Probabilistic Abstract Interpretation

Patrick Cousot and Michael Monerau

Courant Institute, NYU and École Normale Supérieure, France

Abstract. Abstract interpretation has been widely used for verifying properties of computer systems. Here, we present a way to extend this framework to the case of probabilistic systems.

The probabilistic abstraction framework that we propose allows us to systematically lift any classical analysis or verification method to the probabilistic setting by separating in the program semantics the probabilistic behavior from the (non-)deterministic behavior. This separation provides new insights for designing novel probabilistic static analyses and verification methods.

We define the concrete probabilistic semantics and propose different ways to abstract them. We provide examples illustrating the expressiveness and effectiveness of our approach.

1 Introduction

As programs get larger and larger, it has become untractable to verify their properties and/or correctness by hand or testing. Formal methods have thus been developed in order to be able to verify program properties automatically, at least in part. One of them is abstract interpretation which has proved successful both in solving hard problems and scaling up nicely.

When probabilities come into play, the verification of program properties is even more difficult. Our work precisely tackles this issue, that is *verifying properties of probabilistic programs*. We propose a formal, general and modular framework, extending the classical abstract interpretation framework to take probabilities into account, allowing for crafting of new analyses, as well as lifting of existing non-probabilistic analyses to the probabilistic setting.

Probabilities come into play because of *program randomness* (such as calls to a random number generator `rand()`) and *input randomness* (for which a distribution may be known). Usually, all this randomness is forgotten for non-determinism. It is sound but loses a lot of information. So our goal here is to *use* hypotheses on randomness to be able to infer more precise probabilistic program properties.

The goals of having probabilistic static analyses are various, let alone the fact that we can actually verify some probabilistic properties on the program. A couple of more original examples of interesting applications are to enable compilers to gain access to more useful information to decide register allocations or cache/scratchpad allocations, or to provide useful information about branching for Just In Time compilers without having to do any profiling or execution, among many other applications.

There is a lot of work on probabilistic program construction and verification methods [13, 15, 19, 23], probabilistic model-checking [11], probabilistic abstract model-checking [2, 27, 29], probabilistic abstract interpretation [21, 25, 28], with, in the case of model-checking and abstract interpretation, existing applications to biological pathways [1, 3, 18]. One of our objectives is to unify and generalize these frameworks.

2 The Abstract Interpretation Framework

Abstract interpretation is a theory of approximation. Applied to semantics of computer programs, it allows oneself for generic design of static analyses [5].

The *concrete semantics* $S[\![\mathrm{P}]\!]$ of a program P is, by hypothesis, an element $S[\![\mathrm{P}]\!] \in \mathcal{D}$, where $\mathcal{D}$ is a fixed *semantics domain*. It is often expressed as a least fixpoint $S[\![\mathrm{P}]\!] = \mathrm{lfp}^{\preceq} F_{\mathrm{P}}$ where the *concrete transformer* is $F_{\mathrm{P}} : \mathcal{D} \longrightarrow \mathcal{D}$ and $\preceq$ is the *concrete semantic partial order* on $\mathcal{D}$.

Semantic properties of programs are elements of the *concrete domain* $\langle \wp(\mathcal{D}), \subseteq \rangle$ where $\subseteq$ is logical implication. A program P is said to *verify a property* $\Gamma \in \wp(\mathcal{D})$ iff $S[\![\mathrm{P}]\!] \in \Gamma \iff \{S[\![\mathrm{P}]\!]\} \subseteq \Gamma$, which is often undecidable or intractable so that approximations are necessary for total automation.

A partially ordered *abstract domain* $\langle \mathcal{A}, \sqsubseteq \rangle$ is considered and linked to the concrete domain by means of a *Galois connection* $\langle \wp(\mathcal{D}), \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\longleftrightarrow}} \langle \mathcal{A}, \sqsubseteq \rangle$ defined such that $\forall P \in \wp(\mathcal{D}) : \forall Q \in \mathcal{A} : \alpha(P) \sqsubseteq Q \iff P \subseteq \gamma(Q)$. For example the interval abstraction is $\langle \wp(\mathbb{Z}), \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\longleftrightarrow}} \langle \mathcal{I}(\mathbb{Z}), \sqsubseteq_I \rangle$ with $I(\mathbb{Z}) \triangleq \{\bot\} \cup \{[a, b] \mid a \leqslant b\}$, $\alpha(\emptyset) \triangleq \bot$, and $\alpha(S) = [\min S, \max S]$ when $S \neq \emptyset$ where $\min \mathbb{Z} \triangleq -\infty$, $\max \mathbb{Z} \triangleq +\infty$, and $\sqsubseteq_I$ is interval inclusion. In a Galois connection one adjoint uniquely determines the other (which we often leave implicit). Galois connections are used for the sake of simplicity although not necessary (a concretization function γ may be sufficient [7]). The only way to know what is the meaning of *verifying an abstract property* $Q \in \mathcal{A}$ is to evaluate the concretization function γ. Indeed, by definition it means that $S[\![\mathrm{P}]\!] \subseteq \gamma(Q)$, i.e. P verifies the property $\gamma(Q)$.

Static analysis consists in computing an *abstract semantics* $S[\![\mathrm{P}]\!]^{\sharp}$ of the program that is less precise but still *sound* $S[\![\mathrm{P}]\!] \subseteq \gamma(S[\![\mathrm{P}]\!]^{\sharp})$ (and sometimes even *complete* for a given class of properties when it loses no essential information for proofs). Thus the program P is said to satisfy an abstract property $Q \in \mathcal{A}$ iff $S[\![\mathrm{P}]\!]^{\sharp} \sqsubseteq Q$ (which implies $S[\![\mathrm{P}]\!] \subseteq \gamma(S[\![\mathrm{P}]\!]^{\sharp})$ since γ is increasing and $\subseteq$ transitive). An adequate cost/precision ratio consists in choosing $\langle \mathcal{A}, \sqsubseteq \rangle$ and $S[\![\mathrm{P}]\!]^{\sharp}$ to be algorithmically tractable hence imprecise so incomplete but nevertheless precise enough so that $S[\![\mathrm{P}]\!]^{\sharp} \sqsubseteq Q$ implies $S[\![\mathrm{P}]\!] \subseteq \gamma(Q)$. Soundness is always guaranteed along the way by the framework.

3 Probabilistic Concrete Semantics

Our approach relies on basic concepts of classical abstract interpretation that we recalled in Sect. 2 and probability theory [16].

In this section, we introduce how we describe the semantics of probabilistic programs (or systems). It is a very general way of associating a semantics with any probabilistic system. That is, it is not tied to a particular description of probabilities nor to a specific programming language but rather allows for a precise construction of semantics for any probabilistic situation.

3.1 Definition

We look at probabilistic systems as a superposition of (non)-deterministic systems. That is, when a probabilistic program is run we consider that it can be any element of a specific set of (non)-deterministic programs chosen by a random experience. It is as if *all* the random choices that will be made in the subsequent execution are decided by an oracle at startup (although a program knows only during the course of its execution about which random choices have been made up to the current execution point and ignores the later ones[1]).

Definition 1 (Probabilistic semantics). A *probabilistic semantics* $S_p[\![\mathtt{P}]\!] \in \mathcal{D}_p \triangleq \Omega \rightarrowtail \mathcal{D}$ of a program $\mathtt{P}$ is a measurable function of a probability space $\langle \Omega,\ \mathcal{E},\ \mu \rangle$ into a semantics domain $\mathcal{D}$ (considered as a measurable space $\langle \mathcal{D},\ \mathcal{O} \rangle$ with observable semantic properties in $\mathcal{O} \subseteq \wp(\mathcal{D})$). □

By *observable*, we mean that semantic properties in $\mathcal{O}$ will be the ones we eventually have probabilistic information upon.

The meaning of the probabilistic semantics $S_p[\![\mathtt{P}]\!]$ is that when a scenario $\omega \in \Omega$ is picked (randomly according to μ), then the execution of the program $\mathtt{P}$ yields the (non)-deterministic semantics $S_p[\![\mathtt{P}]\!](\omega) \in \mathcal{D}$. That is, ω embodies all the possible random choices that the program will have to make during its execution. $\mathcal{D}$ can be any non-probabilistic semantics domain (e.g. the powerset of maximal execution traces as in Ex. 4 below or any of its abstractions [4] such as the prefix trace semantics in Ex. 1). This definition covers most probabilistic models of computation found in the literature such as program semantics [17], Markov decision processes [2, 3, 10, 11, 22, 29], etc.

Example 1. Suppose the program $\mathtt{P}$ starts by tossing a coin `x = random(1,2)`, and then executes other statements. The prefix trace semantics of $\mathtt{P}$ would be described by $\Omega = \{\omega_1, \omega_2\}$ and $S_p[\![\mathtt{P}]\!] \in \mathcal{D}_p = \Omega \rightarrowtail \mathcal{D}$, where $\mathcal{D} = \wp(\mathcal{S}^+)$ is the set of finite sequences of states and the observable properties are simply $\wp(\mathcal{D})$, defined as $S_p[\![\mathtt{P}]\!](\omega_1) = \{$ prefix traces of $\mathtt{P}$ starting with `x = 1` $\}$ and $S_p[\![\mathtt{P}]\!](\omega_2) = \{$prefix traces of $\mathtt{P}$ starting with `x = 2`$\}$. Then the definition of μ would tell what is the probability of scenarios ω_1 and ω_2. For a non-biased coin, μ would be defined by $\mu(\{\omega_1\}) = 1/2, \mu(\{\omega_2\}) = 1/2, \mu(\emptyset) = 0, \mu(\Omega) = 1$. □

Example 2 (Markov chains). Markov chains can be formalized in our framework by taking $\Omega = [0, 1]^{N}$ (sequences of elements in [0,1]) with the uniform Lebesgue measure. For a specific sequence $u_n \in \Omega$, the execution of the Markov chain is as follows.

From a state s_0, at step $i \geq 0$, where multiple states $s_1, \ldots, s_k$ of the Markov chain can be chosen for the next step and where the probability of going to state s_a is $p_a \in [0, 1]$. By definition, $\sum_{1 \leq a \leq k} p_a = 1$, so $[0, 1]$ can be divided in k segments S_a each of length p_a. Now, choose $s_{i+1} = s_a$ such that $u_i \in S_a$. □

Definition 2 (Probability of a program property). The probability that a program $\mathtt{P}$ has property $\Phi \in \mathcal{O}$ is $\mathbf{Pr}(S_p[\![\mathtt{P}]\!] \in \Phi) = S_p[\![\mathtt{P}]\!](\mu)(\Phi)$. □

Example 3. The semantics $S_p[\![\mathtt{P}]\!] \in \mathcal{D}_p = \Omega_{\mathtt{P}} \rightarrowtail \mathcal{D}_{\mathtt{P}}$ of $\mathtt{P}$ as shown in Fig. 1 can be defined with $\mathcal{D}_{\mathtt{P}} \triangleq \mathbb{Z}^3$ denoting the final value of the variables `x`, `y` and `z` and

[1] This is usually formalized by a filtration in measure theory/probabilities.

P	ω	$S_p[\![\mathtt{P}]\!](\omega)$	$\mu(\{\omega\})$
	$\overleftarrow{x}\,\overleftarrow{y}\,\overleftarrow{z}$	$\langle 1, 0, 2\rangle$	$\frac{1}{2}\cdot\frac{1}{3}\cdot\frac{1}{4}=\frac{1}{24}$
`x = 1` $_{\frac{1}{2}}\oplus$ `x = 2;`	$\overleftarrow{x}\,\overleftarrow{y}\,\overrightarrow{z}$	$\langle 1, 0, 4\rangle$	$\frac{1}{2}\cdot\frac{1}{3}\cdot\frac{3}{4}=\frac{1}{8}$
`y = 0` $_{\frac{x}{3}}\oplus$ `y = 1;`	$\overleftarrow{x}\,\overrightarrow{y}\,\overleftarrow{\overline{z}}$	$\langle 1, 1, 1\rangle$	$\frac{1}{2}\cdot\frac{1}{3}\cdot\frac{1}{5}=\frac{1}{30}$
`if (y = 0) then`	$\overleftarrow{x}\,\overrightarrow{y}\,\overrightarrow{\overline{z}}$	$\langle 1, 1, 3\rangle$	$\frac{1}{2}\cdot\frac{1}{3}\cdot\frac{4}{5}=\frac{2}{15}$
`  z = 2` $_{\frac{1}{4}}\oplus$ `z = 4`	$\overrightarrow{x}\,\overleftarrow{y}\,\overleftarrow{z}$	$\langle 2, 0, 2\rangle$	$\frac{1}{2}\cdot\frac{2}{3}\cdot\frac{1}{4}=\frac{1}{12}$
`else`	$\overrightarrow{x}\,\overleftarrow{y}\,\overrightarrow{z}$	$\langle 2, 0, 4\rangle$	$\frac{1}{2}\cdot\frac{2}{3}\cdot\frac{3}{4}=\frac{1}{4}$
`  z = 1` $_{\frac{1}{5}}\oplus$ `z = 3`	$\overrightarrow{x}\,\overrightarrow{y}\,\overleftarrow{\overline{z}}$	$\langle 2, 1, 1\rangle$	$\frac{1}{2}\cdot\frac{2}{3}\cdot\frac{1}{5}=\frac{1}{15}$
	$\overrightarrow{x}\,\overrightarrow{y}\,\overrightarrow{\overline{z}}$	$\langle 2, 1, 3\rangle$	$\frac{1}{2}\cdot\frac{2}{3}\cdot\frac{4}{5}=\frac{4}{15}$

Fig. 1. Program P and its probabilistic concrete semantics

$\Omega_{\mathtt{P}} \triangleq \Big\{\omega \in \{\overleftarrow{x}, \overrightarrow{x}\}\cdot\{\overleftarrow{y}, \overrightarrow{y}\}\cdot\{\overleftarrow{z}, \overrightarrow{z}, \epsilon\}\cdot\{\overleftarrow{\overline{z}}, \overrightarrow{\overline{z}}, \epsilon\} \;\Big|\; |\omega| = 3\Big\}$ where $\overleftarrow{x}$ (resp. $\overrightarrow{x}$) denotes the left (resp. right) branch of the first probabilistic choice on x, $\overleftarrow{y}$ (resp. $\overrightarrow{y}$) denotes the left (resp. right) branch of the second probabilistic choice on y, and $\overleftarrow{z}$ and $\overrightarrow{z}$ (resp. $\overleftarrow{\overline{z}}$ and $\overrightarrow{\overline{z}}$) denotes the left or right branch of the third (resp. fourth) probabilistic choice on z. Note that the second probabilistic choice depends on the *value* of x.

