Reasoning about states of probabilistic sequential programs[⋆]

R. Chadha, P. Mateus and A. Sernadas

SQIG – IT and IST, Portugal
rch,pmat,acs@math.ist.utl.pt

Abstract. A complete and decidable propositional logic for reasoning about states of probabilistic sequential programs is presented. The state logic is then used to obtain a sound Hoare-style calculus for basic probabilistic sequential programs. The Hoare calculus presented herein is the first probabilistic Hoare calculus with a complete and decidable state logic that has truth-functional propositional (not arithmetical) connectives. The models of the state logic are obtained exogenously by attaching sub-probability measures to valuations over memory cells. In order to achieve complete and recursive axiomatization of the state logic, the probabilities are taken in arbitrary real closed fields.

1 Introduction

Reasoning about probabilistic systems is very important due to applications of probability in distributed systems, security, reliability, and randomized and quantum algorithms. Logics supporting such reasoning have branched in two main directions. Firstly, Hoare-style [27, 21, 6] and dynamic logics [9, 17] have been developed building upon denotational semantics of probabilistic programs [16]. The second approach enriches temporal modalities with probabilistic bounds [10, 13, 23].

Our work is in the area of Hoare-style reasoning about probabilistic sequential programs. A Hoare assertion [11] is a triple of the form $\{\xi_1\}\ s\ \{\xi_2\}$ meaning that if program s starts in state satisfying the state assertion formula ξ_1 and s halts then s ends in a state satisfying the state transition formula ξ_2. The formula ξ_1 is known as the pre-condition and the formula ξ_2 is known as the post-condition. For probabilistic programs the development of Hoare logic has taken primarily two different paths. The common denominator of the two approaches is forward denotational semantics of sequential probabilistic programs [16]: program states are (sub)-probability measures over valuations of memory cells and denotations of programs are (sub)-probability transformations.

The first sound Hoare logic for probabilistic programs was given in [27]. The state assertion language is *truth-functional*, *i.e.*, the formulas of the logic are interpreted as either true and false and the truth value of a formulas is determined by the truth values of the sub-formulas. The state assertion language in [27] consists of two levels: one classical state formulas γ interpreted over the valuations of memory cells and the second

⋆ Supported by FCT and FEDER through POCI via CLC QuantLog POCI/MAT/55796/2004 Project. Additional support for Rohit Chadha came from FCT and FEDER grant SFRH/BPD/26137/2005.

probabilistic state formulas ξ which interpreted over (sub)-probability measures of the valuations. The state assertion language contain terms $(\int \gamma)$ representing probability of γ being true. The language at the probabilistic level is extremely restrictive and is built from term equality using conjunction. Furthermore, the Hoare rule for the alternative if-then-else is incomplete and even simple valid assertions may not be provable.

The reason for incompleteness of the Hoare rule for the alternative composition in [27] as observed in [27, 17] is that the Hoare rule tries to combine absolute information of the two alternates truth-functionally to get absolute information of the alternative composition. This fails because the effects of the two alternatives are not independent. In order to avoid this problem, a probabilistic dynamic logic is given in [17] with an *arithmetical* state assertion logic: the state formulas are interpreted as measurable functions and the connectives are arithmetical operations such as addition and subtraction.

Inspired by the dynamic logic in [17], there are several important works in the probabilistic Hoare logic, *e.g.* [14, 21], in which 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 Hoare logic with arithmetical state logics has yielded several interesting results, the Hoare triples themselves do not seem very intuitive. A high degree of sophistication is required to write down the Hoare assertions needed to verify relatively simple programs. For this reason, it is worthwhile to investigate Hoare logics with truth-functional state logics.

A sound Hoare logic with a truth-functional state logic was presented in [6] and completeness for a fragment of the Hoare-logic is shown for iteration-free programs. In order to deal with alternative composition, a probabilistic sum construct $(\xi_1 + \xi_2)$ is introduced in [6]. Intuitively, the formula $(\xi_1 + \xi_2)$ is satisfied by a (sub)-probability measure μ if μ can be be written as the sum of two measures μ_1 and μ_2 which satisfy ξ_1 and ξ_2 respectively. The drawback of [6] is that no axiomatization is given for the state assertion logic. The essential obstacle in achieving a complete axiomatization for the state language in [6] is the probabilistic sum construct.

This paper addresses the gap between [27] and [6] and provides a sound Hoare logic for iteration-free probabilistic programs with a truth-functional state assertion logic. Our main contribution is that the Hoare logic herein is the first sound probabilistic Hoare logic with a truth-functional state assertion logic that enjoys a complete and decidable axiomatization.

We tackle the Hoare rule for the alternative composition in two steps. The first step is that our alternative choice construct is a slight modification of the usual if-then-else construct: we mark a boolean memory variable bm with the choice taken at the end of the execution of the conditional branch. Please note that this does not pose any restriction over the expressiveness of the programming language. This modification gives us a handle on the Hoare rule for the alternative construct as all the choices are marked by the appropriate memory variable and thus become independent. Please note that a fixed dedicated boolean register could have been used to mark the choices. However, we decided to use a boolean variable in the syntax because the Hoare rule for the alternative composition refers to the marker.

The second step is that in our state assertion language, we have a *conditional construct* (ξ/γ). Intuitively, the formula (ξ/γ) is satisfied by a (sub)-probability measure μ if ξ is true of the (sub)-probability measure obtained by eliminating the measure of all valuations where γ is false. The conditional formulas (ξ/bm) and $(\xi/(\neg\,\mathsf{bm}))$ in the state logic can then be used to combine information of the alternate choices.

The state assertion logic, henceforth referred to as Exogenous Probabilistic Propositional Logic (EPPL), is designed by taking the exogenous semantics approach to enriching a given logic–the models of the enriched logic are sets of models of the given logic with additional structure. A semantic model of EPPL is a set of possible valuations over memory cells which may result from execution of a probabilistic program along with a discrete (sub)-probability space which gives the probability of each possible valuation.

Unlike most works on probabilistic reasoning about programs, we do not confuse possibility with probability: possible valuations may occur with zero probability. This is not a restriction and we can confuse the two, if desired, by adding an axiom to the proof system. On the other hand, this separation yields more expressivity. The exogenous approach to probabilistic logics first appeared in [24, 25] and later in [7, 1, 20]. EPPL is an enrichment of the probabilistic logic proposed in [20]: the conditional construct (ξ/γ) is not present in [20].

For the sake of convenience, we work with finitely additive, discrete and bounded measures and not just (sub)-probability measures. In order to achieve recursive axiomatization for EPPL, we also assume that the measures take values from an arbitrary *real closed field* instead of the set of real numbers. The first order theory of such fields is decidable [12, 3], and this technique of achieving decidability was inspired by other work in probabilistic reasoning [7, 1].

The programming language is a basic imperative language with assignment to memory variables, sequential composition, probabilistic assignment $(\mathsf{toss}(\mathsf{bm}, r))$ and the marked alternative choice. The statement $\mathsf{toss}(\mathsf{bm}, r)$ assigns bm to true with probability r. The term r is a constant and does not depend on the state of the program. This is not a serious restriction. For instance r is taken to be $\frac{1}{2}$ in probabilistic Turing machines.

One of the novelties of our Hoare logic is the rule for $\mathsf{toss}(\mathsf{bm}, r)$ which gives the weakest pre-condition and is not present in other probabilistic Hoare logics with truth-functional state logics. The corresponding rule in the arithmetical setting is discussed in Section 6. We envisage achieving a complete Hoare logic but this is out of the scope of this paper.

The rest of the paper is organized as follows. The syntax, semantics and the complete recursive axiomatization of EPPL is presented in Section 2. The programming language is introduced in Section 3 and the sound Hoare logic is given in Section 4. We illustrate the Hoare calculus with an example in Section 5. Related work is discussed in detail in Section 6. We summarize the results and future work in Section 7. The proofs of the lemmas and theorems are in the appendices.

Acknowledgements. We would like to thank Luís Cruz-Filipe and Peter Selinger for useful and interesting discussions. We will also like to thank the anonymous referees whose useful comments have greatly benefited the presentation.

2 Logic of probabilistic states - EPPL

We assume that in our programming language, there are a finite number of memory cells of two kinds: registers containing real values (with a finite range D fixed once and for all) and registers containing boolean values. In addition to reflecting the usual implementation of real numbers as floating-point numbers, the restriction that real registers take values from a finite range D is also needed for completeness results. Please note that instead of reals, we could have also used any type with finite range.

