Temporalization of probabilistic propositional logic

Pedro Baltazar
Paulo Mateus
*SQIG-Instituto de Telecomunicações and IST, Portugal

October 15, 2008

Abstract

In this paper we study several properties of the Exogenous Probabilistic Propositional Logic (EPPL), a logic for reasoning about probabilities, with the purpose of introducing a temporal version - Exogenous Probabilistic Linear Temporal Logic (EPLTL). In detail, we give a small model theorem for EPPL and introduce a satisfaction and a model checking algorithm for both EPPL and EPLTL. We are also able to provide a (weakly) complete calculus for EPLTL. Finally, we conclude by pointing out some future work.

1 Introduction

Reasoning about probabilistic systems has been a very important research subject with applications in many fields such as security, performance analysis, system verification, traffic analysis and even bioinformatics. In this context, the use of formal methods, and in particular of logic, via syntactic (computer-aided proof systems) and semantic (model-checking tools) approaches, has been highly beneficial to the community.

In this paper we consider a temporalization of the Exogenous Probabilistic Propositional Logic (EPPL) [17] to reason about the evolution of probability distributions described by probabilistic programs and processes. The term exogenous was coined by Kozen in [12] to express that the probabilities had proper syntax and were not hidden in the propositional symbols or connectives (like in PCTL [1]). The state logic is an extension of the probabilistic logic proposed by Fagin et al [9] where we allow to make classical restrictions over probabilistic spaces. EPPL was initially introduced in [16] to reason about quantum states and further developed in the context of a Hoare-like logic [5]. EPPL semantics is obtained

*This work was partially supported by Instituto de Telecomunicações, FCT and EU FEDER through PTDC, namely via QSec PTDC/EIA/67661/2006 Project, IT project QuanTel and European Network of Excellence EURO-NF. Pedro Baltazar also supported by FCT and EU FEDER PhD fellowship SFRH/BD/22698/2005.

by taking the exogenous semantics approach to enrich a given logic–the models of the enriched logic are sets of models of the given logic with additional structure. This approach was inspired by the possible worlds semantics originally proposed by Kripke [13] for modal logic. A model of EPPL is a set of possible valuations over propositional symbols (which, for instance, may denote memory cells of a probabilistic program) along with a probability space that gives the probability of each possible valuation. Indeed, as discussed in this paper, EPPL models can be reformulated more precisely as Bernoulli stochastic processes where the index space is the set of propositional symbols.

EPPL differs significantly from probabilistic *arithmetical* assertion logics, such as the state logic of the probabilistic dynamic logic given in [12], where formulas are interpreted as measurable functions and the connectives are arithmetical operations such as addition and subtraction. Inspired by the dynamic logic in [12], there are several important works in probabilistic Hoare logics, *e.g.* [11, 18], where the state formulas are either measurable functions or arithmetical formulas interpreted as measurable functions. Intuitively, the Hoare triple $\{f\}\, s\, \{g\}$ means that the expected value of the function g after the execution of s is at least as much as the expected value of the function f before the execution. Although research in probabilistic logics with arithmetical state logics has yielded several interesting results, the formulas themselves do not seem very intuitive. Indeed, a high degree of sophistication is required to write down assertions needed to verify relatively simple programs. For this reason, it is worthwhile to investigate dynamic versions of truth-functional probabilistic logics, such as EPPL. In this paper we present in detail a linear temporalization of EPPL, that we call Exogenous Probabilistic Linear Temporal Logic EPLTL.

The contributions of this paper, taking into account the results presented in [17] are significant. We show that we are able to adapt the technique by [9] to obtain a small model theorem with polynomial bound. Capitalizing in the small model theorem we are able to set a PSPACE bound to the SAT problem for EPPL, which was previously thought to be in EXPSPACE [16]. From the SAT algorithm we are able to derive a simpler Hilbert calculus for EPPL than that presented in [17]. We also discuss in details the model-checking of the logic. Moreover, we are able to provide a complete calculus for the temporal extension, EPLTL, together with a SAT and model-checking algorithm.

This paper is structured as follows. In Section 2 we present the main results concerning EPPL. In Section 3 we present the linear temporalization of EPPL, and in Section 4 we point out some future directions. Due to the space constrains, all the proofs are given in the Appendix and will be removed in the final version.

2 Probabilistic state logic

2.1 Syntax

Following the exogenous approach, the language of EPPL consists of formulas at two levels. The formulas of the first level – *basic formulae* – allow us to reason about program variables

and states, that at this point we abstract as a finite set of propositional symbols Λ. The formulas of the second level – *global formulae* – allow us to perform probabilistic reason. We also consider *probabilistic terms*, build over a set of real logic variables Z, to denote real numbers used for quantitative reasoning at the global formulae level. The syntax of the language is given by mutual recursion as presented in Table 1.

$\beta := p \,[\!]\, (\neg\beta) \,[\!]\, (\beta \Rightarrow \beta)$	basic formulae
$t := z \,[\!]\, 0 \,[\!]\, 1 \,[\!]\, (\int \beta) \,[\!]\, (t + t) \,[\!]\, (t \cdot t)$	probabilistic terms
$\delta := (\Box\beta) \,[\!]\, (\beta \perp\!\!\!\perp \beta) \,[\!]\, (t \leq t) \,[\!]\, (\sim\delta) \,[\!]\, (\delta \supset \delta)$	global formulae

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

Table 1: EPPL syntax

Concerning basic formulae, ranged over by $\beta, \beta_1, \ldots$, we assume the usual propositional abbreviations for falsum $\bot$, disjunction $(\beta_1 \vee \beta_2)$, conjunction $(\beta_1 \wedge \beta_2)$ and equivalence $(\beta_1 \Leftrightarrow \beta_2)$.

The probability terms, ranged over by $t, t_1, \ldots$, denote the real numbers. We assume a finite set of (deterministic) real variables, Z, ranging over algebraic real numbers. The probability terms also contain the 0 and 1 real constants that, together with addition, multiplication and the set of logical variables, allow us to express all algebraic real numbers [2]. The probability term $(\int \beta)$ denotes the probability of the set of elements that satisfy β. The terms of the kind $(\int \beta)$ shall henceforth be called *measure terms*.

The global formulas, ranged over by $\delta, \delta_1, \ldots$, are built from modal formulas $(\Box\beta)$, independence formulas $(\beta_1 \perp\!\!\!\perp \beta_2)$, comparison formulas $(t_1 \leq t_2)$ and the connectives $\sim, \supset$. The modal formula $(\Box\beta)$ allows us to impose restrictions on the probability space, namely to impose that all elements of the sample space satisfy β. We shall also use $(\Diamond\beta)$ as an abbreviation for $(\sim(\Box(\neg\beta)))$. Intuitively, $(\Diamond\beta)$ is satisfied if there is at least one valuation in the probability measure that satisfies β. Observe that $\Box$ and $\Diamond$ are not full fledged modalities, since they cannot be nested[1]. The independence formulas $(\beta_1 \perp\!\!\!\perp \beta_2)$ states that the event described by β_1 is independent from the event β_2.

Other global connectives $\{\mathbf{f}, \cup, \cap, \equiv\}$ and comparison predicates $\{=, \neq, \geq, <, >\}$ are introduced as abbreviations in the classical way. For instance, the global falsum $\mathbf{f}$ stands for $(\Box p \cap (\sim\Box p))$ and $(t_1 = t_2)$ stands for $((t_1 \leq t_2) \cap (t_2 \leq t_1))$.

The notion of occurrence of a term t and a global formula δ_1 in the global formula δ is defined as usual. The same holds for the notion of replacing zero or more occurrences of probability terms and global formulas. For the sake of clarity, we shall often drop parentheses in formulas and terms if it does not lead to ambiguity.

We shall also identify here a useful sublanguage of probabilistic state formulas which do not contain any occurrence of a measure term.

$$\begin{array}{lcl} \kappa & := & (\alpha \leq \alpha) \,[\!]\, (\sim\kappa) \,[\!]\, (\kappa \supset \kappa) \\ \alpha & := & z \,[\!]\, 0 \,[\!]\, 1 \,[\!]\, (\alpha + \alpha) \,[\!]\, (\alpha.\alpha) \end{array}$$

[1] We do not have formulas such as $\Box(\Box\beta)$.

The terms of this sublanguage will be called *analytical terms* and the formulas will be called *analytical formulas*. This language is relevant because it is possible to apply the SAT algorithm for the existential theory of the real numbers to any analytical formula.

2.2 Semantics

The models of EPPL are tuples $m = (\Omega, \mathcal{F}, \mu, \mathbf{X})$ where $(\Omega, \mathcal{F}, \mu)$ is a probability space and $\mathbf{X} = (X_p)_{p\in\Lambda}$ is a *stochastic process* over $(\Omega, \mathcal{F}, \mu)$ where each X_p is a Bernoulli random variable, that is, X_p ranges over $\mathbf{2} = \{0, 1\}$. Observe that each basic EPPL formula β induces a Bernoulli random variable $X_\beta : \Omega \to \mathbf{2}$ defined as follows: $X_{(\neg\beta)}(\omega) = 1 - X_\beta(\omega)$; and $X_{(\beta_1\Rightarrow\beta_2)}(\omega) = \max((1 - X_{\beta_1}(\omega)), X_{\beta_2}(\omega))$. So, each basic formula β will represent the measurable subset $\{\omega \in \Omega : X_\beta(\omega) = 1\}$. Moreover, each $\omega \in \Omega$ induces a valuation v_ω over Λ such that $v_\omega(p) = X_p(\omega)$, for all $p \in \Lambda$. Given an EPPL model $m = (\Omega, \mathcal{F}, \mu, \mathbf{X})$ and attribution $\rho : Z \to \mathbb{R}$ for real variables, the denotation of probabilistic terms is as following: $[\![z]\!]_{m,\rho} = \rho(z)$; $[\![0]\!]_{m,\rho} = 0$; $[\![1]\!]_{m,\rho} = 1$; $[\![t_1 + t_2]\!]_{m,\rho} = [\![t_1]\!]_{m,\rho} + [\![t_2]\!]_{m,\rho}$; $[\![t_1.t_2]\!]_{m,\rho} = [\![t_1]\!]_{m,\rho}.[\![t_2]\!]_{m,\rho}$; and $[\![(\int\beta)]\!]_{m,\rho} = \int X_\beta \, d\mu = \mu(X_\beta^{-1}(1))$ is the expected value of X_β, that is, the probability of observing an outcome ω such that v_ω satisfies β.