We suppose that any scenario is observable, so observable properties are simply $\wp(\Omega_{\mathtt{P}})$, and $\sum_{\omega\in\Omega_{\mathtt{P}}}\mu(\{\omega\}) = 1$. The probability that z = 3 is $\frac{2}{5}$ since $\Phi = \{\langle x, y, z\rangle \in \mathbb{Z}^3 \mid z = 3\}$ and $\mathbf{Pr}(S_p[\![\mathtt{P}]\!] \in \Phi) = \frac{2}{15} + \frac{4}{15} = \frac{2}{5}$. □

3.2 Fixpoint Semantics

This formalization allows us to give an easy definition of probabilistic semantics as fixpoints. Indeed, let $F_\omega : \mathcal{D} \longrightarrow \mathcal{D}$ denote the fixpoint semantic transformer for the (non)-deterministic program $\mathtt{P}(\omega)$ such that $S_p[\![\mathtt{P}]\!](\omega) = \mathrm{lfp}^{\preceq} F_\omega$. Now define the lifted operator $F_p : (\Omega \to \mathcal{D}) \longrightarrow (\Omega \to \mathcal{D})$ as $F_p(\lambda\,\omega \bullet X_\omega) \triangleq \lambda\,\omega \bullet F_\omega(X_\omega)$. It easily follows from the definition that $S_p[\![\mathtt{P}]\!] = \mathrm{lfp}^{\dot{\preceq}} F_p$. Thus, we can use the usual abstract interpretation framework since semantics are still fixpoints.

Definition 3 (Probabilistic fixpoint semantics). Let $\langle\mathcal{D}, \preceq\rangle$ be a cpo, $\langle\Omega, \mathcal{E}, \mu\rangle$ where $\mathcal{E} \subseteq \wp(\Omega)$ is a probabilistic space, $F[\![\mathtt{P}]\!] : \Omega \longrightarrow \mathcal{D} \longrightarrow \mathcal{D}$ be a pointwise continuous transformer for program P. The probabilistic fixpoint semantics of P is $S_p[\![\mathtt{P}]\!] \triangleq \mathrm{lfp}^{\dot{\preceq}} F_p[\![\mathtt{P}]\!]$ where $\dot{\preceq}$ is the pointwise extension of $\preceq$ and the probabilistic transformer is $F_p[\![\mathtt{P}]\!](s_{\mathcal{P}})\omega \triangleq F[\![\mathtt{P}]\!](\omega)(s_{\mathcal{P}}(\omega))$ such that $F_p[\![\mathtt{P}]\!] : \mathcal{D}_p \longrightarrow \mathcal{D}_p$. □

Lemma 1. Under the conditions of Def. 1 and 3, $S_p[\![\mathtt{P}]\!] \triangleq \mathrm{lfp}^{\dot{\preceq}} F_p[\![\mathtt{P}]\!] = \lambda\,\omega \bullet \mathrm{lfp}^{\preceq} F[\![\mathtt{P}]\!](\omega)$ is a probabilistic semantics. □

Example 4 (Probabilistic maximal trace semantics). Let $\langle\Omega, \mathcal{E}, \mu\rangle$ be a probability space, Σ be a set of states, Σ^+ be the non-empty finite sequences of states, $\Sigma^* \triangleq \Sigma^+ \cup \{\epsilon\}$ where ϵ is the *empty trace*, Σ^∞ be infinite sequences of states, $\Sigma^{+\infty} \triangleq \Sigma^+ \cup \Sigma^\infty$, and

$\Sigma^{*\infty} \triangleq \Sigma^* \cup \Sigma^\infty$. The *probabilistic maximal trace semantics* is $S_p^{+\infty}[\![\mathrm{P}]\!] \in \Omega \rightarrowtail \wp(\Sigma^{+\infty})$. For each scenario ω, $S_p^{+\infty}[\![\mathrm{P}]\!]\omega$ describes a finite maximal or infinite execution of program P and, following [4], can be defined in fixpoint form.

Define *sequencing* as $X \mathbin{;} Y \triangleq X^\infty \cup \{\sigma s \sigma' \mid \sigma s \in X^+ \wedge s\sigma' \in Y\}$ where $X^\infty \triangleq X \cap \Sigma^\infty$ and $X^+ \triangleq X \cap \Sigma^+$ and the *restriction* $Y\rceil_X \triangleq \{s\sigma' \in Y \mid \exists \sigma : \sigma s \in X^+\}$ so that $X \mathbin{;} Y = X \mathbin{;} (Y\rceil_X)$. This is extended pointwise to $(X \mathbin{;} Y)\omega \triangleq X(\omega) \mathbin{;} Y(\omega)$. For a while language, we would have ($\mathbb{B} \triangleq \{\mathtt{tt}, \mathtt{ff}\}$, $\mathtt{ff} \Rightarrow \mathtt{tt}$)

$$S_p^{+\infty}[\![\mathtt{skip}]\!]\omega \triangleq \{ss \mid s \in \Sigma\}$$

$$S_p^{+\infty}[\![\mathtt{x} := e]\!]\omega \triangleq \left\{ss[\mathtt{x} := \mathcal{E}[\![e]\!](\omega)s] \;\middle|\; s \in \Sigma\right\}^2, \qquad \mathcal{E}[\![e]\!] : \Omega \rightarrowtail (\Sigma \longrightarrow \Sigma)$$

$$S_p^{+\infty}[\![C_1 ; C_2]\!] \triangleq S_p^{+\infty}[\![C_1]\!] \mathbin{;} S_p^{+\infty}[\![C_2]\!]$$

$$S_p^{+\infty}[\![b]\!]\omega \triangleq \left\{s \;\middle|\; \mathcal{E}[\![b]\!](\omega)s\right\}^3, \qquad \mathcal{E}[\![b]\!] : \Omega \rightarrowtail (\Sigma \longrightarrow \mathbb{B})$$

$$S_p^{+\infty}[\![\mathtt{if}\ b\ \mathtt{then}\ C_1\ \mathtt{else}\ C_2]\!] \triangleq S_p^{+\infty}[\![b]\!] \mathbin{;} S_p^{+\infty}[\![C_1]\!] \mathbin{\dot\cup} S_p^{+\infty}[\![\neg b]\!] \mathbin{;} S_p^{+\infty}[\![C_2]\!]$$

$$S_p^{+\infty}[\![\mathtt{while}\ b\ \mathtt{do}\ C]\!] \triangleq \mathrm{lfp}^{\dot\sqsubseteq}\, \lambda X \bullet S_p^{+\infty}[\![b]\!] \mathbin{\dot\cup} S_p^{+\infty}[\![\neg b]\!] \mathbin{;} S_p^{+\infty}[\![C]\!] \mathbin{;} X$$

where $\sqsubseteq$ is the *computational ordering* on infinite traces of [4] (such that $(X \sqsubseteq Y) \triangleq (X^+ \subseteq Y^+ \wedge X^\infty \supseteq Y^\infty)$ and $\dot\sqsubseteq$ is the pointwise extension of $\sqsubseteq$. We do not specify the dependence on ω which would also be possible as e.g. in the *Semantics 2* of [17]. □

3.3 Probabilistic Concrete Transformers

Observe that in Def. 3, probabilistic transformers are defined pointwise. A transformer $F : \mathcal{D}_p \longrightarrow \mathcal{D}_p$ is the lifting of the non-deterministic transformer for each scenario: for all $s_\mathcal{P} \in \mathcal{D}_p$, $F(s_\mathcal{P})(\omega) = F_\omega(s_\mathcal{P}(\omega))$.

It follows that the different probabilistic transformers F_ω do not need to share any common properties. But if they do (e.g. they describe two slightly different paths in the control flow graph of the probabilistic program), it can be exploited by the analysis.

In particular, this framework implies the very important fact that transformers that do not correspond to probabilistic statements have a particular form: all the F_ω are the same. Indeed, this can be understood by the fact that the evolution of the program after a particular non-probabilistic statement does not depend on what scenario has been chosen at the beginning of the execution.

Example 5. If the statement after `x = random(1,2)` is `x = x+1` and has G as its transformer, then for any ω_i, G_{ω_i} has just the effect of incrementing the value of `x` by one, regardless of the fact that `x` took the value 1 or 2. □

However, the F_ω are distinct in full generality (e.g. it is the case for `x = random(1,2)`).

3.4 Examples of Probabilistic Semantics

Since each possible (non)-deterministic semantics of the probabilistic program is an outcome of a scenario, the framework totally separates the probabilistic behavior (on

2 The valuation $\mathcal{E}[\![e]\!]s$ of a pure expression e in state s does not depend on ω when the expression e is not random (i.e. does not use any random variable and/or statement).

3 The valuation $\mathcal{E}[\![b]\!]s$ of a pure condition b in state s does not depend on ω when the condition b is not random.

the Ω and μ side) from the (non)-deterministic semantic one (located in the $\mathcal{D}$ part). As we will see later, it allows for independent and fruitful abstractions.

Example 6 (Trace to transition system abstraction and profiling). For all $s, s' \in \Sigma$, consider the abstractions $\langle \Omega \rightarrowtail \wp(\Sigma^{+\infty}), \dot{\subseteq}\rangle \xleftarrow[\alpha_s]{\gamma_s} \langle \mathbb{B}, \Leftarrow\rangle$ where $\overrightarrow{\mathsf{reach}}(s) \triangleq \{\sigma s \sigma' \mid \sigma \in \Sigma^* \wedge \sigma' \in \Sigma^{*\infty}\}$ and $\alpha_s(s_{\mathcal{P}}) \triangleq (\exists \omega \in \Omega : s_{\mathcal{P}}(\omega) \in \overrightarrow{\mathsf{reach}}(s))$ as well as $\langle \Omega \rightarrowtail \wp(\Sigma^{+\infty}), \dot{\subseteq}\rangle \xleftarrow[\alpha_{\langle s,\, s'\rangle}]{\gamma_{\langle s,\, s'\rangle}} \langle \mathbb{B}, \Leftarrow\rangle$ where $\overrightarrow{\mathsf{succ}}(s, s') \triangleq \{\sigma s s' \sigma' \mid \sigma \in \Sigma^* \wedge \sigma' \in \Sigma^{*\infty}\}$ and $\alpha_{\langle s,\, s'\rangle}(s_{\mathcal{P}}) \triangleq (\exists \omega \in \Omega : s_{\mathcal{P}}(\omega) \in \overrightarrow{\mathsf{succ}}(s, s'))$. The property that a state $s \in \Sigma$ is definitely reached is $\mathsf{reach}(s) \triangleq \alpha_s(S_p^{+\infty}[\![\mathrm{P}]\!])$ which has probability $\mathbf{Pr}_s \triangleq \mathbf{Pr}(\mathsf{reach}(s))$. The property that a transition $\langle s,\ s'\rangle \in \Sigma^2$ is definitely chosen is $\mathsf{succ}(s, s') \triangleq \alpha_{\langle s,\, s'\rangle}(S_p^{+\infty}[\![\mathrm{P}]\!])$ which has probability $\mathbf{Pr}_{\langle s,\, s'\rangle} \triangleq \mathbf{Pr}(\mathsf{succ}(s, s'))$. We have $\mathbf{Pr}_s = \sum_{s' \in \Sigma} \mathbf{Pr}_{\langle s,\, s'\rangle}$. The probability attached to a transition $\langle s,\ s'\rangle \in \Sigma^2$ is the probability of choosing this transition knowing that execution has reached state s which is the conditional probability $\mathbf{Pr}_{\langle s,\, s'\rangle|s} \triangleq \mathbf{Pr}(\mathsf{succ}(s, s') \mid \mathsf{reach}(s)) = \frac{\mathbf{Pr}_{\langle s,\, s'\rangle}}{\mathbf{Pr}_s}$ when state s is reachable. In practice, this conditional probability can often be estimated by statistical profiling. This probabilistic transition system is the abstract probabilistic semantics of probabilistic programs that exhibit discrete probabilistic choices considered in many papers such as [11, 13, 15, 23]. □

Example 7 (Trace to control flow graph abstraction). Continuing Ex. 4 and 6, consider the case of states which are pairs $\langle c,\ m\rangle$ of a control state $c \in \Gamma$ and a memory state $m \in M$ where Γ is finite. Consider the abstraction $\langle \wp(\Sigma \times \Sigma), \subseteq\rangle \xleftarrow[\alpha_G]{\gamma_G} \langle \wp(\Gamma \times \Gamma), \subseteq\rangle$ of states $\langle c,\ m\rangle$ by their control state c, $\alpha_G(S) \triangleq \{\langle c,\ c'\rangle \mid \exists m, m' \in M : \langle\langle c,\ m\rangle,\ \langle c',\ m'\rangle\rangle \in S\}$. The control flow graph (CFG) abstraction $\alpha_G \circ \alpha_\tau$ collects control transitions along traces of T. Similar to Ex. 6, the probability attached to an arc $\langle c,\ c'\rangle \in \Gamma^2$ is the probability of choosing this arc knowing that control has reached c which is the conditional probability $\mathbf{Pr}_{\langle c,\, c'\rangle|c} \triangleq \mathbf{Pr}(\mathsf{succ}(c, c') \mid \mathsf{reach}(c))$ when c is reachable. Compilers construct over-approximations of this CFG syntactically (not taking e.g. conditionals hence code unreachability into account) and often unsoundly (e.g. considering equiprobability of branches or using profiling). □

Ex. 8 below shows that instead of the trace semantics of Ex. 4 we could have considered as well any denotational, predicate transformer, or axiomatic semantics in the abstract interpretation hierarchy of semantics [4].

