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 concurrent programs, formal methods for reasoning about this mixture of effects have lagged behind. In particular, no existing program logics can express specifications about the *distributions of outcomes* resulting from programs that are both probabilistic and concurrent. To address this, we introduce *Probabilistic Concurrent Outcome Logic*, which incorporates ideas from concurrent and probabilistic separation logics into Outcome Logic to introduce new compositional reasoning principles.

1 Introduction

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

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 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. 2025], 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.

While dOL's approach for describing this 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. The reason is that the nondeterminism in the latter is *localized* to the points where the choice operator is used, whereas in a concurrent program, every step can potentially involve nondeterminism from thread interleaving. Reasoning explicitly about this 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 Logic (PSL) uses the notions of independence and conditioning to reason about the interaction between randomness and control flow [Bao et al. 2024; Barthe et al. 2019; Li et al. 2023]. These logics achieve compositional reasoning in concurrent and

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.

probabilistic settings, respectively, so combining their reasoning principles together appears to be a natural way to derive a compositional logic for the combination of these effects. However, as we will see in Section 2, such a combination is challenging to achieve because the semantic models of each logic are quite different and highly specialized to their respective domains. To combine these reasoning principles in pcOL, we develop a new semantic model that captures causality between actions by threads, while also properly sequencing decisions of the scheduler with probabilistic choices made during the program execution. Our contributions are as follows:

Invariant Sensitive Semantics. In Section 3, we formulate a denotational model of probabilistic concurrency relative to an *invariant*—a contract about shared state that must remain true at all times. CSL specifications permit shared state to be nondeterministically altered between steps of execution (*i.e.,* by another thread) [Vafeiadis 2011], an inherently *operational* property. By contrast, probabilistic assertions require us to model the entire distribution of outcomes, which is more readily done using *denotational* methods. While denotational models of CSL do exist [Brookes 2004], the inclusion of probabilistic choice introduces further complexity; interference between threads results in a *distribution* over shared state, not a single value, and the alteration of shared state must be properly sequenced with random sampling and scheduling of threads. We develop an invariant sensitive semantics in Section 3, which ensures that the invariant is always true while retaining the denotation of the program as a distribution of outcomes.

Probabilistic Concurrent Outcome Logic (pcOL). We introduce pcOL in Sections 4 and 5, the first logic to use probabilistic independence and conditioning to reason about concurrent programs. Although our logic has familiar inference rules, the soundness proofs for those rules are substantially more complex. For example, in the parallel composition rule (Par), we not only show that the resources owned by the two threads end up having the proper values, but also that the *joint distribution* over those values is correct. This relies on the fact that the nondeterminism introduced by the scheduler is not *measurable*, a property which has not arisen in prior probabilistic separation logics, since they did not support nondeterminism.

Case Studies. In Section 6 we give three examples using pcOL to prove correctness of protocols that mix randomness from concurrent threads, and privately retrieve information from a database.

2 Overview: Familiar Reasoning Principles in a New Setting

A major hurdle in concurrency analysis is that the semantics of programs is not *compositional*; two programs can have completely different behavior when run in parallel than they do when run in isolation. Reasoning about programs by enumerating all possible interleavings of the threads is not a viable strategy, so restrictions must be introduced to enable sound compositional analysis. This can be achieved via *separation*—if two concurrent threads act on disjoint regions of memory, then their behavior will not change when run in parallel—giving rise to the Par rule of Concurrent Separation Logic [Brookes 2004; O'Hearn 2004], where the *separating conjunction* $\varphi * \psi$ means that the machines's resources (*i.e.,* memory cells) can be divided to satisfy φ and ψ individually.

$$\frac{\langle \varphi_1 \rangle\ C_1\ \langle \psi_1 \rangle \qquad \langle \varphi_2 \rangle\ C_2\ \langle \psi_2 \rangle}{\langle \varphi_1 * \varphi_2 \rangle\ C_1 \parallel C_2\ \langle \psi_1 * \psi_2 \rangle}\ \textsc{Par}$$

We now ask, *what if C_1 and C_2 are probabilistic programs?* For example, suppose that we wanted to flip two coins in parallel and store the results in the variables x and y. After running each thread, 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.

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

After composing these two programs in parallel, the final specification will ideally dictate not only how x and y are distributed, but also their *joint distribution*. This is achieved using a probabilistic interpretation of the separating conjunction $\varphi * \psi$, stating that the events described by φ and ψ are *probabilistically independent* [Barthe et al. 2019]. If the Par rule held under this probabilistic interpretation, we could derive the following specification, which is indeed valid in this case.

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

Here, the independence guarantee of the separating conjunction means that each outcome (*e.g.*, x and y are both 1) occurs with probability $\frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$. Intuitively, a probabilistic interpretation of Par is justifiable because C_1 and C_2 execute without interaction, so the there is no correlation between the random behaviors that they introduce.

Bigger challenges arise when there is shared state, which is handled with *invariants* in CSL. An invariant I is an assertion about shared state that must remain true at all times. Threads can interact with shared state as long as I is preserved by every action. This is achieved in CSL with the following inference rules; a more general Par rule separates the pre- and postcondition, but allows the invariant to be used in both threads. The Atom rule *opens* the invariant, as long as the program is a single atomic commands a.

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

Under a probabilistic interpreation of $*$, this stronger Par rule is valid when the shared state described by the invariant I is *deterministic*, for example, as in:

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

However, when the shared state is randomized, then the rule is unsound: if the threads interact with shared state, then the final events may be correlated, such as in the following program where both x and y are derived from z.

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

Here, x and y are not independently distributed, so how can we enable compositional reasoning when shared state is randomized? The key observation is that in example (1), although x and y are clearly not independently distributed, their distributions are *conditionally* independent on z. Therefore, we can first *condition* on the possible outcomes of the shared state, thereby *determinizing* it, and then use the Par rule.

Logically, conditioning corresponds to breaking down the outcomes of the sampling operation into cases, a common operation in both Outcome Logic [Zilberstein 2024; Zilberstein et al. 2023, 2025] and the probabilistic separation logics Lilac [Li et al. 2023] and Bluebell [Bao et al. 2024]. We expand the $z \sim \mathbf{Ber}\left(\frac{1}{2}\right)$ assertion using an *outcome conjunction* $\bigoplus_{Z \sim \mathbf{Ber}\left(\frac{1}{2}\right)} \lceil z \mapsto Z\rceil$, which binds a new logical variable Z. We can then reason about each outcome individually, where z is deterministic, so the Par applies to obtain the following spec (the full derivation is shown in Appendix D, Figure 9).

$$z \mapsto Z \vdash \langle\lceil x \mapsto -\rceil * \lceil y \mapsto -\rceil\rangle\ x \coloneqq z \parallel y \coloneqq 1 - z\ \langle\lceil x \mapsto Z\rceil * \lceil y \mapsto 1 - Z\rceil\rangle$$

While the outcome conjunction above resembles the *conditioning modalities* from Lilac and Bluebell, its semantics is tailored for a concurrency logic. As we explain in Section 4.3, the conditioning rules used in those logics do not accommodate resources owned by parallel threads. As a result, our outcome conjunction has a different semantics, based on the measure theoretic notion of *direct sums* [Fremlin 2001], which makes our conditioning rules (Cond1 and Cond2) fully compositional.

While it may seem like a natural idea to reinterpret CSL's Par rule using probabilistic separation, the semantic model needed to support this sort of reasoning is considerably more complex. Our semantics needs to capture both the operational notion that the distribution over shared state may change between each *step* of computation, while also retaining the denotation of the program as a distribution of outcomes. In addition, the nondeterminism introduced by scheduling concurrent threads makes independence very subtle, even without shared state. For example, in the following program, the scheduler can choose to make y always equal to x at the end of the program execution, thereby introducing a correlation even though x and y are not even used in the same thread.

$$x :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right) \parallel y \coloneqq 0 \parallel y \coloneqq 1$$

This is achieved by executing the sampling operation first, and then, depending on the outcome, choosing the order of assignments to y so that y is first assigned $1 - x$ and then is overridden to x.

To combat this, we bring the notion of *precise* assertions from standard separation logic [Calcagno et al. 2007] into the probabilistic setting to ensure that the effects of nondeterminism are not measurable. Requiring every event to have a *precise* probability means that all we can say about y in the program above is that $\lceil y \in \{0, 1\} \rceil$, an event with probability 1. This allows us to establish the soundness of Par; the crux of the proof is to show that if two events have precise probabilities p and q when executed individually, then they both occur with probability $p \cdot q$ (Lemma A.8) when executed in parallel. The proof is considerably more complex than that of standard CSL, since it must track probabilities of events in addition to physical resources.

We begin the technical development in Section 3 by defining a denotational model for probabilistic concurrent programs. We then give a measure theoretic model of assertions—including the separation and outcome conjunctions— in Section 4. In Section 5, we define the validity of pcOL triples and provide a proof system. Three examples are shown in Section 6 before we conclude by discussing related work in Section 7 and limitations and future work in Section 8.

3 Denotational Semantics for Probabilistic Concurrency

We begin by describing the syntax and semantics of the programming language that we will be using throughout the paper. The syntax is shown in Figure 1. Programs are represented as commands $C \in \mathsf{Cmd}$, which consist of no-ops (**skip**), sequential composition ($C_1 \fatsemi C_2$), parallel composition ($C_1 \parallel C_2$), if statements, bounded loops, and actions $a \in \mathsf{Act}$.

Whereas commands consist of standard operations for a concurrent language, the novelty lies in actions, which 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 list of values $[v_1, \ldots, v_n]$, that are then each returned with probability $1/n$.

In addition to sampling, we also have deterministic assignment $x \coloneqq e$, 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.

The addition of random sampling presents several challenges in defining a proper semantics for the language, as no existing techniques are adequate. Concurrent semantics for purely nondeterministic languages are often given operationally by a small step relation. This approach is not ideal

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

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

for probabilistic computation, since we want to know not only the states that the program could step to, but also the *distribution* over those states.

In denotational semantics, concurrency is often dealt with using *partially ordered multisets* (pomsets), which record the causality between actions in the program [Gischer 1988; Pratt 1986]. Each pomset represents the possible interleavings of actions within a single trace, but if the program has multiple traces due to control flow branching such as if statements, then a single pomset is insufficient. Instead, a *pomset language*—a set of pomsets—is used, where each pomset in the language corresponds to a particular sequence of branches that the program execution may take.

The problem with pomset languages is that they lose information about when the decision takes place. This effectively forces the scheduler to decide whether or not a test will be true *before* that test is resolved, and presents problems when the language contains randomization, since the test may depend on probabilistic sampling operations that occur *later* in the execution.

We solve this problem with a new pomset structure, which we describe in Section 3.1. Next, in Section 3.2, we describe the interpretation of actions that are both probabilistic and nondeterministic using the convex powerset monad [Jacobs 2008; Mislove 2000; Morgan et al. 1996b; Tix 2000]. Finally, in Section 3.3, we explain how to convert the pomset semantics into a state transformer, so that we can reason about the behavior of programs.

3.1 Trace Semantics with Uninterpreted Actions and Tests

We begin by giving a trace semantics for commands, leaving actions and tests uninterpreted for the time being. The purpose of using uninterpreted atoms is to keep the semantics compositional; since the evaluation of those atoms may depend on the way that they are interleaved, we cannot evaluate them unless we know that no other threads will be composed in parallel. As we mentioned previously, interpreting a probabilistic and concurrent language requires a new semantic structure, as pomset languages do not account for the proper sequencing of probabilistic and nondeterministic actions. This new structure is able to record guarded branching behavior in addition to causality; we call it a *pomset with tests*.

A pomset with tests, similarly to the classical case, will be an isomorphism class of tuples. In our case, we will need 4-tuples $\langle L, \leq, T, \lambda \rangle$, which we will now describe. The first component $L \subseteq_{\mathsf{fin}} \mathsf{Label}$ is a finite set of labels, representing the nodes in the pomset. The relation $\leq\, \subseteq L \times L$ is a partial order on L, which represents causality; if $\ell \leq \ell'$, then ℓ must be scheduled before ℓ'.

Our new addition (to classical pomsets) is the function $T \colon \mathbb{B} \to L \times L$, which we call the *test relation*. If $(\ell, \ell') \in T(1)$, then ℓ' can only be scheduled if the test represented by node ℓ passes, and similarly, $(\ell, \ell') \in T(0)$ means that ℓ' can only be scheduled if the test at node ℓ fails. We additionally require that $T(b) \subseteq\, \leq$ for all $b \in \mathbb{B}$, to ensure that tests are evaluated before the actions that depend on them.

Finally, $\lambda \colon L \to \mathsf{Act} + \mathsf{Test}$ is a *labelling function*, which assigns an action or a test to each node ℓ. We say that two such structures are isomorphic if they are equal modulo relabelling. More precisely, $\langle L, \leq, T, \lambda \rangle \equiv \langle L', \leq', T', \lambda' \rangle$ iff there exists a bijection $f \colon L \to L'$ such that (1) $\ell \leq \ell'$ iff $f(\ell) \leq' f(\ell')$; (2) $(\ell, \ell) \in T(b)$ iff $(f(\ell), f(\ell')) \in T'(b)$; and (3) $\lambda(\ell) = \lambda'(f(\ell))$ for all $\ell, \ell' \in L$. An isomorphism class of these 4-tuples can then be defined in the usual way:

$$\llparenthesis L, \leq, T, \lambda \rrparenthesis = \{ \langle L', \leq', T', \lambda' \rangle \mid \langle L, \leq, T, \lambda \rangle \equiv \langle L', \leq', T', \lambda' \rangle \}$$

A *pomset with tests* is an isomorphism class on these structures, that is an element of the set:

$$\mathcal{P}\mathsf{om} \triangleq \{ \llparenthesis L, \leq, T, \lambda \rrparenthesis \mid \langle L, \leq, T, \lambda \rangle \text{ as described above} \}$$

Since we only use pomsets with tests in this paper, from now on we refer to them as simply pomsets. When we define operations on pomsets, we will define them on a single representative structure, however the fact that they are an isomorphism class will allow us to assume that any two pomsets have disjoint sets of labels (if they were not disjoint, then we could simply pick an isomorphic representative structure that is disjoint). Going forward, we will use the metavariables α, β to denote pomsets, and for some pomset α, we let L_α, $\leq_\alpha$, T_α, and λ_α be the constituent parts of its internal structure. We can now define several operations on pomsets. First of all, the empty pomset $\emptyset_{\mathcal{P}\mathsf{om}} \triangleq \{ \langle \emptyset, \emptyset, T_\emptyset, \lambda_\emptyset \rangle \}$ has an empty label set, an empty order, and $T(b) = \emptyset$ and $\lambda_\emptyset \colon \emptyset \to \mathsf{Act} + \mathsf{Test}$ is the trivial function that takes no inputs. The following function constructs singleton pomsets:

$$\mathsf{singleton}(x) \triangleq \{ \langle \{\ell\}, \{\ell, \ell\}, T_\emptyset, \lambda_{\ell,x} \rangle \mid \ell \in \mathsf{Label} \} \qquad \text{where} \quad \lambda_{\ell,x}(\ell) = x$$

We also provide two injection functions for constructing coproducts from actions and tests:

$$\mathbb{i}_{\mathsf{Act}} \colon \mathsf{Act} \to \mathsf{Act} + \mathsf{Test} \qquad\qquad \mathbb{i}_{\mathsf{Test}} \colon \mathsf{Test} \to \mathsf{Act} + \mathsf{Test}$$

We define the minimal elements of a pomset to be all the labels ℓ for which no other label is smaller than ℓ. Operationally speaking, those minimal nodes are the ones that are ready to be scheduled.

$$\min(\alpha) \triangleq \{ \ell \in L_\alpha \mid \forall \ell' \in \ell.\ \ell' \leq_\alpha \ell \Rightarrow \ell' = \ell \}$$

We now define an operator $\fatsemi$, which performs sequential composition by joining the label sets and making every element of the first set smaller than every element of the second. We achieve this using a sequential composition operation on partial orders $\fatsemi \colon L \times L \to L' \times L' \to (L \uplus L') \times (L \uplus L')$.

$$\leq \fatsemi \leq' \triangleq \leq \cup \leq' \cup (L \times L')$$

We lift this to a $\fatsemi$ operation on pomsets, by using a representative structure of the pomset. Parallel composition $\parallel$ is similar, but there is no causality between the nodes in the two pomsets, so we take a standard union of the partial orders.

$$\alpha \fatsemi \beta \triangleq \llparenthesis L_\alpha \uplus L_\beta, \leq_\alpha \fatsemi \leq_\beta, T_\alpha \uplus T_\beta, \lambda_\alpha \uplus \lambda_\beta \rrparenthesis \qquad \alpha \parallel \beta \triangleq \llparenthesis L_\alpha \uplus L_\beta, \leq_\alpha \cup \leq_\beta, T_\alpha \uplus T_\beta, \lambda_\alpha \uplus \lambda_\beta \rrparenthesis$$

Whereas all of the definitions above have been relatively standard, we now define a novel guarding operation $\mathsf{guard} \colon \mathsf{Test} \times \mathcal{P}\mathsf{om} \times \mathcal{P}\mathsf{om} \to \mathcal{P}\mathsf{om}$, which is how we construct pomsets that contain information about the control flow structure of programs. More precisely, $\mathsf{guard}(b, \alpha, \beta)$ will ensure that the pomset α is executed if b resolves to true, and β will get executed in b resolves to false. We define this operation as follows:

$$\mathsf{guard}(b, \alpha, \beta) \triangleq \llparenthesis L, \leq, T, \lambda \rrparenthesis$$

Where the structure $\llparenthesis L, \leq, T, \lambda \rrparenthesis$ is constructed as follows. Let $\langle \{\ell\}, \leq', T', \lambda' \rangle = \mathsf{singleton}(\mathbb{i}_{\mathsf{Test}}(b))$ be some representative structure. The set of labels $L \triangleq \{\ell\} \uplus L_\alpha \uplus L_\beta$ is the set of labels from both branches, plus the new label for the test. The partial order $\leq' \triangleq (\{\ell\} \times L) \cup \leq_\alpha \cup \leq_\beta$ augments the orders of the branches by adding a causal relationship from the test node to everything else. The test relation is $T(1) \triangleq (\{\ell\} \times L_\alpha) \cup T_\alpha(1) \cup T_\beta(1)$ and $T(0) \triangleq (\{\ell\} \times L_\beta) \cup T_\alpha(0) \cup T_\beta(0)$ adds dependencies from the test node to the two branches, selecting the appropriate branch to

$$\begin{aligned}
[\![\mathbf{skip}]\!] &\triangleq \emptyset_{\mathcal{P}\mathrm{om}} \\
[\![C_1 \fatsemi C_2]\!] &\triangleq [\![C_1]\!] \fatsemi [\![C_2]\!] \\
[\![C_1 \parallel C_2]\!] &\triangleq [\![C_1]\!] \parallel [\![C_2]\!] \\
[\![\mathbf{if}\ b\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2]\!] &\triangleq \mathsf{guard}(b, [\![C_1]\!], [\![C_2]\!]) \\
[\![\mathbf{for}\ n\ \mathbf{do}\ C]\!] &\triangleq [\![C]\!]^n \\
[\![a]\!] &\triangleq \mathsf{singleton}(\mathbb{i}_{\mathsf{Act}}(a))
\end{aligned}$$

Fig. 2. Trace semantics for commands $[\![-]\!] : \mathsf{Cmd} \to \mathcal{P}\mathsf{om}$. Note that $\alpha^0 = \emptyset_{\mathcal{P}\mathrm{om}}$ and $\alpha^{n+1} = \alpha \fatsemi \alpha^n$.

correspond to whether the test is true or false. Finally, the labelling function λ simply agrees with the labelings of the branches, or returns $\mathbb{i}_{\mathsf{Test}}(b)$ on the new label.

We are now ready to define the semantics of our language, shown in Figure 2. The operations of the commands correspond exactly to the pomset operations that we just described, with the exception of loops, which simply repeat the command C the appropriate number of times. We only include bounded iteration because pomsets with tests do not carry an obvious dcpo structure; they will need to be enriched before we can represent computations of (possibly) infinite length [de Bakker and Warmerdam 1990; Meyer and de Vink 1989], which we discuss further in Section 8. We remark that other PSL variants also impose bounds on loops, although those bounds can depend on deterministic program state [Bao et al. 2024; Barthe et al. 2019; Li et al. 2023].

