Probabilistic Concurrent Reasoning in Outcome Logic: Independence, Conditioning, and Invariants

NOAM ZILBERSTEIN, Cornell University, USA
ALEXANDRA SILVA, Cornell University, USA
JOSEPH TASSAROTTI, New York University, USA

Although randomization has long been used in distributed computing, formal methods for reasoning about probabilistic concurrent programs have lagged behind. No existing program logics can express specifications about the full *distributions of outcomes* resulting from programs that are both probabilistic and concurrent. To address this, we introduce *Probabilistic Concurrent Outcome Logic* (pcOL), which incorporates ideas from concurrent and probabilistic separation logics into Outcome Logic to introduce new compositional reasoning principles. At its core, pcOL reinterprets the rules of Concurrent Separation Logic in a setting where separation models probabilistic independence, so as to compositionally describe joint distributions over variables in concurrent threads. Reasoning about outcomes also proves crucial, as case analysis is often necessary to derive precise information about threads that rely on randomized shared state. We demonstrate pcOL on a variety of examples, including to prove almost sure termination of unbounded loops.

CCS Concepts: • **Theory of computation → Separation logic**; **Logic and verification**; **Program verification**; **Concurrency**; **Probabilistic computation**.

Additional Key Words and Phrases: Outcome Logic, Separation Logic, Concurrency, Probabilistic Programming

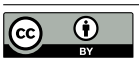
ACM Reference Format:
Noam Zilberstein, Alexandra Silva, and Joseph Tassarotti. 2026. Probabilistic Concurrent Reasoning in Outcome Logic: Independence, Conditioning, and Invariants. *Proc. ACM Program. Lang.* 10, POPL, Article 9 (January 2026), 72 pages. https://doi.org/10.1145/3776651

1 Introduction

Randomization is an important tool in concurrent and distributed computing. Concurrent algorithms can be made more efficient using randomization [Morris 1978; Rabin 1980, 1982] and some distributed synchronization problems have no deterministic solution [Fischer et al. 1985; Lehmann and Rabin 1981]. But despite the prevalence of randomization in concurrent computing over the last several decades, formal methods for such programs are limited. The mixture of *computational effects* in probabilistic concurrent programs is a major source of difficulty in developing verification techniques; random choice is introduced by sampling operations and nondeterminism arises from scheduling the concurrent threads. These two computational effects do not compose in standard ways [Varacca and Winskel 2006], so even just describing the semantics of such programs requires specialized models [He et al. 1997; McIver and Morgan 2005; Zilberstein et al. 2025a,b].

In this paper, we introduce *Probabilistic Concurrent Outcome Logic* (pcOL), a logic for reasoning about programs that are both probabilistic and concurrent. In pcOL, preconditions and postconditions are not just assertions about a single program state. Instead, they describe the *distribution of*

Authors' Contact Information: Noam Zilberstein, noamz@cs.cornell.edu, Cornell University, USA; Alexandra Silva, alexandra.silva@cornell.edu, Cornell University, USA; Joseph Tassarotti, jt4767@nyu.edu, New York University, USA.



This work is licensed under a Creative Commons Attribution 4.0 International License.
© 2026 Copyright held by the owner/author(s).
ACM 2475-1421/2026/1-ART9
https://doi.org/10.1145/3776651

possible outcomes that can arise from executing the program. A key challenge is that, in the concurrent setting, different orderings of threads can give rise to different distributions over program behaviors. To address this, pcOL takes inspiration from the recently introduced Demonic Outcome Logic (dOL) [Zilberstein et al. 2025b], which supports reasoning about sequential probabilistic programs that additionally have a nondeterministic choice operator resolved by an adversary. Different nondeterministic choices can cause different distributions of behaviors in these programs, just as different thread interleavings can cause different distributions in the concurrent setting.

However, while dOL's approach to describing the space of possible distributions provides a basis for pcOL, reasoning about the nondeterminism that arises from concurrent scheduling is substantially more complicated than reasoning about a choice operator. With a choice operator, nondeterminism is *localized* to the points where the operator is used, whereas in a concurrent program, every step can potentially involve nondeterminism from thread interleaving. Reasoning explicitly about nondeterminism at every step is intractable and non-compositional.

To recover compositional reasoning, pcOL incorporates ideas from various *separation logics*. Concurrent Separation Logic (CSL) uses disjointness of resources to ensure that concurrent computations only interact in controlled ways, so that each thread can be analyzed on its own [Brookes 2004; O'Hearn 2004]. Probabilistic Separation Logics (PSL) use the notions of independence and conditioning to reason about the interaction between randomness and control flow [Bao et al. 2021, 2025, 2022; Barthe et al. 2020; Li et al. 2023; Yan et al. 2025]. CSL and PSL achieve compositional reasoning in concurrent and probabilistic settings, respectively, so combining their reasoning principles appears to be a natural way to derive a compositional logic for the combination of both effects. However, as we will see in Section 2, such a combination is challenging to achieve because the metatheories of the two logics are highly specialized to their respective domains, and a direct combination of their rules would not be sound. This paper develops the metatheory in a more complex semantic domain, where concurrency and probabilistic computation can coexist. As a result, pcOL is the first logic to combine all of the following features:

Compositional Concurrency Reasoning. pcOL supports *compositional* concurrency reasoning, meaning that each thread in a concurrent program can be analyzed in isolation, without considering all the possible interleavings or behaviors of the scheduler. Similar to Concurrent Separation Logic (CSL), compositionality stems from *separation*—as long as two threads operate on their own portions of memory, they cannot interfere with each other. In addition, shared state is handled via *resource invariants*, properties about shared state that remain true at every step.

Compositional Probabilistic Reasoning. In a probabilistic context, separation of memory footprints is not sufficient; while it can tell us how the local variables of each thread are distributed, it does not give us the *joint distribution* over the entire global memory. In response, we use a PSL-style model where separation additionally models *probabilistic independence* [Barthe et al. 2020]. Going beyond prior work, our parallel composition rule guarantees that the scheduler cannot introduce any probabilistic correlation between the local states of each thread.

Compositional Outcome Reasoning. When concurrent threads depend on randomized shared state, it is often necessary to do case analysis over the possible values of that shared state in order to capture the probabilistic correlation between the threads (*e.g.,* see Section 2.2). In the style of Demonic Outcome Logic (dOL) [Zilberstein et al. 2025b], pcOL supports compositional reasoning about the outcomes generated via both probabilistic branching and the nondeterministic behavior of the scheduler. But unlike dOL—which does not support separation—case analysis must be done with care, as it can invalidate the independence guarantees of the Frame rule. Compared to prior outcome logics, the *outcome conjunction* of pcOL has a new measure theoretic foundation.

Unbounded Looping and Almost Sure Termination. Our logic includes rules for establishing *almost sure termination*—termination with probability 1—for unbounded loops. This goes beyond the capabilities of all prior separation logics that model separation with probabilistic independence, which either have only bounded looping constructs (*e.g.,* for loops) or require loops to always terminate (which must be established externally to the logic). Unbounded looping is important in randomized concurrent programs, as such programs often only achieve the desired distribution of outcomes in the limit, and not after a bounded number of steps (*e.g.,* Section 6.4).

We begin in Section 2 with an overview of the technical challenges and design of pcOL. Next, in Section 3 we outline the programming language and semantic model that we will use. The logic and inference rules are defined in Sections 4 and 5. We demonstrate the capabilities of pcOL on four case studies in Section 6. Finally, we conclude by discussing related work and future directions in Sections 7 and 8. Omitted proofs and details are given in the appendix [Zilberstein et al. 2025c].

2 Overview: Familiar Reasoning Principles in a New Setting

In this paper, we develop a logic for verifying the correctness of randomized concurrent imperative programs, written in a language that includes control flow operations (if statements and while loops), parallel composition $C_1 \parallel C_2$, and random sampling $x :\approx d$. A major hurdle in concurrency analysis is that the semantics is *non-compositional*; two programs can have completely different behavior when run in parallel than they do when run in isolation. Enumerating all possible interleavings of the threads is not a viable strategy, so abstractions must be introduced for sound compositional analysis. In this section, we explore the mechanisms that enable compositional reasoning about probabilistic concurrent programs in pcOL, including the challenges that arise with shared state.

2.1 Concurrent Separation Logic meets Probabilistic Separation Logic

Concurrent Separation Logic (CSL) achieves compositionality via *separation* [Brookes 2004; O'Hearn 2004]—if two threads act on disjoint memory regions, then their behavior will not change when run in parallel. This idea is encapsulated by the Par rule, where the *separating conjunction* $\varphi * \psi$ means that the machines's memory cells can be divided to satisfy φ and ψ individually. In addition, the Frame rule guarantees that threads cannot interfere with memory outside of their local state.

$$\frac{\langle \varphi_1 \rangle\ C_1\ \langle \psi_1 \rangle \qquad \langle \varphi_2 \rangle\ C_2\ \langle \psi_2 \rangle}{\langle \varphi_1 * \varphi_2 \rangle\ C_1 \parallel C_2\ \langle \psi_1 * \psi_2 \rangle}\textsc{Par} \qquad \frac{\langle \varphi \rangle\ C\ \langle \psi \rangle}{\langle \varphi * \vartheta \rangle\ C\ \langle \psi * \vartheta \rangle}\textsc{Frame}$$

Now suppose that C_1 and C_2 in the Par rule are probabilistic programs. For example, if we flip two coins and store the results in the variables x and y, then the respective variables will be distributed according to Bernoulli distributions with parameter $\frac{1}{2}$, as shown in the following specifications, where $\lceil P \rceil$ means that P holds with probability 1 (almost surely), $x \mapsto -$ means that the current thread has permission to read and write x, and $x \sim d$ means that x is distributed according to d.

$$\langle \lceil x \mapsto - \rceil \rangle\ x :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right)\ \langle x \sim \mathbf{Ber}\left(\tfrac{1}{2}\right) \rangle \qquad \langle \lceil y \mapsto - \rceil \rangle\ y :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right)\ \langle y \sim \mathbf{Ber}\left(\tfrac{1}{2}\right) \rangle$$

Composing these programs in parallel, we would ideally want to derive a specification that dictates not only how x and y are distributed, but also their *joint distribution*. In this case, x and y are *probabilistically independent*, meaning that each outcome (*e.g.,* x and y are both 1) occurs with probability $\frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$. Thus, it is natural to consider using the interpretation of separation proposed in Probabilistic Separation Logic (PSL) [Barthe et al. 2020], in which $\varphi * \psi$ states that the events described by φ and ψ are probabilistically independent. By treating separation as *both* disjointness of memory *and* probabilistic independence, one might hope to validate a probabilistic interpretation

of the Par rule, with which we could derive the following specification.

$$\langle \lceil x \mapsto - \rceil * \lceil y \mapsto - \rceil \rangle\ x :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right) \parallel y :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right)\ \langle x \sim \mathbf{Ber}\left(\tfrac{1}{2}\right) * y \sim \mathbf{Ber}\left(\tfrac{1}{2}\right) \rangle$$

Intuitively, the probabilistic interpretation of Par in this instance is justifiable because C_1 and C_2 execute without interaction, so there is no correlation between their random behaviors. The lack of correlation is due to the fact that there is no shared state, and therefore the nondeterminism introduced by the interleaving behavior of the scheduler is not observable in any way.

Bigger challenges arise when the threads *do* share state, making the nondeterministic behavior of the scheduler observable. CSL handles shared state with *resource invariants*—threads can interact with shared state as long as the invariant I is preserved by every atomic step. More specifically, CSL provides the following inference rules: a more general Par rule allows the invariant resources to be used in both threads; the Atom rule *opens* the invariant, as long as the program is a single atomic command a; and Share allocates an invariant that is true before and after the program execution.

$$\frac{I \vdash \langle \varphi_1 \rangle\ C_1\ \langle \psi_1 \rangle \quad I \vdash \langle \varphi_2 \rangle\ C_2\ \langle \psi_2 \rangle}{I \vdash \langle \varphi_1 * \varphi_2 \rangle\ C_1 \parallel C_2\ \langle \psi_1 * \psi_2 \rangle}\textsc{Par} \qquad \frac{\vdash \langle \varphi * \lceil I \rceil \rangle\ a\ \langle \psi * \lceil I \rceil \rangle}{I \vdash \langle \varphi \rangle\ a\ \langle \psi \rangle}\textsc{Atom} \qquad \frac{I \vdash \langle \varphi \rangle\ C\ \langle \psi \rangle}{\vdash \langle \varphi * \lceil I \rceil \rangle\ C\ \langle \psi * \lceil I \rceil \rangle}\textsc{Share}$$

Under a probabilistic interpretation of $*$, this stronger Par rule is valid when the shared state described by the invariant I is *deterministic*, for example, in the following program, where z has a fixed value, which is read from both threads.

$$z \mapsto 1 \vdash \langle \lceil x \mapsto - \rceil * \lceil y \mapsto - \rceil \rangle\ x :\approx \mathbf{Ber}\left(\tfrac{z}{2}\right) \parallel y :\approx \mathbf{Ber}\left(\tfrac{z}{2}\right)\ \langle (x \sim \mathbf{Ber}\left(\tfrac{1}{2}\right)) * (y \sim \mathbf{Ber}\left(\tfrac{1}{2}\right)) \rangle$$

However, with nondeterministic or randomized shared state, reasoning about programs becomes more complicated because correlations can be introduced in very subtle ways. In the remainder of this section, we will see a few such representative scenarios, and how they are handled in pcOL.

2.2 Handling Randomized Shared State with Outcome Logic

When shared state is randomized, compositional reasoning about outcomes becomes essential. As an example, consider the following program, where x and y read from the shared randomized variable z in different threads.

$$z :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right) \fatsemi (x := z \parallel y := 1 - z) \tag{1}$$

Here, x and y are clearly not independently distributed, since both are derived from z. So it seems dubious that Par could be used to reason about this program. The key observation is that x and y are *conditionally* independent on z, so we can compositionally reason about the threads by first breaking down the outcomes of the sampling operation such that z is deterministic in each case. We draw inspiration from the logics DIBI [Bao et al. 2021], Lilac [Li et al. 2023], and Bluebell [Bao et al. 2025], which include constructs to reason about conditioning, but which are not sufficient for the kind of analysis needed here. DIBI's model of separation is too coarse (see Section 2.3), Lilac has no rule for case analysis over conditioning modalities, and Bluebell's case analysis rule (c-wp-swap) has side conditions which would preclude adapting it for use in parallel composition.

To support *both* case analysis *and* parallel composition, in pcOL we introduce an *outcome conjunction* $\bigoplus_{X \sim d} \varphi$, which binds a new logical variable X that can be referenced in φ, and is distributed according to d, *e.g.*, $z \sim \mathbf{Ber}\left(\tfrac{1}{2}\right)$ is syntactic sugar for $\bigoplus_{Z \sim \mathbf{Ber}\left(\frac{1}{2}\right)} \lceil z \mapsto Z \rceil$. Outcome conjunctions feature in prior logics [Zhang et al. 2024; Zilberstein 2025; Zilberstein et al. 2023, 2025b, 2024], but in this paper we use a new measure theoretic foundation based on *direct sums* [Fremlin 2001], which interacts well with separation, and represents conditional probabilities via Bayes' Law. As a result, the Split rule—shown below—is admissible in pcOL, which is critical for

concurrency reasoning.[1]

$$\frac{I \vdash \langle \varphi \rangle\ C\ \langle \psi \rangle}{I \vdash \left\langle \bigoplus_{X \sim d} \varphi \right\rangle\ C\ \left\langle \bigoplus_{X \sim d} \psi \right\rangle}\textsc{Split}$$

In the premise of Split, X is unbound, and therefore it is implicitly universally quantified. So, the rule allows us to partition the sample space according to d, reason about the program as if X is a deterministic value, and then once again bind X under an outcome conjunction in the conclusion.

Returning to Example (1), by considering each possible outcome of the sampling operation, we end up in a situation where z is deterministic, and therefore the Par rule can apply. Indeed, the Split rule is the missing piece that we need. After executing the sampling operation, we get the assertion $(z \sim \mathbf{Ber}\left(\frac{1}{2}\right)) * \lceil x \mapsto - * y \mapsto - \rceil$, which is equivalent to $\bigoplus_{Z \sim \mathbf{Ber}\left(\frac{1}{2}\right)} \lceil z \mapsto Z * x \mapsto - * y \mapsto - \rceil$. So, to analyze the parallel composition, all we have to do is apply Split to make z deterministic, and then allocate the invariant $z \mapsto Z$, stating that z has some fixed—but universally quantified—value. The derivation is sketched below and shown fully in Appendix F.1.

$$\dfrac{\dfrac{\dfrac{z \mapsto Z \vdash \langle \lceil x \mapsto - \rceil \rangle\ x := z\ \langle \lceil x \mapsto Z \rceil \rangle \quad z \mapsto Z \vdash \langle \lceil y \mapsto - \rceil \rangle\ y := 1 - z\ \langle \lceil y \mapsto 1 - Z \rceil \rangle}{z \mapsto Z \vdash \langle \lceil x \mapsto - * y \mapsto - \rceil \rangle\ x := z \parallel y := 1 - z\ \langle \lceil x \mapsto Z * y \mapsto 1 - Z \rceil \rangle}\textsc{Par}}{\vdash \langle \lceil z \mapsto Z * x \mapsto - * y \mapsto - \rceil \rangle\ x := z \parallel y := 1 - z\ \langle \lceil z \mapsto Z * x \mapsto Z * y \mapsto 1 - Z \rceil \rangle}\textsc{Share}}{\vdash \left\langle \bigoplus_{Z \sim \mathbf{Ber}\left(\frac{1}{2}\right)} \lceil z \mapsto Z * x \mapsto - * y \mapsto - \rceil \right\rangle\ x := z \parallel y := 1 - z\ \left\langle \bigoplus_{Z \sim \mathbf{Ber}\left(\frac{1}{2}\right)} \lceil z \mapsto Z * x \mapsto Z * y \mapsto 1 - Z \rceil \right\rangle}\textsc{Split}$$

Ultimately, we conclude that x and y are independent inside the scope of the $\bigoplus_{Z \sim \mathbf{Ber}\left(\frac{1}{2}\right)}$—where they are deterministic—while still recording exactly how they are probabilistically correlated. This pattern arises frequently, *e.g.,* in Section 6.2, where we prove the correctness of a concurrent shuffling algorithm in pcOL.

2.3 Taming Nondeterminism with Precise Assertions

In the previous section, we saw a new form of compositional reasoning, but it was limited to scenarios where the shared state could be made deterministic via case analysis. In other words, the nondeterministic scheduling order could not affect the outcome of the program. The effects of scheduling become observable when threads *mutate* shared state, because different interleavings can cause the shared variables to take on different values throughout the program execution.

In fact, treating the scheduler adversarially, shared state provides opportunities for the scheduler to introduce probabilistic correlations in unexpected ways. For example, in the following program, the scheduler can force x and y to be equal by first executing the sampling operation; then, with the value of x fixed, choosing an order for the writes to y so that $y := 1 - x$ is first, and $y := x$ is last.

$$\left(x :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right) \fatsemi y := 0\right) \quad \Big\| \quad y := 1 \tag{2}$$

In that case, x and y are both distributed according to $\mathbf{Ber}\left(\frac{1}{2}\right)$, but are certainly not *independently* distributed. To ensure that the correlations between x and y do not invalidate the Par rule in pcOL, we additionally require the postconditions of each thread to be *precise* (Definition 4.1), essentially meaning that they exactly determine the probability of each event. Any pure assertion $\lceil P \rceil$ is precise—P occurs with probability exactly 1—as is $x \sim d$, since d dictates the probability of $\lceil x \mapsto v \rceil$ for all v. Given that, $(x \sim \mathbf{Ber}\left(\frac{1}{2}\right)) * \lceil y \in \{0, 1\} \rceil$ is a valid—and precise—postcondition for (2).

[1] Our approach of using direct sums has tradeoffs: the outcome conjunction $\bigoplus$ is fundamentally discrete, whereas other logics, notably Lilac [Li et al. 2023], have continuous conditioning modalities. Despite this restriction, our examples will illustrate the expressive power of the logic, as many uses of randomization in concurrent and distributed programs only require discrete distributions. See Section 7 for a deeper comparison with other logics.

It may be surprising that $(x \sim \mathbf{Ber}\left(\frac{1}{2}\right)) * \lceil y \in \{0,1\}\rceil$ holds; as we just saw, x and y are not necessarily independently distributed. However, PCOL uses a *probability-space-as-a-resource* model of separation, first explored by Li et al. [2023], to make separation more flexible. That is, we care only about independence of *measurable events* and not *samples*. In the above case, we need only measure the probability of $\lceil y \in \{0,1\}\rceil$, which occurs with probability 1, and nothing finer such as $\lceil y \mapsto 0\rceil$ and $\lceil y \mapsto 1\rceil$. Independence is therefore trivial, *e.g.*, we get that $x \mapsto 1$ and $y \in \{0,1\}$ with probability $\frac{1}{2} \cdot 1 = \frac{1}{2}$. While it is difficult to compositionally determine the exact set of possible joint distributions resulting from different scheduling behaviors, PCOL neatly abstracts away those semantic considerations using simple syntactic checks on the postcondition.

2.4 Weak Separation and Case Analysis over Shared State

Examples (1) and (2) illustrate how PCOL combines both the disjointness and independence interpretations of separation to obtain a probabilistic PAR rule. However, in some cases, the combination of disjointness and independence is too strong and does not hold, because the scheduler *can* induce correlations between concurrent threads through shared state. While we can sometimes *coarsen* what is measurable to recover independence—as we did with y in (2)—it is not always possible.

In response, PCOL uses a second form of separating conjunction, which we call *weak separation*, written $\varphi *_{\mathrm{w}} \psi$, which only requires that φ and ψ hold for disjoint state, and need not be probabilistically independent. Weak separation allows us to still recover some of the benefits of separation logic for reasoning about disjointness, even when we cannot expect independence to hold. To see an example, consider a thread running the following command, in which control flow depends on a variable x that is part of shared state and may be written by another thread:

$$C \quad \triangleq \quad x' \coloneqq x \fatsemi \mathbf{if}\ x'\ \mathbf{then}\ z :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right)\ \mathbf{else}\ z \coloneqq 1$$

To analyze the program above, we need to do case analysis on the value of x at the moment that it is read. Then, we conclude that regardless of x's value, z is distributed according to a Bernoulli distribution with parameter *at least* $\frac{1}{2}$, motivating the following triple.

$$x \mapsto 0 \vee x \mapsto 1 \vdash \langle \lceil z \mapsto - * x' \mapsto - \rceil \rangle\ C\ \langle \exists X \geq \tfrac{1}{2}.\ z \sim \mathbf{Ber}\,(X)\ \rangle \tag{3}$$

Triple (3) is indeed valid on its own, but it is not compatible with the FRAME rule that we saw in Section 2.1. To see why, consider the extended program below in a case where x is initially 0.

$$(y :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right) \fatsemi C) \parallel (x \coloneqq 1)$$

In the left thread, after the $y :\approx \mathbf{Ber}\left(\frac{1}{2}\right)$, if we apply FRAME with $\vartheta = y \sim \mathbf{Ber}\left(\frac{1}{2}\right)$ and use the triple for C in (3), we would get the postcondition $(\exists X \geq \frac{1}{2}.\ z \sim \mathbf{Ber}\,(X)) * (y \sim \mathbf{Ber}\left(\frac{1}{2}\right))$. But that postcondition is not valid because the scheduler can force y and z to be correlated by using the value of y to influence the value of x.[2] As we show in Section 5.3, case analysis on nondeterministic state causes triples to only be *weakly* frame preserving, meaning that only a variant of the FRAME rule using weak separation $\varphi *_{\mathrm{w}} \psi$ can be used. Under the right conditions, strong frame preservation can be restored, as shown in Sections 5.4, 6.1 and 6.4.

Having now given an overview of PCOL's features, we begin the technical development in Section 3 by outlining a denotational model for probabilistic concurrent programs. We then give a measure theoretic model of assertions—including the separating and outcome conjunctions— in Section 4. In Section 5, we define PCOL triples and provide a proof system. Four examples are shown in Section 6 before we conclude by discussing related work in Section 7 and future directions in Section 8.

[2]For the interested reader: if the scheduler makes x equal to y, then x must also be uniformly distributed, and so $z \sim \mathbf{Ber}\left(\frac{1}{2}\right)$ with probability $\frac{1}{2}$ and $z \mapsto 1$ with probability $\frac{1}{2}$, which implies that $z \sim \mathbf{Ber}\left(\frac{3}{4}\right)$. But clearly, $(z \sim \mathbf{Ber}\left(\frac{3}{4}\right)) * (y \sim \mathbf{Ber}\left(\frac{1}{2}\right))$ does not hold, since whenever $y = 0$, then $z = 1$, and so the probability of $\lceil y \mapsto 0 * z \mapsto 1\rceil$ is $\frac{1}{2}$, which is not equal to $\frac{1}{2} \cdot \frac{3}{4}$.

$$
\begin{aligned}
\mathsf{Cmd} \ni C &::= \mathbf{skip} \mid C_1 \fatsemi C_2 \mid C_1 \parallel C_2 \mid \mathbf{if}\ b\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2 \mid \mathbf{while}\ b\ \mathbf{do}\ C \mid a \\
\mathsf{Act} \ni a &::= x \coloneqq e \mid x :\approx d(e) \\
\mathsf{Dist} \ni d &::= \mathbf{Ber}\,(-) \mid \mathbf{geo}\,(-) \mid \mathbf{unif}\,(-) \\
\mathsf{Test} \ni b &::= \mathsf{true} \mid \mathsf{false} \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \mid \neg b \mid e_1 \asymp e_2 \\
\mathsf{Exp} \ni e &::= x \mid v \mid b \mid e[e'] \mid [e_1, \ldots, e_n] \mid e_1 + e_2 \mid e_1 \cdot e_2 \mid \cdots
\end{aligned}
$$

Fig. 1. Syntax of a probabilistic concurrent language, where $x \in \mathsf{Var}$, $v \in \mathsf{Val}$, and $\asymp \in \{=, \leq, <, \ldots\}$.

3 A Probabilistic and Concurrent Programming Language

We begin by describing the syntax and semantics of a probabilistic concurrent programming language, shown in Figure 1. Program commands $C \in \mathsf{Cmd}$ consist of no-ops (**skip**), sequential composition ($C_1 \fatsemi C_2$), parallel composition ($C_1 \parallel C_2$), if statements, while loops, and actions $a \in \mathsf{Act}$.

Actions can perform probabilistic sampling operations $x :\approx d(e)$, where $d \in \mathsf{Dist}$ is a discrete probability distribution with the expression e as a parameter. We include three types of distributions—Bernoulli distributions $\mathbf{Ber}\,(p)$, assigning probability p to 1 and probability $1-p$ to 0; geometric distributions $\mathbf{geo}\,(p)$, assigning probability $(1-p)^n p$ to each $n \in \mathbb{N}$; and uniform distributions $\mathbf{unif}\,(e)$ where e evaluates to a finite set or list of values $[v_1, \ldots, v_n]$, each having probability $1/n$.

Our restriction to discrete distributions was motivated by the applications that we are targeting, including synchronization protocols, which only require fair coin flips [Ben-Or 1983; Lehmann and Rabin 1981]; cryptography, where keys are uniformly sampled fixed length bit-strings; and randomized sketching data structures, where hashes are modeled as uniform random samples over a finite set [Flajolet 1985]. Semantic domains for combining nondeterminism with continuous probability have been explored [Keimel and Plotkin 2017; Tix et al. 2009], but our approach would need modifications to exploit that. See Section 7 for a discussion.

The language also has deterministic assignments $x \coloneqq e$, where e is an expression. Expressions consist of variables x, values v, tests b, list literals $[e_1, \ldots, e_n]$, list accesses $e[e']$ (where e is a list and e' is an index), and standard arithmetic operations. Many more actions could be added to this semantics, including nondeterministic assignment and atomic concurrency primitives such as compare-and-swap, but we do not explore them in this paper.

We use the recently introduced *Pomsets with Formulae* model [Zilberstein et al. 2025a], which augments typical denotational techniques for concurrency semantics [Gischer 1988; Pratt 1986] in order to properly capture probabilistic behavior. We use a denotational model because PCOL describes *distributions* over program states, which differs from the usual CSL setting where assertions are predicates on a single state. Thus, we cannot adapt the Vafeiadis [2011] style operational soundness argument of quantifying over all finite executions. Instead, we must use domain-theoretic techniques to construct the full set of distributions that can occur after an infinite amount of time. While there are several Iris-based separation logics for randomized programs that use an operational proof, all of those logics have predicates over single program states, which they then lift to a probabilistic interpretation using either couplings (for relational proofs) [Gregersen et al. 2024b; Tassarotti and Harper 2019] or using resources to describe *one* property of the randomized program (*e.g.*, error bounds or expected costs) [Aguirre et al. 2024; Haselwarter et al. 2024b]. In contrast, PCOL assertions allow for rich specifications over distributions of states.

3.1 Preliminaries: Memories and the Convex Powerset

Programs must be interpreted in a domain that supports both probabilistic and nondeterministic computation. Although none of our actions are explicitly nondeterministic, nondeterminism arises

due to the interleaving of concurrent threads. The difficulty is that typical representations of probabilistic computation (distributions) do not compose well with nondeterminism (powersets) [Varacca and Winskel 2006; Zwart and Marsden 2019]. We instead use the *convex powerset* C in our denotational semantics, which we describe in this section.

Memories, Expressions, and Tests. A memory $\sigma \in \mathsf{Mem}[S] \triangleq S \to \mathsf{Val}$ is a mapping from a finite set of variables $S \subseteq_{\mathsf{fin}} \mathsf{Var}$ to values, where Val consist of integers, rationals, and lists. The disjoint union $\uplus \colon \mathsf{Mem}[S] \to \mathsf{Mem}[T] \to \mathsf{Mem}[S \cup T]$ combines two memories as long as $S \cap T = \emptyset$, and a similar operation $A * B$ is defined on sets of memories $A \subseteq \mathsf{Mem}[S]$ and $B \subseteq \mathsf{Mem}[T]$.

$$(\sigma \uplus \tau)(x) \triangleq \begin{cases} \sigma(x) & \text{if } x \in S \\ \tau(x) & \text{if } x \in T \end{cases} \qquad A * B \triangleq \{\sigma \uplus \tau \mid \sigma \in A, \tau \in B\}$$

The notation $A * B$ is reminiscent of the *separating conjunction* [O'Hearn and Pym 1999], and indeed we will use it in Section 5 to define the separating conjunction. We define projections $\pi_S \colon \mathsf{Mem}[T] \to \mathsf{Mem}[S \cap T]$ as $\pi_S(\sigma)(x) \triangleq \sigma(x)$ if $x \in S$. Expressions are interpreted in the usual way with $[\![e]\!]_{\mathsf{Exp}} : \mathsf{Mem}[S] \to \mathsf{Val}$ as long as $\mathsf{vars}(e) \subseteq S$, if not then $[\![e]\!]_{\mathsf{Exp}}(\sigma)$ is undefined. The same is true for tests and $[\![b]\!]_{\mathsf{Test}} : \mathsf{Mem}[S] \to \mathbb{B}$ (where $\mathbb{B} = \{0, 1\}$) is defined if $\mathsf{vars}(b) \subseteq S$.

Discrete Probability Distributions. A discrete probability distribution $\mu \in \mathcal{D}(X)$ over a countable set X is a mapping from elements of X to $[0, 1]$ such that $\sum_{x \in X} \mu(x) = 1$. The support of a distribution is the set of elements to which it assigns nonzero probability $\mathsf{supp}(\mu) \triangleq \{x \in X \mid \mu(x) \neq 0\}$. The Dirac, or point-mass, distribution δ_x assigns probability 1 to x and 0 to everything else. The previously defined projections extend to distributions $\pi_S \colon \mathcal{D}(\mathsf{Mem}[T]) \to \mathcal{D}(\mathsf{Mem}[S \cap T])$ by marginalizing as follows $\pi_S(\mu)(\sigma) \triangleq \sum_{\tau \in \mathsf{Mem}[T \setminus S]} \mu(\sigma \uplus \tau)$. Distributions $\mu, \nu \in \mathcal{D}(X \cup \{\bot\})$ are ordered as follows: $\mu \sqsubseteq_{\mathcal{D}} \nu$ iff $\mu(x) \leq \nu(x)$ for all $x \in X$ and $\mu(\bot) \geq \nu(\bot)$. This makes $\langle \mathcal{D}(X \cup \{\bot\}), \sqsubseteq_{\mathcal{D}} \rangle$ a pointed poset with bottom $\bot_{\mathcal{D}} = \delta_\bot$.

The Convex Powerset. A convex powerset is a set of all the possible distributions of outcomes that could result from (nondeterministic) scheduling. For a more complete explanation, refer to He et al. [1997] and Zilberstein et al. [2025b]. Distributions can be added and scaled pointwise: $(\mu + \nu)(x) = \mu(x) + \nu(x)$ and $(p \cdot \mu)(x) = p \cdot \mu(x)$. The convex combination of two distributions is defined as $\mu \oplus_p \nu = p \cdot \mu + (1 - p) \cdot \nu$. A set of distributions $S \subseteq \mathcal{D}(X \cup \{\bot\})$ is *convex* if it is closed under convex combinations: $(\mu \oplus_p \nu) \in S$ for every $\mu, \nu \in S$ and $p \in [0, 1]$.

Additional requirements ensure that the domain is a DCPO, and therefore suitable for representing iterated computations and fixed points. A set S is *up-closed* if for all $\mu, \nu \in \mathcal{D}(X \cup \{\bot\})$, if $\mu \in S$ and $\mu \sqsubseteq_{\mathcal{D}} \nu$ then $\nu \in S$. Finally, S is *Cauchy closed* if it is closed in the product of Euclidean topologies [McIver and Morgan 2005], *i.e.,* it is a finite union of closed regions of $X \cup \{\bot\}$-dimensional Euclidean space. The convex powerset is now defined as follows:

$$C(X) \triangleq \{S \subseteq \mathcal{D}(X \cup \{\bot\}) \mid S \text{ is nonempty, up-closed, convex, and Cauchy closed}\}$$

We include $\bot$ to represent nontermination and undefined behavior such as accessing a variable that is not scope. Nonemptiness ensures that the semantics is not vacuous, since undefined behavior is represented by $\{\delta_\bot\}$ rather than $\emptyset$. Up-closure ensures that C is a partial order in the Smyth [1978] powerdomain, *i.e.,* $S \sqsubseteq_C T$ iff $\forall \nu \in T.\ \exists \mu \in S.\ \mu \sqsubseteq_{\mathcal{D}} \nu$. In fact, due to up-closure, $S \sqsubseteq_C T$ iff $S \supseteq T$, so suprema are given by set intersections. Cauchy closure ensures that the intersections of directed sets are nonempty, and therefore $\langle C(X), \sqsubseteq_C \rangle$ is a pointed DCPO with bottom $\bot_C = \mathcal{D}(X \cup \{\bot\})$.

Convexity ensures that C carries a *monad* structure [Jacobs 2008], making the sequencing of actions compositional. More precisely, there is a unit $\eta \colon X \to C(X)$ and Kleisli extension $(-)^\dagger \colon (X \to C(Y)) \to C(X) \to C(Y)$, which obey the monad laws: $\eta^\dagger = \mathrm{id}$, $f^\dagger \circ \eta = f$, and

$f^\dagger \circ g^\dagger = (f^\dagger \circ g)^\dagger$. These operations are defined as follows:

$$\eta(x) \triangleq \{\delta_x\} \qquad f^\dagger(S) \triangleq \left\{ \sum_{x \in \mathsf{supp}(\mu)} \mu(x) \cdot \nu_x \;\middle|\; \mu \in S, \forall x \in \mathsf{supp}(\mu).\ \nu_x \in f_\perp(x) \right\}$$

where $f_\perp(x) = f(x)$ for $x \in X$ and $f_\perp(\perp) = \perp_C$. As an overloading of notation, we will occasionally write $f^\dagger(\mu)$ to mean $f^\dagger(\{\mu\})$ for any $\mu \in \mathcal{D}(X)$. Convex combinations in C are defined as $S \oplus_p T \triangleq \{\mu \oplus_p \nu \mid \mu \in S, \nu \in T\}$, and convex union is $S \,\&\, T \triangleq \bigcup_{p \in [0,1]} S \oplus_p T$. For some finite index set $I = \{i_1, \ldots, i_n\}$, we let $\&_{i \in I} S_i \triangleq S_{i_1} \,\&\, \cdots \,\&\, S_{i_n}$. Finally, we extend projections to convex sets as $\pi_S(T) \triangleq \{\pi_S(\mu) \mid \mu \in T\}$ where we let $\sigma \uplus \perp = \perp$.

In the next section, we will see how '&' will be used to represent the choices of the *scheduler* when interleaving concurrent threads. The fact that $S \,\&\, T$ is represented as a set of convex combinations operationally corresponds to the idea that the scheduler can use randomness to choose between S and T, rather than making the choice deterministically [Varacca 2002, §6.5].

3.2 Actions and Invariants

We can now use the convex powerset to give semantics to actions. The basic action evaluation is defined below $[\![-]\!]_{\mathsf{Act}} : \mathsf{Act} \to \mathsf{Mem}[S] \to C(\mathsf{Mem}[S])$.

$$[\![x := e]\!]_{\mathsf{Act}}(\sigma) \triangleq \begin{cases} \eta(\sigma[x := [\![e]\!]_{\mathsf{Exp}}(\sigma)]) & \text{if } \mathsf{vars}(e) \cup \{x\} \subseteq S \\ \perp_C & \text{otherwise} \end{cases}$$

$$[\![x :\approx d(e)]\!]_{\mathsf{Act}}(\sigma) \triangleq \begin{cases} \left\{\sum_{v \in \mathsf{supp}(\mu)} \mu(v) \cdot \delta_{\sigma[x:=v]}\right\} & \text{if } \mathsf{vars}(e) \cup \{x\} \subseteq S, \mu = d([\![e]\!]_{\mathsf{Exp}}(\sigma)) \\ \perp_C & \text{otherwise} \end{cases}$$

As we alluded to in Section 2, we will reason about shared state via *invariants*—assertions about shared state that must be preserved by every atomic action. To model the ways in which shared state may be modified by other threads, we define an *invariant sensitive semantics*, in which the scheduler may alter shared state before executing each atomic action. This is based on *semantic invariants*, finite sets of memories $\mathcal{I} \subseteq_{\mathsf{fin}} \mathsf{Mem}[T]$ that represent the legal values of shared state.

We limit invariants to be finite sets in order to avoid issues arising from unbounded nondeterminism. Stemming from the impossibility result of Apt and Plotkin [1986], the semantics of loops cannot be constructed in standard ways using least fixed points in the presence of unbounded nondeterminism. In C specifically, unbounded nondeterminism breaks Cauchy closure [McIver and Morgan 2005, Appendix B.4.2]. Finite invariants are sufficient for a wide variety of verification tasks, such as the examples in Section 6. Looking forward, there are additional algorithms where shared state only takes on finitely many values such as the randomized Dining Philosophers [Lehmann and Rabin 1981] and other synchronization protocols [Rabin 1980, 1982].

The invariant-sensitive model is a family of semantic functions for actions, indexed by a semantic invariant: $[\![a]\!]^{\mathcal{I}}_{\mathsf{Act}} : \mathsf{Mem}[S] \to C(\mathsf{Mem}[S])$, where $\mathcal{I} \subseteq_{\mathsf{fin}} \mathsf{Mem}[T]$ and $T \subseteq S$.

$$[\![a]\!]^{\mathcal{I}}_{\mathsf{Act}} \triangleq (\mathsf{check}^{\mathcal{I}})^\dagger \circ [\![a]\!]^\dagger_{\mathsf{Act}} \circ (\mathsf{replace}^{\mathcal{I}})^\dagger \circ \mathsf{check}^{\mathcal{I}}$$

$$\mathsf{check}^{\mathcal{I}}(\sigma) \triangleq \begin{cases} \eta(\sigma) & \text{if } \pi_T(\sigma) \in \mathcal{I} \\ \perp_C & \text{if } \pi_T(\sigma) \notin \mathcal{I} \end{cases} \qquad \mathsf{replace}^{\mathcal{I}}(\sigma) = \&_{\tau \in \mathcal{I}}\, \eta(\pi_{S \setminus T}(\sigma) \uplus \tau)$$

In the invariant sensitive semantics, $\mathcal{I}$ is first checked to ensure that the current state satisfies the invariant. Next, a new valid state (or distribution thereof) is chosen to replace the current one, simulating a parallel thread which may alter the shared state at any step. The standard action semantics is then executed, followed by another check to ensure that the invariant still holds. If the invariant is ever violated, then $\perp_C$ is returned to indicate that the execution is faulty. Letting $\mathsf{emp} \in \mathsf{Mem}[\emptyset]$ be the empty memory, we remark that $[\![a]\!]_{\mathsf{Act}} = [\![a]\!]^{\{\mathsf{emp}\}}_{\mathsf{Act}}$, meaning that invariant

sensitive execution using the empty invariant is equal to normal execution. In Lemma C.2, we prove that the invariant sensitive semantics is monotonic—*i.e.,* $[\![a]\!]_{\mathrm{Act}}(\sigma) \subseteq [\![a]\!]^{\mathcal{I}}_{\mathrm{Act}}(\sigma)$—adding an invariant can only add behaviors, making it an over-approximation of the program's behavior. So, safety properties about $[\![a]\!]^{\mathcal{I}}_{\mathrm{Act}}(\sigma)$ immediately apply to $[\![a]\!]_{\mathrm{Act}}(\sigma)$ too.

3.3 Semantics of Randomized Concurrent Programs

To give semantics to commands, we use *Partially Ordered Multisets (Pomsets) with Formulae* [Zilberstein et al. 2025a], where a partial order represents the causality between actions in the program. We write $a_1 \to a_2$ to mean that the action a_1 must be scheduled before a_2. Pomsets with formulae are constructed using three composition operators, shown below, which mirror the program syntax.

$$[\![a_1 \fatsemi a_2]\!] = \begin{array}{c} a_2 \\ \uparrow \\ a_1 \end{array} \qquad [\![a_1 \parallel a_2]\!] = \begin{array}{ccc} a_1 & & a_2 \\ & \nwarrow \quad \nearrow & \\ & \bullet & \end{array} \qquad [\![\textbf{if } b \textbf{ then } a_1 \textbf{ else } a_2]\!] = \begin{array}{ccc} a_1 & & a_2 \\ {\scriptstyle \mathrm{T}}\nwarrow & & \nearrow{\scriptstyle \mathrm{F}} \\ & b & \end{array}$$

From left to right, the sequential composition $a_1 \fatsemi a_2$ results in a totally ordered structure, where a_1 must occur before a_2. In a parallel composition $a_1 \parallel a_2$, the actions a_1 and a_2 are not ordered with respect to each other, so they can be scheduled in any order. Finally, if statements result in a guarded branch, where the two successors of the test b both must be scheduled after b, but also will only be scheduled if the outcome of the test matches the label on the arrow.

For the purposes of our program logic, we are interested in the *linearized* version of the model, $\mathcal{L}([\![-]\!]) \colon \mathsf{Cmd} \to \mathsf{Mem}[S] \to \mathcal{C}(\mathsf{Mem}[S])$, which maps input memories to convex sets of output memories, representing the set of possible distributions that can arise due to different interleavings of the parallel threads chosen by the scheduler. Linearization is defined in terms of the semantics for actions and tests from Section 3.2. The semantics of actions $[\![-]\!]^{\mathcal{I}}_{\mathrm{Act}}$ is indexed by an invariant $\mathcal{I}$, and so linearization $\mathcal{L}^{\mathcal{I}}$ is also indexed by an invariant, indicating which semantic function to use for actions. We provide the definition of $\mathcal{L}$—due to Zilberstein et al. [2025a]—in Appendix A. We omit the indexing invariant $\mathcal{I}$ and just write $\mathcal{L}([\![-]\!])$ when $\mathcal{I} = \{\mathsf{emp}\}$.

When the invariant $\mathcal{I}$ is $\{\mathsf{emp}\}$ in $\mathcal{L}$, the scheduler does not simulate any mutations performed by other threads, since all actions are evaluated according to $[\![a]\!]^{\{\mathsf{emp}\}}_{\mathrm{Act}} = [\![a]\!]_{\mathrm{Act}}$, so $\mathcal{L}([\![C]\!])(\sigma)$ can be viewed as the *true* semantics of the program. Similar to action evaluation, linearization is also monotonic with respect to invariants (Lemma 5.3), meaning that $\mathcal{L}([\![C]\!])(\sigma) \subseteq \mathcal{L}^{\mathcal{I}}([\![C]\!])(\sigma)$. This guarantees that adding an invariant will only add new behaviors, so safety properties about $\mathcal{L}^{\mathcal{I}}([\![C]\!])(\sigma)$ will automatically carry over to $\mathcal{L}([\![C]\!])(\sigma)$.

The linearized *state transformer* is ideal for modeling a program logic, where programs are specified in terms of preconditions and postconditions. As shown by Zilberstein et al. [2025a, Lemma 5.2], linearization of non-parallel programming constructs is well-behaved; it is compositional with respect to sequencing, if statements, and while loops, as shown by the following equational rules:

$$\mathcal{L}^{\mathcal{I}}([\![\textbf{skip}]\!]) = \eta \qquad \mathcal{L}^{\mathcal{I}}([\![a]\!]) = [\![a]\!]^{\mathcal{I}}_{\mathrm{Act}} \qquad \mathcal{L}^{\mathcal{I}}([\![C_1 \fatsemi C_2]\!]) = \mathcal{L}^{\mathcal{I}}([\![C_2]\!])^{\dagger} \circ \mathcal{L}^{\mathcal{I}}([\![C_1]\!])$$

$$\mathcal{L}^{\mathcal{I}}([\![\textbf{if } b \textbf{ then } C_1 \textbf{ else } C_2]\!])(\sigma) = \begin{cases} \mathcal{L}^{\mathcal{I}}([\![C_1]\!])(\sigma) & \text{if } [\![b]\!]_{\mathrm{Test}}(\sigma) = \mathsf{true} \\ \mathcal{L}^{\mathcal{I}}([\![C_2]\!])(\sigma) & \text{if } [\![b]\!]_{\mathrm{Test}}(\sigma) = \mathsf{false} \end{cases}$$

$$\mathcal{L}^{\mathcal{I}}([\![\textbf{while } b \textbf{ do } C]\!]) = \mathsf{lfp}\left(\Psi_{\langle b,C,\mathcal{I}\rangle}\right) \qquad \Psi_{\langle b,C,\mathcal{I}\rangle}(f)(\tau) = \begin{cases} f^{\dagger}(\mathcal{L}^{\mathcal{I}}([\![C]\!])(\tau)) & \text{if } [\![b]\!]_{\mathrm{Test}}(\tau) = \mathsf{true} \\ \eta(\tau) & \text{if } [\![b]\!]_{\mathrm{Test}}(\tau) = \mathsf{false} \end{cases}$$

In fact, the parallel-free fragment of the linearized model is equivalent to the model of Demonic Outcome Logic [Zilberstein et al. 2025a, Theorem 5.3], and so some of the metatheory for standard commands carries over from DOL to PCOL. However, there is no straightforward compositional

property for parallel programs, since the input-output behavior of two threads can completely change if they are run in parallel. Building on the insights of Concurrent Separation Logic [O'Hearn 2004] and Probabilistic Separation Logics [Bao et al. 2025; Barthe et al. 2020; Li et al. 2023], we develop compositional reasoning techniques for parallel programs in Sections 4 and 5.

4 The Model of Probabilistic Assertions

Preconditions and postconditions in Probabilistic Concurrent Outcome Logic (pcOL) are inspired by both Demonic Outcome Logic [Zilberstein et al. 2025b] and also probabilistic separation logics [Bao et al. 2021, 2025, 2022; Barthe et al. 2020; Li et al. 2023]. We begin by giving the syntax and semantics for basic assertions about memories in Section 4.1. Next, we discuss background on measure theory and probability spaces in Section 4.2. Like Lilac [Li et al. 2023] and Bluebell [Bao et al. 2025], the model of resources uses probability spaces that only assign probabilities to certain *measurable* sets of memories. Based on this, we define probabilistic assertions in Section 4.3.

4.1 Pure Assertions

We begin by describing pure (non-probabilistic) assertions, which are inspired by standard separation logic [O'Hearn et al. 2001; Reynolds 2002], but where memories range over variables rather than heap cells, as we discussed in Section 3.2. The syntax for these assertions are shown below.

$$P ::= \mathsf{true} \mid \mathsf{false} \mid P \wedge Q \mid P \vee Q \mid P * Q \mid \exists X.\, P \mid e \mapsto E \mid E_1 \asymp E_2 \qquad (\asymp \in \{=, \leq, <, \in, \ldots\})$$
$$E ::= X \mid v \mid E_1 + E_2 \mid E_1 \cdot E_2 \mid \cdots$$

In addition to expressions and variables from Section 3, assertions also depend on logical variables $X, Y, Z \in \mathsf{LVar}$, which cannot be modified by programs. Logical expressions $E \in \mathsf{LExp}$ mirror standard ones, but operate over logical variables $X \in \mathsf{LVar}$ rather than $x \in \mathsf{Var}$. Logical expression evaluation under a context $\Gamma : \mathsf{LVar} \to \mathsf{Val}$ is written $[\![E]\!]_{\mathsf{LExp}}(\Gamma)$ and is defined in a standard way.

Pure assertions are modelled by both a context Γ, and a memory $\sigma \in \mathsf{Mem}[S]$, the satisfaction relation is shown in Figure 2. The meaning of true, false, conjunction, and disjunction are standard. The separating conjunction $P * Q$ means that the memory $\sigma \in \mathsf{Mem}[S]$ can be divided into two smaller memories $\sigma_1 \in \mathsf{Mem}[S_1]$ and $\sigma_2 \in \mathsf{Mem}[S_2]$ to satisfy P and Q individually. By the definition of $\uplus$, S_1 and S_2 must be disjoint. Our logic is an *intuitionistic* [Docherty 2019] or *affine* interpretation of separation logic, meaning that information about variables can be discarded; if $\Gamma, \sigma \vDash P$, then P need not describe the entire memory σ. As such, we only require that $\sigma_1 \uplus \sigma_2 \sqsubseteq \sigma$, which we define as $\sigma \sqsubseteq \tau$ iff $\sigma \uplus \sigma' = \tau$ for some σ', so τ could contain more variables than σ.

We also include a points-to predicate $e \mapsto E$, although it has a slightly different meaning than points-to predicates in the heap model. Here, e does not describe a pointer, but can rather be any concrete expression, and $e \mapsto E$ simply means that the e evaluates to the same value under σ as E does under Γ, allowing us to connect the concrete and logical state. Finally, $E_1 \asymp E_2$ allows us to make assertions about logical state, where $\asymp \in \{=, \leq, \in, \cdots\}$ ranges over similar comparators to the ones in Section 3. We define the following notation to obtain the set of all memories $\sigma \in \mathsf{Mem}[S]$ that satisfy an assertion P, we omit the superscript when we wish to minimize S, so that the memories contain only the free variables of P:

$$(\!|P|\!)^S_\Gamma \triangleq \{\sigma \in \mathsf{Mem}[S] \mid \Gamma, \sigma \vDash P\} \qquad (\!|P|\!)_\Gamma \triangleq (\!|P|\!)^{\mathsf{vars}(P)}_\Gamma$$

Finally, we provide syntactic sugar for restricting the domain of existential quantifiers, asserting membership in a set, and asserting that the resources of e are owned by the current thread.

$$\exists X \in E.\, P \triangleq \exists X.\, P * X \in E \qquad e \in E \triangleq \exists X \in E.\, e \mapsto X \qquad \mathsf{own}(e_1, \ldots, e_n) \triangleq \mathop{\Large *}_{i=1}^{n} \exists X_i.\, e_i \mapsto X_i$$

$$
\begin{array}{ll}
\Gamma, \sigma \vDash \mathsf{true} & \text{always} \\
\Gamma, \sigma \vDash \mathsf{false} & \text{never} \\
\Gamma, \sigma \vDash P \wedge Q & \text{iff} \quad \Gamma, \sigma \vDash P \quad \text{and} \quad \Gamma, \sigma \vDash Q \\
\Gamma, \sigma \vDash P \vee Q & \text{iff} \quad \Gamma, \sigma \vDash P \quad \text{or} \quad \Gamma, \sigma \vDash Q \\
\Gamma, \sigma \vDash P * Q & \text{iff} \quad \exists \sigma_1, \sigma_2. \quad \sigma_1 \uplus \sigma_2 \sqsubseteq \sigma \quad \text{and} \quad \Gamma, \sigma_1 \vDash P \quad \text{and} \quad \Gamma, \sigma_2 \vDash Q \\
\Gamma, \sigma \vDash \exists X.\, P & \text{iff} \quad \Gamma[X := v], \sigma \vDash P \text{ for some } v \in \mathsf{Val} \\
\Gamma, \sigma \vDash e \mapsto E & \text{iff} \quad [\![e]\!]_{\mathsf{Exp}}(\sigma) = [\![E]\!]_{\mathsf{LExp}}(\Gamma) \\
\Gamma, \sigma \vDash E_1 \asymp E_2 & \text{iff} \quad [\![E_1]\!](\Gamma) \asymp [\![E_2]\!](\Gamma)
\end{array}
$$

Fig. 2. Satisfaction relation for pure assertions.

4.2 Measure Theory and Probability Spaces

We now introduce basic definitions from measure theory needed to define probabilistic separation. For a more thorough background, refer to Royden [1968] or Fremlin [2001]. A probability space $\mathcal{P} = \langle \Omega, \mathcal{F}, \mu \rangle$ consists of a sample space Ω, an event space $\mathcal{F}$, and a probability measure μ. For our purposes, the sample space $\Omega \subseteq \mathsf{Mem}[S]$ will consist of memories over a particular set of variables S. The event space $\mathcal{F} \subseteq 2^{\Omega}$ gives the events—*i.e.,* sets of memories—which are measurable. It must be a σ-algebra, meaning that it contains $\emptyset$ and Ω, and it is closed under complementation and countable unions and intersections. The probability measure $\mu \colon \mathcal{F} \to [0, 1]$ assigns probabilities to the events in $\mathcal{F}$, and must obey $\mu(\emptyset) = 0$, $\mu(\Omega) = 1$, and countable additivity: $\mu(\biguplus_{i \in I} A_i) = \sum_{i \in I} \mu(A_i)$ where I is a countable index set and all the A_i sets are pairwise disjoint. For a probability space $\mathcal{P}$, we use $\Omega_{\mathcal{P}}$, $\mathcal{F}_{\mathcal{P}}$, and $\mu_{\mathcal{P}}$ to refer to its respective parts.

We require probability spaces to be *complete*, meaning that they contain all events of measure zero. More formally, $\mathcal{P}$ is complete if for any $A \in \mathcal{F}_{\mathcal{P}}$ such that $\mu_{\mathcal{P}}(A) = 0$, then $B \in \mathcal{F}_{\mathcal{P}}$ for all $B \subseteq A$ [Royden 1968]. We will often also require the sample space to be the full set of memories $\mathsf{Mem}[S]$ for some S. A probability space $\mathcal{P}$ with $\Omega_{\mathcal{P}} \subseteq \mathsf{Mem}[S]$ can be *extended* as follows: $\mathsf{ext}(\mathcal{P}) \triangleq \langle \mathsf{Mem}[S], \mathcal{F}, \mu \rangle$ where $\mathcal{F} \triangleq \{A \subseteq \mathsf{Mem}[S] \mid A \cap \Omega_{\mathcal{P}} \in \mathcal{F}_{\mathcal{P}}\}$ and $\mu(A) \triangleq \mu_{\mathcal{P}}(A \cap \Omega_{\mathcal{P}})$.

We now define a preorder on probability spaces. As is typical in intuitionistic logic, this preorder $\mathcal{P} \preceq \mathcal{Q}$ will indicate when $\mathcal{Q}$ contains more information than $\mathcal{P}$. The information can be gained across two dimensions: by expanding the memory footprint, or by making the event space more granular. Formally, for $\mathcal{P}$ and $\mathcal{Q}$ such that $\Omega_{\mathcal{P}} \subseteq \mathsf{Mem}[S]$, we define $\mathcal{P} \preceq \mathcal{Q}$ as follows:

$$
\begin{array}{lll}
\mathcal{P} \preceq \mathcal{Q} & \text{iff} & \Omega_{\mathcal{P}} \subseteq \pi_S(\Omega_{\mathcal{Q}}) \\
& \text{and} & \mathcal{F}_{\mathcal{P}} \subseteq \{\pi_S(A) \mid A \in \mathcal{F}_{\mathcal{Q}}\} \\
& \text{and} & \forall A \in \mathcal{F}_{\mathcal{P}}.\, \mu_{\mathcal{P}}(A) = \mu_{\mathcal{Q}}\left(\bigcup \{B \in \mathcal{F}_{\mathcal{Q}} \mid \pi_S(B) = A\}\right)
\end{array}
$$

So, $\mathcal{P} \preceq \mathcal{Q}$ iff $\mathcal{P}$ contains smaller sample and event spaces, but $\mathcal{P}$ and $\mathcal{Q}$ agree on the probability of events whose projections are measurable in $\mathcal{P}$. Any proper distribution $\mu \in \mathcal{D}(\mathsf{Mem}[U])$, can be used as a probability space where $\Omega_\mu = \mathsf{Mem}[U]$, $\mathcal{F}_\mu = 2^{\mathsf{Mem}[U]}$ is the greatest σ-algebra on $\mathsf{Mem}[U]$, and $\mu_\mu(A) = \sum_{\sigma \in A} \mu(\sigma)$. Projections of probability spaces are defined as $\pi_U(\mathcal{P}) = \langle \Omega, \mathcal{F}, \mu \rangle$, where $\Omega = \pi_U(\Omega_{\mathcal{P}})$, $\mathcal{F} = \{\pi_U(A) \mid A \in \mathcal{F}_{\mathcal{P}}\}$, and $\mu(A) = \mu_{\mathcal{P}}(A * \pi_{\mathsf{Var} \setminus U}(\Omega_{\mathcal{P}}))$.