Moreover, the satisfaction of global formulas is given by: $m, \rho \Vdash (\Box\beta)$ iff $\Omega = X_\beta^{-1}(1)$; $m, \rho \Vdash (\beta_1 \perp\!\!\!\perp \beta_2)$ iff $X_{\beta_1} \perp\!\!\!\perp X_{\beta_2}$; $m, \rho \Vdash (t_1 \leq t_2)$ iff $[\![t_1]\!]_{m,\rho} \leq [\![t_2]\!]_{m,\rho}$; $m, \rho \Vdash (\sim\delta)$ iff $m, \rho \nVdash \delta$; and $m, \rho \Vdash (\delta_1 \supset \delta_2)$ iff $m, \rho \Vdash \delta_2$ or $m, \rho \nVdash \delta_1$.

Probabilistic terms without occurrences of real variables are called *closed terms*. A global formula only involving closed terms is called a *closed global formula*. Clearly, the denotation of closed terms is independent of the attribution. Consequently, the satisfaction of closed global formulas are also independent of the attribution. So, in these cases, we drop the attribution from the notation.

Remark 2.1 To design a SAT algorithm for EPPL it is important to make some observations on EPPL models. Let $V_m = \{v_\omega : \omega \in \Omega\}$ be the set of all valuations over Λ induced by m. The *basic cylinders*, also called *rectangles*, of an EPPL model m are the subsets $B(b_1 \dots b_k) = \{v \in V_m : v(p_1) = b_1, \dots, v(p_k) = b_k\}$ for $k \geq 0$, $p_1, \dots, p_k \in \Lambda$ and $b_1, \dots, b_k \in \mathbf{2}$. Let $\mathcal{B}_m$ be the set of all basic cylinders of m. Observe that an EPPL model $m = (\Omega, \mathcal{F}, \mu, \mathbf{X})$ induces a probability space $P_m = (V_m, \mathcal{F}_m, \mu_m)$ over valuations, where $\mathcal{F}_m \subseteq \mathbf{2}^{V_m}$ is the σ-algebra generated by the basic cylinders $\mathcal{B}_m$ and μ_m is defined over basic cylinders by $\mu_m(B) = \mu(\{\omega \in \Omega : v_\omega \in B\})$ for all $B \in \mathcal{B}_m$. Moreover, given a probability space over valuations, $P = (V, \mathcal{F}, \mu)$ we can construct an EPPL model $m_P = (V, \mathcal{F}, \mu, \mathbf{X})$ where $X_p(v) = v(p)$. It is easy to see that m and m_{P_m} satisfy precisely the same formulas. This means that it is enough for a SAT algorithm to search for probability spaces over valuations.

Given that we are working towards a complete Hilbert calculus for EPPL through a SAT algorithm, it is relevant to understand whether EPPL fulfills a small model theorem. If this is the case then an upper bound on the size of the satisfying models would imply the decidability of the logic (since it would be enough to search for models up to this bound).

Remark 2.2 *The semantic of the independence formulas* $(\beta_1 \perp\!\!\!\perp \beta_2)$ *allow us to substitute in global formulas all its occurrences by the conjunction*

$$((\int \beta_1 \wedge \beta_2) \leq (\int \beta_1)(\int \beta_2)) \bigcap ((\int \beta_1)(\int \beta_2) \leq (\int \beta_1 \wedge \beta_2)).$$

2.3 Small model theorem

To obtain a small model theorem we start by defining a quotient construction. Let δ be an EPPL formula. We denote the sets of inequalities and basic subformulas occurring in δ by $iq(\delta)$ and $bsf(\delta)$, respectively. Moreover, we denote the (finite) set of propositional symbols that appear in δ by $prop(\delta)$. Given a formula δ and an EPPL model $m = (\Omega, \mathcal{F}, \mu, \mathbf{X})$, we define the following relation on the sample space Ω: $\omega_1 \sim_\delta \omega_2$ iff $X_p(\omega_1) = X_p(\omega_2)$ for all $p \in prop(\delta)$.

Let $prop_\omega(\delta)$ be the subset of propositional symbols of δ such that $X_p(\omega) = 1$. We denote by $[\omega]_\delta$ the $\sim_\delta$ class of ω. Taking an EPPL model $m = (\Omega, \mathcal{F}, \mu, \mathbf{X})$ and an EPPL formula δ, we define the *quotient model* $m/\sim_\delta = (\Omega', \mathcal{F}', \mu', \mathbf{X}')$ where: $\Omega' = \Omega/\sim_\delta$ is the finite set of $\sim_\delta$ classes; $\mathcal{F}' = 2^{\Omega'}$ is the power set σ-algebra; $\mu'(B) = \mu(\cup B)$ for all $B \in \mathcal{F}'$; and $X'_p([\omega]_\delta) = X_p(\omega)$ for all $p \in \Lambda$.

Next, we check that the quotient model is well defined.

Proposition 2.3 Let $m = (\Omega, \mathcal{F}, \mu, \mathbf{X})$ be an EPPL model and δ an EPPL formula, then $m/\sim_\delta = (\Omega', \mathcal{F}', \mu', \mathbf{X}')$ is a finite EPPL model .

Now, we prove that satisfaction is preserved by the quotient construction and, consequentially, that any satisfiable formula has a finite discrete EPPL model of size bounded by the formula length. We take the *length* of a basic formula, probabilistic terms or global formula, to be the number of symbols required to write the formula or term. The length of a formula or term ξ is denoted by $|\xi|$.

We are now able to establish a small model theorem for EPPL. We refer the reader to the Appendix for a detailed proof of the result. Observe that at first sight, to construct a model for a formula δ, we need $O(2^{|\delta|})$ algebraic real numbers to describe the probability measure of the σ-algebra over the propositional symbols occurring in δ. However, adapting a technique for eliminating spurious variables in linear programming (already used in [9]), we are able to set this bound to be just linear.

Theorem 2.4 (Small Model Theorem) *If* δ *is a satisfiable EPPL formula then it has a finite model using at most* $2|\delta| + 1$ *algebraic real numbers.*

The small model theorem does not put a bound on the size of the representation of the algebraic real numbers. Indeed, an algebraic real number can be represented as the root of a polynomial of integers, and this polynomial could increase without any bound. Fortunately, thanks to the fact that the existential theory of the real numbers can be decided in PSPACE [4], we find a bound on the size of the real representations in function of the size of the formula, which will lead to a SAT algorithm for EPPL.

2.4 Decision algorithm for satisfaction

The decision algorithm for EPPL satisfaction uses the decidability of the existential theory of the real numbers and the small model theorem. Before presenting the algorithm we introduce some notation. Like before, given an EPPL formula δ we will denote by $iq(\delta)$ the set of all subformulas of δ of the form $(t_1 \leq t_2)$. Moreover, we denote by $bf_{\Box}(\delta)$ the set of all subformulas of δ of the form $\Box\beta$, by $ip(\delta)$ the set of all subformulas of the form $\beta_1 \perp\!\!\!\perp \beta_2$ and by $at(\delta)$ the set of all global atoms of δ, that is, $at(\delta) = bf_{\Box}(\delta) \cup iq(\delta) \cup ip(\delta)$. By an *exhaustive conjunction ε of literals of $at(\delta)$* we mean that ε is of the form $\alpha_1 \cap \ldots \cap \alpha_k$ where each α_i is either a global atom or a negation of a global atom. Moreover, all global atoms or their negation occur in ε, so, $k = |at(\delta)|$. At this stage, we consider the EPPL formula $\hat{\varepsilon}$ where in ε all global atoms $\beta_1 \perp\!\!\!\perp \beta_2$ are substitute by the global conjunction in Remark 2.2. Given a global formula δ, we denote by $\delta^{\alpha}_{p_\alpha}$ the propositional formula obtained by replacing in δ each global atom α with a fresh propositional symbol p_α, and replacing the global connectives $\sim$ and $\supset$ by the propositional connectives $\neg$ and $\Rightarrow$, respectively. We denote by v_ε the valuation over propositional symbols p_α such that $v_\epsilon(p_\alpha) = 1$ iff α occurs positively in ε.

Given an exhaustive conjunction ε of literals of $at(\delta)$, we denote by $lbf_{\Box}(\varepsilon)$ the set of basic formulas such that $\beta \in lbf_{\Box}(\varepsilon)$ if $\Box\beta$ occurs positively in ε (that is, not negated). Similarly, the set of basic formulas that occur nested by a $\sim\Box$ in ε is denoted by $lbf_{\Diamond\neg}(\varepsilon)$. Finally, we denote all the inequalities occurring in $\hat{\varepsilon}$ by $liq(\varepsilon)$. This last set contains the new inequations introduced by the substitution of the independence formulas.

