

Lilac: a Modal Separation Logic for Conditional Probability

JOHN M. LI, Northeastern University, USA

AMAL AHMED, Northeastern University, USA

STEVEN HOLTZEN, Northeastern University, USA

We present Lilac, a separation logic for reasoning about probabilistic programs where separating conjunction captures probabilistic 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 validates examples from prior work, and give a formal verification of an intricate weighted sampling algorithm whose correctness depends crucially on conditional independence structure.

ACM Reference Format:

John M. Li, Amal Ahmed, and Steven Holtzen. 2023. Lilac: a Modal Separation Logic for Conditional Probability. *Proc. ACM Program. Lang.* 7, PLDI, Article 112 (June 2023), 61 pages. <https://doi.org/10.1145/3591226>

1 INTRODUCTION

Software systems involving probability are pervasive. Such systems naturally appear in diverse domains such as network reliability analysis [Gehr et al. 2018; Smolka et al. 2019], reliability for cyberphysical systems [Holtzen et al. 2021; Lee and Seshia 2016], distributed software systems [Tassarotti and Harper 2019], and many others. As these systems are increasingly deployed in high-consequence domains, there is a growing need for formal frameworks capable of reasoning about and verifying probabilistic correctness properties. We are especially interested in formal frameworks that support *compositional* reasoning: putting probabilistic systems together correctly is a tricky business, as a probabilistic component often makes subtle assumptions about the distribution of its inputs. A formal framework for reasoning about probabilistic systems should facilitate the sound composition of formal verifications of individual components.

In the traditional non-probabilistic setting, *program logics* have become standard kit for compositionally reasoning about heap-manipulating programs at scale [Distefano et al. 2019]. In particular, *separation logic* enables modular reasoning about heap-manipulating programs [Ishtiaq and O'Hearn 2001; O'Hearn et al. 2009; Reynolds 2002, 2009]. The key to this modularity is the *frame rule*,

$$\frac{\{P\} e \{x. Q(x)\}}{\{F * P\} e \{x. F * Q(x)\}}, \quad (1)$$

Authors' addresses: John M. Li, Northeastern University, Boston, MA, USA, li.john@northeastern.edu; Amal Ahmed, Northeastern University, Boston, MA, USA, a.ahmed@northeastern.edu; Steven Holtzen, Northeastern University, Boston, MA, USA, s.holtzen@northeastern.edu.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.

2475-1421/2023/6-ART112

<https://doi.org/10.1145/3591226>

which states that a program e satisfying precondition P and postcondition Q doesn't interfere with any parts of the heap F ("frames") disjoint from parts of the heap described by P [O'Hearn 2012]. This facilitates local reasoning, and is the distinctive advantage of using the substructural separation logic over ordinary predicate logic in reasoning about pointers. An equivalent specification without separation logic leads to an unwieldy proliferation of assertions about inequality of locations and pointer-graph reachability [Reynolds 2002].

What is an effective separation logic for probabilistic programs? In the probabilistic setting, the fundamental source of modularity is *probabilistic independence*. Intuitively, two sources of randomness are independent if knowledge of one does not give any knowledge of the other. In direct analogy to disjointness of heaps, independence structure permeates probabilistic programs: sampling produces a random variable independent of all previously-sampled ones, and commonly-used subroutines (e.g. randomly initializing an array) generate multiple mutually-independent outputs. Just as in the traditional setting, attempting to write specifications for these procedures in ordinary predicate logic leads to a proliferation of independence assertions. A logic for compositional probabilistic reasoning should compositionally support independence, in the same way that ordinary separation logic compositionally supports reasoning about disjoint heaps.

In this paper we present Lilac, a separation logic whose separating conjunction means probabilistic independence. Lilac enjoys a frame rule that is identical to the frame rule of ordinary separation logic; as a consequence, the same modular reasoning principles used on heap-manipulating programs apply directly to the probabilistic setting. Moreover, we prove that Lilac's separating conjunction completely captures probabilistic independence: all probabilistic independence relationships are validated by its semantic model (Lemma 2.5). Both of these points are improvements over prior work [Bao et al. 2022; Barthe et al. 2019], and are consequences of our first core contribution: a new *combining operation on probability spaces*, analogous to *disjoint union of heap fragments* in ordinary separation logic, that serves as the interpretation of separating conjunction.

Our second core contribution is a *modal* treatment of conditional probability. It is common in many probabilistic systems for a property to only hold *conditional* on some random variable: for instance, two random variables might only be independent conditional on a third, a property called *conditional independence*. Conditional reasoning is a second powerful and prevalent source of modularity: 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. Historically, conditional independence has been very difficult to capture in a substructural program logic: it has either gone unsupported [Barthe et al. 2018, 2019] or required a host of new logical connectives and significant changes to the underlying semantic model [Bao et al. 2021]. Lilac captures conditioning via the addition of a single modal operator. Adding support for this conditioning modality doesn't require any changes to our underlying semantic model beyond restricting ourselves to a class of suitably-well-behaved probability spaces. By importing standard theorems of probability theory, we validate a set of derived rules about the conditioning modality that we argue captures the informal flavor of conditional reasoning.

In sum, our contributions are as follows:

- We present Core Lilac, a separation logic whose separating conjunction captures independence (Lemma 2.5), alongside proof rules for reasoning about a simple probabilistic programming language capable of expressing all of the examples we will consider (Section 2). Core Lilac's semantic model is based on a novel combining operation on probability spaces that resembles disjoint union of heap fragments (Lemma 2.4).

- We extend Core Lilac with a modal operator **C** to express conditional reasoning (Section 3). This makes Lilac the first logic that supports conditioning and continuous random variables in combination with a substructural treatment of independence.
- We validate the effectiveness of Lilac as a useful tool for program verification by proving correctness properties of existing examples from the literature as well as a new challenging example. Establishing independence structure is key for proving certain cryptographic protocols correct. We give Lilac proofs for the one-time pad, private information retrieval, and oblivious transfer protocols studied by Barthe et al. [2019] (see Appendix F). Lilac can also establish conditional independence properties: we show this by validating the conditional independence properties of all programs considered by Bao et al. [2021] (Sections 1.1 and 4.2). Finally, we consider a challenging new example that goes beyond the scope of existing separation logics. We validate an intricate reservoir sampling algorithm that uses continuous random variables and whose correctness argument depends crucially on conditional independence structure (Section 4.1).

1.1 A Tour of Lilac

To concretize the discussion, we now present a few simple examples in order to illustrate how Lilac’s separating conjunction encodes independence and how the conditioning modality can be used to establish simple conditional independence relationships. In Section 4.1 we will give a more involved example that simultaneously exercises all of the features described here.

First, consider a program **UNIF2** that samples two reals X, Y uniformly from the interval $[0, 1]$:

$$X \leftarrow \text{unif } [0, 1]; Y \leftarrow \text{unif } [0, 1]; \text{ret } (X, Y) \quad (\text{UNIF2})$$

We will write all probabilistic programs in monadic style, in a manner similar to Haskell’s notation; the keyword **ret** lifts the value (X, Y) into a pure monadic computation. This program satisfies two main properties of interest. First, the outputs X and Y are independent and distributed as $\text{Unif}[0, 1]$. Second, as freshly generated random variables, both X and Y are independent of all other variables. Both properties are asserted by the following Hoare triple:

$$\{\top\} \text{UNIF2} \{(X, Y). X \sim \text{Unif}[0, 1] * Y \sim \text{Unif}[0, 1]\}$$

We write $\{P\} M \{X. Q(X)\}$ for a program M that satisfies precondition P and produces a random variable X satisfying postcondition $Q(X)$. In this case, the precondition is the trivial \top . The proposition $X \sim \text{Unif}[0, 1]$ is a Lilac assertion: it asserts that random variable X is distributed as $\text{Unif}[0, 1]$. This is in direct analogy to the proposition $\ell \mapsto v$ from ordinary separation logic. Ordinarily, the separating conjunction $(\ell_1 \mapsto v_1) * (\ell_2 \mapsto v_2)$ asserts that ℓ_1 and ℓ_2 refer to disjoint heap chunks containing values v_1 and v_2 respectively. Analogously, the postcondition $(X \sim \text{Unif}[0, 1]) * (Y \sim \text{Unif}[0, 1])$ asserts that X is independent of Y (henceforth written $X \perp\!\!\!\perp Y$) and that both are distributed as $\text{Unif}[0, 1]$. Finally, the frame rule implies X and Y are independent of all other random variables, expressing the fact that X and Y are freshly generated.

To establish this postcondition, Lilac provides proof rules that enable the usual forward-symbolic-execution-style reasoning [Reynolds 2009]; we will present these in Section 2.5. The rule for generating a random variable using **unif** $[0, 1]$ is:

$$\{\top\} \text{unif } [0, 1] \{X. X \sim \text{Unif } [0, 1]\},$$

in direct analogy to the rule for allocating a new reference in ordinary separation logic. The rules for monadic operators are standard; see rules H-LET and H-RET in Figure 8.

```

{⊤}
  X ← unif [0, 1];
  {X ∼ Unif[0, 1]}
  Y ← unif [0, 1];
  {X ∼ Unif[0, 1] * Y ∼ Unif[0, 1]}
  ret (X, Y)
  {(X, Y). X ∼ Unif[0, 1] * Y ∼ Unif[0, 1]}

```

Fig. 1. Lilac-annotated **UNIF2**.

This allows derivations in the Lilac program logic to be abbreviated as assertion-annotated programs in the usual way. For example, the postcondition for **UNIF2** can be established by the annotation in Figure 1. So far we have not made use of any probability-specific reasoning principles; a proof analogous to the above could be carried out for a heap-manipulating program that allocates twice. Our next example illustrates how Lilac’s notion of separation can be used together with probability-specific proof rules to compute an expectation. Consider a program **HALVE** that takes a random variable X as input, generates a uniform random variable Y , and computes their product:

$$\mathbf{HALVE}(X) \quad := \quad (Y \leftarrow \mathbf{unif} \ [0, 1]; \ \mathbf{ret} \ XY) \quad (\mathbf{HALVE})$$

Because Y is freshly generated, X and Y must be independent; combined with the fact that Y is uniform, we have $\mathbb{E}[XY] = \mathbb{E}[X] \mathbb{E}[Y] = \mathbb{E}[X]/2$.¹ (Hence the name: **HALVE**(X)’s output is half of X in expectation.) This argument makes use of two key facts: expectation distributes over the product of independent random variables, and for $U \sim \text{Unif}[0, 1]$ we have $\mathbb{E}[U] = 1/2$.

We now show how to express this in Lilac. The claim that **HALVE** produces a random variable with expectation $\mathbb{E}[X]/2$ can be expressed by:

$$\{\text{own } X\} \ \mathbf{HALVE}(X) \ \{Z. \ \mathbb{E}[Z] = \mathbb{E}[X]/2\} \quad (2)$$

The postcondition states that the random variable Z produced by **HALVE** has expectation $\mathbb{E}[X]/2$. The precondition **own** X is a new kind of Lilac assertion: it asserts “probabilistic ownership” of a random variable X but nothing about its distribution. Probabilistic ownership is in direct analogy to the assertion $\ell \mapsto -$ of ordinary separation logic, which asserts ownership of a location ℓ in the heap but nothing about the value stored at ℓ . Probabilistic ownership allows us to describe independence relationships involving X without knowledge of its distribution. In particular, the proposition **own** X $*$ **own** Y asserts that $X \perp\!\!\!\perp Y$, just as the proposition $(\ell_1 \mapsto -) * (\ell_2 \mapsto -)$ asserts that ℓ_1 and ℓ_2 are disjoint locations in ordinary separation logic. In Section 1.2 we will describe precisely what ownership means in this probabilistic context.

The annotation in Figure 2 proves that **HALVE** meets the specification in Equation 2. The proof proceeds in two phases. In the first phase, we apply standard proof rules for **unif** $[0, 1]$ and **ret** to obtain the assertion on Line 5.² As is standard for proofs in separation logic, we implicitly apply the frame rule when needed. In this case, the frame rule guarantees that **own** X is preserved across the invocation of **unif** $[0, 1]$, giving the separating conjunction on Line 3. This formalizes the intuition that Y is freshly generated, and therefore independent of X . Line 5 introduces the proposition $Z \stackrel{\text{as}}{=} XY$,

```

1 {own X}
2   Y ← unif [0, 1];
3 {own X * Y ~ Unif[0, 1]}
4   ret XY
5 {Z. own X * Y ~ Unif[0, 1] * Z as XY}
6 {Z. (E[Y] = 1/2 ∧ E[XY] = E[X] E[Y]) * Z as XY}
7 {Z. E[Y] = 1/2 ∧ E[Z] = E[X] E[Y]}
8 {Z. E[Z] = E[X]/2}

```

Fig. 2. Lilac-annotated **HALVE**.

which asserts that the output Z is almost-surely equal to the product XY . Two random variables X and Y are almost-surely equal if they are equal with probability 1 (i.e., $\Pr(X = Y) = 1$); this is the natural notion of equality for random variables. See Figure 6 for the semantics of $(\stackrel{\text{as}}{=})$. Note that using separating conjunction to combine this proposition with the others does not introduce any spurious independence relationships; unlike **own** X or $Y \sim \text{Unif}[0, 1]$ the proposition $Z \stackrel{\text{as}}{=} XY$ does

¹ $\mathbb{E}[X]$ denotes the expectation of the random variable X .

²Section 2.5 describes these rules in detail.

not assert ownership of any random variable.³ This is in contrast to prior work such as Barthe et al. [2019], where almost-sure equality requires ownership of the variables being compared and, as a result, proofs involving equalities often require a cumbersome mixing of (\wedge) and ($*$).

Lines 6-8 finish the proof using proof rules expressing nontrivial probability-specific reasoning. Line 6 applies the proof rules:

$$\begin{array}{ll} \text{own } X * \text{own } Y & \vdash \mathbb{E}[XY] = \mathbb{E}[X] \mathbb{E}[Y] & (\text{INDEP-PROD}) \\ Y \sim \text{Unif}[0, 1] & \vdash \mathbb{E}[Y] = 1/2 & (\text{EX-UNIF}) \end{array}$$

to obtain a conjunction of equalities about the expectations of Y and XY . Unlike the proof rules we have considered so far, which are structural in nature and express standard logical reasoning principles, these rules express probability-specific facts: **INDEP-PROD** expresses the fact that expectation distributes over products of independent random variables, and **EX-UNIF** expresses the expectation of the standard uniform distribution. As probability-specific proof rules, these do not follow from the ordinary laws of separation logic; instead, they are validated by the probability-specific model that we will describe in Section 1.2. Line 7 uses the almost-sure equality $Z \stackrel{\text{as}}{=} XY$ to replace all occurrences of XY with Z . The rest of the proof follows by calculation.

Lilac’s conditioning modality. Conditioning is at the heart of probabilistic reasoning, and is central to most probabilistic arguments. A core goal of Lilac is to facilitate conditional probability arguments that would be familiar to probability theorists. Informally, conditioning “[turns a] random event or variable into a deterministic one, while preserving the random nature of other events and variables” [Tao 2015]. We capture this intuition with the *conditioning modality* “ $\mathbf{C}_{x \leftarrow X} P(x)$ ”, which asserts that $P(x)$ holds *conditional* on all possible values x the random variable X can take on. A key feature of this modality is that standard separation logic assertions have intuitive conditional readings under it; thus **C** lifts statements about unconditional probability to their natural conditional counterparts. For instance, we have seen that $\text{own } X * \text{own } Y$ asserts that $X \perp\!\!\!\perp Y$, and $\mathbb{E}[X] = v$ asserts that X has expectation v . Accordingly, $\mathbf{C}_{z \leftarrow Z} (\text{own } X * \text{own } Y)$ expresses *conditional independence* of X and Y given the random variable Z , denoted $X \perp\!\!\!\perp Y \mid Z$, and $\mathbf{C}_{y \leftarrow Y} (\mathbb{E}[X] = v)$ asserts that X has *conditional expectation* v given Y , i.e. $\mathbb{E}[X \mid Y] = v$.

Conditional arguments are driven by a careful distinction between random and deterministic quantities. We make these distinctions notationally and semantically explicit in Lilac. We write random expressions as capital letters X ; these variables stand for random variables manipulated by a probabilistic program, and are compared for equality using ($\stackrel{\text{as}}{=}$). *Deterministic* (i.e., non-probabilistic) variables are written in lower-case letters x , and are compared for equality using ordinary ($=$).

Lilac supports familiar ways of transitioning between deterministic and probabilistic quantities, and typing rules (given in Section 2.1) govern precisely where random and deterministic expressions can appear. For instance, $\mathbb{E}[X] = v$ asserts that a random expression X relates to a deterministic quantity v . The conditioning modality permits arguments that mediate between the random and the deterministic. When a probability-theorist says “and now we proceed by conditioning X ” in a pen-and-paper proof, we translate this to entering the conditioning modality. The proposition $\mathbf{C}_{x \leftarrow X} P(x)$ binds a new deterministic variable x for use in $P(x)$. Intuitively, the deterministic x represents an arbitrary but non-probabilistic value that X has been fixed to inside P via conditioning. This allows replacing X with x inside P . For example, the proposition $\mathbf{C}_{x \leftarrow X} (\mathbb{E}[X] = x)$ is valid: under $\mathbf{C}_{x \leftarrow X}$, we can replace X with x to get $\mathbb{E}[X] = \mathbb{E}[x]$, and $\mathbb{E}[x] = x$ because x is deterministic.

This ability to turn random quantities X into deterministic variables x is especially useful because deterministic variables are, generally speaking, better behaved than their random counterparts.

³Section 2.3 will make this precise.

In particular, deterministic variables support case analysis: if b is a deterministic variable of type `bool` then $P[\top/b] \wedge P[\text{F}/b] \vdash P$. This allows Lilac to express a kind of conditional argument by case analysis that is pervasive in probabilistic reasoning. For example, consider the following program:

```

Z ← flip 1/2;  X ← flip 1/2;  Y ← flip 1/2;
A ← ret (X || Z);  B ← ret (Y || Z);
ret (Z, X, Y, A, B)

```

(COMMONCAUSE)

This program is taken from Figure 6(a) of Bao et al. [2021] (and translated into a monadic style), where it is used as an example of conditional independence structure. The program computes A and B from mutually-independent boolean random variables Z, X, Y generated by `flip 1/2`; at exit, we have that $A \perp\!\!\!\perp B \mid Z$. In Bao et al. [2021], this is established via a logic of “doubly-bunched” implications – an extension of separation logic with an additional family of substructural connectives for expressing conditional independence. We will show how Lilac can state and prove this conditional independence without having to introduce new connectives beyond \mathbf{C} .

The conditional independence property $A \perp\!\!\!\perp B \mid Z$ is captured by the following triple:

$$\{\top\} \text{COMMONCAUSE} \left\{ (Z, X, Y, A, B). \mathbf{C}_{z \leftarrow Z} (\text{own } A * \text{own } B) \right\}$$

This postcondition can be established as follows.⁴ First, applying rules for `flip` and `||` gives:

$$Z \sim \text{Ber } 1/2 * X \sim \text{Ber } 1/2 * Y \sim \text{Ber } 1/2 * A \stackrel{\text{as}}{=} (X \vee Z) * B \stackrel{\text{as}}{=} (Y \vee Z)$$

At this point, an informal proof would continue by case analysis on Z as follows. Because Z is a Boolean value, it can only take on one of two values: `T` or `F`. First establish $A \perp\!\!\!\perp B \mid Z = \text{T}$. If $Z = \text{T}$ then $A = B = \text{T}$, and therefore $A \perp\!\!\!\perp B$ because by definition $\text{T} \perp\!\!\!\perp \text{T}$. Now establish $A \perp\!\!\!\perp B \mid Z = \text{F}$. If $Z = \text{F}$ then $A = X$ and $B = Y$. By construction we have $X \perp\!\!\!\perp Y \perp\!\!\!\perp Z$, since they are all independent `flips`. This mutual independence implies that X and Y remain independent after conditioning on $Z = \text{F}$. Hence $A \perp\!\!\!\perp B \mid Z = \text{F}$. This completes the case analysis, so $A \perp\!\!\!\perp B \mid Z$ as desired.

In Lilac, this conditional argument is expressed by introducing the operator $\mathbf{C}_{z \leftarrow Z}$ and performing case analysis on the deterministic z . First, $\mathbf{C}_{z \leftarrow Z}$ is introduced as follows:

$$\mathbf{C}_{z \leftarrow Z} \left(X \sim \text{Ber } 1/2 * Y \sim \text{Ber } 1/2 * A \stackrel{\text{as}}{=} (X \vee Z) * B \stackrel{\text{as}}{=} (Y \vee Z) \right)$$

This step formalizes the idea that anything independent of Z continues to be independent after conditioning. It is justified by an application of the C-INDEP rule, which we will describe in Section 3.

Conditioning on Z allows us to replace occurrences of the random variable Z with the newly-introduced deterministic z :

$$\mathbf{C}_{z \leftarrow Z} \left(X \sim \text{Ber } 1/2 * Y \sim \text{Ber } 1/2 * A \stackrel{\text{as}}{=} (X \vee z) * B \stackrel{\text{as}}{=} (Y \vee z) \right) \quad (3)$$

We are done if we can show that (3) entails $\mathbf{C}_{z \leftarrow Z}(\text{own } A * \text{own } B)$. At this point we use a key property of \mathbf{C} : as a modal operator, it respects entailment, so we must establish the \mathbf{C} -free entailment.⁵

$$X \sim \text{Ber } 1/2 * Y \sim \text{Ber } 1/2 * A \stackrel{\text{as}}{=} (X \vee z) * B \stackrel{\text{as}}{=} (Y \vee z) \vdash \text{own } A * \text{own } B.$$

Now the rest of the proof follows directly by cases on z : if $z = \text{T}$ then after simplifying we have:

$$A \stackrel{\text{as}}{=} \text{T} * B \stackrel{\text{as}}{=} \text{T} \vdash \text{own } A * \text{own } B,$$

⁴Here we prefer to describe the proof as a mixture of prose and Lilac assertions. Appendix C gives a fully annotated program.

⁵That is, if $P \vdash Q$ then $\mathbf{C}_{x \leftarrow X} P \vdash \mathbf{C}_{x \leftarrow X} Q$.

which follows from the rule $X \stackrel{\text{as}}{=} \top \vdash \text{own } X$ specialized to $X = A$ and $X = B$. On the other hand, if $z = \text{F}$ then we are left (again, after simplifications) with

$$A \sim \text{Ber } 1/2 * B \sim \text{Ber } 1/2 \vdash \text{own } A * \text{own } B,$$

which follows from the rule $X \sim \text{Ber } 1/2 \vdash \text{own } X$.


1.2 A Tour of Lilac's Semantic Model

So far we have sketched a system of proof rules for reasoning about probability that would appear quite intuitive to a probability theorist. However, reasonable-looking proof rules are only half of the story. To ensure that the rules are sound and that Lilac propositions have sensible interpretations in terms of existing probability-theoretic objects, we construct a model that validates the reasoning principles described in the previous section and grounds them in probability-theory.