We define two more operations on probability spaces, which will help us to give semantics to the separating conjunction and outcome conjunction in Section 4.3. The first operation is the product space $\mathcal{P} \otimes \mathcal{Q}$, which is defined when $\Omega_{\mathcal{P}} \subseteq \mathsf{Mem}[S]$, $\Omega_{\mathcal{Q}} \subseteq \mathsf{Mem}[T]$, and $S \cap T = \emptyset$. The sample space $\Omega_{\mathcal{P} \otimes \mathcal{Q}} = \Omega_{\mathcal{P}} * \Omega_{\mathcal{Q}}$ is the set of all joined memories in the two spaces, the event space $\mathcal{F}_{\mathcal{P} \otimes \mathcal{Q}}$ is the smallest σ-algebra containing $\{A * B \mid A \in \mathcal{F}_{\mathcal{P}}, B \in \mathcal{F}_{\mathcal{Q}}\}$, and the measure has the property that $\mu_{\mathcal{P} \otimes \mathcal{Q}}(A * B) = \mu_{\mathcal{P}}(A) \cdot \mu_{\mathcal{Q}}(B)$ for any $A \in \mathcal{F}_{\mathcal{P}}$ and $B \in \mathcal{F}_{\mathcal{Q}}$. The full construction uses Carathéodory's method, and is given in Chapter 25 of Fremlin [2001].

$$
\begin{array}{lll}
\Gamma, \mathcal{P} \vDash \top & & \text{always} \\
\Gamma, \mathcal{P} \vDash \bot & & \text{never} \\
\Gamma, \mathcal{P} \vDash \varphi \wedge \psi & \text{iff} & \Gamma, \mathcal{P} \vDash \varphi \quad \text{and} \quad \Gamma, \mathcal{P} \vDash \psi \\
\Gamma, \mathcal{P} \vDash \varphi \vee \psi & \text{iff} & \Gamma, \mathcal{P} \vDash \varphi \quad \text{or} \quad \Gamma, \mathcal{P} \vDash \psi \\
\Gamma, \mathcal{P} \vDash \exists X.\, \varphi & \text{iff} & \Gamma[X := v], \mathcal{P} \vDash \varphi \quad \text{for some} \quad v \in \mathsf{Val} \\
\Gamma, \mathcal{P} \vDash \bigoplus_{X \sim d(E)} \varphi & \text{iff} & \forall v.\, \Gamma[X := v], \mathsf{ext}(\mathcal{P}_v) \vDash \varphi \quad \text{and} \quad \bigoplus_{v \sim \mu} \mathcal{P}_v \preceq \mathcal{P} \quad \text{for some} \quad (\mathcal{P}_v)_{v \in \mathsf{supp}(\mu)} \\
 & & \quad \text{where} \quad \mu = d(\llbracket E \rrbracket_{\mathsf{LExp}}(\Gamma)) \\
\Gamma, \mathcal{P} \vDash \mathop{\&}_{X \in E} \varphi & \text{iff} & \Gamma, \mathcal{P} \vDash \bigoplus_{X \sim \mu} \varphi \quad \text{for some} \quad \mu \in \mathcal{D}(\llbracket E \rrbracket_{\mathsf{LExp}}(\Gamma)) \\
\Gamma, \mathcal{P} \vDash \varphi *_m \psi & \text{iff} & \mathcal{P}' \preceq \mathcal{P} \ \text{ and } \ \Gamma, \mathcal{P}_1 \vDash \varphi \ \text{ and } \ \Gamma, \mathcal{P}_2 \vDash \psi \ \text{ for some } \ \mathcal{P}_1, \mathcal{P}_2, \text{and } \mathcal{P}' \in \mathcal{P}_1 \diamond_m \mathcal{P}_2 \\
\Gamma, \mathcal{P} \vDash \lceil P \rceil & \text{iff} & (\!|P|\!)^S_\Gamma \in \mathcal{F}_{\mathcal{P}} \quad \text{and} \quad \mu_{\mathcal{P}}\left((\!|P|\!)^S_\Gamma\right) = 1
\end{array}
$$

Fig. 3. The satisfaction relation, where $\Gamma : \mathsf{LVar} \to \mathsf{Var}$ is a logical context and $\mathcal{P} = \langle \mathsf{Mem}[S], \mathcal{F}_{\mathcal{P}}, \mu_{\mathcal{P}} \rangle$ is a complete probability space. All the existentially quantified probability spaces are also complete.

Note that this definition is more similar to the initial formulation of PSL [Barthe et al. 2020] (albeit, in a probability space), rather than Lilac and Bluebell, which rely on a theorem stating that independent products are unique [Li et al. 2023, Lemma 2.3]. We use the explicit product construction in order to guarantee that each variable can only occur on one side of the $*$, making mutation rules simpler. Lilac does not allow mutable state, and so mutation is not a factor. On the other hand, Bluebell handles mutation by explicitly tracking permissions, but we found the product construction to be simpler to use than the permission approach.

The next operation is a *direct sum* for combining disjoint probability spaces [Fremlin 2001, 214L]. More precisely, for some countable index set I, discrete distribution $\nu \in \mathcal{D}(I)$, and probability spaces $\mathcal{P}_i = \langle \Omega_i, \mathcal{F}_i, \mu_i \rangle$ such that the Ω_i are pairwise disjoint, we define the direct sum as:

$$
\bigoplus_{i \sim \nu} \mathcal{P}_i \triangleq \langle \biguplus_{i \in I} \Omega_i, \mathcal{F}, \mu \rangle \qquad \mathcal{F} \triangleq \{ A \subseteq \Omega \mid \forall i \in I.\, A \cap \Omega_i \in \mathcal{F}_i \} \qquad \mu(A) \triangleq \sum_{i \in I} \nu(i) \cdot \mu_i(A \cap \Omega_i)
$$

The sample space is the union of all the individual sample spaces, the measurable events are those events whose projections into each Ω_i are measurable according to $\mathcal{F}_i$, and the probability measure is given by a convex sum. The direct sum will be used to give semantics to our outcome conjunction. Finally, we remark that independent products distribute over direct sums (Lemma B.3):

$$
\left(\bigoplus_{i \sim \nu} \mathcal{P}_i\right) \otimes \mathcal{Q} = \bigoplus_{i \sim \nu} (\mathcal{P}_i \otimes \mathcal{Q})
$$

4.3 Probabilistic Assertions

We now define probabilistic assertions, which will serve as pre- and postconditions in pcOL triples. The syntax is shown below and the semantics is in Figure 3.

$$
\varphi ::= \top \mid \bot \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \exists X.\, \varphi \mid \bigoplus_{X \sim d(E)} \varphi \mid \mathop{\&}_{X \in E} \varphi \mid \varphi *_m \psi \mid \lceil P \rceil \qquad (m \in \{\mathsf{s}, \mathsf{w}\})
$$

The semantics for probabilistic assertions is based on a context $\Gamma : \mathsf{LVar} \to \mathsf{Val}$, and a complete probability space $\mathcal{P} = \langle \mathsf{Mem}[S], \mathcal{F}_{\mathcal{P}}, \mu_{\mathcal{P}} \rangle$. The $\top$, $\bot$, conjunction, disjunction, and existential quantification assertions have the usual semantics.

Next, we have two kinds of *outcome conjunctions*, adapted from Demonic OL (dOL) [Zilberstein et al. 2025b], but with a new measure-theoretic semantics based on direct sums. The standard outcome conjunction $\bigoplus_{X \sim d(E)} \varphi$ allocates a new logical variable X, which is distributed according

to $\mu = d([\![E]\!]_{\mathsf{LExp}}(\Gamma))$, and can be referenced in φ. The probability space $\mathcal{P}$ must be a refinement of the direct sum of $(\mathcal{P}_v)_{v \in \mathsf{supp}(\mu)}$. For every v, we then also require that $\Gamma[X := v], \mathsf{ext}(\mathcal{P}_v) \vDash \varphi$, so φ holds in the sub-probability space $\mathcal{P}_v$ with the value of X in Γ updated accordingly. Essentially, the outcome conjunction splits the sample space $\mathsf{Mem}[S]$ according to the support of $d(E)$.

The nondeterministic outcome conjunction $\&_{X \in E}$ is similar, but here only the support of the distribution is specified (as E). This connective is used when disjunctions or existential quantification would be used in a purely nondeterministic logic. For example, as a result of running the concurrent program $x := 1 \parallel x := 2$, it is not correct to say that $\lceil x \mapsto 1 \rceil \vee \lceil x \mapsto 2 \rceil$ since the probabilistic scheduler could choose to make x equal to 1 with some probability $0 < p < 1$. On the other hand $\&_{X \in \{1,2\}} \lceil x \mapsto X \rceil$ means that x takes on value 1 with some (existentially quantified) probability, matching the convex powerset interpretation of nondeterminism.

Next, we have two variations of the separating conjunction, parameterized by a mode $m \in \{\mathsf{s}, \mathsf{w}\}$. Strong separation (mode s) is the interpretation of separation that requires *probabilistic independence* and separation of variables, whereas weak separation (mode w) does not require independence, and only separates the variables. As we explained in Section 2, weak separation is needed when case analysis over nondeterministic shared state leaves us in a scenario where strong frame preservation does not hold. Semantically, the difference is captured by the combinator operation $\diamond_m$, with strong separation using an independent product and weak separation simply requiring that the marginal probability spaces are correct.

$$\mathcal{P}_1 \diamond_{\mathsf{s}} \mathcal{P}_2 \triangleq \{\mathcal{P}_1 \otimes \mathcal{P}_2\} \qquad \mathcal{P}_1 \diamond_{\mathsf{w}} \mathcal{P}_2 \triangleq \{\mathcal{P} \mid \mathcal{P}_1 = \pi_U(\mathcal{P}), \mathcal{P}_2 = \pi_V(\mathcal{P})\}$$

where $\Omega_{\mathcal{P}_1} = \mathsf{Mem}[U]$ and $\Omega_{\mathcal{P}_2} = \mathsf{Mem}[V]$ and $U \cap V = \emptyset$. Clearly, $\mathcal{P}_1 \diamond_{\mathsf{s}} \mathcal{P}_2 \subseteq \mathcal{P}_1 \diamond_{\mathsf{w}} \mathcal{P}_2$, which immediately gives us that $\varphi *_{\mathsf{s}} \psi \Rightarrow \varphi *_{\mathsf{w}} \psi$. Strong separation will be used more commonly, so we will drop the subscript there and write $*$ to mean $*_{\mathsf{s}}$.

Finally, the almost sure assertion $\lceil P \rceil$ states that the pure assertion P, as described in Section 4.1, occurs with probability 1. We also define syntactic sugar below for a binary outcome conjunction $\oplus_E$, a bounded binary outcome conjunction $\oplus_{\geq E}$, and expressions distributed according to some distribution $e \sim d(E)$.

$$\varphi \oplus_E \psi \triangleq \bigoplus_{X \sim \mathbf{Ber}(E)} (\lceil X = 1 \rceil * \varphi) \vee (\lceil X = 0 \rceil * \psi) \qquad e \sim d(E) \triangleq \bigoplus_{X \sim d(E)} \lceil e \mapsto X \rceil$$

$$\varphi \oplus_{\geq E} \psi \triangleq \exists X.\, \lceil X \geq E \rceil * (\varphi \oplus_X \psi)$$

4.4 Convex and Precise Assertions and Entailment Laws

As we mentioned in Section 2, the parallel composition rule of pcOL requires the postcondition from each thread to be *precise*, so that any correlations introduced via concurrent scheduling are not measurable. We now define precision formally in terms of probability spaces.

Definition 4.1 (Precision). An assertion φ is *precise* if for any Γ under which φ is satisfiable there is a unique smallest probability space $\mathcal{P}$ such that $\Gamma, \mathcal{P} \vDash \varphi$ and if $\Gamma, \mathcal{P}' \vDash \varphi$, then $\mathcal{P} \preceq \mathcal{P}'$. We write $\mathsf{precise}(\varphi_1, \ldots, \varphi_n)$ to mean $\mathsf{precise}(\varphi_1) \wedge \cdots \wedge \mathsf{precise}(\varphi_n)$.

Below, we give a few rules to determine that assertions are precise.

$$\frac{}{\mathsf{precise}(\lceil P \rceil)} \qquad \frac{\mathsf{precise}(\varphi, \psi)}{\mathsf{precise}(\varphi * \psi)} \qquad \frac{\mathsf{precise}(\varphi) \qquad \varphi \Rightarrow \lceil e \mapsto X \rceil}{\mathsf{precise}(\bigoplus_{X \sim d(E)} \varphi)}$$

Almost sure assertions are always precise, since the smallest model is the one where $(\!|P|\!)_\Gamma$ occurs with probability 1, and is the smallest measurable set with nonzero probability. Separating conjunctions are precise if their subcomponents are, which follows from monotonicity of the independent product

$$\frac{P \vdash Q}{\lceil P \rceil \vdash \lceil Q \rceil} \qquad \frac{\varphi \vdash \varphi' \qquad \psi \vdash \psi'}{\varphi *_m \psi \vdash \varphi' *_m \psi'} \qquad \varphi * \psi \vdash \varphi *_{\mathsf{w}} \psi \qquad \lceil P * Q \rceil \dashv\vdash \lceil P \rceil *_m \lceil Q \rceil \qquad \varphi * \lceil P \rceil \dashv\vdash \varphi *_{\mathsf{w}} \lceil P \rceil$$

$$\bigoplus_{X \sim d(E)} \varphi \vdash \&_{X \in \mathrm{supp}(d(E))} \varphi \qquad \varphi[E/X] \dashv\vdash \&_{X \in \{E\}} \varphi \qquad \lceil E \subseteq E' \rceil * \&_{X \in E} \varphi \vdash \&_{X \in E'} \varphi$$

$$\frac{\varphi \vdash \psi}{\bigoplus_{X \sim d(E)} \varphi \vdash \bigoplus_{X \sim d(E)} \psi} \qquad \frac{Y \notin \mathsf{fv}(\varphi)}{\bigoplus_{X \sim d(E)} \varphi \vdash \bigoplus_{Y \sim d(E)} \varphi[Y/X]} \qquad \frac{X \notin \mathsf{fv}(\psi)}{(\bigoplus_{X \sim d(E)} \varphi) * \psi \vdash \bigoplus_{X \sim d(E)} (\varphi * \psi)}$$

$$\frac{X \notin \mathsf{fv}(\psi) \qquad \mathsf{precise}(\psi)}{\bigoplus_{X \sim d(E)} (\varphi * \psi) \vdash (\bigoplus_{X \sim d(E)} \varphi) * \psi} \qquad \frac{X \notin \mathsf{fv}(\psi) \qquad \mathsf{convex}(\psi)}{\bigoplus_{X \sim d(E)} (\varphi *_{\mathsf{w}} \psi) \vdash (\bigoplus_{X \sim d(E)} \varphi) *_{\mathsf{w}} \psi} \qquad \frac{X \notin \mathsf{fv}(\varphi) \quad \mathsf{convex}(\varphi)}{\bigoplus_{X \sim d(E)} \varphi \vdash \varphi}$$

Fig. 4. Selected entailment laws, where $\varphi[E/X]$ denotes a syntactic substitution of E for X in φ. Recall that $m \in \{\mathsf{s}, \mathsf{w}\}$ and when m is omitted, $* = *_{\mathsf{s}}$.

(Lemma B.4). Outcome conjunctions are precise if the inner assertion is precise, and implies that $\lceil e \mapsto X \rceil$ for some program expression e, which witnesses how to partition the sample space for the direct sum. Without this partitioning side condition, the result is not necessarily precise. For example, $\lceil x \mapsto 1 \rceil \oplus_{\frac{1}{2}} \lceil x \in \{0, 1\} \rceil$ is not precise, since it is not possible to determine the probability of the event $x = 1$, despite it being measurable in one of the sub-probability spaces. However, $\lceil x \mapsto 1 * y \mapsto 0 \rceil \oplus_{\frac{1}{2}} \lceil x \in \{0, 1\} * y \mapsto 1 \rceil$ is precise, since y witnesses the partition.

Weak separating conjunctions $*_{\mathsf{w}}$ and nondeterministic outcome conjunctions $\&$ are *not* precise, as there are generally many different minimal probability spaces that satisfy them, assigning different probabilities to each event. However, those assertions do obey *convexity*, a weaker condition, which intuitively means that $\varphi \oplus_p \varphi \Rightarrow \varphi$. We give the formal definition below.

Definition 4.2 (Convex Assertions). $\mathsf{convex}(\varphi)$ iff for all Γ under which φ is satisfiable, there exist Ω, $\mathcal{F}$, and a convex set S of probability measures on $\mathcal{F}$ such that:

$$\forall \mathcal{P}. \quad \Gamma, \mathcal{P} \vDash \varphi \qquad \text{iff} \qquad \exists \mu \in S. \quad \langle \Omega, \mathcal{F}, \mu \rangle \preceq \mathcal{P}$$

Clearly any precise assertion is convex, since the set of measures S is just a singleton in that case. So, in addition to analogues of the precision rules above, we also have the following:

$$\frac{\mathsf{convex}(\varphi, \psi)}{\mathsf{convex}(\varphi *_{\mathsf{w}} \psi)} \qquad \frac{\mathsf{convex}(\varphi, \psi) \qquad \begin{smallmatrix} \varphi \Rightarrow \lceil e \mapsto 1 \rceil \\ \psi \Rightarrow \lceil e \mapsto 0 \rceil \end{smallmatrix}}{\mathsf{convex}(\varphi \oplus_{\geq p} \psi)} \qquad \frac{\mathsf{convex}(\varphi) \qquad \varphi \Rightarrow \lceil e \mapsto X \rceil}{\mathsf{convex}(\&_{X \in E} \varphi)}$$

Precision and convexity are useful for formulating entailment laws, which we provide in Figure 4. The first row consists of rules for the interaction between separating conjunctions and pure assertions. Weakening can be performed underneath both pure assertions and separating conjunctions. In addition, strong separation implies weak separation, and weak and strong separation are equivalent when one of the conjuncts is a pure assertion (since independence is trivial in that case).

In the second row, we give some rules pertaining to $\&$. An outcome conjunction $\bigoplus_{X \sim d(E)}$ can be weakened to a $\&$ over $\mathrm{supp}(d(E))$, where $\mathrm{supp}(\mathbf{Ber}\,(E)) \triangleq \{0, 1\}$ and $\mathrm{supp}(\mathbf{unif}\,(E)) \triangleq E$. A $\&_{X \in \{E\}} \varphi$ over a singleton set is the same as substituting E for X in φ. Finally, the bounds of a $\&$ can always be expanded, similar to how $P \Rightarrow P \vee Q$ in classical logic.

The final two rows pertain to outcome conjunctions, and each rule has a corresponding one (not shown) with $\&$ instead of $\bigoplus$. Weakening can be performed inside of an outcome conjunction, and bound variables can be α-renamed as long as the new variable name is fresh. As in Bluebell, the (strong) separating conjunction distributes over the outcome conjunction, so that assertions can be

moved inside of an outcome conjunction, but this rule is invalid if $*$ is replaced by $*_{\mathsf{w}}$, due to the possibility of correlations between ψ and $d(E)$.

Factoring assertions out of an outcome conjunction is only supported in Bluebell for almost-sure assertions $\lceil P \rceil$, whereas in PCOL, it can be performed for any *precise* assertion, due to our semantics based on a direct sum. We saw at the end of Section 4.2 that independent products (which model separating conjunctions) distribute over direct sums (which model outcome conjunctions), however the corresponding entailment $\bigoplus_{X \sim d(E)} (\varphi * \psi) \vdash (\bigoplus_{X \sim d(E)} \varphi) * \psi$ requires that ψ is satisfied by the same model in each case, which can be guaranteed by forcing ψ to be precise. In fact, this exact scenario also arose in the RCOND and RCASE rules of PSL, where an analogous concept called *supported* was used to ensure soundness [Barthe et al. 2020]. Replacing strong separation with weak separation, we only need ψ to be convex—not precise—to factor it out of the outcome conjunction, since independence is not implied. Finally, outcome conjunctions over convex assertions that do not depend on the bound variable X can be collapsed.

5 Probabilistic Concurrent Outcome Logic

Probabilistic Concurrent Outcome Logic (PCOL) specifications are given as triples of the form $I \vDash_m \langle \varphi \rangle\ C\ \langle \psi \rangle$, where φ and ψ are probabilistic assertions (Section 4.3), $C \in \mathsf{Cmd}$ (Figure 1), I is a basic assertion, and $m \in \{\mathsf{s}, \mathsf{w}\}$. Roughly speaking, the meaning of these triples is that if the states are initially distributed according to φ, then any invariant sensitive execution of C with invariant I will satisfy ψ. The mode m dictates what kind of *frame preservation* property the triple has.

Recall from Section 3.3 that invariant sensitive execution requires the invariant states to be drawn from a finite set. For this reason, I must be a *finitary* basic assertion; formally, $\mathsf{finitary}(I)$ iff $(\!|I|\!)_\Gamma$ is a finite set for any context Γ. The formal validity definition of PCOL triples is below.

Definition 5.1 (PCOL Triples). The PCOL triple $I \vDash_m \langle \varphi \rangle\ C\ \langle \psi \rangle$ is valid iff for all $\Gamma : \mathsf{LVar} \to \mathsf{Val}$, μ, and probability spaces $\mathcal{P}$, $\mathcal{P}_F$, and $\mathcal{P}' \in \mathcal{P} \diamond_m \mathcal{P}_F$ such that $\mathcal{P}' \preceq \mu$ and $\Gamma, \mathcal{P} \vDash \varphi * \lceil I \rceil$, then:

$$\forall \nu \in \mathcal{L}^{(\!|I|\!)_\Gamma} (\llbracket C \rrbracket)^\dagger (\mu). \quad \exists Q.\ \exists Q' \in Q \diamond_m \mathcal{P}_F. \quad Q' \preceq \nu \quad \text{and} \quad \Gamma, Q \vDash \psi * \lceil I \rceil$$

As in many separation logics, frame preservation is built into the semantics of the triples [Birkedal and Yang 2007; Jung et al. 2018]; in addition to quantifying over a probability space $\mathcal{P}$ to satisfy φ, we also quantify over a probability space $\mathcal{P}_F$, which describes unused resources and is preserved by the program execution. As with the separating conjunction, we will omit the m when $m = \mathsf{s}$. In addition, since we defined probability spaces to operate over memories $\mathsf{Mem}[S]$, without $\bot$, our triples are *fault avoiding*, which is also a standard choice for separation logics [Yang and O'Hearn 2002]. That is, if $I \vDash \langle \varphi \rangle\ C\ \langle \psi \rangle$ is valid, then we know that C will not encounter a memory fault starting from a distribution satisfying φ.

These triples also imply almost sure termination, or *total correctness*. We chose to pursue total correctness, as it aligns with probabilistic liveness properties that we are interested in (*e.g.,* see Section 6.4). In the probabilistic context, there is no single natural notion of partial correctness, and nontermination breaks parallel composition; composing a thread with a nonterminating thread alters the behavior of the first thread even without shared state.

In the remainder of this section, we will present inference rules for deriving PCOL triples. We write $I \vdash_m \langle \varphi \rangle\ C\ \langle \psi \rangle$ to mean that a triple is derivable using these rules. All of the rules are sound with respect to Definition 5.1.

THEOREM 5.2 (SOUNDNESS). *For all of the rules in Figures 5 to 8, if* $I \vdash_m \langle \varphi \rangle\ C\ \langle \psi \rangle$ *then* $I \vDash_m \langle \varphi \rangle\ C\ \langle \psi \rangle$.

$$\frac{}{I \vdash_m \langle \varphi \rangle\ \mathbf{skip}\ \langle \varphi \rangle}\textsc{Skip} \qquad \frac{I \vdash_m \langle \varphi \rangle\ C_1\ \langle \vartheta \rangle \quad I \vdash_m \langle \vartheta \rangle\ C_2\ \langle \psi \rangle}{I \vdash_m \langle \varphi \rangle\ C_1 \fatsemi C_2\ \langle \psi \rangle}\textsc{Seq}$$

$$\frac{\varphi \Rightarrow \lceil b \mapsto \mathsf{true} \rceil \qquad I \vdash_m \langle \varphi \rangle\ C_1\ \langle \psi \rangle}{I \vdash_m \langle \varphi \rangle\ \mathbf{if}\ b\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2\ \langle \psi \rangle}\textsc{IfT} \qquad \frac{\varphi \Rightarrow \lceil b \mapsto \mathsf{false} \rceil \qquad I \vdash_m \langle \varphi \rangle\ C_2\ \langle \psi \rangle}{I \vdash_m \langle \varphi \rangle\ \mathbf{if}\ b\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2\ \langle \psi \rangle}\textsc{IfF}$$

$$\frac{\varphi \Rightarrow \lceil e \mapsto E \rceil \wedge (\psi * \lceil \mathsf{own}(x) \rceil)}{I \vdash_m \langle \varphi \rangle\ x := e\ \langle \psi * \lceil x \mapsto E \rceil \rangle}\textsc{Assign} \qquad \frac{\varphi \Rightarrow \lceil e \mapsto E \rceil \wedge (\psi * \lceil \mathsf{own}(x) \rceil)}{I \vdash_m \langle \varphi \rangle\ x :\approx d(e)\ \langle \psi * (x \sim d(E) \rangle}\textsc{Samp}$$

Fig. 5. Rules for Sequential Commands

5.1 Rules for Sequential Commands

The rules for sequential commands are given in Figure 5. Although the rules appear like the standard ones for Hoare-like logics [Hoare 1969] and separation logic [O'Hearn et al. 2001; Reynolds 2002], they rely on the properties of linearization shown in Section 3.3. In Skip, the precondition is preserved by a no-op, and Seq is the standard rule for sequential composition.

The rules for if statements are split into two cases, for when the precondition implies that the true or false branch will be taken, respectively, similar to standard Outcome Logic [Zilberstein 2025; Zilberstein et al. 2025b]. These rules can be combined into a single rule for analyzing both branches using the various split rules, which we will introduce in Section 5.3.

Finally, we give rules for atomic actions. Assign requires the precondition to determine that the program expression e evaluates to the logical expression E, and that the variable x, which is being assigned, is owned by the current thread and is disjoint from the assertion ψ. This structure gives the flexibility to apply the rule both when $\varphi = \psi * \lceil x \mapsto E' \rceil$ and $\varphi = \psi * \lceil \mathsf{own}(x) \rceil$. Samp has a similar requirement, but ultimately concludes that x is distributed according to $d(E)$ rather than having a deterministic value.

5.2 Concurrent Separation Logic Rules

Next, in Figure 6, we have a variety of rules inspired by Concurrent Separation Logic (CSL) [Brookes 2004; O'Hearn 2004; Vafeiadis 2011]. First is the Par rule for parallel composition. Although Par looks like the analogous rule from CSL—aside from the condition about precision—the soundness of the rule is substantially more complicated due to the probabilistic interpretation of the separating conjunction. It is not hard to imagine situations where the scheduler can introduce correlation between variables. For example, in the following program (which we previously saw in Section 2.3), the scheduler could choose to schedule the $y := 1$ action *after* the sampling operation is resolved, meaning that it could make $x = y$ with probability 1, a clear correlation.

$$x :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right) \fatsemi y := 0 \quad \Big\| \quad y := 1$$

As such, the outcomes of the two threads will not be independent after being run concurrently, but rather only *observably* independent in some restricted event space. By requiring the postconditions of each thread to be precise, we know that the probability of each measurable event must be specified exactly, so that the nondeterministic behavior of the scheduler will not be measurable (Lemmas C.5 and C.6). In the case of the program above, the strongest precise assertion about y is $\lceil y \in \{0,1\} \rceil$, that y is always either 0 or 1, and $\left(x \sim \mathbf{Ber}\left(\tfrac{1}{2}\right)\right) * \lceil y \in \{0,1\} \rceil$ is a valid postcondition for the program, since almost sure assertions $\lceil P \rceil$ are trivially independent from all other assertions.

More formally, the soundness proof uses the fact that ψ_1 and ψ_2 are precise to obtain unique minimal probability spaces Q_1 and Q_2 satisfying them. We then show that for any distribution ν resulting from running $C_1 \parallel C_2$, and for any events $B_1 \in \mathcal{F}_{Q_1}$ and $B_2 \in \mathcal{F}_{Q_2}$, it must be the case that

$$\frac{I \vdash \langle \varphi_1 \rangle\ C_1\ \langle \psi_1 \rangle \qquad I \vdash \langle \varphi_2 \rangle\ C_2\ \langle \psi_2 \rangle \qquad \mathsf{precise}(\psi_1, \psi_2)}{I \vdash \langle \varphi_1 * \varphi_2 \rangle\ C_1 \parallel C_2\ \langle \psi_1 * \psi_2 \rangle}\textsc{Par}$$

$$\frac{J \vdash_m \langle \varphi * \lceil I \rceil \rangle\ a\ \langle \psi * \lceil I \rceil \rangle}{I * J \vdash_m \langle \varphi \rangle\ a\ \langle \psi \rangle}\textsc{Atom} \qquad \frac{I * J \vdash_m \langle \varphi \rangle\ C\ \langle \psi \rangle \quad \mathsf{finitary}(I)}{J \vdash_m \langle \varphi * \lceil I \rceil \rangle\ C\ \langle \psi * \lceil I \rceil \rangle}\textsc{Share}$$

$$\frac{I \vdash_m \langle \varphi \rangle\ C\ \langle \psi \rangle}{I \vdash_m \langle \varphi *_m \vartheta \rangle\ C\ \langle \psi *_m \vartheta \rangle}\textsc{Frame} \qquad \frac{I \vdash \langle \varphi \rangle\ C\ \langle \psi \rangle}{I \vdash_{\mathsf{w}} \langle \varphi \rangle\ C\ \langle \psi \rangle}\textsc{Weaken} \qquad \frac{I \vdash_{\mathsf{w}} \langle \varphi \rangle\ C\ \langle \psi \rangle \quad \mathsf{precise}(\psi)}{I \vdash \langle \varphi \rangle\ C\ \langle \psi \rangle}\textsc{Strengthen}$$

Fig. 6. Concurrent Separation Logic Rules

$\nu(B_1 * B_2) = \mu_{Q_1}(B_1) \cdot \mu_{Q_2}(B_2)$. Since C_1 and C_2 may not terminate in a bounded amount of time, this probability only converges to the desired product in the limit.

The next two rules are for interacting with invariants. The Atom rule opens the invariant by moving it into the triple as an almost sure assertion, as long as the program is a single atomic action a. The fact that that the program executes atomically, and that I is true before and after execution, means that I is true at every step. Next, the Share rule allows a finitary almost sure assertion I to be moved into the invariant. The soundness of this rule relies on the invariant monotonicity property that we discussed in Section 3.3.

Lemma 5.3 (Invariant Monotonicity). *For any* $U, V, W \subseteq \mathsf{Var}$ *and* $\sigma \in \mathsf{Mem}[W]$ *such that* $U \cap V = \emptyset$, $\mathcal{I} \subseteq \mathsf{Mem}[U]$, $\mathcal{J} \subseteq \mathsf{Mem}[V]$, *and* $U \cup V \subseteq W$*:*

$$\mathcal{L}^{\mathcal{I} * \mathcal{J}}(\boldsymbol{\alpha})(\sigma) \sqsubseteq_C \mathcal{L}^{\mathcal{I}}(\boldsymbol{\alpha})(\sigma)$$

Recall that $\sqsubseteq_C$ is equivalent to $\supseteq$, so invariant monotonicity states that expanding the invariant (via $*$) can only add new behaviors to the set of outcomes. Lemma 5.3 follows from the more general monotonicity property of linearization [Zilberstein et al. 2025a].

The Frame rule allows a local specification to be lifted into a larger memory footprint [Yang and O'Hearn 2002]. The type of separation used depends on the mode m of the triple. If $m = \mathsf{s}$, then the frame ϑ not only represents a disjoint set of physical *resources*, but also that those resources are distributed independently from the information about the present program. A strong triple can always be weakened to a weak triple using the Weaken rule. A weak triple can be strengthened—via Strengthen—as long as the postcondition is precise. Just as with the Par rule, precision here ensures that the scheduler cannot force any correlation between the postcondition and the frame.

5.3 Structural and Outcome Splitting Rules

Additional structural rules are given in Figure 7. The first four splitting rules enable pointwise reasoning over outcome conjunctions, similar to those of Demonic Outcome Logic [Zilberstein et al. 2025b]. All these rules require that the logical variable X, bound by the outcome conjunction, does not appear free in the invariant I, since X is unbound in the premise of the rule. If X is free in I, then the rule can be applied after α-renaming X in the outcome conjunction (see Figure 4).

The first rule, Split1, requires that ψ dictates the partition of the probability spaces in order to construct a final direct sum. This is done in a similar fashion to rules for establishing precision that we saw in Section 4.3—by requiring that $\psi \Rightarrow \lceil e \mapsto X \rceil$ for some expression e. Since X takes on distinct values in each case of the direct sum, then $e \mapsto X$ witnesses that the sample space can be partitioned. If ψ does not witness a partition, then the Split2 rule can instead be used, which requires ψ to be convex and not dependent on X.

We now demonstrate these two modes of use. Below, x is initially distributed according to some distribution $d(E)$, and then it is incremented. The sample space is still partitioned after the

$$\frac{I \vdash_m \langle \varphi \rangle\ C\ \langle \psi \rangle \quad \psi \Rightarrow \lceil e \mapsto X \rceil \quad X \notin \mathsf{fv}(I)}{I \vdash_m \langle \bigoplus_{X \sim d(E)} \varphi \rangle\ C\ \langle \bigoplus_{X \sim d(E)} \psi \rangle}\ \textsc{Split1} \qquad \frac{I \vdash_m \langle \varphi \rangle\ C\ \langle \psi \rangle \quad \psi \Rightarrow \lceil e \mapsto X \rceil \quad X \notin \mathsf{fv}(I)}{I \vdash_m \langle \mathop{\&}_{X \in E} \varphi \rangle\ C\ \langle \mathop{\&}_{X \in E} \psi \rangle}\ \textsc{NSplit1}$$

$$\frac{I \vdash_m \langle \varphi \rangle\ C\ \langle \psi \rangle \quad \mathsf{convex}(\psi) \quad X \notin \mathsf{fv}(I, \psi)}{I \vdash_m \langle \bigoplus_{X \sim d(E)} \varphi \rangle\ C\ \langle \psi \rangle}\ \textsc{Split2} \qquad \frac{I \vdash_m \langle \varphi \rangle\ C\ \langle \psi \rangle \quad \mathsf{convex}(\psi) \quad X \notin \mathsf{fv}(I, \psi)}{I \vdash_m \langle \mathop{\&}_{X \in E} \varphi \rangle\ C\ \langle \psi \rangle}\ \textsc{NSplit2}$$

$$\frac{I \vdash_{\mathsf{w}} \langle \&_{X \in E} \lceil P \rceil \rangle\ C\ \langle \psi \rangle \quad \lceil P \rceil \Rightarrow \lceil e \mapsto X \rceil}{I \vdash_{\mathsf{w}} \langle \lceil \exists X \in E.\ P \rceil \rangle\ C\ \langle \psi \rangle}\ \textsc{Exists} \qquad \frac{\varphi' \Rightarrow \varphi \quad I \vdash_m \langle \varphi \rangle\ C\ \langle \psi \rangle \quad \psi \Rightarrow \psi'}{I \vdash_m \langle \varphi' \rangle\ C\ \langle \psi' \rangle}\ \textsc{Consequence}$$

Fig. 7. Structural and Outcome Splitting Rules

increment, which is witnessed by $\lceil x - 1 \mapsto X \rceil$, so Split1 can be used.

$$\frac{\vdash \langle \lceil x \mapsto X \rceil \rangle\ x := x + 1\ \langle \lceil x \mapsto X + 1 \rceil \rangle \qquad \lceil x \mapsto X + 1 \rceil \Rightarrow \lceil x - 1 \mapsto X \rceil}{\vdash \langle \bigoplus_{X \sim d(E)} \lceil x \mapsto X \rceil \rangle\ x := x + 1\ \langle \bigoplus_{X \sim d(E)} \lceil x \mapsto X + 1 \rceil \rangle}\ \textsc{Split1}$$

On the other hand, the next program samples y, and the resulting sample space is not partitioned according to X. Instead, Split2 is used since the postcondition is convex.

$$\frac{\vdash \langle \lceil x \mapsto X * X \in \{0,1\} \rceil \rangle\ y :\approx \mathbf{Ber}\,(x/2)\ \langle \exists Y.\ \lceil Y \leq 1/2 \rceil * y \sim \mathbf{Ber}\,(Y) \rangle}{\vdash \langle \bigoplus_{X \sim \mathbf{Ber}(1/2)} \lceil x \mapsto X * X \in \{0,1\} \rceil \rangle\ y :\approx \mathbf{Ber}\,(x/2)\ \langle \exists Y.\ \lceil Y \leq 1/2 \rceil * y \sim \mathbf{Ber}\,(Y) \rangle}\ \textsc{Split2}$$

NSplit1 and NSplit2 are the nondeterministic analogues of the two aforementioned rules. They operate in exactly the same way when the distribution over X is not known.

The Exists rule allows for a more complex form of case analysis, where the logical variable being scrutinized, X, is bound by an existential quantifier inside of a pure assertion. Since the precondition is a pure assertion, it may be satisfied by a probability space that cannot measure the probability that X takes on each value of E. Nonetheless, Exists allows us to use a stronger precondition where X is instead bound by a $\&$, allowing us to do case analysis using NSplit1 or NSplit2. The tradeoff is that this form of reasoning is incompatible with strong frame preservation because turning a pure assertion into a nondeterministic outcome conjunction has a bad interaction with the strong separating conjunction, as shown below.

$$\lceil y \in \{0,1\} \rceil * (x \sim \mathbf{Ber}\,(p)) \not\Rightarrow \Big(\mathop{\&}_{Y \in \{0,1\}} \lceil y \mapsto Y \rceil \Big) * (x \sim \mathbf{Ber}\,(p)) \Rightarrow \mathop{\&}_{Y \in \{0,1\}} \Big(\lceil y \mapsto Y \rceil * (x \sim \mathbf{Ber}\,(p)) \Big)$$

On the left hand side, it is possible that x and y are always equal. However, the first implication—which is unsound—implies that information about x's distribution can be distributed into every outcome of y, *i.e.*, x is distributed according to $\mathbf{Ber}\,(p)$ *for every* value of y. So, Exists is compatible only with weak frame preservation, but it nevertheless provides an important capability to do case analysis over shared state, which we will see more concretely in Sections 5.4, 6.1 and 6.4. As long as the postcondition becomes precise at some later point, the Strengthen rule can be used to regain strong frame preservation. Finally, the rule of Consequence allows pre- and postconditions to be manipulated in the standard way. The invariant can be neither strengthened nor weakened, since doing so would break the assumptions of other threads.

$$\frac{\begin{array}{c}\varphi \Rightarrow \lceil \ell \le R \le h \rceil \\ \varphi * \lceil R = \ell \rceil \Rightarrow \lceil b \mapsto \mathsf{false} \rceil \\ \varphi * \lceil R > \ell \rceil \Rightarrow \lceil b \mapsto \mathsf{true} \rceil \end{array} \quad I \vdash_m \langle \varphi * \lceil R = N > \ell \rceil \rangle\ C\ \langle (\&_{R=\ell}^{N-1}\ \varphi) \oplus_{\ge p} (\&_{R=N}^{h}\ \varphi) \rangle \quad \begin{array}{c} 0 < p \le 1 \\ N \notin \mathrm{vars}(\varphi) \\ \mathrm{precise}(\varphi[\ell/R]) \end{array}}{I \vdash_m \langle \&_{R=\ell}^{h}\ \varphi \rangle\ \mathbf{while}\ b\ \mathbf{do}\ C\ \langle \varphi[\ell/R] \rangle}\ \textsc{BoundedRank}$$

Fig. 8. The BoundedRank rule for almost sure termination.

5.4 Loops and Almost Sure Termination

The final proof rule is for analyzing while loops, and proving that they *almost surely* terminate—that is, they terminate with probability 1. The BoundedRank rule, shown in Figure 8, is based on rules for sequential programs due to McIver and Morgan [2005] and Zilberstein et al. [2025b]. It revolves around a loop invariant φ and a rank. The rank R must be integer-valued and bounded between ℓ and h, *i.e.*, $\varphi \Rightarrow \lceil \ell \le R \le h \rceil$. As long as $R > \ell$, the loop continues to iterate ($\varphi * \lceil R > \ell \rceil \Rightarrow \lceil b \mapsto \mathsf{true} \rceil$) and once R reaches ℓ, the loop must terminate ($\varphi * \lceil R = \ell \rceil \Rightarrow \lceil b \mapsto \mathsf{false} \rceil$).

The premise of BoundedRank guarantees that the rank strictly decreases each iteration with probability at least $p > 0$. Since the rank is bounded between ℓ and h, this means that from any start state, the loop is guaranteed to terminate with probability at least $p^{h-\ell} > 0$, which allows us to conclude that the program must almost surely terminate by the Zero-One law of McIver and Morgan [2005, Lemma 2.6.1]. Finally, since the terminating outcome $\varphi[\ell/R]$ is precise, there must be a unique minimal probability space Q that satisfies it. Similar to the soundness proof of the Par rule, we complete the soundness proof of the BoundedRank rule by showing that for all $B \in \mathcal{F}_Q$ and all ν resulting from finite approximations of the loop, $\nu(B)$ converges to $\mu_Q(B)$.

As an example of how to apply this rule, consider the example program below. The program implements a sort of random walk where x moves towards the origin with probability $\frac{1}{2}$, otherwise it is updated to y, which may be altered by a parallel thread.

$$\mathbf{RandWalk} \quad = \quad \mathbf{while}\ x > 0\ \mathbf{do}\ \underbrace{b :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right) \fatsemi \mathbf{if}\ b\ \mathbf{then}\ x := x - 1\ \mathbf{else}\ x := y}_{C_{\mathrm{body}}}$$

As long as the value of y is bounded, this loop almost surely terminates, which we can prove using BoundedRank, subject to the resource invariant $I = y \in \{0, \ldots, 5\}$. We will sketch the proof of almost sure termination here, the full derivation is shown in Appendix F.2. The loop invariant is $\varphi \triangleq \lceil x \mapsto R * 0 \le R \le 5 * \mathsf{own}(b) \rceil$, essentially just stating that x is between 0 and 5. The rank R is the value of x, and so R decreases each iteration with probability at least $\frac{1}{2}$. However, x may also be updated to y, which is nondeterministic, requiring a use of the Exists rule to conclude that $\&_{R=1}^{5} \lceil x \mapsto R \rceil$ after the command $x := y$. Clearly $\&_{R=1}^{5} \lceil x \mapsto R \rceil$ is not precise, so the resulting triple is not strongly frame preserving. Ultimately, we get the following weak triple for the loop body, which states that the rank strictly decreases with probability at least $\frac{1}{2}$.

$$y \in \{0, \ldots, 5\} \vdash_{\mathsf{w}} \langle \lceil x \mapsto N \rceil \rangle\ C_{\mathrm{body}}\ \langle (\&_{R=0}^{N-1} \lceil x \mapsto R \rceil) \oplus_{\ge \frac{1}{2}} (\&_{R=N}^{5} \lceil x \mapsto R \rceil) \rangle \tag{4}$$

At this stage, the weak triple signifies that we do not know exactly how likely the program is to terminate; we can only *bound* the probability. Indeed, the scheduler can influence the likelihood that x takes on particular values, and can force x to be correlated with other state. For example, suppose we wanted to apply the Frame rule with $z \sim \mathbf{unif}\ (\{0, \ldots, 5\})$. The scheduler could choose to always make y and z equal, in which case $z \sim \mathbf{unif}\ (\{0, \ldots, 5\})$ would not be independent of the postcondition of (4). However, the point of BoundedRank is that the scheduler *cannot* affect the probability of eventual termination. As such, the postcondition at the end of the execution is $\varphi[0/R]$, which is equivalent to the precise assertion $\lceil x \mapsto 0 \rceil$, allowing us to apply Strengthen to

get the following strong triple for the entire program.

$$y \in \{0,\dots,5\} \vdash \langle\lceil x \in \{0,\dots,5\}\rceil\rangle\ \texttt{RandWalk}\ \langle\lceil x \mapsto 0\rceil\rangle$$

One remarkable aspect of PCOL is that the triple above also implies that **RandWalk** almost surely terminates when run in parallel with any other almost surely terminating program that obeys the resource invariant I. We get this property for free from the PAR rule, which guarantees that the probability of any infinite interleaving of both programs converges to 0.

This holds without any fairness assumption; our semantics guarantees that each thread almost surely terminates regardless of interference from any other threads, therefore each enabled action is almost surely scheduled within a finite amount of time. Of course, there are many programs that only almost surely terminate subject to a fair scheduler, including probabilistic consensus and synchronization protocols [Ben-Or 1983; Lehmann and Rabin 1981; Rabin 1980]. As we discuss in Section 8, we plan to augment PCOL with capabilities to reason about fair termination in the future.

6 Examples

In this section, we present four examples to demonstrate how the proof rules of PCOL come together into more complex derivations. Proofs are sketched here, and shown fully in Appendix F.

6.1 Entropy Mixer

There are many scenarios where several potential sources of entropy or randomness are available, which must be mixed together with the guarantee that if at least one of the sources of entropy is high quality, then the output will be at least that good. A simplified example of a such scenario is modeled in the following program, where x_2 is a reliable source of entropy, but x_1 is unreliable, because it is derived from y in a way that can be controlled adversarially by the scheduler. Despite that, z, which is derived from x_1 and x_2 is a high quality source of randomness.

$$\texttt{EntropyMixer} \triangleq y \coloneqq 0 \fatsemi \Big(x_1 \coloneqq y \fatsemi x_2 :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right) \fatsemi z \coloneqq \mathbf{xor}(x_1, x_2) \ \Big\|\ y \coloneqq 1 \Big)$$

We will analyze this program using the invariant $I = (y \in \{0,1\})$, and conclude in the end that $z \sim \mathbf{Ber}\left(\frac{1}{2}\right)$. It is easy to see that the second thread satisfies the invariant, so we will focus on the first thread. We first show how information about y can be extracted from the invariant in order to give a specification for the assignment to x_1. The use of the EXISTS rule results in a weak triple.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash_w \langle\lceil y \mapsto Y * \mathsf{own}(x_1)\rceil\rangle\ x_1 \coloneqq y\ \langle\lceil x_1 \mapsto Y * y \mapsto Y\rceil\rangle}\ \textsc{Assign}}{\vdash_w \langle \&_{Y\in\{0,1\}}\lceil y \mapsto Y * \mathsf{own}(x_1)\rceil\rangle\ x_1 \coloneqq y\ \langle \&_{Y\in\{0,1\}}\lceil x_1 \mapsto Y * y \mapsto Y\rceil\rangle}\ \textsc{NSplit1}}{\vdash_w \langle \&_{Y\in\{0,1\}}\lceil y \mapsto Y * \mathsf{own}(x_1)\rceil\rangle\ x_1 \coloneqq y\ \langle (\&_{Y\in\{0,1\}}\lceil x_1 \mapsto Y\rceil) * \lceil y \in \{0,1\}\rceil\rangle}\ \textsc{Consequence}}{\vdash_w \langle \lceil \mathsf{own}(x_1)\rceil * \lceil y \in \{0,1\}\rceil\rangle\ x_1 \coloneqq y\ \langle (\&_{Y\in\{0,1\}}\lceil x_1 \mapsto Y\rceil) * \lceil y \in \{0,1\}\rceil\rangle}\ \textsc{Exists}}{y \in \{0,1\} \vdash_w \langle \lceil \mathsf{own}(x_1)\rceil\rangle\ x_1 \coloneqq y\ \langle \&_{Y\in\{0,1\}}\lceil x_1 \mapsto Y\rceil\rangle}\ \textsc{Atom}$$

These derivations are best read moving up from the lowermost precondition and then down from the topmost postcondition. First, ATOM is applied to open the invariant. Next, we use EXISTS and NSPLIT1 to gain access to the value of y, so that we can apply the ASSIGN rule. After the assignment, we use CONSEQUENCE to weaken the information about y and move it outside the scope of the $\&$ so that we can close the invariant. Now, we move on to the derivation for the remainder of the thread.

$$\dfrac{\dfrac{\dfrac{\vdots}{I \vdash_w \langle\lceil x_1 \mapsto Y * \mathsf{own}(x_2, z)\rceil\rangle\ x_2 :\approx \mathbf{Ber}\ (1/2) \fatsemi z \coloneqq \mathbf{xor}(x_1, x_2)\ \langle \bigoplus_{X\sim\mathbf{Ber}(1/2)}\lceil z \mapsto \mathbf{xor}(Y, X)\rceil\rangle}\ \textsc{Consequence}}{I \vdash_w \langle\lceil x_1 \mapsto Y * \mathsf{own}(x_2, z)\rceil\rangle\ x_2 :\approx \mathbf{Ber}\ (1/2) \fatsemi z \coloneqq \mathbf{xor}(x_1, x_2)\ \langle z \sim \mathbf{Ber}\ (1/2)\rangle}\ \textsc{NSplit2}}{I \vdash_w \langle \&_{Y\in\{0,1\}}\lceil x_1 \mapsto Y * \mathsf{own}(x_2, z)\rceil\rangle\ x_2 :\approx \mathbf{Ber}\ (1/2) \fatsemi z \coloneqq \mathbf{xor}(x_1, x_2)\ \langle z \sim \mathbf{Ber}\ (1/2)\rangle}$$

ConcurrentShuffle :
$a_1 \coloneqq [] \fatsemi a_2 \coloneqq [] \fatsemi i \coloneqq 0 \fatsemi$
while $i < \mathbf{len}(a)$ **do** (
$b :\approx \mathbf{Ber}\left(\frac{1}{2}\right) \fatsemi$
if b **then** $a_1 \coloneqq a_1 +\!\!+ [a[i]]$ **else** $a_2 \coloneqq a_2 +\!\!+ [a[i]] \fatsemi$
$i \coloneqq i + 1$
) $\fatsemi$
$\mathbf{shuffle}_1 \parallel \mathbf{shuffle}_2 \fatsemi$
$a \coloneqq a_1 +\!\!+ a_2$

$\mathbf{shuffle}_k$:
$i_k \coloneqq \mathbf{len}(a_k) \fatsemi$
while $i_k > 1$ **do**
$j_k :\approx \mathbf{unif}\,([0, \ldots, i_k]) \fatsemi$
$a_k \coloneqq \mathbf{swap}(a_k, i_k, j_k) \fatsemi$
$i_k \coloneqq i_k - 1$

Fig. 9. A concurrent shuffling algorithm, where $\ell_1 +\!\!+ \ell_2$ concatenates two lists and $\mathbf{swap}(\ell, i, j)$ returns ℓ with the elements $\ell[i]$ and $\ell[j]$ swapped.

First, we must again do case analysis on the logical variable Y, but this time we use NSplit2, which requires the postcondition to be convex and not depend on Y. In the premise of NSplit2, we we show that z is distributed properly regardless of the value of Y. For any fixed Y, after executing the writes to x_2 and z, we know that $\bigoplus_{X \sim \mathbf{Ber}(1/2)} \lceil z \mapsto \mathbf{xor}(Y, X) \rceil$ (the proof is quite mechanical, and shown in Appendix F.3). Since Y is constant, then $\mathbf{xor}(Y, X)$ is a bijection from $\{0, 1\}$ to $\{0, 1\}$, and we can therefore use the rule of Consequence to conclude that z is uniformly distributed. Since that postcondition is precise, we can strengthen the triple. After combining the two threads with Par, we get the following specification for the whole program.

$$\vdash \langle \lceil \mathsf{own}(x_1, x_2, y, z) \rceil \rangle\ \mathbf{EntropyMixer}\ \langle z \sim \mathbf{Ber}\left(\tfrac{1}{2}\right) \rangle$$

6.2 Concurrent Shuffle

Bacher et al. [2015] showed that shuffling algorithms can be made up to seven times faster through parallelization. They introduced a divide-and-conquer algorithm in which sub-arrays are shuffled concurrently and then merged. In this example, we prove the correctness of a simple concurrent shuffle algorithm using pcOL. The program is shown in Figure 9. First, the elements in the array are randomly assigned to two buckets, a_1 and a_2, the buckets are then shuffled in parallel using the standard Fisher and Yates [1938] algorithm, and then the two shuffled sub-lists are concatenated together. For some list ℓ, let $\Pi(\ell)$ be the set of all permutations of ℓ. For the purposes of this example, we will presume that all lists do not contain duplicate values. The specification of the Fisher-Yates shuffle is shown below (and proven in Appendix F.4). That is, if a list A is stored in a_k, then a_k will hold a uniformly chosen permutation of that list after execution of $\mathbf{shuffle}_k$.

$$\vdash \langle \lceil a_k \mapsto A * \mathsf{own}(i_k, j_k) \rceil \rangle\ \mathbf{shuffle}_k\ \langle a_k \sim \mathbf{unif}\,(\Pi(A)) \rangle$$

For some list ℓ and bit-string x, let $\ell[x]$ be the list obtained by filtering ℓ to only contain the indices i such that $x[i] = 1$, *e.g.*, $[1, 2, 3, 4][1001] = [1, 4]$. After the execution of the bucketing loop, we get the postcondition $\bigoplus_{X \sim \mathbf{unif}\left(\{0,1\}^{\mathbf{len}(A)}\right)} \lceil a_1 \mapsto A[X] * a_2 \mapsto A[\neg X] \rceil$, where X is a uniformly chosen bit-string that dictates into which bucket each element of A is placed, and $\neg X$ is bitwise logical negation. Next, to analyze the concurrent calls to $\mathbf{shuffle}_k$, we first use Split1 so that we can separate a_1 and a_2. Using Par with the triple for $\mathbf{shuffle}_k$, we have that a_1 and a_2 are uniformly and independently distributed permutations of the initial lists after the concurrent shuffles.

$$\frac{\vdash \langle \lceil a_1 \mapsto A[X] \rceil \rangle\ \mathbf{shuffle}_1\ \langle a_1 \sim \mathbf{unif}\,(\Pi(A[X])) \rangle \quad \vdash \langle \lceil a_2 \mapsto A[\neg X] \rceil \rangle\ \mathbf{shuffle}_2\ \langle a_2 \sim \mathbf{unif}\,(\Pi(A[\neg X])) \rangle}{\vdash \langle \lceil a_1 \mapsto A[X] * a_2 \mapsto A[\neg X] \rceil \rangle\ \mathbf{shuffle}_1 \parallel \mathbf{shuffle}_2\ \langle (a_1 \sim \mathbf{unif}\,(\Pi(A[X]))) * (a_2 \sim \mathbf{unif}\,(\Pi(A[\neg X]))) \rangle}\ \textsc{Par}$$

Reapplying the outcome conjunction over X, we can rewrite the postcondition to be:

$$\bigoplus_{X \sim \mathbf{unif}\left(\{0,1\}^{\mathbf{len}(A)}\right)} \bigoplus_{A_1 \sim \mathbf{unif}(\Pi(A[X]))} \bigoplus_{A_2 \sim \mathbf{unif}(\Pi(A[\neg X]))} \lceil a_1 \mapsto A_1 * a_2 \mapsto A_2 \rceil$$

$$\begin{array}{ll}
\textbf{PrivFetch}: & \textbf{fetch}_k: \\
\quad q_1 :\approx \textbf{unif}\ (\{0,1\}^n)\ ; & \quad i_k := 0\ ;\ r_k := 0\ ; \\
\quad q_2 := \textbf{xor}(q_1, x)\ ; & \quad \textbf{while}\ i_k < \textbf{len}(q_k)\ \textbf{do} \\
\quad \textbf{fetch}_1 \parallel \textbf{fetch}_2\ ; & \quad\quad \textbf{if}\ q_k[i_k] = 1\ \textbf{then} \\
\quad r := \textbf{xor}(r_1, r_2) & \quad\quad\quad r_k := \textbf{xor}(r_k, d[i_k])\ ; \\
 & \quad\quad i_k := i_k + 1
\end{array}$$

Fig. 10. A concurrent private information retrieval protocol.

