Lilac: a Modal Separation Logic for Conditional Probability

ANONYMOUS

We present Lilac, a separation logic for reasoning about probabilistic programs where separating conjunction captures statistical independence. Inspired by an analogy with mutable state where sampling corresponds to dynamic allocation, we show how probability spaces over a fixed, ambient sample space appear to be the natural analogue of heap fragments, and present a new combining operation on them such that probability spaces behave like heaps and measurability of random variables behaves like ownership. This combining operation forms the basis for our model of separation, and produces a logic with many pleasant properties. In particular, Lilac has a frame rule identical to the ordinary one, and naturally accommodates advanced features like continuous random variables and reasoning about quantitative properties of programs. Then we propose a new modality based on disintegration theory for reasoning about conditional probability. We show how the resulting modal logic can be used to provide a formal verification of a weighted sampling algorithm.

ACM Reference Format:
Anonymous. 2023. Lilac: a Modal Separation Logic for Conditional Probability. 1, 1 (January 2023), 40 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Software systems involving probability are pervasive. Such systems naturally appear in diverse domains such as network reliability analysis [16, 38], reliability for cyberphysical systems [21], distributed software systems [44], and many others. These systems are increasingly deployed in high-consequence domains; therefore, there is a growing need for formal frameworks capable of reasoning about and verifying probabilistic correctness properties. The most prevalent strategy is to describe the behavior of such systems using a *probabilistic programming language* (PPL) – a programming language whose semantics is given by a probability distribution. Then the probabilistic verification task is to certify that the program satisfies certain safety properties: for instance, that the probability of a bad event occurring is exactly equal to (or upper-bounded by) a desired value [1, 8, 11, 21, 30], that the program terminates with high probability [26, 31], or that the program satisfies certain statistical independence relationships [3, 5].

Program logics are a powerful approach to verifying probabilistic programs [2–5, 36, 44]. In contrast to other methods [11, 22, 30], program logics avoid reasoning explicitly about global execution traces, and instead reason about programs in a compositional manner via assertions capturing local invariants. To accomplish this, program logics must present an expressive and natural axiomatic system for describing probabilistic behavior that is powerful enough to state useful invariants about programs while being simple enough to be usable in practice. The ideal program logic enables "simple proofs for simple programs" [33], which is easier said than done: there is a large and quickly-growing landscape of avenues for designing program logics for PPLs. We identify the following key attributes of an effective general-purpose program logic for PPLs:

Modularity The fundamental decomposition principle for probabilistic programs is *statistical independence*. It should be possible to decompose a probabilistic reasoning task in the logic by exploiting both ordinary and conditional independence.

Support for quantitative reasoning Many probabilistic safety properties are quantitative. For example, they may require proving a program computes the correct value on average, or

Author's address: Anonymous.

2023. XXXX-XXXX/2023/1-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

upper-bounding the probability of deviating significantly from expectation. The logic should be capable of stating and proving facts of this nature.

Compatibility with probability theory Many concise proofs of correctness for randomized algorithms hinge on the ability to import important results from probability theory, such as the law of total expectation, the relationship between expectation and independence, and the expectations of random variables with well-understood distributions. More generally, probabilistic reasoning frequently requires the application of well-known mathematical identities. Ideally, reasoning in the logic should interoperate well with standard mathematical objects, making it easy to integrate such well-known results as derived rules in the logic, and therefore accessible to someone who is familiar with probability theory.

Expressivity The logic should support reasoning about a rich family of programming constructs used in practical PPLs, including loops and continuous random variables.

In this paper we present Lilac, a separation logic for probabilistic programs with the goal of achieving all of the above design criteria. Our core contribution is a novel combining operation on probability spaces, analogous to disjoint union of heap fragments in ordinary separation logic. Based on this combining operation, Lilac interprets separating conjunction as independence of *probability spaces*, rather than the existing approach of representing independence of *random variables* [3, 5]. Our new notion of separation enjoys a close correspondence to ordinary probability – we prove Lilac's separating conjunction precisely captures statistical independence (Lemma 3.5) – and the directness of this correspondence makes it easy to add support for quantitative reasoning and continuous random variables. This makes Lilac the first probabilistic program logic with support for such features in combination with a substructural treatment of statistical independence: existing program logics either treat independence substructurally but have limited support for quantitative reasoning and no support for continuous random variables [2, 5], or support very general kinds of quantitative reasoning but encode (mutual) independence explicitly as a derived notion, leading to a logic that places increased burden on the prover to maintain and establish independence [4].

Lilac's second core contribution is a *disintegration modality*: a *modal* interpretation of conditional probability. Historically, conditional independence has been very difficult to capture in a substructural program logic: it has either gone unsupported [4, 5] or required a host of new logical connectives and significant changes to the underlying semantic model [2]. Lilac captures conditioning via the addition of a single modal operator. Its interpretation, based on *disintegration* [9, 37], doesn't require any changes to our underlying semantic model beyond restricting ourselves to a class of suitably-well-behaved probability spaces. We argue that the disintegration modality captures the flavor of informal reasoning about conditioning. To demonstrate this, we show how it can be used in conjunction with all of Lilac's other features to formally verify a constant-space weighted sampling algorithm in a manner resembling an informal textbook proof.

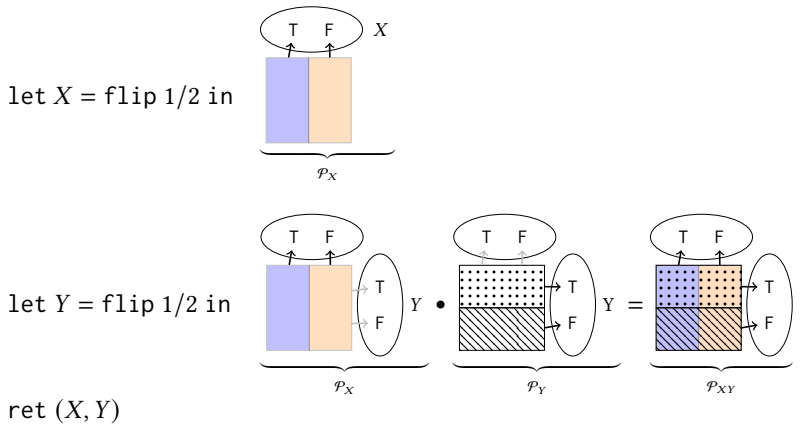
2 MOTIVATION AND OVERVIEW

The design of Lilac is fundamentally motivated by analogy with mutable state: sampling is like allocation, probability spaces are like heap fragments, and random variables are like heap locations. We will illustrate this with a simple example, and show how it forms the basis for our model of separation logic. Then, with this description of our model in mind, we will sketch Lilac's language of assertions alongside informal descriptions of the intended meanings of these assertions. This semantics will be made precise in sections 3 and 4. To conclude this section, we use Lilac to verify a weighted-sampling algorithm, highlighting its capability for reasoning about continuous random variables and conditional probability.

To illustrate the analogy between probability and mutable state, consider the following program:

$$\begin{array}{l}\texttt{let } X = \texttt{flip } 1/2 \texttt{ in}\\ \texttt{let } Y = \texttt{flip } 1/2 \texttt{ in}\\ \texttt{ret } (X, Y)\end{array} \qquad \text{(FLIP2)}$$

This program uses `flip` $1/2$ to sample the result of two fair coin flips (i.e., uniformly distributed values in $\{\mathsf{T}, \mathsf{F}\}$). The idea is to think of this program as carrying along a *probability space* as it executes, analogous to how programs with mutable state carry along a heap. In probability theory, a probability space $\mathcal{P}$ is a tuple $(\Omega, \mathcal{F}, \mu)$. The set Ω is called the sample space and $\mathcal{F}$ is a collection of subsets of Ω called a *σ-algebra*, which is a collection of sets satisfying (1) closure under complements, countable union, and countable intersection; and (2) $\Omega \in \mathcal{F}$ and $\emptyset \in \mathcal{F}$. Elements of $\mathcal{F}$ are called *events*, and $\mu : \mathcal{F} \to [0, 1]$ is a *probability measure* assigning each event its probability. For this example we fix Ω to be the unit square $[0, 1] \times [0, 1]$. This allows us to visualize σ-algebras $\mathcal{F}$ as partitionings of the unit square into a finite number of regions – each region denotes an event – and μ as the map that sends each event to its area.[1] We can visualize execution of FLIP2 as:



Initially, $\mathcal{F}$ only contains the trivial events $\emptyset$ and Ω, analogous to how programs in heap-manipulating languages start with an empty heap. After the first line is executed two things change:

- Two new events are allocated, each covering half the sample space. These are the blue partition on the left and the orange partition on the right; they form the probability space labeled $\mathcal{P}_X$. This is like how `new` allocates a fresh memory cell on the heap: probability spaces correspond to heap fragments.
- The `flip` operation yields a random variable $X : \Omega \to \{\mathsf{T}, \mathsf{F}\}$ that maps blue points in Ω to T and orange points in Ω to F; this is depicted by the arrows.[2] Concretely, $X(\omega_1, \omega_2) = \mathsf{T}$ if $\omega_1 < 1/2$ and F otherwise. This is like how `new` returns the location of the newly allocated heap cell: random variables correspond to locations.

The next line allocates a second probability space $\mathcal{P}_Y$, visualized by the dotted region and dashed region, and a new random variable Y that associates dotted points to T and dashed points to F; concretely, $Y(\omega_1, \omega_2) = \mathsf{T}$ if $\omega_2 > 1/2$ and F otherwise. In heap-manipulating languages, `new` generates a *fresh* heap cell, so that the entire heap after executing `new` is a *disjoint union* of the old heap and the newly allocated cell. Analogously, `flip` allocates a probability space *statistically independent* from the old one, so that the entire space after executing the second `flip` is an

[1] In later technical development we will consider richer families of sample spaces that are more convenient [10].
[2] A *random variable*, typically denoted with a capital letter, is a map out of the sample space Ω.

independent combination of the old space $\mathcal{P}_X$ and the newly allocated one $\mathcal{P}_Y$. This independent combination, written $(\bullet)$, is our new combining operation on probability spaces: it has the same algebraic properties as disjoint union of heap fragments does in ordinary separation logic (Theorem 3.4), and is the heart of Lilac's notion of separation. The space labeled $\mathcal{P}_{XY}$ visualizes the independent combination $\mathcal{P}_X \bullet \mathcal{P}_Y$. The events of $\mathcal{P}_{XY}$ (i.e., the four partitions in the figure) form the σ-algebra generated by the events of $\mathcal{P}_X$ and $\mathcal{P}_Y$; we give a formal definition of this combining operation in Section 3. The insight that disjoint union of heaps corresponds to independent combinations of probability spaces underlies Lilac's interpretation of standard separation logic connectives: in particular, $\mathcal{P} \models P_1 * P_2$ if there exists a "splitting" of $\mathcal{P}$ into an independent combination $\mathcal{P}_1 \bullet \mathcal{P}_2$ such that $\mathcal{P}_1 \models P_1$ and $\mathcal{P}_2 \models P_2$.

In addition to substructural connectives like separating conjunction, ordinary separation logics include atomic propositions for expressing ownership of individual heap fragments; these have analogues in Lilac as well. In ordinary separation logic, the assertion "$\ell \mapsto -$" asserts ownership of the location ℓ but no knowledge of its contents, and $(\ell_1 \mapsto -) * (\ell_2 \mapsto -)$ asserts that ℓ_1 and ℓ_2 are in disjoint heap fragments. By the analogy of locations with random variables, the Lilac proposition "own X" expresses *probabilistic ownership* of X but no knowledge of its distribution, and $(\mathsf{own}\, X) * (\mathsf{own}\, Y)$ asserts the random variables X and Y are statistically independent. Intuitively, the proposition $\mathsf{own}\, X$ holds in probability space $(\Omega, \mathcal{F}, \mu)$ if $\mathcal{F}$ contains every event of the form $X = x$; this is like how, in ordinary separation logic, $h \models \ell \mapsto -$ if the heap h contains location ℓ. The requirement that $\mathcal{F}$ contain every event of the form $X = x$ is well-known to probability theorists: it is exactly *$\mathcal{F}$-measurability* of the random variable X.[3] In the diagram above, X owns $\mathcal{P}_X$ because the event $X = \top$ is the blue rectangle which is in the σ-algebra for $\mathcal{P}_X$. It is useful to clarify what it means for a random variable to *not* be $\mathcal{F}$-measurable (and therefore fail to have ownership of a given probability space): this occurs when there exists some x such that the event $X = x$ is not contained in $\mathcal{F}$. This can be seen in the diagrammatic depictions of $\mathcal{P}_X$ and $\mathcal{P}_Y$ above, where in $\mathcal{P}_X$ the random variable Y is not $\mathcal{P}_X$-measurable, denoted with grayed out arrows; similar for the random variable X in $\mathcal{P}_Y$.

Thus our analogy with mutable state yields two key insights that form the basis for our model of separation logic: (1) disjoint union of heaps corresponds to independent combination of probability spaces, and (2) ownership is measurability. This model closely mirrors the standard mathematical foundations of probability theory, making it natural to capture commonly used probability theory constructs. For instance, Lilac easily accommodates the assertion $X \sim \mathrm{Ber}\, 1/2$ which states that X follows distribution Bernoulli $1/2$; with this assertion in hand, the probability spaces depicted in the two-coin-flip example above can be expressed by the following Lilac-annotated program:

$$\begin{array}{l} \quad \texttt{let}\ X = \texttt{flip}\ 1/2\ \texttt{in} \\ \{X \sim \mathrm{Ber}\, 1/2\} \\ \quad \texttt{let}\ Y = \texttt{flip}\ 1/2\ \texttt{in} \\ \{X \sim \mathrm{Ber}\, 1/2 \ * \ Y \sim \mathrm{Ber}\, 1/2\} \\ \quad \texttt{ret}\ (X, Y) \end{array} \tag{1}$$

2.1 Proving a Weighted Sampling Algorithm Correct

Thus far our development of Lilac has closely mirrored and been inspired by ordinary separation logics. However, arguments for correctness of probabilistic programs often feature a mode of reasoning with no direct analogue in ordinary separation logics: *conditioning*. It is common in many probabilistic systems for a property to only hold *conditional* on some random variable: for

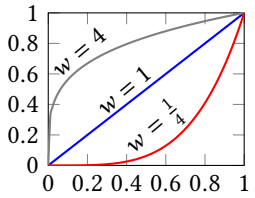
[3] To simplify the presentation we are considering measurability of discrete random variables here; our interpretation of $\mathsf{own}\, X$ generalizes to the continuous case by using the standard definition of $\mathcal{F}$-measurability [24].

```
1 W ← w_1 + ··· + w_n
2 (v_1, ..., v_n) ← (w_1/W, ..., w_n/W)
3 U ~ uniform [0,1]
4 for i in 1...n
5   if v_1 + ··· + v_{i-1} ≤ U < v_1 + ··· + v_i
6     return i
```

(a) Naive linear-space implementation of weighted sampling.

```
1 (M, K) ← (−∞, 0)
2 for i in 1,...,n:
3   S ~ uniform [0,1]
4   U ← S^{1/w_i}
5   if U > M
6     (M, K) ← (U, i)
7 return K
```

(b) Constant-space version.



(c) Visualization of the function $f(x) = x^{1/w}$.

Fig. 1. Motivating weighted sampling example. The constants w_i are inputs.

instance, two random variables might only be independent conditional on a third, a property called *conditional independence*. Conditional independence is a powerful and prevalent source of modularity in PPLs: correctness arguments for probabilistic programs often hinge upon a clever choice of what to condition on, exploiting key conditional independence relationships to complete the proof. To capture this mode of reasoning, Lilac features a modal operator "D"; the proposition "$\mathsf{D}_{x \leftarrow X}\, P$" asserts P holds *conditional* on all possible values x the random variable X can take on. Standard separation logic assertions have intuitive readings under D: while own $X * $ own Y expresses independence of X and Y, $\mathsf{D}_{z \leftarrow Z}$ own $X *$ own Y expresses conditional independence of X and Y given Z; while $\mathbb{E}[X] = v$ is an assertion about the expectation of X, $\mathsf{D}_{y \leftarrow Y}\ \mathbb{E}[X] = v$ is an assertion about the conditional expectation of X given Y.

To highlight the usefulness of this new modality, we will prove a sophisticated constant-space *weighted sampling algorithm* correct using Lilac. Suppose you are given a collection of items $\{x_1, \ldots, x_n\}$ each with associated weight $w_i \in \mathbb{R}^+$. The task is to draw a sample from the collection $\{x_i\}$ in a manner where each item is drawn with probability proportional to its weight. This problem is an example of *reservoir sampling* [12], and is an important primitive in distributed systems.

First, we consider a simple *two-pass* solution to implementing weighted sampling, which requires space linear in the number of weights; pseudocode for this algorithm is presented in Figure 1a. The first pass over the weights occurs on Line 1, which computes the normalizing constant $W = \sum_i w_i$. Line 2 then divides each weight by W so that the result $(v_1, \ldots, v_n)$ forms a probability distribution. This distribution can be thought of as a partitioning of the interval $[0, 1]$ into n subintervals with lengths $(v_1, \ldots, v_n)$; to sample from it we can choose a point U uniformly at random from $[0, 1]$ (Line 3) and select the item corresponding to the subinterval that U lands in (Lines 4–6).

While simple to understand and implement, this naive approach has a critical flaw that makes it inappropriate for application in large-scale systems: it requires storing all previously encountered weights and scanning over them before a single sample can be drawn, and so does not scale to a streaming setting where new weights are acquired one at a time (for instance, as each user visits a website). To fix this limitation, Efraimidis and Spirakis [12] proposed the very clever *constant-space solution* presented in Figure 1b. The core of this approach is to generate a value S uniformly at random from $[0, 1]$ on *every* iteration (Line 3), perturb S according to the next weight w_i in the stream (Line 4), and track only the *greatest* perturbed sample (Lines 5–6). Figure 1c gives some intuition for the perturbed quantity S^{1/w_i} on Line 4: if w_i is large (i.e., item i has high weight), then S^{1/w_i} is likely to be large (visualized by the curve $w = 4$); if w_i is small, then S^{1/w_i} is likely to be small (visualized by the curve $w = 1/4$). The fact that this program is equivalent to the naive one is quite surprising, and proving it requires the simultaneous application of several important theorems from probability theory. We show how this can be done formally in Lilac in a manner similar to a

typical pen-and-paper proof. Correctness is captured by the following Lilac postcondition:

$$\forall i.\, \Pr(K = i) = \frac{w_i}{\sum_j w_j} \tag{2}$$

To establish this postcondition, a typical pen-and-paper proof begins by declaring mutually independent, uniformly distributed random variables $\{S_i\}_{1 \le i \le n}$, where S_i denotes the value sampled by Line 3 on the ith loop iteration, and a random variable $K = \arg\max_i S_i^{1/w_i}$ denotes the final result. Implicit in this setup are the assumptions that each S_i produced by the program is actually independent and uniformly distributed, and that the for-loop actually computes the specified arg max. We can formally establish this by mechanically applying Lilac's proof rules (details in Section 3.5) to conclude the following at program termination:

$$\exists_{\mathrm{rv}} S_1 \ldots S_n.\ \mathop{\ast}_i S_i \sim \mathrm{Unif}\ [0,1]\ *\ K = \arg\max_i S_i^{1/w_i} \tag{3}$$

The proof makes use of the following invariant I_j for the loop on Line 2, which must hold immediately before the execution of the jth iteration for all $1 \le j \le n+1$:

$$I_j \quad := \quad \exists_{\mathrm{rv}} S_1 \ldots S_j.\ \mathop{\ast}_{1 \le i < j} S_i \sim \mathrm{Unif}\ [0,1]\ *\ K = \arg\max_{1 \le i < j} S_i^{1/w_i}\ *\ M = \max_{1 \le i < j} S_i^{1/w_i} \tag{4}$$

The proof that our program maintains this invariant is completely standard for separation logics, so we elide the details and focus on the challenge of deriving the desired post-condition (2) given the setup (3).[4] A traditional proof might continue as follows: to show (2) in the case $i = k$, note that

$$\Pr(K = k) = \Pr\left(S_k^{1/w_k} > S_j^{1/w_j} \text{ for all } j \ne k\right), \tag{5}$$

since K is defined to be the arg max of j over all S_j^{1/w_j}. This is an unwieldy probability to compute. However, there is a classic trick in probability theory to make progress on such problems: we can identify a key quantity to *condition on* that makes the computation tractable. In this case progress is made by conditioning on $S_k = s_k$, fixing the random variable S_k to a deterministic value s_k. This significantly simplifies the computation: for any s_k,

$$\begin{aligned}
\Pr(K = k \mid S_k = s_k) &= \Pr\left(s_k^{1/w_k} > S_j^{1/w_j} \text{ for all } j \ne k\right) && (6)\\
&= \Pr\left(s_k^{w_j/w_k} > S_j \text{ for all } j \ne k\right) && \text{Exponentiating} \quad (7)\\
&= \prod_{j \ne k} \Pr\left(s_k^{w_j/w_k} > S_j\right) && \text{By conditional independence} \quad (8)\\
&= \prod_{j \ne k} s_k^{w_j/w_k} && S_j \text{ uniform}[5] \quad (9)\\
&= \mathrm{pow}\left(s_k, \frac{\sum_{j \ne k} w_j}{w_k}\right). && (10)
\end{aligned}$$

This whole calculation can be expressed in Lilac as a deduction inside the modality $\mathsf{D}_{s_k \leftarrow S_k}$. A critical step occurs in Equation 8: each S_j is conditionally independent from all others given $S_k = s_k$. This permits a critical simplification: the probability of the conjunction becomes a product of simpler probabilities. In Lilac, this reasoning step is expressed by the following entailment:

$$\mathop{\ast}_i \mathsf{own}\, E_i \ \vdash\ \Pr\left(\bigcap_i E_i\right) = \prod_i \Pr(E_i). \tag{Indep-Prod}$$

[4] See appendix C for details.
[5] If U uniform in $[0,1]$ then $\Pr(u > U) = u$.

It may appear that (Indep-Prod) is a statement only about unconditional independence and unconditional probability, whereas Equation (8) is about conditional probability and makes use of independence conditional on $S_k = s_k$. However, the modality $\mathsf{D}_{s_k \leftarrow S_k}$ respects entailment, so (Indep-Prod) continues to hold under it; as such, (Indep-Prod) also expresses a statement about conditional independence. This highlights one of the most important benefits of treating conditioning as a modality: laws that hold in general continue to hold under D, and have the expected interpretations in terms of conditional probability.

