Open access • Journal Article • DOI:10.1093/JIGPAL/JZR054

Decidability and complexity for ω-regular properties of stochastic systems

— Source link

David Henriques, Manuel Biscaia, Pedro Baltazar, Paulo Mateus

Institutions: Technical University of Lisbon

Published on: 01 Dec 2012 - Logic Journal of The Igpl \/ Bulletin of The Igpl (Oxford University Press)

Topics: EXPSPACE, Fragment (logic), Linear temporal logic, Decidability and PSPACE-complete

Decidability and Complexity for ω-Regular Properties of Stochastic Systems

D. Henriques M. Biscaia P. Baltazar P. Mateus

SQIG - Instituto de Telecomunicações
Department of Mathematics, IST - TU Lisbon

Abstract

Herein we study the probabilization of Quantified Linear Temporal Logic, which we call PQLTL. PQLTL can reason about any semialgebraic constrain over probabilities of paths of a Markov chain satisfying a QLTL formulae. PQLTL is related with other commonly used probabilistic temporal logics (such as PLTL, PCTL and PCTL*) that were devised only to specify properties for which model checking algorithms are amenable and whose basic results, such as completeness and decidability, were never investigated.

In this paper, we devise a strong and a weak SAT algorithm for PQLTL. The former relies in [n+2]-EXPSPACE and the latter in [n+1]-EXPSPACE where n is the alternation depth of the quantifiers in the input formula. The weak SAT algorithm for the existential fragment, which has all the expressive power of PQLTL, is in EXPSPACE. Another relevant fragment of PQLTL is the linear closure of PCTL* without nesting of the probability operator. We show that the SAT problem for this fragment is PSPACE complete. Capitalizing in these results, we derive a complete calculus for PQLTL and illustrate it with a toy example.

1 Introduction

Temporal logics, such as CTL, LTL and CTL*, are widely used to reason about distributed and dynamic systems, with multiple applications to diverse fields such as software and hardware verification [6, 11, 20], biological systems [4], or even Philosophy [16]. Despite their expressiveness, these logics are not suited for quantitative reasoning about common probabilistic systems, which has been an active research subject. Because of this, alternative semantics over Markov processes have been proposed and thoroughly studied. Logics like PLTL [12, 26, 27], PCTL or PCTL* [1, 10, 15], have emerged and are nowadays commonly used for these purposes. These logics have been used more as a specification language than as a proper "logic", as they are only used to specify properties intended to be model-checked against some Markov model. As far as the authors know, the completeness and satisfiability problems for these logics were never studied. In this paper we investigate these questions, although acknowledging that the problems we are tackling have less applications than model checking.

We introduce a logic to reason about probabilities of paths of a Markov chain, following the line of research on probabilistic temporal logic (PLTL) introduced in [26, 27]. We consider two enrichments to PLTL. First, we deal with probabilities of sets of paths that can be expressed in quantified linear temporal logic (QLTL). Secondly, we consider semialgebraic sets over probabilized QLTL formulae (i.e., the set of polynomial inequations formed by probabilistic QLTL expressions, algebraic real numbers and variables). We call this language *Probabilistic Quantified Linear Temporal Logic* (PQLTL).

We note that other probabilistic logics (PCTL, PCTL*) allow higher-order probabilities (events that are defined by probabilistic assertions) but these features have shown lack of intuition and applicability, while still not addressing simple issues regarding lack of expressiveness. Our logic, while not allowing for this nesting of probability operators, is still rich enough to specify very relevant properties, such as probabilistic fairness or almost-sure termination but also many previously disregarded requests easily expressible in natural language.

For example, while requests of the form "with probability of 99%, the process eventually reaching the critical state implies that a flag will always be raised" can easily be expressed in PCTL, other reasonable requests such as "The probability of the process entering region 5 is at least double the probability of entering regions 1 to 4" or "The probability of reaching Undecided is at least 10% less than reaching either Accept or Reject" are just not expressible in this logic (or any other logic the authors are aware of). PQLTL is able to express this kind of assertions.

Moreover, the introduction of quantification over propositional symbols allows reasoning about events that only happen with some periodicity. This is a very useful feature when dealing with Markov chains. Indeed, the limit behavior of aperiodic chains is well known, but limit properties of periodic Markov chains are hard to study. By considering only transitions constant modulo the period of the chain, we can eventually reduce a periodic chain to a set of aperiodic subchains, which we know how to handle.

We derive algorithms for the decidability of PQLTL, and a complete

Hilbert calculus. The algorithms are presented in two different versions: a weak SAT algorithm that just decides whether a certain formula is satisfiable, and a strong SAT algorithm that provides a model for a given satisfiable formula.

Concerning the SAT algorithm, the strong version relies in [n+2]-EXPSPACE whereas the weak version is in [n+1]-EXPSPACE, where n is the alternation depth of the quantifiers in the input formula. The weak SAT algorithm for the existential fragment, which has all the expressive power of PQLTL, is in EXPSPACE.

While PQLTL has theoretically interesting properties, the SAT algorithm proposed is very hard, which severely limits its application. However, if we restrict ourselves to *linear* inequalities without quantification, we still have relevant quantitative expressiveness while significantly reducing the requirements of the SAT algorithm. We call this more "practical" logic by PLTL^+. Indeed, we show that the SAT problem for PLTL^+ is PSPACE-complete, which may be surprising, since the SAT algorithm for simple non-probabilistic LTL is already in this complexity class. Moreover, PCTL^* without nesting of the probability operator is a sublanguage of PLTL^+.

Capitalizing in the SAT algorithms, we derive a weakly complete calculus for PQLTL, and consequently for all the sublogics considered. We illustrate the calculus with a toy example using a PONG$^{\mathrm{TM}}$ game. We leave less academic applications of the calculus for future work.

We also present a model-checking algorithm, however the algorithm is straightforwardly obtained by applying well-know reduction of QLTL formulae to deterministic Rabin automata together with the SAT algorithm for the existential theory of the real numbers. This result is a simple generalization of an already existing automata-theoretical algorithm for PCTL^* [10], and so it is only shown in Appendix, for the sake of completeness.

The structure of the paper is the following; in Section 2 we introduce PQLTL, namely by providing its syntax and semantics. Both SAT algorithms for the logic are introduced in Section 3. In Section 4, we present the sublogic containing only linear inequalities for which we obtain a PSPACE SAT algorithm. The complete calculi for all the logics considered are developed in Section 5, where we also illustrate its use with a simple example. Finally, we draw some conclusions and future work.

2 Probabilization of quantified linear temporal logic

2.1 PQLTL Syntax

The construction of PQLTL is the following: a set of formulae is taken at a base level - *basic formulae* - and another set is built over it at an higher level - *global formulae*. A set of *probabilistic terms* is also considered. The syntax is described in Table 1 by mutual recursion.

$$\begin{array}{ll}
\beta := p \ [\!]\ (\neg\beta) \ [\!]\ (\beta \Rightarrow \beta) \ [\!]\ \mathsf{X}\beta \ [\!]\ \beta\mathsf{U}\beta \ [\!]\ \exists p.\beta & \text{basic formulae} \\
t := z \ [\!]\ 0 \ [\!]\ 1 \ [\!]\ \int\beta \ [\!]\ (t + t) \ [\!]\ (t \cdot t) & \text{probabilistic terms} \\
\delta := (t \leq t) \ [\!]\ (\sim\delta) \ [\!]\ (\delta \supset \delta) & \text{global formulae}
\end{array}$$

where $p \in \Lambda$, $z \in Z$.

Table 1: PQLTL syntax

Basic formulae are simply QLTL formulae over a finite set Λ of propositional symbols, allowing for classical quantified temporal reasoning over them. The usual abbreviations for falsum $\bot$, disjunction $(\beta_1 \vee \beta_2)$, conjunction $(\beta_1 \wedge \beta_2)$, equivalence $(\beta_1 \Leftrightarrow \beta_2)$ and universal quantification $\forall p.\beta$, as well as for **f**uture $(\mathsf{F}\beta)$ and **g**lobally $(\mathsf{G}\beta)$ are henceforth used freely.

Probabilistic terms permit quantitative reasoning over the set of algebraic real numbers by introducing a set of algebraic real variables Z which, together with addition, multiplication, 0, 1 and the equality relation of global formulae, allow the representation of any algebraic real number. *Measure terms*, terms of the form $(\int \beta)$ denote the probability of satisfying β.

Global formulae are built by taking comparison formulae $(t_1 \leq t_2)$ as *atoms* and building an analog of the propositional language over them. As in the basic case, we will assume the analogs of usual abbreviations for global falsum $\mathbf{f}$, global disjunction $(\delta_1 \cup \delta_2)$, global conjunction $(\delta_1 \cap \delta_2)$ and global equivalence $(\delta_1 \equiv\ \delta_2)$. The comparison operators $\{=, \neq, \geq, <, >\}$ will also be used as usual.

When no ambiguity arises, we shall drop the parenthesis.

2.2 PQLTL Semantics

In order to define the semantics for PQLTL, we need to build upon the semantics for QLTL. These semantics are detailed in the Appendix and follow the presentation by Sistla, Vardi and Wolper in [24]. From this point on, we shall assume that all QLTL formulae are given in prefix normal form, as in Proposition A.9 in the Appendix, page 30.

The models of PQLTL are pairs, where the first element is a discrete time Markov chain with states labeled by valuations over Λ and the second element is an assignment over Z to the set of algebraic real numbers. In the model-checking community, it is common to call these labeled Markov chains by *probabilistic deterministic transitions systems* (PDTS) and we will adopt this custom. We follow [2] for the formal definition; a PDTS is a tuple $\mathcal{M} = (\mathsf{S}, d_0, \mathcal{T}, \mathsf{L})$ where:

- S is a finite set of *states*, $\mathsf{S} = \{s_1, s_2, ..., s_N\}$;
- d_0 is the *initial distribution*, a probability distribution over S;
- $\mathcal{T}$ is the *probabilistic transition function* which assigns a probability to each transition between states, that is, $\mathcal{T} : \mathsf{S} \times \mathsf{S} \to [0, 1]$, s.t.

$$\forall_{s \in \mathsf{S}} \sum_{s' \in \mathsf{S}} \mathcal{T}(s, s') = 1.$$

- L is the *labeling function*, assigning a valuation over Λ to each state.

To avoid technical complications, we assume $\mathcal{T}$ to be total (possibly enriching it with pairs with probability zero).

There is a uniquely induced probability measure for each PDTS $\mathcal{M}$ and state s_0 over sets of paths that depart from s_0. The measure is defined over the sets of all paths departing from s_0 with common prefixes (*cylinders*):

$$\mu_{\mathcal{M}, s_0}(\{\pi : \pi|_k = s_0 s_1 ... s_k\}) = \mathcal{T}(s_0, s_1) \times \mathcal{T}(s_1, s_2) \times ... \times \mathcal{T}(s_{k-1}, s_k),$$

with the base case of all paths departing from s_0 having measure set to 1 (since $\mu_{\mathcal{M}, s_0}$ is σ-additive, this is enough to fully define the measure [17]). When the context is evident and no ambiguity arises, we shall drop the subscript $\mathcal{M}$.