Any run of a program thus probabilistically assigns values to these registers and such an assignment is henceforth called a *valuation*. If we denote the set of valuations by $\mathcal{V}$ then intuitively a semantic structure of EPPL consists of $V \subseteq \mathcal{V}$, a set of *possible* valuations, along with a finitely additive, discrete and bounded measure μ on $\wp\mathcal{V}$, the power-set of $\mathcal{V}$. A finitely additive, discrete and bounded measure μ on $\wp\mathcal{V}$ is a map from $\wp\mathcal{V}$ to $\mathbb{R}^+$ (the set of non-negative real numbers) such that:

- $\mu(\emptyset) = 0$; and
- $\mu(U_1 \cup U_2) = \mu(U_1) + \mu(U_2)$ if $U_1 \cap U_2 = \emptyset$.

Loosely speaking, $\mu(U)$ denotes the probability of a possible valuation being in the set U. A measure μ is said to be a probability measure if $\mu(\mathcal{V}) = 1$. We work with general measures instead of just probability measures as it is convenient to do so. We will assume that impossible valuations are improbable, *i.e.*, we require $\mu(U) = 0$ for any $U \subset (\mathcal{V} \setminus V)$. Please note that $\mu(U)$ may be 0 for $U \subset \mathcal{V}$.

Furthermore, in order to obtain decidability, we shall assume that the measures take values from an arbitrary *real closed field* instead of the set of real numbers. An ordered field $\mathcal{K} = (K, +, ., 1, 0, \leq)$ is said to be a real closed field if the following hold:

- Every non-negative element of the K has a square root in K.
- Any polynomial of odd degree with coefficients in K has at least one solution.

Examples of real closed fields include the set of real numbers with the usual multiplication, addition and order relation. The set of computable real numbers with the same operations is another example. A measure that takes values from a real closed field $\mathcal{K}$ will henceforth be called a $\mathcal{K}$-measure.

Any real closed field has a copy of integers and rationals. We can also take square roots and n-th roots for odd n in a real closed field. As a consequence, we shall assume that there is a fixed set $\mathcal{R}$ of "real constants" for our purposes.

A semantic structure of EPPL thus consists of a set of possible valuations, a real closed field $\mathcal{K}$ and a $\mathcal{K}$-measure on $\wp\mathcal{V}$. We will call these semantic structures *generalized probabilistic structures*. We start by describing the syntax of the logic.

2.1 Language

The language consists of formulas at two levels. The formulas at first level, *classical state formulas* reason about individual valuations over the memory cells. The formulas at second level, *probabilistic state formulas*, reason about generalized probabilistic structures. There are two kinds of terms in the language: *real terms* used in classical state formulas to denote elements from the set D, and *probability terms* used in probabilistic state formulas to denote elements in an arbitrary real closed field. The syntax of the language is given in Table 1 using the BNF notation and discussed below.

Real terms (with the proviso $c \in D$)
$t := \mathsf{xm} \,[\!]\, x \,[\!]\, c \,[\!]\, (t + t) \,[\!]\, (t\,t)$
Classical state formulas
$\gamma := \mathsf{bm} \,[\!]\, b \,[\!]\, (t \leq t) \,[\!]\, \mathsf{ff} \,[\!]\, (\gamma \Rightarrow \gamma)$

Probability terms (with the proviso $r \in \mathcal{R}$)
$p := r \,[\!]\, y \,[\!]\, (\int \gamma) \,[\!]\, (p + p) \,[\!]\, (p\,p) \,[\!]\, \widetilde{r}$
Probabilistic state formulae:
$\xi := (\Box\gamma) \,[\!]\, (p \leq p) \,[\!]\, (\xi/\gamma) \,[\!]\, \mathsf{fff} \,[\!]\, (\xi \supset \xi)$

Table 1. Language of EPPL

Given fixed $\mathbf{m} = \{0, \ldots, m-1\}$, there are two finite disjoint sets of memory variables: $\mathsf{xM} = \{\mathsf{xm}_k : k \in \mathbf{m}\}$ – representing the contents of real registers, and $\mathsf{bM} = \{\mathsf{bm}_k : k \in \mathbf{m}\}$ – representing the contents of boolean registers. We also have two disjoint sets of rigid variables which are useful in reasoning about programs: $\mathsf{B} = \{b_k : k \in \mathbb{N}\}$ – ranging over the truth values $2 = \{\mathsf{ff}, \mathsf{tt}\}$, and $\mathsf{X} = \{x_k : k \in \mathbb{N}\}$ – ranging over elements of D.

The real terms, ranged over by $t, t_1, \ldots$, are built from the sets D, xM and X using the usual addition and multiplication[1]. The classical state formulas, ranged over by $\gamma, \gamma_1, \ldots$, are built from bM, B and comparison formulas $(p_1 \leq p_2)$ using the classical disjunctive connectives ff and $\Rightarrow$. As usual, other classical connectives $(\neg, \vee, \wedge, \Leftrightarrow)$ are introduced as abbreviations. For instance, $(\neg\,\gamma)$ stands for $(\gamma \Rightarrow \mathsf{ff})$.

The probability terms, ranged over by $p, p_1, \ldots$, denote elements of the real closed field in a semantic structure. We also assume a set of variables, $\mathsf{Y} = \{y_k : k \in \mathbb{N}\}$, ranging over elements of the real closed field. The term $(\int \gamma)$ denotes the measure of the set of valuations that satisfy γ. The denotation of the term $\widetilde{r}$ is r if $0 \leq r \leq 1$, 0 if $r \leq 0$ and 1 otherwise.

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 *conditional formulas* (ξ/γ) using the connectives fff and $\supset$. The formula $(\Box\gamma)$ is true when γ is true of every possible valuation in the semantic structure. Intuitively, the conditional (ξ/γ) is true in a generalized probabilistic structure if it is true in the structure obtained by restricting the possible states to the set where γ is true and eliminating the measure of valuations which satisfy $(\neg\,\gamma)$. Other probabilistic connectives $(\ominus, \cup, \cap, \approx)$ are introduced as abbreviations. For instance, $(\ominus\,\xi)$ stands for $(\xi \supset \mathsf{fff})$. We shall also use $(\Diamond\gamma)$ as an abbreviation for $(\ominus(\Box(\neg\,\gamma)))$. Please note that the $\Box$ and $\Diamond$ are not modalities[2].

[1] The arithmetical operations addition and multiplication are assumed to be defined so as to restrict them to the range D.

[2] 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 also be suitably defined. For the sake of clarity, we shall often drop parenthesis in formulas and terms if it does not lead to ambiguity.

2.2 Semantics

Formally, by a *valuation* we mean a map that provides values to the memory variables and rigid variables– $v : (\mathsf{xM} \to D, \mathsf{bM} \to 2, \mathsf{X} \to D, \mathsf{B} \to 2)$. The set of all possible valuations is denoted by $\mathcal{V}$. Given a valuation v, the denotation of real terms $[\![t]\!]_v$ and satisfaction of classical state formulas $v \Vdash_{\mathsf{C}} \gamma$ are defined inductively as expected. Given $V \subseteq \mathcal{V}$, the *extent* of γ in V is defined as $|\gamma|_V = \{v \in V : v \Vdash_{\mathsf{C}} \gamma\}$.

A *generalized probabilistic state* is a triple $(V, \mathcal{K}, \mu)$ where V is a (possibly empty) subset of $\mathcal{V}$, $\mathcal{K}$ a real closed field and μ is a finitely additive, discrete and finite $\mathcal{K}$-measure over $\wp\mathcal{V}$ such that $\mu(U) = 0$ for every $U \subseteq (\mathcal{V} \setminus V)$. We denote the set of all generalized states by $\mathcal{G}$.

Given a classical formula γ we also need the following sub-measure of μ:

$$\mu_\gamma = \lambda U.\ \mu(|\gamma|_U).$$

That is, μ_γ is null outside of the extent of γ and coincides with μ inside it.

For interpreting the probabilistic variables, we need the concept of an assignment. Given a real closed field $\mathcal{K}$, a *$\mathcal{K}$-assignment* ρ is a map from Y to $\mathcal{K}$.