3.2 Interpreting Actions and Tests

We now discuss the semantics of actions and tests. Actions 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 typical representations of nondeterminism (powersets) [Varacca and Winskel 2006; Zwart and Marsden 2019]. We instead use the *convex powerset* $\mathcal{C}$ in the denotational semantics of actions, whose type is shown below:

$$[\![-]\!]_{\mathsf{Act}} : \mathsf{Act} \to \mathsf{Mem}[S] \to \mathcal{C}(\mathsf{Mem}[S]) \qquad \text{and} \qquad [\![-]\!]_{\mathsf{Test}} : \mathsf{Test} \to \mathsf{Mem}[S] \to \mathbb{B},$$

Above, $\mathsf{Mem}[S]$ is a *memory* over a set of variables S and $\mathbb{B} = \{0, 1\}$ is the Booleans. These components will be described throughout the remainder of the section.

Memories, Expressions, and Tests. A memory $\sigma \in \mathsf{Mem}[S]$ is a mapping from a particular set of variables $S \subseteq \mathsf{Var}$ to values, that is, $\mathsf{Mem}[S] \triangleq S \to \mathsf{Val}$, where values Val consist of integers, rationals, and lists. We can combine two memories using the disjoint union $\uplus\colon \mathsf{Mem}[S] \to \mathsf{Mem}[T] \to \mathsf{Mem}[S \cup T]$ as long as $S \cap T = \emptyset$, and we define a similar operation $A * B$ 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. Dually, 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 $\mathsf{fv}(e) \subseteq S$, if not then $[\![e]\!]_{\mathsf{Exp}}(\sigma)$ is undefined. The same is true for tests and $[\![b]\!]_{\mathsf{Test}} : \mathsf{Mem}[S] \to \mathbb{B}$ is defined as long as $\mathsf{fv}(b) \subseteq S$.

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. We also extend the previously defined projections to distributions $\pi_S : \mathcal{D}(\mathsf{Mem}[T]) \to \mathcal{D}(\mathsf{Mem}[S \cap T])$ by marginalizing as follows $\pi_S(\mu)(\sigma) \triangleq \sum_{\tau \in \mathsf{Mem}[T \setminus S]} \mu(\sigma \uplus \tau)$.

Actions and the Convex Powerset. The convex combination of two distributions $\mu \oplus_p \nu$ is defined as $(\mu \oplus_p \nu)(x) = p \cdot \mu(x) + (1 - p) \cdot \nu(x)$. We call a set of distributions $S \subseteq \mathcal{D}(X)$ *convex* if it is closed under convex combinations. That is, $(\mu \oplus_p \nu) \in S$ for every $\mu, \nu \in S$ and $p \in [0, 1]$. Our domain of computation consists of nonempty and convex sets of distributions.

$$\mathcal{C}(X) \triangleq \{S \subseteq \mathcal{D}(X + \{\bot\}) \mid S \text{ is nonempty and convex}\}$$

We include $\bot$ to represent undefined behavior such as accessing a variable that is not in the current memory. Non-emptiness ensures that the semantics is not vacuous, since undefined behavior is represented by $\{\delta_\bot\}$ rather than $\emptyset$. Finally, convexity ensures that $\mathcal{C}$ carries a *monad* structure [Jacobs 2008], making the sequencing of actions compositional. More precisely, there is a unit $\eta : X \to \mathcal{C}(X)$ and Kleisli extension $(-)^\dagger : (X \to \mathcal{C}(Y)) \to \mathcal{C}(X) \to \mathcal{C}(Y)$, which obey the monad laws: $\eta^\dagger = \mathsf{id}$, $f^\dagger \circ \eta = f$, and $f^\dagger \circ g^\dagger = (f^\dagger \circ g)^\dagger$. These operations are defined as follows:

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

Where above $f_\bot(x) = f(x)$ for $x \in X$ and $f_\bot(\bot) = \eta(\bot)$. As an overloading of notation, we will occasionally write $f^\dagger(\mu)$ to mean $f^\dagger(\{\mu\})$. For a more complete explanation of convex powerset semantics, refer to He et al. [1997] and Zilberstein et al. [2025]. We can now use the convex powerset to give semantics to actions $[\![a]\!]_{\mathsf{Act}} : \mathsf{Mem}[S] \to \mathcal{C}(\mathsf{Mem}[S])$.

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

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

Before we conclude, we define a few more operations on convex sets. First, the convex combination operation $S \oplus_p T \triangleq \{\mu \oplus_p \nu \mid \mu \in S, \nu \in T\}$, and also the convex union $S \,\&\, T \triangleq \bigcup_{p \in [0,1]} S \oplus_p T$. 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 $\sigma \uplus \bot = \bot$.

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].

3.3 State Transformer Semantics via Linearization

Before building a logic to reason about programs, we need a way to interpret programs as state transformers, which we achieve using a *linearization* operation $\mathcal{L} : \mathcal{P}\mathsf{om} \to \mathsf{Mem}[S] \to \mathcal{C}(\mathsf{Mem}[S])$. Typical linearization operations map pomsets to sets of linearly ordered traces, which can then be interpreted [Gischer 1988], however that approach is insufficient in the presence of probabilistic choice. Instead, $\mathcal{L}$ must ensure that scheduling of actions is properly sequenced with respect to probabilistic choices, so that we properly model the scheduler's ability to observe the outcomes of random sampling events, as we discussed in Section 2. Linearizing makes the semantics no longer compositional, so it is only suitable for determining the semantics of a *complete* program, or for two programs that can only interact with shared state in specific ways, as we will see in Section 5.

As we alluded to in Section 2, the way in which we will reason about shared state is via *invariants*—assertions about shared state that must be preserved by every atomic action. Because the shared state may be modified at any instant by another thread, we define an *invariant sensitive semantics*, in which the scheduler may alter shared state before executing each atomic action. This will be based on *semantic invariants*, finite sets of memories $\mathcal{I} \subseteq_{\mathsf{fin}} \mathsf{Mem}[T]$ that represent the legal states, and must include only variables that are disjoint from the standard program states. We limit invariants to be finite sets in order to avoid issues related to unbounded nondeterminism [Apt and Plotkin 1986]. We start by defining invariant sensitive action evaluation $[\![a]\!]^{\mathcal{I}}_{\mathsf{Act}} : \mathsf{Mem}[S] \to \mathcal{C}(\mathsf{Mem}[S])$.

$$[\![a]\!]^{\mathcal{I}}_{\mathsf{Act}}(\sigma) \triangleq \mathop{\&}_{\tau\in\mathcal{I}} \begin{cases} \pi_S\left([\![a]\!]_{\mathsf{Act}}(\sigma \uplus \tau)\right) & \text{if } \bigcup\{\pi_T(\mathsf{supp}(\mu)) \mid \mu \in [\![a]\!]_{\mathsf{Act}}(\sigma \uplus \tau)\} \subseteq \mathcal{I} \\ \eta(\bot) & \text{otherwise} \end{cases}$$

The convex union over $\tau \in \mathcal{I}$ represents the scheduler choosing a memory τ to satisfy the invariant. The action a is then evaluated normally on the combined memory $\sigma \uplus \tau$. If the T projection of every possible resulting state remains within $\mathcal{I}$, then the invariant is upheld and so we get the S projection of the result. If not, then the invariant was violated, so we return $\eta(\bot)$ 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.

Next, we must define two removal operations on pomsets. First, we let $\alpha \setminus L'$ simply remove the labels in L' from the pomset, and accordingly removes them from the relations and labelling function. In addition, we define a filtering operation as follows:

$$\mathsf{filter}(\alpha, \ell, b, \sigma) \triangleq \alpha \setminus \left(\ell \cup \left\{\ell' \in L_\alpha \mid (\ell, \ell') \in T_\alpha\left(\neg [\![b]\!]_{\mathsf{Test}}(\sigma)\right)\right\}\right)$$

This operation removes the nodes from α that disagree on the outcome of the test b, evaluated at state σ. Note that this is not equivalent to keeping the nodes that *do* agree with b, since many nodes do not depend on b at all. It also removes ℓ, the label of b, since ℓ has already been executed.

We are now ready to define invariant sensitive linearization. Given a semantic invariant $\mathcal{I} \subseteq_{\mathsf{fin}} \mathsf{Mem}[T]$ such that $S \cap T = \emptyset$, we define $\mathcal{L}^{\mathcal{I}} : \mathcal{P}\mathsf{om} \to \mathsf{Mem}[S] \to \mathcal{C}(\mathsf{Mem}[S])$ inductively on the structure of the pomset, where below we assume that α is nonempty.

$$\mathcal{L}^{\mathcal{I}}(\emptyset_{\mathcal{P}\mathsf{om}})(\sigma) \triangleq \eta(\sigma) \qquad \mathcal{L}^{\mathcal{I}}(\alpha)(\sigma) \triangleq \mathop{\&}_{\ell\in\min(\alpha)} \begin{cases} \mathcal{L}^{\mathcal{I}}(\alpha \setminus \{\ell\})^\dagger([\![a]\!]^{\mathcal{I}}_{\mathsf{Act}}(\sigma)) & \text{if } \lambda_\alpha(\ell) = \mathbb{i}_{\mathsf{Act}}(a) \\ \mathcal{L}^{\mathcal{I}}(\mathsf{filter}(\alpha, \ell, b, \sigma))(\sigma) & \text{if } \lambda_\alpha(\ell) = \mathbb{i}_{\mathsf{Test}}(b) \end{cases}$$

An empty pomset contains no actions, so its linearization is just the unit function. For a nonempty pomset, the scheduler first picks some minimal node ℓ to schedule. If ℓ represents an action, then the action is evaluated in the invariant sensitive way, and then the result is sequenced with the evaluation of the remainder of the pomset using Kleisli composition. If ℓ is instead a test, then the pomset is filtered based on the resolution of the test on memory σ, and then run on σ.

Linearization has several expected properties, as shown below. That is, it behaves well with respect to sequential composition, if statements, and bounded looping (Lemmas A.2 to A.4).

$$\mathcal{L}^{\mathcal{I}}([\![C_1 \fatsemi C_2]\!])(\sigma) = \mathcal{L}^{\mathcal{I}}([\![C_2]\!])^\dagger(\mathcal{L}^{\mathcal{I}}([\![C_1]\!])(\sigma)) \qquad \mathcal{L}^{\mathcal{I}}([\![\mathbf{for}\ n\ \mathbf{do}\ C]\!])(\sigma) = \mathcal{L}^{\mathcal{I}}([\![C]\!])^n(\sigma)$$

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

Where $f^0 = \eta$ and $f^{n+1} = f^\dagger \circ f^n$. Linearization also obeys a monotonicity property (Lemma A.7) with respect to invariants. That is, moving more resources into the invariant can only add more behaviors, as long as $\bot$ does not appear in the result set. More formally, for any pomset α and any $\sigma \in \mathsf{Mem}[S]$ and $\mathcal{I} \subseteq_{\mathsf{fin}} \mathsf{Mem}[T]$ and $\mathcal{J} \subseteq_{\mathsf{fin}} \mathsf{Mem}[U]$ such that S, T, and U are all pairwise disjoint, if $\bot \notin \mathrm{supp}(\nu)$ for all $\nu \in \mathcal{L}^{\mathcal{I} * \mathcal{J}}(\alpha)(\sigma)$, then:

$$\forall \tau \in \mathcal{I}. \quad \pi_S\left(\mathcal{L}^{\mathcal{J}}(\alpha)(\sigma \uplus \tau)\right) \quad \subseteq \quad \mathcal{L}^{\mathcal{I} * \mathcal{J}}(\alpha)(\sigma)$$

Let $\mathcal{L} \triangleq \mathcal{L}^{\{\mathsf{emp}\}}$ and note that in $\mathcal{L}(\alpha)(\sigma)$, the scheduler does not introduce any new behavior, since all actions are evaluated according to $\llbracket a \rrbracket^{\{\mathsf{emp}\}}_{\mathsf{Act}} = \llbracket a \rrbracket_{\mathsf{Act}}$, so $\mathcal{L}(\alpha)(\sigma)$ can be viewed as the *true* semantics of the program. The monotonicity lemma above guarantees that adding an invariant will only add new behaviors, so safety properties about $\mathcal{L}^{\mathcal{I}}(\alpha)(\sigma)$ will automatically carry over to $\mathcal{L}(\alpha)(\sigma \uplus \tau)$ for any $\tau \in \mathcal{I}$, which shows up in the soundness of the Share rule later on.

4 The Model of Probabilistic Assertions

In this section, we discuss the assertions that we use as pre- and postconditions in Probabilistic Concurrent Outcome Logic (pcOL). These assertions are inspired by standard Outcome Logic [Zilberstein 2024; Zilberstein et al. 2023], but we reinterpret the *outcome conjunction* as a *conditioning modality* like those found in the probabilistic separation logics Lilac [Li et al. 2023] and Bluebell [Bao et al. 2024].

Like Lilac and Bluebell, we use *probability spaces* rather than distributions as models of our assertions. Unlike distributions, which assign probability to every memory, probability spaces only assign mass to measurable events, which are sets of memories in this case. As such, these logics are able to express that particular *events* are independent, rather than the entire distributions resulting from the program execution. For example, in the program $x :\approx \mathbf{Ber}\left(\frac{1}{2}\right) \fatsemi y := x$, clearly x and y are correlated, but the assertion $\left(x \sim \mathbf{Ber}\left(\frac{1}{2}\right)\right) * \lceil y \in \{0, 1\} \rceil$ is still valid, since the event $y \in \{0, 1\}$ occurs with probability 1, and we cannot measure anything finer.

To formalize these ideas, 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. Finally, we give our definition of 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.

$$\begin{aligned} P &::= \mathsf{true} \mid \mathsf{false} \mid P \land Q \mid P \lor Q \mid P * Q \mid \exists X.\, P \mid e \mapsto E \mid E_1 \asymp E_2 \\ E &::= X \mid v \mid E_1 + E_2 \mid E_1 \cdot E_2 \mid \cdots \end{aligned}$$

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 $\Gamma \colon \mathsf{LExp} \to \mathsf{Val}$ is written $\llbracket E \rrbracket_{\mathsf{LExp}}(\Gamma)$ and 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 3. 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.

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

Fig. 3. Satisfaction relation for pure assertions.

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 $\asymp \in \{=, \leq, \cdots\}$ ranges over the same comparators that we saw 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:

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

Finally, we provide syntactic sugar $e \in S$ for asserting that e belongs to a finite set $S = \{v_1, \ldots, v_n\}$, and $\mathsf{own}(e)$ for asserting that the resources of e are owned by the current thread.

$$e \in S \triangleq \exists X.e \mapsto X * (X = v_1 \lor \cdots \lor X = v_n) \qquad\qquad \mathsf{own}(e_1, \ldots, e_n) \triangleq \mathop{\Huge *}_{i=1}^{n} \exists X_i.\, e_i \mapsto X_i$$

4.2 Measure Theory and Probability Spaces

We now introduce basic definitions related to measure theory and probability spaces. 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 $\mathcal{F}$. For a probability space $\mathcal{P}$, we will use $\Omega_\mathcal{P}$, $\mathcal{F}_\mathcal{P}$, and $\mu_\mathcal{P}$ to refer to its respective parts.

We also 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 $\Omega_\mathcal{P} \subset \mathsf{Mem}[S]$ can be *completed* as follows: $\mathsf{comp}(\mathcal{P}) \triangleq \langle \mathsf{Mem}[S], \mathcal{F}, \mu \rangle$ where $\mathcal{F} \triangleq \{A \cup B \mid A \in \mathcal{F}_\mathcal{P}, B \subseteq \mathsf{Mem}[S] \setminus \Omega_\mathcal{P}\}$ and $\mu(A) \triangleq \mu_\mathcal{P}(A \cap \Omega_\mathcal{P})$.

We now define an order on probability spaces. As is typical in intuitionistic logic, this order $\mathcal{P} \preceq \mathcal{Q}$ will indicate when $\mathcal{Q}$ contains more information than $\mathcal{P}$. The information can be gained across two dimensions: we can expand the memory footprint to gain more information about other variables, or we can make 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:

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

So, $\mathcal{P} \preceq \mathcal{Q}$ iff $\mathcal{P}$ contains a smaller sample space and fewer measurable events, but $\mathcal{P}$ and $\mathcal{Q}$ agree on the probability of events whose projections are measurable in $\mathcal{P}$. For a proper distribution $\mu \in \mathcal{D}(\mathsf{Mem}[S])$, we will write $\mathcal{P} \preceq \mu$ to mean that $\mathcal{P} \preceq \mathcal{Q}$, where $\Omega_\mathcal{Q} = \mathsf{Mem}[S]$, $\mathcal{F}_\mathcal{Q} = 2^{\mathsf{Mem}[S]}$ is the greatest σ-algebra, and $\mu_\mathcal{Q}(A) = \sum_{\sigma \in A} \mu(\sigma)$.

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 independent product $\mathcal{P} \otimes \mathcal{Q}$, which is defined when $\Omega_\mathcal{P} \subseteq \mathsf{Mem}[S]$, $\Omega_\mathcal{Q} \subseteq \mathsf{Mem}[T]$, and $S \cap T = \emptyset$.

$$\mathcal{P} \otimes \mathcal{Q} \triangleq \langle \Omega_\mathcal{P} * \Omega_\mathcal{Q}, \{A * B \mid A \in \mathcal{F}_\mathcal{P}, B \in \mathcal{F}_\mathcal{Q}\}, \mu \rangle \quad \text{where} \quad \mu(A) \triangleq \mu_1(\pi_S(A)) \cdot \mu_2(\pi_T(A))$$

Note that this definition is more similar to the initial formulation of PSL [Barthe et al. 2019] (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]. However, Li et al. [2024] recently showed that these two models are equivalent. 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:

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

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, which provides advantages over the conditioning modalities in Lilac and Bluebell, as we will discuss shortly. Finally, we remark that independent products distribute over direct sums (Lemma B.1):

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

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 4.

$$\varphi ::= \top \mid \bot \mid \varphi \land \psi \mid \varphi \lor \psi \mid \bigoplus_{X \sim d(E)} \varphi \mid \varphi * \psi \mid \lceil P \rceil$$

The semantics for probabilistic assertions is based on a context $\Gamma \colon \mathsf{LVar} \to \mathsf{Val}$, and a complete probability space $\mathcal{P} = \langle \mathsf{Mem}[S], \mathcal{F}_\mathcal{P}, \mu_\mathcal{P} \rangle$. The $\top$, $\bot$, conjunction and disjunction assertions have the usual semantics. Next, we have two connectives whose semantics are based on the probability space operations defined in Section 4.2.

The outcome conjunction $\bigoplus_{X \sim d(E)}$ allocates a new logical variable X, which is distributed according to $\mu = d(\llbracket E \rrbracket_{\mathsf{LExp}}(\Gamma))$, and can be referenced in φ. The probability space $\mathcal{P}$ must be larger than the direct sum of $(\mathcal{P}_v)_{v \in \mathsf{supp}(\mu)}$. For every v, we then also require that $\Gamma[X \coloneqq v], \mathsf{comp}(\mathcal{P}_v) \vDash \varphi$, so φ holds in the sub-probability space $\mathcal{P}_v$ with the value of X in Γ updated accordingly. The outcome conjunction is similar to the conditioning modalities of Lilac and Bluebell, which use the notation **C** rather than $\bigoplus$. The key difference is that Lilac and Bluebell do not use a direct sum, but instead

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

Fig. 4. The satisfaction relation, where $\Gamma\colon \mathsf{LVar} \to \mathsf{Var}$ 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.

require that each φ holds in the same σ-algebra for each $v \in \mathsf{supp}(\mu)$. The upside of that approach is that disjointness of the sample space does not need to be established, making some of the entailment rules simpler. The downside is that it makes case analysis over the outcomes of the conditioning operation more difficult. We will discuss this further in Section 5.2.