Given a model $(\mathcal{M} = (\mathsf{S}, d_0, \mathcal{T}, \mathsf{L}), \rho : Z \to \mathbb{R})$, the denotation of probabilistic terms is as follows:

- $[\![z]\!]_{\mathcal{M}, \rho} = \rho(z)$; $[\![0]\!]_{\mathcal{M}, \rho} = 0$; $[\![1]\!]_{\mathcal{M}, \rho} = 1$;
- $[\![t_1 + t_2]\!]_{\mathcal{M}, \rho} = [\![t_1]\!]_{\mathcal{M}, \rho} + [\![t_2]\!]_{\mathcal{M}, \rho}$; $[\![t_1 . t_2]\!]_{\mathcal{M}, \rho} = [\![t_1]\!]_{\mathcal{M}, \rho} . [\![t_2]\!]_{\mathcal{M}, \rho}$; and
- $[\![\int \beta]\!]_{\mathcal{M}, \rho} = \sum_{s_i \in \mathsf{S}} d_0(s_i) \mu_{s_i}(\{\pi : (\mathsf{S}, \pi, \mathsf{L}) \Vdash_{\mathsf{QLTL}} \beta, \pi\,[0] = s_i\})$.

We must be cautious regarding the measurability of the sets $\{\pi : (\mathsf{S}, \pi, \mathsf{L}) \Vdash_{\mathsf{QLTL}} \beta, \pi[0] = s_i\}$. In [22, 24], it is shown that QLTL is exactly as expressive as non-deterministic Büchi automata, which means that for each QLTL formula β, there is a Büchi automaton over the alphabet $\{0,1\}^{\Lambda}$, B_β such that $\{\mathsf{L}(\pi) : (\mathsf{S}, \pi, \mathsf{L}) \Vdash_{\mathsf{QLTL}} \beta\} = L^{\omega}(B_\beta)$. Furthermore, in [26] (see Proposition A.16 in the Appendix, page 32), it is shown that the set of paths whose associated sequence of valuations is accepted by B_β is measureable, and so the sets $\{\pi : (\mathsf{S}, \pi, \mathsf{L}) \Vdash_{\mathsf{QLTL}} \beta, \pi[0] = s_i\}$ are measureable.

Since the denotation of terms of the form $\int\beta$ does not depend on the assignment ρ, we will drop it from the denotation in some statements. It will always be implied, in these cases, that the assertion is true for any assignment ρ. Moreover, the satisfaction of global formulae is given by:

- $\mathcal{M}, \rho \Vdash_{\mathsf{PQLTL}} (t_1 \leq t_2)$ iff $[\![t_1]\!]_{\mathcal{M},\rho} \leq [\![t_2]\!]_{\mathcal{M},\rho}$;
- $\mathcal{M}, \rho \Vdash_{\mathsf{PQLTL}} (\sim\delta)$ iff $\mathcal{M}, \rho \nVdash \delta$; and
- $\mathcal{M}, \rho \Vdash_{\mathsf{PQLTL}} (\delta_1 \supset \delta_2)$ iff $\mathcal{M}, \rho \Vdash \delta_2$ or $\mathcal{M}, \rho \nVdash \delta_1$.

Moreover, the notion of semantic entailment is introduced as usual: $\Gamma \models_{\mathsf{PQLTL}} \delta$ iff, for every model $(\mathcal{M}, \rho)$, $\mathcal{M}, \rho \Vdash_{\mathsf{PQLTL}} \delta$ whenever $\mathcal{M}, \rho \Vdash_{\mathsf{PQLTL}} \gamma$ for each $\gamma \in \Gamma$. Whenever it is clear from the context which logic we are interested in, we will drop the subscript.

Remark 2.1 *Due to Proposition A.13 (in Appendix, page 31), we know that we can rewrite any QLTL formula in order to obtain a formula in existential prenex normal form. This fact motivates the natural fragment of PQLTL, PEQLTL, where we directly assume that the measure terms are built using only formulae in existential prenex normal form. This syntactic fragment allows us to obtain much better complexity bounds in the presented algorithms.*

Lemma 2.2 Let $\beta, \beta_1, ..., \beta_m \in \mathsf{QLTL}$. For any PDTS $\mathcal{M}$, we have that

$$[\![\textstyle\int\beta]\!]_{\mathcal{M}} = [\![\int \bigvee_{i=0}^{2^m-1} (\beta \wedge (\bigwedge_{j=1}^{m} b_{ij}))]\!]_{\mathcal{M}} = \sum_{i=0}^{2^m-1} [\![\int (\beta \wedge (\bigwedge_{j=1}^{m} b_{ij}))]\!]_{\mathcal{M}} \qquad (1)$$

where $b_{ij} = \begin{cases} \beta_j & \text{if the } j\text{-th bit of } i \text{ is } 1, \\ (\neg\beta_j) & \text{otherwise.} \end{cases}$

Proof. For any $\beta_1, \beta_2 \in \mathsf{QLTL}$ the set of paths (in $\mathcal{M}$) that satisfy $\beta_1 \wedge \beta_2$ and the set of paths that satisfy $\beta_1 \wedge (\neg\beta_2)$ are disjoint and their union is the set of paths that satisfy β_1, we have that $[\![\int\beta_1]\!]_{\mathscr{M}} = [\![\int(\beta_1 \wedge \beta_2) \vee (\beta_1 \wedge (\neg\beta_2))]\!]_{\mathscr{M}} = [\![\int(\beta_1 \wedge \beta_2)]\!]_{\mathscr{M}} + [\![\int(\beta_1 \wedge (\neg\beta_2))]\!]_{\mathscr{M}}$ by additivity of the probability measure. It is a simple exercise in induction to generalize this result. △

Notice that the syntax and semantics strictly contain the PLTL informally presented in [26, 27]. In addition, they allow quantitative reasoning, namely through the use of variables and comparisons between terms. Readers familiar with probabilistic temporal logics may wonder about the lack of nesting of the probability operator like, for instance, in PCTL. It is the opinion of the authors that nesting of probabilities is unintuitive, and for the sake of simplicity, we have disregarded it in this work. Still, it must be noticed that this means that some assertions in PCTL cannot be expressed in PQLTL (and vice versa).

3 SAT algorithm for PQLTL

We now derive an algorithm for deciding the satisfiability problem for PQLTL. This algorithm, although computationally demanding for PQLTL formulae in general form, will be adapted to specific (but widely used) cases, resulting in comparatively efficient algorithms. We will develop two versions of the algorithm: one version will solve the classical SAT problem of determining if there is any model that satisfies the input formula; the other version will in fact provide a witness for the satisfiability. We will call these the *weak* and *strong* SAT problems, respectively.

Given a PQLTL global formula δ, let $gatm_\delta = \{a_1, ..., a_n\}$ be the subset of global atoms that occur in δ. Consider a countable set of propositional symbols (or local *atoms*) $\Xi = \{\xi_1, ...\xi_n, ...\}$ and an injective map λ that assigns a propositional symbol $\lambda(a_i) = \xi_i \in \Xi$ to each global atom in $gatm_\delta$; this function can be extended to map PQLTL formulae δ into propositional formulae λ_δ by simple structural induction. Let $V_{\lambda(gatm_\delta)}$ be the set of valuations over $\lambda(gatm_\delta)$, that we will identify with $\{0,1\}^n$ in the expected way. Clearly, we have that $\lambda(gatm(\delta)) = atm(\lambda_\delta)$.

We can use any classical SAT algorithm to check the satisfiability of λ_δ. If λ_δ is not satisfiable, then δ is also not satisfiable and we are done. Otherwise, let $mol(\lambda_\delta)$ (henceforth called the *molecules* of λ_δ) be the set of all $\Phi \subset atm(\lambda_\delta)$ such that the following propositional formula holds:

$$((\bigwedge_{\xi_i \in \Phi} \xi_i) \wedge (\bigwedge_{\xi_i \in atm(\lambda_\delta)\backslash\Phi} \neg\xi_i)) \Rightarrow \lambda_\delta. \qquad (2)$$

Then, each formula λ_δ is equivalent to

$$\bigvee_{\Phi \in mol(\lambda_\delta)} ((\bigwedge_{\xi_i \in \Phi} \xi_i) \wedge (\bigwedge_{\xi_i \in atm(\lambda_\delta) \setminus \Phi} \neg \xi_i)). \tag{3}$$

Which is just a syntactic characterization of the disjunctive normal form.

To prove satisfiability of δ, in addition to λ_δ being satisfiable, we must have that

$$\bigcup_{\Phi \in mol(\lambda_\delta)} ((\bigcap_{\xi_i \in \Phi} \lambda^{-1}(\xi_i)) \cap (\bigcap_{\xi_i \in atm(\lambda_\delta) \setminus \Phi} {\sim}\lambda^{-1}(\xi_i))). \tag{4}$$

must also be satisfiable. For each molecule $\Phi_k \in mol(\lambda_\delta)$, we will denote the expression

$$((\bigcap_{\xi_i \in \Phi} \lambda^{-1}(\xi_i)) \cap (\bigcap_{\xi_i \in atm(\lambda_\delta) \setminus \Phi} {\sim}\lambda^{-1}(\xi_i))). \tag{5}$$

by μ_k, and abusively call it the k-th molecule of δ.

Lemma 3.1 The PQLTL formula δ is satisfiable iff there exists $v \in V_{\lambda(gatm_\delta)}$ such that $v(\lambda_\delta) = 1$ and μ_v is satisfiable.

Proof. ($\Rightarrow$) Suppose δ is satisfiable. Let $(\mathcal{M}, \rho)$ be a model of δ and recall $gatm_\delta = \{a_1, .., a_n\}$. Consider

$$\mu = \bigwedge_{i=1}^{n} \alpha_i \qquad \text{where} \qquad \alpha_i = \begin{cases} a_i & \mathcal{M}, \rho \Vdash a_i, \\ ({\sim}a_i) & \text{otherwise.} \end{cases} \tag{6}$$

Then, obviously $\mathcal{M}, \rho \Vdash \mu$. Furthermore, this μ is associated with $v \in V_{\lambda(gatm_\delta)}$ s.t. $v(a_i) = 1$ iff $\mathcal{M}, \rho \Vdash a_i$, which is a witness for λ_δ.

($\Leftarrow$) Suppose μ_v is satisfiable and $v(\lambda_\delta) = 1$. Let $(\mathcal{M}, \rho)$ be a model of μ_v. Then $\mathcal{M}, \rho \Vdash a_i$ iff $v(\lambda(a_i)) = 1$, by induction in the structure of δ, $\mathcal{M}, \rho \Vdash \delta$:

- if $\delta = a \in gatm(\delta)$, then $v(\lambda_\delta) = v(\lambda(a)) = 1$ iff $\mathcal{M}, \rho \Vdash a$, that is $\mathcal{M}, \rho \Vdash \delta$;

- if $\delta = {\sim}\delta_1$, then $v(\lambda_\delta) = v(\lambda({\sim}\delta_1)) = v(\neg\lambda(\delta_1)) = 1 - v(\lambda(\delta_1)) = 1$ iff $v(\lambda(\delta_1)) = 0$ iff, by IH, $\mathcal{M}, \rho \nVdash \delta_1$ iff $\mathcal{M}, \rho \Vdash {\sim}\delta_1$, that is $\mathcal{M}, \rho \Vdash \delta$;

- if $\delta = \delta_1 \supset \delta_2$, then $v(\lambda_\delta) = v(\lambda(\delta_1 \supset \delta_2)) = v(\lambda(\delta_1) \Rightarrow \lambda(\delta_2)) = max[1 - v(\lambda(\delta_1)), \lambda(\delta_2)] = 1$ iff $v(\lambda(\delta_1)) = 0$ or $v(\lambda(\delta_1)) = 1$ iff, by IH, $\mathcal{M}, \rho \nVdash \delta_1$ or $\mathcal{M}, \rho \Vdash \delta_2$ iff $\mathcal{M}, \rho \Vdash \delta_1 \supset \delta_2$, that is $\mathcal{M}, \rho \Vdash \delta$.

