

Reasoning about probabilistic sequential programs¹

R. Chadha^{b,4,2}, L. Cruz-Filipe^{a,3}, P. Mateus^{a,*}, A. Sernadas^a

^a*SQIG – IT and IST, Portugal*

^b*Department of Computer Science, University of Illinois at Urbana–Champaign*

Abstract

A complete and decidable Hoare-style calculus for iteration-free probabilistic sequential programs is presented using a state logic with truth-functional propositional (not arithmetical) connectives.

Key words: Probabilistic reasoning; Hoare calculus; Completeness; Decidability.

1 Introduction

Reasoning about probabilistic systems is very important due to applications in randomized algorithms, security, reliability, distributed systems and, more recently, quantum computation and information. Logics supporting such reasoning have branched in two main directions. Firstly, Hoare-style [31,25,10] and dynamic logics [13,21] have been developed building upon denotational semantics of probabilistic programs [20]. The second approach enriches temporal modalities with probabilistic bounds [14,17,27].

* Corresponding author.

Email addresses: rch@cs.uiuc.edu (R. Chadha), lcf@math.ist.utl.pt (L. Cruz-Filipe), pmat@math.ist.utl.pt (P. Mateus), acs@math.ist.utl.pt (A. Sernadas).

¹ Supported by FCT and FEDER through POCTI via CLC and project QuantLog POCI/MAT/55796/2004

² Supported by FCT and FEDER grant SFRH/BPD/26137/2005

³ Supported by FCT and FEDER grant SFRH/BPD/16372/2004

⁴ Work done while at IST

Our work is in the area of Hoare-style reasoning about probabilistic sequential programs. A Hoare assertion [15] is a triple of the form $\{\eta_1\} s \{\eta_2\}$ meaning that: if program s starts in state satisfying the state assertion formula η_1 , and if s halts, then s ends in a state satisfying the state transition formula η_2 . Formula η_1 is known as the pre-condition and formula η_2 as the post-condition of the Hoare assertion. For probabilistic programs, the development of Hoare logic has taken primarily two distinct paths. The common denominator of the two approaches is forward denotational semantics of sequential probabilistic programs [20]: 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 [31] using a *truth-functional* state assertion language, *i.e.*, the formulas of the logic are interpreted as either true or false, and the truth value of a formulas is determined by the truth values of the sub-formulas. This state assertion language consists of two levels: i) classical state formulas γ interpreted over the valuations of memory cells; (ii) probabilistic state formulas η interpreted over (sub)-probability measures of the valuations. The latter contains terms of the form $(f\gamma)$ representing probability of γ being true. But the language at the probabilistic level is extremely restrictive, and it 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 [31], as observed in [31,21], 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 [21] 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 [21], there are several important works in the probabilistic Hoare logic, *e.g.* [18,25], where the state formulas are either measurable functions or arithmetical formulas interpreted as measurable functions. Intuitively, the Hoare triple $\{f\} s \{g\}$ means that the expected value of the function g after the execution of s is at least as much as the expected value of the function f before the execution.

Although research in probabilistic 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 [10], and completeness for a fragment of the Hoare-logic was shown for iteration-free programs. In order to deal with alternative composition, a test construct $\mathbf{bm}?\eta$ and a probabilistic sum construct $(\eta_1 + \eta_2)$ was introduced. Intuitively, the formula $\gamma?\eta$ is satisfied by a (sub)-probability measure μ on valuations over memory cell if $\mu(v)$ is non-zero only when the valuation v satisfies γ and there is a valuation μ' such that $\mu + \mu'$ satisfies η . The formula $(\eta_1 + \eta_2)$ is satisfied by a (sub)-probability measure μ if μ can be written as the sum of two measures μ_1 and μ_2 that satisfy η_1 and η_2 respectively. The drawback of [10] is that no axiomatization is given for the state assertion logic. The choice construct and the probabilistic sum constructs are the essential obstacles in achieving a complete axiomatization for the state language.

The gap between [31] and [10] was addressed in [8], which provides a sound Hoare logic with a complete and decidable state assertion logic. The Hoare rule for alternative construct is tackled using two key ingredients. First, the usual if-then-else construct is slightly modified: a boolean memory variable \mathbf{bm} is marked with the choice taken at the end of the conditional branch. This modification gives 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. Secondly, the state assertion language has a conditional construct (η/γ) . Intuitively, the formula (η/γ) is satisfied by a (sub)-probability measure μ if η is true of the (sub)-probability measure obtained by eliminating the measure of all valuations where γ is false. The conditional formulas (η/\mathbf{bm}) and $(\eta/(\neg \mathbf{bm}))$ in the state logic can then be used to combine information of the alternative paths. Another feature of the state language in [8] is that a distinction is maintained between possibility and probability, yielding a more expressive state language. The completeness of the Hoare logic was left as an open question.

This paper addresses the gap between [31] and [10,8], providing a complete and decidable Hoare logic for iteration-free probabilistic programs using a complete and decidable truth-functional probabilistic state assertion logic. The Hoare calculus provided herein was arrived at while attempting to prove the completeness of the Hoare logic proposed in [8].

Our probabilistic state assertion logic, henceforth referred to as Exogenous Probabilistic Propositional Logic (EPPL), is essentially the logic of [11], designed by taking the exogenous semantics approach [11,1,24] to enriching a given logic – the models of the enriched logic are sets of models of the given logic with some additional structure. A semantic model of EPPL is a discrete (sub)-probability space that gives the probability of each possible valuation. For the sake of convenience, we work with finitely additive, discrete and bounded measures and not just (sub)-probability measures. In order to achieve a recursive axiomatization for EPPL, it is also convenient to 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 [16,4], and this technique of achieving decidability is detailed in other works on probabilistic reasoning [11,1]. The exogenous approach to probabilistic logics first appeared in [28,29] and later in [11,1]. The general exogenous mechanism for building new logics is described in detail in [24,6] and used for developing quantum logics in [23,7].

As in [31], there are two levels of formulas in EPPL: classical state formulas γ , interpreted over individual valuations, and probabilistic state formulas η , interpreted over the models of EPPL. Terms p in the language at the probabilistic level represent elements of a real closed field and the probability of γ being true is represented by the term $(f\gamma)$. Probabilistic state formulas are built from probabilistic atoms $p_1 \leq p_2$, meaning that the term p_1 is less than or equal to the term p_2 , using the disjunctive connectives fff and \supset .

The essential difference from our state assertion logic and the logic in [11] is that our terms p contain variables that are interpreted over elements in the real closed field. In order to interpret these variables, our semantic structures also contain an *assignment*. We do not allow quantification over these variables, allowing us to maintain the propositional nature of the state assertion language. The other advantage is that the complexity of the known decision procedures of quantifier-free formulas interpreted over real closed fields is simpler than the complexity of the known decision procedures of the full first-order language [16,4]. As we shall see, variables are crucial in the proof of completeness of the Hoare logic. They are used in the Hoare logic to keep track of individual contributions of the alternate choices to the measure terms $(f\gamma)$. A second difference is that we also allow products in terms. The logic in [11] does not have general product terms and allows only products with constants, mainly for complexity considerations.

The programming language we consider is a basic imperative language with assignment to memory variables, sequential composition, probabilistic assignment and alternative choice. The probabilistic assignment $\text{toss}(\mathbf{bm}, r)$ assigns \mathbf{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. The alternative choice construct described here is the standard if-then-else construct and not the modified *marked* if-then-else proposed in [8]. It turns out that the variables in the state language are sufficient to keep track of individual contributions of the alternate choices and it is not necessary to mark the choices explicitly to achieve completeness. Another difference between our work and the work in [8] is that we do not distinguish between possibility and probability. This is in accordance with standard works on probabilistic programs, and it simplifies the proof of completeness of Hoare logic. The obvious disadvantage

of this decision is that we lose expressiveness.

The completeness and decidability of the proposed Hoare calculus for reasoning about iteration-free probabilistic programs is achieved using the standard technique. First, we define a *weakest precondition* operator $\mathbf{wp}(\cdot, \cdot)$ assigning to each program s and each formula η a new state formula $\mathbf{wp}(s, \eta)$ corresponding to the weakest logical property that a state must satisfy to ensure that η holds after execution of s . The weakest precondition operator is defined in terms of an auxiliary *preterm* operator $\mathbf{pt}(\cdot, \cdot)$ assigning to each program s and each formula p a new state formula $\mathbf{pt}(s, p)$ such that the denotation of $\mathbf{pt}(s, p)$ before execution of s is the same as the denotation of p after the execution of s regardless of initial state. The weakest precondition $\mathbf{wp}(s, \eta)$ is then built by replacing each term p in η by the preterm $\mathbf{pt}(s, p)$.

We then show that, for any program s and formula η , the Hoare calculus derives the judgment $\{\mathbf{wp}(s, \eta)\} s \{\eta\}$; in other words, $\mathbf{wp}(s, \eta)$ is a sufficient precondition for s and η . The proof of completeness concludes after showing that $(V, \mathcal{K}, \mu)\rho \Vdash \mathbf{wp}(s, \eta)$ iff $\llbracket s \rrbracket(V, \mathcal{K}, \mu)\rho \Vdash \eta$. The decidability of EPPL combined with the fact that weakest precondition can be built algorithmically gives decidability of the Hoare logic.

The rest of the paper is organized as follows. The syntax, semantics and the complete recursive axiomatization of EPPL are presented in Section 2. The programming language is introduced in Section 3 and the sound Hoare logic in Section 4. The proofs of completeness and decidability of the Hoare calculus are given in Section 6. We finish by presenting two examples illustrating the Hoare calculus and the generated weakest pre-conditions in Section 7. We discuss related work in Section 8 and summarize the results and future work in Section 9.

Acknowledgements. We would like to thank Peter Selinger and Michael Ben-Or for useful and interesting discussions.

2 Logic of probabilistic states: EPPL

The state logic presented herein is the probability logic proposed in [11] extended with variables that assist in the proof of completeness of the Hoare calculus. In our probabilistic programs, we work with 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. Note that, instead of reals, we could have also

used any type with finite range.

Any run of a program probabilistically assigns values to these registers. 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 is a finitely additive, discrete and bounded measure μ on $\wp\mathcal{V}$, the power-set of \mathcal{V} ; in other words, μ is a map from $\wp\mathcal{V}$ to \mathbb{R}^+ (the set of non-negative real numbers) such that:

- $\mu(\emptyset) = 0$;
- $\mu(U_1 \cup U_2) = \mu(U_1) + \mu(U_2)$ 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 probability measures as it makes the semantics simpler.

Furthermore, it is convenient to assume that the measures take values from an arbitrary *real closed field* instead of the set of real numbers. An ordered field $\mathcal{K} = (K, +, \cdot, 1, 0, \leq)$ 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 ;
- every 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. Another example is the set of computable real numbers with the same operations. 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 copies of the integers and the rationals. In addition, in a real closed field we can take roots of positive elements and odd n -roots. In general, any real algebraic number is definable in a real closed field. The set of real algebraic numbers shall be denoted by \mathcal{A} ; we shall use these numbers as constants in probability terms of our logic.

A semantic structure of EPPL consists of 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 of EPPL consists of formulas at two levels. The formulas of the first level – *classical state formulas* – allow us to reason about individual valuations over the memory cells. The formulas of the second level – *probabilistic state formulas* – allow us to 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 $c \in D$)
$t := \text{xm} \mid X \mid c \mid (t + t) \mid (tt)$
Classical state formulas
$\gamma := \text{bm} \mid B \mid (t \leq t) \mid \text{ff} \mid (\gamma \Rightarrow \gamma)$
Probability terms (with the proviso $r \in \mathcal{A}$)
$p := y \mid r \mid (f\gamma) \mid (p + p) \mid (pp) \mid \tilde{r}$
Probabilistic state formulae:
$\eta := (p \leq p) \mid \text{fff} \mid (\eta \supset \eta)$

Table 1
Language of EPPL

Given a fixed $\mathbf{m} = \{0, \dots, m-1\}$, there are two finite disjoint sets of memory variables: $\text{xM} = \{\text{xm}_k : k \in \mathbf{m}\}$, representing the contents of real registers, and $\text{bM} = \{\text{bm}_k : k \in \mathbf{m}\}$, representing the contents of boolean registers. We also have two sets of (rigid over time and random) logical variables which are useful in parametric reasoning about programs: $\mathbf{B} = \{B_k : k \in \mathbb{N}\}$, ranging over the truth values in $2 = \{\text{ff}, \text{tt}\}$, and $\mathbf{X} = \{X_k : k \in \mathbb{N}\}$, ranging over elements of D . At the end of Subsection 2.2 we will show that the special case in which these random variables behave deterministically except on a set of measure zero can be expressed in the logic. Therefore, we can use these variables as deterministic parameters in applications. On the other hand, the randomness of these variables allow us to have random initial states, which is useful for compositional reasoning about programs; furthermore, it simplifies the theory.

