PRISM: Probabilistic Reasoning Invoking Separation Methods

ANONYMOUS AUTHOR(S)

Probabilistic reasoning presents a challenge due to the 'many-value nature' of random variables and their interdependent probabilities. Accordingly, existing probabilistic reasoning frameworks often lack scalability. To enable local reasoning, Barthe *et al.* [5] extended separation logic to probabilistic programs. However, despite successfully verifying cryptographic algorithms that rely on *independent* probabilities, their framework falls short when it comes to reasoning about randomized algorithms that rely on *dependent* probabilities. Moreover, their logic cannot be used to reason about probabilistic termination.

PRISM is a variation of separation logic that can reason about both independent and dependent probabilities. Local reasoning leverages two separating conjunctions and techniques. Independent separation $\boxtimes$ connects distributions of different variables and enjoys a frame rule closely related to that in traditional separation logic [32]. Dependent separation $\boxplus$ connects distributions of the same variables and achieves locality with a rule closely related to the parallel rule from concurrent separation logic [33]. In addition to supporting local reasoning, PRISM can also verify probabilistic termination, including both almost-sure and partial cases.

To prove the soundness of PRISM, we extend separation algebras to "multiverses," in which parallel "worlds" coexist with different probabilistic "weights." We also introduce a novel limiting semantics for Hoare triples so that they can reason about probabilistic termination. Lastly, we assess the effectiveness of our framework by applying it to verify the correctness of cryptographic algorithms as well as landmark algorithms that utilize probabilistic termination, such as von Neumann's simulation of a fair coin using a biased one [43].

ACM Reference Format:
Anonymous Author(s). 2018. PRISM: Probabilistic Reasoning Invoking Separation Methods. *Proc. ACM Program. Lang.* 1, CONF, Article 1 (January 2018), 48 pages.

1 Introduction

In recent years, there has been significant interest in developing probabilistic reasoning techniques, especially to support the twin fields of artificial intelligence [25, 37, 44, 45] and security protocols [5, 18, 22, 29]. Reasoning about probabilistic distributions poses challenges due to the exponential growth of the sample space and the complexity of relationships among dependent random variables. Previous techniques focused either on proving almost-sure termination (e.g. [1, 7, 12, 15, 28]) or on supporting local reasoning (e.g. [5, 6, 41]). However, to the best of our knowledge, none of these techniques provide a comprehensive solution that can handle both tasks smoothly.

To address this gap we propose PRISM, a variant of separation logic that can reason about both independent and dependent probability distributions. PRISM supports local reasoning, even for programs whose termination depends on a probabilistic process (*e.g.*, almost-sure termination). PRISM can verify security protocols for one-time pads, private information retrievals, oblivious transfers, and multi-party computation. PRISM can also verify the termination probabilities for probabilistic processes. The resulting distributions can be countably infinite, and we can even verify situations in which the termination probability changes dynamically with each iteration.

Probabilistic distributions and local reasoning. The probabilistic "maps-to" predicate, PRISM's analogue of standard separation logic's "points-to" predicate, connects program variables with probabilistic expressions in angle brackets. For example, the predicate $\langle x = (0.5 : 0) \oplus (0.5 : 1)\rangle$ expresses the idea that variable x has a 50/50 probability of being 0 or 1, and the predicate $\langle y = (0.5 : 0) \oplus (0.5 : 1)\rangle$ does the same for y; call these predicates P and Q, respectively[1].

[1]When the probability is 100% we omit the "1 :" for readability; thus $\langle x = 1\rangle$ is technically $\langle x = 1 : 1\rangle$, *i.e.* just $x = 1$.

2018. ACM 2475-1421/2018/1-ART1
https://doi.org/

To facilitate local reasoning about probabilistic distributions, PRISM introduces two logical connectives: the independent $\boxtimes$ and dependent $\boxplus$ separating conjunctions. The predicate $S \boxtimes T$ indicates that the variables in S and T are disjoint and *probabilistically independent*, and that their distributions satisfy S and T, respectively. The P and Q defined just a moment ago indeed do define independent distributions on variables—from an information-theoretic point of view, knowledge of x provides no knowledge of y, and vice versa—and so we can write $P \boxtimes Q$ to express this disjointness. Just as in vanilla separation logic, disjointness yields a frame rule; we also enjoy a weaker frame rule in the non probabilistically-disjoint (*i.e.* dependent) case:

$$\frac{\{P\}\ c\ \{Q\} \quad \mathsf{fv}(F) \cap \mathsf{vars}(c) = \emptyset}{\{P \boxtimes F\}\ c\ \{Q \boxtimes F\}}\ [\textsc{IFrame}] \qquad \frac{\{P\}\ c\ \{Q\} \quad \mathsf{fv}(F) \cap \mathsf{mv}(c) = \emptyset \quad Q \vdash 1}{\{P \wedge F\}\ c\ \{Q \wedge F\}}\ [\textsc{DFrame}]$$

IFrame and DFrame enable local reasoning on independent $\boxtimes$ and dependent $\wedge$ distributions, respectively. Both rules look similar to the classical separation logic frame rule for heap-manipulating programs [32]; beyond the connective used to adjoin the frame, the difference between them is in the side conditions. Just as in classical separation logic, these side conditions eliminate certain pathological frame/command combinations; they are a little subtle and so are discussed in §3.2.

To express the notion of *dependent* probabilities, PRISM's $\langle \cdots \rangle$ predicate can specify multiple variables simultaneously, rather than merely the single-variable version we used to build our P and Q above. To define a predicate R in which x and y's probabilistic values depend on each other, consider $R \triangleq \langle x, y = (0.5 : 1, 0) \oplus (0.5 : 0, 1) \rangle$. This R means that the variables x and y can be in either of two states, both with 50% probability: either x is 0 and y is 1, or x is 1 and y is 0.

To break the monolithic predicate R into smaller pieces we use two key ideas. First, the predicate multiplication operator $\rho \cdot P$ adjusts a predicate's probabilistic norm; thus $0.5 \cdot \langle x, y = 1, 0 \rangle$ means that x and y are 1 and 0 with 50% probability[2]. Second, our other separating connective $\boxplus$ combines predicates that cover disjoint parts of the probability space together. Accordingly, R is equivalent to $(0.5 \cdot \langle x, y = 1, 0 \rangle) \boxplus (0.5 \cdot \langle x, y = 0, 1 \rangle)$. These operators yield simple rules for local reasoning:

$$\frac{\{P\}\ c\ \{Q\}}{\{\rho \cdot P\}\ c\ \{\rho \cdot Q\}}\ [\textsc{Dot}] \qquad \frac{\{P_1\}\ c\ \{Q_1\} \quad \{P_2\}\ c\ \{Q_2\}}{\{P_1 \boxplus P_2\}\ c\ \{Q_1 \boxplus Q_2\}}\ [\textsc{Split}] \qquad \frac{\forall i \in I.\ \{P_i\}\ c\ \{Q_i\}}{\{\boxplus_{i \in I}\ P_i\}\ c\ \{\boxplus_{i \in I}\ Q_i\}}\ [\textsc{iSplit}]$$

The Dot rule says that a specification in a probabilistic distribution carries over to a sub-distribution. The Split rule is reminiscent of the parallel rule from concurrent separation logic [33]: essentially, running a command c on a probabilistic distribution is analogous to running c in parallel with itself on each part of the distribution. Since our distributions can be countably infinite, we also provide an indexed version ISplit that utilizes a countable index set I.

Probabilistic termination. PRISM's Hoare triple verifies probabilistic total correctness using an underlying limiting semantics. That is, by $\{P\}\ c\ \{Q\}$ we mean that if we start from a state satisfying P and execute c, then—with overwhelming probability—we will eventually reach a state satisfying Q. The ratio between the probabilistic norms of P and Q gives the probability of termination. Consider this pair of programs and their respective specifications:

```
1 {⟨x = (0.5 : 0) ⊕ (0.5 : 1)⟩}
2 while b do
3     x := (0.5 : 0) ⊕ (0.5 : 1)
4 {⟨x = 1⟩}
```

```
1 {⟨x = (0.5 : 0) ⊕ (0.5 : 1)⟩}
2 while b do
3     skip
4 {0.5 · ⟨x = 1⟩}
```

Both programs start with a uniform distribution on x between 0 and 1. In the program on the left, we repeatedly flip a fair coin (line 3) until it comes up 1/heads. The postcondition (line 4) means that the program will terminate (with probability 1, *i.e.* almost surely), and afterwards y will be 1.

[2]We require that the norm inside the $\langle \cdots \rangle$ predicate be 1; thus, the attempted predicate $\langle x, y = 0.5 : 1, 0 \rangle$ is ill formed.

In the program on the right, we simply enter an infinite loop if x is 0. The postcondition (line 4) also means that x will be 1, but in this case the termination probability is only 50%.

The traditional Hoare rule for **while** loops for total correctness looks something like this [16]:

$$\frac{\{I \wedge b \wedge \|t = n\|\}\ c\ \{I \wedge \|t < n\|\}}{\{I\}\ \textbf{while}\ b\ \textbf{do}\ c\ \{I \wedge \neg b\}}$$

Here I is the invariant and the termination variant t must decrease over each loop body. Extending this pattern to our probabilistic context is challenging. To verify the left-hand program, we would like to use $\langle x = (0.5 : 0) \oplus (0.5 : 1)\rangle$ for I; in some sense this is the only choice. Moreover, the program text forces $x = 0$ for b. Unfortunately, $I \wedge b$ makes no sense: it asserts that half of the time x is 1, and yet x is also 0! Actually the problem is deeper since the predicate is ill-formed. In traditional Hoare logics, predicates are sets of states. In PRISM, predicates are sets of probabilistic distributions, each of which is a probability-weighted set of sets of states. Loosely speaking, traditional Hoare predicates are first-order whereas PRISM predicates are third-order.

Part of why $I \wedge b$ makes no sense is that b is neither 0 nor 1, but in some informal sense both simultaneously: more precisely, b is 0 on part of the distribution, and 1 on another part. To express this idea formally, we start with the notion of *world* (or *base*) predicates. A world predicate is just a set of states, just like predicates in traditional Hoare logic. We can lift such a base predicate to a PRISM predicate by writing $[P]$, which asserts that P is true on every state within the distribution. Sometimes we write $\langle P\rangle$, which further states that the distribution has norm 1. Now we can express something true about b by writing $\rho \cdot \langle b\rangle \boxplus \overline{\rho} \cdot \langle \neg b\rangle$. This means that b is 1 with probability ρ, and 0 with probability $1 - \rho$, which we abbreviate to $\overline{\rho}$. In the case of our left-hand example, $\rho = \overline{\rho} = 0.5$.

Unfortunately, although $I \wedge \langle b\rangle$ is well-formed, it is still not satisfiable: when $x = 1$ then b is not true. We address the problem by introducing the notions of filter predicates and renormalization predicates. The predicate $P/[Q]$ removes any part of the distributions in P that do not satisfy the world predicate Q. The predicate $P_{\downarrow\rho}$ renormalizes the probability distribution of P to have combined probability ρ. Often we use them together, writing $P/\langle Q\rangle$ for $(P/[Q])_{\downarrow 1}$, *i.e.* filter P by Q and then renormalize to 1. For the specific I and b considered above, $I/\langle b\rangle$ is just $\langle x = 0\rangle$ and $I/\langle \neg b\rangle$ is just $\langle x = 1\rangle$. Given this understanding, the following simplified version of our While rule can be used to verify the left-hand program above:

$$\frac{\{I/\langle b\rangle\}\ c\ \{I \wedge (\rho \cdot \langle b\rangle \boxplus \overline{\rho} \cdot \langle \neg b\rangle)\} \qquad \rho < 1 \qquad I \vdash 1 \qquad \mathrm{convex}(I/\langle \neg b\rangle)}{\{I\}\ \textbf{while}\ b\ \textbf{do}\ c\ \{I/\langle \neg b\rangle\}}\ [\textsc{SWhile}]$$

The premises can be understood as follows. Verifying a loop body should begin from the explained precondition $I/\langle b\rangle$ and reach a postcondition of I. The loop guard b should continue to hold with probability ρ, and fail with probability of $\overline{\rho}$. We require that ρ be strictly less than 1, so with overwhelming probability the loop will eventually terminate. We also require two technicalities: that the total probability in I is 1, and that the ultimate postcondition $I/\langle \neg b\rangle$ be *convex*. Convexity is vaguely analogous to the traditional separation logic property of precision; roughly speaking it means that $(\rho \cdot I) \boxplus (\overline{\rho} \cdot I) \vdash I$. These technicalities both hold for the I considered above for the left-hand program. Thus, for that program, we must verify the loop body (line 3) from precondition $\langle x = 0\rangle$ to postcondition $\langle x = (0.5 : 0) \oplus (0.5 : 1)\rangle \wedge (0.5 \cdot \langle x = 0\rangle \oplus 0.5 \cdot \langle \neg(x = 0)\rangle)$; the right side of the conjunction is implied by the left, so in practice we must verify:

$$\{\langle x = 0\rangle\}\ x := (0.5 : 0) \oplus (0.5 : 1)\ \{\langle x = (0.5 : 0) \oplus (0.5 : 1)\rangle\}$$

Given this, the SWhile rule proves the specification we gave for the left-hand program above. Verifying the right-hand program above requires the more powerful and general While rule we present in §3. Intuitively it is trickier since the infinite loop "eats" part of the probability distribution.

Semantics of multiverses and probabilistic Hoare tuples. We introduce a *multiverse* model to model our assertion language, following in the footsteps of Kriple resource models [36]. Each multiverse consists of worlds associated with a positive weight, with a combined weight no more than 1. We interpret discrete probabilistic distributions using the multiverse framework: worlds and their weights represent members of the sample space and their respective probabilities.

To capture the quantitative aspects of a multiverse, we use the *norm predicate* ρ. To capture the qualitative aspects of a multiverse, we use *base/world predicate* $[P]$, representing the universal properties that all worlds in the multiverse possess in common. The two basic predicates, along with the connectives we have seen so far $\{\boxtimes, \boxplus, \cdot, P/[Q], P_{\downarrow\rho}\}$, let us express and prove many desirable properties about probabilistic programs and their distributions.

Our semantics for probabilistic Hoare triples is also novel, using suprema to express the notion of a limiting postcondition. Consider again the left-hand program presented earlier. It is not guaranteed to terminate, but—given enough time—the probability that $\langle x = 1\rangle$ gets arbitrarily close to 100%. Our definition for the Hoare triple formalizes this idea, and we use it to prove our inference rules.

Contributions and organization. Our paper is organized as follows:

- §2 We introduce our probabilistic programming language and assertion language, and then demonstrate our framework with two simple examples.
- §3 **Contribution 1**: We discuss the key Hoare rules we use to verify probabilistic programs and present a set of entailment rules for predicate transformation.
- §4 **Contribution 2**: We present the multiverse model, show how it models the assertion language, and use it to build our interpretation of probability.
- §5 **Contribution 3**: We define the step semantics of our language and define Hoare triples using a limiting semantics to justify the correctness of our program logic.
- §6,§7 **Contribution 4**: We apply our logic to verify examples drawn from both privacy protocols (interpreted as probabilistic independence) and subtle uses of probabilistic termination.
- §8,§9 We discuss related work and conclude.

2 Programming Language, Assertion Language, and Two Examples

We introduce our probabilistic programming language in §2.1 and our assertion language in §2.2. We then use PRISM to verify probabilistic termination in §2.3 and a one-time pad in §2.4.

2.1 Probabilistic Programming Language

Our probabilistic programs take the following syntax:

$$c \;::=\; \textbf{skip} \mid x := e \mid c;c \mid \textbf{while } b \textbf{ do } c \mid c[e]c$$

where x, e, b represent program variables, arithmetic expressions $(+, -, \times, \ldots)$, and Boolean expressions $(e_1 = e_2, \&, |, \ldots)$. Unlike [5], our language does not differentiate between random variables and deterministic variables. To keep things as simple as possible, our language only modifies program variables; there is no heap/memory. Arithmetic evaluates to $\mathbb{Q}$ (or floating-point); Booleans are treated as a subtype of arithmetic and evaluate to $\{0, 1\}$; both evaluations are deterministic. Commands in our language are entirely standard and deterministic, with one major exception. The statement $c_1[e]c_2$ introduces probabilistic choice: after evaluating e to a value $p \in [0, 1]$, c_1 and c_2 are executed with probabilities p and $\overline{p}$ respectively. Traditional conditional statements are macros:

$$\textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2 \;\triangleq\; c_1[b]c_2 \qquad\qquad \textbf{if } b \textbf{ then } c \;\triangleq\; c[b]\textbf{skip}$$

Another macro is the *distribution assignment* $x := \vec{v}$, *e.g.* $x := (0.5 : 0) \oplus (0.5 : 1)$, which assigns the Bernoulli distribution of a fair coin to x. More precisely, given $\rho_1, \ldots, \rho_n$, with $\Sigma_{i=1}^{n}\rho_i = 1$; and e_1,

197 $\ldots, e_n$, then the *distribution vector* $\vec{v} \triangleq (\rho_1 : e_1) \oplus \ldots \oplus (\rho_n : e_n)$ captures the distribution where e_i
198 occurs with probability ρ_i, and the distribution assignment is defined as

199
200 $$x := \vec{v} \triangleq c_n \quad \text{where} \quad c_1 \triangleq x := e_n, c_{i+1} \triangleq (x := e_{n-i})[\rho_{n-i}/(\rho_{n-i}+\ldots+\rho_n)]c_i \text{ for every } 1 \le i < n$$

201 Lastly, simple looping programs are more naturally expressed with **for** loops, defined as follows:

202
203 $$\textbf{for } i := [n_1, n_2] \textbf{ do } c \triangleq i := n_1; \textbf{while } i \le n_2 \textbf{ do } \{c; i := i + 1\}$$

204 In §5, we will give this programming language a formal operational semantics.

205
206 ### 2.2 PRISM's Assertion Language

207 To verify programs, we provide the following assertion language:

208
209 $$\begin{array}{lll} W, W' & ::= & P(e_1, \ldots, e_n) \text{ where } P \in \mathcal{AP} \mid W \wedge W' \mid W \vee W' \mid \neg W \mid \exists v.\ W \mid \forall v.\ W \\ \phi, \psi & ::= & [W] \mid \rho \mid \phi \boxtimes \psi \mid \phi \boxplus \psi \mid \boxplus_{i \in I} \phi_i \mid \phi/[W] \mid \phi_{\downarrow \rho} \mid \bot \mid \top \mid \phi \wedge \psi \mid \phi \vee \psi \mid \exists x.\phi \mid \forall x.\ \phi \end{array}$$
210

211 *World* (or *base*) predicates W judge individual program states. They include a set of atomic predicates
212 $\mathcal{AP} \triangleq \{=, <, \ldots\}$ such that basic properties about program expressions e_i can be expressed, standard
213 Boolean connectives, and quantifiers. PRISM (or *probabilistic*) predicates ϕ judge probabilistic
214 distributions over states. The *lifting predicate* $[W]$ states that W is true on every state within
215 the distribution. The *norm predicate* ρ asserts that the distribution has norm (total probabilistic
216 weight) of ρ; two common examples are 0 (empty distribution) and 1 (complete distribution). The
217 *independent* (or *vertical*) *separating conjunction* $\phi \boxtimes \psi$ asserts that ϕ and ψ are probabilistically
218 independent. The *dependent* (or *horizontal*) *separating conjunction* $\phi \boxplus \psi$ means that the probability
219 distribution can be divided into two sub-distributions satisfuing ϕ and ψ, respectively. We provide
220 an indexed version of this connective $\boxplus_{i \in I} \phi_i$ to express countably infinite distributions. The *filter*
221 *predicate* $\phi/[W]$ removes all parts of the distributions accepted by ϕ that do not satisfy W. The
222 *normalization predicate* $\phi_{\downarrow \rho}$ renormalizes the distribution to have norm ρ. Lastly, we provide some
223 Boolean connectives and quantifiers. Although our underlying models have them, we omit the
224 implication connectives $\{-\!\!\boxplus, -\!\!\boxtimes, \rightarrow\}$ to keep things simpler. In §3 we provide entailment rules for
225 PRISM's novel connectives (*cf.* Figure 3), and in §4 we provide their formal semantics.

226 We use a few common patterns enough to justify predicate macros. The *fraction* predicate $\rho \cdot \phi$
227 is 0 (empty distribution) when $\rho = 0$, and $\rho \boxtimes \phi$ otherwise. The *world norm* predicate $\langle W \rangle$ is just
228 $[W] \wedge 1$, where 1 asserts the complete distribution. The *filter norm* predicate $\phi/\langle W \rangle$ is just the
229 combination of filtering W and then renormalizing to 1, *i.e.* $(\phi/[W])_{\downarrow 1}$. Lastly, the *distribution*
230 *predicate* $\langle \hat{x} = \vec{v} \rangle$, where $\hat{x}$ is an n-length list of variables and $\vec{v}$ is an n-length distribution vector
231 $(\rho_1 : \hat{v}_1) \oplus \ldots \oplus (\rho_n : \hat{v}_n)$, is just $\rho_1 \cdot \langle \hat{x} = \hat{v}_1 \rangle \boxplus \ldots \boxplus \rho_n \cdot \langle \hat{x} = \hat{v}_n \rangle$.