Lilac's semantic model is designed by analogy with mutable state. The key idea is that *probability spaces* — mathematical objects that model random phenomena — behave like heaps. To make this concrete, consider the following program:


$$X \leftarrow \text{flip } 1/2; Y \leftarrow \text{flip } 1/2; \text{ret } (X, Y) \quad (\text{FLIP2})$$

According to the informal semantics for probabilistic programs used in Section 1.1, this program “generates” two uniformly distributed boolean random variables X and Y . To make this precise, we need some standard definitions. A *probability space* is a tuple $(\Omega, \mathcal{F}, \mu)$. The set Ω is called the *sample space*. The set \mathcal{F} , called the σ -*algebra*, is a collection of subsets of Ω that (1) contains the empty set and Ω , and (2) satisfies closure under countable union and complements. Elements $E \in \mathcal{F}$ are *events*; events models an observable property of the phenomenon. The map $\mu : \mathcal{F} \rightarrow [0, 1]$ is a *probability measure* assigning each event its probability. Given a finite set A , an *A-valued random variable* is a map $X : \Omega \rightarrow A$ that is \mathcal{F} -*measurable*, which means that the set $\{\omega \mid X(\omega) = a\}$ is an event of \mathcal{F} for all $a \in A$. Intuitively, a random variable X represents an object of type A that depends on the random phenomenon modelled by Ω ; the measurability condition ensures that X only depends on observable properties.

To visualize these objects, we will temporarily fix Ω to be the unit square $[0, 1] \times [0, 1]$ and μ to be the function that assigns to each subset of Ω its area, if possible. This allows us to draw a probability space $(\Omega, \mathcal{F}, \mu)$ as a partitioning of the unit square given by the σ -algebra \mathcal{F} . For example, the square  depicts a probability space with σ -algebra $\mathcal{F}_{\square} := \{\emptyset, \text{blue}, \text{orange}, \Omega\}$ and probability measure:

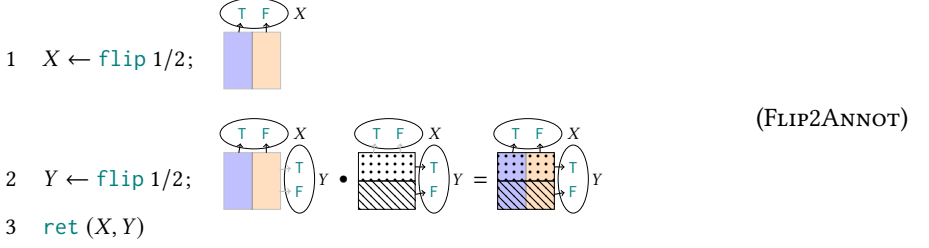
$$\mu(\emptyset) = 0 \quad \mu(\text{blue}) = 1/2 \quad \mu(\text{orange}) = 1/2 \quad \mu(\Omega) = 1.$$

This probability space models a random phenomenon with two events (the blue rectangle and the orange rectangle), each with equal probability.

For example, one can define a random variable on the space  as follows. Let A be the set of boolean values $\{\text{T}, \text{F}\}$ and $X : [0, 1] \times [0, 1] \rightarrow \{\text{T}, \text{F}\}$ be a function that sends blue points to T and orange points to F . Concretely, $X(\omega_1, \omega_2) = \text{T}$ if $\omega_1 < 1/2$ and F otherwise. This is a random variable: the measurability condition is satisfied because $X^{-1}(\text{T}) = \text{blue} \in \mathcal{F}_{\square}$ and $X^{-1}(\text{F}) = \text{orange} \in \mathcal{F}_{\square}$.

We now have the machinery necessary to interpret the **FLIP2** example. The idea is to think of **FLIP2** as carrying along a probability space as it executes, analogous to how programs with mutable

state carry along a heap. We can visualize execution of **FLIP2** as follows:



Before Line 1, the probability space has exactly two events, \emptyset and Ω , with respective probabilities 0 and 1. This is the trivial probability space, analogous to an empty heap. After the first line is executed two things change:

- Two new events are allocated, forming the blue-orange probability space we considered earlier. This is analogous to how **new** allocates a fresh memory cell on the heap: probability spaces correspond to heap fragments.
- The **flip** operation yields the random variable X we considered earlier that maps blue points to **T**; it is depicted by the arrows. This is analogous to how **new** returns the location of the newly allocated heap cell: random variables correspond to locations.

Line 2 allocates a second probability space , whose events are the dotted region and dashed region , and a new random variable Y that associates dotted points to **T** and dashed points to **F**; concretely, $Y(\omega_1, \omega_2) = \text{T}$ if $\omega_2 > 1/2$ and **F** otherwise. There are many other possible alternatives to and Y ; we have simply chosen the ones that are easiest to visualize. This is analogous to how, in heap-manipulating languages, there are many possible locations in the heap that **new** can choose to allocate in. Ordinarily, the location chosen by **new** is also *fresh*, 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 *probabilistically independent* from the old one, so that the entire space after executing the second **flip** is an *independent combination* of the old space and the newly allocated one . The operator (\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 2.4), and is the heart of Lilac's notion of separation. The events of the combined space are generated by the events of and ; we give a formal definition in Section 2. 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, a probability space \mathcal{P} satisfies assertion $P_1 * P_2$ if there exists a "splitting" of \mathcal{P} into an independent combination $\mathcal{P}_1 \bullet \mathcal{P}_2$ such that \mathcal{P}_1 satisfies P_1 and \mathcal{P}_2 satisfies P_2 .

FLIP2ANNOT also illustrates the second key insight that forms the basis for our model of separation logic: ownership is measurability. In Section 1.1 we described the proposition $\text{own } X$ as asserting "probabilistic ownership" of X but no knowledge of its distribution. Now we make this precise: the proposition $\text{own } X$ holds in probability space $(\Omega, \mathcal{F}, \mu)$ when X is \mathcal{F} -measurable. In **FLIP2ANNOT**, $\text{own } X$ holds in the space because X is $\mathcal{F}_{\text{blue-orange}}$ -measurable. On the other hand, $\text{own } Y$ does *not* hold in , because the event $Y^{-1}(\text{F}) = \text{dashed}$ is not contained in $\mathcal{F}_{\text{blue-orange}}$. Similarly, $\text{own } X$ does not hold in because $X^{-1}(\text{T}) = \text{blue}$ $\notin \mathcal{F}_{\text{dotted-dashed}}$. The grayed-out arrows depict non-measurability in **FLIP2ANNOT**.

2 CORE LILAC

Now that we have given informal descriptions of Lilac's proof rules and semantic model, we now begin the formal development of Core Lilac, a subset of Lilac without the conditioning modality.

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

Fig. 3. 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 4.

$$\begin{array}{c}
\begin{array}{l} S \in \text{Set} \\ A \in \text{Meas} \end{array} \quad \begin{array}{l} \Gamma ::= \cdot \mid \Gamma, x : S \\ \Delta ::= \cdot \mid \Delta, X : A \end{array} \quad \begin{array}{l} \llbracket x_1 : S_1, \dots, x_n : S_n \rrbracket = S_1 \times \dots \times S_n \\ \llbracket X_1 : A_1, \dots, X_n : A_n \rrbracket = A_1 \times \dots \times A_n \end{array} \\
\\
\frac{E \in \llbracket \Gamma \rrbracket \rightarrow \llbracket \Delta \rrbracket \xrightarrow{m} A}{\Gamma; \Delta \vdash_{rv} E : A} \text{T-RANDE} \quad \frac{e \in \llbracket \Gamma \rrbracket \rightarrow A}{\Gamma \vdash_{det} e : A} \text{T-DETE} \quad \frac{\mu \in \llbracket \Gamma \rrbracket \rightarrow \mathcal{GA}}{\Gamma \vdash_{det} \mu : A} \text{T-DISTR} \\
\\
\frac{M \in \llbracket \Gamma \rrbracket \rightarrow \llbracket \Delta \rrbracket \xrightarrow{m} \mathcal{GA}}{\Gamma; \Delta \vdash_{prog} M : A} \text{T-PROG} \quad \frac{\Gamma, x:S; \Delta \vdash P}{\Gamma; \Delta \vdash \forall x:S.P} \text{T-DET}\forall \quad \frac{\Gamma; \Delta, X:A \vdash P}{\Gamma; \Delta \vdash \forall_{rv} X:A.P} \text{T-RAND}\forall \\
\\
\frac{\Gamma; \Delta \vdash_{rv} E : A \quad \Gamma \vdash_{det} \mu : A}{\Gamma; \Delta \vdash E \sim \mu} \text{T-SIM} \quad \frac{\Gamma; \Delta \vdash_{rv} E : \mathbb{R} \quad \Gamma \vdash_{det} e : \mathbb{R}}{\Gamma; \Delta \vdash \mathbb{E}[E] = e} \text{T-E} \quad \frac{\Gamma; \Delta \vdash_{rv} E_1 : A \quad \Gamma; \Delta \vdash_{rv} E_2 : A}{\Gamma; \Delta \vdash E_1 \stackrel{as}{=} E_2} \text{T-}\stackrel{as}{=} \\
\\
\frac{\Gamma; \Delta \vdash_{prog} M : A \quad \Gamma; \Delta, X:A \vdash Q}{\Gamma; \Delta \vdash \text{wp}(M, X:A.Q)} \text{T-wp}
\end{array}$$

Fig. 4. Core Lilac typing rules. Contexts Γ contain the types of deterministic variables and contexts Δ contain the types of random variables. Metavariables E range over random expressions and e over deterministic expressions. We write \mathcal{GA} for the set of distributions on A .

First we will introduce the syntax and semantics of Core Lilac propositions. Then we present our combining operation (\bullet) on probability spaces described in Section 1.2, and show that it behaves like disjoint union of heaps. Next, we give the semantics of Lilac propositions, and 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.

2.1 Syntax and Typing of Core Lilac Propositions

The syntax of Core Lilac propositions is given in Figure 3. It includes the standard intuitionistic and substructural connectives, plus the probability-specific ones $\text{own } E$, $E \sim \mu$, $\mathbb{E}[E] = e$, and $E_1 \stackrel{as}{=} E_2$ introduced in Section 1.1. Figure 4 gives selected typing rules for Core Lilac.⁶ The typing judgment has shape $\Gamma; \Delta \vdash P$, where Γ is a context containing the types of deterministic variables and Δ is a context containing the types of random variables. It is defined in terms of auxiliary judgments $\Gamma; \Delta \vdash_{rv} E : A$ for typing random expressions E , which may mention both deterministic and random variables, and $\Gamma \vdash_{det} e : A$ for typing deterministic expressions e , which can only mention deterministic variables, and $\Gamma; \Delta \vdash_{prog} M : A$ for typing programs. Since there are two kinds of variables, there are also two kinds of quantifiers, with typing rules T-DET \forall and T-RAND \forall .

For clarity of presentation the Core Lilac syntax in Figure 3 permits reference to arbitrary measurable spaces in its types and arbitrary measurable functions in its terms, in a manner similar to Shan and Ramsey [2017] and Staton [2020]. The rule T-RANDE characterizes the kinds of functions

⁶The full typing rules are in Appendix B.2.

allowed as random expressions: a random expression E has type A in context $(\Gamma; \Delta)$ if it is a $\llbracket \Gamma \rrbracket$ -indexed family of measurable maps $\llbracket \Delta \rrbracket \xrightarrow{m} A$. For instance, the following random expressions are all well-typed via T-RAND E because $(+)$ and $\text{pow}(-, -)$ are both measurable functions $\mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$:

$$\frac{\Gamma; \Delta \vdash_{\text{rv}} E_1 : \mathbb{R} \quad \Gamma; \Delta \vdash_{\text{rv}} E_2 : \mathbb{R}}{\Gamma; \Delta \vdash_{\text{rv}} E_1 + E_2 : \mathbb{R}} \quad \frac{\Gamma; \Delta \vdash_{\text{rv}} E_1 : \mathbb{R} \quad \Gamma; \Delta \vdash_{\text{rv}} E_2 : \mathbb{R}}{\Gamma; \Delta \vdash_{\text{rv}} \text{pow}(E_1, E_2) : \mathbb{R}}$$

Similarly, T-DETE says a deterministic expression e is well-typed at A in Γ if it is a function $\llbracket \Gamma \rrbracket \rightarrow A$, and T-DISTR says μ is well-typed at A in context Γ if it is a $\llbracket \Gamma \rrbracket$ -indexed family of distributions; we write \mathcal{GA} for the set of distributions on A . Finally, T-PROG says a program M is well-typed in $\Gamma; \Delta$ if it is a $\llbracket \Gamma \rrbracket$ -indexed family of *Markov kernels* — maps $A \xrightarrow{m} \mathcal{GB}$ often used to give semantics to probabilistic programs [Staton 2020].

To reason about the behavior of programs we add a weakest precondition modality in the style of dynamic logic [Harel et al. 2001; Jung et al. 2018]. Intuitively, $\text{wp}(M, X:A.Q)$ asserts that M produces a random variable X satisfying postcondition Q . For example, the fact that **FLIP2** from Section 1.2 produces two independent $\text{Ber } 1/2$ is stated $\text{wp}(\llbracket \text{FLIP2} \rrbracket, (X, Y). X \sim \text{Ber } 1/2 * Y \sim \text{Ber } 1/2)$.

2.2 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 described in Section 1.2, this combining operation behaves like disjoint union of heap fragments, and underlies Lilac's model of separation. Formally, this is captured by the notion of a Kripke resource monoid [Galmiche et al. 2005]:

DEFINITION 2.1. A Kripke resource monoid is a tuple $(\mathcal{M}, \sqsubseteq, \bullet, 1)$ where

- (1) $(\mathcal{M}, \sqsubseteq)$ is a poset,
- (2) (\bullet) is a partial function $\mathcal{M} \times \mathcal{M} \rightarrow \mathcal{M}$,
- (3) $(\mathcal{M}, \bullet, 1)$ is a partial commutative monoid,
- (4) (\bullet) respects (\sqsubseteq) : if $x \sqsubseteq x'$ and $y \sqsubseteq y'$ and $x' \bullet y'$ defined, then $x \bullet y$ defined and $x \bullet y \sqsubseteq x' \bullet y'$.⁷

Intuitively, a KRM models the notion of a resource. The set \mathcal{M} models the space of possible resources; the ordering (\sqsubseteq) models how a given resource can evolve over time. The operation (\bullet) models how resources can be combined; it is partial because not all resources are compatible with each other (e.g., overlapping heap fragments). The choice of (\bullet) , which captures the desired notion of separation, determines the interpretations of the standard separation logic connectives. Our choice, as foreshadowed in Section 1.2, combines two independent probability spaces:

DEFINITION 2.2 (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 1.2. In this example the probability space  is obtained as an independent combination of  and . To show this, the areas of the regions in  must be products of intersections of regions in  and . Consider the events $X^{-1}(\text{blue}) = \text{blue}$ and $Y^{-1}(\text{green}) = \text{green}$. Both of these events have area $1/2$, and their intersection  — the upper-left quadrant of the unit square — has area $1/4 = (1/2)(1/2)$ as desired; clearly this holds for all quadrants.

To form a resource monoid, the combining operation (\bullet) must be a partial function. Definition 2.2 relates probability spaces to their independent combinations. However, it requires a witness ρ of independence; if there are multiple possible choices for ρ , then this relation does not define a partial

⁷This is a specialization of Definition 5.5 from Galmiche et al. [2005].

function. Thankfully – and somewhat surprisingly – it is possible to establish the uniqueness of ρ and therefore of independent combinations:

LEMMA 2.3 (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 $\rho = \rho'$.*

The proof is concise and relies on an application of the well-known *Dynkin π - λ theorem* [Kallenberg 1997]; see Appendix B.3 for details. Given Lemma 2.3, we can safely write $(\mathcal{E}, \mu) \bullet (\mathcal{F}, \nu) = (\mathcal{G}, \rho)$ whenever (\mathcal{G}, ρ) is an independent combination of (\mathcal{E}, μ) and (\mathcal{F}, ν) , making (\bullet) a partial function on probability spaces. In addition to being a partial function, Definition 2.1 also requires that (\bullet) forms a partial commutative monoid and respects a certain ordering relation. This indeed holds of our model: we take the ordering relation to be inclusion of probability spaces, analogous to inclusion of heaps used in ordinary (affine) separation logic:

THEOREM 2.4. *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 $\mu = \nu|_{\mathcal{F}}$.⁸ The tuple $(\mathcal{M}, \sqsubseteq, \bullet, 1)$ is a Kripke resource monoid, where 1 is the trivial probability space (\mathcal{F}_1, μ_1) with $\mathcal{F}_1 = \{\emptyset, \Omega\}$ and $\mu_1(\Omega) = 1$.*

PROOF. The main proof obligation is to establish associativity of (\bullet) . This follows from an application of the π - λ theorem. The proof is intricate; for details, see Appendix B.4. \square

There is a curious contrast between (\bullet) 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 does not require ρ . Instead, Lemma 2.3 guarantees that if any such ρ exists, it is unique. This observation turns the standard definition into 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.

2.3 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 5 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 6 gives the probability-specific rules, which we will discuss after. Both definitions are parameterized by an ambient sample space Ω equipped with a σ -algebra Σ_Ω : all probability spaces \mathcal{P} are assumed to contain sub- σ -algebras of Σ_Ω , and $\text{RV } A$ denotes the set of random variables over Ω . Because Ω is fixed throughout, we write probability spaces simply as (\mathcal{F}, μ) .

Figure 5 defines the meaning of propositions. Each proposition $\Gamma; \Delta \vdash P$ is interpreted as a set of *configurations* of the form (γ, D, \mathcal{P}) by the relation $\gamma, D, \mathcal{P} \models P$. In ordinary separation logic, configurations are of the form (s, h) where h is a heap and s a substitution associating values to variables. Here, the probability space \mathcal{P} plays the role of the heap. The pair (γ, D) plays the role of the substitution; because Lilac has two kinds of variables – random and deterministic – it also has two kinds of substitutions: $\gamma \in \llbracket \Gamma \rrbracket$ maps each deterministic variable to a value, and $D \in \text{RV}[\llbracket \Delta \rrbracket]$ maps each random variable to a mathematical random variable. The last four lines of Figure 5 give

⁸For a distribution $\mu : \mathcal{G} \rightarrow [0, 1]$ and sub- σ -algebra $\mathcal{F} \subseteq \mathcal{G}$, we write $\mu|_{\mathcal{F}}$ for the restriction of μ to \mathcal{F} .

$\gamma, D, \mathcal{P} \models \top$	always
$\gamma, D, \mathcal{P} \models \perp$	never
$\gamma, D, \mathcal{P} \models P \wedge Q$	iff $\gamma, D, \mathcal{P} \models P$ and $\gamma, D, \mathcal{P} \models Q$
$\gamma, D, \mathcal{P} \models P \vee Q$	iff $\gamma, D, \mathcal{P} \models P$ or $\gamma, D, \mathcal{P} \models Q$
$\gamma, D, \mathcal{P} \models P \rightarrow Q$	iff $\gamma, D, \mathcal{P}' \models P$ implies $\gamma, D, \mathcal{P}' \models Q$ for all $\mathcal{P}' \sqsupseteq \mathcal{P}$
$\gamma, D, \mathcal{P} \models P * Q$	iff $\gamma, D, \mathcal{P}_P \models P$ and $\gamma, D, \mathcal{P}_Q \models Q$ for some $\mathcal{P}_P \bullet \mathcal{P}_Q \sqsubseteq \mathcal{P}$
$\gamma, D, \mathcal{P} \models P \multimap Q$	iff $\gamma, D, \mathcal{P}_P \models P$ implies $\gamma, D, \mathcal{P}_P \bullet \mathcal{P} \models Q$ for all \mathcal{P}_P with $\mathcal{P}_P \bullet \mathcal{P}$ defined
$\gamma, D, \mathcal{P} \models \Box P$	iff $\gamma, D, 1 \models P$
$\gamma, D, \mathcal{P} \models \forall x:S.P$	iff $(\gamma, x), D, \mathcal{P} \models P$ for all $x \in S$
$\gamma, D, \mathcal{P} \models \exists x:S.P$	iff $(\gamma, x), D, \mathcal{P} \models P$ for some $x \in S$
$\gamma, D, \mathcal{P} \models \forall_{rv} X:A.P$	iff $\gamma, (D, X), \mathcal{P} \models P$ for all $X : RV A$
$\gamma, D, \mathcal{P} \models \exists_{rv} X:A.P$	iff $\gamma, (D, X), \mathcal{P} \models P$ for some $X : RV A$

Fig. 5. Semantics of basic Lilac connectives.

familiar-looking interpretations of quantifiers. All other lines are standard for separation logics, and follow from the fact that (\bullet) forms a Kripke resource monoid.

$\gamma, D, (\mathcal{F}, \mu) \models \text{own } E$	iff $E(\gamma) \circ D$ is \mathcal{F} -measurable
$\gamma, D, (\mathcal{F}, \mu) \models E \sim \mu'$	iff $E(\gamma) \circ D$ is \mathcal{F} -measurable and $\mu'(\gamma) = \left(\begin{array}{l} \omega \leftarrow \mu; \\ \text{ret } (E(\gamma)(D(\omega))) \end{array} \right)$
$\gamma, D, (\mathcal{F}, \mu) \models \mathbb{E}[E] = e$	iff $E(\gamma) \circ D$ is \mathcal{F} -measurable and $\mathbb{E}_{\omega \sim \mu}[E(\gamma)(D(\omega))] = e(\gamma)$
$\gamma, D, (\mathcal{F}, \mu) \models E_1 \stackrel{\text{as}}{=} E_2$	iff $(E_1(\gamma) \circ D) _F = (E_2(\gamma) \circ D) _F$ for some $F \in \mathcal{F}$ with $\mu(F) = 1$
$\gamma, D, \mathcal{P} \models \text{wp}(M, X:A.Q)$	iff for all $\mathcal{P}_{\text{frame}}$ and μ with $\mathcal{P}_{\text{frame}} \bullet \mathcal{P} \sqsubseteq (\Sigma_\Omega, \mu)$ and all $D_{\text{ext}} : RV \llbracket \Delta_{\text{ext}} \rrbracket$ there exists $X : RV A$ and \mathcal{P}' and μ' with $\mathcal{P}_{\text{frame}} \bullet \mathcal{P}' \sqsubseteq (\Sigma_\Omega, \mu')$ such that $\left(\begin{array}{l} \omega \leftarrow \mu; \\ v \leftarrow M(\gamma)(D(\omega)); \\ \text{ret } (D_{\text{ext}}(\omega), D(\omega), v) \end{array} \right) = \left(\begin{array}{l} \omega \leftarrow \mu'; \\ \text{ret } (D_{\text{ext}}(\omega), D(\omega), X(\omega)) \end{array} \right)$ and $\gamma, (D, X), \mathcal{P}' \models Q$

Fig. 6. Semantics of probability-specific Lilac connectives.