Next, we have the separating conjunction $\varphi * \psi$, which uses an independent product to ensure that the probability spaces satisfying φ and ψ are independent. The fact that $\mathcal{P}_1 \otimes \mathcal{P}_2 \preceq \mathcal{P}$ means that information can be lost so that the events described by φ and ψ are independent, even if some correlation exists in $\mathcal{P}$. The almost sure assertion $\lceil P \rceil$ states that the pure assertion P, as described in Section 4.1, occurs with probability 1. Finally, we define syntactic sugar below for a binary outcome conjunction $\varphi \oplus_p \psi$ and $e \sim d(E)$, meaning that e is distributed according to $d(E)$.

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

Precise Assertions and Entailment Laws. The notion of *precision* often arises 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]. An assertion is precise if it has a unique smallest model. For probabilistic assertions, the formal definition is below.

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 $\mathsf{precise}(\vec{\varphi})$ if the list of assertions $\vec{\varphi}$ are all precise.

We use precision to guarantee that the separating conjunction distributes over outcome conjunctions. 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.

Precision is important in pcOL, as it guarantees that nondeterminism introduced by the scheduler is not observable in the event space. This is crucial for the Par rule, since we want to be able to conclude that the outcomes of two threads are independent despite any correlation that could be introduced by the particular interleaving order of the threads. We will see more concrete examples on this in Section 5. 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 montonicity of the independent product

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

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

Fig. 5. Selected entailment laws for probabilistic assertions, where $\varphi[Y/X]$ denotes a syntactic substitution of Y for X in φ.

(Lemma B.2). 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. 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.

We conclude by giving some entailment laws about probabilistic assertions, shown in Figure 5. Many of these laws echo those of Bluebell, but there are some notable differences highlighting the ways in which our subtle changes to the model result in desirable properties. We first have two rules pertaining to almost sure assertions; weakening can be performed inside of an almost sure assertion[1], and separating conjunctions can be moved in and out of almost sure assertions.

Weakening can also 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 separating conjunction distributes over the outcome conjunction, so that assertions can be moved inside of an outcome conjunction. However, the backward direction of this rule—factoring assertions out of an outcome conjunction—is only supported in Bluebell for almost-sure assertions $\lceil P \rceil$. In our logic, this operation can be performed for any *precise* assertion, due to our semantics based on a direct sum.

5 Probabilistic Concurrent Outcome Logic

Probabilistic Concurrent Outcome Logic (pcOL) specifications are given as triples of the form $I \vDash \langle \varphi \rangle\ C\ \langle \psi \rangle$, where φ and ψ are probabilistic assertions (Section 4.3), $C \in \mathsf{Cmd}$ (Figure 1), and I is a basic assertion. Roughly speaking, the meaning of these triples is that starting in any distribution satisfying φ, any distribution resulting from an invariant sensitive execution of C will satisfy ψ.

Recall from Section 3.3 that invariant sensitive execution required the invariant states to be drawn from a finite set. For this reason, we require I to be a *finitary* basic assertion; formally, we write $\mathsf{finitary}(I)$ iff $(\!| I |\!)_\Gamma$ is a finite set for any context Γ. We will discuss the implications of this restriction further in Section 8. The formal validity definition of pcOL triples is below.

Definition 5.1 (pcOL Triples). The pcOL triple $I \vDash \langle \varphi \rangle\ C\ \langle \psi \rangle$ is valid iff for all $\Gamma \colon \mathsf{LVar} \to \mathsf{Val}$, $\mu \in \mathcal{D}(\mathsf{Mem}[S])$, and probability spaces $\mathcal{P}$ and $\mathcal{P}_F$ such that $\mathcal{P} \otimes \mathcal{P}_F \preceq \mu$ and $\Gamma, \mathcal{P} \vDash \varphi$, then:

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

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. Note that in the initial distribution μ, the variables in $\mathcal{P}$ and $\mathcal{P}_F$ may be correlated, but the fact that $\mathcal{P} \otimes \mathcal{P}_F \preceq \mu$ means that the correlation will not be measurable. In

[1]This rule relies on the fact that probability spaces are *complete*, so that if $A \in \mathcal{F}_\mathcal{P}$ and $\mu_\mathcal{P}(A) = 1$, then $B \in \mathcal{F}_\mathcal{P}$ for all $B \supseteq A$, allowing us to eschew the concepts of *almost-sure equality* [Li et al. 2023, Fig. 6] and *almost measurability* [Bao et al. 2024, Def. 4.7].

$$\frac{}{I \vdash \langle \varphi \rangle\ \mathtt{skip}\ \langle \varphi \rangle}\textsc{Skip} \qquad \frac{I \vdash \langle \varphi \rangle\ C_1\ \langle \vartheta \rangle \quad I \vdash \langle \vartheta \rangle\ C_2\ \langle \psi \rangle}{I \vdash \langle \varphi \rangle\ C_1 \fatsemi C_2\ \langle \psi \rangle}\textsc{Seq} \qquad \frac{\forall 0 \le k \le n-1.\quad I \vdash \langle \varphi_k \rangle\ C\ \langle \varphi_{k+1} \rangle}{I \vdash \langle \varphi_0 \rangle\ \mathtt{for}\ n\ \mathtt{do}\ C\ \langle \varphi_n \rangle}\textsc{For}$$

$$\frac{\varphi \Rightarrow \lceil b \mapsto \mathsf{true} \rceil \qquad I \vdash \langle \varphi \rangle\ C_1\ \langle \psi \rangle}{I \vdash \langle \varphi \rangle\ \mathtt{if}\ b\ \mathtt{then}\ C_1\ \mathtt{else}\ C_2\ \langle \psi \rangle}\textsc{If1} \qquad \frac{\varphi \Rightarrow \lceil b \mapsto \mathsf{false} \rceil \qquad I \vdash \langle \varphi \rangle\ C_2\ \langle \psi \rangle}{I \vdash \langle \varphi \rangle\ \mathtt{if}\ b\ \mathtt{then}\ C_1\ \mathtt{else}\ C_2\ \langle \psi \rangle}\textsc{If2}$$

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

$$\frac{(\varphi * \lceil x \mapsto E \rceil) \Rightarrow \lceil e \mapsto E' \rceil}{I \vdash \langle \varphi * \lceil x \mapsto E \rceil \rangle\ x \coloneqq e\ \langle \varphi * \lceil x \mapsto E' \rceil \rangle}\textsc{Assign} \qquad \frac{(\varphi * \lceil x \mapsto E \rceil) \Rightarrow \lceil e \mapsto E' \rceil}{I \vdash \langle \varphi * \lceil x \mapsto E \rceil \rangle\ x :\approx d(e)\ \langle \varphi * (x \sim d(E')) \rangle}\textsc{Samp}$$

Fig. 6. Rules for Commands

addition, since we defined probability spaces to operate over memories $\mathsf{Mem}[S]$, without $\bot$, this makes the definition of triples *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.

In the remainder of this section, we will present inference rules for deriving pcOL triples. We write $I \vdash \langle \varphi \rangle\ C\ \langle \psi \rangle$ 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 6 and 7, if* $I \vdash \langle \varphi \rangle\ C\ \langle \psi \rangle$ *then* $I \vDash \langle \varphi \rangle\ C\ \langle \psi \rangle$.

5.1 Rules for Commands

The rules for commands are given in Figure 6. Some of the rules appear standard, such as Skip for no-ops, and Seq for sequential composition, however they rely on the properties of linearization that we discussed in Section 3.3. 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 2024; Zilberstein et al. 2025]. These rules can be combined into a single rule for analyzing both branches using Cond1 or Cond2, which we will introduce in Section 5.2. The For rule use a family of assertions $\varphi_0, \ldots, \varphi_n$ to prove the behavior of n iterations of C.

The most involved rule is Par, for parallel composition. Although Par looks like the analogous rules from Concurrent Separation Logic [Brookes 2004; O'Hearn 2004; Vafeiadis 2011], 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, the scheduler could choose to schedule the $y \coloneqq 1$ action *after* the sampling operation is resolved, meaning that it could make $x = y$ with probability 1, a clear correlation.

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

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 for each thread to be precise, we know that the probability of each measurable event must be specified exactly, so that the nondeterminism injected by the scheduler will not be observable, which we prove in Lemmas A.8 and A.9. In the case of the aforementioned program, the only precise assertion about y that subsumes all outcome is $\lceil y \in \{0, 1\} \rceil$, that y is always either 0 or 1. Fortunately, $\left(x \sim \mathbf{Ber}\left(\tfrac{1}{2}\right)\right) * \lceil y \in \{0, 1\} \rceil$ is a valid postcondition for the program, since almost sure assertions $\lceil P \rceil$ are trivially independent from all other assertions.

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 being assigned x is owned by the current thread. Samp has a similar requirement, but ultimately concludes that x is distributed according to $d(E)$ rather than having a deterministic value.

5.2 Structural Rules

We give additional structural rules in Figure 7, which do not depend on the program command. The first two rules interact with invariants and are inspired by Concurrent Separation Logic [O'Hearn 2004]. 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 ensures that the invariant remains true at all times. 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, and the fault avoiding nature of triples.

The Frame rule allows a local specification to be lifted into a larger memory footprint [Yang and O'Hearn 2002]. 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.

We next have two conditioning rules, which allow us to deconstruct an outcome conjunction in the precondition to reason about each event individually. Both rules require that the logical variable X, that is 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 5). The Cond1 rule requires that ψ dictate the partition of the probability spaces in order to construct a direct sum. This is done in a similar fashion to rules for establishing precision that we saw in Section 4.3. If ψ does not dictate a partition, then the Cond2 rule can instead be used, which requires ψ to be precise and not dependent on X.

Although the direct sum semantics of the outcome conjunction imposes some limitations on the conditioning rules, the benefit is that they are fully compositional; the use of conditioning 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. 2024, §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. [2024] show fruitful uses of c-wp-swap, the restriction is not acceptable in the present concurrent setting, since it would preclude use of the Par rule. Because of this, we prefer the direct sum semantics for pcOL.

The Exists rule allows us to reason underneath existential quantifiers in almost sure assertions. This rule resembles a conditioning rule, except that the probabilities with which X takes on each value are unknown. Exists is very important for reasoning about the result of nondeterministic aspects of the program evaluation, as we will see in Section 6.1. 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.

6 Examples

In this section, we present three examples to demonstrate how the proof rules of pcOL come together into more complex derivations.

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. One such example is the following program, where x_2 is a reliable source of entropy, but x_1 is unreliable. Despite that, z, which is derived from

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

$$\frac{I \vdash \langle \varphi \rangle\ C\ \langle \psi \rangle \quad \psi \Rightarrow \lceil e \mapsto X \rceil \quad X \notin \mathsf{fv}(I)}{I \vdash \langle \bigoplus_{X \sim d(E)} \varphi \rangle\ C\ \langle \bigoplus_{X \sim d(E)} \psi \rangle}\ \textsc{Cond1} \qquad \frac{I \vdash \langle \varphi \rangle\ C\ \langle \psi \rangle \quad \mathsf{precise}(\psi) \quad X \notin \mathsf{fv}(I, \psi)}{I \vdash \langle \bigoplus_{X \sim d(E)} \varphi \rangle\ C\ \langle \psi \rangle}\ \textsc{Cond2}$$

$$\frac{I \vdash \langle \lceil P \rceil \rangle\ C\ \langle \psi \rangle \quad \mathsf{precise}(\psi) \quad X \notin \mathsf{fv}(\psi, I)}{I \vdash \langle \lceil \exists X.\, P \rceil \rangle\ C\ \langle \psi \rangle}\ \textsc{Exists} \qquad \frac{\varphi' \Rightarrow \varphi \quad I \vdash \langle \varphi \rangle\ C\ \langle \psi \rangle \quad \psi \Rightarrow \psi'}{I \vdash \langle \varphi' \rangle\ C\ \langle \psi' \rangle}\ \textsc{Consequence}$$

Fig. 7. Structural Rules

x_1 and x_2 is a high quality source of randomness.

$$y \coloneqq 0 \,⨾\, \left(\ x_1 \coloneqq y \,⨾\, x_2 :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right) ⨾\, z \coloneqq \mathsf{xor}(x_1, x_2)\ \middle\|\ y \coloneqq 1\ \right)$$

We will analyze this program using the invariant $y \in \{0,1\}$, and conclude in the end that $z \sim \mathbf{Ber}\left(\frac{1}{2}\right)$. 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.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\langle \lceil y \mapsto Y * (Y = 0 \lor Y = 1) * x_1 \mapsto X \rceil \rangle\ x_1 \coloneqq y\ \langle \lceil x_1 \mapsto Y * y \mapsto Y * (Y = 0 \lor Y = 1) \rceil \rangle}\ \textsc{Assign}}{\langle \lceil y \mapsto Y * (Y = 0 \lor Y = 1) * x_1 \mapsto X \rceil \rangle\ x_1 \coloneqq y\ \langle \lceil x_1 \in \{0,1\} \rceil * \lceil y \in \{0,1\} \rceil \rangle}\ \textsc{Consequence}}{\langle \lceil \mathsf{own}(x_1) \rceil * \lceil y \in \{0,1\} \rceil \rangle\ x_1 \coloneqq y\ \langle \lceil x_1 \in \{0,1\} \rceil * \lceil y \in \{0,1\} \rceil \rangle}\ \textsc{Exists} \times 2}{y \in \{0,1\} \vdash \langle \lceil \mathsf{own}(x_1) \rceil \rangle\ x_1 \coloneqq y\ \langle \lceil x_1 \in \{0,1\} \rceil \rangle}\ \textsc{Atom}$$

First, Atom is applied to open the invariant. Next, we use Exists to gain access to the value of y, so that we can apply the Assign rule. Next, we apply the rule of Consequence in order to reapply the existential quantifiers, so that we can remove the dependence on the logical variable Y, and separate the information about x_1 and y, so that the invariant can be closed. After closing the invariant, we can move on to derive a spec for the remainder of the thread, which we show below (the invariant is not shown for brevity, since it is no longer used).

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\langle \lceil \mathsf{own}(x_2) \rceil \rangle\ x_2 :\approx \mathbf{Ber}\,(1/2)\ \langle x_2 \sim \mathbf{Ber}\,(1/2) \rangle}\ \textsc{Samp}}{\langle \lceil x_1 \mapsto X \rceil * \lceil \mathsf{own}(x_2, z) \rceil \rangle\ x_2 :\approx \mathbf{Ber}\,(1/2)\ \langle \lceil x_1 \mapsto X \rceil * (x_2 \sim \mathbf{Ber}\,(1/2)) * \lceil \mathsf{own}(z) \rceil \rangle}\ \textsc{Frame} \quad (\bigstar)}{\langle \lceil x_1 \mapsto X \rceil * \lceil \mathsf{own}(x_2, z) \rceil \rangle\ x_2 :\approx \mathbf{Ber}\,(1/2) ⨾ z \coloneqq \mathsf{xor}(x_1, x_2)\ \langle z \sim \mathbf{Ber}\,(1/2) \rangle}\ \textsc{Seq}}{\langle \lceil x_1 \in \{0,1\} \rceil * \lceil \mathsf{own}(x_2, z) \rceil \rangle\ x_2 :\approx \mathbf{Ber}\,(1/2) ⨾ z \coloneqq \mathsf{xor}(x_1, x_2)\ \langle z \sim \mathbf{Ber}\,(1/2) \rangle}\ \textsc{Exists}$$

We start by applying Exists again in order to gain access to the 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. The derivation ($\bigstar$) for the second command is shown below.

$$\dfrac{\dfrac{\dfrac{}{\langle \lceil x_1 \mapsto X \rceil * \lceil x_2 \mapsto X' \rceil * \lceil \mathsf{own}(z) \rceil \rangle\ z \coloneqq \mathsf{xor}(x_1, x_2)\ \langle \lceil z \mapsto \mathsf{xor}(X, X') \rceil \rangle}\ \textsc{Assign}}{\langle \bigoplus_{X' \sim \mathbf{Ber}(1/2)} \lceil x_1 \mapsto X \rceil * \lceil x_2 \mapsto X' \rceil * \lceil \mathsf{own}(z) \rceil \rangle\ z \coloneqq \mathsf{xor}(x_1, x_2)\ \langle \bigoplus_{X' \sim \mathbf{Ber}(1/2)} \lceil z \mapsto \mathsf{xor}(X, X') \rceil \rangle}\ \textsc{Cond1}}{\langle \lceil x_1 \mapsto X \rceil * (x_2 \sim \mathbf{Ber}\,(1/2)) * \lceil \mathsf{own}(z) \rceil \rangle\ z \coloneqq \mathsf{xor}(x_1, x_2)\ \langle z \sim \mathbf{Ber}\,(1/2) \rangle}\ \textsc{Consequence}$$

We use Cond1 to condition on the result of the sampling operation, at which point we use Assign to conclude that $z \mapsto \mathsf{xor}(X, X')$. Since X is constant, then $\mathsf{xor}(X, X')$ 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.

6.2 Batch Sampling

Increasing reliance on parallel computing in high-performance applications has exposed random number generation as a bottleneck [Salmon et al. 2011]. Random number generation can be parallelized, but when doing so, care must be taken to ensure that the aggregated result has the right statistical properties. In this example, we use PCOL to prove that a derived form of randomness is uniformly distributed, which relies on the fact that the outcomes of two threads are probabilistically independent. The program shown below samples an n-bit and an m-bit number in parallel, and then combines them to get a uniformly distributed $n + m$-bit number, where $\mathbf{int}_k \triangleq \mathbf{unif}\left([0, \ldots, 2^k - 1]\right)$ is a uniform distribution over the k-bit nonnegative integers.

$$\left(\ x :\approx \mathbf{int}_n \ \middle\|\ y :\approx \mathbf{int}_m \ \right) \fatsemi z \coloneqq 2^m \cdot x + y$$

We can use the PAR and SAMP rules to conclude that $x \sim \mathbf{int}_n * y \sim \mathbf{int}_m$ after the parallel composition. We sketch the remainder of the derivation as follows.

$$\dfrac{\dfrac{\dfrac{}{\langle\lceil x \mapsto X * y \mapsto Y \rceil\rangle\ z \coloneqq 2^m \cdot x + y\ \langle\lceil z \mapsto 2^m \cdot X + Y \rceil\rangle}\ \textsc{Assign}}{\langle \bigoplus_{X\sim\mathbf{int}_n} \bigoplus_{Y\sim\mathbf{int}_m} \lceil x \mapsto X * y \mapsto Y \rceil\rangle\ z \coloneqq 2^m \cdot x + y\ \langle \bigoplus_{X\sim\mathbf{int}_n} \bigoplus_{Y\sim\mathbf{int}_m} \lceil z \mapsto 2^m \cdot X + Y \rceil\rangle}\ \textsc{Cond1} \times 2}{\langle x \sim \mathbf{int}_n * y \sim \mathbf{int}_m \rangle\ z \coloneqq 2^m \cdot x + y\ \langle z \sim \mathbf{int}_{n+m} \rangle}\ \textsc{Consequence}$$

First, we use the rule of CONSEQUENCE with the distributive law from Figure 5 to move the information about y inside of the outcome conjunction over X. Next, we apply COND1 twice in order to remove the two outcome conjunctions and apply the ASSIGN. This gives us the postcondition $\bigoplus_{X\sim\mathbf{int}_n} \bigoplus_{Y\sim\mathbf{int}_m} \lceil z \mapsto 2^m \cdot X + Y \rceil$ after the application of COND1. Now, because $2^m \cdot X + Y$ is a bijection from X and Y to the interval $[0, 2^{n+m} - 1]$, we can fold the two outcome conjunction with the rule of CONSEQUENCE to conclude that z is uniformly sampled from that interval.

We just showed how to use PCOL to prove correctness properties about combined sources of randomness. This paradigm shows us in more examples too. For example, 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—similar to the classic mergesort—in which sub-arrays are shuffled concurrently and then merged using another randomized procedure. A simplified version of their algorithm is shown below.

$$\left(\ \mathtt{shuffle}(x[0, n/2 - 1]) \ \middle\|\ \mathtt{shuffle}(x[n/2, n - 1]) \ \right) \fatsemi \mathtt{merge}(x)$$

Notice that the structure of this program is nearly identical to the batch sampling program above, Indeed, correctness of the concurrent shuffle also relies on the probabilistic independence of the two shuffled sub-arrays, although the proof is out of scope for this paper.