The real terms, ranged over by t, t_1, \dots , are built from the sets D , xM and

X using the usual addition and multiplication⁵. The classical state formulas, ranged over by γ, γ_1, \dots , are built from \mathbf{bM} , \mathbf{B} and comparison formulas ($p_1 \leq p_2$) using the classical disjunctive connectives \mathbf{ff} and \Rightarrow . As usual, other classical connectives ($\neg, \vee, \wedge, \Leftrightarrow, \mathbf{tt}$) are introduced as abbreviations. For instance, $(\neg \gamma)$ stands for $(\gamma \Rightarrow \mathbf{ff})$.

The probability terms, ranged over by p, p_1, \dots , denote elements of the real closed field in a semantic structure. We also assume a set of (rigid and deterministic) logical variables, $Y = \{y_k : k \in \mathbb{N}\}$, ranging over elements of the real closed field. These logical variables, which were not present in [11], are essential in our proof of completeness of the Hoare logic.

The probability terms also contain real algebraic numbers as constants. The denotation of the probability term \tilde{r} is r if $0 \leq r \leq 1$, 0 if $r \leq 0$ and 1 otherwise. The probability term $(f\gamma)$ denotes the measure of the set of valuations that satisfy γ . The terms of the kind $(f\gamma)$ shall henceforth be called *measure terms*. We denote the set of all probability terms by \mathbf{PTerms} .

The probabilistic state formulas, ranged over by η, η_1, \dots , are built from comparison formulas ($p_1 \leq p_2$) using the connectives \mathbf{fff} and \supset . Other probabilistic connectives ($\ominus, \cup, \cap, \approx, \mathbf{ttt}$) and comparison operators ($=, \geq, <, >$) are introduced as abbreviations in the classical way. For instance, $(\ominus \eta)$ stands for $(\eta \supset \mathbf{fff})$ and $(p_1 = p_2)$ stands for $((p_1 \leq p_2) \cap (p_2 \leq p_1))$. We denote the set of all probabilistic state formulas by \mathbf{PForms} .

It is also convenient for applications to introduce as an abbreviation the formula $(\square \gamma)$ which stands for the formula $((f\gamma) = (f\mathbf{tt}))$. Intuitively, $\square \gamma$ is satisfied if the set of the valuations where γ does not hold has measure zero. We shall also use $(\diamond \gamma)$ as an abbreviation for $(\ominus(\square(\neg \gamma)))$. Intuitively, $(\diamond \gamma)$ is satisfied if the set of valuations where γ holds has non-zero measure. We shall see in Section 2.2 that \square and \diamond behave somewhat as necessity and possibility modalities. However, \square and \diamond are not full fledged modalities, since they cannot be nested⁶.

The notion of occurrence of a term p and a probabilistic state formula η_1 in the probabilistic state formula η is defined as usual. The same holds for the notion of replacing zero or more occurrences of probability terms and probabilistic formulas. The set of variables $y \in Y$ occurring in a term p and a formula η will be denoted by $\mathbf{PVar}(p)$ and $\mathbf{PVar}(\eta)$. For the sake of clarity, we shall often drop parentheses in formulas and terms if it does not lead to ambiguity.

⁵ The arithmetical operations addition and multiplication are assumed to be defined so as to restrict them to the range D . This is satisfied if we assume D to be closed under them.

⁶ We do not have formulas such as $\square(\square \gamma)$.

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

$$\begin{aligned}\kappa &:= (a \leq a) \parallel \text{fff} \parallel (\kappa \supset \kappa) \\ a &:= x \parallel r \parallel (a + a) \parallel (aa) \parallel \tilde{r}\end{aligned}$$

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

2.2 Semantics

A *valuation* is a map $v : (\mathbf{xM} \rightarrow D, \mathbf{bM} \rightarrow 2, \mathbf{X} \rightarrow D, \mathbf{B} \rightarrow 2)$ that provides values to the memory variables and corresponding logical variables. The set of all possible valuations is denoted by \mathcal{V} . Given a valuation v , the denotation of real terms $\llbracket t \rrbracket_v$, and satisfaction of classical state formulas $v \Vdash_{\mathbf{C}} \gamma$ are defined inductively as expected. Given $V \subseteq \mathcal{V}$, the *extent* of γ in V is defined as $|\gamma|_V = \{v \in V : v \Vdash_{\mathbf{C}} \gamma\}$.

A *generalized probabilistic state* is a pair (\mathcal{K}, μ) where \mathcal{K} a real closed field and μ is a finitely additive, discrete and finite \mathcal{K} -measure over $\wp\mathcal{V}$. The set of all generalized states is denoted by \mathcal{G} .

Given a classical formula γ we also need the sub-measure of μ defined by

$$\mu_\gamma = \lambda V. \mu(|\gamma|_V).$$

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

To interpret the probabilistic variables $y \in \mathbf{Y}$, we need the concept of assignment. Given a real closed field \mathcal{K} , a *\mathcal{K} -assignment* is a map $\rho : \mathbf{Y} \rightarrow \mathcal{K}$.

Given a generalized state (K, μ) and a \mathcal{K} -assignment ρ , the denotation of probabilistic terms and satisfaction of probabilistic state formulas are defined inductively in Table 2. The formula $(p_1 \leq p_2)$ is satisfied if the term denoted by p_1 is less than or equal to p_2 . The formula $(\eta_1 \supset \eta_2)$ is satisfied by a semantic model if either η_1 is not satisfied by the model or η_2 is satisfied by the model. Observe that the probabilistic connectives behave like the classical ones. Also, the \mathcal{K} -assignment ρ is sufficient to interpret an analytical formula, *i.e.*, a probabilistic formula without measure terms.

Entailment is defined as usual: Λ entails η (written $\Lambda \vDash \eta$) if $(K, \mu)\rho \Vdash \eta$ whenever $(K, \mu)\rho \Vdash \eta_0$ for each $\eta_0 \in \Lambda$. The meta-theorem of entailment holds: $\Lambda, \eta \vDash \eta'$ iff $\Lambda \vDash (\eta \supset \eta')$.

Denotation of probability terms

$$\begin{aligned}
\llbracket r \rrbracket_{(K,\mu)}^\rho &= r \\
\llbracket y \rrbracket_{(K,\mu)}^\rho &= \rho(y) \\
\llbracket (\int \gamma) \rrbracket_{(K,\mu)}^\rho &= \mu(|\gamma|_{\mathcal{V}}) \\
\llbracket p_1 + p_2 \rrbracket_{(K,\mu)}^\rho &= \llbracket p_1 \rrbracket_{(K,\mu)}^\rho + \llbracket p_2 \rrbracket_{(K,\mu)}^\rho \\
\llbracket p_1 p_2 \rrbracket_{(K,\mu)}^\rho &= \llbracket p_1 \rrbracket_{(K,\mu)}^\rho \times \llbracket p_2 \rrbracket_{(K,\mu)}^\rho
\end{aligned}$$

Satisfaction of probabilistic formulas

$$\begin{aligned}
(K, \mu)\rho \Vdash (p_1 \leq p_2) &\text{ iff } (\llbracket p_1 \rrbracket_{(K,\mu)}^\rho \leq \llbracket p_2 \rrbracket_{(K,\mu)}^\rho) \\
(K, \mu)\rho \not\Vdash \text{fff} & \\
(K, \mu)\rho \Vdash (\eta_1 \supset \eta_2) &\text{ iff } (K, \mu)\rho \Vdash \eta_2 \text{ or } (K, \mu)\rho \not\Vdash \eta_1
\end{aligned}$$

Table 2
Semantics of EPPL

We can also define the probabilistic sum construct similar to the one defined in [10] by saying that $(K, \mu)\rho \Vdash \eta_1 + \eta_2$ if there exist μ_1 and μ_2 such that $\mu = \mu_1 + \mu_2$, $(K, \mu_1)\rho \Vdash \eta_1$ and $(K, \mu_2)\rho \Vdash \eta_2$. However, as already observed in Section 1, it is not obvious how to axiomatize this construction.

Recall the derived formula $(\Box\gamma)$ defined above. Clearly, $(K, \mu)\rho \Vdash (\Box\gamma)$ iff $\mu(|\gamma|_{\mathcal{V}}) = \mu(\mathcal{V})$ iff $\mu(|\neg\gamma|_{\mathcal{V}}) = 0$. Similarly, $(K, \mu)\rho \Vdash (\Diamond\gamma)$ iff $\mu(|\gamma|_{\mathcal{V}}) > 0$.

It follows easily from the semantics that $\models (\Box(\gamma_1 \wedge \gamma_2)) \approx ((\Box\gamma_1) \wedge (\Box\gamma_2))$. Hence, $(\Box\gamma)$ behaves as necessity modality. Similarly, $(\Diamond\gamma)$ behaves as possibility modality, *i.e.*, $\models (\Diamond(\gamma_1 \vee \gamma_2)) \approx ((\Diamond\gamma_1) \vee (\Diamond\gamma_2))$. However, it is not the case that $\models ((\Box\gamma) \supset (\Diamond\gamma))$. Consider a generalized probabilistic state (K, μ) where μ is identically zero; then $(K, \mu) \Vdash \Box\gamma$ for all classical state formulas γ , but $(K, \mu) \not\Vdash \Diamond\gamma$ holds for none of them.

Returning to the random nature of our logical variables in \mathbf{X} and \mathbf{B} , observe that we can impose that they behave deterministically except with zero probability. For instance, the formula $\bigcup_{c \in D} (\Box(X_k = c))$ constrains X_k to have a fixed value except with measure zero. Clearly, this is possible because both our data types are finite.

2.3 The axiomatization

We need three new concepts for the axiomatization: that of valid state formula, that of probabilistic tautology and that of valid analytical formula.

A classical state formula γ is said to be *valid* if it holds for 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 η is said to be a *probabilistic tautology* if there exist a propositional tautology β over Q and a map σ from Q to the set of probabilistic state formulas such that η 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 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 κ is satisfied by ρ for any real closed field \mathcal{K} and any \mathcal{K} -assignment ρ . Clearly, a valid analytical formula holds in all semantic structures of EPPL. It is a well-known fact from the theory of quantifier elimination [16,4] 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.

Axiom **CTaut** says that if γ is a valid classical state formula then $(\Box\gamma)$ is an axiom. 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 term $\kappa_{\vec{p}}^{\vec{y}}$ in axiom **RCF** is the term obtained by substituting *all* occurrences of y_i in κ by the probability term p_i . Axiom **RCF** says that if κ is a valid analytical formula, then any formula obtained by replacing variables by probability terms is a tautology. Again, we refrain from spelling out the details as the set of valid analytical formulas is recursive.

Axiom **Meas \emptyset** states simply that the measure of empty set is 0, while axiom **FAdd** expresses finite additivity of measures. Finally, axiom **Mon** relates the classical connectives with probability measures and is a consequence of monotonicity of measures.

The inference rule **PMP** is the *modus ponens* for classical and probabilistic implication.

As usual we say that a set of formulas Λ *derives* η , written $\Lambda \vdash \eta$, if we can

Axioms

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

[**PTaut**] $\vdash \eta$ for each probabilistic tautology η

[**RCF**] $\vdash \kappa_{\vec{p}}^{\vec{y}}$ for any valid analytical formula κ and sequences of probability variables and probability terms \vec{y} and \vec{p} , respectively

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

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

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

Inference rule

[**PMP**] $\eta_1, (\eta_1 \supset \eta_2) \vdash \eta_2$

Table 3

Axioms for EPPL

build a derivation of η from axioms and the inference rules using formulas in Λ as hypotheses. It can be easily shown that the meta-theorem of deduction holds, that is, $\Lambda, \eta_1 \vdash \eta_2$ iff $\Lambda \vdash (\eta_1 \supset \eta_2)$.

Throughout this paper, we shall only be concerned with judgments of the form $\Lambda \vdash \eta$ where Λ is a finite set. Since both meta-theorems of entailment and deduction hold in EPPL, it suffices to consider judgments where Λ is empty.

The soundness of the axiom system is a consequence of the definition.

Theorem 2.1 The axiom system of EPPL is sound, *i.e.*, if $\vdash \eta$ then $\models \eta$.

