Exogenous Probabilistic Computation Tree Logic

Pedro Baltazar[1,6] Paulo Mateus[2,6]

Security and Quantum Information Group, Institute for Telecommunications and Instituto Superior Técnico, Universidade Técnica de Lisboa, Lisbon, Portugal

Rajagopal Nagarajan[3,5] Nikolaos Papanikolaou[4,5]

Department of Computer Science, University of Warwick, Coventry, England

Abstract

We define a logic EpCTL for reasoning about the evolution of probabilistic systems. System states correspond to probability distributions over classical states and the system evolution is modelled by probabilistic Kripke structures that capture both stochastic and non–deterministic transitions. The proposed logic is a temporal enrichment of Exogenous Probabilistic Propositional Logic (EPPL). The model-checking problem for EpCTL is analysed and the logic is compared with PCTL; the semantics of the former is defined in terms of probability distributions over sets of propositional symbols, whereas the latter is designed for reasoning about distributions over paths of possible behaviour. The intended application of the logic is as a specification formalism for properties of communication protocols, and security protocols in particular; to demonstrate this, we specify relevant security properties for a classical contract signing protocol and for the so–called quantum one–time pad.

1 Introduction

There are numerous applications in science where reasoning about probabilistic behaviour is necessary. In computing, applications include probabilistic algorithms, computer modelling and verification of probabilistic systems, including communication protocols with and without security guarantees. The properties of probabilistic programs in particular have been studied before using many different approaches, and it is widely accepted that the development of formal logics for reasoning about such programs is highly beneficial, allowing designers and users of systems to formulate properties which the programs may or may not satisfy.

In this paper we describe a temporal probabilistic logic, EpCTL. Our approach is characterised by the use of an exogenous semantics, such that the models of state formulas are

[1] Email: pbtz@math.ist.utl.pt

[2] Email: pmat@math.ist.utl.pt

[3] Email: biju@dcs.warwick.ac.uk

[4] Email: nikos@dcs.warwick.ac.uk

[5] Partially supported by the EU Sixth Framework Programme (Project SecoQC: *Development of a Global Network for Secure Communication based on Quantum Cryptography*).

[6] Partially supported by FCT and EU FEDER, namely via CLC POCTI (Research Unit 1-601), QuantLog project POCI/MAT/55796/2004 and the recent QSec initiative of SQIG-IT. Pedro Baltazar also supported by FCT and EU FEDER PhD fellowship SFRH/BD/22698/2005.

essentially probability distributions of models of a propositional logic. We build atop earlier work on the probabilistic state logic EPPL [24] by introducing a temporal extension; the result is a branching time logic for reasoning about probabilistic programs. Our intention is to provide a powerful framework for specifying properties of communication protocols, especially security protocols. The proposed logic has enough expressive power to allow specification of relevant security properties, and enables high–level reasoning due to the use of an exogenous semantics.

The exogenous semantics approach [25] involves taking the semantic structures of a base logic (e.g. propositional logic) and combining them together, possibly adding new structure, to provide the semantics for a higher-level logic. This approach has been used to build the probabilistic state logic EPPL and also a logic EQPL for reasoning about states of quantum information systems. The exogenous semantics approach can be considered a variant of the possible-worlds approach of Kripke for modal logic [18], and it is related to the society semantics introduced in [8] for many-valued logic and to the possible translations semantics proposed in [7] for paraconsistent logic.

The logic described here is related to the logic PCTL proposed by Hansson and Jonsson [16]. PRISM [19,20] is a symbolic model-checker for PCTL. There is a fundamental difference between the semantics of the proposed logic EpCTL and PCTL; whereas PCTL enables reasoning about distributions over *paths* in a probabilistic transition system, EpCTL is designed for reasoning about how a probability distribution over a finite set of propositional symbols changes over time. The latter approach is particularly advantageous for reasoning about certain types of systems, such as distributed randomised algorithms. This will be discussed in more detail in Section 3.4.

This paper is structured as follows. First we examine the syntax, semantics, model–checking problem and axiomatisation of the state logic EPPL. We then describe each of these aspects in turn for the temporal extension, namely the new logic EpCTL, and provide intuition for the various constructs. Proofs of theorems have been relegated to the Appendix.

2 Logic of Probabilistic States - EPPL

The state logic of temporal logics like CTL [13], LTL [29] and CTL* [11] is classical propositional logic. Probabilistic temporal extensions, like PCTL [16], also use classical propositional logic for state logic. Here we consider quite a different state logic, exogenous probabilistic propositional logic (EPPL) [23,25,9,24], which was highly inspired by the works of Halpern *et al.* [15]. Given the envisaged application to model–checking, we consider only models over a finite set of propositional symbols Φ, which can be understood as Boolean registers (bits) that constitute the internal state of a protocol or an algorithm. In this setting *an EPPL model*, that henceforth we will call a *probabilistic structure*, is a pair (V,μ) where V is a set of classical valuations [7] over Φ and μ is a map $\mu : V \to [0,1]$ where $\sum_{v\in V}\mu(v) = 1$. For a model (V,μ) we call V the set of *possible valuations*, and μ the probability measure. Observe that μ can be extended to all valuations by assuming that impossible valuations are improbable, i.e, $\mu(v) = 0$ for any $v \in \mathcal{V}\setminus V$ where $\mathcal{V}$ is the set of all valuations over Φ. Finally, it will be useful to consider the probability measure [8], where μ is defined over sets of valuations, $(\mathcal{V}, 2^{\mathcal{V}}, \mu)$ and

[7] Recall that a classical valuation over Φ is a map $v : \Phi \to \{0,1\}$.

[8] Recall that a probability measure is a triple $(\Omega,\mathcal{F},\mu)$ where $\mathcal{F}$ is a σ-algebra over Ω and μ is a measure where $\mu(\Omega) = 1$. Given the finitary assumption over Φ, in this paper we will always have Ω to be $\mathcal{V}$ and $\mathcal{F}$ to be the powerset of $\mathcal{V}$.

$\mu(U) = \sum_{v \in U} \mu(v)$ for any $U \subseteq \mathcal{V}$.

Example 2.1 Consider a variant of the Russian roulette game where the a gambler tosses a coin and if the outcome is tails, the gun is fired. Assume also that the gambler has $1/6$ probability of shooting a bullet. We describe the system with three propositional symbols h (heads), b (bullet was shot), d (gambler is dead). The possible valuations described as sets of propositional symbols are: $\emptyset$ (all propositional symbols are false, the outcome of the coin was not heads); $\{h\}$ (the outcome of the coin was heads but no bullet was shot); and $\{h,b,d\}$ (the outcome of the coin was heads, a bullet was shot and the gambler is dead). The probability measure is $\mu(\emptyset) = 1/2$, $\mu(\{h\}) = 5/12$ and $\mu(\{h,b,d\}) = 1/12$.

We continue by describing the syntax of the logic.

2.1 Language

The language consists of formulas at two levels. The formulas at the first level, *classical formulas*, enable reasoning about propositional symbols. The formulas at the second level, *probabilistic state formulas*, enable reasoning about probabilistic structures. There are also *probability terms* used in probabilistic state formulas to denote real numbers. The syntax of the language, expressed using BNF notation, is as follows.