Figure 6 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 1.2, ownership corresponds to measurability of a random variable with respect to a particular σ -algebra. This intuition is made formal here: the proposition $\text{own } E$ 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.⁹ The expression $E(\gamma) \circ D$ is constructed by performing the relevant substitutions: first all deterministic values are substituted into E , 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 probabilistic notion of independence of random variables:

LEMMA 2.5 (SEPARATING CONJUNCTION IS MUTUAL INDEPENDENCE). *Fix a configuration (γ, D, \mathcal{P}) . Abbreviate $X_i(\gamma) \circ D$ as X'_i . Then, $\perp\!\!\!\perp_i X'_i$ holds with respect to \mathcal{P} if and only if $(\gamma, D, \mathcal{P}) \models \bigstar_i \text{own } X_i$.*

For a proof, see Appendix B.9. 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 values in γ for deterministic variables in the distribution expression μ' . Throughout

⁹Formally, for a probability space $(\Omega, \mathcal{F}, \mu)$, a random variable $X : (\Omega, \mathcal{F}) \rightarrow (A, \mathcal{A})$ is \mathcal{F} -measurable if for every $E \in \mathcal{A}$ it holds that $X^{-1}(E) \in \mathcal{F}$.

$$\begin{aligned}
A, B &::= A \times B \mid \text{bool} \mid \text{real} \mid A^n \mid \text{index} \mid \mathsf{G} A \\
M, N, O &::= X \mid \text{ret } M \mid X \leftarrow M; N \mid (M, N) \mid \text{fst } M \mid \text{snd } M \mid \\
&\quad \mathsf{T} \mid \mathsf{F} \mid \text{if } M \text{ then } N \text{ else } O \mid \text{flip } p \mid r \mid M \oplus N \mid M < N \mid \text{unif } [\emptyset, 1] \mid \\
&\quad [M, \dots, M] \mid M[N] \mid \text{for } (n, M_{\text{init}}, i \ X. M_{\text{step}})
\end{aligned}$$

Fig. 7. APPL syntax. Metavariables p range over probabilities, r over real numbers, and n over natural numbers; \oplus and $<$ range over standard arithmetic and comparison operators.

this figure, we use Haskell-style notation to construct distributions using the Giriy monad [Giry 1982]; here we use this notation on the right hand side of the equation for $\mu'(y)$ 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 proposition $E_1 \stackrel{\text{as}}{=} E_2$ holds with respect to $(\gamma, D, (\mathcal{F}, \mu))$ if E_1 and E_2 are almost-surely equal: formally, we require that there exists some $F \in \mathcal{F}$ with probability 1 (i.e., $\mu(F) = 1$) such that the random variables $E_1(\gamma) \circ D$ and $E_2(\gamma) \circ D$ agree on F . Note that we do not require E_1 or E_2 to be \mathcal{F} -measurable: for $E_1 \stackrel{\text{as}}{=} E_2$ to hold in \mathcal{F} , it need only contain a single F witnessing the almost-sure equality. This makes $E_1 \stackrel{\text{as}}{=} E_2$ a *duplicable proposition*,¹⁰ and allows it to be combined with other propositions using separating conjunction without asserting spurious independence relationships.

The most intricate part of Figure 6 is the interpretation of our weakest-precondition modality wp . Intuitively, configurations of the form (γ, 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 $\text{wp}(M, X:A.Q)$ should hold in configuration (γ, 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 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.

2.4 Syntax and Semantics of APPL

Now we establish a program logic that leverages Core Lilac. We fix a small probabilistic programming language called APPL capable of expressing the examples in Section 1.2. The syntax of APPL is given in Figure 7. It is a simply-typed first-order calculus with a sampling operation, immutable arrays, and bounded loops. It has a simple monadic type-system as in Staton [2020]. The important monadic typing rules are:

$$\begin{array}{c}
\frac{}{\Delta \vdash_{\text{APPL}} \text{unif } [\emptyset, 1] : \mathsf{Greal}} \text{T-UNIF} \qquad \frac{\Delta \vdash_{\text{APPL}} M : A}{\Delta \vdash_{\text{APPL}} \text{ret } M : \mathsf{GA}} \text{T-RET} \qquad \frac{\Delta \vdash_{\text{APPL}} M : \mathsf{GA} \quad \Delta, X : A \vdash_{\text{APPL}} N : \mathsf{GB}}{\Delta \vdash_{\text{APPL}} X \leftarrow M; N : \mathsf{GB}} \text{T-BIND}
\end{array}$$

Monadic computations have type GA ; the G stands for the standard Giriy monad [Giry 1982]. The T-UNIF rule states that $\text{unif } [\emptyset, 1]$ is a probabilistic computation producing a real number.

¹⁰As in Jung et al. [2018], we say a proposition P is duplicable if $P \vdash P * P$.

$$\begin{array}{c}
\frac{P \vdash P' \quad Q' \vdash Q \quad \{P'\} M \{X.Q'\}}{\{P\} M \{X.Q\}} \text{H-CONSEQUENCE} \qquad \frac{\{P\} M \{X.Q\}}{\{F * P\} M \{X.F * Q\}} \text{H-FRAME } (X \notin F) \\
\\
\frac{}{\{Q[\llbracket M \rrbracket / X]\} \text{ret } M \{X.Q\}} \text{H-RET} \qquad \frac{\{P\} M \{X.Q\} \quad \forall_{rv} X. \{Q\} N \{Y.R\}}{\{P\} X \leftarrow M; N \{Y.R\}} \text{H-LET} \\
\\
\frac{}{\{\top\} \text{unif } [\emptyset, 1] \{X.X \sim \text{Unif } [0, 1]\}} \text{H-UNIFORM} \qquad \frac{}{\{\top\} \text{flip } p \{X.X \sim \text{Ber } p\}} \text{H-FLIP} \\
\\
\frac{\forall i:\mathbb{N}. \forall_{rv} X:A. \{I(i, X)\} M \{X'.I(i+1, X')\}}{\{I(1, e)\} \text{for } (n, e, i X. M) \{X:A. I(n+1, X)\}} \text{H-FOR} \qquad \frac{\{P\} M \{X.Q(X)\} \quad \forall_{rv} X. \{Q(X)\} N \{Y.R(\text{if } E \text{ then } X \text{ else } Y)\}}{\{P\} \text{if } E \text{ then } M \text{ else } N \{Z.R(Z)\}} \text{H-IF}
\end{array}$$

Fig. 8. Selected proof rules for reasoning about APPL programs.

The semantics for APPL are standard and follow Staton [2020]. Types A are interpreted as measurable spaces $\llbracket A \rrbracket$ and typing contexts $\Delta = \{x_1 : A_1, \dots, x_n : A_n\}$ as products $\llbracket \Delta \rrbracket = \llbracket A_1 \rrbracket \times \dots \times \llbracket A_n \rrbracket$. Programs $\Delta \vdash_{\text{APPL}} M : \mathbb{G}A$ are interpreted as measurable maps $\llbracket M \rrbracket : \llbracket \Delta \rrbracket \xrightarrow{m} \mathcal{G}\llbracket A \rrbracket$. The full semantics can be found in Appendix A.3.

2.5 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. Using the connectives described in Section 2.1, we define the meaning of Hoare triples $\{P\} M \{X.Q\}$ in terms of wp, following Jung et al. [2018].¹¹ Then, we use the model described in Section 2.3 to validate the proof rules in Figure 8; these rules justify the annotated programs given in Section 1.1.

The structural rules H-CONSEQUENCE and H-FRAME are completely standard, as are H-RET and H-LET. The rules H-UNIFORM and H-FLIP specify APPL’s sampling operations; they formalize the intuition that sampling is like allocation. 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. The rule H-IF is used to reason about if-then-else. Unlike in the traditional setting, a probabilistic program can be thought of as taking both branches of an if-then-else, since it is possible that a Boolean random variable is both true and false with nonzero probability. The H-IF rule reflects this: it states that, to establish $R(Z)$, one can first run the **then**-branch to obtain X , and then run the **else**-branch to obtain Y , and then show that R holds of the *random variable* (if E then X else Y) that combines the outcomes of the two branches.

Now we turn our attention to validating these rules with respect to a suitable model. Thus far we have been rather abstract about the ambient sample space Ω underlying Lilac’s semantic model. At this point we make a concrete choice in order to validate the proof rules in Figure 8. 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.

¹¹Concretely, $\{P\} M \{X.Q\} := \Box(P \multimap \text{wp}(\llbracket M \rrbracket, X.Q))$; see Jung et al. [2018] for a detailed explanation.

$$\begin{array}{c}
\text{C-ENTAIL} \\
\frac{P \vdash Q}{\mathbf{C}_{x \leftarrow E} P \vdash \mathbf{C}_{x \leftarrow E} Q} \\
\\
\text{C-INDEP} \\
(\text{own } E) * P \vdash \mathbf{C}_{x \leftarrow E} P \\
\\
\text{C-SUBST} \\
\vdash \mathbf{C}_{x \leftarrow X} (E \stackrel{\text{as}}{=} E[x/X]) \\
\\
\text{C-TOTAL-EXPECTATION} \\
\mathbf{C}_{x \leftarrow X} (\mathbb{E}[E] = e) \wedge \mathbb{E}[e[X/x]] = v \vdash \mathbb{E}[E] = v
\end{array}$$

Fig. 9. Selected properties of Lilac’s conditioning modality \mathbf{C} .

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 [Culpepper and Cobb 2017; Zhang and Amin 2022]. A probability space has finite footprint if it only uses finitely-many dimensions of the Hilbert cube:

DEFINITION 2.6. *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 2.7. *The proof rules in Figure 8 are sound.*

PROOF. The structural rules, H-RET, 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.14. \square

3 THE CONDITIONING MODALITY

So far we have presented Core Lilac, which defines probabilistic interpretations of the standard separation logic connectives, along with atomic propositions for making probability-specific assertions. Now we describe our second main contribution: Lilac’s modal operator for reasoning about conditioning. We extend Core Lilac with the proposition $\mathbf{C}_{x:A \leftarrow E} P$ which states that P holds conditional on the event $E = x$ for all deterministic x . Its typing rule is:

$$\frac{\Gamma; \Delta \vdash_{\text{TV}} E : A \quad \Gamma, x:A; \Delta \vdash P}{\Gamma; \Delta \vdash \mathbf{C}_{x:A \leftarrow E} P} \text{T-C}$$

Figure 9 lists useful laws about \mathbf{C} (proofs are given in Appendix B.20). The rule C-ENTAIL says \mathbf{C} respects entailment; this allows ordinary logical reasoning to be carried out under \mathbf{C} , automatically lifting statements and proofs about unconditional probability to the conditional setting. The rule C-SUBST captures the intuition that X can be safely replaced by x under $\mathbf{C}_{x \leftarrow X}$. The remaining rules express standard facts about conditioning. The rule C-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 \mathbf{C} . The rule C-TOTAL-EXPECTATION states the Law of Total Expectation, a theorem of probability theory that relates an unconditional expectation to an expectation over conditional expectations. As a rule, 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 deterministic 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 $e[X/x]$. Section 4.1 will give an example illustrating this rule's use.

3.1 Semantics of the conditioning modality

The rules stated in Figure 9 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. Intuitively, a model for entering the conditioning modality involves reasoning under a new conditioned space: $(\gamma, D, \mathcal{P}) \models \mathbf{C}_{x:A \leftarrow X} P$ holds if for all $x \in A$ there exists some *conditioned space* $\mathcal{P}_{X=x}$ such that $\gamma, D, \mathcal{P}_{X=x} \models P$. 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 useful for discrete random variables X , where it is practical to disregard conditioned spaces over null events where $\mu(\{X = x\}) = 0$. However, if X is a continuous random variable, then by definition $\mu(\{X = x\}) = 0$ for all x , so these null events cannot be ignored.

In probability theory, *disintegrations* were developed in order to resolve this issue and give a natural notion of conditioned spaces for continuous random variables [Chang and Pollard 1997]. 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 [Chang and Pollard 1997]. 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 [Chang and Pollard 1997]. We leverage this knowledge here to design a model for \mathbf{C} .

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. 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 3.1. *Let $X : [0, 1]^{\mathbb{N}} \rightarrow (A, \mathcal{A})$ 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 [Srivastava 2008] so any probability measure on it is finite Borel; the result follows from Theorem 1.4 of Chang and Pollard [1997]. \square

The class of spaces (A, \mathcal{A}) required by Lemma 3.1 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 (γ, 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 [Ershov 1975]. This motivates the next step in our search for suitably-well-behaved probability spaces:

THEOREM 3.2. *Let $\mathcal{M}_{\text{Borel}}$ 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 2.4 to $\mathcal{M}_{\text{Borel}}$ is still a KRM.*

A proof of this theorem is in Appendix B.6. The upshot of Theorem 3.2 is that configurations of the form (γ, 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 3.1. The final step in our search is motivated by the desire to validate rule C-INDEP in Figure 9. The soundness of C-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 3.3. *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 2.4 to $\mathcal{M}_{\text{disintegrable}}$ is still a KRM.*

For a proof see Appendix B.16. Using Theorem 3.3, we can finally give an interpretation to **C**:

LEMMA 3.4. *The following interpretation of $\mathbf{C}_{x:A \leftarrow E} P$ validates the rules in Figure 9:*

$$\gamma, D, (\mathcal{F}, \mu) \models \mathbf{C}_{x:A \leftarrow E} P \quad \text{iff} \quad \begin{array}{l} \text{for all } (\Sigma_\Omega, \mu') \sqsupseteq (\mathcal{F}, \mu) \\ \text{and all disintegrations of } \mu' \text{ along } E(\gamma) \circ D \text{ into } \{v_x\}_{x \in A}, \\ \text{and almost all } x \in A, \text{ it holds that } (\gamma, x), D, (\mathcal{F}, v_x|_{\mathcal{P}}) \models P. \end{array}$$

For a detailed proof, see Appendix B.20.

4 FURTHER EXAMPLES OF APPLYING LILAC

An essential component of evaluating any new program logic is applying it to validate interesting correctness properties of programs. Our goal in this section is to further establish (1) that Lilac can validate examples that existing probabilistic separation logic approaches can handle [Bao et al. 2021; Barthe et al. 2019]; and (2) give an example that goes beyond these existing approaches.

4.1 Proving a Weighted Sampling Algorithm Correct

To exercise Lilac’s support for conditional reasoning, continuous random variables, and substructural handling of independence, we now prove a sophisticated constant-space *weighted sampling algorithm* correct using Lilac. Suppose you are given a collection of items $\{x_1, \dots, x_n\}$ 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 instance of *reservoir sampling* [Efrimidis and Spirakis 2006].

A naive solution might first normalize the weights so that they sum to 1 and then sample from the resulting probability distribution. Such an approach is 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, Efrimidis and Spirakis [2006] proposed the *constant-space solution* in Figure 10.

The core idea is to generate a value S uniformly at random from $[0, 1]$ on every iteration

```

1  $W \leftarrow \text{ret } [w_1, \dots, w_n];$ 
2  $M \leftarrow \text{ret } (-\infty); K \leftarrow \text{ret } (0);$ 
   // for i from 1 to n with accumulator (M,K),
3 for ( $n, (M, K), i \ (M, K)$ ).
4    $S \leftarrow \text{unif } [0, 1];$ 
5    $U \leftarrow \text{ret } (S \wedge (1/W[i]));$ 
6   if  $U > M$ 
7   then  $\text{ret } (U, i)$ 
8   else  $\text{ret } (M, K)$ 
```

Fig. 10. Constant-space reservoir sampling.

(Line 3), perturb S according to the next weight w_i in the stream (Line 4), and store only the *greatest* perturbed sample (Lines 5–8). It is a surprising fact that this program is equivalent to the naive one. To prove it, we will establish the postcondition $\forall k. \Pr(K = k) = w_k / \sum_j w_j$. First, mechanically applying the rules given in Section 2.5 allows us to conclude the following at exit (for details, which involve a loop invariant, see Appendix E):

$$\exists_{rv} S_1 \dots S_n. \bigstar_i S_i \sim \text{Unif}[0, 1] \quad * \quad K \stackrel{\text{as}}{=} \arg \max_i S_i^{1/w_i} \quad (4)$$

Here $\{S_i\}_i$ are i.i.d. random variables with S_i denoting the value sampled by Line 4 on the i th iteration, and K denotes the final result. The rest of the proof is devoted to showing that (4) entails the desired postcondition. Given arbitrary k , note that $\Pr(K = k) = \Pr(S_k^{1/w_k} > S_j^{1/w_j} \text{ for all } j \neq k)$, since K is defined to be the arg max of j over all S_j^{1/w_j} . To make computing this probability tractable, we condition on S_k : fixing S_k to a deterministic s_k ,

$$\Pr(K = k \mid S_k = s_k) = \Pr(s_k^{1/w_k} > S_j^{1/w_j} \text{ for all } j \neq k) \quad (5)$$

$$= \Pr(s_k^{w_j/w_k} > S_j \text{ for all } j \neq k) \quad \text{Exponentiating} \quad (6)$$

$$= \prod_{j \neq k} \Pr(s_k^{w_j/w_k} > S_j) \quad \text{By conditional independence} \quad (7)$$

From Equation 7 we proceed by calculation. If $U \sim \text{Unif}[0, 1]$, then $\Pr(u > U) = u$; this lets us conclude that (7) $= \prod_{j \neq k} s_k^{w_j/w_k} = \text{pow}\left(s_k, \frac{\sum_{j \neq k} w_j}{w_k}\right)$.

Formally, this calculation occurs under $\mathbf{C}_{s_k \leftarrow S_k}$, which is introduced via C-INDEP. The expression $\Pr(E)$ abbreviates $\mathbb{E}[\mathbb{1}[E]]$, the expectation of the indicator random variable $\mathbb{1}[E]$.¹² The critical step occurs in Equation 7: since $\perp_{j \neq k} S_j \mid S_k$, we can apply:

$$\bigstar_i \text{own } E_i \vdash \Pr\left(\bigcap_i E_i\right) = \prod_i \Pr(E_i), \quad (\text{INDEP-PROD})$$

an immediate consequence of Lemma 2.5.

Finally, to complete the proof we connect the conditional $\Pr(K = k \mid S_k = s_k)$ to the unconditional $\Pr(K = k)$ using the following instantiation of C-TOTAL-EXPECTATION:

$$\mathbf{C}_{s_k \leftarrow S_k} \left(\underbrace{\mathbb{E}[\mathbb{1}[K = k]]}_E = \underbrace{\text{pow}\left(s_k, \frac{\sum_{j \neq k} w_j}{w_k}\right)}_e \wedge \left(\underbrace{\mathbb{E}\left[\text{pow}\left(S_k, \frac{\sum_{j \neq k} w_j}{w_k}\right)\right]}_{e[S_k/S_k]} = \underbrace{\frac{w_k}{\sum_j w_j}}_v \right) \vdash \underbrace{\mathbb{E}[\mathbb{1}[K = k]]}_E = \underbrace{\frac{w_k}{\sum_j w_j}}_v$$

In the left-hand side of this entailment, the first conjunct follows from the above and the second conjunct follows from a calculation. For a detailed presentation of this proof, see Appendix E.

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 informal proofs.

¹²If E is an event then the random variable $\mathbb{1}[E]$ is 1 if E holds and 0 otherwise.

4.2 An Example of Conditional Independence via Control Flow

For this example, we borrow the **CONDSAMPLES** program from Figure 6(b) of Bao et al. [2021] (translated into a functional style):

```

Z ← flip 1/2;

if Z then (X1 ← flip p;
           Y1 ← flip p;
           ret (Z, X1, Y1)) else (X2 ← flip q;
                                   Y2 ← flip q;
                                   ret (Z, X2, Y2))

```

(CONDSAMPLES)

This program produces a tuple (Z, X, Y) with X and Y conditionally independent given Z . The random variables X and Y are sampled from different distributions depending on the outcome Z of a fair coin flip: if $Z = \mathsf{T}$ then X and Y are Bernoulli random variables with parameter p , and if $Z = \mathsf{F}$ then X and Y are Bernoulli random variables with parameter q . The proof of conditional independence, as in the **COMMONCAUSE** example, goes by case analysis on Z .

Conditional independence of X and Y given Z is expressed by the following triple:

$$\{\mathsf{T}\} \text{ CONDSAMPLES } \left\{ (Z, X, Y). \mathbf{C}_{z \leftarrow Z}(\text{own } X * \text{own } Y) \right\}$$

As usual, the proof begins by mechanically applying proof rules. This yields:

$$Z \sim \text{Ber } 1/2 * \exists_{\text{rv}} X_1 Y_1 X_2 Y_2. \left(X_1 \sim \text{Ber } p * Y_1 \sim \text{Ber } p * X_2 \sim \text{Ber } q * Y_2 \sim \text{Ber } q * \right. \\ \left. X \stackrel{\text{as}}{=} (\text{if } Z \text{ then } X_1 \text{ else } X_2) * Y \stackrel{\text{as}}{=} (\text{if } Z \text{ then } Y_1 \text{ else } Y_2) \right)$$

This mechanically-derived postcondition makes use of existential quantification over random variables, written \exists_{rv} , in order to talk about the random variables produced by the **then** and **else** branches. The subformula $X_1 \sim \text{Ber } p * Y_1 \sim \text{Ber } p$ is the postcondition derived for the **then** branch, and the subformula $X_2 \sim \text{Ber } p * Y_2 \sim \text{Ber } p$ is the postcondition derived for the **else** branch. The almost-sure equalities $X \stackrel{\text{as}}{=} (\text{if } Z \text{ then } X_1 \text{ else } X_2)$ and $Y \stackrel{\text{as}}{=} (\text{if } Z \text{ then } Y_1 \text{ else } Y_2)$ combine the variables produced by the individual branches into the variables X and Y produced by the whole if-then-else.

We now proceed as in the **COMMONCAUSE** example. First, we condition on Z and replace all occurrences of Z with the newly introduced deterministic variable z , giving

$$\mathbf{C}_{z \leftarrow Z} \left(\underbrace{\exists_{\text{rv}} X_1 Y_1 X_2 Y_2. \left(X_1 \sim \text{Ber } p * Y_1 \sim \text{Ber } p * X_2 \sim \text{Ber } q * Y_2 \sim \text{Ber } q * \right. \right.}_{P(z)} \\ \left. \left. X \stackrel{\text{as}}{=} (\text{if } z \text{ then } X_1 \text{ else } X_2) * Y \stackrel{\text{as}}{=} (\text{if } z \text{ then } Y_1 \text{ else } Y_2) \right) \right).$$

The goal is to show $\mathbf{C}_{z \leftarrow Z} P(z) \vdash \mathbf{C}_{z \leftarrow Z}(\text{own } X * \text{own } Y)$. Because \mathbf{C} respects entailment, it suffices to show $P(z) \vdash \text{own } X * \text{own } Y$. This follows by a case analysis on z . If $z = \mathsf{T}$ then $P(z)$ can be simplified to $X \sim \text{Ber } p * Y \sim \text{Ber } p$, and if $z = \mathsf{F}$ then $P(z)$ can be simplified to $X \sim \text{Ber } q * Y \sim \text{Ber } q$. In both cases the simplified form entails $\text{own } X * \text{own } Y$ as desired. See Appendix D for a fully annotated program. For more examples of applying Lilac, see Appendix F.

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.

Properties of the conditioning modality. Here we investigate further some formal properties of the conditioning modality. Specifically, we compare \mathbf{C} to modal necessity \Box [Kripke 1972]. The standard properties of \Box are:

- | | |
|--|--|
| (a) If $\vdash P$ then $\vdash \Box P$ (necessitation). | (d) $\Box P \vee \Box Q \vdash \Box(P \vee Q)$. |
| (b) $\Box(P \rightarrow Q) \vdash \Box P \rightarrow \Box Q$ (distribution). | (e) $\Box P \vdash P$ (axiom M). |
| (c) $\Box(P \wedge Q) \dashv\vdash \Box P \wedge \Box Q$. | (f) $\Box P \vdash \Box \Box P$ (axiom 4). |

The modality $\mathbf{C}_{x \leftarrow X}$ satisfies (a)-(d); for proofs see Appendix B.21. The similarity between \mathbf{C} and modal necessity is somewhat expected, due to the similarity in the logical structure of their interpretations: $\mathbf{C}_{x \leftarrow X} P$ 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. 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 .

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 [Jung et al. 2018; Krebbers et al. 2017]. To do so requires expressing Lilac’s KRM as a *camera* [Jung et al. 2018], 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 $\mathcal{P} \bullet Q = \mathcal{R}$ for any 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 2.4 so that one can talk about equality of probability spaces “up to k steps.” We leave these problems for future work.

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. [2007], 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 methodology 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 [1983]. 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.