232
233 ### 2.3 Example of proving almost-sure termination

234 Consider the probabilistic program in Fig. 1 that tosses a fair coin once, remembering its outcome
235 as x, and then repeatedly tossing the coin until seeing an outcome y that satisfies $y \neq x$. We prove
236 the postcondition $\langle x, y = (0.5 : 0, 1) \oplus (0.5 : 1, 0) \rangle$ which also implies almost-sure termination. Our
237 decorated programs use $\{\searrow, \swarrow\}$ and $\{\searrow, \not\swarrow\}$ to indication the application of the IFrame and
238 DFrame rules from §1, respectively; and $\|$ when applying the Split or ISplit rules. We compress
239 the verification of simple commands into a single line as pre-command-post (*e.g.* lines 1 and 2).

240 We start with the precondition 1, i.e. the initial distribution has norm 1. Upon obtaining the
241 distribution of x, we apply the IFrame rule to infer the distribution of y (line 2). Subsequently, we
242 distribute x over y to establish their joint distribution (line 4) and use it as the loop invariant I after
243 reorganizing the terms based on the loop condition (line 5). By employing the SWhile rule, we enter
244 the loop body with the precondition $I/\langle x = y \rangle$, which is equivalent to $\langle x, y = (0.5 : 0) \oplus (0.5 : 1) \rangle$.
245

1 $\{1\}x := (0.5 : 0) \oplus (0.5 : 1); \{\langle x = (0.5 : 0) \oplus (0.5 : 1)\rangle\}$
2 $\searrow\swarrow \{1\}y := (0.5 : 0) \oplus (0.5 : 1); \{\langle y = (0.5 : 0) \oplus (0.5 : 1)\rangle\}$
3 $\{\langle x = (0.5 : 0) \oplus (0.5 : 1)\rangle \boxtimes \langle y = (0.5 : 0) \oplus (0.5 : 1)\rangle\}$
4 $\{\langle x, y = (0.25 : 0, 0) \oplus (0.25 : 0, 1) \oplus (0.25 : 1, 0) \oplus (0.25 : 1, 1)\rangle\}$
5 $\{0.5 \cdot \langle x, y = (0.5 : 0, 0) \oplus (0.5 : 1, 1)\rangle \boxplus 0.5 \cdot \langle x, y = (0.5 : 0, 1) \oplus (0.5 : 1, 0)\rangle\}$ // I
6 **while** $x = y$ **do**
7 $\{\langle x, y = (0.5 : 0, 0) \oplus (0.5 : 1, 1)\rangle\}$ // $I/\langle x = y\rangle$
8 $\{0.5 \cdot \langle x, y = 0, 0\rangle \boxplus 0.5 \cdot \langle x, y = 1, 1\rangle\}$

9 $\{\langle x, y = 0, 0\rangle\}$	14 $\{\langle x, y = 1, 1\rangle\}$
10 $\searrow\{\langle y = 0\rangle\}$	15 $\searrow\{\langle y = 1\rangle\}$
11 $y := (0.5 : 0) \oplus (0.5 : 1);$	16 $y := (0.5 : 0) \oplus (0.5 : 1);$
12 $\swarrow\{\langle y = (0.5 : 0) \oplus (0.5 : 1)\rangle\}$	17 $\swarrow\{\langle y = (0.5 : 0) \oplus (0.5 : 1)\rangle\}$
13 $\{\langle x, y = (0.5 : 0, 0) \oplus (0.5 : 0, 1)\rangle\}$	18 $\{\langle x, y = (0.5 : 1, 0) \oplus (0.5 : 1, 1)\rangle\}$

19 $\{I \wedge (0.5 \cdot \langle x = y\rangle \boxplus 0.5 \cdot \langle \neg(x = y)\rangle)\}$
20 $\{\langle x, y = (0.5 : 0, 1) \oplus (0.5 : 1, 0)\rangle\}$ // $I/\langle \neg(x = y)\rangle$

Fig. 1. An example of proving almost-sure termination.

By utilizing the Split and Dot rule, we are able to derive $I \wedge (0.5 \cdot \langle x = y\rangle \boxplus 0.5 \cdot \langle \neg(x = y)\rangle)$ at the end of the loop body (line 19). Note that $I/\langle \neg(x = y)\rangle$ is equivalent to the convex predicate $\langle x, y = (0.5 : 0, 1) \oplus (0.5 : 1, 0)\rangle$. Consequently, we exit the loop with the desired postcondition.

2.4 Example of proving probabilistic independence

Alice wants to send a private message m to Bob. A One-Time Pad uses a shared key k drawn from a uniform distribution. Alice encrypts her message as $c := m \operatorname{xor} k$ and Bob decrypts it by computing $m' := c \operatorname{xor} k$. Here we assume that m, k, m' are bit arrays of length $n > 0$ and the operator xor is bitwise. The protocol is considered safe if knowledge of c alone is not sufficient to decode m. Formally, one needs to show that the distributions of m and c are independent, or equivalently that the distribution of c remains unchanged regardless of the content of m, *i.e.* $\mathbb{P}(c) = \mathbb{P}(c|m)$.

The program in Fig. 2 represents the OTP protocol when $n = 2$; the general case is analogous and is in Appendix G. The precondition asserts fixed values for m_1 and m_2 as a_1 and a_2. The vector $\vec{u_k}$ represents the uniform distribution $(1/k : 0) \oplus \ldots \oplus (1/k : k-1)$. By utilizing the IFrame rule, we can deduce the distribution of each bit in k (lines 2-3). Subsequently, the IFrame and Split rule are applied to analyze the assignment $c_1 := m_1 \operatorname{xor} k_1$ (lines 4-12). It is then established that the distribution of c_1 is uniform (line 13) and $\langle c_1 = m_1 \operatorname{xor} k_1\rangle$. Similarly, the distribution of c_2 is also uniform and $\langle c_2 = m_2 \operatorname{xor} k_2\rangle$ (line 16). Consequently, by distributing c_1 over c_2, it can be inferred that their joint distribution is uniform and hence not dependent on the initial values of m_1, m_2.

Finally, we need to show that $\langle m_1' = m_1\rangle \wedge \langle m_2' = m_2\rangle$, i.e. Bob successfully decodes the message. By applying the DFrame rule and utilizing the identity $b \operatorname{xor} b = 0$, we can conclude that $\langle m_1' = m_1\rangle$ (line 19). Likewise, we can establish that $\langle m_2' = m_2\rangle$ (line 20).

3 Inference Framework

We presented our assertion language in §2.2. In §3.1 we present a selection of useful entailment rules to reason about the key predicates in PRISM $\{\boxtimes, \boxplus, \rho \cdot \phi, [W], \phi/[W], \phi_{\downarrow\rho}\}$. Turning to our Hoare rules, many of the key ones were introduced in §1: IFrame and DFrame, to enable local reasoning for independent probabilities; Dot, Split, and ISplit, to enable local reasoning for dependent

1 $\{\langle m_1 = a_1\rangle \boxtimes \langle m_2 = a_2\rangle\}$
2 $\swarrow\!\searrow \{1\} k_1 := \vec{u_2}; \{\langle k_1 = \vec{u_2}\rangle\}$
3 $\searrow\!\swarrow \{1\} k_2 := \vec{u_2}; \{\langle k_2 = \vec{u_2}\rangle\}$
4 $\searrow \{\langle m_1 = a_1\rangle \boxtimes \langle k_1 = \vec{u_2}\rangle\}$
5 $\{0.5 \cdot \langle k_1, m_1 = a_1, 0\rangle \boxplus 0.5 \cdot \langle k_1, m_1 = 1, a_1\rangle\}$

6 $\{\langle m_1, k_1 = a_1, 0\rangle\}$	9 $\{\langle m_1, k_1 = a_1, 1\rangle\}$
7 $c_1 := m_1 \text{ xor } k_1;$	10 $c_1 := m_1 \text{ xor } k_1;$
8 $\{\langle c_1, m_1, k_1 = a_1, a_1, 0\rangle\}$	11 $\{\langle c_1, m_1, k_1 = \overline{a_1}, a_1, 1\rangle\}$

12 $\{0.5 \cdot \langle c_1, m_1, k_1 = a_1, a_1, 0\rangle \boxplus 0.5 \cdot \langle c_1, m_1, k_1 = \overline{a_1}, a_1, 1\rangle\}$
13 $\swarrow \{\langle c_1 = \vec{u_2}\rangle \wedge \langle k_1 = \vec{u_2}\rangle \wedge \langle c_1 = m_1 \text{ xor } k_1\rangle\}$
14 $\searrow \{\langle m_2 = a_2\rangle \boxtimes \langle k_2 = \vec{u_2}\rangle\}$
15 $c_2 := m_2 \text{ xor } k_2;$
16 $\swarrow \{\langle c_2 = \vec{u_2}\rangle \wedge \langle k_2 = \vec{u_2}\rangle \wedge \langle c_2 = m_2 \text{ xor } k_2\rangle\}$
17 $\{(\langle c_1 = \vec{u_2}\rangle \boxtimes \langle c_2 = \vec{u_2}\rangle) \wedge \langle c_1 = m_1 \text{ xor } k_1\rangle \wedge \langle c_2 = m_2 \text{ xor } k_2\rangle\}$
18 $\{(\langle c_1, c_2 = (0.25 : 0, 0) \oplus (0.25 : 0, 1) \oplus (0.25 : 1, 0) \oplus (0.25 : 1, 1)\rangle \wedge \ldots\}$
19 $\searrow\!\swarrow \{\langle c_1 = m_1 \text{ xor } k_1\rangle\} m_1' := c_1 \text{ xor } k_1; \{\langle m_1' = m_1\rangle\}$
20 $\searrow\!\swarrow \{\langle c_2 = m_2 \text{ xor } k_2\rangle\} m_2' := c_2 \text{ xor } k_2; \{\langle m_2' = m_2\rangle\}$
21 $\{\langle c_1, c_2 = (0.25 : 0, 0) \oplus (0.25 : 0, 1) \oplus (0.25 : 1, 0) \oplus (0.25 : 1, 1)\rangle \wedge \langle m_1' = m_1\rangle \wedge \langle m_2' = m_2\rangle\}$

Fig. 2. One-Time Pad example.

probabilities; and SWhile, to verify relatively simple kinds of almost-sure termination. In §3.2 revisit the two frame rules to examine their side conditions more carefully. In §3.3, we explain the other rules of our program logic, except for the relatively complex While rule, which is explained in §3.4. For reference, Figure 13 in Appendix E gathers all of the Hoare rules of PRISM in one place.

3.1 Entailment reasoning

In Fig. 3, we provide the more interesting entailment rules to reason about the predicates in our logic. Here ϕ, ψ, θ are probabilistic predicates; P, Q are world/base predicates; and $0 \leq \alpha, \beta, \rho \leq 1$. A more complete set of rules and their soundness proofs can be found in Appendix C (*cf.* Figure 12).

Many of the entailments are expected: $\boxtimes$ and $\boxplus$ are associative ($\boxtimes^A, \boxplus^A$), commutative ($\boxtimes^C, \boxplus^C$), and distribute over disjunctions ($\boxtimes^\vee, \boxplus^\vee$). Distributing over conjunction is harder: one direction is easy ($\boxtimes^{\wedge 1}$ and $\boxplus^{\wedge 1}$, neither pictured), but the other direction for $\boxtimes$ requires some side conditions ($\boxtimes^{\wedge 2}$); the other direction for $\boxplus$ is not generally true. 1 and 0 are the units for $\boxtimes$ and $\boxplus$ ($\boxtimes^1, \boxplus^0$). If a world predicate W is true on each sub-distribution, then it is true on the whole distribution ($\boxplus^W$).

The distribution rules that relate $\boxtimes$ and $\boxplus$ are a bit more complicated. One direction ($\boxtimes$ over $\boxplus$, *i.e.* $\boxtimes^\boxplus$) is simple enough, but the converse ($\boxplus^\boxtimes$) is a bit delicate. However, it can be used to establish probabilistic independence through $\boxtimes$-factorization. Suppose the distribution of x, y satisfies:

$$\phi = \langle x, y = (0.25 : 0, 0) \oplus (0.25 : 0, 1) \oplus (0.25 : 1, 0) \oplus (0.25 : 1, 1)\rangle$$

We can prove that x and y are probabilistically independent as follows:

$\phi \vdash [\langle x = 0\rangle \boxtimes 0.5 \cdot \langle y = (0.5 : 0) \oplus (0.5 : 1)\rangle] \boxplus [\langle x = 1\rangle \boxtimes 0.5 \cdot \langle y = (0.5 : 0) \oplus (0.5 : 1)\rangle]$
($\boxplus$-split, factorize x, and then normalize y)
$\vdash \langle x = (0.5 : 0) \oplus (0.5 : 1)\rangle \boxtimes \langle y = (0.5 : 0) \oplus (0.5 : 1)\rangle$
(deduce the distribution of x using $\boxplus^\boxtimes$).

Rules for ⊠ predicates

$$\frac{}{\phi \boxtimes \psi \vdash \psi \boxtimes \phi}\,\boxtimes^{C} \qquad \frac{}{(\phi \boxtimes \psi) \boxtimes \theta \dashv\vdash \phi \boxtimes (\psi \boxtimes \theta)}\,\boxtimes^{A} \qquad \frac{}{(\phi \vee \psi) \boxtimes \theta \dashv\vdash (\phi \boxtimes \theta) \vee (\psi \boxtimes \theta)}\,\boxtimes^{\vee}$$

$$\frac{}{\phi \boxtimes 1 \vdash \phi}\,\boxtimes^{1} \qquad \frac{}{\phi \boxtimes (\boxplus_{i\in I}\, \psi_i) \vdash \boxplus_{i\in I}\, (\phi \boxtimes \psi_i)}\,\boxtimes^{\boxplus} \qquad \frac{\phi \vdash 1 \quad \mathsf{fv}(\psi_1) = \mathsf{fv}(\psi_2)}{(\phi \boxtimes \psi_1) \wedge (\phi \boxtimes \psi_2) \vdash \phi \boxtimes (\psi_1 \wedge \psi_2)}\,\boxtimes^{\wedge 2}$$

Rules for ⊞ predicates

$$\frac{}{\phi \boxplus \psi \vdash \psi \boxplus \phi}\,\boxplus^{C} \qquad \frac{}{(\phi \boxplus \psi) \boxplus \theta \vdash \phi \boxplus (\psi \boxplus \theta)}\,\boxplus^{A} \qquad \frac{}{(\phi \vee \psi) \boxplus \theta \dashv\vdash (\phi \boxplus \theta) \vee (\psi \boxplus \theta)}\,\boxplus^{\vee}$$

$$\frac{}{0 \boxplus \phi \vdash \phi}\,\boxplus^{0} \qquad \frac{}{\boxplus_{i\in I}\, [P] \vdash [P]}\,\boxplus^{W} \qquad \frac{}{\boxplus_{i\in I}\, (\alpha_i \cdot ((\hat{x} = \vec{v}) \boxtimes \phi_i)) \vdash (\hat{x} = \vec{v}) \boxtimes (\boxplus_{i\in I}\, (\alpha_i \cdot \phi_i))}\,\boxplus^{\boxtimes}$$

Rules for dot predicates

$$\frac{}{1 \cdot \phi \vdash \phi}\,\mathsf{D}^{1} \qquad \frac{}{0 \cdot \phi \vdash 0}\,\mathsf{D}^{0} \qquad \frac{\beta > 0}{\alpha \cdot \beta \dashv\vdash \alpha \times \beta}\,\mathsf{D}^{\times} \qquad \frac{}{\boxplus_{i\in I}\, \alpha_i \dashv\vdash \sum_{i\in I} \alpha_i}\,\mathsf{D}^{+}$$

Rules for base predicates

$$\frac{}{0 \vdash [P]}\,\mathsf{W}^{0} \qquad \frac{}{\alpha \cdot \langle P\rangle \dashv\vdash \alpha \wedge [P]}\,\mathsf{W}^{\mathsf{D}} \qquad \frac{a \in \mathbb{Q}}{\phi(x) \wedge [x = a] \vdash \phi(a)}\,\mathsf{W}^{=} \qquad \frac{P \vdash_M Q}{[P] \vdash [Q]}\,\mathsf{W}^{\vdash}$$

$$\frac{}{[P] \boxtimes \phi \vdash [P]}\,\mathsf{W}^{\boxtimes} \qquad \frac{a \in \mathbb{Q} \quad x \notin \mathsf{fv}(\phi)}{\langle x = a\rangle \wedge \phi \vdash \langle x = a\rangle \boxtimes \phi}\,\mathsf{W}^{\wedge}_{\boxtimes} \qquad \frac{}{[P] \wedge (\boxplus_{i\in I}\, \phi_i) \vdash \boxplus_{i\in I}\, ([P] \wedge \phi_i)}\,\mathsf{W}^{\wedge}_{\boxplus}$$

Rules for filter and normalization predicates

$$\frac{}{(\alpha \cdot \phi \boxplus \overline{\alpha} \cdot \psi)/[P] \dashv\vdash (\alpha \cdot \phi)/[P] \boxplus (\overline{\alpha} \cdot \psi)/[P]}\,\mathsf{F}^{\boxplus} \qquad \frac{\phi \vdash [P]}{\phi/[P] \dashv\vdash \phi}\,\mathsf{F}^{\mathsf{E}}_{1} \qquad \frac{\phi \vdash [\neg P]}{\phi/[P] \vdash 0}\,\mathsf{F}^{\mathsf{E}}_{2}$$

$$\frac{}{\phi/[P] \vdash [P]}\,\mathsf{F}^{\mathsf{W}} \qquad \frac{}{(\phi/[P])/[Q] \dashv\vdash \phi/[P \wedge W]}\,\mathsf{F}^{\wedge} \qquad \frac{\phi \vdash \rho}{(\alpha \cdot \phi)_{\downarrow\rho} \vdash \phi}\,\mathsf{N}^{\mathsf{E}} \qquad \frac{}{\phi_{\downarrow\rho} \vdash \rho}\,\mathsf{N}^{\|\cdot\|}$$

Fig. 3. Useful entailment rules.

The fractional predicate rules are straightforward: 1 is the unit and 0 is the null element (D^1, D^0); fractionalizing (non-zero) norm predicates together is the same as multiplying the norms ($D^\times$); and adding them together norm predicates with ⊞ is the same as adding the norms directly (D^+).

World predicates are a mix. Some are expected, including consequence ($W^\vdash$) and substitution ($W^=$). The null distribution implies all world predicates vacuously (W^0). We also have some entailments used to isolate ($W^\boxtimes$, W^D) and distribute ($W^\wedge_\boxtimes$, $W^\wedge_\boxplus$) world predicates.

Filter and norm behave straightforwardly. Filter distributes inside ⊞ ($F^\boxplus$), is removed when the world predicate holds (F^E_1), and collapses the distribution when its negation holds (F^E_2).

3.2 Side conditions on IFrame and DFrame

IFrame requires $\mathsf{fv}(F) \cap \mathsf{vars}(c) = \emptyset$ rather than the typical (weaker) $\mathsf{fv}(F) \cap \mathsf{mv}(c) = \emptyset$. Probabilistic independence can be broken simply by reading a variable. Suppose that x and y are independent; now frame out y's independence and run the command $x := y$, which reads y but does not modify it. Clearly x and y are no longer independent, contradicting the returning ⊠-frame.

DFrame requires the more typical restriction on modified variables, which is sufficient to guarantee that any independence facts within F are preserved. DFrame also requires the norm of Q be 1. This ensures that c terminates (almost surely), *i.e.* that c cannot change the aggregate probabilistic weight of the distribution (which has been preserved in F).

3.3 Other interesting Hoare rules

The final two interesting rules in our framework are Choice and If:

$$\frac{\{P\}\, c_1\, \{Q_1\} \quad \{P\}\, c_2\, \{Q_2\} \quad P \vdash [e \Downarrow \rho]}{\{P\}\, c_1 [e] c_2\, \{\rho \cdot Q_1 \boxplus \overline{\rho} \cdot Q_2\}}\text{[Choice]} \qquad \frac{\{P/[b]\}\, c_1\, \{Q_1\} \quad \{P/[\neg b]\}\, c_2\, \{Q_2\}}{\{P\}\ \textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2\, \{Q_1 \boxplus Q_2\}}\text{[If]}$$

The Choice rule for $c_1[e]c_2$ requires 3 side conditions: $\{P\}\, c_1\, \{Q_1\}$, $\{P\}\, c_2\, \{Q_2\}$, and that the precondition P is specific enough to entail $[e \Downarrow \rho]$, *i.e.* e is evaluated to some probability ρ. If e involves random variables, we must first use rules like Split, ISplit, and Dot to make e deterministic in each sub-case to meet this requirement. The If rule is derived from Choice by observing that P can be split into $(\alpha \wedge P/[b]) \boxplus (\overline{\alpha} \wedge P/[\neg b])$ for some probability α.