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]. We achieve this using the program shown in Figure 8. The **`fetch`** procedure takes a bit string query q, which indicates which entries of the database d to return. Those entries are then xor'ed together. Private retrieval is implemented in the **`privFetch`** procedure. The input x is a *one-hot* bit string, with a 1 in the position of the data being requested 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, but with one bit flipped, corresponding to the entry that the user wants. The final data is then an xor of the two responses.

Barthe et al. [2019] proved a similar example in PSL, but their version was sequential; both fetches happened within a single for loop. Our version is slightly more realistic, as the two fetches occur concurrently. We first present a specification for the **`fetch`** procedure, which states that r is

```
fetch(q, i, r, d) :                         privFetch(x) :
  i := 0 ⨾ r := 0 ⨾                           q1 :≈ unif ({0, 1}^n) ⨾
  for n do                                    q2 := xor(q1, x) ⨾
    if q[i] = 1 then                          (fetch(q1, i, r1, d) ∥ fetch(q2, j, r2, d)) ⨾
      r := xor(r, d[i]) ⨾                     r := xor(r1, r2)
    i := i + 1
```

Fig. 8. A concurrent private information retrieval protocol.

an xor of data entries k, such that $q[k] = 1$, subject to the invariant that $d \mapsto D$.

$$d \mapsto D \vdash \langle \lceil q \mapsto Q \rceil * \lceil \mathsf{own}(i, r) \rceil \rangle\ \mathtt{fetch}(q, i, r, d)\ \langle \lceil r \mapsto \operatorname*{xor}_{k=0}^{n-1}(D[k] \land Q[k]) \rceil \rangle$$

We now sketch the derivation of the main procedure. After sampling into q_1, we use Cond2 to make the outcome of that query deterministic. After assigning q_2, we get $\lceil q_1 \mapsto Q \rceil * \lceil q_2 \mapsto \mathsf{xor}(Q, X) \rceil$. We can then apply the Par rule to analyze the concurrent fetches to get the following postcondition:

$$\lceil r_1 \mapsto \operatorname*{xor}_{k=0}^{n-1}(D[k] \land Q[k]) \rceil * \lceil r_2 \mapsto \operatorname*{xor}_{k=0}^{n-1}(D[k] \land \mathsf{xor}(Q, X)[k]) \rceil$$

The final assignment to r is obtained by xor'ing r_1 and r_2, which differ only at the index of the query that we requested. Therefore, we can use the rule of Consequence to conclude that $\lceil r \mapsto D[k] \rceil$, where $X[k] = 1$. Since this assertion is precise and does not depend on Q, we meet the side conditions of the Cond2 rule, and therefore the final postcondition is $\lceil r \mapsto D[k] \rceil$.

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]. Specifications take the form of *refinements*—complex programs are shown to behave equivalently to idealized ones. Probabilistic analysis is then used to prove the expected behavior of the idealized program, but this analysis is external to the program logic. Compared to Polaris, pcOL has two advantages: probabilistic reasoning is done directly in the logic, and specifications give the full *distribution* over the program's outcomes. However, unlike Polaris, pcOL does not support advanced forms of separation and ghost state.

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]. This logic can be used to lower bound the probability of a single outcome, whereas pcOL is more expressive in its ability to give precise probabilities for many outcomes.

Probabilistic Separation Logic. Capturing probabilistic independence in separation logic was first explored by Barthe et al. [2019], 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, but did not include a full program logic [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 reformulated the notion of separation using probability spaces, making it more expressive. Bluebell added the ability to reason about mutable state, and is also relational so as to capture the joint behavior of multiple programs [Bao et al. 2024]. We build on these logics, bringing independence and conditioning into a concurrent setting for the first time, with a modified measure theoretic semantics to support compositional conditioning rules.

A second category of probabilistic program logics is built on top of Iris [Jung et al. 2018], from which they inherit expressive features, including rich support for ghost state, impredicativity, and invariant reasoning. 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 proving an upper bound on the probability that a postcondition will fail to hold. Additional logics have also been developed for relational reasoning and refinement [Gregersen et al. 2024a,b; Haselwarter et al. 2024a]. The tradeoff in these logics 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 (similar to pcOL), but is backed by a denotational model that supports specifications about the distribution of outcomes [Zilberstein et al. 2024].

Semantics for Probability, Nondeterminism, and Concurrency. There is a rich body of research on semantics that combine probabilistic and nondeterministic computation. In this paper, we build on the convex powerset approach [Morgan et al. 1996b], which has been used to define denotational semantics [He et al. 1997] and has well understood equational [Mislove 2000; Mislove et al. 2004] and domain theoretic properties [Keimel and Plotkin 2017; Tix 1999, 2000; Tix et al. 2009]. We refer to Zilberstein et al. [2025, §7] for an overview of alternative other approaches.

The convex powerset was used as a semantic basis for expectation calculi, which bound the expected values of random variables in probabilistic and nondeterministic programs [Kaminski 2019; McIver and Morgan 2005; Morgan et al. 1996a]. More recently, Demonic Outcome Logic (dOL) used the convex powerset to develop a logic for reasoning about these programs in terms of their distributions of outcomes, adding more expressive power [Zilberstein et al. 2025]. However, the convex powerset had not previously intersected with semantics of *concurrent* programs, which uses partially ordered multisets (pomsets) to record the causality between actions [Gischer 1988; Pratt 1986]. As discussed in Section 3, no prior pomset semantics is suitable for use with the convex powerset, as they implicitly reorder probabilistic and nondeterministic choices.

8 Conclusion, Limitations, and Future Work

In this paper, we presented Probabilistic Concurrent Outcome Logic (pcOL), which brings together ideas from concurrent and probabilistic separation logics for reasoning about parallel programs in terms of independence and conditioning. This is a significant step in bringing expressive techniques to analysis of probabilistic concurrent programs and required a new semantics to establish soundness. The result is a proof system with familiar inference rules, however the complexity of the underlying metatheory required us to impose several restrictions, limiting the capabilities of pcOL. We now discuss some of those limitations, and how we plan to address them in future work.

Unbounded Looping and Almost Sure Termination. The programming language in Section 3 only includes bounded looping. We imposed this restriction for two reasons. First, there is no known infinitary structure for pomsets with tests, which were needed to capture the semantics of scheduling interleaved with probabilistic sampling. Second, the soundness of Par does not extend to infinite computation in a straightforward way. We found this limitation acceptable, as no other PSL currently support unbounded iteration [Bao et al. 2024; Barthe et al. 2019; Li et al. 2023]. However, we plan to augment pcOL with the ability to handle unbounded looping by enriching the pomsets from Section 3.1 with a dcpo structure, so that infinitary computations can be represented as the limits of their finite approximations [de Bakker and Warmerdam 1990; Meyer and de Vink 1989].

This will allow us to prove that concurrent programs almost surely terminate—they terminate with probability 1. Almost sure termination has been a focus of sequential probabilistic program analysis [Kaminski 2019; McIver and Morgan 2005; McIver et al. 2018]. We plan to introduce the new concept of *mutual almost sure termination*, in which concurrent loops interacting with shared resources all terminate with probability 1. This will allow us to verify synchronization protocols [Hart et al. 1983] such as the Dining Philosophers problem [Lehmann and Rabin 1981].

Probabilistic Invariants. In this paper, we allow invariants that are *almost sure* statements about shared variables. This helps in establishing soundness of the Par rule, since almost sure assertions are always probabilistically independent from any other assertions. However, going forward we would like to allow invariants to specify probabilistic properties. This would allow us to verify probabilistic approximate counters [Flajolet 1985; Morris 1978], where the invariant must specify that the *expected value* of the approximate count is the true count.

Dynamic Allocation and Resource Algebras. Building on PSL, our resource model uses variables rather than addressed pointers. This simplifies our formulation, as we know that each distribution contains memories over a homogenous set of variables, and therefore we can simply project out separate components of distributions. However, most concurrent programs use pointers, which can be dynamically allocated and alias each other, adding complexity. Further, 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. In particular, *ghost state* governs the ways in which concurrent threads can modify shared resources. Bluebell already included permissions, which can help to duplicate knowledge about read-only variables [Bao et al. 2024], however many other resources are used in practice.

Mechanization. As we saw in Section 6 and Appendix D, pcOL derivations are quite involved—even for small programs—due to the handling of invariants, conditioning, and existential quantification. Verification of larger programs would be infeasible with pen-and-paper proofs, therefore we would like to integrate pcOL into Iris [Jung et al. 2018, 2015] to mechanize the logic and automate the more tedious details of proofs. However, our denotational semantics is quite different from Iris's step-indexed model, so significant work would need to be done in order to fit these two logics together.

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. 2024. Bluebell: An Alliance of Relational Lifting and Independence For Probabilistic Reasoning. arXiv:2402.18708 [cs.LO] https://arxiv.org/abs/2402.18708

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

Kevin Batz, Benjamin Lucien Kaminski, Joost-Pieter Katoen, Christoph Matheja, and Thomas Noll. 2019. Quantitative Separation Logic: A Logic for Reasoning about Probabilistic Pointer Programs. *Proc. ACM Program. Lang.* 3, POPL, Article 34 (Jan 2019), 29 pages. https://doi.org/10.1145/3290347

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

J. W. de Bakker and J. H. A. Warmerdam. 1990. Metric pomset semantics for a concurrent language with recursion. In *Semantics of Systems of Concurrent Processes*, Irène Guessarian (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 21–49. https://doi.org/10.1007/3-540-53479-2_2

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

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

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.

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

Benjamin Lucien Kaminski. 2019. *Advanced weakest precondition calculi for probabilistic programs.* Dissertation. RWTH Aachen University, Aachen. https://doi.org/10.18154/RWTH-2019-01829

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

John M. Li, Jon Aytac, Philip Johnson-Freyd, Amal Ahmed, and Steven Holtzen. 2024. A Nominal Approach to Probabilistic Separation Logic. In *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science* (Tallinn, Estonia) *(LICS '24)*. Association for Computing Machinery, New York, NY, USA, Article 55, 14 pages. https://doi.org/10.1145/3661814.3662135

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

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

Annabelle McIver, Carroll Morgan, Benjamin Lucien Kaminski, and Joost-Pieter Katoen. 2018. A New Proof Rule for Almost-Sure Termination. *Proc. ACM Program. Lang.* 2, POPL, Article 33 (Jan 2018), 28 pages. https://doi.org/10.1145/3158121

J. J. Ch. Meyer and E. P. de Vink. 1989. Pomset semantics for true concurrency with synchronization and recursion. In *Mathematical Foundations of Computer Science 1989*, Antoni Kreczmar and Grazyna Mirkowska (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 360–369.

Michael Mislove. 2000. Nondeterminism and Probabilistic Choice: Obeying the Laws. In *CONCUR 2000 — Concurrency Theory*, Catuscia Palamidessi (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 350–365. https://doi.org/10.1007/3-540-44618-4_26

Michael Mislove, Joël Ouaknine, and James Worrell. 2004. Axioms for Probability and Nondeterminism. *Electronic Notes in Theoretical Computer Science* 96 (2004), 7–28. https://doi.org/10.1016/j.entcs.2004.04.019 Proceedings of the 10th International Workshop on Expressiveness in Concurrency.

Carroll Morgan, Annabelle McIver, and Karen Seidel. 1996a. Probabilistic Predicate Transformers. *ACM Trans. Program. Lang. Syst.* 18, 3 (may 1996), 325–353. https://doi.org/10.1145/229542.229547

Carroll Morgan, Annabelle McIver, Karen Seidel, and J. W. Sanders. 1996b. Refinement-oriented probability for CSP. *Form. Asp. Comput.* 8, 6 (nov 1996), 617–647. https://doi.org/10.1007/bf01213492

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.

John K. Salmon, Mark A. Moraes, Ron O. Dror, and David E. Shaw. 2011. Parallel random numbers: as easy as 1, 2, 3. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis* (Seattle, Washington) *(SC '11)*. Association for Computing Machinery, New York, NY, USA, Article 16, 12 pages. https://doi.org/10.1145/2063384.2063405

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. 1999. *Continuous D-cones: convexity and powerdomain constructions*. Ph.D. Dissertation. Darmstadt University of Technology, Germany. https://d-nb.info/957239157

Regina Tix. 2000. Convex Power Constructions for Continuous D-Cones. *Electronic Notes in Theoretical Computer Science* 35 (2000), 206–229. https://doi.org/10.1016/S1571-0661(05)80746-7 Workshop on Domains IV.

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

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

Noam Zilberstein. 2024. A Relatively Complete Program Logic for Effectful Branching. arXiv:2401.04594 [cs.LO]

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, Dexter Kozen, Alexandra Silva, and Joseph Tassarotti. 2025. 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

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 Semantic Properties

LEMMA A.1. *For any finite index set I, $f\colon X \to \mathcal{C}(Y)$, and $(S_i)_{i\in I} \in \mathcal{C}(X)^I$:*

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

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

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

So, we get:

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

□

A.1 Properties of Linearization

LEMMA A.2. *For any $\alpha, \beta \in \mathcal{P}\mathsf{om}$:*

$$\mathcal{L}^{\mathcal{I}}(\alpha \fatsemi \beta) = \mathcal{L}^{\mathcal{I}}(\beta)^\dagger \circ \mathcal{L}^{\mathcal{I}}(\alpha)$$

PROOF. The proof is by induction on the size of α. If α is empty, then the proof is trivial since $\alpha \fatsemi \beta = \beta$. Now, suppose the claim holds for pomsets smaller than α, we have:

$$\begin{aligned}
&\mathcal{L}^{\mathcal{I}}(\alpha \fatsemi \beta)(\sigma) \\
&= \mathop{\&}_{\ell\in\min(\alpha)} \begin{cases} \mathcal{L}^{\mathcal{I}}(\alpha \setminus \{\ell\} \fatsemi \beta)^\dagger(\llbracket a \rrbracket^{\mathcal{I}}_{\mathsf{Act}}(\sigma)) & \text{if } \lambda_\alpha(\ell) = \mathbb{i}_{\mathsf{Act}}(a) \\ \mathcal{L}^{\mathcal{I}}(\mathsf{filter}(\alpha, \ell, b, \sigma) \fatsemi \beta)(\sigma) & \text{if } \lambda_\alpha(\ell) = \mathbb{i}_{\mathsf{Test}}(b) \end{cases}
\end{aligned}$$

Note above that the filter operation does not affect β, since ℓ was not in β and therefore we know that ℓ does not show up in the test relation of β. We can now apply the induction hypothesis and the monad laws to get:

$$= \mathop{\&}_{\ell\in\min(\alpha)} \begin{cases} \mathcal{L}^{\mathcal{I}}(\beta)^\dagger(\mathcal{L}^{\mathcal{I}}(\alpha \setminus \{\ell\})^\dagger(\llbracket a \rrbracket^{\mathcal{I}}_{\mathsf{Act}}(\sigma))) & \text{if } \lambda_\alpha(\ell) = \mathbb{i}_{\mathsf{Act}}(a) \\ \mathcal{L}^{\mathcal{I}}(\beta)^\dagger(\mathcal{L}^{\mathcal{I}}(\mathsf{filter}(\alpha, \ell, b, \sigma))(\sigma)) & \text{if } \lambda_\alpha(\ell) = \mathbb{i}_{\mathsf{Test}}(b) \end{cases}$$

By Lemma A.1.

$$\begin{aligned}
&= \mathcal{L}^{\mathcal{I}}(\beta)^\dagger \left(\mathop{\&}_{\ell\in\min(\alpha)} \begin{cases} \mathcal{L}^{\mathcal{I}}(\alpha \setminus \{\ell\})^\dagger(\llbracket a \rrbracket^{\mathcal{I}}_{\mathsf{Act}}(\sigma)) & \text{if } \lambda_\alpha(\ell) = \mathbb{i}_{\mathsf{Act}}(a) \\ \mathcal{L}^{\mathcal{I}}(\mathsf{filter}(\alpha, \ell, b, \sigma))(\sigma) & \text{if } \lambda_\alpha(\ell) = \mathbb{i}_{\mathsf{Test}}(b) \end{cases} \right) \\
&= \mathcal{L}^{\mathcal{I}}(\beta)^\dagger(\mathcal{L}^{\mathcal{I}}(\alpha)(\sigma))
\end{aligned}$$

□

LEMMA A.3. *For any* $b \in \mathsf{Test}$ *and* $\alpha, \beta \in \mathcal{P}\mathrm{om}$*:*

$$\mathcal{L}^{\mathcal{I}}(\mathsf{guard}(b,\alpha,\beta))(\sigma) = \begin{cases} \mathcal{L}^{\mathcal{I}}(\alpha)(\sigma) & \text{if } [\![b]\!]_{\mathsf{Test}}(\sigma) = \mathsf{true} \\ \mathcal{L}^{\mathcal{I}}(\beta)(\sigma) & \text{if } [\![b]\!]_{\mathsf{Test}}(\sigma) = \mathsf{false} \end{cases}$$

PROOF. Note that $\min(\mathsf{guard}(b,\alpha,\beta))$ must be a singleton set containing only the label ℓ corresponding to the node of the test b. Therefore, we have:

$$\mathcal{L}^{\mathcal{I}}(\mathsf{guard}(b,\alpha,\beta))(\sigma) = \mathcal{L}^{\mathcal{I}}(\mathsf{filter}(\mathsf{guard}(b,\alpha,\beta),\ell,b,\sigma))(\sigma)$$

If $[\![b]\!]_{\mathsf{Test}}(\sigma) = \mathsf{true}$, then clearly the filter operation will remove β. Similarly, if it is false, then α will be removed.

$$= \begin{cases} \mathcal{L}^{\mathcal{I}}(\alpha)(\sigma) & \text{if } [\![b]\!]_{\mathsf{Test}}(\sigma) = \mathsf{true} \\ \mathcal{L}^{\mathcal{I}}(\beta)(\sigma) & \text{if } [\![b]\!]_{\mathsf{Test}}(\sigma) = \mathsf{false} \end{cases}$$

□

LEMMA A.4. *For any* $\alpha \in \mathcal{P}\mathrm{om}$ *and* $n \in \mathbb{N}$*:*

$$\mathcal{L}^{\mathcal{I}}(\alpha^n) = (\mathcal{L}^{\mathcal{I}}(\alpha))^n$$

PROOF. Recall that $\alpha^0 = \emptyset_{\mathcal{P}\mathrm{om}}$, $\alpha^{n+1} = \alpha \fatsemi \alpha^n$, $f^0 = \eta$, and $f^{n+1} = f^\dagger \circ f^n$. The proof is by induction on n. If $n = 0$, then we have:

$$\mathcal{L}^{\mathcal{I}}(\alpha^0) = \mathcal{L}^{\mathcal{I}}(\emptyset_{\mathcal{P}\mathrm{om}}) = \eta = (\mathcal{L}^{\mathcal{I}}(\alpha))^0$$

So, the claim holds. Now we show the inductive case:

$$\mathcal{L}^{\mathcal{I}}(\alpha^{n+1}) = \mathcal{L}^{\mathcal{I}}(\alpha \fatsemi \alpha^n)$$

By Lemma A.2.

$$= \mathcal{L}^{\mathcal{I}}(\alpha)^\dagger \circ \mathcal{L}^{\mathcal{I}}(\alpha^n)$$

By the induction hypothesis.

$$= \mathcal{L}^{\mathcal{I}}(\alpha)^\dagger \circ (\mathcal{L}^{\mathcal{I}}(\alpha))^n = (\mathcal{L}^{\mathcal{I}}(\alpha))^{n+1}$$