Example 8 (Probabilistic abstract semantics). Let $\langle \Omega, \mathcal{E}, \mu\rangle$ be a probability space and $\mathrm{lfp}^{\dot{\preceq}}\, F_p[\![\mathrm{P}]\!]$ where $F_p : C_p \longrightarrow C_p$ be the probabilistic concrete fixpoint semantics based on the classical concrete semantics $\mathrm{lfp}^{\preceq}\, F_\omega$ where $\langle C, \preceq\rangle$ is a cpo and $F_\omega : C \longrightarrow C$ for all $\omega \in \Omega$. Consider the classical abstraction $\langle C, \preceq\rangle \xleftarrow[\alpha]{\gamma} \langle \mathcal{A}, \sqsubseteq\rangle$. Let $\mathrm{lfp}^{\dot{\sqsubseteq}}\, F_p^\sharp$ where $F_p^\sharp : \mathcal{A}_p \longrightarrow \mathcal{A}_p$ be the probabilistic abstract fixpoint semantics based on the classical sound abstract semantics $\mathrm{lfp}^{\preceq}\, F_\omega \preceq \gamma(\mathrm{lfp}^{\sqsubseteq}\, F_\omega^\sharp)$ where $\langle \mathcal{A}, \sqsubseteq\rangle$ is a cpo and $F_\omega^\sharp : \mathcal{A} \longrightarrow \mathcal{A}$. Then $\mathrm{lfp}^{\dot{\preceq}}\, F_p \dot{\preceq} \gamma_{\mathcal{P}}(\mathrm{lfp}^{\dot{\sqsubseteq}}\, F_p^\sharp)$ so that the probabilistic lifting of a sound classical abstraction is sound in the sense that in scenario ω, the abstract semantics is $(\mathrm{lfp}^{\dot{\sqsubseteq}}\, F_p^\sharp)(\omega) = \mathrm{lfp}^{\sqsubseteq}\, F_\omega^\sharp$. □

$\mathcal{D}$			semantics domain
$\wp(\mathcal{D})$			semantic property domain
$\mathcal{D}_p$	$\triangleq$	$\Omega \rightarrowtail \mathcal{D}$	probabilistic semantics domain
$\mathcal{D}_p^V$	$\subseteq$	$\mathcal{D}_p$	downsized probabilistic semantic domain
$\wp\left(\mathcal{D}_p\right)$	$=$	$\wp(\Omega \rightarrowtail \mathcal{D})$	probabilistic property domain
$\wp\left(\mathcal{D}_p^V\right)$	$\subseteq$	$\wp\left(\mathcal{D}_p\right)$	downsized probabilistic property domain
$\wp(\mathcal{D})_p$	$\triangleq$	$\Omega \rightarrowtail \wp(\mathcal{D})$	collecting semantics domain
$\wp(\mathcal{D})_p^V$	$\subseteq$	$\wp(\mathcal{D})_p$	downsized collecting semantic domain
$\wp\left(\wp(\mathcal{D})_p^V\right)$			properties of collecting semantics domain
$\mathcal{I}_{\dot\subseteq}(\wp(\mathcal{D})_p^V)$			downset properties of collecting semantics domain
$\wp\left(\wp(\mathcal{D})_p^V\right)/_{\dot\equiv}$			probabilistic concrete collecting semantics domain

Fig. 2. Concrete and abstract semantics domains

In practice, the simple abstractions considered in Ex. 8 are not powerful enough, in particular because Ω is in general infinite and needs further abstractions and we want to consider more general probabilistic properties as defined in next Sect. 4.

4 Probabilistic Concrete Collecting Semantics

The concrete/abstract semantics domains introduced here are summarized in Fig. 2.

4.1 Definition

Concrete properties of programs are elements of the usual concrete domain: the powerset of the program semantics domain, denoted by $\wp\left(\mathcal{D}_p\right) = \wp(\Omega \rightarrowtail \mathcal{D})$. The logical implication order is $\subseteq$.

Definition 4 (Probabilistic concrete collecting semantics). Under the conditions of Def. 1, the *probabilistic concrete property domain* is the complete lattice $\langle\wp\left(\mathcal{D}_p\right), \subseteq, \emptyset, \mathcal{D}_p, \cup, \cap\rangle$. The *probabilistic collecting semantics* of a program P is its strongest probabilistic property $\{S_p[\![\mathtt{P}]\!]\}$ [6]. □

The probabilistic concrete property domain $\wp\left(\mathcal{D}_p\right)$ allows us to express any particular probabilistic property.

Example 9 (Probability of a program property). The probabilistic property of verifying a non-probabilistic property $\Gamma \in \wp(\mathcal{D})$ with probability at least 0.7 is:

$$\Phi = \left\{s_{\mathcal{P}} \in \mathcal{D}_p \;\middle|\; \mathbf{Pr}(s_{\mathcal{P}} \in \Gamma) \geq 0.7\right\} = \left\{s_{\mathcal{P}} \in \mathcal{D}_p \;\middle|\; \int_{\Omega} \chi_{\Gamma}(s_{\mathcal{P}}(\omega))d\mu(\omega) \geq 0.7\right\}. \square$$

The probabilistic concrete property domain $\wp\left(\mathcal{D}_p\right)$ also makes it possible to express program properties that are specifically probabilistic, as illustrated by the following examples 10 and 11.

Example 10 (Game gain expectation). Assume a gambling program P allows the owner to win or lose some money at the end of its execution. The win or loss amount for a specific program semantics is given by a measurable function $\kappa : \mathcal{D} \rightarrowtail \mathbb{Z}$, $\mathbb{Z}$ having the σ-algebra $\wp(\mathbb{Z})$. Then it is straightforward to define the property that a probabilistic program is on expectation a winning strategy:

$$\Phi' = \left\{ s_{\mathcal{P}} \in \mathcal{D}_p \,\middle|\, \mathbb{E}(\kappa \circ s_{\mathcal{P}}) > 0 \right\} = \left\{ s_{\mathcal{P}} \in \mathcal{D}_p \,\middle|\, \int_{\Omega} \kappa(s_{\mathcal{P}}(\omega))d\mu(\omega) > 0 \right\} . \quad \square$$

Example 11 (Probabilistic temporal logics). The probabilistic μ-calculus of [22] or the linear-time probabilistic temporal logic of [12] describe probabilistic properties of execution traces. So their semantics can be described by (abstractions of) elements of $\wp(\Omega \rightarrowtail \Sigma^{\infty})$. $\square$

Of course, we basically have no effective way to automatically compute an integral on an arbitrary space Ω. This is not a problem since Def. 4 is a concrete semantics which is not required to be computable nor decidable in any way. This undecidability problem will be tackled by considering abstract semantics.

4.2 Downsizing the Concrete Collecting Domain

Allowing semantics to be any measurable function ensures a good expressivity but may be *too precise*. It is often preferable not to distinguish between similar situations. Indeed, making concrete semantics too verbose makes abstractions less precise, because abstract transformers take meaningless concrete semantics into account. It will become clearer when we design abstract transformers in Sect. 5.3.

Example 12. In the case of Ex. 1 of the non-biased coin above, swapping the values of $S_p[\![\mathsf{P}]\!](\omega_1)$ and $S_p[\![\mathsf{P}]\!](\omega_2)$ is impactless: both objects have exactly the same behavior. What changes is that the scenarios do not have the same *meaning* in both cases: in the first case ω_i stands for the *situation when* $x = i$ whereas it stands for the *situation when* $x = 3 - i$ in the other one. $\square$

To overcome this issue, we simply abstract away similar situations by restricting the concrete domain to the *relevant* semantics. It is not possible to define *relevant* formally as it depends on the specific instance of the framework. Therefore, we assume that there exists a *sanity checker*: it is a characteristic function $V : \mathcal{D}_p \longrightarrow \{0, 1\}$ that decides whether a semantics in $\mathcal{D}_p$ is valid, i.e. is actually of interest. The sanity checker V defines the corresponding set $\mathcal{D}_p^V \triangleq \{s_{\mathcal{P}} \in \mathcal{D}_p \mid V(s_{\mathcal{P}}) = 1\}$.

Thus, the *valid/real* concrete semantics domain is $\wp\left(\mathcal{D}_p^V\right)$ instead of $\wp\left(\mathcal{D}_p\right)$. Actually, $\mathcal{D}_p$ is a particular $\mathcal{D}_p^V$ with V accepting everything.

This process of downsizing a domain $\wp\left(\mathcal{D}_p^{V'}\right)$ to a domain $\wp\left(\mathcal{D}_p^V\right)$ when $\mathcal{D}_p^V \subseteq \mathcal{D}_p^{V'}$ (i.e. V is more restrictive than V') is a simple abstraction where the abstraction $\alpha_{V,V'}(S) \triangleq \{s_{\mathcal{P}} \in S \mid V(s_{\mathcal{P}}) = 1\}$ for $S \subseteq \mathcal{D}_p^{V'}$ simply *forgets* every semantics that is not in $\mathcal{D}_p^V$. It is a Galois connection:

$$\langle \wp\left(\mathcal{D}_p^{V'}\right), \subseteq\rangle \underset{\alpha_{V,V'}}{\overset{\gamma_{V,V'}}{\leftrightarrows}} \langle \wp\left(\mathcal{D}_p^{V}\right), \subseteq\rangle \,.$$

Thus, for any sanity checker V, $\wp\left(\mathcal{D}_p^V\right)$ is an abstraction of $\wp\left(\mathcal{D}_p\right)$. The more restrictive is the sanity checker, the more precise the subsequent abstractions will be (see the abstraction of transformers in Sect. 5.3).

5 Probabilistic Abstract Semantics

We explore here three directions to abstract the probabilistic concrete collecting semantics of Sect. 4. The first one (I) in Sect. 5.1 is to abstract on the semantics side, i.e. abstract $\mathcal{D}$ (this is where it is possible to plug existing non-probabilistic analyses). The second (II) in Sect. 5.2 is to abstract the scenario space Ω by losing some precision on the probabilistic part of the semantics. Finally, the third axis (III) in Sect. 5.3 is to abstract the measurable functions representing the semantics by their distributions.

It is a comprehensive description of the way to *lift* any non-probabilistic analysis to the probabilistic setting. For instance, we can then obtain information such as "$x \in [1, 4]$ with probability 0.7" instead of "x is always in $[1, 4]$" which may not be provable without probabilistic hypotheses.

5.1 (I) Abstracting the Semantics

Given a classical abstract interpretation $\langle \wp(\mathcal{D}),\ \subseteq\rangle \xleftrightarrow[\alpha]{\gamma} \langle \mathcal{A},\ \sqsubseteq\rangle$ such as the interval abstraction, we now describe a way to *lift* any such non-probabilistic analysis to the probabilistic setting. The probabilistic properties considered in Sect. 4 belong to $\wp\left(\mathcal{D}_p^V\right) \subseteq \wp(\Omega \rightarrowtail \mathcal{D})$ where classical properties $\wp(\mathcal{D})$ on which to apply classical abstractions do not appear explicitly. So we have to abstract $\wp\left(\mathcal{D}_p^V\right)$ into a probabilistic collecting semantics domain in which classical properties $\wp(\mathcal{D})$ appear explicitly.

An Inadequate Solution. An immediate solution is to take the classical collecting semantics on each scenario, leading to measurable functions in the set $\wp(\mathcal{D})_p \triangleq \Omega \rightarrowtail \wp(\mathcal{D})$ where the σ-algebra taken on $\wp(\mathcal{D})$ is the powerset of the one on $\mathcal{D}$. The natural logical order between these objects is the pointwise order

$$\forall s, s' \in \wp(\mathcal{D})_p^V,\ s \leq s' \text{ iff } s \mathrel{\dot{\subseteq}} s'\ .$$

Indeed, $\leq$ means that a probabilistic semantic property is more precise than another one if it is the case on every scenario.

However, the problem is now that we cannot reason on $\wp(\mathcal{D})_p \triangleq \Omega \rightarrowtail \wp(\mathcal{D})$ in classical logical terms with the logical implication $\subseteq$ because elements are not sets but functions. And there is no simple order that works with further abstractions.

Probabilistic Collecting Semantics. So, to express properties of these objects, as above, we turn to the powerset $\wp\left(\wp(\mathcal{D})_p\right) = \wp(\Omega \rightarrowtail \wp(\mathcal{D}))$, where the implication order is the inclusion order $\subseteq$ on the sets. This leads to the consideration of properties of the pointwise collecting semantics so that we can manipulate properties of semantic properties. For example, the strongest property of a program semantics $S_p[\![\mathrm{P}]\!] \in \mathcal{D}_p = \Omega \rightarrowtail \mathcal{D}$ is $\{\lambda\,\omega \bullet \{S_p[\![\mathrm{P}]\!]\omega\}\}$. It is interesting to note that while this step is implicit in the non-probabilistic case[4] (see Sect. 5.2), it is essential in the probabilistic setting.

The concrete collecting domain may have to be downsized as in Sect. 4.2 by considering $\wp(\mathcal{D})_p^V$ which is the restriction of $\wp(\mathcal{D})_p$ to functions that are coherent with V in the straightforward sense, i.e. any concretization verifies V.

The correspondence between the downsized probabilistic property domain and the properties of the collecting semantics domain is given by the easily proven Galois connection

$$\langle \wp\left(\mathcal{D}_p^V\right),\ \subseteq\rangle \underset{\alpha_{\mathcal{D}}}{\overset{\gamma_{\mathcal{D}}}{\leftrightarrows}} \langle \wp\left(\wp(\mathcal{D})_p^V\right),\ \subseteq\rangle$$

where $\alpha_{\mathcal{D}}$ and $\gamma_{\mathcal{D}}$ are defined for all $S \in \wp\left(\mathcal{D}_p^V\right)$ and $T \in \wp\left(\wp(\mathcal{D})_p^V\right)$ as:

$$\begin{aligned}
\alpha_{\mathcal{D}}(S) &\triangleq \left\{t_{\mathcal{P}} \in \wp(\mathcal{D})_p^V \mid \exists s_{\mathcal{P}} \in S : \forall\omega \in \Omega : t_{\mathcal{P}}(\omega) = \{s_{\mathcal{P}}(\omega)\}\right\} = \left\{\lambda\,\omega \in \Omega \bullet \{s_{\mathcal{P}}(\omega)\} \;\middle|\; s_{\mathcal{P}} \in S\right\}\\
\gamma_{\mathcal{D}}(T) &\triangleq \left\{s_{\mathcal{P}} \in \mathcal{D}_p^V \mid \exists t_{\mathcal{P}} \in T : \forall\omega \in \Omega : s_{\mathcal{P}}(\omega) \in t_{\mathcal{P}}(\omega)\right\} .
\end{aligned}$$

And actually, the only question we are interested in is to know whether a collecting semantics $C \in \wp(\mathcal{D})_p^V$ satisfies a property $S \in \wp\left(\wp(\mathcal{D})_p^V\right)$ or any more precise property, that is $C \in \downarrow S$ where $\downarrow S \triangleq \left\{s' \in \wp(\mathcal{D})_p^V \;\middle|\; \exists s \in S, s' \,\dot{\subseteq}\, s\right\}$ is the downward closed set of S (or *downset*) for $\dot{\subseteq}$. It shows that the properties of interest are downward closed sets $\mathcal{I}_{\dot{\subseteq}}\left(\wp(\mathcal{D})_p^V\right)$ in $\wp\left(\wp(\mathcal{D})_p^V\right)$ themselves ordered by $\subseteq$.

The correspondence between $\langle\wp\left(\wp(\mathcal{D})_p^V\right),\ \subseteq\rangle$ and $\langle\mathcal{I}_{\dot{\subseteq}}\left(\wp(\mathcal{D})_p^V\right),\ \subseteq\rangle$ is a straightforward Galois connection $\langle\wp\left(\wp(\mathcal{D})_p^V\right),\ \subseteq\rangle \underset{\alpha_{\downarrow}}{\overset{\gamma_{\downarrow}}{\leftrightarrows}} \langle\mathcal{I}_{\dot{\subseteq}}\left(\wp(\mathcal{D})_p^V\right),\ \subseteq\rangle$ defined by $\alpha_{\downarrow}(S) \triangleq \downarrow S = \{s' \in \wp(\mathcal{D})_p^V \mid \exists s \in S, s' \,\dot{\subseteq}\, s\}$, and accordingly $\gamma_{\downarrow}(I) \triangleq \{s \mid \forall s' \in \wp(\mathcal{D})_p^V : (s' \,\dot{\subseteq}\, s) \implies s' \in I\}$. The proof is left to the reader.

