Reasoning about probabilistic sequential programs*

L. Cruz-Filipe, R. Chadha, P. Mateus and A. Sernadas
SQIG – IT and IST, Portugal
lcf,rch,pmat,acs@math.ist.utl.pt
September 29, 2010

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.

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 [?, ?, ?] and dynamic logics [?, ?] have been developed building upon denotational semantics of probabilistic programs [?]. The second approach enriches temporal modalities with probabilistic bounds [?, ?, ?].

Our work is in the area of Hoare-style reasoning about probabilistic sequential programs. A Hoare assertion [?] is a triple of the form \{\xi_1\} s \{\xi_2\} meaning that if program s starts in state satisfying the state assertion formula \(\xi_1\) and s halts then s ends in a state satisfying the state transition formula \(\xi_2\). The formula \(\xi_1\) is known as the pre-condition and the formula \(\xi_2\) is known as the post-condition. For probabilistic programs the development of Hoare logic has taken primarily two different paths. The common denominator of the two approaches is forward denotational semantics of sequential probabilistic programs [?]; 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 [?] using a truth-functional state assertion language, i.e., the formulas of the logic are interpreted as either true and false and the truth value of a formulas is determined by the truth values of the sub-formulas. This state assertion language consists of two levels: i) classical state formulas \(\gamma\) interpreted over the

*Supported by FCT and FEDER through POCI via CLC QuantLog POCI/MAT/55796/2004 Project. Additional support for Luis Cruz-Filipe and Rohit Chadha came from FCT and FEDER grants SFRH/BPD/16372/2004 and SFRH/BPD/26137/2005, respectively.
valuations of memory cells; and (ii) probabilistic state formulas $\xi$ interpreted over (sub)-probability measures of the valuations. The latter contains terms of the form $\langle f \gamma \rangle$ representing probability of $\gamma$ being true. But, the language at the probabilistic level is extremely restrictive and is built from term equality using conjunction. Furthermore, the Hoare rule for the alternative if-then-else is incomplete and even simple valid assertions may not be provable. The reason for incompleteness of the Hoare rule for the alternative composition in [?] as observed in [?, ?] 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 [?] 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 [?], there are several important works in the probabilistic Hoare logic, e.g. [?, ?], in which the state formulas are either measurable functions or arithmetical formulas interpreted as measurable functions. Intuitively, the Hoare triple $\{f\} s \{g\}$ means that the expected value of the function $g$ after the execution of $s$ is at least as much as the expected value of the function $f$ before the execution. Although research in probabilistic Hoare logic with arithmetical state logics has yielded several interesting results, the Hoare triples themselves do not seem very intuitive. A high degree of sophistication is required to write down the Hoare assertions needed to verify relatively simple programs.

For this reason, it is worthwhile to investigate Hoare logics with truth-functional state logics. A sound Hoare logic with a truth-functional state logic was presented in [?] and completeness for a fragment of the Hoare-logic is shown for iteration-free programs. In order to deal with alternative composition, a probabilistic sum construct $(\xi_1 + \xi_2)$ is introduced in [?]. Intuitively, the formula $(\xi_1 + \xi_2)$ is satisfied by a (sub)-probability measure $\mu$ if $\mu$ can be written as the sum of two measures $\mu_1$ and $\mu_2$ which satisfy $\xi_1$ and $\xi_2$ respectively. The drawback of [?] is that no axiomatization is given for the state assertion logic. The essential obstacle in achieving a complete axiomatization for the state language in [?] is the probabilistic sum construct.

This paper addresses the gap between [?] and [?] and provides a complete and decidable Hoare logic for iteration-free probabilistic programs using a truth-functional probabilistic state assertion logic. The Hoare calculus herein was originally proposed in [?] where only its soundness was established. The contribution of this paper is the completeness and decidability of the Hoare calculus.

Following [?], we tackle the Hoare rule for the alternative composition of programs using two key ingredients.

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

Second, we have a conditional construct \((\xi/\gamma)\) in our state assertion language. Intuitively, the formula \((\xi/\gamma)\) is satisfied by a (sub)-probability measure \(\mu\) if \(\xi\) is true of the (sub)-probability measure obtained by eliminating the measure of all valuations where \(\gamma\) is false. The conditional formulas \((\xi/bm)\) and \((\xi/(\neg bm))\) in the state logic can then be used to combine information of the alternative paths.

Our probabilistic state assertion logic, henceforth referred to as Exogenous Probabilistic Propositional Logic (EPPL), is designed by taking the exogenous semantics approach \([?]\) to enriching a given logic – the models of the enriched logic are sets of models of the given logic with some additional structure. A semantic model of EPPL is a set of possible valuations over memory cells which may result from execution of a probabilistic program along with a discrete (sub)-probability space which gives the probability of each possible valuation. For the sake of convenience, we work with finitely additive, discrete and bounded measures and not just (sub)-probability measures. In order to achieve recursive axiomatization for EPPL, 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 \([?, ?]\), and this technique of achieving decidability was inspired by other work in probabilistic reasoning \([?, ?]\).

Like in \([?]\), there are two levels of formulas in EPPL: classical state formulas \(\gamma\) interpreted over individual valuations and probabilistic state formulas \(\xi\) interpreted over the models of EPPL. Terms \(p\) in the language at probabilistic level represent elements of a real closed field and the probability of \(\gamma\) being true is represented by the term \((\int \gamma)\). There are two probabilistic atomic formulas: \(\Box \gamma\) meaning \(\gamma\) is necessarily true in all the possible valuations, and \(p_1 \leq p_2\) meaning that the term \(p_1\) is less than the term \(p_2\). Please note that \(\Box \gamma\) is not a full fledged modality since it cannot be nested. Probabilistic state formulas are built from probabilistic atoms using the disjunctive connectives \(\lor\) and \(\lor\), and a conditional construct \(\xi/\gamma\). The formula \((\xi/\gamma)\) is satisfied in a model of EPPL if \(\xi\) is true of the model obtained by restricting the set of possible valuations to the set where \(\gamma\) is true and eliminating the measure of all valuations where \(\gamma\) is false.

Unlike most works on probabilistic reasoning about programs, we do not confuse possibility with probability: possible valuations may occur with zero probability. This is not a restriction and we can confuse the two, if desired, by adding an axiom to the proof system. On the other hand, this separation yields more expressivity. The exogenous approach to probabilistic logics first appeared in \([?, ?]\) and later in \([?, ?]\). The general exogenous mechanism for building new logics is described in detail in \([?, ?]\) and used for developing quantum logics in \([?, ?]\). EPPL is an enrichment of the probabilistic logic proposed in \([?]\): the conditional construct \((\xi/\gamma)\) is not present in \([?]\). The proof of the completeness and dedidability of EEPL capitalizes on the completeness of the logic in \([?]\). The axioms for \((\xi/\gamma)\) provide an algorithm for eliminating such conditional formulae.

The programming language is a basic imperative language with assignment to memory variables, sequential composition, probabilistic assignment \((\text{toss}(bm, r))\) and the marked alternative choice. The statement \(\text{toss}(bm, r)\) assigns \(bm\) to true with probability \(r\). The term \(r\) is a constant and does not depend on the state of the program. This is not a serious restriction. For instance, \(r\) is taken to be \(\frac{1}{2}\) in probabilistic Turing machines.
One of the novelties of our Hoare logic is the rule for \texttt{toss}(\texttt{bm}, r) which provides the weakest pre-condition and is not present in other probabilistic Hoare logics based upon truth-functional state logics. The corresponding rule in the arithmetical setting is discussed in Section ??.

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 \textit{weakest precondition} operator \texttt{wp}(\cdot, \cdot) assigning to each program \(s\) and each formula \(\xi\) a new state formula \texttt{wp}(s, \xi) corresponding to the weakest logical property that a state must satisfy to ensure that \(\xi\) holds after execution of \(s\). We then show that, for any program \(s\) and formula \(\xi\), the Hoare calculus derives the judgement \(\{\texttt{wp}(s, \xi)\} s \{\xi\}\); in other words, \texttt{wp}(s, \xi) is a sufficient precondition for \(s\) and \(\xi\). The proof concludes after showing that \((V, \mathcal{K}, \mu)\models \texttt{wp}(s, \xi)\) iff \(\models [s](V, \mathcal{K}, \mu)\models \xi\).

The rest of the paper is organized as follows. The syntax, semantics and the complete recursive axiomatization of EPPL is presented in Section ??.

2 Logic of probabilistic states - EPPL

The state logic presented herein is the probability logic proposed in [?]? extended with variables that assist in the proof of completeness of the Hoare calculus. We assume that 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. Please 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 and such an assignment is henceforth called a \textit{valuation}. If we denote the set of valuations by \(V\) then intuitively a semantic structure of EPPL is a finitely additive, discrete and bounded measure \(\mu\) on \(\wp V\), the power-set of \(V\). A finitely additive, discrete and bounded measure \(\mu\) on \(\wp V\) is a map from \(\wp V\) to \(\mathbb{R}^+\) (the set of non-negative real numbers) such that:

\begin{itemize}
  \item \(\mu(\emptyset) = 0\); and
  \item \(\mu(U_1 \cup U_2) = \mu(U_1) + \mu(U_2)\) if \(U_1 \cap U_2 = \emptyset\).
\end{itemize}

Loosely speaking, \(\mu(U)\) denotes the probability of a possible valuation being in the set \(U\). A measure \(\mu\) is said to be a probability measure if \(\mu(V) = 1\). We work with general measures instead of just probability measures as it is convenient to do so.
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 $\mathcal{K} = (\mathbb{K}, +, \cdot, 1, 0, \leq)$ is said to be a real closed field if the following hold:

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

Examples of real closed fields include the set of real numbers with the usual multiplication, addition and order relation. The set of computable real numbers with the same operations is another example. A measure that takes values from a real closed field $\mathcal{K}$ will henceforth be called a $\mathcal{K}$-measure. Any real closed field has a copy of the integers and the rationals. We can also take roots of positive elements and odd $n$-roots in a real closed field. In general, any real algebraic number is definable in a real closed field. We shall denote by $\mathcal{A}$ the set of real algebraic numbers. We shall use them as constants in probability terms of our logic.