□

A.2 Invariant Monotonicity

LEMMA A.5. *For any* $\sigma \in \mathsf{Mem}[S]$, $\mathcal{I} \subseteq_{\mathsf{fin}} \mathsf{Mem}[T]$, *and* $\mathcal{J} \subseteq_{\mathsf{fin}} \mathsf{Mem}[U]$ *such that* S, T, *and* U *are pairwise disjoint, if* $\bot \notin \mathsf{supp}(\nu)$ *for all* $\nu \in [\![a]\!]^{\mathcal{I}*\mathcal{J}}_{\mathsf{Act}}(\sigma)$*, then:*

$$\forall \tau \in \mathcal{I}.\ \pi_S([\![a]\!]^{\mathcal{J}}_{\mathsf{Act}}(\sigma \uplus \tau)) \subseteq [\![a]\!]^{\mathcal{I}*\mathcal{J}}_{\mathsf{Act}}(\sigma) \quad \textit{and} \quad \bigcup \left\{ \mathsf{supp}(\pi_T(\nu)) \mid \nu \in [\![a]\!]^{\mathcal{J}}_{\mathsf{Act}}(\sigma \uplus \tau) \right\} \subseteq \mathcal{I}$$

PROOF. Take any $\tau \in \mathcal{J}$. We first prove the first part of the claim:

$$\pi_S\left([\![a]\!]^{\mathcal{J}}_{\mathsf{Act}}(\sigma \uplus \tau)\right)$$

$$= \pi_S\left(\mathop{\&}_{\tau' \in \mathcal{J}} \begin{cases} \pi_{S\cup T}([\![a]\!]_{\mathsf{Act}}(\sigma \uplus \tau \uplus \tau')) & \text{if } \bigcup\{\mathsf{supp}(\pi_U(\nu)) \mid \nu \in [\![a]\!]_{\mathsf{Act}}(\sigma \uplus \tau \uplus \tau')\} \subseteq \mathcal{I} \\ \eta(\bot) & \text{otherwise} \end{cases} \right)$$

We can move the π_S inside of the $\&$, and remove the projection over T. By ranging τ' over $\mathcal{I} * \mathcal{J}$, we make the set larger, since $\tau \in \mathcal{I}$. We assumed that a has no undefined behavior starting in σ with invariant $\mathcal{I} * \mathcal{J}$, so we can also expand the invariant check to include $\mathcal{I}$.

$$\subseteq \mathop{\&}_{\tau' \in \mathcal{I}*\mathcal{J}} \begin{cases} \pi_S([\![a]\!]_{\mathsf{Act}}(\sigma \uplus \tau')) & \text{if } \bigcup\{\mathsf{supp}(\pi_{T\cup U}(\nu)) \mid \nu \in [\![a]\!]_{\mathsf{Act}}(\sigma \uplus \tau')\} \subseteq \mathcal{I} * \mathcal{J} \\ \eta(\bot) & \text{otherwise} \end{cases}$$

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

The second part of the claim follows directly from the fact that $\bot \notin \mathsf{supp}(\nu)$ for all $\nu \in [\![a]\!]^{\mathcal{I}*\mathcal{J}}_{\mathsf{Act}}(\sigma)$. □

Lemma A.6. *For any* $f\colon \mathsf{Mem}[S] \to \mathcal{C}(\mathsf{Mem}[S])$, $g\colon \mathsf{Mem}[T] \to \mathcal{C}(\mathsf{Mem}[T])$, *and* $U \in \mathcal{C}(\mathsf{Mem}[T])$, *if for all* $\sigma \in \mathsf{Mem}[S]$ *and* $\tau \in \mathsf{Mem}[T \setminus S]$ $\pi_S(g(\sigma \uplus \tau)) \subseteq f(\sigma)$, *then:*

$$\pi_S(g^\dagger(U)) \subseteq f^\dagger(\pi_S(U))$$

Proof.

$$\pi_S(g^\dagger(U)) = \pi_S\left(\left\{ \sum_{\sigma' \in \mathsf{supp}(\mu)} \mu(\sigma') \cdot \nu_{\sigma'} \;\middle|\; \mu \in U, \forall \sigma'.\ \nu_{\sigma'} \in g_\bot(\sigma') \right\}\right)$$

$$= \left\{ \sum_{\sigma' \in \mathsf{supp}(\mu)} \mu(\sigma') \cdot \nu_{\sigma'} \;\middle|\; \mu \in U, \forall \sigma'.\ \nu_{\sigma'} \in \pi_S(g_\bot(\sigma')) \right\}$$

Using the premise, we get:

$$\subseteq \left\{ \sum_{\sigma' \in \mathsf{supp}(\mu)} \mu(\sigma') \cdot \nu_{\sigma'} \;\middle|\; \mu \in U, \forall \sigma'.\ \nu_{\sigma'} \in f_\bot(\pi_S(\sigma')) \right\}$$

We can equivalently choose a $\nu_{\sigma'}$ for each σ' instead of each $\pi_S(\sigma')$, since $f_\bot(\pi_S(\sigma'))$ is convex and the $\nu_{\pi_S(\sigma')}$ below is a convex combination of ν'_σ terms above.

$$= \left\{ \sum_{\sigma' \in \mathsf{supp}(\mu)} \mu(\sigma') \cdot \nu_{\pi_S(\sigma')} \;\middle|\; \mu \in U, \forall \sigma'.\ \nu_{\sigma'} \in f_\bot(\sigma') \right\}$$

$$= \left\{ \sum_{\sigma' \in \mathsf{supp}(\mu)} \mu(\sigma') \cdot \nu_{\sigma'} \;\middle|\; \mu \in \pi_S(U), \forall \sigma'.\ \nu_{\sigma'} \in f_\bot(\sigma') \right\}$$

$$= f^\dagger(\pi_S(U))$$

□

Lemma A.7. *For any* $\sigma \in \mathsf{Mem}[S]$, $\mathcal{I} \subseteq_{\mathsf{fin}} \mathsf{Mem}[T]$ *and* $\mathcal{J} \subseteq_{\mathsf{fin}} \mathsf{Mem}[U]$ *such that* S, T, *and* U *are pairwise disjoint, if* $\bot \notin \mathsf{supp}(\nu)$ *for all* $\nu \in \mathcal{L}^{\mathcal{I}*\mathcal{J}}(\alpha)(\sigma)$, *then:*

$$\forall \tau \in \mathcal{I}.\ \pi_S(\mathcal{L}^{\mathcal{J}}(\alpha)(\sigma \uplus \tau)) \subseteq \mathcal{L}^{\mathcal{I}*\mathcal{J}}(\alpha)(\sigma) \quad \textit{and} \quad \bigcup \left\{ \mathsf{supp}(\pi_T(\nu)) \mid \nu \in \mathcal{L}^{\mathcal{J}}(\alpha)(\sigma \uplus \tau) \right\} \subseteq \mathcal{I}$$

Proof. The proof is by induction on the size of α. If α empty, then clearly the claim holds since:

$$\pi_S(\mathcal{L}^{\mathcal{J}}(\alpha)(\sigma \uplus \tau)) = \pi_S(\eta(\sigma \uplus \tau)) = \eta(\sigma) = \mathcal{L}^{\mathcal{I}*\mathcal{J}}(\alpha)(\sigma)$$

And:

$$\bigcup \left\{ \mathsf{supp}(\pi_T(\nu)) \mid \nu \in \mathcal{L}^{\mathcal{J}}(\alpha)(\sigma \uplus \tau) \right\} = \bigcup \left\{ \mathsf{supp}(\pi_T(\nu)) \mid \nu \in \eta(\sigma \uplus \tau) \right\} = \{\tau\} \subseteq \mathcal{I}$$

Now, if α is nonempty, we have:

$$\pi_S\left(\mathcal{L}^{\mathcal{J}}(\alpha)(\sigma \uplus \tau)\right) = \pi_S\left(\mathop{\&}_{\ell \in \min(\alpha)} \begin{cases} \mathcal{L}^{\mathcal{J}}(\alpha \setminus \{\ell\})^\dagger([\![a]\!]^{\mathcal{J}}_{\mathsf{Act}}(\sigma \uplus \tau)) & \text{if } \lambda_\alpha(\ell) = \mathbb{i}_{\mathsf{Act}}(a) \\ \mathcal{L}^{\mathcal{J}}(\mathsf{filter}(\alpha, \ell, b, \sigma \uplus \tau))(\sigma \uplus \tau) & \text{if } \lambda_\alpha(\ell) = \mathbb{i}_{\mathsf{Test}}(b) \end{cases} \right)$$

$$= \mathop{\&}_{\ell \in \min(\alpha)} \begin{cases} \pi_S \left(\mathcal{L}^{\mathcal{J}}(\alpha \setminus \{\ell\})^\dagger(\llbracket a \rrbracket^{\mathcal{J}}_{\mathsf{Act}}(\sigma \uplus \tau)) \right) & \text{if } \lambda_\alpha(\ell) = \mathbb{i}_{\mathsf{Act}}(a) \\ \pi_S \left(\mathcal{L}^{\mathcal{J}}(\mathsf{filter}(\alpha, \ell, b, \sigma \uplus \tau))(\sigma \uplus \tau) \right) & \text{if } \lambda_\alpha(\ell) = \mathbb{i}_{\mathsf{Test}}(b) \end{cases}$$

By Lemmas A.5 and A.6, the induction hypothesis, and the fact that the test b does not depend on J:

$$\subseteq \mathop{\&}_{\ell \in \min(\alpha)} \begin{cases} \mathcal{L}^{\mathcal{I}*\mathcal{J}}(\alpha \setminus \{\ell\})^\dagger(\llbracket a \rrbracket^{\mathcal{I}*\mathcal{J}}_{\mathsf{Act}}(\sigma)) & \text{if } \lambda_\alpha(\ell) = \mathbb{i}_{\mathsf{Act}}(a) \\ \mathcal{L}^{\mathcal{I}*\mathcal{J}}(\mathsf{filter}(\alpha, \ell, b, \sigma \uplus \tau))(\sigma) & \text{if } \lambda_\alpha(\ell) = \mathbb{i}_{\mathsf{Test}}(b) \end{cases}$$
$$= \mathcal{L}^{\mathcal{I}*\mathcal{J}}(\alpha)(\sigma)$$

The second part of the claim also clearly holds from the fact that $\bot \notin \mathsf{supp}(\nu)$ for all $\nu \in \mathcal{L}^{\mathcal{I}*\mathcal{J}}(\alpha)(\sigma)$. □

A.3 Observable Independence

Lemma A.8. *Let $\alpha, \beta \in \mathcal{P}\mathsf{om}(\mathsf{Act} + \mathsf{Test})$, such that S is the memory footprint of α, T is the footprint of β, $B_1 \subseteq \mathsf{Mem}[S]$, and $B_2 \subseteq \mathsf{Mem}[T]$. If for any $\sigma \in \mathsf{Mem}[S]$, $\tau \in \mathsf{Mem}[T]$, there exists $p, q \in \mathbb{R}$, such that:*

$$\forall \nu_1 \in \mathcal{L}^{\mathcal{I}}(\alpha)(\sigma). \sum_{\sigma' \in B_1} \nu_1(\sigma') = p \qquad \forall \nu_2 \in \mathcal{L}^{\mathcal{I}}(\beta)(\tau). \sum_{\tau' \in B_2} \nu_2(\tau') = q$$

Then:

$$\forall \nu \in \mathcal{L}^{\mathcal{I}}(\alpha \parallel \beta)(\sigma \uplus \tau). \sum_{\sigma' \in B_1} \sum_{\tau' \in B_2} \nu(\sigma' \uplus \tau') = p \cdot q$$

Proof. By induction on the combined size of α and β. If both are empty, then the claim is obvious, since $\mathcal{L}^{\mathcal{I}}(\alpha \parallel \beta)(\sigma \uplus \tau) = \eta(\sigma \uplus \tau)$, so $\nu = \delta_{\sigma \uplus \tau}$. We know by the premise that $\sum_{\sigma' \in B_1} \delta_\sigma(\sigma') = p$, so p must be either 0 or 1 depending on whether $\sigma \in B_1$. A similar claim holds for B_2. So, if $\sigma \in B_1$ and $\tau \in B_2$, then we get that $\sum_{\sigma' \in B_1} \sum_{\tau' \in B_2} \delta_{\sigma \uplus \tau}(\sigma' \uplus \tau') = \delta_{\sigma \uplus \tau}(\sigma \uplus \tau) = 1 = p \cdot q$. If not, then the either p or q is zero, and the sum is also 0.

Now suppose that at least one of α of β is nonempty. We start by establishing the premise so that we will be able to apply the induction hypothesis. More precisely, we must establish that for α' (resp. β') that is structurally smaller than α (resp. β), we have:

$$\forall \nu_1 \in \mathcal{L}^{\mathcal{I}}(\alpha')(\sigma). \sum_{\sigma' \in B_1} \nu_1(\sigma') = p \qquad \forall \nu_2 \in \mathcal{L}^{\mathcal{I}}(\beta')(\tau). \sum_{\tau' \in B_2} \nu_2(\tau') = q$$

Where structurally smaller in this case means that $\alpha' = \alpha \setminus L$ or $\beta' = \beta \setminus L$ for some L. We will not establish this for any label set L, but rather only those L for which we will later need to apply the induction hypothesis.

Take any $\nu_1 \in \mathcal{L}^{\mathcal{I}}(\alpha)(\sigma)$ and suppose that α is not empty. We know by the premise that $\sum_{\sigma' \in B_1} \nu_1(\sigma') = p$. Since this is true for any ν_1, then ν_1 could be the result of choosing any $\ell \in \min(\alpha)$ to go first (or a convex combination thereof). Fix one such ℓ. If $\lambda_\alpha(\ell) = \mathbb{i}_{\mathsf{Act}}(a)$, then we have:

$$p = \sum_{\sigma' \in B_1} \nu_1(\sigma') = \sum_{\sigma' \in B_1} \left(\sum_{\sigma'' \in \mathsf{supp}(\mu)} \mu(\sigma'') \cdot \nu_{\sigma''} \right)(\sigma') = \sum_{\sigma'' \in \mathsf{supp}(\mu)} \mu(\sigma'') \cdot \sum_{\sigma' \in B_1} \nu_{\sigma''}(\sigma')$$

Where $\mu \in \llbracket a \rrbracket^{\mathcal{I}}_{\mathsf{Act}}(\sigma)$ and $\nu_{\sigma''} \in \mathcal{L}^{\mathcal{I}}(\alpha \setminus \{\ell\})_\bot(\sigma'')$ for each σ''. Now, take any $\tau \in \mathsf{supp}(\mu)$ and $\xi \in \mathcal{L}^{\mathcal{I}}(\alpha \setminus \{\ell\})(\sigma'')$. By construction, $\mu(\tau) \cdot \xi + \sum_{\sigma'' \in \mathsf{supp}(\mu) \setminus \{\tau\}} \mu(\sigma'') \cdot \nu_{\sigma''} \in \mathcal{L}^{\mathcal{I}}(\alpha)(\sigma)$, therefore we know that:

$$p = \sum_{\sigma' \in B_1} \left(\mu(\tau) \cdot \xi + \sum_{\sigma'' \in \mathrm{supp}(\mu) \setminus \{\tau\}} \mu(\sigma'') \cdot \nu_{\sigma''} \right)(\sigma') = \mu(\tau) \cdot \sum_{\sigma' \in B_1} \xi(\sigma') + \sum_{\sigma'' \in \mathrm{supp}(\mu) \setminus \{\tau\}} \mu(\sigma'') \cdot \sum_{\sigma' \in B_1} \nu_{\sigma''}(\sigma')$$

And so:

$$p = \mu(\tau) \cdot \sum_{\sigma' \in B_1} \xi(\sigma') + \sum_{\sigma'' \in \mathrm{supp}(\mu) \setminus \{\tau\}} \mu(\sigma'') \cdot \sum_{\sigma' \in B_1} \nu_{\sigma''}(\sigma') = \sum_{\sigma'' \in \mathrm{supp}(\mu)} \mu(\sigma'') \cdot \sum_{\sigma' \in B_1} \nu_{\sigma''}(\sigma')$$

$$\mu(\tau) \cdot \sum_{\sigma' \in B_1} \xi(\sigma') + \cancel{\sum_{\sigma'' \in \mathrm{supp}(\mu) \setminus \{\tau\}} \mu(\sigma'') \cdot \sum_{\sigma' \in B_1} \nu_{\sigma''}(\sigma')} = \mu(\tau) \cdot \sum_{\sigma' \in B_1} \nu_\tau(\sigma') + \cancel{\sum_{\sigma'' \in \mathrm{supp}(\mu) \setminus \{\tau\}} \mu(\sigma'') \cdot \sum_{\sigma' \in B_1} \nu_{\sigma''}(\sigma')}$$

$$\cancel{\mu(\tau)} \cdot \sum_{\sigma' \in B_1} \xi(\sigma') = \cancel{\mu(\tau)} \cdot \sum_{\sigma' \in B_1} \nu_\tau(\sigma')$$

$$\sum_{\sigma' \in B_1} \xi(\sigma') = \sum_{\sigma' \in B_1} \nu_\tau(\sigma')$$

Therefore, we have just shown that $\sum_{\sigma' \in B_1} \xi(\sigma') = \sum_{\sigma' \in B_1} \nu_\tau(\sigma')$ for any $\xi \in \mathcal{L}^{\mathcal{I}}(\alpha \setminus \{\ell\})(\sigma'')$, therefore $\sum_{\sigma' \in B_1} \xi(\sigma')$ is constant.

If instead $\lambda_\alpha(\ell) = \mathbb{i}_{\mathsf{Test}}(b)$, then:

$$p = \sum_{\sigma' \in B_1} \nu_1(\sigma') = \sum_{\sigma' \in B_1} \mu(\sigma')$$

Where $\mu \in \mathcal{L}^{\mathcal{I}}(\mathsf{filter}(\alpha, \ell, b, \sigma)$, so clearly the sum above is also constant. A symmetric argument applies for β. We have therefore established that the premises of the induction hypothesis, so we will be able to apply it shortly.