Proof. The validity of the axioms and the inference rule **PMP** follow from the definition of the semantics. \square

The proofs of completeness and decidability of EPPL go hand-in-hand and essentially follows the lines of the proof of completeness in [11,24]. The main ingredient is the model existence lemma: if a probabilistic formula η is consistent, *i.e.* $\not\vdash (\ominus \eta)$, then there is a model that satisfies η . Furthermore, there is an algorithm that decides the consistency of a probabilistic formula. We give a sketch of the proof and refer the reader to [11] for details.

Theorem 2.2 The proof system of EPPL is weakly complete, *i.e.*, if $\models \eta$ then $\vdash \eta$. Moreover, the set of theorems of EPPL is recursive.

Proof sketch. The central result is to show that if η is consistent (that is, $\not\vdash (\ominus \eta)$) then there is a model $(\mathcal{K}, \mu)\rho$ such that $(\mathcal{K}, \mu)\rho \models \eta$. The decidability follows by showing that the consistency of a formula is decidable.

The proof in [11,24] adapted to EPPL is summarized as follows: (i) compute the (finite) set of valuations over the memory cells and the logical variables in the sets \mathbf{B} and \mathbf{X} occurring in η and let this set of valuations be V ; (ii) let κ_1 be the analytical formula obtained from η by effectively replacing measure terms $(f\gamma)$ by sums $\sum_{v \models \mathcal{C}\gamma, v \in V} y_v$ where y_v represents the probability of the valuation v ; (iii) let κ be the analytical formula $\bigwedge_{y_v, v \in V} (0 \leq y_v)$; (iv) η is consistent iff κ is; (v) finally, consistency of κ is decided by the axiom **RCF** and the model is constructed for a consistent κ by solving for y_v in real closed fields. \square

As there are algorithms for deciding the consistency of analytical formulas, the best-known of which are described in [4], we can get an idea of the complexity of the SAT procedure proposed above. The best algorithm is exponential in the number of variables. This construction generates a variable y_v for each valuation v , hence the total number of variables will be exponential on the number of propositional symbols (since there is an exponential number of valuations). Hence, the algorithm will be doubly exponential on the number of propositional symbols and exponential on the number of EPPL variables in a given formula.

3 Basic probabilistic sequential programs

We now describe the syntax and semantics of our programming language.

3.1 Syntax

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

$$s := \text{skip} \mid \text{xm} \leftarrow t \mid \text{bm} \leftarrow \gamma \mid \text{toss}(\text{bm}, r) \mid s; s \mid \text{if } \gamma \text{ then } s \text{ else } s$$

The statement **skip** does nothing. The statement $\text{xm} \leftarrow t$ assigns to the memory cell **xm** the value denoted by t , and the statement $\text{bm} \leftarrow \gamma$ assigns to the cell **bm** the truth value of γ . For the rest of the paper, by *expression* we shall mean either a term t or a classical state formula γ . Note that both t and γ

may contain variables in the set X (which may be thought of as input to a program).

The statement $\text{toss}(\mathbf{bm}, r)$ sets \mathbf{bm} to true with probability \tilde{r} . Sequential composition of commands is written $s; s$. The statement $\text{if } \gamma \text{ then } s_1 \text{ else } s_2$ is alternative choice: if γ is true then s_1 is executed, else s_2 is executed.

Bounded iteration may be introduced as an abbreviation. Given $k \in \mathbb{N}$, one may define $(\text{while}^k \gamma \text{ do } s)$ as $(\text{if } \gamma \text{ then } s \text{ else skip})^k$.

3.2 Semantics

The semantics of the programming language is basically the forward semantics in [21] adapted to our programming language. Given \mathcal{G} , the set of generalized probabilistic states, the denotation of a program s is a map $\llbracket s \rrbracket : \mathcal{G} \rightarrow \mathcal{G}$ 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 $\llbracket \gamma \rrbracket_v = \mathbf{tt}$ if $v \Vdash_{\mathbf{C}} \gamma$ and $\llbracket \gamma \rrbracket_v = \mathbf{ff}$ otherwise.
- If \mathbf{m} is a memory cell (\mathbf{xm} or \mathbf{bm}) and e is an expression of the same type (t or γ , respectively), then the map $\delta_e^{\mathbf{m}} : \mathcal{V} \rightarrow \mathcal{V}$ is defined as $\delta_e^{\mathbf{m}}(v) = v_{\llbracket e \rrbracket_v}^{\mathbf{m}}$ where $v_{\llbracket e \rrbracket_v}^{\mathbf{m}}$ assigns the value $\llbracket e \rrbracket_v$ to the cell \mathbf{m} and coincides with v elsewhere. As usual, $(\delta_e^{\mathbf{m}})^{-1} : \wp \mathcal{V} \rightarrow \wp \mathcal{V}$ is defined by taking each set $U \subset \mathcal{V}$ to the set of its pre-images.
- $(\mathcal{K}, \mu_1) + (\mathcal{K}, \mu_2) = (\mathcal{K}, \mu_1 + \mu_2)$.
- $r(\mathcal{K}, \mu) = (\mathcal{K}, r\mu)$.

The denotation of classical assignments and sequential composition are as expected. The probabilistic toss $\text{toss}(\mathbf{bm}, r)$ assigns to \mathbf{bm} the value \mathbf{tt} with probability \tilde{r} and the value \mathbf{ff} with probability $1 - \tilde{r}$; therefore, the denotation of the probabilistic toss is the “weighted” sum of the two assignments $\mathbf{bm} \leftarrow \mathbf{tt}$ and $\mathbf{bm} \leftarrow \mathbf{ff}$. The denotation of the alternative composition is as expected: s_1 is executed in the states where γ is true and s_2 is executed in the states where γ is false. It can be easily shown that any probabilistic program preserves the total measure, *i.e.*, if $\llbracket s \rrbracket(\mathcal{K}, \mu) = (\mathcal{K}', \mu')$ then $\mu(\mathcal{V}) = \mu'(\mathcal{V})$.

4 Probabilistic Hoare logic

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

$$\Psi := \eta \parallel \{ \eta \} s \{ \eta \}.$$

$\llbracket \text{skip} \rrbracket$	$= \lambda(\mathcal{K}, \mu). (\mathcal{K}, \mu)$
$\llbracket \text{xm} \leftarrow t \rrbracket$	$= \lambda(\mathcal{K}, \mu). (\mathcal{K}, \mu \circ (\delta_t^{\text{xm}})^{-1})$
$\llbracket \text{bm} \leftarrow \gamma \rrbracket$	$= \lambda(\mathcal{K}, \mu). (\mathcal{K}, \mu \circ (\delta_\gamma^{\text{bm}})^{-1})$
$\llbracket \text{toss}(\text{bm}, r) \rrbracket$	$= \lambda(\mathcal{K}, \mu). (\tilde{r}(\llbracket \text{bm} \leftarrow \text{tt} \rrbracket(\mathcal{K}, \mu)) +$ $(1 - \tilde{r})(\llbracket \text{bm} \leftarrow \text{ff} \rrbracket(\mathcal{K}, \mu)))$
$\llbracket s_1; s_2 \rrbracket$	$= \lambda(\mathcal{K}, \mu). \llbracket s_2 \rrbracket(\llbracket s_1 \rrbracket(\mathcal{K}, \mu))$
$\llbracket \text{if } \gamma \text{ then } s_1 \text{ else } s_2 \rrbracket$	$= \lambda(\mathcal{K}, \mu). (\llbracket s_1 \rrbracket(\mathcal{K}, \mu_\gamma) + \llbracket s_2 \rrbracket(\mathcal{K}, \mu_{(\neg\gamma)}))$

Table 4

Denotation of programs

Satisfaction of Hoare assertions is defined as follows.

- $(\mathcal{K}, \mu)\rho \Vdash_h \eta$ if $(\mathcal{K}, \mu)\rho \Vdash \eta$;
- $(\mathcal{K}, \mu)\rho \Vdash_h \{ \eta_1 \} s \{ \eta_2 \}$ if $(\mathcal{K}, \mu)\rho \Vdash \eta_2$ whenever $\llbracket s \rrbracket(\mathcal{K}, \mu)\rho \Vdash \eta_1$.

We say that a Hoare assertion Ψ is *semantically valid*, and write $\models \Psi$, if $(\mathcal{K}, \mu)\rho \Vdash_h \Psi$ for every generalized probabilistic state (\mathcal{K}, μ) and any \mathcal{K} -assignment ρ .

4.1 Calculus

We shall now give a sound and complete axiomatization of the Hoare calculus. We will only consider judgments of the form $\vdash \Psi$, *i.e.*, judgments with no hypotheses. Hence, in all inference rules the premises are assumed to be theorems of the Hoare calculus. We need some new concepts for the axiomatization: *tossed terms*, *tossed formulas*, *conditional terms* and *conditional formulas*.

Given a memory cell bm , a constant $r \in \mathcal{A}$ and a probabilistic term $p \in \text{PTerms}$, we define the (bm, r) -*tossed term* $\text{toss}(\text{bm}, r; p)$ to be the term obtained from p by replacing *every* occurrence of *each* measure term $(f\gamma)$ by $\tilde{r}(f\gamma_{\text{tt}}^{\text{bm}}) + (1 - \tilde{r})(f\gamma_{\text{ff}}^{\text{bm}})$, where the formula γ_e^{bm} is obtained from γ by replacing *all* occurrences of bm by e . Similarly, we define the probabilistic formula $\text{toss}(\text{bm}, r; \eta)$ to be the formula obtained from η by replacing *every* occurrence of *each* measure term $(f\gamma)$ by $\tilde{r}(f\gamma_{\text{tt}}^{\text{bm}}) + (1 - \tilde{r})(f\gamma_{\text{ff}}^{\text{bm}})$. Formally, $\text{toss}(\text{bm}, r; \cdot)$ can be defined recursively on the set of probabilistic terms PTerms and the set of probabilistic formulas PForms . The recursive definition is given in Table 5. Note that this recursive definition also gives a recursive algorithm for computing $\text{toss}(\text{bm}, r; p)$ and $\text{toss}(\text{bm}, r; \eta)$.

Given a classical state formula γ and a probabilistic term $p \in \text{PTerms}$, we define the γ -*conditioned term* (p/γ) to be the term obtained from p by re-

Tossed terms

$$\begin{aligned}
\text{toss}(\text{bm}, r; r') &= r' \\
\text{toss}(\text{bm}, r; y) &= y \\
\text{toss}(\text{bm}, r; (f\gamma)) &= (\tilde{r}(f\gamma_{\text{tt}}^{\text{bm}}) + (1 - \tilde{r})(f\gamma_{\text{ff}}^{\text{bm}})) \\
\text{toss}(\text{bm}, r; (p + p')) &= (\text{toss}(\text{bm}, r; p) + \text{toss}(\text{bm}, r; p')) \\
\text{toss}(\text{bm}, r; (pp')) &= (\text{toss}(\text{bm}, r; p) \text{toss}(\text{bm}, r; p'))
\end{aligned}$$

Tossed formulas

$$\begin{aligned}
\text{toss}(\text{bm}, r; \text{fff}) &= \text{fff} \\
\text{toss}(\text{bm}, r; (p \leq p')) &= (\text{toss}(\text{bm}, r; p) \leq \text{toss}(\text{bm}, r; p')) \\
\text{toss}(\text{bm}, r; (\eta \supset \eta')) &= (\text{toss}(\text{bm}, r; \eta) \supset \text{toss}(\text{bm}, r; \eta'))
\end{aligned}$$

Table 5

Tossed terms and formulas

placing *every* occurrence of *each* measure term $(f\gamma')$ by $(f(\gamma' \wedge \gamma))$. Similarly, we define the probabilistic formula η/γ to be the formula obtained from η by replacing *every* occurrence of *each* measure term $(f\gamma')$ by $(f(\gamma' \wedge \gamma))$. The recursive definition of $(\cdot)/\gamma$ is given in Table 6. Again, this recursive definition gives a recursive algorithm for computing p/γ and η/γ . Given two probabilistic formulas η_1 and η_2 , we shall use $(\eta_1 \Upsilon_\gamma \eta_2)$ as an abbreviation for $((\eta_1/\gamma) \cap (\eta_2/(\neg\gamma)))$.

A sound and complete Hoare calculus for our probabilistic sequential programs is given in Table 7. The axioms **TAUT** and **SKIP** and the inference rules **SEQ**, **CONS**, **OR** and **AND** are similar to the ones in the case of deterministic sequential programs. The others are briefly discussed below.