A semantic structure of EPPL thus 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 ?? using the BNF notation and discussed below.

| Real terms (with the proviso $c \in D$) | $t \ := \ \text{xm} \ [X \ c \ (t + t) \ \text{tt}]$ |
| Classical state formulas | $\gamma \ := \ \text{bm} \ [B \ (t \leq t) \ \text{ff} \ (\gamma \Rightarrow \gamma)]$ |
| Probability terms (with the proviso $r \in \mathcal{A}$) | $p \ := \ \text{yr} \ (f \gamma) \ (p + p) \ (p) \ \text{rr}$ |
| Probabilistic state formulae | $\eta \ := \ (p \leq p) \ \text{ff} \ (\eta \supset \eta)$ |

**Table 1: Language of EPPL**

Given fixed $m = \{0, \ldots, m - 1\}$, there are two finite disjoint sets of memory variables: $\text{xM} = \{\text{xm}_k : k \in m\}$ – representing the contents of real registers, and
bM = \{bm_k : k \in 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: B = \{B_k : k \in \mathbb{N}\} – ranging over the truth values in \(2 = \{\text{ff}, \text{tt}\}\), and \(X = \{X_k : k \in \mathbb{N}\} – ranging over elements of D. Please note that the special case in which these random variables behave deterministically except on a set of measure zero can be expressed in the logic as we shall explain later on (at the end of Subsection ??). Therefore, we can used 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 \(t, t_1, \ldots\), are built from the sets \(D\), \(\times M\) and \(X\) using the usual addition and multiplication\(^1\). The classical state formulas, ranged over by \(\gamma, \gamma_1, \ldots\), are built from \(bM, B\) and comparison formulas \((p_1 \leq p_2)\) using the classical disjunctive connectives \(\text{ff}\) and \(\Rightarrow\). As usual, other classical connectives \((\neg, \vee, \land, \Leftrightarrow, \text{tt})\) are introduced as abbreviations. For instance, \((\neg \gamma)\) stands for \((\gamma \Rightarrow \text{ff})\).

The probability terms, ranged over by \(p, p_1, \ldots\), denote elements of the real closed field in a semantic structure. We also assume a set of (rigid and deterministic) logical variables, \(Y = \{y_k : k \in \mathbb{N}\}\), ranging over elements of the real closed field. These logical variables were not present in \(\mathcal{?}\) and 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 \(\gamma\). The terms of the kind \((f \gamma)\) shall henceforth be called measure terms. We denote the set of all probability terms by \(\mathcal{P} \text{Terms}\).

The probabilistic state formulas, ranged over by \(\eta, \eta_1, \ldots\), are built the comparison formulas \((p_1 \leq p_2)\) using the connectives \(\text{ff}\) and \(\circ\). Other probabilistic connectives \((\circ, \cup, \cap, \approx, \text{tt})\) and comparison operators \((=, \geq, <, >)\) are introduced as abbreviations in the classical way. For instance, \((\circ \eta)\) stands for \((\eta \circ \text{ff})\) 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 \(\mathcal{P} \text{Forms}\).

It is also convenient for applications to introduce as an abbreviation the formula \((\square \gamma)\) which stands for the formula \(((f \gamma) = (f \text{tt}))\). Intuitively, \(\square \gamma\) is true if the set of the valuations where \(\gamma\) does not hold has measure zero. We shall also use \((\Diamond \gamma)\) as an abbreviation for \((\circ(\square(\neg \gamma)))\). Intuitively, \(\Diamond \gamma\) is true if the set of valuations where \(\gamma\) holds has non-zero measure. We shall see in section ?? that \(\square\) and \(\Diamond\) somewhat behave as necessity and possibility modalities. Please note, however, that the \(\square\) and \(\Diamond\) are not full fledged modalities since they cannot be nested\(^2\).

The notion of occurrence of a term \(p\) and a probabilistic state formula \(\eta\), in the probabilistic state formula \(\eta\) can be easily defined. The notion of replacing zero or more occurrences of probability terms and probabilistic formulas can also be suitably defined. The set of variables \(y \in Y\) occurring in a term \(p\) and a formula \(\eta\) will be denoted by \(\mathcal{P} \text{Var}(p)\) and \(\mathcal{P} \text{Var}(\eta)\). For the sake of clarity,

\(^1\)The arithmetical operations addition and multiplication are assumed to be defined so as to restrict them to the range \(D\).

\(^2\)We do not have formulas such as \(\square(\square \gamma)\).
we shall often drop parentheses in formulas and terms if it does not lead to ambiguity.

We shall also identify here a useful sub-language of probabilistic state formulas which do not contain any occurrence of a measure term:
\[
\kappa := (a \leq a) \hspace{1mm} \text{iff} \hspace{1mm} (\kappa \supset \kappa) \\
\alpha := x \hspace{1mm} \text{iff} \hspace{1mm} (\alpha + a) \hspace{1mm} \text{iff} \hspace{1mm} (aa) \hspace{1mm} \text{iff} \hspace{1mm} \tilde{\gamma}.
\]

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

### 2.2 Semantics

By a valuation we mean a map that provides values to the memory variables and corresponding logical variables—\( v : (xM \to D, bM \to 2, X \to D, B \to 2) \). 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 \( v \models_C \gamma \) are defined inductively as expected. Given \( V \subseteq \mathcal{V} \), the extent of \( \gamma \) in \( V \) is defined as \( |\gamma|_V = \{ v \in V : v \models_C \gamma \} \).

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

Given a classical formula \( \gamma \) we also need the following sub-measure of \( \mu \):
\[
\mu_{\gamma} = \lambda V. \mu(|\gamma|_V).
\]

That is, \( \mu_{\gamma} \) is null outside of the extent of \( \gamma \) and coincides with \( \mu \) inside it.

For interpreting the probabilistic variables \( y \in \mathcal{Y} \), we need the concept of an assignment. Given a real closed field \( \mathcal{K} \), a \( \mathcal{K} \)-assignment \( \rho \) is a map from \( \mathcal{Y} \) to \( \mathcal{K} \).

Given a generalized state \((\mathcal{K}, \mu)\) and a \( \mathcal{K} \)-assignment \( \rho \), the denotation of probabilistic terms and satisfaction of probabilistic state formulas are defined inductively in Table ???. The formula \((p_1 \leq p_2)\) is satisfied if the term denoted by \( p_1 \) is less than \( p_2 \). The formula \((|\eta_1 \supset \eta_2|)\) is satisfied by a semantic model if either \( \eta_1 \) is not satisfied by the model or \( \eta_2 \) is satisfied by the model. Please observe that the probabilistic connectives behave like the classical ones. Please note that the \( \mathcal{K} \)-assignment \( \rho \) is sufficient to interpret an analytical formula, i.e., a probabilistic formula without measure terms.

Entailment is defined as usual: \( \Lambda \) entails \( \eta \) (written \( \Lambda \models \eta \)) if \((\mathcal{K}, \mu)\rho \models \eta \) whenever \((\mathcal{K}, \mu)\rho \models \eta_0 \) for each \( \eta_0 \in \Lambda \). Meta-theorem of entailment holds, i.e., \( \Lambda, \eta \models \eta' \) iff \( \Lambda \models (\eta \supset \eta') \).

Please note that we can also define the probabilistic sum construct similar to the one defined in [?]. We may say that \((\mathcal{K}, \mu)\rho \models \eta_1 + \eta_2 \) if there exist \( \mu_1 \) and \( \mu_2 \) such that \( \mu = \mu_1 + \mu_2 \), \((\mathcal{K}, \mu_1)\rho \models \eta_1 \) and \((\mathcal{K}, \mu_1)\rho \models \eta_2 \). However, as already observed in Section ??, it is not obvious how to axiomatize this construction.

Please recall the derived formula \((\square \gamma) \) defined in Section ?? as \((\diamond \gamma) = (f \circ t)\). Clearly, the semantic model \((\mathcal{K}, \mu)\rho \) satisfies \((\diamond \gamma) \) iff \( \mu(|\gamma|_V) = \mu(V) \) iff \( \mu(|\gamma|_V) = 0 \). Similarly, \((\mathcal{K}, \mu)\rho \) satisfies \((\diamond \gamma) \) defined as \((\circ \diamond (\neg \gamma)) \) iff \( \mu(|\gamma|_V) > 0 \).