Program Logics for Probability. The most closely related work is the probabilistic separation logic (PSL) introduced by Barthe et al. [2019], which gives the first separation logic where separating conjunction explicitly models probabilistic independence. Follow-on work extends PSL to support negative dependence [Bao et al. 2022] and to settings beyond probabilistic computation [Zhou et al. 2021]. PSL interprets separating conjunction as a combining operation on distributions over random stores with disjoint domains, over-approximating the semantic notion of probabilistic 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 $\text{own}(X + Y) *$

own($X - Y$) are inexpressible in PSL due to the occurrence of the random variables X and Y on both sides of $*$. PSL also has a frame rule that imposes a number of extra side-conditions capturing data-flow properties of the program. This is in part a consequence of PSL's notion of separation, and in part because PSL programs are written using mutable variables whereas we have preferred to work with a purely functional language. These side-conditions are nontrivial to check and make applying the frame rule cumbersome. Lilac's frame rule is standard for separation logic, and Lilac's interpretation of separating conjunction coincides with probabilistic independence, as demonstrated by Lemma 2.5, and its semantic model is defined in terms of standard objects of probability theory (i.e., probability spaces and random variables). Moreover, Lilac has support for continuous random variables and a modality for reasoning about conditioning; all of these features in combination seem difficult to add to PSL without significant changes to its semantic model. For those interested in a concrete comparison, we have validated three of the five examples presented in Barthe et al. [2019]: one-time pad, private information retrieval, and oblivious transfer; we do not believe the remaining examples exercise Lilac in ways that go beyond the ones we verified. Validating these examples required no changes to Lilac's semantic model; it suffices to extend APPL with support for bitvectors and to import facts about uniformity and independence via a handful of derived rules. For details, see Appendix F.

Bao et al. [2021] 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 conditioning modality, and this extension does not require any changes to the standard model beyond the restriction to well-behaved probability spaces (Theorem 3.3). As a consequence, Lilac behaves very similarly to existing separation logics while still having facilities for handling conditional independence. For those interested in a concrete comparison, the COMMONCAUSE example presented in Section 1.1 gives a Lilac proof of conditional independence for one of the examples from Bao et al. [2021]; Section 4.2 gives a description of the other example.

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 [Barthe et al. 2018]. 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. [2019] 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 probabilistic 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 probabilistic independence. Tassarotti and Harper [2019] 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. [2019] 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.

Expectation-based approaches. Classically the dominant approach to verifying randomized algorithms has been expectation-based techniques such as PDDL [Kozen 1983] and pGCL [Morgan et al. 1996]. 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 [Gretz et al. 2014; Kaminski et al. 2016; Olmedo et al. 2016]. 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 [Barthe et al. 2018].

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 [Bizjak and Birkedal 2015; Culpepper and Cobb 2017; Wand et al. 2018; Zhang and Amin 2022]. Culpepper and Cobb [2017] and Wand et al. [2018] characterize contextual equivalence of a language with higher-order functions, continuous random variables, and scoring via a biorthogonal logical relation. Zhang and Amin [2022] 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.

Probabilistic Denotational Semantics. An entirely separate method for verifying properties of probabilistic programs seeks to associate programs with a well-behaved mathematical model and perform all reasoning in the model. For example, Staton [2017] 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 [Fritz 2020; Heunen et al. 2017; Staton et al. 2016; Stein 2021]. 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 Girmonad 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 [Heunen et al. 2017] 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 probabilistic 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 conditioning 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. [2021], or if it can handle the exotic forms of negative dependence studied in Bao et al. [2022].

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their careful feedback and suggestions. This work was supported by the National Science Foundation under Grant No. #CCF-2220408.

REFERENCES

- 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.
- 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.
- 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.
- 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>
- 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.
- 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.
- Aleš Bizjak and Lars Birkedal. 2015. Step-indexed logical relations for probability. In *Foundations of Software Science and Computation Structures: 18th International Conference, FOSSACS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11–18, 2015, Proceedings* 18. Springer, 279–294.
- Joseph T Chang and David Pollard. 1997. Conditioning as disintegration. *Statistica Neerlandica* 51, 3 (1997), 287–317.
- 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.
- Dino Distefano, Manuel Fähndrich, Francesco Logozzo, and Peter W O'Hearn. 2019. Scaling static analyses at Facebook. *Commun. ACM* 62, 8 (2019), 62–70.
- Pavlos S Efrimidis and Paul G Spirakis. 2006. Weighted random sampling with a reservoir. *Information processing letters* 97, 5 (2006), 181–185.
- MP Ershov. 1975. Extension of measures and stochastic equations. *Theory of Probability & Its Applications* 19, 3 (1975), 431–444.
- Tobias Fritz. 2020. A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics. *Advances in Mathematics* 370 (2020), 107239.
- 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.
- 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.

- Michele Giry. 1982. A categorical approach to probability theory. In *Categorical aspects of topology and analysis*. Springer, 68–85.
- 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.
- David Harel, Dexter Kozen, and Jerzy Tiuryn. 2001. Dynamic logic. In *Handbook of philosophical logic*. Springer, 99–217.
- 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.
- 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.
- Samin S Ishtiaq and Peter W O’Hearn. 2001. BI as an assertion language for mutable data structures. In *Proceedings of the 28th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. 14–26.
- 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).
- Olav Kallenberg. 1997. *Foundations of modern probability*. Vol. 2. Springer.
- 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.
- Dexter Kozen. 1983. A probabilistic pdl. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*. 291–297.
- 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.
- Saul A Kripke. 1972. Naming and necessity. In *Semantics of natural language*. Springer, 253–355.
- Edward Ashford Lee and Sanjit Arunkumar Seshia. 2016. *Introduction to embedded systems: A cyber-physical systems approach*. Mit Press.
- Carroll Morgan, Annabelle McIver, and Karen Seidel. 1996. Probabilistic predicate transformers. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 18, 3 (1996), 325–353.
- Peter W O’Hearn. 2012. A Primer on Separation Logic (and Automatic Program Verification and Analysis). *Software safety and security* 33 (2012), 286–318.
- 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.
- 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.
- John C Reynolds. 2002. Separation logic: A logic for shared mutable data structures. In *Proceedings 17th Annual IEEE Symposium on Logic in Computer Science*. IEEE, 55–74.
- John C Reynolds. 2009. An introduction to separation logic. In *Engineering Methods and Tools for Software Safety and Security*. IOS Press, 285–310.
- 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.
- 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.
- Sashi Mohan Srivastava. 2008. *A course on Borel sets*. Vol. 180. Springer Science & Business Media.
- Sam Staton. 2017. Commutative semantics for probabilistic programming. In *European Symposium on Programming*. Springer, 855–879.
- Sam Staton. 2020. Probabilistic programs as measures. *Foundations of Probabilistic Programming* (2020), 43.
- 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.
- Dario Maximilian Stein. 2021. Structural foundations for probabilistic programming languages. *University of Oxford* (2021).
- Terence Tao. 2015. 254A, notes 0: A review of probability theory. <https://terrytao.wordpress.com/2010/01/01/254a-notes-0-a-review-of-probability-theory/>
- 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.
- 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.

- 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.
- 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

$$\begin{aligned}
 p &\in [0, 1] \\
 r &\in \mathbb{R} \\
 n &\in \mathbb{N} \\
 \oplus &\in \text{Arith} := \{+, -, \times, /, ^\wedge\} \\
 < &\in \text{Cmp} := \{<, \leq, =\} \\
 A, B &::= A \times B \mid \text{bool} \mid \text{real} \mid A^n \mid \text{index} \mid \text{GA} \\
 M, N, O &::= X \mid \text{ret } M \mid X \leftarrow M; N \mid \\
 &\quad (M, N) \mid \text{fst } M \mid \text{snd } M \mid \\
 &\quad \text{T} \mid \text{F} \mid \text{if } M \text{ then } N \text{ else } O \mid \text{flip } p \mid \\
 &\quad r \mid M \oplus N \mid M < N \mid \text{unif } [\emptyset, 1] \mid \\
 &\quad [M, \dots, M] \mid M[N] \mid \text{for}(n, M_{\text{init}}, i \ X. \ M_{\text{step}})
 \end{aligned}$$

A.2 Typing

$$\begin{array}{c}
 \frac{\Gamma, X : A \vdash_{\text{APPL}} X : A}{\Gamma \vdash_{\text{APPL}} \text{ret } M : \text{GA}} \quad \frac{\Gamma \vdash_{\text{APPL}} M : \text{GA} \quad \Gamma, X : A \vdash_{\text{APPL}} N : \text{GB}}{\Gamma \vdash_{\text{APPL}} X \leftarrow M; N : \text{GB}} \\
 \\
 \frac{\Gamma \vdash_{\text{APPL}} M : A \quad \Gamma \vdash_{\text{APPL}} N : B}{\Gamma \vdash_{\text{APPL}} (M, N) : A \times B} \quad \frac{\Gamma \vdash_{\text{APPL}} M : A \times B}{\Gamma \vdash_{\text{APPL}} \text{fst } M : A} \quad \frac{\Gamma \vdash_{\text{APPL}} M : A \times B}{\Gamma \vdash_{\text{APPL}} \text{snd } M : B} \quad \Gamma \vdash_{\text{APPL}} \text{T} : \text{bool} \\
 \\
 \frac{\Gamma \vdash_{\text{APPL}} \text{F} : \text{bool} \quad \frac{\Gamma \vdash_{\text{APPL}} M : \text{bool} \quad \Gamma \vdash_{\text{APPL}} N : A \quad \Gamma \vdash_{\text{APPL}} O : A}{\Gamma \vdash_{\text{APPL}} \text{if } M \text{ then } N \text{ else } O : A}}{\Gamma \vdash_{\text{APPL}} \text{flip } p : \text{Gbool}} \\
 \\
 \frac{\Gamma \vdash_{\text{APPL}} r : \text{real} \quad \frac{\Gamma \vdash_{\text{APPL}} M : \text{real} \quad \Gamma \vdash_{\text{APPL}} N : \text{real}}{\Gamma \vdash_{\text{APPL}} M \oplus N : \text{real}}}{\Gamma \vdash_{\text{APPL}} M < N : \text{bool}} \\
 \\
 \frac{\Gamma \vdash_{\text{APPL}} \text{unif } [\emptyset, 1] : \text{Greal} \quad \frac{\Gamma \vdash_{\text{APPL}} M_k : A \text{ for all } 1 \leq k \leq n}{\Gamma \vdash_{\text{APPL}} [M_1, \dots, M_n] : A^n}}{\Gamma \vdash_{\text{APPL}} M[N] : A} \quad \frac{\Gamma \vdash_{\text{APPL}} M_{\text{init}} : A \quad \Gamma, i : \text{index}, X : A \vdash_{\text{APPL}} M_{\text{step}} : \text{GA}}{\Gamma \vdash_{\text{APPL}} \text{for}(n, M_{\text{init}}, i \ X. \ M_{\text{step}}) : \text{GA}}
 \end{array}$$

A.3 Semantics

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

$$\begin{aligned}
\llbracket A \times B \rrbracket &= \llbracket A \rrbracket \otimes \llbracket B \rrbracket \\
\llbracket \text{bool} \rrbracket &= (\{\mathsf{T}, \mathsf{F}\}, \mathcal{P}(\{\mathsf{T}, \mathsf{F}\})) \\
\llbracket \text{real} \rrbracket &= (\mathbb{R}, \mathcal{B}(\mathbb{R})) \\
\llbracket A^n \rrbracket &= \underbrace{\llbracket A \rrbracket \otimes \cdots \otimes \llbracket A \rrbracket}_{n \text{ times}} \\
\llbracket \text{index} \rrbracket &= (\mathbb{N}, \mathcal{P}(\mathbb{N})) \\
\llbracket \mathcal{G}A \rrbracket &= \mathcal{G}\llbracket A \rrbracket \\
\llbracket \cdot \rrbracket &= \text{the one-point space} \\
\llbracket \Gamma, X:A \rrbracket &= \llbracket \Gamma \rrbracket \otimes \llbracket A \rrbracket
\end{aligned}
\quad
\begin{aligned}
x \llbracket + \rrbracket y &= x + y \\
x \llbracket - \rrbracket y &= x - y \\
x \llbracket \times \rrbracket y &= xy \\
x \llbracket / \rrbracket y &= \begin{cases} x/y, & y \neq 0 \\ 0, & \text{otherwise} \end{cases} \\
x \llbracket ^ \rrbracket y &= \begin{cases} x^y, & x > 0 \\ 0, & \text{otherwise} \end{cases}
\end{aligned}
\quad
\begin{aligned}
x \llbracket < \rrbracket y &= \begin{cases} \mathsf{T}, & x < y \\ \mathsf{F}, & \text{otherwise} \end{cases} \\
x \llbracket \leq \rrbracket y &= \begin{cases} \mathsf{T}, & x \leq y \\ \mathsf{F}, & \text{otherwise} \end{cases} \\
x \llbracket = \rrbracket y &= \begin{cases} \mathsf{T}, & x = y \\ \mathsf{F}, & \text{otherwise} \end{cases}
\end{aligned}$$

Terms $\Gamma \vdash M : A$ are interpreted as maps $\llbracket M \rrbracket : \llbracket \Gamma \rrbracket \xrightarrow{m} \llbracket A \rrbracket$.

$$\begin{aligned}
\llbracket X \rrbracket \rho &= \rho(X) \\
\llbracket \text{ret } M \rrbracket \rho &= \text{ret } \llbracket M \rrbracket \rho \\
\llbracket X \leftarrow M; N \rrbracket \rho &= v \leftarrow \llbracket M \rrbracket \rho; \llbracket N \rrbracket \rho[X \mapsto v] \\
\llbracket (M, N) \rrbracket \rho &= (\llbracket M \rrbracket \rho, \llbracket N \rrbracket \rho) \\
\llbracket \text{fst } M \rrbracket \rho &= \pi_1(\llbracket M \rrbracket \rho) \\
\llbracket \text{snd } M \rrbracket \rho &= \pi_2(\llbracket M \rrbracket \rho) \\
\llbracket \mathsf{T} \rrbracket \rho &= \mathsf{T} \\
\llbracket \mathsf{F} \rrbracket \rho &= \mathsf{F} \\
\llbracket \text{if } M \text{ then } N \text{ else } O \rrbracket \rho &= \begin{cases} \llbracket N \rrbracket \rho, & \llbracket M \rrbracket \rho = \mathsf{T} \\ \llbracket O \rrbracket \rho, & \llbracket M \rrbracket \rho = \mathsf{F} \end{cases} \\
\llbracket \text{flip } p \rrbracket \rho &= \text{Ber } p \\
\llbracket r \rrbracket \rho &= r \\
\llbracket M \oplus N \rrbracket \rho &= (\llbracket M \rrbracket \rho) \llbracket \oplus \rrbracket (\llbracket N \rrbracket \rho) \\
\llbracket \text{unif } [\emptyset, 1] \rrbracket \rho &= \text{Unif } [0, 1] \\
\llbracket [M_1, \dots, M_n] \rrbracket \rho &= (\llbracket M_1 \rrbracket \rho, \dots, \llbracket M_n \rrbracket \rho) \\
\llbracket M[N] : A \rrbracket \rho &= \begin{cases} \pi_{\llbracket N \rrbracket \rho}(\llbracket M \rrbracket \rho), & 1 \leq \llbracket N \rrbracket \rho \leq n \\ \text{arbitrary}(A), & \text{otherwise} \end{cases} \\
\llbracket \text{for}(n, M_i, i X. M_s) \rrbracket \rho &= \text{loop}(1, \llbracket M_i \rrbracket \rho, \lambda k v. \llbracket M_s \rrbracket \rho[i \mapsto k, X \mapsto v]) \\
\text{where loop}(k, v, f) &= \begin{cases} \text{ret } v, & k > n \\ v' \leftarrow f(k, v); \text{loop}(k+1, v', f), & \text{otherwise} \end{cases}
\end{aligned}$$

To make array indexing total, $\text{arbitrary}(A)$ produces an arbitrary inhabitant of $\llbracket A \rrbracket$.

$$\begin{aligned} \text{arbitrary}(A \times B) &= (\text{arbitrary}(A), \text{arbitrary}(B)) \\ \text{arbitrary}(\text{bool}) &= \top \\ \text{arbitrary}(\text{real}) &= 0 \\ \text{arbitrary}(A^n) &= \underbrace{(\text{arbitrary}(A), \dots, \text{arbitrary}(A))}_{n \text{ times}} \\ \text{arbitrary}(\text{index}) &= 0 \end{aligned}$$

B SYNTAX AND SEMANTICS OF LILAC

B.1 Syntax

$$\begin{aligned} S, T &\in \text{Set} \\ A, B &\in \text{Meas} \\ P, Q &::= \top \mid \perp \mid P \wedge Q \mid P \vee Q \mid P \rightarrow Q \mid \\ &\quad P * Q \mid P \multimap Q \mid \Box P \mid \\ &\quad \forall x:S.P \mid \exists x:S.P \mid \forall_{\text{rv}} X:A.P \mid \exists_{\text{rv}} X:A.P \mid \\ &\quad E \sim \mu \mid \text{own } E \mid E \stackrel{\text{as}}{=} E \mid \mathbb{E}[E] = e \mid \text{wp}(M, X:A.Q) \end{aligned}$$

B.2 Typing

$$\begin{aligned} \Gamma &::= \cdot \mid \Gamma, x : S \\ \Delta &::= \cdot \mid \Delta, X : A \\ \llbracket x_1 : S_1, \dots, x_n : S_n \rrbracket &= S_1 \times \dots \times S_n \\ \llbracket X_1 : A_1, \dots, X_n : A_n \rrbracket &= A_1 \otimes \dots \otimes A_n \\ \frac{E \in \llbracket \Gamma \rrbracket \rightarrow \llbracket \Delta \rrbracket \xrightarrow{m} A}{\Gamma; \Delta \vdash_{\text{rv}} E : A} & \quad \frac{e \in \llbracket \Gamma \rrbracket \rightarrow A}{\Gamma \vdash_{\text{det}} e : A} & \quad \frac{\mu \in \llbracket \Gamma \rrbracket \rightarrow \mathcal{G}A}{\Gamma \vdash_{\text{det}} \mu : A} & \quad \frac{M \in \llbracket \Gamma \rrbracket \rightarrow \llbracket \Delta \rrbracket \xrightarrow{m} \mathcal{G}A}{\Gamma; \Delta \vdash_{\text{prog}} M : A} \\ \Gamma; \Delta \vdash \top & \quad \Gamma; \Delta \vdash \perp & \quad \frac{\Gamma; \Delta \vdash P \quad \Gamma; \Delta \vdash Q}{\Gamma; \Delta \vdash P \wedge Q} & \quad \frac{\Gamma; \Delta \vdash P \quad \Gamma; \Delta \vdash Q}{\Gamma; \Delta \vdash P \vee Q} & \quad \frac{\Gamma; \Delta \vdash P \quad \Gamma; \Delta \vdash Q}{\Gamma; \Delta \vdash P \rightarrow Q} \\ \frac{\Gamma; \Delta \vdash P \quad \Gamma; \Delta \vdash Q}{\Gamma; \Delta \vdash P * Q} & \quad \frac{\Gamma; \Delta \vdash P \quad \Gamma; \Delta \vdash Q}{\Gamma; \Delta \vdash P \multimap Q} & \quad \frac{\Gamma; \Delta \vdash P}{\Gamma; \Delta \vdash \Box P} \\ \frac{\Gamma, x:S; \Delta \vdash P}{\Gamma; \Delta \vdash \forall x:S.P} & \quad \frac{\Gamma, x:S; \Delta \vdash P}{\Gamma; \Delta \vdash \exists x:S.P} & \quad \frac{\Gamma; \Delta, X:A \vdash P}{\Gamma; \Delta \vdash \forall_{\text{rv}} X:A.P} & \quad \frac{\Gamma; \Delta, X:A \vdash P}{\Gamma; \Delta \vdash \exists_{\text{rv}} X:A.P} \\ \frac{\Gamma; \Delta \vdash_{\text{rv}} E : A \quad \Gamma \vdash_{\text{det}} \mu : A}{\Gamma; \Delta \vdash E \sim \mu} & \quad \frac{\Gamma; \Delta \vdash_{\text{rv}} E : A}{\Gamma; \Delta \vdash \text{own } E} & \quad \frac{\Gamma; \Delta \vdash_{\text{rv}} E_1 : A \quad \Gamma; \Delta \vdash_{\text{rv}} E_2 : A}{\Gamma; \Delta \vdash E_1 \stackrel{\text{as}}{=} E_2} \\ \frac{\Gamma; \Delta \vdash_{\text{rv}} E : \mathbb{R} \quad \Gamma \vdash_{\text{det}} e : \mathbb{R}}{\Gamma; \Delta \vdash \mathbb{E}[E] = e} & \quad \frac{\Gamma; \Delta \vdash_{\text{prog}} M : A \quad \Gamma; \Delta, X:A \vdash Q}{\Gamma; \Delta \vdash \text{wp}(M, X:A.Q)} \end{aligned}$$

B.3 Independent combination of probability spaces

LEMMA B.1 (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* [Kallenberg 1997]. 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 $\mu(E)\nu(F)$. \square

LEMMA B.2. *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 $F \in \mathcal{F}, G \in \mathcal{G}$. \square

LEMMA B.3. *If (\mathcal{F}, μ) is a probability space then $\mathcal{F}^\perp := \{E \mid E \perp \mathcal{F}\}^{13}$ 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. \square

THEOREM B.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 $\mu = \nu|_{\mathcal{F}}$.¹⁴ The tuple $(\mathcal{M}, \sqsubseteq, \bullet, \mathbf{1})$ is a Kripke resource monoid, where $\mathbf{1}$ is the trivial probability space (\mathcal{F}_1, μ_1) with $\mathcal{F}_1 = \{\emptyset, \Omega\}$ and $\mu_1(\Omega) = 1$.*

PROOF. $\mathbf{1}$ is indeed a unit: if (\mathcal{F}, μ) is some other probability space on Ω then $\langle \mathcal{F}, \mathcal{F}_1 \rangle = \mathcal{F}$ and μ witnesses the independent combination of itself with μ_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, μ_2) and (\mathcal{F}_3, μ_3) .
- Some $\mu_{1(23)}$ witnesses the combination of (\mathcal{F}_1, μ_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, μ_2) and (\mathcal{F}_3, μ_3) .
- (3) Define $\mu_{1(23)} := \mu_{(12)3}$. This is a witness for (\mathcal{F}_1, μ_1) and $(\mathcal{F}_{23}, \mu_{23})$.

¹³ $E \perp \mathcal{F}$ iff $\mu(E \cap F) = \mu(E)\mu(F)$ for all $F \in \mathcal{F}$.

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.2). 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.3.