3.4 Reasoning about probabilistic termination with the While rule

Our While rule deviates from the traditional rule substantially:

$$\frac{\begin{array}{cc}\sum_{i=0}^{\infty} \rho_i = \rho \le 1 & \{P(n)\}\, c\, \{P(n+1) \boxplus Q(n+1)\} \\ Q(n) \vdash \rho_n \cdot \langle \neg b \rangle & P(n) \vdash (1 - \sum_{i=0}^{n} \rho) \cdot \langle b \rangle\end{array}}{\{P(0) \boxplus Q(0)\}\ \textbf{while}\ b\ \textbf{do}\ c\ \left\{\rho \cdot \langle \neg b \rangle \wedge \left(\boxplus_{i=0}^{\infty} Q(i)\right)\right\}}\text{[While]}$$

We work with indexed invariant pairs $P(n)$ and $Q(n)$ rather than a single invariant I, and also require an infinite indexed sequence of probabilities ρ_n, where ρ_i is the probability that the loop terminates after exactly i iterations. Consequently, the cumulative sum $\alpha_n \triangleq \sum_{i=0}^{n} \rho_i$ and its limit $\rho = \lim_{n\to\infty} \alpha_n$ denote the probabilities of the loop terminating within n iterations and eventually terminating, respectively. The loop terminates almost surely when $\rho = 1$.

$P(n)$, which satisfies $(1 - \alpha_n) \cdot \langle b \rangle$, characterizes the non-terminating program state after exactly n iterations, while $Q(n)$, which satisfies $\rho_n \cdot \langle b \rangle$, characterizes the program configurations that terminate during the n^{th} iteration. $\{P(n)\}\, c\, \{P(n+1) \boxplus Q(n+1)\}$ indicates that during the n^{th} iteration, $P(n)$ is "consumed" by the loop body to "produce" $Q(n+1)$—which exits the loop—and $P(n+1)$ to be utilized in the subsequent iteration. Consequently, with the initial condition $P(0) \boxplus Q(0)$, the loop's final state satisfies $\rho \cdot \langle \neg b \rangle \wedge (\boxplus_{i=0}^{\infty} Q(i))$.

The SWhile rule discussed in §1 is derived from the While rule for when each iteration has an identical terminating probability and a single invariant I can be stated naturally. The convexity side condition in SWhile collapses the infinite $\boxplus$-sum in the postcondition of the While rule. Intuitively, a predicate Q is convex if it is closed under convex combination, e.g. $\rho \cdot Q \boxplus \overline{\rho} \cdot Q$ entails Q. Convex predicates are formalized in Definition 11 and can be syntactically checked using Lemma 4.13.

4 A Model for Probabilistic Program States

In Section 4.1, we introduce the fundamental principles of the multiverse model. Moving on to Section 4.2, we explore its assertion languages and semantics. In Section 4.3, we establish a theoretical foundation for probabilistic program states using the multiverse model. The proofs of the lemmas in this section can be found in the Appendix B.

4.1 The Multiverse Model

We introduce the *multiverse model* to interpret discrete distributions. Intuitively, each multiverse is a weighted set of possible worlds in a Kriple resource model [36]. Fix a Kriple resource model

$\mathcal{M} = (M, \circ, e, \sqsubseteq_M)$ where M is a countable set of *possible worlds*, $\circ$ is an associative binary operator over worlds that has e as the identity element and is preserved under the preorder $\sqsubseteq_M$, i.e. if $m \sqsubseteq_M m'$ and both $m \circ n$, $m' \circ n$ are defined then $m \circ n \sqsubseteq_M m' \circ n$.

A *multiverse* (over $\mathcal{M}$) is the partial mapping $u : M \rightharpoonup (0, 1]$ where $u(w) > 0$ is the *weight* of $w \in \mathrm{dom}(u)$, and the *total weight* (or *norm*) of u — defined as $\|u\| \triangleq \sum_{m \in \mathrm{dom}(u)} u(m)$ — must satisfy $\|u\| \leq 1$. For convenience, we refer u as a Ω-multiverse to indicate that $\mathrm{dom}(u) = \Omega$. The *empty multiverse* $u_\emptyset$ is the only multiverse with total weight of zero. The *unit multiverse* is the singleton mapping $u_e : \{e\} \to \{1\}$. We say u is *normalized* if $\|u\| = 1$.

Let $\Omega_1, \Omega_2 \subseteq M$ and u_1, u_2 be Ω_1-multiverse and Ω_2-multiverse, respectively. We say Ω_1, Ω_2 are *separable*, denoted by $\Omega_1 \perp \Omega_2$, if they are non-empty, $w_1 \circ w_2$ is defined for every $w_1 \in \Omega_1, w_2 \in \Omega_2$, and if $w_1 \circ w_2 = w_1' \circ w_2'$ where either $w_1' \in \Omega_1$ or $w_2' \in \Omega_2$ then $w_1 = w_1'$ and $w_2 = w_2'$, i.e. every world in $\Omega_1 \circ \Omega_2$ has a unique (Ω_1, Ω_2)-decomposition.

The following basic operators and relations are defined over the multiverses u_1, u_2:

Definition 1 (**Singleton**). *Let $0 < \rho \leq 1$. The multiverse $\delta(w, \rho)$ is the singleton $\{w\} \to \{\rho\}$.*

Definition 2 (**Multiplication**). *If $\Omega_1 \perp \Omega_2$ then $u_1 \otimes u_2$ is the $(\Omega_1 \circ \Omega_2)$-multiverse where*

$$[u_1 \otimes u_2](w_1 \circ w_2) \triangleq u_1(w_1) \times u_2(w_2).$$

Definition 3 (**Scalar multiple**). *Let $\alpha \in \mathbb{R}_{>0}$. The α-*multiple *of u_1 is undefined if $\alpha \times \|u\| > 1$. Otherwise, $\alpha \bullet u_1$ is the Ω_1-multiverse where $[\alpha \bullet u_1](w) = \alpha \times u_1(w)$ for every $w \in \Omega_1$.*

Definition 4 (**Mapping**). *Let $f : \Omega_1 \to \Omega_2$ be a surjective mapping. Then $f(u_1)$ is the Ω_2-multiverse where $f(u_1)(w_2) = \sum_{w_1 \in f^{-1}(w_2)} u_1(w_1)$.*

Definition 5 (**Filter**). *Suppose that every world $w \in \Omega_1$ satisfies $w \models_M W \vee \neg W$ where $\models_M$ is the forcing semantics over worlds. Then the $[W]$-filter of u_1 is the multiverse $u/[W]$ where:*

$$u/[W](w) \triangleq \begin{cases} u(w) & \text{, if } w \in \Omega_1 \text{ and } w \models_M W \\ \text{undefined} & \text{, otherwise.} \end{cases}$$

Definition 6 (**Orderings**). *We let $u_1 \preceq u_2$ if $\Omega_1 \subseteq \Omega_2$ and $u_1(w) \leq u_2(w)$ for every $w \in \Omega_1$.*

We let $u_1 \sqsubseteq u_2$ if both u_1, u_2 are empty, or there exist two world sequence $(w_i)_{i \in I}$ and $(w_i')_{i \in I}$ (where the worlds are not necessarily distinct) and a non-zero weight sequence $(\rho_i)_{i \in I}$ such that

$$\forall i \in I.\ w_i \sqsubseteq_M w_i' \quad \text{and} \quad u_1 = \oplus_{i \in I}(\delta(w_i, \rho_i)) \quad \text{and} \quad u_2 = \oplus_{i \in I}(\delta(w_i', \rho_i)).$$

Definition 7 (**Supremum**). *Let $(u_i)_{i \in I}$ be a $\preceq$-ordered sequence (i.e. $u_i \preceq u_{i+1}$ for every i) where $I \subseteq \mathbb{N}$ and each u_i is a Ω_i-multiverse. For each world $w \in \Omega = \bigcup_{i \in I} \Omega_i$, we define the real $u_i^w \triangleq$ if $w \in \Omega_i$ then $u_i(w)$ else 0. The sequence $(u_i^w)_{i \in I}$ is non-decreasing, eventually non-zero, and bounded above by 1. Hence its finite non-zero supremum exists. Thus we can define $\sup (u_i)_{i \in I}$ as the Ω-multiverse u where $u(w) \triangleq \sup (u_i^w)_{i \in I}$ for every world $w \in \Omega$.*

Definition 8 (**Addition**). *If $\|u_1\| + \|u_2\| \leq 1$ then $u_1 \oplus u_2$ is the $\Omega_1 \cup \Omega_2$-multiverse where*

$$[u_1 \oplus u_2](w) \triangleq \begin{cases} u_1(w) + u_2(w) & \text{, if } w \in \Omega_1 \cap \Omega_2 \\ u_i(w) & \text{, if } w \in \Omega_i \text{ and } w \notin \Omega_1 \cap \Omega_2. \end{cases}$$

Generally, let $(u_i)_{i=0}^n$ where $n \in \mathbb{N} \cup \{\infty\}$ and $\sum_{i=0}^n \|u_i\| \leq 1$. Note that their partial sum sequence $(s_i \triangleq \oplus_{k=0}^i u_k)_{i=0}^n$ is $\preceq$-ordered. Thus we let $\oplus_{i=0}^n u_i \triangleq \sup (s_i)_{i=0}^n$.

Lemma 4.1. *Addition, multiplication, scalar multiple, mapping, filter, supremum are well-defined.*

Lemma 4.2. *The relation $\preceq$ is a partial order whereas the relation $\sqsubseteq$ is a preorder. Also, $u_1 \preceq u_2$ implies $\|u_1\| \leq \|u_2\|$ and $u_1 \sqsubseteq u_2$ implies $\|u_1\| = \|u_2\|$. Furthermore, if $u_1 \sqsubseteq u_2$ and both $u \otimes u_1, u \otimes u_2$ are defined then $u \otimes u_1 \sqsubseteq u \otimes u_2$.*

Let $U(M)$ be the set of multiverses over the world domain M. Then:

Lemma 4.3. *The structure $\mathcal{U} = (U(M), \oplus, \otimes, \|\cdot\|, u_\emptyset, u_e)$ is a partial commutative cancellative norm-preserving semi-ring where $\oplus$ has no inverse elements and $\otimes$ is not commutative, i.e.:*

Iden:	$u \oplus u_\emptyset$	$=$	u	$u \otimes u_e$	$=$	u	$= u_e \otimes u$
Comm:	$u_1 \oplus u_2$	$=$	$u_2 \oplus u_1$				
Assoc:	$(u_1 \oplus u_2) \oplus u_3$	$=$	$u_1 \oplus (u_2 \oplus u_3)$	$(u_1 \otimes u_2) \otimes u_3$	$=$	$u_1 \otimes (u_2 \otimes u_3)$	
Canc:	$u_1 \oplus u = u_2 \oplus u$	$\rightarrow$	$u_1 = u_2$	$u_1 \otimes u = u_2 \otimes u$	$\rightarrow$	$u_1 = u_2$	
Norm:	$\|u_1 \oplus u_2\|$	$=$	$\|u_1\| + \|u_2\|$	$\|u_1 \otimes u_2\|$	$=$	$\|u_1\| \times \|u_2\|$	
Dist:	$u = u_1 \oplus u_2 \rightarrow u'' = u' \otimes u \rightarrow u'' = (u' \otimes u_1) \oplus (u' \otimes u_2)$						

Commutatitivity for $\otimes$ can be recovered by forcing $\circ$ to be commutative:

Lemma 4.4. *If $\circ$ is commutative then $\otimes$ is also commutative.*

Our multiverse model, which is built upon an instance of the Kriple resource model, is itself a proper extension of the Kriple resource model:

Corollary 4.5. *The substructure $(U(M), \otimes, u_e, \sqsubseteq)$ is a Kriple resource model.*

4.2 Assertion Language and Semantics of the Multiverses

Our assertion language is a variant of the logic of bunched implication (BI) [35]:

$$\begin{aligned} \phi, \psi \quad ::= \quad & [W] \mid \rho \mid \bot \mid \top \mid \phi \boxtimes \psi \mid \phi \mathrel{-\!\!\boxtimes} \psi \mid \phi \boxplus \psi \mid \boxplus_{i \in I} \phi_i \mid \phi \mathrel{-\!\!\boxplus} \psi \mid \\ & \phi/[W] \mid \phi_{\downarrow \rho} \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \rightarrow \psi \mid \exists x.\, \phi \mid \forall x.\, \psi \end{aligned}$$

Unlike BI, our logic is designed to have a specific set of atomic propositions, i.e. the base/world predicate $[W]$, where W is a predicate over possible worlds, and the norm predicate $\rho \in [0, 1]$. Also, $\boxtimes$ and $\mathrel{-\!\!\boxtimes}$ are analogous to the *separating conjunction* and *magic wand* in BI, respectively. Our assertion language deviates from BI by including several new connectives: *the splitting conjunction* $\boxplus$ and the *splitting wand* $\mathrel{-\!\!\boxplus}$ to analyze sub-multiverses, the *filter predicate* $\phi/[W]$ to remove worlds that fail to satisfy W, and the *norm forcing predicate* $\phi/[\rho]$ to set the norm of the multiverse to ρ. Our logic also includes a different version of $\boxplus$, i.e. $\boxplus_{i \in I} \phi_i$ where the index I is countable, for infinite splitting as well as expressing distributions with countably infinite supports.

We employ the *forcing semantics* $\models$ to interpret our assertions. From a broad perspective, our multiverse semantics is derived from the semantics over worlds, i.e.:

Definition 9 (**Forcing semantics**). *Let $\models_M$ be the forcing semantics over worlds. The forcing semantics for Ω-multiverse u is as follows: $u \models \top$ always holds, $u \models \bot$ never holds. Furthermore:*

$$\begin{aligned}
u \models \rho &\triangleq \|u\| = \rho \\
u \models [W] &\triangleq \forall w \in \Omega.\, w \models_M W \\
u \models \phi \wedge \psi &\triangleq u \models \phi \wedge u \models \psi \\
u \models \phi \vee \psi &\triangleq u \models \phi \vee u \models \psi \\
u \models \phi \rightarrow \psi &\triangleq \forall u'.u \sqsubseteq u' \wedge u' \models \phi \\
&\qquad \rightarrow u' \models \psi \\
u \models \exists x.\, \phi &\triangleq \exists x.\, u \models \psi \\
u \models \forall x.\, \phi &\triangleq \forall x.\, u \models \psi
\end{aligned}$$

$$\begin{aligned}
u \models \phi \boxplus \psi &\triangleq \exists u_1 \exists u_2.\, u = u_1 \oplus u_2 \wedge u_1 \models \phi, u_2 \models \psi \\
u \models \boxplus_{i \in I} \phi_i &\triangleq \exists (u_i)_{i \in I}.\, u = \oplus_{i \in I} u_i \wedge \forall i \in I.\, u_i \models \phi_i \\
u \models \phi \mathrel{-\!\!\boxplus} \psi &\triangleq \forall u'.\, u' \models \phi \wedge (u \oplus u')\downarrow \rightarrow u \oplus u' \models \psi \\
u \models \phi/[W] &\triangleq \exists u'.\, u' \models \phi \wedge (u'/[W])\downarrow \wedge u = u'/[W] \\
u \models \phi \boxtimes \psi &\triangleq \exists u_1 \exists u_2.\, (u \otimes u')\downarrow \wedge u_1 \otimes u_2 \sqsubseteq u \\
&\qquad \wedge u_1 \models \phi \wedge u_2 \models \psi \\
u \models \phi \mathrel{-\!\!\boxtimes} \psi &\triangleq \forall u'.\, u' \models \phi \wedge (u \otimes u')\downarrow \rightarrow u \otimes u' \models \psi \\
u \models \phi_{\downarrow \rho} &\triangleq \|u\| = \rho \wedge \exists a \in \mathbb{R}_{>0}.\, (a \bullet \sigma)\downarrow \wedge a \bullet \sigma \models \phi
\end{aligned}$$

We use $\models_M$ to interpret the semantics of the base predicate $[W]$, i.e. the multiverse u satisfies $[W]$ if every world within it satisfies W. In most cases, our semantics closely resembles the semantics of BI with respect to the Kriple resource model. On the other hand, there are several new scenarios, i.e. $\boxplus_{i\in I}\phi_i$ means that the multiverse can be split into sub-multiverses $(u_i)_{i\in I}$ via $\oplus$, and each u_i satisfies ϕ_i; $\phi \mathrel{-\!\!\boxplus} \psi$ means that if the current multiverse can be combined via $\oplus$ with some multiverse satisfying ϕ then the result multiverse satisfies ψ. The interaction between $\boxplus$ and $-\!\!\boxplus$ is similar to the interaction between $\boxtimes$ and $-\!\!\boxtimes$:

Lemma 4.6. *For every multiverse u and $* \in \{\boxtimes, \boxplus\}$, we have $u \models \phi * (\phi \mathrel{-\!\!*} \psi)$ implies $u \models \psi$.*

The Kripke monotonicity of the multiverses is inherited from their worlds' counterpart:

Lemma 4.7 (Monotonicity). *Suppose $w \models_M W$ implies $w' \models_M W$ for every world predicate W and $w \sqsubseteq_M w'$. Then $u \models \phi$ implies $u' \models \phi$ for every multiverse predicate ϕ and $u \sqsubseteq u'$.*

Thanks to the Kripke monotonicity, $\rho \cdot \phi$ can be interpreted in terms of the ρ-multiple operator:

Lemma 4.8. *$u \models \rho \cdot \phi$ iff either $\rho = 0$ and $u = u_\emptyset$, or $u = \rho \bullet u'$ and $u' \models \phi$ for some u'.*

4.3 Interpretation of Probabilistic Program States

We first provide the *world model* to interpret the world predicates and then explain the mapping from probabilistic program states to multiverses.

World model. We fix a countable set of variables Var whose values are over the rationals $\mathbb{Q}$. A *deterministic program state* is finite partial mapping $m : \mathsf{Var} \rightharpoonup \mathbb{Q}$. For convenience, we call m a *D-map* if $\mathsf{dom}(m) = D$. We use $m_\emptyset$ to denote the $\emptyset$-map and $\mathsf{fmap}(D)$ the set of all D-maps. Let m_1, m_2 be D_1-map and D_2-map respectively. If $D_1 \cap D_2 = \emptyset$ then we define $m_1 \circ m_2$ to be the $D_1 \cup D_2$-map where $m_1 \circ m_2(v) = m_i(v)$ for every $v \in D_1 \cup D_2$ such that $v \in D_i$. Also, we let $m_1 \sqsubseteq_M m_2$ if $D_1 \subseteq D_2$ and $m_1(v) = m_2(v)$ for every $v \in D_1$.

Let $\mathsf{DS}(\mathsf{Var})$ be the set of all deterministic states. Our mappings forms a Kriple resource model:

Lemma 4.9. *$(\mathsf{DS}(\mathsf{Var}), \circ, m_\emptyset, \sqsubseteq_M)$ is a Kriple resource model where $\circ$ is commutative.*

We assume a partial expression evaluation $[\![\cdot]\!](\cdot) : \mathsf{Expr}\times\mathsf{DS}(\mathsf{Var}) \rightharpoonup \mathbb{Q}$ where $[\![e]\!](m)$ is defined if $\mathsf{vars}(e) \subseteq \mathsf{dom}(m)$. The Boolean expression b is treated as subtype of e where $[\![b]\!](m) \in \{0, 1\}$. The forcing semantics over worlds is defined as follows, where $\hat{e} \in (\mathsf{Expr})^n$ and $\hat{v} \in \mathbb{Q}^n$:

$$
\begin{array}{lcl|lcl}
m \models_M P(\hat{e}) & \triangleq & \exists \hat{v}.\ \hat{e} \Downarrow \hat{v} \wedge P(\hat{v}) & m \models_M W \wedge W' & \triangleq & m \models W \wedge m \models W' \\
m \models_M \neg P(\hat{e}) & \triangleq & \exists \hat{v}.\ \hat{e} \Downarrow \hat{v} \wedge \neg P(\hat{v}) & m \models_M W \vee W' & \triangleq & m \models W \vee m \models W' \\
m \models_M \neg W & \triangleq & m \models_M W' \text{ where} & m \models_M \exists v.\ W & \triangleq & \exists v.\ m \models_M W \\
 & & W' \text{ is the DNF of } W & m \models_M \forall v.\ W & \triangleq & \forall v.\ m \models_M W
\end{array}
$$

We adopt the following syntactic sugars: $b \equiv b \Downarrow 1$ and $\neg b \equiv b \Downarrow 0$, i.e. the Boolean expression b is evaluated to 1 and 0, respectively. Consequently, we can treat b as a world predicate, i.e. $[b]$ and $[\neg b]$. Also, note that world predicates like $W \vee \neg W$ is not a tautology in our world model. In fact, $m \models_M W \vee \neg W$ holds only when m contains all of the free variables in W.