Conditioning on $S_k = s_k$ has made progress towards a closed-form answer, but we are not yet done. To complete the proof we need to connect the conditional $\Pr(K = k \mid S_k = s_k)$ to the unconditional $\Pr(K = k)$. This is typically accomplished by using the *law of total expectation*, which relates an unconditional probability to an expectation over conditional probabilities:[6]

$$\Pr(K = k) = \mathbb{E}\left[\Pr(K = k \mid S_k)\right] \qquad \text{Law of Total Expectation} \tag{11}$$

$$= \mathbb{E}\left[\mathrm{pow}\left(S_k, \frac{\sum_{j \neq k} w_j}{w_k}\right)\right] \qquad \text{By (10)} \tag{12}$$

$$= \left(\frac{\sum_{j \neq k} w_j}{w_k} + 1\right)^{-1} \qquad S_k \text{ uniform}[7] \tag{13}$$

$$= \frac{w_k}{\sum_j w_j}. \tag{14}$$

This calculation is also straightforwardly expressible in Lilac. Unlike the calculation in Equations 6–10, which take place inside the modality $\mathsf{D}_{s_k \leftarrow S_k}$, this second calculation (Equations 11–14) takes place outside of it, as it computes the *unconditional* probability $\Pr(K = k)$. The gap between the two calculations is bridged by a proof rule that captures the law of total expectation, which we will explain and prove in Section 4:

$$\mathop{\mathsf{D}}_{x \leftarrow X}\Big(\mathbb{E}[E] = e\Big) \;\wedge\; \mathbb{E}[e[X/x]] = v \;\vdash\; \mathbb{E}[E] = v \tag{15}$$

Putting all this together yields a formal proof of correctness in Lilac.[8]

To sum up, we have shown how Lilac can be used to verify a constant-space weighted sampling algorithm whose correctness argument requires reasoning about conditional independence of continuous random variables and imports several important results from probability theory, including the law of total expectation and key properties of the uniform distribution. Hopefully, the above example illustrates how Lilac's substructural handling of independence, modal treatment of conditioning, and semantic model grounded in familiar constructs from probability theory allow for easy and natural formalizations of standard pen-and-paper proofs.

Now that we've given a high-level overview of Lilac and its semantic model, the remainder of this paper is devoted to making the notions introduced in this section precise. Section 3 gives the syntax and semantics of Core Lilac – the subset of the logic without D – and proof rules for reasoning about a simple probabilistic programming language capable of expressing the programs written in this section. Section 4 describes how to extend Core Lilac with D to support conditional reasoning and investigates its properties as a modal operator. Finally, Section 5 discusses various tradeoffs we considered in the design of Lilac and possible avenues for future research, and Section 6 compares Lilac with other approaches to reasoning about probabilistic programs.

[6] The more familiar statement of the law of total expectation is $\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X|Y]]$. Here we use the fact that the expectation of an indicator function gives a probability; i.e., $\Pr(E) = \mathbb{E}[\mathbb{1}[E]]$.

[7] If U uniform in $[0, 1]$ then $\mathbb{E}[U^\alpha] = 1/(\alpha + 1)$.

[8] Appendix C contains the proof in full detail.

$$
\begin{aligned}
P, Q ::= {} & \top \mid \bot \mid P \wedge Q \mid P \vee Q \mid P \to Q \mid \\
& P * Q \mid P \mathrel{-\!\!*} Q \mid \Box P \mid \\
& \forall x{:}S.P \mid \exists x{:}S.P \mid \forall_{\mathsf{rv}} X{:}A.P \mid \exists_{\mathsf{rv}} X{:}A.P \mid \\
& E \sim \mu \mid \mathsf{own}\, E \mid \mathbb{E}[E] = e \mid \mathsf{wp}(M, X{:}A.Q)
\end{aligned}
$$

Fig. 2. Core Lilac syntax. Metavariables S, T range over sets and A, B over measurable spaces. Metavariables E, e, M, and μ range over arbitrary measurable maps of a certain type described in Figure 3.

$$
\Gamma ::= \cdot \mid \Gamma, x : S \qquad \llbracket x_1 : S_1, \ldots, x_n : S_n \rrbracket = S_1 \times \cdots \times S_n
$$

$$
\Delta ::= \cdot \mid \Delta, X : A \qquad \llbracket X_1 : A_1, \ldots, X_n : A_n \rrbracket = A_1 \times \cdots \times A_n
$$

$$
\frac{E \in \llbracket \Gamma \rrbracket \to \llbracket \Delta \rrbracket \xrightarrow{\mathrm{m}} A}{\Gamma; \Delta \vdash E : A}\ \text{T-RandE} \qquad
\frac{e \in \llbracket \Gamma \rrbracket \to A}{\Gamma \vdash e : A}\ \text{T-PureE} \qquad
\frac{\mu \in \llbracket \Gamma \rrbracket \to \mathcal{G} A}{\Gamma \vdash \mu : A}\ \text{T-Distr}
$$

$$
\frac{M \in \llbracket \Gamma \rrbracket \to \llbracket \Delta \rrbracket \xrightarrow{\mathrm{m}} \mathcal{G} A}{\Gamma; \Delta \vdash M : A}\ \text{T-Prog} \qquad
\frac{\Gamma, x{:}S; \Delta \vdash P}{\Gamma; \Delta \vdash \forall x{:}S.P}\ \text{T-Pure}\forall \qquad
\frac{\Gamma; \Delta, X{:}A \vdash P}{\Gamma; \Delta \vdash \forall_{\mathsf{rv}} X{:}A.P}\ \text{T-Rand}\forall
$$

$$
\frac{\Gamma; \Delta \vdash E : A \qquad \Gamma \vdash \mu : A}{\Gamma; \Delta \vdash E \sim \mu}\ \text{T-Sim} \qquad
\frac{\Gamma; \Delta \vdash E : \mathbb{R} \quad \Gamma \vdash e : \mathbb{R}}{\Gamma; \Delta \vdash \mathbb{E}[E] = e}\ \text{T-}\mathbb{E} \qquad
\frac{\Gamma; \Delta \vdash M : A \qquad \Gamma; \Delta, X{:}A \vdash Q}{\Gamma; \Delta \vdash \mathsf{wp}(M, X{:}A.Q)}\ \text{T-wp}
$$

Fig. 3. Core Lilac typing rules. Contexts Γ contain the types of pure variables and contexts Δ contain the types of random variables. We use capital letters to stand for random expressions and lower-case letters to stand for pure (i.e., deterministic) expressions.

3 CORE LILAC

Now we begin the formal development of Core Lilac, a subset of Lilac without the disintegration modality. First we will introduce the syntax and semantics of Core Lilac propositions. Before discussing semantics of propositions, we formally define the independent combination operation ($\bullet$) foreshadowed in Section 2, and show that it gives a model of separation logic by proving that it is a Kripke resource monoid (KRM) [33]. Then we give semantics of Lilac propositions. Next we fix a small PPL capable of expressing the examples presented in the previous section, called "APPL". Finally, we connect Lilac to APPL by giving proof rules for reasoning about APPL programs.

3.1 Syntax and Typing of Core Lilac Propositions

The syntax of Core Lilac propositions is given in Figure 2. It includes the standard intuitionistic and substructural connectives, plus the following probability-specific ones:

- $E \sim \mu$ asserts a random expression E follows distribution μ.
- $\mathsf{own}\, E$ asserts "ownership" of a random expression E but no knowledge of its distribution.
- $\mathbb{E}[E] = e$ asserts that the random expression E has expectation e.

Figure 3 gives selected typing rules for Core Lilac.[9] The typing judgment has shape $\Gamma; \Delta \vdash P$, where Γ is a context containing the types of pure variables and Δ is a context containing the types of random variables. Lilac propositions include random expressions E, which may mention both pure and random variables, and pure expressions e, which can only mention pure variables. Accordingly, there are two kinds of quantifiers, with typing rules T-Pure$\forall$ and T-Rand$\forall$.

[9] The full typing rules are in Appendix B.2.

For clarity of presentation the Core Lilac syntax in Figure 2 permits reference to arbitrary measurable spaces in its types and arbitrary measurable functions in its terms, in a manner similar to Shan and Ramsey [37] and Staton [41]. The rule T-RandE characterizes the kinds of functions allowed as random expressions: a random expression E has type A in context $(\Gamma;\Delta)$ if it is a $[\![\Gamma]\!]$-indexed family of measurable maps $[\![\Delta]\!] \xrightarrow{\text{m}} A$. For instance, the following random expressions are all well-typed in Lilac via the T-RandE rule:

$$\frac{\Gamma;\Delta \vdash E_1 : \mathbb{R} \qquad \Gamma;\Delta \vdash E_2 : \mathbb{R}}{\Gamma;\Delta \vdash E_1 + E_2 : \mathbb{R}} \qquad \frac{\Gamma;\Delta \vdash E_1 : \mathbb{R} \qquad \Gamma;\Delta \vdash E_2 : \mathbb{R}}{\Gamma;\Delta \vdash \mathrm{pow}\,(E_1, E_2) : \mathbb{R}}$$

Similarly, T-PureE says a pure expression e is well-typed at A in Γ if it is a function $[\![\Gamma]\!] \to A$, and T-Distr says μ is well-typed at A in context Γ if it is a $[\![\Gamma]\!]$-indexed family of distributions on A. Finally, T-Prog says M is well-typed in $\Gamma;\Delta$ if it is a $[\![\Gamma]\!]$-indexed family of *Markov kernels*, that are maps of the form $A \xrightarrow{\text{m}} \mathcal{G}B$, often used to give semantics to probabilistic programs [41].

To reason about the behavior of programs we add a weakest pre-condition modality in the style of dynamic logic [19, 23]. Intuitively, $\mathrm{wp}(M, X{:}A.Q)$ asserts that M produces a random variable X satisfying postcondition Q. For example, the fact that our two-coin-flip program flip2 from Section 2 produces two independent Bernoulli random variables can be stated as follows:

$$\mathrm{wp}([\![\textsc{flip2}]\!], (X, Y).\ X \sim \mathrm{Ber}\ 1/2 \ *\ Y \sim \mathrm{Ber}\ 1/2). \tag{16}$$

3.2 Combining Independent Probability Spaces

Before we present the semantic interpretation of Core Lilac, we first formally describe our novel combining operation on probability spaces as foreshadowed in Section 2, and show that it forms a Kripke resource monoid and thus gives a model of separation logic [15]:

Definition 3.1. *A* Kripke resource monoid *is a tuple* $(\mathcal{M}, \sqsubseteq, \bullet, \mathbf{1})$ *where*

- $(\mathcal{M}, \sqsubseteq)$ *is a poset,*
- $(\bullet)$ *is a partial function* $\mathcal{M} \times \mathcal{M} \rightharpoonup \mathcal{M}$*,*
- $(\mathcal{M}, \bullet, \mathbf{1})$ *is a partial commutative monoid,*
- $(\bullet)$ *respects* $(\sqsubseteq)$*: if* $x \sqsubseteq x'$ *and* $y \sqsubseteq y'$ *and* $x' \bullet y'$ *defined, then* $x \bullet y$ *defined and* $x \bullet y \sqsubseteq x' \bullet y'$*.*[10]

The choice of $(\bullet)$, which captures the desired notion of separation, determines the interpretations of all of the standard separation logic connectives. Our choice, as foreshadowed in Section 2, combines two independent probability spaces:

Definition 3.2 (Independent Combination). *Let* $(\Omega, \mathcal{E}, \mu)$ *and* $(\Omega, \mathcal{F}, \nu)$ *be probability spaces over the same ambient sample space* Ω*. A probability space* $(\Omega, \mathcal{G}, \rho)$ *is an* independent combination *of* $(\Omega, \mathcal{E}, \mu)$ *and* $(\Omega, \mathcal{F}, \nu)$ *if (1)* $\mathcal{G}$ *is the smallest* σ*-algebra containing* $\mathcal{E}$ *and* $\mathcal{F}$*, and (2)* ρ *witnesses the independence of* μ *and* ν *in the sense that for all* $E \in \mathcal{E}$ *and* $F \in \mathcal{F}$ *it holds that* $\rho(E \cap F) = \mu(E)\nu(F)$ *.*

Recall the program flip2 from Section 2. In this example the probability space $\mathcal{P}_{XY} = (\Omega, \mathcal{G}, \rho)$ is obtained as an independent combination of $\mathcal{P}_X = (\Omega, \mathcal{E}, \mu)$ and $\mathcal{P}_Y = (\Omega, \mathcal{F}, \nu)$, and $\Omega = [0,1] \times [0,1]$. To show this, ρ must witness the independence of μ and ν. Consider the event $X^{-1}(\mathsf{T}) \cap Y^{-1}(\mathsf{T}) = \{(x, y) \mid x < 1/2, y > 1/2\}$, which is the upper-left quadrant of the unit square; this event is in $\mathcal{G}$. The measure ρ witnesses the independence of μ and ν for this event since $1/4 = \rho(X^{-1}(\mathsf{T}) \cap Y^{-1}(\mathsf{T})) = \mu(X^{-1}(\mathsf{T}))\nu(Y^{-1}(\mathsf{T})) = 1/2 \times 1/2$; clearly this holds for all quadrants.

To form a resource monoid, the combining operation $(\bullet)$ must be a partial function. Definition 3.2 requires a witness ρ of independence; if there are multiple possible choices for ρ, it's unclear how to

[10] This is a specialization of Definition 5.5 from Galmiche et al. [15].

turn this definition into a partial function. Thankfully – and somewhat surprisingly – it is possible to establish the uniqueness of ρ and therefore of independent combinations:

Lemma 3.3 (independent combinations are unique). *Suppose* $(\Omega, \mathcal{G}, \rho)$ *and* $(\Omega, \mathcal{G}', \rho')$ *are independent combinations of* $(\Omega, \mathcal{E}, \mu)$ *and* $(\Omega, \mathcal{F}, \nu)$. *Then* $\mathcal{G} = \mathcal{G}'$ *and* $\rho = \rho'$.

Proof. It is straightforward to establish that $\mathcal{G} = \mathcal{G}'$: they are both the smallest σ-algebra containing $\mathcal{E}$ and $\mathcal{F}$. Showing $\rho = \rho'$ requires the use of more heavyweight machinery from probability theory: we apply the well-known *Dynkin π-λ theorem* [24]. The set of events on which ρ and ρ' agree forms a λ-system, and the set $\{E \cap F \mid E \in \mathcal{E}, F \in \mathcal{F}\}$ of intersections of events in $\mathcal{E}$ and $\mathcal{F}$ forms a π-system that generates $\langle \mathcal{E}, \mathcal{F} \rangle = \mathcal{G}$. So the π-λ theorem states that it suffices to show $\rho(E \cap F) = \rho'(E \cap F)$ for all $E \in \mathcal{E}$ and $F \in \mathcal{F}$; this follows since by assumption both sides of the equation factorize into $\mu(E)\nu(F)$. □

Given Lemma 3.3, we can safely write $(\mathcal{E}, \mu) \bullet (\mathcal{F}, \nu) = (\mathcal{G}, \rho)$ whenever $(\mathcal{G}, \rho)$ is an independent combination of $(\mathcal{E}, \mu)$ and $(\mathcal{G}, \nu)$, making $(\bullet)$ a partial function on probability spaces. In addition to being a partial function, Definition 3.1 also requires that $(\bullet)$ forms a partial commutative monoid and respects a certain ordering relation:

Theorem 3.4. *Let* $\mathcal{M}$ *be the set of probability spaces over a fixed sample space* Ω. *Let* $(\bullet)$ *be the partial function mapping two probability spaces to their independent combination if it exists. Let* $(\sqsubseteq)$ *be the ordering such that* $(\mathcal{F}, \mu) \sqsubseteq (\mathcal{G}, \nu)$ *iff* $\mathcal{F} \subseteq \mathcal{G}$ *and* $\mu = \nu|_{\mathcal{F}}$.[11] *The tuple* $(\mathcal{M}, \sqsubseteq, \bullet, \mathbf{1})$ *is a Kripke resource monoid, where* $\mathbf{1}$ *is the trivial probability space* $(\mathcal{F}_{\mathbf{1}}, \mu_{\mathbf{1}})$ *with* $\mathcal{F}_{\mathbf{1}} = \{\emptyset, \Omega\}$ *and* $\mu_{\mathbf{1}}(\Omega) = 1$.

Proof. The main proof obligation is to establish associativity of $(\bullet)$. This follows from an application of the π-λ theorem. The proof is quite intricate; for details, see Appendix B.3. □

3.3 Semantics of Lilac Propositions

Having established that independent combination of probability spaces forms a Kripke resource monoid, we are now ready to present Lilac's semantic model with it as the foundation. Figure 4 gives Lilac's interpretations of standard separation logic connectives, along with interpretations of the two kinds of quantifiers. We describe these familiar rules first. Figure 5 gives the probability-specific rules, which we will discuss after.

Figure 4 defines the meaning of propositions $\Gamma; \Delta \vdash P$ via the relation $\gamma, D, \mathcal{P} \models P$. Unpacking this, (1) the substitution $\gamma \in [\![\Gamma]\!]$ gives instantiations for pure variables, (2) the substitution-valued random variable $D \in \mathrm{RV}[\![\Delta]\!]$ gives instantiations for random variables, and (3) $\mathcal{P}$ is a probability space. Furthermore, both D and $\mathcal{P}$ are defined with respect to a fixed, ambient sample space Ω. The last four lines of Figure 4 give familiar-looking interpretations of quantifiers; we use $\mathrm{RV}\, A$ to denote the set of A-valued random variables. All other lines are standard, and follow from the fact that $(\bullet)$ forms a Kripke resource monoid.

Figure 5 describes the probability-specific Lilac connectives. We start with the first line in the figure, which defines the meaning of ownership. Following the intuition from Section 2, ownership corresponds to measurability of a random variable with respect to a particular σ-algebra. This intuition is made formal here: the proposition $\mathsf{own}\, E$ holds with respect to a configuration $(\gamma, D, (\mathcal{F}, \mu))$ if the random variable denoted by the random expression E is $\mathcal{F}$-measurable.[12] The expression E can have free variables that are either pure or random. The random variable $E(\gamma) \circ D$ is constructed by performing the relevant substitutions: first all pure values are substituted into E,

[11] For a distribution $\mu : \mathcal{G} \to [0, 1]$ and sub-σ-algebra $\mathcal{F} \subseteq \mathcal{G}$, we write $\mu|_{\mathcal{F}}$ for the restriction of μ to $\mathcal{F}$.

[12] Formally, for a probability space $(\Omega, \mathcal{F}, \mu)$, a random variable $X : (\Omega, \mathcal{F}) \to (A, \mathcal{A})$ is *$\mathcal{F}$-measurable* if for every $E \in \mathcal{A}$ it holds that $X^{-1}(E) \in \mathcal{F}$.

$$
\begin{array}{lll}
\gamma, D, \mathcal{P} \vDash \top & \text{always} & \\
\gamma, D, \mathcal{P} \vDash \bot & \text{never} & \\
\gamma, D, \mathcal{P} \vDash P \wedge Q & \text{iff} & \gamma, D, \mathcal{P} \vDash P \text{ and } \gamma, D, \mathcal{P} \vDash Q \\
\gamma, D, \mathcal{P} \vDash P \vee Q & \text{iff} & \gamma, D, \mathcal{P} \vDash P \text{ or } \gamma, D, \mathcal{P} \vDash Q \\
\gamma, D, \mathcal{P} \vDash P \to Q & \text{iff} & \gamma, D, \mathcal{P}' \vDash P \text{ implies } \gamma, D, \mathcal{P}' \vDash Q \text{ for all } \mathcal{P}' \sqsupseteq \mathcal{P} \\
\gamma, D, \mathcal{P} \vDash P * Q & \text{iff} & \gamma, D, \mathcal{P}_P \vDash P \text{ and } \gamma, D, \mathcal{P}_Q \vDash Q \text{ for some } \mathcal{P}_P \bullet \mathcal{P}_Q \sqsubseteq \mathcal{P} \\
\gamma, D, \mathcal{P} \vDash P \mathrel{-\!\!*} Q & \text{iff} & \gamma, D, \mathcal{P}_P \vDash P \text{ implies } \gamma, D, \mathcal{P}_P \bullet \mathcal{P} \vDash Q \text{ for all } \mathcal{P}_P \text{ with } \mathcal{P}_P \bullet \mathcal{P} \text{ defined} \\
\gamma, D, \mathcal{P} \vDash \Box P & \text{iff} & \gamma, D, \mathbf{1} \vDash P \\
\gamma, D, \mathcal{P} \vDash \forall x{:}S.P & \text{iff} & (\gamma, x), D, \mathcal{P} \vDash P \text{ for all } x \in S \\
\gamma, D, \mathcal{P} \vDash \exists x{:}S.P & \text{iff} & (\gamma, x), D, \mathcal{P} \vDash P \text{ for some } x \in S \\
\gamma, D, \mathcal{P} \vDash \forall_{\mathrm{rv}} X{:}A.P & \text{iff} & \gamma, (D, X), \mathcal{P} \vDash P \text{ for all } X : \mathrm{RV}\, A \\
\gamma, D, \mathcal{P} \vDash \exists_{\mathrm{rv}} X{:}A.P & \text{iff} & \gamma, (D, X), \mathcal{P} \vDash P \text{ for some } X : \mathrm{RV}\, A
\end{array}
$$

Fig. 4. Semantics of basic Lilac connectives.

$$
\begin{array}{lll}
\gamma, D, (\mathcal{F}, \mu) \vDash \mathsf{own}\, E & \text{iff} & E(\gamma) \circ D \text{ is } \mathcal{F}\text{-measurable} \\
\gamma, D, (\mathcal{F}, \mu) \vDash E \sim \mu' & \text{iff} & E(\gamma) \circ D \text{ is } \mathcal{F}\text{-measurable and } \mu'(\gamma) = \left(\begin{array}{l} \omega \leftarrow \mu; \\ \mathrm{ret}\ (E(\gamma)(D(\omega))) \end{array}\right) \\
\gamma, D, (\mathcal{F}, \mu) \vDash \mathbb{E}[E] = e & \text{iff} & E(\gamma) \circ D \text{ is } \mathcal{F}\text{-measurable and } \mathbb{E}_{\omega \sim \mu}[E(\gamma)(D(\omega))] = e(\gamma) \\
\gamma, D, \mathcal{P} \vDash \mathsf{wp}(M, X{:}A.Q) & \text{iff} & \text{for all } \mathcal{P}_{\mathrm{frame}} \text{ and } \mu \text{ with } \mathcal{P}_{\mathrm{frame}} \bullet \mathcal{P} \sqsubseteq (\Sigma_\Omega, \mu) \\
 & & \text{and all } D_{\mathrm{ext}} : \mathrm{RV}\, [\![\Delta_{\mathrm{ext}}]\!] \\
 & & \text{there exists } X : \mathrm{RV}\, A \text{ and } \mathcal{P}' \text{ and } \mu' \text{ with } \mathcal{P}_{\mathrm{frame}} \bullet \mathcal{P}' \sqsubseteq (\Sigma_\Omega, \mu') \\
 & & \text{such that } \left(\begin{array}{l} \omega \leftarrow \mu; \\ v \leftarrow M(\gamma)(D(\omega)); \\ \mathrm{ret}\ (D_{\mathrm{ext}}(\omega), D(\omega), v) \end{array}\right) = \left(\begin{array}{l} \omega \leftarrow \mu'; \\ \mathrm{ret}\ (D_{\mathrm{ext}}(\omega), D(\omega), X(\omega)) \end{array}\right) \\
 & & \text{and } \gamma, (D, X), \mathcal{P}' \vDash Q
\end{array}
$$

Fig. 5. Semantics of probability-specific Lilac connectives.

and then the resulting measurable map $E(\gamma)$ is composed with D to produce a random variable. With this connective in hand, we can formally relate separating conjunction in Lilac to the familiar statistical notion of independence of random variables:

Lemma 3.5 (separating conjunction is mutual independence). *Fix a configuration $(\gamma, D, \mathcal{P})$. Abbreviating $X_i(\gamma) \circ D$ as X_i', random variables $X_1', \ldots, X_n'$ are mutually independent with respect to $\mathcal{P}$ iff $\gamma, D, \mathcal{P} \vDash \mathsf{own}\, X_1 * \cdots * \mathsf{own}\, X_n$.*

For a proof, see Appendix B.8. Next, proposition $E \sim \mu'$ holds with respect to $(\gamma, D, (\mathcal{F}, \mu))$ if E owns $\mathcal{F}$ and additionally follows distribution $\mu'(\gamma)$, which is the distribution obtained by substituting the pure values in γ for pure variables in the distribution expression μ'. Throughout this figure, we use Haskell-style notation to construct distributions using the Giry monad [17]; here we use this notation on the right hand side of the equation for $\mu'(\gamma)$ to construct the distribution produced by first sampling a value ω from the ambient probability measure μ and then running E on it. Intuitively, this captures the notion that μ' is the push-forward of μ through E. The interpretation of $\mathbb{E}[E] = e$ has a similar structure.