$\triangle$

We will now propose a SAT algorithm for a molecule μ_k. If, for all k, this algorithm returns no model, then δ is not satisfiable. If it does return a model for some k, then that model also satisfies δ.

Before fixing a μ_k, consider the set of QLTL formulae that occur in δ that are not subformulae of other QLTL formula and denote it by $\Theta(\delta)$ (essentially, $\beta \in \Theta(\delta)$ iff $\int\beta$ is subterm of δ). Since $\Theta(\mu_k) = \Theta(\delta)$ for all k, so we will denote this set just by Θ. For the remaining, we fix an enumeration of Θ, $\beta_1, ..., \beta_l$, and denote the set $\{0,1\}^l$ by V_Θ.

Fix now a μ_k. We will consider, for each $\beta_i \in \Theta$, the disjunction of its "molecules" in Θ:

$$\beta_i \equiv \bigvee_{\substack{\sigma \in V_\Theta : \\ \sigma(\beta_i) = 1}} (\beta_i \land (\bigwedge_{\substack{\sigma(\beta_j) = 1, \\ j \neq i}} \beta_j \quad \land \bigwedge_{\substack{\sigma(\beta_j) = 0, \\ j \neq i}} \neg\beta_j)). \tag{7}$$

Where $\sigma(\beta_i) = 1$ if the i-th bit of σ is 1, $\sigma(\beta_i) = 0$ otherwise. Let us now assign, to each $\sigma \in V_\Theta$, an algebraic real variable $x_{\boldsymbol{\sigma}} \notin Z$ indexed by a binary integer $\boldsymbol{\sigma}$ identified with σ in the obvious way. Intuitively, this variable will take on the value of the denotation of its corresponding molecule of formulae of Θ in the output PDTS $\mathcal{M}$:

$$x_{\boldsymbol{\sigma}} = x_{\boldsymbol{\sigma}_1,...,\boldsymbol{\sigma}_l} = [\![\int \bigwedge_{\boldsymbol{\sigma}_j = 1} \beta_j \land \bigwedge_{\boldsymbol{\sigma}_j = 0} \neg\beta_j]\!]_{\mathcal{M}} . \tag{8}$$

Where $\boldsymbol{\sigma}_i$ represents the i-th bit of $\boldsymbol{\sigma}$. Therefore, by Lemma 2.2,

$$[\![\int\beta_i]\!]_{\mathcal{M}} = [\![\int \bigvee_{\substack{\sigma \in V_\Theta : \\ \sigma(\beta_i) = 1}} (\beta_i \land (\bigwedge_{\substack{\sigma(\beta_j) = 1, \\ j \neq i}} \beta_j \quad \land \bigwedge_{\substack{\sigma(\beta_j) = 0, \\ j \neq i}} \neg\beta_j))]\!]_{\mathcal{M}} = \tag{9}$$

$$= \sum_{\substack{\sigma \in V_\Theta) : \\ \sigma(\beta_i) = 1}} [\![\int (\beta_i \land (\bigwedge_{\substack{\sigma(\beta_j) = 1, \\ j \neq i}} \beta_j \quad \land \bigwedge_{\substack{\sigma(\beta_j) = 0, \\ j \neq i}} \neg\beta_j))]\!]_{\mathcal{M}} = \sum_{\substack{\sigma \in V_\Theta : \\ \boldsymbol{\sigma}_i = 1}} x_{\boldsymbol{\sigma}}.$$

Each atom in u_k can now be written as an inequality between polynomials in $x_{\boldsymbol{\sigma}}$ and Z variables, taking $\neg(t_1 \leq t_2)$ as $(t_1 > t_2)$ and the other obvious syntactic shortcuts. In addition, we must have that $x_{\boldsymbol{\sigma}} \geq 0$ and $\sum_{\boldsymbol{\sigma}} x_{\boldsymbol{\sigma}} = 1$. Therefore, any model satisfying μ_k must also satisfy the following system of inequations:

$$\begin{cases} \alpha_{1k} \\ \alpha_{2k} \\ \ldots \\ \alpha_{nk} \\ \sum_{i=0}^{2^l} x_i = 1 \\ x_i \geq 0 \end{cases} \qquad \text{where } \ \alpha_{ik} \text{ is the } i\text{-th literal of } \mu_k \ . \tag{10}$$

Before we solve the system, we must also include one additional restriction in non-feasible $x_{\boldsymbol{\sigma}}$, that is, $x_{\boldsymbol{\sigma}}$ whose associated QLTL molecule

$$\bigwedge_{\boldsymbol{\sigma}_j=1} \beta_j \wedge \bigwedge_{\boldsymbol{\sigma}_j=0} \neg\beta_j \tag{11}$$

is not satisfiable. For each such variable, we add "$x_{\boldsymbol{\sigma}} = 0$" to the system.

By using the well-known SAT algorithm [3] for the existential fragment of the first-order theory of real ordered fields, we can check whether the system has solutions. If this system has at least one solution, then we can construct the witness for the SAT Algorithm for δ, that we describe next.

Consider the individual QLTL models $\mathcal{M}_{\boldsymbol{\sigma}}$ associated with each $x_{\boldsymbol{\sigma}} \neq 0$, regarding them as a Markov chains with transitions of probability 1.[1] Consider the disjoint union of all these models, and starting distribution on the initial state of each model $\mathcal{M}_{\boldsymbol{\sigma}}$ with probability $p_{\boldsymbol{\sigma}} = x_{\boldsymbol{\sigma}}$; then $\mathcal{M}$ is a PDTS. Consider also the assignment ρ that maps each real algebraic variable in δ to its solution in the previous system. And so, $(\mathcal{M}\,, \rho)$ witness the satisfiability of δ.

The procedure is summarized in Algorithm 1.

Theorem 3.2 Algorithm 1 is correct, i.e., if Algorithm 1 returns a model, then that model witnesses the satisfiability of δ.

Proof. If Algorithm 1 returns a model, then each $\mathcal{M}_\sigma$ satisfies ϕ_σ corresponding *only* to that σ, because ϕ_σ contains at least one conjunct that is the negation of the respective conjunct in $\phi_{\sigma'}, \sigma \neq \sigma'$. Then, on $\mathcal{M}_{\sigma'}$ seen as a Markov chain with transitions of probability 1, the measure of the paths that satisfy ϕ_σ is 1 if $\sigma' = \sigma$ (there is only one path and it satisfies ϕ_σ), and 0 otherwise (there is only one path and it *does not* satisfy ϕ_σ). Therefore, $[\![\int \phi_\sigma]\!]_{\mathcal{M}} = \sum_{s_i \in \mathsf{S}} d_0(s_i)\mu_{s_i}(\{\pi : \pi \Vdash_{\mathsf{QLTL}} \int \phi_\sigma, \pi\,[0] = s_i\}) =$

[1] If a QLTL SAT algorithm, for some reason, returns a branching model, there will always be a non-branching model satisfying the same formula obtained by considering a path in the original model

Algorithm 1: StrongSAT$\mathsf{PQLTL}(\delta)$

```
   Input: $\mathsf{PQLTL}$ formula $\delta$
   Output: $\mathsf{PQLTL}$ model $(\mathcal{M} = (\mathsf{S}, d_i, \mathcal{T}, \mathsf{L}), \rho)$ or *no model*
1  compute $\lambda_\delta, atm(\lambda_\delta)$;
2  foreach $v \in V_{\lambda(gatm_\delta)}$ such that $v(\lambda_\delta) = 1$ do
3      compute $\mu_v$, $\Theta$;
4      $\kappa \longleftarrow \begin{cases} \alpha_{1v} \\ \alpha_{2v} \\ ... \\ \alpha_{nv} \\ \sum_{\{\sigma \in V_\Theta\}} x_\sigma = 1 \\ x_\sigma \geq 0 \end{cases}$;
5      foreach $\int \beta_i \in \kappa$ do
6          $\int \beta_i \longleftarrow \sum_{\{\sigma \in V_\Theta : \sigma(\beta_i) = 1\}} x_\sigma$;
7      end
8      foreach $x_\sigma$ do
9          $\phi_\sigma \longleftarrow \bigwedge_{\sigma(i)=1} \beta_i \wedge \bigwedge_{\sigma(i)=0} (\neg \beta_i)$;
10         $\mathcal{M}_\sigma \longleftarrow StrongQLTLSat(\phi_\sigma)$;
11         if $\mathcal{M}_\sigma =$ *no model* then
12             $\kappa \longleftarrow \kappa \cap x_\sigma = 0$;
13         end
14     end
15     $s \longleftarrow \exists StrongRealSat(\kappa)$;
16     if $s =$ *no solution* then
17         goto next $\mu_v$ in line 2;
18     end
19     $\mathcal{M} = \bigcup_{\sigma \in V_\Theta} \mathcal{M}_\sigma$;
20     $S = S_{\mathcal{M}}$; $d_0 = \{x_{\sigma_1}, ..., x_{\sigma_{2^l+1}}\}$ over $\{s_0^{\mathcal{M}_{\sigma_1}}, ..., s_0^{\mathcal{M}_{\sigma_{2^l+1}}}\}$ for $x_\sigma \neq 0$;
21     $\mathsf{L} = \mathsf{L}_{\mathcal{M}}$; $\mathcal{T}(s_1, s_2) = 1$ iff $s_1 \rightarrow s_2$ in the respective $\mathcal{M}_\sigma$;
22     return $m = (\mathsf{S}, d_0, \mathcal{T}, \mathsf{L})$ and $\rho(x) = \kappa(x)$ for $x \in Z$ ;
23 end
24 return *no model*;
```

$\sum_{\sigma' \in V} d_0(s_0^{\sigma'})\mu_{s_0^{\sigma'}}(\{\pi : \pi \Vdash_{\mathsf{QLTL}} \int\phi_\sigma, \pi\,[0] = s_0^{\sigma'}\}) = d_0(s_0^{\sigma})\mu_{s_0^{\sigma}}(\{\pi : \pi \Vdash_{\mathsf{QLTL}} \int\phi_\sigma, \pi\,[0] = s_0^{\sigma}\}) + \sum_{\sigma' \neq \sigma} d_0(s_0^{\sigma'})\mu_{s_0^{\sigma'}}(\{\pi : \pi \Vdash_{\mathsf{QLTL}} \int\phi_{\sigma'}, \pi\,[0] = s_0^{\sigma'}\}) = d_0(s_0^{\sigma})1 + \sum_{\sigma' \neq \sigma} d_0(s_0^{\sigma'})0 = x_\sigma$.

We also have that β_i is satisfied if $\bigvee_{\{\sigma \in V_{p(\mu)} : \sigma(\beta_i)=1\}} \phi_\sigma$ is satisfied since each of the paths that satisfy β_i satisfy also one (and only one) ϕ_σ such that $\sigma(\beta_i) = 1$. On the other hand, since each of these paths satisfy only one such ϕ_σ, we have $[\![\int \bigvee_{\{\sigma \in V_\Theta : \sigma(\beta_i)=1\}} \phi_\sigma]\!]_{\mathcal{M}} = \sum_{\{\sigma \in V_\Theta : \sigma(\beta_i)=1\}} [\![\int \phi_\sigma]\!]_{\mathcal{M}} = \sum_{\{\sigma \in V_\Theta : \sigma(\beta_i)=1\}} x_\sigma$.

Since these values and the values assigned to each $\rho(x)$ are exactly the solutions of the system, each inequality in some μ_v (the one for which output is produced) must hold for this model. But satisfaction of each inequality means satisfaction of μ_v, which means δ is satisfiable by Lemma 3.1. $\triangle$

Lemma 3.3 If μ_v is satisfiable, then the **foreach** command at line 2 must exit in line 22.