Now, for any $\nu \in \mathcal{L}^{\mathcal{I}}(\alpha \parallel \beta)(\sigma \uplus \tau)$, we have:

$$\nu = r \cdot \sum_{\ell \in \mathrm{supp}(\xi_\alpha)} \xi_\alpha(\ell) \cdot \begin{cases} \sum_{\sigma' \in \mathrm{supp}(\mu_{\ell,a,\sigma})} \mu_{\ell,a,\sigma}(\sigma') \cdot \nu_{\alpha \setminus \{\ell\} \parallel \beta, \sigma' \uplus \tau} & \text{if } \lambda_\alpha(\ell) = \mathbb{i}_{\mathsf{Act}}(a) \\ \nu_{\mathsf{filter}(\alpha,b,\sigma) \parallel \beta, \sigma \uplus \tau} & \text{if } \lambda_\alpha(\ell) = \mathbb{i}_{\mathsf{Test}}(b) \end{cases}$$
$$+ (1-r) \cdot \sum_{\ell \in \mathrm{supp}(\xi_\beta)} \xi_\beta(\ell) \cdot \begin{cases} \sum_{\tau' \in \mathrm{supp}(\mu_{\ell,a,\tau})} \mu_{\ell,a,\tau}(\tau') \cdot \nu_{\alpha \parallel \beta \setminus \{\ell\}, \sigma \uplus \tau'} & \text{if } \lambda_\beta(\ell) = \mathbb{i}_{\mathsf{Act}}(a) \\ \nu_{\alpha \parallel \mathsf{filter}(\beta,b,\tau), \sigma \uplus \tau} & \text{if } \lambda_\beta(\ell) = \mathbb{i}_{\mathsf{Test}}(b) \end{cases}$$

Where $r \in [0,1]$, $\xi_\alpha \in \mathcal{D}(\min(\alpha))$, $\xi_\beta \in \mathcal{D}(\min(\beta))$, $\mu_{\ell,a,\sigma} \in [\![a]\!]^{\mathcal{I}}_{\mathsf{Act}}(\sigma)$, and $\nu_{\gamma,\sigma'} \in \mathcal{L}^{\mathcal{I}}(\gamma)_\bot(\sigma')$. This gives us:

$$\sum_{\sigma'' \in B_1} \sum_{\tau'' \in B_2} \nu(\sigma'' \uplus \tau'')$$
$$= r \cdot \sum_{\ell \in \mathrm{supp}(\xi_\alpha)} \xi_\alpha(\ell) \cdot \begin{cases} \sum_{\sigma' \in \mathrm{supp}(\mu_{\ell,a,\sigma})} \mu_{\ell,a,\sigma}(\sigma') \cdot \sum_{\sigma'' \in B_1} \sum_{\tau'' \in B_2} \nu_{\alpha \setminus \{\ell\} \parallel \beta, \sigma' \uplus \tau}(\sigma'' \uplus \tau'') & \text{if } \lambda_\alpha(\ell) = \mathbb{i}_{\mathsf{Act}}(a) \\ \sum_{\sigma'' \in B_1} \sum_{\tau'' \in B_2} \nu_{\mathsf{filter}(\alpha,b,\sigma) \parallel \beta, \sigma \uplus \tau}(\sigma'' \uplus \tau'') & \text{if } \lambda_\alpha(\ell) = \mathbb{i}_{\mathsf{Test}}(b) \end{cases}$$
$$+ (1-r) \cdot \sum_{\ell \in \mathrm{supp}(\xi_\beta)} \xi_\beta(\ell) \cdot \begin{cases} \sum_{\tau' \in \mathrm{supp}(\mu_{\ell,a,\tau})} \mu_{\ell,a,\tau}(\tau') \cdot \sum_{\sigma'' \in B_1} \sum_{\tau'' \in B_2} \nu_{\alpha \parallel \beta \setminus \{\ell\}, \sigma \uplus \tau'}(\sigma'' \uplus \tau'') & \text{if } \lambda_\beta(\ell) = \mathbb{i}_{\mathsf{Act}}(a) \\ \sum_{\sigma'' \in B_1} \sum_{\tau'' \in B_2} \nu_{\alpha \parallel \mathsf{filter}(\beta,b,\tau), \sigma \uplus \tau}(\sigma'' \uplus \tau'') & \text{if } \lambda_\beta(\ell) = \mathbb{i}_{\mathsf{Test}}(b) \end{cases}$$

By the induction hypothesis, and since $\sigma'', \tau'' \neq \bot$:

$$= r \cdot \sum_{\ell \in \mathrm{supp}(\xi_\alpha)} \xi_\alpha(\ell) \cdot \begin{cases} \sum_{\sigma' \in \mathrm{supp}(\mu_{\ell,a,\sigma})} \mu_{\ell,a,\sigma}(\sigma') \cdot p_{\alpha \setminus \{\ell\}, \sigma'} \cdot q_{\beta,\tau} & \text{if } \lambda_\alpha(\ell) = \mathbb{i}_{\mathsf{Act}}(a) \\ p_{\mathsf{filter}(\alpha,b,\sigma),\sigma} \cdot q_{\beta,\tau} & \text{if } \lambda_\alpha(\ell) = \mathbb{i}_{\mathsf{Test}}(b) \end{cases}$$

$$+ (1-r) \cdot \sum_{\ell \in \mathsf{supp}(\xi_\beta)} \xi_\beta(\ell) \cdot \begin{cases} \sum_{\tau' \in \mathsf{supp}(\mu_{\ell,a,\tau})} \mu_{\ell,a,\tau}(\tau') \cdot p_{\alpha,\sigma} \cdot q_{\beta\setminus\{\ell\},\tau'} & \text{if } \lambda_\beta(\ell) = \mathbb{i}_{\mathsf{Act}}(a) \\ p_{\alpha,\sigma} \cdot q_{\mathsf{filter}(\beta,b,\tau),\tau} & \text{if } \lambda_\beta(\ell) = \mathbb{i}_{\mathsf{Test}}(b) \end{cases}$$

$$= r \cdot q_{\beta,\tau} \cdot \sum_{\ell \in \mathsf{supp}(\xi_\alpha)} \xi_\alpha(\ell) \cdot \begin{cases} \sum_{\sigma' \in \mathsf{supp}(\mu_{\ell,a,\sigma})} \mu_{\ell,a,\sigma}(\sigma') \cdot p_{\alpha\setminus\{\ell\},\sigma'} & \text{if } \lambda_\alpha(\ell) = \mathbb{i}_{\mathsf{Act}}(a) \\ p_{\mathsf{filter}(\alpha,b,\sigma),\sigma} & \text{if } \lambda_\alpha(\ell) = \mathbb{i}_{\mathsf{Test}}(b) \end{cases}$$

$$+ (1-r) \cdot p_{\alpha,\sigma} \cdot \sum_{\ell \in \mathsf{supp}(\xi_\beta)} \xi_\beta(\ell) \cdot \begin{cases} \sum_{\tau' \in \mathsf{supp}(\mu_{\ell,a,\tau})} \mu_{\ell,a,\tau}(\tau') \cdot q_{\beta\setminus\{\ell\},\tau'} & \text{if } \lambda_\beta(\ell) = \mathbb{i}_{\mathsf{Act}}(a) \\ q_{\mathsf{filter}(\beta,b,\tau),\tau} & \text{if } \lambda_\beta(\ell) = \mathbb{i}_{\mathsf{Test}}(b) \end{cases}$$

Now, by the premise, we can replace the p and q terms with arbitrary representative distributions, since we know the cumulative mass for B_1 and B_2 is constant.

$$= r \cdot q_{\beta,\tau} \cdot \sum_{\sigma'' \in B_1} \sum_{\ell \in \mathsf{supp}(\xi_\alpha)} \xi_\alpha(\ell) \cdot \begin{cases} \sum_{\sigma' \in \mathsf{supp}(\mu_{\ell,a,\sigma})} \mu_{\ell,a,\sigma}(\sigma') \cdot \nu_{\alpha\setminus\{\ell\},\sigma'}(\sigma'') & \text{if } \lambda_\alpha(\ell) = \mathsf{Act}(a) \\ \nu_{\mathsf{filter}(\alpha,b,\sigma),\sigma}(\sigma'') & \text{if } \lambda_\alpha(\ell) = \mathsf{Test}(b) \end{cases}$$

$$+ (1-r) \cdot p_{\alpha,\sigma} \cdot \sum_{\tau'' \in B_2} \sum_{\ell \in \mathsf{supp}(\xi_\beta)} \xi_\beta(\ell) \cdot \begin{cases} \sum_{\tau' \in \mathsf{supp}(\mu_{\ell,a,\tau})} \mu_{\ell,a,\tau}(\tau') \cdot \nu_{\beta\setminus\{\ell\},\tau'}(\tau'') & \text{if } \lambda_\beta(\ell) = \mathsf{Act}(a) \\ \nu_{\mathsf{filter}(\beta,b,\tau),\tau}(\tau'') & \text{if } \lambda_\beta(\ell) = \mathsf{Test}(b) \end{cases}$$

$$= r \cdot q_{\beta,\tau} \cdot \sum_{\sigma'' \in B_1} \nu_{\alpha,\sigma}(\sigma'') + (1-r) \cdot p_{\alpha,\sigma} \cdot \sum_{\tau'' \in B_2} \nu_{\beta,\tau}\tau''$$

$$= r \cdot q_{\beta,\tau} \cdot p_{\alpha,\sigma} + (1-r) \cdot p_{\alpha,\sigma} \cdot q_{\beta,\tau}$$

$$= p_{\alpha,\sigma} \cdot q_{\beta,\tau}$$

□

Lemma A.9. *For all $k \in \{1,2\}$, $\mathcal{P}_k$, $\mathcal{Q}_k$, α_k, and μ such that $\mathcal{P}_1 \otimes \mathcal{P}_2 \preceq \mu$, if:*

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

Then:

$$\forall \nu \in \mathcal{L}^{\mathcal{I}}(\alpha_1 \parallel \alpha_2)^\dagger(\mu).\ \mathcal{Q}_1 \otimes \mathcal{Q}_2 \preceq \nu$$

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 $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively. Since $\mathcal{P}_1 \otimes \mathcal{P}_2 \preceq \mu$, we know that for any $(i,j) \in I_1 \times I_2$:

$$\sum_{\sigma \in A_{1,i}} \sum_{\tau \in A_{2,j}} \mu(\sigma \uplus \tau) = \mu_{\mathcal{P}_1}(A_{1,i}) \cdot \mu_{\mathcal{P}_2}(A_{2,j}) \tag{2}$$

Also, for any μ_k such that $\mathcal{P}_k \preceq \mu_k$, 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 $\nu_\sigma \in \mathcal{L}^{\mathcal{I}}(\alpha_k)(\sigma)$. Since $\mathcal{Q}_k \preceq \nu_k$, we know that for any $B \in \mathcal{F}_{\mathcal{Q}_k}$:

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

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

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

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

$$
\begin{aligned}
\mu_{Q_k}(B) = \sum_{\tau\in B}\left(\mu_k(\sigma')\cdot\xi + \sum_{\sigma\in\mathsf{supp}(\mu_k)\setminus\{\sigma'\}}\mu_k(\sigma)\cdot\nu_\sigma\right)(\tau) &= \sum_{\sigma\in\mathsf{supp}(\mu_k)}\mu_k(\sigma)\cdot p_{B,\sigma} \\
\mu_k(\sigma')\cdot\sum_{\tau\in B}\xi(\tau) + \cancel{\sum_{\sigma\in\mathsf{supp}(\mu_k)\setminus\{\sigma'\}}\mu_k(\sigma)\cdot p_{B,\sigma}} &= \mu_k(\sigma')\cdot p_{B,\sigma'} + \cancel{\sum_{\sigma\in\mathsf{supp}(\mu_k)\setminus\{\sigma'\}}\mu_k(\sigma)\cdot p_{B,\sigma}} \\
\cancel{\mu_k(\sigma')}\cdot\sum_{\tau\in B}\xi(\tau) &= \cancel{\mu_k(\sigma')}\cdot p_{B,\sigma'} \\
\sum_{\tau\in B}\xi(\tau) &= p_{B,\sigma'}
\end{aligned}
$$

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

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

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

$$
\begin{aligned}
\mu_{Q_k}(B) = \mu_{\mathcal{P}_k}(A_{k,j})\cdot p_{B,\sigma'} + \sum_{i\neq j}\mu_{\mathcal{P}_k}(A_{k,i})\cdot p_{B,\sigma_i} &= \sum_{i\in I_k}\mu_{\mathcal{P}_k}(A_{k,i})\cdot p_{B,k,i} \\
\mu_{\mathcal{P}_k}(A_{k,j})\cdot p_{B,\sigma'} + \cancel{\sum_{i\neq j}\mu_{\mathcal{P}_k}(A_{k,i})\cdot p_{B,k,i}} &= \mu_{\mathcal{P}_k}(A_{k,j})\cdot p_{B,k,j} + \cancel{\sum_{i\neq j}\mu_{\mathcal{P}_k}(A_{k,i})\cdot p_{B,k,i}} \\
\cancel{\mu_{\mathcal{P}_k}(A_{k,j})}\cdot p_{B,\sigma'} &= \cancel{\mu_{\mathcal{P}_k}(A_{k,j})}\cdot p_{B,k,j} \\
p_{B,\sigma'} &= p_{B,k,j}
\end{aligned}
$$

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

Now take any $\nu \in \mathcal{L}^{\mathcal{I}}(\alpha_1 \parallel \alpha_2)^\dagger(\mu)$, which must have the form $\nu = \sum_{\sigma\in\mathsf{supp}(\mu)}\mu(\sigma)\cdot\nu_\sigma$ where $\nu_\sigma \in \mathcal{L}^{\mathcal{I}}(\alpha_1 \parallel \alpha_2)(\sigma)$ 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 $Q_1 \otimes Q_2 \preceq \nu$, which we do as follows:

$$
\begin{aligned}
\sum_{\sigma'\in B_1}\sum_{\tau'\in B_2}\nu(\sigma' \uplus \tau') &= \sum_{\sigma'\in B_1}\sum_{\tau'\in B_2}\sum_{\sigma\in\mathsf{supp}(\mu)}\mu(\sigma)\cdot\nu_\sigma(\sigma' \uplus \tau') \\
&= \sum_{\sigma\in\mathsf{supp}(\mu)}\mu(\sigma)\cdot\sum_{\sigma'\in B_1}\sum_{\tau'\in B_2}\nu_\sigma(\sigma' \uplus \tau')
\end{aligned}
$$

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.

$$
= \sum_{i\in I_1}\sum_{j\in I_2}\sum_{\sigma\in A_{1,i}}\sum_{\tau\in A_{2,j}}\mu(\sigma \uplus \tau)\cdot\sum_{\sigma'\in B_1}\sum_{\tau'\in B_2}\nu_{\sigma\uplus\tau}(\sigma' \uplus \tau')
$$

We previously showed that $\sum_{\sigma'\in B_k}\nu_k(\sigma') = p_{B_k,k,i}$ (a constant) for any $\nu_k \in \mathcal{L}^{\mathcal{I}}(\alpha_k)(\sigma)$ where $\sigma \in A_{k,i}$. So, we can use Lemma A.8 to conclude that:

$$
= \sum_{i\in I_1}\sum_{j\in I_2}\sum_{\sigma\in A_{1,i}}\sum_{\tau\in A_{2,j}}\mu(\sigma \uplus \tau)\cdot p_{B_1,1,i}\cdot p_{B_2,2,j}
$$

By Equation (2).

$$= \sum_{i \in I_1} \sum_{j \in I_2} \mu_{\mathcal{P}_1}(A_{1,i}) \cdot \mu_{\mathcal{P}_2}(A_{2,j}) \cdot p_{B_1,1,i} \cdot p_{B_2,2,j}$$
$$= \left(\sum_{i \in I_1} \mu_{\mathcal{P}_1}(A_{1,i}) \cdot p_{B_1,1,i} \right) \cdot \left(\sum_{j \in I_2} \mu_{\mathcal{P}_2}(A_{2,j}) \cdot p_{B_2,2,j} \right)$$
$$= \mu_{Q_1}(B_1) \cdot \mu_{Q_2}(B_2)$$

□

B Measure Theory Lemmas

Lemma B.1.

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

Proof. Let $\mathcal{P}_v = \langle \Omega_v, \mathcal{F}_v, \mu_v \rangle$ for each v and $Q = \langle \Omega', \mathcal{F}', \mu' \rangle$. If the above are well defined, then there must be S and T such that $\Omega_v \subseteq \mathsf{Mem}[S]$ and $\Omega' \subseteq \mathsf{Mem}[T]$. We first show that both probability spaces have the same sample space, since:

$$(\bigcup_{v \in \mathsf{supp}(\nu)} \Omega_v) * \Omega' = \left\{ \sigma \uplus \sigma' \mid \sigma \in \bigcup_{v \in \mathsf{supp}(\nu)} \Omega_v, \sigma' \in \Omega' \right\} = \bigcup_{v \in \mathsf{supp}(\nu)} \{ \sigma \uplus \sigma' \mid \sigma \in \Omega_v, \sigma' \in \Omega' \} = \bigcup_{v \in \mathsf{supp}(\nu)} \Omega_v * \Omega'$$

The σ-algebras are also the same, since:

$$\left\{ A * B \;\middle|\; A \in \left\{ A' \;\middle|\; A' \subseteq \bigcup_{v \in \mathsf{supp}(\nu)} \Omega_v, \forall v.\, A' \cap \Omega_v \in \mathcal{F}_v \right\}, B \in \mathcal{F}' \right\}$$
$$= \left\{ A * B \;\middle|\; A \subseteq \bigcup_{v \in \mathsf{supp}(\nu)} \Omega_v, \forall v.\, A \cap \Omega_v \in \mathcal{F}_v, B \in \mathcal{F}' \right\}$$

Let $\mathcal{F}'_v = \{ A * B \mid A \in \mathcal{F}_v, B \in \mathcal{F}' \}$.

$$= \left\{ A \;\middle|\; A \subseteq \bigcup_{v \in \mathsf{supp}(\nu)} \Omega_v * \Omega', \forall v.\, A \cap \Omega_v * \Omega' \in \mathcal{F}'_v \right\}$$

Finally, the probability measures agree. Let μ_1 and μ_2 be the probability measures on the left and right, respectively. Then we get:

$$\mu_1(A) = \left(\sum_{v \in \mathsf{supp}(\nu)} \nu(v) \cdot \mu_v(\pi_S(A) \cap \Omega_v) \right) \cdot \mu'(\pi_T(A))$$
$$= \sum_{v \in \mathsf{supp}(\nu)} \nu(v) \cdot \mu_v(\pi_S(A \cap \Omega_v)) \cdot \mu'(\pi_T(A))$$
$$= \sum_{v \in \mathsf{supp}(\nu)} \nu(v) \cdot \mu_v(\pi_S(A \cap (\Omega_v * \Omega'))) \cdot \mu'(\pi_T(A \cap (\Omega_v * \Omega')))$$
$$= \mu_2(A)$$

□

Lemma B.2 (Monotonicity of $\otimes$). *If $\mathcal{P} \preceq \mathcal{P}'$ then $\mathcal{P} \otimes \mathcal{Q} \preceq \mathcal{P}' \otimes \mathcal{Q}$.*

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

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

Now we show the condition on σ-algebras:

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

Finally, we show the condition on probability spaces:

$$\begin{aligned}
&\mu_{\mathcal{P}\otimes\mathcal{Q}}(A) \\
&= \mu_{\mathcal{P}}(\pi_S(A)) \cdot \mu_{\mathcal{Q}}(\pi_T(A))
\end{aligned}$$

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

$$\begin{aligned}
&= \mu_{\mathcal{P}'}\left(\bigcup\{B \in \mathcal{F}_{\mathcal{P}'} \mid \pi_S(B) = \pi_S(A)\}\right) \cdot \mu_{\mathcal{Q}}(\pi_T(A)) \\
&= \mu_{\mathcal{P}'}\left(\bigcup\{\pi_{S'}(B) \mid B \in \mathcal{F}_{\mathcal{P}'\otimes\mathcal{Q}},\ \pi_{S\cup T}(B) = A\}\right) \cdot \mu_{\mathcal{Q}}\left(\bigcup\{\pi_T(B) \mid B \in \mathcal{F}_{\mathcal{P}'\otimes\mathcal{Q}},\ \pi_{S\cup T}(B) = A\}\right) \\
&= \mu_{\mathcal{P}'}\left(\pi_{S'}\left(\bigcup\{B \in \mathcal{F}_{\mathcal{P}'\otimes\mathcal{Q}} \mid \pi_{S\cup T}(B) = A\}\right)\right) \cdot \mu_{\mathcal{Q}}\left(\pi_T\left(\bigcup\{B \in \mathcal{F}_{\mathcal{P}'\otimes\mathcal{Q}} \mid \pi_{S\cup T}(B) = A\}\right)\right) \\
&= \mu_{\mathcal{P}'\otimes\mathcal{Q}}\left(\bigcup\{B \in \mathcal{F}_{\mathcal{P}'\otimes\mathcal{Q}} \mid \pi_{S\cup T}(B) = A\}\right)
\end{aligned}$$

□

Lemma B.3 (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:

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

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

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

□

Lemma B.4. *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 $\mathcal{P}\preceq\mathsf{comp}(\mathcal{P}_i)$ 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:

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

Next, we verify the property on σ-algebras.

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

Note that the completion adds information about events outside of $\Omega_{\mathcal{P}_i}$, so if A is in all of the $\mathcal{F}_{\mathsf{comp}(\mathcal{P}_i)}$ 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.

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

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.

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

□

Lemma B.5. *If* $\mathcal{P} \preceq \mathsf{comp}(\mathcal{P}_i)$ *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:

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

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:

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

□

C Logic and Rules

Lemma C.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.
- $\varphi = \varphi_1 \land \varphi_2$. By the induction hypothesis, we know that $\Gamma, \mathcal{Q} \vDash \varphi_1$ and $\Gamma, \mathcal{Q} \vDash \varphi_2$, therefore $\Gamma, \mathcal{Q} \vDash \varphi_1 \land \varphi_2$.
- $\varphi = \varphi_1 \lor \varphi_2$. 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 $\Gamma, \mathcal{Q} \vDash \varphi_1 \lor \varphi_2$.
- $\varphi = \bigoplus_{X\sim d(E)} \psi$. Immediate, since the semantics stipulates that $\mathcal{P}$ is greater than the direct sum, therefore $\mathcal{Q}$ is also greater than the direct sum.
- $\varphi = \varphi_1 * \varphi_2$. Immediate, since 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 $\mu_{\mathcal{P}}((\!|P|\!)^S_\Gamma) = \mu_{\mathcal{Q}}(\bigcup_{B \mid \pi_S(B) = (\!|P|\!)^S_\Gamma} B) = 1$. Note that by definition $\bigcup_{B \mid \pi_S(B) = (\!|P|\!)^S_\Gamma} B \subseteq (\!|P|\!)^T_\Gamma$. 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.