Using Split1 three times and Assign, we get the following postcondition after the assignment to a.

$$\bigoplus_{X\sim\textbf{unif}(\{0,1\}^{\textbf{len}(A)})} \bigoplus_{A_1\sim\textbf{unif}(\Pi(A[X]))} \bigoplus_{A_2\sim\textbf{unif}(\Pi(A[\neg X]))} \lceil a \mapsto A_1 +\!\!+ A_2 \rceil$$

We now prove the assertion above implies that a is a uniform permutation. Take any $\ell \in \Pi(A)$ and let $n = \textbf{len}(A)$. For every $0 \le k \le n$, there is exactly one split X such that X assigns the elements $\ell[0], \ldots, \ell[k-1]$ to a_1 and the rest to a_2. This split occurs with probability $\frac{1}{2^n}$. Further, given that this correct split has occurred, a_1 and a_2 are shuffled so that $a_1 +\!\!+ a_2 = \ell$ with probability $\frac{1}{k!}\frac{1}{(n-k)!}$. Thus the probability of getting the permutation ℓ is $\sum_{k=0}^{n} \frac{1}{2^n}\frac{1}{k!}\frac{1}{(n-k)!} = \frac{1}{n!}$. Since there are exactly $|\Pi(A)| = n!$ permutations, this means that these permutations are produced uniformly, therefore:

$$\vdash \langle \lceil a \mapsto A * \mathsf{own}(\cdots) \rceil \rangle\ \textbf{ConcurrentShuffle}\ \langle a \sim \textbf{unif}\ (\Pi(A)) \rangle$$

6.3 Private Information Retrieval

Private information retrieval allows a user to fetch data without the database operator learning what data was requested [Chor et al. 1998]. A simple form of private information retrieval is modeled in the program shown in Figure 10. The $\textbf{fetch}_k$ programs process a bit string query q_k, indicating which entries of the database d to return. Those entries are then bitwise xor'ed together. Private retrieval is implemented in **PrivFetch**. The input x is a *one-hot* bit string $\textbf{onehot}(K)$, with a 1 in position K—indicating the index of the data to retrieve—and zeros everywhere else. Two queries are then made concurrently. The first one uses a randomly chosen bit string, and the second uses the same random string xor'ed with $\textbf{onehot}(K)$. The final data is an xor of the two responses, which reveals the data at position K. Barthe et al. [2020] proved a similar example in PSL, but their version was sequential; both fetches happened within a single for loop. Our version better models a distributed system where the computation does not occur in lockstep. We first present a specification for $\textbf{fetch}_k$ (proven in Appendix F.5), which states that r is an xor of data entries i, such that $q_k[i] = 1$, subject to the invariant that $d \mapsto D$.

$$d \mapsto D \vdash \langle \lceil q_k \mapsto Q \rceil * \lceil \mathsf{own}(i_k, r_k) \rceil \rangle\ \textbf{fetch}_k\ \langle \lceil r_k \mapsto \mathop{\textbf{xor}}_{0\le i<n : Q[i]=1} D[i] \rceil \rangle$$

We now sketch the derivation of the main procedure, the full proof is shown in Appendix F.5. After sampling into q_1, we use Split2 to make the outcome of that query deterministic. After assigning q_2, we get $\lceil q_1 \mapsto Q \rceil * \lceil q_2 \mapsto \textbf{xor}(Q, \textbf{onehot}(K)) \rceil$. We can then apply the Par rule to analyze the concurrent fetches to get the following postcondition:

$$\lceil r_1 \mapsto \mathop{\textbf{xor}}_{0\le i<n : Q[i]=1} D[i] \rceil * \lceil r_2 \mapsto \mathop{\textbf{xor}}_{0\le i<n : \textbf{xor}(Q,\textbf{onehot}(K))[i]=1} D[i] \rceil$$

The value of r is the xor of r_1 and r_2, which differ only at index K, therefore we can conclude that $\lceil r \mapsto D[K] \rceil$. Since this assertion is convex and does not depend on Q, we meet the side conditions of the Cond2 rule, and therefore the final postcondition is simply $\lceil r \mapsto D[K] \rceil$.

$$\vdash \langle \lceil x \mapsto \textbf{onehot}(K) * d \mapsto D * \mathsf{own}(q_1, q_1, r_1, r_2, r, i_1, i_2) \rceil \rangle\ \textbf{PrivFetch}\ \langle \lceil r \mapsto D[K] \rceil \rangle$$

6.4 The von Neumann Trick

The von Neumann [1951] trick is a protocol for simulating a fair coin given a coin with unknown bias. The biased coin is flipped twice, if the outcome is 1-0 or 0-1, then the output is the first coin flip, otherwise the experiment is repeated. Below, we have a variant of the von Neumann trick where the coin's bias is stored in shared memory, and its value is not remembered across rounds.

$$\textbf{vonNeumann} \triangleq x := 0 \fatsemi y := 0 \fatsemi \textbf{while}\ x = y\ \textbf{do}\ \ p' := p \fatsemi x :\approx \mathbf{Ber}\,(p') \fatsemi y :\approx \mathbf{Ber}\,(p')$$

Every round, a concurrent thread could change the bias, altering the probability of terminating in that round. Despite this, we show that the program almost surely terminates, and that x is distributed according to a fair coin, subject to the invariant $I = (p \in [\varepsilon, 1-\varepsilon]_\Delta)$ where $0 < \varepsilon \leq \frac{1}{2}$ is an arbitrarily small probability and Δ is a nonzero step size, making the interval finite (if p could be 0 or 1, then the program may not terminate). A variant of this example appeared in Zilberstein et al. [2025b], where the bias was explicitly altered by an adversary. This concurrent version of the program introduces new challenges from a program analysis perspective. The bulk of the derivation involves analyzing the while loop. To do so, we need a loop invariant, which is shown below.

$$\varphi \triangleq \varphi_0 \vee \varphi_1 \quad \varphi_0 \triangleq \bigoplus_{X \sim \mathbf{Ber}(1/2)} \lceil x \mapsto X * y \mapsto \neg X * R = 0 \rceil \quad \varphi_1 \triangleq \lceil x = y \mapsto \mathsf{true} * R = 1 * \mathsf{own}(p') \rceil$$

The rank R is either 0 or 1. When $R = 0$, $x \neq y$ and x is uniformly distributed, so the loop terminates. When $R = 1$, $x = y$, so the loop keeps iterating. Each iteration, the loop terminates with probability at least $2\varepsilon(1-\varepsilon)$. Due to the reliance on shared state, this probability is not exact, but only a bound. The structure of the proof is similar to that of Section 6.1. First, we use Atom and NSplit1 to open the invariant and conclude that p' holds some probability in $[\varepsilon, 1-\varepsilon]_\Delta$. Since the postcondition is imprecise and based on shared state, we are only able to obtain a weak triple at this point.

$$I \vdash_{\mathsf{w}} \langle \lceil \mathsf{own}(p') \rceil \rangle\ p' := p\ \langle\ \&_{X \in [\varepsilon, 1-\varepsilon]_\Delta} \lceil p' \mapsto X \rceil \rangle$$

We then sequence this with the two sampling operations, shown below:

$$\dfrac{\dfrac{\dfrac{\vdots}{I \vdash_{\mathsf{w}} \langle \lceil p' \mapsto X * \mathsf{own}(x,y) \rceil \rangle\ x :\approx \mathbf{Ber}\,(p') \fatsemi y :\approx \mathbf{Ber}\,(p')\ \langle \lceil p' \mapsto X \rceil * x \sim \mathbf{Ber}\,(X) * y \sim \mathbf{Ber}\,(X) \rangle}}{I \vdash_{\mathsf{w}} \langle \lceil p' \mapsto X * \mathsf{own}(x,y) \rceil \rangle\ x :\approx \mathbf{Ber}\,(p') \fatsemi y :\approx \mathbf{Ber}\,(p')\ \langle \varphi_0 \oplus_{\geq 2\varepsilon(1-\varepsilon)} \varphi_1 \rangle}\ \textsc{Consequence}}{I \vdash_{\mathsf{w}} \langle\ \&_{X \in [\varepsilon, 1-\varepsilon]_\Delta} \lceil p' \mapsto X * \mathsf{own}(x,y) \rceil \rangle\ x :\approx \mathbf{Ber}\,(p') \fatsemi y :\approx \mathbf{Ber}\,(p')\ \langle \varphi_0 \oplus_{\geq 2\varepsilon(1-\varepsilon)} \varphi_1 \rangle}\ \textsc{NSplit2}$$

We first use NSplit2 to show that for any fixed probability X, the loop terminates with probability at least $2\varepsilon(1-\varepsilon)$. This is done using straightforward combinatorial reasoning. Given that $p' \mapsto X$, we know that x and y will be independently and identically distributed according to $\mathbf{Ber}\,(X)$. That means that $\lceil x \mapsto 1 * y \mapsto 0 \rceil$ and $\lceil x \mapsto 0 * y \mapsto 1 \rceil$ both occur with probability $X(1-X)$, so φ_0 occurs with probability $2X(1-X)$, and otherwise $x = y$, so φ_1 holds. Since we know that $X \in [\varepsilon, 1-\varepsilon]$, then clearly $2\varepsilon(1-\varepsilon) \leq 2X(1-X)$, so the consequence above is valid. After applying BoundedRank, we get the postcondition $\varphi_0 \Rightarrow (x \sim \mathbf{Ber}\left(\frac{1}{2}\right))$. Since this postcondition is precise, we can use Strengthen to get the following strong triple for the whole program, indicating both that the program almost surely terminates, and that x is distributed like a fair coin. In fact, it also terminates when composed in parallel with other terminating threads that alter p in arbitrary ways!

$$p \in [\varepsilon, 1-\varepsilon]_\Delta \vdash \langle \lceil \mathsf{own}(x, y, p') \rceil \rangle\ \textbf{vonNeumann}\ \langle x \sim \mathbf{Ber}\left(\tfrac{1}{2}\right) \rangle$$

7 Related Work

Logics for Probabilistic Concurrency. Polaris is a relational separation logic built on Iris [Jung et al. 2018] for reasoning about concurrent probabilistic programs [Tassarotti 2018; Tassarotti and

Harper 2019]. Compared to Polaris, pcOL has two main advantages: it supports unbounded looping and it supports direct probabilistic reasoning about the distribution of outcomes. Polaris cannot handle the von Neumann trick, which involves unbounded looping. Because it is a relational logic, Polaris works by relating a randomized concurrent program to a specification program, which is randomized and nondeterministic (but not concurrent). This requires the random choices between the two programs to be coupled in lockstep.

In our concurrent shuffling example, we prove with pcOL that a sequence of many random choices results in a completely different uniform distribution of outcomes. It would be impossible to recreate this proof in Polaris, since the natural specification program for a randomized shuffle would only make a single random choice (a uniform sample over permutations), whereas the actual program makes a large sequence of random choices. It might be possible to do an alternate relational proof in Polaris by instead picking a specification program that makes a similar sequence of random choices, but then one would be left with the challenge of proving that such a specification program actually generated a uniform shuffle, for which Polaris cannot help.

In pcOL, we combine the two steps, avoiding the need to write down a specification program by building quantitative reasoning tools into the logic itself. However, the approaches do have similarities—our invariant sensitive semantics implicitly captures the behavior of the specification program by converting parallel manipulation of shared state into nondeterminism. This is slightly easier, since the user must only write down an invariant rather than an entire specification program. Then, rather than externally analyzing the specification program, we directly prove a quantitative postcondition. While pcOL does not support all the capabilities of Polaris (*e.g.,* ghost state, higher order state, etc.), proofs are carried out in fewer steps using a single, self-contained logic. In the future, it may be fruitful to combine the two approaches. Indeed, Bao et al. [2025] showed the advantages of supporting both relational and unary reasoning in a single probabilistic logic.

Fesefeldt et al. [2022] pursued an alternative technique for reasoning about probabilistic concurrent programs, based on a *quantitative* interpretation of separation logic [Batz et al. 2019]. Their logic can be used to lower bound the probability of a single outcome, making all but the private information retrieval example in this paper out of reach. While Fesefeldt et al. do support unbounded looping, the probability of nontermination is always added to the final expected values, and so it cannot be used to prove almost sure termination.

Fan et al. [2025] developed another logic based on Probabilistic Rely-Guarantee [McIver et al. 2016]. While it supports outcome splitting, the program must be re-instrumented to explicitly declare where splitting occurs, whereas in pcOL splitting is purely logical and can be used anywhere. Fan et al. require postconditions after splitting to be convex; they have a rule similar to Split2, but not Split1. In addition, their parallel composition rules do not give independence guarantees, and accordingly can only make conclusions about the local distributions of all threads, and not the *joint* distributions of all the variables in global memory, and programs must be proven to almost surely terminate externally to the logic. Their logic is based on an oblivious adversary model—a weaker model than our unrestricted adversary. More programs are verifiable in this model, so it is often preferred, however the logic of Fan et al. can only reason about obliviousness in limited ways, *i.e.,* by treating coin flip actions as atomic, so many programs are still out of reach.

Probabilistic Separation Logics. Capturing probabilistic independence in separation logic was first explored by Barthe et al. [2020], however the resulting Probabilistic Separation Logic (PSL) was limited in its ability to reason about control flow, and the frame rule had stringent side conditions. DIBI later extended the PSL model to include conditioning [Bao et al. 2021]. Lilac built on the two aforementioned logics and used conditioning to improve on PSL's handling of control flow,

although without mutable state [Li et al. 2023]. Lilac also added support for continuous distributions, and reformulated the notion of separation using probability spaces, making it more expressive.

Lilac's lack of mutable state means that information about variables can be duplicated; for example, $x \sim \mathbf{Ber}\left(\frac{1}{2}\right) * \lceil x \in \{0,1\} \rceil$ is satisfiable. Bluebell uses a similar model to Lilac, with the ability to reason about mutable state, which requires the logic to track permissions too [Bao et al. 2025]. We chose to use a more restrictive form of separation where information about variables cannot be shared, as it simplified the logical rules by eliminating the need for tracking permissions.

As we mentioned in Section 2, the direct sum semantics of $\bigoplus$ differs from conditioning modalities $\mathsf{C}_{X \sim d} \varphi$, which are based on disintegration. However, we believe that in the future the two modalities can work together to do discrete case analysis over continuous programs. For example, consider the following program, which samples from a continuous distribution and then branches on the result.

$$x :\approx \mathbf{unif}\,([0,1])\,;\, \mathbf{if}\; x \leq \tfrac{1}{2}\; \mathbf{then}\; C_1\; \mathbf{else}\; C_2$$

Lilac's rule for analyzing if statements is incompatible with mutable state, so we would instead need to use Split1 or Split2 to do case analysis on the guard $x \leq \frac{1}{2}$ in order to derive a specification for this program. However, those rules require an outcome conjunction, which cannot express the fact that x is distributed according to a continuous distribution. Instead, we can partition the continuous distribution into two parts, joined with an outcome conjunction, which would then allow us to analyze the two branches of the if statement.

$$\mathsf{C}_{X \sim \mathbf{unif}([0,1])} \lceil x \mapsto X \rceil \quad \Longrightarrow \quad \left(\mathsf{C}_{X \sim \mathbf{unif}([0,1/2])} \lceil x \mapsto X \rceil\right) \oplus_{\frac{1}{2}} \left(\mathsf{C}_{X \sim \mathbf{unif}((1/2,1])} \lceil x \mapsto X \rceil\right)$$

Although the direct sum's partitioning of the sample space imposes some limitations on the splitting rules, the benefit is that they are fully compositional; the use of splitting does not impede the application of other rules later in the derivation. This is in contrast with Bluebell's c-wp-swap [Bao et al. 2025, §5.1], which requires ownership over all program variables (denoted $\mathsf{own}_{\mathbb{X}}$), thereby precluding most later applications of the frame rule. Although Bao et al. [2025] show fruitful uses of c-wp-swap, the restriction is not acceptable for a concurrency logic, since it would preclude use of the Par rule.

This is similar to Exists disabling strong frame preservation, but Exists is used in specific scenarios for case analysis on nondeterministic shared state, which only arises in the concurrency setting. In any case, strong frame preservation can always be reenabled if the postcondition is precise. Non-frame-preserving operations arise in non-probabilistic separation logics too [Spies et al. 2025; Vindum et al. 2025]. The idea of having two types of probabilistic separation was also explored in LINA, where weak separation corresponds to *negative dependence* [Bao et al. 2022].

The notion of *precision* has been previously studied in separation logics, in part to explain when the separating conjunction distributes over other logical connectives, such as the regular conjunction [Calcagno et al. 2007; Vafeiadis 2011]. In PSL, precision (under the name *supported*) was used to ensure that the guard of an if statement remains independent of the postcondition of the two branches. Conditioning à la Lilac and Bluebell provides a more flexible way to reason about control flow without forcing the guard to be independent of the states in the two branches.

Another category of probabilistic separation logics build on Iris [Jung et al. 2018], from which they inherit expressive features, including ghost state and impredicative invariant reasoning. In these logics, separating conjunctions have the usual meaning from CSL, and do not capture probabilistic independence. Lohse and Garg [2024] and Haselwarter et al. [2024b] develop logics for proving bounds on the expected runtime of a randomized program. Aguirre et al. [2024] apply a similar approach for upper bounding the probability that a postcondition will fail to hold in sequential programs, and Li et al. [2025] extended this work to the concurrent setting. Additional logics have also been developed for relational reasoning and refinement [Gregersen et al. 2024a,b; Haselwarter

et al. 2024a]. The tradeoff is that they focus on a narrow property about programs' probabilistic behaviors, *e.g.,* only capturing a bound on an expected cost or probability of a single event. Outcome Separation Logic uses a more primitive form of heap separation, but is backed by a denotational model that supports specifications about the distribution of outcomes [Zilberstein et al. 2024].

8 Conclusion

This paper brings together ideas from concurrent, probabilistic separation logics, and Demonic Outcome Logic in developing pcOL, a new expressive logic for analysis of probabilistic concurrent programs. Although pcOL represents a significant step in reasoning for randomized concurrent programs, more work remains to be done. Fine-grained concurrency analysis is notoriously complex, and we plan to augment pcOL in the following ways to support more expressive verification.

Fair Termination and Synchronization. Our BoundedRank rule makes no assumptions about fairness (meaning that no thread can be indefinitely starved); indeed it applies only to programs for which the probability of eventual termination does not depend on the scheduler. However, many probabilistic synchronization protocols [Hart et al. 1983] such as the Dining Philosophers problem [Lehmann and Rabin 1981] only terminate under a fair scheduler. We would like to extend pcOL to reason about these programs, but it will present significant challenges. Fairness is not a compositional property, so many of the properties of $\mathcal{L}(\llbracket - \rrbracket)$ that we rely on for soundness would not hold.

Dynamic Allocation and Resource Algebras. As with PSL and all logics that build on it, our resource model uses variables rather than pointers. However, most concurrent programs use pointers. Modern CSL implementations such as Iris [Jung et al. 2018, 2015] use resource algebras, so that additional types of physical and logical state can be added to govern the ways in which concurrent threads modify shared resources. Bluebell already includes permissions, which help to duplicate knowledge about read-only variables [Bao et al. 2025], however many other resources are used in practice and it is not yet understood how those resources interact with the independence model of separation. We plan to augment the model of pcOL to support Iris style resource algebras.

Mechanization. As we saw in Section 6 and Appendix F, pcOL derivations are quite involved—even for small programs—due to the handling of invariants, conditioning, and case analysis on shared state. Verification of larger programs would be infeasible with pen-and-paper proofs, therefore we plan to mechanize pcOL in the Lean proof assistant [de Moura et al. 2015]. Lean is the ideal choice because much the underlying probability theory, domain theory, and topology needed to support pcOL are already formalized in mathlib [mathlib Community 2020], although more work would still be needed to formalize all the foundational theories of pcOL, such as the convex powerdomain.

Acknowledgements

This work was supported by the National Science Foundation under awards 2504142 and 2504143 and ARIA's Safeguarded AI programme.

References

Alejandro Aguirre, Philipp G. Haselwarter, Markus de Medeiros, Kwing Hei Li, Simon Oddershede Gregersen, Joseph Tassarotti, and Lars Birkedal. 2024. Error Credits: Resourceful Reasoning about Error Bounds for Higher-Order Probabilistic Programs. *Proc. ACM Program. Lang.* 8, ICFP, Article 246 (Aug. 2024), 33 pages. https://doi.org/10.1145/3674635

Krzysztof Apt and Gordon Plotkin. 1986. Countable nondeterminism and random assignment. *J. ACM* 33, 4 (aug 1986), 724–767. https://doi.org/10.1145/6490.6494

Axel Bacher, Olivier Bodini, Alexandros Hollender, and Jérémie Lumbroso. 2015. MergeShuffle: A Very Fast, Parallel Random Permutation Algorithm. arXiv:1508.03167 [cs.DS] https://arxiv.org/abs/1508.03167

Jialu Bao, Simon Docherty, Justin Hsu, and Alexandra Silva. 2021. A Bunched Logic for Conditional Independence. In *Proceedings of the 36th Annual ACM/IEEE Symposium on Logic in Computer Science* (Rome, Italy) *(LICS '21)*. Association for Computing Machinery, New York, NY, USA, Article 13, 14 pages. https://doi.org/10.1109/LICS52264.2021.9470712

Jialu Bao, Emanuele D'Osualdo, and Azadeh Farzan. 2025. Bluebell: An Alliance of Relational Lifting and Independence for Probabilistic Reasoning. *Proc. ACM Program. Lang.* 9, POPL, Article 58 (Jan. 2025), 31 pages. https://doi.org/10.1145/3704894

Jialu Bao, Marco Gaboardi, Justin Hsu, and Joseph Tassarotti. 2022. A Separation Logic for Negative Dependence. *Proc. ACM Program. Lang.* 6, POPL, Article 57 (jan 2022), 29 pages. https://doi.org/10.1145/3498719

Gilles Barthe, Justin Hsu, and Kevin Liao. 2020. A Probabilistic Separation Logic. *Proc. ACM Program. Lang.* 4, POPL, Article 55 (Jan. 2020), 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. *Proc. ACM Program. Lang.* 3, POPL, Article 34 (Jan 2019), 29 pages. https://doi.org/10.1145/3290347

Michael Ben-Or. 1983. Another advantage of free choice (Extended Abstract): Completely asynchronous agreement protocols. In *Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing* (Montreal, Quebec, Canada) *(PODC '83)*. Association for Computing Machinery, New York, NY, USA, 27–30. https://doi.org/10.1145/800221.806707

Lars Birkedal and Hongseok Yang. 2007. Relational Parametricity and Separation Logic. In *Foundations of Software Science and Computational Structures*. Springer Berlin Heidelberg, Berlin, Heidelberg, 93–107. https://doi.org/10.1007/978-3-540-71389-0_8

Stephen Brookes. 2004. A Semantics for Concurrent Separation Logic. In *CONCUR 2004 - Concurrency Theory*, Philippa Gardner and Nobuko Yoshida (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 16–34. https://doi.org/10.1007/978-3-540-28644-8_2

Cristiano Calcagno, Peter W. O'Hearn, and Hongseok Yang. 2007. Local Action and Abstract Separation Logic. In *22nd Annual IEEE Symposium on Logic in Computer Science (LICS 2007)*. 366–378. https://doi.org/10.1109/LICS.2007.30

Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. 1998. Private information retrieval. *J. ACM* 45, 6 (Nov. 1998), 965–981. https://doi.org/10.1145/293347.293350

Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. 2015. The Lean Theorem Prover (System Description). In *Automated Deduction - CADE-25*, Amy P. Felty and Aart Middeldorp (Eds.). Springer International Publishing, Cham, 378–388. https://doi.org/10.1007/978-3-319-21401-6_26

Simon Docherty. 2019. *Bunched logics: a uniform approach*. Ph.D. Dissertation. University College London. https://discovery.ucl.ac.uk/id/eprint/10073115/

Weijie Fan, Hongjin Liang, Xinyu Feng, and Hanru Jiang. 2025. A Program Logic for Concurrent Randomized Programs in the Oblivious Adversary Model. In *Programming Languages and Systems*, Viktor Vafeiadis (Ed.). Springer Nature Switzerland, Cham, 322–348. https://doi.org/10.1007/978-3-031-91118-7_13

Ira Fesefeldt, Joost-Pieter Katoen, and Thomas Noll. 2022. Towards Concurrent Quantitative Separation Logic. In *33rd International Conference on Concurrency Theory (CONCUR 2022) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 243)*, Bartek Klin, Sławomir Lasota, and Anca Muscholl (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 25:1–25:24. https://doi.org/10.4230/LIPIcs.CONCUR.2022.25

Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. 1985. Impossibility of distributed consensus with one faulty process. *J. ACM* 32, 2 (April 1985), 374–382. https://doi.org/10.1145/3149.214121

Ronald A. Fisher and Frank Yates. 1938. *Statistical Tables: for biological , agricultural and medical research* (4th ed. ed.). Oliver and Boyd, Edinburgh.

Philippe Flajolet. 1985. Approximate counting: A detailed analysis. *BIT* 25, 1 (March 1985), 113–134. https://doi.org/10.1007/BF01934993

David Fremlin. 2001. *Measure Theory, Volume 2*.

Jay L. Gischer. 1988. The equational theory of pomsets. *Theoretical Computer Science* 61, 2 (1988), 199–224. https://doi.org/10.1016/0304-3975(88)90124-7

Simon Oddershede Gregersen, Alejandro Aguirre, Philipp G. Haselwarter, Joseph Tassarotti, and Lars Birkedal. 2024a. Almost-Sure Termination by Guarded Refinement. *Proc. ACM Program. Lang.* 8, ICFP, Article 243 (Aug. 2024), 31 pages. https://doi.org/10.1145/3674632

Simon Oddershede Gregersen, Alejandro Aguirre, Philipp G. Haselwarter, Joseph Tassarotti, and Lars Birkedal. 2024b. Asynchronous Probabilistic Couplings in Higher-Order Separation Logic. *Proc. ACM Program. Lang.* 8, POPL, Article 26 (Jan. 2024), 32 pages. https://doi.org/10.1145/3632868

Sergiu Hart, Micha Sharir, and Amir Pnueli. 1983. Termination of Probabilistic Concurrent Program. *ACM Trans. Program. Lang. Syst.* 5, 3 (July 1983), 356–380. https://doi.org/10.1145/2166.357214

Philipp G. Haselwarter, Kwing Hei Li, Alejandro Aguirre, Simon Oddershede Gregersen, Joseph Tassarotti, and Lars Birkedal. 2024a. Approximate Relational Reasoning for Higher-Order Probabilistic Programs. arXiv:2407.14107 [cs.LO]

https://arxiv.org/abs/2407.14107

Philipp G. Haselwarter, Kwing Hei Li, Markus de Medeiros, Simon Oddershede Gregersen, Alejandro Aguirre, Joseph Tassarotti, and Lars Birkedal. 2024b. Tachis: Higher-Order Separation Logic with Credits for Expected Costs. arXiv:2405.20083 [cs.LO] https://arxiv.org/abs/2405.20083

Jifeng He, Karen Seidel, and Annabelle McIver. 1997. Probabilistic models for the guarded command language. *Science of Computer Programming* 28, 2 (1997), 171–192. https://doi.org/10.1016/S0167-6423(96)00019-6 Formal Specifications: Foundations, Methods, Tools and Applications.

Charles Antony Richard Hoare. 1969. An Axiomatic Basis for Computer Programming. *Commun. ACM* 12, 10 (Oct. 1969), 576–580. https://doi.org/10.1145/363235.363259

Bart Jacobs. 2008. Coalgebraic Trace Semantics for Combined Possibilitistic and Probabilistic Systems. *Electronic Notes in Theoretical Computer Science* 203, 5 (2008), 131–152. https://doi.org/10.1016/j.entcs.2008.05.023 Proceedings of the Ninth Workshop on Coalgebraic Methods in Computer Science (CMCS 2008).

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). https://doi.org/10.1017/S0956796818000151

Ralf Jung, David Swasey, Filip Sieczkowski, Kasper Svendsen, Aaron Turon, Lars Birkedal, and Derek Dreyer. 2015. Iris: Monoids and Invariants as an Orthogonal Basis for Concurrent Reasoning. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (Mumbai, India) *(POPL '15)*. Association for Computing Machinery, New York, NY, USA, 637–650. https://doi.org/10.1145/2676726.2676980

Klaus Keimel and Gordon Plotkin. 2017. Mixed powerdomains for probability and nondeterminism. *Logical Methods in Computer Science* Volume 13, Issue 1 (Jan. 2017). https://doi.org/10.23638/LMCS-13(1:2)2017

Daniel Lehmann and Michael O. Rabin. 1981. On the advantages of free choice: a symmetric and fully distributed solution to the dining philosophers problem. In *Proceedings of the 8th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (Williamsburg, Virginia) *(POPL '81)*. Association for Computing Machinery, New York, NY, USA, 133–138. https://doi.org/10.1145/567532.567547

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 (jun 2023), 24 pages. https://doi.org/10.1145/3591226

Kwing Hei Li, Alejandro Aguirre, Simon Oddershede Gregersen, Philipp G. Haselwarter, Joseph Tassarotti, and Lars Birkedal. 2025. Modular Reasoning about Error Bounds for Concurrent Probabilistic Programs. *Proc. ACM Program. Lang.* 9, ICFP, Article 245 (Aug. 2025), 30 pages. https://doi.org/10.1145/3747514

Janine Lohse and Deepak Garg. 2024. An Iris for Expected Cost Analysis. arXiv:2406.00884 [cs.PL] https://arxiv.org/abs/2406.00884

George Markowsky. 1976. Chain-complete posets and directed sets with applications. *Algebra Universalis* 6 (12 1976), 53–68. https://doi.org/10.1007/BF02485815

The mathlib Community. 2020. The lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs* (New Orleans, LA, USA) *(CPP 2020)*. Association for Computing Machinery, New York, NY, USA, 367–381. https://doi.org/10.1145/3372885.3373824

Annabelle McIver and Carroll Morgan. 2005. *Abstraction, Refinement and Proof for Probabilistic Systems*. Springer. https://doi.org/10.1007/b138392

Annabelle McIver, Tahiry Rabehaja, and Georg Struth. 2016. Probabilistic rely-guarantee calculus. *Theoretical Computer Science* 655 (2016), 120–134. https://doi.org/10.1016/j.tcs.2016.01.016 Quantitative Aspects of Programming Languages and Systems (2013-14).

Robert Morris. 1978. Counting large numbers of events in small registers. *Commun. ACM* 21, 10 (Oct. 1978), 840–842. https://doi.org/10.1145/359619.359627

Peter W. O'Hearn. 2004. Resources, Concurrency and Local Reasoning. In *CONCUR 2004 - Concurrency Theory*. Springer Berlin Heidelberg, Berlin, Heidelberg, 49–67. https://doi.org/10.1016/j.tcs.2006.12.035

Peter W. O'Hearn and David J. Pym. 1999. The Logic of Bunched Implications. *The Bulletin of Symbolic Logic* 5, 2 (1999), 215–244.

Peter W. O'Hearn, John C. Reynolds, and Hongseok Yang. 2001. Local Reasoning about Programs That Alter Data Structures. In *Proceedings of the 15th International Workshop on Computer Science Logic (CSL '01)*. Springer-Verlag, Berlin, Heidelberg, 1–19. https://doi.org/10.1007/3-540-44802-0_1

Vaughan R. Pratt. 1986. Modeling Concurrency with Partial Orders. *Int. J. Parallel Program.* 15, 1 (1986), 33–71. https://doi.org/10.1007/BF01379149

Michael O. Rabin. 1980. N-process synchronization by 4.log2N-valued shared variable. In *21st Annual Symposium on Foundations of Computer Science (sfcs 1980)*. 407–410. https://doi.org/10.1109/SFCS.1980.26

Michael O. Rabin. 1982. The choice coordination problem. *Acta Inf.* 17, 2 (June 1982), 121–134. https://doi.org/10.1007/BF00288965

J.C. Reynolds. 2002. Separation logic: a logic for shared mutable data structures. In *Proceedings 17th Annual IEEE Symposium on Logic in Computer Science*. 55–74. https://doi.org/10.1109/LICS.2002.1029817

H. L. Royden. 1968. *Real Analysis* (2d ed. ed.). Macmillan, New York.

Michael Smyth. 1978. Power domains. *J. Comput. System Sci.* 16, 1 (1978), 23–36. https://doi.org/10.1016/0022-0000(78)90048-X

Simon Spies, Niklas Mück, Haoyi Zeng, Michael Sammler, Andrea Lattuada, Peter Müller, and Derek Dreyer. 2025. Destabilizing Iris. *Proc. ACM Program. Lang.* 9, PLDI, Article 181 (June 2025), 26 pages. https://doi.org/10.1145/3729284

Joseph Tassarotti. 2018. *Verifying Concurrent Randomized Algorithms*. Ph.D. Dissertation. Carnegie Mellon University. https://csd.cmu.edu/academics/doctoral/degrees-conferred/joseph-tassarotti

Joseph Tassarotti and Robert Harper. 2019. A Separation Logic for Concurrent Randomized Programs. *Proc. ACM Program. Lang.* 3, POPL, Article 64 (Jan 2019), 30 pages. https://doi.org/10.1145/3290377

Regina Tix, Klaus Keimel, and Gordon Plotkin. 2009. Semantic Domains for Combining Probability and Non-Determinism. *Electronic Notes in Theoretical Computer Science* 222 (2009), 3–99. https://doi.org/10.1016/j.entcs.2009.01.002

Viktor Vafeiadis. 2011. Concurrent Separation Logic and Operational Semantics. *Electronic Notes in Theoretical Computer Science* 276 (2011), 335–351. https://doi.org/10.1016/j.entcs.2011.09.029 Twenty-seventh Conference on the Mathematical Foundations of Programming Semantics (MFPS XXVII).

Daniele Varacca. 2002. The powerdomain of indexed valuations. In *Proceedings 17th Annual IEEE Symposium on Logic in Computer Science*. 299–308. https://doi.org/10.1109/LICS.2002.1029838

Daniele Varacca and Glynn Winskel. 2006. Distributing probability over non-determinism. *Mathematical Structures in Computer Science* 16, 1 (2006), 87–113. https://doi.org/10.1017/S0960129505005074

Simon Friis Vindum, Aïna Linn Georges, and Lars Birkedal. 2025. The Nextgen Modality: A Modality for Non-Frame-Preserving Updates in Separation Logic. In *Proceedings of the 14th ACM SIGPLAN International Conference on Certified Programs and Proofs* (Denver, CO, USA) *(CPP '25)*. Association for Computing Machinery, New York, NY, USA, 83–97. https://doi.org/10.1145/3703595.3705876

John von Neumann. 1951. Various techniques used in connection with random digits. In *Monte Carlo Method*, A.S. Householder, G.E. Forsythe, and H.H. Germond (Eds.). National Bureau of Standards Applied Mathematics Series, 12, Washington, D.C.: U.S. Government Printing Office, 36–38.

Pengbo Yan, Toby Murray, Olga Ohrimenko, Van-Thuan Pham, and Robert Sison. 2025. Combining Classical and Probabilistic Independence Reasoning to Verify the Security of Oblivious Algorithms. In *Formal Methods*, André Platzer, Kristin Yvonne Rozier, Matteo Pradella, and Matteo Rossi (Eds.). Springer Nature Switzerland, Cham, 188–205. https://doi.org/10.1007/978-3-031-71162-6_10

Hongseok Yang and Peter O'Hearn. 2002. A Semantic Basis for Local Reasoning. In *Foundations of Software Science and Computation Structures*. Springer Berlin Heidelberg, Berlin, Heidelberg, 402–416. https://doi.org/10.1007/3-540-45931-6_28

Linpeng Zhang, Noam Zilberstein, Benjamin Lucien Kaminski, and Alexandra Silva. 2024. Quantitative Weakest Hyper Pre: Unifying Correctness and Incorrectness Hyperproperties via Predicate Transformers. *Proc. ACM Program. Lang.* 8, OOPSLA2, Article 300 (oct 2024), 30 pages. https://doi.org/10.1145/3689740

Noam Zilberstein. 2025. Outcome Logic: A Unified Approach to the Metatheory of Program Logics with Branching Effects. *ACM Trans. Program. Lang. Syst.* 47, 3, Article 14 (Sept. 2025), 71 pages. https://doi.org/10.1145/3743131

Noam Zilberstein, Derek Dreyer, and Alexandra Silva. 2023. Outcome Logic: A Unifying Foundation for Correctness and Incorrectness Reasoning. *Proc. ACM Program. Lang.* 7, OOPSLA1, Article 93 (Apr 2023), 29 pages. https://doi.org/10.1145/3586045

Noam Zilberstein, Daniele Gorla, and Alexandra Silva. 2025a. Denotational Semantics for Probabilistic and Concurrent Programs. In *36th International Conference on Concurrency Theory (CONCUR 2025) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 348)*, Patricia Bouyer and Jaco van de Pol (Eds.). Schloss Dagstuhl – Leibniz-Zentrum f"ur Informatik, Dagstuhl, Germany, 39:1–39:24. https://doi.org/10.4230/LIPIcs.CONCUR.2025.39

Noam Zilberstein, Dexter Kozen, Alexandra Silva, and Joseph Tassarotti. 2025b. A Demonic Outcome Logic for Randomized Nondeterminism. *Proc. ACM Program. Lang.* 9, POPL, Article 19 (Jan 2025), 30 pages. https://doi.org/10.1145/3704855

Noam Zilberstein, Angelina Saliling, and Alexandra Silva. 2024. Outcome Separation Logic: Local Reasoning for Correctness and Incorrectness with Computational Effects. *Proc. ACM Program. Lang.* 8, OOPSLA1 (Apr 2024). https://doi.org/10.1145/3649821

Noam Zilberstein, Alexandra Silva, and Joseph Tassarotti. 2025c. Probabilistic Concurrent Reasoning in Outcome Logic: Independence, Conditioning, and Invariants (Full Version). arXiv:2411.11662 [cs.LO] https://arxiv.org/abs/2411.11662

Maaike Zwart and Dan Marsden. 2019. No-Go Theorems for Distributive Laws. In *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 1–13. https://doi.org/10.1109/lics.2019.8785707

Appendix

A Definition of Program Semantics

In this section, we give details on the *Pomsets with Formulae* semantic model due to Zilberstein et al. [2025a].

A.1 Pomsets with Formulae

Whereas standard pomsets (partially ordered multisets) use a partial order to record the causality between atomic actions in the program (elements of a multiset) [Gischer 1988; Pratt 1986], pomsets with formulae add a Boolean formula to each node, which records which tests must succeed or fail to reach that point in the execution. This is necessary to capture probabilistic concurrency, since tests may not always pass or fail, but rather they may only have some probability of passing, and so both paths must be represented in a single semantic structure. We briefly introduce pomsets with formulae here, refer to Zilberstein et al. [2025a] for a more complete treatment.

Let $\mathit{label} \triangleq \mathrm{Act} \cup \mathrm{Test} \cup \{\bullet, \bot\}$, which can be either an action or test (from Figure 1), a no-op ($\bullet$), or undetermined ($\bot$). Although $\bot$ labels will never appear in the denotation of a *complete* program, they are used to indicate that a finite structure is an approximation of an infinite one, *i.e.,* in the semantics of while loops. Also, let *nodes* be a countable universe of identifiers and *form* be the set of Boolean formulae over *nodes*. For some $\psi \in \mathit{form}$, we write $\mathrm{sat}(\psi)$ to indicate that ψ is satisfiable, and $\mathrm{vars}(\psi) \subseteq \mathit{nodes}$ is the set of variables referenced in ψ. We now define the underlying structure.

Definition A.1 (Labelled Partial Order with Formulae (LPOFs)). An LPOF is a 4-tuple $\langle N, <, \lambda, \varphi \rangle \in \mathit{lpo}$ where:

(1) $N \subseteq \mathit{nodes}$ is a countable set of nodes;
(2) $\langle N, < \rangle$ is a strict poset with a single minimal element such that finitely many nodes appear at every finite distance from the root;
(3) $\lambda \colon N \to \mathit{label}$ is a labelling function such that x has no successors whenever $\lambda(x) = \bot$;
(4) $\varphi \colon N \to \mathit{form}$ is a formula function such that: $\varphi(y) \Rightarrow \varphi(x)$, for all $x < y$ and for all $x \in N$, $\mathrm{sat}(\varphi(x))$ and $y < x$ for all $y \in \mathrm{vars}(\varphi(x))$.

For some $\alpha \in \mathit{lpo}$, we will often use N_α, $<_\alpha$, λ_α, and φ_α to refer to its parts.

The first three components are standard in pomset semantics. The order denotes *causality*, so $x < y$ means that the action $\lambda(x)$ must be scheduled before $\lambda(y)$. The order is partial (not total) because actions that occur in parallel are not related. Formulae are a new addition to this structure, allowing us to encode guarded branching in the structure too. For any node $x \in N$, $\varphi(x)$ is a satisfiable formula comprised of nodes appearing earlier in the trace, since the corresponding tests must be resolved before executing later actions. In addition, formulae can only become stronger as the trace goes on, since dependencies on more and more tests are accumulated over time.

Two LPOFs are isomorphic, denoted $\alpha \equiv \beta$ iff there is a bijection $f \colon N_\alpha \to N_\beta$ such that $x <_\alpha y$ iff $f(x) <_\beta f(y)$, $\lambda_\alpha = \lambda_\beta \circ f$, and $\varphi_\alpha = f^{-1} \circ \varphi_\beta \circ f$, where $f^{-1}(\psi)$ renames the variables of ψ in the obvious way. Given that, a pomset with formulae is an equivalence class of LPOFs.

Definition A.2 (Pomsets with Formulae). We denote the isomorphism class of α as $[\alpha] \triangleq \{\beta \in \mathit{lpo} \mid \alpha \equiv \beta\}$. A pomset with formulae (or, simply, a pomset) $\boldsymbol{\alpha} \in \mathit{pom}$ is an isomorphism class of LPOFs.

$$\mathit{pom} \triangleq \{[\alpha] \mid \alpha \in \mathit{lpo}\}$$

Pomsets with formulae have many nice domain-theoretic properties. For example, they have a DCPO (directed complete partial order) structure, meaning that Scott continuous operations over them have least fixed points. Roughly speaking $\boldsymbol{\alpha} \sqsubseteq_{\mathit{pom}} \boldsymbol{\beta}$ iff $\boldsymbol{\beta}$ is obtained by replacing nodes labelled $\bot$ in $\boldsymbol{\alpha}$ with a new, larger structure. We will see a concrete example of this shortly when we discuss the semantics of loops. Infinite pomsets with formulae can also be represented as the supremum of their finite approximation, and monotone operations on those finite approximations can be extended to continuous operations on infinite pomsets. This is useful, since it is not possible to define operations inductively on infinite structures, as we will see in Appendix A.2.

The trace semantics $[\![-]\!] : \mathrm{Cmd} \to \mathit{pom}$, shown in Figure 11, interprets every command as a pomset with formulae. We use several operators to construct pomsets. First, $\langle - \rangle \colon \mathit{label} \to \mathit{pom}$ constructs a singleton pomset with the given label. This is used for the semantics of **skip** and actions $a \in \mathrm{Act}$. Next, we have three combinators for combining pomsets, which we demonstrate pictorially below. The first is sequential composition $\boldsymbol{\alpha} \fatsemi \boldsymbol{\beta}$, which constructs a pomset where all the actions from $\boldsymbol{\alpha}$ occur before those in $\boldsymbol{\beta}$. In the diagram, $a_1 \to a_2$ indicates causality (a_1 occurs before a_2)

The second combinator is guarded choice $\mathrm{guard}(b, \boldsymbol{\alpha}, \boldsymbol{\beta})$, where the test b becomes the new root and all formulae of the nodes in $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are updated to indicate that the new root must pass or fail, respectively, as indicated by the arrows labelled T and F. Finally, parallel composition $\boldsymbol{\alpha} \parallel \boldsymbol{\beta}$ joins the two pomsets with a new no-op root, so that the actions of $\boldsymbol{\alpha}$

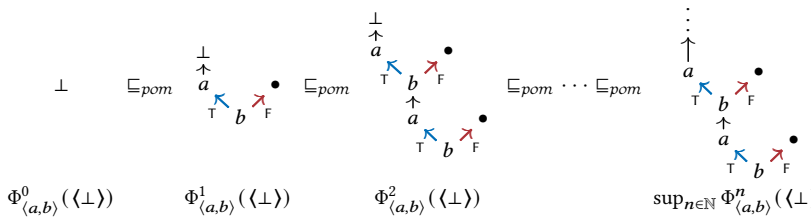
$$
\begin{aligned}
[\![\texttt{skip}]\!] &\triangleq \langle \bullet \rangle \\
[\![C_1 \fatsemi C_2]\!] &\triangleq [\![C_1]\!] \fatsemi [\![C_2]\!] \\
[\![C_1 \parallel C_2]\!] &\triangleq [\![C_1]\!] \parallel [\![C_2]\!] \\
[\![\texttt{if } b \texttt{ then } C_1 \texttt{ else } C_2]\!] &\triangleq \mathrm{guard}(b, [\![C_1]\!], [\![C_2]\!]) \\
[\![\texttt{while } b \texttt{ do } C]\!] &\triangleq \mathrm{lfp}\left(\Phi_{\langle b,C\rangle}\right) \quad \text{where } \Phi_{\langle b,C\rangle}(\boldsymbol{\alpha}) \triangleq \mathrm{guard}\left(b, [\![C]\!] \fatsemi \boldsymbol{\alpha}, \langle \bullet \rangle\right) \\
[\![a]\!] &\triangleq \langle a \rangle
\end{aligned}
$$

Fig. 11. Trace semantics for commands $[\![-]\!] : \mathsf{Cmd} \to \mathit{pom}$.

and $\boldsymbol{\beta}$ can be interleaved without restriction.

$\langle a_1 \rangle \fatsemi \langle a_2 \rangle =$  $\quad \mathrm{guard}(b, \langle a_1 \rangle, \langle a_2 \rangle) =$  $\quad \langle a_1 \rangle \parallel \langle a_2 \rangle =$ 

The semantics of loops is given by a least fixed point of the characteristic function $\Phi_{\langle b,C\rangle}$, which is Scott continuous since it is defined using guard and $\fatsemi$, both of which are Scott continuous [Zilberstein et al. 2025a]. This means that the least fixed point is equal to the supremum of the finite unrollings of $\Phi_{\langle b,C\rangle}$, as shown in the following diagram.



$\Phi^0_{\langle a,b\rangle}(\langle\bot\rangle)$ $\Phi^1_{\langle a,b\rangle}(\langle\bot\rangle)$ $\Phi^2_{\langle a,b\rangle}(\langle\bot\rangle)$ $\sup_{n\in\mathbb{N}} \Phi^n_{\langle a,b\rangle}(\langle\bot\rangle)$

In each unrolling, $\bot$ appears further and further from the root until, in the supremum, it is pushed infinitely far away. That infinite spine represents the execution in which the guard b never becomes false. In the context of probabilistic programs, it is important to include the infinite spine, as the probability of continuing to execute may only converge to 0 in the limit.

A.2 Linearization

The benefit of the pomset model is that it is fully compositional; the parallel composition of two commands is interpreted as a straightforward combination of their syntactic structures. However, for the purposes of designing a program logic, we need a *state transformer* model, which maps each input state to a convex set of distributions over output states. For this purpose, Zilberstein et al. [2025a] also define linearization $\mathcal{L} : \mathit{pom} \to \mathsf{Mem}[S] \to C(\mathsf{Mem}[S])$. The definition is repeated below.

$$
\begin{aligned}
\mathsf{next}(\alpha, \psi, S) &= \{x \in N_\alpha \setminus S \mid \downarrow x \subseteq S, \psi \Rightarrow \varphi_\alpha(x)\} \\
\mathsf{next}^\star(\alpha, \psi, S) &= \{x \in N_\alpha \setminus S \mid \mathsf{sat}(\psi \wedge \varphi_\alpha(x))\} \\
\mathcal{L}^{\mathcal{I}}_{node}(\alpha, \psi, S, x)(\sigma) &= \begin{cases}
\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha, \psi, S \cup \{x\})^\dagger\left([\![\lambda_\alpha(x)]\!]^{\mathcal{I}}_{\mathsf{Act}}(\sigma)\right) & \text{if } \lambda_\alpha(x) \in \mathsf{Act} \\
\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha, \psi \wedge (\!|x = [\![\lambda_\alpha(x)]\!]_{\mathsf{Test}}(\sigma)|\!), S \cup \{x\})(\sigma) & \text{if } \lambda_\alpha(x) \in \mathsf{Test} \\
\bot_C & \text{if } \lambda_\alpha(x) = \bot \\
\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha, \psi, S \cup \{x\})(\sigma) & \text{if } \lambda_\alpha(x) = \bullet
\end{cases} \\
\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha, \psi, S)(\sigma) &= \begin{cases}
\eta(\sigma) & \text{if } \mathsf{next}(\alpha, \psi, S) = \emptyset \\
\&_{x \in \mathsf{next}(\alpha,\psi,S)}\, \mathcal{L}^{\mathcal{I}}_{node}(\alpha, \psi, S, x)(\sigma) & \text{if } \mathsf{next}(\alpha, \psi, S) \neq \emptyset
\end{cases} \\
\mathcal{L}^{\mathcal{I}}_{\mathsf{fin}}([\alpha]) &= \mathcal{L}^{\mathcal{I}}_{lpo}(\alpha, \mathsf{true}, \emptyset) \\
\mathcal{L}^{\mathcal{I}}(\boldsymbol{\alpha}) &= \sup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \mathcal{L}^{\mathcal{I}}_{\mathsf{fin}}(\boldsymbol{\alpha}')
\end{aligned}
$$

The function $\mathsf{next}(\alpha, \psi, S)$ gives the set of nodes that are ready to be scheduled, given a finite LPOF α, a path condition ψ, and the set of nodes S that have already been processed. Linearizing a node $\mathcal{L}_{node}$ does case analysis on the label of the node to decide how to proceed. If the node is an action, then the action is evaluated and composed with the linearization of the remainder of the LPOF. If the node is a test, then the result of the test is added to the path condition. If the node is $\bot$, then the execution is halted. If it is a no-op, then the execution simply continues.

To linearize an entire finite LPOF, we simply perform a nondeterministic choice over all the next nodes, linearize those nodes, and then proceed processing the rest of the structure recursively. To make the recursion well-founded, the structure must be finite, but we extend to the infinite case later. To linearize a finite pomset $\boldsymbol{\alpha}$, we simply linearize any representative LPOF $\alpha \in \boldsymbol{\alpha}$. Finally, to linearize an infinite pomset, we take the supremum over the linearization of all the finite approximations, where $\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}$ indicates that $\boldsymbol{\alpha}'$ is a finite approximation of $\boldsymbol{\alpha}$.

Since most operations are continuous, we get, *e.g.*, that the supremum of finite approximations of a sequential composition is equal to the sequential composition over the supremum of finite approximations:

$$\sup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \sup_{\boldsymbol{\beta}' \ll \boldsymbol{\beta}} \boldsymbol{\alpha}' \fatsemi \boldsymbol{\beta}' = \left(\sup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \boldsymbol{\alpha}' \right) \fatsemi \left(\sup_{\boldsymbol{\beta}' \ll \boldsymbol{\beta}} \boldsymbol{\beta}' \right) = \boldsymbol{\alpha} \fatsemi \boldsymbol{\beta}$$

The exception to this is for parallel composition, which is not continuous for technical reasons. However, if we define $\ll_1$ as follows:

$$\boldsymbol{\gamma} \ll_1 \boldsymbol{\gamma}' \qquad \text{iff} \qquad \boldsymbol{\gamma} \ll \boldsymbol{\gamma}' \quad \text{and} \quad \mathrm{root}(\boldsymbol{\gamma}) = \mathrm{root}(\boldsymbol{\gamma}') = \bullet \quad \text{or} \quad \mathrm{root}(\boldsymbol{\gamma}') \neq \bullet$$

where $\mathrm{root}(\boldsymbol{\gamma})$ is the label of the root node of the pomset $\boldsymbol{\gamma}$, then we get the following weaker pseudo-continuity property:

$$\sup_{\boldsymbol{\alpha}' \ll_1 \boldsymbol{\alpha}} \sup_{\boldsymbol{\beta}' \ll_1 \boldsymbol{\beta}} \boldsymbol{\alpha}' \parallel \boldsymbol{\beta}' = \left(\sup_{\boldsymbol{\alpha}' \ll_1 \boldsymbol{\alpha}} \boldsymbol{\alpha}' \right) \parallel \left(\sup_{\boldsymbol{\beta}' \ll_1 \boldsymbol{\beta}} \boldsymbol{\beta}' \right) = \boldsymbol{\alpha} \parallel \boldsymbol{\beta}$$

B General Lemmas

B.1 Measure Theory Lemmas

We begin by proving some general properties for characterizing discrete product spaces. By Lemma C.1 and C.2 of Bao et al. [2025], for any discrete probability space $\mathcal{P}$, there exists a countable partition $\{A_i \mid i \in I\}$ of $\Omega_{\mathcal{P}}$ such that $\mathcal{F}_{\mathcal{P}} = \{\biguplus_{i \in I'} A_i \mid I' \subseteq I\}$. Let $\mathrm{ev}(\mathcal{P})$ denote this partition. It follows that for every $A \in \mathcal{F}_{\mathcal{P}}$, $\mu_{\mathcal{P}}(A) = \sum_{B \in S} \mu_{\mathcal{P}}(B)$ for some $S \subseteq \mathrm{ev}(\mathcal{P})$.