Given a generalized state $(V, \mathcal{K}, \mu)$ and a $\mathcal{K}$-assignment ρ, the denotation of probabilistic terms and satisfaction of probabilistic state formulas are defined inductively in Table 2. Please note that the semantics ensures that if V is empty, then $(V, \mathcal{K}, \mu)\rho \Vdash \gamma$ for any γ. The formula $(\Box\gamma)$ is satisfied only if all $v \in V$ satisfy γ. For non-empty V, the formula $(p_1 \leq p_2)$ is satisfied if the term denoted by p_1 is less than p_2. The formula (ξ/γ) is satisfied by $(V, \mathcal{K}, \mu)$ and ρ if $(|\gamma|_V, \mathcal{K}, \mu_\gamma)$ and ρ satisfy ξ. 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 $(V, \mathcal{K}, \mu)\rho \Vdash \xi$ whenever $(V, \mathcal{K}, \mu)\rho \Vdash \xi_0$ for each $\xi_0 \in \Xi$.

Please note that the $\mathcal{K}$-assignment ρ is sufficient to interpret a useful sub-language of probabilistic state formulas:

$$\begin{array}{l} \kappa := (a \leq a) \ [\!]\ \mathsf{fff}\ [\!]\ (\kappa \supset \kappa) \\ a := x\ [\!]\ r\ [\!]\ (a + a)\ [\!]\ (aa)\ [\!]\ \widetilde{r}. \end{array}$$

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

2.3 The axiomatization

We need three new concepts for the axiomatization, one of valid state formula, a second one of probabilistic tautology and the third of valid analytical formulas.

Denotation of probability terms

$[\![r]\!]^{\rho}_{(V,\mathcal{K},\mu)}$	$= r$
$[\![y]\!]^{\rho}_{(V,\mathcal{K},\mu)}$	$= \rho(y)$
$[\![(\int\gamma)]\!]^{\rho}_{(V,\mathcal{K},\mu)}$	$= \mu(\lvert\gamma\rvert_V)$
$[\![p_1 + p_2]\!]^{\rho}_{(V,\mathcal{K},\mu)}$	$= [\![p_1]\!]^{\rho}_{(V,\mathcal{K},\mu)} + [\![p_2]\!]^{\rho}_{(V,\mathcal{K},\mu)}$
$[\![p_1 p_2]\!]^{\rho}_{(V,\mathcal{K},\mu)}$	$= [\![p_1]\!]^{\rho}_{(V,\mathcal{K},\mu)} \times [\![p_2]\!]^{\rho}_{(V,\mathcal{K},\mu)}$
$[\![\tilde{r}]\!]^{\rho}_{(V,\mathcal{K},\mu)}$	$= \max(0, \min(r, 1))$

Satisfaction of probabilistic formulas

$(V,\mathcal{K},\mu)\rho \Vdash (\Box\gamma)$	iff $v \Vdash_{\mathsf{C}} \gamma$ for every $v \in V$
$(V,\mathcal{K},\mu)\rho \Vdash (p_1 \leq p_2)$	iff $V \neq \emptyset$ implies $([\![p_1]\!]^{\rho}_{(V,\mathcal{K},\mu)} \leq [\![p_2]\!]^{\rho}_{(V,\mathcal{K},\mu)})$
$(V,\mathcal{K},\mu)\rho \Vdash (\xi/\gamma)$	iff $(\lvert\gamma\rvert_V, \mathcal{K}, \mu_\gamma)\rho \Vdash \xi$
$(V,\mathcal{K},\mu)\rho \Vdash \mathsf{fff}$	iff $V = \emptyset$
$(V,\mathcal{K},\mu)\rho \Vdash (\xi_1 \supset \xi_2)$	iff $(V,\mathcal{K},\mu)\rho \Vdash \xi_2$ or $(V,\mathcal{K},\mu)\rho \nVdash \xi_1$

Table 2. Semantics of EPPL

A classical state formula γ is said to be valid if it is true of all valuations $v \in \mathcal{V}$. As a consequence of the finiteness of D, the set of valid classical state formulas is recursive.

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 is 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 fff, $\rightarrow$ by $\supset$ and $q \in$ 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, if $\mathcal{K}_0$ is the real closed field in a generalized probabilistic structure, then a $\mathcal{K}_0$-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 any $\mathcal{K}$-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 [12, 3] that the set of valid analytical formulas so defined is decidable. We shall not go into details of this result as we want to focus on reasoning about probabilistic aspects only.

The axioms and inference rules of EPPL are listed in Table 3 and better understood in the following groups.

The axiom **CTaut** says that if γ is a valid classical state formula then $(\Box\gamma)$ is an axiom. The axiom **PTaut** says that a probabilistic tautology is an axiom. Since the set of valid classical state 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**$\Rightarrow$, **Eqv**ff and **Ref**$\wedge$ are sufficient to relate (local) classical state reasoning and (global) probabilistic tautological reasoning.

The term $\kappa\{\!|\boldsymbol{y}/\boldsymbol{p}|\!\}$ 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.

Axioms

[CTaut] $\vdash (\Box\gamma)$ for each valid state formula γ

[PTaut] $\vdash \xi$ for each probabilistic tautology ξ

[Lift⇒] $\vdash ((\Box(\gamma_1 \Rightarrow \gamma_2)) \supset (\Box\gamma_1 \supset \Box\gamma_2))$

[Eqvff] $\vdash ((\Box \mathsf{ff}) \approx \mathsf{fff})$

[Ref∧] $\vdash (((\Box\gamma_1) \cap (\Box\gamma_2)) \supset (\Box(\gamma_1 \wedge \gamma_2)))$

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

[Meas∅] $\vdash ((\int \mathsf{ff}) = 0)$

[FAdd] $\vdash (((\int(\gamma_1 \wedge \gamma_2)) = 0) \supset ((\int(\gamma_1 \vee \gamma_2)) = (\int\gamma_1) + (\int\gamma_2)))$

[Mon] $\vdash ((\Box(\gamma_1 \Rightarrow \gamma_2)) \supset ((\int\gamma_1) \leq (\int\gamma_2)))$

[Dist⊃] $\vdash (((\xi_1 \supset \xi_2)/\gamma) \approx ((\xi_1/\gamma) \supset (\xi_2/\gamma)))$

[Elim1] $\vdash (((\Box\gamma_1)/\gamma_2) \approx (\Box(\gamma_2 \Rightarrow \gamma_1)))$

[Elim2] $\vdash (((p_1 \leq p_2)/\gamma) \approx ((\Diamond\gamma) \supset ((p_1 \leq p_2)|^{(\int\gamma_1)}_{(\int(\gamma_1 \wedge \gamma))})))$

Inference rules

[PMP] $\xi_1, (\xi_1 \supset \xi_2) \vdash \xi_2$

[Cond] $\vdash (\xi/\gamma)$ whenever $\vdash \xi$

Table 3. Axioms for EPPL

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 **Dist⊃** says that the connective $\supset$ distributes over the conditional construct. The axioms **Elim1** and **Elim2** eliminate the conditional construct. The probabilistic term

$$(p_1 \leq p_2)|^{(\int\gamma_1)}_{(\int(\gamma_1 \wedge \gamma))}$$

in **Elim2** is the term obtained by replacing <u>all</u> occurrences of $(\int\gamma_1)$ by $(\int(\gamma_1 \wedge \gamma))$ for <u>each</u> classical state formula γ_1.

The inference rule **PMP** is the *modus ponens* for classical and probabilistic implication. The inference rule **Cond** says that if ξ is an theorem. then so is (ξ/γ). The inference rule **Cond** is similar to the generalization rule in modal logics.

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. Please note that while applying the rule **Cond**, we are allowed to use only theorems of the logic (and not any hypothesis or any intermediate step in the derivation).

Every probabilistic formula ξ is equivalent to a probabilistic formula η in which there is no occurrence of a conditional construct:

Lemma 1. Let ξ be an EPPL formula. Then, there is a conditional-free formula η such that $\vdash \xi \approx \eta$. Moreover, there is an algorithm to compute η.

Furthermore, the above set of axioms and rules form a recursive axiomatization:

Theorem 1. EPPL is sound and weakly complete. Moreover, the set of theorems is recursive.

3 Basic probabilistic sequential programs

We shall now describe briefly the syntax and semantics of our basic programs.

3.1 Syntax.

Assuming the syntax of EPPL, the syntax of the programming language in the BNF notation is as follows (with the proviso $r \in \mathcal{R}$):

– $s := \mathsf{skip} \,[]\, \mathsf{xm} \leftarrow t \,[]\, \mathsf{bm} \leftarrow \gamma \,[]\, \mathsf{toss}(\mathsf{bm}, r) \,[]\, s; s \,[]\, \mathsf{bm}\text{–}\mathsf{If}\ \gamma\ \mathsf{then}\ s\ \mathsf{else}\ s.$

The statements $\mathsf{xm} \leftarrow t$ and $\mathsf{bm} \leftarrow \gamma$ are assignments to memory cells xm and bm respectively. For the rest of the paper, by an *expression* we shall mean either the terms t or the classical state formulas γ. Please note that t and γ may contain rigid variables (which may be thought of as input to a program).