Proof. All steps of the algorithm are computations of total functions, therefore, the cycle quits. If the cycle quits, it must exit either in line 24 or in line 22. Suppose that it exits in line 24. Then the system κ has no solution.

On the other hand, if μ_v is satisfiable, there must be a model $\mathcal{M}$ and assignment ρ such that $\mathcal{M}, \rho \Vdash \alpha_{iv}$, where α_{iv} is the i-th literal of μ_v. For this model, the denotation of each measure term must take on a value such that the polynomial equation defined by each α_{iv} is satisfied.

By Lemma 2.2,

$$[\![\textstyle\int\beta]\!]_{\mathcal{M},\rho} = \sum_{i=0}^{2^m-1} [\![\textstyle\int(\beta \wedge (\bigwedge_{j=1}^{l} b_{ij}))]\!]_{\mathcal{M},\rho}$$

for the basic subformulae $\beta_1, ..., \beta_l$ of Θ.

By the same lemma,

$$1 = [\![\textstyle\int\top]\!]_{\mathcal{M},\rho} = \sum_{i=0}^{2^m-1} [\![\textstyle\int(\top \wedge (\bigwedge_{j=1}^{l} b_{ij}))]\!]_{\mathcal{M},\rho} = [\![\textstyle\int(\bigwedge_{j=1}^{l} b_{i,j})]\!]_{\mathcal{M},\rho}.$$

From the semantics of PQLTL, $[\![\int \gamma]\!]_{\mathcal{M}',\rho} \geq 0$ for any basic (QLTL) formula γ, model $(\mathcal{M}', \rho)$.

Finally, if any QLTL formula γ is not satisfied by some path, then $[\![\int\gamma]\!]_{\mathcal{M}',\rho} = 0$ regardless of the model $(\mathcal{M}', \rho)$.

Consider now the values for algebraic variables in Z assigned by ρ and $x_{\boldsymbol{\sigma}} = [\![\bigwedge_{j=1}^{l} b_{ij}]\!]_{\mathcal{M},\rho}$ where $b_{ij} = \beta_j$ if the i-th bit of $\boldsymbol{\sigma}$ is 1, $b_{ij} = (\neg\beta_j)$

otherwise. It is clear from the previous points that these $x_{\boldsymbol{\sigma}}$ witness the consistency of the system κ, which is a contradiction.

Therefore the cycle must exit at line 22. $\triangle$

Theorem 3.4 Algorithm 1 is adequate, i.e., if Algorithm 1 returns *no model*, then formula δ is not satisfiable.

Proof. Suppose δ is satisfiable. Then, by Lemma 3.1, there is μ_v satisfiable, s.t. $v \in V_{\lambda(gatm_\delta)}$ and $v(\lambda_\delta) = 1$. Algorithm 1 will enter the **foreach** at line 2 at least once with such μ_v. By Lemma 3.3, Algorithm 1 must exit at line 22 and therefore, cannot return *no model*. $\triangle$

Theorem 3.5 Algorithm 1 decides the strong satisfiability problem for PQLTL in [n+2]-$\mathsf{EXPSPACE}$, where n is the alternation depth of the quantifiers in the input.

Proof. By Theorem 3.2 and Theorem 3.4, Algorithm 1 clearly decides the satisfiability problem for PQLTL.

Storage of $atm(\lambda_\delta)$ in line 1 requires $\mathcal{O}(|\delta|)$ space.

After each iteration of the cycle in line 2, if the algorithm has not finished, we can discard all computations done inside the cycle before starting over, so the number of iterations does not influence the space requirements of the algorithm.

Computation of μ_v and Θ can be done in $\mathcal{O}(|\delta|)$. Storing system κ, however, requires storing a variable for each $\sigma \in V_\Theta$, which takes $\mathcal{O}(2^{|\Theta|}) = \mathcal{O}(2^{|\delta|})$.

There is a polynomial number of β_i, so the substitution in line 6 does not increase space constraints.

The cycle of line 8 runs $\mathcal{O}(2^{|\delta|})$ times, each time either storing $\mathcal{M}_\sigma$ or adding "$x_\sigma = 0$" to the system. ϕ can be computed in $O(|\delta|)$ and adding "$x_\sigma = 0$" takes constant space. However, $StrongQLTLSat$ is in (n+1)-$\mathsf{EXPSPACE}$ (c.f. Proposition A.15 in the Appendix, page 31), and, because we consider terms of the form $\neg\beta$, we actually need an [n+2]-$\mathsf{EXPSPACE}$ algorithm. Therefore, each $\mathcal{M}_\sigma$ takes [n+2]-$\mathsf{EXPSPACE}$ space in $|\delta|$. There are $\mathcal{O}(2^{|\delta|})$ variables, each requiring saving at most one $\mathcal{M}_\sigma$ this means that this cycle uses space of the order $\mathcal{O}(2^{|\delta|})\mathcal{O}((\mathsf{n}{+}2)\text{-}\mathsf{exp}\{p(|\delta|)\}) = \mathcal{O}((\mathsf{n}{+}2)\text{-}\mathsf{exp}\{p(|\delta|)\})$. Notice the size of the system remains $\mathcal{O}(2^{|\delta|})$.

In [3](c.f. Theorem A.18 in the Appendix, page 32), it is shown that the strong algorithm for the satisfaction of Real Algebraic Numbers is in $\mathsf{EXPSPACE}$. Since we actually need a solution for the system, this means line 15 takes $O(2^{2^{|\delta|}})$ space.

Line 19 adds no complexity, as it is merely a union of previously stored objects and line 20 just collects one subset of already existent structures (x_σ's and s_0's).

Therefore, the space complexity of the algorithm is

$$\mathcal{O}(|\delta|) + \mathcal{O}(2^{2^{|\delta|}}) + \mathcal{O}((\mathsf{n}{+}2)\text{-}\mathsf{exp}\{p(|\delta|)\}) = \mathcal{O}((\mathsf{n}{+}2)\text{-}\mathsf{exp}\{|\delta|\}),$$

which is [n+2]-EXPSPACE. △

Corollary 3.6 Algorithm 1 decides the strong satisfiability problem for PEQLTL in 2-EXPSPACE.

Algorithm 1 can easily be adapted to solve the *weak* version of the SAT problem. The impact of relaxing the requirements is twofold: in line 10, we need only to consider the weak version of the SAT algorithm for QLTL, which is n-EXPSPACE (in our case, since we consider terms of the form $(\neg\int\beta)$, we introduce an extra alternation, leaving us in [n+1]-EXPSPACE). Furthermore, we do not have to save the witnesses, only an indicator of the satisfiability of each ϕ_σ. Therefore, the cycle of line 8 reduces to [n+1]-EXPSPACE complexity. The other consequence of considering the weak SAT is that in line 15, where we need only to check the *consistency* of system κ. By Theorem A.18, this procedure can be done in PSPACE over the size of the system, which is EXPSPACE. Therefore, line 15 needs only EXPSPACE resources. This is of particular importance for the PEQLTL weak SAT problem, since for $n \geq 1$, the QLTL SAT already dominates the 2-EXPSPACE complexity of solving system κ, but for $n = 0$ (PEQLTL), the complexity of the whole algorithm would be dominated by line 15. We refrain from explicitly writing this algorithm for the weak SAT problem since it is very similar to Algorithm 1. Instead, we present the following modified algorithm:

Theorem 3.7 Algorithm 2 decides the weak satisfiability problem for PQLTL in [n+1]-EXPSPACE, where n is the alternation depth of the quantifiers in the input.

Corollary 3.8 Algorithm 2 decides the weak satisfiability problem for PEQLTL in EXPSPACE.

4 Restriction to non-quantified linear inequalities - PLTL$^+$

We now consider the case of non-quantified LTL. The SAT algorithm for a single LTL formula in a probabilistic setting is essentially the same as the

Algorithm 2: WeakSATPQLTL(δ)

```
Input: PQLTL formula $\delta$
Output: $\delta$ satisfiable or no model
Consider Algorithm 1 and:
replace line 10 with $y \longleftarrow WeakQLTLSat(\phi_\sigma)$;
replace line 11 with if $y =$ no model;
replace line 15 with $s \longleftarrow \exists WeakRealSat(\kappa)$;
delete lines 19, 20 and 21;
replace line 22 with return $\delta$ satisfiable.
```

SAT for the non-probabilistic version. However, when we consider quantitative reasoning as introduced in the syntax of PQLTL, the procedure becomes much less obvious, since different measure terms can share propositional symbols. Theorem 3.5 proves the decidability of the SAT problem for PQLTL, which is strictly more complicated than this problem, but the algorithm presented hinges on the fact that an exponential number of variables (in $|\delta|$) is used to represent the probability of each conceivable configuration of QLTL statements or their negations in the original formula δ. This representation can be much simplified. In this section, we show that considering non-quantified LTL formulae and a simple syntactic restriction, we can adapt Algorithm 1 to only consider a number of these variables linear on the size of the original formula, assuming all other to be 0. This will allow a significant reduction on the space requirements of the algorithm.

4.1 Syntatic restriction

A critical requirement for the new algorithm is that we must have a system of linear inequalities instead of a polynomial one. This can be done by weakening PQLTL, removing (besides the quantification, obviously) multiplication at the term level and leaving only sum as an algebraic connective. We baptize this weaker logic by PLTL$^+$. Its syntax is given in Table 4.1

$\beta := p \;[]\; (\neg\beta) \;[]\; (\beta \Rightarrow \beta) \;[]\; \mathsf{X}\beta \;[]\; \beta\mathsf{U}\beta$	basic formulae
$t := z \;[]\; \int\beta \;[]\; (t + t) \;[]\; (c.t)$	probabilistic terms
$\delta := (t \leq t) \;[]\; (\sim\delta) \;[]\; (\delta \supset \delta)$	global formulae

where $p \in \Lambda$, $z \in Z$, c is a real algebraic number.

Table 2: PLTL$^+$ syntax

The semantics for PLTL^+ is the restriction of the semantics of PQLTL obtained by removing the denotation of multiplication. Notice that PLTL^+ is still very expressive and that most of the natural language assertions concerning probabilities (of traces) are expressible within it. The most interesting feature of PLTL^+ is that satisfiability can be checked within PSPACE, as we show next.

4.2 SAT algorithm for PLTL^+

We now present the (weak) SAT algorithm for PLTL^+. It is an adaptation of Algorithm 2 that removes the need to consider all $x_{\boldsymbol{\sigma}}$ variables at once, thus reducing the space complexity. Notice that the strong version of the SAT problem for this problem is in $\mathsf{EXPSPACE}$, since the strong SAT for LTL reduces to it and therefore, in this case, the previous algorithms can be used without increasing the required complexity of the procedure. Algorithm 3 describes the procedure.

Theorem 4.1 Algorithm 3 is correct, i.e., if Algorithm 3 returns δ $satisfiable$, then δ is indeed satisfiable.

Proof. The proof of correction for this algorithm mimics the proof of correction of Algorithm 1 with the obvious changes in the domain of the sums and removing the construction of the witnesses. $\triangle$

Naturally, there is the concern that by considering only subsets of x_σ of size at most $|\delta| + 1$, we might leave out models where the formula would be satisfied. We claim this is not the case by showing that the existence of such models implies the existence of at least one suitably sized ($|\{x_\sigma \neq 0\}| \leq |\delta| + 1$) model that is found by Algorithm 3.

Theorem 4.2 Algorithm 3 is adequate, i.e., if Algorithm 3 returns *no model*, then formula δ is not satisfiable.