Given a global formula α in $liq(\varepsilon)$ we denote by $\hat{\alpha}$ the analytical formula where all terms of the form $(\int\beta)$ are replaced in α by $\sum_{v \in V, v \Vdash \beta} x_v$ where each x_v is a fresh variable. We need the PSPACE SAT algorithm of the existential theory of the reals numbers [4], that we denote by *SatReal*. We assume that this algorithm either returns *no model*, if there is no solution for the input system of inequations, or a *solution array* ρ, where $\rho(x)$ is the solution for variable x. We denote by $var(\delta)$ the set of real logical variables that occur in δ. Given a solution ρ for a system with X variables and a subset $Y \subseteq X$, we denote by $\rho_{|Y}$ the function that maps each element y of Y to $\rho(y)$.

Theorem 2.5 Algorithm 1 decides the satisfiability of an EPPL formula in PSPACE.

2.5 Completeness

In [17] it is shown that a superset of axioms and inference rules in Table 2 is a sound and a (weakly) complete axiomatization of EPPL. Herein, and thanks to the EPPL SAT algorithm, we are able to show that the calculus presented in Table 2 is weakly complete.

It is impossible to obtain a strongly complete axiomatization for EPPL (that is, if $\Delta \Vdash \delta$ then $\Delta \vdash \delta$, for arbitrary large Δ, possibly infinite set) because the logic is not compact [17]. Nevertheless, weakly completeness is enough for software verification, since a program specification generates a finite number of hypotheses.

Algorithm 1: SatEPPL(δ)

Input: EPPL formula δ
Output: (V, μ) (denoting the EPPL model $m = (V, 2^V, \mu, \mathbf{X})$) and attribution ρ or *no model*

compute $bf_{\Box}(\delta), ip(\delta), iq(\delta)$ *and* $at(\delta)$*;*
foreach exhaustive conjunction ε of literals of $at(\delta)$ such that $v_\varepsilon \Vdash \delta^{\alpha}_{p_\alpha}$ **do**
  compute $lbf_{\Box}(\varepsilon)$, $lbf_{\Diamond\neg}(\varepsilon)$ and $liq(\varepsilon)$*;*
  foreach $V \subseteq 2^{prop(\delta)}$ such that $0 < |V| \leq 2|\delta| + 1$, $V \Vdash \wedge lbf_{\Box}(\varepsilon)$ and $V \nVdash \beta$ for all $\beta \in lbf_{\Diamond\neg}(\varepsilon)$ **do**
    $\kappa \longleftarrow \left(\sum_{v\in V} x_v = 1\right) \cap \left(\bigcap_{v\in V} 0 \leq x_v\right)$;
    foreach $\alpha \in liq(\varepsilon)$ **do**
      $\kappa \longleftarrow \kappa \cap \hat{\alpha}$;
    end
    $\rho \longleftarrow SatReal(\kappa)$;
    if $\rho \neq$ *no model* **then**
      $\mu_\rho \longleftarrow \rho_{|\{x_v : v\in V\}}$;
      $\rho_\rho \longleftarrow \rho_{|var(\delta)}$;
      return (V, μ_ρ) and attribution ρ_ρ;
    end
  end
end
return (*no model*);

Concerning the axiomatization of Table 2, 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. Thus, the axioms in Table 2 constitute a recursive set. Note that the **ROF** axiom allow us to separate the reasoning about probabilities from the reasoning about real numbers.

We can simplify the proof in [17] thanks to the SAT algorithm presented before. The soundness of the calculus of Table 2 is straightforward, and so, we focus on the completeness result. Again, we refer the reader to the Appendix for a detailed proof.

Theorem 2.6 *The set of rules and axioms of Table 2 is a weakly complete axiomatization of EPPL.*

2.6 Model Checking

Given a finite set of propositional symbols Λ and using the small model theorem we can consider that all EPPL models are defined over a discrete and finite probability space.

Since for the model-checking procedure we have to deal with computer representation and, in practice, probabilities are represented by floating points and not symbolically by algebraic real numbers, we consider only EPPL models $m = (\Omega, \mathcal{F}, \mu, \mathbf{X})$ specified with

Axioms

[**CTaut**]	$\vdash_{\mathsf{EPPL}}$	$(\Box\beta)$ for each valid propositional formula β;
[**GTaut**]	$\vdash_{\mathsf{EPPL}}$	δ for each instantiation of a propositional tautology δ;
[**Lift** $\Rightarrow$]	$\vdash_{\mathsf{EPPL}}$	$(\Box(\beta_1 \Rightarrow \beta_2) \supset (\Box\beta_1 \supset \Box\beta_2))$;
[**Eqv** $\bot$]	$\vdash_{\mathsf{EPPL}}$	$(\Box\bot \equiv \mathbf{f})$;
[**Indep**]	$\vdash_{\mathsf{EPPL}}$	$(\beta_1 \perp\!\!\!\perp \beta_2) \equiv ((\int \beta_1 \wedge \beta_2) = (\int \beta_1)(\int \beta_2))$;
[**ROF**]	$\vdash_{\mathsf{EPPL}}$	$(t_1 \leq t_2)$ for each instantiation of a valid analytical inequality;
[**Prob**]	$\vdash_{\mathsf{EPPL}}$	$((\int \top) = 1)$;
[**FAdd**]	$\vdash_{\mathsf{EPPL}}$	$(((\int(\beta_1 \wedge \beta_2)) = 0) \supset ((\int(\beta_1 \vee \beta_2)) = (\int \beta_1) + (\int \beta_2)))$;
[**Mon**]	$\vdash_{\mathsf{EPPL}}$	$(\Box(\beta_1 \Rightarrow \beta_2) \supset ((\int \beta_1) \leq (\int \beta_2)))$;

Inference rules

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

Table 2: HC_{EPPL} : complete calculus for EPPL

floating point arrays (like is usual in other probabilistic model checkers, such as PRISM [15, 14]). Observe that, since floating point numbers are rational numbers, they are also algebraic real numbers and so, the semantics given in Section 2.2 does not require any modification to deal with floating points. We represent an EPPL model as a $|\Lambda| \times |\Omega|$-matrix $\mathbf{X}$ of boolean values for the random variables and an $|\Omega|$-array μ of real numbers for the probabilities. The size of Ω is at most $2^{|\Lambda|}$. So, an EPPL models is stored in memory by the record $(\mu, \mathbf{X})$.

Let δ be an EPPL global formula. We consider that in δ we have already replace all occurrences of independence formulas $(\beta_1 \perp\!\!\!\perp \beta_2)$ by inequalities as describe in Remark 2.2.

We define the arrays $bsf(\delta) = (\beta_1, \ldots, \beta_k)$, $pst(\delta) = (t_1, \ldots, t_s)$ and $gsf(\delta) = (\delta_1, \ldots, \delta_m, \delta)$ as the ordered tuples of basic subformulas, probabilistic subterms and global subformulas of δ, respectively, ordered by increasing length. An attribution ρ for real logical variables is also represented by a finite array where the dimension is determined by the number of real logical variables in the formula $|\delta|$, that is bounded by s (the length of $pst(\delta)$). As usual for floating points, we assume that the basic arithmetical operations take $O(1)$ time.

Given an EPPL model $m = (\Omega, \mathcal{F}, \mu, \mathbf{X})$, an attribution ρ and a global formula δ, *the model-checking problem* consists in determining whether $m, \rho \Vdash \delta$. Model checking of EPPL is detailed in Algorithm 2.

Theorem 2.7 Assuming that all basic arithmetical operations and that accessing array/matrix values take $O(1)$ time, Algorithm 2 takes $O(|\delta| \cdot |\Omega|)$ time to decide if an EPPL model $m = (\Omega, \mathcal{F}, \mu, \mathbf{X})$ and attribution ρ satisfy δ.

Algorithm 2: CheckEPPL(m, ρ, δ)

Input: EPPL model $m = (\mu, \mathbf{X})$, attribution ρ and a formula δ
Output: Boolean value $G(|gsf(\delta)|)$

for $i = 1$ **to** $|bsf(\delta)|$ **do** /* this cycle iterates $O(|\delta|)$ times */
  switch β_i **do** /* each case takes $O(|\Omega|)$ time */
    case $p:\ B(i) = X_p$;
    case $(\neg\beta_j):\ B(i) = 1 - B(j)$;
    case $(\beta_j \Rightarrow \beta_l):\ B(i) = \max(1 - B(j), B(l))$;
  end
end
for $i = 1$ **to** $|pst(\delta)|$ **do** /* this cycle iterates $O(|\delta|)$ times */
  switch t_i **do** /* each case takes $O(|\Omega|)$ */
    case $z:\ T(i) = \rho(z)$:;
    case 0 **or** 1 : $T(i) = t_i$;
    case $(\int\beta_j) : T(i) = B(j).\mu$; /* this case takes $O(2|\Omega|)$ */
    case $(t_j + t_l):\ T(i) = T(j) + T(l)$;
    case $(t_j.t_l):\ T(i) = T(j).T(l)$;
  end
end
for $i = 1$ **to** $|gsf(\delta)|$ **do** /* this cycle iterates $O(|\delta|)$ times */
  switch δ_i **do** /* each case takes $O(|\Omega|)$ */
    case $(\Box\beta_j):\ G(i) = \Pi_{l=1}^{|\Omega|} B(j,l)$; /* this case takes $O(|\Omega| - 1)$ */
    case $(t_j \leq t_l):\ G(i) = (T(j) \leq T(l))$;
    case $(\delta_j \supset \delta_l):\ G(i) = \max(1 - G(j), G(l))$;
  end
end

3 Probabilistic Linear Time Logic

3.1 Linear Time Logic

Syntax. We assume that there is a countable set of propositional symbols Ξ. Assuming the set Ξ, the formulas of Linear Time Logic (LTL) are given in BNF notation as

$$\theta := \mathbf{f} \ [\!] \ p \ [\!] \ (\theta \supset \theta) \ [\!] \ \mathsf{X}\theta \ [\!] \ \theta\mathsf{U}\theta$$

where $p \in \Xi$.