It follows easily from semantics that \( \models (\diamond (\gamma_1 \land \gamma_2)) \approx ((\diamond \gamma_1) \cap (\diamond \gamma_2)) \). Hence, \( (\diamond \gamma) \) behaves as necessity modality. Similarly, \( (\diamond \gamma) \) behaves as possibility modality, i.e., \( \models (\diamond (\gamma \lor \gamma_2)) \approx ((\diamond \gamma_1) \cup (\diamond \gamma_2)) \). Please note that it is
Denotation of probability terms
\[
[r]_{r(K,\mu)} = r \\
[y]_{\rho(K,\mu)} = \rho(y) \\
[(\gamma)]_{\rho(K,\mu)} = \mu(\gamma) \\
[p_1 + p_2]_{\rho(K,\mu)} = [p_1]_{\rho(K,\mu)} + [p_2]_{\rho(K,\mu)} \\
[p_1 p_2]_{\rho(K,\mu)} = [p_1]_{\rho(K,\mu)} \times [p_2]_{\rho(K,\mu)}
\]

Satisfaction of probabilistic formulas
\[
(K,\mu) \models (p_1 \leq p_2) \iff ([p_1]_{\rho(K,\mu)} \leq [p_2]_{\rho(K,\mu)}) \\
(K,\mu) \not\models \text{fff} \iff \text{true} \\
(K,\mu) \models (\eta_1 \supset \eta_2) \iff (K,\mu) \models \eta_2 \text{ or } (K,\mu) \not\models \eta_1
\]

Table 2: Semantics of EPPL

not the case that \( \models (\Box \gamma \supset \Diamond \gamma) \). Consider for instance, the generalized probabilistic state \((K,\mu)\) such \(\mu\) is identically zero. In this case for all classical state formula \(\gamma\), \((K,\mu)\) satisfies \(\Box \gamma\) but not \(\Diamond \gamma\).

Returning to the random nature of our logical variables in \(X\) and \(B\), 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, one of valid state formula, a second one of probabilistic tautology and the third of valid analytical formula.

A classical state formula \(\gamma\) is said to be valid if it is true of all valuations \(v \in V\). 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 \(\bot\) and \(\rightarrow\). A probabilistic formula \(\eta\) is said to be a probabilistic tautology if there is a propositional tautology \(\beta\) over \(Q\) and a map \(\sigma\) from \(Q\) to the set of probabilistic state formulas such that \(\eta\) coincides with \(\beta_{\rho} \sigma\) where \(\beta_{\rho} \sigma\) is the probabilistic formula obtained from \(\beta\) by replacing all occurrences of \(\bot\) by \(\text{fff}\), \(\rightarrow\) by \(\supset\) and \(q \in Q\) by \(\sigma(q)\). For instance, the probabilistic formula \(((y_1 \leq y_2) \supset (y_1 \leq y_2))\) is tautological (obtained, for example, from the propositional tautology \(q \rightarrow q\)).

As noted in Section ??, if \(K_0\) is the real closed field in a generalized probabilistic structure, then a \(K_0\)-assignment is enough to interpret all analytical formulas. We say that \(\kappa\) is a valid analytical formula if for any real closed field \(K\) and any \(K\)-assignment \(\rho\), \(\kappa\) is true for \(\rho\). Clearly, a valid analytical formula holds for all semantic structures of EPPL. It is a well-known fact from the theory of quantifier elimination [? , ?] 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 1 and better understood in the following groups.

### Axioms

- **[CTaut]**: \( \vdash (\square \gamma) \) for each valid state formula \( \gamma \)
- **[PTaut]**: \( \vdash \eta \) for each probabilistic tautology \( \eta \)
- **[RCF]**: \( \vdash \kappa_{\vec{y}, \vec{p}} \) where \( \kappa \) is a valid analytical formula, \( \vec{y} \) and \( \vec{p} \) are sequences of probability variables and probability terms respectively
- **[Meas\emptyset]**: \( \vdash ((\int ff) = 0) \)
- **[FAdd]**: \( \vdash (((\int (\gamma_1 \land \gamma_2)) = 0) \supset ((\int (\gamma_1 \lor \gamma_2)) = (\int \gamma_1) + (\int \gamma_2)) \)
- **[Mon]**: \( \vdash ((\square (\gamma_1 \Rightarrow \gamma_2)) \supset ((\int \gamma_1) \leq (\int \gamma_2)) \)

**Inference rule**

- **[PMP]**: if \( \eta_1, (\eta_1 \supset \eta_2) \vdash \eta_2 \)

### Table 3: Axioms for EPPL

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

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

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

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

As usual we say that a set of formulas \( \Lambda \) *derives* \( \eta \), written \( \Lambda \vdash \eta \), if we can build a derivation of \( \eta \) from axioms and the inference rules using formulas in \( \Lambda \) as hypothesis. It can be easily shown that the meta-theorem of deduction holds that is if \( \Lambda, \eta_1 \vdash \eta_2 \) if \( \Lambda \vdash (\eta_1 \supset \eta_2) \).

For this paper, we shall be only concerned with judgments \( \Lambda \vdash \eta \) where \( \Lambda \) is a finite set. Since, both meta-theorems of entailment and deduction hold in for EPPL, it suffices to consider judgments where \( \Lambda \) 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 the
definition of the semantics.

The proofs of completeness and decidability of EPPL go hand-in-hand and
essentially follows the lines of the proof of completeness in [?, ?]. The main
ingredient is the model existence lemma: if a probabilistic formula \( \eta \) is consis-
tent, i.e. \( \not \models (\ominus \eta) \), then there is a model that satisfies \( \eta \). 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 [?] 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 \( \eta \) is consistent (that is, \( \not \models (\ominus \eta) \)) then there
is a model \( (K, \mu) \) such that \( (K, \mu) \models \eta \). The decidability follows by showing
that the consistency of a formula is decidable.

The proof in [?, ?] 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 \( B \) and \( X \) occurring in \( \eta \) and let this set of valuations be \( V \); (ii) let \( \kappa_1 \) be
the analytical formula obtained from \( \eta \) by effectively replacing measure terms
\( (\int \gamma) \) by sums \( \sum_{v \in \mathcal{V}} y_v \) where \( y_v \) represents the probability of the valuation
\( v \); (iii) let \( \kappa \) be the analytical formula \( (\int \kappa) \); (iv) \( \eta \) is consistent
iff \( \kappa \) is; (v) finally, consistency of \( \kappa \) is decided by the axiom RCF and the model
is constructed for a consistent \( \kappa \) by solving for \( y_v \) in real closed fields.

3  Basic probabilistic sequential programs
We shall now describe briefly the syntax and semantics of our basic programs.

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

\[
\bullet \quad s ::= \text{skip} \mid \text{xm} \leftarrow t \mid \text{bm} \leftarrow \bar{\gamma} \mid \text{toss(bm, r)} \mid s; s \mid \text{if } \gamma \text{ then } s \text{ else } s.
\]

The statement \( \text{skip} \) does nothing. The statement \( \text{xm} \leftarrow t \) assigns the memory
cell \( \text{xm} \) the value denoted by \( t \) and the statement \( \text{bm} \leftarrow \bar{\gamma} \) assigns the cell \( \text{bm} \)
the truth value of \( \bar{\gamma} \). For the rest of the paper, by an expression we shall mean
either the terms \( t \) or the classical state formulas \( \gamma \). Please note that \( t \) and \( \gamma \) may
contain variables in the set \( X \) (which may be thought of as input to a program).

The statement \( \text{toss(bm, r)} \) sets \( \text{bm} \) true with probability \( \bar{r} \). The command \( s; s \)
is sequential composition. The statement \( \text{if } \gamma \text{ then } s_1 \text{ else } s_2 \) is the alternative
choice: if \( \gamma \) is true then \( s_1 \) is executed else \( s_2 \) is executed.

Please note that for any \( k \in \mathbb{N} \), one may introduce bounded iteration as an
abbreviation:

\[
(\text{while}^k \gamma \text{ do } s) \text{ for } (\text{if } \gamma \text{ then } s \text{ else skip})^k.
\]
3.2 Semantics

The semantics of the programming language is basically the forward semantics in \( [\gamma]_v \) adapted to our programming language. Given \( G \), the set of generalized probabilistic states, the denotation of a program \( s \) is a map \([s] : G \to G\) defined inductively in Table 4.

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

\[
\begin{align*}
[\text{skip}] & = \lambda(K, \mu). (K, \mu) \\
[\text{xm} \leftarrow t] & = \lambda(K, \mu). (K, \mu \circ (\delta_{\text{tm}})^{-1}) \\
[\text{bm} \leftarrow \gamma] & = \lambda(K, \mu). (K, \mu \circ (\delta_{\text{bm}})^{-1}) \\
[\text{toss}(bm, r)] & = \lambda(K, \mu). ((1 - \tilde{r}) ([bm \leftarrow ff](K, \mu)) + \tilde{r} ([bm \leftarrow tt](K, \mu))) \\
[s_1; s_2] & = \lambda(K, \mu). ([s_1](K, \mu)) + [s_2](K, \mu_{(-\gamma)}) \\
[\text{if } \gamma \text{ then } s_1 \text{ else } s_2] & = \lambda(K, \mu). ([s_1](K, \mu_{\gamma}) + [s_2](K, \mu_{(-\gamma)}))
\end{align*}
\]

Table 4: Denotation of programs

4 Probabilistic Hoare logic

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

- \( \Psi := \eta \{ \eta \} s \{ \eta \} \).

Satisfaction of Hoare assertions is defined as

- \( (K, \mu) \rho \vdash_h \eta \) if \( (K, \mu) \rho \vdash \eta \),
\( (\mathcal{K}, \mu) \) \( \models_h \{ \eta_1 \} \) \( \models \{ \eta_2 \} \) if \( (\mathcal{K}, \mu) \models \eta_1 \) implies \( [s](\mathcal{K}, \mu) \models \eta_2 \).