Recall that an analytical formula is a probabilistic formula that does not contain any measure terms (terms of the kind $(f\gamma)$). Since an analytical formula

Conditional terms

$$\begin{aligned}
r/\gamma &= r \\
y/\gamma &= y \\
(f\gamma')/\gamma &= (f(\gamma \wedge \gamma')) \\
(p + p')/\gamma &= (p/\gamma + p'/\gamma) \\
(pp')/\gamma &= ((p/\gamma)(p'/\gamma))
\end{aligned}$$

Conditional formulas

$$\begin{aligned}
\text{fff}/\gamma &= \text{fff} \\
(p \leq p')/\gamma &= (p/\gamma \leq p'/\gamma) \\
(\eta \supset \eta')/\gamma &= (\eta/\gamma \supset \eta'/\gamma)
\end{aligned}$$

Table 6

Conditional terms and formulas

does not contain any memory cells, a execution of a program does not change the truth value of an an analytical formula κ . This fact is reflected in the axiom $f\mathbf{FREE}$. (Actually, this axiom is only needed in the case s is an alternative statement. It can be derived in other cases by induction.)

In the axioms \mathbf{ASGR} and \mathbf{ASGB} , the notation η_e^m stands for the formula obtained from η by replacing *all* occurrences of the memory variable m by the expression e . The axioms \mathbf{ASGR} and \mathbf{ASGB} are analogous to the Hoare rules for assignment in the case of deterministic sequential programs. The axiom \mathbf{TOSS} covers the case of probabilistic tosses.

Axioms

$[\mathbf{TAUT}]$	$\vdash \eta$ if η is an EPPL theorem
$[f\mathbf{FREE}]$	$\vdash \{\kappa\} s \{\kappa\}$ if κ is an analytical formula
$[\mathbf{SKIP}]$	$\vdash \{\eta\} \text{skip} \{\eta\}$
$[\mathbf{ASGR}]$	$\vdash \{\eta_t^{xm}\} xm \leftarrow t \{\eta\}$
$[\mathbf{ASGB}]$	$\vdash \{\eta_\gamma^{bm}\} bm \leftarrow \gamma \{\eta\}$
$[\mathbf{TOSS}]$	$\vdash \{\text{toss}(bm, \eta; r)\} \text{toss}(bm, r) \{\eta\}$

Inference rules

$[\mathbf{SEQ}]$	$\{\eta_0\} s_1 \{\eta_1\}, \{\eta_1\} s_2 \{\eta_2\}$	$\vdash \{\eta_0\} s_1; s_2 \{\eta_2\}$
$[\mathbf{IF}]$	$\{\eta_1\} s_1 \{y_1 = (f\gamma_0)\}, \{\eta_2\} s_2 \{y_2 = (f\gamma_0)\}$	$\vdash \{\eta_1 \vee_\gamma \eta_2\} \text{if } \gamma \text{ then } s_1 \text{ else } s_2 \{y_1 + y_2 = (f\gamma_0)\}$
$[\mathbf{ELIMV}]$	$\{\eta \cap (y = p)\} s \{\eta\}$	$\vdash \{\eta_p^y\} s \{\eta\}$ if $y \notin \text{PVar}(p) \cup \text{PVar}(\eta)$
$[\mathbf{CONS}]$	$\eta_0 \supset \eta_1, \{\eta_1\} s \{\eta_2\}, \eta_2 \supset \eta_3$	$\vdash \{\eta_0\} s \{\eta_3\}$
$[\mathbf{OR}]$	$\{\eta_0\} s \{\eta_2\}, \{\eta_1\} s \{\eta_2\}$	$\vdash \{\eta_0 \cup \eta_1\} s \{\eta_2\}$
$[\mathbf{AND}]$	$\{\eta_0\} s \{\eta_1\}, \{\eta_0\} s \{\eta_2\}$	$\vdash \{\eta_0\} s \{\eta_1 \cap \eta_2\}$

Table 7

Hoare calculus

For the inference rule \mathbf{IF} , recall that $\eta_1 \vee_{\gamma_0} \eta_2$ is an abbreviation for the formula $((\eta_1/\gamma_0) \cap (\eta_2/(\neg\gamma_0)))$. This inference rule keeps track of $(f\gamma)$, the measure of γ . The variables y_1 and y_2 account for the contributions to $(f\gamma)$ from the alternative branches s_1 and s_2 , respectively. Although this rule might seem a bit restrictive, it is sufficient to guarantee the completeness of the Hoare calculus along with the axiom $f\mathbf{FREE}$ and the inference rule \mathbf{ELIMV} .

The inference rule \mathbf{ELIMV} eliminates variables in the set \mathbf{Y} . In this rule, η

cannot have any conditional constructs and the variable y does not occur in either the probabilistic term p or the post-condition η . This inference rule is essential for proving the completeness of Hoare logic and is not present in [8]. It can be viewed as a special case of the inference rule for existential quantifiers in first-order Hoare logic, which is often stated as

$$\{\varphi\} s \{\psi\} \vdash \{(\exists z. \varphi)\} s \{\psi\} \text{ if } z \text{ does not occur in } \psi.$$

The inference rule **ELIMV** can then be viewed as a special instance of this rule by observing that the first-order formula $(\exists z. (\varphi(z) \wedge (z = r)))$ is equivalent to $(\exists z. (\varphi(r) \wedge (z = r)))$, which in turn is equivalent to $\varphi(r)$ if z does not occur in r .

5 Soundness of the Hoare logic

We now show that the Hoare calculus presented in Section 4 is sound, *i.e.*, if $\vdash \Psi$ then $\Vdash_h \Psi$. It is sufficient to show that all the axioms and inference rules of the Hoare calculus are sound.

The proofs of soundness of the axioms **ASGB** and **ASGR** rely on the substitution lemma for classical valuations. This situation is similar to the one in deterministic sequential programs, where the key ingredient for the soundness of the axiom for assignments is also a substitution lemma. Recall that the valuation $v_{[[e]]_v}^m$ assigns the value $[[e]]_v$ to the cell m and coincides with the valuation v elsewhere.

Lemma 5.1 (Substitution Lemma for classical valuations) For every valuation $v \in \mathcal{V}$, classical state formula γ , memory cell m (xm or bm) and term e of the same type (t or γ' , respectively),

$$v_{[[e]]_v}^m \Vdash_{\mathbf{C}} \gamma \text{ iff } v \Vdash_{\mathbf{C}} \gamma_e^m.$$

Proof. The proof is by induction on the structure of γ and is similar to the one for deterministic sequential programs. \square

We now extend the substitution lemma for classical valuations to a substitution lemma for probabilistic terms and formulas, which will imply the soundness of **ASGB** and **ASGR**. Recall that $\delta_e^m : \mathcal{V} \rightarrow \mathcal{V}$ is the map that takes each valuation v to $v_{[[e]]_v}^m$.

Lemma 5.2 (Substitution Lemma for assignment) Let (\mathcal{K}, μ) be a generalized probabilistic structure and ρ be a \mathcal{K} -assignment. Given a memory cell

m and a term e of the same type, let $\mu' = \mu \circ (\delta_e^m)^{-1}$. Then

$$\llbracket (f\gamma) \rrbracket_{(\mathcal{K}, \mu')}^\rho = \llbracket (f\gamma_e^m) \rrbracket_{(\mathcal{K}, \mu)}^\rho$$

for any classical state formula γ . Furthermore, for any probabilistic term p ,

$$\llbracket p \rrbracket_{(\mathcal{K}, \mu')}^\rho = \llbracket p_e^m \rrbracket_{(\mathcal{K}, \mu)}^\rho,$$

and, for any probabilistic formula η ,

$$(\mathcal{K}, \mu')\rho \Vdash \eta \text{ iff } (\mathcal{K}, \mu)\rho \Vdash \eta_e^m.$$

Proof. As a consequence of Lemma 5.1,

$$(\delta_e^m)^{-1}(|\gamma|_{\mathcal{V}}) = |\gamma_e^m|_{\mathcal{V}} \text{ and hence } \mu((\delta_e^m)^{-1}(|\gamma|_{\mathcal{V}})) = \mu(|\gamma_e^m|_{\mathcal{V}}).$$

Therefore, by definition,

$$\llbracket (f\gamma) \rrbracket_{(\mathcal{K}, \mu')}^\rho = \mu \circ (\delta_e^m)^{-1}(|\gamma|_{\mathcal{V}}) = \mu(|\gamma_e^m|_{\mathcal{V}}) = \llbracket (f\gamma_e^m) \rrbracket_{(\mathcal{K}, \mu)}^\rho.$$

The result is extended to probabilistic terms and formulas by induction. \square

The soundness of the axiom for probabilistic toss, **TOSS**, is an easy consequence of the following lemma.

Lemma 5.3 (Substitution Lemma for probabilistic tosses) Let (K, μ) be a generalized probabilistic structure, ρ be a \mathcal{K} -assignment, $r \in \mathcal{A}$ be a constant and $\mu' = \tilde{r} \mu \circ (\delta_{\text{tt}}^{\text{bm}})^{-1} + (1 - \tilde{r}) \mu \circ (\delta_{\text{ff}}^{\text{bm}})^{-1}$. Then, for any classical state formula γ ,

$$\llbracket (f\gamma) \rrbracket_{(\mathcal{K}, \mu')}^\rho = \tilde{r} \llbracket (f\gamma_{\text{tt}}^{\text{bm}}) \rrbracket_{(K, \mu)}^\rho + (1 - \tilde{r}) \llbracket (f\gamma_{\text{ff}}^{\text{bm}}) \rrbracket_{(K, \mu)}^\rho.$$

Furthermore, for any probabilistic term p ,

$$\llbracket p \rrbracket_{(\mathcal{K}, \mu')}^\rho = \llbracket \text{toss}(\text{bm}, r; p) \rrbracket_{(K, \mu)}^\rho,$$

and, for any probabilistic formula η ,

$$(\mathcal{K}, \mu')\rho \Vdash \eta \text{ iff } (K, \mu)\rho \Vdash \text{toss}(\text{bm}, r; \eta).$$

Proof. Let $\mu_1 = \mu \circ (\delta_{\text{tt}}^{\text{bm}})^{-1}$ and $\mu_2 = \mu \circ (\delta_{\text{ff}}^{\text{bm}})^{-1}$. Then

$$\llbracket (f\gamma) \rrbracket_{(\mathcal{K}, \mu')}^\rho = \tilde{r} \llbracket (f\gamma) \rrbracket_{(\mathcal{K}, \mu_1)}^\rho + (1 - \tilde{r}) \llbracket (f\gamma) \rrbracket_{(\mathcal{K}, \mu_2)}^\rho$$

by definition; by Lemma 5.2

$$\llbracket (f\gamma) \rrbracket_{(\mathcal{K}, \mu_1)}^\rho = \llbracket (f\gamma_{\text{tt}}^{\text{bm}}) \rrbracket_{(\mathcal{K}, \mu)}^\rho \text{ and } \llbracket (f\gamma) \rrbracket_{(\mathcal{K}, \mu_2)}^\rho = \llbracket (f\gamma_{\text{ff}}^{\text{bm}}) \rrbracket_{(\mathcal{K}, \mu)}^\rho.$$

The claim for probabilistic terms and probabilistic formulas then follows by induction. \square

The following proposition asserts the soundness of the axiom **fFREE**.

Proposition 5.4 (Soundness of fFREE) For any statement s , any analytical formula κ , any generalized state (\mathcal{K}, μ) and \mathcal{K} assignment ρ ,

$$(\llbracket s \rrbracket(\mathcal{K}, \mu))\rho \Vdash \kappa \text{ iff } (\mathcal{K}, \mu)\rho \Vdash \kappa.$$

Proof. The claim follows from the fact that interpretation of analytical depends only on the assignment ρ . \square

Proposition 5.5 For any generalized state (\mathcal{K}, μ) , \mathcal{K} -assignment ρ and classical state formulas γ and γ' ,

$$\llbracket (f\gamma')/\gamma \rrbracket_{(\mathcal{K}, \mu)}^\rho = \llbracket (f\gamma') \rrbracket_{(\mathcal{K}, \mu_\gamma)}^\rho.$$

Furthermore, for any probability term p ,

$$\llbracket p/\gamma \rrbracket_{(\mathcal{K}, \mu)}^\rho = \llbracket p \rrbracket_{(\mathcal{K}, \mu_\gamma)}^\rho,$$

and, for any probabilistic formula η ,

$$(\mathcal{K}, \mu)\rho \Vdash \eta/\gamma \text{ iff } (\mathcal{K}, \mu_\gamma)\rho \Vdash \eta.$$

Proof. By definition,

$$\llbracket (f\gamma') \rrbracket_{(\mathcal{K}, \mu_\gamma)}^\rho = \mu_\gamma(|\gamma'|_{\mathcal{V}}) = \mu(|\gamma'|_{\mathcal{V}} \cap |\gamma|_{\mathcal{V}}) = \mu(|\gamma' \wedge \gamma|_{\mathcal{V}}) = \llbracket (f\gamma')/\gamma \rrbracket_{(\mathcal{K}, \mu)}^\rho.$$

The claims for probabilistic terms and formulas now follow by induction. \square

We can now establish the soundness of the inference rule **IF**.