Probabilistic states. Let $D \subseteq \mathsf{Var}$ be finite. A *D-state* is an Ω-multiverse for some $\Omega \subseteq \mathsf{fmap}(D)$. We let $\mathsf{vdom}(\sigma) \triangleq \Omega$. Intuitively, each D-state σ expresses a sub-distribution[3] over D. Consequently, the sum $\oplus_{i\in I}\ \sigma_i$ is defined only when $\sum_{i\in I}\ \|\sigma_i\| \leq 1$ and $\mathsf{vdom}(\sigma_i) = \mathsf{vdom}(\sigma_j)$ for every $i, j \in I$. It follows that separability can be interpreted in terms of domain disjointness, i.e.:

Lemma 4.10. *Let σ_1, σ_2 be D_1-state, D_2-state, respectively. Then $\mathsf{dom}(\sigma_1) \perp \mathsf{dom}(\sigma_1)$ iff $D_1 \cap D_2 = \emptyset$.*

[3] As we relax the condition that the measure of the entire sample space is at most 1 rather than equal to 1.

Next, we define the notions of probability and probabilistic independence in D-states:

Definition 10 (Probability interpretation). *Let σ a normalized D-state. The probability that a world predicate W holds in σ is defined as:*

$$\mathbb{P}_\sigma(W) \triangleq \sum_i \sigma(m_i) \text{ where } m_i \in \mathsf{dom}(\sigma) \text{ and } m_i \models_W W$$

Furthermore, let $\hat{x}, \hat{y}$ be disjoint tuples of distinct variables in D. Then $\hat{x}$ and $\hat{y}$ are probabilistic independent in σ if $\mathbb{P}_\sigma(\hat{x} = \hat{u} \wedge \hat{y} = \hat{v}) = \mathbb{P}_\sigma(\hat{x} = \hat{u}) \times \mathbb{P}_\sigma(\hat{y} = \hat{v})$ for every possible values $\hat{u}, \hat{v}$.

Hereafter, we may omit the subscript σ if it is clear from the context. The following result allows us to use the connective $\boxtimes$ to reason about probabilistic independence:

Lemma 4.11. *Let σ be a normalized D-state, $\hat{x}, \hat{y} \subseteq D$ and $\hat{x} \cap \hat{y} = \emptyset$. Then $\hat{x}$ and $\hat{y}$ are probabilistic independent in σ iff $\sigma \models \langle \hat{x} = \hat{x} \rangle \boxtimes \langle \hat{y} = \hat{y} \rangle$.*

Let $S \subseteq D$. The *S-restriction* of a D-map m is the S-map $\pi(m, S)$ where $\pi(m, S)(v) = m(v)$ for every $m \in S$. Similarly, the (lifted) S-restriction of a D-state σ is the S-state $\pi(\sigma, S)$ such that $\pi(\sigma, S)(m) = \sum_{m'} \sigma(m')$ where $m' \in \mathsf{dom}(\sigma)$ and $m = \pi(m', S)$. From a probabilistic point of view, $\pi(\sigma, S)$ captures the marginal distribution of σ over the restrictive variable domain S. It follows that the multiverse preorder $\sqsubseteq$ can be defined via state restriction and the forcing semantics for states only depends on the free variables in the formula:

Lemma 4.12. *Let σ, σ' be non-empty D-state and D'-state, respectively. Then:*

(1) *$\sigma \sqsubseteq \sigma'$ iff $D \subseteq D'$ and $\sigma = \pi(\sigma', D)$.*
(2) *$\sigma \models \phi$ iff $\mathsf{fv}(\phi) \subseteq D$ and $\pi(\sigma, \mathsf{fv}(\phi)) \models \phi$.*

Lastly, we provide the formal definition of convex predicates used in the SWhile rule:

Definition 11. *A predicate ϕ is convex if for every state σ and probability sequence $(\rho_i)_{i=0}^{\infty}$ such that $\sum_{i=0}^{\infty} \rho_i = 1$, if $\sigma \models \bigoplus_{i=0}^{\infty} (\rho_i \cdot \phi)$ then $\sigma \models \phi$.*

The following lemma is employed to syntactically verify the convexity of a predicate:

Lemma 4.13. *The following predicates are convex: ρ, $[P]$, $\hat{x} = \vec{v}$. Furthermore, if the predicates ϕ, ψ are convex then $\rho \cdot \phi$, $(\hat{x} = \vec{v}) \boxtimes \phi$, and $\phi \wedge \psi$ are also convex.*

5 Semantics of Hoare Triples

Configurations. We utilize the constructs from the multiverse model to define the operational semantics for probabilistic states. Let $D \subseteq \mathsf{Var}$ be finite. A *basic D-configuration* (D-config) $\langle c, m \rangle$ is a pair of program c and D-map m. A *probabilistic D-config* κ is a multiverse over D-configs. We may omit the domain D if it is not important or clear from the context.

We introduce the following notations using the mapping constructs for multiverses: $\langle c, \sigma \rangle$ — meaning every mapping in σ is associated with the program c — is the probabilistic config lifted from the mapping $\lambda m.\ \langle c, m \rangle$ and $\kappa \fatsemi c'$ — meaning the program c' is appended to every basic config in κ — is the probabilistic config lifted from the mapping $\lambda \langle c, m \rangle.\ \langle c, m \rangle \fatsemi c'$ where $\langle c, m \rangle \fatsemi c' \triangleq \langle c, c', m \rangle$. It follows that $\langle c_1; c_2, \sigma \rangle = \langle c_1, \sigma \rangle \fatsemi c_2$.

A basic config is final if its program is **skip**. A probabilistic config is *final* if it can be written as $\langle \textbf{skip}, \sigma \rangle$ for some state σ, i.e. all of its basic config is final. A nonempty probabilistic config is *running* if none of its basic configs is final. Consequently, each probabilistic config κ can be expressed uniquely as the $\oplus$-sum between a final config and a running config, i.e. $\kappa = \langle \textbf{skip}, \sigma \rangle \oplus \kappa_R$. We let $\lfloor \kappa \rfloor \triangleq \sigma$ the *terminating state* of κ and $\lceil \kappa \rceil \triangleq \kappa_R$ the *running config* of κ.

$$\frac{}{\langle \mathbf{skip}, m\rangle \rightsquigarrow \delta(\langle \mathbf{skip}, m\rangle, 1)}\ (\text{skStp}) \qquad \frac{v \in \mathrm{dom}(m) \quad [\![e]\!](m) = a \quad m' = m[v \Leftarrow a]}{\langle v := e, m\rangle \rightsquigarrow \delta(\langle \mathbf{skip}, m'\rangle, 1)}\ (\text{asStp})$$

$$\frac{}{\langle \mathbf{skip}; c, m\rangle \rightsquigarrow \delta(\langle c, m\rangle, 1)}\ (\text{seqStpF}) \qquad \frac{[\![e]\!](m) = \rho \in [0,1]}{\langle c_1 [e] c_2, m\rangle \rightsquigarrow \langle c_1, m\rangle \oplus_\rho \langle c_2, m\rangle}\ (\text{chStp})$$

$$\frac{c \neq \mathbf{skip} \quad \langle c, m\rangle \rightsquigarrow \kappa}{\langle c; c', m\rangle \rightsquigarrow \kappa \fatsemi c'}(\text{seqStpR}) \qquad \frac{}{\langle \mathbf{while}\ b\ \mathbf{do}\ c, m\rangle \rightsquigarrow \delta(\langle \mathbf{if}\ b\ \mathbf{then}\ \{c; \mathbf{while}\ b\ \mathbf{do}\ c\}, m\rangle, 1)}(\text{whStp})$$

(a) Basic step relation

$$\frac{\kappa \text{ is empty}}{\kappa \rightsquigarrow\!\!\!\!\rightarrow \kappa}(\text{emStp}) \qquad \frac{\kappa \text{ is not empty} \quad \kappa = \bigoplus_{i\in I} (\delta(\langle c_i, m_i\rangle, \rho_i)) \quad \forall i \in I.\ \langle c_i, m_i\rangle \rightsquigarrow \kappa_i}{\kappa \rightsquigarrow\!\!\!\!\rightarrow \bigoplus_{i\in I} (\rho_i \bullet \kappa_i)}(\text{okStp})$$

(b) Probabilistic step relation via lifting

Fig. 4. Step semantics for probabilistic programs

Step semantics. Our step semantics in Fig. 4 is two-fold: we start with the basic step $\rightsquigarrow$ (Fig. 4a) from a basic config to a normalized probabilistic config and then lift it into probabilistic step $\rightsquigarrow\!\!\!\!\rightarrow$ (Fig. 4b) between two probabilistic configs. It is worth noticing that the step relations are between configs of the same variable domain. While the sequential $\rightsquigarrow$-steps is almost standard, we still need to check to ensure that the program state contains the necessary variables to run the statement. The chStp step for probabilistic choice utilizes the modified $\oplus_\rho$-sum for the result config where

$$w_0 \oplus_\rho w_1 \triangleq \begin{cases} w_\rho & , \text{ if } \rho \in \{0,1\} \\ (\delta(w_1, \rho)) \oplus (\delta(w_2, 1-\rho)) & , \text{ otherwise.} \end{cases}$$

The lifted probabilistic step $\kappa \rightsquigarrow\!\!\!\!\rightarrow \kappa'$ is divided into two cases: (1) if κ is empty then κ' is also empty, (2) otherwise κ' is the config in which each basic config in κ takes a $\rightsquigarrow$-step.

We write $\kappa \rightsquigarrow\!\!\!\!\rightarrow_n \kappa'$ if κ' can be reached from κ in exactly n steps. The following result implies that $\rightsquigarrow\!\!\!\!\rightarrow$ is deterministic and the induced sequence of terminating states of κ is $\preceq$-ordered:

Lemma 5.1. *Let $\kappa \rightsquigarrow\!\!\!\!\rightarrow_{n_1} \kappa_1$ and $\kappa \rightsquigarrow\!\!\!\!\rightarrow_{n_2} \kappa_2$. Then*

(1) *If $n_1 = n_2$ then $\kappa_1 = \kappa_2$.*
(2) *If $n_1 \leq n_2$ then $\lfloor \kappa_1 \rfloor \preceq \lfloor \kappa_2 \rfloor$. Consequently, let $(\kappa_i)_{i=0}^n$ where $n \in \mathbb{N} \cup \{\infty\}$ be a sequence of configs such that $\kappa_i \rightsquigarrow\!\!\!\!\rightarrow \kappa_{i+1}$ for every i. Then there exists some state σ such that $\sigma = \sup(\lfloor \kappa_i \rfloor)_{i=0}^n$.*

Accordingly, we define the following notion of state convergence $\sigma \Conv_c \sigma'$ which indicates that an initial config $\langle c, \sigma\rangle$ converges to some terminating state σ':

Definition 12 (**State convergence**). *The* convergence relation *$\sigma \Conv_c \sigma'$ holds if there exists an infinite sequence of configs $(\kappa_i)_{i=0}^\infty$ such that $\kappa_0 = \langle c, \sigma\rangle$, $\kappa_i \rightsquigarrow\!\!\!\!\rightarrow \kappa_{i+1}$ for every i, and $\sigma' = \sup \lfloor \kappa_i \rfloor_{i=0}^\infty$.*

The semantics of our Hoare triples is defined based on the concept of state convergence:

Definition 13 (**Hoare triple**). *The triple $\{P\}\, c\, \{Q\}$ holds if for every state σ that contains all of the variables in c, if σ satisfies the precondition P then the config $\langle c, \sigma\rangle$ converges to some terminating state σ' that satisfies the postcondition Q. Formally:*

$$\{P\}\, c\, \{Q\} \ \triangleq\ \forall \sigma \in \mathcal{S}^+(\mathrm{vars}(c)).\ \sigma \models P \rightarrow \exists \sigma'.\ \sigma \Conv_c \sigma' \wedge \sigma' \models Q$$

where $\sigma \in \mathcal{S}^+(D)$ if and only if $D \subseteq \mathrm{vdom}(\sigma)$.

1 $\{\langle x = 0\rangle \boxtimes \langle y = 0\rangle\}$
2 $x := (\alpha : 0) \oplus (\overline{\alpha} : 1); y := (\alpha : 0) \oplus (\overline{\alpha} : 1)$
3 $\{\langle x = (\alpha : 0) \oplus (\overline{\alpha} : 1)\rangle \boxtimes \langle y = (\alpha : 0) \oplus (\overline{\alpha} : 1)\rangle\}$
4 $\left\{\left\langle x, y = (\alpha^2 : 0, 0) \oplus (\alpha\overline{\alpha} : 0, 1) \oplus (\alpha\overline{\alpha} : 1, 0) \oplus (\overline{\alpha}^2 : 1, 1)\right\rangle\right\}$
5 $\left\{\rho \cdot \left\langle x, y = (\alpha^2/\rho : 0, 0) \oplus (\overline{\alpha}^2/\rho : 1, 1)\right\rangle \boxplus \overline{\rho} \cdot \langle x, y = (0.5 : 0, 1) \oplus (0.5 : 1, 0)\rangle\right\}$ // I
6 **while** $x = y$ **do**
7 $\quad\left\{\left\langle x, y = (\alpha^2/\rho : 0, 0) \oplus (\overline{\alpha}^2/\rho : 1, 1)\right\rangle\right\}$ // $I/\langle x = y\rangle$
8 $\quad x := (\alpha : 0) \oplus (\overline{\alpha} : 1); y := (\alpha : 0) \oplus (\overline{\alpha} : 1)$
9 $\quad\{\langle x = (\alpha : 0) \oplus (\overline{\alpha} : 1)\rangle \boxtimes \langle y = (\alpha : 0) \oplus (\overline{\alpha} : 1)\rangle\}$ // $\vdash I \wedge (\rho \cdot \langle x = y\rangle \boxplus \overline{\rho} \cdot \langle \neg(x = y)\rangle)$
10 $\{\langle x, y = (0.5 : 0, 1) \oplus (0.5 : 1, 0)\rangle\}$ // $I/\langle \neg(x = y)\rangle$
11 $\{\langle x = (0.5 : 0) \oplus (0.5 : 1)\rangle\}$

Fig. 5. Verification of fair coin simulation.

Our rules, including those mentioned in §1 and §3, are sound with respect to the given semantics[4]:

Theorem 5.2 (Soundness). *If* $\vdash \{P\}\, c\, \{Q\}$ *then* $\models \{P\}\, c\, \{Q\}$.

6 Evaluation for Proving Almost-sure Termination

We apply our framework to prove almost-sure termination of probabilistic programs under different conditions: beginning with von Neumann's trick involving parameterized probability in §6.1, moving on to martingale betting with infinite support in §6.2, and concluding with probabilistic loops featuring dynamic distributions in §6.3.

6.1 Parameterized Distributions

Our framework can reason about distributions with parameterized probabilities. In Figure 5, we validate the von Neumann's trick [43] which simulates a fair coin using a biased coin by flipping it twice repeatedly until the two outcomes (i.e. x and y) are different. The biased coin's distribution vector is determined by the parameter α, as follows:

$$\vec{v} \triangleq (\alpha : 0) \oplus (\overline{\alpha} : 1) \qquad \text{where } 0 < \alpha < 1$$

We show that $\langle x = (0.5 : 0) \oplus (0.5 : 1)\rangle$, i.e. the process eventually terminates and x has the desired distribution. Following the first attempt, the combined distribution of x, y satisfies the predicate $\rho \cdot P \boxplus \overline{\rho} \cdot Q$ (line 5) where:

$$\rho \triangleq \alpha^2 + \overline{\alpha}^2 \qquad P \triangleq \left\langle x, y = (\alpha^2/\rho : 0, 0) \oplus (\overline{\alpha}^2/\rho : 1, 1)\right\rangle \qquad Q \triangleq \langle x, y = (0.5 : 0, 1) \oplus (0.5 : 1, 0)\rangle$$

Accordingly, we apply the SWhile rule with the invariant $I \triangleq \rho \cdot P \boxplus \overline{\rho} \cdot Q$. Note that $P \vdash \langle x = y\rangle$ whereas $Q \vdash \langle \neg(x = y)\rangle$. As a result, P and Q are equivalent to $I/\langle x = y\rangle$ and $I/\langle \neg(x = y)\rangle$, respectively. We enter the loop body with P and use it to derive the loop body's postcondition $I \wedge (\rho \cdot \langle x = y\rangle \boxplus \overline{\rho} \cdot \langle \neg(x = y)\rangle)$ (line 9). Also, the predicate Q is convex by Lemma 4.13. Consequently, we exit the loop with Q which in turn implies $\langle x = (0.5 : 0) \oplus (0.5 : 1)\rangle$ as intended.

6.2 Distributions with Infinite Support

In Fig. 6, we apply our framework to reason about distribution with infinite support. The given program adopts the widely recognized martingale strategy in betting games, where a player bets

[4] A full set of inference rules together with their soundness proofs can be found in the Appendix E.

$\{\langle x, y, z = 0, 1, 0\rangle\}$ // $P(0) \boxplus Q(0)$
while $x = 0$ **do**
  $\{0.5^n \cdot \langle x, y, z = 0, 2^n, 1 - 2^n\rangle\}$ // $P(n)$
  ↘ $\{\langle x, y, z = 0, 2^n, 1 - 2^n\rangle\}$
    $x := (0.5 : 0) \oplus (0.5 : 1)$;
    $\{\langle x, y, z = (0.5 : 0, 2^n, 1 - 2^n) \oplus (0.5 : 1, 2^n, 1 - 2^n)\rangle\}$
    $z := z - y; y := y + y$;
    $\left\{\left\langle x, y, z = (0.5 : 0, 2^{n+1}, 1 - 2^{n+1}) \oplus (0.5 : 1, 2^{n+1}, 1 - 2^{n+1})\right\rangle\right\}$
    if $x = 1$ **then** $z := z + y$;
  ↙ $\left\{\left\langle x, y, z = (0.5 : 0, 2^{n+1}, 1 - 2^{n+1}) \oplus (0.5 : 1, 2^{n+1}, 1)\right\rangle\right\}$
  $\left\{0.5^{n+1} \cdot \left\langle x, y, z = 0, 2^{n+1}, 1 - 2^{n+1}\right\rangle \boxplus 0.5^{n+1} \cdot \langle x, z = 1, 1\rangle\right\}$ // $P(n+1) \boxplus Q(n+1)$
$\{\langle z = 1\rangle\}$

Fig. 6. Generating a distribution (i.e. y) with infinitely many outcomes

y on the outcome of a coin flip x to win either $2y$ (if $x = 1$) or 0 (if $x = 0$). The player's strategy involves doubling y until he wins. Consequently, he is assured of a positive net profit z, provided that he has an infinite amount of money to play the game.

Initially, $x = z = 0$ and $y = 1$. Our postcondition is $\langle z = 1\rangle$, i.e. the game terminates almost surely and the player gains a net profit of 1. Note that the distribution of y has infinitely many outcomes, with n consecutive losses resulting in $y = 2^n$. We apply the While rule with the following invariant:

$$\rho_0 \triangleq 0, \rho_{n+1} \triangleq 0.5^{n+1} \quad P(n) \triangleq 0.5^n \cdot \langle x, y, z = 0, 2^n, 1 - 2^n\rangle \quad Q(n) \triangleq \rho_n \cdot \langle x, z = 1, 1\rangle$$

Note that $\rho_{n+1} = 1 - \sum_{i=0}^{n+1} \rho_i$. Thus $P(n) \vdash \left(1 - \sum_{i=0}^{n} \rho_i\right) \cdot \langle x = 0\rangle$ and $Q(n) \vdash \rho_n \cdot \langle x \neq 0\rangle$. As $\sum_{i=0}^{\infty} \rho_i = 1$, we can deduce the following postcondition of the loop:

$$\langle x \neq 0\rangle \wedge \left(\boxplus_{i=0}^{\infty} Q(i)\right)$$

As $Q(n) \vdash \rho_n \cdot \langle z = 1\rangle$ and $\langle z = 1\rangle$ is convex (Lemma 4.13), we conclude that $\boxplus_{i=0}^{\infty} Q(i) \vdash \langle z = 1\rangle$.