- Classical formulas
  $\gamma := \Phi \,[\!]\, \perp \,[\!]\, (\gamma \Rightarrow \gamma)$
- Probability terms
  $p := 0 \,[\!]\, 1 \,[\!]\, y \,[\!]\, (\textstyle\int \gamma) \,[\!]\, (p + p) \,[\!]\, (p\,p)$
- Probabilistic state formulae
  $\cdot\ \xi := (\Box \gamma) \,[\!]\, (p \leq p) \,[\!]\, \perp\!\!\!\perp \,[\!]\, (\xi \supset \xi)$

The classical state formulas, ranged over by $\gamma, \gamma_1, \ldots$, are built from the propositional symbols Φ and the classical disjunctive connectives $\perp$ (falsum) and $\Rightarrow$ (implication). As usual, other classical connectives $(\neg, \vee, \wedge, \Leftrightarrow)$ are introduced as abbreviations. For instance, $(\neg\gamma)$ stands for $(\gamma \Rightarrow \perp)$.

The probability terms, ranged over by $p, p_1, \ldots$, denote elements of the reals. We also assume a set of variables, $\mathsf{Y} = \{y_k : k \in \mathbb{N}\}$, ranging over the reals. The term $(\int\gamma)$ denotes the measure of the set of valuations that satisfy γ.

The probabilistic state formulas, ranged over by $\xi, \xi_1, \ldots$, are built from the *necessity formulas* $(\Box\gamma)$, the comparison formulas $(p_1 \leq p_2)$ and the connectives $\perp\!\!\!\perp$ and $\supset$. The formula $(\Box\gamma)$ is true when γ is true of every possible valuation in the semantic structure. Other probabilistic connectives $(\ominus, \cup, \cap, \approx)$ are introduced as abbreviations. For instance, $(\ominus\xi)$ stands for $(\xi \supset \perp\!\!\!\perp)$. We shall also use $(\Diamond\gamma)$ as an abbreviation for $(\ominus(\Box(\neg\gamma)))$. Please note that the $\Box$ and $\Diamond$ are not modalities[9]. We also use any algebraic real number as a constant since the language of EPPL has enough expressiveness to specify these constants. For instance, $\sqrt{2} \leq y_1$ can be written as $(y_2 y_2 = (1+1) \cap y_2 \geq 0) \supset y_2 \leq y_1$. We will use subtraction and division freely since they can also be expressed in EPPL, for instance $x/y = -2$ can be written as $((z + (1+1) = 0) \cap (\ominus(y = 0))) \supset x = y \cdot z$. Finally, the conditional probability term $(\int \gamma_1 | \gamma_2)$ is an abbreviation of $(\int (\gamma_1 \wedge \gamma_2))/(\int \gamma_2)$.

[9] We do not have formulas such as $\Box(\Box\gamma)$.

The notion of occurrence of a term p and a probabilistic state formula ξ_1 in the probabilistic state formula ξ can be easily defined. The notion of replacing zero or more occurrences of probability terms and probabilistic formulas can be similarly defined. For the sake of clarity, we shall often drop parentheses in formulas and terms if it does not lead to ambiguity.

Example 2.2 Consider again the variant of the Russian roulette described in Example 2.1. Stating that the coin is fair can be expressed by $(\int h = 1/2)$. We can also say the bullet is shot only if the outcome of the coin is heads by $\Box(b \Rightarrow h)$. Similarly, the gambler is dead only if the outcome of the coin toss is heads and the bullet is shot, which can be expressed by $\Box(d \Rightarrow b \wedge h)$. Finally, the fact that the probability of the bullet being shot is $1/6$ can be captured by $(\int b|h) = 1/6$.

2.2 *Semantics*

Given $V \subseteq \mathcal{V}$, the *extent* of a classical formula γ in V is defined as $|\gamma|_V = \{v \in V : v \Vdash_C \gamma\}$. For interpreting the probabilistic variables, we need the concept of an assignment. An *assignment* ρ is a map such that $\rho(y) \in \mathbb{R}$ for each $y \in \mathsf{Y}$.

Given a probabilistic structure (V, μ) and an assignment ρ, the denotation of probabilistic terms and satisfaction of probabilistic state formulas are defined inductively as follows.

- Denotation of probability terms
  - $[\![0]\!]^{\rho}_{(V,\mu)} = 0$
  - $[\![1]\!]^{\rho}_{(V,\mu)} = 1$
  - $[\![y]\!]^{\rho}_{(V,\mu)} = \rho(y)$
  - $[\![(\int\gamma)]\!]^{\rho}_{(V,\mu)} = \mu(|\gamma|_V)$
  - $[\![p_1 + p_2]\!]^{\rho}_{(V,\mu)} = [\![p_1]\!]^{\rho}_{(V,\mu)} + [\![p_2]\!]^{\rho}_{(V,\mu)}$
  - $[\![p_1 p_2]\!]^{\rho}_{(V,\mu)} = [\![p_1]\!]^{\rho}_{(V,\mu)} \times [\![p_2]\!]^{\rho}_{(V,\mu)}$
- Satisfaction of probabilistic formulas
  - $(V,\mu)\rho \Vdash (\Box\gamma)$ iff $v \Vdash_C \gamma$ for every $v \in V$
  - $(V,\mu)\rho \Vdash (p_1 \leq p_2)$ iff $V \neq \emptyset$ implies $([\![p_1]\!]^{\rho}_{(V,\mu)} \leq [\![p_2]\!]^{\rho}_{(V,\mu)})$
  - $(V,\mu)\rho \nVdash \perp\!\!\!\perp$
  - $(V,\mu)\rho \Vdash (\xi_1 \supset \xi_2)$ iff $(V,\mu)\rho \Vdash \xi_2$ or $(V,\mu)\rho \nVdash \xi_1$

The formula $(\Box\gamma)$ is satisfied only if all $v \in V$ satisfy γ. The formula $(p_1 \leq p_2)$ is satisfied if the term denoted by p_1 is less than p_2. The formula $(\xi_1 \supset \xi_2)$ is satisfied by a semantic model if either ξ_1 is not satisfied by the model or ξ_2 is satisfied by the model. Entailment is defined as usual: Ξ entails ξ (written $\Xi \vDash \xi$) if $(V,\mu)\rho \Vdash \xi$ whenever $(V,\mu)\rho \Vdash \xi_0$ for each $\xi_0 \in \Xi$.

Please note that an assignment ρ is sufficient to interpret a useful sub-language of probabilistic state formulas:

$$\begin{aligned} \kappa &:= (a \leq a) \mathbin{[\!]} \perp\!\!\!\perp \mathbin{[\!]} (\kappa \supset \kappa) \\ a &:= 0 \mathbin{[\!]} 1 \mathbin{[\!]} x \mathbin{[\!]} (a + a) \mathbin{[\!]} (aa). \end{aligned}$$

Henceforth, the terms of this sub-language will be called *analytical terms* and the formulas will be called *analytical formulas*.