The most intricate part of Figure 5 is the interpretation of our weakest-precondition modality wp. Intuitively, configurations of the form $(\gamma, D, \mathcal{P})$ represent fragments of a machine state, much like how a configuration (s, h) in ordinary separation logic represents a fragment of the full heap. The idea is that $\mathsf{wp}(M, X{:}A.Q)$ should hold in configuration $(\gamma, D, \mathcal{P})$ if (1) running M with any state containing fragment $\mathcal{P}$ produces a new state containing a new fragment $\mathcal{P}'$ and a new

$$
\begin{array}{rcl}
A, B & ::= & A \times B \mid \texttt{bool} \mid \texttt{real} \mid A^n \mid \texttt{index} \mid \texttt{G}\,A \\
M, N, O & ::= & X \mid \texttt{ret}\ M \mid \texttt{let}\ X = M\ \texttt{in}\ N \mid (M, N) \mid \texttt{fst}\ M \mid \texttt{snd}\ M \mid \\
 & & \texttt{T} \mid \texttt{F} \mid \texttt{if}\ M\ \texttt{then}\ N\ \texttt{else}\ O \mid \texttt{flip}\ p \mid r \mid M \oplus N \mid M \prec N \mid \texttt{uniform} \mid \\
 & & [M, \ldots, M] \mid M[N] \mid \texttt{for}(n, M_{\text{init}}, i\ X.\ M_{\text{step}})
\end{array}
$$

Fig. 6. APPL syntax. Metavariables p range over probabilities, r over real numbers, and n over natural numbers; $\oplus$ and $\prec$ range over standard arithmetic and comparison operators.

random variable X; (2) the new fragment $\mathcal{P}'$ satisfies postcondition Q; (3) any fragments $\mathcal{P}_{\text{frame}}$ independent of $\mathcal{P}$ are preserved by M, which is necessary to establish a frame rule. To enforce (1), we quantify over all probability spaces (Σ_Ω, μ) containing $\mathcal{P}$ and require that running M in μ produce a new probability space (Σ_Ω, μ') containing a new fragment $\mathcal{P}'$ and new random variable X whose distribution is equal to the distribution produced by M. To enforce (2), we require that the new configuration $(\gamma, (D, X), \mathcal{P}')$ satisfy Q. To enforce (3), we quantify over all possible "frames" $\mathcal{P}_{\text{frame}}$, and require that the new space μ' contain the exact same frame unchanged. Finally, in order to prove a fundamental substitution lemma, we quantify over arbitrary extensions D_{ext} to the random substitution D; for details on this technical point see Appendix B.4.1.

3.4 Syntax and Semantics of APPL

Our next goal is to establish a program logic that leverages Core Lilac. To do this here we fix a small probabilistic programing language called APPL capable of expressing the examples in Section 2. The syntax of APPL is given in Figure 6. It is a simply-typed first-order calculus with a sampling operation, arrays, and bounded loops. It has a simple monadic type-system as in Staton [41]. The important monadic typing rules are:

$$
\frac{}{\Gamma \vdash \texttt{uniform} : \texttt{G real}}\ \text{T-Unif} \qquad \frac{}{\Gamma \vdash \texttt{ret}\ M : \texttt{G}\,A}\ \text{T-Ret} \qquad \frac{\Gamma \vdash M : \texttt{G}\,A \qquad \Gamma, X : A \vdash N : \texttt{G}\,B}{\Gamma \vdash \texttt{let}\ X = M\ \texttt{in}\ N : \texttt{G}\,B}\ \text{T-Bind}
$$

Monadic computations have type $\texttt{G}\,A$; the G stands for the standard Giry monad [17]. The T-Unif rule states that uniform is a probabilistic computation producing a real number.

The semantics for APPL are standard and follow Staton [41]. Types A are interpreted as measurable spaces $[\![A]\!]$ and typing contexts $\Gamma = \{x_1 : A_1, \ldots, n_n : A_N\}$ as products $[\![\Gamma]\!] = [\![A_1]\!] \times \cdots \times [\![A_n]\!]$. Terms $\Gamma \vdash M : \texttt{G}\,A$ are interpreted as measurable maps $[\![M]\!] : [\![\Gamma]\!] \xrightarrow{\text{m}} \mathcal{G}[\![A]\!]$, where $\mathcal{G}[\![A]\!]$ is the set of distributions on $[\![A]\!]$. The full semantics can be found in Appendix A.3.

3.5 Reasoning About APPL Programs

We now show how the semantic model described in the previous section validates standard proof rules for reasoning about APPL programs. As in Iris [23], we give meaning to Hoare triples using the wp and $\Box$ modalities: for any APPL program prog, we define

$$
\{P\}\ \texttt{prog}\ \{X.\, Q\} \quad := \quad \Box\,(P \mathrel{-\!\!*} \texttt{wp}\,([\![\texttt{prog}]\!], X.\, Q))\,. \tag{17}
$$

Standard proof rules for reasoning about APPL programs are consequences of laws about the wp modality. Figure 7 shows a selection of such rules. The structural rules Consequence, Frame, and Disjunction are completely standard; in contrast to prior work on probabilistic separation logics [5], our frame rule requires no extra side conditions about the program's dataflow properties. Notably missing from this collection of structural rules is a rule for conjunction; for discussion of this point, see Section 5.

The remaining rules in Figure 7 facilitate reasoning about APPL constructs. The rules H-Pure and H-Let are the standard ones for pure computations and let-bindings. The rules H-Uniform and H-Flip specify the behavior of APPL's sampling operations; they closely resemble the standard

$$\frac{P \vdash P' \qquad Q' \vdash Q \qquad \{P'\}\ M\ \{X.\, Q'\}}{\{P\}\ M\ \{X.\, Q\}}\ \text{H-Consequence} \qquad \frac{\{P\}\ M\ \{X.\, Q\}}{\{F * P\}\ M\ \{X.\, F * Q\}}\ \text{H-Frame}\ (X \notin F)$$

$$\frac{\{P\}\ M\ \{X.\, Q\} \qquad \{P'\}\ M\ \{X.\, Q'\}}{\{P \vee P'\}\ M\ \{X.\, Q \vee Q'\}}\ \text{H-Disjunction} \qquad \frac{}{\{Q[\![M]\!]/X]\}\ \texttt{ret}\ M\ \{X.\, Q\}}\ \text{H-Pure}$$

$$\frac{\{P\}\ M\ \{X.\, Q\} \qquad \forall_{\mathsf{rv}} X.\ \{Q\}\ N\ \{Y.\, R\}}{\{P\}\ \texttt{let}\ X = M\ \texttt{in}\ N\ \{Y.\, R\}}\ \text{H-Let} \qquad \frac{}{\{\top\}\ \texttt{uniform}\ \{X.\, X \sim \mathrm{Unif}\ [0,1]\})}\ \text{H-Uniform}$$

$$\frac{}{\{\top\}\ \texttt{flip}\ p\ \{X.\, X \sim \mathrm{Ber}\ p\}}\ \text{H-Flip} \qquad \frac{\forall i{:}\mathbb{N}.\ \forall_{\mathsf{rv}} X{:}A.\ \{I(i,X)\}\ M\ \{X'.\, I(i+1,X')\}}{\{I(1,e)\}\ \texttt{for}(n, e, i\ X.\ M)\ \{X{:}A.\, I(n+1,X)\}}\ \text{H-For}$$

Fig. 7. Selected proof rules for reasoning about APPL programs.

rule for allocating a new reference in ordinary separation logic, formalizing the intuition that sampling is like allocation. Finally, the rule H-For is a standard proof principle for reasoning about APPL's for-loops: it states that one can conclude postcondition $I(n+1, X)$ after running a for-loop if an invariant $I(i, X)$ – a proposition indexed by the loop iteration i and value of the accumulator variable X – holds on entry of the initial accumulator e and is maintained by every loop iteration.

Thus far we have been rather abstract about the ambient sample space Ω underlying Lilac's semantic model. At this point we must make a concrete choice in order to validate the proof rules in Figure 7. The soundness of H-Uniform and H-Flip require constructing a new probability space independent of an existing one. To ensure that it is always possible to construct such a fresh probability space, we fix a particular choice of Ω and restrict our Kripke resource monoid to a class of probability spaces on Ω with so-called "finite footprint"; this guarantees that there is always enough "room" in Ω for new probability spaces to be allocated.

Specifically, we fix Ω to be the Hilbert cube $[0,1]^{\mathbb{N}}$, the collection of infinite streams of real numbers in the interval $[0,1]$; these infinite streams can be thought of as infinitely-replenishable randomness sources for use throughout a probabilistic program's execution [10, 46]. A probability space has finite footprint if it only uses finitely-many dimensions of the Hilbert cube:

Definition 3.6. *A σ-algebra $\mathcal{F}$ on $[0,1]^{\mathbb{N}}$ has* finite footprint *if there is some finite n such that every $F \in \mathcal{F}$ is of the form $F' \times [0,1]^{\mathbb{N}}$ for some $F' \subseteq [0,1]^n$.*

Then we restrict our Kripke resource monoid on probability spaces to only those probability spaces with finite footprint. With this choice of Ω and a restriction to suitably-well-behaved probability spaces in hand, we can validate the above proof rules:

Theorem 3.1. *The proof rules in Figure 7 are sound.*

Proof. The structural rules, H-Pure, and H-Let follow straightforwardly from unwinding the definitions of Hoare triples and the interpretations of the logical connectives. The rule H-For follows by induction on the number of loop iterations. As foreshadowed, the rules H-Uniform and H-Flip require constructing a new probability space independent of an existing one; because the existing space only exhausts some finite n dimensions of the Hilbert cube, we are free to allocate the new probability space in dimensions $n+1$ and above. For details see Appendix B.11. □

$$\textsc{D-Entail}\ \frac{P \vdash Q}{\mathsf{D}_{x\leftarrow E}\, P \vdash \mathsf{D}_{x\leftarrow E}\, Q} \qquad \textsc{D-And}\ \mathsf{D}_{x\leftarrow E}(P \wedge Q) \dashv\vdash \mathsf{D}_{x\leftarrow E}\, P \wedge \mathsf{D}_{x\leftarrow E}\, Q \qquad \textsc{D-Indep}\ (\mathsf{own}\, E) * P \vdash \mathsf{D}_{x\leftarrow E}\, P$$

$$\textsc{D-Subst}\ \vdash \mathsf{D}_{x\leftarrow X}\left(\mathbb{E}[E] = \mathbb{E}[E[x/X]]\right) \qquad \textsc{D-Total-Expectation}\ \mathsf{D}_{x\leftarrow X}\left(\mathbb{E}[E] = e\right) \wedge \mathbb{E}[e[X/x]] = v \vdash \mathbb{E}[E] = v$$

Fig. 8. Selected properties of Lilac's conditioning modality D.

4 A DISINTEGRATION-BASED MODALITY FOR CONDITIONAL PROBABILITY

So far we have presented Core Lilac, which defines probabilistic interpretations of the standard separation logic connectives, along with some atomic propositions for making probability-specific assertions. We now describe the second main contribution of this paper: Lilac's modal operator for reasoning about conditional probability. We extend Core Lilac with the proposition $\mathsf{D}_{x:A\leftarrow E}\, P$ and the following typing rule:

$$\frac{\Gamma;\Delta \vdash E : A \qquad \Gamma, x{:}A;\Delta \vdash P}{\Gamma;\Delta \vdash \mathsf{D}_{x:A\leftarrow E}\, P}\ \text{T-D}$$

The intended reading of $\mathsf{D}_{x:A\leftarrow E}\, P$ is that P holds conditional on the event $E = x$ for all x. Intuitively, we can think of this mode of reasoning as "freezing" a random quantity E, temporarily setting its random nature aside and reasoning about it as if it were a deterministic quantity x in order to establish P. This is reflected in the typing rule by the fact that x is a pure variable.

Giving a semantics to D requires the notion of a *disintegration* [9, 37]. Before getting out the heavy machinery, we first give some intuition for this new modality and its properties. Figure 8 lists useful laws about D. The first two rules, D-Entail and D-And, are structural in nature: D-Entail says D respects entailment and D-And says it distributes over conjunction. Together these rules allow ordinary logical reasoning to be carried out under D, with D-Entail automatically lifting statements and proofs about unconditional probability to the conditional setting. We saw an example of this phenomenon in Section 2.1, where Indep-Prod was automatically lifted to a statement about the conditional probability of a conjunction of conditionally independent events.

The remaining rules in Figure 8 express standard probability-theoretic statements about conditioning. The rule D-Indep states that if P holds independent of some random expression E, then P also holds conditional on $E = x$ for any x; this acts as a form of introduction rule for D. The rule D-Subst captures the intuition that, for an expression E with random free variable X, substituting x for X does not change conditional expectations given $X = x$. Finally, D-Total-Expectation states the Law of Total Expectation, which acts as a form of elimination rule for D. It says that, to compute the expectation of a random expression E, one can proceed in two stages: first, compute the conditional expectation of E given $X = x$, yielding some pure expression e in terms of the conditioned x; then, compute the desired unconditional expectation by putting the random X back into e and taking the expectation of the resulting expression $e[X/x]$. The proof of correctness of the weighted sampling algorithm in Section 2.1 made use of the following instantiation of this proof rule:

$$\mathsf{D}_{s_k\leftarrow S_k}\left(\mathbb{E}[\underbrace{\mathbb{1}[K=k]}_{E}] = \underbrace{\mathrm{pow}\left(s_k, \frac{\sum_{j\neq k} w_j}{w_k}\right)}_{e}\right) \wedge \left(\mathbb{E}\left[\underbrace{\mathrm{pow}\left(S_k, \frac{\sum_{j\neq k} w_j}{w_k}\right)}_{e[S_k/s_k]}\right] = \underbrace{\frac{w_k}{\sum_j w_j}}_{v}\right) \vdash \mathbb{E}[\underbrace{\mathbb{1}[K=k]}_{E}] = \underbrace{\frac{w_k}{\sum_j w_j}}_{v}$$

4.1 Semantics of the Disintegration Modality

The rules stated in Figure 8 give a powerful and intuitive framework for reasoning about conditioning that would be familiar to an experienced probability theorist. Our goal in this section is to identify a model that validates these rules. In an ideal world, $(\gamma, D, \mathcal{P}) \vDash \mathsf{D}_{x:A \leftarrow X}\, P$ holds if for all $x \in A$ there exists some *conditioned space* $\mathcal{P}_{X=x}$ such that $\gamma, D, \mathcal{P}|_{X=x} \vDash P$. We would like to define $\mathcal{P}|_{X=x}$ using the standard definition of conditional probability: let $\mathcal{P} = (\Omega, \mathcal{F}, \mu)$ and define $\mathcal{P}|_{X=x} = (\Omega, \mathcal{F}, \mu|_{X=x})$ where $\mu|_{X=x}(E) = \mu(E \cap \{X = x\})/\mu(\{X = x\})$. This definition for the conditioned space $\mathcal{P}|_{X=x}$ is well-defined for discrete random variables X, where it is common to disregard the case when $\mu(\{X = x\}) = 0$. However, if X is a continuous random variable, then $\mu(\{X = x\}) = 0$ for all x, so it is impossible to disregard this case. This restriction is a problem for us because we would like Lilac to support continuous random variables.

In probability theory, *disintegrations* were developed in order to resolve this issue and give a natural notion of conditioned spaces for continuous random variables [9]. A *disintegration* for a probability space $\mathcal{P}$ with respect to a random variable X is defined as a collection of all conditioned spaces $\{\mathcal{P}|_{X=x}\}_{x \in A}$ satisfying certain measurability and concentration properties [9]. The existence of a disintegration for a probability space and random variable is a very strong condition, and not all probability spaces $\mathcal{P}$ will have a well-defined disintegration for all random variables X. The study of disintegrations has formally characterized some of the conditions under which there exist well-defined notions of disintegration [9]. We leverage this knowledge here to design a model for our disintegration modality in Lilac.

Our strategy will be to identify a suitable class of probability spaces that is both large enough to accommodate all of our design criteria and examples, and well-behaved enough to admit all reasonable disintegrations. We use this class to validate the rules in Figure 8. Our starting point in this search is the *Hilbert cube*, the countable product of unit intervals $[0, 1]^{\mathbb{N}}$. The Hilbert cube is disintegrable with respect to a large class of random variables (those whose codomain has a well-behaved σ-algebra):

Lemma 4.1. *Let $X : [0, 1]^{\mathbb{N}} \to (A, \mathcal{A})$ be a random variable and $\mathcal{P}$ be a probability space on the Hilbert cube. If $\mathcal{A}$ is countably-generated and contains all singletons, then there exists a disintegration of $\mathcal{P}$ with respect to X.*

Proof. The Hilbert cube is a complete separable metric space [39] and any probability measure on it is by definition finite Borel; the result follows from Theorem 1.4 of Chang and Pollard [9]. □

The class of spaces $(A, \mathcal{A})$ required by Lemma 4.1 includes many familiar examples, such as $\mathbb{R}^n$, $\mathbb{N}$, and all finite spaces with the usual powerset σ-algebra. Since the Hilbert cube is the sample space Ω underlying Lilac's semantic model, this result allows us to disintegrate configurations $(\gamma, D, \mathcal{P})$ whenever $\mathcal{P}$ is a probability space whose σ-algebra is exactly the Borel σ-algebra on the Hilbert cube, and whose measure μ is correspondingly a Borel measure. However, our configurations are not quite of this form: μ may be a probability measure on a sub-σ-algebra on the Hilbert cube, and such measures unfortunately cannot in general be extended to a Borel measure [13]. This motivates the next step in our search for suitably-well-behaved probability spaces:

Theorem 4.1. *Let $\mathcal{M}_{Borel}$ be the set of probability spaces on the Hilbert cube of the form $(\Omega, \mathcal{F}, \mu)$, where μ can be extended to a Borel measure. The restriction of the KRM given by Theorem 3.4 to $\mathcal{M}_{Borel}$ is still a KRM.*

A proof of this theorem is in Appendix B.6. The upshot of Theorem 4.1 is that configurations of the form $(\gamma, D, \mathcal{P})$ where $\mathcal{P} \in \mathcal{M}_{\text{Borel}}$ can be extended to the Hilbert cube, where they are disintegrable with respect to suitably-well-behaved random variables following Lemma 4.1. The final step in our

search is motivated by the desire to validate rule D-Indep in Figure 8. The soundness of D-Indep requires the ability to show that a union of negligible sets (a set with measure 0) remains negligible. In general this is not the case, so we need to further specialize our model. We force these unions to be countable – from which the result follows straightforwardly from the axioms of probability – by restricting ourselves to probability spaces with *countably-generated σ-algebras*. Putting this all together yields the final Kripke resource monoid underlying Lilac's semantic model:

Theorem 4.2. *Let $\mathcal{M}_{\text{disintegrable}}$ be the set of countably-generated probability spaces $\mathcal{P}$ that have finite footprint and can be extended to a Borel measure on the entire Hilbert cube. The restriction of the KRM given by Theorem 3.4 to $\mathcal{M}_{\text{disintegrable}}$ is still a KRM.*

For a proof see Appendix B.14. Using Theorem 4.2, we can finally give an interpretation to D:

Lemma 4.2. *The following interpretation of $\mathsf{D}_{x:A\leftarrow E}\, P$ is well-defined and validates the rules in Figure 8: $\gamma, D, \mathcal{P} \vDash \mathsf{D}_{x:A\leftarrow E}\, P$ iff for all $(\Sigma_\Omega, \mu) \sqsupseteq \mathcal{P}$ and all disintegrations of μ along $E(\gamma) \circ D$ into $\{\nu_x\}_{x\in A}$, and almost all $x \in A$, it holds that $G, (\mathcal{F}, \nu_x|_{\mathcal{P}}) \vDash P$.*

For a detailed proof, see Appendix B.18.

5 DISCUSSION AND FUTURE WORK

In this section we explore various possible extensions to Lilac and expound on the more subtle consequences of some of the design decisions we made while validating certain proof rules. First we will discuss why the structural rule for conjunction is absent from Figure 7. Then we discuss possible avenues for integrating Lilac into Iris [23], along with possible extensions to support languages with first-class support for conditioning. Next we further explore the modal interpretation of D, and discuss the degree to which it behaves like well-known modalities from modal logic. Then we describe connections between our notion of independent combination of probability spaces and the more standard notion of independence of σ-algebras from probability theory. Finally we discuss possible future work towards a more principled model of Lilac with the aim of supporting languages with more advanced features such as higher-order functions and recursion [7].

Structural rule for conjunction. Standard separation logics often admit the following rule:

$$\frac{\{P\}\ M\ \{X.\, Q\} \qquad \{P'\}\ M\ \{X.\, Q'\}}{\{P \wedge P'\}\ M\ \{X.\, Q \wedge Q'\}}$$

This rule is notably absent from our rules in Figure 7. We have found neither a proof nor counterexample of its validity. Such a rule is not straightforward to validate due to the nondeterministic choice of $\mathcal{P}'$ in the interpretation of wp. Intuitively, the postconditions Q and Q' may be witnessed by two different probability spaces $\mathcal{P}$ and $\mathcal{P}'$; to prove the rule sound, one must somehow combine these two probability spaces into one which witnesses the conjunction $Q \wedge Q'$. A similar challenge with adding a rule for conjunction arises in concurrent separation logic, where the conjunction rule requires the invariants to be *precise* [34]. We leave the resolution of this rule – perhaps by a notion of precision for Lilac assertions – to future work.