Lemma B.1. *For any probability spaces $\mathcal{P}_1$ and $\mathcal{P}_2$*

$$\mathcal{F}_{\mathcal{P}_1 \otimes \mathcal{P}_2} = \left\{ \biguplus_{(A,B) \in S} A * B \;\middle|\; S \subseteq \mathrm{ev}(\mathcal{P}_1) \times \mathrm{ev}(\mathcal{P}_2) \right\}$$

Proof. The forward inclusion follows immediately from Lemma C.5 of Bao et al. [2025], so it suffices to only show the reverse inclusion. Take any $S \subseteq \mathrm{ev}(\mathcal{P}_1) \times \mathrm{ev}(\mathcal{P}_2)$. For each $(A, B) \in S \subseteq \mathrm{ev}(\mathcal{P}_1) \times \mathrm{ev}(\mathcal{P}_2)$, clearly $A \in \mathrm{ev}(\mathcal{P}_1)$ and $B \in \mathrm{ev}(\mathcal{P}_2)$, so $A * B \in \mathcal{F}_{\mathcal{P}_1 \otimes \mathcal{P}_2}$ since by definition $\mathcal{F}_{\mathcal{P}_1 \otimes \mathcal{P}_2}$ is the smallest sigma algebra containing $\{A * B \mid A \in \mathcal{F}_{\mathcal{P}_1}, B \in \mathcal{F}_{\mathcal{P}_2}\}$. Since $\mathcal{F}_{\mathcal{P}_1 \otimes \mathcal{P}_2}$ is closed under countable unions, then $\biguplus_{(A,B) \in S} A * B \in \mathcal{F}_{\mathcal{P}_1 \otimes \mathcal{P}_2}$. □

Lemma B.2. *$\mathcal{Q} = \mathcal{P}_1 \otimes \mathcal{P}_2$ iff:*

(1) $\Omega_{\mathcal{Q}} = \Omega_{\mathcal{P}_1} * \Omega_{\mathcal{P}_2}$
(2) $\mathcal{F}_{\mathcal{Q}} = \{\biguplus_{(A,B) \in S} A * B \mid S \subseteq \mathrm{ev}(\mathcal{P}_1) \times \mathrm{ev}(\mathcal{P}_2)\}$
(3) $\mu_{\mathcal{Q}}(A * B) = \mu_{\mathcal{P}_1}(A) \cdot \mu_{\mathcal{P}_2}(B)$ *for all* $A \in \mathrm{ev}(\mathcal{P}_1)$ *and* $B \in \mathrm{ev}(\mathcal{P}_2)$

Proof. The forward direction follows immediately from the definition of product spaces and Lemma B.1. We now show the reverse direction. By (1) and the definition of product spaces, $\Omega_{\mathcal{Q}} = \Omega_{\mathcal{P}_1} * \Omega_{\mathcal{P}_2} = \Omega_{\mathcal{P}_1 \otimes \mathcal{P}_2}$. By (2) and Lemma B.1, $\mathcal{F}_{\mathcal{Q}} = \mathcal{F}_{\mathcal{P}_1 \otimes \mathcal{P}_2}$. Now, take any event $A \in \mathcal{F}_{\mathcal{Q}} = \mathcal{F}_{\mathcal{P}_1 \otimes \mathcal{P}_2}$. We already know that $A = \biguplus_{(A_1,A_2) \in S} A_1 * A_2$ for some $S \subseteq \mathrm{ev}(\mathcal{P}_1) \times \mathrm{ev}(\mathcal{P}_2)$, so:

$$\begin{aligned}
\mu_{\mathcal{Q}}(A) &= \mu_{\mathcal{Q}}\left(\biguplus_{(A_1,A_2) \in S} A_1 * A_2 \right) \\
&= \sum_{(A_1,A_2) \in S} \mu_{\mathcal{Q}}(A_1 * A_2) \\
&= \sum_{(A_1,A_2) \in S} \mu_{\mathcal{P}_1}(A_1) \cdot \mu_{\mathcal{P}_2}(A_2) \\
&= \sum_{(A_1,A_2) \in S} \mu_{\mathcal{P}_1 \otimes \mathcal{P}_2}(A_1 * A_2) \\
&= \mu_{\mathcal{P}_1 \otimes \mathcal{P}_2}\left(\biguplus_{(A_1,A_2) \in S} A_1 * A_2 \right) = \mu_{\mathcal{P}_1 \otimes \mathcal{P}_2}(A)
\end{aligned}$$

□

Lemma B.3. *If both sides of the following equality are well-defined, then:*

$$\left(\bigoplus_{i\sim\nu} \mathcal{P}_i\right) \otimes \mathcal{Q} = \bigoplus_{i\sim\nu} (\mathcal{P}_i \otimes \mathcal{Q})$$

Proof. Let $\mathcal{P}_i = \langle \Omega_i, \mathcal{F}_i, \mu_i \rangle$ for each $i \in I = \mathrm{supp}(\nu)$. If the above are well defined, then there must be S and T such that $\Omega_i \subseteq \mathsf{Mem}[S]$ and $\Omega_{\mathcal{Q}} \subseteq \mathsf{Mem}[T]$. We first show that both probability spaces have the same sample space, since:

$$\begin{aligned}
\Omega_{(\bigoplus_{i\sim\nu}\mathcal{P}_i)\otimes\mathcal{Q}} &= (\bigcup_{i\in\mathrm{supp}(\nu)} \Omega_i) * \Omega_{\mathcal{Q}} \\
&= \left\{ \sigma \uplus \sigma' \;\middle|\; \sigma \in \bigcup_{i\in I} \Omega_i v, \sigma' \in \Omega_{\mathcal{Q}} \right\} \\
&= \bigcup_{i\in I} \{\sigma \uplus \sigma' \mid \sigma \in \Omega_i, \sigma' \in \Omega_{\mathcal{Q}}\} \\
&= \bigcup_{i\in I} \Omega_i * \Omega_{\mathcal{Q}} \;=\; \Omega_{\bigoplus_{i\sim\nu}(\mathcal{P}_i\otimes\mathcal{Q})}
\end{aligned}$$

To show that the σ-algebras are the same, it will suffice to show that they are generated from the same disjoint partitions [Bao et al. 2025, Lemma C.1]. We start by using Lemma B.1 to conclude that:

$$\mathrm{ev}\left(\left(\bigoplus_{i\sim\nu} \mathcal{P}_i\right) \otimes \mathcal{Q}\right) = \mathrm{ev}\left(\bigoplus_{i\sim\nu} \mathcal{P}_i\right) \times \mathrm{ev}(\mathcal{Q})$$

Since all the $\mathcal{P}_i$ have disjoint sample spaces, their partitions must also be disjoint:

$$\begin{aligned}
&= \left(\biguplus_{i\in I} \mathrm{ev}(\mathcal{P}_i)\right) \times \mathrm{ev}(\mathcal{Q}) \\
&= \biguplus_{i\in I} \mathrm{ev}(\mathcal{P}_i) \times \mathrm{ev}(\mathcal{Q}) \\
&= \biguplus_{i\in I} \mathrm{ev}(\mathcal{P}_i \otimes \mathcal{Q}) = \mathrm{ev}\left(\bigoplus_{i\sim\nu} \mathcal{P}_i \otimes \mathcal{Q}\right)
\end{aligned}$$

Finally, by Lemma B.2, it suffices to show that the product measures agree on events of the form $A * B$, where $A \in \mathcal{F}_{\bigoplus_{i\sim\nu}\mathcal{P}_i}$ and $B \in \mathcal{F}_{\mathcal{Q}}$:

$$\begin{aligned}
\mu_{(\bigoplus_{i\sim\nu}\mathcal{P}_i)\otimes\mathcal{Q}}(A * B) &= \mu_{\bigoplus_{i\sim\nu}\mathcal{P}_i}(A) \cdot \mu_{\mathcal{Q}}(B) \\
&= \left(\sum_{i\in I} \nu(i) \cdot \mu_i(A \cap \Omega_i)\right) \cdot \mu_{\mathcal{Q}}(B) \\
&= \sum_{i\in I} \nu(i) \cdot \mu_i(A \cap \Omega_i) \cdot \mu_{\mathcal{Q}}(B) \\
&= \sum_{i\in I} \nu(i) \cdot \mu_{\mathcal{P}_i\otimes\mathcal{Q}}((A \cap \Omega_i) * B) \\
&= \sum_{i\in I} \nu(i) \cdot \mu_{\mathcal{P}_i\otimes\mathcal{Q}}((A * B) \cap (\Omega_i * \Omega_{\mathcal{Q}})) \\
&= \mu_{\bigoplus_{i\sim\nu}\mathcal{P}_i\otimes\mathcal{Q}}(A * B)
\end{aligned}$$

□

Lemma B.4 (Monotonicity of $\otimes$). *If $\mathcal{P} \preceq \mathcal{P}'$ and $\mathcal{Q} \preceq \mathcal{Q}'$, then $\mathcal{P} \otimes \mathcal{Q} \preceq \mathcal{P}' \otimes \mathcal{Q}'$.*

Proof. Let S, S', T, and T' be sets such that $\Omega_{\mathcal{P}} \subseteq S$, $\Omega_{\mathcal{P}'} \subseteq \mathsf{Mem}[S']$, $\Omega_{\mathcal{Q}} \subseteq \mathsf{Mem}[T]$, and $\Omega_{\mathcal{Q}'} \subseteq \mathsf{Mem}[T']$. First we show the condition on sample spaces:

$$\begin{aligned}
\Omega_{\mathcal{P}\otimes\mathcal{Q}} &= \Omega_{\mathcal{P}} * \Omega_{\mathcal{Q}} \\
&= \{\sigma \uplus \tau \mid \sigma \in \Omega_{\mathcal{P}}, \tau \in \Omega_{\mathcal{Q}}\} \\
&\subseteq \{\sigma \uplus \tau \mid \sigma \in \pi_S(\Omega_{\mathcal{P}'}), \tau \in \pi_T(\Omega_{\mathcal{Q}'})\} \\
&= \{\pi_S(\sigma) \uplus \pi_T(\tau) \mid \sigma \in \Omega_{\mathcal{P}'}, \tau \in \Omega_{\mathcal{Q}'}\} \\
&= \{\pi_{S\cup T}(\sigma \uplus \tau) \mid \sigma \in \Omega_{\mathcal{P}'}, \tau \in \Omega_{\mathcal{Q}'}\} = \pi_{S\cup T}(\Omega_{\mathcal{P}'\otimes\mathcal{Q}})
\end{aligned}$$

Now we show the condition on σ-algebras:

$$\mathcal{F}_{\mathcal{P}\otimes\mathcal{Q}} = \sigma(\{A * B \mid A \in \mathcal{F}_{\mathcal{P}}, B \in \mathcal{F}_{\mathcal{Q}}\})$$

Since σ-closure is monotonic:

$$\begin{aligned}
&\subseteq \sigma(\{A * B \mid A \in \pi_S(\mathcal{F}_{\mathcal{P}'}), B \in \pi_T(\mathcal{F}_{\mathcal{Q}'})\}) \\
&= \sigma(\{\pi_S(A) * \pi_T(B) \mid A \in \mathcal{F}_{\mathcal{P}'}, B \in \mathcal{F}_{\mathcal{Q}'}\}) \\
&= \sigma(\pi_{S\cup T}(\{A * B \mid A \in \mathcal{F}_{\mathcal{P}'}, B \in \mathcal{F}_{\mathcal{Q}'}\})) \\
&= \pi_{S\cup T}(\sigma(\{A * B \mid A \in \mathcal{F}_{\mathcal{P}'}, B \in \mathcal{F}_{\mathcal{Q}'}\})) \\
&= \pi_{S\cup T}(\mathcal{F}_{\mathcal{P}'\otimes\mathcal{Q}'})
\end{aligned}$$

Finally, by Lemma B.2, it suffices to show that the product measure is equal only for events of the form $A * B$ where $A \in \mathcal{F}_{\mathcal{P}}$ and $B \in \mathcal{F}_{\mathcal{Q}}$:

$$\mu_{\mathcal{P}\otimes\mathcal{Q}}(A * B) = \mu_{\mathcal{P}}(A) \cdot \mu_{\mathcal{Q}}(B)$$

Since $\mathcal{P} \preceq \mathcal{P}'$ and $\mathcal{Q} \preceq \mathcal{Q}'$:

$$\begin{aligned}
&= \mu_{\mathcal{P}'}\left(\bigcup\{A' \in \mathcal{F}_{\mathcal{P}'} \mid \pi_S(A') = A\}\right) \cdot \mu_{\mathcal{Q}'}\left(\bigcup\{B' \in \mathcal{F}_{\mathcal{Q}'} \mid \pi_T(B') = B\}\right) \\
&= \mu_{\mathcal{P}'\otimes\mathcal{Q}'}\left(\left(\bigcup\{A' \in \mathcal{F}_{\mathcal{P}'} \mid \pi_S(A') = A\}\right) * \left(\bigcup\{B' \in \mathcal{F}_{\mathcal{Q}'} \mid \pi_T(B') = B\}\right)\right)
\end{aligned}$$

Let $A'' = \bigcup\{A' \in \mathcal{F}_{\mathcal{P}'} \mid \pi_S(A') = A\}$ and $B'' = \bigcup\{B' \in \mathcal{F}_{\mathcal{Q}'} \mid \pi_T(B') = B\}$. Clearly $A'' \in \mathcal{F}_{\mathcal{P}'}$ and $B'' \in \mathcal{F}_{\mathcal{Q}'}$ since both sets are closed under countable unions. Therefore:

$$= \mu_{\mathcal{P}'\otimes\mathcal{Q}'}\left(A'' * B''\right)$$

We show that the sets above and below are equal by showing the inclusion in both directions. The forward inclusion is trivial; $A'' * B''$ is clearly an element of $\mathcal{F}_{\mathcal{P}'\otimes\mathcal{Q}'}$ and has the property $\pi_{S\cup T}(A'' * B'') = \pi_S(A'') * \pi_T(B'') = A * B$. For the reverse inclusion, take some element of the set below, which must have the form $\sigma \uplus \tau$ where $\sigma \in \Omega_{\mathcal{P}'}$ and $\tau \in \Omega_{\mathcal{Q}'}$ and $\sigma \uplus \tau \in E$ for some $E \in \mathcal{F}_{\mathcal{P}'\otimes\mathcal{Q}'}$ such that $\pi_{S\cup T}(E) = A * B$. By Lemma B.1, $E = \biguplus_{(E_1,E_2)\in S} E_1 * E_2$ for some $S \subseteq \mathsf{ev}(\mathcal{P}') \times \mathsf{ev}(\mathcal{Q}')$, so $\sigma \in E_1$ and $\tau \in E_2$ for some $(E_1, E_2) \in S$. Since $\pi_{S\cup T}(E) = A * B$, then $\pi_S(E) = \biguplus_{(E_1,-)\in S} E_1 = A$ and $\pi_T(E) = \biguplus_{(-,E_2)\in S} E_2 = B$, therefore $\sigma \in E_1 \subseteq \pi_S(E) \subseteq A''$ and $\tau \in E_2 \subseteq \pi_T(E) \subseteq B''$, so $\sigma \uplus \tau \in A'' * B''$.

$$= \mu_{\mathcal{P}'\otimes\mathcal{Q}'}\left(\bigcup\{E \in \mathcal{F}_{\mathcal{P}'\otimes\mathcal{Q}'} \mid \pi_{S\cup T}(E) = A * B\}\right)$$

□

Lemma B.5 (Monotonicity of $\oplus$). *If $\mathcal{P}_v \preceq \mathcal{P}'_v$ for each $v \in \mathsf{supp}(\nu)$, then $\bigoplus_{v\sim\nu} \mathcal{P}_v \preceq \bigoplus_{v\sim\nu} \mathcal{P}'_v$.*

Proof. Let $\mathcal{P} = \bigoplus_{v\sim\nu} \mathcal{P}_v$ and $\mathcal{P}' = \bigoplus_{v\sim\nu} \mathcal{P}'_v$, so we need to prove that $\mathcal{P} \preceq \mathcal{P}'$. Also, let S and S' be sets such that $\Omega_{\mathcal{P}_v} \subseteq \mathsf{Mem}[S]$ and $\Omega_{\mathcal{P}'_v} \subseteq \mathsf{Mem}[S']$ for all v. We first establish the required property on the sample space:

$$\Omega_{\mathcal{P}} = \bigcup_{v\in\mathsf{supp}(\nu)} \Omega_{\mathcal{P}_v} \subseteq \bigcup_{v\in\mathsf{supp}(\nu)} \pi_S(\Omega_{\mathcal{P}'_v}) = \pi_S(\Omega_{\mathcal{P}'})$$

Now, we establish the property on σ-algebras.

$$\begin{aligned}
\mathcal{F}_{\mathcal{P}} &= \{A \mid A \subseteq \Omega_{\mathcal{P}}, \forall v.\, A \cap \Omega_{\mathcal{P}_v} \in \mathcal{F}_{\mathcal{P}_v}\} \\
&\subseteq \{A \mid A \subseteq \pi_S(\Omega_{\mathcal{P}'}), \forall v.\, A \cap \pi_S(\Omega_{\mathcal{P}'_v}) \in \{\pi_S(B) \mid B \in \mathcal{F}_{\mathcal{P}'_v}\}\} \\
&= \{\pi_S(A) \mid A \subseteq \Omega_{\mathcal{P}'}, \forall v.\, A \cap \Omega_{\mathcal{P}'_v} \in \mathcal{F}_{\mathcal{P}'_v}\} \\
&= \{\pi_S(A) \mid A \in \mathcal{F}_{\mathcal{P}'}\}
\end{aligned}$$

Finally, we establish the condition on probability measures:

$$\begin{aligned}
\mu_{\mathcal{P}}(A) &= \sum_{v\in\mathsf{supp}(\nu)} \nu(v) \cdot \mu_{\mathcal{P}_v}(A \cap \Omega_{\mathcal{P}_v}) \\
&= \sum_{v\in\mathsf{supp}(\nu)} \nu(v) \cdot \mu_{\mathcal{P}'_v}\left(\bigcup\{B \in \mathcal{F}_{\mathcal{P}'_v} \mid \pi_S(B) = A \cap \Omega_{\mathcal{P}_v}\}\right)
\end{aligned}$$

Since $A \subseteq \Omega_{\mathcal{P}} \subseteq \pi_S(\Omega_{\mathcal{P}'})$, then, we can move the intersection out of the set limits:

$$\begin{aligned}
&= \sum_{v\in\mathsf{supp}(\nu)} \nu(v) \cdot \mu_{\mathcal{P}'_v}\left(\left(\bigcup\{B \in \mathcal{F}_{\mathcal{P}'} \mid \pi_S(B) = A\}\right) \cap \Omega_{\mathcal{P}'_v}\right) \\
&= \mu_{\mathcal{P}'}\left(\bigcup\{B \in \mathcal{F}_{\mathcal{P}'} \mid \pi_S(B) = A\}\right)
\end{aligned}$$

□

Lemma B.6. *For any complete probability spaces $\mathcal{P}$, $(\mathcal{P}_i)_{i\in I}$, and $\mu \in \mathcal{D}(I)$ such that $\Omega_{\mathcal{P}} \subseteq \mathsf{Mem}[S]$, each $\Omega_{\mathcal{P}_i} \subseteq \mathsf{Mem}[T]$, and $\bigcup_{i\in I} \Omega_{\mathcal{P}_i} = \mathsf{Mem}[T]$, if $\mathcal{P} \preceq \mathsf{ext}(\mathcal{P}_i)$ for all $i \in \mathsf{supp}(\mu)$, then $\mathcal{P} \preceq \bigoplus_{i\sim\mu} \mathcal{P}_i$*

Proof. Let $\mathcal{Q} = \bigoplus_{i\sim\mu} \mathcal{P}_i$. Since $\mathcal{P} \preceq \mathcal{P}_i$, then it must be that $S \subseteq T$. The property on sample spaces is simple:

$$\Omega_{\mathcal{P}} = \bigcup_{i\in\mathsf{supp}(\mu)} \Omega_{\mathcal{P}} \subseteq \bigcup_{i\in\mathsf{supp}(\mu)} \pi_S(\Omega_{\mathsf{ext}(\mathcal{P}_i)}) = \bigcup_{i\in\mathsf{supp}(\mu)} \pi_S(\mathsf{Mem}[T]) = \pi_S\left(\bigcup_{i\in\mathsf{supp}(\mu)} \mathsf{Mem}[T]\right) = \pi_S(\Omega_{\mathcal{Q}})$$

Next, we verify the property on σ-algebras.

$$\begin{aligned}\mathcal{F}_{\mathcal{P}} &= \bigcap_{i\in I} \mathcal{F}_{\mathcal{P}} \\ &\subseteq \bigcap_{i\in I} \{\pi_S(A) \mid A \in \mathcal{F}_{\mathsf{ext}(\mathcal{P}_i)}\} \\ &= \{\pi_S(A) \mid \forall i \in I.\, A \in \mathcal{F}_{\mathsf{ext}(\mathcal{P}_i)}\}\end{aligned}$$

Note that the completion adds information about events outside of $\Omega_{\mathcal{P}_i}$, so if A is in all of the $\mathcal{F}_{\mathsf{ext}(\mathcal{P}_i)}$ sets, then its projection into each $\Omega_{\mathcal{P}_i}$ must be in $\mathcal{F}_{\mathcal{P}_i}$.

$$\begin{aligned}&= \{\pi_S(A) \mid A \subseteq \Omega_{\mathcal{Q}}, \forall i.\, A \cap \Omega_{\mathcal{P}_i} \in \mathcal{F}_{\mathcal{P}_i}\} \\ &= \{\pi_S(A) \mid A \in \mathcal{F}_{\mathcal{Q}}\}\end{aligned}$$

Finally, we show the property on probability measures.

$$\begin{aligned}\mu_{\mathcal{P}}(A) &= \sum_{i\in\mathsf{supp}(\mu)} \mu(i) \cdot \mu_{\mathcal{P}}(A) \\ &= \sum_{i\in\mathsf{supp}(\mu)} \mu(i) \cdot \mu_{\mathsf{ext}(\mathcal{P}_i)}\left(\bigcup \{B \in \mathcal{F}_{\mathsf{ext}(\mathcal{P}_i)} \mid \pi_S(B) = A\}\right)\end{aligned}$$

The completion assigns zero probability to events outside of $\Omega_{\mathcal{P}_i}$, so removing the completion and projecting into $\Omega_{\mathcal{P}_i}$ will yield the same value.

$$\begin{aligned}&= \sum_{i\in\mathsf{supp}(\mu)} \mu(i) \cdot \mu_{\mathcal{P}_i}\left(\left(\bigcup \{B \in \mathcal{F}_{\mathcal{Q}} \mid \pi_S(B) = A\}\right) \cap \Omega_{\mathcal{P}_i}\right) \\ &= \mu_{\mathcal{Q}}\left(\bigcup \{B \in \mathcal{F}_{\mathcal{Q}} \mid \pi_S(B) = A\}\right)\end{aligned}$$

□

Lemma B.7. *If $\mathcal{P} \preceq \mathsf{ext}(\mathcal{P}_i)$ for all $i \in \mathsf{supp}(\mu)$, then $\mathcal{P} \preceq \bigoplus_{i\sim\mu} \mathcal{P}_i$.*

Proof. Let S be the set such that $\Omega_{\mathcal{P}} = \mathsf{Mem}[S]$, T be such that $\mathsf{Mem}[T] = \bigcup_{i\in\mathsf{supp}(\mu)} \Omega_{\mathcal{P}_i}$, and $\mathcal{Q} = \preceq \bigoplus_{i\sim\mu} \mathcal{P}_i$. The condition on sample spaces is simple, since $\Omega_{\mathcal{P}} = \mathsf{Mem}[S]$ and $\Omega_{\mathcal{Q}} = \mathsf{Mem}[T]$, and clearly $\mathsf{Mem}[S] = \pi_S(\mathsf{Mem}[T])$. For the condition on σ-algebras, we have:

$$\begin{aligned}\mathcal{F}_{\mathcal{P}} &= \bigcap_{i\in\mathsf{supp}(\mu)} \mathcal{F}_{\mathcal{P}} \\ &\subseteq \bigcap_{i\in\mathsf{supp}(\mu)} \{\pi_S(A) \mid A \in \mathcal{F}_{\mathsf{ext}(\mathcal{P}_i)}\} \\ &= \bigcap_{i\in\mathsf{supp}(\mu)} \{\pi_S(A \cup B) \mid A \in \mathcal{F}_{\mathcal{P}_i}, B \subseteq \mathsf{Mem}[T] \setminus \Omega_{\mathcal{P}_i}\}\end{aligned}$$

Since for each i in the intersection above, we can measure every sample outside of $\Omega_{\mathcal{P}_i}$, then after taking the intersection we are only able to measure samples from $\Omega_{\mathcal{P}_i}$ according to the information provided by $\mathcal{F}_{\mathcal{P}_i}$, which provides the *least* information about those samples.

$$\begin{aligned}&= \{\pi_S(A) \mid A \subseteq \mathsf{Mem}[T], \forall i.\, A \cap \Omega_{\mathcal{P}_i} \in \mathcal{F}_{\mathcal{P}_i}\} \\ &= \{\pi_S(A) \mid A \in \mathcal{F}_{\mathcal{Q}}\}\end{aligned}$$

Now, we show the condition on probability measures:

$$\mu_{\mathcal{P}}(A) = \sum_{i\in\mathsf{supp}(\mu)} \mu(i) \cdot \mu_{\mathcal{P}}(A)$$

$$= \sum_{i \in \mathrm{supp}(\mu)} \mu(i) \cdot \mu_{\mathrm{ext}(\mathcal{P}_i)} \left(\bigcup \{ B \in \mathcal{F}_{\mathrm{ext}(\mathcal{P}_i)} \mid \pi_S(B) = A \} \right)$$
$$= \sum_{i \in \mathrm{supp}(\mu)} \mu(i) \cdot \mu_{\mathcal{P}_i} \left(\left(\bigcup \{ B \in \mathcal{F}_Q \mid \pi_S(B) = A \} \right) \cap \Omega_{\mathcal{P}_i} \right)$$
$$= \mu_Q \left(\bigcup \{ B \in \mathcal{F}_Q \mid \pi_S(B) = A \} \right)$$

□

B.2 The Convex Powerset

Lemma B.8. *For any finite index set I, $f \colon X \to C(Y)$, and $(S_i)_{i \in I} \in C(X)^I$:*

$$f^\dagger(\mathop{\&}_{i \in I} S_i) = \mathop{\&}_{i \in I} f^\dagger(S_i)$$

Proof. Let $g \colon I \to C(X)$ be defined as $g(i) \triangleq S_i$. Now, observe that:

$$\mathop{\&}_{i \in I} S_i = \left\{ \sum_{i \in \mathrm{supp}(\mu)} \mid \mu \in \mathcal{D}(I), \forall i.\ v_i \in g(i) \right\} = g^\dagger(\mathcal{D}(I))$$

So, we get:

$$\begin{aligned} f^\dagger(\mathop{\&}_{i \in I} S_i) &= f^\dagger(g^\dagger(\mathcal{D}(I))) \\ &= (f^\dagger \circ g)^\dagger(\mathcal{D}(I)) \\ &= \left\{ \sum_{i \in \mathrm{supp}(\mu)} \mu(i) \cdot v_i \;\middle|\; \mu \in \mathcal{D}(I), \forall i.\ v_i \in f^\dagger(g(i)) \right\} \\ &= \mathop{\&}_{i \in I} f^\dagger(S_i) \end{aligned}$$

□

For any $S \in C(\mathsf{Mem}[V])$ and $A \subseteq \mathsf{Mem}[V]$, let:

$$\mathsf{minProb}(S, A) \triangleq \inf_{\mu \in S} \mu(A)$$

Since $S \in C(\mathsf{Mem}[V])$, it is a closed subset of $\mathcal{D}(\mathsf{Mem}[V]_\perp)$, and so there must be a $\mu \in S$ such that $\mu(A) = \mathsf{minProb}(S, A)$, therefore $\mathsf{minProb}(S, A) = \min_{\mu \in S} \mu(A)$.

Lemma B.9 (Monotonicity of minProb). *If $S \sqsubseteq_C T$, then:*

$$\mathsf{minProb}(S, A) \leq \mathsf{minProb}(T, A)$$

Proof. By definition $S \sqsubseteq_C T$ iff $S \supseteq T$. So, we get:

$$\mathsf{minProb}(S, A) = \inf_{\mu \in S} \mu(A) = \min(\inf_{\mu \in T} \mu(A), \inf_{\mu \in S \setminus T} \mu(A)) \leq \inf_{\mu \in T} \mu(A) = \mathsf{minProb}(T, A)$$

□

Lemma B.10. *For any directed set $D \subseteq C(\mathsf{Mem}[V])$:*

$$\sup_{S \in D} \mathsf{minProb}(S, A) \leq \mathsf{minProb}(\sup D, A)$$

Proof. Since $S \sqsubseteq_C \sup D$ for all $S \in D$, then by Lemma B.9 we know that $\mathsf{minProb}(S, A) \leq \mathsf{minProb}(\sup D, A)$. Therefore, since the supremum is the least upper bound, it must be that $\sup_{S \in D} \mathsf{minProb}(S, A) \leq \mathsf{minProb}(\sup D, A)$. □

Lemma B.11 (Scott Continuity of minProb). *For any directed set $D \subseteq C(\mathsf{Mem}[S])$ such that $\mu(A) = p$ for all $\mu \in \sup D$:*

$$\sup_{S \in D} \mathsf{minProb}(S, A) = \mathsf{minProb}(\sup D, A)$$

Proof. By **?**, Corollary 3, we know that any chain-continuous function on a DCPO is Scott continuous, therefore it will suffice to show that minProb is chain continuous. Let $(S_\delta)_{\delta<\zeta}$ be a transfinite chain, so that $S_\delta \in C(\mathsf{Mem}[V])$ for all $\delta < \zeta$, and $S_\delta \sqsubseteq_C S_{\delta'}$ for all $\delta < \delta'$. We will now prove that:

$$\sup_{\delta<\zeta} \mathsf{minProb}(S_\delta, A) = \mathsf{minProb}\left(\sup_{\delta<\zeta} S_\delta, A\right)$$

From Lemma B.10, we already know that $\sup_{S\in D} \mathsf{minProb}(S, A) \leq \mathsf{minProb}(\sup D, A)$. We will now also show that $\sup_{S\in D} \mathsf{minProb}(S, A) < \mathsf{minProb}(\sup D, A)$ is a contradiction, and therefore the two quantities are equal. Let $p = \mathsf{minProb}(\sup_{\delta<\zeta} S_\delta, A)$, and suppose for the sake of contradiction that:

$$\sup_{\delta<\zeta} \mathsf{minProb}(S_\delta, A) < \mathsf{minProb}(\sup_{\delta<\zeta} S_\delta, A)$$

So, there is some $\varepsilon > 0$ such that $\sup_{\delta<\zeta} \mathsf{minProb}(S_\delta, A) = p - \varepsilon$.

Let $T_\delta = \{\mu \in S_\delta \mid \mu(A) \leq p - \varepsilon\}$ for all $\delta < \zeta$. If there exists a δ such that $T_\delta = \emptyset$, then $\sup_{\delta'<\zeta} \mathsf{minProb}(S_{\delta'}, A) \geq \mathsf{minProb}(S_\delta, A) > p - \varepsilon$, which is a contradiction, therefore $T_\delta \neq \emptyset$ for all $\delta < \zeta$. Further, by Lemma B.4.3 of McIver and Morgan [2005], the T_δ sets are closed, therefore their intersection must be nonempty by the finite intersection property. That means that there is some $\mu \in \bigcap_{\delta<\zeta} T_\delta \subseteq \bigcap_{\delta<\zeta} S_\delta = \sup_{\delta<\zeta} S_\delta$ such that $\mu(A) \leq p-\varepsilon$, but this is a contradiction too since we know that $\mathsf{minProb}(\sup_{\delta<\zeta} S_\delta, A) = p$. Therefore, it cannot be that $\sup_{\delta<\zeta} \mathsf{minProb}(S_\delta, A) < \mathsf{minProb}(\sup_{\delta<\zeta} S_\delta, A)$, and instead $\sup_{\delta<\zeta} \mathsf{minProb}(S_\delta, A) = \mathsf{minProb}(\sup_{\delta<\zeta} S_\delta, A)$.

□

Lemma B.12.

$$\mathsf{minProb}(S \,\&\, T, A) = \min(\mathsf{minProb}(S, A), \mathsf{minProb}(T, A))$$

Proof.

$$\begin{aligned}
\mathsf{minProb}(S \,\&\, T, A) &= \mathsf{minProb}(\{\mu \oplus_p \nu \mid \mu \in S, \nu \in T, p \in [0,1]\}, A) \\
&= \inf_{\mu\in S} \inf_{\nu\in T} \inf_{p\in[0,1]} p \cdot \mu(A) + (1-p) \cdot \nu(A) \\
&= \inf_{p\in[0,1]} p \cdot \left(\inf_{\mu\in S} \mu(A)\right) + (1-p) \cdot \left(\inf_{\nu\in T} \nu(A)\right) \\
&= \inf_{p\in[0,1]} p \cdot \mathsf{minProb}(S, A) + (1-p) \cdot \mathsf{minProb}(T, A)
\end{aligned}$$

Now, there are three cases, if $\mathsf{minProb}(S, A) < \mathsf{minProb}(T, A)$, then the infimum occurs when $p = 1$. If instead $\mathsf{minProb}(S, A) > \mathsf{minProb}(T, A)$, then the infimum occurs when $p = 0$. If $\mathsf{minProb}(S, A) = \mathsf{minProb}(T, A)$, then the expression is equal for all p. So, the infimum always occurs at one of the extremes ($p = 0$ or $p = 1$), and therefore:

$$= \min(\mathsf{minProb}(S, A), \mathsf{minProb}(T, A))$$

□

Lemma B.13. *For any* $f\colon \mathsf{Mem}[U] \to C(\mathsf{Mem}[U])$, $S \in C(\mathsf{Mem}[U])$, *and* $A \subseteq \mathsf{Mem}[U]$:

$$\mathsf{minProb}(f^\dagger(S), A) = \inf_{\mu\in S} \sum_{\sigma\in\mathsf{supp}(\mu)\cap\mathsf{Mem}[U]} \mu(\sigma) \cdot \mathsf{minProb}(f(\sigma), A)$$

Proof.

$$\begin{aligned}
\mathsf{minProb}(f^\dagger(S), A) &= \mathsf{minProb}\left(\left\{ \sum_{\sigma\in\mathsf{supp}(\mu)} \mu(\sigma) \cdot \nu_\sigma \;\middle|\; \mu \in S, \forall \sigma \in \mathsf{supp}(\mu).\ \nu_\sigma \in f_\perp(\sigma) \right\}, A\right) \\
&= \inf_{\mu\in S} \inf_{\nu_\sigma\in f_\perp(\sigma), \forall\sigma\in\mathsf{supp}(\mu)} \sum_{\sigma\in\mathsf{supp}(\mu)} \mu(\sigma) \cdot \nu_\sigma(A) \\
&= \inf_{\mu\in S} \sum_{\sigma\in\mathsf{supp}(\mu)} \mu(\sigma) \cdot \inf_{\nu_\sigma\in f_\perp(\sigma)} \nu_\sigma(A) \\
&= \inf_{\mu\in S} \sum_{\sigma\in\mathsf{supp}(\mu)} \mu(\sigma) \cdot \mathsf{minProb}(f_\perp(\sigma), A) \\
&= \inf_{\mu\in S} \sum_{\sigma\in\mathsf{supp}(\mu)\cap\mathsf{Mem}[U]} \mu(\sigma) \cdot \mathsf{minProb}(f_\perp(\sigma), A) + \mu(\perp) \cdot \mathsf{minProb}(f_\perp(\perp), A) \\
&= \inf_{\mu\in S} \sum_{\sigma\in\mathsf{supp}(\mu)\cap\mathsf{Mem}[U]} \mu(\sigma) \cdot \mathsf{minProb}(f(\sigma), A) + 0
\end{aligned}$$

$$= \inf_{\mu \in S} \sum_{\sigma \in \mathsf{supp}(\mu) \cap \mathsf{Mem}[U]} \mu(\sigma) \cdot \mathsf{minProb}(f(\sigma), A)$$

□

C Concurrency Lemmas

C.1 Invariant Sensitive Execution

Lemma C.1 (Check Monotonicity). *For any $U, V, W \subseteq \mathsf{Var}$ such that $U \cap V = \emptyset$, $\mathcal{I} \subseteq \mathsf{Mem}[U]$, $\mathcal{J} \subseteq \mathsf{Mem}[V]$, and $U \cup V \subseteq W$:*

$$\mathsf{check}^{\mathcal{I} * \mathcal{J}} \sqsubseteq_C^{\bullet} \mathsf{check}^{\mathcal{I}}$$

Proof. Take any $\sigma \in \mathsf{Mem}[W]$. If $\pi_U(\sigma) \notin \mathcal{I}$ or $\pi_V(\sigma) \notin \mathcal{J}$, then we get:

$$\mathsf{check}^{\mathcal{I} * \mathcal{J}}(\sigma) = \bot_C \sqsubseteq_C \mathsf{check}^{\mathcal{I}}(\sigma)$$

So, we are done. If not, then $\pi_U(\sigma) \in \mathcal{I}$ and $\pi_V(\sigma) \in \mathcal{J}$, so we get:

$$\mathsf{check}^{\mathcal{I} * \mathcal{J}}(\sigma) = \eta(\sigma) = \mathsf{check}^{\mathcal{I}}(\sigma)$$

□

Lemma C.2 (Action Monotonicity). *For any $U, V, W \subseteq \mathsf{Var}$ such that $U \cap V = \emptyset$, $\mathcal{I} \subseteq \mathsf{Mem}[U]$, $\mathcal{J} \subseteq \mathsf{Mem}[V]$, and $U \cup V \subseteq W$:*

$$[\![a]\!]_{\mathsf{Act}}^{\mathcal{I} * \mathcal{J}} \sqsubseteq_C^{\bullet} [\![a]\!]^{\mathcal{I}}$$

Proof. Let $W' = W \setminus (U \cup V)$. Take any $\sigma \in \mathsf{Mem}[W]$. if $\pi_V(\sigma) \notin \mathcal{J}$ or $\pi_U(\sigma) \notin \mathcal{I}$, then:

$$[\![a]\!]_{\mathsf{Act}}^{\mathcal{I} * \mathcal{J}}(\sigma) = \bot_C \sqsubseteq_C [\![a]\!]^{\mathcal{I}}(\sigma)$$

So, we are done. If instead $\pi_V(\sigma) \in \mathcal{J}$ and $\pi_U(\sigma) \in \mathcal{I}$, then we get:

$$\begin{aligned}
[\![a]\!]_{\mathsf{Act}}^{\mathcal{I} * \mathcal{J}}(\sigma) &= (\mathsf{check}^{\mathcal{I} * \mathcal{J}})^{\dagger}\left([\![a]\!]_{\mathsf{Act}}^{\dagger}\left((\mathsf{replace}^{\mathcal{I} * \mathcal{J}})^{\dagger}(\mathsf{check}^{\mathcal{I} * \mathcal{J}}(\sigma))\right)\right) \\
&= (\mathsf{check}^{\mathcal{I} * \mathcal{J}})^{\dagger}\left([\![a]\!]_{\mathsf{Act}}^{\dagger}\left(\mathsf{replace}^{\mathcal{I} * \mathcal{J}}(\sigma)\right)\right) \\
&= (\mathsf{check}^{\mathcal{I} * \mathcal{J}})^{\dagger}\left(\mathop{\&}_{\tau \in \mathcal{I} * \mathcal{J}} [\![a]\!]_{\mathsf{Act}}(\pi_{W'}(\sigma) \uplus \tau)\right)
\end{aligned}$$

Since $\pi_V(\sigma) \in \mathcal{J}$ and $\sqsubseteq_C$ is $\supseteq$.

$$\sqsubseteq_C (\mathsf{check}^{\mathcal{I} * \mathcal{J}})^{\dagger}\left(\mathop{\&}_{\tau \in \mathcal{I}} [\![a]\!]_{\mathsf{Act}}(\pi_{W' \cup V}(\sigma) \uplus \tau)\right)$$

By Lemma C.1 and monotonicity of Kleisli extension.

$$\begin{aligned}
&\sqsubseteq_C (\mathsf{check}^{\mathcal{I}})^{\dagger}\left(\mathop{\&}_{\tau \in \mathcal{I}} [\![a]\!]_{\mathsf{Act}}(\pi_{W' \cup V}(\sigma) \uplus \tau)\right) \\
&= (\mathsf{check}^{\mathcal{I}})^{\dagger}\left([\![a]\!]_{\mathsf{Act}}^{\dagger}\left(\mathsf{replace}^{\mathcal{I}}(\sigma)\right)\right)
\end{aligned}$$

Since $\pi_U(\sigma) \in \mathcal{I}$

$$\begin{aligned}
&= (\mathsf{check}^{\mathcal{I}})^{\dagger}\left([\![a]\!]_{\mathsf{Act}}^{\dagger}\left((\mathsf{replace}^{\mathcal{I}})^{\dagger}(\mathsf{check}^{\mathcal{I}}(\sigma))\right)\right) \\
&= [\![a]\!]_{\mathsf{Act}}^{\mathcal{I}}(\sigma)
\end{aligned}$$

□

Lemma 5.3 (Invariant Monotonicity). *For any $U, V, W \subseteq \mathsf{Var}$ and $\sigma \in \mathsf{Mem}[W]$ such that $U \cap V = \emptyset$, $\mathcal{I} \subseteq \mathsf{Mem}[U]$, $\mathcal{J} \subseteq \mathsf{Mem}[V]$, and $U \cup V \subseteq W$:*

$$\mathcal{L}^{\mathcal{I} * \mathcal{J}}(\boldsymbol{\alpha})(\sigma) \sqsubseteq_C \mathcal{L}^{\mathcal{I}}(\boldsymbol{\alpha})(\sigma)$$

Proof. For any pomset $\boldsymbol{\alpha}$, let $\boldsymbol{\alpha}^{\mathcal{I}}$ be the pomset obtained by replacing each action a with the tuple $\langle a, \mathcal{I} \rangle$. The evaluation function for these actions and order is as follows:

$$[\![\langle a, \mathcal{I} \rangle]\!]_{\mathsf{Act}}(\sigma) \triangleq [\![a]\!]_{\mathsf{Act}}^{\mathcal{I}}(\sigma) \qquad \langle a, \mathcal{I} \rangle \sqsubseteq_{\mathsf{Act}} \langle a', \mathcal{J} \rangle \quad \text{iff} \quad a = a' \quad \text{and} \quad \exists \mathcal{J}'.\ \mathcal{I} = \mathcal{J} * \mathcal{J}'$$

So, clearly $\boldsymbol{\alpha}^{\mathcal{I}*\mathcal{J}} \sqsubseteq_{pom} \boldsymbol{\alpha}^{\mathcal{I}}$, and $\mathcal{L}^{\mathcal{I}}(\boldsymbol{\alpha}) = \mathcal{L}(\boldsymbol{\alpha}^{\mathcal{I}})$. Note that $\sqsubseteq_{\mathsf{Act}}$ is not pointed or finitely proceeded, but this doesn't matter; as we will see in the following proof, we only need this order for monotonicity, not for the extension lemma [Zilberstein et al. 2025a, Lemma 3.8]. We now complete the proof as follows:

$$\begin{aligned}\mathcal{L}^{\mathcal{I}*\mathcal{J}}(\boldsymbol{\alpha}) &= \sup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \mathcal{L}^{\mathcal{I}*\mathcal{J}}_{\mathsf{fin}}(\boldsymbol{\alpha}') \\ &= \sup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \mathcal{L}_{\mathsf{fin}}(\boldsymbol{\alpha}'^{\mathcal{I}*\mathcal{J}})\end{aligned}$$

By Lemma H.3 of Zilberstein et al. [2025a] and Lemma C.2.

$$\begin{aligned}&\sqsubseteq_C \sup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \mathcal{L}_{\mathsf{fin}}(\boldsymbol{\alpha}'^{\mathcal{I}}) \\ &= \sup_{\boldsymbol{\alpha}' \ll \boldsymbol{\alpha}} \mathcal{L}^{\mathcal{I}}_{\mathsf{fin}}(\boldsymbol{\alpha}') = \mathcal{L}^{\mathcal{I}}(\boldsymbol{\alpha})\end{aligned}$$

□

C.2 Parallel Composition

Lemma C.3. *For any pairwise disjoint $U_1, U_2, V \subseteq \mathsf{Var}$, α and β such that $\mathsf{vars}_{\mathsf{Act}}(\alpha) \subseteq U_1 \cup V$, $\mathsf{vars}_{\mathsf{Test}}(\alpha) \subseteq U_1$, $\mathsf{vars}_{\mathsf{Act}}(\beta) \subseteq U_2 \cup V$, and $\mathsf{vars}_{\mathsf{Test}}(\beta) \subseteq U_2$; $\mathcal{I} \subseteq \mathsf{Mem}[V]$, $S \subseteq N_{\alpha \|_x \beta}$, $\psi_1, \psi_2 \in$ form, $\sigma_1 \in \mathsf{Mem}[U_1]$, $\sigma_2 \in \mathsf{Mem}[U_2]$, $\tau \in \mathcal{I}$, $A \subseteq \mathsf{Mem}[U_1]$, and $B \subseteq \mathsf{Mem}[U_2]$:*

$$\begin{aligned}&\mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha \|_x \beta, \psi_1 \wedge \psi_2, S)(\sigma_1 \uplus \sigma_2 \uplus \tau), A * B * \mathcal{I}) \\ &= \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha, \psi_1, S \cap N_\alpha)(\sigma_1 \uplus \tau), A * \mathcal{I}) \cdot \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\beta, \psi_2, S \cap N_\beta)(\sigma_2 \uplus \tau), B * \mathcal{I})\end{aligned}$$

Proof. The proof is by induction on the size of $\mathsf{next}^\star(\alpha \|_x \beta, \psi_1 \wedge \psi_2, S)$. If the set is empty, then we have:

$$\begin{aligned}&\mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha \|_x \beta, \psi_1 \wedge \psi_2, S)(\sigma_2 \uplus_2 \uplus \tau), A * B * \mathcal{I}) \\ &= \mathsf{minProb}(\eta(\sigma_1 \uplus \sigma_2 \uplus \tau), A * B * \mathcal{I}) \\ &= \sum_{\sigma_1' \in A} \sum_{\sigma_2' \in B} \sum_{\tau' \in \mathcal{I}} \delta_{\sigma_1 \uplus \sigma_2 \uplus \tau}(\sigma_1' \uplus \sigma_2' \uplus \tau')\end{aligned}$$

Since we already know that $\tau \in \mathcal{I}$:

$$\begin{aligned}&= \sum_{\sigma_1' \in A} \sum_{\sigma_2' \in B} \sum_{\tau' \in \mathcal{I}} \delta_{\sigma_1}(\sigma_1') \cdot \delta_{\sigma_2}(\sigma_2') \cdot \delta_\tau(\tau') \\ &= \left(\sum_{\sigma_1' \in A} \delta_{\sigma_1}(\sigma_1')\right) \cdot \left(\sum_{\sigma_2' \in B} \delta_{\sigma_2}(\sigma_2')\right) \\ &= \left(\sum_{\sigma' \in A} \sum_{\tau' \in \mathcal{I}} \delta_{\sigma_1 \uplus \tau}(\sigma_1' \uplus \tau')\right) \cdot \left(\sum_{\tau' \in B} \sum_{\tau' \in \mathcal{I}} \delta_{\sigma_2 \uplus \tau}(\sigma_2' \uplus \tau')\right) \\ &= \mathsf{minProb}(\eta(\sigma_1 \uplus \tau), A * \mathcal{I}) \cdot \mathsf{minProb}(\eta(\sigma_2 \uplus \tau), B * \mathcal{I}) \\ &= \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha, \psi_1, S \cap N_\alpha)(\sigma_1 \uplus \tau), A * \mathcal{I}) \cdot \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\beta, \psi_2, S \cap N_\beta)(\sigma_2 \uplus \tau, B * \mathcal{I})\end{aligned}$$

Now suppose that the set of next elements is not empty. If $\mathsf{next}(\alpha \|_x \beta, \psi_1 \wedge \psi_2, S) = \{x\}$, then we know that $\lambda_{\alpha \|_x \beta}(x) = \bullet$, and so:

$$\begin{aligned}&\mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha \|_x \beta, \psi_1 \wedge \psi_2, S)(\sigma_1 \uplus \sigma_2 \uplus \tau), A * B * \mathcal{I}) \\ &= \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha \|_x \beta, \psi_1 \wedge \psi_2, S \cup \{x\})(\sigma_1 \uplus \sigma_2 \uplus \tau), A * B * \mathcal{I})\end{aligned}$$

By the induction hypothesis:

$$= \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha, \psi_1, S \cap N_\alpha)(\sigma_1 \uplus \tau), A * \mathcal{I}) \cdot \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\beta, \psi_2, S \cap N_\beta)(\sigma_2 \uplus \tau), B * \mathcal{I})$$

If not, then x has already been scheduled and so $\mathsf{next}(\alpha \|_x \beta, \psi_1 \wedge \psi_2, S) = \mathsf{next}(\alpha, \psi_1, S \cap N_\alpha) \cup \mathsf{next}(\beta, \psi_2, S \cap N_\beta)$. Take any $y \in N_\alpha$, we will now show that:

$$\begin{aligned}&\mathsf{minProb}\left(\mathcal{L}^{\mathcal{I}}_{node}(\alpha \|_x \beta, \psi_1 \wedge \psi_2, S, y)(\sigma_1 \uplus \sigma_2 \uplus \tau), A * B * \mathcal{I}\right) \\ &= \mathsf{minProb}\left(\mathcal{L}^{\mathcal{I}}_{node}(\alpha, \psi_1, S \cap N_\alpha, y)(\sigma_1 \uplus \tau), A * \mathcal{I}\right) \cdot \mathsf{minProb}\left(\mathcal{L}^{\mathcal{I}}_{lpo}(\beta, \psi_2, S \cap N_\beta)(\sigma_2 \uplus \tau), B * \mathcal{I}\right)\end{aligned}$$

There are four cases:

(1) $\lambda_{\alpha\|_x\beta}(y) = \lambda_\alpha(y) \in \mathsf{Act}$. Let $a = \lambda_{\alpha\|_x\beta}(y)$, so:

$$\begin{aligned}
&\mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{node}(\alpha \|_x \beta, \psi_1 \wedge \psi_2, S, y)(\sigma_1 \uplus \sigma_2 \uplus \tau), A * B * \mathcal{I}) \\
&= \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha \|_x \beta, \psi_1 \wedge \psi_2, S \cup \{y\})^\dagger([\![a]\!]^{\mathcal{I}}_{\mathsf{Act}}(\sigma_1 \uplus \sigma_2 \uplus \tau)), A * B * \mathcal{I})
\end{aligned}$$

By Lemma B.13.

$$= \inf_{\mu \in [\![a]\!]^{\mathcal{I}}_{\mathsf{Act}}(\sigma_1 \uplus \sigma_2 \uplus \tau)} \sum_{\rho \in \mathsf{supp}(\mu) \cap \mathsf{Mem}[U_1 \cup V]} \mu(\rho) \cdot \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha \|_x \beta, \psi_1 \wedge \psi_2, S \cup \{y\})(\rho), A * B * \mathcal{I})$$

Since $\mathsf{vars}(\alpha) \cap \mathsf{dom}(\sigma_2) = \emptyset$.

$$= \inf_{\mu \in [\![a]\!]^{\mathcal{I}}_{\mathsf{Act}}(\sigma_1 \uplus \tau)} \sum_{\rho \in \mathsf{supp}(\mu) \cap \mathsf{Mem}[U_1 \cup V]} \mu(\rho) \cdot \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha \|_x \beta, \psi_1 \wedge \psi_2, S \cup \{y\})(\pi_{U_1}(\rho) \uplus \sigma_2 \uplus \pi_V(\rho)), A * B * \mathcal{I})$$

Note that by the definition of invariant sensitive execution, either $\pi_V(\rho) \subseteq \mathcal{I}$, or $[\![a]\!]^{\mathcal{I}}_{\mathsf{Act}}(\sigma_1 \uplus \tau) = \bot_C$. In the former case, we can apply the induction hypothesis. In the latter case, the entire expression must be zero, and therefore so is $\mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{node}(\alpha, \psi_1, S, y)(\sigma_1 \uplus \tau), A * \mathcal{I})$, so the claim holds trivially. So, we presume the former is true and apply the induction hypothesis to get:

$$= \inf_{\mu \in [\![a]\!]^{\mathcal{I}}_{\mathsf{Act}}(\sigma_1 \uplus \tau)} \sum_{\rho \in \mathsf{supp}(\mu)} \mu(\rho) \cdot \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha, \psi_1, S \cap N_\alpha \cup \{y\})(\rho), A * \mathcal{I}) \cdot \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\beta, \psi_2, S \cap N_\beta)(\sigma_2 * \pi_V(\rho)), B * \mathcal{I})$$

Note that $\mathcal{L}^{\mathcal{I}}_{lpo}(\beta, \psi_2, S \cap N_\beta)(\sigma_2 \uplus \tau) = \mathcal{L}^{\mathcal{I}}_{lpo}(\beta, \psi_2, S \cap N_\beta)(\sigma_2 \uplus \tau')$ for any $\tau' \in \mathcal{I}$, since the invariant sensitive execution reassigns the V variables at each step and tests only depend on local state.

$$\begin{aligned}
&= \left(\inf_{\mu \in [\![a]\!]^{\mathcal{I}}_{\mathsf{Act}}(\sigma_1 \uplus \tau)} \sum_{\rho \in \mathsf{supp}(\mu)} \mu(\rho) \cdot \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha, \psi_1, S \cap N_\alpha \cup \{y\})(\rho), A * \mathcal{I}) \right) \cdot \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\beta, \psi_2, S \cap N_\beta)(\sigma_2 \uplus \tau, B * \mathcal{I}) \\
&= \left(\mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha, \psi_1, S \cap N_\alpha \cup \{y\})^\dagger([\![a]\!]^{\mathcal{I}}_{\mathsf{Act}}(\sigma_1 \uplus \tau)), A * \mathcal{I}) \right) \cdot \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\beta, \psi_2, S \cap N_\beta)(\sigma_2 \uplus \tau), B * \mathcal{I}) \\
&= \mathsf{minProb}(\mathcal{L}_{node}(\alpha, \psi_1, S \cap N_\alpha, y)(\sigma_1 \uplus \tau), A * \mathcal{I}) \cdot \mathsf{minProb}(\mathcal{L}_{lpo}(\beta, \psi, S \cap N_\beta)(\sigma_2 \uplus \tau), B * \mathcal{I})
\end{aligned}$$

(2) $\lambda_{\alpha\|_x\beta}(y) = \lambda_\alpha(y) \in \mathsf{Test}$. Let $b = \lambda_{\alpha\|_x\beta}(y)$, and we therefore have:

$$\begin{aligned}
&\mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{node}(\alpha \|_x \beta, \psi_1 \wedge \psi_2, S, y)(\sigma_1 \uplus \sigma_2 \uplus \tau), A * B * \mathcal{I}) \\
&= \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha \|_x \beta, \psi_1 \wedge \psi_2 \wedge (\!|y = [\![b]\!]_{\mathsf{Test}}(\sigma_1 \uplus \sigma_2 \uplus \tau)|\!), S \cup \{y\})(\sigma_1 \uplus \sigma_2 \uplus \tau), A * B * \mathcal{I})
\end{aligned}$$

By the induction hypothesis, and the fact that $\mathsf{vars}(b) \subseteq \mathsf{vars}_{\mathsf{Test}}(\alpha) \subseteq U_1$ (therefore it does not depend on $\sigma_2 \in \mathsf{Mem}[U_2]$ and $\tau \in \mathsf{Mem}[V]$.

$$\begin{aligned}
&= \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha, \psi_1 \wedge (\!|y = [\![b]\!](\sigma_1)|\!), S \cap N_\alpha \cup \{y\})(\sigma_1 \uplus \tau), A * \mathcal{I}) \\
&\quad \cdot \mathsf{minProb}(\mathcal{L}_{lpo}(\beta, \psi_2, S \cap N_\beta)(\sigma_2 \uplus \tau), B * \mathcal{I}) \\
&= \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{node}(\alpha, \psi_1, S \cap N_\alpha, y)(\sigma_1 \uplus \sigma_2 \uplus \tau), A * \mathcal{I}) \cdot \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\beta, \psi_2, S \cap N_\beta)(\sigma_1 \uplus \sigma_2 \uplus \tau), B * \mathcal{I})
\end{aligned}$$

(3) $\lambda_{\alpha\|_x\beta}(y) = \lambda_\alpha(y) = \bot$.

$$\begin{aligned}
&\mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{node}(\alpha \|_x \beta, \psi_1 \wedge \psi_2, S, y)(\sigma_1 \uplus \sigma_2 \uplus \tau), A * B * \mathcal{I}) \\
&= \mathsf{minProb}(\bot_C, A * B) \\
&= 0 \\
&= 0 \cdot \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\beta, \psi_2, S \cap N_\beta)(\sigma_2 \uplus \tau), B * \mathcal{I}) \\
&= \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{node}(\alpha, \psi_1, S \cap N_\alpha, y)(\sigma_1 \uplus \tau), A * \mathcal{I}) \cdot \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\beta, \psi_2, S \cap N_\beta)(\sigma_2 \uplus \tau), B * \mathcal{I})
\end{aligned}$$

(4) $\lambda_{\alpha\|_x\beta}(y) = \lambda_\alpha(y) = \bullet$.

$$\begin{aligned}
&\mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{node}(\alpha \|_x \beta, \psi_1 \wedge \psi_2, S, y)(\sigma_1 \uplus \sigma_2 \uplus \tau), A * B * \mathcal{I}) \\
&= \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha \|_x \beta, \psi_1 \wedge \psi_2, S \cup \{y\})(\sigma_1 \uplus \sigma_2 \uplus \tau), A * B * \mathcal{I})
\end{aligned}$$

By the induction hypothesis.

$$\begin{aligned}
&= \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha, \psi_1, S \cap N_\alpha \cup \{y\})(\sigma_1 \uplus \tau), A * \mathcal{I}) \cdot \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\beta, \psi_2, S \cap N_\beta)(\sigma_2 \uplus \tau), B * \mathcal{I}) \\
&= \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{node}(\alpha, \psi_1, S \cap N_\alpha, y)(\sigma_1 \uplus \tau), A * \mathcal{I}) \cdot \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\beta, \psi_2, S \cap N_\beta)(\sigma_1 \uplus \tau), B * \mathcal{I})
\end{aligned}$$

By a nearly identical argument, we also get that for any $y \in N_\beta$:

$$\begin{aligned}&\mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{node}(\alpha \parallel_x \beta, \psi_1 \wedge \psi_2, S, y)(\sigma_1 \uplus \sigma_2 \uplus \tau), A * B * \mathcal{I})\\ &= \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha, \psi_1, S \cap N_\alpha)(\sigma_1 \uplus \tau), A) \cdot \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{node}(\beta, \psi_2, S \cap N_\beta, y)(\sigma_2 \uplus \tau), B * \mathcal{I})\end{aligned}$$

We now complete the proof as follows:

$$\begin{aligned}&\mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha \parallel_x \beta, \psi_1 \wedge \psi_2, S)(\sigma_1 \uplus \sigma_2 \uplus \tau), A * B * \mathcal{I})\\
&= \mathsf{minProb}\left(\bigwedge_{y \in \mathsf{next}(\alpha \parallel_x \beta, \psi_1 \wedge \psi_2, S)} \mathcal{L}^{\mathcal{I}}_{node}(\alpha \parallel_x \beta, \psi_1 \wedge \psi_2, S, y)(\sigma_1 \uplus \sigma_2 \uplus \tau), A * B * \mathcal{I}\right)\end{aligned}$$