2.3 *Model–checking* EPPL

For the model–checking procedure we assume that the probabilistic structure and assignment are represented using a floating point data structure. We assume that a probabilistic structure (V,μ) for Φ propositional symbols is modelled by a V-array of real numbers; the size of V is at most 2^n with with $n = |\Phi|$. We also assume that the basic arithmetical operations take $O(1)$ time. Moreover, we assume that we use only a finite number of variables Y and that assignment is a vector of real number of size $|\mathsf{Y}|$.

We also assume the definition of the *length* of a classical formula γ or a quantum formula ξ as the number of symbols required to write the formula. The length of a formula ξ (classical or probabilistic) is given is represented by $|\xi|$.

Given a probabilistic structure (V,μ), assignment ρ and a probabilistic formula ξ, the first step is to evaluate all the terms occurring in ξ. For the probability terms $\int\gamma$, the evaluation takes $|V|\cdot|\gamma|$ steps as we check the set of valuations that satisfy γ. Once the terms are evaluated, the model–checking algorithm is straightforward.

Theorem 2.3 Assuming that all basic arithmetical operations take unit time, there is an algorithm $O(|\xi|\cdot|V|)$ to decide if a probabilistic structure over Φ and an assignment ρ satisfy ξ.

Clearly, since in the worst case the probability distribution will span all possible valuations, we have $|V| = 2^n$ where $n = |\Phi|$. Observe that in many cases the set of possible valuations is small and it is possible to describe this set in a compact manner, as well as the probabilities associated, we will return to this discussion when we discuss the model–checking procedure of the temporal extension of the logic.

2.4 *Axiomatisation of* EPPL

The axiomatisation of the EPPL logic presented here relies entirely on that in [9] and will be presented in a summarised way, since its details are outside the scope of this paper. We need two new concepts for the axiomatisation, namely the notions of *probabilistic tautology* and of *valid analytical formulas*.

Consider propositional formulas built from a countable set of propositional symbols Q using the classical connectives $\perp$ and $\rightarrow$. A probabilistic formula ξ is said to be a *probabilistic tautology* if there exists a propositional tautology β over Q, and a map σ from Q to the set of probabilistic state formulas, such that ξ coincides with $\beta_p\sigma$ (where $\beta_p\sigma$ is the probabilistic formula obtained from β by replacing all occurrences of $\perp$ by $\perp\!\!\!\perp$, $\rightarrow$ by $\supset$ and $q \in \mathrm{Q}$ by $\sigma(q)$). For instance, the probabilistic formula $((y_1 \leq y_2) \supset (y_1 \leq y_2))$ is tautological (obtained, for example, from the propositional tautology $q \rightarrow q$).

As noted in Section 2.2 an assignment is enough to interpret all analytical formulas. We say that κ is a *valid analytical formula* if for any real closed field $\mathcal{K}$ and assignment ρ, κ is true for ρ. Clearly, a valid analytical formula holds for all semantic structures of EPPL. It is a well-known fact from the theory of quantifier elimination [17,3] that the set of valid analytical formulas so defined is decidable over algebraic ordered fields. Moreover, since the real numbers consitute a representative model of algebraic ordered fields (that is, if there exists a solution for a systems of inequations written with the terms of EPPL in an algebraic order field, there is also a solution for the real numbers), the decidability result extends over the real numbers. We shall not go into details of this result as we want to focus exclusively on reasoning about probabilistic aspects.

The axioms and inference rules of EPPL are listed below.

- Axioms
  - · [**CTaut**] $\vdash (\Box\gamma)$ for each valid formula γ;
  - · [**PTaut**] $\vdash \xi$ for each probabilistic tautology ξ

  - · [**Lift⇒**] $\vdash ((\Box(\gamma_1 \Rightarrow \gamma_2)) \supset (\Box\gamma_1 \supset \Box\gamma_2))$
  - · [**Eqv⊥**] $\vdash ((\Box\bot) \approx \bot\!\!\!\bot)$
  - · [**Ref∧**] $\vdash (((\Box\gamma_1) \cap (\Box\gamma_2)) \supset (\Box(\gamma_1 \wedge \gamma_2)))$

  - · [**RCF**] $\vdash \kappa\{\!|\vec{y}/\vec{p}|\!\}$ where κ is a valid analytical formula, $\vec{y}$ and $\vec{p}$ are sequences of probability variables and probability terms respectively

  - · [**Meas∅**] $\vdash ((\int\bot) = 0)$
  - · [**FAdd**] $\vdash (((\int(\gamma_1 \wedge \gamma_2)) = 0) \supset ((\int(\gamma_1 \vee \gamma_2)) = (\int\gamma_1) + (\int\gamma_2)))$
  - · [**Prob**] $\vdash ((\int\top) = 1)$
  - · [**Mon**] $\vdash ((\Box(\gamma_1 \Rightarrow \gamma_2)) \supset ((\int\gamma_1) \leq (\int\gamma_2)))$
- Inference rules
  - · [**CMP**] $(\Box\gamma_1), (\Box(\gamma_1 \Rightarrow \gamma_2)) \vdash (\Box\gamma_2)$
  - · [**PMP**] $\xi_1, (\xi_1 \supset \xi_2) \vdash \xi_2$

The axiom **CTaut** says that if γ is a valid classical formula then $(\Box\gamma)$ is an axiom. The axiom **PTaut** says that a probabilistic tautology is an axiom. Since the set of valid classical formulas and the set of probabilistic tautologies are both recursive, there is no need to spell out the details of tautological reasoning.

The axioms **Lift⇒**, **Eqv⊥** and **Ref∧** are sufficient to relate (local) classical state reasoning and (global) probabilistic tautological reasoning.

The term $\kappa\{\!|\vec{y}/\vec{p}|\!\}$ in the axiom **RCF** is the term obtained by substituting all occurrences of y_i in κ by p_i. The axiom **RCF** says that if κ is a valid analytical formula, then any formula obtained by replacing variables with probability terms is a tautology. We refrain from spelling out the details as the set of valid analytical formulas is recursive.

The axiom **Meas∅** says that the measure of empty set is 0. The axiom **FAdd** is the finite additivity of the measures. The axiom **Mon** relates the classical connectives with probability measures and is a consequence of monotonicity of measures. The axiom **Prob** says that the measure is a probability measure.

The inference rules **CMP** and **PMP** are the *modus ponens* for classical and probabilistic implication respectively.

As usual we say that a set of formulas Γ *derives* ξ, written $\Gamma \vdash \xi$, if we can build a derivation of ξ from axioms and the inference rules using formulas in Γ as hypothesis.

Theorem 2.4 EPPL is sound and weakly complete. Moreover, the set of theorems is recursive.

3 The Computation Tree Extension - EpCTL

In this section we define the computation tree extension to EPPL, which we call exogenous probabilistic computation tree logic (EpCTL). The idea is to consider several probabilistic structures together with a transition relation between them, in other words, a Kripke structure whose nodes are probability structures. This structure is particularly interesting for two reasons. Firstly, it captures the idea, which arises in the study of probabilistic transition systems, that the state space should be described as a distribution of classical states [30,14,9]. Secondly, it is a step towards reasoning about quantum systems, since in such systems a state is described as a probabilistic ensemble of pure quantum states (*cf.* mixed states, density operators). We will explore both aspects in Section 4 by presenting two detailed examples.

We proceed to present the syntax of EpCTL.