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 $\rho(F \cap G) = \rho'(F \cap G) = \rho'(F)\rho'(G) = \rho(F)\rho(G)$ 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. \square

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

LEMMA B.7. 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.4 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. \square

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

LEMMA B.8. The following interpretations of basic connectives is well-formed:

$\gamma, D, \mathcal{P} \models \top$	<i>always</i>
$\gamma, D, \mathcal{P} \models \perp$	<i>never</i>
$\gamma, D, \mathcal{P} \models P \wedge Q$	<i>iff</i> $\gamma, D, \mathcal{P} \models P$ and $\gamma, D, \mathcal{P} \models Q$
$\gamma, D, \mathcal{P} \models P \vee Q$	<i>iff</i> $\gamma, D, \mathcal{P} \models P$ or $\gamma, D, \mathcal{P} \models Q$
$\gamma, D, \mathcal{P} \models P \rightarrow Q$	<i>iff</i> $\gamma, D, \mathcal{P}' \models P$ implies $\gamma, D, \mathcal{P}' \models Q$ for all $\mathcal{P}' \sqsupseteq \mathcal{P}$
$\gamma, D, \mathcal{P} \models P * Q$	<i>iff</i> $\gamma, D, \mathcal{P}_P \models P$ and $\gamma, D, \mathcal{P}_Q \models Q$ for some $\mathcal{P}_P \bullet \mathcal{P}_Q \sqsubseteq \mathcal{P}$
$\gamma, D, \mathcal{P} \models P \multimap Q$	<i>iff</i> $\gamma, D, \mathcal{P}_P \models P$ implies $\gamma, D, \mathcal{P}_P \bullet \mathcal{P} \models Q$ for all \mathcal{P}_P with $\mathcal{P}_P \bullet \mathcal{P}$ defined
$\gamma, D, \mathcal{P} \models \Box P$	<i>iff</i> $\gamma, D, 1 \models P$
$\gamma, D, \mathcal{P} \models \forall x:S.P$	<i>iff</i> $(\gamma, x), D, \mathcal{P} \models P$ for all $x \in S$
$\gamma, D, \mathcal{P} \models \exists x:S.P$	<i>iff</i> $(\gamma, x), D, \mathcal{P} \models P$ for some $x \in S$
$\gamma, D, \mathcal{P} \models \forall_{\text{rv}} X:A.P$	<i>iff</i> $\gamma, (D, X), \mathcal{P} \models P$ for all $X : \text{RV } A$
$\gamma, D, \mathcal{P} \models \exists_{\text{rv}} X:A.P$	<i>iff</i> $\gamma, (D, X), \mathcal{P} \models P$ for some $X : \text{RV } A$
$\gamma, D, (\mathcal{F}, \mu) \models E \sim \mu'$	<i>iff</i> $E(\gamma) \circ D$ is \mathcal{F} -measurable and $\mu'(\gamma) = \begin{pmatrix} \omega \leftarrow \mu; \\ \text{ret } (E(\gamma)(D(\omega))) \end{pmatrix}$
$\gamma, D, (\mathcal{F}, \mu) \models \text{own } E$	<i>iff</i> $E(\gamma) \circ D$ is \mathcal{F} -measurable
$\gamma, D, (\mathcal{F}, \mu) \models E_1 \stackrel{\text{as}}{=} E_2$	<i>iff</i> $(E_1(\gamma) \circ D) _F = (E_2(\gamma) \circ D) _F$ for some $F \in \mathcal{F}$ with $\mu(F) = 1$
$\gamma, D, (\mathcal{F}, \mu) \models \mathbb{E}[E] = e$	<i>iff</i> $E(\gamma) \circ D$ is \mathcal{F} -measurable and $\mathbb{E}_{\omega \sim \mu}[E(\gamma)(D(\omega))] = e(\gamma)$
$\gamma, D, \mathcal{P} \models \text{wp}(M, X:A.Q)$	<i>iff</i> for all $\mathcal{P}_{\text{frame}}$ and μ with $\mathcal{P}_{\text{frame}} \bullet \mathcal{P} \sqsubseteq (\Sigma_\Omega, \mu)$ and all $D_{\text{ext}} : \text{RV } \llbracket \Delta_{\text{ext}} \rrbracket$ there exists $X : \text{RV } A$ and \mathcal{P}' and μ' with $\mathcal{P}_{\text{frame}} \bullet \mathcal{P}' \sqsubseteq (\Sigma_\Omega, \mu')$ such that $\begin{pmatrix} \omega \leftarrow \mu; \\ v \leftarrow M(\gamma)(D(\omega)); \\ \text{ret } (D_{\text{ext}}(\omega), D(\omega), v) \end{pmatrix} = \begin{pmatrix} \omega \leftarrow \mu'; \\ \text{ret } (D_{\text{ext}}(\omega), D(\omega), X(\omega)) \end{pmatrix}$ and $\gamma, (D, X), \mathcal{P}' \models Q$

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

LEMMA B.9 (SEPARATING CONJUNCTION IS MUTUAL INDEPENDENCE). *Fix a configuration (γ, D, \mathcal{P}) . Abbreviating $X_i(\gamma) \circ D$ as X'_i , random variables X'_1, \dots, X'_n are mutually independent with respect to \mathcal{P} iff $\gamma, D, \mathcal{P} \models \text{own } X_1 * \dots * \text{own } X_n$.*

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

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

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

For the converse, suppose X'_1, \dots, X'_n are mutually independent with respect to some probability space \mathcal{P} . For each $1 \leq i \leq n$, let \mathcal{P}_i be the probability space (\mathcal{F}_i, μ_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 $\mathcal{P}_1 \bullet \dots \bullet \mathcal{P}_n$ is defined, as then $\mathcal{P}_1 \bullet \dots \bullet \mathcal{P}_n \sqsubseteq \mathcal{P}$ by lemma 2.3. This follows by induction on n . Cases $n = 0$ and $n = 1$ are immediate. Now suppose $\mathcal{P}_1 \bullet \dots \bullet \mathcal{P}_k$ is defined. It's straightforward to show that $\mathcal{P}_1 \bullet \dots \bullet \mathcal{P}_k$ is the pullback of the random variable (X'_1, \dots, X'_k) , and \mathcal{P}_{k+1} is the pullback of X'_{k+1} by definition. Mutual independence of $X'_1, \dots, X'_k, X'_{k+1}$ implies independence of (X'_1, \dots, X'_k) and X'_{k+1} : intersections of events $X_1'^{-1}(E_1) \cap \dots \cap X_k'^{-1}(E_k)$ form a π -system that generates $\mathcal{F}_1 \bullet \dots \bullet \mathcal{F}_k$, events independent of \mathcal{F}_{k+1} with respect to μ form a λ -system, and each

intersection $X_1'^{-1}(E_1) \cap \dots \cap X_k'^{-1}(E_k)$ is independent of \mathcal{F}_{k+1} because $X_1', \dots, X_k', X_{k+1}'$ are mutually independent. Thus the pullback of $(X_1', \dots, X_k', X_{k+1}')$ is an independent combination of $\mathcal{P}_1 \bullet \dots \bullet \mathcal{P}_k$ and \mathcal{P}_{k+1} , and $\mathcal{P}_1 \bullet \dots \bullet \mathcal{P}_k \bullet \mathcal{P}_{k+1}$ is defined. This closes the induction, so $\gamma, D, \mathcal{P} \models \text{own } X_1' \bullet \dots \bullet X_n'$ as desired. \square

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

$$\frac{s \in \llbracket \Gamma' \rrbracket \rightarrow \llbracket \Gamma \rrbracket \quad S \in \llbracket \Delta' \rrbracket \xrightarrow{m} \llbracket \Delta \rrbracket}{\Gamma'; \Delta' \vdash (s, S) : \Gamma; \Delta}$$

$$\frac{\Gamma; \Delta \vdash_{\text{rv}} E : A \quad \Gamma'; \Delta' \vdash (s, S) : \Gamma; \Delta}{\Gamma'; \Delta' \vdash E[s, S] : A}$$

$$E[s, S](\gamma) = E(s(\gamma)) \circ S$$

$$\frac{\Gamma \vdash_{\text{det}} e : A \quad \Gamma' \vdash s : \Gamma}{\Gamma' \vdash e[s] : A}$$

$$e[s] = e \circ s$$

$$\frac{\Gamma \vdash_{\text{det}} \mu : A \quad \Gamma' \vdash s : \Gamma}{\Gamma' \vdash \mu[s] : A}$$

$$\mu[s] = \mu \circ s$$

$$\frac{\Gamma; \Delta \vdash_{\text{prog}} M : A \quad \Gamma'; \Delta' \vdash (s, S) : \Gamma; \Delta}{\Gamma'; \Delta' \vdash M[s, S] : A}$$

$$M[s, S](\gamma) = M(s(\gamma)) \circ S$$

$$\top[s, S] = \top$$

$$\perp[s, S] = \perp$$

$$(P \wedge Q)[s, S] = (P[s, S] \wedge Q[s, S])$$

$$(P \vee Q)[s, S] = (P[s, S] \vee Q[s, S])$$

$$(P \rightarrow Q)[s, S] = (P[s, S] \rightarrow Q[s, S])$$

$$(P * Q)[s, S] = (P[s, S] * Q[s, S])$$

$$(P \multimap Q)[s, S] = (P[s, S] \multimap Q[s, S])$$

$$(\Box P)[s, S] = \Box P[s, S]$$

$$(\forall x:T.P)[s, S] = \forall x:T.P[s \times 1_T, S]$$

$$(\exists x:T.P)[s, S] = \exists x:T.P[s \times 1_T, S]$$

$$(\forall_{\text{rv}} X:A.P)[s, S] = \forall_{\text{rv}} X:A.P[s, S \times 1_A]$$

$$(\exists_{\text{rv}} X:A.P)[s, S] = \exists_{\text{rv}} X:A.P[s, S \times 1_A]$$

$$(E \sim \mu)[s, S] = E[s, S] \sim \mu[s]$$

$$(\text{own } E)[s, S] = \text{own } E[s, S]$$

$$(E_1 \stackrel{\text{as}}{=} E_2)[s, S] = E_1[s, S] \stackrel{\text{as}}{=} E_2[s, S]$$

$$(\mathbb{E}[E] = e)[s, S] = \mathbb{E}[E[s, S]] = e[s]$$

$$\text{wp}(M, X:A.Q)[s, S] = \text{wp}(M[s, S], X:A.Q[s, S \times 1_A])$$

LEMMA B.10 (SYNTACTIC AND SEMANTIC SUBSTITUTION COINCIDE). $\gamma, D, \mathcal{P} \models P[s, S]$ iff $s(\gamma), S \circ D, \mathcal{P} \models P$.

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

- Case $E \sim \mu$:

$$\gamma, D, \mathcal{P} \models (E \sim \mu)[s, S]$$

$$\text{iff } \gamma, D, \mathcal{P} \models ((\gamma \mapsto E(\gamma) \circ S) \sim (\mu \circ S))$$

$$\text{iff } E(s(\gamma)) \circ S \circ D \text{ is } \mathcal{P}\text{-measurable and } \mu(\gamma) = \left(\begin{array}{l} \omega \leftarrow \mathcal{P} \\ \text{ret } E(\gamma)(S(D(\omega))) \end{array} \right)$$

$$\text{iff } s(\gamma), S \circ D, \mathcal{P} \models E \sim \mu$$

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

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

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

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

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

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

so that (\mathcal{P}', μ', X) witnesses $s(\gamma), S \circ D, \mathcal{P} \models \text{wp}(M, X:A.Q)$ as desired.

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

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

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

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

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

□

B.5 Derived rules

LEMMA B.11. *The following structural rules hold:*

$$\begin{array}{c}
 \text{H-CONSEQUENCE} \\
 \frac{P \vdash Q}{\text{wp}(M, X.P) \vdash \text{wp}(M, X.Q)} \\
 \\
 \text{H-DISJUNCTION} \\
 \text{wp}(M, X.P) \vee \text{wp}(M, X.Q) \vdash \text{wp}(M, X.P \vee Q) \\
 \\
 \text{H-FRAME} \\
 F * \text{wp}(M, X.Q) \vdash \text{wp}(M, X.F * Q) \quad (X \notin F)
 \end{array}$$

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

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

and $\gamma, (D, X), \mathcal{P}' \models Q$. Since $X \notin F$, $F[\text{weak}_X] = F$ so $\gamma, (D, X), \mathcal{P}_F \models F$ by lemma B.10. And since the composition $(\mathcal{P}_{\text{frame}} \bullet \mathcal{P}_F) \bullet \mathcal{P}'$ is defined, the composition $\mathcal{P}_F \bullet \mathcal{P}'$ must be as well, so $\gamma, (D, X), \mathcal{P}_F \bullet \mathcal{P}' \models F * Q$ as desired. \square

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

$$\begin{aligned}
 Q[e/X] \vdash \text{wp}(\text{ret } e, X.Q) \quad & \text{wp}(M, X.\text{wp}(N, Y.Q)) \vdash \text{wp}((X \leftarrow M; N), Y.Q) \\
 (\forall_{\text{rv}} X:A. X \sim \text{Unif}[0, 1] \multimap Q) \vdash & \text{wp}(\text{Unif}[0, 1], X:A.Q) \\
 (\forall_{\text{rv}} X:A. X \sim \text{Ber } p \multimap Q) \vdash & \text{wp}(\text{Ber } p, X:A.Q)
 \end{aligned}$$

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

$$\text{wp}(M, X:A. \text{wp}(N, Y:A. Q(\text{if } E \text{ then } X \text{ else } Y))) \vdash \text{wp}(\text{if } E \text{ then } M \text{ else } N, X:A. Q)$$

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

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

PROOF.

- Ret: suppose $\gamma, D, \mathcal{P} \models Q[e(\gamma)/X]$. By lemma B.10 this is equivalent to $\gamma, (D, e(\gamma)), \mathcal{P} \models Q$. To show $\text{wp}(\text{ret } e, X.Q)$ suppose $\mathcal{P}_{\text{frame}} \bullet \mathcal{P} \sqsubseteq (\Sigma_\Omega, \mu)$ and $D_{\text{ext}} : \text{RV} \llbracket \Delta_{\text{ext}} \rrbracket$. Choose $\mathcal{P}' := \mathcal{P}$ and $\mu' := \mu$ and $X(\omega) := e(\gamma)$. Then

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

and $\gamma, (D, e(\gamma)), \mathcal{P} \models Q$ as desired.

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

$$\left(\begin{array}{l} \omega \leftarrow \mu; \\ x \leftarrow M(\gamma)(D(\omega)); \\ \text{ret } (D_{\text{ext}}(\omega), D(\omega), x) \end{array} \right) = \left(\begin{array}{l} \omega \leftarrow \mu_X; \\ \text{ret } (D_{\text{ext}}(\omega), D(\omega), X(\omega)) \end{array} \right) \quad (8)$$

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

$$\left(\begin{array}{l} \omega \leftarrow \mu_X; \\ y \leftarrow N(\gamma)((D, X)(\omega)); \\ \text{ret } (D_{\text{ext}}(\omega), (D, X)(\omega), y) \end{array} \right) = \left(\begin{array}{l} \omega \leftarrow \mu_Y; \\ \text{ret } (D_{\text{ext}}(\omega), (D, X)(\omega), Y(\omega)) \end{array} \right) \quad (9)$$

and $\gamma, (D, X, Y), \mathcal{P}_Y \models Q$. Since $X \notin Q$, this implies $\gamma, (D, Y), \mathcal{P}_Y \models Q$ by B.10, so it only remains to show

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

Calculate:

$$\begin{aligned} & \left(\begin{array}{l} \omega \leftarrow \mu; \\ y \leftarrow (X \leftarrow M; N)(\gamma)(D(\omega)); \\ \text{ret } (D_{\text{ext}}(\omega), D(\omega), v) \end{array} \right) = \left(\begin{array}{l} \omega \leftarrow \mu; \\ x \leftarrow M(\gamma)(D(\omega)); \\ y \leftarrow N(\gamma)(D(\omega), x); \\ \text{ret } (D_{\text{ext}}(\omega), D(\omega), y) \end{array} \right) \\ &= \left(\begin{array}{l} (\delta_{\text{ext}}, \delta, x) \leftarrow \\ \left(\begin{array}{l} \omega \leftarrow \mu; \\ x \leftarrow M(\gamma)(D(\omega)); \\ \text{ret } (D_{\text{ext}}(\omega), D(\omega), x) \end{array} \right) \\ y \leftarrow N(\gamma)(\delta, x); \\ \text{ret } (\delta_{\text{ext}}, \delta, y) \end{array} \right) \stackrel{8}{=} \left(\begin{array}{l} (\delta_{\text{ext}}, \delta, x) \leftarrow \\ \left(\begin{array}{l} \omega \leftarrow \mu_X; \\ \text{ret } (D_{\text{ext}}(\omega), D(\omega), X(\omega)) \end{array} \right) \\ y \leftarrow N(\gamma)(\delta, x); \\ \text{ret } (\delta_{\text{ext}}, \delta, y) \end{array} \right) \\ &= \left(\begin{array}{l} \omega \leftarrow \mu_X; \\ y \leftarrow N(\gamma)(\delta, X(\omega)); \\ \text{ret } (D_{\text{ext}}(\omega), D(\omega), y) \end{array} \right) \stackrel{9}{=} \left(\begin{array}{l} \omega \leftarrow \mu_Y; \\ \text{ret } (D_{\text{ext}}(\omega), D(\omega), Y(\omega)) \end{array} \right) \end{aligned}$$

- Uniform: suppose (1) $\gamma, D, \mathcal{P} \models \forall_{rv} X:A. X \sim \text{Unif}[0, 1] \multimap Q$. 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 $\pi_{n+1} = (\dots, \omega_{n+1}, \dots) \mapsto \omega_{n+1}$. 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 $\gamma, (D, X), \mathcal{P} \bullet \mathcal{P}_n \models Q$ 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:

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

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

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

We generalize, and show

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

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

– Case $k = 0$:

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

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

$$\begin{aligned} & I(n+1-(j+1), V) \multimap \text{wp}(\text{loop}(n+1-(j+1), V, M), X'. I(n+1, X')) \\ & \vdash I(n-j, V) \multimap \text{wp}(\text{loop}(n-j, V, M), X'. I(n+1, X')) \\ & \vdash I(n-j, V) \multimap \text{wp}((V' \leftarrow M[n-j/i, V/X]; \text{loop}(n+1-j, V', M)), X'. I(n+1, X')) \\ & \vdash I(n-j, V) \multimap \text{wp}(M[n-j/i, V/X], V'. \text{wp}(\text{loop}(n+1-j, V', M), X'. I(n+1, X'))) \end{aligned}$$

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

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

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

$$I(n+1-j, V') \multimap \text{wp}(\text{loop}(n+1-j, V', M), X'. I(n+1, X'))$$

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

- If: applying properties of Markov kernels and rules for Let and Ret,

$$\begin{aligned}
& \text{wp}(\text{if } E \text{ then } M \text{ else } N, Z. Q(Z)) \\
& \vdash \text{wp} \left(\left(\begin{array}{l} X \leftarrow M; \\ Y \leftarrow N; \\ \text{ret } (\text{if } E \text{ then } X \text{ else } Y) \end{array} \right), Z. Q(Z) \right) \\
& \vdash \text{wp} \left(M, X. \text{wp} \left(\left(\begin{array}{l} Y \leftarrow N; \\ \text{ret } (\text{if } E \text{ then } X \text{ else } Y) \end{array} \right), Z. Q(Z) \right) \right) \\
& \vdash \text{wp}(M, X. \text{wp}(N, Y. \text{wp}(\text{ret } (\text{if } E \text{ then } X \text{ else } Y), Z. Q(Z)))) \\
& \vdash \text{wp}(M, X. \text{wp}(N, Y. Q(\text{if } E \text{ then } X \text{ else } Y)))
\end{aligned}$$

as desired.

□

COROLLARY B.13. *The following wp laws hold:*

$$\begin{array}{ll}
W\text{-RET} & W\text{-LET} \\
Q[\llbracket M \rrbracket / X] \vdash \text{wp}(\llbracket \text{ret } M \rrbracket, X.Q) & \text{wp}(\llbracket M \rrbracket, X. \text{wp}(\llbracket N \rrbracket, Y.Q)) \vdash \text{wp}(\llbracket X \leftarrow M; N \rrbracket, Y.Q)
\end{array}$$

$$\begin{array}{l}
W\text{-UNIFORM} \\
(\forall_{rv} X:A. X \sim \text{Unif}[0, 1] \multimap Q) \vdash \text{wp}(\llbracket \text{unif } [\emptyset, 1] \rrbracket, X:A.Q)
\end{array}$$

$$\begin{array}{l}
W\text{-FLIP} \\
(\forall_{rv} X:A. X \sim \text{Ber } p \multimap Q) \vdash \text{wp}(\llbracket \text{flip } p \rrbracket, X:A.Q)
\end{array}$$

$$\begin{array}{l}
W\text{-FOR} \\
I(1, e) * (\forall i:\mathbb{N}. \forall_{rv} X:A. \{I(i, X)\} M \{X'. I(i+1, X')\}) \vdash \text{wp}(\llbracket \text{for}(n, e, i X. M) \rrbracket, X:A. I(n+1, X))
\end{array}$$

$$\text{wp}(\llbracket M \rrbracket, X:A. \text{wp}(\llbracket N \rrbracket, Y:A. Q(\text{if } E \text{ then } X \text{ else } Y))) \vdash \text{wp}(\llbracket \text{if } EMN \rrbracket, X:A. Q)$$

PROOF. Unfold $\llbracket - \rrbracket$ and apply lemma B.12.

□

LEMMA B.14. *The following proof rules hold:*

$$\begin{array}{c}
\frac{P \vdash P' \quad Q' \vdash Q \quad \{P'\} M \{X. Q'\}}{\{P\} M \{X. Q\}} \text{H-CONSEQUENCE} \\
\\
\frac{\{P\} M \{X. Q\}}{\{F * P\} M \{X. F * Q\}} \text{H-FRAME } (X \notin F) \quad \frac{\{P\} M \{X. Q\} \quad \{P'\} M \{X. Q'\}}{\{P \vee P'\} M \{X. Q \vee Q'\}} \text{H-DISJUNCTION} \\
\\
\frac{}{\{Q[\llbracket M \rrbracket / X]\} \text{ ret } M \{X. Q\}} \text{H-RET} \quad \frac{\{P\} M \{X. Q\} \quad \forall_{rv} X. \{Q\} N \{Y. R\}}{\{P\} X \leftarrow M; N \{Y. R\}} \text{H-LET} \\
\\
\frac{}{\{\top\} \text{ unif } [0, 1] \{X. X \sim \text{Unif } [0, 1]\}} \text{H-UNIFORM} \quad \frac{}{\{\top\} \text{ flip } p \{X. X \sim \text{Ber } p\}} \text{H-FLIP} \\
\\
\frac{\forall i: \mathbb{N}. \forall_{rv} X: A. \{I(i, X)\} M \{X'. I(i + 1, X')\}}{\{I(1, e)\} \text{ for } (n, e, i X. M) \{X: A. I(n + 1, X)\}} \text{H-FOR} \\
\\
\frac{\{P\} M \{X. Q(X)\} \quad \forall_{rv} X. \{Q(X)\} N \{Y. R(\text{if } E \text{ then } X \text{ else } Y)\}}{\{P\} \text{ if } E \text{ then } M \text{ else } N \{Z. R(Z)\}} \text{H-IF}
\end{array}$$

PROOF. Rules H-CONSEQUENCE, H-FRAME, and H-DISJUNCTION follow from Lemma B.11; the remaining rules follow from Corollary B.13. All proofs go by unfolding the definition of the Hoare triple and applying the relevant wp law. \square

B.6 Disintegration

LEMMA B.15. *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) . \square

THEOREM B.16. *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 2.4 to $\mathcal{M}_{\text{disintegrable}}$ is still a KRM.*

PROOF. By Lemma B.7, 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 countably-generated. Then, the restriction to spaces that can be extended to a Borel measure still yields a KRM by Lemma B.15, as the set of spaces that can be extended to a Borel measure is downward-closed. \square

$$\frac{\Gamma; \Delta \vdash_{\text{rv}} E : A \quad \Gamma, x:A; \Delta \vdash P}{\Gamma; \Delta \vdash \mathbf{C}_{x:A \leftarrow E} P}$$

$\gamma, D, \mathcal{P} \models \mathbf{C}_{x:A \leftarrow E} P$ iff for all $(\Sigma_\Omega, \mu) \sqsupseteq \mathcal{P}$
 and all (Σ_Ω, μ) -disintegrations $\{v_x\}_{x \in A}$ with respect to $E(\gamma) \circ D$
 and $(E \circ D)_* \mu$ -almost-all $x \in A$,
 $(\gamma, x), D, v_x|_{\mathcal{P}} \models P$

LEMMA B.17. If μ and ν are probability measures on a space (Ω, \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. \square

LEMMA B.18. $\text{own } E * P \vdash \mathbf{C}_{x:A \leftarrow E} P$.