Lemma 5.6 (Soundness of IF) Given probabilistic state formulas η_1 and η_2 , programs s_1 and s_2 , variables $y_1 \in \mathcal{Y}$ and $y_2 \in \mathcal{Y}$ and a classical state formula γ ,

$$\models_h \{\eta_1\} s_1 \{y_1 = (f\gamma)\} \text{ and } \models_h \{\eta_2\} s_2 \{y_2 = (f\gamma)\}$$

iff, for any classical state formula γ_0 ,

$$\models_h \{\eta_1 \Upsilon_{\gamma_0} \eta_2\} \text{ if } \gamma_0 \text{ then } s_1 \text{ else } s_2 \{y_1 + y_2 = (f\gamma)\}.$$

Proof. Let (\mathcal{K}, μ) be a generalized probabilistic state and ρ be a \mathcal{K} -assignment such that $(\mathcal{K}, \mu)\rho \Vdash \eta_1 \Upsilon_{\gamma_0} \eta_2$. Then $(\mathcal{K}, \mu)\rho \Vdash \eta_1/\gamma_0$ and $(\mathcal{K}, \mu)\rho \Vdash \eta_2/(\neg\gamma_0)$. Thus, $(\mathcal{K}, \mu_{\gamma_0})\rho \Vdash \eta_1$ and $(\mathcal{K}, \mu_{(\neg\gamma_0)})\rho \Vdash \eta_2$ by Proposition 5.5.

Let $(\mathcal{K}, \mu_1) = \llbracket s_1 \rrbracket(\mathcal{K}, \mu_{\gamma_0})$, $(\mathcal{K}, \mu_2) = \llbracket s_2 \rrbracket(\mathcal{K}, \mu_{(\neg\gamma_0)})$ and $\mu' = \mu_1 + \mu_2$. We need to show that $(\mathcal{K}, \mu')\rho \Vdash (y_1 + y_2 = (f\gamma))$.

Since $\Vdash_h \{\eta_1\} s_1 \{y_1 = (f\gamma)\}$ and $(\mathcal{K}, \mu_{\gamma_0})\rho \Vdash \eta_1$, it follows that $(\mathcal{K}, \mu_1) \Vdash_h y_1 = (f\gamma)$. Thus, by definition $\rho(y_1) = \mu_1(|\gamma|_{\mathcal{V}})$. Similarly, $\rho(y_2) = \mu_2(|\gamma|_{\mathcal{V}})$.

Hence, $\mu'(|\gamma|_{\mathcal{V}}) = \mu_1(|\gamma|_{\mathcal{V}}) + \mu_2(|\gamma|_{\mathcal{V}}) = \rho(y_1) + \rho(y_2) = \rho(y_1 + y_2)$. Therefore, $(\mathcal{K}, \mu')\rho \Vdash (y_1 + y_2 = (f\gamma))$ as required. \square

We now show that the inference rule **ELIMV** is sound. In order to do this, we shall first establish a substitution result for variables $y \in \mathbf{Y}$. For rest of the paper, given a \mathcal{K} -assignment ρ , a variable $y \in \mathbf{Y}$ and an element $k \in \mathcal{K}$, the \mathcal{K} -assignment ρ_k^y denotes the assignment that assigns the value k to y and coincides with ρ elsewhere.

Proposition 5.7 Let $y \in \mathbf{Y}$ be a variable and p be a probabilistic term. Given a general probabilistic structure (\mathcal{K}, μ) and a \mathcal{K} -assignment ρ , let $k = \llbracket p \rrbracket_{(\mathcal{K}, \mu)}^\rho$ and $\rho_1 = \rho_k^y$. Then:

- for any probabilistic term p_0 , $\llbracket p_0 \rrbracket_{(\mathcal{K}, \mu)}^{\rho_1} = \llbracket p_0 \rrbracket_{(\mathcal{K}, \mu)}^\rho$;
- for any probabilistic formula η , $(\mathcal{K}, \mu)\rho_1 \Vdash \eta$ iff $(\mathcal{K}, \mu)\rho \Vdash \eta_p^y$.

Proof. The first part of the proposition is proved by induction on the structure of p_0 . We consider the case when p_0 is a variable y_0 , the other cases being straightforward. If y_0 is y , then by definition $\llbracket y \rrbracket_{(\mathcal{K}, \mu)}^{\rho_1} = k = \llbracket p \rrbracket_{(\mathcal{K}, \mu)}^\rho = \llbracket y_p \rrbracket_{(\mathcal{K}, \mu)}^\rho$. Otherwise, $\llbracket y_0 \rrbracket_{(\mathcal{K}, \mu)}^{\rho_1} = \rho_1(y_0) = \rho(y_0) = \llbracket y_0 \rrbracket_{(\mathcal{K}, \mu)}^\rho = \llbracket y_0 \rrbracket_{(\mathcal{K}, \mu)}^\rho$.

The second part of the proposition follows by induction. \square

We make one more observation before we prove the soundness of **ELIMV**. Let $y \in \mathbf{Y}$ be a variable and η be a probabilistic formula such that η does not contain any occurrence of y . For any general probabilistic structure (\mathcal{K}, μ) and \mathcal{K} -assignments ρ_1 and ρ_2 such that $\rho_1(y') = \rho_2(y')$ for any y' distinct from y , a straightforward induction shows that

$$(\mathcal{K}, \mu)\rho_1 \Vdash \eta \text{ iff } (\mathcal{K}, \mu)\rho_2 \Vdash \eta.$$

Lemma 5.8 (Soundness of ELIMV) Given a probabilistic formula η , a

probabilistic term p , a probabilistic formula η , a variable $y \in \mathbf{Y}$ that does not occur either in p or in η and a statement s ,

if $\Vdash_h \{\eta \cap (y = p)\} s \{\eta\}$ then $\Vdash_h \{\eta_p^y\} s \{\eta\}$.

Proof. Assume that $\Vdash_h \{\eta \cap (y = p)\} s \{\eta\}$ and let (\mathcal{K}, μ) be a generalized state and ρ be a \mathcal{K} -assignment such that $(\mathcal{K}, \mu)\rho \Vdash \eta_p^y$. We need to show that $(\llbracket s \rrbracket(\mathcal{K}, \mu))\rho \Vdash \eta$.

Let $k = \llbracket p \rrbracket_{(\mathcal{K}, \mu)}^\rho$ and $\rho_1 = \rho_k^y$. By Proposition 5.7, $(\mathcal{K}, \mu)\rho_1 \Vdash \eta$.

By definition, $\llbracket y \rrbracket_{(\mathcal{K}, \mu)}^{\rho_1} = k$. By Proposition 5.7, $\llbracket p \rrbracket_{(\mathcal{K}, \mu)}^{\rho_1} = \llbracket p_p^y \rrbracket_{(\mathcal{K}, \mu)}^\rho$; since y does not occur in p , p_p^y is p itself, hence $\llbracket p \rrbracket_{(\mathcal{K}, \mu)}^{\rho_1} = \llbracket p \rrbracket_{(\mathcal{K}, \mu)}^\rho = k$. Therefore, $(\mathcal{K}, \mu)\rho_1 \Vdash (y = p)$.

Since $\Vdash_h \{\eta \cap (y = p)\} s \{\eta\}$, it follows that $(\llbracket s \rrbracket(\mathcal{K}, \mu))\rho_1 \Vdash \eta$. Since ρ_1 and ρ differ only in the value assigned to y and y does not occur in η , also $(\llbracket s \rrbracket(\mathcal{K}, \mu))\rho \Vdash \eta$ as required. \square

We are ready to prove the soundness of Hoare calculus.

Theorem 5.9 (Soundness of Hoare calculus) For any Hoare assertion Ψ , if $\vdash \Psi$ then $\models \Psi$.

Proof. The proof is by induction on the length of the derivation of $\vdash \Psi$; it suffices to show that each of the axioms and inference rules is sound.

The soundness of axioms **TAUT** and **SKIP** and of the inference rules **SEQ**, **AND**, **OR** and **CONS** is straightforward.

The soundness of axioms **ASGR** and **ASGB** follows from Lemma 5.2 and that of axiom **TOSS** from Lemma 5.3. The soundness of the axiom **fFREE** follows from Proposition 5.4, while Lemmas 5.6 and 5.8 establish the soundness of inference rules **IF** and **ELIMV** respectively. \square

6 Completeness and decidability of the Hoare calculus

We now show that the Hoare calculus provided in Section 4 is complete, *i.e.*, if $\Vdash_h \Psi$ then $\vdash \Psi$. Furthermore, there is an algorithm that given a probabilistic Hoare formula Ψ determines whether $\Vdash_h \Psi$ or $\not\Vdash_h \Psi$. The proof of completeness and decidability of the Hoare logic uses the completeness and decidability of EPPL (see Theorem 2.2).

The proof of completeness of the Hoare logic employs the standard tech-

nique [10] of defining the *weakest precondition* operator. Intuitively, the weakest precondition operator $\mathbf{wp}(\cdot, \cdot)$ assigns to each statement $s \in \mathbf{S}$ and each formula $\eta \in \mathbf{PForms}$ a new state formula $\mathbf{wp}(s, \eta)$ that corresponds to the weakest logical property that a state must satisfy to ensure that η holds after execution of s . The weakest precondition itself uses the *preterm* operator. Intuitively, the preterm operator $\mathbf{pt}(\cdot, \cdot)$ assigns to each statement $s \in \mathbf{S}$ and each probabilistic term $p \in \mathbf{PTerms}$ a new term $\mathbf{pt}(s, p)$ whose denotation in a given initial state is the same as the denotation of p after execution of s .

We then show that for any program s and EPPL formula η the Hoare calculus derives the judgment $\vdash \{\mathbf{wp}(s, \eta)\} s \{\eta\}$, establishing by correctness (Theorem 5.9) that $\mathbf{wp}(s, \eta)$ is a sufficient precondition for s and η . Furthermore, $(\mathcal{K}, \mu)\rho \Vdash \mathbf{wp}(s, \eta)$ iff $\llbracket s \rrbracket(\mathcal{K}, \mu)\rho \Vdash \eta$, implying that if $\models_h \{\eta'\} s \{\eta\}$ then $\models (\eta' \supset \mathbf{wp}(s, \eta))$. The completeness of EPPL will allow us to conclude that $(\eta' \supset \mathbf{wp}(s, \eta))$ is an EPPL theorem, and we can then use the Hoare inference rule **CONS** to conclude that $\vdash \{\eta'\} s \{\eta\}$. The decidability of the Hoare calculus follows from the fact that the weakest precondition can be computed algorithmically and decidability of EPPL.

6.1 Preterms

The preterm $\mathbf{pt}(s, p)$ is defined recursively on the structure of the statement s and the probability term p .

Recall that, given a memory cell \mathbf{bm} , a constant $r \in \mathcal{A}$ and a probabilistic term p , the term $\mathbf{toss}(\mathbf{bm}, r; p)$ is the term obtained from p by replacing every occurrence of each measure term $(f\gamma)$ by $\tilde{r}(f\gamma_{\text{tt}}^{\mathbf{bm}}) + (1 - \tilde{r})(f\gamma_{\text{ff}}^{\mathbf{bm}})$ and, given a classical state formula γ and a probabilistic term p , the term (p/γ) is the term obtained from p by replacing every occurrence of each measure term $(f\gamma')$ by $(f(\gamma' \wedge \gamma))$. The definition of $\mathbf{pt}(s, p)$ is shown in Table 8.

The preterm operator acts as the identity on the constants and the variables. Furthermore, the set of variables occurring in the term is unchanged.

Proposition 6.1 For any statement s , the following hold:

- $\mathbf{pt}(s, r) = r$ for all $r \in \mathcal{A}$;
- $\mathbf{pt}(s, y) = y$ for all $y \in \mathbf{Y}$;
- $\mathbf{PVar}(p) = \mathbf{PVar}(\mathbf{pt}(s, p))$ for all probabilistic terms p .