$C \in \downarrow S$ can also be expressed as $\forall s \in C : \exists s' \in S : s \,\dot{\subseteq}\, s'$. This leads to define a pre-order $\ddot{\subseteq}$ where the Hoare preorder $\ddot{\sqsubseteq}$ is defined for any $\dot{\sqsubseteq}$ as follows

$$\forall S, S' \in \wp\left(\wp(\mathcal{D})_p^V\right), S \,\ddot{\sqsubseteq}\, S' \text{ iff } \forall s \in S : \exists s' \in S' : s \,\dot{\sqsubseteq}\, s' .$$

To get a partial order, it is necessary to quotient by the associated equivalence relation $S \,\ddot{\equiv}\, S' \triangleq S \,\ddot{\sqsubseteq}\, S' \wedge S' \,\ddot{\sqsubseteq}\, S$. In the rest of the paper, we denote by $[S]_{\equiv} \triangleq \{S' \mid S' \equiv S\}$ the equivalence class of the element S for the equivalence relation $\equiv$, or simply $[S]$ when the relation $\equiv$ is obvious from the context.

We have $\langle\mathcal{I}_{\dot{\subseteq}}(\wp(\mathcal{D})_p^V),\ \subseteq\rangle \underset{\ddot{\alpha}_{\mathcal{I}}}{\overset{\ddot{\gamma}_{\mathcal{I}}}{\leftrightarrows}} \langle\wp\left(\wp(\mathcal{D})_p^V\right)/_{\ddot{\equiv}_{\subseteq}},\ \ddot{\subseteq}\rangle$ meaning that the complete downset lattice of initial segments $\langle\mathcal{I}_{\dot{\subseteq}}(\wp(\mathcal{D})_p^V),\ \subseteq\rangle$ is Galois-isomorphic to the complete lattice $\langle\wp\left(\wp(\mathcal{D})_p^V\right)/_{\ddot{\equiv}_{\subseteq}},\ \ddot{\subseteq}\rangle$ where $\ddot{\alpha}_{\mathcal{I}}(I) \triangleq [I]_{\ddot{\equiv}_{\subseteq}}$ and $\ddot{\gamma}_{\mathcal{I}}([S]_{\ddot{\equiv}_{\subseteq}}) \triangleq \{s \mid \exists s' \in S : s \,\dot{\subseteq}\, s'\}$. The proof is left to the reader.

[4] When $\Omega = \{\bullet\}$, $\wp(\Omega \rightarrowtail \mathcal{D})$ is isomorphic to $\wp(\mathcal{D})$, so we essentially get $\wp(\wp(\mathcal{D}))$ which, in the classical case, is often abstracted into $\wp(\mathcal{D})$ by $\langle\wp(\wp(\mathcal{D})),\ \subseteq\rangle \underset{\alpha_{\cup}}{\overset{\gamma_{\cup}}{\leftrightarrows}} \langle\wp(\mathcal{D}),\ \subseteq\rangle$ where $\alpha_{\cup}(P) \triangleq \bigcup P$ and $\gamma_{\cup}(Q) \triangleq \wp(Q)$, which amounts to taking initial segments for the order $\subseteq$. See Sect. 5.2 for more details.

The two visions $\langle \mathcal{I}_{\dot\subseteq}(\wp(\mathcal{D})^V_p), \subseteq\rangle \underset{\ddot\alpha_{\mathcal{I}}}{\overset{\ddot\gamma_{\mathcal{I}}}{\leftrightarrows}} \langle \wp\left(\wp(\mathcal{D})^V_p\right)/_{\ddot{=}}, \ddot\subseteq\rangle$ are equivalent, but we find the "$\ddot\subseteq$-approach" much more intuitive for the rest of this paper. It accounts to looking at sets of properties simply as "what may happen is over-approximated by these elements" instead of "everything that can happen is to be found in this set".

Example 13. Consider Ω and the interval property $\Gamma = \{\lambda\omega \bullet x \in [1, 10]\}$ (i.e. the set of mesurable functions where for each scenario ω, x is in $[1, 10]$) where $\sqsubseteq$ is interval inclusion. Let the program semantics be $S_p\llbracket \mathtt{P} \rrbracket = \{\lambda\omega \bullet x \in [3, 3], \lambda\omega \bullet x \in [7, 7]\}$. The fact that program "P satisfies property Γ" is $\left[S_p\llbracket \mathtt{P}\rrbracket\right] \mathrel{\ddot\sqsubseteq} [\Gamma]$, i.e. $S_p\llbracket \mathtt{P}\rrbracket \mathrel{\ddot\sqsubseteq} \Gamma$ or equivalently $\forall s \in S_p\llbracket \mathtt{P}\rrbracket : \exists s' \in \Gamma : s \mathrel{\dot\sqsubseteq} s'$ that is $\forall s \in S_p\llbracket \mathtt{P}\rrbracket : \exists s' \in \Gamma : \forall\omega \in \Omega : s(\omega) \sqsubseteq s'(\omega)$ which holds since $[3, 3] \sqsubseteq [1, 10]$ and $[7, 7] \sqsubseteq [1, 10]$. Note that we do *not* have the inclusion $\{S_p\llbracket \mathtt{P}\rrbracket\} \subseteq \Gamma$, so the Hoare order is really what is meaningful for us. □

In particular, a set with only $\top$ is larger than any other one. The above explanations justify the following definition.

Definition 5 (Probabilistic concrete collecting semantics domain). The *probabilistic concrete collecting semantics domain* is

$$\langle \wp\left(\wp(\mathcal{D})^V_p\right)/_{\ddot{=}}, \ddot\subseteq\rangle\,. \qquad \square$$

This will be the base domain for the abstractions we describe below, coming from the Galois connection

$$\langle \wp\left(\mathcal{D}^V_p\right), \subseteq\rangle \underset{\alpha_{\mathcal{D}}}{\overset{\gamma_{\mathcal{D}}}{\leftrightarrows}} \langle \wp\left(\wp(\mathcal{D})^V_p\right), \subseteq\rangle \underset{\ddot\alpha_{\mathcal{I}}\circ\alpha_{\downarrow}}{\overset{\gamma_{\downarrow}\circ\ddot\gamma_{\mathcal{I}}}{\leftrightarrows}} \langle \wp\left(\wp(\mathcal{D})^V_p\right)/_{\ddot{=}_{\subseteq}}, \ddot\subseteq\rangle$$

such that lem. 2 below is satisfied.

Lemma 2. Given a probabilistic property $\Phi \in \wp(\mathcal{D})^V_p$, we have

$$\ddot\alpha_{\mathcal{I}} \circ \alpha_{\downarrow} \circ \alpha_{\mathcal{D}}(\Phi) = \left[\left\{\lambda\omega \bullet \{s_{\mathcal{P}}(\omega)\} \;\middle|\; s_{\mathcal{P}} \in \Phi\right\}\right]_{\ddot{=}_{\subseteq}}. \qquad \square$$

Proof. $\ddot\alpha_{\mathcal{I}} \circ \alpha_{\downarrow} \circ \alpha_{\mathcal{D}}(\Phi)$

$= \ddot\alpha_{\mathcal{I}} \circ \alpha_{\downarrow}\left(\left\{t_{\mathcal{P}} \in \wp(\mathcal{D})^V_p \;\middle|\; \exists s'_{\mathcal{P}} \in \Phi : \forall\omega \in \Omega : t_{\mathcal{P}}(\omega) = \{s'_{\mathcal{P}}(\omega)\}\right\}\right)$ $\wr$def. $\alpha_{\mathcal{D}}$ and $\wp(\mathcal{D})^V_p$$\wr$

$= \ddot\alpha_{\mathcal{I}}\left(\left\{s_{\mathcal{P}} \in \wp(\mathcal{D})^V_p \;\middle|\; \exists t_{\mathcal{P}} \in \wp(\mathcal{D})^V_p : \exists s'_{\mathcal{P}} \in \Phi : \forall\omega \in \Omega : t_{\mathcal{P}}(\omega) = \{s'_{\mathcal{P}}(\omega)\} \wedge \forall\omega \in \Omega : s_{\mathcal{P}}(\omega) \subseteq t_{\mathcal{P}}(\omega)\right\}\right)$ $\wr$def. $\alpha_{\downarrow}$, $\wp(\mathcal{D})^V_p$ and $\dot\subseteq$$\wr$

$= \left[\left\{s_{\mathcal{P}} \in \wp(\mathcal{D})^V_p \;\middle|\; \exists s'_{\mathcal{P}} \in \Phi : \forall\omega \in \Omega : s_{\mathcal{P}}(\omega) \subseteq \{s'_{\mathcal{P}}(\omega)\}\right\}\right]_{\ddot{=}_{\subseteq}}$ $\wr$set theory and def. $\ddot\alpha_{\mathcal{I}}$$\wr$

$= \left[\left\{\lambda\omega \bullet \emptyset\right\} \cup \left\{\lambda\omega \bullet \{s'_{\mathcal{P}}(\omega)\} \;\middle|\; s'_{\mathcal{P}} \in \Phi\right\}\right]_{\ddot{=}_{\subseteq}}$

$\wr$since $s_{\mathcal{P}}(\omega) \subseteq \{s'_{\mathcal{P}}(\omega)\}$ implies $s_{\mathcal{P}}(\omega) = \emptyset$ or $s_{\mathcal{P}}(\omega) = \{s'_{\mathcal{P}}(\omega)\}$$\wr$

$= \left[\left\{\lambda\omega \bullet \{s'_{\mathcal{P}}(\omega)\} \;\middle|\; s'_{\mathcal{P}} \in \Phi\right\}\right]_{\ddot{=}_{\subseteq}}$ $\wr$def. $\ddot{=}_{\subseteq}$.$\wr$ □

Example 14 (Probabilistic maximal trace collecting semantics). Continuing Ex. 4, the probabilistic maximal trace semantics is $S^{+\infty}_p\llbracket \mathtt{P}\rrbracket \in \Omega \rightarrowtail \mathcal{D}$ where $\mathcal{D} \triangleq \wp(\Sigma^{+\infty})$ so that the *probabilistic maximal powertraces collecting semantics* is $S^{\{\{+\infty\}\}}_p\llbracket \mathtt{P}\rrbracket \triangleq \ddot\alpha_{\mathcal{I}} \circ \alpha_{\downarrow} \circ \alpha_{\mathcal{D}}(\{S^{+\infty}_p\llbracket \mathtt{P}\rrbracket\})$ proving, by Lem. 2, that

$$S^{\{\{+\infty\}\}}_p\llbracket \mathtt{P}\rrbracket = \left[\left\{\lambda\omega \bullet \{S^{+\infty}_p\llbracket \mathtt{P}\rrbracket(\omega)\}\right\}\right]_{\ddot{=}_{\sqsubseteq}} \in \wp(\Omega \rightarrowtail \wp(\wp(\Sigma^{+\infty})))/_{\ddot{=}}. \qquad \square$$

Lemma 3. A probabilistic semantics $s_{\mathcal{P}} \in \mathcal{D}_p$ satisfies a probabilistic property $\Phi \in \wp(\mathcal{D})_p^V$ if and only if $s_{\mathcal{P}} \in \gamma_{\mathcal{D}}(\Phi)$ if and only if $\ddot{\alpha}_{\mathcal{I}} \circ \alpha_{\downarrow} \circ \alpha_{\mathcal{D}}(\{s_{\mathcal{P}}\}) \mathrel{\ddot{\subseteq}} \left[\{\Phi\}\right]_{\ddot{\equiv}_{\subseteq}}$. $\square$

The proof is straightforward from the definitions and is left to the reader.

Example 15 (Probability of trace properties). Continuing Ex. 4, the probability that the trace semantics $S_p^{+\infty}[\![\mathrm{P}]\!]$ satisfies an observable property $\Phi \in \mathcal{F} \subseteq \wp(\wp(\Sigma^{+\infty}))$ (such as determinism $\Phi = \{\{\sigma\} \mid \sigma \in \Sigma^{+\infty}\}$) is given by the distribution $S_p^{+\infty}[\![\mathrm{P}]\!](\mu) : \mathcal{F} \longrightarrow [0,1]$ such that $S_p^{+\infty}[\![\mathrm{P}]\!](\mu)\Phi = \mathbf{Pr}\left(S_p^{+\infty}[\![\mathrm{P}]\!] \in \Phi\right) = \mathbf{Pr}\left(\forall\omega : S_p^{+\infty}[\![\mathrm{P}]\!](\omega) \in \Phi\right) = \mathbf{Pr}\left(S_p^{+\infty}[\![\mathrm{P}]\!](\omega) \in \{\lambda\,\omega \bullet \Phi' \mid \Phi' \subseteq \Phi\}\right) = \mathbf{Pr}\left(S_p^{+\infty}[\![\mathrm{P}]\!](\omega) \in {\downarrow}\{\lambda\,\omega \bullet \Phi\}\right)$ which, by Lem. 3, is

$$\mathbf{Pr}\left(\ddot{\alpha}_{\mathcal{I}} \circ \alpha_{\downarrow} \circ \alpha_{\mathcal{D}}(\{s_{\mathcal{P}}\}) \mathrel{\ddot{\subseteq}} \left[\{{\downarrow}\{\lambda\,\omega \bullet \Phi\}\}\right]_{\ddot{\equiv}_{\sqsubseteq}}\right) = \mathbf{Pr}\left(\ddot{\alpha}_{\mathcal{I}} \circ \alpha_{\downarrow} \circ \alpha_{\mathcal{D}}(\{s_{\mathcal{P}}\}) \mathrel{\ddot{\subseteq}} \left[\{\lambda\,\omega \bullet \Phi\}\right]_{\ddot{\equiv}_{\sqsubseteq}}\right)$$

$$= \int_{\Omega} \chi_{\left[\{\lambda_{\omega} \bullet \Phi\}\right]_{\ddot{\equiv}_{\sqsubseteq}}} (\ddot{\alpha}_{\mathcal{I}} \circ \alpha_{\downarrow} \circ \alpha_{\mathcal{D}} \circ S_p^{+\infty}[\![\mathrm{P}]\!](\omega))\mathrm{d}\mu(\omega).$$

$\square$

Semantics Abstraction. Now that we gained access to semantic properties, we can generalize $\langle\wp(\mathcal{D}),\ \subseteq\rangle$ to any concrete domain $\langle C,\ \leq\rangle$. We assume that we have a Galois connection with an abstract domain $\mathcal{A}$: $\langle C,\ \leq\rangle \xleftrightarrow[\alpha]{\gamma} \langle\mathcal{A},\ \sqsubseteq\rangle$ as mentioned above. However, it is required that C and $\mathcal{A}$ are measurable spaces (as before, their σ-algebra express observable behaviors), and that α and γ are measurable functions.