The statement $\mathsf{toss}(\mathsf{bm}, r)$ sets bm true with probability $\widetilde{r}$. The command $s; s$ is sequential composition. The statement bm–$\mathsf{If}\ \gamma\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2$ is the bm–marked alternative choice: if γ is true then s_1 is executed and bm is set to true after the execution of s_1 else s_2 is executed and bm is set to false.

3.2 Semantics

The semantics of the programming language is basically the forward semantics in [17] adapted to our programming language. Given $\mathcal{G}$, the set of generalized probabilistic states, the denotation of a program s is a map $[\![s]\!] : \mathcal{G} \to \mathcal{G}$ defined inductively in Table 4. The definition uses the following notations:

– The denotation of a real term t given a valuation v can be extended to classical state formulas as: $[\![\gamma]\!]_v = \mathsf{tt}$ if $v \Vdash_{\mathsf{c}} \gamma$ otherwise $[\![\gamma]\!]_v = \mathsf{ff}$.
– If m is a memory cell (xm or bm) and e is an expression of the same type (t or γ, respectively) then the map $\delta^{\mathsf{m}}_e : \mathcal{V} \to \mathcal{V}$ is defined as $\delta^{\mathsf{m}}_e(v) = v^{\mathsf{m}}_{[\![e]\!]_v}$, where $v^{\mathsf{m}}_{[\![e]\!]_v}$ assigns the value $[\![e]\!]_v$ to the cell m and coincides with v elsewhere. As usual, $(\delta^{\mathsf{m}}_e)^{-1} : \wp\mathcal{V} \to \wp\mathcal{V}$ is defined by taking each set $U \subset \mathcal{V}$ to the set of its pre-images.
– $(V_1, \mathcal{K}, \mu_1) + (V_2, \mathcal{K}, \mu_2) = (V_1 \cup V_2, \mathcal{K}, \mu_1 + \mu_2)$.
– $r(V, \mathcal{K}, \mu) = (V, \mathcal{K}, r\mu)$.

The denotation of classical assignments, sequential composition and marked alternative are as expected. The probabilistic toss $\mathsf{toss}(\mathsf{bm}, r,)$ assigns bm the value tt with probability $\widetilde{r}$ and the value ff with probability $1 - \widetilde{r}$. Therefore, the denotation of the probabilistic toss is the "weighted" sum of the two assignments $\mathsf{bm} \leftarrow \mathsf{tt}$ and $\mathsf{bm} \leftarrow \mathsf{ff}$.

4 Probabilistic Hoare logic

We are ready to define the Hoare logic. As expected, the Hoare assertions are :

– $\delta := \xi \,[]\, \{\xi\}\, s\, \{\xi\}$.

$$
\begin{array}{ll}
[\![\mathsf{skip}]\!] & = \lambda(V,\mathcal{K},\mu).\,(V,\mathcal{K},\mu)\\
[\![\mathsf{xm} \leftarrow t]\!] & = \lambda(V,\mathcal{K},\mu).\,(\delta_t^{\mathsf{xm}}(V),\mathcal{K},\mu\circ(\delta_t^{\mathsf{xm}})^{-1})\\
[\![\mathsf{bm} \leftarrow \gamma]\!] & = \lambda(V,\mathcal{K},\mu).\,(\delta_\gamma^{\mathsf{bm}}(V),\mathcal{K},\mu\circ(\delta_\gamma^{\mathsf{bm}})^{-1})\\
[\![\mathsf{toss}(\mathsf{bm},r)]\!] & = \lambda(V,\mathcal{K},\mu).\,((1-\widetilde{r})\,([\![\mathsf{bm}\leftarrow\mathsf{ff}]\!](V,\mathcal{K},\mu))+\\
 & \qquad\qquad\qquad\qquad \widetilde{r}\,([\![\mathsf{bm}\leftarrow\mathsf{tt}]\!](V,\mathcal{K},\mu)))\\
[\![s_1;s_2]\!] & = \lambda(V,\mathcal{K},\mu).\,[\![s_2]\!]\,([\![s_1]\!](V,\mathcal{K},\mu))\\
[\![\mathsf{bm\text{–}If}\ \gamma\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2]\!] & = \lambda(V,\mathcal{K},\mu).\,([\![s_1;\mathsf{bm}\leftarrow\mathsf{tt}]\!](|\gamma|_V,\mathcal{K},\mu_\gamma)+\\
 & \qquad\qquad\qquad\qquad [\![s_2;\mathsf{bm}\leftarrow\mathsf{ff}]\!](|(\neg\,\gamma)|_V,\mathcal{K},\mu_{(\neg\,\gamma)}))
\end{array}
$$

Table 4. Denotation of programs

Satisfaction of Hoare assertions is defined as

- $(V,\mathcal{K},\mu)\rho \Vdash_h \xi$ if $(V,\mathcal{K},\mu)\rho \Vdash \xi$,
- $(V,\mathcal{K},\mu)\rho \Vdash_h \{\xi_1\}\,s\,\{\xi_2\}$ if $(V,\mathcal{K},\mu)\rho \Vdash \xi_1$ implies $[\![s]\!](V,\mathcal{K},\mu)\rho \Vdash \xi_2$.

Semantic entailment is defined as expected: we say that $\varDelta$ entails δ (written $\varDelta \vDash \delta$) if $(V,\mathcal{K},\mu)\rho \Vdash \delta$ whenever $(V,\mathcal{K},\mu)\rho \Vdash \delta_0$ for each $\delta_0 \in \varDelta$.

4.1 Calculus

A sound Hoare calculus for our probabilistic sequential programs is given in Table 5. In the axioms **ASGR** and **ASGB**, the notation ξ_e^m means the formula obtained from ξ by replacing <u>all</u> occurrences (including those in conditionals and probability terms) of the memory variable m by the expression e. The axioms **TAUT**, **SKIP**, **ASGR** and **ASGB** are similar to the ones in the case of sequential programs.

Axioms

[**TAUT**]	$\vdash \xi$ if ξ is an EPPL theorem
[**SKIP**]	$\vdash \{\xi\}\,\mathsf{skip}\,\{\xi\}$
[**ASGR**]	$\vdash \{\xi_t^{\mathsf{xm}}\}\,\mathsf{xm}\leftarrow t\,\{\xi\}$
[**ASGB**]	$\vdash \{\xi_\gamma^{\mathsf{bm}}\}\,\mathsf{bm}\leftarrow \gamma\,\{\xi\}$
[**TOSS**]	$\vdash \{\eta\,\|_{\Box(\gamma_{\mathsf{ff}}^{\mathsf{bm}}\wedge\gamma_{\mathsf{tt}}^{\mathsf{bm}})}^{\Box\gamma}\,\|_{(1-\widetilde{r})(\int\gamma_{\mathsf{ff}}^{\mathsf{bm}})+\widetilde{r}(\int\gamma_{\mathsf{tt}}^{\mathsf{bm}})}^{(\int\gamma)}\}\ \mathsf{toss}(\mathsf{bm},r)\,\{\eta\}$

Inference rules

[**SEQ**]	$\{\xi_0\}\,s_1\,\{\xi_1\},\{\xi_1\}\,s_2\,\{\xi_2\}$	$\vdash \{\xi_0\}\,s_1;s_2\,\{\xi_2\}$
[**IF**]	$\{\xi_0\}\,s_1;\mathsf{bm}\leftarrow\mathsf{tt}\,\{\xi_2\}$	
	$\{\xi_1\}\,s_2;\mathsf{bm}\leftarrow\mathsf{ff}\,\{\xi_3\}$	$\vdash \{\xi_0 \curlyvee_\gamma \xi_1\}\,\mathsf{bm\text{–}If}\ \gamma\ \mathsf{then}\ s_1\ \mathsf{else}\ s_2\,\{\xi_2 \curlyvee_{\mathsf{bm}} \xi_3\}$
[**CONS**]	$\xi_0 \supset \xi_1, \{\xi_1\}\,s\,\{\xi_2\}, \xi_2 \supset \xi_3$	$\vdash \{\xi_1\}\,s\,\{\xi_3\}$
[**OR**]	$\{\xi_0\}\,s\,\{\xi_2\},\{\xi_1\}\,s\,\{\xi_2\}$	$\vdash \{\xi_0 \cup \xi_1\}\,s\,\{\xi_2\}$
[**AND**]	$\{\xi_0\}\,s\,\{\xi_1\},\{\xi_0\}\,s\,\{\xi_2\}$	$\vdash \{\xi_0\}\,s\,\{\xi_1 \cap \xi_2\}$