Embedding Lilac into Iris. In the future we would like to embed Lilac in Iris, so that we can take advantage of Iris's rich support for reasoning about recursive and higher-order programs, and its interface for carrying out interactive higher-order separation logic proofs [23, 28]. To do so requires expressing Lilac's KRM as a *camera* [23], which is an algebraic structure similar to a KRM that additionally supports step-indexed reasoning. One difference between cameras and KRMs is that cameras are not parameterized by an extra ordering relation: for cameras, $(\sqsubseteq)$ is implicitly defined to be relation $x \sqsubseteq y \Leftrightarrow \exists z.x \bullet z = y$. However, Lilac's KRM includes ordering relations $\mathcal{P} \sqsubseteq \mathcal{R}$

that are not of the form $\mathcal{P} \bullet \mathcal{Q} = \mathcal{R}$ for any $\mathcal{Q}$. Therefore, embedding Lilac into Iris would require finding a way to bridge this gap between KRMs and cameras. Additionally, a naive adaptation of Lilac's KRM into a camera would not suffice for increasing its reasoning power: making use of Iris's support for step-indexed reasoning requires developing a suitable step-indexed generalization of the KRM in Theorem 3.4 so that one can talk about equality of probability spaces "up to k steps." We leave these problems for future work.

Observations. In this paper we focused on applying Lilac to verifying randomized algorithms. In this setting the programming language being reasoned about does not typically have a notion of observation or Bayesian conditioning. Many probabilistic programing languages in the machine learning setting do support Bayesian reasoning, and it would be interesting to be able to apply Lilac to these languages too [22, 41]. Languages of this sort have a syntax like `observe` E to condition on the event E, whose semantics is usually to reject all program executions that do not satisfy E. A minor extension to APPL's semantic model and small adjustment to Lilac's interpretation of wp could enable Lilac to express proof rules of the following form:[13]

$$\{\mathsf{own}\, E\}\ \texttt{observe}\ E\ \{\Pr(E = \mathsf{T}) = 1\}.$$

This rule states that, given ownership of the random expression E, observation alters the underlying probability space so that E is equal to T with probability 1. The ability for `observe` to modify the underlying probability space suggests an analogy between conditioning and mutable update; indeed, the above proof rule closely resembles a proof rule for mutable update in a typical imperative separation logic. This connection between conditioning and mutable update is unlikely to be a tight correspondence, since conditioning operations can be arbitrarily reordered [40] while writes to memory cannot; nonetheless, in future work we would like to further explore whether this connection can be exploited to bring more techniques from the world of mutable state to bear on probabilistic program verification tasks.

Properties of the disintegration modality. Here we investigate further some formal properties of the disintegration modality. Specifically, we compare D to modal necessity $\Box$ [29]. The standard properties of $\Box$ are:

- Necessitation: if $\vdash P$ then $\vdash \Box P$.
- Distribution: $\Box(P \to Q) \vdash \Box P \to \Box Q$.
- Distributes over $(\wedge)$: $\Box(P \wedge Q) \dashv\vdash \Box P \wedge \Box Q$.
- Semidistributes over $(\vee)$: $\Box P \vee \Box Q \vdash \Box(P \vee Q)$.
- Axiom M: $\Box P \vdash P$.
- Axiom 4: $\Box P \vdash \Box\Box P$.

The disintegration modality satisfies the following analogous properties, stated and proved in Lemma B.19 in the appendix:

- Necessitation: if $\vdash P$ then $\vdash \mathsf{D}_{x \leftarrow X} P$.
- Distribution: $\mathsf{D}_{x \leftarrow X}(P \to Q) \vdash \mathsf{D}_{x \leftarrow X} P \to \mathsf{D}_{x \leftarrow X} Q$.
- Distributes over $(\wedge)$: $\mathsf{D}_{x \leftarrow X}(P \wedge Q) \dashv\vdash \mathsf{D}_{x \leftarrow X} P \wedge \mathsf{D}_{x \leftarrow X} Q$.
- Semidistributes over $(\vee)$: $\mathsf{D}_{x \leftarrow X} P \vee \mathsf{D}_{x \leftarrow X} Q \vdash \mathsf{D}_{x \leftarrow X}(P \vee Q)$.

This similarity between our disintegration modality and modal necessity is somewhat expected, due to the similarity in the logical structure of their interpretations: $\mathsf{D}_{x \leftarrow E} P$ requires P to hold in almost-all conditional probability spaces $\mathcal{P}|_{X=x}$, similar to how the usual interpretation of $\Box P$ in modal logic requires that P hold in all reachable worlds. Notably absent from our list above are

[13] Currently, APPL's semantic domain of Markov kernels assumes program's denotations are normalized; observations induce unnormalized distributions.

analogues of Axiom 4 and Axiom M. We are not sure whether Axiom 4 holds. Axiom M however has a counterexample – this is to be expected, as Axiom M in standard modal logic says that what is necessary is the case, whereas we do not expect something that holds conditional on $X = x$ to hold unconditionally, even if it holds conditional on $X = x$ for all x.

Independence of σ-algebras. There is a curious mismatch between our definition of independent combination and the standard definition of independence of σ-algebras in probability theory. The standard notion of independence of two sub-σ-algebras $\mathcal{E}, \mathcal{F} \subseteq \mathcal{G}$ with respect to an ambient probability space $(\Omega, \mathcal{G}, \rho)$ states that ρ factorizes along $\mathcal{E}$ and $\mathcal{F}$: i.e., it says that for all $E \in \mathcal{E}, F \in \mathcal{F}$, it holds that $\rho(E \cap F) = \rho(E)\rho(F)$. This definition presupposes the existence of an ambient measure ρ by which the independence of $\mathcal{E}$ and $\mathcal{F}$ can be judged. In contrast, our independent combination *constructs* a combined measure ρ from the measures μ and ν. Our contribution is that this combining operation forms a partial function on probability spaces with the structure of a partial commutative monoid; once one has combined μ and ν to obtain ρ in this way, our definition coincides with the standard one.

Formal structure of Lilac models. In constructing a model for Lilac that validates our proof rules we made a number of design decisions about what sort of probability spaces to include. Following Biering et al. [7], it would be interesting future work to pursue a principled methodology for characterizing the space of such valid probabilistic models of separation logic. Such a methodolgy would potentially facilitate future extensions to support probabilistic languages with more sophisticated features such as higher-order functions, polymorphism, mutable state, and concurrency.

6 RELATED WORK

Verification of probabilistic programs has a long history going back to Kozen [27]. In this section we sketch the broad themes that are most closely related to program logics for probabilistic programs. First, we discuss approaches that make use of separation logic. Then, we discuss alternative approaches to probabilistic program verification, based on expectations, logical relations, and denotational semantics.

6.1 Program Logics for Probability

The most closely related work is the probabilistic separation logic (PSL) introduced by Barthe et al. [5], which gives the first separation logic where separating conjunction explicitly models statistical independence. Follow-on work extends PSL to support negative dependence [3] and to settings beyond probabilistic computation [47]. PSL interprets separating conjunction as a combining operation on distributions over random stores with disjoint domains, over-approximating the semantic notion of statistical independence with a semi-syntactic criterion on stores. As a consequence, PSL's notion of independence is linked to the occurrences of free variables in logical formulas, and so statements such as $\mathsf{own}(X + Y) * \mathsf{own}(X - Y)$ are inexpressible in PSL due to the occurrence of the random variables X and Y on both sides of $*$. A further consequence of PSL's notion of separation is that its frame rule imposes a number of extra side-conditions capturing data-flow properties of the program that hampers its application. Lilac's frame rule is standard for separation logic. Lilac's interpretation of separating conjunction coincides with statistical independence, as demonstrated by Lemma 3.5, and its semantic model is defined in terms of standard objects of probability theory (i.e., probability spaces and random variables). This makes it easier to import standard mathematical constructions. In particular, we have exploited this close correspondence to easily support continuous random variables (unsupported by PSL and its extensions), to add logical connectives expressing the expectations of random variables, and to import results from disintegration theory in the construction of our modal operator for

expressing conditional probability. All these features in combination seem difficult to add to PSL without significant changes to its semantic model.

Bao et al. [2] extends PSL to handle conditional independence by extending the standard logic of bunched implications underlying separation logic with a family of specially-designed connectives in a new logic called *doubly-bunched implications* (DIBI). The corresponding model required for proving soundness of DIBI deviates significantly from the usual model of separation logic. Lilac handles conditional independence via the disintegration modality, and this extension does not require any changes to the standard model beyond the restriction to well-behaved probability spaces (Theorem 4.2). In Lilac, conditional independence is a derived concept expressible as a combination of disintegration and separating conjunction. As a consequence, Lilac behaves very similarly to existing separation logics while still having facilities for handling conditional independence.

A separate line of probabilistic program logics seeks to verify probabilistic programs correct without a substructural notion of independence. An example of this style of logic is Ellora [4]. In Ellora, independence is encoded as an assertion about factorization of probabilities. This is similar to how in program logics without separating conjunction, aliasing can be ruled out by assertions stating the pairwise-disjointness of heap locations. This limited the modularity of Ellora proofs significantly, since independence is often a key notion in simplifying an argument for correctness. To address this limitation Ellora is equipped with the ability to embed logics such as a *law and independence logic*, which makes it capable of reasoning about mutual independence relationships. However, these embedded logics are rather limited: Barthe et al. [5] note that the resulting independence logic cannot handle conditional control flow and that it is more ergonomic to use a substructural logic to implicitly handle statistical independence. This limitation was a primary motivation for developing PSL.

Another strategy for designing a separation logic for probabilistic programs is to identify a notion of separation other than statistical independence. Tassarotti and Harper [44] introduced Polaris, an extension of Iris for verifying concurrent randomized algorithms. The goal of Polaris is very different from Lilac's, and so it makes different design choices. The notion of separation in Polaris is the standard one in separation logic enforcing ownership of disjoint heap fragments. To reason about probability, Polaris enriches base Iris with the ability to make coupling-style arguments. As a consequence, Polaris has no substructural treatment of independence or method for stating facts involving conditioning. Additionally, Polaris does not support continuous random variables. Yet another way to generalize separation logic to the probabilistic setting is given by Batz et al. [6] who introduced quantitative separation logic (QSL). QSL generalizes the meaning of assertions: rather than interpreting assertions as predicates on configurations, i.e. functions from configurations to Boolean values, QSL interprets predicates as functions from configurations to expectations. In QSL, separating conjunction does not model independence as in Lilac or PSL.

6.2 Expectation-based approaches

Classically the dominant approach to verifying randomized algorithms has been expectation-based techniques such as PPDL [27] and pGCL [32]. These approaches reason about expected quantities of probabilistic programs, and design a weakest-pre-expectation operator that propagates information about expected values backwards through the program. These methods have been widely-used in practice, verifying properties such as probabilistic bounds and running-times of randomized algorithms [18, 25, 35]. However, expectation-based approaches verify a single property about expectations at a time. As a consequence, verifying multiple interwoven properties of expectations can require repeated separate verification passes, leading to cumbersome and non-modular proofs. These limitations in expectation-based approaches were an important motivation for the development of probabilistic program logics like Ellora [4].

6.3 Logical Relations for Probabilistic Programs

A separate method for reasoning about probabilistic programs recasts the reasoning problem as a relational one, seeking to verify that two programs are equivalent. Logical relations are a proof-technique for establishing equivalence of programs, and recent work has generalized this strategy to the probabilistic setting [10, 45, 46]. Culpepper and Cobb [10] and Wand et al. [45] characterize contextual equivalence of a language with higher-order functions, continuous random variables, and scoring via a biorthogonal logical relation. Zhang and Amin [46] characterize contextual equivalence of a language that additionally supports nested queries via a step-indexed biorthogonal logical relation. These models are all based on an operational semantics defined using the Hilbert cube, and serve as the inspiration for our use of the Hilbert cube in Lilac's semantic model. Though logical relations are well-suited for proving the validity of program rewrite rules, they are less well-suited for proving intricate post-conditions that can be stated in a program logic.

6.4 Probabilistic Denotational Semantics

An entirely separate method for verifying properties of probabilistic program seeks to associate programs with a well-behaved mathematical model and perform all reasoning in the model. For example, Staton [40] gives a denotational model for validating intuitive commutativity and rewrite rules for programs by associating them with an appropriate category that witnesses the correctness of these transformations. Recently there have been significant developments towards designing general-purpose models for probabilistic programs that are convenient for proving various properties [14, 20, 42, 43]. The drawbacks of using denotational approaches to verify probabilistic programs are similar to the drawbacks of using denotational approaches for verifying non-probabilistic programs: extensions to the language often require drastic changes to the denotational model. Nonetheless there are interesting connections between Lilac and denotational semantics due to our use of the Giry monad in giving a semantics to our wp-modality. One possible avenue for future work is to replace this with a richer domain such as quasi-Borel spaces [20] in order to enrich Lilac with capability for reasoning about higher-order functions.

7 CONCLUSION

Lilac is a probabilistic separation logic with support for continuous random variables and conditional reasoning whose interpretation of separating conjunction coincides with the ordinary notion of statistical independence. The core contributions of Lilac are (1) a novel notion of separation based on independent combination of probability spaces; and (2) a modal treatment of conditional probability, which includes a set of proof rules for reasoning about conditional expectations that would be intuitive to an experienced probability theorist. To demonstrate Lilac, we derived proof rules for reasoning about a simple probabilistic programming language and showed how they can be used in combination with Lilac's other features to prove a sophisticated weighted sampling algorithm correct. Notably, the derived proof rules mirror those of ordinary separation logic: rules for sampling resemble the usual rules for allocation, and our frame rule is completely standard. Ultimately, we envision Lilac becoming a standard tool in the toolkit for verifying probabilistic programs. For future work, we are curious if Lilac can be extended to the quantum programming setting in a style similar to Zhou et al. [47], or if it can handle the exotic forms of negative dependence studied in Bao et al. [3].

REFERENCES

[1] Aws Albarghouthi, Loris D'Antoni, Samuel Drews, and Aditya V Nori. 2017. Fairsquare: probabilistic verification of program fairness. *Proceedings of the ACM on Programming Languages* 1, OOPSLA (2017), 1–30.

[2] Jialu Bao, Simon Docherty, Justin Hsu, and Alexandra Silva. 2021. A bunched logic for conditional independence. In *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE, 1–14.

[3] Jialu Bao, Marco Gaboardi, Justin Hsu, and Joseph Tassarotti. 2022. A separation logic for negative dependence. *Proceedings of the ACM on Programming Languages* 6, POPL (2022), 1–29.

[4] Gilles Barthe, Thomas Espitau, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. 2018. An assertion-based program logic for probabilistic programs. In *European Symposium on Programming*. Springer, Cham, 117–144.

[5] Gilles Barthe, Justin Hsu, and Kevin Liao. 2019. A Probabilistic Separation Logic. *Proc. ACM Program. Lang.* 4, POPL, Article 55 (dec 2019), 30 pages. https://doi.org/10.1145/3371123

[6] Kevin Batz, Benjamin Lucien Kaminski, Joost-Pieter Katoen, Christoph Matheja, and Thomas Noll. 2019. Quantitative separation logic: a logic for reasoning about probabilistic pointer programs. *Proceedings of the ACM on Programming Languages* 3, POPL (2019), 1–29.

[7] Bodil Biering, Lars Birkedal, and Noah Torp-Smith. 2007. BI-hyperdoctrines, higher-order separation logic, and abstraction. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 29, 5 (2007), 24–es.

[8] James Bornholt, Todd Mytkowicz, and Kathryn S. McKinley. 2014. Uncertain<T>: A First-Order Type for Uncertain Data. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems* (Salt Lake City, Utah, USA) *(ASPLOS '14)*. Association for Computing Machinery, New York, NY, USA, 51–66. https://doi.org/10.1145/2541940.2541958

[9] Joseph T Chang and David Pollard. 1997. Conditioning as disintegration. *Statistica Neerlandica* 51, 3 (1997), 287–317.

[10] Ryan Culpepper and Andrew Cobb. 2017. Contextual equivalence for probabilistic programs with continuous random variables and scoring. In *European Symposium on Programming*. Springer, 368–392.

[11] Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. 2017. A storm is coming: A modern probabilistic model checker. In *International Conference on Computer Aided Verification*. Springer, 592–600.

[12] Pavlos S Efraimidis and Paul G Spirakis. 2006. Weighted random sampling with a reservoir. *Information processing letters* 97, 5 (2006), 181–185.

[13] MP Ershov. 1975. Extension of measures and stochastic equations. *Theory of Probability & Its Applications* 19, 3 (1975), 431–444.

[14] Tobias Fritz. 2020. A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics. *Advances in Mathematics* 370 (2020), 107239.

[15] Didier Galmiche, Daniel Méry, and David Pym. 2005. The semantics of BI and resource tableaux. *Mathematical Structures in Computer Science* 15, 6 (2005), 1033–1088.

[16] Timon Gehr, Sasa Misailovic, Petar Tsankov, Laurent Vanbever, Pascal Wiesmann, and Martin Vechev. 2018. Bayonet: probabilistic inference for networks. *ACM SIGPLAN Notices* 53, 4 (2018), 586–602.

[17] Michele Giry. 1982. A categorical approach to probability theory. In *Categorical aspects of topology and analysis*. Springer, 68–85.

[18] Friedrich Gretz, Joost-Pieter Katoen, and Annabelle McIver. 2014. Operational versus weakest pre-expectation semantics for the probabilistic guarded command language. *Performance Evaluation* 73 (2014), 110–132.

[19] David Harel, Dexter Kozen, and Jerzy Tiuryn. 2001. Dynamic logic. In *Handbook of philosophical logic*. Springer, 99–217.

[20] Chris Heunen, Ohad Kammar, Sam Staton, and Hongseok Yang. 2017. A convenient category for higher-order probability theory. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE, 1–12.

[21] Steven Holtzen, Sebastian Junges, Marcell Vazquez-Chanlatte, Todd Millstein, Sanjit A Seshia, and Guy Van den Broeck. 2021. Model checking finite-horizon Markov chains with probabilistic inference. In *International Conference on Computer Aided Verification*. Springer, 577–601.

[22] Steven Holtzen, Guy Van den Broeck, and Todd Millstein. 2020. Scaling exact inference for discrete probabilistic programs. *Proceedings of the ACM on Programming Languages* 4, OOPSLA (2020), 1–31.

[23] Ralf Jung, Robbert Krebbers, Jacques-Henri Jourdan, Aleš Bizjak, Lars Birkedal, and Derek Dreyer. 2018. Iris from the ground up: A modular foundation for higher-order concurrent separation logic. *Journal of Functional Programming* 28 (2018).

[24] Olav Kallenberg. 1997. *Foundations of modern probability*. Vol. 2. Springer.

[25] Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. 2016. Inferring covariances for probabilistic programs. In *International Conference on Quantitative Evaluation of Systems*. Springer, 191–206.

[26] Benjamin Lucien Kaminski, Joost-Pieter Katoen, Christoph Matheja, and Federico Olmedo. 2016. Weakest precondition reasoning for expected run–times of probabilistic programs. In *European Symposium on Programming*. Springer, 364–389.

[27] Dexter Kozen. 1983. A probabilistic pdl. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*. 291–297.

[28] Robbert Krebbers, Amin Timany, and Lars Birkedal. 2017. Interactive proofs in higher-order concurrent separation logic. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*. 205–217.

[29] Saul A Kripke. 1972. Naming and necessity. In *Semantics of natural language*. Springer, 253–355.

[30] Marta Kwiatkowska, Gethin Norman, and David Parker. 2002. PRISM: Probabilistic symbolic model checker. In *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*. Springer, 200–204.

[31] Marcel Moosbrugger, Ezio Bartocci, Joost-Pieter Katoen, and Laura Kovács. 2021. Automated termination analysis of polynomial probabilistic programs. In *European Symposium on Programming*. Springer, Cham, 491–518.

[32] Carroll Morgan, Annabelle McIver, and Karen Seidel. 1996. Probabilistic predicate transformers. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 18, 3 (1996), 325–353.

[33] Peter W O'Hearn. 2012. A Primer on Separation Logic (and Automatic Program Verification and Analysis). *Software safety and security* 33 (2012), 286–318.

[34] Peter W O'Hearn, Hongseok Yang, and John C Reynolds. 2009. Separation and information hiding. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 31, 3 (2009), 1–50.

[35] Federico Olmedo, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. 2016. Reasoning about recursive probabilistic programs. In *2016 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE, 1–10.

[36] Robert Rand and Steve Zdancewic. 2015. VPHL: A verified partial-correctness logic for probabilistic programs. *Electronic Notes in Theoretical Computer Science* 319 (2015), 351–367.

[37] Chung-chieh Shan and Norman Ramsey. 2017. Exact Bayesian inference by symbolic disintegration. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*. 130–144.

[38] Steffen Smolka, Praveen Kumar, David M Kahn, Nate Foster, Justin Hsu, Dexter Kozen, and Alexandra Silva. 2019. Scalable verification of probabilistic networks. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*. 190–203.

[39] Sashi Mohan Srivastava. 2008. *A course on Borel sets*. Vol. 180. Springer Science & Business Media.

[40] Sam Staton. 2017. Commutative semantics for probabilistic programming. In *European Symposium on Programming*. Springer, 855–879.

[41] Sam Staton. 2020. Probabilistic programs as measures. *Foundations of Probabilistic Programming* (2020), 43.

[42] Sam Staton, Hongseok Yang, Frank Wood, Chris Heunen, and Ohad Kammar. 2016. Semantics for probabilistic programming: higher-order functions, continuous distributions, and soft constraints. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*. 525–534.

[43] Dario Maximilian Stein. 2021. Structural foundations for probabilistic programming languages. *University of Oxford* (2021).

[44] Joseph Tassarotti and Robert Harper. 2019. A separation logic for concurrent randomized programs. *Proceedings of the ACM on Programming Languages* 3, POPL (2019), 1–30.

[45] Mitchell Wand, Ryan Culpepper, Theophilos Giannakopoulos, and Andrew Cobb. 2018. Contextual equivalence for a probabilistic language with continuous random variables and recursion. *Proceedings of the ACM on Programming Languages* 2, ICFP (2018), 1–30.

[46] Yizhou Zhang and Nada Amin. 2022. Reasoning about "reasoning about reasoning": semantics and contextual equivalence for probabilistic programs with nested queries and recursion. *Proceedings of the ACM on Programming Languages* 6, POPL (2022), 1–28.

[47] Li Zhou, Gilles Barthe, Justin Hsu, Mingsheng Ying, and Nengkun Yu. 2021. A quantum interpretation of bunched logic & quantum separation logic. In *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE, 1–14.