The semantics abstraction is now by composition. Thus, noting $C_p = \Omega \rightarrowtail C$ and $\mathcal{A}_p = \Omega \rightarrowtail \mathcal{A}$, the concrete and abstract semantics domains are $\langle\wp\left(C_p\right)/_{\ddot{\equiv}_{\leq}},\ \ddot{\leq}\rangle$ and $\langle\wp\left(\mathcal{A}_p\right)/_{\ddot{\equiv}_{\sqsubseteq}},\ \ddot{\sqsubseteq}\rangle$.

The abstraction is defined by composition in terms of elements of C_p and $\mathcal{A}_p$, and it is then lifted to powersets and equivalence classes to be coherent with the domains just mentioned. So, for $s_{\mathcal{P}} \in C_p$, $\alpha \circ s_{\mathcal{P}} \in \mathcal{A}_p$ and conversely, if $t_{\mathcal{P}} \in \mathcal{A}_p$, then $\gamma \circ t_{\mathcal{P}} \in C_p$. It defines the Galois connection

$$\langle\wp\left(C_p\right)/_{\ddot{\equiv}_{\leq}},\ \ddot{\leq}\rangle \xleftrightarrow[\ddot{\alpha}_{\alpha}]{\ddot{\gamma}_{\alpha}} \langle\wp\left(\mathcal{A}_p\right)/\ \ddot{\equiv}_{\sqsubseteq},\ \ddot{\sqsubseteq}\rangle$$

pointwise where

$$\ddot{\alpha}_{\alpha} \triangleq \lambda[S]\bullet\left[\left\{\lambda\,\omega\bullet\alpha\circ s_{\mathcal{P}}(\omega)\ \middle|\ s_{\mathcal{P}} \in S\right\}\right]_{\ddot{\equiv}_{\sqsubseteq}} \quad \ddot{\gamma}_{\alpha} \triangleq \lambda[T]\bullet\left[\left\{s_{\mathcal{P}} \in C_p\ \middle|\ \exists t \in T,\ s_{\mathcal{P}} \mathrel{\dot{\leq}} \gamma\circ t\right\}\right]_{\ddot{\equiv}_{\leq}}$$

(it is easy to verify that these functions are well-defined, i.e. they do not depend on the representent picked for $[S]$ and $[T]$, and that they are properly measurable).

Example 16 (Set of traces to traces abstraction). Continuing Ex. 4 and 14, consider the abstraction of sets of traces into traces $\langle\wp(\wp(\Sigma^{+\infty})),\ \subseteq\rangle \xleftrightarrow[\alpha_{\cup}]{\gamma_{\cup}} \langle\wp(\Sigma^{+\infty}),\ \subseteq\rangle$ with $\alpha_{\cup}(S) \triangleq \bigcup S$ and $\gamma_{\cup}(T) = \wp(T)$ as first performed in most classical static analyses. The *probabilistic trace collecting semantics* is

$$\begin{aligned}
&S_p^{\{+\infty\}}[\![\mathtt{P}]\!] \triangleq \ddot{\alpha}_{\alpha_\cup}(S_p^{\{\{+\infty\}\}}[\![\mathtt{P}]\!]) \in \wp(\Omega \rightarrowtail \wp(\Sigma^{+\infty}))/_{\doteq_\subseteq} \\
&= \ddot{\alpha}_{\alpha_\cup}\Big(\Big[\big\{\lambda\,\omega \bullet \{S_p^{+\infty}[\![\mathtt{P}]\!](\omega)\}\big\}\Big]_{\doteq_\subseteq}\Big) && \text{⦃def. } S_p^{\{\{+\infty\}\}}[\![\mathtt{P}]\!] \text{ in Ex. 14⦄} \\
&= \Big[\Big\{\lambda\,\omega \bullet \alpha_\cup(s_{\mathcal{P}}(\omega)) \;\Big|\; s_{\mathcal{P}} \in \Big[\big\{\lambda\,\omega \bullet \{S_p^{+\infty}[\![\mathtt{P}]\!](\omega)\}\big\}\Big]_{\doteq_\subseteq}\Big\}\Big]_{\doteq_\subseteq} && \text{⦃def. } \ddot{\alpha}_{\alpha_\cup}\text{⦄} \\
&= \Big[\Big\{\lambda\,\omega \bullet \alpha_\cup\big(\{S_p^{+\infty}[\![\mathtt{P}]\!](\omega)\}\big)\Big\}\Big]_{\doteq_\subseteq} && \text{⦃def. } \doteq_\subseteq\text{⦄} \\
&= \Big[\big\{\lambda\,\omega \bullet S_p^{+\infty}[\![\mathtt{P}]\!](\omega)\big\}\Big]_{\doteq_\subseteq} && \text{⦃def. } \alpha_\cup\text{⦄} \quad \square
\end{aligned}$$

Example 17 (Traces to reachability abstraction). Continuing Ex. 4, 14, and 16, consider the reachability abstraction $\langle\wp(\Sigma^{+\infty}),\ \subseteq\rangle \underset{\alpha_r}{\overset{\gamma_r}{\leftrightarrows}} \langle\wp(\Sigma),\ \subseteq\rangle$ such that $\alpha_r(T) \triangleq \{s \in \Sigma \mid \exists\sigma,\sigma' : \sigma s \sigma' \in T\}$ collecting states along traces of T. Applying the above semantics abstraction, the *probabilistic reachability semantics* is

$$\begin{aligned}
&S_p^r[\![\mathtt{P}]\!] \triangleq \ddot{\alpha}_{\alpha_r}(S_p^{\{+\infty\}}[\![\mathtt{P}]\!]) \in \wp(\Omega \rightarrowtail \wp(\Sigma^{+\infty}))/_{\doteq} \\
&= \ddot{\alpha}_{\alpha_r}\Big(\Big[\big\{\lambda\,\omega \bullet S_p^{+\infty}[\![\mathtt{P}]\!](\omega)\big\}\Big]_{\doteq_\subseteq}\Big) && \text{⦃def. } S_p^{\{+\infty\}}[\![\mathtt{P}]\!] \text{ in Ex. 16⦄} \\
&= \Big[\Big\{\lambda\,\omega \bullet \alpha_r(s_{\mathcal{P}}(\omega)) \;\Big|\; s_{\mathcal{P}} \in \Big[\big\{\lambda\,\omega \bullet S_p^{+\infty}[\![\mathtt{P}]\!](\omega)\big\}\Big]_{\doteq_\subseteq}\Big\}\Big]_{\doteq_\subseteq} && \text{⦃def. } \ddot{\alpha}_{\alpha_r}\text{⦄} \\
&= \Big[\Big\{\lambda\,\omega \bullet \alpha_r\big(S_p^{+\infty}[\![\mathtt{P}]\!](\omega)\big)\Big\}\Big]_{\doteq_\subseteq} && \text{⦃def. } \doteq_\subseteq\text{⦄} \\
&= \Big[\Big\{\lambda\,\omega \bullet \{s \in \Sigma \mid \exists\sigma,\sigma' : \sigma s \sigma' \in S_p^{+\infty}[\![\mathtt{P}]\!](\omega)\}\Big\}\Big]_{\doteq_\subseteq} && \text{⦃def. } \alpha_r\text{⦄}
\end{aligned}$$

The probabilistic reachability semantics is therefore the downward closed set of the function taking each scenario to the minimal reachability abstraction of its behavior.

□

Example 18 (Probability of invariance properties). Continuing the trace to reachability abstraction example 17, the probability that a program invariant $I \in \wp(\Sigma)$ holds during execution (assuming that the abstract property I is properly measurable) is

$$\begin{aligned}
&S_p^{+\infty}[\![\mathtt{P}]\!](\mu)(\gamma_r(I)) \\
&\triangleq \mathbf{Pr}(S_p^{+\infty}[\![\mathtt{P}]\!] \in \gamma_r(I)) && \text{⦃def. 2 of property probability⦄} \\
&= \mathbf{Pr}(\ddot{\alpha}_{\mathcal{I}} \circ \alpha_\downarrow \circ \alpha_{\mathfrak{D}}(S_p^{+\infty}[\![\mathtt{P}]\!]) \mathrel{\ddot{\subseteq}} [\{\gamma_r(I)\}]_{\doteq_\subseteq}) && \text{⦃Lem. 3⦄} \\
&= \mathbf{Pr}(\ddot{\alpha}_{\alpha_r} \circ \ddot{\alpha}_{\mathcal{I}} \circ \alpha_\downarrow \circ \alpha_{\mathfrak{D}}(S_p^{+\infty}[\![\mathtt{P}]\!]) \mathrel{\ddot{\subseteq}} [\{\lambda\,\omega \bullet I\}]_{\doteq_\subseteq}) && \text{⦃Galois connexion } \alpha_r, \gamma_r\text{⦄} \\
&= \mathbf{Pr}(S_p^r[\![\mathtt{P}]\!] \mathrel{\ddot{\subseteq}} [\{\lambda\,\omega \bullet I\}]_{\doteq_\subseteq}) && \text{⦃def. } S_p^r[\![\mathtt{P}]\!] \text{ in Ex. 17.⦄}
\end{aligned}$$

Therefore we can define the *invariant probability semantics* $S_i[\![\mathtt{P}]\!] \triangleq \lambda I \bullet \mathbf{Pr}(S_p^{+\infty}[\![\mathtt{P}]\!] \in \gamma_r(I)) = \mathbf{Pr}(S_p^r[\![\mathtt{P}]\!] \mathrel{\ddot{\subseteq}} [\{\lambda\,\omega \bullet I\}])$. An axiomatic definition of the abstract semantics $S_i[\![\mathtt{P}]\!]$ can be calculated from the definition of $S_p^{+\infty}[\![\mathtt{P}]\!]$ in Ex. 4 using standard abstract interpretation techniques. For example

$$\begin{aligned}
&- \; S_i[\![\mathtt{skip}]\!]I \triangleq \mathbf{Pr}(S_p^{+\infty}[\![\mathtt{skip}]\!] \in \gamma_r(I)) && \text{⦃def. } S_i[\![\mathtt{skip}]\!]\text{⦄} \\
&= \mathbf{Pr}(\{ss \mid s \in \Sigma\} \in \gamma_r(I)) && \text{⦃def. } S_p^{+\infty}[\![\mathtt{skip}]\!]\text{⦄} \\
&= \mathbf{Pr}(s \in I) = \int_\Omega \chi_I \,\mathrm{d}\mu && \text{⦃def. } \gamma_r \text{ and distributions⦄} \\
&- \; S_i[\![C_1\,;C_2]\!]I \triangleq \mathbf{Pr}(S_p^{+\infty}[\![C_1\,;C_2]\!] \in \gamma_r(I)) && \text{⦃def. } S_i[\![C_1\,;C_2]\!]\text{⦄} \\
&= \mathbf{Pr}(S_p^{+\infty}[\![C_1]\!] \mathbin{⨾} S_p^{+\infty}[\![C_2]\!] \in \gamma_r(I)) && \text{⦃def. } S_p^{+\infty}[\![C_1\,;C_2]\!]\text{⦄}
\end{aligned}$$

$= \mathbf{Pr}(S_p^{+\infty}[\![C_1]\!] \in \gamma_r(I) \wedge S_p^{+\infty}[\![C_2]\!] \in \gamma_r(I))$ ⦃def. $\fatsemi$ and γ_r⦄

$= \mathbf{Pr}(S_p^{+\infty}[\![C_1]\!] \in \gamma_r(I)) \times \mathbf{Pr}(S_p^{+\infty}[\![C_2]\!] \in \gamma_r(I))$ ⦃Probability theory⦄

$= S_i[\![C_1]\!]I \times S_i[\![C_2]\!]I$ ⦃def. $S_i[\![C]\!]$⦄

and similarly for other commands using fixpoint abstraction [6, Th. 7.1.0.4-(3)] for loops. □

The series of examples 4, 14, 16, 17, and 18 shows that the probabilistic abstract interpretation framework is compositional in that the abstraction of an abstraction is an abstraction. Sec. 5.2 below makes the link with classical static analysis approaches.

5.2 (II) Abstracting the Scenario Space Ω

Definition. The scenario space Ω is chosen arbitrarily among all the measured spaces that could describe the random behavior at hand. Several Ω spaces could describe the same probabilistic system, or we might want to "group" several scenarios together because they look the same from the level of details we need.

Satisfyingly enough, it is possible to change the Ω space by a simple abstraction. Let Ω be a measurable space with a distribution μ, and Ω' be a set. Suppose there exists a surjective mapping $q : \Omega \twoheadrightarrow \Omega'$, then it is possible to abstract a probabilistic semantics expressed over Ω by one over Ω'.

First, we define the observable events on Ω' as the smallest set making q measurable. We note $\mathcal{A}_p(\Omega) \triangleq \Omega \rightarrowtail \mathcal{A}$ for the probabilistic semantics domain over Ω and $\langle \mathcal{A}, \sqsubseteq, \sqcup \rangle$. Then

$$\langle \wp\left(\mathcal{A}_p(\Omega)\right)/_{\ddot{\equiv}}, \ddot{\sqsubseteq}\rangle \underset{\alpha_{\Omega,\Omega'}}{\overset{\gamma_{\Omega,\Omega'}}{\leftrightarrows}} \langle \wp\left(\mathcal{A}_p(\Omega')\right)/_{\ddot{\equiv}}, \ddot{\sqsubseteq}\rangle$$

where

$$\gamma_{\Omega,\Omega'} \triangleq \left[\lambda [S] \bullet \left\{s_{\mathcal{P}} \in \mathcal{A}_p \;\middle|\; \exists s'_{\mathcal{P}} \in S' : \forall \omega \in \Omega : s_{\mathcal{P}}(\omega) \sqsubseteq s'_{\mathcal{P}}(q(\omega))\right\}\right]_{\ddot{\equiv}_{\sqsubseteq}}$$
$$\alpha_{\Omega,\Omega'} \triangleq \left[\lambda [S'] \bullet \left\{\lambda \omega' \in \Omega' \bullet \sqcup_{\omega \in q^{-1}(\{\omega'\})} s(\omega) \;\middle|\; s \in S\right\}\right]_{\ddot{\equiv}_{\sqsubseteq}}$$

and it can be verified that these definitions do not depend on the chosen representants S and S'.

The law μ' on Ω' is the image of the law μ by q, i.e. for all measurable sets $X' \subseteq \Omega', \mu'(X') = \mu(q^{-1}(X'))$.

Non-Determinism as an Abstraction. Merging scenarios by using a surjective q that identifies their image amounts to forgetting the probabilistic information on them, and seeing them just as a "new scenario". It means that when in the new compound scenario, the program can actually *non-deterministically* be in either one of the initial scenarios. That is why all their semantics are joined in the $\alpha_{\Omega,\Omega'}$ definition, and the probability of the new scenario is the sum of the probabilities of the source ones.

