of the 34th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS'14)*, volume 29 of *Leibniz International Proceedings in Informatics*, pages 531–543. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014.

[RSV03] Dinah Rosenberg, Eilon Solan, and Nicolas Vieille. Stochastic games with imperfect monitoring. Technical Report 1376, Northwestern University, July 2003.

[RW87] Peter J. Ramadge and W. Murray Wonham. Supervisory control of a class of discrete-event processes. *SIAM Journal of Control and Optimization*, 25(1):206–230, 1987.

[Saf88] Shmuel Safra. On the complexity of ω-automata. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science (FOCS'88)*, pages 319–327. IEEE Computer Society Press, 1988.

[SBMR13] Ocan Sankur, Patricia Bouyer, Nicolas Markey, and Pierre-Alain Reynier. Robust controller synthesis in timed automata. In *Proceedings of the 24th International Conference on Concurrency Theory (CONCUR'13)*, volume 8052 of *Lecture Notes in Computer Science*, pages 546–560. Springer, 2013.

[Sch02] Philippe Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity. *Information Processing Letters*, 83(5):251–261, 2002.

[Sha48] Claude Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[SLDP09] Jun Sun, Yang Liu, Jin Song Dong, and Jun Pang. PAT: towards flexible verification under fairness. In *21st International Conference on Computer Aided Verification (CAV'09)*, volume 5643 of *Lecture Notes in Computer Science*, pages 709–714. Springer, 2009.

[SLT98] Meera Sampath, Stéphane Lafortune, and Demosthenis Teneketzis. Active diagnosis of discrete-event systems. *IEEE Transactions on Automatic Control*, 43(7):908–929, 1998.

[Son71] E. J. Sondik. *The Optimal Control of Partially Observable Markov Processes.* PhD thesis, Stanford University, 1971.

[Sor02] Sylvain Sorin. *A first course on zero-sum repeated games.* Springer, 2002.

[SPKM08] P. Vijay Suman, Paritosh K. Pandya, Shankara Narayanan Krishna, and Lakshmi Manasa. Timed automata with integer resets: Language inclusion and expressiveness. In *Proceedings of the 6th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'08)*, volume 5215 of *Lecture Notes in Computer Science*, pages 78–92. Springer, 2008.

[SS12] Sylvain Schmitz and Philippe Schnoebelen. Algorithmic aspects of WQO theory. Lecture Notes, 2012.

[SSL+95] Meera Sampath, Raja Sengupta, Stéphane Lafortune, Kasim Sinnamohideen, and Demosthenis Teneketzis. Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, 1995.

[ST08] Julien Schmaltz and Jan Tretmans. On conformance testing for timed systems. In *Proceedings of the 6th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'08)*, volume 5215 of *Lecture Notes in Computer Science*, pages 250–264, 2008.

[Sta12] Amélie Stainer. Frequencies in forgetful timed automata. In *Proceedings of the 10th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'12)*, volume 7595 of *Lecture Notes in Computer Science*, pages 236–251. Springer, 2012.

[Sto03] Mariëlle Stoelinga. Fun with FireWire: A comparative study of formal verification methods applied to the IEEE 1394 root contention protocol. *Formal Aspects of Computing*, 14(3):328–337, 2003.

[SV89] Shmuel Safra and Moshe Y. Vardi. On ω-automata and temporal logic. In *Proceedings of the 21st ACM Symposium on Theory of Computing (STOC'89)*, pages 127–137. ACM, 1989.

[Szn13] Tali Sznajder, December 2013. Personal communication.

[Tra11] Mathieu Tracol. Recurrence and transcience for finite probabilistic tables. *Theoretical Computer Science*, 412(12-14):1154–1168, 2011.

[Tre96] Jan Tretmans. Test generation with inputs, outputs, and quiescence. In *Proc. of the second International workshop on Tools and Algorithms for Construction and Analysis of Systems (TACAS'96)*, volume 1055 of *Lecture Notes in Computer Science*, pages 127–146. Springer, 1996.

[Tri06] Stavros Tripakis. Folk theorems on the determinization and minimization of timed automata. *Information Processing Letters*, 99(6):222–226, 2006.

[TT05] David Thorsley and Demosthenis Teneketzis. Diagnosability of stochastic discrete-event systems. *IEEE Transactions on Automatic Control*, 50(4):476–492, 2005.

[WJ06] Nicolás Wolovick and Sven Johr. A characterization of meaningful schedulers for continuous-time Markov decision processes. In *Proceedings of the 4th International Conference Formal Modeling and Analysis of Timed Systems (FORMATS'06)*, volume 4202 of *Lecture Notes in Computer Science*, pages 352–367. Springer, 2006.

[WSL+14] Ting Wang, Jun Sun, Yang Liu, Xinyu Wang, and Shanping Li. Are timed automata bad for a specification language? Language inclusion checking for timed automata. In *Proceedings of the 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'14)*, volume 8413 of *Lecture Notes in Computer Science*, pages 310–325. Springer, 2014.

[ZJNH11] Lijun Zhang, David N. Jansen, Flemming Nielson, and Holger Hermanns. Automata-based CSL model checking. In *Proceedings of the 38th International Colloquium on Automata, Languages and Programming (ICALP'11)*, volume 6756 of *Lecture Notes in Computer Science*, pages 271–282. Springer, 2011.