A SYNTAX AND SEMANTICS OF APPL

A.1 Syntax

$$
\begin{array}{rcl}
p & \in & [0,1] \\
r & \in & \mathbb{R} \\
n & \in & \mathbb{N} \\
\oplus & \in & \text{Arith} := \{+, -, \times, /, \hat{}\} \\
\prec & \in & \text{Cmp} := \{<, \leq, =\} \\
A, B & ::= & A \times B \mid \mathsf{bool} \mid \mathsf{real} \mid A^n \mid \mathsf{index} \mid \mathsf{G}\, A \\
M, N, O & ::= & X \mid \mathsf{ret}\ M \mid \mathsf{let}\ X = M\ \mathsf{in}\ N \mid \\
 & & (M, N) \mid \mathsf{fst}\ M \mid \mathsf{snd}\ M \mid \\
 & & \mathsf{T} \mid \mathsf{F} \mid \mathsf{if}\ M\ \mathsf{then}\ N\ \mathsf{else}\ O \mid \mathsf{flip}\ p \mid \\
 & & r \mid M \oplus N \mid M \prec N \mid \mathsf{uniform} \mid \\
 & & [M, \ldots, M] \mid M[N] \mid \mathsf{for}(n, M_{\text{init}}, i\ X.\ M_{\text{step}})
\end{array}
$$

A.2 Typing

$$\Gamma, X : A \vdash X : A \qquad \Gamma \vdash \mathsf{ret}\ M : \mathsf{G}\, A \qquad \frac{\Gamma \vdash M : \mathsf{G}\, A \qquad \Gamma, X : A \vdash N : \mathsf{G}\, B}{\Gamma \vdash \mathsf{let}\ X = M\ \mathsf{in}\ N : \mathsf{G}\, B} \qquad \frac{\Gamma \vdash M : A \qquad \Gamma \vdash N : B}{\Gamma \vdash (M, N) : A \times B}$$

$$\frac{\Gamma \vdash M : A \times B}{\Gamma \vdash \mathsf{fst}\ M : A} \qquad \frac{\Gamma \vdash M : A \times B}{\Gamma \vdash \mathsf{snd}\ M : B} \qquad \Gamma \vdash \mathsf{T} : \mathsf{bool} \qquad \Gamma \vdash \mathsf{F} : \mathsf{bool}$$

$$\frac{\Gamma \vdash M : \mathsf{bool} \qquad \Gamma \vdash N : A \qquad \Gamma \vdash O : A}{\Gamma \vdash \mathsf{if}\ M\ \mathsf{then}\ N\ \mathsf{else}\ O : A} \qquad \Gamma \vdash \mathsf{flip}\ p : \mathsf{G}\,\mathsf{bool} \qquad \Gamma \vdash r : \mathsf{real}$$

$$\frac{\Gamma \vdash M : \mathsf{real} \qquad \Gamma \vdash N : \mathsf{real}}{\Gamma \vdash M \oplus N : \mathsf{real}} \qquad \frac{\Gamma \vdash M : \mathsf{real} \qquad \Gamma \vdash N : \mathsf{real}}{\Gamma \vdash M \prec N : \mathsf{bool}} \qquad \Gamma \vdash \mathsf{uniform} : \mathsf{G}\,\mathsf{real}$$

$$\frac{\Gamma \vdash M_k : A \text{ for all } 1 \leq k \leq n}{\Gamma \vdash [M_1, \ldots, M_n] : A^n} \qquad \frac{\Gamma \vdash M : A^n \qquad \Gamma \vdash N : \mathsf{index}}{\Gamma \vdash M[N] : A}$$

$$\frac{\Gamma \vdash M_{\text{init}} : A \qquad \Gamma, i : \mathsf{index}, X : A \vdash M_{\text{step}} : \mathsf{G}\, A}{\Gamma \vdash \mathsf{for}(n, M_{\text{init}}, i\ X.\ M_{\text{step}}) : \mathsf{G}\, A}$$

A.3 Semantics

Types and typing contexts are interpreted as measurable spaces, arithmetic operators as maps $\mathbb{R} \times \mathbb{R} \xrightarrow{\text{m}} \mathbb{R}$, and comparison operators as maps $\mathbb{R} \times \mathbb{R} \xrightarrow{\text{m}} [\![\texttt{bool}]\!]$.

$$
\begin{aligned}
[\![A \times B]\!] &= [\![A]\!] \otimes [\![B]\!] \\
[\![\texttt{bool}]\!] &= (\{\mathsf{T}, \mathsf{F}\}, \mathcal{P}(\{\mathsf{T}, \mathsf{F}\})) \\
[\![\texttt{real}]\!] &= (\mathbb{R}, \mathcal{B}(\mathbb{R})) \\
[\![A^n]\!] &= \underbrace{[\![A]\!] \otimes \cdots \otimes [\![A]\!]}_{n \text{ times}} \\
[\![\texttt{index}]\!] &= (\mathbb{N}, \mathcal{P}(\mathbb{N})) \\
[\![\mathsf{G}\, A]\!] &= \mathcal{G}[\![A]\!] \\
[\![\cdot]\!] &= \text{the one-point space} \\
[\![\Gamma, X{:}A]\!] &= [\![\Gamma]\!] \otimes [\![A]\!]
\end{aligned}
$$

$$
\begin{aligned}
x[\![+]\!]y &= x + y \\
x[\![-]\!]y &= x - y \\
x[\![\times]\!]y &= xy \\
x[\![/]\!]y &= \begin{cases} x/y, & y \neq 0 \\ 0, & \text{otherwise} \end{cases} \\
x[\![\hat{}]\!]y &= \begin{cases} x^y, & x > 0 \\ 0, & \text{otherwise} \end{cases}
\end{aligned}
\qquad
x[\![<]\!]y = \begin{cases} \mathsf{T}, & x < y \\ \mathsf{F}, & \text{otherwise} \end{cases}
$$

$$
x[\![\leq]\!]y = \begin{cases} \mathsf{T}, & x \leq y \\ \mathsf{F}, & \text{otherwise} \end{cases}
\qquad
x[\![=]\!]y = \begin{cases} \mathsf{T}, & x = y \\ \mathsf{F}, & \text{otherwise} \end{cases}
$$

Terms $\Gamma \vdash M : A$ are interpreted as maps $[\![M]\!] : [\![\Gamma]\!] \xrightarrow{\text{m}} [\![A]\!]$.

$$
\begin{aligned}
[\![X]\!]\rho &= \rho(X) \\
[\![\texttt{ret}\ M]\!]\rho &= \text{ret}\, [\![M]\!]\rho \\
[\![\texttt{let}\ X = M\ \texttt{in}\ N]\!]\rho &= v \leftarrow [\![M]\!]\rho;\ [\![N]\!]\rho[X \mapsto v] \\
[\![(M, N)]\!]\rho &= ([\![M]\!]\rho, [\![N]\!]\rho) \\
[\![\texttt{fst}\ M]\!]\rho &= \pi_1([\![M]\!]\rho) \\
[\![\texttt{snd}\ M]\!]\rho &= \pi_2([\![M]\!]\rho) \\
[\![\mathsf{T}]\!]\rho &= \mathsf{T} \\
[\![\mathsf{F}]\!]\rho &= \mathsf{F} \\
[\![\texttt{if}\ M\ \texttt{then}\ N\ \texttt{else}\ O]\!]\rho &= \begin{cases} [\![N]\!]\rho, & [\![M]\!]\rho = \mathsf{T} \\ [\![O]\!]\rho, & [\![M]\!]\rho = \mathsf{F} \end{cases} \\
[\![\texttt{flip}\ p]\!]\rho &= \text{Ber}\, p \\
[\![r]\!]\rho &= r \\
[\![M \oplus N]\!]\rho &= ([\![M]\!]\rho)[\![\oplus]\!]([\![N]\!]\rho) \\
[\![\texttt{uniform}]\!]\rho &= \text{Unif}\ [0, 1] \\
[\![\texttt{[}M_1, \ldots, M_n\texttt{]}]\!]\rho &= ([\![M_1]\!]\rho, \ldots, [\![M_n]\!]\rho) \\
[\![M\texttt{[}N\texttt{]} : A]\!]\rho &= \begin{cases} \pi_{[\![N]\!]\rho}([\![M]\!]\rho), & 1 \leq [\![N]\!]\rho \leq n \\ \text{arbitrary}(A), & \text{otherwise} \end{cases} \\
[\![\texttt{for}(n, M_\text{i}, i\ X.\ M_\text{s})]\!]\rho &= \text{loop}(1, [\![M_\text{i}]\!]\rho, \lambda\, k\ v.\ [\![M_\text{s}]\!]\rho[i \mapsto k, X \mapsto v])
\end{aligned}
$$

$$
\text{where loop}(k, v, f) = \begin{cases} \text{ret}\, v, & k > n \\ v' \leftarrow f(k, v);\ \text{loop}(k+1, v', f), & \text{otherwise} \end{cases}
$$

To make array indexing total, $\mathrm{arbitrary}(A)$ produces an arbitrary inhabitant of $[\![A]\!]$.

$$\begin{aligned}\mathrm{arbitrary}(A \times B) &= (\mathrm{arbitrary}(A), \mathrm{arbitrary}(B))\\ \mathrm{arbitrary}(\texttt{bool}) &= \mathsf{T}\\ \mathrm{arbitrary}(\texttt{real}) &= 0\\ \mathrm{arbitrary}(A^n) &= \underbrace{(\mathrm{arbitrary}(A), \ldots, \mathrm{arbitrary}(A))}_{n \text{ times}}\\ \mathrm{arbitrary}(\texttt{index}) &= 0\end{aligned}$$

B SYNTAX AND SEMANTICS OF LILAC

B.1 Syntax

$$\begin{aligned}S, T &\in \mathbf{Set}\\ A, B &\in \mathbf{Meas}\\ P, Q &::= \top \mid \bot \mid P \wedge Q \mid P \vee Q \mid P \to Q \mid\\ &\quad P * Q \mid P \mathrel{-\!\!*} Q \mid \Box P \mid\\ &\quad \forall x{:}S.P \mid \exists x{:}S.P \mid \forall_{\mathsf{rv}} X{:}A.P \mid \exists_{\mathsf{rv}} X{:}A.P \mid\\ &\quad E \sim \mu \mid \mathsf{own}\, E \mid E =_{\mathsf{a.s.}} E \mid \mathbb{E}[E] = e \mid \mathrm{wp}(M, X{:}A.Q)\end{aligned}$$

B.2 Typing

$$\begin{aligned}\Gamma &::= \cdot \mid \Gamma, x : S\\ \Delta &::= \cdot \mid \Delta, X : A\end{aligned}$$

$$\begin{aligned}[\![x_1 : S_1, \ldots, x_n : S_n]\!] &= S_1 \times \cdots \times S_n\\ [\![X_1 : A_1, \ldots, X_n : A_n]\!] &= A_1 \otimes \cdots \otimes A_n\end{aligned}$$

$$\frac{E \in [\![\Gamma]\!] \to [\![\Delta]\!] \xrightarrow{\mathrm{m}} A}{\Gamma; \Delta \vdash E : A} \qquad \frac{e \in [\![\Gamma]\!] \to A}{\Gamma \vdash e : A} \qquad \frac{\mu \in [\![\Gamma]\!] \to \mathcal{G}A}{\Gamma \vdash \mu : A} \qquad \frac{M \in [\![\Gamma]\!] \to [\![\Delta]\!] \xrightarrow{\mathrm{m}} \mathcal{G}A}{\Gamma; \Delta \vdash M : A}$$

$$\Gamma; \Delta \vdash \top \qquad \Gamma; \Delta \vdash \bot \qquad \frac{\Gamma; \Delta \vdash P \qquad \Gamma; \Delta \vdash Q}{\Gamma; \Delta \vdash P \wedge Q} \qquad \frac{\Gamma; \Delta \vdash P \qquad \Gamma; \Delta \vdash Q}{\Gamma; \Delta \vdash P \vee Q} \qquad \frac{\Gamma; \Delta \vdash P \qquad \Gamma; \Delta \vdash Q}{\Gamma; \Delta \vdash P \to Q}$$

$$\frac{\Gamma; \Delta \vdash P \qquad \Gamma; \Delta \vdash Q}{\Gamma; \Delta \vdash P * Q} \qquad \frac{\Gamma; \Delta \vdash P \qquad \Gamma; \Delta \vdash Q}{\Gamma; \Delta \vdash P \mathrel{-\!\!*} Q} \qquad \frac{\Gamma; \Delta \vdash P}{\Gamma; \Delta \vdash \Box P}$$

$$\frac{\Gamma, x{:}S; \Delta \vdash P}{\Gamma; \Delta \vdash \forall x{:}S.P} \qquad \frac{\Gamma, x{:}S; \Delta \vdash P}{\Gamma; \Delta \vdash \exists x{:}S.P} \qquad \frac{\Gamma; \Delta, X{:}A \vdash P}{\Gamma; \Delta \vdash \forall_{\mathsf{rv}} X{:}A.P} \qquad \frac{\Gamma; \Delta, X{:}A \vdash P}{\Gamma; \Delta \vdash \exists_{\mathsf{rv}} X{:}A.P}$$

$$\frac{\Gamma; \Delta \vdash E : A \qquad \Gamma \vdash \mu : A}{\Gamma; \Delta \vdash E \sim \mu} \qquad \frac{\Gamma; \Delta \vdash E : A}{\Gamma; \Delta \vdash \mathsf{own}\, E} \qquad \frac{\Gamma; \Delta \vdash E_1 : A \qquad \Gamma; \Delta \vdash E_2 : A}{\Gamma; \Delta \vdash E_1 =_{\mathsf{a.s.}} E_2}$$

$$\frac{\Gamma; \Delta \vdash E : \mathbb{R} \qquad \Gamma \vdash e : \mathbb{R}}{\Gamma; \Delta \vdash \mathbb{E}[E] = e} \qquad \frac{\Gamma; \Delta \vdash M : A \qquad \Gamma; \Delta, X{:}A \vdash Q}{\Gamma; \Delta \vdash \mathrm{wp}(M, X{:}A.Q)}$$

B.3 Independent combination of probability spaces

Lemma B.1. *If $\mathcal{F}$ and $\mathcal{G}$ are σ-algebras on Ω then the set $\mathcal{E} := \{F \cap G \mid F \in \mathcal{F}, G \in \mathcal{G}\}$ of intersections of events in $\mathcal{F}$ and $\mathcal{G}$ is a π-system that generates $\langle \mathcal{F}, \mathcal{G} \rangle$.*

Proof. First let's show that $\mathcal{E}$ is a π-system. The set $\mathcal{E}$ is nonempty because it at least has to contain $\emptyset$. It's closed under finite intersections because if $(F_1 \cap G_1) \in \mathcal{E}$ and $(F_2 \cap G_2) \in \mathcal{E}$ then $(F_1 \cap G_1) \cap (F_2 \cap G_2) = \underbrace{(F_1 \cap F_2)}_{\in \mathcal{F}} \cap \underbrace{(G_1 \cap G_2)}_{\in \mathcal{G}} \in \mathcal{E}$, where the last step follows from the fact that $\mathcal{F}$ and $\mathcal{G}$ are both σ-algebras and hence closed under intersections.

Now we just have to show $\langle \mathcal{E} \rangle = \langle \mathcal{F}, \mathcal{G} \rangle$. As sets of generators, $\mathcal{E}$ contains the union of $\mathcal{F}$ and $\mathcal{G}$: because $\mathcal{F}$ and $\mathcal{G}$ are σ-algebras $\mathcal{E}$ includes intersections of the form $F \cap \Omega = F$ and $\Omega \cap G = G$ for all $F \in \mathcal{F}$ and $G \in \mathcal{G}$. This implies $\langle \mathcal{F}, \mathcal{G} \rangle \subseteq \langle \mathcal{E} \rangle$. For the other direction, note that every generator $(F \cap G) \in \mathcal{E}$ is an intersection of generators $F \in \mathcal{F}, G \in \mathcal{G}$. □

Lemma B.2. *If $(\mathcal{F}, \mu)$ is a probability space then $\mathcal{F}^\perp := \{E \mid E \perp \mathcal{F}\}$[14] is a λ-system.*

Proof. Clearly $\emptyset \in \mathcal{F}^\perp$ because $\emptyset \perp E$ for any E. If $E \perp \mathcal{F}$ then $E^c \perp \mathcal{F}$, so $\mathcal{F}$ is closed under complements. Finally, if $\{A_n\}_{n \in \mathbb{N}}$ is a collection of disjoint sets in $\mathcal{F}^\perp$ then $\mu(\bigcup_n A_n \cap E) = \sum_n \mu(A_n \cap E) = \sum_n \mu(A_n)\mu(E) = \mu(E)\mu(\bigcup_n A_n)$ for all E, so $\mathcal{F}^\perp$ is closed under countable disjoint union. □

Theorem B.3. *Let $\mathcal{M}$ be the set of probability spaces over a fixed sample space Ω. Let $(\bullet)$ be the partial function mapping two probability spaces to their independent combination if it exists. Let $(\sqsubseteq)$ be the ordering such that $(\mathcal{F}, \mu) \sqsubseteq (\mathcal{G}, \nu)$ iff $\mathcal{F} \subseteq \mathcal{G}$ and $\mu = \nu|_{\mathcal{F}}$.*[15] *The tuple $(\mathcal{M}, \sqsubseteq, \bullet, \mathbf{1})$ is a Kripke resource monoid, where $\mathbf{1}$ is the trivial probability space $(\mathcal{F}_\mathbf{1}, \mu_\mathbf{1})$ with $\mathcal{F}_\mathbf{1} = \{\emptyset, \Omega\}$ and $\mu_\mathbf{1}(\Omega) = 1$.*

Proof. $\mathbf{1}$ is indeed a unit: if $(\mathcal{F}, \mu)$ is some other probability space on Ω then $\langle \mathcal{F}, \mathcal{F}_\mathbf{1} \rangle = \mathcal{F}$ and μ witnesses the independent combination of itself with $\mu_\mathbf{1}$. And the relation "$\mathcal{P}$ is an independent combination of $\mathcal{Q}$ and $\mathcal{R}$" is clearly symmetric in $\mathcal{Q}$ and $\mathcal{R}$, so $(\bullet)$ is commutative. We just need to show $(\bullet)$ is associative and respects $(\sqsubseteq)$.

For associativity, suppose $(\mathcal{F}_1, \mu_1) \bullet (\mathcal{F}_2, \mu_2) = (\mathcal{F}_{12}, \mu_{12})$ and $(\mathcal{F}_{12}, \mu_{12}) \bullet (\mathcal{F}_3, \mu_3) = (\mathcal{F}_{(12)3}, \mu_{(12)3})$. There are three things to check:

- Some μ_{23} witnesses the combination of $(\mathcal{F}_2, \mu_2)$ and $(\mathcal{F}_3, \mu_3)$.
- Some $\mu_{1(23)}$ witnesses the combination of $(\mathcal{F}_1, \mu_1)$ and $(\mathcal{F}_{23}, \mu_{23})$.
- $(\langle \mathcal{F}_1, \langle \mathcal{F}_2, \mathcal{F}_3 \rangle \rangle, \mu_{1(23)}) = (\langle \langle \mathcal{F}_1, \mathcal{F}_2 \rangle, \mathcal{F}_3 \rangle, \mu_{(12)3})$.

We'll show this as follows:

(1) $\langle \mathcal{F}_1, \langle \mathcal{F}_2, \mathcal{F}_3 \rangle \rangle = \langle \langle \mathcal{F}_1, \mathcal{F}_2 \rangle, \mathcal{F}_3 \rangle$.
(2) Define $\mu_{23} := \mu_{(12)3}|_{\mathcal{F}_{23}}$. This is a witness for $(\mathcal{F}_2, \mu_2)$ and $(\mathcal{F}_3, \mu_3)$.
(3) Define $\mu_{1(23)} := \mu_{(12)3}$. This is a witness for $(\mathcal{F}_1, \mu_1)$ and $(\mathcal{F}_{23}, \mu_{23})$.

To show the left-to-right inclusion for (1): by the universal property of freely-generated σ-algebras, we just need to show $\langle \langle \mathcal{F}_1, \mathcal{F}_2 \rangle, \mathcal{F}_3 \rangle$ is a σ-algebra containing $\mathcal{F}_1$ and $\langle \mathcal{F}_2, \mathcal{F}_3 \rangle$. It clearly contains $\mathcal{F}_1$. To show it contains $\langle \mathcal{F}_2, \mathcal{F}_3 \rangle$, we just need to show it contains $\mathcal{F}_2$ and $\mathcal{F}_3$ (by the universal property again), which it clearly does. The right-to-left inclusion is similar.

For (2), if $E_2 \in \mathcal{F}_2$ and $E_3 \in \mathcal{F}_3$ then $\mu_{23}(E_2 \cap E_3) = \mu_{(12)3}(E_2 \cap E_3) = \mu_{(12)3}((\Omega \cap E_2) \cap E_3) = \mu_{12}(\Omega \cap E_2)\mu_3(E_3) = \mu_1(\Omega)\mu_2(E_2)\mu_3(E_3) = \mu_2(E_2)\mu_3(E_3)$ as desired.

For (3), we need $\mu_{(12)3}(E_1 \cap E_{23}) = \mu_1(E_1)\mu_{23}(E_{23})$ for all $E_1 \in \mathcal{F}_1$ and $E_{23} \in \langle \mathcal{F}_2, \mathcal{F}_3 \rangle$. For this we use the π-λ theorem. Let $\mathcal{E}$ be the set $\{E_2 \cap E_3 \mid E_2 \in \mathcal{F}_2, E_3 \in \mathcal{F}_3\}$ of intersections of events in $\mathcal{F}_2$ and $\mathcal{F}_3$. $\mathcal{E}$ is a π-system that generates $\langle \mathcal{F}_2, \mathcal{F}_3 \rangle$ (lemma B.1). Let $\mathcal{G}$ be the set of events E_{23} such that $\mu_{(12)3}(E_1 \cap E_{23}) = \mu_1(E_1)\mu_{23}(E_{23})$ for all $E_1 \in \mathcal{F}_1$. We are done if $\langle \mathcal{E} \rangle \subseteq \mathcal{G}$. By the π-λ theorem, we just need to check that $\mathcal{E} \subseteq \mathcal{G}$ and that $\mathcal{G}$ is a λ-system. We have $\mathcal{E} \subseteq \mathcal{G}$ because if $E_2 \in \mathcal{F}_2$ and $E_3 \in \mathcal{F}_3$ then $\mu_{(12)3}(E_1 \cap (E_2 \cap E_3)) = \mu_1(E_1)\mu_2(E_2)\mu_3(E_3) = \mu_1(E_1)\mu_{23}(E_2 \cap E_3)$. To see that $\mathcal{G}$ is a λ-system, note that $\mu_1(E_1)\mu_{23}(E_{23}) = \mu_{(12)3}(E_1)\mu_{(12)3}(E_{23})$ and so $\mathcal{G}$ is actually equal to $\mathcal{F}_1^\perp$ (the set of events independent of $\mathcal{F}_1$), a λ-system by Lemma B.2.