Proof. By induction on the structure of s and p . \square

Lemma 6.2 For any probabilistic term p , statement s , any generalized struc-

$\text{pt}(\text{skip}, p)$	$= p$
$\text{pt}(\text{bm} \leftarrow \gamma, p)$	$= p_\gamma^{\text{bm}}$
$\text{pt}(\text{xm} \leftarrow t, p)$	$= p_t^{\text{xm}}$
$\text{pt}(\text{toss}(\text{bm}, r), p)$	$= \text{toss}(\text{bm}, r; p)$
$\text{pt}(s_1; s_2, p)$	$= \text{pt}(s_1, \text{pt}(s_2, p))$
$\text{pt}(\text{if } \gamma \text{ then } s_1 \text{ else } s_2, r)$	$= r$
$\text{pt}(\text{if } \gamma \text{ then } s_1 \text{ else } s_2, y)$	$= y$
$\text{pt}(\text{if } \gamma \text{ then } s_1 \text{ else } s_2, (f\gamma_0))$	$= (\text{pt}(s_1, (f\gamma_0))/\gamma + \text{pt}(s_2, (f\gamma_0))/(\neg\gamma))$
$\text{pt}(\text{if } \gamma \text{ then } s_1 \text{ else } s_2, (p_1 + p_2))$	$= (\text{pt}(\text{if } \gamma \text{ then } s_1 \text{ else } s_2, p_1) +$ $\text{pt}(\text{if } \gamma \text{ then } s_1 \text{ else } s_2, p_2))$
$\text{pt}(\text{if } \gamma \text{ then } s_1 \text{ else } s_2, (p_1 p_2))$	$= (\text{pt}(\text{if } \gamma \text{ then } s_1 \text{ else } s_2, p_1) \times$ $\text{pt}(\text{if } \gamma \text{ then } s_1 \text{ else } s_2, p_2))$

Table 8
Preterms

ture (\mathcal{K}, μ) and \mathcal{K} -assignment ρ ,

$$\llbracket \text{pt}(s, p) \rrbracket_{(\mathcal{K}, \mu)}^\rho = \llbracket p \rrbracket_{\llbracket s \rrbracket_{(\mathcal{K}, \mu)}}^\rho.$$

Proof. By induction on the structure of s . The case when s is `skip` follows from the definition. The cases when s is an assignment to a memory cell or a probabilistic toss follow respectively from Lemmas 5.2 and 5.3.

If s is $s_1; s_2$, then applying the induction hypothesis twice to a given a probabilistic term p yields

$$\llbracket \text{pt}(s_1, \text{pt}(s_2, p)) \rrbracket_{(\mathcal{K}, \mu)}^\rho = \llbracket \text{pt}(s_2, p) \rrbracket_{\llbracket s_1 \rrbracket_{(\mathcal{K}, \mu)}}^\rho = \llbracket p \rrbracket_{\llbracket s_1; s_2 \rrbracket_{(\mathcal{K}, \mu)}}^\rho$$

as required.

If s is `if γ then s_1 else s_2` , we proceed by induction on p . The case when p is a constant $r \in \mathcal{A}$ is immediate from the definition; the case when p is the variable y follows from the fact that the interpretation of a variable depends only on the \mathcal{K} -assignment ρ .

If p is the term $(f\gamma_0)$ then by definition

$$\llbracket (f\gamma_0) \rrbracket_{\llbracket s \rrbracket_{(\mathcal{K}, \mu)}}^\rho = \kappa_1 + \kappa_2$$

where

$$\kappa_1 = \llbracket (f\gamma_0) \rrbracket_{\llbracket s_1 \rrbracket(\mathcal{K}, \mu_\gamma)}^\rho \text{ and } \kappa_2 = \llbracket (f\gamma_0) \rrbracket_{\llbracket s_2 \rrbracket(\mathcal{K}, \mu_{(\neg\gamma)})}^\rho.$$

Applying the induction hypothesis to s_1 and s_2 yields respectively

$$\kappa_1 = \llbracket \mathbf{pt}(s_1, (f\gamma_0)) \rrbracket_{(\mathcal{K}, \mu_\gamma)}^\rho \text{ and } \kappa_2 = \llbracket \mathbf{pt}(s_2, (f\gamma_0)) \rrbracket_{(\mathcal{K}, \mu_{(\neg\gamma)})}^\rho.$$

By Proposition 5.5,

$$\kappa_1 = \llbracket \mathbf{pt}(s_1, (f\gamma_0)) \rrbracket_{(\mathcal{K}, \mu_\gamma)}^\rho = \llbracket \mathbf{pt}(s_1, (f\gamma_0)) / \gamma \rrbracket_{(\mathcal{K}, \mu)}^\rho$$

and

$$\kappa_2 = \llbracket \mathbf{pt}(s_2, (f\gamma_0)) \rrbracket_{(\mathcal{K}, \mu_{(\neg\gamma)})}^\rho = \llbracket \mathbf{pt}(s_2, (f\gamma_0)) / (\neg\gamma) \rrbracket_{(\mathcal{K}, \mu)}^\rho;$$

the result now follows.

If p is $(p_1 + p_2)$ or $(p_1 p_2)$, then by induction hypothesis

$$\llbracket \mathbf{pt}(s, p_i) \rrbracket_{(\mathcal{K}, \mu)}^\rho = \llbracket p_i \rrbracket_{\llbracket s \rrbracket(\mathcal{K}, \mu)}^\rho$$

for $i = 1, 2$ and the result follows immediately. \square

6.2 Weakest preconditions

The weakest precondition operator $\mathbf{wp} : \mathbf{S} \times \mathbf{PForms} \rightarrow \mathbf{PForms}$ is defined using the preterm operator. The weakest precondition $\mathbf{wp}(s, \eta)$ is obtained by replacing each comparison formula $(p_1 \leq p_2)$ occurring in $\mathbf{wp}(s, \eta)$ by $(\mathbf{pt}(s, p_1) \leq \mathbf{pt}(s, p_2))$. The formal definition can be found in Table 9.

$\mathbf{wp}(s, \mathbf{fff})$	=	\mathbf{fff}
$\mathbf{wp}(s, (p_1 \leq p_2))$	=	$(\mathbf{pt}(s, p_1) \leq \mathbf{pt}(s, p_2))$
$\mathbf{wp}(s, (\eta_1 \supset \eta_2))$	=	$(\mathbf{wp}(s, \eta_1) \supset \mathbf{wp}(s, \eta_2))$

Table 9
Weakest preconditions

It follows from the definition and Lemma 6.2 that $\mathbf{wp}(s, \eta)$ is indeed the weakest precondition for η to hold after execution of s .

Theorem 6.3 For any statement s , probabilistic formula η , generalized structure (\mathcal{K}, μ) and \mathcal{K} -assignment ρ ,

$$(\mathcal{K}, \mu)\rho \Vdash_h \mathbf{wp}(s, \eta) \text{ iff } (\llbracket s \rrbracket(\mathcal{K}, \mu))\rho \Vdash_h \eta.$$

Proof. The proof is by induction on the structure of η . The base case where η is fff is immediate. The other base is when η is $(p_1 \leq p_2)$; by Lemma 6.2,

$$\llbracket \mathbf{pt}(s, p_i) \rrbracket_{(\mathcal{K}, \mu)}^\rho = \llbracket p_i \rrbracket_{\llbracket s \rrbracket(\mathcal{K}, \mu)}^\rho$$

for $i = 1, 2$, and the result follows.

Finally, if η is $(\eta_1 \supset \eta_2)$ then by induction hypothesis

$$(\mathcal{K}, \mu)\rho \Vdash_h \mathbf{wp}(s, \eta_i) \text{ iff } (\llbracket s \rrbracket(\mathcal{K}, \mu))\rho \Vdash_h \eta_i$$

for $i = 1, 2$, and the result follows. \square

The following corollary is straightforward.

Corollary 6.4 For any statement s and probabilistic formulas η and η' ,

$$\models_h \{\eta'\} s \{\eta\} \text{ iff } \models (\eta' \supset \mathbf{wp}(s, \eta)).$$

Proof. (\Rightarrow) Suppose $\models_h \{\eta'\} s \{\eta\}$. Consider an arbitrary generalized probabilistic state (\mathcal{K}, μ) and an arbitrary \mathcal{K} -assignment ρ such that $(\mathcal{K}, \mu)\rho \Vdash \eta'$; then $(\llbracket s \rrbracket(\mathcal{K}, \mu))\rho \Vdash \eta$, since $\models_h \{\eta'\} s \{\eta\}$. By Theorem 6.3, $(\mathcal{K}, \mu)\rho \Vdash \mathbf{wp}(s, \eta)$; since (\mathcal{K}, μ) and ρ are arbitrary, $\models (\eta' \supset \mathbf{wp}(s, \eta))$.

(\Leftarrow) Suppose $\models (\eta' \supset \mathbf{wp}(s, \eta))$. Consider an arbitrary generalized probabilistic state (\mathcal{K}, μ) and an arbitrary \mathcal{K} -assignment ρ such that $(\mathcal{K}, \mu)\rho \Vdash \eta'$. Then $(\mathcal{K}, \mu)\rho \Vdash \mathbf{wp}(s, \eta)$, since $\models (\eta' \supset \mathbf{wp}(s, \eta))$; by Theorem 6.3, $(\llbracket s \rrbracket(\mathcal{K}, \mu))\rho \Vdash \eta$. Since (\mathcal{K}, μ) and ρ are arbitrary, $\models_h \{\eta'\} s \{\eta\}$. \square

The next step is to show that the Hoare axiomatization allows us to derive the judgment

$$\vdash \{\mathbf{wp}(s, \eta)\} s \{\eta\}.$$

We start by showing this the special case when η is $y = p$ for some variable $y \in \mathbf{Y}$ and probabilistic term p .

Lemma 6.5 For any probabilistic term p , statement s and variable $y \in \mathbf{Y}$,

$$\vdash \{y = \mathbf{pt}(s, p)\} s \{y = p\}.$$

Proof. By induction on the structure of s . If s is **skip**, an assignment to a memory cell or a probabilistic toss, then the required judgment can be derived by axioms **SKIP**, **ASGB**, **ASGR** or **TOSS**.

If s is $s_1; s_2$, then $\mathbf{pt}(s_1; s_2, p) = \mathbf{pt}(s_1, \mathbf{pt}(s_2, p))$ by definition, and the induction hypothesis applied to the programs s_1 and s_2 gives

$$\vdash \{y = \mathbf{pt}(s_1, \mathbf{pt}(s_2, p))\} s_1 \{y = \mathbf{pt}(s_2, p)\}$$

and $\vdash \{y = \mathbf{pt}(s_2, p)\} s_2 \{y = p\}$ respectively. By **SEQ** it follows that

$$\vdash \{y = \mathbf{pt}(s_1; s_2, p)\} s_1; s_2 \{y = p\}.$$

If s is the alternative **if** γ **then** s_1 **else** s_2 we proceed by induction on p . If p is a constant or a variable, then $\vdash \{y = p\} s \{y = p\}$ by axiom **fFREE** and the result follows by observing that in these cases $\mathbf{pt}(s, p) = p$ by Proposition 6.1.

Suppose p is $(f\gamma_0)$ for some classical state formula γ_0 and pick two distinct variables $y_1, y_2 \in \mathbf{Y}$ different from y . Let η_1 be $(y_1 = \mathbf{pt}(s_1, (f\gamma_0)))$, η_2 be $(y_2 = \mathbf{pt}(s_2, (f\gamma_0)))$ and η^\dagger be

$$(y = y_1 + y_2) \cap (y_1 = \mathbf{pt}(s_1, (f\gamma_0))/\gamma) \cap (y_2 = \mathbf{pt}(s_2, (f\gamma_0))/(\neg\gamma)).$$

By the outer induction hypothesis (on s_1 and s_2),

$$\vdash \{y_i = \mathbf{pt}(s_i, (f\gamma_0))\} s_i \{y_i = (f\gamma_0)\}$$

for $i = 1, 2$. Since $\mathbf{pt}(s, p) = \mathbf{pt}(s_1, (f\gamma_0))/\gamma + \mathbf{pt}(s_2, (f\gamma_0))/(\neg\gamma)$, we can

derive $\{y = \mathbf{pt}(s, p)\} s \{y = p\}$ as follows.