6.3 Dynamic Distributions

Our framework can handle the cases where the terminating probabilities of each iteration are different. For instance, each iteration of the loop in Fig. 7a terminates with a probability of $1 - y$ (i.e. when $x = 1$) and the value of y is halved after each iteration. Initially, $x = 0$ and $y = 1$. We utilize the While rule with the following invariant pair:

$$P(n) \triangleq \beta_n \cdot \langle x, y = 0, 0.5^n\rangle \qquad Q(n) \triangleq \rho_n \cdot \langle x = 1\rangle$$

where $(\rho_i)_{i=0}^{\infty}$ will be determined later and $(\beta_n)_{n=0}^{\infty}$ is defined as $\beta_n \triangleq 1 - \sum_{i=0}^{n} \rho_i$.

Using $P(n)$ as the precondition, we then derive the following postcondition of the loop body:

$$0.5^{n+1}\beta_n \cdot \left\langle x, y = 0, 0.5^{n+1}\right\rangle \boxplus (1 - 0.5^{n+1})\beta_n \cdot \left\langle x, y = 1, 0.5^{n+1}\right\rangle$$

To recover the invariant $P(n+1) \boxplus Q(n+1)$, the following conditions must satisfy:

$$\beta_{n+1} = 0.5^{n+1}\beta_n \quad \text{and} \quad \rho_{n+1} = (1 - 0.5^{n+1})\beta_n$$

Note that $\rho_0 = 0$ and so $\beta_0 = 1$. Thus we deduce that:

$$\beta_n = 0.5^{n(n+1)/2} \quad \text{and} \quad \rho_n = (1 - 0.5^{n+1})0.5^{n(n+1)/2}$$

$\{\langle x, y = 0, 1\rangle\}$ // $P(0) \boxplus Q(0)$
while $x = 0$ **do**
  $\{\beta_n \cdot \langle x, y = 0, 0.5^n\rangle\}$ // $P(n)$
  $\searrow$ $\{\langle x, y = 0, 0.5^n\rangle\}$
    $y := y/2; x := 0$ $[y]$ $x := 1$
  $\swarrow$ $\left\{\left\langle x, y = (0.5^{n+1} : 0, 0.5^{n+1}) \oplus (1 - 0.5^{n+1} : 1, 0.5^{n+1})\right\rangle\right\}$
  $\left\{0.5^{n+1}\beta_n \cdot \left\langle x, y = 0, 0.5^{n+1}\right\rangle \boxplus (1 - 0.5^{n+1})\beta_n \cdot \langle x = 1\rangle\right\}$
  $\left\{\beta_{n+1} \cdot \left\langle x, y = 0, 0.5^{n+1}\right\rangle \boxplus \rho_{n+1} \cdot \langle x = 1\rangle\right\}$ // $P(n+1) \boxplus Q(n+1)$
$\{\langle x = 1\rangle\}$

(a) A while loop in which the terminating probability increases after each iteration.

$\{\langle x, y = 0, 1\rangle\}$ // $P(0) \boxplus Q(0)$
while $x = 0$ **do**
  $\{\beta_n \cdot \langle x = 0\rangle \boxtimes \langle 0 \le y \le 0.5^n\rangle\}$ // $P(n)$
  $\searrow$ $\{\langle 0 \le y \le 0.5^n\rangle\}$
    $y := y/2$ $[1/2]$ $y := y/3$;
  $\swarrow$ $\left\{\left\langle 0 \le y \le 0.5^{n+1}\right\rangle\right\}$
  $\searrow$ $\left\{\langle x = 0\rangle \boxtimes \left(\left\langle y = \bigoplus_{i\in I} (\alpha_i : \mathcal{A}_i)\right\rangle \wedge \left\langle 0 \le y \le 0.5^{n+1}\right\rangle\right)\right\}$
    $\left\{\boxplus_{i\in I} \left(\alpha_i \cdot \left\langle x, y = 0, \mathcal{A}_i \wedge 0 \le \mathcal{A}_i \le 0.5^{n+1}\right\rangle\right)\right\}$
    $x := 0$ $[y]$ $x := 1$;
  $\swarrow$ $\left\{\boxplus_{i\in I} \left(\alpha_i\mathcal{A}_i \cdot \left\langle x, y = 0, \mathcal{A}_i \wedge 0 \le \mathcal{A}_i \le 0.5^{n+1}\right\rangle \boxplus \alpha_i(1 - \mathcal{A}_i) \cdot \left\langle x, y = 1, \mathcal{A}_i \wedge 0 \le \mathcal{A}_i \le 0.5^{n+1}\right\rangle\right)\right\}$
  $\left\{\beta_n(\sum_{i\in I} \alpha_i\mathcal{A}_i) \cdot \langle x = 0\rangle \boxtimes \left\langle 0 \le y \le 0.5^{n+1}\right\rangle \boxplus (\sum_{i\in I} \alpha_i(1 - \mathcal{A}_i)) \cdot \langle x = 1\rangle\right\}$
  $\left\{\beta_{n+1} \cdot \langle x = 0\rangle \boxtimes \left\langle 0 \le y \le 0.5^{n+1}\right\rangle \boxplus \rho_{n+1} \cdot \langle x = 1\rangle\right\}$ // $P(n+1) \boxplus Q(n+1)$
$\{\langle x = 1\rangle\}$

(b) A while loop in which the terminating probability is probabilistic.

Fig. 7. Reasoning with dynamic terminating probabilities.

As $\lim_{n\to\infty} \beta_n = 0$, we obtain the loop's postcondition $\langle \neg(x = 0)\rangle \wedge (\boxplus_{i=0}^{\infty} Q(n))$. Note that $\langle x = 1\rangle$ is convex by Lemma 4.13. Hence, $\boxplus_{i=0}^{\infty} Q(i) \vdash \langle x = 1\rangle$ as desired.

In Fig. 7b, we consider a more complicated example where the probability in the probabilistic choice is random. Here each iteration terminates with a probability of y (i.e. when $x = 0$) where the value of y is determined by the probabilistic choice $y := y/2[1/2]y := y/3$. We apply the While rule with the following invariant pair:

$$P(n) \triangleq \beta_n \cdot \langle x = 0\rangle \boxtimes \langle 0 \le y \le 0.5^n\rangle \qquad Q(n) \triangleq \rho_n \cdot \langle x = 1\rangle$$

where $(\rho_i)_{i=0}^{\infty}$ is unknown and $\beta_n \triangleq 1 - \sum_{i=0}^{n} \rho_i$.

Instead of computing the exact forms of $(\rho_i)_{i=0}^{\infty}$ and $(\beta_i)_{i=0}^{\infty}$, it suffices to show that $\lim_{n\to\infty} \beta_n = 0$ to prove almost-sure termination. Following the probabilistic assignment for y, we are able to derive the inequality $\left\langle 0 \le y \le 0.5^{n+1}\right\rangle$. Next, we assume that y adheres to an arbitrary distribution $\bigoplus_{i\in I} (\alpha_i : \mathcal{A}_i)$, where I is countable, $0 \le \mathcal{A}_i \le 0.5^{n+1}$, and $\sum_{i\in I} \alpha_i = 1$. Using a combination of Split/ISplit and Dot, we make y deterministic in each sub-case and subsequently derive the

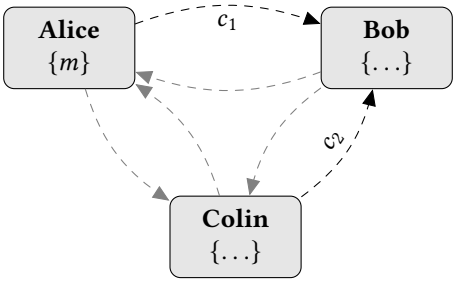


Fig. 8. Alice's secret m is non-leaking to Bob if $\mathbb{P}(c_1, c_2 | m = m_1) = \mathbb{P}(c_1, c_2 | m = m_2)$ for every m_1, m_2

following postcondition for the loop body:

$$\beta_n \left(\sum_{i \in I} \alpha_i \mathcal{A}_i \right) \cdot \langle x = 0 \rangle \boxtimes \left\langle 0 \leq y \leq 0.5^{n+1} \right\rangle \boxplus \left(\sum_{i \in I} \alpha_i (1 - \mathcal{A}_i) \right) \cdot \langle x = 1 \rangle$$

Accordingly, the invariant $P(n+1) \boxplus Q(n+1)$ can be recovered by forcing the following equations:

$$\beta_{n+1} = \beta_n \left(\sum_{i \in I} \alpha_i \mathcal{A}_i \right) \quad \text{and} \quad \rho_{n+1} = \sum_{i \in I} \alpha_i \, (1 - \mathcal{A}_i)$$

As $\mathcal{A}_i \leq 0.5^{n+1}$ for every i, we obtain the following upper bound for β_{n+1}

$$\beta_{n+1} \;\leq\; \beta_n \left(\sum_{i \in I} 0.5^{n+1} \alpha_i \right) \;=\; 0.5^{n+1} \beta_n \;\leq \ldots \leq\; 0.5^{(n+1)(n+2)/2} \beta_0 \;=\; 0.5^{(n+1)(n+2)/2}$$

Hence, $\lim_{n \to \infty} \beta_n = 0$ which implies that the program terminates almost surely.

7 Evaluation for Proving Probabilistic Independence

In §7.1, we explain how the correctness of various security protocols can be reduced to probabilistic independence. We then apply our framework in §7.2-§7.4 to verify the relevant security protocols.

7.1 Privacy Correctness for Security Protocols

Security privacy. We benchmark the local reasoning feature of our framework using the security protocols mentioned in [4] for secured information sharing. In detail, the setting of a security protocol Γ involves multiple entities $\{E_i\}_{i=1}^n$, where each E_i wants communicate with the others in a secured manner without leaking its private information to unintended recipients.

For instance, Fig. 8 illustrates the communication among Alice, Bob, and Colin. Here Bob receives two public ciphertexts c_1 and c_2 from Alice and Colin, respectively. Meanwhile, Alice possesses another message, denoted as m, which she intends to keep confidential from Bob. We define the secret message m as being *non-leaking* to Bob if he is unable to deduce the content of m solely based on the received ciphertexts c_1 and c_2. It is crucial to emphasize that solely considering the direct ciphertext c_1 from Alice does not guarantee the desired level of secrecy, as information regarding m can potentially be indirectly leaked through c_2 via Colin.

For the purpose of this work, we formalize privacy as follows.

Definition 14 (**Probabilistic non-interference (PN)**). *An immutable*[5] *message m is* non-leaking *to an entity Bob if he either 1) does not receive any message, or 2) receives $n > 0$ messages*

[5] i.e. $m \notin \mathrm{mv}(\Gamma)$ where Γ is the program expressing the security protocol.

$\hat{c} = (c_1, \ldots, c_n)$ *whose distribution is the same regardless of the content of* m*, i.e.*

$$\mathbb{P}(\hat{c} \mid m = a_1) \;=\; \mathbb{P}(\hat{c} \mid m = a_2) \quad \textit{for every possible values } a_1, a_2 \textit{ of } m.$$

The above property can be translated into the following Hoare triple:

$$\exists \vec{v} \forall a.\ \{\langle m = a\rangle \boxtimes F\}\, \Gamma\, \{\hat{c} = \vec{v}\}\,.$$

Intuitively, it means that $\hat{c}$ *do not disclose any information about* m*.*

Remark. There is another privacy condition called **Input independence (II)** which assumes m is a random variable with some arbitrary distribution and then shows that the distributions of m and $\hat{c}$ (the ciphertexts) are independent. Its respective Hoare triple is:

$$\exists \vec{v} \forall \vec{a}.\ \{(m = \vec{a}) \boxtimes F\}\, \Gamma\, \{(m = \vec{a}) \boxtimes (\hat{c} = \vec{v})\}$$

In many cases, (PN) and (II) are equivalent (e.g. [21]) even though their proofs could be vastly different [4]. It turns our that once they are formalized in our framework (w.r.t. our definitions), their equivalence can be derived directly:

Lemma 7.1. *The Hoare triples for non-interference and input independence are equivalent.*

Proof strategy. Given a security protocol Γ. We derive the following Hoare triple to prove that Γ satisfies both computation correctness and privacy correctness, i.e.

$$\exists \vec{v} \forall a.\ \{\langle m = a\rangle \boxtimes F\}\, \Gamma\, \{c = \vec{v} \wedge \langle Q\rangle\}$$

where the normalized predicate $\langle Q\rangle$ represents the desired computational result.

Assumption and notations. Unless stated otherwise, the messages are bit arrays of the same length. We override assignments for bit arrays as follows, where x, y, z are of length $n > 0$:

$$x := \vec{v}\ (\forall i.\ x[i] := \vec{v}) \qquad x := y\ (\forall i.\ x[i] := y[i]) \qquad x := y \text{ op } z\ (\forall i.\ x[i] := y[i] \text{ op } z[i])$$

For example, $x := (0.5 : 0) \oplus (0.5 : 1)$ is equivalent to the sequence $c_0; \ldots; c_{n-1}$ where c_i is simply $x[i] := (0.5 : 0) \oplus (0.5 : 1)$. Similarly, the following equality predicates are interpreted bit-wise:

$$\langle x = y\rangle \triangleq \langle x[0] = y[0] \wedge x[1] = y[1] \wedge \ldots\rangle \qquad \langle x = y \text{ op } z\rangle \triangleq \langle x[0] = y[0] \text{ op } z[0] \wedge \ldots\rangle$$

We use the following results to reason about array assignments:

Lemma 7.2. *Let* x, y, z *be arrays of length* $n > 0$*. The following Hoare triples hold:*

$$\{1\}\, x := \vec{v}\, \left\{\boxtimes_{i=0}^{n-1}\ (x[i] = \vec{v})\right\} \qquad \{1\}\, x := y\, \{\langle x = y\rangle\} \qquad \{1\}\, x := y \text{ op } z\, \{\langle x = y \text{ op } z\rangle\}\,.$$

Let S be a non-empty finite set. We write $\mathsf{U}(x) \triangleq x = \sum_{a \in S}\ (1/|S| : a)$ to mean that the distribution of x is uniform over S. We use the following results to reason about uniform distributions:

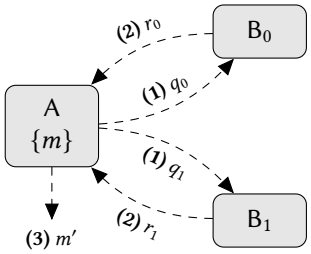
Lemma 7.3. *Let* x, y, z *be of length* $n > 0$ *and their elements are drawn from* S*. Then:*

(1) *For every state* σ*, we have* $\sigma \models \mathsf{U}(x)$ *iff* $\sigma \models \boxtimes_{i=0}^{n-1} \mathsf{U}(x[i])$.
(2) $\forall \hat{a}.\ \{\mathsf{U}(x) \boxtimes \langle y = \hat{a}\rangle\}\, z := x \text{ xor } y\, \{\mathsf{U}(x) \wedge \mathsf{U}(z)\}$.

7.2 Private Information Retrieval

We consider the Private Information Retrieval (PIR) scheme [10], where a client A wants retrieve an item from a database server without letting the server know about the retrieved item. An efficient PIR scheme is given in Fig. 9a given that the database is stored in two servers B_0 and B_1. We assume that the database is a bit array d whilst the client's secret selection is a bit array m where $m[k] = 1$ if and only if A wants to retrieve $d[k]$. The following steps are executed:

(1) The client A sends a random bit arrays q_0 to B_0 and $q_1 := q_0 \text{ xor } m$ to B_1, respectively.
(2) Each B_i computes a bit $r_i := \mathsf{XOR}_{k=0}^{n-1}\ (q_i[k] \times d[k])$ and sends r_i to A.



$q_0 := (0.5 : 0) \oplus (0.5 : 1)$;
$q_1 := q_0$ xor m;
for $i := [0, n-1]$ **do** $r_0 := r_0$ xor $(q_0[i] \times d[i])$;
for $i := [0, n-1]$ **do** $r_1 := r_1$ xor $(q_1[i] \times d[i])$;
$m' := r_0$ xor r_1;

(a) Private Information Retrieval

1 $\left\{\langle m = \hat{a}\rangle \boxtimes \left\langle d = \hat{b}\right\rangle \boxtimes \langle r_0 = 0\rangle \boxtimes \langle r_1 = 0\rangle\right\}$
2 $\searrow \{\langle m = \hat{a}\rangle\}$
3 $q_0 := (0.5 : 0) \oplus (0.5 : 1); q_1 := q_0$ xor m;
4 $\swarrow \{\mathsf{U}(q_0) \wedge \mathsf{U}(q_1) \wedge \langle q_1 = q_0 \text{ xor } m\rangle\}$ //privacy correctness
5 $\searrow \left\{\left\langle r_0 = 0 \wedge d = \hat{b}\right\rangle\right\} \rightsquigarrow F : \left\langle r_1 = 0 \wedge d = \hat{b}\right\rangle \wedge \langle q_1 = q_0 \text{ xor } m\rangle \wedge \mathsf{U}(q_0) \wedge \mathsf{U}(q_1)$
6 **for** $i := [0, n-1]$ **do**
7 $\{I_0(i) \triangleq \langle r_0 = \mathrm{XOR}_{k=0}^{i-1}\ (q_0[k] \times d[k])\rangle\}$ //loop invariant
8 $r_0 := r_0$ xor $(q_0[i] \times d[i])$;
9 $\{\langle r_0 = \mathrm{XOR}_{k=0}^{i}\ (q_0[k] \times d[k])\rangle\}$ //$\vdash I_0(i+1)$
10 $\swarrow \{\langle r_0 = \mathrm{XOR}_{k=0}^{n-1}\ (q_0[k] \times d[k])\rangle\}$
11 $\searrow \left\{\left\langle r_1 = 0 \wedge d = \hat{b}\right\rangle\right\} \rightsquigarrow F : \langle r_0 = \mathrm{XOR}_{k=0}^{n-1}(q_0[k] \times d[k])\rangle \wedge \langle q_1 = q_0 \text{ xor } m\rangle \wedge \mathsf{U}(q_0) \wedge \mathsf{U}(q_1)$
12 **for** $i := [0, n-1]$ **do** $r_1 := r_1$ xor $(q_1[i] \times d[i])$;
13 $\swarrow \{\langle r_1 = \mathrm{XOR}_{k=0}^{n-1}\ (q_1[k] \times d[k])\rangle\}$
14 $\searrow \{\langle q_1 = q_0 \text{ xor } m \wedge r_0 = \mathrm{XOR}_{k=0}^{n-1}\ (q_0[k] \times d[k]) \wedge r_1 = \ldots\rangle\} \rightsquigarrow F : \mathsf{U}(q_0) \wedge \mathsf{U}(q_1)$
15 $m' := r_0$ xor r_1;
16 $\swarrow \{\langle m' = \mathrm{XOR}_{k=0}^{n-1}\ (m[k] \times d[k])\rangle\}$ //computation correctness
17 $\{\langle m' = \mathrm{XOR}_{k=0}^{n-1}\ (m[k] \times d[k])\rangle \wedge \mathsf{U}(q_0) \wedge \mathsf{U}(q_1)\}$

(b) Proof sketch

Fig. 9. Verification of Private Information Retrieval where the frame predicates of DFrame are colored in grey.

(3) The client A computes $m' := r_0$ xor r_1 to retrieve the content of the item.

In Figure 9b, we prove that m is non-leaking to B$_0$, B$_1$, and A computes the correct result:

$$\mathsf{U}(q_0) \wedge \mathsf{U}(q_1) \wedge \left\langle m' = \mathrm{XOR}_{k=0}^{n-1}\ (m[k] \times d[k])\right\rangle$$ [6]

For privacy correctness, we apply lemmas 7.2 and 7.3 to prove that q_0 and q_1 are uniform (line 4). For computation correctness, we use the For rule to infer the values of r_0 (line 10) and r_1 (line 13):

$$\frac{\{\langle I(i) \wedge i \le n_2\rangle\}\ c\ \{\langle I(i+1)\rangle\} \quad n_1 \le n_2 + 1 \quad i \notin \mathrm{fv}(I) \cup \mathrm{vars}(c)}{\{\langle I(n_1)\rangle\}\ \textbf{for}\ i := [n_1, n_2]\ \textbf{do}\ c\ \{\langle I(n_2+1)\rangle\}}\ [\textsc{For}]$$

[6]Note that $m[k] \times d[k] = d[k]$ if $k = i$, the index of the chosen item, otherwise $m[k] \times d[k] = 0$. Thus $m' = d[i]$.

Consequently, we deduce that $\langle r_i = \mathsf{XOR}_{k=0}^{n-1}\ (q_i[k] \times d[k])\rangle$ holds for $i \in \{0,1\}$. Note that $\langle q_1 = q_0 \text{ xor } m\rangle$ implies $\langle m = q_0 \text{ xor } q_1\rangle$. Hence, we can deduce the value of m' as follows (line 16):

$$\langle m' = r_0 \text{ xor } r_1\rangle \vdash \langle m' = \mathsf{XOR}_{k=0}^{n-1}\ ((q_0[k] \text{ xor } q_1[k]) \times d[k])\rangle \vdash \langle m' = \mathsf{XOR}_{k=0}^{n-1}\ (m[k] \times d[k])\rangle$$

As a result, we conclude that the client A computes the correct answer.