PROOF. Suppose $\gamma, D, \underbrace{\mathcal{P}_E \bullet \mathcal{P}_P}_{\mathcal{P}} \models \text{own } E * P$ and $\gamma, D, \mathcal{P}_E \models \text{own } E$ and (1) $\gamma, D, \mathcal{P}_P \models P$. To show

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

Write $\mathcal{P}_P = (\mathcal{F}_P, \mu_P)$ and let $S := \{x \in A \mid v_x|_{\mathcal{F}_P} = \mu_P\}$. 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.17, we can write S as the countable intersection $S = \bigcap_{n \in \mathbb{N}} S_n$ where $S_n := \{x \in A \mid v_x(B_n) = \mu_P(B_n)\}$. 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 $v_{(-)}(B_n)$; since v 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 $v_x(B_k) \neq \mu_P(B_k)$ 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:

$$N_< := \{x \in N \mid v_x(B_k) < \mu_P(B_k)\}$$

$$N_> := \{x \in N \mid v_x(B_k) > \mu_P(B_k)\}$$

These are both measurable, since they can be written as preimages of $[0, \mu_P(B_k))$ and $(\mu_P(B_k), 1]$ under $v_{(-)}(B_k)$. 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 v is a disintegration of μ with respect to $E(\gamma) \circ D$, we have

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

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

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

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

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

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. \square

LEMMA B.19 (LAW OF TOTAL EXPECTATION). *The following entailment holds:*

$$\mathbb{E}[e[X/x]] = v \wedge \mathbf{C}_{x:A \leftarrow X} \mathbb{E}[E] = e \vdash \mathbb{E}[E] = v$$

PROOF. Fix (γ, 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

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

as desired. \square

LEMMA B.20. *The following entailments hold:*

$$\begin{array}{c} \text{C-ENTAIL} \\ \frac{P \vdash Q}{\mathbf{C}_{x \leftarrow E} P \vdash \mathbf{C}_{x \leftarrow E} Q} \end{array} \quad \begin{array}{c} \text{C-INDEP} \\ \text{own } E * P \vdash \mathbf{C}_{x \leftarrow E} P \end{array} \quad \begin{array}{c} \text{C-SUBST} \\ \vdash \mathbf{C}_{x \leftarrow X} \left(E \stackrel{\text{as}}{=} E[x/X] \right) \end{array}$$

$$\begin{array}{c} \text{C-TOTAL-EXPECTATION} \\ \mathbf{C}_{x \leftarrow X} \mathbb{E}[E] = e \wedge \mathbb{E}[e[X/x]] = v \vdash \mathbb{E}[E] = v \end{array}$$

PROOF. C-INDEP and C-TOTAL-EXPECTATION follow from lemmas B.18 and B.19 respectively.

- C-ENTAIL: suppose $P \vdash Q$ and $\gamma, D, \mathcal{P} \models \mathbf{C}_{x:A \leftarrow E} P$. 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, $\gamma, D, \mathcal{P}_x \models P$ for almost-all $x \in A$. Since $P \vdash Q$, this implies $\gamma, D, \mathcal{P}_x \models Q$ for almost-all $x \in A$ as desired.
- C-SUBST: Fix (γ, D, \mathcal{P}) . Let $\{\mu_x\}_{x \in A}$ be a disintegration of \mathcal{P} with respect to $E(\gamma) \circ D$. Let E_x be the event that $E(\gamma) \circ D$ and $E[x/X](\gamma) \circ D$ agree. We need to show that E_x holds almost-surely with respect to μ_x for almost all $x \in A$. By the definition of disintegration, the event $\{\omega \mid X(\omega) = x\}$ has probability 1 under μ_x for almost all x . Since $\{\omega \mid X(\omega) = x\} \subseteq \{\omega \mid E(\gamma)(D\omega) = E[x/X](\gamma)(D\omega)\} = E_x$ this implies $\mu_x(E_x) = 1$ for almost-all x as desired.

□

LEMMA B.21. *The following entailments hold:*

- Necessitation: if $\vdash P$ then $\vdash \mathbf{C}_{x \leftarrow X} P$.
- Distribution: $\mathbf{C}_{x \leftarrow X} (P \rightarrow Q) \vdash \mathbf{C}_{x \leftarrow X} P \rightarrow \mathbf{C}_{x \leftarrow X} Q$.
- Distributes over (\wedge) : $\mathbf{C}_{x \leftarrow X} (P \wedge Q) \vdash \mathbf{C}_{x \leftarrow X} P \wedge \mathbf{C}_{x \leftarrow X} Q$.
- Semidistributes over (\vee) : $\mathbf{C}_{x \leftarrow X} P \vee \mathbf{C}_{x \leftarrow X} Q \vdash \mathbf{C}_{x \leftarrow X} (P \vee Q)$.

PROOF.

- Necessitation: if P holds in all configurations then it holds for all disintegrated configurations as well.
- Distribution: it suffices to show $D_{x \leftarrow X}(P \rightarrow Q) \wedge \mathbf{C}_{x \leftarrow X} P \vdash \mathbf{C}_{x \leftarrow X} Q$. By C-AND the premise is equivalent to $\mathbf{C}_{x \leftarrow X}((P \rightarrow Q) \wedge P)$; the result then follows from C-ENTAIL via the entailment $(P \rightarrow Q) \wedge P \vdash Q$.
- Distributes over (\wedge) : the left-to-right direction follows from C-ENTAIL via the entailments $P \wedge Q \vdash P$ and $P \wedge Q \vdash Q$. For the right-to-left entailment, suppose $\gamma, D, \mathcal{P} \models \mathbf{C}_{x \leftarrow E} P$ and $\gamma, D, \mathcal{P} \models \mathbf{C}_{x \leftarrow E} Q$ 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 $\gamma, D, \mathcal{P}_x \models P$ for all $x \in F_1$ and $\gamma, D, \mathcal{P}_x \models Q$ for all $x \in F_2$. Therefore, $\gamma, D, \mathcal{P}_x \models P \wedge Q$ for all $x \in F_1 \cap F_2$. Moreover, $F_1 \cap F_2$ has measure 1 by subadditivity, so $\gamma, D, \mathcal{P}_x \models P \wedge Q$ for almost-all x as desired.
- Semidistributes over (\vee) : it suffices to show $\mathbf{C}_{x \leftarrow X} P \vdash \mathbf{C}_{x \leftarrow X} (P \vee Q)$ and $\mathbf{C}_{x \leftarrow X} Q \vdash \mathbf{C}_{x \leftarrow X} (P \vee Q)$. These follow from C-ENTAIL via the entailments $P \vdash P \vee Q$ and $Q \vdash P \vee Q$ respectively.

□

C ANNOTATED COMMONCAUSE PROGRAM

```

{⊤}
  Z ← flip 1/2;
{Z ∼ Ber 1/2}
  X ← flip 1/2;
{Z ∼ Ber 1/2 * X ∼ Ber 1/2}
  Y ← flip 1/2;
{Z ∼ Ber 1/2 * X ∼ Ber 1/2 * Y ∼ Ber 1/2}
  A ← X || Z;
{Z ∼ Ber 1/2 * X ∼ Ber 1/2 * Y ∼ Ber 1/2 * A as (X ∨ Z)}
  B ← Y || Z;
{Z ∼ Ber 1/2 * X ∼ Ber 1/2 * Y ∼ Ber 1/2 * A as (X ∨ Z) * B as (Y ∨ Z)}
  ret (Z, X, Y, A, B)

{ Cz←Z (X ∼ Ber 1/2 * Y ∼ Ber 1/2 * A as (X ∨ Z) * B as (Y ∨ Z)) } (C-INDEP)

{ Cz←Z (  $\underbrace{X \sim \text{Ber } 1/2 * Y \sim \text{Ber } 1/2 * A^{\text{as}}(X \vee z) * B^{\text{as}}(Y \vee z)}_P$  ) } (C-SUBST)

{ Cz←Z ((z = T → P[T/z]) ∧ (z = F → P[F/z])) }

{ Cz←Z ( (z = T → X ∼ Ber 1/2 * Y ∼ Ber 1/2 * A as (X ∨ T) * B as (Y ∨ T)) ∧
  (z = F → X ∼ Ber 1/2 * Y ∼ Ber 1/2 * A as (X ∨ F) * B as (Y ∨ F)) ) }

{ Cz←Z ( (z = T → A as T * B as T) ∧
  (z = F → X ∼ Ber 1/2 * Y ∼ Ber 1/2 * A as X * B as Y) ) }

{ Cz←Z ( (z = T → own A * own B) ∧
  (z = F → A ∼ Ber 1/2 * B ∼ Ber 1/2) ) }

{ Cz←Z ( (z = T → own A * own B) ∧
  (z = F → own A * own B) ) }

{ Cz←Z (own A * own B) }

```

D AN EXAMPLE OF CONDITIONAL INDEPENDENCE VIA CONTROL FLOW

```

{⊤}
  Z ← flip 1/2;
{Z ∼ Ber 1/2}
  if Z then
    X1 ← flip p;
    {X1 ∼ Ber p}
    Y1 ← flip p;
    {X1 ∼ Ber p * Y1 ∼ Ber p}
    ret (Z, X1, Y1)
  else
    {∃rv X1 Y1. X1 ∼ Ber p * Y1 ∼ Ber p}
    X2 ← flip q;
    {(∃rv X1 Y1. X1 ∼ Ber p * Y1 ∼ Ber p) * X2 ∼ Ber q}
    Y2 ← flip q;
    {(∃rv X1 Y1. X1 ∼ Ber p * Y1 ∼ Ber p) * X2 ∼ Ber q * Y2 ∼ Ber q}
    ret (Z, X2, Y2)

```

$$\left\{ \begin{array}{l} Z \sim \text{Ber } 1/2 * \exists_{rv} X_1 Y_1 X_2 Y_2. X_1 \sim \text{Ber } p * Y_1 \sim \text{Ber } p * X_2 \sim \text{Ber } q * Y_2 \sim \text{Ber } q * \\ X \stackrel{\text{as}}{=} (\text{if } Z \text{ then } X_1 \text{ else } X_2) * Y \stackrel{\text{as}}{=} (\text{if } Z \text{ then } Y_1 \text{ else } Y_2) \end{array} \right\}$$

$$\left\{ \mathbf{C}_{z \leftarrow Z} \left(\begin{array}{l} \exists_{rv} X_1 Y_1 X_2 Y_2. X_1 \sim \text{Ber } p * Y_1 \sim \text{Ber } p * X_2 \sim \text{Ber } q * Y_2 \sim \text{Ber } q * \\ X \stackrel{\text{as}}{=} (\text{if } z \text{ then } X_1 \text{ else } X_2) * Y \stackrel{\text{as}}{=} (\text{if } z \text{ then } Y_1 \text{ else } Y_2) \end{array} \right) \right\} \text{ (C-INDEP)}$$

$$\left\{ \mathbf{C}_{z \leftarrow Z} \left(\underbrace{\begin{array}{l} \exists_{rv} X_1 Y_1 X_2 Y_2. X_1 \sim \text{Ber } p * Y_1 \sim \text{Ber } p * X_2 \sim \text{Ber } q * Y_2 \sim \text{Ber } q * \\ X \stackrel{\text{as}}{=} (\text{if } z \text{ then } X_1 \text{ else } X_2) * Y \stackrel{\text{as}}{=} (\text{if } z \text{ then } Y_1 \text{ else } Y_2) \end{array}}_P \right) \right\} \text{ (C-SUBST)}$$

$$\left\{ \mathbf{C}_{z \leftarrow Z} \left(\begin{array}{l} (z = \mathbf{T} \rightarrow P[\mathbf{T}/z]) \wedge \\ (z = \mathbf{F} \rightarrow P[\mathbf{F}/z]) \end{array} \right) \right\}$$

$$\left\{ \mathbf{C}_{z \leftarrow Z} \left(\begin{array}{l} (z = \mathbf{T} \rightarrow \exists_{rv} X_1 Y_1. X_1 \sim \text{Ber } p * Y_1 \sim \text{Ber } p * X \stackrel{\text{as}}{=} X_1 * Y \stackrel{\text{as}}{=} Y_1) \wedge \\ (z = \mathbf{F} \rightarrow \exists_{rv} X_2 Y_2. X_2 \sim \text{Ber } q * Y_2 \sim \text{Ber } q * X \stackrel{\text{as}}{=} X_2 * Y \stackrel{\text{as}}{=} Y_2) \end{array} \right) \right\}$$

$$\left\{ \mathbf{C}_{z \leftarrow Z} \left(\begin{array}{l} (z = \mathbf{T} \rightarrow X \sim \text{Ber } p * Y \sim \text{Ber } p) \wedge \\ (z = \mathbf{F} \rightarrow X \sim \text{Ber } q * Y \sim \text{Ber } q) \end{array} \right) \right\}$$

$$\left\{ \mathbf{C}_{z \leftarrow Z} \left(\begin{array}{l} (z = \mathbf{T} \rightarrow \text{own } X * \text{own } Y) \wedge \\ (z = \mathbf{F} \rightarrow \text{own } X * \text{own } Y) \end{array} \right) \right\}$$

$$\left\{ \mathbf{C}_{z \leftarrow Z} (\text{own } X * \text{own } Y) \right\}$$

```

1  $W \leftarrow \text{ret } (w_1 + \dots + w_n);$ 
2  $(v_1, \dots, v_n) \leftarrow \text{ret } (w_1/W, \dots, w_n/W);$ 
3  $U \leftarrow \text{unif } [0, 1];$ 
4 for  $(n, 0, i, J).$ 
5   if  $v_1 + \dots + v_{i-1} \leq U < v_1 + \dots + v_i$ 
6     then ret  $i$ 
7   else ret  $J$ 

```

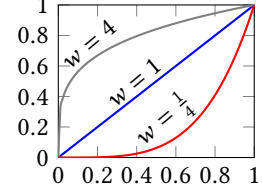
(a) Naive linear-space implementation of weighted sampling.

```

1  $w \leftarrow \text{ret } [w_1, \dots, w_n];$ 
2  $M \leftarrow \text{ret } (-\infty); K \leftarrow \text{ret } (0);$ 
3 for  $(n, (M, K), i, (M, K)).$ 
4    $S \leftarrow \text{unif } [0, 1];$ 
5    $U \leftarrow \text{ret } (S^{1/w[i]});$ 
6   if  $U > M$ 
7     then ret  $(U, i)$ 
8   else ret  $(M, K)$ 

```

(b) Constant-space version.



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

Fig. 11. Weighted sampling example. The constants w_i are inputs.

E PROVING A WEIGHTED SAMPLING ALGORITHM CORRECT (FULL)

To exercise Lilac’s support for conditional reasoning, continuous random variables, and substructural handling of independence, we now prove a sophisticated constant-space *weighted sampling algorithm* correct using Lilac. Suppose you are given a collection of items $\{x_1, \dots, x_n\}$ 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 instance of *reservoir sampling* [Efrimidis and Spirakis 2006], and is an important primitive in distributed systems.

First, we consider a naive solution that requires space linear in the number of weights; pseudocode for this algorithm is presented in Figure 11a. 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, \dots, 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, \dots, 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 4–7).

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, Efrimidis and Spirakis [2006] proposed the very clever *constant-space solution* presented in Figure 11b. 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 5–8). Figure 11c 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 informal proof. Correctness is captured by the following Lilac postcondition:

$$\forall k. \Pr(K = k) = \frac{w_k}{\sum_j w_j} \quad (10)$$

To establish this postcondition, a typical informal proof begins by declaring mutually independent, uniformly distributed random variables $\{S_i\}_{1 \leq i \leq n}$, where S_i denotes the value sampled by Line 4 on the i th loop iteration, and a random variable $K = \arg \max_i S_i^{1/w_i}$ that 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 the proof rules described in Section 2.5 to conclude the following at program termination:

$$\exists_{rv} S_1 \dots S_n. \bigstar_i S_i \sim \text{Unif } [0, 1] \quad * \quad K \stackrel{\text{as}}{=} \arg \max_i S_i^{1/w_i} \quad (11)$$

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 j th iteration for all $1 \leq j \leq n + 1$:

$$I_j \quad := \quad \exists_{rv} S_1 \dots S_j. \bigstar_{1 \leq i < j} S_i \sim \text{Unif } [0, 1] \quad * \quad K \stackrel{\text{as}}{=} \arg \max_{1 \leq i < j} S_i^{1/w_i} \quad * \quad M \stackrel{\text{as}}{=} \max_{1 \leq i < j} S_i^{1/w_i} \quad (12)$$

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 (10) given the setup (11). To show (10) in the case $i = k$, note that

$$\Pr(K = k) = \Pr\left(S_k^{1/w_k} > S_j^{1/w_j} \text{ for all } j \neq k\right), \quad (13)$$

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 directly. The trick is to use conditioning: in this case, fixing S_k to a deterministic s_k gives

$$\Pr(K = k \mid S_k = s_k) = \Pr\left(s_k^{1/w_k} > S_j^{1/w_j} \text{ for all } j \neq k\right) \quad (14)$$

$$= \Pr\left(s_k^{w_j/w_k} > S_j \text{ for all } j \neq k\right) \quad \text{Exponentiating} \quad (15)$$

$$= \prod_{j \neq k} \Pr\left(s_k^{w_j/w_k} > S_j\right) \quad \text{By conditional independence} \quad (16)$$

$$= \prod_{j \neq k} s_k^{w_j/w_k} \quad S_j \text{ uniform}^{15} \quad (17)$$

$$= \text{pow}\left(s_k, \frac{\sum_{j \neq k} w_j}{w_k}\right). \quad (18)$$

Formally, this calculation occurs under the modality $\mathbf{C}_{s_k \leftarrow S_k}$, which is introduced via C-INDEP. The expression $\Pr(E)$ abbreviates $\mathbb{E}[\mathbb{1}[E]]$, the expectation of the indicator random variable $\mathbb{1}[E]$.¹⁶ A critical step occurs in Equation 16: 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. This is an application of the derived rule

$$\bigstar_i \text{ own } E_i \vdash \Pr\left(\bigcap_i E_i\right) = \prod_i \Pr(E_i), \quad (\text{INDEP-PROD})$$

an immediate consequence of Lemma 2.5. Note that our modal treatment of conditioning leads to a nice separation of concerns here. Because \mathbf{C} respects entailment, facts like INDEP-PROD that appear to be only about unconditional independence and unconditional probability are automatically lifted to facts like Equation (16), with the expected conditional reading.

¹⁵If U uniform in $[0, 1]$ then $\Pr(u > U) = u$.

¹⁶If E is an event then the random variable $\mathbb{1}[E]$ is 1 if E holds and 0 otherwise.

Finally, to complete the proof we connect the conditional $\Pr(K = k \mid S_k = s_k)$ to the unconditional $\Pr(K = k)$ using the law of total expectation:

$$\Pr(K = k) = \mathbb{E} [\Pr(K = k \mid S_k)] \quad \text{Law of Total Expectation} \quad (19)$$

$$= \mathbb{E} \left[\text{pow} \left(S_k, \frac{\sum_{j \neq k} w_j}{w_k} \right) \right] \quad \text{By (18)} \quad (20)$$

$$= \left(\frac{\sum_{j \neq k} w_j}{w_k} + 1 \right)^{-1} \quad S_k \text{ uniform}^{17} \quad (21)$$

$$= \frac{w_k}{\sum_j w_j}. \quad (22)$$

Unlike the calculation in Equations 14–18, which take place inside the modality $\mathbf{C}_{s_k \leftarrow S_k}$, this second calculation (Equations 19–22) takes place outside of it, as it computes the *unconditional* probability $\Pr(K = k)$. The gap between the two calculations is bridged by the following instantiation of C-TOTAL-EXPECTATION:

$$\mathbf{C}_{s_k \leftarrow S_k} \left(\underbrace{\mathbb{E}[\mathbb{1}[K = k]]}_E = \underbrace{\text{pow} \left(s_k, \frac{\sum_{j \neq k} w_j}{w_k} \right)}_e \right) \wedge \left(\underbrace{\mathbb{E} \left[\text{pow} \left(S_k, \frac{\sum_{j \neq k} w_j}{w_k} \right) \right]}_{e[S_k/s_k]} = \underbrace{\frac{w_k}{\sum_j w_j}}_v \right) \vdash \underbrace{\mathbb{E}[\mathbb{1}[K = k]]}_E = \underbrace{\frac{w_k}{\sum_j w_j}}_v$$

Putting all this together yields a formal proof of correctness in Lilac. The next page gives a fully annotated program.