3.1 Syntax

The syntax of EpCTL can be easily obtained from the syntax of EPPL. The idea is that at the level of probabilistic state formulas, we also introduce the usual CTL modalities. For the sake of clarity, we recall the definition of classical formulas and probability terms.

- Classical formulas
  $\gamma := \Phi \,[\!]\, \perp \,[\!]\, (\gamma \Rightarrow \gamma)$
- Probability terms
  $p := 0 \,[\!]\, 1 \,[\!]\, y \,[\!]\, (\int \gamma) \,[\!]\, (p + p) \,[\!]\, (p\, p)$
- Exogenous probabilistic computation tree logic formulas
  · $\delta := (\Box\gamma) \,[\!]\, (p \leq p) \,[\!]\, \perp\!\!\!\perp \,[\!]\, (\delta \supset \delta) \,[\!]\, (\mathsf{EX}\delta) \,[\!]\, (\mathsf{AF}\delta) \,[\!]\, (\mathsf{E}[\delta \mathsf{U} \delta])$

The intuitive semantics of the temporal modalities is similar to that in classical CTL. The modalities are composed by two symbols, where the first one is chosen among E or A, and the second one among X, F, G and the bi-modality U. The second symbol is used for temporal reasoning: X stands for ne**x**t; F for sometime in the **f**uture; G for always in the future; and U for **u**ntil. The first symbol quantifies over all computation paths: an existential (E - for there **e**xists) path or a universal (A - for **a**ll) paths. The combination of the two symbols can be easily understood. For example, the formula $\mathsf{EX}\delta$ holds in a probability structure (V, p) if there **e**xists a ne**x**t structure of (V, p) (that is, a structure reachable from (V, p) with a single transition) that satisfies δ. As usual, all CTL modalities are obtained as abbreviations from EX, AF and EU.

- $(\mathsf{AX}\,\delta)$ for $\ominus \mathsf{EX}(\ominus\delta)$;
- $(\mathsf{EF}\,\delta)$ for $\ominus(\mathsf{E}[(\ominus \perp\!\!\!\perp)\mathsf{U}\delta])$;
- $(\mathsf{AG}\,\delta)$ for $\ominus(\mathsf{EF}(\ominus\delta))$;
- $(\mathsf{EG}\,\delta)$ for $\ominus(\mathsf{AF}(\ominus\delta))$;
- $\mathsf{A}[\delta_1 \mathsf{U} \delta_2]$ for $\ominus(\mathsf{E}[(\ominus\delta_2)\mathsf{U}(\ominus\delta_1 \cap \boxminus\delta_2)]) \cap (\ominus(\mathsf{EG}(\ominus\delta_2)))$.

Example 3.1 Consider again the Russian roulette variant from Example 2.1 together with some temporal primitives. First, we will like to state that the bullet can not be shot before the outcome of the coin is heads, which can be expressed as $\mathsf{A}[((\int b) = 0)\mathsf{U}((\int h) > 0)]$. Suppose that the gambler is always playing this game alone, clearly the probability of killing himself tends asymptotically to 1, we can capture this statement with $((x < 1) \supset AF((\int d) > x))$.

3.2 Semantics

A *probabilistic Kripke structure* is a pair $(\mathcal{P}, R)$ where $\mathcal{P}$ is a set of probabilistic structures and $R \subseteq \mathcal{P} \times \mathcal{P}$ is a *total transition relation*, that is, for any $(V, \mu) \in \mathcal{P}$ there exists (V', μ') such that $(V,\mu)\,R\,(V',\mu')$. The notion of probabilistic Kripke structure is very general, and, as we shall see, it is capable of capturing Markov transitions (and more) as well as systems with both non–deterministic and probabilistic transitions.

Example 3.2 Consider the Russian roulette from Example 2.1, and consider that the gambler plays the game twice and then, if alive, halts. The probabilistic Kripke structure is such that all probability structures involved have the set of admissible valuations $V = \{\emptyset, \{h\}, \{h,b,d\}\}$. Assume that the initial distribution μ_0 is $\mu(\emptyset) = 1$. The probability distribution over V evolves accordingly to the following stochastic matrix

$$M = \begin{bmatrix} \frac{1}{2} & \frac{5}{12} & \frac{1}{12} \\ \frac{1}{2} & \frac{5}{12} & \frac{1}{12} \\ 0 & 0 & 1 \end{bmatrix}$$

so, assuming that the gambler's only choice is to play twice and then halt if alive, we have that $\mu_0\,R\,\mu_1$ and $\mu_1(\emptyset) = 1/2, \mu_1(\{h\}) = 5/12$ and $\mu_1(\{h,b,d\}) = 1/12$; moreover, $\mu_1\,R\,\mu_2$, with $\mu_2(\emptyset) = 11/24, \mu_2(\{h\}) = 55/144$ and $\mu_2(\{h,b,d\}) = 23/144$; and finally $\mu_2\,R\,\mu_2$.

The interpretation of probabilistic terms is defined as before. The satisfaction of a temporal formula is defined over a probabilistic Kripke structure $(\mathcal{P},R)$, a probabilistic structure $(V,\mu) \in \mathcal{P}$ and an assignment ρ.

- $(\mathcal{P},R),(V,\mu),\rho \Vdash (\Box\gamma)$ iff $(V,\mu),\rho \Vdash (\Box\gamma)$;
- $(\mathcal{P},R),(V,\mu),\rho \Vdash (p_1 \leq p_2)$ iff $(V,\mu),\rho \Vdash (p_1 \leq p_2)$;
- $(\mathcal{P},R),(V,\mu),\rho \nVdash \perp\!\!\!\perp$;
- $(\mathcal{P},R),(V,\mu),\rho \Vdash (\delta_1 \supset \delta_2)$ iff $(\mathcal{P},R),(V,\mu),\rho \nVdash \delta_1$ or $(\mathcal{P},R),(V,\mu),\rho \Vdash \delta_2$;
- $(\mathcal{P},R),(V,\mu),\rho \Vdash (\mathsf{EX}\delta)$ iff $(\mathcal{P},R),(V',\mu'),\rho \Vdash \delta$ with $(V,\mu)\,R\,(V,\mu)$;
- $(\mathcal{P},R),(V,\mu),\rho \Vdash (\mathsf{AF}\delta)$ iff for all path π over R starting in (V,μ) there exist $k \in \mathbb{N}$ such that $(\mathcal{P},R),\pi_k,\rho \Vdash \delta$;
- $(\mathcal{P},R),(V,\mu),\rho \Vdash (\mathsf{E}[\delta_1 \mathsf{U} \delta_2])$ iff there exist path π over R starting in (V,μ) and $k \in \mathbb{N}$ such that $(\mathcal{P},R),\pi_k,\rho \Vdash \delta_2$ and $(\mathcal{P},R),\pi_i,\rho \Vdash \delta_1$ for every $i \leq k$;

where π_i denotes the i-element of the path π.

We say that $(\mathcal{P},R) \Vdash \delta$ iff $(\mathcal{P},R),(V,\mu),\rho \Vdash \delta$ for all $(V,\mu) \in \mathcal{P}$ and assignment ρ.