By Lemma B.12

$$\begin{aligned}&= \min_{y \in \mathsf{next}(\alpha \parallel_x \beta, \psi_1 \wedge \psi_2, S)} \mathsf{minProb}\left(\mathcal{L}^{\mathcal{I}}_{node}(\alpha \parallel_x \beta, \psi_1 \wedge \psi_2, S, y)(\sigma_1 \uplus \sigma_2 \uplus \tau), A * B * \mathcal{I}\right)\\
&= \min\Big(\min_{y \in \mathsf{next}(\alpha, \psi_1, S \cap N_\alpha)} \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{node}(\alpha \parallel_x \beta, \psi_1 \wedge \psi_2, S, y)(\sigma_1 \uplus \sigma_2 \uplus \tau), A * B * \mathcal{I}),\\
&\qquad \min_{y \in \mathsf{next}(\beta, \psi_2, S \cap N_\beta)} \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{node}(\alpha \parallel_x \beta, \psi_1 \wedge \psi_2, S, y)(\sigma_1 \uplus \sigma_2 \uplus \tau), A * B * \mathcal{I})\Big)\\
&= \min\Big(\min_{y \in \mathsf{next}(\alpha, \psi_1, S \cap N_\alpha)} \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{node}(\alpha, \psi_1, S \cap N_\alpha, y)(\sigma_1 \uplus \tau), A * \mathcal{I}) \cdot \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\beta, \psi_2, S \cap N_\beta)(\sigma_2 \uplus \tau), B * \mathcal{I}),\\
&\qquad \min_{y \in \mathsf{next}(\beta, \psi_2, S \cap N_\beta)} \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha, \psi_1, S \cap N_\alpha)(\sigma_1 \uplus \tau), A * \mathcal{I}) \cdot \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{node}(\beta, \psi_2, S \cap N_\beta, y)(\sigma_2 \uplus \tau), B * \mathcal{I})\Big)\\
&= \min\Big(\Big(\min_{y \in \mathsf{next}(\alpha, \psi_1, S \cap N_\alpha)} \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{node}(\alpha, \psi_1, S \cap N_\alpha, y)(\sigma_1 \uplus \tau), A * \mathcal{I})\Big) \cdot \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\beta, \psi_2, S \cap N_\beta)(\sigma_2 \uplus \tau), B * \mathcal{I}),\\
&\qquad \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha, \psi_1, S \cap N_\alpha)(\sigma_1 \uplus \tau), A * \mathcal{I}) \cdot \Big(\min_{y \in \mathsf{next}(\beta, \psi_2, S \cap N_\beta)} \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{node}(\beta, \psi_2, S \cap N_\beta, y)(\sigma_2 \uplus \tau), B * \mathcal{I})\Big)\Big)\\
&= \min\Big(\mathsf{minProb}\left(\bigwedge_{y \in \mathsf{next}(\alpha, \psi_1, S \cap N_\alpha)} \mathcal{L}^{\mathcal{I}}_{node}(\alpha, \psi_1, S \cap N_\alpha, y)(\sigma_1 \uplus \tau), A * \mathcal{I}\right) \cdot \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\beta, \psi_2, S \cap N_\beta)(\sigma_2 \uplus \tau), B * \mathcal{I}),\\
&\qquad \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha, \psi_1, S \cap N_\alpha)(\sigma_1 \uplus \tau), A * \mathcal{I}) \cdot \mathsf{minProb}\left(\bigwedge_{y \in \mathsf{next}(\beta, \psi_2, S \cap N_\beta)} \mathcal{L}^{\mathcal{I}}_{node}(\beta, \psi_2, S \cap N_\beta, y)(\sigma_2 \uplus \tau), B * \mathcal{I}\right)\Big)\\
&= \min\Big(\mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha, \psi_1, S \cap N_\alpha)(\sigma_1 \uplus \tau), A * \mathcal{I}) \cdot \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\beta, \psi_2, S \cap N_\beta)(\sigma_2 \uplus \tau), B * \mathcal{I}),\\
&\qquad \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha, \psi_1, S \cap N_\alpha)(\sigma_1 \uplus \tau), A * \mathcal{I}) \cdot \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\beta, \psi_2, S \cap N_\beta)(\sigma_2 \uplus \tau), B * \mathcal{I})\Big)\\
&= \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha, \psi_1, S \cap N_\alpha)(\sigma_1 \uplus \tau), A * \mathcal{I}) \cdot \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}_{lpo}(\beta, \psi_2, S \cap N_\beta)(\sigma_2 \uplus \tau), B * \mathcal{I})\end{aligned}$$

□

Lemma C.4. *For any pairwise disjoint $U_1, U_2, V \subseteq \mathsf{Var}$ and $\boldsymbol{\alpha}, \boldsymbol{\beta} \in pom_{\mathsf{fin}}$ such that $\mathsf{vars}_{\mathsf{Act}}(\boldsymbol{\alpha}) \subseteq U_1 \cup V$, $\mathsf{vars}_{\mathsf{Test}}(\boldsymbol{\alpha}) \subseteq U_1$, $\mathsf{vars}_{\mathsf{Act}}(\boldsymbol{\beta}) \subseteq U_2 \cup V$, and $\mathsf{vars}_{\mathsf{Test}}(\boldsymbol{\beta}) \subseteq U_2$; $\mathcal{I} \subseteq \mathsf{Mem}[V]$, $\sigma_1 \in \mathsf{Mem}[U_1]$, $\sigma_2 \in \mathsf{Mem}[U_2]$, $\tau \in \mathcal{I}$, $A \subseteq \mathsf{Mem}[U_1]$, and $B \subseteq \mathsf{Mem}[U_2]$:*

$$\mathsf{minProb}\left(\mathcal{L}^{\mathcal{I}}_{\mathsf{fin}}(\boldsymbol{\alpha} \parallel \boldsymbol{\beta})(\sigma_1 \uplus \sigma_2 \uplus \tau), A * B * \mathcal{I}\right) = \mathsf{minProb}\left(\mathcal{L}^{\mathcal{I}}_{\mathsf{fin}}(\boldsymbol{\alpha})(\sigma_1 \uplus \tau), A * \mathcal{I}\right) \cdot \mathsf{minProb}\left(\mathcal{L}^{\mathcal{I}}_{\mathsf{fin}}(\boldsymbol{\beta})(\sigma_2 \uplus \tau), B * \mathcal{I}\right)$$

Proof. Fix any $\alpha \in \boldsymbol{\alpha}$, $\beta \in \boldsymbol{\beta}$, and $x \notin N_\alpha \cup N_\beta$. This give us $\boldsymbol{\alpha} \parallel \boldsymbol{\beta} = [\alpha \parallel_x \beta]$. So, we get:

$$\begin{aligned}&\mathsf{minProb}\left(\mathcal{L}^{\mathcal{I}}_{\mathsf{fin}}(\boldsymbol{\alpha} \parallel \boldsymbol{\beta})(\sigma_1 \uplus \sigma_2 \uplus \tau), A * B * \mathcal{I}\right)\\
&= \mathsf{minProb}\left(\mathcal{L}^{\mathcal{I}}_{\mathsf{fin}}([\alpha \parallel_x \beta])(\sigma_1 \uplus \sigma_2 \uplus \tau), A * B * \mathcal{I}\right)\\
&= \mathsf{minProb}\left(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha \parallel_x \beta, \mathsf{true}, \emptyset)(\sigma_1 \uplus \sigma_2 \uplus \tau), A * B * \mathcal{I}\right)\end{aligned}$$

By Lemma C.3.

$$\begin{aligned}&= \mathsf{minProb}\left(\mathcal{L}^{\mathcal{I}}_{lpo}(\alpha, \mathsf{true}, \emptyset)(\sigma_1 \uplus \tau), A * \mathcal{I}\right) \cdot \mathsf{minProb}\left(\mathcal{L}^{\mathcal{I}}_{lpo}(\beta, \mathsf{true}, \emptyset)(\sigma_2 \uplus \tau), B * \mathcal{I}\right)\\
&= \mathsf{minProb}\left(\mathcal{L}^{\mathcal{I}}_{\mathsf{fin}}([\alpha])(\sigma_1 \uplus \tau), A * \mathcal{I}\right) \cdot \mathsf{minProb}\left(\mathcal{L}^{\mathcal{I}}_{lpo}([\beta])(\sigma_2 \uplus \tau), B * \mathcal{I}\right)\end{aligned}$$

$$= \mathsf{minProb}\left(\mathcal{L}^{\mathcal{I}}_{\mathsf{fin}}(\boldsymbol{\alpha})(\sigma_1 \uplus \tau), A * \mathcal{I}\right) \cdot \mathsf{minProb}\left(\mathcal{L}^{\mathcal{I}}_{lpo}(\boldsymbol{\beta})(\sigma_2 \uplus \tau), B * \mathcal{I}\right)$$

□

Lemma C.5. *For any pairwise disjoint* $U_1, U_2, V \subseteq \mathsf{Var}$ *and* $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathit{pom}$ *such that* $\mathsf{vars}_{\mathsf{Act}}(\boldsymbol{\alpha}) \subseteq U_1 \cup V$, $\mathsf{vars}_{\mathsf{Test}}(\boldsymbol{\alpha}) \subseteq U_1$, $\mathsf{vars}_{\mathsf{Act}}(\boldsymbol{\beta}) \subseteq U_2 \cup V$, *and* $\mathsf{vars}_{\mathsf{Test}}(\boldsymbol{\beta}) \subseteq U_2$; $\mathcal{I} \subseteq \mathsf{Mem}[V]$, $\sigma_1 \in \mathsf{Mem}[U_1]$, $\sigma_2 \in \mathsf{Mem}[U_2]$, $\tau \in \mathcal{I}$, $A \subseteq \mathsf{Mem}[U_1]$, *and* $B \subseteq \mathsf{Mem}[U_2]$, *if there exists* p *and* q *such that:*

$$\forall \nu_1 \in \mathcal{L}^{\mathcal{I}}(\boldsymbol{\alpha})(\sigma_1 \uplus \tau).\ \nu_1(B_1 * \mathcal{I}) = p \qquad \textit{and} \qquad \forall \nu_2 \in \mathcal{L}^{\mathcal{I}}(\boldsymbol{\beta})(\sigma_2 \uplus \tau).\ \nu_2(B_2 * \mathcal{I}) = q$$

Then:

$$\forall \nu \in \mathcal{L}^{\mathcal{I}}(\boldsymbol{\alpha} \parallel \boldsymbol{\beta})(\sigma_1 \uplus \sigma_2 \uplus \tau).\ \nu(B_1 * B_2 * \mathcal{I}) \geq p \cdot q$$

Proof. Take any $\nu \in \mathcal{L}^{\mathcal{I}}(\boldsymbol{\alpha} \parallel \boldsymbol{\beta})(\sigma_1 \uplus \sigma_2 \uplus \tau)$. We have:

$$\begin{aligned}
\nu(B_1 * B_2 * \mathcal{I}) &\geq \mathsf{minProb}(\mathcal{L}^{\mathcal{I}}(\boldsymbol{\alpha} \parallel \boldsymbol{\beta})(\sigma_1 \uplus \sigma_2 \uplus \tau), B_1 * B_2 * \mathcal{I}) \\
&= \mathsf{minProb}\left(\sup_{\boldsymbol{\gamma} \ll \boldsymbol{\alpha} \parallel \boldsymbol{\beta}} \mathcal{L}^{\mathcal{I}}_{\mathsf{fin}}(\boldsymbol{\gamma})(\sigma_1 \uplus \sigma_2 \uplus \tau), B_1 * B_2 * \mathcal{I}\right)
\end{aligned}$$

By Lemma B.11.

$$\begin{aligned}
&= \sup_{\boldsymbol{\gamma} \ll \boldsymbol{\alpha} \parallel \boldsymbol{\beta}} \mathsf{minProb}\left(\mathcal{L}^{\mathcal{I}}_{\mathsf{fin}}(\boldsymbol{\gamma})(\sigma_1 \uplus \sigma_2 \uplus \tau), B_1 * B_2 * \mathcal{I}\right) \\
&= \sup_{\boldsymbol{\alpha}' \ll_1 \boldsymbol{\alpha}} \sup_{\boldsymbol{\beta}' \ll_1 \boldsymbol{\beta}} \mathsf{minProb}\left(\mathcal{L}^{\mathcal{I}}_{\mathsf{fin}}(\boldsymbol{\alpha}' \parallel \boldsymbol{\beta}')(\sigma_1 \uplus \sigma_2 \uplus \tau), B_1 * B_2 * \mathcal{I}\right)
\end{aligned}$$

By Lemma C.4.

$$\begin{aligned}
&= \sup_{\boldsymbol{\alpha}' \ll_1 \boldsymbol{\alpha}} \sup_{\boldsymbol{\beta}' \ll_1 \boldsymbol{\beta}} \mathsf{minProb}\left(\mathcal{L}^{\mathcal{I}}_{\mathsf{fin}}(\boldsymbol{\alpha}')(\sigma_1 \uplus \tau), B_1 * \mathcal{I}\right) \cdot \mathsf{minProb}\left(\mathcal{L}^{\mathcal{I}}_{\mathsf{fin}}(\boldsymbol{\beta}')(\sigma_2 \uplus \tau), B_2 * \mathcal{I}\right) \\
&= \left(\sup_{\boldsymbol{\alpha}' \ll_1 \boldsymbol{\alpha}} \mathsf{minProb}\left(\mathcal{L}^{\mathcal{I}}_{\mathsf{fin}}(\boldsymbol{\alpha}')(\sigma_1 \uplus \tau), B_1 * \mathcal{I}\right)\right) \cdot \left(\sup_{\boldsymbol{\beta}' \ll_1 \boldsymbol{\beta}} \mathsf{minProb}\left(\mathcal{L}^{\mathcal{I}}_{\mathsf{fin}}(\boldsymbol{\beta}')(\sigma_2 \uplus \tau), B_2 * \mathcal{I}\right)\right)
\end{aligned}$$

By Lemma B.11.

$$\begin{aligned}
&= \mathsf{minProb}\left(\mathcal{L}^{\mathcal{I}}(\boldsymbol{\alpha})(\sigma_1 \uplus \tau), B_1 * \mathcal{I}\right) \cdot \mathsf{minProb}\left(\mathcal{L}^{\mathcal{I}}(\boldsymbol{\beta})(\sigma_2 \uplus \tau), B_2 * \mathcal{I}\right) \\
&= p \cdot q
\end{aligned}$$

□

Lemma C.6. *For any pairwise disjoint* $U_1, U_2, V \subseteq \mathsf{Var}$ *and* $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2 \in \mathit{pom}$ *such that* $\mathsf{vars}_{\mathsf{Act}}(\boldsymbol{\alpha}_k) \subseteq U_k \cup V$ *and* $\mathsf{vars}_{\mathsf{Test}}(\boldsymbol{\alpha}_k) \subseteq U_k$; $\mathcal{I} \subseteq \mathsf{Mem}[V]$, *let* $\mathcal{P}_{\mathcal{I}}$ *be the trivial probability space where* $\mu_{\mathcal{P}_{\mathcal{I}}}(\mathcal{I}) = 1$. *For all* $k \in \{1, 2\}$, $\mathcal{P}_k$, $\mathcal{Q}_k$, *and* $\mu \in \mathcal{D}(\mathsf{Mem}[U_1 \cup U_2 \cup V])$ *such that* $\mathcal{P}_1 \otimes \mathcal{P}_2 \otimes \mathcal{P}_{\mathcal{I}} \preceq \mu$, *if:*

$$\forall \mu_k.\ \mathcal{P}_k \otimes \mathcal{P}_{\mathcal{I}} \preceq \mu_k \implies \forall \nu_k \in \mathcal{L}^{\mathcal{I}}(\boldsymbol{\alpha}_k)^{\dagger}(\mu_k).\ \mathcal{Q}_k \otimes \mathcal{P}_{\mathcal{I}} \preceq \nu_k$$

Then:

$$\forall \nu \in \mathcal{L}^{\mathcal{I}}(\boldsymbol{\alpha}_1 \parallel \boldsymbol{\alpha}_2)^{\dagger}(\mu).\ \mathcal{Q}_1 \otimes \mathcal{Q}_2 \otimes \mathcal{P}_{\mathcal{I}} \preceq \nu$$

Proof. Let $(A_{1,i})_{i \in I_1}$ and $(A_{2,i})_{i \in I_2}$ be the most precise, disjoint measurable events from $\mathsf{ev}(\mathcal{P}_1)$ and $\mathsf{ev}(\mathcal{P}_2)$, respectively. Since $\mathcal{P}_1 \otimes \mathcal{P}_2 \otimes \mathcal{P}_{\mathcal{I}} \preceq \mu$, we know that for any $(i, j) \in I_1 \times I_2$:

$$\sum_{\sigma_1 \in A_{1,i}} \sum_{\sigma_2 \in A_{2,j}} \sum_{\tau \in \mathcal{I}} \mu(\sigma_1 \uplus \sigma_2 \uplus \tau) = \mu_{\mathcal{P}_1}(A_{1,i}) \cdot \mu_{\mathcal{P}_2}(A_{2,j}) = \mu_{\mathcal{P}_1 \otimes \mathcal{P}_{\mathcal{I}}}(A_{1,i} * \mathcal{I}) \cdot \mu_{\mathcal{P}_2 \otimes \mathcal{I}}(A_{2,j} * \mathcal{I}) \tag{5}$$

Also, for any μ_k such that $\mathcal{P}_k \otimes \mathcal{P}_{\mathcal{I}} \preceq \mu_k$, we know that any $\nu_k \in \mathcal{L}^{\mathcal{I}}(\alpha_k)^{\dagger}(\mu_k)$ has the form:

$$\nu_k = \sum_{\sigma \in \mathsf{supp}(\mu_k)} \mu_k(\sigma) \cdot \nu_\sigma$$

Where each $\nu_\sigma \in \mathcal{L}^{\mathcal{I}}(\boldsymbol{\alpha}_k)(\sigma)$. Since $\mathcal{Q}_k \otimes \mathcal{P}_{\mathcal{I}} \preceq \nu_k$, we know that for any $B \in \mathcal{F}_{\mathcal{Q}_k}$:

$$\mu_{\mathcal{Q}_k \otimes \mathcal{P}_{\mathcal{I}}}(B * \mathcal{I}) = \sum_{\tau \in B * \mathcal{I}} \nu_k(\tau) = \sum_{\sigma \in \mathsf{supp}(\mu_k)} \mu_k(\sigma) \cdot \sum_{\tau \in B * \mathcal{I}} \nu_\sigma(\tau)$$

Let $p_{B,\sigma} = \sum_{\tau \in B * \mathcal{I}} \nu_\sigma(\tau)$.

$$= \sum_{\sigma \in \mathsf{supp}(\mu_k)} \mu_k(\sigma) \cdot p_{B,\sigma}$$

We will now show that $\sum_{\tau \in B * \mathcal{I}} \xi(\tau) = p_{B,\sigma'}$ for any $\sigma' \in \mathrm{supp}(\mu_k)$ and $\xi \in \mathcal{L}^{\mathcal{I}}(\boldsymbol{\alpha}_k)(\sigma')$. Take any such ξ. By construction, $\mu_k(\sigma') \cdot \xi + \sum_{\sigma \in \mathrm{supp}(\mu_k) \setminus \{\sigma'\}} \mu_k(\sigma) \cdot \nu_\sigma \in \mathcal{L}^{\mathcal{I}}(\boldsymbol{\alpha}_k)^\dagger(\mu_k)$, therefore:

$$\mu_{Q_k \otimes \mathcal{P}_{\mathcal{I}}}(B * \mathcal{I}) = \sum_{\tau \in B * \mathcal{I}} \left(\mu_k(\sigma') \cdot \xi + \sum_{\sigma \in \mathrm{supp}(\mu_k) \setminus \{\sigma'\}} \mu_k(\sigma) \cdot \nu_\sigma \right)(\tau) = \sum_{\sigma \in \mathrm{supp}(\mu_k)} \mu_k(\sigma) \cdot p_{B,\sigma}$$

$$\mu_k(\sigma') \cdot \sum_{\tau \in B * \mathcal{I}} \xi(\tau) + \cancel{\sum_{\sigma \in \mathrm{supp}(\mu_k) \setminus \{\sigma'\}} \mu_k(\sigma) \cdot p_{B,\sigma}} = \mu_k(\sigma') \cdot p_{B,\sigma'} + \cancel{\sum_{\sigma \in \mathrm{supp}(\mu_k) \setminus \{\sigma'\}} \mu_k(\sigma) \cdot p_{B,\sigma}}$$

$$\cancel{\mu_k(\sigma')} \cdot \sum_{\tau \in B * \mathcal{I}} \xi(\tau) = \cancel{\mu_k(\sigma')} \cdot p_{B,\sigma'}$$

$$\sum_{\tau \in B * \mathcal{I}} \xi(\tau) = p_{B,\sigma'}$$

Now, let μ_k be constructed by fixing a single state $\sigma_i \in A_{k,i} * \mathcal{I}$ for each $i \in I_k$ and setting $\mu_k(\sigma_i) = \mu_{\mathcal{P}_k \otimes \mathcal{P}_{\mathcal{I}}}(A_{k,i})$, so clearly $\mathcal{P}_k \otimes \mathcal{P}_{\mathcal{I}} \preceq \mu_k$. Now let $p_{B,k,i} = p_{B,\sigma_i}$. Based on what we have just showed, this gives us:

$$\mu_{Q_k \otimes \mathcal{P}_{\mathcal{I}}}(B * \mathcal{I}) = \sum_{\sigma \in \mathrm{supp}(\mu_k)} \mu_k(\sigma) \cdot p_{B,\sigma} = \sum_{i \in I_k} \mu_k(\sigma_i) \cdot p_{B,\sigma_i} = \sum_{i \in I_k} \mu_{\mathcal{P}_k \otimes \mathcal{P}_{\mathcal{I}}}(A_{k,i} * \mathcal{I}) \cdot p_{B,k,i}$$

We will now show that $p_{B,\sigma'} = p_{B,k,j}$ for any $j \in I_k$ and $\sigma' \in A_{k,j} * \mathcal{I}$. Take any such σ', then we get:

$$\mu_{Q_k \otimes \mathcal{P}_{\mathcal{I}}}(B * \mathcal{I}) =$$

$$\mu_{\mathcal{P}_k \otimes \mathcal{P}_{\mathcal{I}}}(A_{k,j} * \mathcal{I}) \cdot p_{B,\sigma'} + \sum_{i \neq j} \mu_{\mathcal{P}_k \otimes \mathcal{P}_{\mathcal{I}}}(A_{k,i} * \mathcal{I}) \cdot p_{B,\sigma_i} = \sum_{i \in I_k} \mu_{\mathcal{P}_k \otimes \mathcal{P}_{\mathcal{I}}}(A_{k,i} * \mathcal{I}) \cdot p_{B,k,i}$$

$$\mu_{\mathcal{P}_k \otimes \mathcal{P}_{\mathcal{I}}}(A_{k,j} * \mathcal{I}) \cdot p_{B,\sigma'} + \cancel{\sum_{i \neq j} \mu_{\mathcal{P}_k \otimes \mathcal{P}_{\mathcal{I}}}(A_{k,i} * \mathcal{I}) \cdot p_{B,k,i}} = \mu_{\mathcal{P}_k \otimes \mathcal{P}_{\mathcal{I}}}(A_{k,j} * \mathcal{I}) \cdot p_{B,k,j} + \cancel{\sum_{i \neq j} \mu_{\mathcal{P}_k \otimes \mathcal{P}_{\mathcal{I}}}(A_{k,i} * \mathcal{I}) \cdot p_{B,k,i}}$$

$$\cancel{\mu_{\mathcal{P}_k \otimes \mathcal{P}_{\mathcal{I}}}(A_{k,j} * \mathcal{I})} \cdot p_{B,\sigma'} = \cancel{\mu_{\mathcal{P}_k \otimes \mathcal{P}_{\mathcal{I}}}(A_{k,j} * \mathcal{I})} \cdot p_{B,k,j}$$

$$p_{B,\sigma'} = p_{B,k,j}$$

We have therefore shown that $\sum_{\tau \in B} \nu_k(\tau) = p_{B,i,k}$ for any $\nu_k \in \mathcal{L}^{\mathcal{I}}(\boldsymbol{\alpha}_k)(\sigma)$ where $B \in \mathcal{F}_{Q_k}$, $i \in I_k$, and $\sigma \in A_{k,i}$

Now take any $\nu \in \mathcal{L}^{\mathcal{I}}(\boldsymbol{\alpha}_1 \parallel \boldsymbol{\alpha}_2)^\dagger(\mu)$, which must have the form $\nu = \sum_{\sigma \in \mathrm{supp}(\mu)} \mu(\sigma) \cdot \nu_\sigma$ where $\nu_\sigma \in \mathcal{L}^{\mathcal{I}}(\boldsymbol{\alpha}_1 \parallel \boldsymbol{\alpha}_2)(\sigma)$ for each σ. Since Q_1 and Q_2 operate over different address spaces, clearly $Q_1 \otimes Q_2$ exists. It just remains to show that $Q_1 \otimes Q_2 \otimes \mathcal{P}_{\mathcal{I}} \preceq \nu$, which we do as follows. By Lemma B.2, it suffices to show that the probability measures agree on the product of disjoint partitions. Let $\{B_{k,j} \mid j \in J_k\} = \mathrm{ev}(Q_k)$ for $k \in \{1, 2\}$. For any $i \in J_1$ and $j \in J_2$, we have:

$$\nu(B_{1,i} * B_{2,j} * \mathcal{I}) = \sum_{\sigma \in \mathrm{supp}(\mu)} \mu(\sigma) \cdot \nu_\sigma(B_{1,i} * B_{2,j} * \mathcal{I})$$

Instead of summing over the support of σ, we can alternatively sum over the elements of the $A_{1,i'}$ and $A_{2,j'}$ sets.

$$= \sum_{i' \in I_1} \sum_{j' \in I_2} \sum_{\sigma_1 \in A_{1,i'}} \sum_{\sigma_2 \in A_{2,j'}} \sum_{\tau \in \mathcal{I}} \mu(\sigma_1 \uplus \sigma_2 \uplus \tau) \cdot \nu_\sigma(B_{1,i} * B_{2,j} * \mathcal{I})$$

We previously showed that $\nu_k(B_{k,\ell} * \mathcal{I}) = p_{B_{k,\ell},k,i'}$ (a constant) for any $\nu_k \in \mathcal{L}^{\mathcal{I}}(\boldsymbol{\alpha}_k)(\sigma)$ where $\sigma \in A_{k,i'}$. So, we can use Lemma C.5 to conclude that:

$$\geq \sum_{i' \in I_1} \sum_{j' \in I_2} \sum_{\sigma_1 \in A_{1,i'}} \sum_{\sigma_2 \in A_{2,j'}} \sum_{\tau \in \mathcal{I}} \mu(\sigma_1 \uplus \sigma_2 \uplus \tau) \cdot p_{B_{1,i},1,i'} \cdot p_{B_{2,j},2,j'}$$

$$= \sum_{i' \in I_1} \sum_{j' \in I_2} \mu(A_{1,i'} * A_{2,j'} * \mathcal{I}) \cdot p_{B_{1,i},1,i'} \cdot p_{B_{2,j},2,j'}$$

By Equation (5).

$$= \sum_{i' \in I_1} \sum_{j' \in I_2} \mu_{\mathcal{P}_1 \otimes \mathcal{P}_{\mathcal{I}}}(A_{1,i'} * \mathcal{I}) \cdot \mu_{\mathcal{P}_2 \otimes \mathcal{P}_{\mathcal{I}}}(A_{2,j'} * \mathcal{I}) \cdot p_{B_{1,i},1,i'} \cdot p_{B_{2,j},2,j'}$$

$$= \left(\sum_{i' \in I_1} \mu_{\mathcal{P}_1 \otimes \mathcal{P}_{\mathcal{I}}}(A_{1,i'} * \mathcal{I}) \cdot p_{B_{1,i},1,i'} \right) \cdot \left(\sum_{j' \in I_2} \mu_{\mathcal{P}_2 \otimes \mathcal{P}_{\mathcal{I}}}(A_{2,j'} * \mathcal{I}) \cdot p_{B_{2,j},2,j'} \right)$$

$$= \mu_{Q_1 \otimes \mathcal{P}_{\mathcal{I}}}(B_{1,i} * \mathcal{I}) \cdot \mu_{Q_2 \otimes \mathcal{P}_{\mathcal{I}}}(B_{2,j} * \mathcal{I})$$

$$= \mu_{Q_1}(B_{1,i}) \cdot \mu_{Q_2}(B_{2,j}) \cdot \mu_{\mathcal{P}_{\mathcal{I}}}(\mathcal{I})$$

Now, we have shown that $\nu(B_{1,i} * B_{2,j} * \mathcal{I}) \geq \mu_{Q_1 \otimes Q_2 \otimes \mathcal{P}_\mathcal{I}}(B_{1,i} * B_{2,j} * \mathcal{I})$ for all $i \in J_1$ and $j \in J_2$. Since $\sum_{i \in J_1} \sum_{j \in J_2} \mu_{Q_1 * Q_2}(B_{1,i} * B_{2,j}) = 1$, then $\nu(B_{1,i} * B_{2,j} * \mathcal{I})$ cannot be strictly greater then $\mu_{Q_1}(B_{1,i}) \cdot \mu_{Q_2}(B_{2,j})$ for any i or j, and therefore the quantities must be equal.

□

D Almost Sure Termination

For any $S \in C(\mathsf{Mem}[V])$, let $\mathsf{minterm}(S) = \mathsf{minProb}(S, \mathsf{Mem}[V])$, *i.e.,* it is the minimum probability that the program terminates. Also, let $\Psi_{\langle b,C,\mathcal{I}\rangle} : (\mathsf{Mem}[V] \to C(\mathsf{Mem}[V])) \to \mathsf{Mem}[V] \to C(\mathsf{Mem}[V])$ be defined as follows:

$$\Psi_{\langle b,C,\mathcal{I}\rangle}(f)(\sigma) \triangleq \begin{cases} f^\dagger(\mathcal{L}^\mathcal{I}(\llbracket C \rrbracket)(\sigma)) & \text{if } \llbracket b \rrbracket_{\mathsf{Test}}(\sigma) = \mathsf{true} \\ \eta(\sigma) & \text{if } \llbracket b \rrbracket_{\mathsf{Test}}(\sigma) = \mathsf{false} \end{cases}$$

By Zilberstein et al. [2025a, Lemma 5.2]:

$$\mathcal{L}^\mathcal{I}(\llbracket \textbf{while } b \textbf{ do } C \rrbracket) = \mathsf{lfp}\left(\Psi_{\langle b,C,\mathcal{I}\rangle}\right) = \sup_{n \in \mathbb{N}} \Psi^n_{\langle b,C,\mathcal{I}\rangle}(\bot^\bullet_C)$$

Now, for any test b and distribution μ we define a conditioning operator as follows:

$$(b?\mu)(\sigma) \triangleq \begin{cases} \frac{\mu(\sigma)}{\mu(b)} & \text{if } \llbracket b \rrbracket_{\mathsf{Test}}(\sigma) = \mathsf{true} \\ 0 & \text{if } \llbracket b \rrbracket_{\mathsf{Test}}(\sigma) = \mathsf{false} \end{cases} \qquad \text{where} \quad \mu(b) \triangleq \sum_{\sigma \in \mathsf{supp}(\mu) \mid \llbracket b \rrbracket_{\mathsf{Test}}(\sigma) = \mathsf{true}} \mu(\sigma)$$

Note that if $\mu(b) = 0$, then $b?\mu$ is not well-defined. In that case, we just let $b?\mu = \bot_\mathcal{D}$. It is also clearly true that $\mu = (b?\mu) \oplus_{\mu(b)} (\neg b?\mu)$ for any b and μ. In addition, we call $\langle \varphi, \psi \rangle$ an loop invariant pair for $\textbf{while } b \textbf{ do } C$ under the resource invariant I iff:

(1) $\varphi \Rrightarrow \lceil b \mapsto \mathsf{true} \rceil$
(2) $\psi \Rrightarrow \lceil b \mapsto \mathsf{false} \rceil$
(3) $I \vDash_\mathsf{w} \langle \varphi \rangle\ C\ \langle \varphi\ \&\ \psi \rangle$
(4) $\mathsf{precise}(\psi)$

Given these new definitions, we prove some partial correctness results, which show that the ψ (as defined above) holds on the terminating portion of the result of a while loop.

Lemma D.1. *Take any Γ, let $\mathcal{I} = (\!|I|\!)_\Gamma$, $\langle \varphi, \psi \rangle$ be an invariant pair for $\textbf{while } b \textbf{ do } C$ under I, and Q be the unique smallest probability space satisfying ψ (which exists since $\mathsf{precise}(\psi)$). For any $\mu, n \in \mathbb{N}$, and $A \in \mathcal{F}_Q$ such that $\Gamma, \mu \vDash \varphi * \lceil I \rceil$:*

$$\mathsf{minProb}\left(\Psi^n_{\langle b,C,\mathcal{I}\rangle}\left(\bot^\bullet_C\right)^\dagger(\mu), A * \mathcal{I}\right) = \mathsf{minterm}\left(\Psi^n_{\langle b,C,\mathcal{I}\rangle}\left(\bot^\bullet_C\right)^\dagger(\mu)\right) \cdot \mu_Q(A)$$

Proof. The proof is by induction on n. Suppose that $n = 0$, and so we have:

$$\begin{aligned} \mathsf{minProb}\left(\Psi^0_{\langle b,C,\mathcal{I}\rangle}\left(\bot^\bullet_C\right)^\dagger(\mu), A * \mathcal{I}\right) &= \mathsf{minProb}\left(\bot_C, A * \mathcal{I}\right) \\ &= 0 \\ &= \mathsf{minterm}\left(\bot_C\right) \cdot \mu_Q(A) \\ &= \mathsf{minterm}\left(\Psi^0_{\langle b,C,\mathcal{I}\rangle}\left(\bot^\bullet_C\right)^\dagger(\mu)\right) \cdot \mu_Q(A) \end{aligned}$$

Now suppose that $n = 1$, and so we have:

$$\begin{aligned} \mathsf{minProb}\left(\Psi^1_{\langle b,C,\mathcal{I}\rangle}\left(\bot^\bullet_C\right)^\dagger(\mu), A * \mathcal{I}\right) &= \mathsf{minProb}\left(\left(\bot^\bullet_C\right)^\dagger\left(\mathcal{L}^\mathcal{I}(\llbracket C \rrbracket)^\dagger(\mu)\right), A * \mathcal{I}\right) \\ &= \mathsf{minProb}\left(\bot_C, A * \mathcal{I}\right) \\ &= 0 \\ &= \mathsf{minterm}\left(\bot_C\right) \cdot \mu_Q(A) \\ &= \mathsf{minterm}\left(\Psi^1_{\langle b,C,\mathcal{I}\rangle}\left(\bot^\bullet_C\right)^\dagger(\mu)\right) \cdot \mu_Q(A) \end{aligned}$$

Now suppose that $n > 1$. Then, we have:

$$\begin{aligned} &\mathsf{minProb}\left(\Psi^n_{\langle b,C,\mathcal{I}\rangle}\left(\bot^\bullet_C\right)^\dagger(\mu), A * \mathcal{I}\right) \\ &= \mathsf{minProb}\left(\Psi^{n-1}_{\langle b,C,\mathcal{I}\rangle}\left(\bot^\bullet_C\right)^\dagger\left(\mathcal{L}^\mathcal{I}(\llbracket C \rrbracket)^\dagger(\mu)\right), A * \mathcal{I}\right) \end{aligned}$$

$$= \inf_{\nu \in \mathcal{L}^{\mathcal{I}}(\llbracket C \rrbracket)^\dagger(\mu)} \mathsf{minProb}\left(\Psi^{n-1}_{\langle b,C,\mathcal{I}\rangle}\left(\bot^\bullet_C\right)^\dagger(\nu), A * \mathcal{I}\right)$$

Since $\langle \varphi, \psi \rangle$ is an invariant pair and $\Gamma, \mu \vDash \varphi * \lceil I \rceil$, then every such ν above can be split into $\nu = b?\nu \oplus_{\nu(b)} \neg b?\nu$ such that $\Gamma, b?\nu \vDash \varphi * \lceil I \rceil$ and $\Gamma, \neg b?\nu \vDash \psi * \lceil I \rceil$.

$$= \inf_{\nu \in \mathcal{L}^{\mathcal{I}}(\llbracket C \rrbracket)^\dagger(\mu)} \mathsf{minProb}\left(\Psi^{n-1}_{\langle b,C,\mathcal{I}\rangle}\left(\bot^\bullet_C\right)^\dagger(b?\nu \oplus_{\nu(b)} \neg b?\nu), A * \mathcal{I}\right)$$

$$= \inf_{\nu \in \mathcal{L}^{\mathcal{I}}(\llbracket C \rrbracket)^\dagger(\mu)} \mathsf{minProb}\left(\Psi^{n-1}_{\langle b,C,\mathcal{I}\rangle}\left(\bot^\bullet_C\right)^\dagger(b?\nu) \oplus_{\nu(b)} \Psi^{n-1}_{\langle b,C,\mathcal{I}\rangle}\left(\bot^\bullet_C\right)^\dagger(\neg b?\nu), A * \mathcal{I}\right)$$

Since $n > 1$, then $\Psi^{n-1}_{\langle b,C,\mathcal{I}\rangle}(\bot^\bullet_C)^\dagger(\neg b?\nu) = \{\neg b?\nu\}$.

$$= \inf_{\nu \in \mathcal{L}^{\mathcal{I}}(\llbracket C \rrbracket)^\dagger(\mu)} \mathsf{minProb}\left(\Psi^{n-1}_{\langle b,C,\mathcal{I}\rangle}\left(\bot^\bullet_C\right)^\dagger(b?\nu) \oplus_{\nu(b)} \{\neg b?\nu\}, A * \mathcal{I}\right)$$

$$= \inf_{\nu \in \mathcal{L}^{\mathcal{I}}(\llbracket C \rrbracket)^\dagger(\mu)} \nu(b) \cdot \mathsf{minProb}\left(\Psi^{n-1}_{\langle b,C,\mathcal{I}\rangle}\left(\bot^\bullet_C\right)^\dagger(b?\nu), A * \mathcal{I}\right) + (1 - \nu(b)) \cdot (\neg b?\nu)(A * \mathcal{I})$$

Since $\Gamma, \neg b?\nu \vDash \psi * \lceil I \rceil$, then $(\neg b?\nu)(A * \mathcal{I}) = \mu_Q(A)$. Also using the induction hypothesis, we get:

$$= \inf_{\nu \in \mathcal{L}^{\mathcal{I}}(\llbracket C \rrbracket)^\dagger(\mu)} \nu(b) \cdot \mathsf{minterm}\left(\Psi^{n-1}_{\langle b,C,\mathcal{I}\rangle}\left(\bot^\bullet_C\right)^\dagger(b?\nu)\right) \cdot \mu_Q(A) + (1 - \nu(b)) \cdot \mu_Q(A)$$

$$= \left(\inf_{\nu \in \mathcal{L}^{\mathcal{I}}(\llbracket C \rrbracket)^\dagger(\mu)} \mathsf{minterm}\left(\Psi^{n-1}_{\langle b,C,\mathcal{I}\rangle}\left(\bot^\bullet_C\right)^\dagger(b?\nu) \oplus_{\nu(b)} \{\neg b?\nu\}\right)\right) \cdot \mu_Q(A)$$

$$= \left(\inf_{\nu \in \mathcal{L}^{\mathcal{I}}(\llbracket C \rrbracket)^\dagger(\mu)} \mathsf{minterm}\left(\Psi^{n-1}_{\langle b,C,\mathcal{I}\rangle}\left(\bot^\bullet_C\right)^\dagger(\nu)\right)\right) \cdot \mu_Q(A)$$

$$= \mathsf{minterm}\left(\Psi^{n}_{\langle b,C,\mathcal{I}\rangle}\left(\bot^\bullet_C\right)^\dagger(\mu)\right) \cdot \mu_Q(A)$$

□

Lemma D.2. *Take any Γ, let $\mathcal{I} = \langle\!|I|\!\rangle_\Gamma$, $\langle \varphi, \psi \rangle$ be an invariant pair for* **while** b **do** C *under I, and Q be the unique smallest probability space satisfying ψ (which exists since* $\mathsf{precise}(\psi)$*). For any μ such that $\Gamma, \mu \vDash \varphi * \lceil I \rceil$ and $A \in \mathcal{F}_Q$:*

$$\mathsf{minProb}\left(\mathcal{L}^{\mathcal{I}}\left(\llbracket \textbf{while } b \textbf{ do } C \rrbracket\right)^\dagger(\mu), A * \mathcal{I}\right) = \mathsf{minterm}\left(\mathcal{L}^{\mathcal{I}}\left(\llbracket \textbf{while } b \textbf{ do } C \rrbracket\right)^\dagger(\mu)\right) \cdot \mu_Q(A)$$

Proof. The proof proceeds as follows:

$$\mathsf{minProb}\left(\mathcal{L}^{\mathcal{I}}\left(\llbracket \textbf{while } b \textbf{ do } C \rrbracket\right)^\dagger(\mu), A * \mathcal{I}\right) = \mathsf{minProb}\left(\sup_{n \in \mathbb{N}} \Psi^{n}_{\langle b,C,\mathcal{I}\rangle}\left(\bot^\bullet_C\right)^\dagger(\mu), A * \mathcal{I}\right)$$

$$= \sup_{n \in \mathbb{N}} \mathsf{minProb}\left(\Psi^{n}_{\langle b,C,\mathcal{I}\rangle}\left(\bot^\bullet_C\right)^\dagger(\mu), A * \mathcal{I}\right)$$

By Lemma D.1.

$$= \sup_{n \in \mathbb{N}} \mathsf{minterm}\left(\Psi^{n}_{\langle b,C,\mathcal{I}\rangle}\left(\bot^\bullet_C\right)^\dagger(\mu)\right) \cdot \mu_Q(A)$$

$$= \mathsf{minterm}\left(\sup_{n \in \mathbb{N}} \Psi^{n}_{\langle b,C,\mathcal{I}\rangle}\left(\bot^\bullet_C\right)^\dagger(\mu)\right) \cdot \mu_Q(A)$$

$$= \mathsf{minterm}\left(\mathcal{L}^{\mathcal{I}}\left(\llbracket \textbf{while } b \textbf{ do } C \rrbracket\right)^\dagger(\mu)\right) \cdot \mu_Q(A)$$

□

Corollary D.3 (Almost Sure Termination). *Take any Γ and $0 < p \leq 1$, let $\mathcal{I} = \langle\!|I|\!\rangle_\Gamma$, and $\langle \varphi, \psi \rangle$ be an invariant pair for* **while** b **do** C *under I. If additionally* $\mathsf{minterm}(\mathcal{L}^{\mathcal{I}}(\llbracket \textbf{while } b \textbf{ do } C \rrbracket)^\dagger(\mu)) \geq p$ *for all μ such that $\Gamma, \mu \vDash \varphi * \lceil I \rceil$, then* $\mathsf{minterm}(\mathcal{L}^{\mathcal{I}}(\llbracket \textbf{while } b \textbf{ do } C \rrbracket)^\dagger(\mu)) = 1$.

Proof. Follows by an identical argument to Lemma D.3 of Zilberstein et al. [2025b]. □

E Logic and Rules

Lemma E.1 (Monotonicity). *If* $\Gamma, \mathcal{P} \vDash \varphi$ *and* $\mathcal{P} \preceq \mathcal{Q}$, *then* $\Gamma, \mathcal{Q} \vDash \varphi$.

Proof. By induction on the structure of φ.

- $\varphi = \top$. Since $\Gamma, \mathcal{Q} \vDash \top$ for all $\mathcal{Q}$, the claim holds trivially.
- $\varphi = \bot$. The premise is false, therefore this case is vacuous.
- $\varphi = \varphi_1 \wedge \varphi_2$. By the induction hypothesis, we know that $\Gamma, \mathcal{Q} \vDash \varphi_1$ and $\Gamma, \mathcal{Q} \vDash \varphi_2$, therefore $\Gamma, \mathcal{Q} \vDash \varphi_1 \wedge \varphi_2$.
- $\varphi = \varphi_1 \vee \varphi_2$. Without loss of generality, suppose that $\Gamma, \mathcal{P} \vDash \varphi_1$. By the induction hypothesis, we know that $\Gamma, \mathcal{Q} \vDash \varphi_1$, therefore by weakening $\Gamma, \mathcal{Q} \vDash \varphi_1 \vee \varphi_2$.
- $\varphi = \exists X.\psi$. We know that there is some $v \in \mathsf{Val}$ such that $\Gamma[X := v], \mathcal{P} \vDash \psi$. By the induction hypothesis, we get that $\Gamma[X := v], \mathcal{Q} \vDash \psi$. This implies that $\Gamma, \mathcal{Q} \vDash \exists X.\psi$.
- $\varphi = \bigoplus_{X \sim d(E)} \psi$. Immediate, since the semantics stipulates that $\mathcal{P}$ is greater than the direct sum, therefore $\mathcal{Q}$ is also greater than the direct sum.
- $\varphi = \&_{X \in E} \psi$. We know that $\Gamma, \mathcal{P} \vDash \bigoplus_{X \sim \mu}$ for some $\mu \in \mathcal{D}(\llbracket E \rrbracket_{\mathsf{LExp}}(\Gamma))$. By the previous case, we get that $\Gamma, \mathcal{Q} \vDash \bigoplus_{X \sim \mu} \psi$. This implies that $\Gamma, \mathcal{Q} \vDash \&_{X \in E} \psi$.
- $\varphi = \varphi_1 *_m \varphi_2$. Immediate, since we know that $\mathcal{P}_1 \diamond_m \mathcal{P}_2 \preceq \mathcal{P}$ such that $\Gamma, \mathcal{P}_i \vDash \varphi_i$ for each i, and therefore $\mathcal{P}_1 \diamond_m \mathcal{P}_2 \preceq \mathcal{Q}$ as well.
  the semantics stipulates that $\mathcal{P}$ is greater than the independent product, therefore $\mathcal{Q}$ is also greater than the independent product.
- $\varphi = \lceil P \rceil$. Let $\Omega_{\mathcal{P}} = \mathsf{Mem}[S]$ and $\Omega_{\mathcal{Q}} = \mathsf{Mem}[T]$, and note that $S \subseteq T$ since $\mathcal{P} \preceq \mathcal{Q}$. We know that $(\!|P|\!)^S_\Gamma \in \mathcal{F}_{\mathcal{P}}$ and $\mu_{\mathcal{P}}((\!|P|\!)^S_\Gamma) = 1$. We also know that $\mu_{\mathcal{P}}((\!|P|\!)^S_\Gamma) = \mu_{\mathcal{Q}}(\bigcup_{B|\pi_S(B)=(\!|P|\!)^S_\Gamma} B) = 1$. Note that by definition $\bigcup_{B|\pi_S(B)=(\!|P|\!)^S_\Gamma} B \subseteq (\!|P|\!)^T_\Gamma$. Since that set has probability 1 and $\mathcal{Q}$ is a complete probability space, then $(\!|P|\!)^T_\Gamma \in \mathcal{F}_{\mathcal{Q}}$, and also has probability 1.

□

E.1 Precise and Convex Assertions

Lemma E.2. $\mathsf{precise}(\lceil P \rceil)$

Proof. Take any Γ. If $(\!|P|\!)_\Gamma = \emptyset$, then P is unsatisfiable under Γ, so we are done. If not, then let $\Omega = \mathsf{Mem}[\mathsf{fv}(P)]$, $\mathcal{F} = \{A \subseteq \Omega \mid (\!|P|\!)_\Gamma \subseteq A\} \cup \{A \subseteq \Omega \mid A \cap (\!|P|\!)_\Gamma = \emptyset\}$ and:

$$\mu(A) = \begin{cases} 1 & \text{if } (\!|P|\!)_\Gamma \subseteq A \\ 0 & \text{otherwise} \end{cases}$$

It is relatively easy to see that μ is a probability measure since $(\!|P|\!)_\Gamma$ is the smallest measurable set with nonzero probability, and it has probability 1, so the countable additivity property holds. By definition, $\Gamma, \langle \Omega, \mathcal{F}, \mu \rangle \vDash \lceil P \rceil$. Clearly it is also minimal, since any other $\mathcal{P}$ such that $\Gamma, \mathcal{P} \vDash \lceil P \rceil$ must also include $\mathcal{F}$ as measurable sets by definition, and must assign probability 1 to the event $(\!|P|\!)_\Gamma$. □

Lemma E.3. *If* $\mathsf{precise}(\varphi, \psi)$, *then* $\mathsf{precise}(\varphi * \psi)$.

Proof. Take any Γ, if either φ or ψ is not satisfiable under Γ, then neither is $\varphi * \psi$, and then the claim holds vacuously. If both are satisfiable, then there are unique smallest $\mathcal{P}_1$ and $\mathcal{P}_2$ such that $\Gamma, \mathcal{P}_1 \vDash \varphi$ and $\Gamma, \mathcal{P}_2 \vDash \psi$. Clearly, this means that $\Gamma, \mathcal{P}_1 \otimes \mathcal{P}_2 \vDash \varphi * \psi$. We now argue that $\mathcal{P}_1 \otimes \mathcal{P}_2$ is minimal. Take any $\mathcal{Q}$ such that $\Gamma, \mathcal{Q} \vDash \varphi * \psi$. This means that there are $\mathcal{Q}_1 \otimes \mathcal{Q}_2 \preceq \mathcal{Q}$ such that $\Gamma, \mathcal{Q}_1 \vDash \varphi$ and $\Gamma, \mathcal{Q}_2 \vDash \psi$. By precision of φ and ψ, we know that $\mathcal{P}_1 \preceq \mathcal{Q}_1$ and $\mathcal{P}_2 \preceq \mathcal{Q}_2$. Using Lemma B.4, we get:

$$\mathcal{P}_1 \otimes \mathcal{P}_2 \preceq \mathcal{Q}_1 \otimes \mathcal{Q}_2 \preceq \mathcal{Q}$$

□

Lemma E.4. *If* $\mathsf{precise}(\varphi)$ *and* $\varphi \Rightarrow \lceil e \mapsto X \rceil$, *then* $\mathsf{precise}(\bigoplus_{X \sim d(E)} \varphi)$.

Proof. Take any Γ and let $\nu = d(\llbracket E \rrbracket_{\mathsf{LExp}}(\Gamma))$, if φ is unsatisfiable under any $\Gamma[X := v]$, then so is $\bigoplus_{X \sim d(E)} \varphi$, so the claim holds vacuously. If not, then there is a unique smallest $\mathcal{P}_v$ such that $\Gamma[X := v], \mathcal{P}_v \vDash \varphi$ for each $v \in \mathsf{supp}(\nu)$. Since $\varphi \Rightarrow \lceil e \mapsto X \rceil$, we can create new disjoint probability spaces $\mathcal{P}'_v$, where each $\Omega_{\mathcal{P}'_v} = \{\sigma \in \Omega_{\mathcal{P}_v} \mid \llbracket e \rrbracket_{\mathsf{Exp}}(\sigma) = v\}$. Note that this does not remove any samples that have positive probability, and clearly $\mathcal{P}_v = \mathsf{ext}(\mathcal{P}'_v)$. Let $\mathcal{P} = \bigoplus_{v \sim \nu} \mathcal{P}'_v$, then $\Gamma, \mathcal{P} \vDash \bigoplus_{X \sim d(E)} \varphi$. It only remains to show that $\mathcal{P}$ is minimal.

Take any $\mathcal{Q}$ such that $\Gamma, \mathcal{Q} \vDash \bigoplus_{X \sim d(E)} \varphi$. That means that $\Gamma[X := v], \mathsf{ext}(\mathcal{Q}_v) \vDash \varphi$, where $\bigoplus_{v \sim \nu} \mathcal{Q}_v \preceq \mathcal{Q}$. Since φ is precise, then $\mathcal{P}_v = \mathsf{ext}(\mathcal{P}'_v) \preceq \mathsf{ext}(\mathcal{Q}_v)$ for each v. Since the completion only expands the sample space with zero probability events, this must mean that $\mathcal{P}'_v \preceq \mathcal{Q}_v$ as well. Therefore, by Lemma B.5:

$$\mathcal{P} = \bigoplus_{v \sim \nu} \mathcal{P}'_v \preceq \bigoplus_{v \sim \nu} \mathcal{Q}_v \preceq \mathcal{Q}$$

□

Lemma E.5. *If* $\mathsf{convex}(\varphi_1, \varphi_2)$, *then* $\mathsf{convex}(\varphi_1 *_{\mathsf{w}} \varphi_2)$.

Proof. Take any Γ, if $\varphi_1 *_{\mathsf{w}} \varphi_2$ is satisfiable under Γ, then so are each φ_k. Since each φ_k is convex, we know that there exist Ω_k, $\mathcal{F}_k$, and S_k such that $\Gamma, \mathcal{P} \vDash \varphi$ iff $\langle \Omega_k, \mathcal{F}_k, \mu \rangle \preceq \mathcal{P}$ for some $\mu \in S_k$. Let U and V be sets such that $\Omega_1 = \mathsf{Mem}[U]$ and $\Omega_2 = \mathsf{Mem}[V]$, let $\Omega = \Omega_1 * \Omega_2$, $\mathcal{F}$ be the smallest σ-algebra containing $\{A * B \mid A \in \mathcal{F}_1, B \in \mathcal{F}_2\}$, and $S = \{\mu \mid \pi_U(\mu) \in S_1, \pi_V(\mu) \in S_2\}$ (S is clearly convex since S_1 and S_2 are convex).

We complete the proof by showing that $\Gamma, \mathcal{P} \vDash \varphi_1 *_{\mathsf{w}} \varphi_2$ iff $\langle \Omega, \mathcal{F}, \mu \rangle \preceq \mathcal{P}$ for some $\mu \in S$. For the forward direction, suppose that $\Gamma, \mathcal{P} \vDash \varphi_1 *_{\mathsf{w}} \varphi_2$, so $\Gamma, \pi_U(\mathcal{P}) \vDash \varphi_1$ and $\Gamma, \pi_V(\mathcal{P}) \vDash \varphi_2$. Due to convexity of φ_1 and φ_2, there must be $\mu_1 \in S_1$ and $\mu_2 \in S_2$ such that $\langle \Omega_1, \mathcal{F}_1, \mu_1 \rangle \preceq \pi_U(\mathcal{P})$ and $\langle \Omega_2, \mathcal{F}_2, \mu_2 \rangle \preceq \pi_V(\mathcal{P})$. Now, let:

$$\mu(A) \triangleq \mu_{\mathcal{P}}\left(\bigcup \{B \in \mathcal{F}_{\mathcal{P}} \mid \pi_{U \cup V}(B) = A\}\right)$$

This gives us:

$$\pi_U(\mu)(A) = \mu(A * \mathsf{Mem}[V]) = \mu_{\mathcal{P}}\left(\bigcup \{B \in \mathcal{F}_{\mathcal{P}} \mid \pi_{U \cup V}(B) = A * \mathsf{Mem}[V]\}\right) = \mu_{\mathcal{P}}\left(\bigcup \{B \in \mathcal{F}_{\mathcal{P}} \mid \pi_U(B) = A\}\right) = \mu_1(A)$$

And similarly, $\pi_V(\mu)(B) = \mu_2(B)$, therefore $\mu \in S$ by construction, and so also by construction $\langle \Omega, \mathcal{F}, \mu \rangle \preceq \mathcal{P}$.

For the reverse direction, suppose that $\langle \Omega, \mathcal{F}, \mu \rangle \preceq \mathcal{P}$ for some $\mu \in S$. That means that $\pi_U(\mu) \in S_1$ and $\pi_V(\mu) \in S_2$, and therefore $\Gamma, \langle \Omega_1, \mathcal{F}_1, \pi_U(\mu) \rangle \vDash \varphi_1$ and $\Gamma, \langle \Omega_2, \mathcal{F}_2, \pi_V(\mu) \rangle \vDash \varphi_2$. Since $\langle \Omega, \mathcal{F}, \mu \rangle \in \langle \Omega_1, \mathcal{F}_1, \pi_U(\mu) \rangle \diamond_{\mathsf{w}} \langle \Omega_2, \mathcal{F}_2, \pi_V(\mu) \rangle$, then $\Gamma, \mathcal{P} \vDash \varphi_1 *_{\mathsf{w}} \varphi_2$. □

Lemma E.6. *If* $\mathsf{convex}(\varphi_1, \varphi_2)$, $\varphi_1 \Rightarrow \lceil e \mapsto 1 \rceil$, *and* $\varphi_2 \Rightarrow \lceil e \mapsto 0 \rceil$, *then* $\mathsf{convex}(\varphi_1 \oplus_{\geq E} \varphi_2)$.