Proof. Suppose δ is a PLTL^+ satisfiable formula; then Algorithm 1 applied to δ returns a PDTS $\mathcal{M}$ and assignment ρ such that $\mathcal{M}, \rho \Vdash \delta$. Consider now the system stored in memory when Algorithm 1 is at line 22, just before exiting. Remove the 2^l conditions $x_\sigma \geq 0$, all conditions of the form $x_\sigma = 0$ and all the corresponding x_σ. This transformed system has $|atms(\delta)| + 1$ inequations and these inequations are linear, since δ is a PLTL^+ formula. Furthermore, the system has a nonnegative solution (the set of values of x_σ that would be outputted). Then, from linear programming [9], there

Algorithm 3: Sat $\mathsf{PLTL}^+(\delta)$

```
   Input: $\mathsf{PLTL}^+$ formula $\delta$
   Output: $\delta$ $satisfiable$ or $no\ model$
1  compute $\lambda_\delta, atm(\lambda_\delta)$;
2  foreach $v \in V_{\lambda(gatm_\delta)}$ such that $v(\lambda_\delta) = 1$ do
3      compute $\mu_v$, $\Theta$;
4      $\kappa \longleftarrow \begin{cases} \alpha_{1v} \\ \alpha_{2v} \\ \ldots \\ \alpha_{nv} \end{cases}$;
5      foreach $V \subseteq V_\Theta$ s. t. $0 < |V| \leq |\delta| + 1$ do
6          foreach $\int\beta_i \in \kappa$ do
7              $\int\beta_i \longleftarrow \sum_{\{\sigma \in V : \sigma(p_i)=1\}} x_\sigma$;
8          end
9          $\kappa \longleftarrow \kappa \cap \begin{cases} \sum_{\sigma \in V} x_\sigma = 1 \\ x_\sigma \geq 0 \text{ for } \sigma \in V \end{cases}$;
10         $s \longleftarrow LinearSolve(\kappa)$;
11         if $s = no\ solution$ then
12             break;
13         end
14         foreach $x_\sigma \neq 0$ do
15             $\phi_\sigma \longleftarrow \bigwedge_{\sigma(i)=1} \beta_i \wedge \bigwedge_{\sigma(i)=0} (\neg\beta_i)$;
16             $y \longleftarrow WeakLTLSat(\phi_\sigma)$;
17             if $y = no\ model$ then
18                 goto next $V$ in line 5;
19             end
20         end
21         return $\delta$ $satisfiable$;
22     end
23 end
24 return $no\ model$;
```

is a solution for the system with at most $|atms(\delta)| + 1$ variables taking values different from 0. Consider this solution and construct $\mathcal{M}'$ from $\mathcal{M}$ reassigning probabilities according to the new values of x_σ and discarding unneeded $\mathcal{M}_\sigma$. It is clear that $\mathcal{M}', \rho \Vdash \delta$, since the denotations of terms yield the same values.

Furthermore, this particular set of x_σ (different form 0) has size at most $|atms(\delta)| + 1$, so it will eventually be considered in the cycle in line 2 of Algorithm 3. At least in this iteration of the cycle, the algorithm must exit at line 21 and therefore, cannot return *no model.* $\triangle$

Theorem 4.3 Algorithm 3 decides the weak satisfiability problem for PLTL^+ in PSPACE.

Proof. By Theorem 4.1 and Theorem 4.2, Algorithm 3 clearly decides the satisfiability problem for PLTL^+.

Storage of $atm(\lambda_\delta)$ in line 1 requires $\mathcal{O}(|\delta|)$ space.

After each iteration of the cycle in line 2, if the algorithm has not finished, we can discard all computations done inside the cycle before starting over, so the number of iterations does not influence the space requirements of the algorithm.

Computation of μ_v and Θ can be done in $\mathcal{O}(|\delta|)$. Storing system κ, requires storing a variable for each of $|atms(\delta)|+1$ $\sigma \in V_\Theta$, which also takes $\mathcal{O}(|\delta|)$.

There is a polynomial number of β_i, so the substitution in line 7 leaves the algorithm still in $\mathcal{O}(p_1(|\delta|))$, for some polynomial p_1.

Line 9 requires an additional $\mathcal{O}(|\delta|)$ space. Notice the size of the system remains $\mathcal{O}(p_2(|\delta|))$, for some polynomial p_2.

It is well known from linear algebra that the algorithm for the satisfaction of systems of *linear* inequations is in P, which means that at line 10, the algorithm is still in $\mathcal{O}(p_3(|\delta|))$ space, for some polynomial p_3. Notice that considering only linear systems avoided once again an increase in the complexity of the algorithm.

The cycle of line 14 runs $\mathcal{O}(|\delta|)$ times, but it does not need to save any information at each iterations, so the space complexity does not increase with each iteration. Moreover, ϕ_σ can be computed in $\mathcal{O}(|\delta|)$, and $WeakLTLSat$ is in PSPACE [23]. These means that this cycle uses space of the order $p_4(\mathcal{O}(p_5(|\delta|))) = \mathcal{O}(p_6(|\delta|))$, for polynomials p_4, p_5 and p_6.

Therefore, the space complexity of the algorithm is $\mathcal{O}(|\delta|) + \mathcal{O}(p_3(|\delta|)) + p_4(\mathcal{O}(p_5(|\delta|)))$ which is PSPACE. $\triangle$

Theorem 4.4 The satisfiability problem for PLTL^+ is PSPACE-complete.

Proof. Algorithm 3 shows the satisfiability problem for PLTL$^+$ can be solved in PSPACE.

LTLSat is PSPACE complete [23]. Given a LTL formula β, we can obviously build the formula $(\int\beta > 0)$, from β in constant time. If there is a model for β, at least that model seen as a Markov chain satisfies $(\int\beta > 0)$ and an algorithm for the PLTL$^+$ SAT returns a model. If β is not satisfiable, then no path satisfies β and, in particular, all probability measures over paths that satisfy β in any model (the empty set) yield 0 and no model can be returned by a PLTL$^+$ SAT. △

5 Complete Hilbert calculus

We now turn our attention towards using the previous algorithms to obtain a complete Hilbert calculus for PQLTL and its reducts. The proof technique closely follows those in [8, 13, 18].

We present the axiomatization in Table 3. We consider an Hilbert system - recursive set of axioms and finitary rules. We recall the axiom schema **ROF** is decidable thanks to Tarski's result on the decidability of real ordered fields and for the decidability of QLTL, one can reference [14]. Thus, the axioms in Table 3 constitute a recursive set.

Axioms

[**GTaut**]	$\vdash_{\mathsf{PQLTL}}$	δ	for each PQLTL instantiation δ of a tautological propositional formula;
[**Prob**]	$\vdash_{\mathsf{PQLTL}}$	$(\int\varphi = 1)$	for each QLTL valid formula φ;
[**ROF**]	$\vdash_{\mathsf{PQLTL}}$	$(t_1 \leq t_2)$	for each instantiation of a valid analytical inequality;
[**FAdd**]	$\vdash_{\mathsf{PQLTL}}$	$((\int(\neg(\beta_1 \wedge \beta_2)) = 1) \supset (\int(\beta_1 \vee \beta_2) = \int\beta_1 + \int\beta_2));$	
[**Mon**]	$\vdash_{\mathsf{PQLTL}}$	$((\int(\beta_1 \Rightarrow \beta_2) = 1) \supset (\int\beta_1 \leq \int\beta_2));$	

Inference rules

[**MP**] δ_1, $(\delta_1 \supset \delta_2) \vdash_{\mathsf{PQLTL}} \delta_2$.

Table 3: HC_{PQLTL} : complete calculus for PQLTL.

Theorem 5.1 The calculus presented on Table 3 is sound, that is $\vdash_{\mathsf{PQLTL}} \delta$ implies $\models_{\mathsf{PQLTL}} \delta$.

Proof. Proof of correction is straightforward and will not be detailed; Axioms **GTaut** and **Prob** follow trivially from the definition of the semantics, the correctness of **ROF** comes from Tarski's result; **FAdd** and **Mon** are a consequence, respectively of the finite additivity and monotonicity of probability measures. The **MP** rule follows from the definition of the semantics of $\supset$. $\triangle$

The proof of completeness will follow by the usual contrapositive approach: If $\not\vdash_{\mathsf{PQLTL}} \delta$ then $\not\models_{\mathsf{PQLTL}} \delta$. Then, there is a PDTS $\mathscr{M}$ and an assignment ρ such that $\mathscr{M}, \rho \not\Vdash_{\mathsf{PQLTL}} \delta$. A formula δ is said *PQLTL-consistent* if $\not\vdash_{\mathsf{PQLTL}} (\sim\delta)$. We must first show that the consistency of a global formula is propagated to the consistency of at least one of its molecules.

Lemma 5.2 Let δ be a PQLTL-consistent formula. Then there is a molecule $\mathbf{\Phi}_k \in mol(\delta)$ such that μ_k is consistent.

Proof. Suppose, by contradiction, that for each $\mathbf{\Phi}_k \in mol(\delta)$, $(\sim\mu_k)$ is a theorem, then, by tautological reasoning (**GTaut**),

$$\bigcap_{\mathbf{\Phi}_k \in mol(\delta)} (\sim\mu_k) = \bigcap_{\mathbf{\Phi} \in mol(\delta)} (\sim((\bigcap_{\varphi\in\Phi} \varphi) \cap (\bigcap_{\varphi \in atm(\delta)\setminus\Phi} (\sim\varphi)))). \qquad (12)$$

is also a theorem. Therefore,

$$\sim(\bigcup_{\mathbf{\Phi} \in mol(\delta)} ((\bigcap_{\varphi\in\Phi} \varphi) \cap (\bigcap_{\varphi \in atm(\delta)\setminus\Phi} (\sim\varphi)))). \qquad (13)$$

is also a theorem. And so $\sim\delta$ is also a theorem and δ is not PQLTL-consistent, which is a contradiction. $\triangle$

Theorem 5.3 The calculus presented on Table 3 is complete, that is $\models_{\mathsf{PQLTL}} \delta$ implies $\vdash_{\mathsf{PQLTL}} \delta$.

Proof. We will prove that every PQLTL-consistent formula has a model. This will suffice, since $\not\vdash_{\mathsf{PQLTL}} \delta$ implies $\not\vdash_{\mathsf{PQLTL}} (\sim(\sim\delta))$, that is, $(\sim\delta)$ is PQLTL-consistent and so $(\sim\delta)$ is satisfiable which means $\not\models_{\mathsf{PQLTL}} \delta$.

Suppose then that γ is a PQLTL-consistent formula. By Lemma 5.2 there is a molecule of (γ), $\mu_\gamma = (\bigcap_{i\in I}(t \leq t')_i) \cap (\bigcap_{j\in J}(t \leq t')_j)$ that is PQLTL-consistent.

Assume, by contradiction, that the SAT Algorithm 1 returns *no model* for μ_γ. If the SAT algorithm returns *no model* for μ_γ it has to be for one of the following two reasons: (i) it can not find a v at line 2; (ii) for all viable v the SatReal algorithm returns no model at line 15. We will now show that for both cases we can contradict the consistency of μ_γ.

In case (i), λ_{μ_γ} (a propositional formula) is not satisfied by any valuation, i.e. $(\neg\lambda_{\mu_\gamma})$ is a valid formula and, by completeness of the propositional calculus, a theorem. Therefore, by **GTaut**, $\vdash_{PQLTL} (\sim\mu_\gamma)$.