1. $\{y_1 = \mathbf{pt}(s_1, (f\gamma_0))\} s_1 \{y_1 = (f\gamma_0)\}$ Lemma
2. $\{y_2 = \mathbf{pt}(s_2, (f\gamma_0))\} s_2 \{y_2 = (f\gamma_0)\}$ Lemma
3. $\{\eta_1 \Upsilon_\gamma \eta_2\}$ if γ then s_1 else $s_2 \{y_1 + y_2 = (f\gamma_0)\}$ **IF** 1,2
4. $\eta^\dagger \supset (\eta_1 \Upsilon_\gamma \eta_2)$ **TAUT**
5. $\{\eta^\dagger\}$ if γ then s_1 else $s_2 \{y_1 + y_2 = (f\gamma_0)\}$ **CONS** 3,4
6. $\{y = y_1 + y_2\}$ if γ then s_1 else $s_2 \{y = y_1 + y_2\}$ **fFREE**
7. $\eta^\dagger \supset (y = y_1 + y_2)$ **TAUT**
8. $\{\eta^\dagger\}$ if γ then s_1 else $s_2 \{y = y_1 + y_2\}$ **CONS** 6,7
9. $\{\eta^\dagger\}$ if γ then s_1 else $s_2 \{(y = y_1 + y_2) \cap (y_1 + y_2 = (f\gamma_0))\}$ **AND** 5,8
10. $((y = y_1 + y_2) \cap (y_1 + y_2 = (f\gamma_0))) \supset (y = (f\gamma_0))$ **fFREE**
11. $\{\eta^\dagger\}$ if γ then s_1 else $s_2 \{y = (f\gamma_0)\}$ **CONS** 9,10
12. $\{(y = y_1 + \mathbf{pt}(s_2, (f\gamma_0))) \cap (y_1 = \mathbf{pt}(s_1, (f\gamma_0)) / \gamma)\}$
if γ then s_1 else $s_2 \{y = (f\gamma_0)\}$ **ELIMV** 11
13. $\{y = \mathbf{pt}(s_1, (f\gamma_0)) / \gamma + \mathbf{pt}(s_2, (f\gamma_0)) / (\neg \gamma)\}$
if γ then s_1 else $s_2 \{y = (f\gamma_0)\}$ **ELIMV**12

If p is $(p_1 + p_2)$, pick $y_1, y_2 \in \mathbf{Y}$ different from y such that y_1 and y_2 do not occur in either p_1 or p_2 . Let η^\dagger be

$$(y = y_1 + y_2) \cap (y_1 = p_1) \cap (y_2 = p_2)$$

and define η^\ddagger as

$$(y = y_1 + y_2) \cap (y_1 = \mathbf{pt}(s, p_1)) \cap (y_2 = \mathbf{pt}(s, p_2)).$$

By the inner induction hypothesis (in y_1 and y_2), $\vdash \{y_i = \mathbf{pt}(s, p_i)\} s \{y_i = p_i\}$ for $i = 1, 2$. Then the judgment $\{y = \mathbf{pt}(s, p)\} s \{y = p\}$ can be derived as

follows.

- | | | |
|-----|--|-------------------|
| 1. | $\{y_1 = \mathbf{pt}(s, p_1)\} s \{y_1 = p_1\}$ | Lemma |
| 2. | $\eta^\dagger \supset (y_1 = \mathbf{pt}(s, p_1))$ | TAUT |
| 3. | $\{\eta^\dagger\} s \{y_1 = p_1\}$ | CONS 1,2 |
| 4. | $\{y_2 = \mathbf{pt}(s, p_2)\} s \{y_2 = p_2\}$ | Lemma |
| 5. | $\eta^\dagger \supset (y_2 = \mathbf{pt}(s, p_2))$ | TAUT |
| 6. | $\{\eta^\dagger\} s \{y_2 = p_2\}$ | CONS 4,5 |
| 7. | $\{\eta^\dagger\} s \{(y_1 = p_1) \cap (y_2 = p_2)\}$ | AND 3,6 |
| 8. | $\{y = y_1 + y_2\} s \{y = y_1 + y_2\}$ | fFREE |
| 9. | $\eta^\dagger \supset (y = y_1 + y_2)$ | TAUT |
| 10. | $\{\eta^\dagger\} s \{y = y_1 + y_2\}$ | CONS 8,9 |
| 11. | $\{\eta^\dagger\} s \{\eta^\dagger\}$ | AND 7,10 |
| 12. | $\eta^\dagger \supset (y = (p_1 + p_2))$ | fFREE |
| 13. | $\{\eta^\dagger\} s \{y = (p_1 + p_2)\}$ | CONS 11,12 |
| 14. | $\{(y = y_1 + \mathbf{pt}(s, p_2)) \cap (y_1 = \mathbf{pt}(s, p_1))\} s \{y = (p_1 + p_2)\}$ | ELIMV 13 |
| 15. | $\{y = \mathbf{pt}(s, p_1) + \mathbf{pt}(s, p_2)\} s \{y = (p_1 + p_2)\}$ | ELIMV 14 |

The case where p is $(p_1 p_2)$ is similar. \square

We are now ready to show that the judgment $\{\mathbf{wp}(s, \eta)\} s \{\eta\}$ is derivable in the Hoare logic for any η . Given a probabilistic formula η and a probabilistic term p we say that p *occurs as a comparison term in η* if there is some probabilistic term q such that either the comparison formula $(p \leq q)$ or the comparison formula $(q \leq p)$ occurs in η .

Theorem 6.6 For any statement s and any conditional free formula η ,

$$\vdash \{\mathbf{wp}(s, \eta)\} s \{\eta\}.$$

Proof. Let p_1, p_2, \dots, p_n be all the comparison terms occurring in η . Pick n distinct variables $y_1, y_2, \dots, y_n \in \mathbf{Y}$ that do not occur in η . Let p'_1, p'_2, \dots, p'_n be the terms $\mathbf{pt}(s, p_1), \mathbf{pt}(s, p_2), \dots, \mathbf{pt}(s, p_n)$ respectively and let η^\dagger be the formula obtained from η by replacing each occurrence of a comparison formula

$(p_i \leq p_j)$ by $(y_i \leq y_j)$. Finally, take

$$\eta_a \equiv \eta^\dagger \cap \left(\bigcap_i (y_i = p_i) \right) \quad \text{and} \quad \eta_b \equiv \eta^\dagger \cap \left(\bigcap_i (y_i = p'_i) \right).$$

Clearly, the following hold:

- η^\dagger is an analytical formula;
- $\eta_{\substack{y_1 y_2 \dots y_n \\ p_1 p_2 \dots p_n}}^\dagger$ is η ;
- $(\eta_a \supset \eta^\dagger)$ and $(\eta_b \supset \eta^\dagger)$ are EPPL theorems;
- $(\eta_b \supset (y_i = p'_i))$ are EPPL theorems for all $1 \leq i \leq n$;
- $\text{wp}(s, \eta)$ is $\eta_{\substack{y_1 y_2 \dots y_n \\ p'_1 p'_2 \dots p'_n}}^\dagger$.

By axiom **JFREE**, $\vdash \{\eta^\dagger\} s \{\eta^\dagger\}$; by Lemma 6.5, $\vdash \{y_i = p'_i\} s \{y_i = p_i\}$ for all $1 \leq i \leq n$. Since $(\eta_b \supset \eta^\dagger)$ and $(\eta_b \supset (y_i = p'_i))$ are EPPL theorems for all $1 \leq i \leq n$, by application of **CONS** it follows that

$$\vdash \{\eta_b\} s \{\eta^\dagger\} \quad \text{and} \quad \vdash \{\eta_b\} s \{y_i = p_i\}$$

for all $1 \leq i \leq n$. Several applications of the inference rule **AND** then give $\vdash \{\eta_b\} s \{\eta_a\}$; since $(\eta_a \supset \eta)$ is an EPPL theorem, another application of **CONS** yields $\vdash \{\eta_b\} s \{\eta\}$. Finally, several applications of **ELIMV** show that

$$\vdash \{\eta_{\substack{y_1 y_2 \dots y_n \\ p'_1 p'_2 \dots p'_n}}^\dagger\} s \{\eta\}$$

as required. \square

We are ready to show the Hoare calculus is complete and decidable.

Theorem 6.7 (Completeness and decidability) Let s be a probabilistic sequential program and η be an EPPL formula. If $\models_h \{\eta'\} s \{\eta\}$, then $\vdash \{\eta'\} s \{\eta\}$. Moreover, the set of theorems of the Hoare calculus is recursive.

Proof.

Completeness. Suppose that $\models_h \{\eta'\} s \{\eta\}$. By Corollary 6.4, $\models (\eta' \supset \text{wp}(s, \eta))$. By completeness of EPPL, $\vdash (\eta' \supset \text{wp}(s, \eta))$. Theorem 6.6 implies that $\vdash \{\text{wp}(s, \eta)\} s \{\eta\}$, whence $\vdash \{\eta'\} s \{\eta\}$ by **CONS**.

Decidability. By soundness and completeness, $\vdash \{\eta'\} s \{\eta\}$ iff $\models_h \{\eta'\} s \{\eta\}$. By Corollary 6.4 and completeness of EPPL, it follows that $\vdash \{\eta'\} s \{\eta\}$ iff $\vdash (\eta' \supset \text{wp}(s, \eta))$. The decidability is now a consequence of the decidability of EPPL and the fact that $\text{wp}(s, \eta)$ can be computed algorithmically. \square

The complexity of the formula $\text{wp}(s, \eta)$ is exponential on the number of nested if-then-else commands containing terms denoting probabilities of formulas, as

seen in Table 8. Furthermore, EPPL is double exponential in the number of propositional symbols and exponential in the number of variables, as discussed at the end of Section 2.

7 Examples

We now present two examples and compute the weakest pre-condition of two programs.

One-time pad. A *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 the bitwise xor of m and k , where k is a key that will be used only once.

We model this via the following program S_{enc} , which generates a random 1-bit key \mathbf{bm}_k and encrypts the 1-bit plain-text \mathbf{bm}_p ⁷.

```
toss( $\mathbf{bm}_k, \frac{1}{2}$ );
 $\mathbf{bm}_c \leftarrow \neg(\mathbf{bm}_k \Leftrightarrow \mathbf{bm}_p)$ 
```

The security of this one-time pad is equivalent to requiring that the probability of the cipher-text \mathbf{xm}_c being \mathbf{tt} be $\frac{1}{2}$ regardless of the probability distribution on the possible values of the plain-text \mathbf{xm}_p . This can be expressed by the following Hoare assertion:

$$\Psi \equiv \{(f\mathbf{tt}) = 1\} S_{\text{enc}} \{(f\mathbf{bm}_c) = \frac{1}{2}\}.$$

The pre-condition $(f\mathbf{tt}) = 1$ means that the total measure of the space of valuations is 1. Although Ψ is derivable in our Hoare calculus, as shown in [8], we shall show that there exists a derivation not by building one directly, but simply by computing weakest preconditions and applying the above results.

By definition,

$$\left(\mathbf{wp}(S_{\text{enc}}, (f\mathbf{bm}_c) = \frac{1}{2})\right) \equiv \left(\mathbf{pt}(S_{\text{enc}}, (f\mathbf{bm}_c)) = \mathbf{pt}(S_{\text{enc}}, \frac{1}{2})\right).$$

Now, $\mathbf{pt}(S_{\text{enc}}, \frac{1}{2})$ is $\frac{1}{2}$ by Proposition 6.1. On the other hand,

⁷ Observe that $\neg(\mathbf{bm}_k \Leftrightarrow \mathbf{bm}_p)$ is a way of computing $(\mathbf{bm}_k \text{ xor } \mathbf{bm}_p)$.

$$\begin{aligned}
& \text{pt}(S_{\text{enc}}, (f\mathbf{bm}_c)) \\
&= \text{pt}(\text{toss}(\mathbf{bm}_k, \frac{1}{2}), \text{pt}(\mathbf{bm}_c \leftarrow \neg(\mathbf{bm}_k \Leftrightarrow \mathbf{bm}_p), (f\mathbf{bm}_c))) \\
&= \text{pt}(\text{toss}(\mathbf{bm}_k, \frac{1}{2}), (f\neg(\mathbf{bm}_k \Leftrightarrow \mathbf{bm}_p))) \\
&= \frac{\tilde{1}}{2}(f\neg(\mathbf{bm}_k \Leftrightarrow \mathbf{tt})) + (1 - \frac{\tilde{1}}{2})(f\neg(\mathbf{bm}_k \Leftrightarrow \mathbf{ff})).
\end{aligned}$$

Hence,

$$\begin{aligned}
& \left(\text{wp}(S_{\text{enc}}, (f\mathbf{bm}_c) = \frac{1}{2}) \right) \equiv \\
& \left(\frac{\tilde{1}}{2}(f\neg(\mathbf{bm}_k \Leftrightarrow \mathbf{tt})) + (1 - \frac{\tilde{1}}{2})(f\neg(\mathbf{bm}_k \Leftrightarrow \mathbf{ff})) = \frac{1}{2} \right).
\end{aligned}$$

The derivability of Ψ now follows from the fact that

$$((f\mathbf{tt}) = 1) \approx \left(\frac{\tilde{1}}{2}(f\neg(\mathbf{bm}_k \Leftrightarrow \mathbf{tt})) + (1 - \frac{\tilde{1}}{2})(f\neg(\mathbf{bm}_k \Leftrightarrow \mathbf{ff})) = \frac{1}{2} \right)$$

is an EPPL theorem, since both sides of the equivalence are clearly equivalent to $(f(\mathbf{bm}_k \Leftrightarrow \mathbf{ff})) + (f(\mathbf{bm}_k \Leftrightarrow \mathbf{tt})) = 1$.