Semantics. The semantics of the temporal logic LTL is also given using a Kripke structure. A *Kripke structure* over a set of propositions Ξ is a tuple $\mathcal{K} = (\mathsf{S}, \mathsf{R}, \mathsf{L})$ where S is a set, elements of which are called *states*; $\mathsf{R} \subseteq S \times S$ is a said to be the *accessibility relation* and it is assumed that for every $s \in \mathsf{S}$ there exists $s' \in \mathsf{S}$ such that $(s, s') \in \mathsf{R}'$; and $\mathsf{L} : \mathsf{S} \to \wp(\Xi)$ is said to be a *labeling function*. Given a Kripke structure, $\mathcal{K} = (\mathsf{S}, \mathsf{R}, \mathsf{L})$, an

infinite sequence of states $\pi = s_1 s_2 \ldots$ is said to be a *computation path* if $(s_i, s_{i+1}) \in \mathsf{R}$ for all $i \geq 1$. The semantics of LTL is defined in terms of a Kripke structure $\mathcal{K}$ and a computation path π. The LTL modalities contain symbols for temporal reasoning: X stands for ne**x**t; and U for **u**ntil. The remaining temporal modalities, F and G, are easily obtained by abbreviation: $(\mathsf{F}\theta)$ for $((\sim\mathbf{f})\mathsf{U}\theta)$; and $(\mathsf{G}\theta)$ for $(\sim\mathsf{F}(\sim\theta))$.

Given a Kripke structure $\mathcal{K}$, a computation path $\pi = s_1 \ldots$ of the Kripke structure, and a LTL formula θ, the formal semantics is defined inductively in terms of a relation $\mathcal{K}, \pi \Vdash \theta$ and is given in Table 3. We denote by π^i the i-th suffix of π, that is, the path $s_i, s_{i+1} \ldots$

$\mathcal{K}, \pi \nVdash_{\mathsf{LTL}} \mathbf{f}$;		
$\mathcal{K}, \pi \Vdash_{\mathsf{LTL}} p$	iff	$p \in \mathsf{L}(s_1)$ with $\pi = s_1, \ldots$;
$\mathcal{K}, \pi \Vdash_{\mathsf{LTL}} (\theta_1 \supset \theta_2)$	iff	$\mathcal{K}, \pi \nVdash_{\mathsf{LTL}} \theta_1$ or $\mathcal{K}, \pi \Vdash_{\mathsf{LTL}} \theta_2$;
$\mathcal{K}, \pi \Vdash_{\mathsf{LTL}} \mathsf{X}\theta$	iff	$\mathcal{K}, \pi^2 \Vdash_{\mathsf{LTL}} \theta$;
$\mathcal{K}, \pi \Vdash_{\mathsf{LTL}} (\theta_1 \mathsf{U} \theta_2)$	iff	there is some $i \geq 1$ such that $\mathcal{K}, \pi^i \Vdash_{\mathsf{LTL}} \theta_2$ and $\mathcal{K}, \pi^j \Vdash_{\mathsf{LTL}} \theta_1$ for $1 \leq j < i$.

Table 3: Semantics of LTL

A Kripke strucute $\mathcal{K}$ is a model of (or satisfies) the formula θ if $\mathcal{K}, \pi \Vdash \theta$ for every path π in $\mathcal{K}$. As usual, we say that a set of formulas Θ entails the formula θ, which we write $\Theta \vDash \theta$, if a Kripke structure satisfying all the formulas in Θ also satisfies θ.

Axiomatization. The temporal logic LTL enjoys a sound and complete axiomatization. The proof system HC_{LTL} of LTL is given in Table 4.

The following result is proved in [10].

Theorem 3.1 *The proof system HC_{LTL} is sound and weakly complete with respect to Kripke structures.*

Axioms		
[**Taut**]		All propositional tautologies with propositional symbols substituted by LTL formulas;
[**X1**]	$\vdash_{\mathsf{LTL}}$	$(\sim\mathsf{X}\theta_1) \equiv (\mathsf{X}\sim\theta_1)$
[**X2**]	$\vdash_{\mathsf{LTL}}$	$(\mathsf{X}(\theta_1 \supset \theta_2)) \supset (\mathsf{X}\theta_1 \supset \mathsf{X}\theta_2)$
[**G**]	$\vdash_{\mathsf{LTL}}$	$(\mathsf{G}\theta_1) \supset (\theta_1 \cap (\mathsf{X}\mathsf{G}\theta_1))$
[**U1**]	$\vdash_{\mathsf{LTL}}$	$(\theta_1 \mathsf{U} \theta_2) \supset (\mathsf{F}\theta_2)$
[**U2**]	$\vdash_{\mathsf{LTL}}$	$(\theta_1 \mathsf{U} \theta_2) \equiv (\theta_2 \cup (\theta_1 \cap \mathsf{X}(\theta_1 \mathsf{U} \theta_2)))$
Inference rules		
[**MP**]	$\theta_1, (\theta_1 \supset \theta_2) \vdash_{\mathsf{LTL}} \theta_2$	
[**XGen**]	$\theta_1 \vdash_{\mathsf{LTL}} (\mathsf{X}\theta_1)$	
[**Ind**]	$(\theta_1 \supset \theta_2), (\theta_1 \supset (\mathsf{X}\theta_1)) \vdash_{\mathsf{LTL}} (\theta_1 \supset (\mathsf{G}\theta_2))$	

Table 4: HC_{LTL} : complete calculus for LTL

3.2 Exogenous Probabilistic Linear Time logic

Syntax. The formulas of Exogenous Probabilistic Linear Time Logic (EPLTL) are obtained by enriching the probabilistic formulas with LTL modalities and are depicted in Table 5. The temporal modalities $\mathsf{F}\theta$ and $\mathsf{G}\theta$ are introduced as abbreviations. Observe

$$\theta \quad := \quad \mathbf{f} \,[\!]\, \gamma \,[\!]\, (\theta \supset \theta) \,[\!]\, (\mathsf{X}\theta) \,[\!]\, (\theta \mathsf{U} \theta) \text{ where } \gamma \text{ is an EPPL formula.}$$

Table 5: Language of EPLTL

that the connectives $\mathbf{f}$ and $\supset$ are shared with EPPL.

Semantics. We now provide a semantics for EPLTL based on EPPL-parametrized Kripke structures. An *EPPL-parametrized Kripke structure* is a tuple $\mathcal{T} = (\mathsf{S}, \mathsf{R}, \mathsf{L})$, where S is a non-empty set of states; $\mathsf{R} \subseteq \mathsf{S} \times \mathsf{S}$ is a total relation; and a function L such that $\mathsf{L}(s)$ is a pair (m, ρ), for each $s \in S$, where m is an EPPL model and ρ is an attribution. Like for Kripke structures, a computation path is a infinite sequence $\pi = m_1\rho_1, m_2\rho_2 \ldots$ such that for any $i \geq 1$, we have $(m_i\rho_i, m_{i+1}\rho_2) \in \mathsf{R}$. Given an EPPL-Kripke structure $\mathcal{T}$, a computational path π in $\mathcal{T}$ and a EPLTL formula θ, the semantics of EPLTL is defined in terms of a relation $\mathcal{T}, \pi \Vdash_{\mathsf{EPLTL}} \gamma$ given in Table 6.

$\mathcal{T}, \pi \nVdash_{\mathsf{EPLTL}} \mathbf{f}$		
$\mathcal{T}, \pi \Vdash_{\mathsf{EPLTL}} \gamma$	iff	$m_1, \rho_1 \Vdash_{\mathsf{EPPL}} \gamma$;
$\mathcal{T}, \pi \Vdash_{\mathsf{EPLTL}} (\theta_1 \supset \theta_2)$	iff	$\mathcal{T}, \pi \nVdash_{\mathsf{EPLTL}} \theta_1$ or $\mathcal{T}, \pi \Vdash_{\mathsf{EPLTL}} \theta_2$;
$\mathcal{T}, \pi \Vdash_{\mathsf{EPLTL}} (\mathsf{X}\theta)$	iff	$\mathcal{T}, \pi^2 \Vdash_{\mathsf{EPLTL}} \theta$;
$\mathcal{T}, \pi \Vdash_{\mathsf{EPLTL}} (\theta_1 \mathsf{U} \theta_2)$	iff	there is some $i \geq 1$ such that $\mathcal{T}, \pi^i \Vdash_{\mathsf{EPLTL}} \theta_2$ and $\mathcal{T}, \pi^j \Vdash_{\mathsf{EPLTL}} \theta_1$ for $1 \leq j < i$.

Table 6: Semantics of EPLTL

An EPPL-Kripke structure $\mathcal{T}$ is said to satisfy an EPLTL formula θ, which we denote by $\mathcal{T} \Vdash_{\mathsf{EPLTL}} \theta$, if $\mathcal{T}, \pi \Vdash_{\mathsf{EPLTL}} \theta$ for all computational paths π in $\mathcal{T}$. The entailement relation is defined as for LTL.

3.3 Axiomatization

We are able to provide a weakly complete axiomatization of EPLTL capitalizing on the complete LTL calculus HC_{LTL}, which we present in Table 7. Please note that although the completeness of the calculus may look trivial, the proof of completeness is subtle. This is because the connectives $\mathbf{f}$ and $\supset$ are shared between EPPL and LTL which may create new theorems that would not be obtained by just adding the EPPL axioms to LTL axioms.

It is straightforward to check the soundness of the calculus, for this reason we omit here the lengthy exercise of verifying that all axioms and inference rules are sound.

Axioms

[**PTeo**]	All EPPL theorems;
[**LTLTaut**]	All LTL tautologies with propositional symbols substituted by EPLTL formulas.