¹⁷If U uniform in $[0, 1]$ then $\mathbb{E}[U^\alpha] = 1/(\alpha + 1)$.

```

{⊤}
  W ← ret [w1, ..., wn];
{W as (w1, ..., wn)}
  M ← ret (−∞);
{W as (w1, ..., wn) * M as −∞}
  K ← ret 0;
{W as (w1, ..., wn) * M as −∞ * K as 0}
Let I(i, M, K) :=  $\left( W \overset{\text{as}}{=} (w_1, \dots, w_n) * 1 \leq i \leq n * \right.$ 
 $\left. \exists S_1 \dots S_i. \bigstar_{1 \leq j < i} S_j \sim \text{Unif } [0, 1] * K \overset{\text{as}}{=} \arg \max_{1 \leq j < i} S_j^{1/w_j} * M \overset{\text{as}}{=} \max_{1 \leq j < i} S_j^{1/w_j} \right)$ 
{I(1, M, K)}
  for(n, (M, K), i (M, K).
    {I(i, M, K)}
      S ← unif [0, 1];
      {I(i, M, K) * S ~ Unif [0, 1]}
      U ← ret (S ^ (1/w[i]));
      {I(i, M, K) * S ~ Unif [0, 1] * U as S1/wi}
      if U > M
        then ret (U, i)
      else ret (M, K)
       $\left\{ (M', K'). \begin{array}{l} I(i, M, K) * S \sim \text{Unif } [0, 1] * U \overset{\text{as}}{=} S^{1/w_i} * \\ M' \overset{\text{as}}{=} \max(U, M) * U' \overset{\text{as}}{=} \text{if } U > M \text{ then } i \text{ else } K \end{array} \right\}$ 
      {(M', K'). I(i + 1, M', K')}
    {(M', K'). I(n + 1, M', K')}
     $\left\{ \exists_{\text{rv}} S_1 \dots S_n. \bigstar_i S_i \sim \text{Unif } [0, 1] * K \overset{\text{as}}{=} \arg \max_i S_i^{1/w_i} \right\}$ 
     $\left\{ \forall k. \Pr(K = k) = \frac{w_k}{\sum_j w_j} \right\}$ 

```

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:

$$\begin{array}{c}
\frac{\begin{array}{c} * S_i \sim \text{Unif } [0, 1] \quad K \stackrel{\text{as}}{=} \arg \max_i S_i^{1/w_i} \\ \hline \Pr[K = k] = \frac{w_k}{\sum_j w_j} \end{array}}{\begin{array}{c} * S_i \sim \text{Unif } [0, 1] \quad K \stackrel{\text{as}}{=} \arg \max_i S_i^{1/w_i} \\ \hline \Pr[\forall j \neq k. S_k^{1/w_k} > S_j^{1/w_j}] = \frac{w_k}{\sum_j w_j} \end{array}} \\
\frac{\begin{array}{c} * S_i \sim \text{Unif } [0, 1] \quad K \stackrel{\text{as}}{=} \arg \max_i S_i^{1/w_i} \\ \hline \Pr[\forall j \neq k. S_k^{w_j/w_k} > S_j] = \frac{w_k}{\sum_j w_j} \end{array}}{\begin{array}{c} * S_i \sim \text{Unif } [0, 1] \quad K \stackrel{\text{as}}{=} \arg \max_i S_i^{1/w_i} \\ \hline \mathbb{E} \left[\mathbb{1}[\forall j \neq k. S_k^{w_j/w_k} > S_j] \right] = \frac{w_k}{\sum_j w_j} \end{array}} \\
\frac{\begin{array}{c} * S_i \sim \text{Unif } [0, 1] \quad K \stackrel{\text{as}}{=} \arg \max_i S_i^{1/w_i} \\ \hline \mathbb{E} \left[\prod_{j \neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j] \right] = \frac{w_k}{\sum_j w_j} \end{array}}{\begin{array}{c} * S_i \sim \text{Unif } [0, 1] \quad K \stackrel{\text{as}}{=} \arg \max_i S_i^{1/w_i} \\ \hline \mathbb{E} \left[\prod_{j \neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j] \right] = \frac{w_k}{\sum_j w_j} \end{array}}
\end{array}$$

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

$$\begin{array}{c}
\frac{S_k \sim \text{Unif } [0, 1] \wedge \mathbf{C}_{s_k \leftarrow S_k} * S_j \sim \text{Unif } [0, 1] \quad K \stackrel{\text{as}}{=} \arg \max_i S_i^{1/w_i}}{\mathbb{E} \left[\prod_{j \neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j] \right] = \frac{w_k}{\sum_j w_j}} \\
\frac{S_k \sim \text{Unif } [0, 1] \wedge \mathbf{C}_{s_k \leftarrow S_k} * S_j \sim \text{Unif } [0, 1] \wedge \left(\mathbb{E} \left[\prod_{j \neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j] \right] = \mathbb{E} \left[\prod_{j \neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j] \right] \right)}{K \stackrel{\text{as}}{=} \arg \max_i S_i^{1/w_i}} \\
\hline
\mathbb{E} \left[\prod_{j \neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j] \right] = \frac{w_k}{\sum_j w_j}
\end{array}$$

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

$$S_k \sim \text{Unif } [0, 1] \wedge \mathbf{C}_{s_k \leftarrow S_k} * S_j \sim \text{Unif } [0, 1] \wedge \left(\mathbb{E} \left[\prod_{j \neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j] \right] \right) = \prod_{j \neq k} \mathbb{E}[\mathbb{1}[S_k^{w_j/w_k} > S_j]]$$

$$K \stackrel{\text{as}}{=} \arg \max_i S_i^{1/w_i}$$

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

Now significant simplifications are possible, completing the first calculation (Equations 14–18):

$$S_i \sim \text{Unif } [0, 1] \wedge \mathbf{C}_{s_k \leftarrow S_k} * S_j \sim \text{Unif } [0, 1] \wedge \left(\mathbb{E} \left[\prod_{j \neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j] \right] \right) = \prod_{j \neq k} S_k^{w_j/w_k}$$

$$K \stackrel{\text{as}}{=} \arg \max_i S_i^{1/w_i}$$

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

$$S_k \sim \text{Unif } [0, 1] \wedge \mathbf{C}_{s_k \leftarrow S_k} * S_j \sim \text{Unif } [0, 1] \wedge \left(\mathbb{E} \left[\prod_{j \neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j] \right] \right) = \exp \left(s_k, \frac{\sum_{j \neq k} w_j}{w_k} \right)$$

$$K \stackrel{\text{as}}{=} \arg \max_i S_i^{1/w_i}$$

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

$$S_k \sim \text{Unif } [0, 1] \wedge \mathbf{C}_{s_k \leftarrow S_k} \mathbb{E} \left[\prod_{j \neq k} \mathbb{1}[S_k^{w_j/w_k} > S_j] \right] = \exp \left(s_k, \frac{\sum_{j \neq k} w_j}{w_k} \right)$$

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

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 conditioning modality by applying the law of total expectation (C-TOTAL-EXPECTATION):

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

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

The remainder of the calculation is straightforward, following Equations 19–22:

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

QED

F EXAMPLES FROM BARTHE ET. AL.

In this section we consider three of the five examples presented in Barthe et al. [2019]: one-time pad, oblivious transfer, and private information retrieval. In each example, the goal is to verify the perfect secrecy of a cryptographic protocol. Perfect secrecy is established via two methods: uniformity, which aims to show that each agent’s view of others’ data is uniformly distributed at exit, and input independence, which aims to show that the encrypted output of the protocol is independent of the input.

Barthe et al. [2019] use PSL to establish perfect secrecy of one-time pad and private information retrieval via both uniformity and input independence, and perfect secrecy of oblivious transfer via uniformity. We will show how the same can be done in Lilac. Barthe et al. [2019] also observe that the input independence proof for oblivious transfer gets stuck, mentioning that even an informal proof sketch does not seem easy. We will show that the postcondition specifying input independence for oblivious transfer is in fact unsatisfiable by giving a countermodel.

To do this, we add some support for length- n bitvectors and reasoning about uniformity. For bitvectors,

- Let $\text{Ber}^n 1/2$ be the uniform distribution on boolean-valued n -tuples.
- Let $\text{flip}^n p$ be the n -ary generalization of flip that produces n -tuples of i.i.d. $\text{Ber } p$ random variables, with the evident semantics.
- If X is a random variable valued in boolean n -tuples, let $\bigoplus X$ be the random variable given by the \oplus of all n components.
- Let $\&\&^n$ and \oplus^n be the lifting of boolean $\&\&$ and \oplus to n -tuples.
- We will make use of algebraic properties of the bitvector xor operator \oplus^n throughout; in particular the property that $x \oplus^n -$ is invertible.

Next, we import the requisite probability theory facts as derived rules. For clarity of exposition, we suppress components of Lilac’s semantic model (like underlying probability spaces, the random substitution, and the deterministic substitution) in the proofs of these rules in favor of a presentation that more closely mirrors textbook probability. The first few facts concern uniformity of random bitvectors:

LEMMA F.1. *If X is a random n -bitvector and $f : \llbracket \text{bool} \rrbracket^n \rightarrow \llbracket \text{bool} \rrbracket^n$ a bijection then*

$$X \sim \text{Ber}^n 1/2 \vdash f(X) \sim \text{Ber}^n 1/2.$$

PROOF. We have $\Pr[f(X) = x] = \Pr[X = f^{-1}(x)] = 1/2^n$ for all x . □

LEMMA F.2. *If X is a random n -bitvector and Y a random m -bitvector then*

$$X \sim \text{Ber}^n 1/2 * Y \sim \text{Ber}^m 1/2 \dashv\vdash (X, Y) \sim \text{Ber}^{m+n} 1/2$$

PROOF. Calculation gives

$$\Pr[(X, Y) = (x, y)] = \Pr[X = x, Y = y] \stackrel{(a)}{=} \Pr[X = x] \Pr[Y = y] = (1/2^n)(1/2^m) = 1/2^{m+n}$$

for all x, y as desired. Equation (a) follows from independence of X and Y . □

This next lemma encodes the key fact of probability theory underlying the perfect secrecy of the examples we will consider in the next section. Intuitively, it states that any random variable which is “conditionally uniformly distributed” (that is, uniformly distributed conditional on some other random variable) is uniformly distributed proper.

LEMMA F.3. *If X is a random variable taking on finitely many values¹⁸ and Y a random n -bitvector then*

$$\text{own } X \wedge \bigoplus_{x \leftarrow X} (Y \sim \text{Ber}^n 1/2) \vdash \text{own } X * Y \sim \text{Ber}^n 1/2$$

PROOF. It suffices to show X and Y are independent and Y uniform. This amounts to showing the equality $\Pr[X = x, Y = y] = \Pr[X = x]/2^n$ for all x, y . By assumption Y is uniformly distributed conditional on X , so

$$\Pr[X = x, Y = y] = \Pr[Y = y \mid X = x] \Pr[X = x] = \frac{1}{2^n} \Pr[X = x]$$

as desired. □

To avoid verbosity, we use the abbreviation $\bigoplus_{x \leftarrow X}^{\text{own}} P := \text{own } X \wedge \bigoplus_{x \leftarrow X} P$.

Thus the entailment given by Lemma F.3 can be written $\bigoplus_{x \leftarrow X}^{\text{own}} (Y \sim \text{Ber}^n 1/2) \vdash \text{own } X * Y \sim \text{Ber}^n 1/2$.

We will also frequently make use of the following variant of C-INDEP:

LEMMA F.4. *The following entailment holds:*

$$\text{own } X * P \vdash \bigoplus_{x \leftarrow X}^{\text{own}} P.$$

PROOF. By the following chain:

$$\begin{array}{ll} \text{own } X * P & \\ \vdash (\text{own } X * P) \wedge (\text{own } X * P) & \text{idempotency of } \wedge \\ \vdash \text{own } X \wedge (\text{own } X * P) & \text{drop a conjunct} \\ \vdash \text{own } X \wedge \bigoplus_{x \leftarrow X} P & \text{C-INDEP} \\ \vdash \bigoplus_{x \leftarrow X}^{\text{own}} P & \text{definition of } \bigoplus^{\text{own}} \end{array}$$

□

¹⁸Though we expect this restriction can be lifted, $\text{cod}(X)$ finite suffices for our examples.

The final derived rule we will make use of encodes the probability-theoretic fact that, when proving an assertion of the form $X \sim \mu * P$, one can first establish that X has distribution μ , and then separately establish independence of X from other random variables.

LEMMA F.5. *Let X be a random variable, μ a distribution, and P a proposition.*

$$(X \sim \mu) \wedge (\text{own } X * P) \vdash (X \sim \mu) * P$$

PROOF. Suppose X is distributed as μ with respect to probability space \mathcal{P}_X , that P holds in space \mathcal{P} , and that there exists a space \mathcal{P}'_X independent of \mathcal{P} for which X is \mathcal{P}'_X -measurable. Let \mathcal{Q} be the pullback σ -algebra of X . We have that $\mathcal{P}_X \supseteq \mathcal{Q} \subseteq \mathcal{P}'_X$ and that \mathcal{P}_X and \mathcal{P}'_X agree on \mathcal{Q} . Thus, \mathcal{Q} is independent of \mathcal{P} and the composite $\mathcal{Q} \bullet \mathcal{P}$ witnesses $X \sim \mu * P$ as desired. \square

F.1 Verification of one-time pad example

The one-time pad protocol is modelled by the following probabilistic program, parameterized by a constant m representing the message being encrypted:

```

K ← flip 1/2;
C ← ret (K ⊕ m);
ret C
(ONETIMEPAD)
```

F.1.1 Uniformity. Uniformity is specified by the triple $\{\top\} \text{ONETIMEPAD}(m) \{C. C \sim \text{flip } 1/2\}$. This can be established by the following annotation:

```

{⊤}
  K ← flip 1/2;
  {K ~ Ber 1/2}
  C ← ret (K ⊕ m);
  {K ~ Ber 1/2 * C as K ⊕ m}
  ret C
  {C. K ~ Ber 1/2 * C as K ⊕ m}
  {C. K ~ Ber 1/2 * C ⊕ m as K}           rearranging the equality
  {C. (C ⊕ m) ~ Ber 1/2}                 substituting away K
  {C. C ~ Ber 1/2}                       Lemma F.1
```

F.1.2 Input independence. Input independence is specified by the triple

$$\{\text{own } M\} \text{ONETIMEPAD}(M) \{C. \text{own } M * C \sim \text{Ber } 1/2\}.$$

This can be established by the following annotation:

$$\begin{array}{ll}
\{\text{own } M\} & \\
K \leftarrow \text{flip } 1/2; & \\
\{\text{own } M * K \sim \text{Ber } 1/2\} & \\
C \leftarrow \text{ret } (K \oplus M); & \\
\{\text{own } M * K \sim \text{Ber } 1/2 * C \stackrel{\text{as}}{=} K \oplus M\} & \\
\text{ret } C & \\
\{C. \text{own } M * K \sim \text{Ber } 1/2 * C \stackrel{\text{as}}{=} K \oplus M\} & \\
\{C. \text{own } M * K \sim \text{Ber } 1/2 * C \oplus M \stackrel{\text{as}}{=} K\} & \text{rearranging the equality} \\
\{C. \text{own } M * (C \oplus M) \sim \text{Ber } 1/2\} & \text{substituting away } K \\
\left\{ C. \bigoplus_{m \leftarrow M}^{\text{own}} ((C \oplus M) \sim \text{Ber } 1/2) \right\} & \text{Lemma F.4} \\
\left\{ C. \bigoplus_{m \leftarrow M}^{\text{own}} ((C \oplus m) \sim \text{Ber } 1/2) \right\} & \text{replacing } M \text{ with } m \\
\left\{ C. \bigoplus_{m \leftarrow M}^{\text{own}} (C \sim \text{Ber } 1/2) \right\} & \text{Lemma F.1} \\
\{C. \text{own } M * C \sim \text{Ber } 1/2\} & \text{Lemma F.3}
\end{array}$$

F.2 Verification of private information retrieval example

Let i be an n -tuple of boolean values with only a single component set to \top . The private information retrieval protocol is modelled by the following probabilistic program, parameterized by i :

$$\begin{array}{ll}
Q_0 \leftarrow \text{flip}^n 1/2; & \\
Q_1 \leftarrow Q_0 \oplus^n i; & \\
A_0 \leftarrow Q_0 \&\&^n d; & \\
A_1 \leftarrow Q_1 \&\&^n d; & \\
R_0 \leftarrow \text{ret } \left(\bigoplus A_0 \right); & (\text{PRIVATEINFORMATIONRETRIEVAL}) \\
R_1 \leftarrow \text{ret } \left(\bigoplus R_1 \right); & \\
R \leftarrow \text{ret } (R_0 \oplus R_1); & \\
\text{ret } (Q_0, Q_1, A_0, A_1, R_0, R_1, R) &
\end{array}$$

F.2.1 Uniformity. Uniformity is specified by the triple

$\{\top\} \text{PRIVATEINFORMATIONRETRIEVAL}(i) \{(Q_0, Q_1, A_0, A_1, R_0, R_1, R). Q_0 \sim \text{Ber}^n 1/2 \wedge Q_1 \sim \text{Ber}^n 1/2\}.$

This can be established by the following annotation:

$$\begin{aligned}
& \{\top\} \\
& Q_0 \leftarrow \text{flip}^n 1/2; \\
& \{Q_0 \sim \text{Ber}^n 1/2\} \\
& Q_1 \leftarrow Q_0 \oplus^n i; \\
& \{Q_0 \sim \text{Ber}^n 1/2 * Q_1 \stackrel{\text{as}}{=} Q_0 \oplus^n i\} \\
& A_0 \leftarrow Q_0 \&\&^n d; \\
& \{Q_0 \sim \text{Ber}^n 1/2 * Q_1 \stackrel{\text{as}}{=} Q_0 \oplus^n i * A_0 \stackrel{\text{as}}{=} Q_0 \&\&^n d\} \\
& A_1 \leftarrow Q_1 \&\&^n d; \\
& \{Q_0 \sim \text{Ber}^n 1/2 * Q_1 \stackrel{\text{as}}{=} Q_0 \oplus^n i * A_0 \stackrel{\text{as}}{=} Q_0 \&\&^n d * A_1 \stackrel{\text{as}}{=} Q_1 \&\&^n d\} \\
& R_0 \leftarrow \text{ret} \left(\bigoplus A_0 \right); \\
& \left\{ Q_0 \sim \text{Ber}^n 1/2 * Q_1 \stackrel{\text{as}}{=} Q_0 \oplus^n i * A_0 \stackrel{\text{as}}{=} Q_0 \&\&^n d * A_1 \stackrel{\text{as}}{=} Q_1 \&\&^n d * R_0 = \bigoplus A_0 \right\} \\
& R_1 \leftarrow \text{ret} \left(\bigoplus A_1 \right); \\
& \left\{ Q_0 \sim \text{Ber}^n 1/2 * Q_1 \stackrel{\text{as}}{=} Q_0 \oplus^n i * A_0 \stackrel{\text{as}}{=} Q_0 \&\&^n d * A_1 \stackrel{\text{as}}{=} Q_1 \&\&^n d * R_0 = \bigoplus A_0 * R_1 = \bigoplus A_1 \right\} \\
& R \leftarrow \text{ret} (R_0 \oplus R_1); \\
& \left\{ \begin{array}{l} Q_0 \sim \text{Ber}^n 1/2 * Q_1 \stackrel{\text{as}}{=} Q_0 \oplus^n i * A_0 \stackrel{\text{as}}{=} Q_0 \&\&^n d * A_1 \stackrel{\text{as}}{=} Q_1 \&\&^n d * \\ R_0 = \bigoplus A_0 * R_1 = \bigoplus A_1 * R \stackrel{\text{as}}{=} R_0 \oplus R_1 \end{array} \right\} \\
& \text{ret} (Q_0, Q_1, A_0, A_1, R_0, R_1, R) \\
& \left\{ \begin{array}{l} (Q_0, Q_1, A_0, A_1, R_0, R_1, R). \\ Q_0 \sim \text{Ber}^n 1/2 * Q_1 \stackrel{\text{as}}{=} Q_0 \oplus^n i * A_0 \stackrel{\text{as}}{=} Q_0 \&\&^n d * A_1 \stackrel{\text{as}}{=} Q_1 \&\&^n d * \\ R_0 = \bigoplus A_0 * R_1 = \bigoplus A_1 * R \stackrel{\text{as}}{=} R_0 \oplus R_1 \end{array} \right\} \\
& \vdots \\
& \{(Q_0, Q_1, A_0, A_1, R_0, R_1, R). Q_0 \sim \text{Ber}^n 1/2 \wedge Q_1 \sim \text{Ber}^n 1/2\}
\end{aligned}$$

The final entailment (hidden by the vertical ellipses) can be established as follows:

$$\begin{aligned}
& \left(\begin{array}{l} Q_0 \sim \text{Ber}^n 1/2 * Q_1 \stackrel{\text{as}}{=} Q_0 \oplus^n i * A_0 \stackrel{\text{as}}{=} Q_0 \&\&^n d * \\ R_0 = \bigoplus A_0 * R_1 = \bigoplus A_1 * R \stackrel{\text{as}}{=} R_0 \oplus R_1 \end{array} \right) \\
& \vdash Q_0 \sim \text{Ber}^n 1/2 * Q_1 \stackrel{\text{as}}{=} Q_0 \oplus^n i && \text{dropping some conjuncts} \\
& \vdash (Q_0 \sim \text{Ber}^n 1/2 * Q_1 \stackrel{\text{as}}{=} Q_0 \oplus^n i) \wedge (Q_0 \sim \text{Ber}^n 1/2 * Q_1 \stackrel{\text{as}}{=} Q_0 \oplus^n i) && \text{by } P \vdash P \wedge P \\
& \vdash (Q_0 \sim \text{Ber}^n 1/2) \wedge (Q_0 \sim \text{Ber}^n 1/2 * Q_1 \stackrel{\text{as}}{=} Q_0 \oplus^n i) && \text{weakening first conjunct} \\
& \vdash (Q_0 \sim \text{Ber}^n 1/2) \wedge (Q_0 \sim \text{Ber}^n 1/2 * Q_1 \oplus^n i \stackrel{\text{as}}{=} Q_0) && \text{rearranging the equality} \\
& \vdash (Q_0 \sim \text{Ber}^n 1/2) \wedge ((Q_1 \oplus^n i) \sim \text{Ber}^n 1/2) && \text{substituting away } Q_0 \\
& \vdash (Q_0 \sim \text{Ber}^n 1/2) \wedge (Q_1 \sim \text{Ber}^n 1/2) && \text{Lemma F.1}
\end{aligned}$$

F.2.2 Input independence. Let I be a random n -bitvector.¹⁹ Input independence is specified by the triple

$$\{\text{own } I\} \text{ PRIVATEINFORMATIONRETRIEVAL}(I) \{C. (\text{own } I * \text{own } Q_0) \wedge (\text{own } I * \text{own } Q_1)\}.$$

This can be established by the following annotation:

$$\begin{aligned}
& \{\text{own } I\} \\
& \quad Q_0 \leftarrow \text{flip}^n 1/2; \\
& \{\text{own } I * Q_0 \sim \text{Ber}^n 1/2\} \\
& \quad Q_1 \leftarrow Q_0 \oplus^n I; \\
& \{\text{own } I * Q_0 \sim \text{Ber}^n 1/2 * Q_1 \stackrel{\text{as}}{=} Q_0 \oplus^n I\} \\
& \quad A_0 \leftarrow Q_0 \&\&^n d; \\
& \{\text{own } I * Q_0 \sim \text{Ber}^n 1/2 * Q_1 \stackrel{\text{as}}{=} Q_0 \oplus^n I * A_0 \stackrel{\text{as}}{=} Q_0 \&\&^n d\} \\
& \quad A_1 \leftarrow Q_1 \&\&^n d; \\
& \{\text{own } I * Q_0 \sim \text{Ber}^n 1/2 * Q_1 \stackrel{\text{as}}{=} Q_0 \oplus^n I * A_0 \stackrel{\text{as}}{=} Q_0 \&\&^n d * A_1 \stackrel{\text{as}}{=} Q_1 \&\&^n d\} \\
& \quad R_0 \leftarrow \text{ret} \left(\bigoplus A_0 \right); \\
& \{\text{own } I * Q_0 \sim \text{Ber}^n 1/2 * Q_1 \stackrel{\text{as}}{=} Q_0 \oplus^n I * A_0 \stackrel{\text{as}}{=} Q_0 \&\&^n d * A_1 \stackrel{\text{as}}{=} Q_1 \&\&^n d * R_0 = \bigoplus A_0\} \\
& \quad R_1 \leftarrow \text{ret} \left(\bigoplus R_1 \right); \\
& \left\{ \begin{array}{l} \text{own } I * Q_0 \sim \text{Ber}^n 1/2 * Q_1 \stackrel{\text{as}}{=} Q_0 \oplus^n I * A_0 \stackrel{\text{as}}{=} Q_0 \&\&^n d * A_1 \stackrel{\text{as}}{=} Q_1 \&\&^n d * \\ R_0 = \bigoplus A_0 * R_1 = \bigoplus A_1 * R \stackrel{\text{as}}{=} R_0 \oplus R_1 \end{array} \right\} \\
& \quad R \leftarrow \text{ret} (R_0 \oplus R_1); \\
& \left\{ \begin{array}{l} \text{own } I * Q_0 \sim \text{Ber}^n 1/2 * Q_1 \stackrel{\text{as}}{=} Q_0 \oplus^n I * A_0 \stackrel{\text{as}}{=} Q_0 \&\&^n d * A_1 \stackrel{\text{as}}{=} Q_1 \&\&^n d * \\ R_0 = \bigoplus A_0 * R_1 = \bigoplus A_1 * R \stackrel{\text{as}}{=} R_0 \oplus R_1 \end{array} \right\} \\
& \quad \text{ret} (Q_0, Q_1, A_0, A_1, R_0, R_1, R) \\
& \left\{ \begin{array}{l} (Q_0, Q_1, A_0, A_1, R_0, R_1, R). \text{own } I * Q_0 \sim \text{Ber}^n 1/2 * Q_1 \stackrel{\text{as}}{=} Q_0 \oplus^n I * A_0 \stackrel{\text{as}}{=} Q_0 \&\&^n d * \\ A_1 \stackrel{\text{as}}{=} Q_1 \&\&^n d * R_0 = \bigoplus A_0 * R_1 = \bigoplus A_1 * R \stackrel{\text{as}}{=} R_0 \oplus R_1 \end{array} \right\} \\
& \left\{ (Q_0, Q_1, A_0, A_1, R_0, R_1, R). \text{own } I * Q_0 \sim \text{Ber}^n 1/2 * Q_1 \stackrel{\text{as}}{=} Q_0 \oplus^n I \right\} \\
& \quad \vdots \\
& \{(Q_0, Q_1, A_0, A_1, R_0, R_1, R). (\text{own } I * \text{own } Q_0) \wedge (\text{own } I * \text{own } Q_1)\}
\end{aligned}$$

The final entailment (hidden by the vertical ellipses) can be established as follows. The left conjunct of the postcondition follows from $Q_0 \sim \text{Ber}^n 1/2 \vdash \text{own } Q_0$ and dropping the equality $Q_1 \stackrel{\text{as}}{=} Q_0 \oplus^n I$.