Proof. Take any Γ, if $\varphi_1 \oplus_{\geq E} \varphi_2$ is satisfiable under Γ, then so are φ_1 and φ_2. Let $p = [\![E]\!]_{\mathsf{LExp}}(\Gamma)$ and let X be the variable that is bound by $\oplus_{\geq E}$. Since the φ_k are convex, then for each $k \in \{1, 2\}$ there exist Ω_k, $\mathcal{F}_k$, and S_k such that $\Gamma[X := 2 - k], \mathcal{P} \vDash \varphi_k$ iff $\langle \Omega_k, \mathcal{F}_k, \mu \rangle \preceq \mathcal{P}$ for some $\mu \in S_k$. Now, let:

$$\Omega \triangleq \Omega_1 \cup \Omega_2 \qquad \mathcal{F} \triangleq \left\{A \subseteq \Omega \mid \{\sigma \in A \mid [\![e]\!]_{\mathsf{Exp}}(\sigma) = 1\} \in \mathcal{F}_1, \{\sigma \in A \mid [\![e]\!]_{\mathsf{Exp}}(\sigma) = 0\} \in \mathcal{F}_2\right\}$$

$$S \triangleq \{\mu \oplus_q \nu \mid \mu \in S_1, \nu \in S_2, p \leq q \leq 1\}$$

$$\text{where} \quad (\mu \oplus_q \nu)(A) = q \cdot \mu(\{\sigma \in A \mid [\![e]\!]_{\mathsf{Exp}}(\sigma) = 1\}) + (1 - q) \cdot \nu(\{\sigma \in A \mid [\![e]\!]_{\mathsf{Exp}}(\sigma) = 0\})$$

We complete the proof by showing that $\Gamma, \mathcal{P} \vDash \varphi_1 \oplus_{\geq E} \varphi_2$ iff $\langle \Omega, \mathcal{F}, \mu \rangle \preceq \mathcal{P}$ for some $\mu \in S$. For the forward direction, suppose that $\Gamma, \mathcal{P} \vDash \varphi_1 \oplus_{\geq E} \varphi_2$, so there exists a $q \geq p$ such that $\Gamma[X := 1], \mathsf{ext}(\mathcal{P}_1) \vDash \varphi_1$ and $\Gamma[X := 0], \mathsf{ext}(\mathcal{P}_2) \vDash \varphi_2$ for some $\mathcal{P}_1$ and $\mathcal{P}_2$ such that $\mathcal{P}_1 \oplus_q \mathcal{P}_2 \preceq \mathcal{P}$. This means that $\langle \Omega_k, \mathcal{F}_k, \mu_k \rangle \preceq \mathsf{ext}(P_k)$ for some $\mu_k \in S_k$ for each $k \in \{1, 2\}$. So, letting $\mu = \mu_1 \oplus_q \mu_2$, clearly $\mu \in S$ by construction. Finally, we get:

$$\langle \Omega, \mathcal{F}, \mu \rangle = \langle \Omega_1, \mathcal{F}_1, \mu_1 \rangle \oplus_q \langle \Omega_2, \mathcal{F}_2, \mu_2 \rangle \preceq \mathcal{P}_1 \oplus_q \mathcal{P}_2 \preceq \mathcal{P}$$

For the reverse direction, suppose that $\langle \Omega, \mathcal{F}, \mu \rangle \preceq \mathcal{P}$ for some $\mu \in S$. Since $\mu \in S$, then there exist $\mu_1 \in S_1$, $\mu_2 \in S_2$, and $q \geq p$ such that $\mu = \mu_1 \oplus_q \mu_2$. We know by construction that $\Gamma[X := 2 - k], \langle \Omega_k, \mathcal{F}_k, \mu_k \rangle \vDash \varphi_k$ for each k, therefore clearly $\Gamma, \langle \Omega, \mathcal{F}, \mu \rangle \vDash \varphi_1 \oplus_q \varphi_2$. Therefore, by Lemma E.1, we know that $\Gamma, \mathcal{P} \vDash \varphi_1 \oplus_q \varphi_2$. Finally, we weaken this assertion to get $\Gamma, \mathcal{P} \vDash \varphi_1 \oplus_{\geq E} \varphi_2$. □

Lemma E.7. *If* $\mathsf{convex}(\varphi)$ *and* $\varphi \Rightarrow \lceil e \mapsto X \rceil$, *then* $\mathsf{convex}(\&_{X \in E} \varphi)$.

Proof. Take any Γ, if $\bigoplus_{X \in E} \varphi$ is satisfiable under Γ, then so is φ. Let $S = [\![E]\!]_{\mathsf{LExp}}(\Gamma)$, so for each $v \in S$, there must be Ω_v, $\mathcal{F}_v$, and S_v such that $\Gamma[X := v], \mathcal{P} \vDash \varphi$ iff $\langle \Omega_v, \mathcal{F}_v, \mu \rangle \preceq \mathcal{P}$ for some $\mu \in S_v$. Now let:

$$\Omega \triangleq \bigcup_{v \in S} \Omega_v \qquad \mathcal{F} \triangleq \left\{A \subseteq \Omega \mid \forall v \in S.\ \{\sigma \in A \mid [\![e]\!]_{\mathsf{Exp}}(\sigma) = v\} \in \mathcal{F}_v\right\}$$

$$T \triangleq \left\{\sum_{v \in S} \xi(v) \cdot \mu_v \mid \xi \in \mathcal{D}(S), \forall v.\ \mu_v \in S_v\right\}$$

Clearly T is convex, since it is constructed as countable convex combinations of convex sets [Zilberstein et al. 2025b, Lemma B.1]. We now show that $\Gamma, \mathcal{P} \vDash \&_{X \in E}\, \varphi$ iff $\langle \Omega, \mathcal{F}, \mu \rangle \preceq \mathcal{P}$ for some $\mu \in T$. For the forward direction, suppose that $\Gamma, \mathcal{P} \vDash \&_{X \in E}\, \varphi$, so $\Gamma[X := v], \mathrm{ext}(\mathcal{P}_v) \vDash \varphi$ for each $v \in \mathrm{supp}(\nu)$ where $\nu \in \mathcal{D}(S)$ and $\bigoplus_{v \sim \nu} \mathcal{P}_v \preceq \mathcal{P}$. This means that for each v, there is a μ_v such that $\langle \Omega_v, \mathcal{F}_v, \mu_v \rangle \preceq \mathrm{ext}(\mathcal{P}_v)$. So, letting $\mu = \sum_{v \in S} \nu(v) \cdot \mu_v$, we get that $\langle \Omega, \mathcal{F}, \mu \rangle \preceq \mathcal{P}$ by construction.

For the reverse direction, suppose that $\langle \Omega, \mathcal{F}, \mu \rangle \preceq \mathcal{P}$ for some $\mu \in T$. Since $\mu \in T$, then there exists a $\xi \in \mathcal{D}(S)$ and $\mu_v \in S_v$ for all $v \in S$ such that $\mu = \sum_{v \in S} \xi(v) \cdot \mu_v$. By definition, $\Gamma[X := v], \langle \Omega_v, \mathcal{F}_v, \mu_v \rangle \vDash \varphi$. Let $\mathcal{P}_v$ be the restriction of $\langle \Omega_v, \mathcal{F}_v, \mu_v \rangle$ to the states where $[\![e]\!]_{\mathrm{Exp}}(\sigma) = v$, then in order to show that $\Gamma, \mathcal{P} \vDash \&_{X \in E}\, \varphi$, it will suffice to show that $\bigoplus_{v \sim \xi} \mathcal{P}_v \preceq \langle \Omega, \mathcal{F}, \mu \rangle$, and then we can conclude that $\bigoplus_{v \sim \xi} \mathcal{P}_v \preceq \mathcal{P}$ by transitivity and complete the proof by Lemma E.1. The conditions on Ω and $\mathcal{F}$ hold by construction. For the probability measure, we have:

$$\begin{aligned} \mathcal{P}_{\bigoplus_{v \sim \xi} \mathcal{P}_v}(A) &= \sum_{v \in S} \xi(v) \cdot \mu_{\mathcal{P}_v}(A \cap \Omega_v) \\ &= \sum_{v \in S} \xi(v) \cdot \mu_v(A \cap \Omega_v) \end{aligned}$$

Since each μ_v assigns 0 probability outside of Ω_v, we can remove the intersection.

$$\begin{aligned} &= \sum_{v \in S} \xi(v) \cdot \mu_v(A) \\ &= \mu(A) \end{aligned}$$

□

E.2 Entailment Rules

Lemma E.8. *The entailment rules in Figure 4 are valid.*

Proof.

(1) $\dfrac{P \vdash Q}{\lceil P \rceil \vdash \lceil Q \rceil}$

Suppose that $\Gamma, \mathcal{P} \vDash \lceil P \rceil$, where $\Omega_{\mathcal{P}} = \mathsf{Mem}[S]$. That means that $(\!|P|\!)^S_\Gamma \in \mathcal{F}_{\mathcal{P}}$ and $\mu((\!|P|\!)^S_\Gamma) = 1$. Since $P \vdash Q$, then it must be that $(\!|P|\!)^S_\Gamma \subseteq (\!|Q|\!)^S_\Gamma$, therefore $(\!|Q|\!)^S_\Gamma \in \mathcal{F}_{\mathcal{P}}$ because $\mathcal{P}$ is a complete probability space and all samples outside of $(\!|P|\!)^S_\Gamma$ must have measure 0. By the additivity property of probability measures $\mu((\!|Q|\!)^S_\Gamma) = 1$.

(2) $\dfrac{\varphi \vdash \varphi' \qquad \psi \vdash \psi'}{\varphi *_m \psi \vdash \varphi' *_m \psi'}$

Suppose that $\Gamma, \mathcal{P} \vDash \varphi *_m \psi$, so $\Gamma, \mathcal{P}_1 \vDash \varphi$ and $\Gamma, \mathcal{P}_2 \vDash \psi$ and $\mathcal{P}' \preceq \mathcal{P}$ for some $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}' \in \mathcal{P}_1 \diamond_m \mathcal{P}_2$. Since $\varphi \vdash \varphi'$ and $\psi \vdash \psi'$, then $\Gamma, \mathcal{P}_1 \vDash \varphi'$ and $\Gamma, \mathcal{P}_2 \vDash \psi'$, therefore we immediately conclude that $\Gamma, \mathcal{P} \vDash \varphi' *_m \psi'$.

(3) $\varphi * \psi \vdash \varphi *_{\mathrm{w}} \psi$

Suppose that $\Gamma, \mathcal{P} \vDash \varphi * \psi$, so $\Gamma, \mathcal{P}_1 \vDash \varphi$ and $\Gamma, \mathcal{P}_2 \vDash \psi$ for some $\mathcal{P}_1$ and $\mathcal{P}_2$ such that $\mathcal{P}_1 \otimes \mathcal{P}_2 \preceq \mathcal{P}$. Clearly, $\mathcal{P}_1 \otimes \mathcal{P}_2 \in \mathcal{P}_1 \diamond_{\mathrm{w}} \mathcal{P}_2$, therefore this immediately implies that $\Gamma, \mathcal{P} \vDash \varphi *_{\mathrm{w}} \psi$.

(4) $\lceil P * Q \rceil \dashv\vdash \lceil P \rceil *_m \lceil Q \rceil$

We first show that $\lceil P * Q \rceil \vdash \lceil P \rceil *_m \lceil Q \rceil$. Suppose that $\Gamma, \mathcal{P} \vDash \lceil P * Q \rceil$, where $\Omega_{\mathcal{P}} = \mathsf{Mem}[S]$. That means that $(\!|P * Q|\!)^S_\Gamma \in \mathcal{F}_{\mathcal{P}}$ and $\mu_{\mathcal{P}}((\!|P * Q|\!)^S_\Gamma) = 1$. Now let $\mathcal{P}_1$ and $\mathcal{P}_2$ be the smallest probability spaces such that $\Gamma, \mathcal{P}_1 \vDash \lceil P \rceil$ and $\Gamma, \mathcal{P}_2 \vDash \lceil Q \rceil$, so clearly $\mathcal{P}_1 \diamond_{\mathrm{w}} \mathcal{P}_2 = \mathcal{P}_1 \diamond_{\mathrm{s}} \mathcal{P}_2 = \{\mathcal{P}_1 \otimes \mathcal{P}_2\}$ and therefore $\Gamma, \mathcal{P}_1 \otimes \mathcal{P}_2 \vDash \lceil P \rceil *_m \lceil Q \rceil$. It is also clearly the case that $\mathcal{P}_1 \otimes \mathcal{P}_2 \preceq \mathcal{P}$ since the smallest event with nonzero measure in $\mathcal{P}_1 \otimes \mathcal{P}_2$ is $(\!|P|\!)_\Gamma * (\!|Q|\!)_\Gamma$, which has probability 1, therefore everything larger also has probability 1, and it suffices to show that:

$$\mu_{\mathcal{P}_1 \otimes \mathcal{P}_2}((\!|P|\!)_\Gamma * (\!|Q|\!)_\Gamma) = 1 = \mu_{\mathcal{P}}\left((\!|P * Q|\!)^S_\Gamma\right) = \mu_{\mathcal{P}}\left(\bigcup_{B \mid \pi_{\mathrm{fv}(P,Q)}(B) = (\!|P*Q|\!)^S_\Gamma} B\right)$$

Therefore, by Lemma E.1, $\Gamma, \mathcal{P} \vDash \lceil P \rceil *_m \lceil Q \rceil$.

Now we show that $\lceil P \rceil *_m \lceil Q \rceil \vdash \lceil P * Q \rceil$, suppose that $\Gamma, \mathcal{P} \vDash \lceil P \rceil *_m \lceil Q \rceil$. That means that $\mathcal{P}' \preceq \mathcal{P}$ and $\Gamma, \mathcal{P}_1 \vDash \lceil P \rceil$ and $\Gamma, \mathcal{P}_2 \vDash \lceil Q \rceil$ for some $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}' \in \mathcal{P}_1 \diamond_m \mathcal{P}_2$. This also means that $\mu_{\mathcal{P}_1}((\!|P|\!)^S_\Gamma) = 1$ and $\mu_{\mathcal{P}_2}((\!|Q|\!)^T_\Gamma) = 1$ (where $\Omega_{\mathcal{P}_1} = \mathsf{Mem}[S]$ and $\Omega_{\mathcal{P}_2} = \mathsf{Mem}[T]$). Therefore, we know that $\pi_S(\mu_{\mathcal{P}'})((\!|P|\!)^S_\Gamma) = 1$ and $\pi_T(\mu_{\mathcal{P}'})((\!|Q|\!)^T_\Gamma) = 1$, and therefore we know that anything outside of P and Q has measure zero, and therefore it must be the case that $\mu_{\mathcal{P}'}((\!|P * Q|\!)^{S \cup T}_\Gamma) = 1$. So, $\Gamma, \mathcal{P}' \vDash (\!|P * Q|\!)$, and since $\mathcal{P}' \preceq \mathcal{P}$, then by Lemma E.1, $\Gamma, \mathcal{P} \vDash (\!|P * Q|\!)$.

(5) $\varphi * \lceil P \rceil \dashv\vdash \varphi *_{\mathsf{w}} \lceil P \rceil$
The forward direction follows immediately from Item 3, so we prove only the reverse direction. Suppose that $\Gamma, \mathcal{P} \vDash \varphi *_{\mathsf{w}} \lceil P \rceil$. This means that $\Gamma, \mathcal{P}_1 \vDash \varphi$ and $\Gamma, \mathcal{P}_2 \vDash \lceil P \rceil$ for some $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}' \in \mathcal{P}_1 \diamond_{\mathsf{w}} \mathcal{P}_2$. Now let $\mathcal{P}_2'$ be the smallest probability space such that $\Gamma, \mathcal{P}_2' \vDash \lceil P \rceil$. Since $\mathcal{P}_2'$ contains events only of measure 0 or 1, then $\mathcal{P}_1 \diamond_{\mathsf{w}} \mathcal{P}_2' = \{\mathcal{P}_1 \otimes \mathcal{P}_2'\}$, therefore it must be that $\mathcal{P}_1 \otimes \mathcal{P}_2' \preceq \mathcal{P}' \preceq \mathcal{P}$. Therefore, by definition, $\Gamma, \mathcal{P} \vDash \varphi * \lceil P \rceil$.

(6) $\bigoplus_{X \sim d(E)} \varphi \vdash \&_{X \in \mathsf{supp}(d(E))} \varphi$
Suppose that $\Gamma, \mathcal{P} \vDash \bigoplus_{X \sim d(E)} \varphi$, therefore $\Gamma[X \coloneqq v], \mathsf{ext}(\mathcal{P}_v) \vDash \varphi$ for each $v \in \mathsf{supp}(\xi)$ where $\xi = d(\llbracket E \rrbracket_{\mathsf{LExp}}(\Gamma))$. Obviously $\xi \in \mathcal{D}(\mathsf{supp}(\xi))$, so this immediately implies that $\Gamma, \mathcal{P} \vDash \&_{\mathsf{supp}(d(E))} \varphi$.

(7) $\varphi[E/X] \dashv\vdash \&_{X \in \{E\}} \varphi$
For the forward direction, suppose that $\Gamma, \mathcal{P} \vDash \varphi[E/X]$. Let $v = \llbracket E \rrbracket_{\mathsf{LExp}}(\Gamma)$. We therefore have that $\Gamma[X \coloneqq v] \vDash \varphi$. Obviously, $\delta_v \in \mathcal{D}(\{v\})$, therefore we get that $\Gamma, \mathcal{P} \vDash \&_{X \in \{E\}} \varphi$.
For the reverse direction, suppose that $\Gamma, \mathcal{P} \vDash \&_{X \in \{E\}} \varphi$. Since the only distribution over a singleton support is the point-mass distribution, this immediately gives us $\Gamma[X \coloneqq v] \vDash \varphi$, where again $v = \llbracket E \rrbracket_{\mathsf{LExp}}(\Gamma)$. Finally, we conclude that $\Gamma, \mathcal{P} \vDash \varphi[E/X]$.

(8) $\lceil E \subseteq E' \rceil * \&_{X \in E} \varphi \vdash \&_{X \in E'} \varphi$
Suppose that $\Gamma, \mathcal{P} \vDash \lceil E \subseteq E' \rceil * \&_{X \in E} \varphi$. Let $S = \llbracket E \rrbracket_{\mathsf{LExp}}(\Gamma)$ and $S' = \llbracket E' \rrbracket_{\mathsf{LExp}}(\Gamma)$. We therefore know that $S \subseteq S'$ and $\Gamma[X \coloneqq v], \mathsf{ext}(\mathcal{P}_v) \vDash \varphi$ for all $v \in \mathsf{supp}(\xi)$ and some $\xi \in \mathcal{D}(S)$ such that $\bigoplus_{v \sim \xi} \mathcal{P}_v \preceq \mathcal{P}$. Obviously, it is also the case that $\xi \in \mathcal{D}(S')$, since $S \subseteq S'$, therefore we immediately have that $\Gamma, \mathcal{P} \vDash \&_{X \in E'} \varphi$.

(9) $$\frac{\varphi \vdash \psi}{\bigoplus_{X \sim d(E)} \varphi \vdash \bigoplus_{X \sim d(E)} \psi}$$
Suppose that $\Gamma, \mathcal{P} \vDash \bigoplus_{X \sim d(E)} \varphi$ and let $\nu = d(\llbracket E \rrbracket_{\mathsf{LExp}}(\Gamma))$. So, $\bigoplus_{v \sim \nu} \mathcal{P}_v \preceq \mathcal{P}$ such that $\Gamma[X \coloneqq v], \mathsf{ext}(\mathcal{P}_v) \vDash \varphi$ for each v. Since $\varphi \vdash \psi$, we get that $\Gamma[X \coloneqq v], \mathsf{ext}(\mathcal{P}_v) \vDash \psi$ for each v. Therefore, $\Gamma, \mathcal{P} \vDash \bigoplus_{X \sim d(E)} \psi$. Note that ψ may not witness a partition of the sample space, but the $\mathcal{P}_v$s are still disjoint.

(10) $$\frac{Y \notin \mathsf{fv}(\varphi)}{\bigoplus_{X \sim d(E)} \varphi \vdash \bigoplus_{Y \sim d(E)} \varphi[Y/X]}$$
Suppose that $\Gamma, \mathcal{P} \vDash \bigoplus_{X \sim d(E)} \varphi$, and let $\nu = d(\llbracket E \rrbracket_{\mathsf{LExp}}(\Gamma))$. This means that $\bigoplus_{v \sim \nu} \mathcal{P}_v \preceq \mathcal{P}$ such that $\Gamma[X \coloneqq v], \mathsf{ext}(\mathcal{P}_v) \vDash \varphi$ for each $v \in \mathsf{supp}(\nu)$. Since $Y \notin \mathsf{fv}(\varphi)$, then clearly $\Gamma[Y \coloneqq v], \mathsf{ext}(\mathcal{P}_v) \vDash \varphi[Y/X]$. Therefore, we get that $\Gamma, \bigoplus_{v \sim \nu} \mathcal{P}_v \vDash \bigoplus_{Y \sim d(E)} \varphi[Y/X]$, and since $\bigoplus_{v \sim \nu} \mathcal{P}_v \preceq \mathcal{P}$, then $\Gamma, \mathcal{P} \vDash \bigoplus_{Y \sim d(E)} \varphi[Y/X]$ by Lemma E.1.

(11) $$\frac{X \notin \mathsf{fv}(\psi)}{(\bigoplus_{X \sim d(E)} \varphi) * \psi \vdash \bigoplus_{X \sim d(E)} (\varphi * \psi)}$$
Suppose that $\Gamma, \mathcal{P} \vDash (\bigoplus_{X \sim d(E)} \varphi) * \psi$ and let $\nu = d(\llbracket E \rrbracket_{\mathsf{LExp}}(\Gamma))$. So, $(\bigoplus_{v \sim \nu} \mathcal{P}_v) \otimes \mathcal{P}' \preceq \mathcal{P}$ such that $\Gamma[X \coloneqq v], \mathsf{ext}(\mathcal{P}_v) \vDash \varphi$ for each v and $\Gamma, \mathcal{P}' \vDash \psi$. Since $X \notin \mathsf{fv}(\psi)$, then $\Gamma[X \coloneqq v], \mathcal{P}' \vDash \psi$ for each v. Also note that $\mathsf{ext}(\mathcal{P}_v) \otimes \mathcal{P}' = \mathsf{ext}(\mathcal{P}_v \otimes \mathcal{P}')$ since $\mathcal{P}'$ is already a complete probability space. This gives us $\Gamma[X \coloneqq v], \mathsf{ext}(\mathcal{P}_v \otimes \mathcal{P}') \vDash \varphi * \psi$. Now, by Lemma B.3, we have:

$$\bigoplus_{v \sim \nu} (\mathcal{P}_v \otimes \mathcal{P}') = \left(\bigoplus_{v \sim \nu} \mathcal{P}_v\right) \otimes \mathcal{P}' \preceq \mathcal{P}$$

Therefore, $\Gamma, \mathcal{P} \vDash \bigoplus_{X \sim d(E)} (\varphi * \psi)$.

(12) $$\frac{X \notin \mathsf{fv}(\psi) \qquad \mathsf{precise}(\psi)}{\bigoplus_{X \sim d(E)} (\varphi * \psi) \vdash (\bigoplus_{X \sim d(E)} \varphi) * \psi}$$
Suppose that $\Gamma, \mathcal{P} \vDash \bigoplus_{X \sim d(E)} (\varphi * \psi)$, and let $\nu = d(\llbracket E \rrbracket_{\mathsf{LExp}}(\Gamma))$. This means that $\bigoplus_{v \sim \nu} (\mathcal{P}_v \otimes \mathcal{Q}_v) \preceq \mathcal{P}$ such that $\Gamma[X \coloneqq v], \mathsf{ext}(\mathcal{P}_v) \vDash \varphi$ and $\Gamma[X \coloneqq v], \mathsf{ext}(\mathcal{Q}_v) \vDash \psi$. Since $X \notin \mathsf{fv}(\psi)$, we also know that $\Gamma, \mathsf{ext}(\mathcal{Q}_v) \vDash \psi$, and since ψ is precise, there is a unique $\mathcal{Q}$ such that $\Gamma, \mathcal{Q} \vDash \psi$ and $\mathcal{Q} \preceq \mathsf{ext}(\mathcal{Q}_v)$ for all v. Therefore, by recombining the components, we get that $\Gamma, \mathcal{P} \vDash (\bigoplus_{X \sim d(E)} \varphi) * \psi$.

(13) $$\frac{X \notin \mathsf{fv}(\psi) \qquad \mathsf{convex}(\psi)}{\bigoplus_{X \sim d(E)} (\varphi *_{\mathsf{w}} \psi) \vdash (\bigoplus_{X \sim d(E)} \varphi) *_{\mathsf{w}} \psi}$$

Suppose that $\Gamma, \mathcal{P} \vDash \bigoplus_{X\sim d(E)} (\varphi *_{\mathsf{w}} \psi)$, which means that $\Gamma[X := v], \mathsf{ext}(\mathcal{P}_v) \vDash \varphi *_{\mathsf{w}} \psi$ for each $v \in \mathsf{supp}(\xi)$ where $\xi = d([\![E]\!]_{\mathsf{LExp}}(\Gamma))$ and $\bigoplus_{v\sim\xi} \mathcal{P}_v \preceq \mathcal{P}$. Let S be the set such that $\Omega_{\mathcal{P}} = \mathsf{Mem}[S]$ and $T = \mathsf{vars}(\psi)$, so $\Gamma[X := v], \pi_{S\setminus T}(\mathsf{ext}(\mathcal{P}_v)) \vDash \varphi$, and therefore $\Gamma, \pi_{S\setminus T}(\mathcal{P}) \vDash \varphi$. In addition, we know that $\Gamma[X := v], \pi_T(\mathsf{ext}(\mathcal{P}_v)) \vDash \psi$ for each v, and since ψ is convex, then $\Gamma, \pi_T(\mathcal{P}) \vDash \psi$ (see Item 14 for more details). Combining these two facts, we get that $\Gamma, \mathcal{P} \vDash (\bigoplus_{X\sim d(E)} \varphi) *_{\mathsf{w}} \psi$.

(14) $$\frac{X \notin \mathsf{fv}(\varphi) \qquad \mathsf{convex}(\varphi)}{\bigoplus_{X\sim d(E)} \varphi \vdash \varphi}$$

Suppose that $\Gamma, \mathcal{P} \vDash \bigoplus_{X\sim d(E)} \varphi$, and let $\nu = d([\![E]\!]_{\mathsf{LExp}}(\Gamma))$. This means that $\bigoplus_{v\sim\nu} \mathcal{P}_v \preceq \mathcal{P}$ and $\Gamma[X := v], \mathsf{ext}(\mathcal{P}_v) \vDash \varphi$. Since $X \notin \mathsf{fv}(\varphi)$,this also means that $\Gamma, \mathsf{ext}(\mathcal{P}_v) \vDash \varphi$. Since φ is convex, there exist Ω, $\mathcal{F}$, and S and for each v $\mu_v \in S$ such that $\Gamma, \langle \Omega, \mathcal{F}, \mu_v\rangle \vDash \varphi$ and $\langle \Omega, \mathcal{F}, \mu_v\rangle \preceq \mathsf{ext}(\mathcal{P}_v)$. Now, let $\mu = \sum_{x\in\mathsf{supp}(\nu)} \nu(v) \cdot \mu_v$, so clearly $\mu \in S$ since it is a convex combination of elements of S, and therefore, $\Gamma, \langle \Omega, \mathcal{F}, \mu\rangle \vDash \varphi$. It remains only to show that $\langle \Omega, \mathcal{F}, \mu\rangle \preceq \mathcal{P}$. The conditions on Ω and $\mathcal{F}$ hold trivially since $\langle \Omega, \mathcal{F}, \mu_v\rangle \preceq \mathsf{ext}(\mathcal{P}_v)$ for all v. Now, we show the condition on $\mu_{\mathcal{P}}$:

$$\mu(A) = \sum_{v\in\mathsf{supp}(\nu)} \nu(v) \cdot \mu_v(A)$$

Let U be the set such that $\Omega = \mathsf{Mem}[U]$. Since $\langle \Omega, \mathcal{F}, \mu_v\rangle \preceq \mathsf{ext}(\mathcal{P}_v)$.

$$= \sum_{v\in\mathsf{supp}(\nu)} \nu(v) \cdot \mu_{\mathcal{P}_v}\left(\bigcup \{B \in \mathcal{F}_{\mathcal{P}_v} \mid \pi_U(B) = A\}\right)$$

Since $\langle \Omega, \mathcal{F}, \mu_v\rangle \preceq \mathsf{ext}(\mathcal{P}_v)$, then μ_v gives measure 0 to all events outside of $\Omega_{\mathcal{P}_v}$, therefore we can add the following intersection.

$$= \sum_{v\in\mathsf{supp}(\nu)} \nu(v) \cdot \mu_{\mathcal{P}_v}\left(\bigcup \{B \in \mathcal{F}_{\mathcal{P}_v} \mid \pi_U(B) = A\} \cap \Omega_{\mathcal{P}_v}\right)$$

$$= \mu_{\bigoplus_{v\sim\nu} \mathcal{P}_v}\left(\bigcup \{B \in \mathcal{F}_{\mathcal{P}} \mid \pi_U(B) = A\}\right)$$

Since $\bigoplus_{v\sim\nu} \mathcal{P}_v \preceq \mathcal{P}$.

$$= \mu_{\mathcal{P}}\left(\bigcup \{B \in \mathcal{F}_{\mathcal{P}} \mid \pi_U(B) = A\}\right)$$

□

E.3 Soundness of Inference Rules

We start by providing a lemma stating that weak triples can be stated without the frame preservation property.

Lemma E.9 (Alternative Characterization of Weak Triples).

$$I \vDash_{\mathsf{w}} \langle\varphi\rangle\ C\ \langle\psi\rangle \qquad \textit{iff} \qquad \forall \Gamma, \mu.\ \Gamma, \mu \vDash \varphi * \lceil I\rceil \implies \forall \nu \in \mathcal{L}^{\langle\!|I|\!\rangle_\Gamma}([\![C]\!])^\dagger(\mu).\ \Gamma, \nu \vDash \psi * \lceil I\rceil$$

Proof. We show both directions:

- ($\Rightarrow$) Suppose that $\Gamma, \mu \vDash \varphi * \lceil I\rceil$. Let $\mathcal{P}_F$ be the trivial probability space on $\mathsf{Mem}[\emptyset]$, so clearly $\mu \diamond_{\mathsf{w}} \mathcal{P}_F = \{\mu\}$. Now take any $\nu \in \mathcal{L}^{\langle\!|I|\!\rangle_\Gamma}([\![C]\!])^\dagger(\mu)$. Since $I \vDash_{\mathsf{w}} \langle\varphi\rangle\ C\ \langle\psi\rangle$, we get that there exists a $\mathcal{Q}$ and $\mathcal{Q}' \in \mathcal{Q} \diamond_{\mathsf{w}} \mathcal{P}_F$ such that $\mathcal{Q}' \preceq \nu$ and $\Gamma, \mathcal{Q} \vDash \psi * \lceil I\rceil$. Since $\mathcal{Q} \diamond_{\mathsf{w}} \mathcal{P}_F = \{\mathcal{Q}\}$, then $\mathcal{Q}' = \mathcal{Q}$, therefore $\Gamma, \mathcal{Q}' \vDash \psi * \lceil I\rceil$. In addition, since $\mathcal{Q}' \preceq \nu$, then by Lemma E.1, $\Gamma, \nu \vDash \psi * \lceil I\rceil$.
- ($\Leftarrow$) Suppose that $\mathcal{P}' \in \mathcal{P} \diamond_{\mathsf{w}} \mathcal{P}_F$ such that $\mathcal{P}' \preceq \mu$ and $\Gamma, \mathcal{P} \vDash \varphi * \lceil I\rceil$. Let $U \subseteq \mathsf{Var}$ be the variables of $\mathcal{P}$ and I and V be the variables of $\mathcal{P}_F$. Clearly $\Gamma, \pi_U(\mu) \vDash \varphi * \lceil I\rceil$. Since $\pi_U(\mu) \preceq \mu$, then by Lemma E.1, $\Gamma, \mu \vDash \varphi * \lceil I\rceil$ too. So, by the premise, $\Gamma, \nu \vDash \psi * \lceil I\rceil$ for any $\nu \in \mathcal{L}^{\langle\!|I|\!\rangle_\Gamma}([\![C]\!])^\dagger(\mu)$. But ψ and I only depend on the variables U, so $\Gamma, \pi_U(\nu) \vDash \psi * \lceil I\rceil$ too. In addition, it must be the case that $\mathcal{P}_F \preceq \pi_V(\nu)$ since C does not alter any variables in V. Now, let $\mathcal{Q} = \pi_U(\nu)$ and construct $\mathcal{Q}'$ as follows:

  $$\Omega_{\mathcal{Q}'} \triangleq \Omega_{\mathcal{P}'} \qquad \mathcal{F}_{\mathcal{Q}'} \triangleq \sigma(\{A_1 * A_2 \mid A_1 \in \mathcal{F}_{\mathcal{Q}}, A_2 \in \mathcal{F}_{\mathcal{P}_F}\}) \qquad \mu_{\mathcal{Q}'}(A) = \nu(A)$$

  So clearly by construction, $\mathcal{Q}' \in \mathcal{Q} \diamond_{\mathsf{w}} \mathcal{P}_F$ and $\mathcal{Q}' \preceq \nu$ and $\Gamma, \mathcal{Q} \vDash \psi * \lceil I\rceil$. Therefore, we are done.

□

Theorem 5.2 (Soundness). *For all of the rules in Figures 5 to 8, if $I \vdash_m \langle\varphi\rangle\ C\ \langle\psi\rangle$ then $I \vDash_m \langle\varphi\rangle\ C\ \langle\psi\rangle$.*

Proof. The proof is by induction on the derivation.

$$\frac{I \vdash_m \langle \varphi_1 \rangle\ C\ \langle \psi_1 \rangle \qquad I \vdash_m \langle \varphi_2 \rangle\ C\ \langle \psi_2 \rangle}{I \vdash_m \langle \varphi_1 \vee \varphi_2 \rangle\ C\ \langle \psi_1 \vee \psi_2 \rangle}\textsc{Disj} \qquad \frac{I \vdash_m \langle \varphi \rangle\ C\ \langle \psi \rangle \qquad X \notin \mathsf{vars}(\psi, I)}{I \vdash_m \langle \exists X.\ \varphi \rangle\ C\ \langle \psi \rangle}\textsc{Exists2}$$

$$\frac{I \vdash_m \langle \varphi \rangle\ C\ \langle \psi \rangle}{I[E/X] \vdash_m \langle \varphi[E/X] \rangle\ C\ \langle \psi[E/X] \rangle}\textsc{Subst}$$

Fig. 12. Additional Inference Rules

- Skip.

$$\frac{}{I \vdash_m \langle \varphi \rangle\ \mathbf{skip}\ \langle \varphi \rangle}\textsc{Skip}$$

Suppose that $\mathcal{P}' \in \mathcal{P} \diamond_m \mathcal{P}_F$, $\mathcal{P}' \preceq \mu$, and $\Gamma, \mathcal{P} \vDash \varphi * \lceil I \rceil$. By Zilberstein et al. [2025a, Lemma 5.2], we have:

$$\mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket \mathbf{skip} \rrbracket)^\dagger(\mu) = \eta^\dagger(\mu) = \{\mu\}$$

So, letting $\mathcal{Q} = \mathcal{P}$ and $\mathcal{Q}' = \mathcal{P}'$, clearly $\mathcal{Q}' \in \mathcal{Q} \diamond_m \mathcal{P}_F$ and $\mathcal{Q}' \preceq \mu$ and $\Gamma, \mathcal{Q} \vDash \varphi * \lceil I \rceil$.

- Seq.

$$\frac{I \vdash_m \langle \varphi \rangle\ C_1\ \langle \vartheta \rangle \qquad I \vdash_m \langle \vartheta \rangle\ C_1\ \langle \psi \rangle}{I \vdash_m \langle \varphi \rangle\ C_1 \fatsemi C_2\ \langle \psi \rangle}\textsc{Seq}$$

Suppose that $\mathcal{P}' \in \mathcal{P} \diamond_m \mathcal{P}_F$, $\mathcal{P}' \preceq \mu$, and $\Gamma, \mathcal{P} \vDash \varphi * \lceil I \rceil$. By Zilberstein et al. [2025a, Lemma 5.2], we have:

$$\begin{aligned}\mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C_1 \fatsemi C_2 \rrbracket)^\dagger(\mu) &= \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C_1 \rrbracket \fatsemi \llbracket C_2 \rrbracket)^\dagger(\mu) \\ &= \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C_2 \rrbracket)^\dagger(\mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C_1 \rrbracket)^\dagger(\mu)) \\ &= \bigcup_{\mu' \in \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C_1 \rrbracket)^\dagger(\mu)} \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C_2 \rrbracket)^\dagger(\mu')\end{aligned}$$

So, for every $\nu \in \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C_1 \fatsemi C_2 \rrbracket)^\dagger(\mu)$, there is a $\mu' \in \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C_1 \rrbracket)^\dagger(\mu)$ such that $\nu \in \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C_2 \rrbracket)^\dagger(\mu')$. By the induction hypotheses, we know there exist $\mathcal{Q}_1$ and $\mathcal{Q}_1' \in \mathcal{Q}_1 \diamond_m \mathcal{P}_F$ such that $\mathcal{Q}_1' \preceq \mu'$ and $\Gamma, \mathcal{Q}_1 \vDash \vartheta * \lceil I \rceil$. By the induction hypothesis again, we get that there exist $\mathcal{Q}$ and $\mathcal{Q}' \in \mathcal{Q} \diamond_m \mathcal{P}_F$ such that $\mathcal{Q}' \preceq \nu$ and $\Gamma, \mathcal{Q} \vDash \psi * \lceil I \rceil$.

- IfT.

$$\frac{\varphi \Rightarrow \lceil b \mapsto \mathsf{true} \rceil \qquad I \vdash_m \langle \varphi \rangle\ C_1\ \langle \psi \rangle}{I \vdash_m \langle \varphi \rangle\ \mathbf{if}\ b\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2\ \langle \psi \rangle}\textsc{IfT}$$

Suppose that $\mathcal{P}' \in \mathcal{P} \diamond_m \mathcal{P}_F$, $\mathcal{P}' \preceq \mu$, and $\Gamma, \mathcal{P} \vDash \varphi * \lceil I \rceil$. We therefore know that $\llbracket b \rrbracket_{\mathsf{Test}}(\sigma) = \mathsf{true}$ for all $\sigma \in \mathsf{supp}(\mu)$. So, by Zilberstein et al. [2025a, Lemma 5.2], we get that:

$$\begin{aligned}\mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket \mathbf{if}\ b\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2 \rrbracket)^\dagger(\mu) &= \mathcal{L}^{(\!|I|\!)_\Gamma}(\mathsf{guard}(b, \llbracket C_1 \rrbracket, \llbracket C_2 \rrbracket))^\dagger(\mu) \\ &= \left(\lambda\sigma.\begin{cases} \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C_1 \rrbracket)(\sigma) & \text{if } \llbracket b \rrbracket_{\mathsf{Test}}(\sigma) = \mathsf{true} \\ \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C_2 \rrbracket)(\sigma) & \text{if } \llbracket b \rrbracket_{\mathsf{Test}}(\sigma) = \mathsf{false} \end{cases}\right)^\dagger(\mu) \\ &= \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C_1 \rrbracket)^\dagger(\mu)\end{aligned}$$

So any $\nu \in \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket \mathbf{if}\ b\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2 \rrbracket)^\dagger(\mu)$ must also be in $\mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C_1 \rrbracket)^\dagger(\mu)$, and therefore we can use the induction hypothesis to conclude that there is a $\mathcal{Q}$ and $\mathcal{Q}' \in \mathcal{Q} \diamond_m \mathcal{P}_F$ such that $\mathcal{Q}' \preceq \nu$ and $\Gamma, \mathcal{Q} \vDash \psi * \lceil I \rceil$.

- IfF. Symmetric to the IfT case.
- Assign.

$$\frac{\varphi \Rightarrow \lceil e \mapsto E \rceil \wedge (\psi * \lceil \mathsf{own}(x) \rceil)}{I \vdash_m \langle \varphi \rangle\ x := e\ \langle \psi * \lceil x \mapsto E \rceil \rangle}\textsc{Assign}$$

We prove only the $m = \mathsf{s}$ case, as the $m = \mathsf{w}$ case follows from the soundness of the Weakening rule. Suppose that $\mathcal{P} \otimes \mathcal{P}_F \preceq \mu$ and $\Gamma, \mathcal{P} \vDash \varphi * \lceil x \mapsto E \rceil * \lceil I \rceil$. Let S be the set of variables such that $\mu \in \mathcal{D}(\mathsf{Mem}[S])$. Since $\varphi \Rightarrow \lceil e \mapsto E \rceil$, we know that $\llbracket e \rrbracket_{\mathsf{Exp}}(\sigma) = \llbracket E \rrbracket_{\mathsf{LExp}}(\Gamma)$ for all $\sigma \in \mathsf{supp}(\mu)$. Now take any $\nu \in \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket x := e \rrbracket)^\dagger(\mu)$. We know from Zilberstein et al. [2025a, Lemma 5.2] that:

$$\begin{aligned}\mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket x := e \rrbracket)^\dagger(\mu) &= \mathcal{L}^{(\!|I|\!)_\Gamma}(\langle x := e \rangle)^\dagger(\mu) \\ &= \left(\llbracket x := e \rrbracket_{\mathsf{Act}}^{(\!|I|\!)_\Gamma}\right)^\dagger(\mu)\end{aligned}$$

From the premise of the rule, we know that $x := e$ does not depend on $\mathsf{vars}(I)$, so the invariant sensitive semantics is the same as the regular semantics.

$$
\begin{aligned}
&= [\![x := e]\!]^{\dagger}_{\mathsf{Act}}(\mu) \\
&= \left\{ \sum_{\sigma \in \mathsf{supp}(\mu)} \mu(\sigma) \cdot \nu_\sigma \;\middle|\; \forall \sigma.\ \nu_\sigma \in [\![x := e]\!]_{\mathsf{Act}}(\sigma) \right\} \\
&= \left\{ \sum_{\sigma \in \mathsf{supp}(\mu)} \mu(\sigma) \cdot \delta_{\sigma[x := [\![e]\!]_{\mathsf{Exp}}(\sigma)]} \right\} \\
&= \left\{ \sum_{\sigma \in \mathsf{supp}(\mu)} \mu(\sigma) \cdot \delta_{\sigma[x := [\![E]\!]_{\mathsf{LExp}}(\Gamma)]} \right\}
\end{aligned}
$$

Now, since $\varphi \Rightarrow \psi * \lceil \mathsf{own}(x) \rceil$, we get that $\Gamma, \mathcal{P} \vDash \psi * \lceil \mathsf{own}(x) \rceil * \lceil I \rceil$, so there exist $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ such that $\mathcal{P}_1 \otimes \mathcal{P}_2 \otimes \mathcal{P}_3 \preceq \mathcal{P}$ and $\Gamma, \mathcal{P}_1 \vDash \psi$ and $\Gamma, \mathcal{P}_2 \vDash \lceil \mathsf{own}(x) \rceil$ and $\Gamma, \mathcal{P}_3 \vDash \lceil I \rceil$. Since $\lceil \mathsf{own}(x) \rceil$ is precise, let $\mathcal{P}'_2$ be the unique smallest probability space such that $\Gamma, \mathcal{P}'_2 \vDash \lceil \mathsf{own}(x) \rceil$, and note that $\mathcal{P}'_2 \preceq \mathcal{P}_2$ and $\Omega_{\mathcal{P}'_2} = \mathsf{Mem}[\{x\}]$. So, by Lemma B.4, we have:

$$\mathcal{P}_1 \otimes \mathcal{P}'_2 \otimes \mathcal{P}_3 \otimes \mathcal{P}_F \preceq \mathcal{P}_1 \otimes \mathcal{P}_2 \otimes \mathcal{P}_3 \otimes \mathcal{P}_F \preceq \mathcal{P} \otimes \mathcal{P}_F \preceq \mu$$

From above, we know that $\nu = \sum_{\sigma \in \mathsf{supp}(\mu)} \mu(\sigma) \cdot \delta_{\sigma[x := [\![E']\!]_{\mathsf{LExp}}(\Gamma)]}$, so it is easy to see that $\pi_{S \setminus \{x\}}(\nu) = \pi_{S \setminus \{x\}}(\mu)$, therefore:

$$\mathcal{P}_1 \otimes \mathcal{P}_3 \otimes \mathcal{P}_F \preceq \pi_{S \setminus \{x\}}(\mu) = \pi_{S \setminus \{x\}}(\nu)$$

Let $\mathcal{Q}$ be the trivial probability space that satisfies $\lceil x \mapsto E \rceil$, so that $\Omega_{\mathcal{Q}} = \mathsf{Mem}[\{x\}]$ all events in $\mathcal{F}_{\mathcal{Q}}$ have probability 0 or 1, and $\mathcal{Q} \preceq \pi_{\{x\}}(\nu)$. So, clearly $\Gamma, \mathcal{P}_1 \otimes \mathcal{Q} \otimes \mathcal{P}_3 \vDash \psi * \lceil x \mapsto E \rceil * \lceil I \rceil$ and: $(\mathcal{P}_1 \otimes \mathcal{Q} \otimes \mathcal{P}_3) \otimes \mathcal{P}_F \preceq \nu$.

- Samp.

$$\frac{\varphi \Rightarrow \lceil e \mapsto E \rceil \wedge (\psi * \lceil \mathsf{own}(x) \rceil}{I \vdash_m \langle \varphi \rangle\ x :\approx d(e)\ \langle \psi * (x \sim d(E)) \rangle}\ \textsc{Samp}$$

We prove only the $m = \mathsf{s}$ case, as the $m = \mathsf{w}$ case follows from the soundness of the Weakening rule. Suppose that $\mathcal{P} \otimes \mathcal{P}_F \preceq \mu$ and $\Gamma, \mathcal{P} \vDash \varphi * \lceil I \rceil$, where $\mu \in \mathcal{D}(\mathsf{Mem}[S])$. Since $\varphi \Rightarrow \lceil e \mapsto E \rceil$, we know that $[\![e]\!]_{\mathsf{Exp}}(\sigma) = [\![E]\!]_{\mathsf{LExp}}(\Gamma)$ for all $\sigma \in \mathsf{supp}(\mu)$. Now take any $\nu \in \mathcal{L}^{(\!|I|\!)_\Gamma}([\![x :\approx d(e)]\!])^{\dagger}(\mu)$. We know that:

$$\mathcal{L}^{(\!|I|\!)_\Gamma}([\![x :\approx d(e)]\!])^{\dagger}(\mu) = \left([\![x :\approx d(e)]\!]^{(\!|I|\!)_\Gamma}_{\mathsf{Act}} \right)^{\dagger}(\mu)$$

From the premise of the rule, we know that $x :\approx e$ does not depend on $\mathsf{vars}(I)$, so the invariant sensitive semantics is the same as the regular semantics.

$$
\begin{aligned}
&= [\![x :\approx d(e)]\!]^{\dagger}_{\mathsf{Act}}(\mu) \\
&= \left\{ \sum_{\sigma \in \mathsf{supp}(\mu)} \mu(\sigma) \cdot \nu_\sigma \;\middle|\; \forall \sigma.\ \nu_\sigma \in [\![x :\approx d(e)]\!]_{\mathsf{Act}}(\sigma) \right\} \\
&= \left\{ \sum_{\sigma \in \mathsf{supp}(\mu)} \mu(\sigma) \cdot \sum_{v \in \mathsf{Val}} d([\![e]\!]_{\mathsf{Exp}}(\sigma))(v) \cdot \delta_{\sigma[x := v]} \right\} \\
&= \left\{ \sum_{v \in \mathsf{Val}} d([\![E]\!]_{\mathsf{LExp}}(\Gamma))(v) \cdot \sum_{\sigma \in \mathsf{supp}(\mu)} \mu(\sigma) \cdot \delta_{\sigma[x := v]} \right\}
\end{aligned}
$$

Let $\nu_v = \sum_{\sigma \in \mathsf{supp}(\mu)} \mu(\sigma) \cdot \delta_{\sigma[x := v]}$, therefore $\nu = \sum_{v \in \mathsf{Val}} d([\![E]\!]_{\mathsf{LExp}}(\Gamma))(v) \cdot \nu_v$. Note also that $\pi_{S \setminus \{x\}}(\mu) = \pi_{S \setminus \{x\}}(\nu)$. Since $\varphi \Rightarrow \psi * \lceil \mathsf{own}(x) \rceil$, we know that $\Gamma, \mathcal{P} \vDash \psi * \lceil \mathsf{own}(x) \rceil * \lceil I \rceil$, so there exist $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ such that $\Gamma, \mathcal{P}_1 \vDash \psi$, $\Gamma, \mathcal{P}_2 \vDash \lceil \mathsf{own}(x) \rceil$, and $\Gamma, \mathcal{P}_3 \vDash \lceil I \rceil$. This gives us:

$$\mathcal{P}_1 \otimes \mathcal{P}_3 \otimes \mathcal{P}_F \preceq \pi_{S \setminus \{x\}}(\mu) = \pi_{S \setminus \{x\}}(\nu) = \pi_{S \setminus \{x\}}(\nu_v)$$

Now, let $\mathcal{Q}_v$ be the trivial probability space such that $\Omega_{\mathcal{Q}_v} = \{\sigma \in \mathsf{Mem}[\{x\}] \mid \sigma(x) = v\}$ and $\Gamma[X := v], \mathcal{Q}_v \vDash \lceil x \mapsto X \rceil$. By Construction:

$$\mathcal{P}_1 \otimes \mathsf{ext}(\mathcal{Q}_v) \otimes \mathcal{P}_3 \otimes \mathcal{P}_F = (\mathcal{P}_1 \otimes \mathcal{P}_3 \otimes \mathcal{P}_F) \otimes \mathsf{ext}(\mathcal{Q}_v) \preceq \pi_{S \setminus \{x\}}(\nu_v) \otimes \pi_{\{x\}}(\nu_v) = \nu_v$$

Let $\xi = d(\llbracket E \rrbracket_{\mathsf{LExp}}(\Gamma))$ and $\mathcal{Q} = \mathcal{P}_1 \otimes (\bigoplus_{v \sim \xi} \mathcal{Q}_v) \otimes \mathcal{P}_3$, then using Lemmas B.3 and B.5 we get:

$$\mathcal{Q} \otimes P_F = \mathcal{P}_1 \otimes \left(\bigoplus_{v \sim \xi} \mathcal{Q}_v\right) \otimes \mathcal{P}_3 \otimes \mathcal{P}_F = \bigoplus_{v \sim \xi} \mathcal{P}_1 \otimes \mathcal{Q}_v \otimes \mathcal{P}_3 \otimes \mathcal{P}_F \preceq \sum_{v \in \mathsf{supp}(\xi)} \xi(v) \cdot \nu_v = \nu$$

By construction, we also have $\Gamma, \mathcal{Q} \vDash \psi * (x \sim d(E)) * \lceil I \rceil$.

- Par.

$$\frac{I \vdash \langle \varphi_1 \rangle\ C_1\ \langle \psi_1 \rangle \qquad I \vdash \langle \varphi_2 \rangle\ C_2\ \langle \psi_2 \rangle \qquad \mathsf{precise}(\psi_1, \psi_2)}{I \vdash \langle \varphi_1 * \varphi_2 \rangle\ C_1 \parallel C_2\ \langle \psi_1 * \psi_2 \rangle}\textsc{Par}$$

Since we are using a strong triple, suppose that $\mathcal{P}_1 \otimes \mathcal{P}_2 \otimes \mathcal{P}_I \otimes \mathcal{P}_F \preceq \mu$ such that $\Gamma, \mathcal{P}_1 \vDash \varphi_1$ and $\Gamma, \mathcal{P}_2 \vDash \varphi_2$ and $\Gamma, \mathcal{P}_I \vDash \lceil I \rceil$. Without loss of generality, suppose that $\mathcal{P}_I$ is minimal. To complete the proof, we need to establish the premise of Lemma C.6. Since ψ_1 and ψ_2 are precise, let $\mathcal{Q}_1$ and $\mathcal{Q}_2$ be the unique smallest probability spaces that satisfy them under Γ (note that if ψ_1 or ψ_2 is unsatisfiable, then the premise of the rule is false, so the claim holds vacuously). Take any μ_1 such that $\mathcal{P}_1 \otimes \mathcal{P}_I \preceq \mu_1$ and any $\nu_1 \in \mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}(\llbracket C_1 \rrbracket)^\dagger(\mu_1)$. By the premise of the Par rule, we know that there is a $\mathcal{Q}_1' \preceq \nu_1$ such that $\Gamma, \mathcal{Q}_1' \vDash \psi_1 * \lceil I \rceil$, therefore $\mathcal{Q}_1 * \mathcal{P}_I \preceq \mathcal{Q}_1'$, and therefore we have shown that:

$$\forall \mu_1.\ \mathcal{P}_1 \otimes \mathcal{P}_I \preceq \mu_1 \implies \forall \nu_1 \in \mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}(\llbracket C_1 \rrbracket)^\dagger(\mu_1).\ \mathcal{Q}_1 \otimes \mathcal{P}_I \preceq \nu_1$$

We now perform a nearly identical argument for C_2, but we also handle the frame. Take any μ_2 such that $\mathcal{P}_2 \otimes \mathcal{P}_I \otimes \mathcal{P}_F \preceq \mu_2$ and any $\nu_2 \in \mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}(\llbracket C_2 \rrbracket)^\dagger(\mu_2)$. By the second premise of the Par rule, we get that there is a $\mathcal{Q}_2'$ such that $\mathcal{Q}_2' \otimes \mathcal{P}_F \preceq \nu_2$ and $\Gamma, \mathcal{Q}_2' \vDash \psi_2 * \lceil I \rceil$. Due to precision, we know that $\mathcal{Q}_2 * \mathcal{P}_I \preceq \mathcal{Q}_2'$ and so by Lemma B.4, $\mathcal{Q}_2 \otimes \mathcal{P}_I \otimes \mathcal{P}_F \preceq \mathcal{Q}_2' \otimes \mathcal{P}_F \preceq \nu_2$, so we have shown that:

$$\forall \mu_2.\ \mathcal{P}_2 \otimes \mathcal{P}_I \otimes \mathcal{P}_F \preceq \mu_2 \implies \forall \nu_2.\ \mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}(\llbracket C_2 \rrbracket)^\dagger(\mu_2).\ \mathcal{Q}_2 \otimes \mathcal{P}_I \otimes \mathcal{P}_F \preceq \nu_2$$

Now, by Lemma C.6, $\mathcal{Q}_1 \otimes \mathcal{Q}_2 \otimes \mathcal{P}_I \otimes \mathcal{P}_F \preceq \nu$ for all $\nu \in \mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}(\llbracket C_1 \rrbracket \parallel \llbracket C_2 \rrbracket)^\dagger(\mu)$. Let $\mathcal{Q} = \mathcal{Q}_1 \otimes \mathcal{Q}_2 \otimes \mathcal{P}_I$. So $\mathcal{Q} \otimes \mathcal{P}_F \preceq \nu$ and since $\Gamma, \mathcal{Q}_1 \vDash \psi_1$ and $\Gamma, \mathcal{Q}_2 \vDash \psi_2$, we get that $\Gamma, \mathcal{Q} \vDash \psi_1 * \psi_2 * \lceil I \rceil$, so we are done.