Inference rules

[**PMP**]	$\theta_1, (\theta_1 \supset \theta_2) \vdash_{\mathsf{QCTL}} \theta_2$;
[**Gen**]	$\theta_1 \vdash_{\mathsf{LTL}} \mathsf{G}\theta_1$.

Table 7: HC_{EPLTL} calculus for EPLTL

Theorem 3.2 (Soundness) *The axiomatization HC_{EPLTL} is sound.*

The completeness of the calculus is established by translating probabilistic atoms into propositional symbols. Consider the subset of atomic EPPL formulas pAtom (*i.e.*, the set constituted by box formula ($\Box\beta$), independence formulas ($\beta_1 \perp\!\!\!\perp \beta_2$) and comparison formulas ($t_1 \leq t_2$)). Let Ξ be the countable set of propositional symbols used to write LTL formulas. Given a fixed bijective map $\lambda : \mathsf{pAtom} \rightarrow \Xi$ (that translates each global atom to a LTL propositional symbol) we can translate each EPPL formula θ to a LTL formula $\lambda(\theta)$ by extending inductively λ on the structure of the formula θ (and preserving all connectives). For simplicity, we denote $\lambda(\theta)$ just by $\widetilde{\theta}$. The map λ can also be used to translate an EPPL-Kripke structure $\mathcal{T} = (\mathsf{S}, \mathsf{R}, \mathsf{L})$ to the Kripke structure $\widetilde{\mathcal{T}} = (\mathsf{S}, \mathsf{R}, \widetilde{\mathsf{L}})$, where $p \in \widetilde{\mathsf{L}}(s)$ if $\mathsf{L}(s) \Vdash_{\mathsf{EPPL}} \lambda^{-1}(p)$.

Lemma 3.3 *Let $\mathcal{T}$ be an EPPL-Kripke structure. Then, $\mathcal{T}, \pi \Vdash_{EPLTL} \theta$ iff $\widetilde{\mathcal{T}}, \pi \Vdash_{LTL} \widetilde{\theta}$.*

The next lemma shows that EPLTL incorporates both LTL and EPPL reasoning.

Lemma 3.4 *Let θ be and EPLTL formula, if $\vdash_{LTL} \widetilde{\theta}$ then $\vdash_{EPLTL} \theta$. Moreover, let γ be an EPPL formula, if $\vdash_{EPPL} \gamma$ then $\vdash_{EPLTL} \gamma$.*

Proof: Follows directly from axioms **LTLTaut** and **PTeo**.

If one restricts just to EPPL formulas, EPLTL reasoning coincides with that of EPPL.

Lemma 3.5 *Let γ be an EPPL formula. Then $\vdash_{EPLTL} \gamma$ iff $\vdash_{EPPL} \gamma$.*

The following lemma is crucial to the proof of completeness.

Lemma 3.6 *Let θ be an EPLTL formula such that $\Vdash_{EPLTL} \theta$. Then there is an EPPL formula γ_θ such that $\vdash_{EPLTL} \gamma_\theta$ and $\Vdash_{LTL} (\mathsf{G}\widetilde{\gamma}_\theta \supset \widetilde{\theta})$.*

We are now able to show the completeness of HC_{EPLTL}.

Theorem 3.7 *The axiomatization HC_{EPLTL} is weakly complete.*

Proof: Let $\Vdash_{\mathsf{EPLTL}} \theta$ be a valid EPLTL formula. With γ_θ as in Lemma 3.6 we have that $\Vdash_{\mathsf{LTL}} (\mathsf{G}\widetilde{\gamma}_\theta \supset \widetilde{\theta})$. Using LTL completeness we have $\vdash_{\mathsf{LTL}} (\mathsf{G}\widetilde{\gamma}_\theta \supset \widetilde{\theta})$. Now, from Lemma 3.4 we get $\vdash_{\mathsf{EPLTL}} (\mathsf{G}\gamma_\theta \supset \theta)$.

Hence, we are able do the following derivation in EPLTL:

1)	$\vdash_{\mathsf{EPLTL}} \gamma_\theta$	Lemma 3.6
2)	$\vdash_{\mathsf{EPLTL}} (\mathsf{G}\gamma_\theta)$	Rule **Gen** at 1
3)	$\vdash_{\mathsf{EPLTL}} (\mathsf{G}\gamma_\theta \supset \theta)$	Lemma 3.6,Lemma 3.4
4)	$\vdash_{\mathsf{EPLTL}} \theta$	Rule **PMP** at 2,3

Therefore, HC_{EPLTL} is complete.

3.4 SAT problem

Let θ be the EPLTL formula that we want to test for satisfiability and $at = \{\gamma_1, \ldots, \gamma_k\}$ be the set of atomic EPPL formulas that are atoms of θ. Now for each k-vector $i \in \{0,1\}^k$, consider the EPPL formula

$$\delta_i = \prod_{j=1}^{k} \varphi_j \qquad \text{where} \qquad \varphi_j = \begin{cases} \gamma_j & \text{if } j\text{-th bit of } i \text{ is } 1 \\ (\sim\gamma_j) & \text{otherwise} \end{cases} \tag{1}$$

Let $K \subseteq \{0,1\}^k$ be such that δ_i is an EPPL consistent formula and let $\gamma_\theta = \bigsqcup_{i \in K} \delta_i$. Observe that $\Vdash_{\mathsf{EPPL}} \gamma_\theta$ and that for each EPPL model m, ρ there exists a unique $i \in K$ such that $m, \rho \Vdash_{\mathsf{EPPL}} \delta_i$. Given a Kripke structure $\mathcal{K} = (\mathsf{S}, \mathsf{R}, \mathsf{L})$ that satisfies $(\mathsf{G}(\widetilde{\gamma}_\theta) \cap \widetilde{\theta})$ and a path π starting at $s \in \mathsf{S}$ we denote by (m_s, ρ_s) an EPPL model that satisfies δ_i whenever $\mathcal{K}, \pi \Vdash_{\mathsf{LTL}} \widetilde{\delta}_i$. Moreover, choose $(m_s, \rho_s) \neq (m_{s'}, \rho_{s'})$ whenever $s \neq s'$ (this can be done just by changing the assignments of variables not occurring in θ). We denote by $\mathcal{T}_\mathcal{K}$ the EPPL-Kripke structure $(S_\mathcal{K}, R_\mathcal{K}, id : S_\mathcal{K} \to S_\mathcal{K})$ where $S_\mathcal{K} = \{(m_s, \rho_s) : s \in S\}$ and $((m_s, \rho_s), (m_{s'}, \rho_{s'})) \in R_\mathcal{K}$ iff $(s, s') \in R$. Finally, given a computation path $\pi = s_1, \ldots$ in $\mathcal{K}$ we denote by π_K the computation path $(m_{s_1}, \rho_{s_1}), \ldots$ in $\mathcal{T}_\mathcal{K}$.

The following theorem is the kernel of the EPLTL SAT algorithm.

Theorem 3.8 *Let θ be a EPLTL formula. Then, $(\mathsf{G}(\widetilde{\gamma}_\theta) \cap \widetilde{\theta})$ is LTL-satisfiable iff θ is EPLTL-satisfiable. Moreover, $\mathcal{K}, \pi \Vdash_{\mathsf{LTL}} (\mathsf{G}(\widetilde{\gamma}_\theta) \cap \widetilde{\theta})$ iff $\mathcal{T}_\mathcal{K}, \pi_\mathcal{K} \Vdash_{\mathsf{EPLTL}} \theta$.*

We are now able to show the SAT algorithm for EPLTL.

The SAT algorithm for EPLTL that we present is double-EXPSPACE, due to applying the EXPSPACE SAT algorithm of LTL to a formula that has increased exponentially. It is possible to improve this bound by adapting the LTL SAT algorithm in order to cope with EPPL formulas. In a full version of the paper it is worthwhile presenting this improvement, but for the sake of space, we preferred herein to present a simpler algorithm.

Algorithm 3: SAT Algorithm for EPLTL– Sat$\mathsf{EPLTL}(\theta)$

Input: EPLTL formula θ
Output: $\mathcal{T}$ (an EPPL-Kripke structure satisfying θ) or *no model*

compute δ_i *as in Equation* (1) *for all* $i \in 2^k$*;*
let $K = \{i \in 2^k : \mathit{SatEPPL}(\delta_i) \neq$ "*no model*"$\}$ *and*
  $\mathcal{M} = \{(m_i, \rho_i) : i \in K$ *and* $\mathit{SatEPPL}(\delta_i) = (m_i, \rho_i)\}$*;*
let $\gamma_\theta = \bigsqcup_{i \in K} \delta_i$*;*
let $\mathcal{K} = \mathit{SatLTL}(\mathsf{G}(\widetilde{\gamma}_\theta) \cap \widetilde{\theta})$*;*
if $\mathcal{K} =$*no model* **then**
  return *(no model)*;
end
return *(*$\mathcal{T}_\mathcal{K}$ *constructed from the models stored in* $\mathcal{M}$*)*;

3.5 Model-checking problem

Similarly to the SAT algorithm, we provide a model-checking algorithm for EPLTL that uses directly the PSPACE model-checking algorithm for LTL. This makes the problem more or less trivial thanks to Lemma 3.3. Given the EPLTL formula θ and a EPPL-Kripke structure $\mathcal{T}$, we start by transforming $\mathcal{T}$ into a classical Kripke structure $\widetilde{\mathcal{T}}$ by checking whether the EPPL models in $\mathcal{T}$ satisfy or not the probabilistic atoms in θ. Then, it remains to model check in LTL the Kripke structure $\widetilde{\mathcal{T}}$ against $\widetilde{\theta}$. Clearly, the model-checking procedure is in PSPACE, since the translation of $\mathcal{T}$ into $\widetilde{\mathcal{T}}$ can be done in polynomial space.