¹⁹Following Barthe et al. [2019], we don't even need require that I always have only a single component set to \top to establish input independence.

The right conjunct is established by the following chain of entailments:

$$\begin{aligned}
& \text{own } I * Q_0 \sim \text{Ber}^n 1/2 * Q_1 \stackrel{\text{as}}{=} Q_0 \oplus^n I \\
& \vdash \mathbf{C}_{i \leftarrow I}^{\text{own}} (Q_0 \sim \text{Ber}^n 1/2 * Q_1 \stackrel{\text{as}}{=} Q_0 \oplus^n I) \quad \text{Lemma F.4} \\
& \vdash \mathbf{C}_{i \leftarrow I}^{\text{own}} (Q_0 \sim \text{Ber}^n 1/2 * Q_1 \stackrel{\text{as}}{=} Q_0 \oplus^n i) \quad \text{substituting } i \text{ for } I \\
& \vdash \mathbf{C}_{i \leftarrow I}^{\text{own}} (Q_0 \sim \text{Ber}^n 1/2 * (Q_1 \oplus^n i) \stackrel{\text{as}}{=} Q_0) \quad \text{rearranging the equality} \\
& \vdash \mathbf{C}_{i \leftarrow I}^{\text{own}} ((Q_1 \oplus^n i) \sim \text{Ber}^n 1/2) \quad \text{substituting away } Q_0 \\
& \vdash \mathbf{C}_{i \leftarrow I}^{\text{own}} (Q_1 \sim \text{Ber}^n 1/2) \quad \text{Lemma F.1} \\
& \vdash \text{own } I * Q_1 \sim \text{Ber}^n 1/2 \quad \text{Lemma F.3} \\
& \vdash \text{own } I * \text{own } Q_1 \quad \text{by } X \sim \mu \vdash \text{own } X
\end{aligned}$$

F.3 Verification of oblivious transfer example

The oblivious transfer protocol is modelled by the following probabilistic program:

```

R0 ← flipn 1/2;
R1 ← flipn 1/2;
D ← flip 1/2;
(RD, R¬D) ← if D then ret (R1, R0) else ret (R0, R1);
E ← ret (c ⊕ D);
(F0, F1) ← if E then ret (m0 ⊕ R1, m1 ⊕ R0) else ret (m0 ⊕ R0, m1 ⊕ R1);
(mc, F¬c) ← if c then ret (F1 ⊕ RD, m0 ⊕ R¬D) else ret (F0 ⊕ RD, m1 ⊕ R¬D);
ret (R0, R1, D, RD, R¬D, E, F0, F1, mc, F¬c)

```

(OBLIVIOUSTRANSFER)

This program is parameterized by the two messages m_0 and m_1 on offer and the bit c encoding the receiver's choice.

F.3.1 Uniformity. Uniformity is specified by the triple

$$\{\top\} \text{ OBLIVIOUSTRANSFER}(c, m_0, m_1) \left\{ \begin{pmatrix} R_0, R_1, D, R_D, R_{\neg D} \\ E, F_0, F_1, m_c, F_{\neg c} \end{pmatrix} \cdot ((R_0, R_1) \sim \text{Ber}^{2n} 1/2 * E \sim \text{Ber} 1/2) \wedge \right. \\
\left. (D \sim \text{Ber} 1/2 * (R_D, F_{\neg c}) \sim \text{Ber}^{2n} 1/2) \right\}.$$

The next page gives an annotated program that establishes this specification.

$$\begin{aligned}
& \{\top\} \\
& R_0 \leftarrow \text{flip}^n 1/2; \\
& \{R_0 \sim \text{Ber}^n 1/2\} \\
& R_1 \leftarrow \text{flip}^n 1/2; \\
& \{R_0 \sim \text{Ber}^n 1/2 * R_1 \sim \text{Ber}^n 1/2\} \\
& D \leftarrow \text{flip} 1/2; \\
& \{R_0 \sim \text{Ber}^n 1/2 * R_1 \sim \text{Ber}^n 1/2 * D \sim \text{Ber} 1/2\} \\
& (R_D, R_{\neg D}) \leftarrow \text{if } D \text{ then ret } (R_1, R_0) \text{ else ret } (R_0, R_1); \\
& \left\{ \begin{array}{l} R_0 \sim \text{Ber}^n 1/2 * R_1 \sim \text{Ber}^n 1/2 * D \sim \text{Ber} 1/2 * \\ R_D \stackrel{\text{as}}{=} (\text{if } D \text{ then } R_1 \text{ else } R_0) * R_{\neg D} \stackrel{\text{as}}{=} (\text{if } D \text{ then } R_0 \text{ else } R_1) \end{array} \right\} \\
& E \leftarrow \text{ret } (c \oplus D); \\
& \left\{ \begin{array}{l} R_0 \sim \text{Ber}^n 1/2 * R_1 \sim \text{Ber}^n 1/2 * D \sim \text{Ber} 1/2 * \\ R_D \stackrel{\text{as}}{=} (\text{if } D \text{ then } R_1 \text{ else } R_0) * R_{\neg D} \stackrel{\text{as}}{=} (\text{if } D \text{ then } R_0 \text{ else } R_1) * E \stackrel{\text{as}}{=} c \oplus D \end{array} \right\} \\
& (F_0, F_1) \leftarrow \text{if } E \text{ then ret } (m_0 \oplus R_1, m_1 \oplus R_0) \text{ else ret } (m_0 \oplus R_0, m_1 \oplus R_1); \\
& \left\{ \begin{array}{l} R_0 \sim \text{Ber}^n 1/2 * R_1 \sim \text{Ber}^n 1/2 * D \sim \text{Ber} 1/2 * \\ R_D \stackrel{\text{as}}{=} (\text{if } D \text{ then } R_1 \text{ else } R_0) * R_{\neg D} \stackrel{\text{as}}{=} (\text{if } D \text{ then } R_0 \text{ else } R_1) * E \stackrel{\text{as}}{=} c \oplus D * \\ F_0 \stackrel{\text{as}}{=} (\text{if } E \text{ then } m_0 \oplus R_1 \text{ else } m_1 \oplus R_0) * F_1 \stackrel{\text{as}}{=} (\text{if } D \text{ then } m_0 \oplus R_0 \text{ else } m_1 \oplus R_1) \end{array} \right\} \\
& (m_c, F_{\neg c}) \leftarrow \text{if } c \text{ then ret } (F_1 \oplus R_D, m_0 \oplus R_{\neg D}) \text{ else ret } (F_0 \oplus R_D, m_1 \oplus R_{\neg D}); \\
& \underbrace{\left\{ \begin{array}{l} R_0 \sim \text{Ber}^n 1/2 * R_1 \sim \text{Ber}^n 1/2 * D \sim \text{Ber} 1/2 * \\ R_D \stackrel{\text{as}}{=} (\text{if } D \text{ then } R_1 \text{ else } R_0) * R_{\neg D} \stackrel{\text{as}}{=} (\text{if } D \text{ then } R_0 \text{ else } R_1) * E \stackrel{\text{as}}{=} c \oplus D * \\ F_0 \stackrel{\text{as}}{=} (\text{if } E \text{ then } m_0 \oplus R_1 \text{ else } m_1 \oplus R_0) * F_1 \stackrel{\text{as}}{=} (\text{if } D \text{ then } m_0 \oplus R_0 \text{ else } m_1 \oplus R_1) * \\ m_c \stackrel{\text{as}}{=} (\text{if } c \text{ then } F_1 \oplus R_D \text{ else } F_0 \oplus R_D) * F_{\neg c} \stackrel{\text{as}}{=} (\text{if } c \text{ then } m_0 \oplus R_{\neg D} \text{ else } m_1 \oplus R_{\neg D}) \end{array} \right\}}_P \\
& \text{ret } (R_0, R_1, D, R_D, R_{\neg D}, E, F_0, F_1, m_c, F_{\neg c}) \\
& \{(R_0, R_1, D, R_D, R_{\neg D}, E, F_0, F_1, m_c, F_{\neg c}). P\} \\
& \vdots \\
& \underbrace{\left\{ \begin{array}{l} (R_0, R_1) \sim \text{Ber}^{2n} 1/2 * E \sim \text{Ber} 1/2 \wedge \\ (D \sim \text{Ber} 1/2 * (R_D, F_{\neg c}) \sim \text{Ber}^{2n} 1/2) \end{array} \right\}}_Q \\
& \underbrace{\hspace{10em}}_R
\end{aligned}$$

The final entailment (hidden by the vertical ellipses), abbreviated $P \vdash Q \wedge R$, can be established as follows. First, to show $P \vdash Q$,

$$\begin{array}{ll}
 P & \\
 \vdash R_0 \sim \text{Ber}^n 1/2 * R_1 \sim \text{Ber}^n 1/2 * D \sim \text{Ber} 1/2 * E \stackrel{\text{as}}{=} c \oplus D & \text{dropping conjuncts} \\
 \vdash R_0 \sim \text{Ber}^n 1/2 * R_1 \sim \text{Ber}^n 1/2 * D \sim \text{Ber} 1/2 * E \oplus c \stackrel{\text{as}}{=} D & \text{rearranging the equality} \\
 \vdash R_0 \sim \text{Ber}^n 1/2 * R_1 \sim \text{Ber}^n 1/2 * (E \oplus c) \sim \text{Ber} 1/2 & \text{substituting away } D \\
 \vdash R_0 \sim \text{Ber}^n 1/2 * R_1 \sim \text{Ber}^n 1/2 * E \sim \text{Ber} 1/2 & \text{Lemma F.1} \\
 \vdash (R_0, R_1) \sim \text{Ber}^{2n} 1/2 * E \sim \text{Ber} 1/2 & \text{Lemma F.2} \\
 = Q &
 \end{array}$$

Second, $P \vdash R$ can be established by the following chain:

$$\begin{array}{ll}
 P & \\
 \vdash \left(R_0 \sim \text{Ber}^n 1/2 * R_1 \sim \text{Ber}^n 1/2 * D \sim \text{Ber} 1/2 * \right. & \\
 \quad \left. R_D \stackrel{\text{as}}{=} (\text{if } D \text{ then } R_1 \text{ else } R_0) * R_{\neg D} \stackrel{\text{as}}{=} (\text{if } D \text{ then } R_0 \text{ else } R_1) * \right. & \text{dropping some conjuncts} \\
 \quad \left. F_{\neg c} \stackrel{\text{as}}{=} (\text{if } c \text{ then } m_0 \oplus R_{\neg D} \text{ else } m_1 \oplus R_{\neg D}) \right) & \\
 \vdash \dots & \\
 \vdash \left(R_D \sim \text{Ber}^n 1/2 * R_{\neg D} \sim \text{Ber}^n 1/2 * D \sim \text{Ber} 1/2 * \right. & \\
 \quad \left. F_{\neg c} \stackrel{\text{as}}{=} (\text{if } c \text{ then } m_0 \oplus R_{\neg D} \text{ else } m_1 \oplus R_{\neg D}) \right) & \\
 \vdash \left(R_D \sim \text{Ber}^n 1/2 * R_{\neg D} \sim \text{Ber}^n 1/2 * D \sim \text{Ber} 1/2 * \right. & \text{commuting conversion} \\
 \quad \left. F_{\neg c} \stackrel{\text{as}}{=} (\text{if } c \text{ then } m_0 \text{ else } m_1) \oplus R_{\neg D} \right) & \\
 \vdash \left(R_D \sim \text{Ber}^n 1/2 * R_{\neg D} \sim \text{Ber}^n 1/2 * D \sim \text{Ber} 1/2 * \right. & \text{property of } \oplus \\
 \quad \left. (\text{if } c \text{ then } m_0 \text{ else } m_1) \oplus F_{\neg c} \stackrel{\text{as}}{=} R_{\neg D} \right) & \\
 \vdash \left(R_D \sim \text{Ber}^n 1/2 * D \sim \text{Ber} 1/2 * \right. & \text{substitute away } R_{\neg D} \\
 \quad \left. ((\text{if } c \text{ then } m_0 \text{ else } m_1) \oplus F_{\neg c}) \sim \text{Ber}^n 1/2 \right) & \\
 \vdash R_D \sim \text{Ber}^n 1/2 * D \sim \text{Ber} 1/2 * F_{\neg c} \sim \text{Ber}^n 1/2 & \text{Lemma F.1} \\
 \vdash D \sim \text{Ber} 1/2 * (R_D, F_{\neg c}) \sim \text{Ber}^{2n} 1/2 & \text{Lemma F.2} \\
 = R &
 \end{array}$$

The entailment hidden by ellipses establishes the mutual independence of D , R_D , and $R_{\neg D}$, the key property that perfect secrecy hinges on. The proof goes by case analysis on D , and is shown on the next page.

$$\begin{array}{ll}
\frac{\left(\begin{array}{l} R_0 \sim \text{Ber}^n 1/2 * R_1 \sim \text{Ber}^n 1/2 * D \sim \text{Ber} 1/2 * \\ R_D \stackrel{\text{as}}{=} (\text{if } D \text{ then } R_1 \text{ else } R_0) * R_{\neg D} \stackrel{\text{as}}{=} (\text{if } D \text{ then } R_0 \text{ else } R_1) \end{array} \right)}{\vdash \left(\begin{array}{l} (R_0, R_1) \sim \text{Ber}^{2n} 1/2 * D \sim \text{Ber} 1/2 * \\ R_D \stackrel{\text{as}}{=} (\text{if } D \text{ then } R_1 \text{ else } R_0) * R_{\neg D} \stackrel{\text{as}}{=} (\text{if } D \text{ then } R_0 \text{ else } R_1) \end{array} \right)} & \text{Lemma F.2} \\
\hline
\vdash S[D \sim \text{Ber} 1/2] & S[D \sim \text{Ber} 1/2] \\
\vdash S[D \sim \text{Ber} 1/2] \wedge S[D \sim \text{Ber} 1/2] & \text{property of } \wedge \\
\vdash S[D \sim \text{Ber} 1/2] \wedge (D \sim \text{Ber} 1/2) & \text{drop conjuncts} \\
\vdash S[\text{own } D] \wedge (D \sim \text{Ber} 1/2) & X \sim \mu \vdash \text{own } X \\
\vdash \mathbf{C}_{d \leftarrow D}^{\text{own}} \left((R_0, R_1) \sim \text{Ber}^{2n} 1/2 * R_D \stackrel{\text{as}}{=} (\text{if } D \text{ then } R_1 \text{ else } R_0) * \dots \right) \wedge (D \sim \text{Ber} 1/2) & \text{Lemma F.4} \\
\vdash \mathbf{C}_{d \leftarrow D}^{\text{own}} \left(\begin{array}{l} (R_0, R_1) \sim \text{Ber}^{2n} 1/2 * \\ R_D \stackrel{\text{as}}{=} (\text{if } d \text{ then } R_1 \text{ else } R_0) * R_{\neg D} \stackrel{\text{as}}{=} \dots \end{array} \right) \wedge (D \sim \text{Ber} 1/2) & \text{replace } D \text{ with } d \\
\hline
\vdash \mathbf{C}_{d \leftarrow D}^{\text{own}} \left(\underbrace{Q(d)} \right) \wedge (D \sim \text{Ber} 1/2) & \\
\vdash \mathbf{C}_{d \leftarrow D}^{\text{own}} (Q(\top) \wedge Q(\text{F})) \wedge (D \sim \text{Ber} 1/2) & \text{cases on } d \\
\hline
\vdash \mathbf{C}_{d \leftarrow D}^{\text{own}} \left(\underbrace{Q(\top)} \right) \wedge (D \sim \text{Ber} 1/2) & \\
\vdash \mathbf{C}_{d \leftarrow D}^{\text{own}} \left(\underbrace{\left(\begin{array}{l} (R_0, R_1) \sim \text{Ber}^{2n} * R_D \stackrel{\text{as}}{=} R_1 * R_{\neg D} \stackrel{\text{as}}{=} R_0 \\ (R_0, R_1) \sim \text{Ber}^{2n} 1/2 * R_D \stackrel{\text{as}}{=} R_0 * R_{\neg D} \stackrel{\text{as}}{=} R_1 \end{array} \right) \wedge}_{Q(\text{F})} \right) \wedge (D \sim \text{Ber} 1/2) & \text{unfold } Q \\
\vdash \mathbf{C}_{d \leftarrow D}^{\text{own}} \left(\left((R_{\neg D}, R_D) \sim \text{Ber}^{2n} 1/2 \wedge ((R_D, R_{\neg D}) \sim \text{Ber}^{2n} 1/2) \right) \wedge (D \sim \text{Ber} 1/2) \right) & \text{substitute} \\
\vdash \mathbf{C}_{d \leftarrow D}^{\text{own}} \left(\begin{array}{l} (R_{\neg D} \sim \text{Ber}^n 1/2 * R_D \sim \text{Ber}^n 1/2) \wedge \\ (R_D \sim \text{Ber}^n 1/2 * R_{\neg D} \sim \text{Ber}^n 1/2) \end{array} \right) \wedge (D \sim \text{Ber} 1/2) & \text{Lemma F.2} \\
\vdash \mathbf{C}_{d \leftarrow D}^{\text{own}} (R_{\neg D} \sim \text{Ber}^n 1/2 * R_D \sim \text{Ber}^n 1/2) \wedge (D \sim \text{Ber} 1/2) & \text{by } P \wedge P \vdash P \\
\vdash \mathbf{C}_{d \leftarrow D}^{\text{own}} ((R_D, R_{\neg D}) \sim \text{Ber}^{2n} 1/2) \wedge (D \sim \text{Ber} 1/2) & \text{Lemma F.2} \\
\vdash (\text{own } D * R_{\neg D} \sim \text{Ber}^n 1/2 * R_D \sim \text{Ber}^n 1/2) \wedge (D \sim \text{Ber} 1/2) & \text{Lemma F.3} \\
\vdash D \sim \text{Ber} 1/2 * R_{\neg D} \sim \text{Ber}^n 1/2 * R_D \sim \text{Ber}^n 1/2 & \text{Lemma F.5}
\end{array}$$

F.3.2 Input independence. Following Barthe et al. [2019], the following Hoare triple specifies input independence for **OBLIVIOUSTRANSFER**:

$$\begin{array}{l}
\{ \text{own}(C, M_0, M_1) \} \\
\text{OBLIVIOUSTRANSFER}(M_0, M_1, C) \\
\left\{ (R_0, R_1, D, R_D, R_{\neg D}, E, F_0, F_1, M_C, M_{1-C}, F_{\neg C}). \begin{array}{l} (\text{own } C * \text{own}(R_0, R_1, E)) \wedge \\ (\text{own } M_{1-C} * \text{own}(D, R_D, F_0, F_1)) \end{array} \right\}
\end{array}$$

In the postcondition, M_{1-C} denotes the random variable (if C then M_0 else M_1).

Barthe et al. [2019] observe that the proof gets stuck, mentioning that even an informal proof sketch does not seem easy. We show that this triple is in fact impossible to establish by giving an explicit counterexample. This takes the form of three random variables C , M_0 , and M_1 such that the postcondition fails. In particular, we choose $C = 0$ and $M_0 = M_1 = M$ for some uniformly-distributed M . Now the triple reads

$$\begin{aligned} & \{\text{own}(C, M, M)\} \\ & \text{OBLIVIOUSTRANSFER}(M, M, C) \\ & \left\{ (R_0, R_1, D, R_D, R_{\neg D}, E, F_0, F_1, M_C, M_{1-C}, F_{\neg C}). \begin{array}{l} (\text{own } C * \text{own}(R_0, R_1, E)) \wedge \\ (\text{own } M * \text{own}(D, R_D, F_0, F_1)) \end{array} \right\} \end{aligned}$$

Suppose this triple holds. Then so does the following triple, where we have dropped the first conjunct of the postcondition:

$$\begin{aligned} & \{\text{own}(C, M, M)\} \\ & \text{OBLIVIOUSTRANSFER}(M, M, C) \\ & \{(R_0, R_1, D, R_D, R_{\neg D}, E, F_0, F_1, M_C, M_{1-C}, F_{\neg C}). (\text{own } M * \text{own}(D, R_D, F_0, F_1))\} \end{aligned}$$

Lilac's semantic model validates the following derived rule:

LEMMA F.6. *Let X be a random variable and $f : \text{cod}(X) \rightarrow B$ measurable. Then $\text{own } X \vdash \text{own}(f \circ X)$.*

PROOF. The composition of measurable maps remains measurable. \square

Thus we have that $\text{own}(D, R_D, F_0, F_1) \vdash \text{own}(R_D \oplus F_0)$, so the following triple holds:

$$\begin{aligned} & \{\text{own}(C, M, M)\} \\ & \text{OBLIVIOUSTRANSFER}(M, M, C) \\ & \{(R_0, R_1, D, R_D, R_{\neg D}, E, F_0, F_1, M_C, M_{1-C}, F_{\neg C}). (\text{own } M * \text{own}(R_D \oplus F_0))\} \end{aligned}$$

At this point we transition from working in Lilac to reading off the meanings of Lilac propositions in our semantic model. We know that, by forward symbolic execution as in the previous section, the OT protocol sets

$$\begin{aligned} F_0 & \stackrel{\text{as}}{=} (\text{if } E \text{ then } M_0 \oplus R_1 \text{ else } M_1 \oplus R_0) \\ E & \stackrel{\text{as}}{=} C \oplus D \\ R_D & \stackrel{\text{as}}{=} (\text{if } D \text{ then } R_1 \text{ else } R_0) \end{aligned}$$

We have set $M_0 = M_1 = M$ and $C = 0$ for the sake of contradiction, so these equations become

$$\begin{aligned} F_0 & \stackrel{\text{as}}{=} M \oplus (\text{if } E \text{ then } R_1 \text{ else } R_0) \\ E & \stackrel{\text{as}}{=} D \\ R_D & \stackrel{\text{as}}{=} (\text{if } D \text{ then } R_1 \text{ else } R_0) \end{aligned}$$

Further substitution gives

$$F_0 \stackrel{\text{as}}{=} M \oplus (\text{if } D \text{ then } R_1 \text{ else } R_0) \stackrel{\text{as}}{=} M \oplus R_D.$$

Reading the final Hoare triple obtained above in terms of the model, we have that $\llbracket \text{OBLIVIOUSTRANSFER} \rrbracket$ is a Markov kernel whose pushforward along the distribution on (C, M, M) gives a distribution on

$$(R_0, R_1, D, R_D, R_{\neg D}, E, F_0, F_1, M_C, M_{1-C}, F_{\neg C})$$

in which M is independent of $(R_D \oplus F_0)$. But we also have

$$R_D \oplus F_0 = R_D \oplus (M \oplus R_D) = M$$

by the above deduction, so this Hoare triple asserts the self-independence of M . This is a contradiction: we have chosen M to be a uniformly-distributed n -tuple of boolean values, which cannot be self-independent.

Intuitively, the special case $C = 0$ and $M_0 = M_1 = M$ that we have chosen is the situation where both messages offered by the sender are exactly the same and the receiver always opts to receive message 0. The failure of input independence corresponds to the fact that the receiver manages to learn what message 1 is. But it only learns message 1 because in this particular situation the two messages happen to be the same! Thus this counterexample appears to be more an issue with this particular specification of perfect secrecy via input independence than a vulnerability in the OT protocol.