In case (ii), using **Prob**, **FAdd** and **Mon**, and considering Θ the set of subformulae of μ_γ as in Subsection 3 we can rewrite each $\int\beta_j \in \mu_\gamma$ as

$$\sum_{\{\sigma\in\{0,1\}^{|\Theta|}:\sigma_j=1\}} \int(\beta_j \wedge (\bigwedge_{\sigma_i=1}\beta_i \wedge \bigwedge_{\sigma_i=0}(\neg\beta_i))). \tag{14}$$

We will refer to this rewritten form as μ^*_γ. The same axioms also allow us to derive the theorems

$$\sum_{\sigma\in\{0,1\}^{|\Theta|}} \int(\bigwedge_{\sigma_i=1}\beta_i \wedge \bigwedge_{\sigma_i=0}(\neg\beta_i)) = 1. \tag{15}$$

and

$$\int(\bigwedge_{\sigma_i=1}\beta_i \wedge \bigwedge_{\sigma_i=0}(\neg\beta_i)) \geq 0. \tag{16}$$

for all $\sigma \in \{0,1\}^{|\Theta|}$. Finally, we can derive which of these conjuncts are impossible with the use of **Prob**. We will call the set of all σ s.t. $\bigwedge_{\sigma_i=1}\beta_i \wedge \bigwedge_{\sigma_i=0}(\neg\beta_i)$ is impossible by $\mathbf{I}$.

By **GTaut** we can conjunct these theorems to μ^*_γ :

$$\begin{array}{l}
\mu_\gamma \equiv \\
\quad \mu^*_\gamma \;\; \wedge \\
\qquad \bigwedge_{\sigma\in\mathbf{I}} \;\; (\int(\bigwedge_{\sigma_i=1}\beta_i \wedge \bigwedge_{\sigma_i=0}(\neg\beta_i)) = 0) \quad \wedge \\
\qquad\quad (\sum_{\sigma\in\{0,1\}^{|\Theta|}} \int(\bigwedge_{\sigma_i=1}\beta_i \wedge \bigwedge_{\sigma_i=0}(\neg\beta_i)) = 1) \quad \wedge \\
\qquad\qquad \bigwedge_{\sigma\in\{0,1\}^{|\Theta|}} \;\; (\int(\bigwedge_{\sigma_i=1}\beta_i \wedge \bigwedge_{\sigma_i=0}(\neg\beta_i)) \geq 0).
\end{array} \tag{17}$$

The system generated by $\kappa_{x_\sigma}^{(\bigwedge_{\sigma_i=1}\beta_i \wedge \bigwedge_{\sigma_i=0}(\neg\beta_i))}$, which is just the right hand side of the expression in (17) with each conjunct of the form $(\bigwedge_{\sigma_i=1}\beta_i \wedge \bigwedge_{\sigma_i=0}(\neg\beta_i))$ substituted by a fresh variable $x_\sigma$is exactly the system κ in line 15 of Algorithm 1 , which we are assuming has no solution. Therefore, by completeness of the axiomatization of the real ordered fields, it must be possible to derive $(\sim\kappa)$, and, using **ROF**, derive its instantiation $(\sim\mu_\gamma)$, which completes the proof. $\triangle$

Corollary 5.4 The calculus presented Table 3 with axioms **GTaut** and **Prob** substituted by

[**GTaut2**] $\vdash$ δ for each PEQLTL instantiation δ of a tautological propositional formula;

[**Prob2**] $\vdash$ $(\int \varphi = 1)$ for each EQLTL valid formula φ;

is a complete axiomatization of the PEQLTL fragment of PQLTL.

Corollary 5.5 The calculus presented Table 3 with axioms **GTaut**, **Prob** and **ROF** substituted by

[**GTaut3**] $\vdash$ δ for each PLTL instantiation δ of a tautological propositional formula;

[**Prob3**] $\vdash$ $(\int \varphi = 1)$ for each LTL valid formula φ;

[**LROF**] $\vdash$ $(t_1 \leq t_2)$ for each instantiation of a valid *linear* analytical inequality;

is a complete axiomatization of the PLTL$^+$ fragment of PQLTL.

5.1 Example: Training for the PONG$^{\mathrm{TM}}$ world championship

We now present a very basic toy example, just to illustrate the use of the calculus. The setting is as follows. PONG$^{\mathrm{TM}}$ is a primitive computer game, akin to tennis, where players control a pad on each side of a screen and bounce back and forth a "ball". The world champion of PONG$^{\mathrm{TM}}$ is training for the next series, but also has to watch over his young nephew. He decides to play the game with him, but the youngster has a non-negligible chance of not being able to return the "ball" successfully. Therefore, the world champion sets a "wall" in the nephew's side of the field which automatically returns the "ball" should the nephew fail to do it. Figure 1 summarizes the setting.

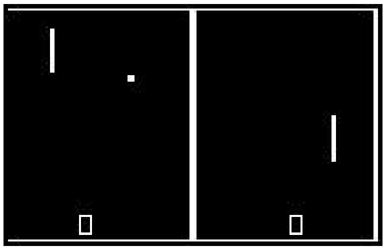


Figure 1: A PONG$^{\mathrm{TM}}$game between the world champion and his nephew.

Assume that the world champion never fails to return the ball; we will use PQLTL to show that on all even transitions, the ball is returned by a human player. Notice that the "intuitive" approach of using LTL to state $(\mathsf{G}(hum \Rightarrow \mathsf{XX}hum))$ fails in this model, since the nephew would be required to keep returning the ball after his first success.

In order to model this situation, we will take as hypothesis the following formulae:

a) $\int \mathsf{G}((s_1 \Rightarrow hum) \wedge (s_2 \Rightarrow hum) \wedge (s_3 \Rightarrow \neg hum)) = 1$;

b) $\int \mathsf{G}(s_1 \dot{\vee} s_2 \dot{\vee} s_3) = 1$;

c) $\int \mathsf{G}(s_1 \Rightarrow \mathsf{X}(s_2 \vee s_3) \wedge (s_2 \Rightarrow \mathsf{X}s_1) \wedge (s_3 \Rightarrow \mathsf{X}s_1)) = 1$;

d) $\int s_1 = 1$.

Where $\dot{\vee}$ denotes the *exclusive or* connective. Let Γ be the set of formulae *a*), *b*), *c*), *d*). The Markov chain of Figure 5.1 is one of many that satisfies Γ and we will use it as a guideline for following the example. In this case, s_1 is a state that represents the world champion, s_2 the nephew and s_3 the wall. The propositional symbol *hum* indicates if the player is human.

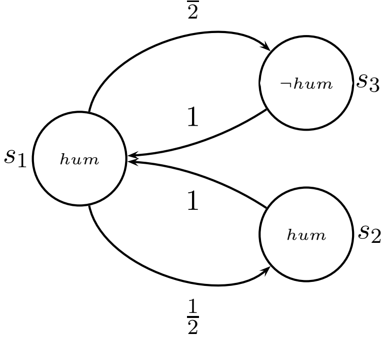


Figure 2: A model satisfying Γ, assuming the nephew has $\frac{1}{2}$ probability of failing to return the ball

Condition *a*) fixes which states pertain to human players; *b*) states that the ball can only be played by one player at a time and ensures that the players are different entities; *c*) expresses the possible transitions of the ball; *d*) just states that the world champion has the service (starts the game).

We are trying to derive the expression:

$$\int (\exists p.(p \wedge (\mathsf{X}\neg p) \wedge (\mathsf{G}(p \Leftrightarrow \mathsf{XX}p)) \wedge (\mathsf{G}(p \Rightarrow hum)))) = 1. \qquad (18)$$

The following lemma will be extensively used in the following derivations. It essentially allows propositional reasoning to be used *inside* measure terms of probability 1.

Lemma 5.6 Let $\Delta \vdash (\int(\beta_1 \Rightarrow \beta_2) = 1)$ and $\Delta \vdash (\int\beta_1 = 1)$. Then $\Delta \vdash (\int\beta_2 = 1)$.

Proof. By **Mon**, $\Delta \vdash ((\int(\beta_1 \Rightarrow \beta_2) = 1) \supset (\int\beta_1 \leq \int\beta_2))$, therefore, by **MP**, $\Delta \vdash (\int\beta_1 \leq \int\beta_2)$. Now, by **ROF** and **Prob** we get $\Delta \vdash (1 \leq \int\beta_2) \cap (\int\beta_2 \leq 1)$ and once again by **ROF**, $\Delta \vdash (\int\beta_2 = 1)$. △

From Γ, we have directly

$$\Gamma \vdash \int s_1 = 1; \tag{19}$$

$$\Gamma \vdash \int(\mathsf{G}(s_1 \Rightarrow hum)) = 1. \tag{20}$$

From *b*), we derive, by tautological reasoning permitted by Lemma 5.6, $\int\mathsf{G}((s_2 \vee s_3) \Leftrightarrow (\neg s_1)) = 1$, and more concretely $\int((s_2 \vee s_3) \Leftrightarrow (\neg s_1)) = 1$. From *c*) we derive $\int\mathsf{G}(s_1 \Rightarrow \mathsf{X}(s_2 \vee s_3)) = 1$. From these assertions, *d*), and Lemma 5.6 again, we get

$$\Gamma \vdash \int(\mathsf{X}(\neg s_1)) = 1. \tag{21}$$

$\int\mathsf{G}(s_1 \Rightarrow \mathsf{X}(s_2 \vee s_3)) = 1$ allows us to get $\int\mathsf{G}((s_1 \Rightarrow \mathsf{X}s_2) \vee (s_1 \Rightarrow \mathsf{X}s_3)) = 1$. With this assertion and *c*), we derive

$$\Gamma \vdash \int\mathsf{G}(s_1 \Rightarrow \mathsf{XX}s_1) = 1. \tag{22}$$

Finally, from $\int\mathsf{G}(s_2 \Rightarrow \mathsf{X}s_1) = 1$ and $\int\mathsf{G}(s_1 \Rightarrow \mathsf{X}(s_2 \vee s_3)) = 1$ (both from *c*)) we derive $\int\mathsf{G}(s_2 \Rightarrow \mathsf{XX}(s_2 \vee s_3)) = 1$ and, in a similar way, $\int\mathsf{G}(s_3 \Rightarrow \mathsf{XX}(s_2 \vee s_3)) = 1$. Both these assertions together yield $\int\mathsf{G}((s_2 \vee s_3) \Rightarrow \mathsf{XX}(s_2 \vee s_3)) = 1$. Joining $\int((s_2 \vee s_3) \Leftrightarrow (\neg s_1)) = 1$, which we have already justified, we get $\int\mathsf{G}((\neg s_1) \Rightarrow \mathsf{XX}(\neg s_1)) = 1$, which is classically equivalent to

$$\Gamma \vdash \int\mathsf{G}(\mathsf{XX}s_1 \Rightarrow s_1) = 1. \tag{23}$$

Once again by classical reasoning permitted by Lemma 5.6, we can join (19), (20), (21), (22), (23) and get

$$\Gamma \vdash \int(s_1 \wedge (\mathsf{X}\neg s_1) \wedge (\mathsf{G}(s_1 \Leftrightarrow \mathsf{XX}s_1)) \wedge (\mathsf{G}(s_1 \Rightarrow hum))) = 1. \tag{24}$$

From the contrapositive version of Axiom $\mathsf{QX2}$ from QLTL calculus in [14], we have

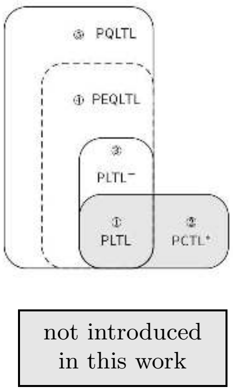
$$\begin{aligned}\Gamma \vdash \int((s_1 \wedge (\mathsf{X}\neg s_1) \wedge (\mathsf{G}(s_1 \Leftrightarrow \mathsf{XX}s_1)) \wedge (\mathsf{G}(s_1 \Rightarrow hum))) \Rightarrow \\ (\exists s_1.(s_1 \wedge (\mathsf{X}\neg s_1) \wedge (\mathsf{G}(s_1 \Leftrightarrow \mathsf{XX}s_1)) \wedge (\mathsf{G}(s_1 \Rightarrow hum)))) = 1.\end{aligned} \tag{25}$$

Therefore, by **MP**, we have

$$\int(\exists s_1.(s_1 \wedge (\mathsf{X}\neg s_1) \wedge (\mathsf{G}(s_1 \Leftrightarrow \mathsf{XX}s_1)) \wedge (\mathsf{G}(s_1 \Rightarrow hum)))) = 1 \tag{26}$$

as we wanted.