We say that a Hoare assertion \( \Psi \) is \textit{semantically valid} (written \( \models \Psi \)) if \( (\mathcal{K}, \mu) \models_h \Psi \) for every generalized probabilistic state \( (\mathcal{K}, \mu) \) and any \( \mathcal{K} \)-assignment \( \rho \).

4.1 Calculus

We shall now give a sound and complete axiomatization of the Hoare calculus. Please note that we shall only consider judgments of the form \( \models \Psi \), \textit{i.e.}, judgments with no hypothesis. Hence, in all inference rules the premises are assumed to be theorems of the Hoare calculus. We need some new concepts for the axiomatization: \textit{tossed terms}, \textit{tossed formulas}, \textit{conditional terms} and \textit{conditional formulas}.

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

<table>
<thead>
<tr>
<th>Tossed terms</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{toss}(bm, r; r') )</td>
<td>( = r' )</td>
<td></td>
</tr>
<tr>
<td>( \text{toss}(bm, r; y) )</td>
<td>( = y )</td>
<td></td>
</tr>
<tr>
<td>( \text{toss}(bm, r; (f \gamma)) )</td>
<td>( = ((1 - \tilde{\gamma})(\gamma_{bm}^f) + \tilde{\gamma}(\gamma_{tt}^f)) )</td>
<td></td>
</tr>
<tr>
<td>( \text{toss}(bm, r; (p + p')) )</td>
<td>( = (\text{toss}(bm, r; p) + \text{toss}(bm, r; p')) )</td>
<td></td>
</tr>
<tr>
<td>( \text{toss}(bm, r; (pp')) )</td>
<td>( = (\text{toss}(bm, r; p) \text{toss}(bm, r; p')) )</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tossed formulas</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{toss}(bm, r; fff) )</td>
<td>( = fff )</td>
<td></td>
</tr>
<tr>
<td>( \text{toss}(bm, r; (p \leq p')) )</td>
<td>( = (\text{toss}(bm, r; p) \leq \text{toss}(bm, r; p')) )</td>
<td></td>
</tr>
<tr>
<td>( \text{toss}(bm, r; (\eta \supset q')) )</td>
<td>( = (\text{toss}(bm, r; \eta) \supset \text{toss}(bm, r; q')) )</td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Tossed terms and formulas

Given a classical state formula \( \gamma \) and a probabilistic term \( p \in \text{PTerms} \) we define the \( \gamma \)-\textit{conditioned term}, \( (p/\gamma) \), to be the term obtained from \( p \) by replacing \textit{every} occurrence of \textit{each} measure term \( (f \gamma') \) by \( (f(\gamma' \land \gamma)) \). Similarly, we define the probabilistic formula \( \eta/\gamma \) to be the formula obtained from \( \eta \) by replacing \textit{every} occurrence of \textit{each} measure term \( (f \gamma') \) by \( (f(\gamma' \land \gamma)) \). Formally, the recursive definition of \( (\cdot)/\gamma \) is given in Table 6. Please note that the recursive definition also gives a recursive algorithm for computing \( p/\gamma \) and \( \eta/\gamma \). Also, given two probabilistic formulas \( \eta_1 \) and \( \eta_2 \), we shall use \( (\eta_1 \gamma \eta_2) \) as an abbreviation for \( ((\eta_1/\gamma) \cap (\eta_2/(\neg \gamma))) \).
Conditional terms

\[
\begin{align*}
    r/\gamma &= r \\
    y/\gamma &= y \\
    (f/\gamma)/\gamma &= (f(\gamma \wedge \gamma')) \\
    (p + p)/\gamma &= (p/\gamma + p'/\gamma) \\
    (pp'/\gamma) &= ((p/\gamma) (p'/\gamma)) \\
\end{align*}
\]

Conditional formulas

\[
\begin{align*}
    \begin{aligned}
        s \leq s' \quad &\text{iff} \\
        \eta \supset \eta' \quad &\text{iff}
    \end{aligned}
\end{align*}
\]

Table 6: Conditional terms and formulas

A sound and complete Hoare calculus for our probabilistic sequential programs is given in Table ???. 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. We discuss the others briefly here.

Please recall that an analytical formula is a probabilistic formula that does not contain any measure terms (i.e. terms of the kind \(\int \gamma\)). Since an analytical formula does not contain any memory cells, a statement does not change the truth value of an an analytical formula \(\kappa\). This fact is reflected in the axiom ANAL\(^3\).

In the axioms ASGR and ASGB, the notation \(\eta^m\) means the formula obtained from \(\eta\) by replacing all occurrences (i.e., those measure terms) of the memory variable \(m\) by the expression \(e\). The axioms ASGR and ASGB are reminiscent of the Hoare rules for assignment in the case of deterministic sequential programs. The axiom TOSS gives the Hoare for probabilistic tosses.

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

The inference rule ELIMV eliminates variables in the set \(Y\). Please note that in this rule \(\eta\) does not have any conditional constructs and the variable \(y\) does not occur either in the probabilistic term \(p\) in the post-condition \(\eta\). This inference rule is essential for proving the completeness of Hoare logic and is not present in [?]. This inference rule can be viewed as a special case of the inference rule for existential quantifiers in first-order Hoare logic. The rule for existential quantifiers in first-order Hoare logic for deterministic programs is often stated as:

\[
\{\varphi\} s \{\psi\} \vdash \{\exists z. \varphi\} s \{\psi\} \quad \text{if } z \text{ does not occur in } \psi.
\]

\(^3\)Actually, the axiom is only needed in the case \(s\) is an alternative statement. It can be derived in other cases by induction.
The inference rule ELIMV can then be viewed as a special instance of the rule for existential quantifiers by observing that the first-order formula \((\exists z. (\varphi(z) \land (z = r)))\) is equivalent to \((\exists z. (\varphi(r) \land (z = r)))\) which in turn is equivalent to \(\varphi(r)\) if \(z\) does not occur in \(r\).

## 5 Soundness of Hoare Logic

We shall now show that the Hoare calculus presented in [??] is sound, i.e., if \(\vdash \Psi\) then \(\models_\Pi \Psi\). It is sufficient to show that all the axioms and inference rules of the Hoare calculus are sound. We start by showing that the axioms for assignments to memory variables (ASGB and ASGR) are sound.

The proofs of soundness of the axioms ASGB and ASGR rely on substitution lemma for classical valuations. Please note that the substitution lemma for classical valuations is also the key ingredient for soundness of the axiom for assignments in deterministic sequential programs [??]. Please recall that the valuation \(v^m_{[e]}\) assigns the value \([e]_v\) to the cell \(m\) and coincides with the valuation \(v\) elsewhere. We have:

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

\[
v^m_{[e]} \models_c \gamma \iff v \models_c \gamma^m_{[e]}
\]
Proof: The proof is by induction on the structure of $\gamma$ and is similar to the one for deterministic sequential programs.

We shall now extend the substitution lemma for classical valuations to a substitution lemma for probabilistic terms and formula which will imply the soundness of ASGB and ASGR. Please recall that $\delta^m_e : V \rightarrow V$ is the map that takes each valuation $v$ to $v\{e\leftarrow m\}$. We have:

Lemma 5.2 (Substitution Lemma for assignment) Let $(K, \mu)$ be a generalized probabilistic structure and $\rho$ be a $K$-assignment. Given a memory cell $m$ and a term $e$ of the same type, let $\mu' = \mu \circ (\delta^m_e)^{-1}$. Then for any classical state formula $\gamma$:

$$[\langle f \gamma \rangle]_{(K, \mu')}^\rho = [\langle f \gamma^m_e \rangle]_{(K, \mu)}^\rho.$$ 

Furthermore, for any probabilistic term $p$, we have

$$[p]_{(K, \mu')}^\rho = [p^m_e]_{(K, \mu)}^\rho$$

and for any probabilistic formula $\eta$, we have

$$(K, \mu') \rho \models \eta \text{ iff } (K, \mu) \rho \models \eta^m_e.$$ 

Proof: Please note that as a consequence of Lemma 5.2, we have

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

Therefore, by definition,

$$[\langle f \gamma \rangle]_{(K, \mu')}^\rho = \mu(\delta^m_e)^{-1}(|\gamma|_V) = \mu(|\gamma^m_e|_V) = [\langle f \gamma^m_e \rangle]_{(K, \mu)}^\rho.$$ 

The result can then be extended to probabilistic terms and formulas by induction.