Table 5. Hoare calculus

For the axiom **TOSS**, we do not consider arbitrary probabilistic formulas. Instead, we just have probabilistic formulas η which do not have any conditional sub-terms. This is not a serious limitation as every EPPL formula is equivalent to another EPPL formula without conditionals (see Lemma 1). Furthermore, the formula

$$\eta\,|^{\Box\gamma}_{\Box(\gamma^{\mathsf{bm}}_{\mathsf{ff}}\wedge\gamma^{\mathsf{bm}}_{\mathsf{tt}})}|^{(\int\gamma)}_{(1-\widetilde{r})(\int\gamma^{\mathsf{bm}}_{\mathsf{ff}})+\widetilde{r}(\int\gamma^{\mathsf{bm}}_{\mathsf{tt}})}$$

is the formula obtained from η by replacing every occurrence of a necessity formula $(\Box\gamma)$ by $(\Box(\gamma^{\mathsf{bm}}_{\mathsf{ff}} \wedge \gamma^{\mathsf{bm}}_{\mathsf{tt}}))$ and every occurrence of a probability term $(\int\gamma)$ by $(1 - \widetilde{r})(\int\gamma^{\mathsf{bm}}_{\mathsf{ff}}) + \widetilde{r}(\int\gamma^{\mathsf{bm}}_{\mathsf{tt}})$. Here, the formula γ^{bm}_{e} is obtained by replacing $\underline{\text{all}}$ occurrences of bm by e. The soundness of this Hoare rule is a consequence of the following lemma:

Lemma 2 (Substitution lemma for probabilistic tosses). *For any formula η,*

$$([\![\mathsf{toss}(\mathsf{bm}, r)]\!]\,(V, \mathcal{K}, \mu))\rho \Vdash \eta \textit{ iff } (V, \mathcal{K}, \mu)\rho \Vdash \eta\,|^{\Box\gamma}_{\Box(\gamma^{\mathsf{bm}}_{\mathsf{ff}}\wedge\gamma^{\mathsf{bm}}_{\mathsf{tt}})}|^{(\int\gamma)}_{(1-\widetilde{r})(\int\gamma^{\mathsf{bm}}_{\mathsf{ff}})+\widetilde{r}(\int\gamma^{\mathsf{bm}}_{\mathsf{tt}})}.$$

The inference rules **SEQ**, **CONS**, **OR** and **AND** are similar to the ones in sequential programs. For the inference rule **IF**, $\xi \curlyvee_\gamma \xi'$ is an abbreviation for the formula $((\xi/\gamma) \cap (\xi'/\neg\gamma))$. It follows from the definition of semantics of EPPL that $(V, \mathcal{K}, \mu)\rho \Vdash \xi \curlyvee_\gamma \xi'$ if and only if $(|\gamma|_V, \mathcal{K}, \mu_\gamma)\rho \Vdash \xi$ and $(|\neg\gamma|_V, \mathcal{K}, \mu_{\neg\gamma})\rho \Vdash \xi'$. We have:

Theorem 2. The Hoare calculus is sound.

The completeness for the Hoare logic was being worked upon in collaboration with Luís Cruz-Filipe at the time of submission of this paper. The Hoare rule for probabilistic tosses is its weakest pre-condition form as a consequence of the substitution lemma for probabilistic tosses (see Lemma 2 above). For the alternative, we have shown that if ξ_0 and ξ_1 are weakest pre-conditions corresponding to the two marked choices then so is $\xi_0 \curlyvee_\gamma \xi_1$. Furthermore, any EPPL formula ξ is essentially equivalent to a disjunct of formulas of the kind $\xi' \curlyvee_\gamma \xi''$. Roughly, two EPPL formulas are essentially equivalent if the semantic structures satisfying them differ only in the $\mathcal{K}$-assignment part.

5 Example

We illustrate our Hoare calculus on a variation of the 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 [2] 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 bm_k suffices. We restrict our attention to this special case. If the key $\mathsf{bm}_k = 1$ then the qubit is (unitarily) encrypted as the pair $(\beta, -\alpha)$ otherwise it remains the same. The following program S_{qenc} simulates this process by first generating a random key and then encrypting the qubit $(\mathsf{xm}_1, \mathsf{xm}_2)$:

$$\mathsf{toss}(\mathsf{bm}_k, \tfrac{1}{2}); \mathsf{bm}\text{–}\mathsf{If}\ \mathsf{bm}_k\ \mathsf{then}\ \mathsf{Pauli}_{XZ}\ \mathsf{else}\ \mathsf{skip}$$

where Pauli_{XZ} is $\mathsf{xm}_3 \leftarrow \mathsf{xm}_1; \mathsf{xm}_1 \leftarrow \mathsf{xm}_2; \mathsf{xm}_2 \leftarrow -\mathsf{xm}_3$ [3].

[3] The name Pauli_{XZ} has its roots in quantum mechanics.

Assume that the initial values of xm_1 and xm_2 are c_1 and c_2 respectively (with $c_1 \neq c_2$). 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 xm_1 being c_1 is $\frac{1}{2}$ (and hence of xm_1 being c_2 is also $\frac{1}{2}$). We can use our logic to show the above for S_{qenc}. In particular, assuming η_I is $\Box((\mathsf{xm}_1 = c_1) \wedge (\mathsf{xm}_2 = c_2) \wedge (c_1 < c_2))$, we derive the following in our Hoare calculus:

$$\vdash \{((\textstyle\int \mathsf{tt}) = 1) \cap \eta_I\}\, S_{\text{qenc}}\, \{(\textstyle\int(\mathsf{xm}_1 = c_1)) = \frac{1}{2}\}.$$

Abbreviating the statement bm–If bm_k then Pauli_{XZ} else skip as IF, the derivation is:

1	$\{(\int(\mathsf{xm}_1 = c_1)) = \frac{1}{2}\}\, \mathsf{skip}\, \{(\int(\mathsf{xm}_1 = c_1)) = \frac{1}{2}\}$	SKIP
2	$\{(\int(c_2 = c_1)) = 0\}\, \mathsf{Pauli}_{XZ}\, \{(\int(\mathsf{xm}_1 = c_1)) = 0\}$	ASGR, SEQ
3	$(((\int \mathsf{tt}) = \frac{1}{2}) \cap \eta_I) \supset (\int(\mathsf{xm}_1 = c_1)) = \frac{1}{2}$	TAUT
4	$(((\int \mathsf{tt}) = \frac{1}{2}) \cap \eta_I) \supset (\int(c_2 = c_1)) = 0$	TAUT
5	$\{(((\int \mathsf{tt}) = \frac{1}{2}) \cap \eta_I) \curlyvee_{\mathsf{bm}_k} (((\int \mathsf{tt}) = \frac{1}{2}) \cap \eta_I)\}\ \mathsf{IF}$ $\{(\int(\mathsf{xm}_1 = c_1)) = \frac{1}{2} \curlyvee_{\mathsf{bm}} (\int(\mathsf{xm}_1 = c_1)) = 0\}$	IF, CONS 1,2,3,4
6	$((\int(\mathsf{xm}_1 = c_1)) = \frac{1}{2} \curlyvee_{\mathsf{bm}} (\int(\mathsf{xm}_1 = c_1)) = 0) \supset (\int(\mathsf{xm}_1 = c_1)) = \frac{1}{2}$	TAUT
7	$(\eta_I \cap ((\int \mathsf{bm}) = \frac{1}{2}) \cap ((\int \neg\, \mathsf{bm}) = \frac{1}{2}))\supset$ $(((\int \mathsf{tt}) = \frac{1}{2}) \cap \eta_I) \curlyvee_{\mathsf{bm}_k} (((\int \mathsf{tt}) = \frac{1}{2}) \cap \eta_I)$	TAUT
8	$\{(\eta_I \cap ((\int \mathsf{bm}) = \frac{1}{2}) \cap ((\int \neg\, \mathsf{bm}) = \frac{1}{2}))\}\ \mathsf{IF}\ \{(\int(\mathsf{xm}_1 = c_1)) = \frac{1}{2}\}$	CONS 5,6,7
9	$\{((\int \mathsf{tt}) = 1) \cap \eta_I\}\{\mathsf{toss}(\mathsf{bm}, \frac{1}{2})\}$ $\{(\eta_I \cap ((\int \mathsf{bm}) = \frac{1}{2}) \cap ((\int \neg\, \mathsf{bm}) = \frac{1}{2}))\}$	TOSS, TAUT
10	$\{((\int \mathsf{tt}) = 1) \cap \eta_I\}\, S_{\text{qenc}}\, \{(\int(\mathsf{xm}_1 = c_1)) = \frac{1}{2}\}$	SEQ 8,9.