□

C.1 Precision

Lemma C.2. 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:

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

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 C.3. *If* precise(φ, ψ), *then* 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.2, we get:

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

□

Lemma C.4. *If* precise(φ) *and* $\varphi \Rightarrow \lceil e \mapsto X \rceil$, *then* 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 $\mathcal{P}_v = \mathsf{comp}(\mathcal{P}'_v)$. 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 $\Gamma[X := v], \mathsf{comp}(\mathcal{Q}_v) \vDash \varphi$, where $\bigoplus_{v \sim \nu} \mathcal{Q}_v \preceq \mathcal{Q}$. Since φ is precise, then $\mathcal{P}_v = \mathsf{comp}(\mathcal{P}'_v) \preceq \mathsf{comp}(\mathcal{Q}_v)$ 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.3:

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

□

C.2 Entailment Rules

Lemma C.5. *The entailment rules in Figure 5 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) $\lceil P * Q \rceil \dashv\vdash \lceil P \rceil * \lceil Q \rceil$

We first show that $\lceil P * Q \rceil \vdash \lceil P \rceil * \lceil Q \rceil$. 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 $\Gamma, \mathcal{P}_1 \otimes \mathcal{P}_2 \vDash \lceil P \rceil * \lceil Q \rceil$. 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:

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

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

Now we show that $\lceil P \rceil * \lceil Q \rceil \vdash \lceil P * Q \rceil$, suppose that $\Gamma, \mathcal{P} \vDash \lceil P \rceil * \lceil Q \rceil$. That means that $\mathcal{P}_1 \otimes \mathcal{P}_2 \preceq \mathcal{P}$ such that $\Gamma, \mathcal{P}_1 \vDash \lceil P \rceil$ and $\Gamma, \mathcal{P}_2 \vDash \lceil Q \rceil$ where $\Omega_{\mathcal{P}_1} = \mathsf{Mem}[S]$ and $\Omega_{\mathcal{P}_2} = \mathsf{Mem}[T]$. This also means that $\mu_{\mathcal{P}_1}((\!|P|\!)^S_\Gamma) = 1$ and $\mu_{\mathcal{P}_2}((\!|Q|\!)^T_\Gamma) = 1$. Therefore, we have that:

$$\begin{aligned}
\mu_{\mathcal{P}_1 \otimes \mathcal{P}_2}\left((\!|P * Q|\!)^{S \cup T}_\Gamma\right) &= \mu_{\mathcal{P}_1 \otimes \mathcal{P}_2}\left(\{\sigma \in \mathsf{Mem}[S \cup T] \mid \Gamma, \sigma \vDash P * Q\}\right) \\
&= \mu_{\mathcal{P}_1 \otimes \mathcal{P}_2}\left(\{\sigma \uplus \tau \mid \Gamma, \sigma \vDash P,\ \Gamma, \tau \vDash Q\}\right) \\
&= \mu_{\mathcal{P}_1 \otimes \mathcal{P}_2}\left((\!|P|\!)^S_\Gamma * (\!|Q|\!)^T_\Gamma\right) \\
&= \mu_{\mathcal{P}_1}\left((\!|P|\!)^S_\Gamma\right) \cdot \mu_{\mathcal{P}_2}\left((\!|Q|\!)^T_\Gamma\right) = 1 \cdot 1 = 1
\end{aligned}$$

So, $\Gamma, \mathcal{P}_1 \otimes \mathcal{P}_2 \vDash (\!|P * Q|\!)$, and since $\mathcal{P}_1 \otimes \mathcal{P}_2 \preceq \mathcal{P}$, then by Lemma C.1, $\Gamma, \mathcal{P} \vDash (\!|P * Q|\!)$.

(3) $\dfrac{\varphi \vdash \psi}{\bigoplus_{X\sim d(E)} \varphi \vdash \bigoplus_{X\sim d(E)} \psi}$

Suppose that $\Gamma, \mathcal{P} \vDash \bigoplus_{X\sim d(E)} \varphi$ and let $\nu = d([\![E]\!]_{\mathsf{LExp}}(\Gamma)$. So, $\bigoplus_{v\sim\nu} \mathcal{P}_v \preceq \mathcal{P}$ such that $\Gamma[X := v], \mathsf{comp}(\mathcal{P}_v) \vDash \varphi$ for each v. Since $\varphi \vdash \psi$, we get that $\Gamma[X := v], \mathsf{comp}(\mathcal{P}_v) \vDash \psi$ 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.

(4) $\dfrac{X \notin \mathsf{fv}(\psi)}{(\bigoplus_{X\sim d(E)} \varphi) * \psi \vdash \bigoplus_{X\sim d(E)} (\varphi * \psi)}$

Suppose that $\Gamma, \mathcal{P} \vDash (\bigoplus_{X\sim d(E)} \varphi) * \psi$ and let $\nu = d([\![E]\!]_{\mathsf{LExp}}(\Gamma))$. So, $(\bigoplus_{v\sim\nu} \mathcal{P}_v) \otimes \mathcal{P}' \preceq \mathcal{P}$ such that $\Gamma[X := v], \mathsf{comp}(\mathcal{P}_v) \vDash \varphi$ for each v and $\Gamma, \mathcal{P}' \vDash \psi$. Since $X \notin \mathsf{fv}(\psi)$, then $\Gamma[X := v], \mathcal{P}' \vDash \psi$ for each v. Also note that $\mathsf{comp}(\mathcal{P}_v) \otimes \mathcal{P}' = \mathsf{comp}(\mathcal{P}_v \otimes \mathcal{P}')$ since $\mathcal{P}'$ is already a complete probability space. This gives us $\Gamma[X := v], \mathsf{comp}(\mathcal{P}_v \otimes \mathcal{P}') \vDash \varphi * \psi$. Now, by Lemma B.1, 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, $\Gamma, \mathcal{P} \vDash \bigoplus_{X\sim d(E)}(\varphi * \psi)$.

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

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 $\Gamma[X := v], \mathsf{comp}(\mathcal{P}_v) \vDash \varphi$ for each $v \in \mathsf{supp}(\nu)$. Since $Y \notin \mathsf{fv}(\varphi)$, then clearly $\Gamma[Y := v], \mathsf{comp}(\mathcal{P}_v) \vDash \varphi[Y/X]$. 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 C.1.

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

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 $\Gamma[X := v], \mathsf{comp}(\mathcal{P}_v) \vDash \varphi$ and $\Gamma[X := v], \mathsf{comp}(\mathcal{Q}_v) \vDash \psi$. Since $X \notin \mathsf{fv}(\psi)$, we also know that $\Gamma, \mathsf{comp}(\mathcal{Q}_v) \vDash \psi$, and since ψ is precise, there is a unique $\mathcal{Q}$ such that $\Gamma, \mathcal{Q} \vDash \psi$ and $\mathcal{Q} \preceq \mathsf{comp}(\mathcal{Q}_v)$ for all v. Therefore, by recombining the components, we get that $\Gamma, \mathcal{P} \vDash (\bigoplus_{X\sim d(E)} \varphi) * \psi$.

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

Suppose that $\Gamma, \mathcal{P} \vDash \bigoplus_{X\sim d(E)} \varphi$, and let $\nu = d(\llbracket E\rrbracket_{\mathsf{LExp}}(\Gamma))$. This means that $\bigoplus_{v\sim\nu}\mathcal{P}_v \preceq \mathcal{P}$ and $\Gamma[X := v], \mathsf{comp}(\mathcal{P}_v) \vDash \varphi$. Since $X \notin \mathsf{fv}(\varphi)$,this also means that $\Gamma, \mathsf{comp}(\mathcal{P}_v) \vDash \varphi$. Since φ is precise, there is some $\mathcal{Q} \preceq \mathsf{comp}(\mathcal{P}_v)$ such that $\Gamma, \mathcal{Q} \vDash \varphi$. By Lemma B.5, we know that $\mathcal{Q} \preceq \bigoplus_{v\sim\nu}\mathcal{P}_v \preceq \mathcal{P}$, so by Lemma C.1 we conclude that $\Gamma, \mathcal{P} \vDash \varphi$.

□

C.3 Soundness of Inference Rules

Theorem 5.2 (Soundness). *For all of the rules in Figures 6 and 7, if $I \vdash \langle\varphi\rangle\ C\ \langle\psi\rangle$ then $I \vDash \langle\varphi\rangle\ C\ \langle\psi\rangle$.*

Proof. By induction on the derivation.

▹ Skip.

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

Suppose that $\Gamma, \mathcal{P} \vdash \varphi$ and $\mathcal{P} \otimes \mathcal{P}_F \preceq \mu$. So, we have:

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

So clearly $\mathcal{P} \otimes \mathcal{P}_F \preceq \mu$ and $\Gamma, \mathcal{P} \vDash \varphi$.

▹ Seq.

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

Suppose that $\Gamma, \mathcal{P} \vdash \varphi$ and $\mathcal{P} \otimes \mathcal{P}_F \preceq \mu$. By Lemma A.2, we also have that:

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

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 is some $\mathcal{P}'$ such that $\mathcal{P}' \otimes \mathcal{P}_F \preceq \mu'$ and $\Gamma, \mathcal{P}' \vDash \vartheta$ and therefore there is some $\mathcal{Q}$ such that $\mathcal{Q} \otimes \mathcal{P}_F \preceq \nu$ and $\Gamma, \mathcal{Q} \vDash \psi$.

▷ Par.

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

Suppose that $\mathcal{P}_1 \otimes \mathcal{P}_2 \otimes \mathcal{P}_F \preceq \mu$ such that $\Gamma, \mathcal{P}_1 \vDash \varphi_1$ and $\Gamma, \mathcal{P}_2 \vDash \varphi_2$. To complete the proof, we need to establish the premise of Lemma A.9. 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 $\mathcal{P}_1 \preceq \mu_2$ and any $\nu_1 \in \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C_1 \rrbracket)^\dagger(\mu_1)$. By the premise of the Par rule, we know that there is a $\mathcal{Q}_1' \preceq \nu_1$ such that $\Gamma, \mathcal{Q}_1' \vDash \psi_1$, therefore $\mathcal{Q}_1 \preceq \mathcal{Q}_1'$, and therefore we have shown that:

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

We now perform a nearly identical argument for C_2, but we also handle the frame. Take any μ_2 such that $\mathcal{P}_2 \otimes \mathcal{P}_F \preceq \mu_2$ and any $\nu_2 \in \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C_2 \rrbracket)^\dagger(\mu_2)$. 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 $\Gamma, \mathcal{Q}_2' \vDash \psi_2$. Due to precision, we know that $\mathcal{Q}_2 \preceq \mathcal{Q}_2'$ and so by Lemma B.2, $\mathcal{Q}_2 \otimes \mathcal{P}_F \preceq \mathcal{Q}_2' \otimes \mathcal{P}_F \preceq \nu_2$, so we have shown that:

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

Now, by Lemma A.9, $\mathcal{Q}_1 \otimes \mathcal{Q}_2 \otimes \mathcal{P}_F \preceq \nu$ for all $\nu \in \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C_1 \rrbracket \parallel \llbracket C_2 \rrbracket)^\dagger(\mu)$. Since we already established that $\Gamma, \mathcal{Q}_1 \vDash \psi_1$ and $\Gamma, \mathcal{Q}_2 \vDash \psi_2$, we get that $\Gamma, \mathcal{Q}_1 \otimes \mathcal{Q}_2 \vDash \psi_1 * \psi_2$, so we are done.

▷ If1.

$$\frac{\varphi \Rightarrow \lceil b \mapsto \mathsf{true} \rceil \qquad I \vdash \langle \varphi \rangle\ C_1\ \langle \psi \rangle}{I \vdash \langle \varphi \rangle\ \mathbf{if}\ b\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2\ \langle \psi \rangle}\textsc{If1}$$

Suppose that $\mathcal{P} \otimes \mathcal{P}_F \preceq \mu$ and $\Gamma, \mathcal{P} \vDash \varphi$. We therefore know that $\llbracket b \rrbracket_{\mathsf{Test}}(\sigma) = \mathsf{true}$ for all $\sigma \in \mathsf{supp}(\mu)$. By Lemma A.3, we therefore get that:

$$\mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket \mathbf{if}\ b\ \mathbf{then}\ C_1\ \mathbf{else}\ C_2 \rrbracket)^\dagger(\mu) = \mathcal{L}^{(\!|I|\!)_\Gamma}(\mathsf{guard}(b, \llbracket C_1 \rrbracket, \llbracket C_2 \rrbracket))^\dagger(\mu) = \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C_1 \rrbracket)^\dagger(\mu)$$

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 premise to conclude that there is a $\mathcal{Q}$ such that $\mathcal{Q} \otimes \mathcal{P}_F \preceq \nu$ and $\Gamma, \mathcal{Q} \vDash \psi$.

▷ If2. Symmetric to the previous case.

▷ Assign.

$$\frac{(\varphi * \lceil x \mapsto E \rceil) \Rightarrow \lceil e \mapsto E' \rceil}{I \vdash \langle \varphi * \lceil x \mapsto E \rceil \rangle\ x := e\ \langle \varphi * \lceil x \mapsto E' \rceil \rangle}\textsc{Assign}$$

Suppose that $\mathcal{P}_1 \otimes \mathcal{P}_2 \otimes \mathcal{P}_F \preceq \mu$ and $\Gamma, \mathcal{P}_1 \vDash \varphi$ and $\Gamma, \mathcal{P}_2 \vDash \lceil x \mapsto E \rceil$, where $\mu \in \mathcal{D}(\mathsf{Mem}[S])$. Since $(\varphi * \lceil x \mapsto E \rceil) \Rightarrow \lceil e \mapsto E' \rceil$, 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 that:

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

From the premise of the rule, we know that $x := e$ does not depend on the variables if I, so the invariant sensitive semantics is the same as the regular semantics.

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

Therefore, we know that $\nu = \sum_{\sigma \in \mathsf{supp}(\mu)} \mu(\sigma) \cdot \delta_{\sigma[x := \llbracket E' \rrbracket_{\mathsf{LExp}}(\Gamma)]}$. It is easy to see that $\pi_{S \setminus \{x\}}(\nu) = \pi_{S \setminus \{x\}}(\mu)$, therefore $\mathcal{P}_1 \otimes \mathcal{P}_F \preceq \pi_{S \setminus \{x\}}(\nu)$ and so $\Gamma, \mathcal{P}_1 \vDash \varphi$. Letting Q be the trivial probability space that satisfies $\lceil x \mapsto E' \rceil$, we also get that $\mathcal{P}_1 \otimes Q \otimes \mathcal{P}_F \preceq \nu$, therefore we are done.

▹ Samp.

$$\frac{(\varphi * \lceil x \mapsto E \rceil) \Rightarrow \lceil e \mapsto E' \rceil}{I \vdash \langle \varphi * \lceil x \mapsto E \rceil \rangle\ x :\approx d(e)\ \langle \varphi * (x \sim d(E')) \rangle}\ \textsc{Samp}$$

Suppose that $\mathcal{P}_1 \otimes \mathcal{P}_2 \otimes \mathcal{P}_F \preceq \mu$ and $\Gamma, \mathcal{P}_1 \vDash \varphi$ and $\Gamma, \mathcal{P}_2 \vDash \lceil x \mapsto E \rceil$, where $\mu \in \mathcal{D}(\mathsf{Mem}[S])$. Since $(\varphi * \lceil x \mapsto E \rceil) \Rightarrow \lceil e \mapsto E' \rceil$, 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 $\nu \in \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket x :\approx d(e) \rrbracket)^\dagger(\mu)$. We know that:

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

From the premise of the rule, we know that $x :\approx e$ does not depend on the variables if I, so the invariant sensitive semantics is the same as the regular semantics.

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

Let $\nu_v = \sum_{\sigma \in \mathsf{supp}(\mu)} \mu(\sigma) \cdot \delta_{\sigma[x := v]}$, therefore $\nu = \sum_{v \in \mathsf{Val}} d(\llbracket E' \rrbracket_{\mathsf{LExp}}(\Gamma))(v) \cdot \nu_v$. By similar logic to the soundness of the Assign rule, we know that $\mathcal{P}_1 \otimes Q_v \otimes \mathcal{P}_F \preceq \nu_v$ where Q_v is the trivial probability space satisfying $\lceil x \mapsto X \rceil$ in context $\Gamma[X := v]$. Let $\xi = d(\llbracket E' \rrbracket_{\mathsf{LExp}}(\Gamma))$, then using Lemma B.3 we get that $\bigoplus_{v\sim\xi} \mathcal{P}_1 \otimes \mathcal{Q}_v \otimes \mathcal{P}_F \preceq \nu$, which by Lemma B.1 is equal to $\mathcal{P}_1 \otimes \left(\bigoplus_{v\sim\xi} \mathcal{Q}_v\right) \otimes \mathcal{P}_F$, and clearly $\Gamma, \mathcal{P}_1 \otimes \left(\bigoplus_{v\sim\xi} \mathcal{Q}_v\right) \vDash \varphi * (x \sim d(E'))$.

▹ Atom.

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

Suppose that $\mathcal{P} \otimes \mathcal{P}_F \preceq \mu$ and $\Gamma, \mathcal{P} \vDash \varphi$. Let $\mathcal{P}_I$ be the trivial probability space such that $\Gamma, \mathcal{P}_I \vDash \lceil I \rceil$. Clearly $\Gamma, \mathcal{P} \otimes \mathcal{P}_I \vDash \varphi * \lceil I \rceil$. Let S be the variables in μ, $S_I = \mathsf{fv}(I)$ and $S_J = \mathsf{fv}(J)$.

$$\mathcal{L}^{(\!|I|\!)_\Gamma * (\!|J|\!)_\Gamma}(\llbracket a \rrbracket)^\dagger(\mu) = (\llbracket a \rrbracket_{\mathsf{Act}}^{(\!|I|\!)_\Gamma * (\!|J|\!)_\Gamma})^\dagger(\mu)$$

$$= \left\{ \sum_{\sigma\in\mathsf{supp}(\mu)} \mu(\sigma)\cdot\nu_\sigma \;\middle|\; \forall\sigma.\nu_\sigma \in \mathop{\&}_{\sigma'\in(\!|I|\!)_\Gamma * (\!|J|\!)_\Gamma} \begin{cases} \pi_S(\llbracket a \rrbracket_{\mathsf{Act}}(\sigma \uplus \sigma')) & \text{if } \pi_{S_I\cup S_J}(\llbracket a \rrbracket_{\mathsf{Act}}(\sigma \uplus \sigma')) \subseteq (\!|I|\!)_\Gamma * (\!|J|\!)_\Gamma \\ \eta(\bot) & \text{otherwise} \end{cases} \right\}$$

Now, take any $\nu \in \mathcal{L}^{(\!|I|\!)_\Gamma * (\!|J|\!)_\Gamma}(\llbracket a \rrbracket)^\dagger(\mu)$, which must have the form:

$$\nu = \sum_{\sigma\in\mathsf{supp}(\mu)} \mu(\sigma) \cdot \sum_{\sigma_I\in\mathsf{supp}(\mu_I)} \mu_I(\sigma_I) \cdot \sum_{\sigma_J\in\mathsf{supp}(\mu_J)} \mu_J(\sigma_J)\cdot \nu_{\sigma\uplus\sigma_I\uplus\sigma_J}$$

Where $\mu_I \in \mathcal{D}(\llbracket I \rrbracket_\bot^\dagger(\!)\Gamma)$, $\mu_J \in \mathcal{D}((\!|J|\!)_\Gamma)$, and $\nu_\tau \in \pi_S(\llbracket a \rrbracket_{\mathsf{Act}}(\tau))$ if $\pi_{S_I\cup S_J}(\llbracket a \rrbracket_{\mathsf{Act}}(\tau)) \subseteq (\!|I|\!)_\Gamma * (\!|J|\!)_\Gamma$ and $\delta_\bot$ otherwise. Now, let $\mu'(\tau) = \mu(\pi_S(\tau)) \cdot \mu_I(\pi_{S_I}(\tau))$, so clearly $\mathcal{P} \otimes \mathcal{P}_F \otimes \mathcal{P}_I \preceq \mu'$. Now, let:

$$\nu' = \sum_{\tau\in\mathsf{supp}(\mu')} \mu'(\tau)\cdot \sum_{\sigma_J\in\mathsf{supp}(\mu_J)} \mu_J(\sigma_J)\cdot\nu'_{\tau\uplus\sigma_J}$$

Where $\nu'_{\tau\uplus\sigma_J}$ is some distribution from $\pi_{S\cup S_I}(\llbracket a \rrbracket_{\mathsf{Act}}(\tau \uplus \sigma_J))$ such that $\pi_S(\nu'_{\tau\uplus\sigma_J}) = \nu_{\tau\uplus\sigma_J}$, which must exist. By construction, $\nu' \in \mathcal{L}^{(\!|J|\!)_\Gamma}(\llbracket a \rrbracket)^\dagger(\mu')$ and therefore, by the induction hypothesis, there is a $\mathcal{Q}$ such that $\mathcal{Q} \otimes \mathcal{P}_F \preceq \nu$ and $\Gamma, \mathcal{Q} \vDash \psi * \lceil I \rceil$, therefore there are $\mathcal{Q}' \otimes \mathcal{Q}_I \preceq \nu'$ such that $\Gamma, \mathcal{Q}' \vDash \psi$ and $\Gamma, \mathcal{Q}_I \vDash \lceil I \rceil$. Note that $\nu = \pi_S(\nu')$, and therefore $\mathcal{Q}' \preceq \nu$, so we are done.