Example 3.3 Consider the probabilistic Kripke structure $(\mathcal{P},R)$ of Example 3.2. The structure satisfies the property that the probability of dying is non decreasing, that is, $(\mathcal{P},R) \Vdash (((\int d) = x) \supset (\mathsf{AG}((\int d) \geq x)))$.

3.3 Model–checking EpCTL

We now address the problem of model–checking a temporal formula. Following the usual model–checking technique for CTL, the goal is to compute the set

$$Sat_{(\mathcal{P},R),\rho}(\delta) := \{(V,\mu) \in \mathcal{P} : (\mathcal{P},R),(V,\mu),\rho \Vdash \delta\}$$

for a probabilistic Kripke structure $(\mathcal{P},R)$, assignment ρ and formula δ. This is called the *global model–checking problem*. Before presenting the model–checking algorithm, it is useful to introduce some notation for relations, namely in the context of a probabilistic Kripke structure $(\mathcal{P},R)$. We denote by R^{-1} the inverse relation of R, that is, $(V,\mu)R^{-1}(V',\mu')$ iff $(V',\mu')R(V,\mu)$. Given a set of probabilistic strucutures $X \subseteq \mathcal{P}$, we denote by RX the set $\{(V,\mu) \in \mathcal{P} : \text{there exists } (V',\mu') \in X \text{ such that } (V',\mu')R(V,\mu)\}$. We are now able to present a model–checking algorithm, adapted from the usual algorithm for CTL:

(i) $Sat_{(\mathcal{P},R),\rho}(\Box\gamma) = \{(V,\mu) \in \mathcal{P} : (V,\mu),\rho \Vdash (\Box\gamma)\}$;

(ii) $Sat_{(\mathcal{P},R),\rho}(p_1 \leq p_2) = \{(V,\mu) \in \mathcal{P} : (V,\mu),\rho \Vdash (p_1 \leq p_2)\}$;

(iii) $Sat_{(\mathcal{P},R),\rho}(\delta_1 \supset \delta_2) = (\mathcal{P} \setminus Sat_{(\mathcal{P},R),\rho}(\delta_1) \cup Sat_{(\mathcal{P},R),\rho}(\delta_2)$;

(iv) $Sat_{(\mathcal{P},R),\rho}(\mathsf{EX}\delta) = R^{-1}Sat_{(\mathcal{P},R),\rho}(\delta)$;

(v) $Sat_{(\mathcal{P},R),\rho}(\mathsf{AF}\delta) = \mathbf{FixedPoint}[F_{(\mathsf{AF}\delta)}]$ with

$$F_{(\mathsf{AF}\delta)}(X) = Sat_{(\mathcal{P},R),\rho}(\delta) \cup \{(V,\mu) \in \mathcal{P} : R\{(V,\mu)\} \subseteq X\};$$

(vi) $Sat_{(\mathcal{P},R),\rho}(\mathsf{E}[\delta_1 \mathsf{U} \delta_2]) = \mathbf{FixedPoint}[F_{\mathsf{E}[\delta_1 \mathsf{U} \delta_2]}]$ with

$$F_{\mathsf{E}[\delta_1 \mathsf{U} \delta_2]}(X) = Sat_{(\mathcal{P},R),\rho}(\delta_2) \cup (Sat_{(\mathcal{P},R),\rho}(\delta_1) \cap R^{-1}X).$$

In general, a probabilistic Kripke structure requires exponential space (over the number of propositional symbols) due to the exponential spanning of probabilities on the distribution over the valuations. For this reason, the model–checking algorithm takes exponential time on the number of propositional symbols, but it is polynomial on the size of the probabilistic Kripke structure and the complexity of the formula.

Theorem 3.4 Assuming that all basic arithmetical operations take unit time, the model–checking algorithm for EpCTL takes $O(|\delta|^2 \cdot |\mathcal{P}|^2 \cdot 2^n)$ time for inputs $(\mathcal{P},R),\rho$ and δ, where δ is written with n propositional symbols.

It is well known that a slightly better algorithm can be obtained if EG is taken as a basic modality instead of AF, but we refrain from doing so here. Although the algorithm is exponential in the worst case, it assumes that a probability distribution over valuations is encoded as a vector of probabilities. Clearly, for particular relevant cases, there are much more efficient and compact encodings; we are currently investigating which probability distributions can be encoded efficiently.

3.4 Expressiveness and Scope

In this section we discuss the scope and expressiveness of EpCTL. Substantial work has been done on model–checking probabilistic behaviour, namely using PCTL and probabilistic transition systems [2]. In order to fully understand the scope of EpCTL we will show how one can construct a probabilistic Kripke structure from a probabilistic transition system encompassing both probability and non–determinism [19]. To this end we consider the following simplified notion of probabilistic transition systems that we adapted from [2].

Definition 3.5 A *probabilistic transition system* is a tuple $T = (S,A,M)$ where S is a set of states, A is a set of (non-deterministic) actions and $M = \{M_a\}_{a \in A^*}$ is a Markov chain over S for each $a \in A$.

A probabilistic transition system evolves by a (non–deterministic) scheduler that chooses an action $a \in A$ and then the stochastic transition M_a occurs. As usual in temporal reasoning, we disregard the set of actions A that induce the transitions, and care only about the transitions. It is useful to describe a probabilistic transition systems as a stochastic process $\{X_w\}_{w\in A^*}$ where each random variable X ranges over S and A^* (words correspond to generative choices of the scheduler) constitutes the index space. To generate a probabilistic Kripke structure one requires an initial distribution over the states, say X_ε, and a labeling function $\ell : S \to 2^\Phi$. For the sake of simplicity, we assume that S is denumerable, and thus each random variable X_w can be presented as a map $X_w : S \to [0,1]$ associating to each state its probability. The labeling function ℓ maps each state to the set of propositional symbols holding in that state. Given a choice by the scheduler $w \in A^*$ we describe the evolution of the state by the stochastic matrix $M_w = M_{a_n} \dots M_{a_1}$ with $w = a_1 \dots a_n$, and the resulting state is $X_w = M_w.X_\varepsilon$ [10] . It is easy to extract a probabilistic structure (V, μ_w) from each X_w by taking $V = 2^\Phi$ and $\mu_w(v) = \sum_{s:\ell(s)=v} X_w(s)$. It is easy to construct a probabilistic Kripke structure $(\mathcal{P}, R)$ over Φ, by taking $\mathcal{P} = \{(2^\Phi, \mu_w) : w \in A^*\}$ and $(2^\Phi, \mu_w) R (2^\Phi, \mu_{w'})$ iff $w' = w.a$ for some $a \in A$ [11] . Observe that the constructed relation R does not need to be a tree, since it may happen that $(2^\Phi, \mu_w) = (2^\Phi, \mu_{w'})$ for $w \neq w$.