6 Conclusions and future work



① PLTL - “With probability of 99%, the process eventually reaching the critical state implies that a flag will always eventually be raised.”

② PCTL* - “With probability of 98%, the process eventually reaching the critical state implies that a flag will always eventually be raised with probability 99%.”

③ PLTL$^+$ - “The probability of reaching `Undecided` is at least 10% less than reaching either `Accept` or `Reject`.”

④ PQLTL/ ⑤ PEQLTL- “The probability that process A refrains from reading from the common channel in all even steps and that process B refrains from doing so in all odd steps is at least 99.9%.”
“The probability of reaching `Accept` is inversely proportional to that of reaching `Undecided`.”

Logic	Strong SAT	Weak SAT	MC
PQLTL	[n+2]-EXPSPACE	[n+1]-EXPSPACE	[n+2]-EXPSPACE
PEQLTL	2-EXPSPACE	EXPSPACE	2-EXPSPACE
PLTL$^+$	EXPSPACE	PSPACE	PSPACE
PLTL	EXPSPACE	PSPACE	PSPACE
PCTL*	unknown	unknown	PSPACE

Table 4: Relations between PLTL, PCTL*, PLTL$^+$, PEQLTL and PQLTL.

Herein we have proposed a new probabilistic logic to reason about semi-algebraic constrains of probabilities of sets of path of a Markov chains specified by a QLTL formula. Moreover, we presented a model-checking and SAT [n+2]-EXPSPACE algorithm for the logic and a weakly complete Hilbert calculus. We have considered relevant subfragments with a more efficient SAT algorithm, namely PLTL$^+$ which extends PLTL. In Table 4 we summarize the results obtained. We note that SAT algorithms presented also work for models that incorporate both non-deterministic and probabilistic transitions, such as those in [5, 10, 26], since Markov chains are particular cases of these models.

For future work we intend to enrich PQLTL with global temporal reasoning, and eventually, for the sake of exhaustiveness, nesting of the probability operator. We also intend to explore the calculus in less academic examples.

Acknowledgments

This work was partially supported by project SQIG at IT, IT Project QuantTel, Network of Excellence –Euro-NF, the SQIG LAP initiative, and the FCT and EU FEDER projects QSec PTDC/EIA/67661/2006, QuantPrivTel PTDC/EEA-TEL/103402/2008, AMDSC UTAustin/MAT/0057/2008 project of IST, PhD fellowship SFRH/BD/22698/2005, and PhD fellowship within the program CMU | Portugal.

References

[1] A. Aziz, V. Singhal, F. Balarin, and R. K. Brayton. It usually works: The temporal logic of stochastic systems. In *Lecture Notes in Computer Science*, pages 155–165. Springer, 1995.

[2] C. Baier and J. P. Katoen. *Principles of Model Checking.* The MIT Press, May 2008.

[3] S. Basu, R. Pollack, and M. F. Roy. *Algorithms in Real Algebraic Geometry.* Springer-Verlag, 2003.

[4] G. Bernot, J. P. Comet, A. Richard, and J. Guespin. Application of formal methods to biological regulatory networks: extending Thomas' asynchronous logical approach with temporal logic. *Journal of Theoretical Biology*, 220:339–347, 2004.

[5] A. Bianco and L. De Alfaro. Model checking of probabilistic and nondeterministic systems. *Lecture Notes in Computer Science*, 1026:499–??, 1995.

[6] G. V. Bochmann. Hardware specification with temporal logic: An example. *IEEE Transactions on Computers*, C-31:223–231, 1982.

[7] J. R. Büchi. On a Decision Method in Restricted Second Order Arithmetic. In *Proceedings of the 1960 International Congress of Logic, Methodology and Philosophy of Science*, pages 1–12. Stanford University Press, 1960. June.

[8] R. Chadha, P. Mateus, A. Sernadas, and C. Sernadas. Extending classical logic for reasoning about quantum systems. In D. Gabbay

K. Engesser and D. Lehmann, editors, *Handbook of Quantum Logic and Quantum Structures: Quantum Logic*, pages 325–372. Elsevier, 2009.

[9] V Chávtal. *Linear programming.* WH Freeman, 1983.

[10] F. Ciesinski and M. Größer. On probabilistic computation tree logic. In C. Baier, B. R. Haverkort, H. Hermanns, J. P. Katoen, and M. Siegle, editors, *Validation of Stochastic Systems*, volume 2925 of *Lecture Notes in Computer Science*, pages 147–188. Springer, 2004.

[11] E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking.* MIT Press, 1999.

[12] C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of ACM*, 42:857–907, 1995.

[13] R. Fagin, J. Y. Halpern, and N. Megiddo. A logic for reasoning about probabilities. *Inf. Comput.*, 87(1-2):78–128, 1990.

[14] T. French and M. Reynolds. A sound and complete proof system for QPTL. In Philippe Balbiani, Nobu-Yuki Suzuki, Frank Wolter, and Michael Zakharyaschev, editors, *Advances in Modal Logic*, pages 127–148. King's College Publications, 2002.

[15] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Asp. Comput*, 6(5):512–535, 1994.

[16] P. Øhrstrøm and P. F. V. Hasle. *Temporal Logic: From Ancient Ideas to Artificial Intelligence.* Kluwer, 1995.

[17] J. G. Kemeny, J. L. Snell, and A. W. Knapp. *Denumerable Markov Chains.* Van Nostrand, New Jersey, 1966.

[18] P. Mateus and A. Sernadas. Weakly complete axiomatization of exogenous quantum propositional logic. *Information and Computation*, 204, 2006.

[19] M. Mukund. Finite-state automata on infinite inputs, July 09 1996.

[20] A. Pnueli. The temporal logic of programs. In *focs77*, pages 46–57, 1977.

[21] M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. Amer.Math.Soc.*, 141:1–35, 1969.

[22] A. P. Sistla. *Theoretical issues in the design and verification of distributed systems.* PhD thesis, Harvard University, Cambridge, MA, USA, 1983.

[23] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. In *STOC '82: Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 159–168, New York, NY, USA, 1982. ACM.

[24] A. P. Sistla, M. Y. Vardi, and P. Wolper. The complementation problem for büchi automata with applications to temporal logic (extended abstract). In *Proceedings of the 12th Colloquium on Automata, Languages and Programming*, pages 465–474, London, UK, 1985. Springer-Verlag.

[25] A. Tarski. *A Decision Method for Elementary Algebra and Geometry.* University of California Press, 2d edition, 1951.

[26] M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Foundations of Computer Science 85*, pages 327–338, 1985.

[27] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Lectures in Computer Science 86*, pages 332–344, 1986.

A Appendix

A.1 Automata notations and needed results

A Büchi automaton is a finite state automaton with a suitable accepting condition for infinite sequences. It was proposed by Büchi in [7]. For a detailed introduction to Automata on infinite words, see [19].

Definition A.1 A tuple $B = (S, \Sigma, \tau, s_0, F)$ is said to be a *Büchi automaton* if S and Σ are finite, $\tau \subseteq S \times \Sigma \times S$, $s_0 \subseteq S$ and $F \subseteq S$.

We say that an infinite sequence δ^ω is *fireable in N* if $\forall_{i \geq 1} \quad pr_1(\delta\,[i]) = pr_3(\delta\,[i-1])$ and $pr_1(\delta\,[0]) \in s_0$, where pr_i are the projections on the i-th component. Furthermore $N(\alpha)$ represents the sequence of states visited in τ.

We define the language accepted by N as:

$$L^\omega(N) = \{pr_2(\alpha) : \alpha \in \tau^\omega, \alpha \text{ is fireable in } N, inf(N(\alpha)) \cap F \neq \emptyset\} .$$

The map $inf(.) : \Sigma^\omega \mapsto \Sigma$ produces the set of symbols appearing infinitely often in the given infinite sequence.

Definition A.2 Let $G = (V, E)$ be a finite directed graph. We call $C \subseteq V$ an ergodic set if

- $\forall (u, v) \in E, u \in C \Rightarrow v \in C$,
- $\forall u, v \in C, \exists$ a path from u to v.

Proposition A.3 (see [7]) The set of all languages accepted by Büchi automata is precisely the set of ω-regular languages.

Proposition A.4 (see [24]) The emptiness problem for the language accepted by a Büchi automaton is decidable in NLOGSPACE over the size of the automaton.

Proposition A.5 (see [24]) The emptiness problem for the complementary language accepted by a Büchi automaton is decidable in PSPACE over the size of the automaton.

Looking closer at the definition of the transition relation τ in the definition of Büchi automata, we are requiring that the underlying automata should be non-deterministic. Furthermore, unlike finite state automata in Automata Theory over finite sequences, deterministic Büchi automata are strictly less expressive than non-deterministic Büchi automata [19]. This fact motivates the introduction of the following automata over infinite sequences.

Definition A.6 (see [21]) A *Rabin automaton* is a tuple $R = (S, \Sigma, \tau, s_0, L)$ where S, Σ, τ and s_0 are as in Definition A.1 and $L = \{(A_1, B_1), \dots (A_k, B_k)\}$ where $A_i, B_i \subseteq S$.

Definition A.7 Let $S \subseteq Q$. We say that S satisfies $L = \{(A_1, B_1), \dots (A_k, B_k)\}$ if $\exists_{1 \le i \le k}$ such that $S \cap A_i \neq \emptyset$ and $S \cap B_i = \emptyset$.

With this table of pairs, the accepted language of a Rabin automaton is defined as:

$$L^\omega(R) = \{pr_2(\alpha) : \alpha \in \delta^\omega, \alpha \text{ is fireable in R}, \alpha \text{ satisfies } L\}.$$

Proposition A.8 (see [19, 26]) The set of all languages accepted by deterministic Rabin automata are the ω-regular languages. Furthermore, it is possible to convert a non-deterministic Büchi automaton into a deterministic Rabin automaton in exponential space relative to the size of the non-deterministic Büchi automaton.

A.2 Quantified Propositional Linear Temporal Logic

Quantified Propositional Linear Temporal Logic was introduced in [22] to address the lack of expressiveness of LTL. Let Λ be a finite set of propositional symbols. The syntax of quantified propositional linear temporal logic (QLTL) formulae over Λ is defined as, where $p \in \Lambda$:

$$\beta := p \;[\!]\; (\neg\beta) \;[\!]\; (\beta \Rightarrow \beta) \;[\!]\; \mathsf{X}\beta \;[\!]\; \beta\mathsf{U}\beta \;[\!]\; \exists p.\beta(p)$$

The usual abbreviations for $\bot, \top, \wedge, \vee, \Leftrightarrow, \mathsf{F}, \mathsf{G}, \forall$ are assumed. Furthermore, when using multiple quantifiers of the same type, the symbols $\exists, \forall$ may be used.

Regarding the semantics, let S be a set of states, and $\mathsf{L} : \mathsf{S} \to \{0,1\}^{\Lambda}$. A *path* π over S is an element of S^{ω}. To each path is therefore associated an infinite sequence of valuations over Λ. The k-th state of a path π is s_k and is denoted by $\pi\left[k\right]$. Given a path $\pi = s_0 s_1 ... s_k ...$, we denote the *$k$-prefix* $s_0 s_1 ... s_k$ by $\pi|_k$ and say its *length* is $k+1$.