7.3 Oblivious Transfer

Our next example is the Oblivious Transfer scheme (OT) [38, 40]. Here the sender, Alice, possesses two private messages m_0 and m_1, while the receiver, Bob, holds a private binary choice $c \in \{0,1\}$. Through the protocol, Bob gains knowledge of the message m_c, while remaining oblivious to the other message m_{1-c}. On the other hand, Alice remains unaware of the choice c. This scheme can be modeled by introducing a trusted third-party Colin [4] as follows (Fig. 10a):

(1) Colin generates two random bit arrays r_0, r_1 and a random bit d. He then sends r_0, r_1 to Alice and d, r_d to Bob.
(2) Bob encrypts his choice as $e := d \text{ xor } c$ and sends e to Alice.
(3) Alice computes two bit arrays $f_0 := m_0 \text{ xor } r_e, f_1 := m_1 \text{ xor } r_{1-e}$ and sends them to Bob.
(4) Bob retrieves his chosen message by computing $m' := f_c \text{ xor } r_d$.

As Alice and Bob have different perspectives, their correctness proofs require different pre/post-conditions. For instance, we prove that the message m_{1-c} is non-leaking to Bob by showing that his received messages (d, r_d, f_0, f_1) has a fixed distribution regardless of the values of the message m_{1-c} (yet may depend on the other message m_c):

$$\begin{array}{ll}\text{The case } c = 0: & \forall \hat{s}_0 \exists \vec{v} \forall \hat{s}_1.\ \{\langle c = 0\rangle \boxtimes \langle m_0 = \hat{s}_0\rangle \boxtimes \langle m_1 = \hat{s}_1\rangle\}\ \Gamma\ \{d, r_d, f_0, f_1 = \vec{v}\}\\ \text{The case } c = 1: & \forall \hat{s}_1 \exists \vec{v} \forall \hat{s}_0.\ \{\langle c = 1\rangle \boxtimes \langle m_0 = \hat{s}_0\rangle \boxtimes \langle m_1 = \hat{s}_1\rangle\}\ \Gamma\ \{d, r_d, f_0, f_1 = \vec{v}\}\end{array}$$

Here the precondition assumes that m_c is deterministic (yet arbitrary) as Bob can decode it at the end. We also prove that $\langle m' = m_c\rangle$, i.e. Bob successfully receives his desired message.

On the other hand, we prove that c is non-leaking to Alice by showing that her received messages (r_0, r_1, e) has a fixed distribution regardless of the value of c:

$$\exists \vec{v} \forall a.\ \{\langle c = a\rangle\}\ \Gamma\ \{r_0, r_1, e = \vec{u}\}$$

Figure 10b contains the proof sketch of privacy correctness and computation correctness for Bob when $c = 0$ (the case $c = 1$ is similar). First, it is easy to show that distributions of r_0, r_1, d are uniform and pair-wise independent (line 4). We then utilize the Split rule on d to analyze the assignments of r_d, e, f_0, f_1 (lines 4-12). Accordingly, we prove on line 12 that d, r_d, f_0, f_1 satisfy:

$$\mathsf{U}(d) \boxtimes \mathsf{U}(f_1) \boxtimes (\mathsf{U}(r_d) \wedge \langle f_0 = r_d \text{ xor } m_0 \wedge m_0 = \hat{s}_0\rangle)$$

i.e. d, f_1, r_d are uniform and pairwise independent, and $f_0 = r_d \text{ xor } m_0$ (thus f_0 is also uniform since m_0 is deterministic). Accordingly, the joint distribution of (d, r_d, f_0, f_1) is independent of m_1:

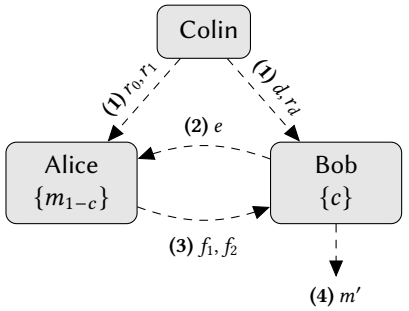
$$d, r_d, f_0, f_1 = \sum_{a,\hat{v},\hat{w}} 1/2^{2n+1}\ \langle a, \hat{v}, \hat{v} \text{ xor } \hat{s}_0, \hat{w}\rangle \qquad \text{where } a \in \{0,1\} \text{ and } \hat{v}, \hat{w} \in \{0,1\}^n$$

Similarly, the joint distribution of (d, r_d, f_0, f_1) is independent of m_0 when $c = 1$:

$$d, r_d, f_0, f_1 = \sum_{a,\hat{v},\hat{w}} 1/2^{2n+1}\ \langle a, \hat{v}, \hat{w}, \hat{v} \text{ xor } \hat{s}_1\rangle \qquad \text{where } a \in \{0,1\} \text{ and } \hat{v}, \hat{w} \in \{0,1\}^n$$

Hence, we conclude that the message m_{1-c} is non-leaking to Bob.

For computation correctness, observe that $\langle f_0 = m_0 \text{ xor } r_d\rangle$ (line 14). Since $c = 0$, the assignment $m' := f_0 \text{ xor } r_d$ implies $\langle m' = m_0\rangle$. Similarly, $\langle m' = m_1\rangle$ holds when $c = 1$. Hence $m' = m_c$.



$r_0 := (0.5 : 0) \oplus (0.5 : 1);$
$r_1 := (0.5 : 0) \oplus (0.5 : 1);$
$d := (0.5 : 0) \oplus (0.5 : 1);$
if $d = 0$ **then** $r_d := r_0$ **else** $r_d := r_1;$
$e := d$ xor $c;$
if $e = 0$ **then** $f_0 := m_0$ xor $r_0; f_1 := m_1$ xor r_1
else $f_0 := m_0$ xor $r_1; f_1 := m_1$ xor $r_0;$
if $c = 0$ **then** $m' := f_0$ xor r_d **else** $m' := f_1$ xor r_d

(a) Oblivious Transfer

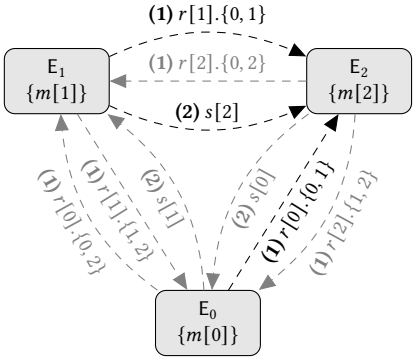
$\{\langle c = 0\rangle \boxtimes \langle m_0 = \hat{s}_0\rangle \boxtimes \langle m_1 = \hat{s}_1\rangle\}$//the case c = 0
$\searrow$ $\{1\}$
$r_0 := (0.5 : 0) \oplus (0.5 : 1); r_1 := (0.5 : 0) \oplus (0.5 : 1); d := (0.5 : 0) \oplus (0.5 : 1);$
$\{\mathsf{U}(r_0) \boxtimes \mathsf{U}(r_1) \boxtimes \mathsf{U}(d)\}$
if $d = 0$ **then** $r_d := r_0$ **else** $r_d := r_1;$
$\swarrow$ $\{0.5 \cdot \mathsf{U}(r_1) \boxtimes (\langle d, r_d = 0, r_0\rangle \wedge \mathsf{U}(r_0)) \quad \boxplus \quad 0.5 \cdot \mathsf{U}(r_0) \boxtimes (\langle d, r_d = 1, r_1\rangle \wedge \mathsf{U}(r_1))\}$
$\searrow$ $\{(0.5 \cdot \mathsf{U}(r_1) \boxtimes (\langle c, d, r_d = 0, 0, r_0\rangle \wedge \mathsf{U}(r_0)) \quad \boxplus \quad 0.5 \cdot (\ldots))\}$
$e := d$ xor $c;$
$\swarrow$ $\{0.5 \cdot \mathsf{U}(r_1) \boxtimes (\langle c, d, r_d, e = 0, 0, r_0, 0\rangle \wedge \mathsf{U}(r_0)) \boxplus 0.5 \cdot (\ldots))\}$
$\searrow$ $\{\langle m_0 = \hat{s}_0\rangle \boxtimes \langle m_1 = \hat{s}_1\rangle \boxtimes (0.5 \cdot \mathsf{U}(r_1) \boxtimes (\langle d, e, r_d = 0, 0, r_0\rangle \wedge \mathsf{U}(r_0)) \boxplus 0.5 \cdot (\ldots))\}$
if $e = 0$ **then** $f_0 := m_0$ xor $r_0; f_1 := m_1$ xor r_1 **else** $f_0 := m_0$ xor $r_1; f_1 := m_1$ xor $r_0;$
$\{\mathsf{U}(d) \boxtimes \mathsf{U}(f_1) \boxtimes (\mathsf{U}(r_d) \wedge \langle f_0 = r_d \text{ xor } m_0 \wedge m_0 = \hat{s_0}\rangle)\}$
$\swarrow$ $\left\{(d, r_d, f_0, f_1 = \sum_{a,\hat{v},\hat{w}} 1/2^{2n+1} \langle a, \hat{v}, \hat{v} \text{ xor } \hat{s}_0, \hat{w}\rangle) \wedge \langle f_0 = r_d \text{ xor } m_0\rangle\right\}$//privacy correctness
$\{\langle c = 0\rangle \boxtimes \langle f_0 = r_d \text{ xor } m_0\rangle\}$
if $c = 0$ **then** $m' := f_0$ xor r_d **else** $m' := f_1$ xor r_d
$\{\langle m' = m_0\rangle\}$//computation correctness

(b) Proof of computation correctness and privacy correctness for Bob when $c = 0$.

$\{\langle c = \hat{a}\rangle\}$
$r_0 := (0.5 : 0) \oplus (0.5 : 1); r_1 := (0.5 : 0) \oplus (0.5 : 1); d := (0.5 : 0) \oplus (0.5 : 1);$
$\{\langle c = \hat{a}\rangle \boxtimes \mathsf{U}(d) \boxtimes \mathsf{U}(r_0) \boxtimes \mathsf{U}(r_1)\}$
$\searrow$ $\{\langle c = \hat{a}\rangle \boxtimes \mathsf{U}(d)\}$
$e := d$ xor $c;$
$\swarrow$ $\{\langle c = \hat{a}\rangle \boxtimes (\mathsf{U}(d) \wedge \mathsf{U}(e))\}$
$\{\mathsf{U}(r_0) \boxtimes \mathsf{U}(r_1) \boxtimes \mathsf{U}(e)\}$//privacy correctness
...

(c) Proof of privacy correctness for Alice.

Fig. 10. Verification of Oblivious Transfer



for $i := [0,2]$ **do**
  $r[i].0 := (1/p : 0) \oplus \ldots \oplus (1/p : p-1)$;
  $r[i].1 := (1/p : 0) \oplus \ldots \oplus (1/p : p-1)$;
  $r[i].2 := (m[i] - r[i].0 - r[i].1)$ **mod** p;
for $i := [0,2]$ **do**
  $s[i] := (r[0].i + r[1].i + r[2].i)$ **mod** p;
$m' := (s[0] + s[1] + s[2])$ **mod** p

(a) Multi-Party Computation

1 $\{\langle m[0] = a_0\rangle \boxtimes \langle m[1] = a_1\rangle \boxtimes \langle m[2] = a_2\rangle\}$
2 **for** $i := [0,2]$ **do**
3 $r[i].0 := (1/p : 0) \oplus \ldots \oplus (1/p : p-1)$;
4 $r[i].1 := (1/p : 0) \oplus \ldots \oplus (1/p : p-1)$;
5 $r[i].2 := (m[i] - r[i].1 - r[i].2)$ **mod** p
6 $\left\{\boxtimes_{k=0}^{2} \left(\mathsf{U}(r[k].0) \boxtimes \mathsf{U}(r[k].1) \wedge \left\langle \wedge_{k=0}^{2} \; r[k].2 =_p a_k - r[k].0 - r[k].1\right\rangle\right) \boxtimes \langle m[0] = a_0\rangle \boxtimes \ldots\right\}$
7 **for** $i := [0,2]$ **do** $s[i] := (r[0].i + r[1].i + r[2].i)$ **mod** p;
8 $\left\{\left(\boxtimes_{k=0}^{2} \mathsf{U}(r[k].0) \boxtimes \mathsf{U}(r[k].1)\right) \wedge \left\langle s[2] = \sum_{k=0}^{2} (a_k - r[k].0 - r[k].1)\right\rangle \wedge \ldots\right\}$
9 $\left\{\mathsf{U}(s[2]) \boxtimes \left(\boxtimes_{k=0}^{1} \mathsf{U}(r[k].0) \boxtimes \mathsf{U}(r[k].1)\right) \wedge \ldots\right\}$ //privacy correctness
10 $\searrow \left\{\wedge_{k=0}^{2} \; \left\langle s[k] =_p \sum_{j=0}^{2} r[j].k\right\rangle \wedge \left\langle m[k] =_p \sum_{j=0}^{2} r[k].j\right\rangle\right\} \hookrightarrow F : \mathsf{U}(s[2]) \boxtimes \ldots$
11 $m' := (s[0] + s[1] + s[2])$ **mod** p
12 $\swarrow \{\langle m' = m[0] + m[1] + m[2]\rangle\}$ //computation correctness
13 ...

(b) Proof of computation correctness and privacy correctness for E_2.

Fig. 11. Verification of Multi-Party Computation

Fig. 10c outlines the privacy proof for Alice. On line 3, we show that r_0, r_1, d are uniform and pairwise independent. Next, we apply the IFrame rule to isolate r_0, r_1 and show that e, after the assignment, is uniform (line 6). Consequently, we infer that r_0, r_1, e are uniform and pairwise independent (line 7). By Lemma 7.3, their joint distribution is uniform and thus is independent of c.

7.4 Multi-Party Computation

In secured multi-party computation (MPC), multiple entities $\{\mathsf{E}_0, \ldots, \mathsf{E}_n\}$ want to compute a common function $f(x_0, \ldots, x_n)$ in which E_i holds the private input x_i. Fig. 11a is an instance of MPC in which each E_i in $\{\mathsf{E}_0, \mathsf{E}_1, \mathsf{E}_2\}$ holds a private integer $m[i] > 0$ and the goal is to compute the sum $m[0]+m[1]+m[2]$. Assume that they share a sufficiently large prime number $p > m[0]+m[1]+m[2]$. Then their goal can be achieved as follows:

(1) Each E_i generates two random number $r[i].0, r[i].1 \in \{0, \ldots, p-1\}$ and uses them to compute $r[i].2 := (m[i] - r[i].0 - r[i].1)$ **mod** p. Then each E_i receives $r[k].j$ if $i \notin \{k, j\}$.

(2) Each E_i computes $s[k] := (r(1).k + r(2).k + r(3).k)$ **mod** p where $k \in \{0,1,2\}\backslash\{i\}$. Then each E_i receives $s[i]$, e.g. from $\mathsf{E}_{(i-1) \bmod 3}$.
(3) Each E_i computes the sum as $m' := (s[0] + s[1] + s[2])$ **mod** p.

In Fig. 11b, we prove that $m[0]$ and $m[1]$ are non-leaking to E_2[7], i.e.:

$$\forall a_2 \exists \vec{v} \forall a_0 \forall a_1.\ \{\langle m[0] = a_0\rangle \boxtimes \langle m[1] = a_1\rangle \boxtimes \langle m[2] = a_2\rangle\}\, \Gamma\, \{\hat{c} = \vec{v}\}$$

where $\hat{c} = (r[0].0, r[0].1, r[1].0, r[1].1, s[2])$ contains the ciphertexts received by E_2. In fact, we show that the distribution of $\hat{c}$ is uniform which is sufficient to imply the existence of $\vec{v}$:

$$\mathsf{U}(r[0].0) \boxtimes \mathsf{U}(r[0].1) \boxtimes \mathsf{U}(r[1].0) \boxtimes \mathsf{U}(r[1].1) \boxtimes \mathsf{U}(s[2])$$

For computation correctness, we prove $\langle m' = m[0] + m[1] + m[2]\rangle$, i.e. E_2 computes the correct sum. For convenience, we write $e_1 =_p e_2$ as a shorthand for $e_1 = e_2$ **mod** p.

In Fig. 11b, after the assignments of $\{r[i].j\}_{i,j=0}^{2}$, it can be inferred that the variables in the set $K = \bigcup_{k=0}^{2}\{r[k].0, r[k].1\}$ have uniform distributions and are pairwise independent (line 6):

$$\boxtimes_{k=0}^{2}\ (\mathsf{U}(r[k].0) \boxtimes \mathsf{U}(r[k].1))$$

Since these variables are not modified hereafter, their distributions and independence relations are maintained throughout the program. Following the assignments of $\{s[i]\}_{i=0}^{2}$, it can be inferred (line 8) that $s[2], r[k].0$ satisfies the following predicate, where $A_k \triangleq a_k - r[k].0 - r[k].1$:

$$\left\langle s[2] =_p A_0 + A_1 + A_2 \right\rangle \wedge \left(\boxtimes_{k=0}^{2}\ (\mathsf{U}(r[k].0) \boxtimes \mathsf{U}(r[k].1))\right)$$

We then utilize the following result, where $b = r[2].0$, to establish that $s[2]$ is uniform and independent of $(r[0].0, r[0].1, r[1].0, r[1].1)$, thus completing the privacy proof for $s[2]$ (line 9):

Lemma 7.4. *Let $n, p \in \mathbb{N}_{>0}$, $s \in \mathbb{N}$, and $(a_i)_{i=1}^{n}, b, c$ are random variables drawn from the finite sample set $S = \{0, \ldots, p-1\}$. Then for every distribution vector $\vec{v}$ of $\hat{a} = (a_1, \ldots, a_n)$, we have:*

$$(\hat{a} = \vec{v}) \boxtimes \mathsf{U}(b) \wedge \left\langle c =_p s - b - (a_1 + \ldots + a_n)\right\rangle \vdash (\hat{a} = \vec{v}) \boxtimes \mathsf{U}(c).$$

For computation correctness, note that $s[k]$ and $m[k]$ satisfy the following predicates (lines 10):

$$\left\langle s[k] =_p r[0].k + r[1].k + r[2].k\right\rangle \quad \text{and} \quad \left\langle m[k] =_p r[k].0 + r[k].1 + r[k].2\right\rangle$$

As a result, $m' =_p m[0] + m[1] + m[2]$. As $p > m[i] + m[1] + m[2]$, we can deduce that $m' = m[0] + m[1] + m[2]$. Hence, E_2 computes the correct sum.

8 Related Work

Our work integrates local reasoning and termination reasoning for probabilistic programs within a unified framework. Our original inspiration came from Barthe, Hsu, and Liao [5], who presented Probabilistic Separation Logic (PSL) based on the logic of Bunched Implications (BI) [13, 35, 36] by treating "probabilistic independence as separation". Nevertheless, our work diverges from theirs in several key aspects. We incorporate termination reasoning, focusing on total correctness as opposed to their emphasis on partial correctness. Our framework supports the reasoning of both probabilistic dependence and independence, whereas they focus only on independence. Our IFrame rule is notably simpler than theirs, requiring fewer side conditions. Lastly, as discussed in §7.1, probabilistic noninterference (NP) and input independence (II) are not equivalent in their framework, requiring distinct methods for resolution; notably, they were unable to verify II for the OT scheme in §7.3. In contrast, PN and II are shown to be equivalent in our framework, allowing for the transformation of the proof of one into the proof of the other with ease.

[7]The other two cases for E_0 and E_1 are similar.

Barthe *et al.* [5] has inspired a number of papers beyond ours. Bao *et al.* expanded the baseline logic to reason about negative dependence [2] and conditional independence [3]. Li *et al.* introduced modal operators to capture conditional probabilistic behaviors for the verification of probabilistic programs with pointers and continuous distributions. Slightly earlier, Batz *et al.* introduced Quantitative Separation Logic [6], which combines SL and weakest preexpectations [30] for the backward reasoning of probabilistic programs with pointers. Predicates are evaluated as real numbers that represent expectations, and the frame rule only provides a lower bound. In contrast, our logic is designed for forward reasoning and our predicates are propositional, allowing us capture the exact distributions of the program states. Tassarotti and Harper proposed a relational separation program logic for concurrent probabilistic programs with data structures [41]. All of these ultimately build on the Separation Logic of O'Hearn, Ishtiaq, and Reynods [17, 32, 39] to enable local reasoning.