Taking into account the above construction we have a common ground to understand better the differences between EpCTL and PCTL. PCTL enables reasoning about the distribution over paths in probabilistic transition systems while EpCTL is designed for reasoning about the evolution of the distribution of propositional symbols. Depending on the application, it can be better or worse to model a given property using distributions over paths or over the propositional symbols, since both approaches are valid, but quite different. Hence, the PCTL formula $\mathsf{AG}_{>q}\varphi$ states that, for any choice of the scheduler, the measure of paths satisfying $G\varphi$ is greater than q. On the other hand the EpCTL formula $AG(\int \varphi > q)$ means that for any choice of the scheduler, all the state distributions reached are such that the probability of φ holding is greater than q. Given that in PCTL the probabilities are endogenous in the modalities, it does not seem to be possible to express some more sophisticated types of property, such as: $AG((\int \varphi_1 \cdot \int \varphi_2) > q)$.

Observe also that it is common to describe the state of a random (distributed) algorithm or protocol as a probability distribution over the memory cells [30,14,9]. In this case, EpCTL is particularly suitable because the state logic EPPL reasons about distributions over memory cells (Boolean registers = propositional symbols), while state evolution is captured by the temporal modalities. This is the main motivation behind the design of EpCTL, and is fully explored in Section 4, where two examples are described in detail. On the other hand, it is worthwhile noting that for model–checking EpCTL one requires the number of reachable probability distributions over states to be finite. While in most security protocols and terminating random algorithms the number of reachable distributions is finite, in some distributed settings where the processes run for ever this might not be the case.

Observe that a probabilistic Kripke structure $(\mathcal{P}, R)$ does not impose any restrictions on two related distributions (V, μ) and (V', μ'). This makes it possible to model general stochastic processes, not only Markov chains. This matters little in practice, since, in the majority of

[10] Observe that $\{X_w\}_{w\in A^*}$ constitutes a so–called *heterogeneous Markov chain*, given that for each temporal line of choices, that is, infinite word w, the associated (sub)stochastic process $Y^w = \{Y^w_i\}_{i\in\mathbb{N}}$ with $Y^w_i = X_{w_0 \dots w_i}$ is a Markov chain

[11] The stochastic process $\{\mu_w\}_{w\in A^*}$ ranging over 2^Φ is a so–called *Markov modulated process,* given that it is modulated by $\{X_w\}_{w\in A^*}$

applications, it is heterogeneous Markov chains that are used to describe system evolution. Clearly the differences between PCTL and EpCTLneed to be investigated further, and we will endeavour to do so in the full version of this paper.

3.5 Axiomatisation of EpCTL

Given a complete axiomatisation for the state logic EPPL, a sound proof system for the EpCTL follows directly; we adapt known complete axiomatisations for CTL.

- Axioms
  - · [**Inq**] All valid EPPL formulas;
  - · [**Taut**] All tautologies with propositional symbols substituted by EpCTL formulas;
  - · [**EX**] $\vdash \mathsf{EX}(\delta_1 \cup \delta_2) \equiv \mathsf{EX}\delta_1 \cup \mathsf{EX}\delta_2$
  - · [**X**] $\vdash \mathsf{AX}(\ominus \perp\!\!\!\perp) \cap \mathsf{EX}(\ominus \perp\!\!\!\perp)$
  - · [**EU**] $\vdash \mathsf{E}[\delta_1 \mathsf{U} \delta_2] \equiv \delta_2 \cup (\delta_1 \cap \mathsf{EXE}[\delta_1 \mathsf{U} \delta_2])$
  - · [**AU**] $\vdash \mathsf{A}[\delta_1 \mathsf{U} \delta_2] \equiv \delta_2 \cup (\delta_1 \cap \mathsf{AXA}[\delta_1 \mathsf{U} \delta_2])$
  - · [**AG1**] $\vdash \mathsf{AG}(\delta_3 \supset ((\ominus \delta_2) \cap \mathsf{EX}\delta_3)) \supset (\delta_3 \supset (\ominus \mathsf{A}[\delta_1 \mathsf{U} \delta_2]))$
  - · [**AG2**] $\vdash \mathsf{AG}(\delta_3 \supset ((\ominus \delta_2) \cap (\delta_1 \supset \mathsf{AX}\delta_3))) \supset (\delta_3 \supset (\ominus \mathsf{E}[\delta_1 \mathsf{U} \delta_2]))$
  - · [**AG3**] $\vdash \mathsf{AG}(\delta_1 \supset \delta_2) \supset (\mathsf{EX}\delta_1 \supset \mathsf{EX}\delta_2)$
- Inference rules
  - · [**QMP**] $\delta_1, (\delta_1 \sqsupset \delta_2) \vdash \delta_2$
  - · [**AGen**] $\delta_1 \vdash \mathsf{AG}\delta_1$

While it is out of the scope of this paper to give a (weakly) complete set of axioms for EpCTL, we intend to provide them in the final version.

4 Illustrative Examples

In order to demonstrate the expressiveness of EpCTL, we consider a couple of examples from the literature, starting with a model of the contract-signing protocol due to Ben-Or *et al.* [6]. We then consider a simple example of a protocol from the area of quantum cryptography, the quantum one–time pad [1], noting that this particular protocol may be modelled entirely in a probabilistic setting, and its properties formalised in a classical probabilistic (as opposed to specifically quantum) formalism.

4.1 A Contract Signing Protocol

The problem of contract signing is to find a way of getting two users, A and B, to commit to a contract $\mathcal{C}$ in such a way that neither party may falsely convince the other that the former has signed. In other words, A and B must sign the contract together, without one party gaining any advantage over the other. The traditional solution to the problem is for A and B to sign $\mathcal{C}$ simultaneously, but this is only possible if A and B are in physical proximity. Assuming that A and B are spatially separated, the only way for contract signing to be achieved is through a communications protocol, although it is likely that, at different stages of such a protocol, one party will have a relative advantage over the other. The objective of a 'fair' contract signing protocol, such as the one proposed by Ben-Or *et al.* [6] (henceforth referred to simply as the BGMR protocol) is to constrain this relative advantage so that it remains within specific bounds tolerated and agreed upon by both users.

The BGMR protocol assumes the setting of a network of users (we focus only on the two user case) with a signature scheme in operation. Only user U is assumed capable of producing U's signature on message m (*unforgeability*) and any other user is assumed capable of verifying the validity of U's signature on m (*universal verifiability*). The protocol assumes that neither user A nor user B wants to be committed to contract $\mathcal{C}$ unless the other user is, and makes it possible for A and B to sign $\mathcal{C}$ by exchanging commitments. The notion of *fairness* for the protocol is defined as the property that, the conditional probability with which "B is not privileged" given that "A is privileged" is always small. Formally, the BGMR protocol is said to be (v,ε)–fair, this being defined as follows.

Definition 4.1 A contract signing protocol is (v,ε)–fair for A if the following holds, for any contract $\mathcal{C}$, when A follows the protocol properly: At any step of the protocol in which the probability that "B is privileged" is greater than v, the conditional probability that "A is not privileged" given that "B is privileged" is at most ε.

If A and B are assumed to be dishonest, then a third party – a *judge* – must be invoked during the protocol in order to provide an independent judgement as to whether the contract is to be considered binding for both users. During the protocol, A and B exchange signed messages of the following form:

$$m = (C,p,U) = \text{" With probability } p\text{, the contract } C \text{ shall be valid. Signed, User } U. \text{ "}$$

When message m is received, the recipient is said to be privileged with probability p, meaning that invocation of the judge will result in him ruling that contract $\mathcal{C}$ is binding to A and B with probability p. If the protocol does not terminate successfully by a pre–agreed date D, one of the two users invokes an early stopping procedure.