To show $(\bullet)$ respects $(\sqsubseteq)$, suppose $(\mathcal{F}, \mu) \sqsubseteq (\mathcal{F}', \mu')$ and $(\mathcal{G}, \nu) \sqsubseteq (\mathcal{G}', \nu')$ and $(\mathcal{F}', \mu') \bullet (\mathcal{G}', \nu') = (\langle \mathcal{F}', \mathcal{G}' \rangle, \rho')$. We need to show (1) $(\mathcal{F}, \mu) \bullet (\mathcal{G}, \nu) = (\langle \mathcal{F}, \mathcal{G} \rangle, \rho)$ and (2) $(\langle \mathcal{F}, \mathcal{G} \rangle, \rho) \sqsubseteq (\langle \mathcal{F}', \mathcal{G}' \rangle, \rho')$ for some ρ. Define ρ to be the restriction of ρ' to $\langle \mathcal{F}, \mathcal{G} \rangle$. Now (1) holds because $\rho(F \cap G) = \rho'(F \cap G) =$

[14] $E \perp \mathcal{F}$ iff $\mu(E \cap F) = \mu(E)\mu(F)$ for all $F \in \mathcal{F}$.

$\rho'(F)\rho'(G) = \rho(F)\rho(G)$ for all $F \in \mathcal{F}$ and $G \in \mathcal{G}$ (the second step follows from $\mathcal{F} \subseteq \mathcal{F}'$ and $\mathcal{G} \subseteq \mathcal{G}'$). For (2), $\langle \mathcal{F}, \mathcal{G} \rangle \subseteq \langle \mathcal{F}', \mathcal{G}' \rangle$ because $\mathcal{F} \subseteq \mathcal{F}'$ and $\mathcal{G} \subseteq \mathcal{G}'$, and $\rho = \rho'|_{\langle \mathcal{F}, \mathcal{G} \rangle}$ by construction. □

B.4 Semantics

Let Ω be the Hilbert cube $[0,1]^{\mathbb{N}}$, and let Σ_Ω be the standard Borel σ-algebra on the Hilbert cube generated by the product topology.

Definition B.4. *A sub-σ-algebra $\mathcal{F}$ of Σ_Ω has* finite footprint *if there is some n such that every $F \in \mathcal{F}$ is of the form $F' \times [0,1]^{\mathbb{N}}$ for some $F' \subseteq [0,1]^n$.*

Definition B.5. *A random variable $X : (\Omega, \Sigma_\Omega) \to (A, \Sigma_A)$ has* finite footprint *if the pullback σ-algebra $\{X^{-1}(E) \mid E \in \Sigma_A\}$ has finite footprint.*

Lemma B.6. *Let $\mathcal{M}_{\text{finite}}$ be the set of probability spaces $\mathcal{P}$ with finite footprint whose σ-algebras are sub-σ-algebras of the standard Borel σ-algebra on $[0,1]^{\mathbb{N}}$. The restriction of the KRM given by Theorem B.3 to $\mathcal{M}_{\text{finite}}$ is still a KRM.*

Proof. If m and n witness the finite footprints of independently-combinable probability spaces $\mathcal{P}$ and $\mathcal{Q}$ then $\max(m,n)$ witnesses the finite footprint of their independent combination $\mathcal{P} \bullet \mathcal{Q}$, and if $\mathcal{F}$ and $\mathcal{G}$ are two sub-σ-algebras of the Borel σ-algebra on the Hilbert cube, then so is the σ-algebra $\langle \mathcal{F}, \mathcal{G} \rangle$. Thus $(\bullet)$ remains closed under $\mathcal{M}_{\text{finite}}$, which suffices to show that it remains a KRM. □

Let $\mathrm{RV}\, A$ be the set of measurable maps $[0,1]^{\mathbb{N}} \xrightarrow{\text{m}} A$ with finite footprint. Interpret propositions $\Gamma; \Delta \vdash P$ as sets of configurations $(\gamma, D, \mathcal{P})$ where $\gamma \in [\![\Gamma]\!]$, $D \in \mathrm{RV}\, [\![\Delta]\!]$, and $\mathcal{P} \in \mathcal{M}_{\text{finite}}$.

Lemma B.7. *The following interpretations of basic connectives is well-formed:*

$\gamma, D, \mathcal{P} \vDash \top$	*always*	
$\gamma, D, \mathcal{P} \vDash \bot$	*never*	
$\gamma, D, \mathcal{P} \vDash P \wedge Q$	*iff*	$\gamma, D, \mathcal{P} \vDash P$ *and* $\gamma, D, \mathcal{P} \vDash Q$
$\gamma, D, \mathcal{P} \vDash P \vee Q$	*iff*	$\gamma, D, \mathcal{P} \vDash P$ *or* $\gamma, D, \mathcal{P} \vDash Q$
$\gamma, D, \mathcal{P} \vDash P \to Q$	*iff*	$\gamma, D, \mathcal{P}' \vDash P$ *implies* $\gamma, D, \mathcal{P}' \vDash Q$ *for all* $\mathcal{P}' \sqsupseteq \mathcal{P}$
$\gamma, D, \mathcal{P} \vDash P * Q$	*iff*	$\gamma, D, \mathcal{P}_P \vDash P$ *and* $\gamma, D, \mathcal{P}_Q \vDash Q$ *for some* $\mathcal{P}_P \bullet \mathcal{P}_Q \sqsubseteq \mathcal{P}$
$\gamma, D, \mathcal{P} \vDash P \mathrel{-\!\!*} Q$	*iff*	$\gamma, D, \mathcal{P}_P \vDash P$ *implies* $\gamma, D, \mathcal{P}_P \bullet \mathcal{P} \vDash Q$ *for all* $\mathcal{P}_P$ *with* $\mathcal{P}_P \bullet \mathcal{P}$ *defined*
$\gamma, D, \mathcal{P} \vDash \Box P$	*iff*	$\gamma, D, 1 \vDash P$
$\gamma, D, \mathcal{P} \vDash \forall x{:}S.P$	*iff*	$(\gamma, x), D, \mathcal{P} \vDash P$ *for all* $x \in S$
$\gamma, D, \mathcal{P} \vDash \exists x{:}S.P$	*iff*	$(\gamma, x), D, \mathcal{P} \vDash P$ *for some* $x \in S$
$\gamma, D, \mathcal{P} \vDash \forall_{\text{rv}} X{:}A.P$	*iff*	$\gamma, (D, X), \mathcal{P} \vDash P$ *for all* $X : \mathrm{RV}\, A$
$\gamma, D, \mathcal{P} \vDash \exists_{\text{rv}} X{:}A.P$	*iff*	$\gamma, (D, X), \mathcal{P} \vDash P$ *for some* $X : \mathrm{RV}\, A$
$\gamma, D, (\mathcal{F}, \mu) \vDash E \sim \mu'$	*iff*	$E(\gamma) \circ D$ *is* $\mathcal{F}$*-measurable and* $\mu'(\gamma) = \begin{pmatrix} \omega \leftarrow \mu; \\ \mathrm{ret}\ (E(\gamma)(D(\omega))) \end{pmatrix}$
$\gamma, D, (\mathcal{F}, \mu) \vDash \mathsf{own}\, E$	*iff*	$E(\gamma) \circ D$ *is* $\mathcal{F}$*-measurable*
$\gamma, D, (\mathcal{F}, \mu) \vDash E_1 =_{\text{a.s.}} E_2$	*iff*	$(E_1(\gamma) \circ D)\vert_F = (E_2(\gamma) \circ D)\vert_F$ *for some* $F \in \mathcal{F}$ *with* $\mu(F) = 1$
$\gamma, D, (\mathcal{F}, \mu) \vDash \mathbb{E}[E] = e$	*iff*	$E(\gamma) \circ D$ *is* $\mathcal{F}$*-measurable and* $\mathbb{E}_{\omega \sim \mu}[E(\gamma)(D(\omega))] = e(\gamma)$
$\gamma, D, \mathcal{P} \vDash \mathsf{wp}(M, X{:}A.Q)$	*iff*	*for all* $\mathcal{P}_{\text{frame}}$ *and* μ *with* $\mathcal{P}_{\text{frame}} \bullet \mathcal{P} \sqsubseteq (\Sigma_\Omega, \mu)$ *and all* $D_{ext} : \mathrm{RV}\, [\![\Delta_{ext}]\!]$ *there exists* $X : \mathrm{RV}\, A$ *and* $\mathcal{P}'$ *and* μ' *with* $\mathcal{P}_{\text{frame}} \bullet \mathcal{P}' \sqsubseteq (\Sigma_\Omega, \mu')$ *such that* $\begin{pmatrix} \omega \leftarrow \mu; \\ v \leftarrow M(\gamma)(D(\omega)); \\ \mathrm{ret}\ (D_{ext}(\omega), D(\omega), v) \end{pmatrix} = \begin{pmatrix} \omega \leftarrow \mu'; \\ \mathrm{ret}\ (D_{ext}(\omega), D(\omega), X(\omega)) \end{pmatrix}$ *and* $\gamma, (D, X), \mathcal{P}' \vDash Q$

Proof. We must verify that the extended random substitutions (D, X) in the interpretations of $\forall_{\text{rv}}$, $\exists_{\text{rv}}$, and wp have finite footprint; in all cases this follows from the fact that D and X have finite footprint. □

Lemma B.8 (separating conjunction is mutual independence). *Fix a configuration $(\gamma, D, \mathcal{P})$. Abbreviating $X_i(\gamma) \circ D$ as X_i', random variables $X_1', \ldots, X_n'$ are mutually independent with respect to $\mathcal{P}$ iff $\gamma, D, \mathcal{P} \vDash \mathsf{own}\, X_1 * \cdots * \mathsf{own}\, X_n$.*

Proof. First suppose $\gamma, D, \mathcal{P} \vDash \mathsf{own}\, X_1 * \cdots * \mathsf{own}\, X_n$, so each X_i' is $\mathcal{P}_i$-measurable for some $\mathcal{P}_1 \bullet \ldots \bullet \mathcal{P}_n \sqsubseteq \mathcal{P}$. Write $\mathcal{P} = (\mathcal{F}, \mu)$ and $\mathcal{P}_i = (\mathcal{F}_i, \mu_i)$ for all $1 \le i \le n$. For any subset J of $\{1, \ldots, n\}$ and any collection of events $\{E_j \in \mathcal{F}_j\}_{j \in J}$, we have

$$\Pr\left[\bigwedge_{j \in J} X_j' \in E_j\right] = \mu\left(\bigcap_{j \in J} X_j'^{-1}(E_j)\right) \overset{(a)}{=} \prod_{j \in J} \mu_j(X_j'^{-1}(E_j)) \overset{(b)}{=} \prod_{j \in J} \mu(X_j'^{-1}(E_j)) = \prod_{j \in J} \Pr[X_j' \in E_j]$$

where (a) and (b) hold because $\mathcal{P}_1 \bullet \ldots \bullet \mathcal{P}_n \sqsubseteq \mathcal{P}$ and $X_j'^{-1}(E_j) \in \mathcal{F}_j$ for all j. Hence $X_1', \ldots, X_n'$ are mutually independent.

For the converse, suppose $X_1', \ldots, X_n'$ are mutually independent with respect to some probability space $\mathcal{P}$. For each $1 \le i \le n$, let $\mathcal{P}_i$ be the probability space $(\mathcal{F}_i, \mu_i)$ where $\mathcal{F}_i$ is the pullback σ-algebra along X_i' and μ_i the restriction of μ to $\mathcal{F}_i$. It's enough to show that the composition $\mathcal{P}_1 \bullet \ldots \bullet \mathcal{P}_n$ is defined, as then $\mathcal{P}_1 \bullet \ldots \bullet \mathcal{P}_n \sqsubseteq \mathcal{P}$ by lemma 3.3. This follows by induction on n. Cases $n = 0$ and $n = 1$ are immediate. Now suppose $\mathcal{P}_1 \bullet \ldots \bullet \mathcal{P}_k$ is defined. It's straightforward to show that $\mathcal{P}_1 \bullet \ldots \bullet \mathcal{P}_k$ is the pullback of the random variable $(X_1', \ldots, X_k')$, and $\mathcal{P}_{k+1}$ is the pullback of X_{k+1}' by definition. Mutual independence of $X_1', \ldots, X_k', X_{k+1}'$ implies independence of $(X_1', \ldots, X_k')$ and X_{k+1}': intersections of events $X_1'^{-1}(E_1) \cap \cdots \cap X_k'^{-1}(E_k)$ form a π-system that generates $\mathcal{F}_1 \bullet \ldots \bullet \mathcal{F}_k$, events independent of $\mathcal{F}_{k+1}$ with respect to μ form a λ-system, and each intersection $X_1'^{-1}(E_1) \cap \cdots \cap X_k'^{-1}(E_k)$ is independent of $\mathcal{F}_{k+1}$ because $X_1', \ldots, X_k', X_{k+1}'$ are mutually independent. Thus the pullback of $(X_1', \ldots, X_k', X_{k+1}')$ is an independent combination of $\mathcal{P}_1 \bullet \ldots \bullet \mathcal{P}_k$ and $\mathcal{P}_{k+1}$, and $\mathcal{P}_1 \bullet \ldots \bullet \mathcal{P}_k \bullet \mathcal{P}_{k+1}$ is defined. This closes the induction, so $\gamma, D, \mathcal{P} \vDash \mathsf{own}\, X_1' \bullet \ldots \bullet X_n'$ as desired. □

B.4.1 Substitution. Substitutions take the form (s, S) where s is a substitution of pure values and S a substitution of random variables.

$$\frac{s \in [\![\Gamma']\!] \to [\![\Gamma]\!] \qquad S \in [\![\Delta']\!] \xrightarrow{\mathrm{m}} [\![\Delta]\!]}{\Gamma'; \Delta' \vdash (s, S) : \Gamma; \Delta}$$

$$\frac{\Gamma;\Delta \vdash E : A \qquad \Gamma';\Delta' \vdash (s,S) : \Gamma;\Delta}{\Gamma';\Delta' \vdash E[s,S] : A} \qquad E[s,S](\gamma) = E(s(\gamma)) \circ S$$

$$\frac{\Gamma \vdash e : A \qquad \Gamma' \vdash s : \Gamma}{\Gamma' \vdash e[s] : A} \qquad e[s] = e \circ s$$

$$\frac{\Gamma \vdash \mu : A \qquad \Gamma' \vdash s : \Gamma}{\Gamma' \vdash \mu[s] : A} \qquad \mu[s] = \mu \circ s$$

$$\frac{\Gamma;\Delta \vdash M : A \qquad \Gamma';\Delta' \vdash (s,S) : \Gamma;\Delta}{\Gamma';\Delta' \vdash M[s,S] : A} \qquad M[s,S](\gamma) = M(s(\gamma)) \circ S$$

$$\begin{aligned}
\top[s,S] &= \top \\
\bot[s,S] &= \bot \\
(P \wedge Q)[s,S] &= (P[s,S] \wedge Q[s,S]) \\
(P \vee Q)[s,S] &= (P[s,S] \vee Q[s,S]) \\
(P \to Q)[s,S] &= (P[s,S] \to Q[s,S]) \\
(P * Q)[s,S] &= (P[s,S] * Q[s,S]) \\
(P \mathrel{-\!\!*} Q)[s,S] &= (P[s,S] \mathrel{-\!\!*} Q[s,S]) \\
(\Box P)[s,S] &= \Box P[s,S] \\
(\forall x{:}T.P)[s,S] &= \forall x{:}T.P[s \times 1_T, S] \\
(\exists x{:}T.P)[s,S] &= \exists x{:}T.P[s \times 1_T, S] \\
(\forall_{\mathsf{rv}} X{:}A.P)[s,S] &= \forall_{\mathsf{rv}} X{:}A.P[s, S \times 1_A] \\
(\exists_{\mathsf{rv}} X{:}A.P)[s,S] &= \exists_{\mathsf{rv}} X{:}A.P[s, S \times 1_A] \\
(E \sim \mu)[s,S] &= E[s,S] \sim \mu[s] \\
(\mathsf{own}\, E)[s,S] &= \mathsf{own}\, E[s,S] \\
(E_1 =_{\text{a.s.}} E_2)[s,S] &= E_1[s,S] =_{\text{a.s.}} E_2[s,S] \\
(\mathbb{E}[E] = e)[s,S] &= \mathbb{E}[E[s,S]] = e[s] \\
\mathsf{wp}(M, X{:}A.Q)[s,S] &= \mathsf{wp}(M[s,S], X{:}A.Q[s, S \times 1_A])
\end{aligned}$$

$$\frac{\Gamma;\Delta \vdash P \qquad \Gamma';\Delta' \vdash (s,S) : \Gamma;\Delta}{\Gamma';\Delta' \vdash P[s,S]}$$

Lemma B.9 (syntactic and semantic substitution coincide). *$\gamma, D, \mathcal{P} \vDash P[s,S]$ iff $s(\gamma), S \circ D, \mathcal{P} \vDash P$.*

Proof. By induction on the syntax of propositions. The interesting cases are:

- Case $E \sim \mu$:

$$\begin{aligned}
&\gamma, D, \mathcal{P} \vDash (E \sim \mu)[s,S] \\
&\text{iff } \gamma, D, \mathcal{P} \vDash ((\gamma \mapsto E(\gamma) \circ S) \sim (\mu \circ S)) \\
&\text{iff } E(s(\gamma)) \circ S \circ D \text{ is } \mathcal{P}\text{-measurable and } \mu(\gamma) = \left(\begin{array}{l} \omega \leftarrow \mathcal{P} \\ \text{ret } E(\gamma)(S(D(\omega))) \end{array} \right) \\
&\text{iff } s(\gamma), S \circ D, \mathcal{P} \vDash E \sim \mu
\end{aligned}$$

- Case $\mathsf{wp}(M, X{:}A.Q)$: For the left-to-right direction, suppose (1) $\gamma, D, \mathcal{P} \vDash \mathsf{wp}(M[s,S], X{:}A.Q[s, S \times 1_A])$ and (2) $\mathcal{P}_{\text{frame}} \bullet \mathcal{P} \sqsubseteq (\Sigma_\Omega, \mu)$ and (3) $D_{\text{ext}} : \mathsf{RV}\,[\![\Delta_{\text{ext}}]\!]$. By (1) there exist $\mathcal{P}'$ and μ' with $\mathcal{P}_{\text{frame}} \bullet \mathcal{P}' \sqsubseteq$

(Σ_Ω, μ') and $X : \mathrm{RV}\, A$ such that

$$\begin{pmatrix} \omega \leftarrow \mu; \\ v \leftarrow M[s,S](\gamma)(D(\omega)); \\ \mathrm{ret}\ (D_{\mathrm{ext}}, D(\omega), v) \end{pmatrix} = \begin{pmatrix} \omega \leftarrow \mu'; \\ \mathrm{ret}\ (D_{\mathrm{ext}}, D(\omega), X(\omega)) \end{pmatrix}$$

and $\gamma, (D, X), \mathcal{P} \vDash Q[s, S \times 1_A]$. By IH this is equivalent to $s(\gamma), (S \circ D, X), \mathcal{P} \vDash Q$ and simplifying the above equation gives

$$\begin{pmatrix} \omega \leftarrow \mu; \\ v \leftarrow M(s(\gamma))(S(D(\omega))); \\ \mathrm{ret}\ (D_{\mathrm{ext}}, D(\omega), v) \end{pmatrix} = \begin{pmatrix} \omega \leftarrow \mu'; \\ \mathrm{ret}\ (D_{\mathrm{ext}}, D(\omega), X(\omega)) \end{pmatrix}$$

Now postcomposing both sides with the map $(\delta_{\mathrm{ext}}, \delta, v) \mapsto (\delta_{\mathrm{ext}}, S(\delta), v)$ gives

$$\begin{pmatrix} \omega \leftarrow \mu; \\ v \leftarrow M(s(\gamma))(S(D(\omega))); \\ \mathrm{ret}\ (D_{\mathrm{ext}}, S(D(\omega)), v) \end{pmatrix} = \begin{pmatrix} \omega \leftarrow \mu'; \\ \mathrm{ret}\ (D_{\mathrm{ext}}, S(D(\omega)), X(\omega)) \end{pmatrix}$$

so that $(\mathcal{P}', \mu', X)$ witnesses $s(\gamma), S \circ D, \mathcal{P} \vDash \mathrm{wp}(M, X{:}A.Q)$ as desired.
For the right-to-left direction, suppose (1) $s(\gamma), S \circ D, \mathcal{P} \vDash \mathrm{wp}(M, X{:}A.Q)$ and (2) $\mathcal{P}_{\mathrm{frame}} \bullet \mathcal{P} \sqsubseteq (\Sigma_\Omega, \mu)$ and (3) $D_{\mathrm{ext}} : \mathrm{RV}\, [\![\Delta_{\mathrm{ext}}]\!]$. Specialize (1) with $D_{\mathrm{ext}} := (D_{\mathrm{ext}}, D)$ to get $\mathcal{P}'$, μ' and $X : \mathrm{RV}\, A$ such that

$$\begin{pmatrix} \omega \leftarrow \mu; \\ v \leftarrow M(s(\gamma))(S(D(\omega))); \\ \mathrm{ret}\ ((D_{\mathrm{ext}}, D), S(D(\omega)), v) \end{pmatrix} = \begin{pmatrix} \omega \leftarrow \mu'; \\ \mathrm{ret}\ ((D_{\mathrm{ext}}, D), S(D(\omega)), X(\omega)) \end{pmatrix}$$

and $s(\gamma), (S \circ D, X), \mathcal{P}' \vDash Q$. By IH this is equivalent to $\gamma, (D, X), \mathcal{P}' \vDash Q[s, S \times 1_A]$, and postcomposing both sides of the above equation with the map $((\delta_{\mathrm{ext}}, \delta), \_, v) \mapsto (\delta_{\mathrm{ext}}, \delta, v)$ and rewriting $M(s(\gamma))(S(D(\omega)))$ in terms of $M[s, S]$ gives

$$\begin{pmatrix} \omega \leftarrow \mu; \\ v \leftarrow M[s,S](\gamma)(D(\omega)); \\ \mathrm{ret}\ (D_{\mathrm{ext}}, D, v) \end{pmatrix} = \begin{pmatrix} \omega \leftarrow \mu'; \\ \mathrm{ret}\ (D_{\mathrm{ext}}, D, X(\omega)) \end{pmatrix}$$

so that $(\mathcal{P}', \mu', X)$ witnesses $\gamma, D, \mathcal{P} \vDash \mathrm{wp}(M[s, S], X{:}A.Q[s, S \times 1_A])$ as desired.