Thus, non-determinism is simply expressible in our framework by the Ω-abstraction. And while non-determinism is expressible between some scenarios, all the other probabilistic informations about the other scenarios are kept unchanged and used. Moreover, the non-determinism impacts as little as possible because the new compound scenario still behaves well with respect to the rest of the semantics.

Classical Abstract Interpretation as an Abstraction. Along those lines, it is natural to find classical abstract interpretation as a limit Ω-abstraction: forgetting all probabilistic information in the semantics should give back the classical abstract interpretation framework.

It is exactly what happens if Ω' is taken as a singleton $\Omega_\bullet = \{\bullet\}$ with the trivial probability measure on it (in this case, the semantics describes *anything* that can happen as the join of all possible outcomes, without knowing what is the probability for each actual behavior). We call this abstraction the "safe abstraction".

$$\langle \wp\left(\mathcal{A}_p(\Omega)\right)/_{\ddot{\equiv}_{\sqsubseteq}}, \ddot{\sqsubseteq}\rangle \xrightleftharpoons[\alpha_{\Omega,\{\bullet\}}]{\gamma_{\Omega,\{\bullet\}}} \langle \wp\left(\mathcal{A}_p(\Omega_\bullet)\right)/_{\ddot{\equiv}_{\sqsubseteq}}, \ddot{\sqsubseteq}\rangle$$

where $\mathcal{A}_p(\Omega_\bullet) \triangleq \{\bullet\} \rightarrowtail \mathcal{A}$ is isomorphic to $\mathcal{A}$, and so $\langle \wp\left(\mathcal{A}_p(\Omega_\bullet)\right)/_{\ddot{\equiv}_{\sqsubseteq}}, \ddot{\sqsubseteq}\rangle$ is order-isomorphic to $\langle \wp\left(\mathcal{A}\right)/_{\ddot{\equiv}_{\sqsubseteq}}, \ddot{\sqsubseteq}\rangle$.

In classical abstract interpretation, we are usually just interested in properties such as $S[\![\mathrm{P}]\!]^\sharp \sqsubseteq Q$. It means that when we have a semantics that can be any element of $Q_{\mathcal{P}} \in \wp\left(\mathcal{A}\right)$, we say that the most precise abstract state describing it is $\sqcup Q_{\mathcal{P}}$. It amounts to applying the following *join-abstraction*

$$\langle \wp\left(\mathcal{A}\right)/_{\ddot{\equiv}}, \ddot{\sqsubseteq}\rangle \xrightleftharpoons[\alpha_{\sqcup}]{\gamma_{\sqcup}} \langle \mathcal{A}, \sqsubseteq\rangle$$

where

$$\alpha_{\sqcup} \triangleq \lambda\, [S] \bullet \bigsqcup_{Q\in S} Q \quad \text{and} \quad \gamma_{\sqcup} \triangleq \lambda\, Q \bullet [\downarrow Q]_{\ddot{\equiv}} \ .$$

This Galois connection abstracts the probabilistic abstract interpretation framework back to the classical abstract interpretation framework, an abstraction which is not always expressible in other more specific frameworks e.g. [24, 26, 25].

5.3 (III) Abstracting Probabilistic Semantics by Distributions

Law-Abstraction. Starting from the *abstract probabilistic semantics* of Sect. 5.1

$$S_p[\![\mathrm{P}]\!]^\sharp \in \mathcal{A}_p \triangleq \Omega \rightarrowtail \mathcal{A}, \quad \text{where} \quad \langle \mathcal{A}, \sqsubseteq\rangle \text{ is a cpo,}$$

we have the semantic properties in the domain

$$\left[\downarrow\{S_p[\![\mathrm{P}]\!]^\sharp\}\right]_{\ddot{\equiv}_{\sqsubseteq}} \in \wp\left(\mathcal{A}_p\right)/_{\ddot{\equiv}_{\sqsubseteq}} \ .$$

In this semantics, the dependencies between scenarios and the associated abstract semantics have been preserved. But this is something that we may not desire for static analysis because it would lead to combinatorial explosion. One solution considered in Sect. 5.2 is to abstract the scenario space Ω. Another abstraction is to consider the distribution of the abstract semantics, that is, the function giving the probability of any observable abstract property. Remembering only the distribution from a measurable function is actually an abstraction. Note that usual probabilistic analysis tools start actually from (abstractions of) this level of abstraction to build their analysis e.g.

[1, 3, 13, 15, 21, 28, 29], lacking the insight and soundness justifications that we developed above.

The order between the laws should reflect the intuition we have on lattices and logical implication. The information that we need from the distribution is actually restricted to downward closed sets because we want to answer questions like "What is $\mathbf{Pr}(S_p[\![\mathrm{P}]\!]^\sharp \sqsubseteq Q)$?", which is given by the function $\lambda Q \bullet S_p[\![\mathrm{P}]\!]^\sharp(\mu)(\downarrow Q)$ (where $\downarrow$ is this time the classical downward operator in the lattice $\mathcal{A}$).

Thus, we say that a law $\nu \in \mathcal{L}_{\mathcal{A}}$ ($\mathcal{L}_{\mathcal{A}}$ denotes the set of probability laws on $\mathcal{A}$, $\mathcal{L}_{\mathcal{A}} \subseteq \wp(\mathcal{A}) \longrightarrow [0,1]$) is more precise than another one ν' if it puts more weight on the bottom of the abstract lattice $\mathcal{A}$. That is, the logical order between laws on $\mathcal{A}$ is

$$\nu \preceq \nu' \iff \quad \forall Q \in \mathcal{A} \;:\; \nu(\downarrow Q) \geq \nu'(\downarrow Q)$$

The idea behind this logical order is essential to the understanding of the whole approach. As usual, logical orders should reflect that smaller abstract properties imply greater ones. Here, the intuition on the order $\nu \preceq \nu'$ is that ν assigns a higher probability than ν' to more precise properties in $\langle \mathcal{A}, \sqsubseteq \rangle$, so more precise properties have better chances to hold.

Classically, it is safe to approximate $x \in [1,10]$ by $x \in [1,20]$. It is just less precise, because $[1,10] \subseteq [1,20]$. In the probabilistic case, the analogous situation would be "$x \in [1,10]$ is true with probability one", approximated by "$x \in [1,10]$ with probability $1/2$ and $x \in [1,20]$ with probability $1/2$". Of course, the former situation is more precise than the second one, and this is reflected by the $\preceq$ order.

Formally, the $\preceq$ order checks that anywhere in the lattice, the most precise law is at least as precise as the other one, with at least as much probability.

It is interesting to note that as we mentioned before, if Ω is shrunk to a singleton $\Omega_\bullet$, the only valid probabilities for properties are 0 and 1, and the $\preceq$ order boils down to $\sqsubseteq$ between abstract states and gives back the classical abstract interpretation framework.

The order $\preceq$ is then lifted to the powersets by using the Hoare order once again, with $N, N' \subseteq \mathcal{L}_{\mathcal{A}}$

$$N \ddot{\preceq} N' \iff \forall \nu \in N \;:\; \exists \nu' \in N' \;:\; \nu \preceq \nu' \;.$$

In fact, we take for $\mathcal{L}_{\mathcal{A}}$ a subset of the laws on $\mathcal{A}$ because some laws do not have a meaning for the semantics at hand. If a non-biased coin is tossed, it makes no sense to speak of having tails with probability $1/3$. It is not a proper abstract semantics. To circumvent this issue, we restrict from now on $\mathcal{L}_{\mathcal{A}}$ to the elements l that have at least one corresponding function, i.e. a function in $\mathcal{A}_p$ such that $f(\mu) = l$.

We are now ready to define the Galois connection that unifies all of this

$$\langle \wp\left(\mathcal{A}_p\right)/_{\ddot{\equiv}_{\sqsubseteq}}, \ddot{\sqsubseteq} \rangle \xrightleftharpoons[\alpha_{\mathcal{L}}]{\gamma_{\mathcal{L}}} \langle \wp\left(\mathcal{L}_{\mathcal{A}}\right)/_{\ddot{\equiv}_{\preceq}}, \ddot{\preceq} \rangle$$

where $\alpha_{\mathcal{L}} \triangleq \lambda [S]_{\ddot{\equiv}_{\sqsubseteq}} \bullet [\{s(\mu) \mid s \in S\}]_{\ddot{\equiv}_{\preceq}}$ and $\gamma_{\mathcal{L}} \triangleq \lambda [N]_{\ddot{\equiv}_{\preceq}} \bullet \left[\{s \in \mathcal{A}_p \mid s(\mu) \in N\}\right]_{\ddot{\equiv}_{\sqsubseteq}}$. As usual, it is easily shown that these functions are well-defined regardless of the chosen representant of the equivalence classes.

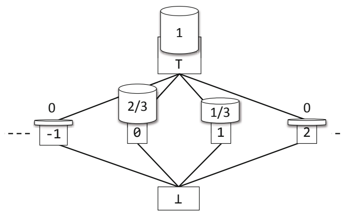
Example 19 (Probabilistic constant propagation). Consider the very simple probabilistic program `P : x = 0` $_{\frac{2}{3}}\oplus$ `x = 1` whose abstract probabilistic semantics is defined by $\Omega = \{\omega_0, \omega_1\}$ and the constant propagation lattice $\mathcal{A} \triangleq \{\bot, \top\} \cup \mathbb{Z}$ ordered by $\forall z \in \mathbb{Z} : \bot \sqsubset z \sqsubset \top$ as

$$S_p[\![\mathrm{P}]\!]^\sharp(\omega_0) \;=\; 0, \qquad \mu(\{\omega_0\}) \;=\; \tfrac{2}{3}, \qquad S_p[\![\mathrm{P}]\!]^\sharp(\omega_1) \;=\; 1, \qquad \mu(\{\omega_1\}) \;=\; \tfrac{1}{3}\,.$$

The strongest probabilistic program property is

$$\left[\downarrow S_p[\![\mathrm{P}]\!]^\sharp\right]_{\ddot{\equiv}} \;=\; \left[\left\{\lambda\omega\bullet\left(\!\!\left[\ \omega = \omega_0 \mathrel{\text{?}} \{\bot, 0\} \mathrel{\text{⸰}} \{\bot\} \ \Big\|\ \omega = \omega_1 \mathrel{\text{?}} \{\bot, 1\} \mathrel{\text{⸰}} \{\bot\}\ \right]\!\!\right)\right\}\right]_{\ddot{\equiv}}$$

The order $\ddot{\sqsubseteq}$ is such that e.g. $[\lambda\omega\bullet\{\bot, 0\}]_{\ddot{\equiv}} \mathrel{\ddot{\sqsubseteq}} [\lambda\omega\bullet\{\bot, 0, \top\}]_{\ddot{\equiv}}$ since $0 \sqsubseteq \top$. We have $[\{\nu\}]_{\ddot{\equiv}_{\preceq}} = \alpha_{\mathcal{L}}\left(\left[\downarrow\{S_p[\![\mathrm{P}]\!]^\sharp\}\right]_{\ddot{\equiv}}\right)$ and $\nu \prec \nu'$ as follows (assuming $Z \subseteq (\mathbb{Z} \setminus \{0, 1\}) \cup \{\top\}$)



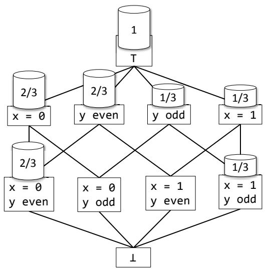
P	$\{\bot, Z\}$	$\{\bot, 0, Z\}$	$\{\bot, 1, Z\}$	$\{\bot, 0, 1, Z\}$
$\nu(\downarrow P)$	0	$\frac{2}{3}$	$\frac{1}{3}$	1
$\nu'(\downarrow P)$	0	0.5	0.2	1

□

Example 20. The final distribution of the constant and parity analysis of a simplified version `P'` of the probabilistic program `P` of Ex. 3 is provided below

```
x = 0 ⅔⊕ x = 1;
if (x = 0) then
  y = 2 ¼⊕ y = 4
else
  y = 1 ⅕⊕ y = 3
```

Program `P'`



□

Example 21. Note that it is not a paradox to have in the abstract, for example :

$$\mathbf{Pr}\,(x \in [0, 10]) > \mathbf{Pr}\,(x \in [0, 5]) + \mathbf{Pr}\,(x \in [5, 10])$$

Indeed the analysis may not have managed to infer the exact value of $\mathbf{Pr}\,(x \in [0, 5])$ by lack of completeness, but only an under-estimation. □

In practice, distributions need only to be considered for atoms of atomic lattices (`x = 0, y even` $_{\frac{2}{3}}\oplus$ `x = 1, y odd` in Ex. 20) and more generally only for the join-irreducible elements. Further examples based on sets of probability distributions are given in [21].

Law-Abstraction Transformers. Along with the abstract domain that we just described, it is essential to construct the corresponding abstract transformers.

They are operators that take as input a (set of) semantic properties distribution and transform it according to their corresponding statement, over-approximating the concrete semantics of the statement.

Let us say that a statement S has a corresponding concrete transformer $F^{\mathsf{S}} : \mathcal{D}_p \longrightarrow \mathcal{D}_p$ defined as $F^{\mathsf{S}}_p(\lambda\,\omega \bullet X_\omega) \triangleq \lambda\,\omega \bullet F^{\mathsf{S}}_\omega(X_\omega)$, following Sec. 3.2.

It follows from this definition that for any $s \in \mathcal{D}^V_p$, the distribution of s is transformed by F^{S} in the following way, where $\Phi \subseteq \mathcal{D}$

$$\mathbf{Pr}\left(F^{\mathsf{S}}(s) \in \Phi\right) = \int_\Omega \chi_\Phi\left(F^{\mathsf{S}}_\omega(s(\omega))\right)\mathrm{d}\mu(\omega) = \int_\Omega \chi_{(F^{\mathsf{S}}_\omega)^{-1}(\Phi)}(s(\omega))\mathrm{d}\mu(\omega) \qquad (1)$$

i.e. to know the probability that a semantic property is verified after applying a transformer, we measure the probability of the scenarios leading to that property after the transformation.

In the general case, this integral cannot be simplified. In particular, it cannot be expressed generally as a function of the input distribution s only. As a consequence, there is no straightforward way to go from a concrete transformer to an abstract one that transforms elements of $\langle\wp(\mathcal{L}_\mathcal{A})/_{\ddot{\equiv}_\leq}, \ddot{\leq}\rangle$, other than by using the classical formula: $(F^{\mathsf{S}})^\sharp = \alpha \circ F^{\mathsf{S}} \circ \gamma$ where α and γ are the appropriate abstraction and concretization functions that link the abstract domain to the most concrete one.

In practice, one has to design the transformers by hand, making sure that they are over-approximations of the above mentioned optimal abstract transformers. Note that this is a process that was made silently in the related works, but taking them as axioms without proving their soundness in respect to the concrete semantics (see e.g. Sect. 7.4 and 7.5).