6 Related Work

The area of formal methods in probabilistic programs has attracted a lot of work ranging from semantics [16, 15, 29, 22] to logic-based reasoning [9, 17, 27, 10, 13, 21, 23, 6].

Our work is in the field of probabilistic dynamic logics. Dynamic logic is a modal logic in which the modalities are of the form $\langle s \rangle \varphi$ where s is a program and φ is a state assertion formula. For probabilistic programs, there are two distinct approaches to dynamic logic. The main difference in the two approaches is that one uses truth-functional state logic while the other one uses state logic with arithmetical connectives.

The first truth-functional probabilistic state logic based works appear in the context of dynamic logic [28, 18, 26, 9, 8]. In the context of probabilistic truth-functional dynamic logics, the state language has terms representing probabilities (*e.g.*, $(\int \gamma)$ represents the probability of γ being true). An infinitary complete axiom system for probabilistic dynamic logic is given in [18]. Later, a complete finitary axiomatization of probabilistic dynamic logic was given in [9]. However, the state logic is second-order (to deal with iteration) and no axiomatization of the state logic is achieved. In [8], decidability of a less expressive dynamic logic is achieved.

Hoare logic can be viewed as a fragment of dynamic logic and the first probabilistic Hoare logic with truth-functional propositional state logic appears in [27]. However, as discussed in Section 1, even simple assertions in this logic may not be provable. For instance, the valid Hoare assertion (adapting somewhat the syntax)

$$\{(\int \mathsf{tt}) = 1\}\ \mathsf{If}\ x = 0\ \mathsf{then}\ \mathsf{skip}\ \mathsf{else}\ \mathsf{skip}\ \{(\int \mathsf{tt}) = 1\}$$

is not provable in the logic. As noted in [27, 17], the reason for incompleteness is the Hoare rule for the alternative if-then-else which tries to combine absolute information of the two alternatives truth-functionally. The Hoare logic in [6] circumvents the problem of the alternative by defining the probabilistic sum connective as already discussed in Section 1. Although this logic is more expressive than the one in [27] and completeness is achieved for a fragment of the Hoare logic, it is not clear how to axiomatize the probabilistic sum connective [6].

The other approach to dynamic logic uses arithmetical state logic instead of truth-functional state logic [17, 15, 14, 21]. For example, instead of the if-then-else construct the programming language in [17] has the construct $\gamma?s_1 + (\neg\gamma)?s_2$ which is closely bound to the forward denotational semantics proposed in [16]. This leads to a probabilistic dynamic logic in which measurable functions are used as state formulas and the connectives are interpreted as arithmetical operations.

In the context of Hoare logics, the approach of arithmetical connectives is the one that has attracted more research. The Hoare triple in this context naturally leads to the definition of *weakest pre-condition* for a measurable function g and a program s: the weakest pre-condition $\mathsf{wp}(g, s)$ is the function that has the greatest expected value amongst all functions f such that $\{f\}\, s\, \{g\}$ is a valid Hoare triple. The weakest pre-condition can thus be thought of as a backward semantics which transforms a post-state g in the context of a program s to a pre-state $\mathsf{wp}(g, s)$. The important result in this area is the duality between the forward semantics and the backwards semantics [14].

Later, [21] extended this framework to address non-determinism and proved the duality between forward semantics and backward semantics. Instead of just using functions f and g as pre-conditions and post-conditions, [21] also allows a rudimentary state language with basic classical state formulas α, negation, disjunction and conjunction. The classical state formula α is interpreted as the function that takes the value 1 in the memory valuations where α is true and 0 otherwise. Conjunction and disjunction are interpreted as minimum and maximum respectively, and negation as subtraction from the constant function 1. For example, the following Hoare assertion is valid in this logic:

$$\{r\}\ \mathsf{toss}(\mathsf{bm}, r)\ \{\mathsf{bm}\}.$$

Here r in the pre-condition is the constant function r and bm is the function that take value 1 when bm is true and 0 otherwise. The validity of the above Hoare assertion says that the probability of bm being true after the probabilistic toss is at least r.

We tackle the problem of alternative if-then-else by marking the choices at the end of the execution and by introducing the conditional construct (ξ/γ) in the state logic. The state logic itself is the probabilistic logic in [20] extended with the conditional construct. The logic is designed by the exogenous semantics approach to probabilistic logics [24, 25, 7, 1, 20]. The main difference from the logic in [20] is that the state logic herein has the conditional construct which is not present in [20]. The axioms **Dist**$\supset$, **Elim1** and **Elim2** are used to deal with this conditional construct. Using these, we can demonstrate that every formula is equivalent to another formula without conditionals and the proof of completeness then follows the lines of the proof in [20]. The other difference is that the probabilities in [20] are taken in the set of real numbers and terms contain real computable numbers. The proof of completeness is obtained relative to an (undecidable) oracle for reasoning about reals.

Finally, one main contribution of our paper is the Hoare rule in the weakest pre-condition form for probabilistic toss in the context of truth-functional state logic. The Hoare rule for probabilistic tosses does appear in the context of arithmetical Hoare logics and takes the form

$$\mathsf{wp}(\mathsf{toss}(\mathsf{bm}, r), \alpha) = r \times \mathsf{wp}(\mathsf{bm} \leftarrow \mathsf{tt}, \alpha) + (1 - r) \times \mathsf{wp}(\mathsf{bm} \leftarrow \mathsf{ff}, \alpha).$$

7 Conclusions and Future Work

Our main contribution is a sound probabilistic Hoare calculus with a truth-functional state assertion logic that enjoys recursive axiomatization. The Hoare rule for the if-then-else statement avoids the probabilistic sum construct in [6] by marking the choices taken and by taking advantage of a conditional construct in the state assertion language. Another important contribution is the axiom for probabilistic toss which gives the weakest pre-condition in truth-functional setting and is the counterpart of the weakest pre-condition for probabilistic toss in Hoare logics with arithmetical state logics.

As discussed in Section 4, we are currently working towards complete axiomatization for the Hoare-calculus for the iteration free language. We plan to include the iteration construct and demonic non-determinsim in future work. For iteration, we will investigate completeness using an oracle for arithmetical reasoning.

Our long-term interests are in reasoning about quantum programs and protocols. Probabilities are inevitable in quantum programs because measurements of quantum states yield probabilistic mixtures of quantum states. We aim to investigate Hoare-style reasoning and dynamic logics for quantum programming. Towards this end, we have already designed logics for reasoning about individual quantum states [19, 5] and a sound Hoare logic for basic quantum imperative programs [4].

References

1. M. Abadi and J. Y. Halpern. Decidability and expressiveness for first-order logics of probability. *Information and Computation*, 112(1):1–36, 1994.
2. A. Ambainis, 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*, page 547. IEEE Computer Society, 2000.
3. S. Basu, R. Pollack, and R. Marie-Françoise. *Algorithms in Real Algebraic Geometry*. Springer, 2003.
4. R. Chadha, P. Mateus, and A. Sernadas. Reasoning about quantum imperative programs. *Electronic Notes in Theoretical Computer Science*, 158:19–40, 2006. Invited talk at the Twenty-second Conference on the Mathematical Foundations of Programming Semantics.
5. R. Chadha, P. Mateus, A. Sernadas, and C. Sernadas. Extending classical logic for reasoning about quantum systems. Preprint, CLC, Department of Mathematics, Instituto Superior Técnico, 2005. Invited submission to the Handbook of Quantum Logic.