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 $(K, \mu)$ be a generalized probabilistic structure, $\rho$ be a $K$-assignment, $r \in A$ be a constant and

$$\mu' = \tilde{r} \mu \circ (\delta^m_{\text{tt}})^{-1} + (1 - \tilde{r}) \mu \circ (\delta^m_{\text{ff}})^{-1}.$$ 

Then, for any classical state formula $\gamma$:

$$[\langle f \gamma \rangle]_{(K, \mu')}^\rho = \tilde{r} [\langle f \gamma^m_{\text{tt}} \rangle]_{(K, \mu)}^\rho + (1 - \tilde{r}) [\langle f \gamma^m_{\text{ff}} \rangle]_{(K, \mu)}^\rho.$$ 

For any probabilistic term $p$, we have

$$[p]_{(K, \mu')}^\rho = [\text{toss}(m, r; p)]_{(K, \mu')}^\rho.$$ 

Furthermore, for any probabilistic formula $\eta$,

$$(K, \mu') \rho \models \eta \text{ iff } (K, \mu) \rho \models \text{toss}(m, r; \eta).$$
Proof: Let \( \mu_1 = \mu \circ (\delta_{\text{bbm}})^{-1} \) and \( \mu_2 = \mu \circ (\delta_{\text{bb}})^{-1} \). We have by definition:

\[
[(\bar{f} \gamma)]^\rho_{(K,\mu')} = \tilde{\gamma}[(\bar{f} \gamma)]^\rho_{(K,\mu)} + (1 - \tilde{\gamma})[(\bar{f} \gamma)]^\rho_{(K,\mu_2)}.
\]

We have by Lemma 2.4:

\[
[(\bar{f} \gamma)]^\rho_{(K,\mu)} = [(\bar{f} \gamma_{\text{bb}})]^\rho_{(K,\mu)} \quad \text{and} \quad [(\bar{f} \gamma)]^\rho_{(K,\mu_2)} = [(\bar{f} \gamma_{\text{bb}})]^\rho_{(K,\mu)}.
\]

Finally, the claim for probabilistic terms and probabilistic formulas can be proved by induction. \( \triangle \)

The following proposition asserts the soundness of the axiom ANAL:

**Proposition 5.4 (Soundness of ANAL)** For any statement \( s \), any analytical formula \( \kappa \), any generalized state \((K,\mu)\) and K-assignment \( \rho \),

\[
([s](K,\mu))\rho \models \kappa \iff (K,\mu)\rho \models \kappa. \]

**Proof:** The claim follows easily from the fact that interpretation of analytical depends only on the assignment \( \rho \). \( \triangle \)

The following proposition is used in the soundness of IF rule:

**Proposition 5.5** For any generalized state \( (K,\mu) \), a K-assignment \( \rho \) and two classical state formulas \( \gamma \) and \( \gamma' \), we have

\[
[(\bar{f} \gamma')/\gamma]^\rho_{(K,\mu)} = [(\bar{f} \gamma')]^\rho_{(K,\mu)}.
\]

Furthermore, for any probability term \( p \),

\[
[p/\gamma]^\rho_{(K,\mu)} = [p]^\rho_{(K,\mu)}
\]

and probabilistic formula \( \eta \)

\[
(K,\mu)\rho \models \eta/\gamma \iff (K,\mu_\gamma)\rho \models \eta.
\]

**Proof:** We have by definition,

\[
[(\bar{f} \gamma')]^\rho_{(K,\mu)} = \mu_\gamma(\{\gamma' \mid y\}) = \mu(\{\gamma' \mid y \cap \gamma \mid y\}) = \mu(\{\gamma' \land \gamma \mid y\}) = [(\bar{f} \gamma')/\gamma]^\rho_{(K,\mu)}.
\]

The claims for probabilistic terms and formulas now follow by induction. \( \triangle \)

The following Lemma asserts the soundness of the inference rule IF:

**Lemma 5.6 (Soundness of IF rule)** Given probabilistic state formulas \( \eta_1 \) and \( \eta_2 \), statements \( s_1 \) and \( s_2 \), variables \( y_1 \in \mathcal{Y} \) and \( y_2 \in \mathcal{Y} \), and a classical state formula \( \gamma \) such that

\[
\models \{\eta_1\} s_1 \{y_1 = (f \gamma)\} \quad \text{and} \quad \models \{\eta_2\} s_2 \{y_2 = (f \gamma)\}.
\]

Then for any classical state formula \( \gamma_0 \),

\[
\models \{\eta_1 \land \gamma_0 \land \eta_2\} \text{ if } \gamma_0 \text{ then } s_1 \text{ else } s_2 \{y_1 + y_2 = (f \gamma)\}.
\]
Proof: Let $(K, \mu)$ be a generalized probabilistic state and let $\rho$ be a $K$-assignment such that $(K, \mu)\rho \models \eta_1 \wedge \eta_2$. Therefore, $(K, \mu)\rho \models \eta_1 / \gamma_0$ and $(K, \mu)\rho \models \eta_2 / (-\gamma_0)$. Thus, $(K, \mu, \gamma_0)\rho \models \eta_1$ and $(K, \mu, (-\gamma_0))\rho \models \eta_2$ by Proposition ??.

Let $(K, \mu_1) \models [s_1](K, \mu_{\gamma_0})$, $(K, \mu_2) \models [s_2](K, \mu_{-\gamma_0})$ and let $\mu' = \mu_1 + \mu_2$.

We need to show that $(K, \mu')\rho \models (y_1 + y_2 = (\gamma))$.

Please observe that since $\models_{h} \{ \eta_1 \}$ $s_1 \{ y_1 = (f_\gamma) \}$ and $(K, \mu_{\gamma_0})\rho \models \eta_1$, we get $(K, \mu_1)\models_{h} y_1 = (\gamma)$. Thus, we have by definition, $\rho(y_1) = \mu_1(\gamma|_Y)$. Similarly, we have $\rho(y_2) = \mu_2(\gamma|_Y)$.

Hence, we get $\mu'(\gamma|_Y) = \mu_1(\gamma|_Y) + \mu_2(\gamma|_Y) = \rho(y_1) + \rho(y_2) = \rho(y_1 + y_2)$. Therefore, $(K, \mu')\rho \models (y_1 + y_2 = (\gamma))$ as required. △

Now, we shall show that the inference rule ELIMV is sound. In order to do this, we shall first establish a substitution result for variables $y \in Y$. Please note that for rest of the paper, given a $K$-assignment $\rho$, a variable $y \in Y$ and an element $k \in K$, the $K$-assignment $\rho^p_k$ denotes the assignment that assigns the value $k$ to $y$ and coincides with $\rho$ elsewhere.

Proposition 5.7 Let $y \in Y$ be a variable and $p$ be a probabilistic term. Given a general probabilistic structure $(K, \mu)$ and a $K$-assignment $\rho$, let $k = \overline{p}\rho_{(K, \mu)}$ and $\rho_1 = \rho^p_k$. Then,

1. for any probabilistic term $p_0$, $[p_0]_{(K, \mu)}^{\rho_1} = [p_0]_{(K, \mu)}^{\rho_2}$ and

2. for any probabilistic formula $\eta$, $(K, \mu)\rho_1 \models \eta$ iff $(K, \mu)\rho \models \eta^p_0$.

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 are straightforward. If $y_0$ is $y$ then by definition, $[y_0]_{(K, \mu)}^{\rho_1} = k = [y_0]_{(K, \mu)}^{\rho_2} = [y_0]_{(K, \mu)}^{\rho_2}$.

If $y_0$ is not $y$ then $[y_0]_{(K, \mu)}^{\rho_1} = \rho_1(y_0) = \rho(y_0) = [y_0]_{(K, \mu)}^{\rho_2} = [y_0]_{(K, \mu)}^{\rho_2}$.

The second part of the proposition follows easily from the first part by induction. △

We make one more observation before we prove the soundness of ELIMV rule. Let $y \in Y$ be a variable and $\gamma$ be a probabilistic formula such that $\gamma$ does not contain any occurrence of $y$. For any general probabilistic structure $(K, \mu)$ and $K$-assignments $\rho_1$ and $\rho_2$ such that $\forall y' \neq y \Rightarrow \rho_1(y') = \rho_2(y')$, we have

$(K, \mu)\rho_1 \models \eta$ iff $(K, \mu)\rho_2 \models \eta$.

The above observation can be proved by using induction on the structure of $\eta$.

Lemma 5.8 (Soundness of ELIMV rule) Given a probabilistic formula $\eta$, a probabilistic term $p$, a probabilistic formula $\eta$, a variable $y \in Y$ that does not occur either in $p$ or in $\eta$, and a statement $s$ such that

$\models_{h} \{ \eta \cap (y = p) \} s \{ \eta \}$.

Then

$\models_{h} \{ \eta^p_0 \} s \{ \eta \}$.
Proof: Let \((K, \mu)\) be a generalized state and \(p\) be a \(K\)-assignment such that \((K, \mu)\models \eta\). We need to show that \(((s)(K, \mu))\rho \models \eta\).

Let \(k = [p]_{\rho(K, \mu)}^\rho\) and \(\rho_1 = \rho_k^\rho\). We have \((K, \mu)\models \eta\) by Proposition ??.

Also, \([y]_{\rho(K, \mu)} = k\) by definition. Now, \([p]_{\rho(K, \mu)} = \rho_{k}^\rho\) by Proposition ??.

Since \(y\) does not occur in \(p\), \(p_{\rho}^\rho\) is \(p\) itself. Hence \([p]_{\rho(K, \mu)} = \rho_{(K, \mu)}^\rho = k\). Thus \((K, \mu)\models \eta\) also.

Therefore, by hypothesis, \(((s)(K, \mu))\rho_1 \models \eta\). Since, \(\rho_1\) and \(\rho\) differ only in the value assigned to \(y\) and \(y\) does not occur in \(\eta\), we get \(((s)(K, \mu))\rho \models \eta\) as required.

We are ready to prove soundness of Hoare calculus:

Theorem 5.9 (Soundness of Hoare calculus) The Hoare calculus is sound, i.e., \(\models \Psi \Rightarrow \vdash \Psi\) for any Hoare assertion \(\Psi\).