We see here that the precision of the sanity checker V (see Sect. 4.2) is crucial to the precision of the abstract transformers. Indeed, the smaller the set $F^{\mathsf{S}} \circ \gamma$, the more precise $(F^{\mathsf{S}})^\sharp$ is. It makes sense: if $(F^{\mathsf{S}})^\sharp$ has to be sound with respect to the "right" concrete semantics *and* some "useless" ones, it is less precise than if it just has to account for the right ones. That is why defining precise sanity checkers is so important to easily craft sound and precise abstract transformers.

But the issue is not as problematic as it may seem. Indeed, in the vast majority of cases, equation (1) can be further simplified to only depend upon the distribution of s.

When the transformer corresponds to a *non-random* statement, then by definition the operators F_ω are all equal as seen in Ex. 5, and the equation boils down to

$$\mathbf{Pr}\left(F^{\mathsf{S}}(s) \in \Phi\right) = \int_\Omega \chi_\Phi\left(F^{\mathsf{S}}(s(\omega))\right)\mathrm{d}\mu(\omega) = \int_\Omega \chi_\Phi(F^{\mathsf{S}}(\omega))\mathrm{d}s(\mu)(\omega)$$

where "$\mathrm{d}s(\mu)(\omega)$" denotes that the integral is taken according to the probability measure of s. Thus the new distribution is now computed as a simple function of the distribution of s, an information that is kept in the abstract state in the $\langle\wp(\mathcal{L}_\mathcal{A})/_{\ddot{\equiv}_\leq}, \ddot{\leq}\rangle$ domain.

Of course, to apply to the $\langle\wp(\mathcal{L}_\mathcal{A})/_{\ddot{\equiv}_\leq}, \ddot{\leq}\rangle$ domain, this process has to be lifted pointwise to sets (and thus equivalence classes) — it is straighforward.

Example 22. Suppose that x is a random integer variable in a fixed program, the statement x++ would have such an abstract transformer. Indeed, the action of the statement does not depend on the actual value of x. In any scenario, it increments the value of x by one. Evaluating the above integral, we see that, for instance, the probability of x being 4 after the transformer is the probability of x being 3 before, which is exactly what we expect. □

As we just saw, it is far easier to define abstract transformers for non-random statements than for random ones. So how should we craft transformers for random statements?

First, let us note it is a good thing that non-random transformers are seamlessly lifted to the probabilistic case. It is certainly desirable. On the other hand, building the abstract transformers for random statements requires more knowledge because we have to create a function as precise as possible verifying the soundness equation, without formal indication on how to do it. It looks pretty normal after all, because handling probabilistic behaviors must necessarily imply more work at some point.

That being said, our experience is that in most practical instantiations of the framework, there will not be that many probabilistic constructs to find transformers for (typically, just calls to `rand()`-like functions). For these statements, the probabilistic behavior is well-known, and sound abstract transformers are pretty straightforward to build.

6 Iterating in the Abstract and Branch Prediction

The goal of this section is to show how to instrumentalize all the theory that has been developed so far to build a probabilistic static analyzer. Essentially, it boils down to building as precise abstract transformers as possible for classic programming languages constructs such as conditional and loops. Once this is done, it just remains to use classical abstract interpretation based fixpoint approximation through custom iteration schemes, e.g. [9].

6.1 Conditionals

Knowing the semantic properties distribution after a conditional requires to know as precisely as possible the probability that the condition is actually true or false. It is intuitively clear: the more a branch is likely to be executed, the more it will have an impact on the final outcome.

Formally, assume that $Q \in \mathcal{A}$, $l_s \in \mathcal{L}_{\mathcal{A}}$ is the law of a semantics $s \in \mathcal{D}_p^V$, $\mathtt{S}$ is the statement "`if` b `then` C_1 `else` C_2", and assume that the probability that the condition b is true when evaluated is fixed and equal to p_b, then for any $\Phi \in \wp(\mathcal{A})$

$$\begin{aligned}
&[\![\mathtt{S}]\!](l_s)(\Phi) = \mathbf{Pr}([\![\mathtt{S}]\!](s) \in \Phi) && \wr\text{def. distribution}\wr\\
&= \mathbf{Pr}\Big([\![C_1]\!](s) \in \Phi \wedge [\![\mathtt{b}]\!](s)\Big) \vee ([\![C_2]\!](s) \in \Phi \wedge [\![\neg b]\!](s)) \\
&= \mathbf{Pr}\Big([\![C_1]\!](s) \in \Phi \wedge [\![\mathtt{b}]\!](s)\Big) + \mathbf{Pr}\Big([\![C_2]\!](s) \in \Phi \wedge [\![\neg b]\!](s)\Big) && \wr\text{probability theory}\wr\\
&= p_b \times \mathbf{Pr}\Big([\![C_1]\!](s) \in \Phi \mid [\![\mathtt{b}]\!](s)\Big) + (1 - p_b) \times \mathbf{Pr}\Big([\![C_2]\!](s) \in \Phi \mid [\![\neg \mathtt{b}]\!](s)\Big) && \wr\text{cond. prob.}\wr
\end{aligned}$$

The abstract transformer of statement $\mathtt{S}$ depends heavily on p_b. Unfortunately, it may be the case that the analysis cannot determine the exact value of p_b. There can be two main reasons for that

- *Lack of precision*: the evaluation of the condition may involve variables that we do not have precise enough information about. Moreover, as we do not have always the optimal abstract transfer functions, we are likely to lose precision along the way: the probability that we know for a condition to be true is unfortunately just a minoration (because, for example, the analyzer could show that the condition is met with probability only 0.5 instead of 0.7 by lack of completeness).
- *Measurability*: the condition is not probabilistic, or we do not have the necessary probabilistic setting to determine it (it may have been abstracted away by an Ω-abstraction from Sect. 5.2). Indeed, the previous calculus is valid only if the events $[\![\mathtt{b}]\!](s)$ (b is true) and $[\![\neg\mathtt{b}]\!](s)$ (b is false) are observable. Otherwise, the value of p_b is not even defined.

Whatever the cause of the uncertainty may be, we end up with p_b being unknown in a set $p_b \in P_b \subseteq [0,1]$. At this point, the best is to separately analyze the branches of the conditional and compute $P_1(\Phi) = \mathbf{Pr}\,([\![C_1]\!](s) \in \Phi \mid [\![\mathtt{b}]\!](s))$ and $P_2(\Phi) = \mathbf{Pr}\,([\![C_2]\!](s) \in \Phi \mid [\![\neg\mathtt{b}]\!](s))$. Then the set of possible outcome distributions is $\{l \in \mathcal{L}_A \mid \exists p \in P_b : \forall \Phi \in \wp(\mathcal{A}) : l(\Phi) = pP_1(\Phi) + (1-p)P_2(\Phi)\}$.

In the same spirit, if P_1 and/or P_2 cannot be accurately determined, then their values belong to some subsets of $[0,1]$ that we try to compute as precisely as possible.

This process must then be lifted to sets (and then easily to equivalence classes) to accomodate the abstract domain $\langle \wp(\mathcal{L}_{\mathcal{A}})/_{\ddot{\equiv}_{\leq}}, \ddot{\preceq}\rangle$.

6.2 Loops

As usual, loops are even more difficult to analyze. It combines the issues of evaluating conditional probabilities with the need to evaluate the number and effects of iterating through the loop.

We describe here a few strategies to design abstract transformers for `while` loops. There are many others that could apply to more specific cases, but we will remain as general as possible to give a good overview. We assume we have the statement `S`: "`while` b `do` C".

The General Case. In the favorable case, the probability of entering the loop after $i \geq 0$ iterations is known, and we denote it by $p_{\mathtt{loop}}(i)$. Following the same idea than in the case of the conditional, we have

$$\begin{aligned}
& [\![\mathtt{S}]\!](l_s)(\Phi) = \mathbf{Pr}([\![\mathtt{S}]\!](s) \in \Phi) && \text{(def. distribution)} \\
=\ & \mathbf{Pr}([\![\mathtt{S}]\!](s) \in \Phi \wedge 0 \text{ iteration}) + \mathbf{Pr}([\![\mathtt{S}]\!](s) \in \Phi \wedge \geq 1 \text{ iterations}) && \text{(dichotomy)} \\
=\ & \mathbf{Pr}([\![\mathtt{S}]\!](s) \in \Phi \wedge 0 \text{ iteration}) + \mathbf{Pr}([\![\mathtt{S}]\!](s) \in \Phi \wedge 1 \text{ iterations}) + \mathbf{Pr}([\![\mathtt{S}]\!](s) \in \Phi \wedge \\
& \geq 2 \text{ iterations}) \\
& && \text{(dichotomy)} \\
=\ & \ldots \\
=\ & \sum_{i\geq 0} \mathbf{Pr}([\![\mathtt{S}]\!](s) \in \Phi \wedge i \text{ iterations}) && \text{(converges because positive terms and sum} \leq 1\text{)} \\
=\ & \sum_{i\geq 0} p_{\mathtt{loop}}(i) \times \mathbf{Pr}([\![\mathtt{S}]\!](s) \in \Phi \mid i \text{ iterations}) && \text{(cond. prob.)}
\end{aligned}$$

The nice thing here is that the computation of the iterations is separate for each number of iterations. For $i \geq 0$ iterations, the transformer of the loop is simply the composition of the conditional evaluation and the body execution i times. The second term that accounts for the probability that the loop actually does not terminate cannot be *a priori* eliminated, although it can sometimes be ruled out if the analysis does have more information on the context.

As in the conditional case, the crux of the matter is to obtain as good evaluations as possible for $p_{\texttt{loop}}(i)$ and the body transformer.

Non-Probabilistic Loops. If the truth of the condition b of the loop is not a measurable semantic property, then the analysis cannot determine what is the probability to enter the loop. Thus $p_{\texttt{loop}}(i)$ is only known to be anything in $[0, 1]$, and the analysis has to contain the set of all corresponding possible probabilistic measures.

As usual, custom widening operators may have to be used to guarantee termination depending on the underlying abstract domain.

An Example of an Ad-Hoc Loop Transfer Function. We now present a particular case of a loop transformer that may apply in a variety of cases, as an example to show how to craft specific loop abstract transformers for specific situations.

Suppose that the analyzer knows that the loop always terminates and that $p_{\texttt{loop}}(i)$ decreases as i increases, but that it cannot deduce from the body of the loop how it does so. In that case, the above equation is of no practical use. One way would be to go with the transfer function from 6.2 as it is sound, but it can be quite imprecise.

The approach we choose here is to unroll the loop for $N > 0$ iterations and over-approximate anything that can happen after. Reusing the above calculus, we have

$$\begin{aligned}
&[\![\mathtt{S}]\!](l_s)(\Phi) = \mathbf{Pr}([\![\mathtt{S}]\!](s) \in \Phi) && \text{(def. distribution)}\\
&= \sum_{i\geq 0} p_{\texttt{loop}}(i) \times \mathbf{Pr}([\![\mathtt{S}]\!](s) \in \Phi \mid i \text{ iterations}) && \text{(conditional probability)}\\
&= \sum_{i=0}^{N} p_{\texttt{loop}}(i) \times \mathbf{Pr}([\![\mathtt{S}]\!](s) \in \Phi \mid i \text{ iterations}) + \sum_{i>N} p_{\texttt{loop}}(i) \times \mathbf{Pr}([\![\mathtt{S}]\!](s) \in \Phi \mid i \text{ iterations})
\end{aligned}$$

By hypothesis, for all $i > N$, $p_{\texttt{loop}}(i) \leq p_{\texttt{loop}}(N)$. So we deduce that

$$\sum_{i>N} p_{\texttt{loop}}(i) \times \mathbf{Pr}([\![\mathtt{S}]\!](s) \in \Phi \mid i \text{ iterations}) \leq p_{\texttt{loop}}(N)$$

In that case, the transfer function for the first iterations is thus calculated by simply composing the body transfer function and the conditional N times, we note it l_N. Then to take the second term into account, the set of resulting distributions is $\{l \in \mathcal{L}_{\mathcal{A}} \mid \forall \Phi \in \wp(\mathcal{A}) : |l(\Phi) - l_N(\Phi)| \leq p_{\texttt{loop}}(N)\}$. The soundness is guaranteed by the above calculus.

Note that the transformer could be made more precise because the uncertainty applies only to properties that are impacted by the execution of the body, we do not take that into account in the above definition.

This approach can be made even more precise as N need not be fixed in advance: the loop can be iterated until the probability of going through it again is less than a specified cutoff $\varepsilon > 0$ (so that the source of imprecision $p_{\texttt{loop}}(N)$ is tightly bounded) ; and if it is not witnessed after a specified number of iterations $N_{\max}$, then the above mechanism is used.

7 Related Work: Some Well-Known Techniques as Probabilistic Abstractions

7.1 Markov Chains/Decision Processes

Markov chains are random discrete transitions systems with a finite or countable number of possible states such that the next state depends only on the current state and not on the past or the future. Assuming in Ex. 4 that $S_p^{+\infty}[\![\mathrm{P}]\!]$ is a stationary stochastic process (all executions do terminate) on a countable state space Σ (for simplicity on the non-negative integers), the Markov chain with the transition matrix $[\mathsf{succ}(s, s')]_{s,s' \in \Sigma}$ has the same steady-state behavior, and similar short-term statistics [20, Proposition A.1.1]. In case of non-stationarity (non-termination), alternatives are to add history (considering states in $\Sigma' \triangleq \Sigma^+$) or to define $\mathbf{Pr}_{\langle s,\, s' \rangle} \triangleq \lim_{n \to \infty} \frac{1}{n} \mathbf{Pr}(S_p^{+\infty}[\![\mathrm{P}]\!] \in \{\sigma s s' \sigma' \mid \sigma s \in \Sigma^+ \wedge s' \sigma' \in \Sigma^{+\infty}\})$. So every process is (almost) Markov, which justifies this standard abstraction of probabilistic program semantics [22].

7.2 Probabilistic Model Checking

Probabilistic model checking [11] is often based on the Markov chain abstraction of Sect. 7.1. The fundamental notion of *probabilistic reachability* for Markov decision processes can be generalized to programs by considering the abstraction $\alpha(X) \triangleq \lambda s \bullet \mathbf{Pr}(S_p^{+\infty}[\![\mathrm{P}]\!] 1_{\{s\}} \cap \gamma_r(X) \neq \emptyset)$ of the maximal trace semantics similar to Ex. 17. It is further abstracted by the *probability interval abstraction* $\alpha_m(X) \triangleq \min\{\alpha(X)s \mid s \in \Sigma\}$ and $\alpha_M(X) \triangleq \max\{\alpha(X)s \mid s \in \Sigma\}$ which is computable for finite systems [3, Sect. 6], [10, Sect. 3], [11, Sect. 4], or their reduced product [29, Sect. 3], etc. However programs generally have an unbounded concrete semantics so a (traditional) finite abstraction is often too imprecise [8]. This is the main reason for considering infinitary abstractions in this paper.