We are now ready to state the BGMR protocol in detail. Steps 1–5 are for initialization of protocol parameters.

(i) Parties A and B agree who goes first and set a termination date D. We assume that A is to go first.

(ii) Party A chooses the conditional probability v that "B is privileged" while "A is not privileged."

(iii) Party A chooses the parameter $\alpha > 1$ such that, the conditional probability that "A is privileged" given that "B is privileged" is at least $\frac{1}{\alpha}$.

(iv) Party B chooses $\beta > 1$ such that, the conditional probability that "B is privileged" given that "A is privileged" is at least $\frac{1}{\beta}$.

(v) The protocol is initialised with $\lambda_A = \lambda_B = 0$. The symbol λ_A stores the probability mentioned in the message last sent from A, and similarly λ_B stores the probability mentioned in the message last transmitted from B.

(vi) A and B perform the following procedures alternately:

A-step. User A denotes the probability mentioned in the last message received by p. A then checks whether $p \geqslant \lambda_A$. If so, then A sets $\lambda_A := \max(v, \min(1, p\cdot\alpha))$. Otherwise, A assumes the protocol has been terminated. A then transmits message $(\mathcal{C},\lambda_A,A)$ to B.

B-step. User B denotes the probability mentioned in the last message received by p. B then checks whether $p \geqslant \lambda_B$. If so, then B sets $\lambda_B := \min(1, p\cdot\beta)$. Otherwise, B assumes the protocol has been terminated. B then transmits message $(\mathcal{C},\lambda_B,B)$ to B.

The details of the judge's procedure, and of the early stopping procedure, are to be found in [6]. An analysis of the protocol has been performed using the PRISM model-checker by Norman and Shmatikov [27].

The essential point about the BGMR protocol is that it ensures a specified degree of fairness, characterised by the constants v and ε. At the end of the protocol, both parties need to be privileged. We formalise the notion of (v,ε)–fairness using EpCTL in what follows.

We establish the set $\Phi = \{\varphi_A, \varphi_B\}$ of propositional constants, where φ_A corresponds to the truth of the event "A is privileged," and similarly φ_B is true if "B is privileged." To express the fairness property, we regard the protocol parameter ε as a real variable, namely a member of the set Y defined in Section 2. The probability v which party A fixes in step (ii) above may be expressed as the following term in EpCTL :

$$v = (\textstyle\int \varphi_B|(\neg\varphi_A))$$

In steps (iii) and (iv) of the protocol, parties A and B fix the parameters α and β respectively such that the following EpCTL properties are true:

(1) $(\int \varphi_A|\varphi_B) \geqslant \dfrac{1}{\alpha}$

(2) $(\int \varphi_B|\varphi_A) \geqslant \dfrac{1}{\beta}$

The property of (v,ε)–fairness may be expressed thus:

$$AG(((\textstyle\int \varphi_B) > v) \supset ((\int(\neg\varphi_A)|\varphi_B) \leqslant \varepsilon))$$

that is to say, in all paths of the protocol, the probability that "A is not privileged" given that B is, assuming that the probability of "B being privileged" is greater than v, remains less than ε. Note that we are using the comparators $>, \geqslant$ freely; these may be expressed in terms of the $\leqslant$ operator in the formal syntax of EpCTL.

4.2 *Quantum One Time Pad*

A qubit is the basic memory unit in quantum computation (just as a bit is the basic memory unit in classical computation). The state of a qubit is a pair (α,β) of complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$. A *quantum one time pad* [1] encrypts a qubit using two key (classical) bits in a secure way: observing the encrypted qubit yields two results, both with equal probability. In the special case that α and β are real numbers one bit key φ^K suffices. If $\varphi^K = 1$ then the qubit is encrypted as the pair $(\beta, -\alpha)$, otherwise it remains the same. We consider that a real number α is encoded using floating point representation, namely a vector of propositional symbols $\varphi_1^\alpha \dots \varphi_n^\alpha$ which we denote just by φ^α. We will abbreviate by $\alpha = \beta$ the classical formula $(\varphi_1^\alpha \Leftrightarrow \varphi_1^\beta) \wedge \dots \wedge (\varphi_n^\alpha \Leftrightarrow \varphi_n^\beta)$.

The following program simulates this process by first generating a random key and then encrypting the qubit (α,β):

```
(i) Let $\varphi^K$ := outcome of a fair Bernoulli trial;
(ii) If ($\varphi^K = 1$) then
    (a) $\gamma := \alpha$
    (b) $\alpha := \beta$
    (c) $\beta := -\gamma$
```

Assume that the initial values of α and β are c and d respectively (with $c \neq d$). It follows from quantum information theory that in order to prove the security of the quantum one-time pad, it suffices to show that the probability after the encryption of α being c is $\frac{1}{2}$ (and hence of α being d is also $\frac{1}{2}$). We can use our logic and model checking procedure to show the above by considering the probabilistic Kripke structure induced by the encryption program. Assume that the program induces a single transition in the Kripke structure, and from that point on the probability distribution over the states remains the same. Therefore, the security of the quantum one-time pad is equivalent to checking that all initial states fulfill

$$(\Box(\alpha = c \wedge \beta = d \wedge (\neg(c = d)))) \supset \mathsf{AX}((\textstyle\int(\alpha = c)) = \frac{1}{2}).$$

5 Summary and Conclusion

In this paper we have introduced a probabilistic branching-time logic, EpCTL, which may be regarded as a temporal extension of the exogenous probabilistic state logic EPPL. We have stated the syntax and semantics of EPPL, considered the model–checking problem for formulas in this state logic, and presented an axiomatisation for it. We described the EpCTL extension, stating syntax, semantics, and model–checking issues in an analogous way to EPPL. The expressiveness of EpCTL was discussed, and an axiomatisation was given. We demonstrated the use of EpCTL as a means of expressing properties of two security protocols: a classical probabilistic contract signing protocol, and the quantum one–time pad.

Our approach has been inspired by earlier work by Halpern, and we expect the probabilistic temporal logic EpCTL to serve as a useful alternative to related classical logics such as PCTL. Future work will include refining the axiomatisation of EpCTL, considering possible improvements to the model–checking algorithm, and implementing the algorithm. Further work on case studies is necessary, especially with a view to classifying and verifying the types of properties which typically arise in security.

We hope this work will serve as a basis for ongoing work in developing an exogenous, temporal quantum logic for model–checking general quantum protocols. A quantum state logic, exogenous quantum propositional logic (EQPL) was proposed in [23]; we intend to provide a temporal extension of that logic, extending the techniques described in the present paper. Thus we have the necessary ingredients for building a dedicated model–checking tool for the analysis of quantum cryptographic and communication systems, which is one of our long-term goals.

References

[1] Ambainis, A., M. Mosca, A. Tapp and R. de Wolf, *Private quantum channels*, in: *FOCS '00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science* (2000), p. 547.