□

B.5 Derived rules

Lemma B.10. *The following structural rules hold:*

H-Consequence
$$\frac{P \vdash Q}{\mathrm{wp}(M, X.P) \vdash \mathrm{wp}(M, X.Q)}$$

H-Frame
$$F * \mathrm{wp}(M, X.Q) \vdash \mathrm{wp}(M, X.F * Q)\ (X \notin F)$$

H-Disjunction
$$\mathrm{wp}(M, X.P) \vee \mathrm{wp}(M, X.Q) \vdash \mathrm{wp}(M, X.P \vee Q)$$

Proof. We show the proof of the frame rule; the others are standard. Suppose (1) $\gamma, D, \mathcal{P}_F \bullet \mathcal{P}_M \vDash F * \mathrm{wp}(M, X.Q)$ for some $\gamma, D, \mathcal{P}_F \vDash F$ and $\gamma, D, \mathcal{P}_M \vDash \mathrm{wp}(M, X.Q)$. To show $\mathrm{wp}(M, X.F * Q)$, further suppose $\mathcal{P}_{\mathrm{frame}} \bullet (\mathcal{P}_F \bullet \mathcal{P}_M) \sqsubseteq (\Sigma_\Omega, \mu)$ and $D_{\mathrm{ext}} : \mathrm{RV}\, [\![\Delta_{\mathrm{ext}}]\!]$. By associativity, $\mathcal{P}_{\mathrm{frame}} \bullet (\mathcal{P}_F \bullet \mathcal{P}_M) = (\mathcal{P}_{\mathrm{frame}} \bullet \mathcal{P}_F) \bullet \mathcal{P}_M$ so specializing (1) with $\mathcal{P}_{\mathrm{frame}} := \mathcal{P}_{\mathrm{frame}} \bullet \mathcal{P}_F$ gives $(\mathcal{P}_{\mathrm{frame}} \bullet \mathcal{P}_F) \bullet \mathcal{P}' \sqsubseteq (\Sigma_\Omega, \mu')$ and X such that

$$\begin{pmatrix} \omega \leftarrow \mu; \\ v \leftarrow M(\gamma)(D(\omega)); \\ \mathrm{ret}\ (D_{\mathrm{ext}}(\omega), D(\omega), v) \end{pmatrix} = \begin{pmatrix} \omega \leftarrow \mu'; \\ \mathrm{ret}\ (D_{\mathrm{ext}}(\omega), D(\omega), X(\omega)) \end{pmatrix}$$

and $\gamma, (D, X), \mathcal{P}' \vDash Q$. Since $X \notin F, F[\mathrm{weak}_X] = F$ so $\gamma, (D, X), \mathcal{P}_F \vDash F$ by lemma B.9. And since the composition $(\mathcal{P}_{\mathrm{frame}} \bullet \mathcal{P}_F) \bullet \mathcal{P}'$ is defined, the composition $\mathcal{P}_F \bullet \mathcal{P}'$ must be as well, so $\gamma, (D, X), \mathcal{P}_F \bullet \mathcal{P}' \vDash F * Q$ as desired. □

Lemma B.11. *The following* wp *laws hold:*

$$Q[e/X] \vdash \mathrm{wp}(\mathrm{ret}\ e, X.Q) \qquad \mathrm{wp}(M, X.\mathrm{wp}(N, Y.Q)) \vdash \mathrm{wp}\ ((X \leftarrow M; N), Y.Q)$$

$$(\forall_{\mathrm{rv}} X{:}A.\ X \sim \mathrm{Unif}[0,1] \mathrel{-\!\!*} Q) \vdash \mathrm{wp}(\mathrm{Unif}\ [0,1], X{:}A.Q)$$

$$(\forall_{\mathrm{rv}} X{:}A.\ X \sim \mathrm{Ber}\, p \mathrel{-\!\!*} Q) \vdash \mathrm{wp}(\mathrm{Ber}\, p, X{:}A.Q)$$

$$I(1, e) * (\forall i{:}\mathbb{N}.\ \forall_{\mathrm{rv}} X{:}A.\ \{I(i, X)\}\ M\ \{X'.\ I(i+1, X')\}) \vdash \mathrm{wp}(\mathrm{for}(n, e, M), X{:}A.\ I(n+1, X))$$

where $\mathrm{for}(n, e, f)$ *is defined by*

$$\mathrm{for}(n, e, f) = \mathrm{loop}(1, e, f) \textit{ where } \mathrm{loop}(k, e, f) = \begin{cases} \mathrm{ret}\ e, & k > n \\ v \leftarrow f(k, e);\ \mathrm{loop}(k+1, v, f), & \textit{otherwise} \end{cases}$$

Proof.

- Pure: suppose $\gamma, D, \mathcal{P} \vDash Q[e(\gamma)/X]$. By lemma B.9 this is equivalent to $\gamma, (D, e(\gamma)), \mathcal{P} \vDash Q$. To show $\mathrm{wp}(\mathrm{ret}\ e, X.Q)$ suppose $\mathcal{P}_{\mathrm{frame}} \bullet \mathcal{P} \sqsubseteq (\Sigma_\Omega, \mu)$ and $D_{\mathrm{ext}} : \mathrm{RV}[\![\Delta_{\mathrm{ext}}]\!]$. Choose $\mathcal{P}' := \mathcal{P}$ and $\mu' := \mu$ and $X(\omega) := e(\gamma)$. Then

$$\begin{pmatrix} \omega \leftarrow \mu; \\ v \leftarrow (\mathrm{ret}\ e)(\gamma)(D(\omega)); \\ \mathrm{ret}\ (D_{\mathrm{ext}}(\omega), D(\omega), v) \end{pmatrix} = \begin{pmatrix} \omega \leftarrow \mu; \\ \mathrm{ret}\ (D_{\mathrm{ext}}(\omega), D(\omega), e(\gamma)) \end{pmatrix} = \begin{pmatrix} \omega \leftarrow \mu'; \\ \mathrm{ret}\ (D_{\mathrm{ext}}(\omega), D(\omega), X(\omega)) \end{pmatrix}$$

  and $\gamma, (D, e(\Gamma)), \mathcal{P} \vDash Q$ as desired.
- Let: suppose $\gamma, D, \mathcal{P} \vDash \mathrm{wp}(M, X.\mathrm{wp}(N, Y.Q))$. To show $\mathrm{wp}((X \leftarrow M; N), Y.Q)$ suppose $\mathcal{P}_{\mathrm{frame}} \bullet \mathcal{P} \sqsubseteq (\Sigma_\Omega, \mu)$ and $D_{\mathrm{ext}} : \mathrm{RV}[\![\Delta_{\mathrm{ext}}]\!]$. By assumption, there exist $\mathcal{P}_{\mathrm{frame}} \bullet \mathcal{P}_X \sqsubseteq (\Sigma_\Omega, \mu_X)$ and X such that

$$\begin{pmatrix} \omega \leftarrow \mu; \\ x \leftarrow M(\gamma)(D(\omega)); \\ \mathrm{ret}\ (D_{\mathrm{ext}}(\omega), D(\omega), x) \end{pmatrix} = \begin{pmatrix} \omega \leftarrow \mu_X; \\ \mathrm{ret}\ (D_{\mathrm{ext}}(\omega), D(\omega), X(\omega)) \end{pmatrix} \tag{18}$$

  and $\gamma, (D, X), \mathcal{P}_X \vDash \mathrm{wp}(N, Y.Q)$. Applying this assumption gives $\mathcal{P}_{\mathrm{frame}} \bullet \mathcal{P}_Y \sqsubseteq (\Sigma_\Omega, \mu_Y)$ and Y with

$$\begin{pmatrix} \omega \leftarrow \mu_X; \\ y \leftarrow N(\gamma)((D, X)(\omega)); \\ \mathrm{ret}\ (D_{\mathrm{ext}}(\omega), (D, X)(\omega), y) \end{pmatrix} = \begin{pmatrix} \omega \leftarrow \mu_Y; \\ \mathrm{ret}\ (D_{\mathrm{ext}}(\omega), (D, X)(\omega), Y(\omega)) \end{pmatrix} \tag{19}$$

  and $\gamma, (D, X, Y), \mathcal{P}_Y \vDash Q$. Since $X \notin Q$, this implies $\gamma, (D, Y), \mathcal{P}_Y \vDash Q$ by B.9, so it only remains to show

$$\begin{pmatrix} \omega \leftarrow \mu; \\ y \leftarrow (X \leftarrow M; N)(\gamma)(D(\omega)); \\ \mathrm{ret}\ (D_{\mathrm{ext}}(\omega), D(\omega), y) \end{pmatrix} = \begin{pmatrix} \omega \leftarrow \mu_Y; \\ \mathrm{ret}\ (D_{\mathrm{ext}}(\omega), D(\omega), Y(\omega)) \end{pmatrix}$$

Calculate:

$$\left(\begin{array}{l}\omega \leftarrow \mu;\\ y \leftarrow (X \leftarrow M; N)(\gamma)(D(\omega));\\ \text{ret } (D_{\text{ext}}(\omega), D(\omega), v)\end{array}\right) = \left(\begin{array}{l}\omega \leftarrow \mu;\\ x \leftarrow M(\gamma)(D(\omega));\\ y \leftarrow N(\gamma)(D(\omega), x);\\ \text{ret } (D_{\text{ext}}(\omega), D(\omega), y)\end{array}\right)$$

$$= \left(\begin{array}{l}(\delta_{\text{ext}}, \delta, x) \leftarrow\\ \quad\left(\begin{array}{l}\omega \leftarrow \mu;\\ x \leftarrow M(\gamma)(D(\omega));\\ \text{ret } (D_{\text{ext}}(\omega), D(\omega), x)\end{array}\right)\\ y \leftarrow N(\gamma)(\delta, x);\\ \text{ret } (\delta_{\text{ext}}, \delta, y)\end{array}\right) \overset{18}{=} \left(\begin{array}{l}(\delta_{\text{ext}}, \delta, x) \leftarrow\\ \quad\left(\begin{array}{l}\omega \leftarrow \mu_X;\\ \text{ret } (D_{\text{ext}}(\omega), D(\omega), X(\omega))\end{array}\right)\\ y \leftarrow N(\gamma)(\delta, x);\\ \text{ret } (\delta_{\text{ext}}, \delta, y)\end{array}\right)$$

$$= \left(\begin{array}{l}\omega \leftarrow \mu_X;\\ y \leftarrow N(\gamma)(\delta, X(\omega));\\ \text{ret } (D_{\text{ext}}(\omega), D(\omega), y)\end{array}\right) \overset{19}{=} \left(\begin{array}{l}\omega \leftarrow \mu_Y;\\ \text{ret } (D_{\text{ext}}(\omega), D(\omega), Y(\omega))\end{array}\right)$$

- Uniform: suppose (1) $\gamma, D, \mathcal{P} \vDash \forall_{\text{rv}} X{:}A.\ X \sim \text{Unif}[0,1] \mathrel{-\!\!*} Q$. To show $\text{wp}(\text{Unif } [0,1], X{:}A.Q)$, suppose $\mathcal{P}_{\text{frame}} \bullet \mathcal{P} \sqsubseteq (\Sigma_\Omega, \mu)$ and $D_{\text{ext}} : \text{RV}[\![\Delta_{\text{ext}}]\!]$. Let n witness $(D_{\text{ext}}, \mathcal{P}_{\text{frame}} \bullet \mathcal{P})$'s finite footprint. Write the Hilbert cube as $[0,1]^{\mathbb{N}} \cong [0,1]^n \otimes [0,1]^{\mathbb{N}}$. Define μ' via this isomorphism as the product measure $\mu|_{[0,1]^n} \otimes \lambda$, where λ assigns to each finite-dimensional box $\prod_{i=1}^n [a_i, b_i] \times [0,1]^{\mathbb{N}}$ the measure $\prod_{i=1}^n |b_i - a_i|$ and extends to a measure on the whole Hilbert cube by the Carathéodory extension theorem. Let $\mathcal{P}_n$ be the restriction of μ' to measurable sets of the form $[0,1]^n \times F \times [0,1]^{\mathbb{N}}$. Let X be the projection $\pi_{n+1} = (\ldots, \omega_{n+1}, \ldots) \mapsto \omega_{n+1}$. By construction, the composite $\mathcal{P}_{\text{frame}} \bullet \mathcal{P} \bullet \mathcal{P}_n$ is defined and $\mathcal{P}_{\text{frame}} \bullet \mathcal{P} \bullet \mathcal{P}_n \sqsubseteq (\Sigma_\Omega, \mu')$ and X is $\mathcal{P}_n$-measurable and uniformly distributed in $[0,1]$. Therefore $\gamma, (D, X), \mathcal{P} \bullet \mathcal{P}_n \vDash Q$ by (1), and it only remains to show

$$\left(\begin{array}{l}\omega \leftarrow \mu;\\ v \leftarrow \text{Unif } [0,1];\\ \text{ret } (D_{\text{ext}}(\omega), D(\omega), v)\end{array}\right) = \left(\begin{array}{l}\omega \leftarrow \mu';\\ \text{ret } (D_{\text{ext}}(\omega), D(\omega), X(\omega))\end{array}\right)$$

Calculate:

$$\left(\begin{array}{l}\omega \leftarrow \mu;\\ v \leftarrow \text{Unif } [0,1];\\ \text{ret } (D_{\text{ext}}(\omega), D(\omega), v)\end{array}\right) = \left(\begin{array}{l}\omega_{1\cdots n} \leftarrow \mu|_{[0,1]^n};\\ v \leftarrow \text{Unif } [0,1];\\ \text{ret } (D_{\text{ext}}(\omega_{1\cdots n}), D(\omega_{1\cdots n}), v)\end{array}\right)$$

$$= \left(\begin{array}{l}(\omega_{1\cdots n}, v) \leftarrow \mu|_{[0,1]^n} \otimes \text{Unif } [0,1];\\ \text{ret } (D_{\text{ext}}(\omega_{1\cdots n}), D(\omega_{1\cdots n}), v)\end{array}\right) = \left(\begin{array}{l}\omega_{1\cdots n+1} \leftarrow \mu'|_{[0,1]^{n+1}};\\ \text{ret } (D_{\text{ext}}(\omega_{1\cdots n}), D(\omega_{1\cdots n}), \omega_{n+1})\end{array}\right)$$

$$= \left(\begin{array}{l}\omega \leftarrow \mu';\\ \text{ret } (D_{\text{ext}}(\omega), D(\omega), X(\omega))\end{array}\right)$$

- Flip: analogous to Uniform.
- For: suppose (1) $\forall i{:}\mathbb{N}.\ \forall_{\text{rv}} X{:}\text{RV}\, A.\ \{I(i, X)\}\ M\ \{X'.\, I(i+1, X')\}$. We need to show

$$I(1, e) \mathrel{-\!\!*} \text{wp}(\text{for}(n, e, M), X{:}A.\ I(n+1, X)).$$

We generalize, and show

$$I(n+1-k, V) \mathrel{-\!\!*} \text{wp}(\text{loop}(n+1-k, V, M), X'.\, I(n+1, X'))$$

for all V and all $0 \le k \le n$ by induction on k, from which this follows at $k = n$.

– Case $k = 0$:

$$\begin{aligned}\top &\vdash I(n+1,V) \mathrel{-\!\!*} I(n+1,V)\\ &\vdash I(n+1,V) \mathrel{-\!\!*} \mathrm{wp}(\mathrm{ret}\ V, X'.\ I(n+1,X'))\\ &\vdash I(n+1,V) \mathrel{-\!\!*} \mathrm{wp}(\mathrm{loop}(n+1,V,M), X'.\ I(n+1,X'))\\ &\vdash I(n+1-k,V) \mathrel{-\!\!*} \mathrm{wp}(\mathrm{loop}(n+1-k,V,M), X'.\ I(n+1,X'))\end{aligned}$$

– Case $k = j+1 \le n$: backwards reasoning from the goal gives

$$\begin{aligned}&I(n+1-(j+1),V) \mathrel{-\!\!*} \mathrm{wp}(\mathrm{loop}(n+1-(j+1),V,M), X'.\ I(n+1,X'))\\ &\dashv I(n-j,V) \mathrel{-\!\!*} \mathrm{wp}(\mathrm{loop}(n-j,V,M), X'.\ I(n+1,X'))\\ &\dashv I(n-j,V) \mathrel{-\!\!*} \mathrm{wp}((V' \leftarrow M[n-j/i, V/X];\ \mathrm{loop}(n+1-j,V',M)), X'.\ I(n+1,X'))\\ &\dashv I(n-j,V) \mathrel{-\!\!*} \mathrm{wp}(M[n-j/i,V/X], V'.\ \mathrm{wp}(\mathrm{loop}(n+1-j,V',M), X'.\ I(n+1,X')))\end{aligned}$$

Now $I(n-j,V) \mathrel{-\!\!*} \mathrm{wp}(M[n-j/i,V/X], V'.\ I(n-j+1,V'))$ by (1) so it suffices to show

$$\begin{aligned}&\mathrm{wp}(M[n-j/i,V/X], V'.\ I(n+1-j,V'))\\ &\mathrel{-\!\!*} \mathrm{wp}(M[n-j/i,V/X], V'.\ \mathrm{wp}(\mathrm{loop}(n+1-j,V',M), X'.\ I(n+1,X'))).\end{aligned}$$

The outer wps are the same, so by the consequence rule it suffices to show

$$I(n+1-j,V') \mathrel{-\!\!*} \mathrm{wp}(\mathrm{loop}(n+1-j,V',M), X'.\ I(n+1,X'))$$

for all V', which is exactly the induction hypothesis at j.

□

Corollary B.12. *The following* wp *laws hold:*

W-pure
$Q[[\![M]\!]/X] \vdash \mathrm{wp}([\![\mathtt{ret}\ M]\!], X.Q)$

W-Let
$\mathrm{wp}([\![M]\!], X.\mathrm{wp}([\![N]\!], Y.Q)) \vdash \mathrm{wp}\ ([\![\mathtt{let}\ X = M\ \mathtt{in}\ N]\!], Y.Q)$

W-Uniform
$(\forall_{\mathrm{rv}} X{:}A.\ X \sim \mathrm{Unif}[0,1] \mathrel{-\!\!*} Q) \vdash \mathrm{wp}([\![\mathtt{uniform}]\!], X{:}A.Q)$

W-Flip
$(\forall_{\mathrm{rv}} X{:}A.\ X \sim \mathrm{Ber}\,p \mathrel{-\!\!*} Q) \vdash \mathrm{wp}([\![\mathtt{flip}\ p]\!], X{:}A.Q)$

W-For
$I(1,e) * (\forall i{:}\mathbb{N}.\ \forall_{\mathrm{rv}} X{:}A.\ \{I(i,X)\}\ M\ \{X'.\ I(i+1,X')\}) \vdash \mathrm{wp}([\![\mathtt{for}(n,e,i\ X.\ M)]\!], X{:}A.\ I(n+1,X))$

Proof. Unfold $[\![-]\!]$ and apply lemma B.11. □

B.6 Disintegration

Lemma B.13. *Let $(M, \bullet, \sqsubseteq, 1)$ be a KRM with $1 \sqsubseteq x$ for all x. Let A be a downward-closed subset of M (i.e., $x \sqsubseteq y \in A$ implies $x \in A$). Let $(\bullet')$ be the restriction of $(\bullet)$ to A; that is,*

$$x \bullet' y := \begin{cases} x \bullet y, & x \bullet y \in A \\ \text{undefined}, & \text{otherwise} \end{cases}$$

Then $(A, \bullet', \sqsubseteq, 1)$ is a KRM.

Proof. Unit and commutativity are straightforward. For associativity, note that if $1 \sqsubseteq x$ for all x then by monotonicity of $(\bullet)$ we have $x \sqsubseteq x \bullet y$ for all x, y. Now suppose $x \bullet' y$ defined and $(x \bullet' y) \bullet' z$ defined. Then $(x \bullet y) \bullet z \in A$ and by downward closure so is $y \bullet z$ and by associativity so is $x \bullet (y \bullet z)$, so both $y \bullet' z$ and $x \bullet' (y \bullet' z)$ are defined and associativity is inherited from associativity of $(\bullet)$. □

Theorem B.14. *Let $\mathcal{M}_{\mathrm{disintegrable}}$ be the set of countably-generated probability spaces $\mathcal{P}$ that have finite footprint and can be extended to a Borel measure on the entire Hilbert cube. The restriction of the KRM given by Theorem 3.4 to $\mathcal{M}_{\mathrm{disintegrable}}$ is still a KRM.*

Proof. By Lemma B.6, the restriction to $\mathcal{M}_{\text{finite}}$ is a KRM, so it suffices to show that restricting to countably-generated spaces that can be extended to a Borel measure still yields a KRM. First, restricting to countably-generated spaces still yields a KRM because the independent combination of two countably-generated spaces remains couuntably-generated. Then, the restriction to spaces that can be extended to a Borel measure still yields a KRM by Lemma B.13, as the set of spaces that can be extended to a Borel measure is downward-closed. □

$$\frac{\Gamma; \Delta \vdash E : A \qquad \Gamma, x{:}A; \Delta \vdash P}{\Gamma; \Delta \vdash \mathop{\mathsf{D}}_{x:A \leftarrow E} P}$$

$$\begin{aligned} \gamma, D, \mathcal{P} \vDash \mathsf{D}_{x:A \leftarrow E}\, P \quad \text{iff} \quad & \text{for all } (\Sigma_\Omega, \mu) \sqsupseteq \mathcal{P} \\ & \text{and all } (\Sigma_\Omega, \mu)\text{-disintegrations } \{\nu_x\}_{x \in A} \text{ with respect to } E \circ D \\ & \text{and } (E \circ D)_* \mu\text{-almost-all } x \in A, \\ & (\gamma, x), D, \nu_x|_{\mathcal{P}} \vDash P \end{aligned}$$

Lemma B.15. *If μ and ν are probability measures on a space $(\Omega, \mathcal{F})$ generated by a π-system $\mathcal{B}$, then $\mu = \nu$ iff $\mu(B) = \nu(B)$ for all $B \in \mathcal{B}$.*

Proof. The left-to-right direction is straightforward. The right-to-left direction follows from the π-λ theorem: the set $S := \{B \in \mathcal{B} \mid \mu(B) = \nu(B)\}$ is a λ-system and μ and ν agree on a π-system that generates $\mathcal{F}$ by assumption. □

Lemma B.16. $\mathsf{own}\, E * P \vdash \mathop{\mathsf{D}}_{x:A \leftarrow E} P$.

Proof. Suppose $\gamma, D, \underbrace{\mathcal{P}_E \bullet \mathcal{P}_P}_{\mathcal{P}} \vDash \mathsf{own}\, E * P$ and $\gamma, D, \mathcal{P}_E \vDash \mathsf{own}\, E$ and (1) $\gamma, D, \mathcal{P}_P \vDash P$. To show $\mathsf{D}_{x \leftarrow E}\, P$, suppose $\mathcal{P}_E \bullet \mathcal{P}_P \sqsubseteq (\Sigma_\Omega, \mu)$ and let $\{\nu_x\}_{x \in A}$ be a μ-disintegration with respect to $E(\gamma) \circ D$. We need to show $(\gamma, x), D, \nu_x|_{\mathcal{P}} \vDash P$ for almost all $x \in A$. Since $x \notin P$, $(\gamma, x), D, \nu_x|_{\mathcal{P}} \vDash P$ is equivalent to $\gamma, D, \nu_x|_{\mathcal{P}} \vDash P$, so by assumption (1) and monotonicity it suffices to show $\nu_x|_{\mathcal{P}_P} = \mathcal{P}_P$ for almost all $x \in A$.