Proof: The proof is by induction on the length of the derivation of \(\vdash \Psi\) and it suffices to show that each of the axioms and inference rules are sound. The soundness of axioms TAUT and SKIP and of the inference rules SEQ, AND, OR and CONS can be easily established.

The soundness of axioms ASGR and ASGB follows by Lemma ??, The soundness of the axiom TOSS follows by Lemma ??, The soundness of the axiom ANAL follows by Proposition ??, Lemma ?? and Lemma ?? establish the the soundness of inference rules IF and ELIMV respectively.

6 Completeness and decidability of the Hoare calculus

We shall now show that the Hoare calculus provided in Section ?? is complete, i.e., if \(\models_h \Psi\) then \(\vdash \Psi\). Furthermore, we will show that there is an algorithm that given a probabilistic Hoare formula \(\Psi\) determines if \(\models_h \Psi\) or \(\models_h \Psi\). The proof of completeness and decidability of the Hoare logic uses the completeness and decidability of EPPL (see Theorem ??).

The proof of completeness of the Hoare logic employs the standard technique [??, ??] of defining the weakest precondition operator. Intuitively, the weakest precondition operator \(\text{wp}(\cdot, \cdot)\) assigns to each statement \(s \in \mathcal{S}\) and each formula \(\eta \in \mathcal{PForms}\) a new state formula \(\text{wp}(s, \eta)\) that corresponds to the weakest logical property that a state must satisfy to ensure that \(\eta\) holds after execution of \(s\). The weakest precondition itself shall use the weakest preterm operator. Intuitively, the weakest preterm operator \(\text{wpt}(\cdot, \cdot)\) assigns to each statement \(s \in \mathcal{S}\) and each probabilistic term \(p \in \mathcal{PTerms}\) a new term \(\text{wpt}(s, p)\) such that the denotation of \(\text{wpt}(s, p)\) in a given initial state is the same as the the denotation of \(p\) after the execution of \(s\).

We will then show that for any program \(s\) and EPPL formula \(\eta\), the Hoare calculus derives the judgment \(\vdash \{\text{wp}(s, \eta)\} s \{\eta\}\), i.e., \(\text{wp}(s, \eta)\) is a sufficient precondition for \(s\) and \(\eta\). Furthermore, we shall that \((K, \mu)\rho \models \text{wp}(s, \eta)\) iff \([s]_{(K, \mu)}\rho \models \eta\). This will imply that if \(\models_h \{\eta\} s \{\eta\}\) then \(\vdash (\eta \supset \eta)\). The completeness of EPPL will allow us to conclude that \((\eta \supset \eta)\) is an EPPL theorem and we can then use the Hoare inference rule CONS to conclude \(\vdash \{\eta\} s \{\eta\}\). The decidability of the Hoare calculus will then follow from the fact that the
weakest precondition can be computed algorithmically and the fact that EPPL is decidable. We start by defining the weakest preterm operator.

### 6.1 Weakest preterms

The weakest preterm $\text{wpt}(s, p)$ is defined recursively on the structure of the statement $s$ and the probability term $p$. Once we have defined the weakest preterm, we shall prove certain properties by using induction on the structure of the statement $s$ and the probability term $p$. Please note that for the rest of the paper, the underlying pre-order for the above mentioned recursion and induction to make sense is the lexicographic order on the set $S \times \text{PTerms}$.

Please recall that given a memory cell $bm$, a constant $r \in A$ and a probabilistic term $p$ the term $\text{toss}(bm, r; p)$ is the term obtained from $p$ by replacing every occurrence of each measure term $(\int \gamma)$ by $(1 - \tilde{r})(\int \gamma_{\text{ff}}) + \tilde{r}(\int \gamma_{\text{tt}})$. Also, given a classical state formula $\gamma$ and a probabilistic term $p$ the term $(p/\gamma)$ is the term obtained from $p$ by replacing every occurrence of each measure term $(\int \gamma')$ by $(\int (\gamma' \land \gamma))$.

**Definition 6.1** The map $\text{wpt} : S \times \text{PTerms} \rightarrow \text{PTerms}$ is defined recursively as:

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

The weakest precondition operator acts as the identity on the constants and the variables. The set of variables occurring in the preterm is also unchanged:

**Proposition 6.2** For any statement $s$,

- $\text{wp}(s, r) = r$ for all $r \in A$,
- $\text{wp}(s, y) = y$ for all $y \in Y$ and
- $\text{PVar}(p) = \text{PVar}(\text{wpt}(s, p))$ all probabilistic terms $p$.

**Proof:** The proof is by induction on the structure of $s$ and $p$. 

---

19
Lemma 6.3 For any probabilistic term $p$, statement $s$, any generalized structure $(\mathcal{K}, \mu)$ and $\mathcal{K}$-assignment $\rho$,

$$[\text{wpt}(s, p)]^{p}_{(\mathcal{K}, \mu)} = [p]^{p}_{s}[\mathcal{K}, \mu].$$

Proof: The proof is by induction on the structure of $s$ and $p$. The case when $s$ is skip follows from definition. The case when $s$ is an assignment to a memory cell follows from Lemma ???. The case when $s$ is a probabilistic toss follows from Lemma ???.

If $s$ is the statement $s_1; s_2$ then given a fixed probabilistic term $p$ we have by induction hypothesis (applied to statement $s_2$):

$$[\text{wpt}(s_2, p)]^{p}_{s_2}[\mathcal{K}, \mu] = [p]^{p}_{s_2}[\mathcal{K}, \mu] = [p]^{p}_{s_2}[\mathcal{K}, \mu].$$

Also, for all probabilistic terms $p'$, we have by induction hypothesis (applied to statement $s_1$):

$$[\text{wpt}(s_1, p')]^{p'}_{s_1}[\mathcal{K}, \mu] = [p']^{p'}_{s_1}[\mathcal{K}, \mu].$$

Thus, by letting $p'$ to be $\text{wpt}(s_2, p)$, we get

$$[\text{wpt}(s_1, \text{wpt}(s_2, p))]^{p}_{s_1}[\mathcal{K}, \mu] = [\text{wpt}(s_2, p)]^{p}_{s_2}[\mathcal{K}, \mu] = [p]^{p}_{s_2}[\mathcal{K}, \mu]$$

as required.

If $s$ is the statement if $\gamma$ then $s_1$ else $s_2$, then the case case when $p$ is just the 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 $\rho$.

If $s$ is the statement if $\gamma$ then $s_1$ else $s_2$ and $p$ is the term $(\int\gamma_0)$ then by definition:

$$[\int\gamma_0]^{p}_{s_1}[\mathcal{K}, \mu] = \kappa_1 + \kappa_2$$

where

$$\kappa_1 = [\int\gamma_0]^{p}_{s_1}[\mathcal{K}, \mu]$$

and

$$\kappa_2 = [\int\gamma_0]^{p}_{s_2}[\mathcal{K}, \mu].$$

Now, by induction hypothesis applied to $s_1$ and $s_2$ respectively, we get:

$$\kappa_1 = [\text{wpt}(s_1, (\int\gamma_0))]^{p}_{s_1}[\mathcal{K}, \mu]$$

and

$$\kappa_2 = [\text{wpt}(s_2, (\int\gamma_0))]^{p}_{s_2}[\mathcal{K}, \mu].$$

Now, by Proposition ??, we get

$$\kappa_1 = [\text{wpt}(s_1, (\int\gamma_0))]^{p}_{s_1}[\mathcal{K}, \mu]$$

and

$$\kappa_2 = [\text{wpt}(s_2, (\int\gamma_0))]^{p}_{s_2}[\mathcal{K}, \mu].$$

The result now follows in this case.

If $s$ is the statement if $\gamma$ then $s_1$ else $s_2$ and $p$ is $(p_1 + p_2)$ or $(p_1 p_2)$, then by induction hypothesis,

$$[\text{wpt}(s, p_1)]^{p}_{s}[\mathcal{K}, \mu] = [p_1]^{p}_{s}[\mathcal{K}, \mu]$$

and

$$[\text{wpt}(s, p_2)]^{p}_{s}[\mathcal{K}, \mu] = [p_2]^{p}_{s}[\mathcal{K}, \mu].$$

The result now follows immediately. △
6.2 Weakest preconditions

The weakest precondition operator \( \text{wp}: S \times \text{PForms} \rightarrow \text{PForms} \) is defined using the weakest preterm operator. The weakest precondition \( \text{wp}(s, \eta) \) is obtained by replacing each comparison formula \( (p_1 \leq p_2) \) occurring in \( \text{wp}(s, \eta) \) by \( (\text{wpt}(s, p_1) \leq \text{wpt}(s, p_2)) \). Formally,

**Definition 6.4** The map \( \text{wp}: S \times \text{PForms} \rightarrow \text{PForms} \) is defined recursively as:

\[
\begin{align*}
\text{wp}(s, \text{fff}) &= \text{fff} \\
\text{wp}(s, (p_1 \leq p_2)) &= (\text{wpt}(s, p_1) \leq \text{wpt}(s, p_2)) \\
\text{wp}(s, (\eta_1 \supset \eta_2)) &= (\text{wp}(s, \eta_1) \supset \text{wp}(s, \eta_2)).
\end{align*}
\]

It follows easily from the definition and Lemma ?? that \( \text{wp}(s, \eta) \) is indeed the weakest precondition:

**Theorem 6.5 (Necessity and sufficiency of weakest precondition)** For any statement \( s \), probabilistic formula \( \eta \), any generalized structure \( (K, \mu) \) and \( K \)-assignment \( \rho \),

\[
(K, \mu)\rho \models_h \text{wp}(s, \eta) \iff ([s](K, \mu))\rho \models_h \eta.
\]

**Proof:** The proof is by induction on the structure of \( \eta \). The base case where \( \eta \) is \text{fff} is immediate. The other base is \( \eta \) is \( (p_1 \leq p_2) \). Please note that by Lemma ??, we have

\[
[wpt(s, p_1)]_{(K, \mu)} = [p_1]_{([s](K, \mu))} \text{ and } [wpt(s, p_2)]_{(K, \mu)} = [p_2]_{([s](K, \mu))}.
\]

The result now follows immediately.