7.3 Quantitative Abstraction

[24, 26] propose a formulation of abstract interpretation on Hilbert spaces for real or complex quantitative abstractions of distribution-based semantics which can be reformulated using abstraction (2) of traces (e.g. where states are sets of λ-terms and transitions are reductions of these λ-terms). However, they do not stick to the usual soundness notion [26, Sect.5.2]: they are interested in behaviors on expectations and the "strict" soundness that we enforced from the beginning has to be relaxed using more permissive concretization functions.

7.4 Probabilistic Strongest Postcondition Semantics

Following Ex. 4, we let $\langle \Omega, \mathcal{E}, \mu\rangle$ be a probability space. The probabilistic semantics postulated in [14] is a distribution transformer abstracting the probabilistic maximal trace semantics $S_p^{+\infty}[\![\mathtt{P}]\!] : \Omega \rightarrowtail \Sigma^{+\infty}$.

Given a distribution $\delta \in \mathcal{L}_\Sigma$ of the initial states, the abstraction $\alpha_s : (\Omega \rightarrowtail \wp(\Sigma^{+\infty})) \longrightarrow (\mathcal{L}_\Sigma \longrightarrow \mathcal{L}_\Sigma)$ of $X \in \wp(\Sigma^{+\infty})$ is the distribution of the final states, if any, so that

$$\alpha_s(\lambda\,\omega \bullet X(\omega))\delta s' \triangleq \sum_{s\in\Sigma} \delta(s) \times \mathbf{Pr}(\exists\sigma : s\sigma s' \in X^+) \tag{2}$$

The abstract semantics is $S_s[\![\mathtt{P}]\!] \triangleq \alpha_s(S_p^{+\infty}[\![\mathtt{P}]\!])$. For example

$$\begin{aligned}
&- \ \alpha_s(S_p^{+\infty}[\![\mathtt{skip}]\!])\delta s' = \alpha_s(\{ss \mid s \in \Sigma\})\delta s' && \langle\text{def. } S_p^{+\infty}[\![\mathtt{skip}]\!]\rangle\\
&= \sum_{s\in\Sigma} \delta(s) \times \mathbf{Pr}(s = s') = \delta(s') && \langle\text{def. } \alpha_s \text{ so that } S_s[\![\mathtt{skip}]\!]\delta = \delta\rangle
\end{aligned}$$

$$\begin{aligned}
&- \ \alpha_s(S_p^{+\infty}[\![\mathtt{if}\ c\ \mathtt{then}\ A\ \mathtt{else}\ B]\!])\delta s'\\
&= \sum_{s\in\Sigma} \delta(s) \times \mathbf{Pr}(\exists\sigma : s\sigma s' \in \{s\sigma' \mid \mathcal{E}[\![c]\!]s \wedge s\sigma' \in S_p^{+\infty}[\![A]\!]^+\} \cup \{s\sigma' \mid \mathcal{E}[\![\neg c]\!]s \wedge s\sigma' \in\\
&\quad S_p^{+\infty}[\![B]\!]^+\})\\
&\hspace{6cm} \langle\text{def. } \alpha_s \text{ and } S_p^{+\infty}[\![\mathtt{if}\ c\ \mathtt{then}\ A\ \mathtt{else}\ B]\!]\rangle\\
&= \sum_{s\in\Sigma} \delta(s) \times (\mathbf{Pr}(\mathcal{E}[\![c]\!]s) \times \mathbf{Pr}(\exists\sigma : s\sigma s' \in S_p^{+\infty}[\![A]\!]^+) + (1 - \mathbf{Pr}(\mathcal{E}[\![c]\!]s)) \times \mathbf{Pr}(\exists\sigma : s\sigma s' \in\\
&\quad S_p^{+\infty}[\![B]\!]^+\})) && \langle\text{probability law}\rangle\\
&= \sum_{s\in\Sigma} \delta(s) \times (c \times \mathbf{Pr}(\exists\sigma : s\sigma s' \in S_p^{+\infty}[\![A]\!]^+) + (1 - c) \times \mathbf{Pr}(\exists\sigma : s\sigma s' \in S_p^{+\infty}[\![B]\!]^+))\\
&\hspace{4cm} \langle\text{by [14] implicitly assuming that } \mathbf{Pr}(\mathcal{E}[\![c]\!]s) = c \text{ where } c \in \mathbb{R}^*\rangle\\
&= c \times \alpha_s(S_p^{+\infty}[\![A]\!])\delta s' + (1 - c) \times \alpha_s(S_p^{+\infty}[\![B]\!])\delta s' && \langle\text{def. } \alpha_s\rangle
\end{aligned}$$

proving that $S_s[\![\mathtt{if}\ c\ \mathtt{then}\ A\ \mathtt{else}\ B]\!] = c \times S_s[\![A]\!] + (1 - c) \times S_s[\![B]\!]$ pointwise and similarly for other commands using fixpoint abstraction [6, Th. 7.1.0.4-(3)] for loops.

These theorems are, up to logical notations, the axioms postulated in [14]. The probabilistic strongest postcondition abstraction in equation (2) is frequently used as collecting semantics for forward static analysis e.g. [21] for Markov decision processes.

7.5 Probabilistic Weakest Precondition Semantics

Whereas [14] is a forward abstraction as explained in Sect. 7.4, [15, 23] is the corresponding backward abstraction providing probabilistic weakest preconditions

$$\alpha_w(X)\delta s \triangleq \sum_{s'\in\Sigma} \mathbf{Pr}(\exists\sigma : s\sigma s' \in X^+) \times \delta(s') \tag{3}$$

The abstract semantics is $S_w[\![\mathtt{P}]\!] \triangleq \alpha_w(S_p^{+\infty}[\![\mathtt{P}]\!])$. For example

$$\begin{aligned}
&S_w[\![C_1\,;C_2]\!]\delta = \alpha_w(S_p^{+\infty}[\![C_1\,;C_2]\!])\delta && \langle\text{def. } S_w[\![\mathtt{P}]\!]\rangle\\
&= \lambda\,s \bullet \sum_{s'\in\Sigma} \mathbf{Pr}(\exists\sigma : s\sigma s' \in S_p^{+\infty}[\![C_1]\!]^+ \mathbin{⨾} S_p^{+\infty}[\![C_2]\!]^+) \times \delta(s') && \langle\text{def. } \alpha_w \text{ and } S_p^{+\infty}[\![C_1\,;C_2]\!]\rangle
\end{aligned}$$

$$= \lambda s \bullet \sum_{s' \in \Sigma} \mathbf{Pr}(\exists \sigma', s'', \sigma'' : s\sigma' s'' \in S_p^{+\infty}[\![C_1]\!]^+ \wedge s''\sigma s' \in S_p^{+\infty}[\![C_2]\!]^+) \times \delta(s')$$
$$\text{\textlbrackdbl def. } ; \text{ with } s\sigma s' = s\sigma' s''\sigma s' \text{\textrbrackdbl}$$

$$= \lambda s \bullet \sum_{s' \in \Sigma} \mathbf{Pr}(\exists \sigma : s\sigma s' \in S_p^{+\infty}[\![C_1]\!]^+) \times \left(\sum_{s'' \in \Sigma} \mathbf{Pr}(\exists \sigma'' : s'\sigma'' s'' \in S_p^{+\infty}[\![C_2]\!]^+) \times \delta(s'') \right)$$
$$\text{(conditional probabability)}$$

$$= \lambda s \bullet \alpha_w(S_p^{+\infty}[\![C_1]\!])(\lambda s' \bullet \sum_{s'' \in \Sigma} \mathbf{Pr}(\exists \sigma'' : s'\sigma'' s'' \in S_p^{+\infty}[\![C_2]\!]^+) \times \delta(s''))(s) \quad \text{(def. } \alpha_w\text{)}$$

$$= \lambda s \bullet \alpha_w(S_p^{+\infty}[\![C_1]\!])(\alpha_w(S_p^{+\infty}[\![C_2]\!])(\delta))(s) \quad \text{(def. } \alpha_w\text{)}$$

$$= \lambda s \bullet S_w[\![C_1]\!](S_w[\![C_2]\!](\delta))(s) \quad \text{(def. } S_w[\![C]\!]\text{)}$$

$$= S_w[\![C_1]\!] \circ S_w[\![C_2]\!](\delta) \quad \text{(def. } \circ\text{)}$$

which is the definition of $S_w[\![C_1\,;C_2]\!]$ postulated in [15, Sect. 4]. The probabilistic choice $C_1\ {}_p\!\oplus C_2$ requires additional hypotheses as in Sect. 7.1 while iteration is handled by fixpoint abstraction [6, Th. 7.1.0.4-(3)]. The probabilistic weakest precondition abstraction (3), or at least its discrete equivalent, is frequently used as collecting semantics for backward static analysis e.g. [22, 29] for Markov decision processes and further abstracted by the probabilistic intervals of Sect. 7.2.

8 Future Work and Conclusion

We have introduced new principles of probabilistic abstract interpretation for designing probabilistic semantics and static analysis methods. The framework is very general, highly expressive so as to set forth any probabilistic and computational situation. The framework separates probabilities (μ) from semantics ($S_p[\![\mathrm{P}]\!]$) so the probabilistic and semantics abstractions are self-reliant. Their abstractions can each be fine-tuned independently by easy adaptation of standard proof and static analysis methods.

Future work includes the case of absence of a best abstraction, the study of relational law-abstractions, improvement of branch prediction, implementation and experiments. It will also be essential to develop precise widening operators and abstract transformers to keep enough precision during the fixpoint calculation.

Work supported in part by the CMACS NSF Expeditions in Computing award 0926166.

References

[1] Camporesi, F., Feret, J., Koeppl, H., Petrov, T.: Automatic reduction of stochastic rules-based models in a nutshell. Amer. Inst. of Physics, AIP 1281(2) (2010)

[2] Chadha, R., Viswanathan, M., Viswanathan, R.: Least Upper Bounds for Probability Measures and Their Applications to Abstractions. In: van Breugel, F., Chechik, M. (eds.) CONCUR 2008. LNCS, vol. 5201, pp. 264–278. Springer, Heidelberg (2008)

[3] Coletta, A., Gori, R., Levi, F.: Approximating probabilistic behaviors of biological systems using abstract interpretation 229(1), 165–182 (2009)

[4] Cousot, P.: Constructive design of a hierarchy of semantics of a transition system by abstract interpretation. TCS 277(1-2), 47–103 (2002)

[5] Cousot, P., Cousot, R.: Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: POPL, pp. 238–252 (1977)

[6] Cousot, P., Cousot, R.: Systematic design of program analysis frameworks. In: POPL, pp. 269–282 (1979)

[7] Cousot, P., Cousot, R.: Abstract interpretation frameworks. J. Logic and Comp. 2(4), 511–547 (1992)

[8] Cousot, P., Cousot, R.: Comparing the Galois Connection and Widening/Narrowing Approaches to Abstract Interpretation. In: Bruynooghe, M., Wirsing, M. (eds.) PLILP 1992. LNCS, vol. 631, pp. 269–295. Springer, Heidelberg (1992)

[9] Cousot, P., Cousot, R., Feret, J., Mauborgne, L., Miné, A., Rival, X.: Why does Astrée scale up? FMSD 35(3), 229–264 (2009)

[10] D'Argenio, P.R., Jeannet, B., Jensen, H.E., Larsen, K.G.: Reduction and Refinement Strategies for Probabilistic Analysis. In: Hermanns, H., Segala, R. (eds.) PAPM-PROBMIV 2002. LNCS, vol. 2399, pp. 57–76. Springer, Heidelberg (2002)

[11] Forejt, V., Kwiatkowska, M., Norman, G., Parker, D.: Automated Verification Techniques for Probabilistic Systems. In: Bernardo, M., Issarny, V. (eds.) SFM 2011. LNCS, vol. 6659, pp. 53–113. Springer, Heidelberg (2011)

[12] Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. FAC 6(5), 512–535 (1994)

[13] Hehner, E.: Probabilistic Predicative Programming. In: Kozen, D. (ed.) MPC 2004. LNCS, vol. 3125, pp. 169–185. Springer, Heidelberg (2004)

[14] Hehner, E.: A probability perspective. FAC 23(4), 391–419 (2011)

[15] Katoen, J.-P., McIver, A.K., Meinicke, L.A., Morgan, C.C.: Linear-Invariant Generation for Probabilistic Programs: Automated Support for Proof-Based Methods. In: Cousot, R., Martel, M. (eds.) SAS 2010. LNCS, vol. 6337, pp. 390–406. Springer, Heidelberg (2010)

[16] Klenke, A.: Probability Theory: A Comprehensive Course. Springer, Heidelberg (2007)

[17] Kozen, D.: Semantics of probabilistic programs. JCSS 22, 328–350 (1981)

[18] Kwiatkowska, M., Norman, G., Parker, D.: Using probabilistic model checking in systems biology. PER 35(4), 14–21 (2008)

[19] McIver, A., Morgan, C.: Abstraction, Refinement and Proof for Probabilistic Systems. Springer, Heidelberg (2005)

[20] Meyn, S.: Control Techniques for Complex Networks. CUP (2007)

[21] Monniaux, D.: Abstract Interpretation of Probabilistic Semantics. In: SAS 2000. LNCS, vol. 1824, pp. 322–340. Springer, Heidelberg (2000)

[22] Monniaux, D.: Abstract interpretation of programs as Markov decision processes. SCP 58(1–2), 179–205 (2005)

[23] Morgan, C., McIver, A., Seidel, K., Sanders, J.: Probabilistic predicate transformers. TOPLAS 18(3), 325–353 (1996)

[24] Di Pierro, A., Hankin, C., Wiklicky, H.: Probabilistic lambda-calculus and quantitative program analysis. JLC 15(2), 159–179 (2005)

[25] Di Pierro, A., Wiklicky, H.: Concurrent constraint programming: towards probabilistic abstract interpretation. In: PPDP, pp. 127–138. ACM (2000)

[26] Di Pierro, A., Wiklicky, H.: Probabilistic Abstract Interpretation and Statistical Testing (Extended Abstract). In: Hermanns, H., Segala, R. (eds.) PAPM-PROBMIV 2002. LNCS, vol. 2399, pp. 211–212. Springer, Heidelberg (2002)

[27] Roy, P., Parker, D., Norman, G., de Alfaro, L.: Symbolic magnifying lens abstraction in Markov decision processes. In: QEST 2008, pp. 103–112. IEEE (2008)

[28] Smith, M.: Probabilistic abstract interpretation of imperative programs using truncated normal distributions 220(3), 43–59 (2008)

[29] Wachter, B., Zhang, L.: Best Probabilistic Transformers. In: Barthe, G., Hermenegildo, M. (eds.) VMCAI 2010. LNCS, vol. 5944, pp. 362–379. Springer, Heidelberg (2010)