▹ Share.

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

Suppose that $\mathcal{P} \otimes \mathcal{P}_I \otimes \mathcal{P}_F \preceq \mu$ and $\Gamma, \mathcal{P} \vDash \varphi$, and $\Gamma, \mathcal{P}_I \vDash \lceil I \rceil$. Now, take any $\nu \in \mathcal{L}^{(\!|J|\!)_\Gamma}(\llbracket C \rrbracket)^\dagger(\mu)$. By the invariant monotonicity property (Lemmas A.6 and A.7), we know that:

$$\pi_S(\mathcal{L}^{(\!|J|\!)_\Gamma}(\llbracket C \rrbracket)^\dagger(\mu)) \subseteq \mathcal{L}^{(\!|I|\!)_\Gamma * (\!|J|\!)_\Gamma}(\llbracket C \rrbracket)^\dagger(\pi_S(\mu))$$

Where above, S is the footprint of $\mathcal{P} \otimes \mathcal{P}_F$. So, $\pi_S(\nu) \in \mathcal{L}^{(\!|I|\!)_\Gamma * (\!|J|\!)_\Gamma}(\llbracket C \rrbracket)^\dagger(\pi_S(\mu))$. Note that $\mathcal{P} \otimes \mathcal{P}_S \preceq \pi_S(\mu)$, so by the premise of the rule, we know that there is a $\mathcal{Q}$ such that $\mathcal{Q} \otimes \mathcal{P}_F \preceq \pi_S(\nu)$ and $\Gamma, \mathcal{Q} \vDash \psi$. Let $\mathcal{Q}_I$ be the trivial probability space satisfying $\lceil I \rceil$, so clearly $\mathcal{Q} \otimes \mathcal{Q}_I \otimes \mathcal{P}_F \preceq \nu$, and $\Gamma, \mathcal{Q} \otimes \mathcal{Q}_I \vDash \psi * \lceil I \rceil$, so we are done.

▹ Cond1.

$$\frac{I \vdash \langle \varphi \rangle\ C\ \langle \psi \rangle \qquad \psi \Rightarrow \lceil e \mapsto X \rceil \qquad X \notin \mathsf{fv}(I)}{I \vdash \langle \bigoplus_{X\sim d(E)} \varphi \rangle\ C\ \langle \bigoplus_{X\sim d(E)} \psi \rangle}\textsc{Cond1}$$

Suppose that $\mathcal{P} \otimes \mathcal{P}_F \preceq \mu$ and $\Gamma, \mathcal{P} \vDash \bigoplus_{X\sim d(E)} \varphi$. This means that $\bigoplus_{v\sim\xi} \mathcal{P}_v \preceq \mathcal{P}$ such that $\Gamma[X := v], \mathsf{comp}(\mathcal{P}_v) \vDash \varphi$ for each v and $\xi = d(\llbracket E \rrbracket_{\mathsf{LExp}}(\Gamma))$. By Lemmas B.1 and B.2 we have:

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

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

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

So $\nu = \sum_{v\in\mathsf{supp}(\xi)} \xi(v) \cdot \nu_v$ where $\nu_v \in \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C \rrbracket)^\dagger(\mu_v)$ for each v. Note that $X \notin \mathsf{fv}(I)$, so $(\!|I|\!)_\Gamma = (\!|I|\!)_{\Gamma[X:=v]}$, 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 $\Gamma[X := v], \mathcal{Q}_v \vDash \psi$. Since $\psi \Rightarrow \lceil e = X \rceil$, we can restrict the $\mathcal{Q}_v$s to disjoint probability spaces $\mathcal{Q}'_v$ such that $\mathcal{Q}_v = \mathsf{comp}(\mathcal{Q}'_v)$ as follows:

$$\mathcal{Q}'_v \triangleq \langle \{\sigma \in \Omega_{\mathcal{Q}_v} \mid \llbracket e \rrbracket_{\mathsf{Exp}}(\sigma) = v\}, \{A \cap \Omega_{\mathcal{Q}'_v} \mid A \in \mathcal{F}_{\mathcal{Q}_v}\}, \mu_{\mathcal{Q}_v} \rangle$$

Therefore $\bigoplus_{v\sim\nu} \mathcal{Q}'_v$ exists and $\Gamma, \bigoplus_{v\sim\xi} \mathcal{Q}'_v \vDash \bigoplus_{v\sim d(E)} \psi$. Also, clearly $\mathcal{Q}'_v \preceq \mathcal{Q}_v$ since both agree on the probability of event with nonzero mass, and $\mathcal{Q}_v$ contains strictly more measurable events. Therefore by Lemma B.2, $\mathcal{Q}'_v \otimes \mathcal{P}_F \preceq \nu_v$. It is also easy to see that $\bigoplus_{v\sim\xi} \mathcal{Q}'_v \otimes \mathcal{P}_F \preceq \sum_{v\in\mathsf{supp}(\xi)} \xi(v) \cdot \nu_v = \nu$, and since $\bigoplus_{v\sim\xi}(\mathcal{Q}'_v \otimes \mathcal{P}_F) = \left(\bigoplus_{v\sim\xi} \mathcal{Q}'_v\right) \otimes \mathcal{P}_F$, then $\left(\bigoplus_{v\sim\xi} \mathcal{Q}'_v\right) \otimes \mathcal{P}_F \preceq \nu$.

▷ Cond2.

$$\frac{I \vdash \langle \varphi \rangle\ C\ \langle \psi \rangle \qquad \mathsf{precise}(\psi) \qquad X \notin \mathsf{fv}(I, \psi)}{I \vdash \langle \bigoplus_{X\sim d(E)} \varphi \rangle\ C\ \langle \psi \rangle}\ \textsc{Cond2}$$

Suppose that $\mathcal{P} \otimes \mathcal{P}_F \preceq \mu$ and $\Gamma, \mathcal{P} \vDash \bigoplus_{X\sim d(E)} \varphi$. This means that $\bigoplus_{v\sim\xi} \mathcal{P}_v \preceq \mathcal{P}$ such that $\Gamma[X := v], \mathsf{comp}(\mathcal{P}_v) \vDash \varphi$ for each v and $\xi = d(\llbracket E \rrbracket_{\mathsf{LExp}}(\Gamma))$. By Lemmas B.1 and B.2 we have:

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

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

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

So $\nu = \sum_{v\in\mathsf{supp}(\xi)} \xi(v) \cdot \nu_v$ where $\nu_v \in \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C \rrbracket)^\dagger(\mu_v)$ for each v. Note that $X \notin \mathsf{fv}(I)$, so $(\!|I|\!)_\Gamma = (\!|I|\!)_{\Gamma[X:=v]}$, 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 $\Gamma[X := v], \mathcal{Q}_v \vDash \psi$. Since $X \notin \mathsf{fv}(\psi)$, we can remove the update of X to conclude that $\Gamma, \mathcal{Q}_v \vDash \psi$, and since ψ is precise, we know that there is some $\mathcal{Q} \preceq \mathcal{Q}_v$ such that $\Gamma, \mathcal{Q} \vDash \psi$. It remains only to show that $\mathcal{Q} \otimes \mathcal{P}_F \preceq \nu$, which we do as follows. Taking any $A \in \mathcal{F}_{\mathcal{Q}}$ and $B \in \mathcal{F}_{\mathcal{P}_F}$, we get:

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

Since $\mathcal{Q} \preceq \mathcal{Q}_v$, and letting $\Omega_{\mathcal{Q}} = \mathsf{Mem}[S]$ and $\Omega_{\mathcal{P}_F} = \mathsf{Mem}[T]$:

$$= \sum_{v \in \mathsf{supp}(\xi)} \xi(v) \cdot \mu_{\mathcal{Q}_v}\left(\bigcup\{A' \in \mathcal{F}_{\mathcal{Q}_v} \mid \pi_S(A') = A\}\right) \cdot \mu_{\mathcal{P}_F}(B)$$

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

$$\begin{aligned}&= \sum_{v \in \mathsf{supp}(\xi)} \xi(v) \cdot \sum_{\sigma \mid \pi_S(\sigma) \in A} \sum_{\tau \mid \pi_T(\tau) \in B} \nu_v(\sigma \uplus \tau) \\ &= \sum_{\sigma \mid \pi_S(\sigma) \in A} \sum_{\tau \mid \pi_T(\tau) \in B} \sum_{v \in \mathsf{supp}(\xi)} \xi(v) \cdot \nu_v(\sigma \uplus \tau) \\ &= \sum_{\sigma \mid \pi_S(\sigma) \in A} \sum_{\tau \mid \pi_T(\tau) \in B} \nu(\sigma \uplus \tau)\end{aligned}$$

▹ Frame.

$$\frac{I \vdash \langle \varphi \rangle\ C\ \langle \psi \rangle}{I \vdash \langle \varphi * \vartheta \rangle\ C\ \langle \psi * \vartheta \rangle}$$

Suppose that $\mathcal{P} \otimes \mathcal{P}' \otimes \mathcal{P}_F \preceq \mu$ and $\Gamma, \mathcal{P} \vDash \varphi$ and $\Gamma, \mathcal{P}' \vDash \vartheta$. Now, take any $\nu \in \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C \rrbracket)^\dagger(\mu)$. let $\mathcal{P}'_F = \mathcal{P}' \otimes \mathcal{P}_F$, so since $\mathcal{P} \otimes \mathcal{P}'_F \preceq \mu$, by the induction hypothesis, there is a $\mathcal{Q}$ such that $\mathcal{Q} \otimes \mathcal{P}'_F \preceq \nu$ and $\Gamma, \mathcal{Q} \vDash \psi$, therefore $\mathcal{Q} \otimes \mathcal{P}' \vDash \psi * \vartheta$. Expanding $\mathcal{P}'_F$, we get that $\mathcal{Q} \otimes \mathcal{P}' \otimes \mathcal{P}_F \preceq \nu$, so we are done.

▹ Exists.

$$\frac{I \vdash \langle \lceil P \rceil \rangle\ C\ \langle \psi \rangle \qquad \mathsf{precise}(\psi) \qquad X \notin \mathsf{fv}(\psi, I)}{I \vdash \langle \lceil \exists X. P \rceil \rangle\ C\ \langle \psi \rangle}\textsc{Exists}$$

The proof follows from Lemma A.9, so we must first establish the premises. Take any μ_1 such that $\mathcal{P} \preceq \mu_1$. We know that $\Gamma, \sigma \vDash \exists X.\ P$ for each $\sigma \in \mathsf{supp}(\mu_1)$, which means that $\Gamma[X := v_\sigma], \sigma \vDash P$ for some $v_\sigma \in \mathsf{Val}$. Now, let $\xi \triangleq \sum_{\sigma \in \mathsf{supp}(\mu)} \mu(\sigma) \cdot \delta_{v_\sigma}$, so $\xi(v)$ is the probability that $\Gamma[X := v]$ is the context that satisfies P. Also, let:

$$\mu_v(\sigma) \triangleq \frac{1}{\xi(v)} \begin{cases} \mu_1(\sigma) & \text{if } v = v_\sigma \\ 0 & \text{if } v \neq v_\sigma \end{cases}$$

Now, let $\mathcal{P}_v$ be the unique smallest probability space that satisfies $\lceil P \rceil$ in context $\Gamma[X := v]$. By construction, $\Gamma[X := v], \mathcal{P}_v \vDash \lceil P \rceil$ and $(\!|I|\!)_\Gamma = (\!|I|\!)_{\Gamma[X:=v]}$ since $X \notin \mathsf{fv}(I)$, so by the premise of the Exists rule, for any $\nu \in \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C \rrbracket)^\dagger(\mu_v)$ there is a $\mathcal{Q}_v \preceq \nu$ such that $\Gamma[X := v], \mathcal{Q}_v \vDash \psi$. Since $X \notin \mathsf{fv}(\psi)$, this means that $\Gamma, \mathcal{Q}_v \vDash \psi$, and since ψ is precise, there is a unique smallest $\mathcal{Q}$ such that $\Gamma, \mathcal{Q} \vDash \psi$. Now, since by construction $\mu_1 = \sum_{v \in \mathsf{supp}(\xi)} \xi(v) \cdot \mu_v$, we have:

$$\mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C \rrbracket)^\dagger(\mu_1) = \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C \rrbracket)^\dagger\Big(\sum_{v \in \mathsf{supp}(\nu)} \xi(v) \cdot \mu_v\Big)$$

$$= \left\{ \sum_{v \in \mathrm{supp}(\nu)} \xi(v) \cdot \nu_v \;\middle|\; \forall v. \nu_v \in \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C \rrbracket)^\dagger(\mu_v) \right\}$$

So, any element of $\nu \in \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C \rrbracket)^\dagger(\mu_1)$ is a convex combination of $\nu = \sum_{v \in \mathrm{supp}(\xi)} \xi(v) \cdot \nu_v$ such that $\mathcal{Q} \preceq \nu_v$ and $\Gamma, \mathcal{Q} \vDash \psi$ for each v. Therefore, $\mathcal{Q} \preceq \nu$ since:

$$\mu_{\mathcal{Q}}(A) = \sum_{v \in \mathrm{supp}(\xi)} \xi(v) \cdot \mu_{\mathcal{Q}}(A) = \sum_{v \in \mathrm{supp}(\xi)} \xi(v) \cdot \sum_{\sigma \in A} \nu_v(\sigma) = \sum_{\sigma \in A} \sum_{v \in \mathrm{supp}(\xi)} \xi(v) \cdot \nu_v(\sigma) = \sum_{\sigma \in A} \nu(\sigma)$$

So, we have just shown that:

$$\forall \mu_1. \mathcal{P} \preceq \mu_1 \implies \forall \nu_1 \in \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C \rrbracket)^\dagger(\mu_1). \mathcal{Q} \preceq \nu_1$$

Since $\mathcal{L}^{(\!|I|\!)_\Gamma}(\emptyset_{\mathcal{P}\mathrm{om}})^\dagger(\mu_2) = \{\mu_2\}$, it is also obvious that:

$$\forall \mu_2. \mathcal{P}_F \preceq \mu_2 \implies \forall \nu_2 \in \mathcal{L}^{(\!|I|\!)_\Gamma}(\emptyset_{\mathcal{P}\mathrm{om}})^\dagger(\mu_2). \mathcal{P}_F \preceq \nu_2$$

Therefore, by Lemma A.9, we get that:

$$\forall \mu. \mathcal{P} \otimes \mathcal{P}_F \preceq \mu \implies \forall \nu \in \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C \rrbracket \parallel \emptyset_{\mathcal{P}\mathrm{om}})^\dagger(\mu). \mathcal{Q} \otimes \mathcal{P}_F \preceq \nu$$

Now, since $\llbracket C \rrbracket \parallel \emptyset_{\mathcal{P}\mathrm{om}} = \llbracket C \rrbracket$, and we already established that $\Gamma, \mathcal{Q} \vDash \psi$, we are done.

▹ Consequence.

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

Suppose that $\mathcal{P} \otimes \mathcal{P}_F \preceq \mu$ and $\Gamma, \mathcal{P} \vDash \varphi'$. By the premise, we know that $\Gamma, \mathcal{P} \vDash \varphi$. Now take any $\nu \in \mathcal{L}^{(\!|I|\!)_\Gamma}(\llbracket C \rrbracket)(\mu)$. By the induction hypothesis, there is a $\mathcal{Q}$ such that $\mathcal{Q} \otimes \mathcal{P}_F \preceq \nu$ and $\Gamma, \mathcal{Q} \vDash \psi$. By the second implication, $\Gamma, \mathcal{Q} \vDash \psi'$.

□

D Examples

In Figure 9 we give the full derivation for program (1), shown below, which was introduced in Section 2.

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

In this figure, we show the derivation for the program that was introduced in Section 2. The program is repeated below:

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

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 $(\bigstar)$ 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 $(\bigstar)$ derivation. The first step is to rearrange the precondition using the entailment laws from Figure 5 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.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\langle \mathsf{own}(x) * \lceil z \mapsto Z\rceil\rangle\ x \coloneqq z\ \langle \lceil x \mapsto Z\rceil * \lceil z \mapsto Z\rceil\rangle}\ \textsc{Assign}}{z \mapsto Z \vdash \langle \mathsf{own}(x)\rangle\ x \coloneqq z\ \langle \lceil x \mapsto Z\rceil\rangle}\ \textsc{Atom} \qquad \dfrac{\dfrac{}{\langle \mathsf{own}(y) * \lceil z \mapsto Z\rceil\rangle\ y \coloneqq 1 - z\ \langle \lceil y \mapsto 1 - Z\rceil * \lceil z \mapsto Z\rceil\rangle}\ \textsc{Assign}}{z \mapsto Z \vdash \langle \mathsf{own}(y)\rangle\ y \coloneqq 1 - z\ \langle \lceil y \mapsto 1 - Z\rceil\rangle}\ \textsc{Atom}}{z \mapsto Z \vdash \langle \mathsf{own}(x,y)\rangle\ x \coloneqq z \parallel y \coloneqq 1 - z\ \langle \lceil x \mapsto Z\rceil * \lceil y \mapsto 1 - Z\rceil\rangle}\ \textsc{Par}}{\langle \mathsf{own}(x,y) * \lceil z \mapsto Z\rceil\rangle\ x \coloneqq z \parallel y \coloneqq 1 - z\ \langle \lceil x \mapsto Z\rceil * \lceil y \mapsto 1 - Z\rceil * \lceil z \mapsto Z\rceil\rangle}\ \textsc{Share}}{\langle \bigoplus_{\mathbf{Ber}(1/2)} \mathsf{own}(x,y) * \lceil z \mapsto Z\rceil\rangle\ x \coloneqq z \parallel y \coloneqq 1 - z\ \langle \bigoplus_{\mathbf{Ber}(1/2)} \lceil x \mapsto Z\rceil * \lceil y \mapsto 1 - Z\rceil * \lceil z \mapsto Z\rceil\rangle}\ \textsc{Cond1}}{\langle \mathsf{own}(x,y) * z \sim \mathbf{Ber}\,(1/2)\rangle\ x \coloneqq z \parallel y \coloneqq 1 - z\ \langle \bigoplus_{\mathbf{Ber}(1/2)} \lceil x \mapsto Z\rceil * \lceil y \mapsto 1 - Z\rceil * \lceil z \mapsto Z\rceil\rangle}\ \textsc{Consequence}$$

Fig. 9. Derivation of the program from Section 2.