Another thread in the related work is proving almost-sure termination. One common approach involves constructing the super-martingales, stochastic processes whose expected values are non-increasing over the loop iterations (*e.g.* McIver *et al.* [28] and Chatterjee *et al.* [8, 9]). The invariant predicates $P(n)$ and $Q(n)$ in our While rule exhibit similar behavior to super-martingales. In $P(n)$, the probability mass of the distribution decreases with each iteration, while in $Q(n)$, the probability mass increases and is utilized to verify the postcondition of the loop. Another paper by Barthe *et al.* proposed an assertion-based logic for analyzing expectations and termination [4]. Their predicates are evaluated to sets of distributions similar to ours; however, their framework does not contain a viable treatment for local reasoning as ours does. Gregersen *et al.* introduced a refinement logic for proving almost-sure termination of higher-order probabilistic programs [14]. Other work (*e.g.* [7, 26, 31]) offer techniques for proving positive almost-sure termination, a stronger termination condition that requires the expected number of transitions in the program to be finite.

Denotational semantics is commonly used to model the semantics of probabilistic programs (*e.g.* Kozen [23, 24], Ehrhard *et al.* [11], and Vákár *et al.* [42]). In contrast, our work uses operational semantics, where the step relation exists between two sets of distributions over program configurations. He *et al.* [19] and McIver [27] presented operational semantics for probabilistic programs based on expectations. More recently, Olmedo and Kaminski *et al.* [20, 34] introduced a wp-style calculus for obtaining bounds on the expected run-time of probabilistic programs.

9 Conclusion and Future Work

We presented PRISM, a program logic for probabilistic programs that supports both local reasoning and probabilistic termination reasoning. We support two frame rules: IFrame for independent distributions and DFrame for dependent distributions. We also support locality for dependent distributions with our Dot, Split, and ISplit rules. Our SWhile rule verifies typical almost-sure termination loops in a way that is analogous to the while rule for total correctness in traditional Hoare logic. Our While rule can reason about complex termination scenarios using an indexed pair of predicate invariants $P(n)$ and $Q(n)$, with the former gradually "consumed" in each iteration to generate the latter, which is then utilized to establish the post-distribution and termination probability of the loop. We introduced the multiverse model for interpreting probabilistic states along with a variation of BI for PRISM's assertion language. We provided a semantics for our Hoare triple using suprema/limits. We applied our framework to prove probabilistic termination on various loops and various privacy and security protocols.

We have identified several directions for future work. Firstly, we aim to expand our logic to handle probabilistic programs with pointers and/or other program types like concurrent programs and quantum programs. Secondly, we intend to incorporate program statements such as non-deterministic choice, sampling from continuous distributions, or "observing" in probabilistic inference. Lastly, we plan to enhance our logic to address reasoning about conditional independence.

References

[1] Sheshansh Agrawal, Krishnendu Chatterjee, and Petr Novotný. 2017. Lexicographic Ranking Supermartingales: An Efficient Approach to Termination of Probabilistic Programs. *Proc. ACM Program. Lang.* 2, POPL, Article 34 (dec 2017), 32 pages. https://doi.org/10.1145/3158122

[2] 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

[3] Jialu Bao, Marco Gaboardi, Justin Hsu, and Joseph Tassarotti. 2022. A separation logic for negative dependence. *Proc. ACM Program. Lang.* 6, POPL, Article 57 (jan 2022), 29 pages. https://doi.org/10.1145/3498719

[4] Gilles Barthe, Thomas Espitau, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. 2018. An Assertion-Based Program Logic for Probabilistic Programs. In *Programming Languages and Systems - 27th European Symposium on Programming, ESOP 2018 (Lecture Notes in Computer Science, Vol. 10801)*, Amal Ahmed (Ed.). Springer, 117–144.

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

[6] Kevin Batz, Benjamin Lucien Kaminski, Joost-Pieter Katoen, Christoph Matheja, and Thomas Noll. 2019. Quantitative separation logic: a logic for reasoning about probabilistic pointer programs. *Proc. ACM Program. Lang.* 3, POPL (2019), 34:1–34:29.

[7] Olivier Bournez and Florent Garnier. 2005. Proving Positive Almost-Sure Termination. In *Term Rewriting and Applications*, Jürgen Giesl (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 323–337.

[8] Krishnendu Chatterjee, Amir Kafshdar Goharshady, Tobias Meggendorfer, and Đorđe Žikelić. 2022. Sound and Complete Certificates for Quantitative Termination Analysis of Probabilistic Programs. In *Computer Aided Verification: 34th International Conference, CAV 2022, Haifa, Israel, August 7–10, 2022, Proceedings, Part I* (Haifa, Israel). Springer-Verlag, Berlin, Heidelberg, 55–78. https://doi.org/10.1007/978-3-031-13185-1_4

[9] Krishnendu Chatterjee, Petr Novotný, and Đorđe Žikelić. 2017. Stochastic invariants for probabilistic termination. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages* (Paris, France) *(POPL '17)*. Association for Computing Machinery, New York, NY, USA, 145–160. https://doi.org/10.1145/3009837.3009873

[10] 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

[11] Thomas Ehrhard, Michele Pagani, and Christine Tasson. 2017. Measurable Cones and Stable, Measurable Functions: A Model for Probabilistic Higher-Order Programming. *Proc. ACM Program. Lang.* 2, POPL, Article 59 (Dec. 2017), 28 pages. https://doi.org/10.1145/3158147

[12] Luis María Ferrer Fioriti and Holger Hermanns. 2015. Probabilistic Termination: Soundness, Completeness, and Compositionality. *SIGPLAN Not.* 50, 1 (jan 2015), 489–501. https://doi.org/10.1145/2775051.2677001

[13] Alexander V. Gheorghiu, Tao Gu, and David J. Pym. 2024. Inferentialist Resource Semantics. arXiv:2402.09217 [cs.LO] https://arxiv.org/abs/2402.09217

[14] Simon Oddershede Gregersen, Alejandro Aguirre, Philipp G. Haselwarter, Joseph Tassarotti, and Lars Birkedal. 2024. Almost-Sure Termination by Guarded Refinement. arXiv:2404.08494 [cs.LO]

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

[16] C. A. R. Hoare. 1969. An axiomatic basis for computer programming. *Commun. ACM* 12, 10 (oct 1969), 576–580. https://doi.org/10.1145/363235.363259

[17] Samin S. Ishtiaq and Peter W. O'Hearn. 2001. BI as an Assertion Language for Mutable Data Structures. In *Proceedings of the 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (London, United Kingdom) *(POPL '01)*. Association for Computing Machinery, New York, NY, USA, 14–26. https://doi.org/10.1145/360204.375719

[18] Aaron D. Jaggard, Catherine Meadows, Michael Mislove, and Roberto Segala. 2010. Reasoning about Probabilistic Security Using Task-PIOAs. In *Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security*, Alessandro Armando and Gavin Lowe (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 2–22.

[19] He Jifeng, K. Seidel, and A. McIver. 1997. Probabilistic Models for the Guarded Command Language. *Sci. Comput. Program.* 28, 2–3 (April 1997), 171–192. https://doi.org/10.1016/S0167-6423(96)00019-6

[20] Benjamin Lucien Kaminski, Joost-Pieter Katoen, Christoph Matheja, and Federico Olmedo. 2018. Weakest Precondition Reasoning for Expected Runtimes of Randomized Algorithms. *J. ACM* 65, 5, Article 30 (Aug. 2018), 68 pages. https://doi.org/10.1145/3208102

[21] Jonathan Katz and Yehuda Lindell. 2007. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC.

[22] Barbara Kordy, Marc Pouly, and Patrick Schweitzer. 2016. Probabilistic reasoning with graphical security models. *Information Sciences* 342 (2016), 111–131. https://doi.org/10.1016/j.ins.2016.01.010

[23] Dexter Kozen. 1979. Semantics of Probabilistic Programs *(SFCS '79)*. IEEE Computer Society, USA, 101–114. https://doi.org/10.1109/SFCS.1979.38

[24] Dexter Kozen. 1983. A Probabilistic PDL. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing (STOC '83)*. Association for Computing Machinery, New York, NY, USA, 291–297. https://doi.org/10.1145/800061.808758

[25] C. Louizos. 2022. *Probabilistic reasoning for uncertainty & compression in deep learnin*. University of Amsterdam. https://pure.uva.nl/ws/files/67639124/Thesis.pdf

[26] Rupak Majumdar and V. R. Sathiyanarayana. 2024. Positive Almost-Sure Termination: Complexity and Proof Rules. *Proc. ACM Program. Lang.* 8, POPL, Article 37 (jan 2024), 29 pages. https://doi.org/10.1145/3632879

[27] A.K. McIver and Carroll Morgan. 2001. Partial correctness for probabilistic demonic programs. *Theoretical Computer Science* 266, 1-2 (2001), 513–541.

[28] Annabelle McIver, Carroll Morgan, Benjamin Lucien Kaminski, and Joost-Pieter Katoen. 2017. A New Proof Rule for Almost-Sure Termination. *Proc. ACM Program. Lang.* 2, POPL, Article 33 (dec 2017), 28 pages. https://doi.org/10.1145/3158121

[29] B.C. Montalto. 2014. *Equivalence Properties and Probabilistic Reasoning in Symbolic Security Protocol Analysis*. ETH-Zürich. https://books.google.com.sg/books?id=lEgYrgEACAAJ

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

[31] Van Chan Ngo, Quentin Carbonneaux, and Jan Hoffmann. 2018. Bounded expectations: resource analysis for probabilistic programs. In *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation* (Philadelphia, PA, USA) *(PLDI 2018)*. Association for Computing Machinery, New York, NY, USA, 496–512. https://doi.org/10.1145/3192366.3192394

[32] Peter O'Hearn, John Reynolds, and Hongseok Yang. 2001. Local Reasoning about Programs that Alter Data Structures. In *Computer Science Logic*, Laurent Fribourg (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–19.

[33] Peter W. OHearn. 2007. Resources, Concurrency, and Local Reasoning. *Theor. Comput. Sci.* 375, 1–3 (apr 2007), 271–307. https://doi.org/10.1016/j.tcs.2006.12.035

[34] Federico Olmedo, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. 2016. Reasoning about Recursive Probabilistic Programs. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science* (New York, NY, USA) *(LICS '16)*. Association for Computing Machinery, New York, NY, USA, 672–681. https://doi.org/10.1145/2933575.2935317

[35] David J. Pym. 2002. The semantics and proof theory of the logic of bunched implications. In *Applied logic series*.

[36] David J. Pym, Peter W. O'Hearn, and Hongseok Yang. 2004. Possible Worlds and Resources: The Semantics of BI. *Theor. Comput. Sci.* 315, 1 (may 2004), 257–305. https://doi.org/10.1016/j.tcs.2003.11.020

[37] Meng Qu and Jian Tang. 2019. *Probabilistic Logic Neural Networks for Reasoning*. Curran Associates Inc., Red Hook, NY, USA.

[38] Michael O. Rabin. 2005. How To Exchange Secrets with Oblivious Transfer. *IACR Cryptol. ePrint Arch.* (2005), 187. http://eprint.iacr.org/2005/187

[39] John C. Reynolds. 2002. Separation Logic: A Logic for Shared Mutable Data Structures. In *Proceedings of the 17th Annual IEEE Symposium on Logic in Computer Science (LICS '02)*. IEEE Computer Society, USA, 55–74.

[40] Ronald L. Rivest. 1999. *Unconditionally Secure Commitment and Oblivious Transfer Schemes Using Private Channels and a Trusted Initializer*. Technical Report.

[41] Joseph Tassarotti and Robert Harper. 2019. A separation logic for concurrent randomized programs. *Proc. ACM Program. Lang.* 3, POPL (2019), 64:1–64:30.

[42] Matthijs Vákár, Ohad Kammar, and Sam Staton. 2019. A Domain Theory for Statistical Probabilistic Programming. 3, POPL, Article 36 (Jan. 2019), 29 pages. https://doi.org/10.1145/3290349

[43] John von Neumann. 1951. Various Techniques Used in Connection with Random Digits. In *Monte Carlo Method*, A. S. Householder, G. E. Forsythe, and H. H. Germond (Eds.). National Bureau of Standards Applied Mathematics Series, Vol. 12. US Government Printing Office, Washington, DC, Chapter 13, 36–38.

[44] Xia Zeng, Zhengfeng Yang, Li Zhang, Xiaochao Tang, Zhenbing Zeng, and Zhiming Liu. 2023. Safety Verification of Nonlinear Systems with Bayesian Neural Network Controllers. *Proceedings of the AAAI Conference on Artificial Intelligence* 37, 12 (Jun. 2023), 15278–15286. https://doi.org/10.1609/aaai.v37i12.26782

[45] Yuyu Zhang, Xinshi Chen, Yuan Yang, Arun Ramamurthy, Bo Li, Yuan Qi, and Le Song. 2020. Efficient Probabilistic Logic Reasoning with Graph Neural Networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=rJg76kStwH

In Section A, the readers can find the definitions for $\mathsf{fv}(\phi)$, $\mathsf{vars}(c)$, $\mathsf{mv}(c)$ along with the definitions and notations used in the appendix. The proofs of the lemmas used in the paper are detailed as follows: §B for lemmas in §4, §C for a more complete set of the entailment rules (including those in Figure 3) together with their soundness proofs, §D for lemmas in §5, §E for the complete inference rule set (Fig. 13) and their proofs, §F for lemmas in §7. Lastly, in §G, the general case of the OTP scheme mentioned in §2.4 is presented.

A Definitions and notations

We define $\mathsf{fv}(\phi)$ — the set of free variables in ϕ, $\mathsf{vars}(c)$ — the set of program variables in c, $\mathsf{mv}(c)$ — the set of modified variables in c — as follows:

$$\mathsf{fv}([W]) \triangleq \mathsf{fv}(W) \qquad \mathsf{fv}(\phi) \triangleq \emptyset \text{ for } \phi \in \{\rho, \top, \bot\} \qquad \mathsf{fv}(\rho_{\downarrow\phi}) \triangleq \mathsf{fv}(\phi)$$
$$\mathsf{fv}(\phi * \psi) \triangleq \mathsf{fv}(\phi) \cup \mathsf{fv}(\psi) \text{ for } * \in \{\boxtimes, \boxplus, -\!\!\boxtimes, -\!\!\boxplus\}$$
$$\mathsf{fv}(\phi/[W]) \triangleq \mathsf{fv}(\phi) \cup \mathsf{fv}(W) \qquad \mathsf{fv}(\forall x.\ \phi) = \mathsf{fv}(\exists x.\ \phi) \triangleq \mathsf{fv}(\phi)$$

$$\mathsf{vars}(\textbf{skip}) \triangleq \emptyset \qquad \mathsf{vars}(x := e) \triangleq \{x\} \cup \mathsf{fv}(e) \qquad \mathsf{vars}(c_1[e]c_2) \triangleq \mathsf{vars}(c_1) \cup \mathsf{vars}(c_2) \cup \mathsf{fv}(e)$$
$$\mathsf{vars}(c_1; c_2) \triangleq \mathsf{vars}(c_1) \cup \mathsf{vars}(c_2) \qquad \mathsf{vars}(\textbf{while } b \textbf{ do } c) \triangleq \mathsf{vars}(c) \cup \mathsf{fv}(b)$$

$$\mathsf{mv}(\textbf{skip}) \triangleq \emptyset \qquad \mathsf{mv}(x := e) \triangleq \{x\} \qquad \mathsf{mv}(c_1[e]c_2) \triangleq \mathsf{mv}(c_1) \cup \mathsf{mv}(c_2)$$
$$\mathsf{mv}(c_1; c_2) \triangleq \mathsf{mv}(c_1) \cup \mathsf{mv}(c_2) \qquad \mathsf{mv}(\textbf{while } b \textbf{ do } c) \triangleq \mathsf{mv}(c)$$

We define partial operator as follow, where $* \in \{+, \times\}$:

$$a \mathbin{\hat{*}} b \triangleq \begin{cases} 0 & \text{, if both } a, b \text{ are undefined} \\ a & \text{, if } a\downarrow \text{ and } b \text{ is undefined} \\ b & \text{, if } b\downarrow \text{ and } a \text{ is undefined} \\ a * b & \text{, otherwise.} \end{cases}$$

For convenience, we introduce/recall the following definitions and notations:

Definition 15. *Let $D \cap D' = \emptyset$, D-state σ, D'-map m', D'-config $\kappa = \oplus_{i\in I}\ \delta(\langle c_i, m_i\rangle, \rho_i)$. Then:*

1. *The variable domain of σ is $\mathsf{vdom}(\sigma) \triangleq D$ and of κ is $\mathsf{vdom}(\kappa) \triangleq D'$.*
2. *We override $\sigma \otimes m'$ to be the $(D \cup D')$-state lifted from $\lambda m \in \mathsf{fmap}(D).\ m \circ m'$.*
3. *We override $\mathsf{mv}(\kappa) \triangleq \bigcup_{i\in I} \mathsf{mv}(c_i)$.*
4. *We override $\sigma \otimes \kappa \triangleq \oplus_{i\in I}\ \langle(\rho_i \bullet \sigma) \otimes m_i, c_i\rangle$.*
5. *Let $S \subseteq D'$. We override $\pi(\kappa, S) \triangleq \oplus_{i\in I}\ \delta(\langle c_i, \pi(m_i, S)\rangle, \rho_i)$.*
6. *The state snapshot of κ is the D'-state $\lfloor\kappa\rfloor \triangleq \oplus_{i\in I}\ \delta(m_i, \rho_i)$.*
7. *A sequence of config $(\kappa_i)_{i=0}^n$ where $n \in \mathbb{N} \cup \{\infty\}$ is an induced run of a config κ if $\kappa_0 = \kappa$ and $\kappa_i \rightsquigarrow \kappa_{i+1}$ for every $i < n$.*

B Proofs of lemmas in Section 4

We often use the following result as an alternative definition for supremum:

Proposition B.1. *Given a $\preceq$-ordered sequence of Ω-multiverses. Then for every Ω-multiverse u:*

1. *If $u = \sup\,(u_i)_{i=0}^n$ then $\|u\| \le 1$.*
2. *$u = \sup\,(u_i)_{i=0}^n$ iff $u_i \preceq u$ for every $i \le n$ and for every $\epsilon > 0$, there exists some $i \le n$ such that $\|u_i\| > \|u\| - \epsilon$.*

Proof. If n is finite then $u = u_n$ and we are done. Otherwise, we assume $n = \infty$.

(1) Suppose $\|u\| > 1$. Thus there exists some finite set $S \subseteq \mathrm{dom}(u)$ s.t. $A \triangleq \sum_{w\in S} u(w) > 1$. Let $\epsilon \triangleq (A-1)/(|S|+1) > 0$. For every $w \in S$, observe that $(u_i(w))_{i=0}^{\infty}$ is non-decreasing and $\sup(u_i(w))_{i=0}^{\infty} = u(w)$. As a result, there exists some n_w s.t. $u_n(w) > u(w) - \epsilon$ for every $n \geq n_w$. Accordingly, if we let $n \triangleq \max\{n_w \mid w \in S\}$ then $u_n(w) > u(w) - \epsilon$ for every $w \in S$. Thus:

$$1 \;\geq\; \|u_n\| \;\geq\; \sum_{w\in S} u_n(w) \;>\; \sum_{w\in S} u(w) - \epsilon \times (|S|+1) \;=\; 1$$

This is a contradiction. Hence $\|u\| \leq 1$.

(2) $\Rightarrow$ direction. Note that by definition of the supremum, $u_k(w) \leq u(w)$ holds for every k and $w \in \mathrm{dom}(u_k)$. Thus $u_k \preceq u$ for every k.
For the sake of contradiction, assume that there exists some $\epsilon > 0$ s.t. $\|u_k\| \leq \|u\| - \epsilon$ for every $k \leq n$. Since $u_k \preceq u$, we can find some u'_k s.t. $u = u_k \oplus u'_k$. It follows that:

$$\left\|u'_k\right\| = \|u\| - \|u_k\| \geq \epsilon \quad \text{for every } k$$

On the other hand, let $S \subseteq \mathrm{dom}(u)$ be finite s.t. $\sum_{w\in S} u(w) > \|u\| - \epsilon/2$. Then:

$$\sum_{w\in S'} u(w) < \epsilon/2 \quad \text{where } S^C \triangleq \mathrm{dom}(u'_m)\backslash S \tag{1}$$

Now we pick $\epsilon' \triangleq \epsilon/2(|S|+1) > 0$. As $u = \sup\{u_n\}_{n=0}^{\infty}$, there exists some u_m (i.e. by a similar argument as in part (a)) such that:

$$u_m(w) > u(w) - \epsilon' \quad \text{for every } w \in S \tag{2}$$

Pick u'_m such that $u_m \oplus u'_m = u$. Then:

$$\begin{array}{rcl} \epsilon \leq \left\|u'_m\right\| & = & \sum_{w\in S} u'_m(w) + \sum_{w\in S^c} u'_m(w) \\ & \leq & \sum_{w\in S} u(w) - \sum_{w\in S} u_m(w) + \sum_{w\in S^C} u(w) \\ & < & (|S|+1) \times \epsilon' + \epsilon/2 \\ & = & \epsilon \end{array}$$

This is a contradiction and so we are done.
$\Leftarrow$ direction. It suffices to show that for every $w \in \mathrm{dom}(u)$, we have:

$$u(w) \;=\; \sup(u_i(w))_{i=0}^{\infty}$$

Since $u_i \preceq u$ for every i, it follows that:

$$u_i(w) \;\leq\; u(w) \quad \text{and so} \quad \sup(u_i(w))_{i=0}^{\infty} \;\leq\; u(w)$$

Suppose that $\sup(u_i(w))_{i=0}^{\infty} < u(w)$. Let:

$$\epsilon \;\triangleq\; u(w) - \sup(u_i(w))_{i=0}^{\infty} \;>\; 0$$

There exists some u_n s.t. $\|u_n\| > \|u\| - \epsilon$. Thus:

$$\epsilon \;>\; \|u\| - \|u_n\| \;\geq\; u(w) - u_n(w) \;\geq\; u(w) - \sup\{u_i(w)\}_{i=0}^{\infty} \;=\; \epsilon$$

This is a contradiction. Hence $u(w) = \sup(u_i(w))_{i=0}^{\infty}$.