[2] Baier, C. and M. Z. Kwiatkowska, *Model checking for a probabilistic branching time logic with fairness*, Distributed Computing **11** (1998), pp. 125–155.
URL `citeseer.ist.psu.edu/article/kwiatkowska96model.html`

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

[4] Beauquier, D., A. M. Rabinovich and A. Slissenko, *A logic of probability with decidable model-checking*, in: *CSL '02: Proceedings of the 16th International Workshop and 11th Annual Conference of the EACSL on Computer Science Logic* (2002), pp. 306–321.

[5] Ben-Ari, M., Z. Manna and A. Pnueli, *The temporal logic of branching time*, in: *POPL '81: Proceedings of the 8th ACM SIGPLAN-SIGACT symposium on Principles of programming languages* (1981), pp. 164–176.

[6] Ben-Or, M., O. Goldreich, S. Micali and R. L. Rivest, *A fair protocol for signing contracts*, IEEE Transactions on Information Theory **36** (1990), pp. 40–46.

[7] Carnielli, W. A., *Possible-translations semantics for paraconsistent logics*, in: *Frontiers of Paraconsistent Logic (Ghent, 1997)*, Studies in Logic and Computation **8** (2000), p. 149163.

[8] Carnielli, W. A. and M. Lima-Marques, *Society semantics and multiple–valued logics*, in: *Advances in Contemporary Logic and Computer Science (Salvador, 1996)*, Contemporary Mathematics **235** (1999), pp. 33–52.

[9] Chadha, R., P. Mateus and A. Sernadas, *Reasoning about states of probabilistic sequential programs*, in: Z. Ésik, editor, *Computer Science Logic 2006 (CSL06)*, Lecture Notes in Computer Science **4207**, Springer-Verlag, 2006 pp. 240–255.

[10] Clarke, E. M. and E. A. Emerson, *Design and synthesis of synchronization skeletons using branching time temporal logics*, in: *Proceeding of the Workshop on Logics of Programs*, LNCS **131**, Springer-Verlag, 1981 .

[11] Clarke, E. M. and E. A. Emerson, *Design and synthesis of synchronization skeletons using branching-time temporal logic*, in: *Logic of Programs, Workshop* (1982), pp. 52–71.

[12] Clarke, E. M., E. A. Emerson and A. P. Sistla, *Automatic verification of finite-state concurrent systems using temporal logic specifications*, ACM Trans. Program. Lang. Syst. **8** (1986), pp. 244–263.

[13] Clarke, E. M. and J. M. Wing, *Formal methods: state of the art and future directions*, ACM Comput. Surv. **28** (1996), pp. 626–643.

[14] den Hartog, J. and E. de Vink, *Verifying probabilistic programs using a hoare like logic*, International Journal of Foundations of Computer Science **13** (2002), pp. 315–340.

[15] Fagin, R., J. Y. Halpern and N. Megiddo, *A logic for reasoning about probabilities*, Information and Computation **87** (1990), pp. 78–128.
URL `citeseer.ist.psu.edu/fagin90logic.html`

[16] Hansson, H. and B. Jonsson, *A logic for reasoning about time and reliability*, Formal Aspects of Computing **6** (1994), pp. 512–535.
URL `citeseer.ist.psu.edu/hansson94logic.html`

[17] Hodges, W., "Model Theory," Cambridge University Press, 1993.

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

[19] Kwiatkowska, M., 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* (2002), pp. 200–204.

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

[21] Kwon, Y. and G. Agha, *Linear inequality LTL (iLTL): A model checker for discrete time Markov chains*, in: J. Davies, W. Schulte and M. Barnett, editors, *Proceedings of 6th International Conference on Formal Engineering Methods (ICFEM 2004)*, Lecture Notes in Computer Science **3308** (2004).

[22] Mateus, P. and A. Sernadas, *Exogenous quantum logic*, in: W. Carnielli, F. Dionísio and P. Mateus, editors, *Proceedings of CombLog'04, Workshop on Combination of Logics: Theory and Applications* (2004), pp. 141–149.

[23] Mateus, P. and A. Sernadas, *Reasoning about quantum systems*, in: J. Alferes and J. Leite, editors, *Logics in Artificial Intelligence, Ninth European Conference, JELIA'04*, Lecture Notes in Artificial Intelligence **3229** (2004), pp. 239–251.

[24] Mateus, P. and A. Sernadas, *Weakly complete axiomatization of exogenous quantum propositional logic*, Information and Computation **204** (2006), pp. 771–794, arXiv math.LO/0503453.
URL
`http://www.elsevier.com/wps/find/journaldescription.cws_home/622844/description#description`

[25] Mateus, P., A. Sernadas and C. Sernadas, *Exogenous semantics approach to enriching logics.*, in: G. Sica, editor, *Essays on the Foundations of Mathematics and Logic* (2005), pp. 165–194.

[26] Nilsson, N. J., *Probabilistic logic*, Artificial Intelligence **28** (1986), pp. 71–87.

[27] Norman, G. and V. Shmatikov, *Analysis of probabilistic contract signing*, Journal of Computer Security (2006), to appear.

[28] Papanikolaou, N., "Techniques for Design and Validation of Quantum Protocols," Master's thesis, Department of Computer Science, University of Warwick (2005), also available as Research Report CS-RT-413.

[29] Pnueli, A., *The temporal logic of programs*, in: *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science (FOCS 1977)*, 1977, p. 4657.

[30] Ramshaw, L. H., "Formalizing the analysis of algorithms." Ph.D. thesis, Stanford University (1979).

Appendix

A Proof of Theorem 2.3

Proof. First notice that the terms that take longer to evaluate are those of the type $(\int \gamma)$ and $(\Box\gamma)$. The number of terms of type $(\int \gamma)$ is bounded by $|\xi|$. To evaluate one of these terms we require $O(|V|)$ time corresponding to traveling throughout all the valuations satisfying γ and summing all the associated probabilities. So, computing all $(\int \gamma)$ terms takes $O(|\xi|.|V|)$ time. The same expression will be obtained to check the satisfaction of $(\Box\gamma)$.

After these values are obtained, the remaining computation (comparing terms, negating a boolean value, and making implications between boolean values) takes at most $O(|\xi|)$ time. Hence, the total time to decide if a if a probabilistic structure over Φ and an assignment ρ satisfy ξ is $O(|\xi|.|V|+|\xi|) = O(|\xi|.|V|)$. $\Box$

B Proof of Theorem 2.4

Proof. The result follows from the fact that EPPL logic present here is a sublanguage of that presented in [9], for which the corresponding axiomatisation is proved to be sound and weakly complete. Hence, for further details look at [9]. $\Box$

C Proof of Theorem 3.4

Proof. The propositional CTL model–checking algorithm takes $O(|\delta| \cdot |\mathcal{P}|^2)$ (see [10] for a detailed analysis). So, if we consider each $(\Box\gamma)$ and $p_1 \leq p_2$ to be a propositional symbol, the time complexity of the algorithm would be $O(|\delta| \cdot |\mathcal{P}|^2)$. Finally, since checking if these formulas are satisfied by a $(\mathcal{P},\mu)$ and ρ takes $O(|\delta| \cdot 2^n)$ (c.f. Theorem 2.3) we derive the desired upper bound. Recall that we consider all arithmetic computations to be $O(1)$ by using floating point representation for the real numbers. $\Box$