Write $\mathcal{P}_P = (\mathcal{F}_P, \mu_P)$ and let $S := \{x \in A \mid \nu_x|_{\mathcal{F}_P} = \mu_P\}$. It's enough to show that S is Σ_A-measurable and has probability 1. Let $\mathcal{B} = \{B_n\}_{n \in \mathbb{N}}$ be a countable basis of $\mathcal{F}_P$; without loss of generality we may assume $\mathcal{B}$ is a π-system because any countable collection of sets has countable closure under finite intersections. By lemma B.15, we can write S as the countable intersection $S = \bigcap_{n \in \mathbb{N}} S_n$ where $S_n := \{x \in A \mid \nu_x(B_n) = \mu_P(B_n)\}$. Because σ-algebras are closed under countable intersections and measures are countably subadditive, S is measurable with probability 1 if each S_n is.

Each S_n is Σ_A-measurable: S_n is equal to the preimage of the singleton set $\{\mu_P(B_n)\}$ under the map $\nu_{(-)}(B_n)$; since ν is a Markov kernel and singletons are Borel, this preimage must be Σ_A-measurable. It only remains to show each S_n has probability 1. Now, suppose for the sake of contradiction that there is some k such that S_k does not have probability 1, so $\nu_x(B_n) \neq \mu_P(B_n)$ for all $x \in N := A \setminus S_k$. We can write N as a disjoint union of two subsets $N_<$ and $N_>$, defined as follows:

$$\begin{aligned} N_< &:= \{x \in N \mid \nu_x(B_n) < \mu_P(B_n)\} \\ N_> &:= \{x \in N \mid \nu_x(B_n) > \mu_P(B_n)\} \end{aligned}$$

These are both measurable, since they can be written as preimages of $[0, \mu_P(B_n))$ and $(\mu_P(B_n), 1]$ under $\nu_{(-)}(B_n)$. Because N has nonzero probability, at least one of $N_<$ or $N_>$ must have nonzero probability too. Suppose it's $N_<$; the case where $N_>$ has nonzero probability is analogous. Because ν is a disintegration of μ with respect to $E(\gamma) \circ D$, we have

$$\mathop{\mathbb{E}}_{\omega \sim \mu} f(\omega) = \mathop{\mathbb{E}}_{x \sim (E(\gamma) \circ D)_* \mu} \mathop{\mathbb{E}}_{\omega \sim \nu_x} f(\omega)$$

for all $f : \Omega \xrightarrow{\text{m}} \mathbb{R}_{\geq 0}$. Choose $f(\omega) := \mathbb{1}[\omega \in B_n]\mathbb{1}[E(\gamma)(D(\omega)) \in N_<]$. Then simplifying LHS gives

$$\begin{aligned}\mathop{\mathbb{E}}_{\omega\sim\mu} f(\omega) &= \mathop{\mathbb{E}}_{\omega\sim\mu} \mathbb{1}[\omega \in B_n]\mathbb{1}[E(\gamma)(D(\omega)) \in N_<] \\ &\overset{(a)}{=} \mathop{\mathbb{E}}_{\omega\sim\mu} \mathbb{1}[\omega \in B_n] \mathop{\mathbb{E}}_{\omega\sim\mu} \mathbb{1}[E(\gamma)(D(\omega)) \in N_<] \\ &= \mu(B_n)(E(\gamma)\circ D)_*\mu(N_<)\end{aligned}$$

Step (a) uses independence of B_n and $(E(\gamma)\circ D)^{-1}(N_<)$: $\mathcal{P}_E \bullet \mathcal{P}_P$ defined and $B_n \in \mathcal{P}_P$ and $(E(\gamma)\circ D)^{-1}(N_<) \in \mathcal{P}_E$. Meanwhile, simplifying RHS gives

$$\begin{aligned}\mathop{\mathbb{E}}_{x\sim(E(\gamma)\circ D)_*\mu} \mathop{\mathbb{E}}_{\omega\sim\nu_x} f(\omega) &= \mathop{\mathbb{E}}_{x\sim(E(\gamma)\circ D)_*\mu} \mathop{\mathbb{E}}_{\omega\sim\nu_x} \mathbb{1}[\omega \in B_n]\mathbb{1}[E(\gamma)(D(\omega)) \in N_<] \\ &\overset{(a)}{=} \mathop{\mathbb{E}}_{x\sim(E(\gamma)\circ D)_*\mu} \mathop{\mathbb{E}}_{\omega\sim\nu_x} \mathbb{1}[\omega \in B_n]\mathbb{1}[x \in N_<] \\ &= \mathop{\mathbb{E}}_{x\sim(E(\gamma)\circ D)_*\mu} \mathbb{1}[x \in N_<] \mathop{\mathbb{E}}_{\omega\sim\nu_x} \mathbb{1}[\omega \in B_n] \\ &= \mathop{\mathbb{E}}_{x\sim(E(\gamma)\circ D)_*\mu} \mathbb{1}[x \in N_<]\nu_x(B_n) \\ &\overset{(b)}{<} \mathop{\mathbb{E}}_{x\sim(E(\gamma)\circ D)_*\mu} \mathbb{1}[x \in N_<]\mu_P(B_n) \\ &= \mu_P(B_n) \mathop{\mathbb{E}}_{x\sim(E(\gamma)\circ D)_*\mu} \mathbb{1}[x \in N_<] \\ &= \mu_P(B_n)(E(\gamma)\circ D)_*\mu(N_<) \\ &= \mu(B_n)(E(\gamma)\circ D)_*\mu(N_<)\end{aligned}$$

Step (a) holds because $(E(\gamma)\circ D)_*\nu_x(\{x\}) = 1$ for almost all x. Step (b) holds because the expectation is taken over $x \in N_<$, where the inequality holds by assumption; the inequality remains strict because $N_<$ is nonnegligible. Putting these two together gives LHS = RHS and LHS < RHS, a contradiction. □

Lemma B.17 (law of total expectation). *The following entailment holds:*

$$\mathbb{E}[e[X/x]] = v \;\wedge\; \mathop{\mathsf{D}}_{x:A\leftarrow X} \mathbb{E}[E] = e \;\vdash\; \mathbb{E}[E] = v$$

Proof. Fix $(\gamma, D, \mathcal{P})$. By the first conjunct $\mathbb{E}_{\omega\sim\mathcal{P}}\, e(\gamma, X(\gamma)(D(\omega))) = v(\gamma)$. By assumption, $\mathcal{P}$ extends to a Borel measure μ on the Hilbert cube. By the disintegration theorem, there exists at least one μ-disintegration with respect to $X(\gamma)\circ D$; call it $\{\nu_x\}_{x\in A}$. By the second conjunct $\mathbb{E}_{\omega\sim\nu_x}[E(\gamma)(D(\omega))] = e(\gamma, x)$ for almost all $x \in A$. This along with the existence of the disintegration ν implies

$$\mathop{\mathbb{E}}_{\omega\sim\mathcal{P}} E(\gamma)(D(\omega)) = \mathop{\mathbb{E}}_{x\sim(X(\gamma)\circ D)_*\mu} \mathop{\mathbb{E}}_{\omega\sim\nu_x} E(\gamma)(D(\omega)) = \mathop{\mathbb{E}}_{x\sim(X(\gamma)\circ D)_*\mu} e(\gamma, x) = \mathop{\mathbb{E}}_{\omega\in\mu} e(\gamma, X(\gamma)(D(\omega))) = v(\gamma)$$

as desired. □

Lemma B.18. *The following entailments hold:*

D-Entail
$$\frac{P \;\vdash\; Q}{\mathop{\mathsf{D}}_{x\leftarrow E} P \;\vdash\; \mathop{\mathsf{D}}_{x\leftarrow E} Q}$$

D-And
$$\mathop{\mathsf{D}}_{x\leftarrow E}(P \wedge Q) \;\dashv\vdash\; \mathop{\mathsf{D}}_{x\leftarrow E} P \;\wedge\; \mathop{\mathsf{D}}_{x\leftarrow E} Q$$

D-Indep
$$\mathsf{own}\, E * P \;\vdash\; \mathop{\mathsf{D}}_{x\leftarrow E} P$$

D-Subst
$$\mathop{\mathsf{D}}_{x\leftarrow X} \mathbb{E}[E[x/X]] = e \;\vdash\; \mathop{\mathsf{D}}_{x\leftarrow X} \mathbb{E}[E] = e$$

D-Total-Expectation
$$\mathop{\mathsf{D}}_{x\leftarrow X} \mathbb{E}[E] = e \;\wedge\; \mathbb{E}[e[X/x]] = v \;\vdash\; \mathbb{E}[E] = v$$

Proof. D-Indep and D-Total-Expectation follow from lemmas B.16 and B.17 respectively.

- D-Entail: suppose $P \vdash Q$ and $\gamma, D, \mathcal{P} \vDash \mathsf{D}_{x:A\leftarrow E} P$. Let $\{\mu_x\}_{x\in A}$ be a disintegration of $\mathcal{P}$ with respect to $E(\gamma)\circ D$; let $\{\mathcal{P}_x\}_{x\in A}$ be the corresponding restrictions of $\{\mu_x\}_{x\in A}$ to $\mathcal{P}$. By assumption, $\gamma, D, \mathcal{P}_x \vDash P$ for almost-all $x \in A$. Since $P \vdash Q$, this implies $\gamma, D, \mathcal{P}_x \vDash Q$ for almost-all $x \in A$ as desired.

- D-And: the left-to-right direction follows from D-Entail via the entailments $P \wedge Q \vdash P$ and $P \wedge Q \vdash Q$. For the right-to-left entailment, suppose $\gamma, D, \mathcal{P} \vDash \mathsf{D}_{x \leftarrow E}\, P$ and $\gamma, D, \mathcal{P} \vDash \mathsf{D}_{x \leftarrow E}\, Q$ and let $\{\mu_x\}_{x \in A}$ be a disintegration of $\mathcal{P}$ with respect to $E(\gamma) \circ D$; let $\{\mathcal{P}_x\}_{x \in A}$ be the corresponding restrictions of $\{\mu_x\}_{x \in A}$ to $\mathcal{P}$. By assumption, there are two sets $F_1, F_2 \subseteq A$ of measure 1 such that $\gamma, D, \mathcal{P}_x \vDash P$ for all $x \in F_1$ and $\gamma, D, \mathcal{P}_x \vDash Q$ for all $x \in F_2$. Therefore, $\gamma, D, \mathcal{P}_x \vDash P \wedge Q$ for all $x \in F_1 \cap F_2$. Moreover, $F_1 \cap F_2$ has measure 1 by subadditivity, so $\gamma, D, \mathcal{P}_x \vDash P \wedge Q$ for almost-all x as desired.
- D-Subst: Suppose $\gamma, D, \mathcal{P} \vDash \mathsf{D}_{x \leftarrow X}\, \mathbb{E}[E[x/X]]$ and let $\{\mu_x\}_{x \in A}$ be a disintegration of $\mathcal{P}$ with respect to $X \circ D$; let $\{\mathcal{P}_x\}_{x \in A}$ be the corresponding restrictions of $\{\mu_x\}_{x \in A}$ to $\mathcal{P}$. By assumption, $\mathbb{E}_{\omega \sim \mu_x}\, E(\gamma)(D(\omega)[X \mapsto x]) = e(\gamma)$ for almost-all $x \in A$. Since $\mu_x((X \circ D)^{-1}(x)) = 1$ for almost-all $x \in A$, this is equivalent to $\mathbb{E}_{\omega \sim \mu_x}\, E(\gamma)(D(\omega)) = e(\gamma)$ for almost-all $x \in A$, so $\gamma, D, \mathcal{P}_x \vDash \mathbb{E}[E] = e$ for almost-all $x \in A$ as desired.

□

Lemma B.19. *The following entailments hold:*

- *Necessitation: if* $\vdash P$ *then* $\vdash \mathop{\mathsf{D}}_{x \leftarrow X} P$.
- *Distribution:* $\mathop{\mathsf{D}}_{x \leftarrow X} (P \to Q) \vdash \mathop{\mathsf{D}}_{x \leftarrow X} P \to \mathop{\mathsf{D}}_{x \leftarrow X} Q$.
- *Distributes over* $(\wedge)$*:* $\mathop{\mathsf{D}}_{x \leftarrow X} (P \wedge Q) \dashv\vdash \mathop{\mathsf{D}}_{x \leftarrow X} P \wedge \mathop{\mathsf{D}}_{x \leftarrow X} Q$.
- *Semidistributes over* $(\vee)$*:* $\mathop{\mathsf{D}}_{x \leftarrow X} P \vee \mathop{\mathsf{D}}_{x \leftarrow X} Q \vdash \mathop{\mathsf{D}}_{x \leftarrow X} (P \vee Q)$.

Proof.

- Necessitation: if P holds in all configurations then it holds for all disintegrated configurations as well.
- Distribution: it suffices to show $D_{x \leftarrow X}(P \to Q) \wedge \mathsf{D}_{x \leftarrow X}\, P \vdash \mathsf{D}_{x \leftarrow X}\, Q$. By D-And the premise is equivalent to $\mathsf{D}_{x \leftarrow X}((P \to Q) \wedge P)$; the result then follows from D-Entail via the entailment $(P \to Q) \wedge P \vdash Q$.
- Distributes over $(\wedge)$: this is D-And.
- Semidistributes over $(\vee)$: it suffices to show $\mathsf{D}_{x \leftarrow X}\, P \vdash \mathsf{D}_{x \leftarrow X}(P \vee Q)$ and $\mathsf{D}_{x \leftarrow X}\, Q \vdash \mathsf{D}_{x \leftarrow X}(P \vee Q)$. These follow from D-Entail via the entailments $P \vdash P \vee Q$ and $Q \vdash P \vee Q$ respectively.

□

C VERIFICATION OF CONSTANT-SPACE WEIGHTED SAMPLING

$\{\top\}$

  $\texttt{let}\ w = [w_1, \ldots, w_n]\ \texttt{in}$

$\{w =_{\text{a.s.}} (w_1, \ldots, w_n)\}$

  $\texttt{let}\ M = -\infty\ \texttt{in}$

$\{w =_{\text{a.s.}} (w_1, \ldots, w_n) \ast M =_{\text{a.s.}} -\infty\}$

  $\texttt{let}\ K = 0\ \texttt{in}$

$\{w =_{\text{a.s.}} (w_1, \ldots, w_n) \ast M =_{\text{a.s.}} -\infty \ast K =_{\text{a.s.}} 0\}$

$$\text{Let } I(i, M, K) \quad := \quad \begin{pmatrix} w =_{\text{a.s.}} (w_1, \ldots, w_n) \ast 1 \le i \le n \ast \\ \exists S_1 \ldots S_i.\ \mathop{\Large\ast}_{1 \le j < i} S_j \sim \text{Unif}\ [0,1] \ast K = \arg\max_{1 \le j < i} S_j^{1/w_j} \ast M = \max_{1 \le j < i} S_j^{1/w_j} \end{pmatrix}$$

$\{I(1, M, K)\}$

  $\texttt{for}(n, (M, K), i\ (M, K).$

  $\{I(i, M, K)\}$

    $\texttt{let}\ S = \texttt{uniform}\ \texttt{in}$

  $\{I(i, M, K) \ast S \sim \text{Unif}\ [0,1]\}$

    $\texttt{let}\ U = S\ \texttt{\^{}}\ (1/w[i])\ \texttt{in}$

  $\{I(i, M, K) \ast S \sim \text{Unif}\ [0,1] \ast U =_{\text{a.s.}} S^{1/w_i}\}$

    $\texttt{if}\ U > M$

    $\texttt{then ret}\ (U, i)$

    $\texttt{else ret}\ (M, K))$

  $\{(M', K').\ I(i, M, K) \ast S \sim \text{Unif}\ [0,1] \ast U =_{\text{a.s.}} S^{1/w_i} \ast M' =_{\text{a.s.}} \max(U, M) \ast U' =_{\text{a.s.}} \text{if } U > M \text{ then } i \text{ else } K\}$

  $\{(M', K').\ I(i+1, M', K')\}$

$\{(M', K').\ I(n+1, M', K')\}$

$\{\exists_{\text{rv}} S_1 \ldots S_n.\ \mathop{\Large\ast}_i S_i \sim \text{Unif}\ [0,1] \ast K = \arg\max_i S_i^{1/w_i}\}$

$\{\forall i.\ \Pr(K = i) = \dfrac{w_i}{\sum_j w_j}\}$

To illustrate the proof of the final entailment, we animate the proof state at each step in inference-rule notation, in the style of interactive theorem provers such as Coq. First we work backwards from the goal:

$$\frac{\circledast_i S_i \sim \text{Unif}\ [0,1] \qquad K = \arg\max_i S_i^{1/w_i}}{\Pr[K = k] = \frac{w_k}{\sum_j w_j}}$$

$$\frac{\circledast_i S_i \sim \text{Unif}\ [0,1] \qquad K = \arg\max_i S_i^{1/w_i}}{\Pr[\forall j \neq k.\ S_k^{1/w_k} > S_j^{1/w_j}] = \frac{w_k}{\sum_j w_j}}$$

$$\frac{\circledast_i S_i \sim \text{Unif}\ [0,1] \qquad K = \arg\max_i S_i^{1/w_i}}{\Pr[\forall j \neq k.\ S_k^{w_j/w_k} > S_j] = \frac{w_k}{\sum_j w_j}}$$

$$\frac{\circledast_i S_i \sim \text{Unif}\ [0,1] \qquad K = \arg\max_i S_i^{1/w_i}}{\mathbb{E}\left[\mathbb{1}[\forall j \neq k.\ S_k^{w_j/w_k} > S_j]\right] = \frac{w_k}{\sum_j w_j}}$$

$$\frac{\circledast_i S_i \sim \text{Unif}\ [0,1] \qquad K = \arg\max_i S_i^{1/w_i}}{\mathbb{E}\left[\prod_{j\neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j]\right] = \frac{w_k}{\sum_j w_j}}$$

At this point we begin working forwards from the hypotheses, using D-Indep to introduce the disintegration modality $\mathsf{D}_{s_k \leftarrow S_k}$ with the aim of computing the conditional probability $\Pr(K = k \mid S_k = s_k)$.

$$\frac{S_k \sim \text{Unif}\ [0,1] \wedge \mathop{\mathsf{D}}_{s_k \leftarrow S_k} \circledast_{j\neq k} S_j \sim \text{Unif}\ [0,1] \qquad K = \arg\max_i S_i^{1/w_i}}{\mathbb{E}\left[\prod_{j\neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j]\right] = \frac{w_k}{\sum_j w_j}}$$

$$\frac{S_k \sim \text{Unif}\ [0,1] \wedge \mathop{\mathsf{D}}_{s_k \leftarrow S_k} \circledast_{j\neq k} S_j \sim \text{Unif}\ [0,1] \wedge \left(\mathbb{E}\left[\prod_{j\neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j]\right] = \mathbb{E}\left[\prod_{j\neq k} \mathbb{1}[s_k^{w_j/w_k} > S_j]\right]\right) \quad K = \arg\max_i S_i^{1/w_i}}{\mathbb{E}\left[\prod_{j\neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j]\right] = \frac{w_k}{\sum_j w_j}}$$

Next, we use conditional independence of $\{S_j\}_{j\neq k}$ given S_k, encoded in the iterated separating conjunction underneath $\mathsf{D}_{s_k \leftarrow S_k}$, to interchange product and expectation:

$$\frac{S_k \sim \mathrm{Unif}\ [0,1] \land \mathop{\mathsf{D}}_{s_k \leftarrow S_k} \mathop{\Large *}_{j\neq k} S_j \sim \mathrm{Unif}\ [0,1] \land \left(\mathbb{E}\left[\prod_{j\neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j]\right] = \prod_{j\neq k} \mathbb{E}[\mathbb{1}[s_k^{w_j/w_k} > S_j]]\right) \quad K = \arg\max_i S_i^{1/w_i}}{\mathbb{E}\left[\prod_{j\neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j]\right] = \frac{w_k}{\sum_j w_j}}$$

Now significant simplifications are possible, completing the first calculation (Equations 6–10) of Section 2.1:

$$\frac{S_i \sim \mathrm{Unif}\ [0,1] \land \mathop{\mathsf{D}}_{s_k \leftarrow S_k} \mathop{\Large *}_{j\neq k} S_j \sim \mathrm{Unif}\ [0,1] \land \left(\mathbb{E}\left[\prod_{j\neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j]\right] = \prod_{j\neq k} s_k^{w_j/w_k}\right) \quad K = \arg\max_i S_i^{1/w_i}}{\mathbb{E}\left[\prod_{j\neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j]\right] = \frac{w_k}{\sum_j w_j}}$$

$$\frac{S_k \sim \mathrm{Unif}\ [0,1] \land \mathop{\mathsf{D}}_{s_k \leftarrow S_k} \mathop{\Large *}_{j\neq k} S_j \sim \mathrm{Unif}\ [0,1] \land \left(\mathbb{E}\left[\prod_{j\neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j]\right] = \exp\left(s_k, \frac{\sum_{j\neq k} w_j}{w_k}\right)\right) \quad K = \arg\max_i S_i^{1/w_i}}{\mathbb{E}\left[\prod_{j\neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j]\right] = \frac{w_k}{\sum_j w_j}}$$

$$\frac{S_k \sim \mathrm{Unif}\ [0,1] \land \mathop{\mathsf{D}}_{s_k \leftarrow S_k} \mathbb{E}\left[\prod_{j\neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j]\right] = \exp\left(s_k, \frac{\sum_{j\neq k} w_j}{w_k}\right)}{\mathbb{E}\left[\prod_{j\neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j]\right] = \frac{w_k}{\sum_j w_j}}$$

Having completed the computation of the conditional probability $\Pr(K = k \mid S_k = s_k)$ by working forwards from the hypotheses, we eliminate the disintegration modality by applying the law of total expectation (D-Total-Expectation):

$$\frac{S_k \sim \mathrm{Unif}\ [0,1] \land \mathbb{E}\left[\prod_{j\neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j]\right] = \mathbb{E}\left[\exp\left(S_k, \frac{\sum_{j\neq k} w_j}{w_k}\right)\right]}{\mathbb{E}\left[\prod_{j\neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j]\right] = \frac{w_k}{\sum_j w_j}}$$

The remainder of the calculation is straightforward, following Equations 11–14 of Section 2.1:

$$\frac{S_k \sim \text{Unif } [0,1] \wedge \mathbb{E}\left[\prod_{j \neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j]\right] = \frac{1}{\frac{\sum_{j \neq k} w_j}{w_k} + 1}}{\mathbb{E}\left[\prod_{j \neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j]\right] = \frac{w_k}{\sum_j w_j}}$$

$$\frac{\mathbb{E}\left[\prod_{j \neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j]\right] = \frac{w_k}{\sum_j w_j}}{\mathbb{E}\left[\prod_{j \neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j]\right] = \frac{w_k}{\sum_j w_j}}$$

QED