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

\[
(K, \mu)\rho \models_h \text{wp}(s, \eta_1) \iff ([s](K, \mu))\rho \models_h \eta_1
\]

and

\[
(K, \mu)\rho \models_h \text{wp}(s, \eta_2) \iff ([s](K, \mu))\rho \models_h \eta_2.
\]

The result now follows immediately. \( \triangle \)

We get the following corollary:

**Corollary 6.6** For any statement \( s \) and probabilistic formulas \( \eta \) and \( \eta' \), we have

\[
\models_h \{\eta'\} s \{\eta\} \iff (\eta' \supset \text{wp}(s, \eta)).
\]

**Proof:** \((\Rightarrow)\) Suppose \( \models_h \{\eta'\} s \{\eta\} \). Consider an arbitrary generalized probabilistic state \( (K, \mu) \) and an arbitrary \( K \)-assignment \( \rho \) such that \( (K, \mu)\rho \models \eta' \). Since \( \models_h \{\eta'\} s \{\eta\} \), we get \( ([s](K, \mu))\rho \models \eta \). Therefore, by Theorem ??, \( (K, \mu)\rho \models \text{wp}(s, \eta) \). Since \( (K, \mu) \) and \( \rho \) are arbitrary, we get \( \models (\eta' \supset \text{wp}(s, \eta)) \).

\((\Leftarrow)\) Suppose \( \models (\eta' \supset \text{wp}(s, \eta)) \). Consider an arbitrary generalized probabilistic state \( (K, \mu) \) and an arbitrary \( K \)-assignment \( \rho \) such that \( (K, \mu)\rho \models \eta' \). Since \( \models (\eta' \supset \text{wp}(s, \eta)) \), we get \( (K, \mu)\rho \models \text{wp}(s, \eta) \). Therefore, by Theorem ??, \( ([s](K, \mu))\rho \models \eta \). Since \( (K, \mu) \) and \( \rho \) are arbitrary, we get \( \models_h \{\eta'\} s \{\eta\} \).
The next step is to show that the Hoare axiomatization allows us to derive the judgment:

\[ \vdash \{ \text{wp}(s, \eta) \} s \{ \eta \}. \]

We start by showing this by showing it in the special case when the formula \( \eta \) is \( y = p \) for some given variable \( y \in \mathcal{Y} \) and probabilistic term \( p \):

**Lemma 6.7** For any probabilistic term \( p \), statement \( s \) and any variable \( y \in \mathcal{Y} \)

\[ \vdash \{ y = \text{wpt}(s, p) \} s \{ y = p \}. \]

**Proof:**

The proof is by induction on the structure of \( s \) and \( p \). If \( s \) is \texttt{skip}, then the judgment \( \vdash \{ y = \text{wpt}(s, p) \} s \{ y = p \} \) can be derived by the axiom \texttt{SKIP}. If \( s \) is an assignment to a memory cell then the required judgment can be derived by the axiom \texttt{ASGB} or the axiom \texttt{ASGR}. If \( s \) is a probabilistic toss, the judgment follows from the axiom \texttt{TOSS}.

If \( s \) is the sequential composition \( s_1; s_2 \) then \( \text{wpt}(s_1; s_2, p) \overset{df}{=} \text{wpt}(s_1, \text{wpt}(s_2, p)) \) and by induction hypothesis applied on the statements \( s_1 \) and \( s_2 \) we have:

\[ \vdash \{ y = \text{wpt}(s_1, \text{wpt}(s_2, p)) \} s_1 \{ y = \text{wpt}(s_2, p) \} \]

and

\[ \vdash \{ y = \text{wpt}(s_2, p) \} s_2 \{ y = p \} \]

respectively. The inference rule \texttt{SEQ} then gives us

\[ \vdash \{ y = \text{wpt}(s_1; s_2, p) \} s_1; s_2 \{ y = p \}. \]

If \( s \) is the alternative if \( \gamma \) then \( s_1 \) else \( s_2 \) and \( p \) is a constant \( r \in \mathcal{A} \) then by the axiom \texttt{ANAL}, we have

\[ \vdash \{ y = r \} s \{ y = r \}. \]

The result follows by observing that \( \text{wpt}(s, r) = r \) by Proposition ???. The case when \( p \) is a variable \( y \in \mathcal{Y} \) is similar.

Now consider the case where \( s \) is the alternative if \( \gamma \) then \( s_1 \) else \( s_2 \) and \( p \) is \( (f \gamma_0) \) for some classical state formula \( \gamma_0 \). Now, pick two distinct variable \( y_1, y_2 \in \mathcal{Y} \) different from \( y \). Let \( \eta_1 \) be \( (y_1 = \text{wpt}(s_1, (f \gamma_0))) \) and \( \eta_2 \) be \( (y_2 = \text{wpt}(s_2, (f \gamma_0))) \). Let \( \eta \) be

\[ (y = y_1 + y_2) \cap (y_1 = \text{wpt}(s_1, (f \gamma_0)))/\gamma \cap (y_2 = \text{wpt}(s_2, (f \gamma_0))/(\neg \gamma)). \]

Now, by induction hypothesis (applied to \( s_1 \) and \( s_2 \) respectively), we have:

\[ \vdash \{ y_1 = \text{wpt}(s_1, (f \gamma_0)) \} s_1 \{ y_1 = (f \gamma_0) \} \]

and

\[ \vdash \{ y_2 = \text{wpt}(s_2, (f \gamma_0)) \} s_2 \{ y_2 = (f \gamma_0) \}. \]

Please recall that \( \text{wpt}(s_1, (f \gamma_0))/\gamma + \text{wpt}(s_2, (f \gamma_0))/(\neg \gamma) \) is \( \text{wpt}(s, p) \) by definition. We can derive the judgment

\[ \vdash \{ y = \text{wpt}(s, p) \} s \{ y = p \} \]

as follows:
Now, we can derive the judgment as follows

1. \( \{ y_1 = \text{wpt}(s_1, (f \theta_0)) \} s_1 \{ y_1 = \text{wpt}(s_1, \gamma_0) \} \)
   \( \text{Ind. Hyp} \)

2. \( \{ y_2 = \text{wpt}(s_2, (f \theta_0)) \} s_2 \{ y_2 = \text{wpt}(s_2, \gamma_0) \} \)
   \( \text{Ind. Hyp} \)

3. \( \{ \eta_1 \gamma, \eta_2 \} \) if \( \gamma \) then \( s_1 \) else \( s_2 \) \( (y_1 + y_2 = (f \theta_0)) \)
   \( \text{IF: 1, 2} \)

4. \( \eta^1 \supset (\gamma_1 \gamma, \gamma_2) \)
   \( \text{TAUT} \)

5. \( \{ \eta^1 \} \) if \( \gamma \) then \( s_1 \) else \( s_2 \) \( (y_1 + y_2 = (f \theta_0)) \)
   \( \text{Cons: 3, 4} \)

6. \( \{ y = y_1 + y_2 \} \) if \( \gamma \) then \( s_1 \) else \( s_2 \) \( \{ y = y_1 + y_2 \} \)
   \( \text{ANAL} \)

7. \( \eta^1 \supset (y = y_1 + y_2) \)
   \( \text{TAUT} \)

8. \( \{ \eta^1 \} \) if \( \gamma \) then \( s_1 \) else \( s_2 \) \( \{ y = y_1 + y_2 \} \)
   \( \text{Cons: 6, 7} \)

9. \( \{ \eta^1 \} \) if \( \gamma \) then \( s_1 \) else \( s_2 \) \( \{ (y = y_1 + y_2) \cap (y_1 + y_2 = (f \theta_0)) \} \)
   \( \text{AND: 5, 8} \)

10. \( \{ (y = y_1 + y_2) \cap (y_1 + y_2 = (f \theta_0)) \supset (y = (f \theta_0)) \} \)
    \( \text{TAUT} \)

11. \( \{ \eta^1 \} \) if \( \gamma \) then \( s_1 \) else \( s_2 \) \( \{ y = (f \theta_0) \} \)
    \( \text{Cons: 9, 10} \)

12. \( \{ (y = y_1 + \text{wpt}(s_2, (f \theta_0))) \cap (y_1 = \text{wpt}(s_1, (f \theta_0))/\gamma) \} \)
    
    \( \text{if } \gamma \text{ then } s_1 \text{ else } s_2 \{ y = (f \theta_0) \} \)
    \( \text{ELIMV: 11} \)

13. \( \{ y = \text{wpt}(s_1, (f \theta_0))/\gamma + \text{wpt}(s_2, (f \theta_0))/\neg \gamma) \}
    
    \( \text{if } \gamma \text{ then } s_1 \text{ else } s_2 \{ y = (f \theta_0) \} \)
    \( \text{ELIMV: 12} \)