- Atom.

$$\frac{J \vdash_m \langle \varphi * \lceil I \rceil \rangle\ a\ \langle \psi * \lceil I \rceil \rangle}{I * J \vdash_m \langle \varphi \rangle\ a\ \langle \psi \rangle}\textsc{Atom}$$

Suppose that $\mathcal{P}' \in \mathcal{P} \diamond_m \mathcal{P}_F$, $\mathcal{P}' \preceq \mu$, and $\Gamma, \mathcal{P} \vDash \varphi * \lceil I * J \rceil$. This means that there exist $\mathcal{P}_1$ and $\mathcal{P}_2$ such that $\mathcal{P}_1 \otimes \mathcal{P}_2 \otimes \mathcal{P}_3 \preceq \mathcal{P}$ and $\Gamma, \mathcal{P}_1 \vDash \varphi$ and $\Gamma, \mathcal{P}_2 \vDash \lceil I * J \rceil$. Let $\mathcal{P}_I$ and $\mathcal{P}_J$ be the trivial probability spaces that satisfy $\lceil I \rceil$ and $\lceil J \rceil$, respectively, so clearly $\mathcal{P}_I \otimes \mathcal{P}_J \preceq \mathcal{P}_2$. Now, take any $\nu \in \mathcal{L}^{\langle\!| I * J |\!\rangle_\Gamma}(\llbracket a \rrbracket)^\dagger(\mu)$ and note that:

$$\begin{aligned}
\mathcal{L}^{\langle\!| I * J |\!\rangle_\Gamma}(\llbracket a \rrbracket)^\dagger(\mu) &= (\llbracket a \rrbracket_{\mathsf{Act}}^{\langle\!| I * J |\!\rangle_\Gamma})^\dagger(\mu) \\
&= \left(\left(\mathsf{check}^{\langle\!| I * J |\!\rangle_\Gamma}\right)^\dagger \circ \llbracket a \rrbracket_{\mathsf{Act}}^\dagger \circ \left(\mathsf{replace}^{\langle\!| I * J |\!\rangle_\Gamma}\right)^\dagger \circ \left(\mathsf{check}^{\langle\!| I * J |\!\rangle_\Gamma}\right)^\dagger\right)(\mu) \\
&= \left(\mathsf{check}^{\langle\!| I |\!\rangle_\Gamma * \langle\!| J |\!\rangle_\Gamma}\right)^\dagger \left(\llbracket a \rrbracket_{\mathsf{Act}}^\dagger \left(\left(\mathsf{replace}^{\langle\!| I |\!\rangle_\Gamma * \langle\!| J |\!\rangle_\Gamma}\right)^\dagger \left(\left(\mathsf{check}^{\langle\!| I |\!\rangle_\Gamma * \langle\!| J |\!\rangle_\Gamma}\right)^\dagger(\mu)\right)\right)\right) \\
&= \left(\mathsf{check}^{\langle\!| I |\!\rangle_\Gamma}\right)^\dagger \left(\mathcal{L}^{\langle\!| J |\!\rangle_\Gamma}(\llbracket a \rrbracket)^\dagger \left(\left(\mathsf{replace}^{\langle\!| I |\!\rangle_\Gamma}\right)^\dagger \left(\left(\mathsf{check}^{\langle\!| I |\!\rangle_\Gamma}\right)^\dagger(\mu)\right)\right)\right)
\end{aligned}$$

Since we already assumed that μ satisfies $I * J$, the first check does nothing.

$$= \left(\mathsf{check}^{\langle\!| I |\!\rangle_\Gamma}\right)^\dagger \left(\mathcal{L}^{\langle\!| J |\!\rangle_\Gamma}(\llbracket a \rrbracket)^\dagger \left(\left(\mathsf{replace}^{\langle\!| I |\!\rangle_\Gamma}\right)^\dagger(\mu)\right)\right)$$

Therefore, we know that $\nu = (\mathsf{check}^{\langle\!| I |\!\rangle_\Gamma})^\dagger(\nu')$ for some $\nu' \in \mathcal{L}^{\langle\!| J |\!\rangle_\Gamma}(\llbracket a \rrbracket)^\dagger(\mu')$ and $\mu' \in (\mathsf{replace}^{\langle\!| I |\!\rangle_\Gamma})^\dagger(\mu))$. We know that $\mathcal{P}_1 \otimes \mathcal{P}_I \otimes \mathcal{P}_J \preceq \mathcal{P}_1 \otimes \mathcal{P}_2 \preceq \mathcal{P}$, so there must be a $\mathcal{P}'' \in (\mathcal{P}_1 \otimes \mathcal{P}_I \otimes \mathcal{P}_J) \diamond_m \mathcal{P}_F$ such that $\mathcal{P}'' \preceq \mathcal{P}'$ and therefore $\mathcal{P}'' \preceq \mu$. Since $\mathcal{P}''$ only contains trivial information about I, and μ and μ' differ only in the states that satisfy I, then $\mathcal{P}'' \preceq \mu'$ too. Therefore, by the induction hypothesis, there must be $\mathcal{Q}$ and $\mathcal{Q}' \in \mathcal{Q} \diamond_m \mathcal{P}_F$ such that $\mathcal{Q}' \preceq \nu'$ and $\Gamma, \mathcal{Q} \vDash \psi * \lceil I \rceil * \lceil J \rceil$. This implies that $\Gamma, \nu' \vDash \lceil I \rceil$, and therefore $\nu = \nu'$, since the check operation does nothing. Therefore, we get that $\mathcal{Q}' \preceq \nu' = \nu$, and so we are done.

- Share.

$$\frac{I * J \vdash_m \langle \varphi \rangle\ C\ \langle \psi \rangle \qquad \mathsf{finitary}(I)}{J \vdash_m \langle \varphi * \lceil I \rceil \rangle\ C\ \langle \psi * \lceil I \rceil \rangle}\textsc{Share}$$

Suppose that $\mathcal{P}' \in \mathcal{P} \diamond_m \mathcal{P}_F$, $\mathcal{P}' \preceq \mu$, and $\Gamma, \mathcal{P} \vDash \varphi * \lceil I \rceil * \lceil J \rceil$. Now, take any $\nu \in \mathcal{L}^{\langle\!| J |\!\rangle_\Gamma}(\llbracket C \rrbracket)^\dagger(\mu)$. By Lemma 5.3, we know that:

$$\mathcal{L}^{\langle\!| I * J |\!\rangle_\Gamma}(\llbracket C \rrbracket)^\dagger(\mu) = \mathcal{L}^{\langle\!| I |\!\rangle_\Gamma * \langle\!| J |\!\rangle_\Gamma}(\llbracket C \rrbracket)^\dagger(\mu) \sqsubseteq_C \mathcal{L}^{\langle\!| J |\!\rangle_\Gamma}(\llbracket C \rrbracket)^\dagger(\mu)$$

And since $\sqsubseteq_C$ is equivalent to $\supseteq$, then $\nu \in \mathcal{L}^{\langle\!| I * J |\!\rangle_\Gamma}([\![C]\!])^\dagger(\mu)$ as well. Therefore, by the induction hypothesis, we know that there exist $\mathcal{Q}$ and $\mathcal{Q}' \in \mathcal{Q} \diamond_m \mathcal{P}_F$ such that $\mathcal{Q}' \preceq \nu$ and $\Gamma, \mathcal{Q} \vDash \psi * \lceil I * J \rceil$. Since $\lceil I * J \rceil \Leftrightarrow \lceil I \rceil * \lceil J \rceil$, we are done.

- Frame.

$$\frac{I \vdash_m \langle \varphi \rangle\ C\ \langle \psi \rangle}{I \vdash_m \langle \varphi *_m \vartheta \rangle\ C\ \langle \psi *_m \vartheta \rangle}\textsc{Frame}$$

Suppose that $\mathcal{P}' \in \mathcal{P} \diamond_m \mathcal{P}_F$ and $\mathcal{P}' \preceq \mu$ and $\Gamma, \mathcal{P} \vDash (\varphi *_m \vartheta) * \lceil I \rceil$. That means that there exist $\mathcal{P}_1$ and $\mathcal{P}_I$ such that $\mathcal{P}_1 \otimes \mathcal{P}_I \preceq \mathcal{P}$ and $\Gamma, \mathcal{P}_1 \vDash \varphi *_m \vartheta$, and $\Gamma, \mathcal{P}_I \vDash \lceil I \rceil$, and without loss of generality, suppose that $\mathcal{P}_I$ is the trivial probability space that satisfies I. We also know that $\mathcal{P}_1 \in \mathcal{P}_2 \diamond_m \mathcal{P}_3$ where $\Gamma, \mathcal{P}_2 \vDash \varphi$ and $\Gamma, \mathcal{P}_3 \vDash \vartheta$. Since $\mathcal{P}_I$ is a trivial probability space, then $\mathcal{P}_2 \diamond_m \mathcal{P}_I = \{\mathcal{P}_2 \otimes \mathcal{P}_I\}$, so by associativity of projections, we get that there must be a $\mathcal{P}'_F \in \mathcal{P}_F \diamond_m \mathcal{P}_3$ such that $\mathcal{P}' \in (\mathcal{P}_2 \otimes \mathcal{P}_I) \diamond_m \mathcal{P}'_F$.

Now, take any $\nu \in \mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}([\![C]\!])^\dagger(\mu)$. By the induction hypothesis, there exist $\mathcal{Q}$ and $\mathcal{Q}' \in \mathcal{Q} \diamond_m \mathcal{P}'_F$ such that $\mathcal{Q}' \preceq \nu$ and $\mathcal{Q} \vDash \psi * \lceil I \rceil$. So, there must be a $\mathcal{Q}'' \in \mathcal{Q} \diamond_m \mathcal{P}_3$ and $\mathcal{Q}' \in \mathcal{Q}'' \diamond_m \mathcal{P}_F$, and by constructions $\Gamma, \mathcal{Q}'' \vDash (\psi * \lceil I \rceil) *_m \vartheta$. Since $\lceil I \rceil$ is a pure assertion, this also means that $\Gamma, \mathcal{Q}'' \vDash (\psi *_m \vartheta) * \lceil I \rceil$.

- Weaken.

$$\frac{I \vdash \langle \varphi \rangle\ C\ \langle \psi \rangle}{I \vdash_{\mathsf{w}} \langle \varphi \rangle\ C\ \langle \psi \rangle}\textsc{Weaken}$$

Immediate from the induction hypothesis, letting $\mathcal{P}_F$ be the trivial probability space on $\mathsf{Mem}[\emptyset]$, and Lemma E.9.

- Strengthen.

$$\frac{I \vdash_{\mathsf{w}} \langle \varphi \rangle\ C\ \langle \psi \rangle \qquad \mathsf{precise}(\psi)}{I \vdash \langle \varphi \rangle\ C\ \langle \psi \rangle}\textsc{Strengthen}$$

Suppose that $\mathcal{P} \otimes \mathcal{P}_F \preceq \mu$, and $\Gamma, \mathcal{P} \vDash \varphi * \lceil I \rceil$. This means that there exists a $\mathcal{P}'$ such that $\Gamma, \mathcal{P}' \vDash \varphi$ and $\Gamma, \mathcal{P}_I \vDash \lceil I \rceil$ where $\mathcal{P}_I$ is the trivial probability space satisfying $\lceil I \rceil$ and $\mathcal{P}' \otimes \mathcal{P}_I \preceq \mathcal{P}$. Since ψ is precise, let $\mathcal{Q}$ be the unique smallest probability space such that $\Gamma, \mathcal{Q} \vDash \psi$. From the premise of the rule, we know that:

$$\forall \mu_1.\ \mathcal{P}' \otimes \mathcal{P}_I \preceq \mu_1 \implies \forall \nu_1 \in \mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}([\![C]\!])^\dagger(\mu_1).\ \mathcal{Q} \otimes \mathcal{P}_I \preceq \nu_1$$

In addition, since $\mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}([\![\mathbf{skip}]\!]) = \eta$ [Zilberstein et al. 2025a, Lemma 5.2] it is obvious that:

$$\forall \mu_2.\ \mathcal{P}_F \otimes \mathcal{P}_I \preceq \mu_2 \implies \forall \nu_2 \in \mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}([\![\mathbf{skip}]\!])^\dagger(\mu_2).\ \mathcal{P}_F \otimes \mathcal{P}_I \preceq \nu_2$$

So, by Lemma C.6, we get that $\mathcal{Q} \otimes \mathcal{P}_F \otimes \mathcal{P}_I \preceq \nu$ for any $\nu \in \mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}([\![C]\!] \parallel [\![\mathbf{skip}]\!])^\dagger(\mu)$. But clearly $\mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}([\![C]\!] \parallel [\![\mathbf{skip}]\!]) = \mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}([\![C]\!])$, so we are done.

- Split1.

$$\frac{I \vdash_m \langle \varphi \rangle\ C\ \langle \psi \rangle \qquad \psi \Rightarrow \lceil e \mapsto X \rceil \qquad X \notin \mathsf{fv}(I)}{I \vdash_m \langle \bigoplus_{X \sim d(E)} \varphi \rangle\ C\ \langle \bigoplus_{X \sim d(E)} \psi \rangle}\textsc{Split1}$$

We first prove the claim for the case where $m = \mathsf{s}$. Suppose that $\mathcal{P} \otimes \mathcal{P}_F \preceq \mu$ and $\Gamma, \mathcal{P} \vDash (\bigoplus_{X \sim d(E)} \varphi) * \lceil I \rceil$. This means that $(\bigoplus_{v \sim \xi} \mathcal{P}_v) \otimes \mathcal{P}_I \preceq \mathcal{P}$ such that $\Gamma[X \coloneqq v], \mathsf{ext}(\mathcal{P}_v) \vDash \varphi$ for each v and $\xi = d([\![E]\!]_{\mathsf{LExp}}(\Gamma))$ and $\mathcal{P}_I$ is the trivial probability space satisfying $\lceil I \rceil$. By Lemmas B.3 and B.4 we have:

$$\bigoplus_{v \sim \xi}(\mathcal{P}_v \otimes \mathcal{P}_I \otimes \mathcal{P}_F) = \left(\bigoplus_{v \sim \xi} \mathcal{P}_v\right) \otimes \mathcal{P}_I \otimes \mathcal{P}_F \preceq \mathcal{P} \otimes \mathcal{P}_F \preceq \mu$$

Now, for each $v \in \mathsf{supp}(\xi)$, let $\mu_v(\sigma) \triangleq \frac{1}{\xi(v)} \cdot \mu(\sigma)$ if $\sigma \in \Omega_{\mathcal{P}_v} * \Omega_{\mathcal{P}_I} * \Omega_{\mathcal{P}_F}$, which clearly gives us $\mathcal{P}_v \otimes \mathcal{P}_I \otimes \mathcal{P}_F \preceq \mu_v$ and $\mu = \sum_{v \in \mathsf{supp}(\xi)} \xi(v) \cdot \mu_v$. Now, take any $\nu \in \mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}([\![C]\!])^\dagger(\mu)$. We know that:

$$\begin{aligned}\mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}([\![C]\!])^\dagger(\mu) &= \mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}([\![C]\!])^\dagger\Big(\sum_{v \in \mathsf{supp}(\xi)} \nu(v) \cdot \mu_v\Big) \\ &= \left\{ \sum_{v \in \mathsf{supp}(\xi)} \xi(v) \cdot \nu_v \;\middle|\; \forall v.\ \nu_v \in \mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}([\![C]\!])^\dagger(\mu_v) \right\}\end{aligned}$$

So $\nu = \sum_{v \in \mathsf{supp}(\xi)} \xi(v) \cdot \nu_v$ where $\nu_v \in \mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}([\![C]\!])^\dagger(\mu_v)$ for each v. Note that $X \notin \mathsf{fv}(I)$, so $\langle\!| I |\!\rangle_\Gamma = \langle\!| I |\!\rangle_{\Gamma[X:=v]}$, and therefore by the induction hypothesis, we get that there exists $\mathcal{Q}_v$ such that $\mathcal{Q}_v \otimes \mathcal{P}_F \preceq \nu_v$ and $\Gamma[X \coloneqq v], \mathcal{Q}_v \vDash \psi * \lceil I \rceil$. Since $\psi \Rightarrow \lceil e = X \rceil$, we can restrict the $\mathcal{Q}_v$s to disjoint probability spaces $\mathcal{Q}'_v$ such that $\mathsf{ext}(\mathcal{Q}'_v) \otimes \mathcal{P}_I \preceq \mathcal{Q}_v$.

Therefore $\bigoplus_{v\sim\nu} \mathcal{Q}'_v$ exists and $\Gamma, \bigoplus_{v\sim\xi} \mathcal{Q}'_v \vDash \bigoplus_{v\sim d(E)} \psi$. So, we get that:

$$\left(\bigoplus_{v\sim\xi} \mathcal{Q}'_v\right) \otimes \mathcal{P}_I \otimes \mathcal{P}_F = \bigoplus_{v\sim\xi} \left(\mathcal{Q}'_v \otimes \mathcal{P}_I \otimes \mathcal{P}_F\right) \preceq \sum_{v\in\mathrm{supp}(\xi)} \xi(v) \cdot \nu_v = \nu$$

The case where $m = \mathsf{w}$ follows immediately from Lemma E.9 using the proof above with $\mathcal{P}_F$ being the trivial probability space on empty memories.

- NSplit1.

$$\frac{I \vdash_m \langle \varphi \rangle\ C\ \langle \psi \rangle \qquad \psi \Rightarrow \lceil e \mapsto X \rceil \qquad X \notin \mathsf{fv}(I)}{I \vdash_m \langle \mathop{\&}_{X\in E} \varphi \rangle\ C\ \langle \mathop{\&}_{X\in E} \psi \rangle}\ \textsc{NSplit1}$$

Suppose that $\mathcal{P}' \in \mathcal{P} \diamond_m \mathcal{P}_F$ and $\mathcal{P}' \preceq \mu$ and $\Gamma, \mathcal{P} \vDash (\&_{X\in E}\, \varphi) * \lceil I \rceil$. This means that there is some $\xi \in \mathcal{D}(\llbracket E \rrbracket_{\mathsf{LExp}}(\Gamma))$ such that $\Gamma, \mathcal{P} \vDash (\bigoplus_{X\sim\xi} \varphi) * \lceil I \rceil$. Now take any $\nu \in \mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}(\llbracket C \rrbracket)^\dagger(\mu)$. Using the Split1 rule, we get that there exists a $\mathcal{Q}$ and $\mathcal{Q}' \in \mathcal{Q} \diamond_m \mathcal{P}_F$ such that $\mathcal{Q}' \preceq \nu$ and $\Gamma, \mathcal{Q} \vDash (\bigoplus_{X\sim\xi} \psi) * \lceil I \rceil$. This immediately implies that $\Gamma, \mathcal{Q} \vDash (\&_{X\in E}\, \psi) * \lceil I \rceil$.

- Split2.

$$\frac{I \vdash_m \langle \varphi \rangle\ C\ \langle \psi \rangle \qquad \mathsf{convex}(\psi) \qquad X \notin \mathsf{vars}(I, \psi)}{I \vdash_m \langle \bigoplus_{X\sim d(E)} \varphi \rangle\ C\ \langle \psi \rangle}\ \textsc{Split2}$$

We start with the case where $m = \mathsf{s}$. Suppose that $\mathcal{P} \otimes \mathcal{P}_F \preceq \mu$ and $\Gamma, \mathcal{P} \vDash (\bigoplus_{X\sim d(E)} \varphi) * \lceil I \rceil$ and let $\xi = d(\llbracket E \rrbracket_{\mathsf{LExp}}(\Gamma))$. Since $X \notin \mathsf{vars}(I)$, then $\Gamma, \mathcal{P} \vDash \bigoplus_{X\sim d(E)} \varphi * \lceil I \rceil$ and so there exists a family of $\mathcal{P}_v$ such that $\bigoplus_{v\sim\xi} \mathcal{P}_v \preceq \mathcal{P}$ and $\Gamma[X \coloneqq v], \mathsf{ext}(\mathcal{P}_v) \vDash \varphi * \lceil I \rceil$ for all $v \in \mathrm{supp}(\xi)$. By Lemmas B.3 and B.5, we also have that:

$$\bigoplus_{v\sim\xi} (\mathcal{P}_v \otimes \mathcal{P}_F) = \left(\bigoplus_{v\sim\xi} \mathcal{P}_v\right) \otimes \mathcal{P}_F \preceq \mathcal{P} \otimes \mathcal{P}_F \preceq \mu$$

Now, for each $v \in \mathrm{supp}(\xi)$, let $\mu_v(\sigma) \triangleq \frac{1}{\xi(v)} \cdot \mu(\sigma)$ if $\sigma \in \Omega_{\mathcal{P}_v}$, which clearly gives us $\mathsf{ext}(\mathcal{P}_v) \otimes \mathcal{P}_F \preceq \mu_v$ and $\mu = \sum_{v\in\mathrm{supp}(\xi)} \xi(v) \cdot \mu_v$. Now, take any $\nu \in \mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}(\llbracket C \rrbracket)^\dagger(\mu)$. We know that:

$$\begin{aligned}\mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}(\llbracket C \rrbracket)^\dagger(\mu) &= \mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}(\llbracket C \rrbracket)^\dagger\Big(\sum_{v\in\mathrm{supp}(\xi)} \nu(v) \cdot \mu_v\Big) \\ &= \left\{ \sum_{v\in\mathrm{supp}(\xi)} \xi(v) \cdot \nu_v \;\middle|\; \forall v.\ \nu_v \in \mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}(\llbracket C \rrbracket)^\dagger(\mu_v) \right\}\end{aligned}$$

So $\nu = \sum_{v\in\mathrm{supp}(\xi)} \xi(v) \cdot \nu_v$ where $\nu_v \in \mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}(\llbracket C \rrbracket)^\dagger(\mu_v)$ for each v. Note that $X \notin \mathsf{vars}(I)$, so $\langle\!| I |\!\rangle_\Gamma = \langle\!| I |\!\rangle_{\Gamma[X:=v]}$, and therefore by the induction hypothesis, we get that there exist a family of $\mathcal{Q}_v$ such $\mathcal{Q}_v \otimes \mathcal{P}_F \preceq \nu_v$ and $\Gamma[X \coloneqq v], \mathcal{Q}_v \vDash \psi * \lceil I \rceil$. Since $X \notin \mathsf{vars}(\psi)$, we can remove the update of X to conclude that $\Gamma, \mathcal{Q}_v \vDash \psi * \lceil I \rceil$. Since ψ is convex, then we can presume that there exists U, $\mathcal{F}$, and S such that $\mathcal{Q}_v = \langle \mathsf{Mem}[U], \mathcal{F}, \xi_v \rangle$ for some $\xi_v \in S$. Now, let $\mathcal{Q} = \langle \mathsf{Mem}[U], \mathcal{F}, \sum_{v\in\mathrm{supp}(\xi)} \xi(v) \cdot \xi_v \rangle$. Since $\sum_{v\in\mathrm{supp}(\xi)} \xi(v) \cdot \xi_v$ is a convex combinations of elements of S, then $\sum_{v\in\mathrm{supp}(\xi)} \xi(v) \cdot \xi_v \in S$ and therefore by construction $\Gamma, \mathcal{Q} \vDash \psi * \lceil I \rceil$. We now complete the proof by showing that $\mathcal{Q} \otimes \mathcal{P}_F \preceq \nu$. Take any $A \in \mathcal{F}_{\mathcal{Q}}$ and $B \in \mathcal{F}_{\mathcal{P}_F}$, and let V be the set such that $\Omega_{\mathcal{P}_F} = \mathsf{Mem}[V]$:

$$\begin{aligned}\mu_{\mathcal{Q}\otimes\mathcal{P}_F}(A * B) &= \mu_{\mathcal{Q}}(A) \cdot \mu_{\mathcal{P}_F}(B) \\ &= \left(\sum_{v\in\mathrm{supp}(\xi)} \xi(v) \cdot \xi_v(A)\right) \cdot \mu_{\mathcal{P}_F}(B) \\ &= \sum_{v\in\mathrm{supp}(\xi)} \xi(v) \cdot \xi_v(A) \cdot \mu_{\mathcal{P}_F}(B) \\ &= \sum_{v\in\mathrm{supp}(\xi)} \xi(v) \cdot \mu_{\mathcal{Q}_v}(A) \cdot \mu_{\mathcal{P}_F}(B)\end{aligned}$$

Since $\mathcal{Q}_v \otimes \mathcal{P}_F \preceq \nu_v$:

$$\begin{aligned}&= \sum_{v\in\mathrm{supp}(\xi)} \xi(v) \cdot \pi_{U\cup V}(\nu_v)(A * B) \\ &= \pi_{U\cup V}(\nu)(A)\end{aligned}$$

The case where $m = \mathsf{w}$ follows from Lemma E.9 using the same reasoning as above, but where $\mathcal{P}_F$ is the trivial probability space on $\mathsf{Mem}[\emptyset]$.

- NSplit2.

$$\frac{I \vdash_m \langle \varphi \rangle\ C\ \langle \psi \rangle \qquad \mathsf{convex}(\psi) \qquad X \notin \mathsf{fv}(I, \psi)}{I \vdash_m \langle \mathop{\&}_{X \in E} \varphi \rangle\ C\ \langle \psi \rangle}\ \textsc{NSplit2}$$

Suppose that $\mathcal{P}' \in \mathcal{P} \diamond_m \mathcal{P}_F$ and $\mathcal{P}' \preceq \mu$ and $\Gamma, \mathcal{P} \vDash (\mathop{\&}_{X \in E} \varphi) * \lceil I \rceil$. This means that there is some $\xi \in \mathcal{D}(\llbracket E \rrbracket_{\mathsf{LExp}}(\Gamma))$ such that $\Gamma, \mathcal{P} \vDash (\bigoplus_{X \sim \xi} \varphi) * \lceil I \rceil$. Now take any $\nu \in \mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}(\llbracket C \rrbracket)^\dagger(\mu)$. Using the Split2 rule, we get that there exists a $\mathcal{Q}$ and $\mathcal{Q}' \in \mathcal{Q} \diamond_m \mathcal{P}_F$ such that $\mathcal{Q}' \preceq \nu$ and $\Gamma, \mathcal{Q} \vDash \psi * \lceil I \rceil$.

- Exists.

$$\frac{I \vdash_{\mathsf{w}} \langle \mathop{\&}_{X \in E} \lceil P \rceil \rangle\ C\ \langle \psi \rangle \qquad \lceil P \rceil \Rightarrow \lceil e \mapsto X \rceil}{I \vdash_{\mathsf{w}} \langle \lceil \exists X \in E.\ P \rceil \rangle\ C\ \langle \psi \rangle}\ \textsc{Exists}$$

By Lemma E.9, it suffices to show that the claim holds without frame preservation. Suppose that $\Gamma, \mu \vDash \lceil \exists X \in E. P \rceil * \lceil I \rceil$. Without loss of generality, suppose that X is not free in I (if not, then we could α-rename X in P to obtain an equivalent assertion, *i.e.,* $\lceil \exists X \in E.\ P \rceil \Leftrightarrow \lceil \exists Y \in E.\ P[Y/X] \rceil$ where Y is some fresh variable). So, we get that $\Gamma, \mu \vDash \lceil \exists X \in E.\ P * I \rceil$, which means that for every $\sigma \in \mathsf{supp}(\mu)$ there exists $v_\sigma \in \llbracket E \rrbracket_{\mathsf{LExp}}(\Gamma)$ such that $\Gamma[X \coloneqq v_\sigma], \sigma \vDash P * I$. Let $\xi \triangleq \sum_{\sigma \in \mathsf{supp}(\mu)} \mu(\sigma) \cdot \delta_{v_\sigma}$, so $\xi(v)$ is the probability that $\Gamma[X \coloneqq v]$ is the context that satisfies P. Also, let:

$$\mu_v(\sigma) \triangleq \frac{1}{\xi(v)} \begin{cases} \mu(\sigma) & \text{if } v = v_\sigma \\ 0 & \text{if } v \neq v_\sigma \end{cases}$$

Clearly, by construction $\Gamma[X \coloneqq v], \mu_v \vDash \lceil P \rceil * \lceil I \rceil$ for all $v \in \mathsf{supp}(\xi)$. Also, since $\lceil P \rceil \Rightarrow \lceil e \mapsto X \rceil$, then $\llbracket e \rrbracket_{\mathsf{Exp}}(\sigma) = v$ for each $\sigma \in \mathsf{supp}(\mu_v)$, and so each $\mu_v = \mathsf{ext}(\mu'_v)$ where μ'_v has a restricted sample space to only be over states where e evaluates to v. Thus, we clearly have $\bigoplus_{v \sim \xi} \mu'_v \preceq \mu$, and therefore $\Gamma, \mu \vDash (\mathop{\&}_{X \in E} \lceil P \rceil) * \lceil I \rceil$. So, by the induction hypothesis and Lemma E.9, $\Gamma, \nu \vDash \psi * \lceil I \rceil$ for all $\nu \in \mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}(\llbracket C \rrbracket)^\dagger(\mu)$.

- Consequence.

$$\frac{\varphi' \Rightarrow \varphi \qquad I \vdash_m \langle \varphi \rangle\ C\ \langle \psi \rangle \qquad \psi \Rightarrow \psi'}{I \vdash_m \langle \varphi' \rangle\ C\ \langle \psi' \rangle}\ \textsc{Consequence}$$

Suppose that $\mathcal{P}' \in (\mathcal{P} \otimes \mathcal{P}_{\langle\!| I |\!\rangle_\Gamma}) \diamond_m \mathcal{P}_F$, $\mathcal{P}' \preceq \mu$, and $\Gamma, \mathcal{P} \vDash \varphi'$. Since $\varphi' \Rightarrow \varphi$, then $\Gamma, \mathcal{P} \vDash \varphi$. By the induction hypothesis, there exists $\mathcal{Q}$ and $\mathcal{Q}' \in (\mathcal{Q} \otimes \mathcal{P}_{\langle\!| I |\!\rangle_\Gamma}) \diamond_m \mathcal{P}_F$ such that $\mathcal{Q}' \preceq \nu$ and $\Gamma, \mathcal{Q} \vDash \psi$ for any $\nu \in \mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}(\llbracket C \rrbracket)^\dagger(\mu)$. Since $\psi \Rightarrow \psi'$, then $\Gamma, \mathcal{Q} \vDash \psi'$, so we are done.

- BoundedRank. Subject to the following conditions:

(1) $\varphi \Rightarrow \lceil \ell \leq R \leq h \rceil$ (2) $\varphi * \lceil R = \ell \rceil \Rightarrow \lceil b \mapsto \mathsf{false} \rceil$ (3) $\varphi * \lceil R > \ell \rceil \Rightarrow \lceil b \mapsto \mathsf{true} \rceil$
(4) $\mathsf{precise}(\varphi[\ell/R])$ (5) $N \notin \mathsf{vars}(\varphi)$ (6) $0 < p \leq 1$

The following inference is valid:

$$\frac{I \vdash_m \langle \varphi * \lceil R = N > \ell \rceil \rangle\ C\ \langle (\mathop{\&}_{R=\ell}^{N-1} \varphi) \oplus_{\geq p} (\mathop{\&}_{R=N}^{h} \varphi) \rangle}{I \vdash_m \langle \mathop{\&}_{R=\ell}^{h} \varphi \rangle\ \mathbf{while}\ b\ \mathbf{do}\ C\ \langle \varphi[\ell/R] \rangle}\ \textsc{BoundedRank}$$

We prove the rule without frame preservation, as frame preservation follows from Lemma E.9 if $m = \mathsf{w}$ and the Strengthen rule if $m = \mathsf{s}$ (since the postcondition is precise). Suppose that $\Gamma, \mu \vDash \mathop{\&}_{R=\ell}^{h} \varphi * \lceil I \rceil$. Let $\varphi' = \mathop{\&}_{R=\ell+1}^{h} \varphi$ and $\psi = \varphi[\ell/R]$. We first show that $\langle \varphi', \psi \rangle$ is an invariant pair for $\mathbf{while}\ b\ \mathbf{do}\ C$ under I:

(1) $\varphi' = \mathop{\&}_{R=\ell+1}^{h} \varphi \Rightarrow \mathop{\&}_{R=\ell+1}^{h} \lceil b \mapsto \mathsf{true} \rceil \Rightarrow \lceil b \mapsto \mathsf{true} \rceil$
(2) $\psi = \varphi[\ell/R] \Rightarrow \lceil b \mapsto \mathsf{false} \rceil$
(3) We can weaken the postcondition of the premise of rule as follows:

$$(\mathop{\&}_{R=\ell}^{N-1} \varphi) \oplus_{\geq p} (\mathop{\&}_{R=N}^{h} \varphi) \quad \Rightarrow \quad (\mathop{\&}_{R=\ell}^{N-1} \varphi)\ \&\ (\mathop{\&}_{R=N}^{h} \varphi) \quad \Rightarrow \quad \mathop{\&}_{R=\ell}^{h} \varphi \quad \Rightarrow \quad \varphi'\ \&\ \psi$$

So, after an application of NSplit2, we get that $I \vDash_m \langle \varphi' \rangle\ C\ \langle \varphi'\ \&\ \psi \rangle$. If $m = \mathsf{s}$, Weaken can be used to obtain $I \vDash_{\mathsf{w}} \langle \varphi' \rangle\ C\ \langle \varphi'\ \&\ \psi \rangle$.

(4) By assumption, we have $\mathsf{precise}(\psi)$.

In addition, the premise of the rule implies that each iteration, the rank decreases by at least 1 with probability at least p, so starting in any state satisfying φ, the loop will terminate with probability at least $p^{h-\ell} > 0$. Therefore by Corollary D.3, $\mathsf{minterm}(\mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}(\llbracket \mathbf{while}\ b\ \mathbf{do}\ C \rrbracket)^\dagger(\mu)) = 1$, *i.e.,* the loop almost surely terminates.

Since $\varphi[\ell/R]$ is precise, let $\mathcal{Q}$ be the unique smallest probability space such that $\Gamma, \mathcal{Q} \vDash \varphi[\ell/R]$. Take any event $B \in \mathcal{F}_\mathcal{Q}$. By Lemma D.2, we get:

$$\mathsf{minProb}\left(\mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}(\llbracket \mathbf{while}\ b\ \mathbf{do}\ C \rrbracket)^\dagger(\mu), B * \langle\!| I |\!\rangle_\Gamma\right) = \mathsf{minterm}\left(\mathcal{L}^{\langle\!| I |\!\rangle_\Gamma}(\llbracket \mathbf{while}\ b\ \mathbf{do}\ C \rrbracket)^\dagger(\mu)\right) \cdot \mu_\mathcal{Q}(B) = \mu_\mathcal{Q}(B)$$

So, $\mu_Q(B) \leq \nu(B * (\!|I|\!)_\Gamma)$ for any $\nu \in \mathcal{L}^{(\!|I|\!)_\Gamma}([\![\texttt{while } b \texttt{ do } C]\!])^\dagger(\mu)$. However, we already established that the loop almost surely terminates, so $\nu(\bot) = 0$, therefore $\nu(B * (\!|I|\!)_\Gamma) = 1 - \nu(\mathsf{Mem}[V] \setminus (B * (\!|I|\!)_\Gamma))$ (where V is the domain of ν), which gives us:

$$\begin{aligned}\mu_Q(B) &\leq \nu(B * (\!|I|\!)_\Gamma)\\ &= 1 - \nu(\mathsf{Mem}[V] \setminus (B * (\!|I|\!)_\Gamma))\\ &\leq 1 - \mu_Q(\mathsf{Mem}[V] \setminus (B * (\!|I|\!)_\Gamma))\\ &= 1 - (1 - \mu_Q(B * (\!|I|\!)_\Gamma))\\ &= \mu_Q(B)\end{aligned}$$

Therefore $\mu_Q(B) = \nu(B * (\!|I|\!)_\Gamma)$, and since this is true for any $B \in \mathcal{F}_Q$, then $Q \otimes \mathcal{P}_I \preceq \nu$ where $\mathcal{P}_I$ is the trivial probability space satisfying I. By definition, $\Gamma, Q \vDash \varphi[\ell/R]$, so we are done.

- Disj.

$$\frac{I \vdash_m \langle \varphi_1 \rangle\ C\ \langle \psi_1 \rangle \qquad I \vdash_m \langle \varphi_2 \rangle\ C\ \langle \psi_2 \rangle}{I \vdash_m \langle \varphi_1 \vee \varphi_2 \rangle\ C\ \langle \psi_1 \vee \psi_2 \rangle}\ \textsc{Disj}$$

Suppose that $\mathcal{P}' \in \mathcal{P} \diamond \mathcal{P}_F$ and $\mathcal{P}' \preceq \mu$ and $\Gamma, \mathcal{P} \vDash (\varphi_1 \vee \varphi_2) * \lceil I \rceil$. This means that $\mathcal{P}_1 \otimes \mathcal{P}_2 \preceq \mathcal{P}$ such that $\Gamma, \mathcal{P}_1 \vDash \varphi_1 \vee \varphi_2$ and $\Gamma, \mathcal{P}_2 \vDash \lceil I \rceil$. Without loss of generality, suppose that $\Gamma, \mathcal{P}_1 \vDash \varphi_1$, therefore $\Gamma, \mathcal{P} \vDash \varphi_1 * \lceil I \rceil$. Now take any $\nu \in \mathcal{L}^{(\!|I|\!)_\Gamma}([\![C]\!])^\dagger(\mu)$. By the induction hypothesis, we know that there exists Q and $Q' \in Q \diamond_m \mathcal{P}_F$ such that $Q' \preceq \nu$ and $\Gamma, Q \vDash \psi_1 * \lceil I \rceil$. We can weaken this to conclude that $\Gamma, Q \vDash (\psi_1 \vee \psi_2) * \lceil I \rceil$.

- Exists2.

$$\frac{I \vdash_m \langle \varphi \rangle\ C\ \langle \psi \rangle \qquad X \notin \mathsf{vars}(\psi, I)}{I \vdash_m \langle \exists X.\ \varphi \rangle\ C\ \langle \psi \rangle}\ \textsc{Exists2}$$

Suppose that $\mathcal{P}' \in \mathcal{P} \diamond \mathcal{P}_F$ and $\mathcal{P}' \preceq \mu$ and $\Gamma, \mathcal{P} \vDash (\exists X.\varphi) * \lceil I \rceil$. This means that $\mathcal{P}_1 \otimes \mathcal{P}_2 \preceq \mathcal{P}$ such that $\Gamma, \mathcal{P}_1 \vDash \exists X.\varphi$ and $\Gamma, \mathcal{P}_2 \vDash \lceil I \rceil$. Therefore, $\Gamma[X \coloneqq v], \mathcal{P}_1 \vDash \varphi$ for some $v \in \mathrm{Val}$. Since $X \notin \mathsf{vars}(I)$, then we also have that $\Gamma[X \coloneqq v], \mathcal{P} \vDash \varphi * \lceil I \rceil$. Now take any $\nu \in \mathcal{L}^{(\!|I|\!)_\Gamma}([\![C]\!])^\dagger(\mu)$. Since $I \notin \mathsf{vars}(I)$, then $(\!|I|\!)_{\Gamma[X:=v]} = (\!|I|\!)_\Gamma$, so $\nu \in \mathcal{L}^{(\!|I|\!)_{\Gamma[X:=v]}}([\![C]\!])^\dagger(\mu)$ as well. By the induction hypothesis, we know that there exists Q and $Q' \in Q \diamond_m \mathcal{P}_F$ such that $Q' \preceq \nu$ and $\Gamma[X \coloneqq v], Q \vDash \psi * \lceil I \rceil$. Since $X \notin \mathsf{vars}(\psi, I)$, then this implies that $\Gamma, Q \vDash \psi * \lceil I \rceil$.

- Subst

$$\frac{I \vdash_m \langle \varphi \rangle\ C\ \langle \psi \rangle}{I[E/X] \vdash_m \langle \varphi[E/X] \rangle\ C\ \langle \psi[E/X] \rangle}\ \textsc{Subst}$$

Suppose that $\mathcal{P}' \in \mathcal{P} \diamond_m \mathcal{P}_F$ such that $\mathcal{P}' \preceq \mu$ and $\Gamma, \mathcal{P} \vDash \varphi[E/X] * \lceil I[E/X] \rceil$. Let $\Gamma' = \Gamma[X \coloneqq [\![E]\!]_{\mathsf{LExp}}(\Gamma)]$, so by construction $\Gamma', \mathcal{P} \vDash \varphi * \lceil I \rceil$, and $(\!|I[E/X]|\!)_\Gamma = (\!|I|\!)_{\Gamma'}$. Now, take any $\nu' \in \mathcal{L}^{(\!|I[E/X]|\!)_\Gamma}([\![C]\!])^\dagger(\mu) = \mathcal{L}^{(\!|I|\!)_{\Gamma'}}([\![C]\!])^\dagger(\mu)$. By the induction hypothesis, we know that there exist Q' and $Q' \in Q \diamond_m \mathcal{P}_F$ such that $Q' \preceq \nu$ and $\Gamma', Q \vDash \psi$. This means that $\Gamma, Q \vDash \psi[E/X]$.

□

E.4 Derived Rules

Lemma E.10. *The following inference rule is derivable:*

$$\frac{I \vdash_m \langle \varphi * \lceil X = \mathrm{true} \rceil \rangle\ C_1\ \langle \psi \rangle \qquad I \vdash_m \langle \varphi * \lceil X = \mathrm{false} \rceil \rangle\ C_2\ \langle \psi \rangle \qquad \varphi \Rightarrow \lceil b \mapsto X \rceil \qquad \psi \Rightarrow \lceil b \mapsto X \rceil}{I \vdash_m \langle \bigoplus_{X \sim \mathrm{Ber}(p)} \varphi \rangle\ \texttt{if } b \texttt{ then } C_1 \texttt{ else } C_2\ \langle \bigoplus_{X \sim \mathrm{Ber}(p)} \psi \rangle}\ \textsc{If}$$

Proof. Note in the proof below that true = 1 and false = 0.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{I \vdash_m \langle \varphi * \lceil X = 1 \rceil \rangle\ C_1\ \langle \psi \rangle}\ \textsc{IfT}}{I \vdash_m \langle \varphi * \lceil X = 1 \rceil \rangle\ \texttt{if } b \texttt{ then } C_1 \texttt{ else } C_2\ \langle \psi \rangle} \qquad \dfrac{}{I \vdash_m \langle \varphi * \lceil X = 0 \rceil \rangle\ \texttt{if } b \texttt{ then } C_1 \texttt{ else } C_2\ \langle \psi \rangle}\ \textsc{IfF}}{I \vdash_m \langle (\varphi * \lceil X = 1 \rceil) \vee (\varphi * \lceil X = 0 \rceil) \rangle\ \texttt{if } b \texttt{ then } C_1 \texttt{ else } C_2\ \langle \psi \vee \psi \rangle}\ \textsc{Disj}}{I \vdash_m \langle \bigoplus_{X \sim \mathrm{Ber}(p)} (\varphi * \lceil X = 1 \rceil) \vee (\varphi * \lceil X = 0 \rceil) \rangle\ \texttt{if } b \texttt{ then } C_1 \texttt{ else } C_2\ \langle \bigoplus_{X \sim \mathrm{Ber}(p)} \psi \vee \psi \rangle}\ \textsc{Split1}}{I \vdash_m \langle \bigoplus_{X \sim \mathrm{Ber}(p)} \varphi \rangle\ \texttt{if } b \texttt{ then } C_1 \texttt{ else } C_2\ \langle \bigoplus_{X \sim \mathrm{Ber}(p)} \psi \rangle}\ \textsc{Consequence}$$

□

Lemma E.11. *The following inference rule is derivable:*

$$\frac{I \vdash_m \langle \lceil P \wedge b \mapsto 1 \rceil \rangle\ C_1\ \langle \lceil Q \rceil \rangle \qquad I \vdash_m \langle \lceil P \wedge b \mapsto 0 \rceil \rangle\ C_2\ \langle \lceil Q \rceil \rangle \qquad \lceil P \rceil \Rightarrow \lceil b \in \{0,1\} \rceil}{I \vdash_m \langle \lceil P \rceil \rangle\ \mathbf{if}\ b\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2\ \langle \lceil Q \rceil \rangle}\ \textit{IfPure}$$

Proof. We only show the derivation of the weak version. The strong version can be easily derived using Strengthen, since the postcondition is precise (*e.g.*, see Lemma E.12). Let X be some fresh logical variable such that $X \notin \mathrm{vars}(P, I)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{I \vdash_w \langle \lceil P \wedge b \mapsto 1 \rceil \rangle\ C_1\ \langle \lceil Q \rceil \rangle}\ \textsc{IfT}}{I \vdash_w \langle \lceil P \wedge b \mapsto 1 \rceil \rangle\ \mathbf{if}\ b\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2\ \langle \lceil Q \rceil \rangle} \qquad \vdots}{I \vdash_w \langle \lceil P \wedge b \mapsto X \wedge (X = 0 \vee X = 1) \rceil \rangle\ \mathbf{if}\ b\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2\ \langle \lceil Q \rceil \rangle}\ \textsc{Disj}}{I \vdash_w \langle \&_{X \in \{0,1\}} \lceil P \wedge b \mapsto X \wedge (X = 0 \vee X = 1) \rceil \rangle\ \mathbf{if}\ b\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2\ \langle \lceil Q \rceil \rangle}\ \textsc{NSplit2}}{I \vdash_w \langle \lceil \exists X.\ P \wedge b \mapsto X \wedge (X = 0 \vee X = 1) \rceil \rangle\ \mathbf{if}\ b\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2\ \langle \lceil Q \rceil \rangle}\ \textsc{Exists}}{I \vdash_w \langle \lceil P \rceil \rangle\ \mathbf{if}\ b\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2\ \langle \lceil Q \rceil \rangle}\ \textsc{Consequence}$$

□

Lemma E.12. *The following inference rule is derivable:*

$$\frac{I \vdash \langle \&_{X \in E} \lceil P \rceil \rangle\ C\ \langle \psi \rangle \qquad \lceil P \rceil \Rightarrow \lceil e \mapsto X \rceil \qquad \mathrm{precise}(\psi)}{I \vdash \langle \lceil \exists X \in E.\ P \rceil \rangle\ C\ \langle \psi \rangle}\ \textit{ExistsStrong}$$

Proof.

$$\dfrac{\dfrac{\dfrac{I \vdash \langle \&_{X \in E} \lceil P \rceil \rangle\ C\ \langle \psi \rangle}{I \vdash_w \langle \&_{X \in E} \lceil P \rceil \rangle\ C\ \langle \psi \rangle}\ \textsc{Weaken} \qquad \lceil P \rceil \Rightarrow \lceil e \mapsto X \rceil}{I \vdash_w \langle \lceil \exists X \in E.\ P \rceil \rangle\ C\ \langle \psi \rangle}\ \textsc{Exists} \qquad \mathrm{precise}(\psi)}{I \vdash \langle \lceil \exists X \in E.\ P \rceil \rangle\ C\ \langle \psi \rangle}\ \textsc{Strengthen}$$

□

F Examples

In this section, we provide derivations that were omitted in the main text. The content is rotated to take advantage of the full width of the page.

F.1 Conditional Independence Example

In this section, we show the derivation for the program that was introduced in Section 2. The program is repeated below:

$$z :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right) \fatsemi \left(x \coloneqq z \parallel y \coloneqq 1 - z\right)$$

The first step is to break up the sequential composition, and derive a specification for the sampling operation. This is simple using the Seq, Frame, and Samp rules. However, the derivation for the second part of the program is more difficult, and will be filled in shortly where the (★) appears.

$$\dfrac{\dfrac{\dfrac{}{\langle \mathsf{own}(z)\rangle\ z :\approx \mathbf{Ber}\,(1/2)\ \langle z \sim \mathbf{Ber}\,(1/2)\rangle}\ \textsc{Samp}}{\langle \mathsf{own}(x,y,z)\rangle\ z :\approx \mathbf{Ber}\,(1/2)\ \langle \mathsf{own}(x,y) * z \sim \mathbf{Ber}\,(1/2)\rangle}\ \textsc{Frame} \qquad (\bigstar)}{\langle \mathsf{own}(x,y,z)\rangle\ z :\approx \mathbf{Ber}\,(1/2) \fatsemi (x \coloneqq z \parallel y \coloneqq 1-z)\ \langle \bigoplus_{\mathbf{Ber}(1/2)} \lceil x \mapsto Z\rceil * \lceil y \mapsto 1-Z\rceil * \lceil z \mapsto Z\rceil\rangle}\ \textsc{Seq}$$

We now show the (★) derivation. The first step is to rearrange the precondition using the entailment laws from Figure 4 to bring the outcome conjunction to the outside, so that we can apply the Cond1 rule. Conditioning makes z deterministic, and therefore trivially independent from the rest of the state, so we can allocate an invariant with the Share rule. The rest of the proof is straightforward.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\langle \mathsf{own}(x) * \lceil z \mapsto Z\rceil\rangle\ x \coloneqq z\ \langle \lceil x \mapsto Z\rceil * \lceil z \mapsto Z\rceil\rangle}\ \textsc{Assign}}{z \mapsto Z \vdash \langle \mathsf{own}(x)\rangle\ x \coloneqq z\ \langle \lceil x \mapsto Z\rceil\rangle}\ \textsc{Atom} \quad \dfrac{}{\langle \mathsf{own}(y) * \lceil z \mapsto Z\rceil\rangle\ y \coloneqq 1-z\ \langle \lceil y \mapsto 1-Z\rceil * \lceil z \mapsto Z\rceil\rangle}\ \textsc{Assign}}{z \mapsto Z \vdash \langle \mathsf{own}(y)\rangle\ y \coloneqq 1-z\ \langle \lceil y \mapsto 1-Z\rceil\rangle}\ \textsc{Atom}}{z \mapsto Z \vdash \langle \mathsf{own}(x,y)\rangle\ x \coloneqq z \parallel y \coloneqq 1-z\ \langle \lceil x \mapsto Z\rceil * \lceil y \mapsto 1-Z\rceil\rangle}\ \textsc{Par}}{\langle \mathsf{own}(x,y) * \lceil z \mapsto Z\rceil\rangle\ x \coloneqq z \parallel y \coloneqq 1-z\ \langle \lceil x \mapsto Z\rceil * \lceil y \mapsto 1-Z\rceil * \lceil z \mapsto Z\rceil\rangle}\ \textsc{Share}}{\langle \bigoplus_{\mathbf{Ber}(1/2)} \mathsf{own}(x,y) * \lceil z \mapsto Z\rceil\rangle\ x \coloneqq z \parallel y \coloneqq 1-z\ \langle \bigoplus_{\mathbf{Ber}(1/2)} \lceil x \mapsto Z\rceil * \lceil y \mapsto 1-Z\rceil * \lceil z \mapsto Z\rceil\rangle}\ \textsc{Cond1}}{\langle \mathsf{own}(x,y) * z \sim \mathbf{Ber}\,(1/2)\rangle\ x \coloneqq z \parallel y \coloneqq 1-z\ \langle \bigoplus_{\mathbf{Ber}(1/2)} \lceil x \mapsto Z\rceil * \lceil y \mapsto 1-Z\rceil * \lceil z \mapsto Z\rceil\rangle}\ \textsc{Consequence}$$

F.2 Almost Sure Termination of a Random Walk

Recall the following random walk program from Section 5.4.

$$\mathtt{while}\ x > 0\ \mathtt{do}\ b :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right) \fatsemi \mathtt{if}\ b\ \mathtt{then}\ x \coloneqq x - 1\ \mathtt{else}\ x \coloneqq y$$

We now give the full derivation for this program. Below in (6), we derive a specification for the sampling operation.

$$\dfrac{\dfrac{\dfrac{}{y \in \{0,\dots,5\} \vdash \langle \lceil \mathsf{own}(b) \rceil \rangle\ b :\approx \mathbf{Ber}\left(\frac{1}{2}\right)\ \langle b \sim \mathbf{Ber}\left(\frac{1}{2}\right) \rangle}\ \textsc{Samp}}{y \in \{0,\dots,5\} \vdash \langle \lceil \mathsf{own}(b) \rceil * \lceil x \mapsto R * 0 < R = N \le 5 \rceil \rangle\ b :\approx \mathbf{Ber}\left(\frac{1}{2}\right)\ \langle b \sim \mathbf{Ber}\left(\frac{1}{2}\right) * \lceil x \mapsto R * 0 < R = N \le 5 \rceil \rangle}\ \textsc{Frame}}{y \in \{0,\dots,5\} \vdash \langle \lceil x \mapsto R * \mathsf{own}(b) * 0 < R = N \le 5 \rceil \rangle\ b :\approx \mathbf{Ber}\left(\frac{1}{2}\right)\ \langle \bigoplus_{X \sim \mathbf{Ber}\left(\frac{1}{2}\right)} \lceil b \mapsto X * x \mapsto R * 0 < R = N \le 5 \rceil \rangle}\ \textsc{Consequence}}{y \in \{0,\dots,5\} \vdash_{\mathsf{w}} \langle \lceil x \mapsto R * \mathsf{own}(b) * 0 < R = N \le 5 \rceil \rangle\ b :\approx \mathbf{Ber}\left(\frac{1}{2}\right)\ \langle \bigoplus_{X \sim \mathbf{Ber}\left(\frac{1}{2}\right)} \lceil b \mapsto X * x \mapsto R * 0 < R = N \le 5 \rceil \rangle}\ \textsc{Weaken} \tag{6}$$

Next, in (7), we show the 'false' branch of the if statement.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash_{\mathsf{w}} \langle \lceil b \mapsto \mathsf{false} * x \mapsto N * y \mapsto R \rceil \rangle\ x := y\ \langle \lceil x \mapsto R * b \mapsto \mathsf{false} * y \mapsto R \rceil \rangle}\ \textsc{Assign}}{\vdash_{\mathsf{w}} \langle \&_{R=0}^{5} \lceil b \mapsto \mathsf{false} * x \mapsto N * y \mapsto R \rceil \rangle\ x := y\ \langle \&_{R=0}^{5} \lceil x \mapsto R * b \mapsto \mathsf{false} * y \mapsto R \rceil \rangle}\ \textsc{NSplit}}{\vdash_{\mathsf{w}} \langle \lceil b \mapsto \mathsf{false} * x \mapsto N \rceil * \lceil y \in \{0,\dots,5\} \rceil \rangle\ x := y\ \langle (\&_{R=0}^{5} \lceil x \mapsto R * b \mapsto \mathsf{false} \rceil) * \lceil y \in \{0,\dots,5\} \rceil \rangle}\ \textsc{Exists+Consequence}}{y \in \{0,\dots,5\} \vdash_{\mathsf{w}} \langle \lceil b \mapsto \mathsf{false} * x \mapsto N \rceil \rangle\ x := y\ \langle \&_{R=0}^{5} \lceil x \mapsto R * b \mapsto \mathsf{false} \rceil \rangle}\ \textsc{Atom} \tag{7}$$

In (8), we give the full if statement.

$$\dfrac{\dfrac{\dfrac{}{y \in \{0,\dots,5\} \vdash_{\mathsf{w}} \langle \lceil b \mapsto \mathsf{true} * x \mapsto N \rceil \rangle\ x := x - 1\ \langle \lceil x \mapsto N - 1 * b \mapsto \mathsf{true} \rceil \rangle}\ \textsc{Assign} \quad (7)}{y \in \{0,\dots,5\} \vdash_{\mathsf{w}} \langle \bigoplus_{X \sim \mathbf{Ber}\left(\frac{1}{2}\right)} \lceil b \mapsto X * x \mapsto N \rceil \rangle\ \mathbf{if}\ b\ \mathbf{then}\ x := x - 1\ \mathbf{else}\ x := y\ \langle \lceil x \mapsto N - 1 * b \mapsto \mathsf{true} \rceil \oplus_{\frac{1}{2}} \&_{R=0}^{5} \lceil x \mapsto R * b \mapsto \mathsf{false} \rceil \rangle}\ \textsc{If}}{y \in \{0,\dots,5\} \vdash_{\mathsf{w}} \langle \bigoplus_{X \sim \mathbf{Ber}\left(\frac{1}{2}\right)} \lceil b \mapsto X * x \mapsto R * 0 < R = N \le 5 \rceil \rangle\ \mathbf{if}\ b\ \mathbf{then}\ x := x - 1\ \mathbf{else}\ x := y\ \langle \&_{R=0}^{N-1} \lceil x \mapsto R * \mathsf{own}(b) \rceil \oplus_{\geq \frac{1}{2}} \&_{R=N}^{5} \lceil x \mapsto R * \mathsf{own}(b) \rceil \rangle}\ \textsc{Consequence} \tag{8}$$

Finally, we complete the derivation below. Recall that the loop invariant used in BoundedRank is $\varphi \triangleq \lceil x \mapsto R * 0 \le R \le 5 * \mathsf{own}(b) \rceil$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(6) \qquad (8)}{y \in \{0,\dots,5\} \vdash_{\mathsf{w}} \langle \lceil x \mapsto R * \mathsf{own}(b) * 0 < R = N \le 5 \rceil \rangle\ b :\approx \mathbf{Ber}\left(\frac{1}{2}\right) \fatsemi \mathbf{if}\ b\ \mathbf{then}\ x := x - 1\ \mathbf{else}\ x := y\ \langle \&_{R=0}^{N-1} \lceil x \mapsto R * \mathsf{own}(b) \rceil \oplus_{\geq \frac{1}{2}} \&_{R=N}^{5} \lceil x \mapsto R * \mathsf{own}(b) \rceil \rangle}\ \textsc{Seq}}{y \in \{0,\dots,5\} \vdash_{\mathsf{w}} \langle \&_{R=0}^{5} \lceil x \mapsto R * \mathsf{own}(b) \rceil \rangle\ \mathbf{while}\ x > 0\ \mathbf{do}\ b :\approx \mathbf{Ber}\left(\frac{1}{2}\right) \fatsemi \mathbf{if}\ b\ \mathbf{then}\ x := x - 1\ \mathbf{else}\ x := y\ \langle \lceil x \mapsto 0 * \mathsf{own}(b) \rceil \rangle}\ \textsc{BoundedRank}}{y \in \{0,\dots,5\} \vdash_{\mathsf{w}} \langle \&_{R=0}^{5} \lceil x \mapsto R * \mathsf{own}(b) \rceil \rangle\ \mathbf{while}\ x > 0\ \mathbf{do}\ b :\approx \mathbf{Ber}\left(\frac{1}{2}\right) \fatsemi \mathbf{if}\ b\ \mathbf{then}\ x := x - 1\ \mathbf{else}\ x := y\ \langle \lceil x \mapsto 0 \rceil \rangle}\ \textsc{Consequence}}{y \in \{0,\dots,5\} \vdash_{\mathsf{w}} \langle \lceil x \in \{0,\dots,5\} * \mathsf{own}(b) \rceil \rangle\ \mathbf{while}\ x > 0\ \mathbf{do}\ b :\approx \mathbf{Ber}\left(\frac{1}{2}\right) \fatsemi \mathbf{if}\ b\ \mathbf{then}\ x := x - 1\ \mathbf{else}\ x := y\ \langle \lceil x \mapsto 0 \rceil \rangle}\ \textsc{Exists}}{y \in \{0,\dots,5\} \vdash \langle \lceil x \in \{0,\dots,5\} * \mathsf{own}(b) \rceil \rangle\ \mathbf{while}\ x > 0\ \mathbf{do}\ b :\approx \mathbf{Ber}\left(\frac{1}{2}\right) \fatsemi \mathbf{if}\ b\ \mathbf{then}\ x := x - 1\ \mathbf{else}\ x := y\ \langle \lceil x \mapsto 0 \rceil \rangle}\ \textsc{Strengthen}$$

F.3 Entropy Mixer

Recall the following entropy mixer program from Section 6.1.

$$y \coloneqq 0 \fatsemi \Big(\; x_1 \coloneqq y \fatsemi x_2 :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right) \fatsemi z \coloneqq \mathbf{xor}(x_1, x_2) \;\; \Big\| \;\; y \coloneqq 1 \; \Big)$$

We will analyze this program using the invariant $y \in \{0,1\}$, and conclude in the end that $z \sim \mathbf{Ber}\left(\frac{1}{2}\right)$. We begin by analyzing the read from shared state in the first thread. The derivation is given below in (9).