6. J.I. den Hartog and E.P. de Vink. Verifying probabilistic programs using a hoare like logic. *International Journal of Foundations of Computer Science*, 13(3):315–340, 2002.
7. R. Fagin, J. Y. Halpern, and N. Megiddo. A logic for reasoning about probabilities. *Information and Computation*, 87(1-2):78–128, 1990.
8. Y. A. Feldman. A decidable propositional dynamic logic with explicit probabilities. *Information and Control*, 63((1/2)):11–38, 1984.
9. Y. A. Feldman and David Harel. A probabilistic dynamic logic. *Journal of Computer and System Sciences*, 28:193–215, 1984.
10. H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6:512–535, 1995.
11. C. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12:576–583, 1969.
12. W. Hodges. *Model Theory*. Cambridge University Press, 1993.
13. M. Huth and M. Kwiatkowska. Quantitative analysis and model checking. In *12th Annual IEEE Symposium on Logic in Computer Science (LICS'97)*, pages 111–122, 1997.
14. C. Jones. *Probabilistic Non-determinism*. PhD thesis, U. Edinburgh, 1990.
15. C. Jones and G. D. Plotkin. A probabilistic powerdomain of evaluations. In *Proceedings of the Fourth Annual Symposium on Logic in Computer Science*, pages 186–195. IEEE Computer Society, 1989.
16. D. Kozen. Semantics of probabilistic programs. *Journal of Computer System Science*, 22:328–350, 1981.
17. D. Kozen. A probabilistic PDL. *Journal of Computer System Science*, 30:162–178, 1985.
18. J.A. Makowsky and M.L.Tiomkin. Probabilistic propositional dynamic logic, 1980. manuscript.
19. P. Mateus and A. Sernadas. Weakly complete axiomatization of exogenous quantum propositional logic. *Information and Computation*, to appear.
20. P. Mateus, A. Sernadas, and C. Sernadas. Exogenous semantics approach to enriching logics. In *Essays on the Foundations of Mathematics and Logic*, volume 1 of *Advanced Studies in Mathematics and Logic*, pages 165–194. Polimetrica, 2005.
21. C. Morgan, A. McIver, and K. Seidel. Probabilistic predicate transformers. *ACM Transactions on Programming Languages and Systems*, 18(3):325–353, 1996.
22. M. A. Moshier and A. Jung. A logic for probabilities in semantics. In *Computer Science Logic*, volume 2471 of *Lecture Notes in Computer Science*, pages 216–231. Springer Verlag, 2002.
23. M. Narasimha, R. Cleaveland, and P. Iyer. Probabilistic temporal logics via the modal mucalculus. In *Foundations of Software Science and Computation Structures (FOSSACS 99)*, volume 1578 of *Lecture Notes in Computer Science*, pages 288–305. 1999.
24. N. J. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28(1):71–87, 1986.
25. N. J. Nilsson. Probabilistic logic revisited. *Artificial Intelligence*, 59(1-2):39–42, 1993.
26. R. Parikh and A. Mahoney. A theory of probabilistic programs. In *Proceedings of the Carnegie Mellon Workshop on Logic of Programs*, volume 64 of *Lecture Notes in Computer Science*, pages 396–402. Springer-Verlag, 1983.
27. L. H. Ramshaw. *Formalizing the analysis of algorithms.* PhD thesis, Stanford University, 1979.
28. J. H. Reif. Logics for probabilistic programming (extended abstract). In *STOC '80: Proceedings of the twelfth annual ACM symposium on Theory of computing*, pages 8–13, 1980.
29. R. Tix, K. Keimel, and G.D. Plotkin. Semantic domains for combining probability and nondeterminism. *Electronic Notes in Theoretical Computer Science*, 129:1–104, 2005.

A EPPL – soundness, weak completeness and decidability

We briefly outline the proof of soundness, weak completeness and decidability of EPPL.

Lemma 3 (Principle of substitution of equivalent formulas). Given three formulas ξ, ξ_1 and ξ_2, let ξ' be obtained from ξ by replacing zero or more occurrences of ξ_1 in ξ by ξ_2. Then, $\vdash (\xi \approx \xi')$ whenever $\vdash (\xi_1 \approx \xi_2)$.

Proof. Since all (global) tautological formulas are theorems in EPPL, the only interesting case is when $\xi \overset{\text{def}}{=} (\xi_1/\gamma)$. In this case, if $\vdash (\xi_1 \approx \xi_2)$ then $\vdash (\xi_1 \supset \xi_2)$. Thus, by rule **Cond**, $\vdash ((\xi_1 \supset \xi_2)/\gamma)$, and moreover, by rule **Dist**$\supset$, $\vdash ((\xi_1/\gamma) \supset (\xi_2/\gamma))$. Similarly $\vdash ((\xi_2/\gamma) \supset (\xi_1/\gamma))$, which concludes the proof. □

Lemma 4. Let η be an EPPL formula without conditionals. Then for every γ, there exists another conditional-free formula η' such that $\vdash \eta/\gamma \approx \eta'$. Moreover, there is an algorithm to compute η'.

Proof. The proof follows by induction on the structure of η. Basis: (i) $\eta \overset{\text{def}}{=} \Box\gamma_1$ - the result follows by **Elim1**; (ii) $\eta \overset{\text{def}}{=} (p_1 \leq p_2)$ - the result yields by **Elim2**; (iii) $\eta \overset{\text{def}}{=} \mathsf{fff}$. The axiom **Eqv**ff says that $\vdash \mathsf{fff} \approx (\Box\mathsf{ff})$. The result now follows from part (i) and Lemma 3.

Induction step: $\eta \overset{\text{def}}{=} (\eta_1 \supset \eta_2)$ - by **Dist**$\supset$, $\vdash ((\eta_1 \supset \eta_2)/\gamma) \approx ((\eta_1/\gamma) \supset (\eta_2/\gamma))$. By induction hypothesis, $\vdash (\eta_1/\gamma) \approx \eta_1'$ and $\vdash (\eta_2/\gamma) \approx \eta_2'$ where both η_1' and η_2' have no conditional operators. Finally, by Lemma 3, $\vdash ((\eta_1 \supset \eta_2)/\gamma) \approx (\eta_1' \supset \eta_2')$. Moreover, the inductive proof above induces a recursive algorithm to obtain η' from η. □

We are ready to show that EPPL is complete and decidable. First, we have:

Lemma 1. Let ξ be an EPPL formula. Then, there is a conditional-free formula η such that $\vdash \xi \approx \eta$. Moreover, there is an algorithm to compute η.

Proof. The proof is by induction using Lemma 4. The inductive proof also provides a recursive algorithm.

Theorem 1. The proof system of EPPL is sound and weakly complete. Moreover, the set of theorems of EPPL is recursive.

Proof.
Soundness - Straightforward induction in the length of a EPPL derivation. All the axioms are valid formulas, and the rules are sound.

Weak completeness and decidability - The central result is to show that if ξ is consistent (that is, $\not\vdash (\ominus\,\xi)$) then there is a model $(V, \mathcal{K}, \mu)\rho$ such that $(V, \mathcal{K}, \mu)\rho \Vdash \xi$. The decidability follows by showing that the consistency of a formula is decidable. The proofs of model existence and decidability of consistency go hand-in-hand.

The proof follows the steps in [19, 5] restricted to the probabilistic sub-language. The main difference is that the language presented in Table 1 enriches that of [19, 5] with the conditional operator. Note that, due to Lemma 1, if ξ is consistent then there is a conditonal-free consistent η equivalent to ξ. Moreover, η can be computed from ξ. For this reason, we restrict our attention to formulas η which have no conditional operators. The proof in [19, 5] can now be adapted to EPPL and is summarized as follows: (i) η has an equivalent probabilistic disjunctive normal form computable from η; (ii) η is consistent iff one of its conjunctive molecules of its DNF is consistent, and therefore one can work only with this probabilistic conjunctive molecule; (iii) there exists a formula η' that is maximal with respect to admissible valuation and which can be computed from η (global Henkin construction); moreover, one can effectively obtain a (finite) component V of the model from η' ; (iv) the probabilistic terms $(\int\gamma)$ of a molecule η' can be effectively replaced by sums $\sum_{v\Vdash_{\mathsf{C}}\gamma, v\in V} y_v$ resulting in another formula η'' where y_v represents the probability of the valuation v and hence η is consistent iff η'' is; (v) finally, one can obtain the remaining parts of the model by effectively solving (in real closed fields) a system of in-equations induced by η''. □

B Soundness of Hoare Logic

The proof of soundness relies on substitution lemma for probabilistic states and probabilistic tosses. They are proved using the substitution lemma for classical valuations:

Lemma 5 (Substitution Lemma for classical valuations). For any valuation $v \in \mathcal{V}$, any classical state formula γ, any memory cell m (xm or bm) and a term e of the same type (t or γ', respectively):

$$v^m_{[\![e]\!]_v} \Vdash_{\mathsf{C}} \gamma \text{ iff } v \Vdash_{\mathsf{C}} \gamma^m_e .$$

Proof. The proof is standard and is similar to the ones for deterministic sequential programs. □

Recall that $\delta^m_e : \mathcal{V} \to \mathcal{V}$ is the map that takes each valuation v to $v^m_{[\![e]\!]_v}$.