If \( s \) is is the alternative if \( \gamma \) then \( s_1 \) else \( s_2 \) and \( p \) is \( (p_1 + p_2) \) then pick \( y_1, y_2 \in Y \) different from \( y \) such that \( y_1 \) and \( y_2 \) do not occur in \( p_1 \) or \( p_2 \). Let \( \eta^\dagger \)

\[
(((y = y_1 + y_2) \cap (y_1 = p_1)) \cap (y_2 = p_2)).
\]

and \( \eta^{\dagger\dagger} \) be defined as

\[
(((y = y_1 + y_2) \cap ((y_1 = \text{wpt}(s, p_1)) \cap (y_2 = \text{wpt}(s, p_2)))).
\]

Now, by induction hypothesis (applied to \( p_1 \) and \( p_2 \)), we get

\[
\vdash \{ y_1 = \text{wpt}(s, p_1) \} s \{ y_1 = p_1 \}
\]

and

\[
\vdash \{ y_2 = \text{wpt}(s, p_2) \} s \{ y_1 = p_2 \}.
\]

Now, we can derive the judgment

\[
\vdash \{ y = \text{wpt}(s, p) \} s \{ y = p \}
\]

as follows
\begin{enumerate}
\item \{y_1 = \text{wpt}(s, p_1)\} \cap \{y_1 = p_1\}
\item \eta' \supset ((y_1 = \text{wpt}(s, p_1)))
\item \{(\eta') s \{y_1 = p_1\}\)
\item \{y_2 = \text{wpt}(s, p_2)\} \cap \{y_2 = p_2\}
\item \eta' \supset ((y_2 = \text{wpt}(s, p_2)))
\item \{(\eta') s \{y_2 = p_2\}\)
\item \{(\eta') s ((y_1 = \text{wpt}(s, p_1)) \cap (y_1 = \text{wpt}(s, p_1)))\}
\item \{y = y_1 + y_2\} \cap \{y = y_1 + y_2\}
\item \eta' \supset (y = y_1 + y_2)
\item \{(\eta') s \{y = y_1 + y_2\}\)
\item \{(\eta') s \{\eta'\}\)
\item \eta' \supset (y = (p_1 + p_2))
\item \{(\eta') s \{y = (p_1 + p_2)\}\)
\item \{(y = y_1 + \text{wpt}(s, p_2)) \cap (y_1 = \text{wpt}(s, p_1))\} \cap \{y = (p_1 + p_2)\}
\item \{y = \text{wpt}(s, p_1) + \text{wpt}(s, p_2)\} \cap \{y = (p_1 + p_2)\}
\end{enumerate}

The case where \(p\) is \((p_1p_2)\) is similar.

We are now ready to extend the above result for any \(\eta\), \(i.e.,\) to show that the judgment \(\vdash \{\text{wp}(s, \eta)\} \cap \{\eta\}\) is derivable in the Hoare logic. Given a probabilistic formula \(\eta\) and a probabilistic term \(p\) we say that \(p\) occurs as a comparison term in \(\eta\) 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 \(\eta\). We have:

**Theorem 6.8 (Weakest precondition is derivable)** For any statement \(s\) and any conditional free formula \(\eta\),

\[\vdash \{\text{wp}(s, \eta)\} \cap \{\eta\}\].

**Proof:** Let \(p_1, p_2, \ldots, p_n\) be the comparison terms occurring in \(\eta\). Pick \(n\) distinct variables \(y_1, y_2, \ldots, y_n \in Y\) that do not occur in \(\eta\). Let \(p'_1, p'_2, \ldots, p'_n\) be the terms \(\text{wpt}(s, p_1), \text{wpt}(s, p_2), \ldots, \text{wpt}(s, p_n)\) respectively.

Let \(\eta'\) be the formula obtained by replacing each occurrence of a comparison formula \((p_i \leq p_j)\) in \(\eta\) by the comparison formula \((y_i \leq y_j)\). Let \(\eta_a\) be the formula

\[\eta' \cap (\bigcap_i (y_i = p_i))\]

and \(\eta_b\) be the formula

\[\eta' \cap (\bigcap_i (y_i = p'_i)).\]

Clearly, we have

- \(\eta'\) is an analytical formula;
- \(\eta'_{y_1, y_2, \ldots, y_n}\) is \(\eta\);
- \((\eta_a \supset \eta')\) and \((\eta_b \supset \eta')\) are EPPL theorems;
• \((\eta \triangleright (y = p_i'))\) are EPPL theorems for all \(1 \leq i \leq n\); and

• \(wp(s, \eta)\) is \(\eta^1_{p_1'^1} p_2'^2 \ldots p_n'^n\)

Now, by axiom \textbf{ANAL}, we have

\[ \vdash \{ \eta^1 \} s \{ \eta \} \]

Also, by Lemma ??, we have for all \(1 \leq i \leq n\),

\[ \vdash \{ y_i = p_i' \} s \{ y = p_i \} \]

Now, \((\eta \triangleright \eta^1)\) and \((\eta \triangleright (y = p_i'))\) are EPPL theorems for all \(1 \leq i \leq n\). Hence, we have by the inference rule \textbf{CONS},

\[ \vdash \{ \eta_b \} s \{ \eta \} \text{ and } \vdash \{ \eta_b \} s \{ y = p_i \} \]

for all \(1 \leq i \leq n\). Several applications of the inference rule \textbf{AND} then gives

\[ \vdash \{ \eta_b \} s \{ \eta \} \]

Since \((\eta_a \triangleright \eta)\) is an EPPL theorem, we get by the inference rule \textbf{CONS},

\[ \vdash \{ \eta_b \} s \{ \eta \} \]

Finally, several applications of the inference rule \textbf{ELIMV} gives

\[ \vdash \{ \eta^1_{p_1'^1} p_2'^2 \ldots p_n'^n \} s \{ \eta \} \]

as required. \(\triangle\)

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

\textbf{Theorem 6.9} The Hoare calculus is complete, \(i.e.,\) if \(\vdash_h \{ \eta' \} s \{ \eta \}\) then \(\vdash \{ \eta' \} s \{ \eta \}\). Moreover, the set of theorems of the Hoare calculus is recursive.

\textbf{Proof:}

Completeness. Suppose that \(\vdash_h \{ \eta' \} s \{ \eta \}\). Then by Corollary ??, we get

\[ \vdash (\eta' \triangleright wp(s, \eta)) \]

The completeness of EPPL then gives \(\vdash (\eta' \triangleright wp(s, \eta))\). Also, by Theorem ??, we get \(\vdash \{ wp(s, \eta) \} s \{ \eta \}\). Hence, by the inference rule \textbf{CONS}, we get \(\vdash \{ \eta' \} s \{ \eta \}\).

Decidability. As a consequence of completeness and soundness, we have \(\vdash \{ \eta' \} s \{ \eta \} \text{ iff } \vdash_h \{ \eta' \} s \{ \eta \}\). Thus, by Corollary ?? and completeness of EPPL, \(\vdash \{ \eta' \} s \{ \eta \} \text{ iff } \vdash (\eta' \triangleright wp(s, \eta))\). The decidability now follows from decidability of EPPL and the fact that \(wp(s, \eta)\) can be computed algorithmically.

\section{Examples}

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

$$\text{toss}(bm_k, \frac{1}{2}); bm_c \leftarrow \neg (bm_k \Leftrightarrow bm_p).$$

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

$$\Psi_{ot} \overset{df}{=} \{(\int tt) = 1\} S_{\text{enc}} \{(\int bm_c) = \frac{1}{2}\}.$$

The pre-condition $(\int tt) = 1$ means that the total measure is 1. We can derive the above in our Hoare calculus. We shall however the use of weakest preconditions.

The weakest precondition $wp(S_{\text{enc}}, (\int bm_c) = \frac{1}{2})$ is $wp(S_{\text{enc}}, (\int bm_c)) = wp(S_{\text{enc}}, \frac{1}{2})$.

Now, $wp(S_{\text{enc}}, \frac{1}{2})$ is $\frac{1}{2}$ by Proposition ???. Also, by definition,

$$wp(S_{\text{enc}}, (\int bm_c)) = wp(\text{toss}(bm_k, \frac{1}{2}), wp(bm_c \leftarrow \neg (bm_k \Leftrightarrow bm_p), (\int bm_c)))$$

$$= wp(\text{toss}(bm_k, \frac{1}{2}), (\int \neg (bm_k \Leftrightarrow bm_p)))$$

$$= \frac{1}{2}(\int \neg (bm_k \Leftrightarrow tt)) + \frac{1}{2}(\int \neg (bm_k \Leftrightarrow ff)).$$

Hence,

$$wp(S_{\text{enc}}, (\int bm_c) = \frac{1}{2}) \approx (\frac{1}{2}(\int \neg (bm_k \Leftrightarrow tt)) + \frac{1}{2}(\int \neg (bm_k \Leftrightarrow ff)) = \frac{1}{2}).$$

Now, the assertion $\Psi_{ot}$ follows from the fact that the following is an EPPL theorem:

$$((\int tt) = 1) \approx (\frac{1}{2}(\int \neg (bm_k \Leftrightarrow tt)) + \frac{1}{2}(\int \neg (bm_k \Leftrightarrow ff)) = \frac{1}{2}).$$

\footnote{Please note that $\neg (bm_k \Leftrightarrow bm_p)$ is one way of computing xor.}