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash_w \langle \lceil y \mapsto Y * \mathsf{own}(x_1) \rceil \rangle \; x_1 \coloneqq y \; \langle \lceil x_1 \mapsto Y * y \mapsto Y \rceil \rangle}\,\textsc{Assign}}{\vdash_w \langle \&_{Y \in \{0,1\}} \lceil y \mapsto Y * \mathsf{own}(x_1) \rceil \rangle \; x_1 \coloneqq y \; \langle \&_{Y \in \{0,1\}} \lceil x_1 \mapsto Y * y \mapsto Y \rceil \rangle}\,\textsc{NSplit1}}{\vdash_w \langle \&_{Y \in \{0,1\}} \lceil y \mapsto Y * \mathsf{own}(x_1) \rceil \rangle \; x_1 \coloneqq y \; \langle (\&_{Y \in \{0,1\}} \lceil x_1 \mapsto Y \rceil) * \lceil y \in \{0,1\} \rceil \rangle}\,\textsc{Consequence}}{\vdash_w \langle \lceil \mathsf{own}(x_1) \rceil * \lceil y \in \{0,1\} \rceil \rangle \; x_1 \coloneqq y \; \langle (\&_{Y \in \{0,1\}} \lceil x_1 \mapsto Y \rceil) * \lceil y \in \{0,1\} \rceil \rangle}\,\textsc{Exists}}{y \in \{0,1\} \vdash_w \langle \lceil \mathsf{own}(x_1) \rceil \rangle \; x_1 \coloneqq y \; \langle \&_{Y \in \{0,1\}} \lceil x_1 \mapsto Y \rceil \rangle}\,\textsc{Atom} \tag{9}$$

First, Atom is applied to open the invariant. Next, we use Exists and NSplit1 to gain access to the value of y, so that we can apply the Assign rule. The rule of Consequence is used to weaken the information about y and move it out of the scope of the $\&$, so that the invariant can be closed. Since the postcondition at this stage is not precise, we are forced to use a weak triple. We now move on to derive a specification for the write to z below in (10).

$$\dfrac{\dfrac{\dfrac{}{\langle \lceil x_1 \mapsto X \rceil * \lceil x_2 \mapsto X' \rceil * \lceil \mathsf{own}(z) \rceil \rangle \; z \coloneqq \mathbf{xor}(x_1, x_2) \; \langle \lceil z \mapsto \mathbf{xor}(X, X') \rceil \rangle}\,\textsc{Assign}}{\langle \bigoplus_{X' \sim \mathbf{Ber}(1/2)} \lceil x_1 \mapsto X \rceil * \lceil x_2 \mapsto X' \rceil * \lceil \mathsf{own}(z) \rceil \rangle \; z \coloneqq \mathbf{xor}(x_1, x_2) \; \langle \bigoplus_{X' \sim \mathbf{Ber}(1/2)} \lceil z \mapsto \mathbf{xor}(X, X') \rceil \rangle}\,\textsc{Split1}}{\langle \lceil x_1 \mapsto X \rceil * (x_2 \sim \mathbf{Ber}\,(1/2)) * \lceil \mathsf{own}(z) \rceil \rangle \; z \coloneqq \mathbf{xor}(x_1, x_2) \; \langle z \sim \mathbf{Ber}\,(1/2) \rangle}\,\textsc{Consequence} \tag{10}$$

At this point, we have that x_1 is deterministic and x_2 is distributed according to a Bernoulli distribution. We use Split1 to do case analysis on the result of the sampling operation, at which point we use Assign to conclude that $z \mapsto \mathbf{xor}(X, X')$. Since X is constant, then $\mathbf{xor}(X, X')$ is a bijection from $\{0,1\}$ to $\{0,1\}$, and we can therefore use the rule of Consequence to conclude that z is uniformly distributed. We can now compose this information with the sampling operation immediately before.

$$\dfrac{\dfrac{\dfrac{}{y \in \{0,1\} \vdash_w \langle \lceil \mathsf{own}(x_2) \rceil \rangle \; x_2 :\approx \mathbf{Ber}\,(1/2) \;\; \langle x_2 \sim \mathbf{Ber}\,(1/2) \rangle \rangle}\,\textsc{Samp}}{y \in \{0,1\} \vdash_w \langle \lceil x_1 \mapsto X \rceil * \lceil \mathsf{own}(x_2, z) \rceil \rangle \; x_2 :\approx \mathbf{Ber}\,(1/2) \;\; \langle \lceil x_1 \mapsto X \rceil * (x_2 \sim \mathbf{Ber}\,(1/2)) * \lceil \mathsf{own}(z) \rceil \rangle}\,\textsc{Frame} \quad (10)}{y \in \{0,1\} \vdash_w \langle \lceil x_1 \mapsto X \rceil * \lceil \mathsf{own}(x_2, z) \rceil \rangle \; x_2 :\approx \mathbf{Ber}\,(1/2) \fatsemi z \coloneqq \mathbf{xor}(x_1, x_2) \; \langle z \sim \mathbf{Ber}\,(1/2) \rangle}\,\textsc{Seq}}{y \in \{0,1\} \vdash_w \langle \lceil \mathsf{own}(x_2, z) \rceil * \&_{Y \in \{0,1\}} \lceil x_1 \mapsto Y \rceil \rangle \; x_2 :\approx \mathbf{Ber}\,(1/2) \fatsemi z \coloneqq \mathbf{xor}(x_1, x_2) \; \langle z \sim \mathbf{Ber}\,(1/2) \rangle}\,\textsc{Consequence}, \textsc{NSplit2} \tag{11}$$

We start by applying NSplit2 in order to gain access to the nondeterministic value of x_1. It is important to do this *before* the sampling operation, since we need to ensure that x_2 is uniformly distributed given any fixed value for x_1. Next, we apply Seq and derive specs for the individual commands. The sampling operation is simple, we complete the proof with the Frame and Samp rules. Now, we can derive a specification for the entire first thread in (12).

$$\dfrac{\dfrac{\dfrac{(9)}{y \in \{0,1\} \vdash_w \langle \lceil \mathsf{own}(x_1, x_2, z) \rceil \rangle \; x_1 \coloneqq y \; \langle \lceil \mathsf{own}(x_2, z) \rceil * \&_{Y \in \{0,1\}} \lceil x_1 \mapsto Y \rceil \rangle}\,\textsc{Frame} \quad (11)}{y \in \{0,1\} \vdash_w \langle \lceil \mathsf{own}(x_1, x_2, z) \rceil \rangle \; x_1 \coloneqq y \fatsemi x_2 :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right) \fatsemi z \coloneqq \mathbf{xor}(x_1, x_2) \; \langle z \sim \mathbf{Ber}\left(\tfrac{1}{2}\right) \rangle}\,\textsc{Seq}}{y \in \{0,1\} \vdash \langle \lceil \mathsf{own}(x_1, x_2, z) \rceil \rangle \; x_1 \coloneqq y \fatsemi x_2 :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right) \fatsemi z \coloneqq \mathbf{xor}(x_1, x_2) \; \langle z \sim \mathbf{Ber}\left(\tfrac{1}{2}\right) \rangle}\,\textsc{Strengthen} \tag{12}$$

This step essentially just involves using Seq and Frame to compose the triples that we previously derived. At the end, we also use Strengthen, since the postcondition is now precise. Finally, we complete the derivation for the whole program.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\langle \lceil y \in \{0,1\} \rceil \rangle\ y \coloneqq 1\ \langle \lceil y \mapsto 1 \rceil \rangle}\ \textsc{Assign}}{\langle \lceil y \in \{0,1\} \rceil \rangle\ y \coloneqq 1\ \langle \lceil y \in \{0,1\} \rceil \rangle}\ \textsc{Consequence}}{y \in \{0,1\} \vdash \langle \lceil y \in \{0,1\} \rceil \rangle\ y \coloneqq 1\ \langle \lceil \mathsf{true} \rceil \rangle}\ \textsc{Atom}\ (12) \qquad \dfrac{\vdots}{\vdash \langle \lceil \mathsf{own}(x_1,x_2,y,z) \rceil \rangle\ y \coloneqq 0\ \langle \lceil y \mapsto 0 * \mathsf{own}(x_1,x_2,z) \rceil \rangle}\ \textsc{Assign}}{y \in \{0,1\} \vdash \langle \lceil \mathsf{own}(x_1,x_2,z) \rceil \rangle\ x_1 \coloneqq y \fatsemi x_2 :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right) \fatsemi z \coloneqq \mathbf{xor}(x_1,x_2) \parallel y \coloneqq 1\ \langle z \sim \mathbf{Ber}\left(\tfrac{1}{2}\right) \rangle}\ \textsc{Par}}{\vdash \langle \lceil \mathsf{own}(x_1,x_2,z) \rceil * \lceil y \in \{0,1\} \rceil \rangle\ x_1 \coloneqq y \fatsemi x_2 :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right) \fatsemi z \coloneqq \mathbf{xor}(x_1,x_2) \parallel y \coloneqq 1\ \langle (z \sim \mathbf{Ber}\left(\tfrac{1}{2}\right)) * \lceil y \in \{0,1\} \rceil \rangle}\ \textsc{Share}}{\vdash \langle \lceil y \mapsto 0 * \mathsf{own}(x_1,x_2,z) \rceil \rangle\ x_1 \coloneqq y \fatsemi x_2 :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right) \fatsemi z \coloneqq \mathbf{xor}(x_1,x_2) \parallel y \coloneqq 1\ \langle z \sim \mathbf{Ber}\left(\tfrac{1}{2}\right) \rangle}\ \textsc{Consequence}$$

$$\dfrac{\vdots \qquad \vdots}{\vdash \langle \lceil \mathsf{own}(x_1,x_2,y,z) \rceil \rangle\ y \coloneqq 0 \fatsemi (x_1 \coloneqq y \fatsemi x_2 :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right) \fatsemi z \coloneqq \mathbf{xor}(x_1,x_2) \parallel y \coloneqq 1)\ \langle z \sim \mathbf{Ber}\left(\tfrac{1}{2}\right) \rangle}\ \textsc{Seq}$$

After concluding that $y \mapsto 0$ after the first command, we can weaken this information and use Share to move y into the invariant. Next, we use Par to compositionally analyze the two threads. We have already seen the proof of the first thread above. The second thread has a simple derivation, since $y \coloneqq 1$ is clearly atomic and obeys the invariant.

F.4 Concurrent Shuffling

Recall the following program from Section 6.2 for concurrent shuffling a list using two parallel threads.

$$\begin{array}{l}
a_1 \coloneqq [\,] \fatsemi a_2 \coloneqq [\,] \fatsemi i \coloneqq 0 \fatsemi \\
\mathbf{while}\ i < \mathbf{len}(a)\ \mathbf{do}\ (\\
\quad b :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right) \fatsemi \\
\quad \mathbf{if}\ b\ \mathbf{then}\ a_1 \coloneqq a_1 +\!\!+ [a[i]]\ \mathbf{else}\ a_2 \coloneqq a_2 +\!\!+ [a[i]] \fatsemi \\
\quad i \coloneqq i + 1 \\
) \fatsemi \\
\mathbf{shuffle}_1 \parallel \mathbf{shuffle}_2 \fatsemi \\
a \coloneqq a_1 +\!\!+ a_2
\end{array}
\qquad\qquad
\begin{array}{l}
\mathbf{shuffle}_k : \\
\quad i_k \coloneqq \mathbf{len}(a_k) - 1 \fatsemi \\
\quad \mathbf{while}\ i_k > 0\ \mathbf{do} \\
\quad\quad j_k :\approx \mathbf{unif}\ ([0,\ldots,i_k]) \fatsemi \\
\quad\quad a_k \coloneqq \mathbf{swap}(a_k,i_k,j_k) \fatsemi \\
\quad\quad i_k \coloneqq i_k - 1
\end{array}$$

We now give the complete derivation of the correctness proof. We begin with the $\mathbf{shuffle}_k$ program. To define the loop invariant, we first recursively define $\mathbf{swaps}(n,\ell)$ which gives the set of possible lists obtained when $i_k \mapsto n$. It is easy to see that $\mathbf{swaps}(0,\ell) = \Pi(\ell)$, since once i reaches n, all permutations of the elements above position n are accounted for, so once i reaches 0, all permutations of the entire list are accounted for (the final position does not explicitly need to be chosen, since only one element remains).

$$\mathbf{swaps}(n,\ell) = \begin{cases} \{\ell\} & \text{if } n = \mathbf{len}(\ell) - 1 \\ \{\mathbf{swap}(\ell', n+1, j) \mid \ell' \in \mathbf{swaps}(n+1,\ell), j \in \{0,\ldots,n+1\}\} & \text{if } 0 \le n < \mathbf{len}(\ell) - 1 \end{cases}$$

We now define the loop invariant φ in using $\mathbf{swaps}$. The rank R is given by i_k, which is bounded between 0 and $\mathbf{len}(A) - 1$. The final postcondition $\varphi[0/R]$ can also be simplified as follows.

$$\varphi \triangleq \lceil i_k \mapsto R * 0 \le R \le \mathbf{len}(A) - 1 * \mathsf{own}(j_k) \rceil * (a_k \sim \mathbf{unif}\ (\mathbf{swaps}(R,A))) \qquad \varphi[0/R] = \lceil i_k \mapsto 0 * \mathsf{own}(j_k) \rceil * (a_k \sim \mathbf{unif}\ (\Pi(A)))$$

We now show the derivation for the loop body as a decorated program:

$$
\begin{array}{ll}
 & \langle \varphi * \lceil R = N > 0 \rceil \rangle \\
\textsc{Consequence} & \langle \lceil i_k \mapsto N * 0 < N \leq \mathbf{len}(A) - 1 * \mathsf{own}(j_k) \rceil * (a_k \sim \mathbf{unif}\ (\mathbf{swaps}(\mathbf{len}(A) - 1 - N, A))) \rangle \\
\textsc{Samp} & \quad j_k :\approx \mathbf{unif}\ ([0, \ldots, i_k]) \fatsemi \\
 & \langle \lceil i_k \mapsto N * 0 < N \leq \mathbf{len}(A) - 1 \rceil * (a_k \sim \mathbf{unif}\ (\mathbf{swaps}(N, A))) * (j_k \sim \mathbf{unif}\ ([0, \ldots, N]) \rangle \\
\textsc{Consequence} & \langle \bigoplus_{X \sim \mathbf{unif}(\mathbf{swaps}(N,A))} \bigoplus_{Y \sim \mathbf{unif}([0,\ldots,N])} \lceil i_k \mapsto N * 0 < N \leq \mathbf{len}(A) - 1 * a_k \mapsto X * j_k \mapsto Y \rceil \rangle \\
\textsc{Split1}, \textsc{Assign} & \quad a_k := \mathbf{swap}(a_k, i_k, j_k) \fatsemi \\
 & \langle \bigoplus_{X \sim \mathbf{unif}(\mathbf{swaps}(N,A))} \bigoplus_{Y \sim \mathbf{unif}([0,\ldots,N])} \lceil i_k \mapsto N * 0 < N \leq \mathbf{len}(A) - 1 * a_k \mapsto \mathbf{swap}(X, N, Y) * j_k \mapsto Y \rceil \rangle \\
\textsc{Consequence} & \langle \lceil i_k \mapsto N * 0 < N \leq \mathbf{len}(A) - 1 * \mathsf{own}(j_k) \rceil * (a_k \sim \mathbf{unif}\ (\mathbf{swaps}(N - 1, A))) \rangle \\
\textsc{Assign} & \quad i_k := i_k - 1 \\
 & \langle \lceil i_k \mapsto N - 1 * 0 < N \leq \mathbf{len}(A) - 1 * \mathsf{own}(j_k) \rceil * (a_k \sim \mathbf{unif}\ (\mathbf{swaps}(N - 1, A))) \rangle \\
\textsc{Consequence} & \langle (\&_{R=0}^{N-1}\, \varphi) \oplus_1 (\&_{R=N}^{\mathbf{len}(A)-1}\, \varphi) \rangle
\end{array}
\tag{13}
$$

When entering the loop body, we know that the tail of a_k is already shuffled. After performing the swap of i_k and j_k, we extend the shuffled tail by one position. Clearly, the shuffles remain uniformly distributed, since X is uniformly distributed, and the element at position N is also chosen uniformly. Now, we give the remaining derivation of $\mathbf{shuffle}_k$, which is quite mechanical given the derivation of the loop body above:

$$
\dfrac{\dfrac{}{\vdash \langle \lceil a_k \mapsto A * \mathsf{own}(i_k, j_k) \rceil \rangle\ i_k := \mathbf{len}(a_k) - 1\ \langle \lceil a_k \mapsto A * i_k \mapsto \mathbf{len}(A) - 1 * \mathsf{own}(j_k) \rceil \rangle}\ \textsc{Assign} \quad \vdots \quad \dfrac{\dfrac{\dfrac{(13)}{\vdash \langle \&_{R=0}^{\mathbf{len}(A)-1}\, \varphi \rangle\ \mathbf{while}\ i_k > 0\ \mathbf{do}\ (\cdots)\ \langle (a_k \sim \mathbf{unif}\ (\Pi(A))) * \lceil i_k \mapsto 0 * \mathsf{own}(j_k) \rceil \rangle}\ \textsc{BoundedRank}}{\vdash \langle \lceil a_k \mapsto A * i_k \mapsto \mathbf{len}(A) - 1 * \mathsf{own}(j_k) \rceil \rangle\ \mathbf{while}\ i_k > 0\ \mathbf{do}\ (\cdots)\ \langle a_k \sim \mathbf{unif}\ (\Pi(A)) \rangle}\ \textsc{Consequence}}{\vdash \langle \lceil a_k \mapsto A * \mathsf{own}(i_k, j_k) \rceil \rangle\ i_k := \mathbf{len}(a_k) - 1 \fatsemi \mathbf{while}\ i_k > 0\ \mathbf{do}\ (\cdots)\ \langle a_k \sim \mathbf{unif}\ (\Pi(A)) \rangle}\ \textsc{Seq}
\tag{14}
$$

We now move to deriving the specification for the main program. To analyze the loop, we use the following loop invariant, where $M = \mathbf{len}(A)$. Clearly, the rank R is bounded between 0 and M, and when $R = 0$ the loop must terminate since $i = \mathbf{len}(a)$.

$$
\psi = \lceil i \mapsto M - R * 0 \leq R \leq M * a \mapsto A * \mathsf{own}(b) \rceil * \bigoplus_{X \sim \mathbf{unif}\left(\{0,1\}^{M-R}\right)} \lceil a_1 \mapsto A[X] * a_2 \mapsto A[\neg X] \rceil
$$

We now show the derivation of the loop body as a decorated program. The key idea is to merge the newly sampled value of b into the $\bigoplus$ over X to extend the length of the bit-string on each iteration.

$$
\begin{array}{ll}
 & \langle \psi * \lceil R = N > 0 \rceil \rangle \\
\textsc{Consequence} & \left\langle \lceil i \mapsto M - N * 0 < N \le M * a \mapsto A * \mathsf{own}(b) \rceil * \bigoplus_{X \sim \mathbf{unif}(\{0,1\}^{M-N})} \lceil a_1 \mapsto A[X] * a_2 \mapsto A[\neg X] \rceil \right\rangle \\
\textsc{Samp} & \quad b :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right) \fatsemi \\
 & \left\langle \lceil i \mapsto M - N * 0 < N \le M * a \mapsto A \rceil * (b \sim \mathbf{Ber}\left(\tfrac{1}{2}\right)) * \bigoplus_{X \sim \mathbf{unif}(\{0,1\}^{M-N})} \lceil a_1 \mapsto A[X] * a_2 \mapsto A[\neg X] \rceil \right\rangle \\
\textsc{If} & \quad \mathbf{if}\ b\ \mathbf{then} \\
 & \left\langle \lceil i \mapsto M - N * 0 < N \le M * a \mapsto A * b \mapsto Y * Y = 1 \rceil * \bigoplus_{X \sim \mathbf{unif}(\{0,1\}^{M-N})} \lceil a_1 \mapsto A[X] * a_2 \mapsto A[\neg X] \rceil \right\rangle \\
\textsc{Assign} & \quad\quad a_1 := a_1 +\!\!+ [a[i]] \\
 & \left\langle \lceil i \mapsto M - N * 0 < N \le M * a \mapsto A * b \mapsto Y * Y = 1 \rceil * \bigoplus_{X \sim \mathbf{unif}(\{0,1\}^{M-N})} \lceil a_1 \mapsto A[X +\!\!+ [Y]] * a_2 \mapsto A[\neg(X +\!\!+ [Y])] \rceil \right\rangle \\
 & \quad \mathbf{else} \\
 & \left\langle \lceil i \mapsto M - N * 0 < N \le M * a \mapsto A * b \mapsto Y * Y = 0 \rceil * \bigoplus_{X \sim \mathbf{unif}(\{0,1\}^{M-N})} \lceil a_1 \mapsto A[X] * a_2 \mapsto A[\neg X] \rceil \right\rangle \\
\textsc{Assign} & \quad\quad a_2 := a_2 +\!\!+ [a[i]] \fatsemi \\
 & \left\langle \lceil i \mapsto M - N * 0 < N \le M * a \mapsto A * b \mapsto Y * Y = 0 \rceil * \bigoplus_{X \sim \mathbf{unif}(\{0,1\}^{M-N})} \lceil a_1 \mapsto A[X +\!\!+ [Y]] * a_2 \mapsto A[\neg(X +\!\!+ [Y])] +\!\!+ A[M-N] \rceil \right\rangle \\
 & \left\langle \bigoplus_{Y \sim \mathbf{Ber}\left(\frac{1}{2}\right)} \lceil i \mapsto M - N * 0 < N \le M * a \mapsto A * b \mapsto Y \rceil * \bigoplus_{X \sim \mathbf{unif}(\{0,1\}^{M-N})} \lceil a_1 \mapsto A[X +\!\!+ [Y]] * a_2 \mapsto A[\neg(X +\!\!+ [Y])] +\!\!+ A[M-N] \rceil \right\rangle \\
\textsc{Consequence} & \left\langle \lceil i \mapsto M - N * 0 < N \le M * a \mapsto A * \mathsf{own}(b) \rceil * \bigoplus_{X \sim \mathbf{unif}\left(\{0,1\}^{M-(N-1)}\right)} \lceil a_1 \mapsto A[X] * a_2 \mapsto A[\neg X] \rceil \right\rangle \\
\textsc{Assign} & \quad i := i + 1 \\
 & \left\langle \lceil i \mapsto M - (N-1) * 0 < N \le M * a \mapsto A * \mathsf{own}(b) \rceil * \bigoplus_{X \sim \mathbf{unif}\left(\{0,1\}^{M-(N-1)}\right)} \lceil a_1 \mapsto A[X] * a_2 \mapsto A[\neg X] \rceil \right\rangle \\
\textsc{Consequence} & \langle \psi[N-1/R] \rangle
\end{array}
\tag{15}
$$

Finally, we conclude by showing the derivation of the main program, which is also quite mechanical. The final consequence relies on some combinatorial reasoning, which is explained in Section 6.2.

$\langle \lceil a \mapsto A * \mathsf{own}(b, a_1, a_2, i, i_1, i_2, j_1, j_2) \rceil \rangle$

Assign $\quad a_1 \coloneqq [] \fatsemi$

$\langle \lceil a \mapsto A * a_1 \mapsto [] * \mathsf{own}(b, a_2, i, i_1, i_2, j_1, j_2) \rceil \rangle$

Assign $\quad a_2 \coloneqq [] \fatsemi$

$\langle \lceil a \mapsto A * a_1 \mapsto [] * a_2 \mapsto [] * \mathsf{own}(b, i, i_1, i_2, j_1, j_2) \rceil \rangle$

Assign $\quad i \coloneqq 0 \fatsemi$

$\langle \lceil a \mapsto A * a_1 \mapsto [] * a_2 \mapsto [] * i \mapsto 0 * \mathsf{own}(b, i_1, i_2, j_1, j_2) \rceil \rangle$

Consequence $\quad \langle \psi[\mathbf{len}(A)/R] \rangle$

BoundedRank, (15) $\quad \mathbf{while}\ i < \mathbf{len}(A)\ \mathbf{do}\ (\cdots) \fatsemi$

$\left\langle \lceil i \mapsto \mathbf{len}(A) * a \mapsto A * \mathsf{own}(\cdots) \rceil * \bigoplus_{X \sim \mathbf{unif}\left(\{0,1\}^{\mathbf{len}(A)}\right)} \lceil a_1 \mapsto A[X] * a_2 \mapsto A[\neg X] \rceil \right\rangle$

Consequence
$\left\langle \bigoplus_{X \sim \mathbf{unif}\left(\{0,1\}^{\mathbf{len}(A)}\right)} \lceil a_1 \mapsto A[X] * a_2 \mapsto [A[\neg X] * \mathsf{own}(\cdots) \rceil \right\rangle$

Split1 $\quad \langle \lceil a_1 \mapsto A[X] * a_2 \mapsto A[\neg X] * \mathsf{own}(\cdots) \rceil \rangle$

Frame, Par, (14)
$$\left(\begin{array}{l} \langle \lceil a_1 \mapsto A[X] * \mathsf{own}(i_1, j_1) \rceil \rangle \\ \quad \mathbf{shuffle}_1 \\ \langle a_1 \sim \mathbf{unif}\ (\Pi(A[X])) \rangle \end{array} \middle\| \begin{array}{l} \langle \lceil a_2 \mapsto A[\neg X] * \mathsf{own}(i_2, j_2) \rceil \rangle \\ \quad \mathbf{shuffle}_2 \\ \langle a_2 \sim \mathbf{unif}\ (\Pi(A[\neg X])) \rangle \end{array} \right) \fatsemi$$

$\langle a_1 \sim \mathbf{unif}\ (\Pi(A[X])) * a_2 \sim \mathbf{unif}\ (\Pi(A[\neg X])) * \mathsf{own}(\cdots) \rangle$

$\left\langle \bigoplus_{X \sim \mathbf{unif}\left(\{0,1\}^{\mathbf{len}(A)}\right)} a_1 \sim \mathbf{unif}\ (\Pi(A[X])) * a_2 \sim \mathbf{unif}\ (\Pi(A[\neg X])) * \lceil \mathsf{own}(a) \rceil \right\rangle$

Consequence
$\left\langle \bigoplus_{X \sim \mathbf{unif}\left(\{0,1\}^{\mathbf{len}(A)}\right)} \bigoplus_{A_1 \sim \mathbf{unif}(\Pi(A[X]))} \bigoplus_{A_2 \sim \mathbf{unif}(\Pi(A[\neg X]))} \lceil a_1 \mapsto A_1 * a_2 \mapsto A_2 * \mathsf{own}(a) \rceil \right\rangle$

Split1, Assign $\quad a \coloneqq a_1 +\!\!+ a_2$

$\left\langle \bigoplus_{X \sim \mathbf{unif}\left(\{0,1\}^{\mathbf{len}(A)}\right)} \bigoplus_{A_1 \sim \mathbf{unif}(\Pi(A[X]))} \bigoplus_{A_2 \sim \mathbf{unif}(\Pi(A[\neg X]))} \lceil a \mapsto A_1 +\!\!+ A_2 \rceil \right\rangle$

Consequence $\quad \langle a \sim \mathbf{unif}\ (\Pi(A)) \rangle$

F.5 Private Information Retrieval

Recall the following private information retrieval program from Section 6.3.

$$\begin{array}{ll}
\textbf{PrivFetch}: & \textbf{fetch}_k: \\
\quad q_1 :\approx \textbf{unif}\,(\{0,1\}^n)\,; & \quad i_k := 0\,; r_k := 0\,; \\
\quad q_2 := \textbf{xor}(q_1, x)\,; & \quad \textbf{while}\ i_k < \textbf{len}(q_k)\ \textbf{do} \\
\quad \textbf{fetch}_1 \parallel \textbf{fetch}_2\,; & \quad\quad \textbf{if}\ q_k[i_k] = 1\ \textbf{then} \\
\quad r := \textbf{xor}(r_1, r_2) & \quad\quad\quad r_k := \textbf{xor}(r_k, d[i_k])\,; \\
 & \quad\quad i_k := i_k + 1
\end{array}$$

We now give the complete correctness proof, showing that **PrivFetch** correctly fetches the data entry indicated by x from the database. We begin with the $\textbf{fetch}_k$ procedure, whose proof is shown below in (16).

$$\dfrac{\dfrac{\dfrac{}{d \mapsto D \vdash \langle \lceil q_k \mapsto Q * i_k \mapsto 0 * \mathsf{own}(r_k) \rceil \rangle\ r_k := 0\ \langle \lceil q_k \mapsto Q * i_k \mapsto 0 * r_k \mapsto 0 \rceil \rangle}\ \textsc{Assign} \qquad (17)}{d \mapsto D \vdash \langle \lceil q_k \mapsto Q * i_k \mapsto 0 * \mathsf{own}(r_k) \rceil \rangle\ r_k := 0\,; \textbf{while}\ i_k < \textbf{len}(q_k)\ \textbf{do}\ (\cdots)\ \langle \lceil r_k \mapsto \underset{0 \le i < n : Q[i]=1}{\textbf{xor}} D[i] \rceil \rangle}\ \textsc{Seq}}{\dfrac{\dfrac{}{d \mapsto D \vdash \langle \lceil q_k \mapsto Q * \mathsf{own}(i_k, r_k) \rceil \rangle\ i_k := 0\ \langle \lceil q_k \mapsto Q * i_k \mapsto 0 * \mathsf{own}(r_k) \rceil \rangle}\ \textsc{Assign} \quad \vdots}{d \mapsto D \vdash \langle \lceil q_k \mapsto Q * \mathsf{own}(i_k, r_k) \rceil \rangle\ i_k := 0\,; r_k := 0\,; \textbf{while}\ i_k < \textbf{len}(q_k)\ \textbf{do}\ (\cdots)\ \langle \lceil r_k \mapsto \underset{0 \le i < n : Q[i]=1}{\textbf{xor}} D[i] \rceil \rangle}\ \textsc{Seq} \tag{16}$$

The proof above is quite mechanical, just using Seq and Assign to analyze the initialization. The next step is to analyze the loop. To do so, we use the following invariant, which states that after i_k iterations, r_k holds a dot product of the first i_k elements of D and Q.

$$\varphi \triangleq \lceil q_k \mapsto Q * i_k \mapsto n - R * 0 \le R \le n * r_k \mapsto \underset{0 \le i < n-R : Q[i]=1}{\textbf{xor}} D[i] \rceil$$

The rank R indicates the remaining number of iterations until termination, and it clearly bounded between 0 and n. We can also specialize φ to obtain the pre- and postconditions of the whole loop, shown below:

$$\varphi[n/R] = \lceil q_k \mapsto Q * i_k \mapsto 0 * r_k \mapsto 0 \rceil \qquad \varphi[0/R] = \lceil q_k \mapsto Q * i_k \mapsto n - R * r_k \mapsto \underset{0 \le i < n : Q[i]=1}{\textbf{xor}} D[i] \rceil$$

Analyzing the loop is simply a matter of applying BoundedRank.

$$\dfrac{\dfrac{(18)}{d \mapsto D \vdash \langle \&_{R=0}^{n}\, \varphi \rangle\ \textbf{while}\ i_k < \textbf{len}(q_k)\ \textbf{do}\ (\cdots)\ \langle \varphi[0/R] \rangle}\ \textsc{BoundedRank}}{d \mapsto D \vdash \langle \lceil q_k \mapsto Q * i_k \mapsto 0 * r_k \mapsto 0 \rceil \rangle\ \textbf{while}\ i_k < \textbf{len}(q_k)\ \textbf{do}\ (\cdots)\ \langle \lceil r_k \mapsto \underset{0 \le i < n : Q[i]=1}{\textbf{xor}} D[i] \rceil \rangle}\ \textsc{Consequence} \tag{17}$$

Now, we show that the loop body upholds the invariant. We show this as a decorated program. We use the derived IfPure rule to analyze the if statement, which is similar to the rule for if statements in Hoare Logic. In each case of the if statement, the respective value of $Q[n-N]$ allows us to extend the dot product.

$$
\begin{array}{ll}
 & \langle \varphi * \lceil R = N > 0 \rceil \rangle \\
\textsc{Consequence} & \langle \lceil q_k \mapsto Q * i_k \mapsto n-N * r_k \mapsto \mathbf{xor}_{0\le i<n-N:Q[i]=1}\, D[i] \rceil \rangle \\
\textsc{IfPure} & \quad \mathbf{if}\ q_k[i_k]\ \mathbf{then} \\
\textsc{Atom} & \quad \langle \lceil q_k \mapsto Q * i_k \mapsto n-N * r_k \mapsto \mathbf{xor}_{0\le i<n-N:Q[i]=1}\, D[i] * Q[n-N]=1 \rceil \rangle \\
 & \quad\quad \langle \lceil q_k \mapsto Q * i_k \mapsto n-N * r_k \mapsto \mathbf{xor}_{0\le i<n-N:Q[i]=1}\, D[i] * Q[n-N]=1 \rceil * \lceil d \mapsto D \rceil \rangle \\
\textsc{Assign} & \quad\quad r_k = \mathbf{xor}(r_k, d[i_k]) \\
 & \quad\quad \langle \lceil q_k \mapsto Q * i_k \mapsto n-N * r_k \mapsto \mathbf{xor}\left(\mathbf{xor}_{0\le i<n-N:Q[i]=1}\, D[i], D[n-N]\right) * Q[n-N]=1 \rceil * \lceil d \mapsto D \rceil \rangle \\
 & \quad \langle \lceil q_k \mapsto Q * i_k \mapsto n-N * r_k \mapsto \mathbf{xor}\left(\mathbf{xor}_{0\le i<n-N:Q[i]=1}\, D[i], D[n-N]\right) * Q[n-N]=1 \rceil \rangle \\
\textsc{Consequence} & \quad \langle \lceil q_k \mapsto Q * i_k \mapsto n-N * r_k \mapsto \mathbf{xor}_{0\le i<n-(N-1):Q[i]=1}\, D[i] \rceil \rangle \\
 & \mathbf{else} \\
 & \quad \langle \lceil q_k \mapsto Q * i_k \mapsto n-N * r_k \mapsto \mathbf{xor}_{0\le i<n-N:Q[i]=1}\, D[i] * Q[n-N]=0 \rceil \rangle \\
\textsc{Skip} & \quad\quad \mathbf{skip} \\
 & \quad \langle \lceil q_k \mapsto Q * i_k \mapsto n-N * r_k \mapsto \mathbf{xor}_{0\le i<n-N:Q[i]=1}\, D[i] * Q[n-N]=0 \rceil \rangle \\
\textsc{Consequence} & \quad \langle \lceil q_k \mapsto Q * i_k \mapsto n-N * r_k \mapsto \mathbf{xor}_{0\le i<n-(N-1):Q[i]=1}\, D[i] \rceil \rangle \\
 & \langle \lceil q_k \mapsto Q * i_k \mapsto n-N * r_k \mapsto \mathbf{xor}_{0\le i<n-(N-1):Q[i]=1}\, D[i] \rceil \rangle \\
\textsc{Assign} & i_k := i_k + 1 \\
 & \langle \lceil q_k \mapsto Q * i_k \mapsto n-(N-1) * r_k \mapsto \mathbf{xor}_{0\le i<n-(N-1):Q[i]=1}\, D[i] \rceil \rangle \\
\textsc{Consequence} & \langle \varphi[N-1/R] \rangle \\
\textsc{Consequence} & \langle (\&_{R=0}^{N-1}\, \varphi) \oplus_1 (\&_{R=N}^{n}\, \varphi) \rangle
\end{array}
\tag{18}
$$

Now we move on to deriving the specification for the main program. Ultimately, the postcondition states that $\lceil r \mapsto D[K] \rceil$, *i.e.*, that we selected the correct data from the database.

$$
\dfrac{\dfrac{}{\vdash \langle \lceil x \mapsto \mathbf{onehot}(K) * d \mapsto D * \mathrm{own}(\cdots) \rceil \rangle\ q_1 :\approx \mathbf{unif}\,(\{0,1\}^n)\ \langle \lceil x \mapsto \mathbf{onehot}(K) * d \mapsto D * \mathrm{own}(\cdots) \rceil * (q_1 \sim \mathbf{unif}\,(\{0,1\}^n)) \rangle}\ \textsc{Samp} \quad (19)}{\vdash \langle \lceil x \mapsto \mathbf{onehot}(K) * d \mapsto D * \mathrm{own}(\cdots) \rceil \rangle\ q_1 :\approx \mathbf{unif}\,(\{0,1\}^n) \fatsemi q_2 := \mathbf{xor}(q_1, x) \fatsemi (\mathbf{fetch}_1 \parallel \mathbf{fetch}_2) \fatsemi r := \mathbf{xor}(r_1, r_2)\ \langle \lceil r \mapsto D[K] \rceil \rangle}\ \textsc{Seq}
$$

After dispatching the first sampling command, we move on to analyze the remainder of the program. To do so, we must do case analysis on the sampled bit-string using the Split2 rule, as shown below in (19).

$$\dfrac{\dfrac{\dfrac{}{\vdash \langle\lceil x \mapsto \mathbf{onehot}(K) * d \mapsto D * q_1 \mapsto Q * \mathsf{own}(\cdots)\rceil\rangle\ q_2 := \mathbf{xor}(q_1, x)\ \langle\lceil x \mapsto \mathbf{onehot}(K) * d \mapsto D * q_1 \mapsto Q * q_2 \mapsto \mathbf{xor}(Q, \mathbf{onehot}(K)) * \mathsf{own}(\cdots)\rceil\rangle}\ \textsc{Assign} \quad \vdots\ (20)}{\vdash \langle\lceil x \mapsto \mathbf{onehot}(K) * d \mapsto D * q_1 \mapsto Q * \mathsf{own}(\cdots)\rceil\rangle\ q_2 := \mathbf{xor}(q_1, x) \fatsemi (\mathbf{fetch}_1 \parallel \mathbf{fetch}_2) \fatsemi r := \mathbf{xor}(r_1, r_2)\ \langle\lceil r \mapsto D[X]\rceil\rangle}\ \textsc{Seq}}{\vdash \langle \bigoplus_{Q \sim \mathbf{unif}(\{0,1\}^n)} \lceil x \mapsto \mathbf{onehot}(K) * d \mapsto D * q_1 \mapsto Q * \mathsf{own}(\cdots)\rceil\rangle\ q_2 := \mathbf{xor}(q_1, x) \fatsemi (\mathbf{fetch}_1 \parallel \mathbf{fetch}_2) \fatsemi r := \mathbf{xor}(r_1, r_2)\ \langle\lceil r \mapsto D[K]\rceil\rangle}\ \textsc{Split2}}{\vdash \langle\lceil x \mapsto \mathbf{onehot}(K) * d \mapsto D * \mathsf{own}(\cdots)\rceil * (q_1 \sim \mathbf{unif}(\{0,1\}^n))\rangle\ q_2 := \mathbf{xor}(q_1, x) \fatsemi (\mathbf{fetch}_1 \parallel \mathbf{fetch}_2) \fatsemi r := \mathbf{xor}(r_1, r_2)\ \langle\lceil r \mapsto D[K]\rceil\rangle}\ \textsc{Consequence} \tag{19}$$

Next, we analyze the concurrent fetch commands. Since the shared state d is deterministic, it is easy to allocate the invariant with Share. Then, we use Par and the derivation for $\mathbf{fetch}_k$ that we showed previously. In the second thread, we must use the Subst rule, since the query is not Q, but rather a more complex logical expression.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(16)}{d \mapsto D \vdash \langle\lceil q_2 \mapsto \mathbf{xor}(Q, \mathbf{onehot}(K)) * \mathsf{own}(i_2, r_2)\rceil\rangle\ \mathbf{fetch}_2\ \langle\lceil r_2 \mapsto \mathop{\mathbf{xor}}_{0 \le i < n : \mathbf{xor}(Q, \mathbf{onehot}(K))[i]=1} D[i]\rceil\rangle}\ \textsc{Subst} \quad (16)}{d \mapsto D \vdash \langle\lceil q_1 \mapsto Q * \mathsf{own}(i_1, r_1)\rceil * \lceil q_2 \mapsto \mathbf{xor}(Q, \mathbf{onehot}(K)) * \mathsf{own}(i_2, r_2)\rceil\rangle\ \mathbf{fetch}_1 \parallel \mathbf{fetch}_2\ \langle\lceil r_1 \mapsto \mathop{\mathbf{xor}}_{0 \le i < n : Q[i]=1} D[i]\rceil * \lceil r_2 \mapsto \mathop{\mathbf{xor}}_{0 \le i < n : \mathbf{xor}(Q, \mathbf{onehot}(K))[i]=1} D[i]\rceil\rangle}\ \textsc{Par}}{d \mapsto D \vdash \langle\lceil q_1 \mapsto Q * q_2 \mapsto \mathbf{xor}(Q, \mathbf{onehot}(K)) * \mathsf{own}(\cdots)\rceil\rangle\ \mathbf{fetch}_1 \parallel \mathbf{fetch}_2\ \langle\lceil r_1 \mapsto \mathop{\mathbf{xor}}_{0 \le i < n : Q[i]=1} D[i] * r_2 \mapsto \mathop{\mathbf{xor}}_{0 \le i < n : \mathbf{xor}(Q, \mathbf{onehot}(K))[i]=1} D[i] * \mathsf{own}(r)\rceil\rangle}\ \textsc{Frame}}{\vdash \langle\lceil d \mapsto D * q_1 \mapsto Q * q_2 \mapsto \mathbf{xor}(Q, \mathbf{onehot}(K)) * \mathsf{own}(\cdots)\rceil\rangle\ \mathbf{fetch}_1 \parallel \mathbf{fetch}_2\ \langle\lceil r_1 \mapsto \mathop{\mathbf{xor}}_{0 \le i < n : Q[i]=1} D[i] * r_2 \mapsto \mathop{\mathbf{xor}}_{0 \le i < n : \mathbf{xor}(Q, \mathbf{onehot}(K))[i]=1} D[i] * d \mapsto D * \mathsf{own}(r)\rceil\rangle}\ \textsc{Share} \quad \vdots\ (21)}{\vdash \langle\lceil d \mapsto D * q_1 \mapsto Q * q_2 \mapsto \mathbf{xor}(Q, \mathbf{onehot}(K)) * \mathsf{own}(\cdots)\rceil\rangle\ (\mathbf{fetch}_1 \parallel \mathbf{fetch}_2) \fatsemi r := \mathbf{xor}(r_1, r_2)\ \langle\lceil r \mapsto D[K]\rceil\rangle}\ \textsc{Seq} \tag{20}$$

Finally, we give a derivation for the final assignment.

$$\dfrac{\dfrac{}{\vdash \langle\lceil r_1 \mapsto \mathop{\mathbf{xor}}_{0 \le i < n : Q[i]=1} D[i] * r_2 \mapsto \mathop{\mathbf{xor}}_{0 \le i < n : \mathbf{xor}(Q, \mathbf{onehot}(K))[i]=1} D[i] * \mathsf{own}(r)\rceil\rangle\ r := \mathbf{xor}(r_1, r_2)\ \langle\lceil r \mapsto D[K]\rceil\rangle}\ \textsc{Assign}}{\vdash \langle\lceil r_1 \mapsto \mathop{\mathbf{xor}}_{0 \le i < n : Q[i]=1} D[i] * r_2 \mapsto \mathop{\mathbf{xor}}_{0 \le i < n : \mathbf{xor}(Q, \mathbf{onehot}(K))[i]=1} D[i] * d \mapsto D * \mathsf{own}(r)\rceil\rangle\ r := \mathbf{xor}(r_1, r_2)\ \langle\lceil r \mapsto D[K]\rceil\rangle}\ \textsc{Consequence} \tag{21}$$

Below, we show why $\lceil \mathbf{xor}(r_1, r_2) \mapsto D[K]\rceil$:

$$\mathbf{xor}\left(\mathop{\mathbf{xor}}_{0 \le i < n : Q[i]=1} D[i], \mathop{\mathbf{xor}}_{0 \le i < n : \mathbf{xor}(Q, \mathbf{onehot}(K))[i]=1} D[i]\right) = \mathop{\mathbf{xor}}_{i=0}^{n-1} \begin{cases} \mathbf{xor}(D[i], D[i]) & \text{if } Q[i] = \mathbf{xor}(Q, \mathbf{onehot}(K))[i] = 1 \\ D[i] & \text{if } Q[i] = \neg\, \mathbf{xor}(Q, \mathbf{onehot}(K))[i] \\ 0 & \text{otherwise} \end{cases}$$

Since $\mathbf{xor}(D[i], D[i]) = 0$, we can combine the first and last cases.

$$= \underset{i=0}{\overset{n-1}{\mathbf{xor}}} \begin{cases} D[i] & \text{if } Q[i] = \neg\, \mathbf{xor}(Q, \mathbf{onehot}(K))[i] \\ 0 & \text{otherwise} \end{cases}$$

$Q[i] = \neg\, \mathbf{xor}(Q, \mathbf{onehot}(K))[i]$ occurs only when $i = K$.

$$= \underset{i=0}{\overset{n-1}{\mathbf{xor}}} \begin{cases} D[i] & \text{if } i = K \\ 0 & \text{otherwise} \end{cases}$$

Since $\mathbf{xor}(x, 0) = x$, then we can drop all the zero terms.

$$= D[K]$$

F.6 The von Neumann Trick

Recall the following program from Section 6.4, which simulates a fair coin given a coin whose bias can be altered by a parallel thread.

$$\begin{array}{l} x := 0 \fatsemi \\ y := 0 \fatsemi \\ \textbf{while } x = y \textbf{ do} \\ \quad p' := p \fatsemi \\ \quad x :\approx \mathbf{Ber}\,(p') \fatsemi \\ \quad y :\approx \mathbf{Ber}\,(p') \end{array}$$

In this section, we provide the complete derivation for the correctness of this program, showing both that it almost surely terminates, and also that x is distributed like a fair coin flip at the end of the program execution. We are going to use the resource invariant $I \triangleq (p \in \ell)$, where ℓ is a finite list of values between ε and $1 - \varepsilon$, with $0 < \varepsilon \leq \frac{1}{2}$. For the purposes of analyzing the while loop, we will use the loop invariant φ below:

$$\varphi \triangleq \varphi_0 \vee \varphi_1 \qquad \varphi_0 \triangleq \bigoplus_{X \sim \mathbf{Ber}(1/2)} \lceil x \mapsto X * y \mapsto \neg X * R = 0 \rceil \qquad \varphi_1 \triangleq \lceil x = y \mapsto \mathsf{true} * R = 1 * \mathsf{own}(p') \rceil$$

Note that this gives us $\varphi \implies \lceil x = y \mapsto R \rceil$, so when $R = 0$ the loop will terminate. Clearly, this also implies that R is bounded between 0 and 1. In addition, $\varphi[0/R] = \bigoplus_{X \in \mathbf{Ber}(1/2)} \lceil x \mapsto X * y \mapsto \neg X \rceil$ and it is precise. We start with the derivation of the command $p := p'$ in (22), which reads a value from shared state. This derivation proceeds by using the Atom rule to open the invariant and the Exists rule to get the specific value of p' at the point of the read.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash_{\mathsf{w}} \langle \lceil p \mapsto X * \mathsf{own}(x, y, p') \rceil \rangle\ p' := p\ \langle \lceil p' \mapsto X * p \mapsto X * \mathsf{own}(x, y) \rceil \rangle}\ \textsc{Assign'}}{\vdash_{\mathsf{w}} \langle \&_{X \in \ell} \lceil p \mapsto X * \mathsf{own}(x, y, p') \rceil \rangle\ p' := p\ \langle \&_{X \in \ell} \lceil p' \mapsto X * p \mapsto X * \mathsf{own}(x, y) \rceil \rangle}\ \textsc{NSplit1}}{\vdash_{\mathsf{w}} \langle \lceil x = y \mapsto \mathsf{true} * R = 1 * \mathsf{own}(p') \rceil * \lceil p \in \ell \rceil \rangle\ p' := p\ \langle (\&_{X \in \ell} \lceil p' \mapsto X * \mathsf{own}(x, y) \rceil) * \lceil p \in \ell \rceil \rangle}\ \textsc{Exists+Consequence}}{p \in \ell \vdash_{\mathsf{w}} \langle \lceil x = y \mapsto \mathsf{true} * R = N = 1 * \mathsf{own}(p') \rceil \rangle\ p' := p\ \langle \&_{X \in \ell} \lceil p' \mapsto X * \mathsf{own}(x, y) \rceil \rangle}\ \textsc{Atom} \tag{22}$$

Next, we derive a specification for the two sampling operations in (24). First, the NSplit2 rule is used to compositionally reason about the nondeterministic outcomes. In the premise of that rule, the value of p' is deterministic, so that Samp can be used twice for the writes to x and y. In the end, x and y are independently and identically distributed

according to $\mathbf{Ber}(X)$. We complete the proof using the rule of Consequence, with implication (23).

$$\begin{aligned}
\lceil p' \mapsto X \rceil * (x \sim \mathbf{Ber}(X)) * (y \sim \mathbf{Ber}(X)) &\implies \lceil p' \mapsto X \rceil * \left(\bigoplus_{Y \sim \mathbf{Ber}(X)} \lceil x \mapsto Y \rceil\right) * \left(\bigoplus_{Z \sim \mathbf{Ber}(X)} \lceil y \mapsto Z \rceil\right) \\
&\implies \bigoplus_{Y \sim \mathbf{Ber}(X)} \bigoplus_{Z \sim \mathbf{Ber}(X)} \lceil p' \mapsto X * x \mapsto Y * y \mapsto Z \rceil \\
&\implies \left(\bigoplus_{Y \sim \mathbf{Ber}(1/2)} \lceil p' \mapsto X * x \mapsto Y * y \mapsto \neg Y \rceil\right) \oplus_{2X(1-X)} \lceil x = y \mapsto \mathsf{true} * \mathsf{own}(p') \rceil \\
&\implies \varphi_0 \oplus_{2X(1-X)} \varphi_1 \\
&\implies \varphi_0 \oplus_{\geq 2\varepsilon(1-\varepsilon)} \varphi_1
\end{aligned} \tag{23}$$

Now, the derivation is shown below.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{I \vdash_{\mathsf{w}} \langle \lceil p' \mapsto X * \mathsf{own}(y) \rceil * (x \sim \mathbf{Ber}(X)) \rangle\ y :\approx \mathbf{Ber}(p')\ \langle \lceil p' \mapsto X \rceil * (x \sim \mathbf{Ber}(X)) * (y \sim \mathbf{Ber}(X)) \rangle}\ \textsc{Samp}}{\vdots}}{I \vdash_{\mathsf{w}} \langle \lceil p' \mapsto X * \mathsf{own}(x, y) \rceil \rangle\ x :\approx \mathbf{Ber}(p')\ \langle \lceil p' \mapsto X * \mathsf{own}(y) \rceil * (x \sim \mathbf{Ber}(X)) \rangle}\ \textsc{Samp}}{I \vdash_{\mathsf{w}} \langle \lceil p' \mapsto X * \mathsf{own}(x, y) \rceil \rangle\ x :\approx \mathbf{Ber}(p') \fatsemi y :\approx \mathbf{Ber}(p')\ \langle \lceil p' \mapsto X \rceil * (x \sim \mathbf{Ber}(X)) * (y \sim \mathbf{Ber}(X)) \rangle}\ \textsc{Seq}}{I \vdash_{\mathsf{w}} \langle \lceil p' \mapsto X * \mathsf{own}(x, y) \rceil \rangle\ x :\approx \mathbf{Ber}(p') \fatsemi y :\approx \mathbf{Ber}(p')\ \langle \varphi_0 \oplus_{\geq 2\varepsilon(1-\varepsilon)} \varphi_1 \rangle}\ \textsc{Consequence},(23)}{I \vdash_{\mathsf{w}} \langle \&_{X \in \ell} \lceil p' \mapsto X * \mathsf{own}(x, y) \rceil \rangle\ x :\approx \mathbf{Ber}(p') \fatsemi y :\approx \mathbf{Ber}(p')\ \langle \varphi_0 \oplus_{\geq 2\varepsilon(1-\varepsilon)} \varphi_1 \rangle}\ \textsc{NSplit2} \tag{24}$$

Note that the use of NSplit2 results in a weak triple, and given that the postcondition of (24) is not precise, it is not possible to use a strong version of the rule. However, after the loop terminates, the postcondition *is* precise, and so we can use Strengthen later, as seen in (25). Note that for the derivation of the loop body, the only possibility for $R > 0$ is $R = 1$, so $\varphi * \lceil R = N > 0 \rceil$ is equivalent to $\lceil x = y \mapsto \mathsf{true} * \mathsf{own}(p') * R = N = 1 \rceil = \varphi_1 * \lceil N = 1 \rceil$. Now, letting $p = 2\varepsilon(1-\varepsilon)$ be the minimum probability that the rank decreases, the premise of BoundedRank requires us to show that φ_0 occurs with probability at least p. This derivation is shown below.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{I \vdash_{\mathsf{w}} \langle \varphi_1 * \lceil N = 1 \rceil \rangle\ p' := p \fatsemi x :\approx \mathbf{Ber}(p') \fatsemi y :\approx \mathbf{Ber}(p')\ \langle \varphi_0 \oplus_{\geq 2\varepsilon(1-\varepsilon)} \varphi_1 \rangle}\ \textsc{Seq}}{I \vdash_{\mathsf{w}} \langle \&_{R \in \{0,1\}} \varphi \rangle\ \mathbf{while}\ x = y\ \mathbf{do}\ p' := p \fatsemi x :\approx \mathbf{Ber}(p') \fatsemi y :\approx \mathbf{Ber}(p')\ \langle \varphi_0 \rangle}\ \textsc{BoundedRank}}{I \vdash \langle \&_{R \in \{0,1\}} \varphi \rangle\ \mathbf{while}\ x = y\ \mathbf{do}\ p' := p \fatsemi x :\approx \mathbf{Ber}(p') \fatsemi y :\approx \mathbf{Ber}(p')\ \langle \varphi_0 \rangle}\ \textsc{Strengthen}}{I \vdash \langle \lceil x \mapsto 0 * y \mapsto 0 * \mathsf{own}(p') \rceil \rangle\ \mathbf{while}\ x = y\ \mathbf{do}\ p' := p \fatsemi x :\approx \mathbf{Ber}(p') \fatsemi y :\approx \mathbf{Ber}(p')\ \langle x \sim \mathbf{Ber}(1/2) \rangle}\ \textsc{Consequence} \tag{25}$$

Finally, we include the initial assignments to x and y to complete the proof.

$$\dfrac{\dfrac{\dfrac{}{I \vdash \langle \lceil \mathsf{own}(x, y, p') \rceil \rangle\ x := 0\ \langle \lceil x \mapsto 0 * \mathsf{own}(y, p') \rceil \rangle}\ \textsc{Assign} \qquad \dfrac{\dfrac{}{I \vdash \langle \lceil x \mapsto 0 * \mathsf{own}(y, p') \rceil \rangle\ y := 0\ \langle \lceil x \mapsto 0 * y \mapsto 0 * \mathsf{own}(p') \rceil \rangle}\ \textsc{Assign} \qquad (25)}{I \vdash \langle \lceil x \mapsto 0 * \mathsf{own}(y, p') \rceil \rangle\ y := 0 \fatsemi \mathbf{while}\ x = y\ \mathbf{do}\ p' := p \fatsemi x :\approx \mathbf{Ber}(p') \fatsemi y :\approx \mathbf{Ber}(p')\ \langle x \sim \mathbf{Ber}(1/2) \rangle}\ \textsc{Seq}}{I \vdash \langle \lceil \mathsf{own}(x, y, p') \rceil \rangle\ x := 0 \fatsemi y := 0 \fatsemi \mathbf{while}\ x = y\ \mathbf{do}\ p' := p \fatsemi x :\approx \mathbf{Ber}(p') \fatsemi y :\approx \mathbf{Ber}(p')\ \langle x \sim \mathbf{Ber}(1/2) \rangle}\ \textsc{Seq}$$

Received 2025-07-02; accepted 2025-11-06