Lemma 6 (Substitution Lemma for probabilistic states). Let $(V, \mathcal{K}, \mu)$ be a generalized probabilistic structure and ρ a $\mathcal{K}$-assignment. Given a memory cell m and a term e of the same type, let $(V', \mathcal{K}, \mu') = (\delta^m_e(V), \mathcal{K}, \mu \circ (\delta^m_e)^{-1})$. Then for any classical state formula γ:

1. $(V', \mathcal{K}, \mu')\rho \Vdash (\Box\gamma)$ iff $(V, \mathcal{K}, \mu)\rho \Vdash (\Box\gamma^m_e)$, and
2. $[\![(\int\gamma)]\!]^\rho_{(V', \mathcal{K}, \mu')} = [\![(\int\gamma^m_e)]\!]^\rho_{(V, \mathcal{K}, \mu)}$.

Furthermore, for any formula ξ, $(V', \mathcal{K}, \mu')\rho \Vdash \xi$ iff $(V, \mathcal{K}, \mu)\rho \Vdash \xi^m_e$.

Proof. Please note that as a consequence of Lemma 5, we have

$$|\gamma|_{\delta^m_e(V)} = \delta^m_e(|\gamma^m_e|_V) \text{ and } (\delta^m_e)^{-1}(|\gamma|_{\delta^m_e(V)}) = (|\gamma^m_e|_V).$$

The two parts of the lemma then follow by definition. Using this and induction, we can show that ξ, $(V', \mathcal{K}, \mu')\rho \Vdash \xi$ iff $(V, \mathcal{K}, \mu)\rho \Vdash \xi^m_e$ for any ξ. In the case of conditionals, we have to use the above observation again. □

Lemma 2 (Substitution lemma for probabilistic tosses). Let $(V, \mathcal{K}, \mu)$ be a generalized probabilistic structure and ρ a $\mathcal{K}$-assignment. Let

$$\begin{aligned}(V_1, \mathcal{K}, \mu_1) &= (\delta^{\mathsf{bm}}_{\mathsf{tt}}(V), \mathcal{K}, \mu \circ (\delta^{\mathsf{bm}}_{\mathsf{tt}})^{-1}),\\ (V_2, \mathcal{K}, \mu_2) &= (\delta^{\mathsf{bm}}_{\mathsf{ff}}(V), \mathcal{K}, \mu \circ (\delta^{\mathsf{bm}}_{\mathsf{ff}})^{-1}), \mathit{and}\\ (V', \mathcal{K}, \mu') &= \widetilde{r}(V_1, \mathcal{K}, \mu_1) + (1 - \widetilde{r})(V_2, \mathcal{K}, \mu_2).\end{aligned}$$

Then for any classical state formula γ:

1. $(V', \mathcal{K}, \mu')\rho \Vdash (\Box\gamma)$ iff $(V, \mathcal{K}, \mu)\rho \Vdash \Box(\gamma^{\mathsf{bm}}_{\mathsf{tt}} \wedge \gamma^{\mathsf{bm}}_{\mathsf{ff}})$, and
2. $[\![(\int\gamma)]\!]^{\rho}_{(V',\mathcal{K},\mu')} = \widetilde{r}[\![(\int\gamma^{\mathsf{bm}}_{\mathsf{tt}})]\!]^{\rho}_{(V,\mathcal{K},\mu)} + (1 - \widetilde{r})[\![(\int\gamma^{\mathsf{bm}}_{\mathsf{ff}})]\!]^{\rho}_{(V,\mathcal{K},\mu)}$.

Furthermore, for any conditional free formula η,

$$(V', \mathcal{K}, \mu')\rho \Vdash \eta \text{ iff } (V, \mathcal{K}, \mu)\rho \Vdash \eta\,|^{\Box\gamma}_{\Box(\gamma^{\mathsf{bm}}_{\mathsf{ff}} \wedge \gamma^{\mathsf{bm}}_{\mathsf{tt}})}|^{(\int\gamma)}_{(1-\widetilde{r})(\int\gamma^{\mathsf{bm}}_{\mathsf{ff}})+\widetilde{r}(\int\gamma^{\mathsf{bm}}_{\mathsf{tt}})}.$$

Proof. Please observe that by Lemma 5, we have for any $v \in \mathcal{V}$:

$$v \Vdash_{\mathsf{C}} \gamma^{\mathsf{bm}}_{\mathsf{tt}} \wedge \gamma^{\mathsf{bm}}_{\mathsf{ff}} \text{ iff } v^{\mathsf{bm}}_{\mathsf{tt}} \Vdash_{\mathsf{C}} \gamma \text{ and } v^{\mathsf{bm}}_{\mathsf{ff}} \Vdash_{\mathsf{C}} \gamma.$$

The first part then follows by definition. For the second part, by definition, we have:

$$[\![(\int\gamma)]\!]^{\rho}_{(V',\mathcal{K},\mu')} = \widetilde{r}[\![(\int\gamma)]\!]^{\rho}_{(V_1,\mathcal{K},\mu_1)} + (1 - \widetilde{r})[\![(\int\gamma)]\!]^{\rho}_{(V_2,\mathcal{K},\mu_2)}.$$

The second part then follows from the second part of Lemma 6. Finally, the claim for the conditional-free η can thus be proved using induction on the structure of η. $\Box$

Theorem 2. The Hoare-calculus is sound.

Proof. The proof is by cases on the axioms and inference rules. For assignments to memory variables, we use Lemma 6 and for probabilistic assignments we use Lemma 2. For the alternative construct we divide our generalized state (including the measure) into two parts, one where the guard is true and the other where it is false.

C Classical one-time pad

We demonstrated the use of our Hoare logic for proving the correctness of a variation of quantum one-time pad in Section5. Here we demonstrate the correctness of classical one-time pad.

One-time pad is a provably secure way of encrypting a bit-string. Given a plain-text message m and a key k of same length, the cipher-text c is computed as bitwise xor of m and k. We can prove the security of the one-time pad in our calculus. The following program S_{enc}, for instance, generates a random 1-bit key bm_k and encrypts the 1-bit plain-text bm_p:

$$\mathsf{toss}(\mathsf{bm}_k, \tfrac{1}{2});\ \mathsf{bm}_c \leftarrow \neg(\mathsf{bm}_k \Leftrightarrow \mathsf{bm}_p).$$

The following Hoare assertion states the security of the one-time pad (the probability of the cipher-text xm_c being tt is $\frac{1}{2}$ regardless of the probability distribution on the possible values of the plain-text xm_p):

$$\{(\textstyle\int\mathsf{tt}) = 1\}\, S_{\text{enc}}\, \{(\textstyle\int\mathsf{bm}_c) = \tfrac{1}{2}\}.$$

The pre-condition $(\int \mathsf{tt}) = 1$ means that the total measure is 1. We can derive the above in our Hoare calculus:

1	$((\int \mathsf{tt}) = 1) \supset (\frac{1}{2}(\int \neg(\mathsf{tt} \Leftrightarrow \mathsf{bm}_p)) + \frac{1}{2}(\int \neg(\mathsf{ff} \Leftrightarrow \mathsf{bm}_p)) = \frac{1}{2})$	TAUT
2	$\{\frac{1}{2}(\int \neg(\mathsf{tt} \Leftrightarrow \mathsf{bm}_p)) + \frac{1}{2}(\int \neg(\mathsf{ff} \Leftrightarrow \mathsf{bm}_p)) = \frac{1}{2}\}$ $\mathsf{toss}(\mathsf{bm}_k, \frac{1}{2})\, \{(\int \neg(\mathsf{bm}_k \Leftrightarrow \mathsf{bm}_p)) = \frac{1}{2}\}$	TOSS
3	$\{(\int \mathsf{tt}) = 1\}\, \mathsf{toss}(\mathsf{bm}_k, \frac{1}{2})\, \{(\int \neg(\mathsf{bm}_k \Leftrightarrow \mathsf{bm}_p)) = \frac{1}{2}\}$	CONS 1,2
4	$\{(\int \neg(\mathsf{bm}_k \Leftrightarrow \mathsf{bm}_p)) = \frac{1}{2}\}\, \mathsf{bm}_c \leftarrow \neg(\mathsf{bm}_k \Leftrightarrow \mathsf{bm}_p)\, \{(\int \mathsf{bm}_c) = \frac{1}{2}\}$	ASGB
5	$\{(\int \mathsf{tt}) = 1\}\, S\text{enc}\, \{(\int \mathsf{bm}_c) = \frac{1}{2}\}$	SEQ 3,4.