4 Future Work

A research line we will pursue is on using SAT solvers for predicate abstraction [8]. We will explore how EPPL can be applied on probabilistic predicate abstraction. Following the work on non-probabilistic verification of C-like programs [7] we also intend to analyze the probabilistic version of the bit-vector logic [3]. We intend to implement and apply the model-checking algorithm to case studies and check the power of the formalism against established temporal probabilistic logics. The temporalization can also be generalized to more rich logics such as the μ-calculus.

References

[1] C. Baier, E. Clarke, V. Hartonas-Garmhausen, M. Kwiatkowska, and M. Ryan. Symbolic model checking for probabilistic processes. In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, *Proc. 24th International Colloquium on Automata, Languages and Programming (ICALP'97)*, volume 1256 of *LNCS*, pages 430–440. Springer, 1997.

[2] S. Basu, R. Pollack, and R. Marie-Françoise. *Algorithms in Real Algebraic Geometry*. Springer, 2003.

[3] Raik Brinkmann and Rolf Drechsler. Rtl-datapath verification using integer linear programming. In *VLSI Design*, pages 741–746, 2002.

[4] John Canny. Some algebraic and geometric computations in pspace. In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 460–469, New York, NY, USA, 1988. ACM.

[5] R. Chadha, L. Cruz-Filipe, P. Mateus, and A. Sernadas. Reasoning about probabilistic sequential programs. *Theoretical Computer Science*, 379(1-2):142–165, 2007.

[6] V. Chvátal. *Linear programming*. Freeman, 1983.

[7] Edmund M. Clarke, Daniel Kroening, Natasha Sharygina, and Karen Yorav. Satabs: Sat-based predicate abstraction for ansi-c. In *TACAS*, pages 570–574, 2005.

[8] Edmund M. Clarke, Muralidhar Talupur, Helmut Veith, and Dong Wang. Sat based predicate abstraction for hardware verification. In *SAT*, pages 78–92, 2003.

[9] R. Fagin, J. Y. Halpern, and N. Megiddo. A logic for reasoning about probabilities. *Information and Computation*, 87(1/2):78–128, 1990.

[10] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. The temporal analysis of fairness. In *Proceedings 7th Symp. on Principles of Programming Languages, POPL'80*, pages 163–173. ACM, 1980.

[11] C. Jones. *Probabilistic Non-Determinism*. PhD thesis, U. Edinburgh, 1990.

[12] D. Kozen. A probabilistic PDL. *Journal of Computer System Science*, 30:162–178, 1985.

[13] S.A. Kripke. Semantical analysis of modal logic. I. Normal modal propositional calculi. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 9:67–96, 1963.

[14] M. Kwiatkowska, G. Norman, and D. Parker. Prism: Probabilistic symbolic model checker. In *TOOLS '02: Proceedings of the 12th International Conference on Computer Performance Evaluation, Modelling Techniques and Tools*, pages 200–204, London, UK, 2002. Springer-Verlag.

[15] M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic model checking in practice: case studies with prism. *SIGMETRICS Perform. Eval. Rev.*, 32(4):16–21, 2005.

[16] P. Mateus and A. Sernadas. Weakly complete axiomatization of exogenous quantum propositional logic. *Information and Computation*, 204(5):771–794, 2006. ArXiv math.LO/0503453.

[17] P. Mateus, A. Sernadas, and C. Sernadas. Exogenous semantics approach to enriching logics. In G. Sica, editor, *Essays on the Foundations of Mathematics and Logic*, volume 1, pages 165–194. Polimetrica, 2005.

[18] C. Morgan, A. McIver, and K. Seidel. Probabilistic predicate transformers. *ACM Transactions on Programming Languages and Systems*, 18(3):325–353, 1996.

Appendix

Proof of Lemma 2.3

To prove this lemma we need two auxiliary lemmas.

Lemma 4.1 *The relation $\sim_\delta$ is a finite index equivalence relation on Ω and if $\omega_1 \sim_\delta \omega_2$ then $X_\beta(\omega_1) = X_\beta(\omega_2)$ for all $\beta \in bsf(\delta)$.*

Proof: Clearly $\sim_\delta$ is an equivalence relation. The set $prop(\delta)$ is finite, so, it allows only a finite number of different $\sim_\delta$ classes. The second part of the lemma is straightforward by structural induction in β. Let $\omega_1, \omega_2 \in \Omega$ such that $\omega_1 \sim_\delta \omega_2$.

Base: true by definition.

Step:

- In the case $(\neg\beta)$ we have
  $X_{(\neg\beta)}(\omega_1) = 1 - X_\beta(\omega_1) = 1 - X_\beta(\omega_2) = X_{(\neg\beta)}(\omega_2)$;
- In the case $(\beta_1 \Rightarrow \beta_2)$ we have
  $X_{(\beta_1\Rightarrow\beta_2)}(\omega_1) = \max(1-X_{\beta_1}(\omega_1), X_{\beta_2}(\omega_1)) = \max(1-X_{\beta_1}(\omega_2), X_{\beta_2}(\omega_2)) = X_{(\beta_1\Rightarrow\beta_2)}(\omega_2)$.

△

Lemma 4.2 *Let $\omega \in \Omega$,*

$$[\omega]_\delta = \left(\bigcap_{p\in prop_\omega(\delta)} \{\omega' : X_p(\omega') = 1\} \right) \bigcap \left(\bigcap_{p\in prop(\delta)\setminus prop_\omega(\delta)} \{w' : X_p(\omega') = 0\} \right).$$

Moreover, $[\omega]_\delta \in \mathcal{F}$.

Proof: For the first claim, observe that if $\omega_1 \sim_\delta \omega_2$ then $prop_{\omega_1}(\delta) = prop_{\omega_2}(\delta)$. Using the facts that $(X_p)_{p\in\Lambda}$ are random variables and $\mathcal{F}$ is closed to finite intersections we prove the last claim. △

Finally we are able to prove Lemma 2.3.

Proof:[Lemma 2.3] By Lemma 4.1, Ω' is a finite set. If $B \in \mathcal{F}'$, then $\cup B = \cup\{[s]_\delta : [s]_\delta \in B\}$ is in $\mathcal{F}$ by Lemma 4.2. By definition we get that $\mu'([s]_\delta) = \mu([s]_\delta)$ and $\mu'(\Omega') = \mu(\cup\Omega') = \mu(\Omega) = 1$. So, μ' is a finite probabilistic measure over Ω'. △

Proof of Theorem 2.4

Let $m = (\Omega, \mathcal{F}, \mu, \mathbf{X})$ be an EPPL model of δ. We start by computing the quotient model $m' = m/\sim_\delta$ that is a finite discrete EPPL model of size $2^{|prop(\delta)|} \in O(2^{|\delta|})$ and then we will reduce the model to use $2|\delta| + 1$ real numbers.

Observe that in the quotient model m' we need a real number for each valuation over $prop(\delta)$, therefore we need at most $2^{|\delta|}$ real numbers. We start by proving that m' satisfies δ. Note that m and m' agree in the denotation of probabilistic terms. The only non trivial case are the terms $(\int \beta)$. For $\beta \in bsf(\delta)$ and using Lemma 4.2 we get

$$[\![\int \beta]\!]_{m',\gamma} = \mu'(X'_\beta = 1) = \mu(\cup\{X'_\beta = 1\}) = \mu(X_\beta = 1) = [\![(\int \beta)]\!]_{m,\gamma},$$

for any attribution γ of the real variables. By structural induction on terms of δ we get that m and m' agree on inequations, and on independence formulas.

For any subformula $\Box\beta$ of δ we have that $m, \gamma \Vdash \Box\beta$ iff $\Omega = X_\beta^{-1}(1)$ iff $\Omega' = (X'_\beta)^{-1}(1)$ iff $m', \gamma \Vdash \Box\beta$. Now, since m and m' agree on modal formulas $\Box\beta$, inequations $(t_1 \leq t_2)$ and independence formulas $(\beta_1 \perp\!\!\!\perp \beta_2)$ we get by structural induction that m' is a model for δ.

Finally, we will simplify m' to obtain a model $m'' = (\Omega'', 2^{\Omega''}, \mu'', \mathbf{X}'')$ of δ such that $|\Omega''| \leq 2|\delta| + 1$.

Let $bsf(\delta) = \{\beta_1, \ldots, \beta_k\}$ and $\Omega'_{\beta_i} = X'^{-1}_{\beta_i}(1) \subseteq \Omega'$. Observe that $k \leq |\delta|$. Then, we can build a system of $k+1$ equations

$$\begin{cases} \sum_{\omega \in \Omega'_{\beta_1}} x_\omega = \mu'(X_{\beta_1} = 1) \\ \ldots \\ \sum_{\omega \in \Omega'_{\beta_k}} x_\omega = \mu'(X_{\beta_k} = 1) \\ \sum_{\omega \in \Omega'} x_\omega = 1 \end{cases}$$

for which we know that there is a non-negative solution $x_\omega = \mu'(\{\omega\})$ for all $\omega \in \Omega'$. From linear programming it is well known that if a system of $k+1$ linear equations has a non-negative solution, then there is a solution ρ for the system with at most $k+1$ variables taking *positive values* (see, for instance, Theorem 9.3 in [6]). Then, we can construct a model m''' such that $\Omega''' = \{\omega \in \Omega' : \rho(x_\omega) > 0\}$ and $\mu'''(\{\omega\}) = \rho(x_\omega)$. Observe that $m''' \Vdash (t_1 \leq t_2)$ iff $m' \Vdash (t_1 \leq t_2)$ for each inequation $(t_1 \leq t_2)$ occurring in δ. The same happens to independence formulas, since by construction m' and m'' agree in the denotations of terms. However, it might be the case that $m''' \Vdash \Box\beta$ and $m' \nVdash \Box\beta$ for some subformula $\Box\beta$ of δ, since $\Omega''' \subseteq \Omega'$. Then, for each subformula $\Box\beta$ of δ such that $m' \nVdash \Box\beta$ and $m''' \Vdash \Box\beta$ there exists $\omega_\beta \in \Omega' \setminus \Omega'''$ such that $v_{\omega_\beta} \nVdash \beta$. We can now construct the model m'' where

$$\Omega'' = \Omega''' \cup \{\omega_\beta \in \Omega' \setminus \Omega''' : m' \nVdash \Box\beta \text{ and } m''' \Vdash \Box\beta\},$$

$$\mu''(\omega) = \begin{cases} \mu'''(\omega) & \text{if } \omega \in \Omega''' \\ 0 & \text{otherwise} \end{cases}$$

and $X''_p(\omega) = X'_p(\omega)$ for all $\omega \in \Omega''$. Clearly, $|\Omega''| \leq 2|\delta| + 1$ and $m'' \Vdash \delta$. Finally, from the first order theory of real ordered fields, if there is a model using real numbers for a real closed formula, then there is a model using only algebraic real numbers [2] (since the logic can not specify transcendental real numbers with a single formula – we need an infinite number of formulas to be able to specify a transcendental real number), so the solution of the system can be made just with algebraic real numbers. $\triangle$