Quantum one-time pad. We now present a quantum variation of the previous example. 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, a one-bit key \mathbf{bm}_k suffices; we restrict our attention to this special case.

If the key \mathbf{bm}_k is 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 in state $(\mathbf{xm}_1, \mathbf{xm}_2)$.

`toss($\mathbf{bm}_k, \frac{1}{2}$); if \mathbf{bm}_k then PauliXZ else skip`

Here, Pauli_{XZ} is $\mathbf{xm}_3 \leftarrow \mathbf{xm}_1; \mathbf{xm}_1 \leftarrow \mathbf{xm}_2; \mathbf{xm}_2 \leftarrow -\mathbf{xm}_3$; the name Pauli_{XZ} has its roots in quantum mechanics.

Assume that the initial values of \mathbf{xm}_1 and \mathbf{xm}_2 are c_1 and c_2 respectively, with $c_1 \neq c_2$. It follows from quantum information theory that the quantum one-time pad is secure if the probability of \mathbf{xm}_1 being c_1 after encryption is $\frac{1}{2}$ (and hence that of \mathbf{xm}_1 being c_2 is also $\frac{1}{2}$). Assuming η_I is $\square((\mathbf{xm}_1 = c_1) \wedge (\mathbf{xm}_2 =$

$c_2) \wedge (c_1 < c_2)$), this can be expressed by the following Hoare assertion.

$$\Psi \equiv \{((f\mathbf{tt}) = 1) \cap \eta_I\} S_{\text{qenc}} \{(f(\mathbf{xm}_1 = c_1)) = \frac{1}{2}\}.$$

This Hoare assertion can be shown to hold by the method of weakest preconditions.

By definition of preterm,

$$\begin{aligned} \text{pt}(\text{Pauli}_{XZ}, (f(\mathbf{xm}_1 = c_1))) &\text{ is } (f(\mathbf{xm}_2 = c_1)) \\ \text{pt}(\text{skip}, (f(\mathbf{xm}_1 = c_1))) &\text{ is } (f(\mathbf{xm}_1 = c_1)). \end{aligned}$$

Hence,

$$\begin{aligned} \text{pt}(\text{if } \mathbf{bm}_k \text{ then } \text{Pauli}_{XZ} \text{ else skip}, (f(\mathbf{xm}_1 = c_1))) \\ = (f(\mathbf{xm}_2 = c_1) \wedge \mathbf{bm}_k) + (f(\mathbf{xm}_1 = c_1) \wedge \neg \mathbf{bm}_k) \end{aligned}$$

Therefore,

$$\begin{aligned} \text{pt}(S_{\text{qenc}}, (f(\mathbf{xm}_1 = c_1))) \\ = ((\frac{1}{2})(f(\mathbf{xm}_2 = c_1) \wedge \mathbf{tt})) + (1 - \frac{1}{2})(f(\mathbf{xm}_2 = c_1) \wedge \mathbf{ff})) + \\ (\frac{1}{2})(f(\mathbf{xm}_1 = c_1) \wedge \neg \mathbf{tt})) + (1 - \frac{1}{2})(f(\mathbf{xm}_1 = c_1) \wedge \neg \mathbf{ff})). \end{aligned}$$

Also, $\text{pt}(S_{\text{qenc}}, \frac{1}{2})$ is $\frac{1}{2}$ by Proposition 6.1. The Hoare assertion Ψ now follows from the fact that

$$(((f\mathbf{tt}) = 1) \cap \eta_I) \supset (\text{pt}(S_{\text{qenc}}, (f(\mathbf{xm}_1 = c_1))) = \frac{1}{2})$$

is an EPPL theorem.

8 Related Work

The area of formal methods in probabilistic programs has attracted a lot of work ranging from logic-based reasoning [13,21,31,14,17,25,27,10] to semantics [20,19,33,26].

This 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 works based on truth-functional probabilistic state logic appeared in the context of dynamic logic [32,22,30,13,12]. In the context of probabilistic truth-functional dynamic logics, the state language has terms representing probabilities (*e.g.*, $(f\gamma)$ represents the probability of γ being true). An infinitary complete axiom system for probabilistic dynamic logic is given in [22]. Later, a complete finitary axiomatization of probabilistic dynamic logic was given in [13]. However, the state logic is second-order (to deal with iteration) and undecidable. In [12], 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 [31]. 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)

$$\{(f\text{tt}) = 1\} \text{ if } x = 0 \text{ then skip else skip } \{(f\text{tt}) = 1\}$$

is not provable in the logic. As noted in [31,21], 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 [10] 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 [31] and completeness is achieved for a fragment of the Hoare logic, it is not clear how to axiomatize the test construct and the probabilistic sum connective [10].

The other approach to dynamic logic uses arithmetical state logic instead of truth-functional state logic [21,19,18,25]. For example, instead of the if-then-else construct, the programming language in [21] has the construct $\gamma?s_1 + (\neg\gamma)?s_2$ which is closely bound to the forward denotational semantics proposed in [20]. 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 $\text{wp}(g, s)$ 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 $\text{wp}(g, s)$. The important result in this area is the duality between the forward semantics and the backwards semantics [18].

Later, [25] 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, [25] 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.

$$\{r\} \text{toss}(\mathbf{bm}, r) \{\mathbf{bm}\}$$

In the pre-condition, r is the constant function r , and \mathbf{bm} is the function that takes value 1 when \mathbf{bm} is true and 0 otherwise. The above Hoare assertion states that the probability of \mathbf{bm} being true after the probabilistic toss is at least r . The Hoare rule for probabilistic tosses in the context of arithmetical Hoare logics takes the form

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

The problem of alternative of if-then-else construct is tackled in [8] by marking the choices at the end of the execution. However, our proof of completeness shows that this is not needed and variables in the state logic are sufficient to account for individual contributions to the measure terms ($f\gamma$).

Our state logic itself is the probabilistic logic in [11] extended with variables that aid in the proof of completeness of the Hoare logic. The logic is designed by the exogenous semantics approach to probabilistic logics [28,29,11,1,24]. A second difference is that we also allow products in terms. The probability logic in [11] does not have general product terms and allows only products with constants. The constants are rational numbers and this makes the logic NP-complete. We can also keep this restriction in our state assertion language.

The main distinction between the state logic herein and the logic in [8] is that we do not distinguish between possibility and probability. The semantic structure in [24] also contains a set of possible valuations along with a probability measure with the restriction that impossible valuations are improbable. The formula $\Box\gamma$ is an atomic formula of the state logic in [8] and is true of a semantic structure if γ holds for all possible valuations. The conditional formula η/γ also appears as an atomic formula in [8]. It was then shown as a lemma that the conditional construct could be removed from the language without loss of expressivity (in other words, for each formula η there was a provably equivalent conditional-free formula η'). However, as we do not distinguish between probability and possibility, the conditional construct can be easily defined by recursion and is hence removed from the primitives of the state language.

9 Concluding remarks

Our main contribution is a complete and decidable probabilistic Hoare calculus with a truth-functional state assertion logic that enjoys recursive axiomatization.

The truth-functional state assertion logic is essentially the probability logic in [11] extended with variables that aid in the proof of the completeness of the Hoare logic. For the sake of convenience, we also assumed that the measures take values from an arbitrary *real closed field* instead of the set of real numbers. The first-order theory of real closed fields is complete for real numbers [16,4] and hence the results in this paper will still hold if we work only with real numbers.

The proof of completeness of the Hoare logic employs the standard technique of defining weakest preconditions. The algorithmic definition of weakest preconditions uses an auxiliary preterm operator. The decidability of the Hoare logic then follows from the decidability of the state logic.

There are several directions in which this work can be extended. First, the complexity analysis of both the state logic and Hoare logic needs to be carried out. This will entail the complexity analysis of the first-order theory of real closed fields⁸. We also plan to include the iteration construct and demonic non-determinism 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 [23,9], a sound Hoare logic for basic quantum imperative programs [7] and a sound quantum temporal logic [3].

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.

⁸ Some earlier results [5] claimed that the complexity of the decision procedure of satisfiability of first-order theory of real closed fields is PSPACE complete. However, in personal communication, Michael Ben-Or (one of the authors of [5]) informed us that the proof of this result has been called into question.

- [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] P. Baltazar, R. Chadha, P. Mateus, and A. Sernadas. Towards model-checking quantum security protocols. Technical report, CLC, Department of Mathematics, Instituto Superior Técnico, Lisboa, Portugal, 2006. Submitted for publication.
- [4] S. Basu, R. Pollack, and R. Marie-Françoise. *Algorithms in Real Algebraic Geometry*. Springer Verlag, 2003.
- [5] M. Ben-Or, D. Kozen, and J. Reif. The complexity of elementary algebra and geometry. *Journal of Computer and System Sciences*, 18:251–264, 1986.
- [6] C. Caleiro, P. Mateus, A. Sernadas, and C. Sernadas. Quantum institutions. In K. Futatsugi, J.-P. Jouannaud, and J. Meseguer, editors, *Algebra, Meaning, and Computation – Essays Dedicated to Joseph A. Goguen on the Occasion of His 65th Birthday*, volume 4060 of *Lecture Notes in Computer Science*, pages 50–64. Springer Verlag, 2006.
- [7] 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.
- [8] R. Chadha, P. Mateus, and A. Sernadas. Reasoning about states of probabilistic sequential programs. In *Computer Science Logic 2006 (CSL06)*, Lecture Notes in Computer Science. Springer Verlag, in print.
- [9] 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.
- [10] 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.
- [11] R. Fagin, J.Y. Halpern, and N. Megiddo. A logic for reasoning about probabilities. *Information and Computation*, 87(1–2):78–128, 1990.
- [12] Y.A. Feldman. A decidable propositional dynamic logic with explicit probabilities. *Information and Control*, 63(1/2):11–38, 1984.
- [13] Y.A. Feldman and D. Harel. A probabilistic dynamic logic. *Journal of Computer and System Sciences*, 28:193–215, 1984.
- [14] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6:512–535, 1995.
- [15] C. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12:576–583, 1969.

- [16] W. Hodges. *Model Theory*. Cambridge University Press, 1993.
- [17] 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.
- [18] C. Jones. *Probabilistic Non-Determinism*. PhD thesis, U. Edinburgh, 1990.
- [19] 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.
- [20] D. Kozen. Semantics of probabilistic programs. *Journal of Computer System Science*, 22:328–350, 1981.
- [21] D. Kozen. A probabilistic PDL. *Journal of Computer System Science*, 30:162–178, 1985.
- [22] J.A. Makowsky and M.L. Tiomkin. Probabilistic propositional dynamic logic, 1980. Manuscript.
- [23] P. Mateus and A. Sernadas. Weakly complete axiomatization of exogenous quantum propositional logic. *Information and Computation*, 204(5):771–794, 2006. ArXiv math.LO/0503453.
- [24] P. Mateus, A. Sernadas, and C. Sernadas. Exogenous semantics approach to enriching logics. In G. Sica, editor, *Essays on the Foundations of Mathematics and Logic*, volume 1 of *Advanced Studies in Mathematics and Logic*, pages 165–194. Polimetria, 2005.
- [25] C. Morgan, A. McIver, and K. Seidel. Probabilistic predicate transformers. *ACM Transactions on Programming Languages and Systems*, 18(3):325–353, 1996.
- [26] 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.
- [27] M. Narasimha, R. Cleaveland, and P. Iyer. Probabilistic temporal logics via the modal mu-calculus. In *Foundations of Software Science and Computation Structures (FOSSACS 99)*, volume 1578 of *Lecture Notes in Computer Science*, pages 288–305. Springer Verlag, 1999.
- [28] N.J. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28(1):71–87, 1986.
- [29] N.J. Nilsson. Probabilistic logic revisited. *Artificial Intelligence*, 59(1–2):39–42, 1993.
- [30] 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.
- [31] L.H. Ramshaw. *Formalizing the Analysis of Algorithms*. PhD thesis, Stanford University, 1979.

- [32] 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.
- [33] R. Tix, K. Keimel, and G.D. Plotkin. Semantic domains for combining probability and non-determinism. *Electronic Notes in Theoretical Computer Science*, 129:1–104, 2005.