A model for QLTL is a triple $m = (\mathsf{S}, \pi, \mathsf{L})$. We say that two models, $m = (\mathsf{S}, \pi, \mathsf{L})$ and $m' = (\mathsf{S}, \pi, \mathsf{L}')$ are p_i-equivalent, for some $p_i \in \Lambda$, if for all k and $p_j \neq p_i$, $\mathsf{L}(\pi\left[k\right](p_j)) = \mathsf{L}'(\pi\left[k\right](p_j))$. It is clear that p_i-equivalence is an equivalence relation. The satisfaction relation $\Vdash_{QLTL}$ over QLTL formulae is defined inductively in the following way:

$(\mathsf{S}, \pi, \mathsf{L}) \Vdash p$ iff $\pi\left[0\right](p) = 1$,
$(\mathsf{S}, \pi, \mathsf{L}) \Vdash \neg\beta$ iff $(\mathsf{S}, \pi, \mathsf{L}) \nVdash \beta$
$(\mathsf{S}, \pi, \mathsf{L}) \Vdash \beta_1 \Rightarrow \beta_2$ iff $(\mathsf{S}, \pi, \mathsf{L}) \nVdash \beta_1$ or $(\mathsf{S}, \pi, \mathsf{L}) \Vdash \beta_2$,
$(\mathsf{S}, \pi, \mathsf{L}) \Vdash \mathsf{X}\beta$ iff $(\mathsf{S}, \pi^1, \mathsf{L}) \Vdash \beta$,
$(\mathsf{S}, \pi, \mathsf{L}) \Vdash \beta_1\mathsf{U}\beta_2$ iff $\exists_{i \geq 0}\left((\mathsf{S}, \pi^i, \mathsf{L}) \Vdash \beta_2 \text{ and } \forall_{0 \leq j < i}\ (\mathsf{S}, \pi^j, \mathsf{L}) \Vdash \beta_1\right)$,
$(\mathsf{S}, \pi, \mathsf{L}) \Vdash \exists p.\beta$ iff there exists a p-equivalent model $(\mathsf{S}, \pi, \mathsf{L}')$ s.t. $(\mathsf{S}, \pi, \mathsf{L}') \Vdash \beta$.

Sistla, in his dissertation thesis [22], presented the following proposition:

Proposition A.9 If $\beta \in \mathsf{QLTL}$ then there exist formulae β_1, ψ s. t.

$$\beta_1 \equiv_{abv} \exists\forall \ldots \exists\forall\psi,$$

where $\psi \in \mathsf{LTL}$ and $\models_{\mathsf{QLTL}} \beta \Leftrightarrow \beta_1$.

We say that a formula β is in *normal form* or *prenex normal form*, if $\beta \equiv \exists\forall \ldots \exists\forall\psi$. This normal form theorem motivates the definition of the set of formulae Σ_k^{QLTL}. A formula QLTL in normal form is in Σ_k^{QLTL} if and only if the first quantifier is an $\exists$ quantifier, and there are further k alternations of quantifiers.

Proposition A.10 LTL is not expressive enough to encode the proposition $\mathsf{G}_2 p \equiv$ p is true at all even instants.

However, Proposition A.10 does not hold for QLTL; the formula $\exists p'.(p' \wedge \mathsf{X}\neg p' \wedge \mathsf{G}(p' \Leftrightarrow \mathsf{XX}p') \wedge \mathsf{G}(p' \Rightarrow p)$ expresses $G_2 p$.

A.3 Regarding Automata theory and QLTL

There are some deep connections between LTL and QLTL and automata theory.

Proposition A.11 (see [14, 22, 24]) For each LTL formula ψ, there exists a Büchi automaton s.t. its accepted language is precisely the set of models of ψ.

This translation procedure uses exponential space on the size of the LTL formula.

Proposition A.12 (see [22]) For each QLTL formula β, there exists a Büchi automaton such that its accepted language is precisely the set of models of β; furthermore, for each Büchi automaton B there exists a formula $\beta \in \mathsf{QLTL}$ such that the set of models of β is precisely $L^\omega(B)$.

In fact, this statement can even be strengthened as Σ_0^{QLTL} is as expressive as non-deterministic Büchi automata. This fact allows us to assume an existential normal form for all formulae in QLTL. Usually, this fragment is named EQLTL. This translation, however, is not simple, and it is done using Proposition A.12.

Proposition A.13 (see [14]) For each QLTL formula β, there exists $\psi \in \Sigma_0^{QLTL}$ such that $\models_{\mathsf{QLTL}} \beta \Leftrightarrow \psi$.

We can use the translation between QLTL and Büchi automata to obtain complexity bounds for the satisfiability problem of QLTL formulae.

Proposition A.14 (see [23]) The satisfiability problem for LTL formulae is PSPACE-complete on the size of the formula if a witness is not required and $\mathsf{EXPSPACE}$ otherwise.

Proposition A.15 (see [24]) The satisfiability problem for QLTL formulae in normal form is in n-$\mathsf{EXPSPACE}$ in the alternation depth of the normal form for $\mathsf{n} \geq 1$, and in $[\mathsf{n}+1]$-$\mathsf{EXPSPACE}$ while providing a witness.

In our case, we need to consider QLTL formulae over probabilistic deterministic transition systems (PDTS). Each probabilistic deterministic transition system induces a measure over the Borel σ-algebra of the basic cylinders of S^ω, where S is the set of states of the PDTS. We need then to know if the set of all models of a given QLTL formula is in the Borel σ-algebra. This result was proved directly for LTL for QLTL the result was proved in [26], using automata-theoretical tools.

Proposition A.16 (see [26]) Let $\beta \in$ QLTL. Then the sets $\{\pi : (\mathsf{S}, \pi, \mathsf{L}) \Vdash_{QLTL} \beta, \pi\,[0] = s_i\}$ are measurable in the Borel σ-algebra over the basic cylinders of S^ω.

A.4 Real Ordered Fields

Theorem A.17 (see [25]) The theory of real ordered fields is decidable.

Theorem A.18 ([3] page 488) The satisfiability of quantifier-free formulae in the theory of real ordered fields is decidable in PSPACE. Providing a witness is EXPSPACE.

B Model-checking algorithm

The model-checking problem for PQLTL consists in deciding whether a certain PQLTL formula δ, is satisfied by a given PDTS $\mathcal{M} = (\mathsf{S}, d_0, \mathcal{T}, \mathsf{L})$ for some non-specified assignment ρ. There are two main conceptual steps on the model-checking algorithm; the first step is to compute the denotation of each measure term. In order to do so, we will use an automata-theoretical algorithm for probabilistic deterministic transition systems proposed in [26], which can be seen, for instance, in [10]. We will only adapt it to use the already latent possibilities in the referenced algorithm to enclose the more expressive QLTL.

After this step, we only need to compute whether there exists an assignment ρ that satisfies the set of polynomial inequalities obtained from δ; this can be accomplished through the decidability result about real algebraic closed fields originally proved by Tarski (Theorem A.17 in Appendix, page 32).

We will now describe the first step of the model-checking algorithm. The algorithm receives as input a measure term of the form $\int \beta$ and a PDTS $\mathcal{M} = (\mathsf{S}, \mathcal{T}, d_0, L)$, where β is a Σ_k^{QLTL} formula; it returns as output an algebraic real number, corresponding to $[\![\int \beta]\!]_{\mathcal{M}}$. This automata-theoretic

```
Algorithm 4: MeasureTermCalc($\beta, \mathcal{M}$)
   Input: $\mathsf{QLTL}$ formula $\beta$, PDTS $\mathcal{M} = (\mathsf{S}, \mathcal{T}, d_0, L)$
   Output: $[\![\int \beta]\!]_{\mathcal{M}}$
 1 compute DRA $B_\beta = (Q, 2^\Lambda, q_0, \tau, \alpha)$ s.t. $L^\omega(B_\beta) = L(\beta)$;
 2 compute $\mathcal{M} \times B_\beta = (\mathsf{S} \times Q, \mathcal{T}', d'_0, L)$:
 3         $\mathcal{T}'((s,q),(s',q')) = \begin{cases} \mathcal{T}(s,s'), & \text{if } (q, L(s), q') \in \tau \\ 0, & \text{otherwise.} \end{cases}$;
 4         $d'_0 = (d_0, q_0)$; $L'(s,q) = L(s)$;
 5 compute $\alpha' = \{(S \times L, S \times U) : (L, U) \in \alpha\}$;
 6 compute $A = \bigcup\{E \subseteq S \times Q : E \text{ is ergodic}, E \text{ satisfies } \alpha'\}$;
 7 foreach $(s,q) \in A$ do
 8     label $(s,q)$ with $acc$;
 9 end
10 compute
       $P^\infty(s,q) = \begin{cases} 1, & \text{if } (s,q) \text{ is labeled by } acc; \\ 0, & \begin{array}{l}\text{if there are only null measure paths} \\ \text{connecting (s,q) to a state labeled by } acc;\end{array} \\ \sum_{(s',q') \in \mathsf{S} \times Q} \mathcal{T}'((s,q),(s',q')) P^\infty(s',q'), & \text{otherwise.} \end{cases}$
11 return $\sum_{(s',q_0)} P^\infty(s', q_0) d(s')$
```

approach uses the equivalence between QLTL and deterministic Rabin automata (DRA) to build a product PDTS containing the information about the QLTL formula and the PDTS given as inputs. Afterwards, as in [10], it is only a matter of computing the ergodic sets of the product PDTS, and check which ergodic sets satisfy the accepting condition. In fact, we are just reducing the computation of $\int \beta$, for some $\beta \in \mathsf{QLTL}$ on the input PDTS to the computation of the probability of the formula $\int \mathsf{F}acc$ in a labeled product PDTS obtained from the input and the Rabin automaton of the input formula. This reduction already appears in [10].

Finally, we need to compute the probability in each state of satisfying the accepting condition. Although the definition in Line 10 is recursive, it can be solved using a system of linear equations.

Therefore, the model-checking procedure for PQLTL is fully described in

Algorithm 5.

```
Algorithm 5: PQLTLModelChecker($\delta, \mathcal{M}$)
Input: PQLTL formula $\delta$, PDTS $\mathcal{M} = (\mathsf{S}, \mathcal{T}, d_0, L)$
Output: satisfied or not satisfied
foreach measure term $\int\beta$ in $\delta$ do
    compute $c_\beta = \text{MeasureTermCalc}(\beta, \mathcal{M})$;
    replace $\int\beta \hookleftarrow c_\beta$ in $\delta$;
end

return $\exists$WeakRealSat($\delta$);
```

The space complexity of the model-checking algorithm depends mainly on the complexity of the algorithm that translates PQLTL formulae into deterministic Rabin automata. If we assume that our PQLTL formulae are in Σ_k^{QLTL}, the size of the deterministic Rabin automata generated is $k + 2$ exponential on the size of the inner LTL formulae, using Proposition A.8 and Proposition A.12 in the Appendix (pages 29 and 31, respectively). The remaining instructions of Algorithm 4 can be carried out in space polynomial in the size of the system. Finally, the last instructions in Algorithm 5 will only use polynomial space, since we do not need to produce an assignment ρ. Thus, assuming that the initial formula is in EQLTL we obtain precisely the same space and time complexity as the algorithms referenced in [10].

Proposition B.1 *Algorithm 5 is correct.*

Proof. The proof follows immediately from the proof of the correction of the model-checking algorithm referenced in [10]. △