Proof of Theorem 2.5

We now explain the algorithm and show its soundness. Given an EPPL formula δ, we start by computing (line 1) its global atoms $bf_{\Box}(\delta), ip(\delta), iq(\delta)$ and $at(\delta)$. Observe that computing these sets can be done in PSPACE. Next, we see δ as the propositional formula $\delta^{\alpha}_{p_\alpha}$ (which is obtained by replacing each global atom α with a propositional symbol p_α) and cycle over all exhaustive conjunctions ε such that $v_\varepsilon \Vdash \delta^{\alpha}_{p_\alpha}$ (line 2). Observe that if δ has a model m, γ then this model will either satisfy or not all global atoms $at(\delta)$. Therefore, there is an exhaustive conjunction ε such that m, γ is a model of ε and, moreover, in this case, $v_\varepsilon \Vdash \delta^{\alpha}_{p_\alpha}$. On the other hand, if δ has no model, then all ε such $v_\varepsilon \Vdash \delta^{\alpha}_{p_\alpha}$ have no model. Hence, to find a model for δ it is enough to find a model for an ε such that $v_\varepsilon \Vdash \delta^{\alpha}_{p_\alpha}$. Observe that at each step of the cycle of line 2 we only need to store one such ε, which requires only polynomial space. In the body of the cycle we check whether, given such ε, there is an EPPL model that satisfies all literals occurring in ε.

Using Remark 2.2 we rewrite ε yielding an equivalente EPPL formula without independence formulas. To check if there is an EPPL model for ε we start by computing $lbf_{\Box}(\varepsilon), lbf_{\Diamond\neg}(\varepsilon)$ and $liq(\varepsilon)$ (line 3), which can also be stored in polynomial space. Given Remark 2.1, it is enough to check for models where the outcome space Ω is given as a set of valuations of the basic propositional symbols. Moreover, thanks to the small model theorem (Theorem 2.4), it is enough to search for sets of valuations V such that $|V| \leq 2|\delta| + 1$. Observe that V has to satisfy the modal literals $\Box\beta$ and $\sim\Box\beta$ occurring in ε, that is: (i) for all $\beta \in lbf_{\Box}(\varepsilon)$ and $v \in V$ we have that $v \Vdash \beta$; (ii) for all $\beta \in lbf_{\Diamond\sim}(\varepsilon)$ there exists $v \in V$ such that $v \nVdash \beta$. We can rewrite (i) as $V \Vdash \wedge lbf_{\Box}(\varepsilon)$ and (ii) as $V \nVdash \beta$ for all $\beta \in lbf_{\Diamond\neg}(\varepsilon)$. Hence, it is enough to construct a model with a set of valuations that fulfills the guard of the cycle of line 4. In the body of this cycle we will check if there is a model of ε taking such V as the set of outcomes, that is, if there is a solution for the inequations in $liq(\epsilon)$. Since we only have to store a set of valuations V with $|V| \leq 2|\delta| + 1$ at each step of the cycle, once again we need only polynomial space.

Next, we search for a model of the inequations in $liq(\varepsilon)$ having a set of outcomes V (line 5). To this end we consider a fresh real logical variable x_v for each $v \in V$ representing its probability. The idea behind this step is to build an analytical formula κ that specifies the two probability constraint expressed in line 5 and the inequations in $liq(\varepsilon)$. This formula, κ, is constructed in line 7, by replacing the terms $(\int \beta)$ in $liq(\varepsilon)$ by $\sum_{v \in V : v \Vdash \beta} x_v$, that is, we rewrite the measure terms by the measure of its outcomes. In line 9 we call the *SatReal* algorithm for a solution (model) to κ. Since $|\kappa|$ is polynomial on $|\delta|$ and the set of variables in κ is polynomially bounded by $|\delta|$, the *SatReal* will compute the solution in

PSPACE. If such solution ρ exists, we have succeeded in finding a model for δ. Hence, we return (V, μ_ρ) and γ_ρ, where $\mu_\rho(v)$ is $\rho(x_v)$ (line 11) and γ_ρ is the restriction of ρ to $var(\delta)$ (line 12). As stated in Remark 2.1 this is enough to construct an EPPL model. If there is no solution ρ then we could not find a solution for the set V of valuations, and have to try with another V. Finally, if for all ε and V we are not able to find a solution, then there is no model for δ. $\triangle$

Proof of Theorem 2.6

To show the completeness of the system, we use a contrapositive argument: if $\nvdash \delta$ then $\nVdash \delta$. By definition, a formula δ is *consistent* if $\nvdash \sim\delta$. So, if we prove that every consistent formula δ has a model we get the completeness result. To check this fact, observe that if $\nvdash \delta$ then $\nvdash \sim\sim\delta$, that is, $\sim\delta$ is consistent. If $\sim\delta$ is consistent it has a model and therefore, $\nVdash \delta$.

So, we will prove that every consistent formula δ has a model. Assume by contradiction that δ is consistent and the SAT algorithm returns *no model*. Let $A = \{\varepsilon$ exhaustive conjunction of literals: $v_\varepsilon \Vdash \delta^\alpha_{p_\alpha}\}$. By the completeness of propositional logic it follows that $\vdash (\vee_{\varepsilon\in A}\varepsilon^\alpha_{p_\alpha}) \Leftrightarrow \delta^\alpha_{p_\alpha}$, and by **GTaut** we have that $\vdash \cup A \equiv \delta$. If δ is consistent then there is ε consistent, and if δ has no model, then the consistent ε has no model as well. If the SAT algorithm returns *no model* for ε it has to be for one of the following two reasons: (i) it can not find a V at line 4; (ii) for all viable V the $SatReal$ algorithm returns *no model* at line 9. We will now show that for both cases we can contradict the consistency of ε.

In case (i) – no V can be found at line 4 – it cannot be because $|V| > 2|\delta| + 1$, thanks to the small model theorem. This means that if we remove the bound $0 < |V| \leq 2|\delta| + 1$ in line 4, and consider all possible sets of valuations the algorithm would also fail. In particular, take $V = 2^{prop(\delta)}$, it must happen (a) $V \nVdash \wedge lbf_\Box(\varepsilon)$ *or* (b) $V \Vdash \beta$ for some $\beta \in lbf_{\Diamond\neg}(\varepsilon)$. For case (a) we have that $\nVdash \wedge lbf_\Box(\varepsilon)$ and so, $\nVdash \beta$ for some $\beta \in lbf_\Box(\varepsilon)$, or equivalently $\Vdash \beta \Rightarrow \bot$. But by completeness of the propositional calculus we have that $\vdash \beta \Rightarrow \bot$, by **CTaut** we have that $\vdash \Box(\beta \Rightarrow \bot)$ and by **Lift**$\Rightarrow$ and **Eqv**$\bot$ we have that $\vdash \sim(\Box\beta)$ from which follows $\vdash \sim\varepsilon$ which contradicts the consistency of ε. In case (b) there is $\beta \in lbf_{\Diamond\neg}(\varepsilon)$ such that β is a tautology. Then, by **CTaut**, $\vdash \Box\beta$. From the last derivation we get $\vdash \sim\varepsilon$, which contradicts the consistency of ε.

In case (ii), the algorithm fails at line 9 for all viable V computed in line 4. Thanks to the small model theorem it means that the algorithm would also fail at line 9 for all V such that

$$V \Vdash \wedge lbf_\Box(\varepsilon) \text{ and } V \nVdash \beta \text{ for all } \beta \in lbf_{\Diamond\neg}(\varepsilon), \tag{2}$$

independently of the bound on the size of V. It is easy to see that the sets of valuations satisfying (2) are closed under unions, and therefore there is the largest V fulfilling (2), say $V_{\max}$, and for this set the algorithm would fail at line 9. Let $V^c = 2^{prop(\delta)} \setminus V_{\max}$, since ε is consistent it is easy to see that $\varepsilon' = \varepsilon \cap (\cap_{v\in V^c}\Box\neg\beta_v)$ is consistent, where β_v is a propositional formula that is satisfied only by valuation v. Indeed, $\vdash \wedge lbf_\Box(\varepsilon) \Rightarrow \neg\beta_v$ for

all $v \in V^c$, from which we derive that

$$\vdash (\cap_{\beta \in lbf_{\Box}(\varepsilon)} \Box\beta) \supset (\cap_{v \in V^c} \Box\neg\beta_v)$$

and so $\vdash \varepsilon' \equiv \varepsilon$. Thus, if ε is consistent then ε' is also consistent, and if there is no model for ε then there is no model for ε' as well, and the algorithm will fail precisely in line 9.

By **RCF** we have $\vdash \sim\kappa^{x_v}_{(\int\beta_v)}$, where $\kappa^{x_v}_{(\int\beta_v)}$ is the formula κ where we replace each variable x_v by the term $(\int\beta_v)$ with β_v a propositional formula that is satisfied only by v. By **Prob**, **FAdd** and **Mon** we have $\vdash (\Box\neg\beta_v) \supset ((\int\beta_v) = 0)$, thus we can derive

$$\vdash \varepsilon' \supset (\cap_{v \in V^c}((\int\beta_v) = 0)). \tag{3}$$

From $\vdash \sim\kappa^{x_v}_{(\int\beta_v)}$ and **FAdd** and **RCF** we obtain that

$$\vdash \cap_{v \in V^c}((\int\beta_v) = 0) \supset \sim \cap_{\alpha \in liq(\delta)} \alpha. \tag{4}$$

Finally, by **CTaut** we have

$$\vdash \sim \cap_{\alpha \in liq(\delta)} \alpha \supset \sim\varepsilon'. \tag{5}$$

So, from (3), (4) and (5) we obtain with tautological reasoning $\vdash \varepsilon' \supset \sim\varepsilon'$ from which we conclude $\vdash \sim\varepsilon'$. This contradicts the consistency of ε' and thus, the consistency of δ. For this reason there must be a model for ε' and consequently, a model for δ. △

Proof of Theorem 2.7

The first part of the model-checking algorithm (lines 1–7) consists in writing a Boolean $|bsf(\delta)| \times |\Omega|$-matrix B where the entry $B(i,j)$ is $X_{\beta_i}(\omega_j)$, for all $1 \leq i \leq |bsf(\delta)|$ and $1 \leq j \leq |\Omega|$. In the second part of the algorithm (lines 8–16), we evaluate all the subterms to a real $|pst(\delta)|$-array T, where $T(i) = [\![t_i]\!]_{m,\gamma}$, for all $1 \leq i \leq |pst(\delta)|$. In this part, denotation of the term $(\int\beta_i)$ is calculated in line 12 by the matrix product of the two arrays, $[\![(\int\beta_i)]\!]_{m,\gamma} = B(i).\mu$, for all $1 \leq i \leq |bsf(\delta)|$. Finally, in the third part of the algorithm (lines 17–23), we evaluate all global subformulas to a Boolean $|gsf(\delta)|$-array G, where $G(i) = 1$ iff $m, \gamma \Vdash \delta_i$, for all $1 \leq i \leq |gsf(\delta)|$, and return as output $G(|gsf|)$. This result is formalized in the following theorem. △

Proof of Lemma 3.3

The proof follows by straightforward induction on the structure of θ.

- Base: If θ is $\mathbf{f}$ or $\theta \in \mathsf{pAtom}$ then $\mathcal{T}, \pi \Vdash_{\mathsf{EPLTL}} \theta$ iff $m_1\rho_1 \Vdash_{\mathsf{EPPL}} \theta$ iff $\widetilde{\mathcal{T}}, \pi \Vdash_{\mathsf{LTL}} \widetilde{\theta}$ by definition of $\widetilde{\mathcal{T}}$.
- Step: For the sake of simplicity, we just consider the case when θ is $\mathsf{X}\theta_1$. The other cases can be similarly handled. Now, $\mathcal{T}, \pi \Vdash_{\mathsf{EPLTL}} \mathsf{X}\theta_1$ iff $\mathcal{T}, \pi^2 \Vdash_{\mathsf{EPLTL}} \theta_1$ iff, by induction, $\widetilde{\mathcal{T}}, \pi^2 \Vdash_{\mathsf{LTL}} \widetilde{\theta}_1$ iff $\widetilde{\mathcal{T}}, \pi \Vdash_{\mathsf{LTL}} \mathsf{X}\widetilde{\theta}_1$ iff, by definition, $\widetilde{\mathcal{T}}, \pi \Vdash_{\mathsf{LTL}} \widetilde{\theta}$. △

Proof of Lemma 3.5

Thanks to Lemma 3.4 it suffices to show that if $\vdash_{\mathsf{EPLTL}} \gamma$ then $\vdash_{\mathsf{EPPL}} \gamma$. Suppose $\vdash_{\mathsf{EPLTL}} \gamma$. Then $\Vdash_{\mathsf{EPLTL}} \gamma$ by soundness of EPLTL. Let m be an arbitrary EPPL model and ρ an arbitrary attribution. Consider the EPPL-Kripke structure $\mathcal{T} = (\{m\rho\}, \{(m\rho, m\rho), id_{\{m\rho\}}\})$ and π its unique path. We have that $\mathcal{T}, \pi \Vdash_{\mathsf{EPLTL}} \gamma$. By definition, we get $m\rho \Vdash_{\mathsf{EPPL}} \gamma$. Since ψ and ρ are arbitrary, we get $\Vdash_{\mathsf{EPPL}} \gamma$. By completeness of EPPL, we get $\vdash_{\mathsf{EPPL}} \gamma$. △

Proof of Lemma 3.6

Let $at = \{\gamma_1, \ldots, \gamma_k\}$ be the set of atomic EPPL formulas that are atoms of θ. Now for each k-vector $i \in \{0,1\}^k$, consider the EPPL formula

$$\delta_i = \prod_{j=1}^{k} \varphi_j \qquad \text{where} \qquad \varphi_j = \begin{cases} \gamma_j & \text{if } j\text{-th bit of } i \text{ is } 1 \\ (\sim\gamma_j) & \text{otherwise} \end{cases}$$

Let $K \subseteq \{0,1\}^k$ be such that δ_i is a EPPL consistent formula and let $\gamma_\theta = \bigsqcup_{i \in K} \delta_i$. Clearly, $\vdash_{\mathsf{EPPL}} \gamma_\theta$ and therefore by Lemma 3.5, $\vdash_{\mathsf{EPLTL}} \gamma_\theta$. Also please note for any EPPL model m and assignment ρ, $m\rho \Vdash \delta_i$ for exactly one $i \in K$.

We shall prove $\Vdash_{\mathsf{LTL}} (\mathsf{G}\widetilde{\gamma}_\theta \supset \widetilde{\theta})$ by contradiction. Suppose that $\mathcal{K} = (\mathsf{S}, \mathsf{R}, \mathsf{L})$ is a Kripke structure such that $\mathcal{K}, \pi \not\Vdash_{\mathsf{LTL}} (\mathsf{G}\widetilde{\gamma}_\theta \supset \widetilde{\theta})$ for some $s \in \mathsf{S}$. Then $\mathcal{K}, \pi \Vdash_{\mathsf{LTL}} \mathsf{G}\widetilde{\gamma}_\theta$ and $\mathcal{K}, \pi \not\Vdash_{\mathsf{LTL}} \widetilde{\theta}$. Let $\mathsf{S}' = \{s' \in \mathsf{S} : s' \text{ occurs in } \pi\}$.

Since $\mathcal{K}, \pi \Vdash_{\mathsf{LTL}} \mathsf{G}\widetilde{\gamma}_\theta$, we get that $\mathcal{K}, \pi \Vdash_{\mathsf{LTL}} \widetilde{\gamma}_\theta$. Hence, there is some $i_{s_1} \in K$ such that $\mathcal{K}, \pi \Vdash_{\mathsf{LTL}} \widetilde{\delta}_{i_{s_1}}$. Since $\delta_{i_{s_1}}$ is consistent EPPL formula, there is an EPPL model m_{s_1} and an assignment ρ_{s_1} such that $m_{s_1}\rho_{s_1} \Vdash_{\mathsf{EPPL}} \delta_{i_{s_1}}$. For each s_j fix one such m_{s_j} and ρ_{s_j} ensuring that $\rho_{s_j} \neq \rho_{s_k}$ whenever $j \neq k$ (this can be ensured by modifying the assignments on real variables not occurring in θ). Consider the set $S_\theta = \{(m_{s'}, \rho_{s'}) : s' \in \mathsf{S}'\}$ and the EPLTL model $\mathcal{T} = (S_\theta, R_\theta, id_{S_\theta})$, where $(m_{s'}\rho_{s'}, m_{s''}\rho_{s''}) \in R_\theta$ iff $(s', s'') \in R$. Denote by π' the path $m_{s_1}\rho_{s_1}, m_{s_2}\rho_{s_2} \ldots$ Using the fact that $\mathcal{K}, \pi \not\Vdash \widetilde{\theta}$, it follows from Lemma 3.3 $\mathcal{T}, \pi' \not\Vdash \theta$ which contradicts $\vdash_{\mathsf{EPLTL}} \gamma_\theta$. △

Proof of Theorem 3.8

Proof: ⇐) Follows directly from Lemma 3.3 and from the fact that $\Vdash_{\mathsf{EPPL}} \gamma_\theta$. Concerning the second assertion, it follows by noticing that $\widetilde{\mathcal{T}_\mathcal{K}}$ is equal to $\mathcal{K}$ up to relabeling of states.

⇒) For this direction, it is sufficient to show the second assertion. Since $\widetilde{\mathcal{T}_\mathcal{K}}$ is equal to $\mathcal{K}$ up to relabeling of states, by Lemma 3.3 we have that $\mathcal{T}_\mathcal{K}, \pi_\mathcal{K} \Vdash_{\mathsf{EPLTL}} \mathsf{G}(\gamma_\theta) \cap \theta$ and therefore $\mathcal{T}_\mathcal{K}, \pi_\mathcal{K} \Vdash_{\mathsf{EPLTL}} \theta$. △