□

Proof of Lemma 4.1. We need to check that the result mapping u satisfies (1) $u(m) > 0$ for every $m \in \mathrm{dom}(u)$ and (2) $\|u\| \leq 1$. The first condition is easy to verify. For the second condition, it is not hard to deduce that $\|u_1 \otimes u_2\| = \|u_1\| \times \|u_2\|$, $\|\rho \bullet u\| = \rho\,\|u\|$, $\left\|\oplus_{i=0}^{n} u_i\right\| = \sum_{i=0}^{n} \|u_i\|$, $\|f(u)\| = \|u\|$, $\|u/[W]\| \leq \|u\|$, and $\left\|\sup(u_i)_{i=0}^{n}\right\| = \sup(\|u_i\|)_{i=0}^{n}$. In each equality/inequality, the RHS is at most 1. Hence, we are done. □

Proof of Lemma 4.2. We only need to check the last property as the others are easy to verify. Assume $u_1 \sqsubseteq u_2$ and both $u \otimes u_1, u \otimes u_2$ are defined. Clearly, both u_1, u_2 are non-empty. Let $u_1 = \oplus_{i \in I}(\delta(w_i, \rho_i))$ and $u_2 = \oplus_{i \in I}(\delta(w'_i, \rho_i))$ where $w_i \sqsubseteq_W w'_i$ for every $i \in I$. Also, assume $u = \oplus_{j \in J}(\delta(v_j, \alpha_j))$. Then $u \otimes u_1 = \oplus_{j \in J, i \in I}(\delta(v_j, \alpha_j)) \otimes (\delta(w_i, \rho_i)) = \oplus_{j \in I, i \in I}(\delta(v_j \circ w_i, \alpha_j \rho_i))$. Similarly, $u \otimes u_2 = \oplus_{j \in I, i \in I}(\delta(v_j \circ w'_i, \alpha_j \rho_i))$. Clearly, $v_j \circ w_i \sqsubseteq_M v_j \circ w'_i$ for every $j \in J, i \in I$. Hence, $u \otimes u_1 \sqsubseteq u \otimes u_2$. □

Proof of Lemma 4.3. The properties for $\oplus$ are easy to verify. We prove the properties for $\otimes$ where each u_i is a non-empty Ω_i-multiverse.

(1) $\otimes$-iden. Assume u is a non-empty Ω-multiverse. Note that $\Omega \perp \{e\}$ and for, for every $w \in \Omega$, we have:

$$u \otimes u_e(w) \;=\; u \otimes u_e(w \circ e) \;=\; u(w) \times u_e(e) \;=\; u(w)$$

Thus, $u \otimes u_e = u$. Similarly, $u_e \otimes u = u$.

(2) $\otimes$-assoc. Note that $\circ$ is associative, so $(\Omega_1 \circ \Omega_2) \circ \Omega_3 = \Omega_1 \circ (\Omega_2 \circ \Omega_3)$ and:

$$\Omega_1 \perp \Omega_2 \wedge \Omega_1 \circ \Omega_2 \perp \Omega_3 \qquad \text{iff} \qquad \Omega_2 \perp \Omega_3 \wedge \Omega_1 \perp \Omega_2 \circ \Omega_3$$

Let $(w_1 \circ w_2) \circ w_3 \in (\Omega_1 \circ \Omega_2) \circ \Omega_3$ where $w_i \in \Omega_i$. Then:

$$\begin{array}{rcl} [(u_1 \circ u_2) \circ u_3]\,((w_1 \circ w_2) \circ w_3) & = & u_1(w_1) \times u_2(w_2) \times u_3(w_3) \\ & = & [u_1 \circ (u_2 \circ u_3)]\,(w_1 \circ (w_2 \circ w_3)) \end{array}$$

Thus $(u_1 \circ u_2) \circ u_3 = u_1 \circ (u_2 \circ u_3)$.

(3) $\otimes$-canc. Let $\Omega_1 \perp \Omega_3$, $\Omega_2 \perp \Omega_3$ and $u_1 \otimes u_3 = u_2 \otimes u_3$ where $u_i \in \Omega_i$. It follows that $\Omega_1 \circ \Omega_3 = \Omega_2 \circ \Omega_3$. We will show that $\Omega_1 = \Omega_2$. Let $w_1 \in \Omega_1$ and $w_3 \in \Omega_3$. Then $w_1 \circ w_3 \in \Omega_1 \circ \Omega_3 = \Omega_2 \circ \Omega_3$. Thus there exist $w_2 \in \Omega_2$ and $w'_3 \in \Omega_3$ s.t. $w_1 \circ w_3 = w_2 \circ w'_3$. As $\Omega_2 \perp \Omega_3$, we have $w_1 = w_2$ and $w_3 = w'_3$. Thus, $w_1 \in \Omega_2$ which implies that $\Omega_1 \subseteq \Omega_2$. Similarly, $\Omega_2 \subseteq \Omega_1$. Hence $\Omega_1 = \Omega_2$.
Now let $w_1 \in \Omega_1$ and $w \in \Omega_3$. Then:

$$u_1(w_1) \times u_3(w_3) \;=\; [u_1 \otimes u_3](w_1 \circ w_3) \;=\; [u_2 \otimes u_3](w_1 \circ w_3) \;=\; u_2(w_1) \times u_3(w_3)$$

Thus, $u_1(w_1) = u_2(w_1)$ for every $w_1 \in \Omega_1$. Hence, $u_1 = u_2$.

(4) $\otimes$-norm. We have:

$$\begin{array}{rcl} \|u_1 \otimes u_2\| & = & \sum_{w_1 \in \Omega_1, w_2 \in \Omega_2} [u_1 \otimes u_2]\,(w_1 \circ w_2) \\ & = & \sum_{w_1 \in \Omega_1, w_2 \in \Omega_2} u_1(w_1) \times u_2(w_2) \\ & = & \left(\sum_{w_1 \in \Omega_1} u_1(w_1)\right) \times \left(\sum_{w_2 \in \Omega_2} u_2(w_2)\right) \\ & = & \|u_1\| \times \|u_2\| \end{array}$$

(5) $\otimes$-dist. Let Ω, Ω' be the non-empty domains of u, u', respectively. Note that $\Omega = \Omega_1 \cup \Omega_2$ and $\Omega' \perp \Omega$. It follows that $\Omega' \perp \Omega_1$ and $\Omega' \perp \Omega_2$. Also, $\Omega' \circ \Omega = (\Omega' \circ \Omega_1) \cup (\Omega' \circ \Omega_2)$. For every $w \in \Omega, w \in \Omega'$, we have:

$$\begin{array}{rcl} [u' \otimes u]\,(w' \circ w) & = & u'(w') \times u(w) \\ & = & u'(w') \times (u_1(w) \hat{+} u_2(w)) \\ & = & u'(w') \hat{\times} u_1(w) + u'(w') \hat{\times} u_2(w) \\ & = & [(u' \otimes u_1) \oplus (u' \otimes u_2)]\,(w' \circ w) \end{array}$$

where $\hat{+}, \hat{\times}$ are the partial addition and multiplication defined in §A
As a result, $u' \otimes u = (u' \otimes u_1) \oplus (u' \otimes u_2)$.

Hence, we are done. □

Proof of Lemma 4.4. Assume $\circ$ is commutative. Let Ω_1, Ω_2 be the non-empty domain of u_1, u_2, respectively. Then $\Omega_1 \perp \Omega_2$ iff $\Omega_2 \perp \Omega_1$ and $\Omega_1 \circ \Omega_2 = \Omega_2 \circ \Omega_1$. Furthermore, for every worlds $w_1 \in \Omega_1, w_2 \in \Omega_2$, we have:

$$[u_1 \otimes u_2]\,(w_1 \circ w_2) = u_1(w_1) \times u_2(w_2) = u_2(w_2) \times u_1(w_1) = [u_2 \otimes u_1]\,(w_2 \circ w_1)$$

Note that $w_1 \circ w_2 = w_2 \circ w_1$. Thus, $u_1 \otimes u_2 = u_2 \otimes u_1$. □

Proof of Corollary 4.5. Direct from Lemma 4.2 and Lemma 4.3. □

Proof of Lemma 4.6. We prove the following cases:

(1) Let $u \models \phi \boxplus (\phi \mathrel{-\!\!\boxplus} \psi)$. Accordingly, there exist u_1, u_2 such that $u_1 \oplus u_2 = u$ and $u_1 \models \phi$, $u_2 \models \phi \mathrel{-\!\!\boxplus} \psi$. By the semantics of $-\!\!\boxplus$, we have $u \models \psi$.
(2) Let $u \models \phi \boxtimes (\phi \mathrel{-\!\!\boxtimes} \psi)$. Accordingly, there exist u_1, u_2 such that $u_1 \boxtimes u_2 \sqsubseteq u$ and $u_1 \models \phi$, $u_2 \models \phi \mathrel{-\!\!\boxtimes} \psi$. By the semantics of $-\!\!\boxtimes$, $u_1 \boxtimes u_2 \models \psi$. By the Monotonicity Lemma 4.7, $u \models \psi$.

Hence, we are done. □

We need the following result to prove Lemma 4.7:

Proposition B.2. *Let $u \sqsubseteq u'$. Suppose $u = u_1 \oplus u_2$. Then there exist u_1', u_2' such that $u' = u_1' \oplus u_2'$ and $u_1 \sqsubseteq u_1'$, $u_2 \sqsubseteq u_2'$. Generally, suppose $u = \oplus_{i\in I} u_i$ for some $(u_i)_{i\in I}$. Then there exists $(u_i')_{i\in I}$ such that $u' = \oplus_{i\in I} u_i'$ and $u_i \sqsubseteq u_i'$ for every $i \in I$.*

Proof. First, we prove the case $u = u_1 \oplus u_2$. Suppose $u_1 = \oplus_{i\in I_1} \delta(w_i, \rho_i)$ and $u_2 = \oplus_{i\in I_2} \delta(w_i, \rho_i')$. Since $u \sqsubseteq u'$, there exist $(w_i)_{i\in I_1\cup I_2}$ and $(\rho_i'')_{i\in I_1\cup I_2}$ such that $w_i \sqsubseteq_M w_i'$ for every $i \in I_1 \cup I_2$ and $u = \oplus_{i\in I_1\cup I_2} \delta(w_i, \rho_i'')$, $u' = \oplus_{i\in I_1\cup I_2} \delta(w_i', \rho_i'')$. We define:

$$u_1' \triangleq \oplus_{i\in I_1} \delta(w_i', \rho_i) \qquad \text{and} \qquad u_2' \triangleq \oplus_{i\in I_2} \delta(w_i', \rho_i')$$

As a result, $u_1 \sqsubseteq u_1'$ and $u_2 \sqsubseteq u_2'$. As $u_1 \oplus u_2 = u$, we also have $u_1' \oplus u_2' = u'$. Thus the case $u = u_1 \oplus u_2$ is proven.

Using induction (i.e. on the sum's length), one can show that if $u = \oplus_{i=1}^n u_i$ then $u' = \oplus_{i=1}^n u_i'$ where $u_i \sqsubseteq u_i'$ for every $i \in [1, n]$. Now suppose I is finitely countable. W.l.o.g., we assume that $I = \mathbb{N}_{>0}$. Let $u = \oplus_{i=1}^\infty u_i$. Note that $u = \sup (s_n)_{n=1}^\infty$ where $s_n = \oplus_{i=1}^n u_i$. Let $\bigcup_{n=1}^\infty (u_i')_{i=1}^n$ be the infinite sequence where its finite subsequence$(u_i')_{i=1}^n$ is derived from $(u_i)_{i=1}^n$. It follows that $u' = \oplus_{i=1}^\infty u_i'$ and $u_i \sqsubseteq u_i'$ for every i. Hence, we are done. □

Proof of Lemma 4.7. Let $u \models \phi$ and $u \sqsubseteq u'$, we show that $u' \models \phi$. We apply structural induction on the formula ϕ:

(1) $\phi = [W]$. Let w' be a world of u'. Then there exists some world w of u such that $w \sqsubseteq_M w'$. As $w \models_M W$ and $\sqsubseteq_M$ satisfies the Kripke monotonicity, it follows that $w' \models_M W$. Thus $u' \models [W]$.
(2) $\phi = \rho$. By Lemma 4.3, we have $\|u\| = \|u'\|$. Thus $u \models \rho$ implies $u' \models \rho$.
(3) $\phi = \phi_1 \boxplus \phi_2$. Since $u \models \phi_1 \boxplus \phi_2$, there exist u_1, u_2 such that $u_1 \oplus u_2 = u$ and $u_1 \models \phi_1$, $u_2 \models \phi_2$. By Prop. B.2, there exist u_1', u_2' such that $u' = u_1' \oplus u_2'$ and $u_1 \sqsubseteq u_1'$, $u_2 \sqsubseteq u_2'$. By the induction hypothesis, we have $u_1' \models \phi_1$, $u_2' \models \phi_2$. Thus $u' \models \phi_1 \boxplus \phi_2$.
(4) $\phi = \boxplus_{i\in I} \phi_i$. Similar to the previous case.
(5) $\phi = \phi \mathrel{-\!\!\boxplus} \psi$. Let u_1', u_2' satisfy $u_1' \models \phi$ and $u' \oplus u_1' = u_2'$. Note that $\|u\| = \|u'\|$ and so $\|u\| + \|u_1'\| = \|u'\| + \|u_1'\| \leq 1$. Thus there exists some u_2 such that $u \oplus u_1' = u_2$. As $u \sqsubseteq u'$ and $u_1' \sqsubseteq u_1'$, we have $u_2 \sqsubseteq u_2'$. As $u \models \phi \mathrel{-\!\!\boxplus} \psi$ and $u_1' \models \phi$, we deduce that $u_2 \models \psi$. As a result, the induction hypothesis gives us $u_2' \models \psi$. Thus $u' \models \phi \mathrel{-\!\!\boxplus} \psi$.
(6) $\phi = \phi_1 \boxtimes \phi_2$. Let u_1, u_2 satisfy $u_1 \otimes u_2 \sqsubseteq u$ and $u_1 \models \phi_1$, $u_2 \models \phi_2$. As $\sqsubseteq$ is transitive, we also have $u_1 \otimes u_2 \sqsubseteq u'$. Thus $u' \models \phi_1 \boxtimes \phi_2$.

(7) $\phi = \phi_1 \mathrel{-\!\!\boxtimes} \phi_2$. Let u'_1, u'_2 satisfy $u' \otimes u'_1 = u'_2$ and $u'_1 \models \phi$. By Lemma 4.2, there exists u_2 such that $u \otimes u'_1 = u_2$ and $u_2 \sqsubseteq u'_2$. As $u \models \phi \mathrel{-\!\!\boxtimes} \psi$, and $u'_1 \models \phi$, we have $u_2 \models \psi$. Using the induction hypothesis, we deduce that $u'_2 \models \phi$. Thus $u' \models \phi \mathrel{-\!\!\boxtimes} \psi$.

(8) $\phi = \phi_1/[W]$. Let $u \models \phi_1/[W]$. Thus there exists some u_1 such that $u_1 \models \phi_1$ and $u = u_1/[W]$. Note that $u \preceq u_1$. Thus there exists some u_2 such that $u \oplus u_2 = u_1$ and none of the worlds in u_2 satisfies W. Note that $u \sqsubseteq u'$ and $u_2 \sqsubseteq u_2$. Thus $u \oplus u_2 \sqsubseteq u' \oplus u_2$, or equivalently $u_1 \sqsubseteq u'_1$ where $u'_1 \triangleq u' \oplus u_2$. Consequently, $u'_1 \models \phi_1$ by the induction hypothesis. Now, observe that $u_1 = u'_1/[W]$ since $u' \models [W]$ (by the induction hypothesis) and none of the worlds in u_2 satisfies W. Hence, $u' \models \phi_1/[W]$.

(9) $\phi = \phi_{1\downarrow\rho}$. Let $u \models \phi_{1\downarrow\rho}$. Thus there exists some $a \geq 0$ such that $\|u\| = \rho$ and $a \bullet u \models \phi$. Note that $a \bullet u \sqsubseteq a \bullet u'$ and so $a \bullet u' \models \phi$ by the induction hypothesis. Besides, $\|u\| = \|u'\|$. Consequently, $u' \models \phi_{1\downarrow\rho}$.

(10) The other cases are direct from the induction hypothesis.

Hence, we are done. □

Proof of Lemma 4.8. The case $\rho = 0$ is easy to verify. Thus we can assume that $\rho \in (0, 1]$ and so $\rho \cdot \phi$ is equivalent to $\rho \boxtimes \phi$.

For $\Rightarrow$ direction, let $u \models \rho \boxtimes \phi$. Thus there exist u_1, u_2 such that $u_1 \otimes u_2 \sqsubseteq u$ and $u_1 \models \rho$, $u_2 \models \phi$. As $\|u\| = \|u_1\| \times \|u_2\| \leq \rho$, there exists u' such that $u = \rho \bullet u'$. It follows that $u_2 \sqsubseteq u'$. By Lemma 4.7, we deduce that $u' \models \phi$.

For $\Leftarrow$ direction, let $u = \rho \bullet u'$ and $u' \models \phi$. Note that $u = u_1 \otimes u'$ where $u_1 = (\delta(e, \rho)) \models \rho$. As the relation $\sqsubseteq$ is reflexive, we deduce that $u \models \rho \boxtimes \phi$. □

Proof of Lemma 4.9. Direct from the definitions. □

Proof of Lemma 4.10. Direct from the fact that $(m_1 \circ m_2)\downarrow$ iff their domains are disjoint. □

Proof of Lemma 4.11. Let $\sigma_1 = \pi(\sigma, \hat{x})$, $\sigma_2 = \pi(\sigma, \hat{y})$, $\sigma_3 = \pi(\sigma, \hat{x} \cup \hat{y})$. By Lemma 4.12, we have $\sigma_1, \sigma_2 \sqsubseteq \sigma_3 \sqsubseteq \sigma$. It follows that $\hat{x}$ and $\hat{y}$ are independent iff $\sigma_1 \otimes \sigma_2 = \sigma_3$.

(1) $\Rightarrow$ (2). Note that $\sigma_1 \models \langle \hat{x} = \hat{x} \rangle$, $\sigma_2 \models \langle \hat{y} = \hat{y} \rangle$. As $\sigma_3 \sqsubseteq \sigma$, we have $\sigma \models [\langle \hat{x} = \hat{x} \rangle] \boxtimes \langle \hat{y} = \hat{y} \rangle$.

(2) $\Rightarrow$ (1). We have $\sigma \models \langle \hat{x} = \hat{x} \rangle \boxtimes \langle \hat{y} = \hat{y} \rangle$. Thus there exist σ_a, σ_b such that $\sigma_a \otimes \sigma_b \sqsubseteq \sigma$ and $\sigma_a \models \langle \hat{x} = \hat{x} \rangle$, $\sigma_b \models \langle \hat{y} = \hat{y} \rangle$. This implies that the variable domains of σ_a and of σ_b contain $\hat{x}$ and $\hat{y}$, respectively. As a result, we deduce that $\sigma_1 = \pi(\sigma_a, \hat{x})$, $\sigma_2 = \pi(\sigma_b, \hat{y})$, and $\sigma_3 = \pi(\sigma_a \otimes \sigma_b, \hat{x} \cup \hat{y},)$. As $\hat{x} \cap \hat{y} = \emptyset$, we have $(\sigma_1 \otimes \sigma_2)\downarrow$ and so $\sigma_1 \otimes \sigma_2 \sqsubseteq \sigma_a \otimes \sigma_b$ by the $\otimes$-preserving property of $\sqsubseteq$. Note that $\sigma_3 \sqsubseteq \sigma_a \otimes \sigma_b$ and σ_3 has the same variable domain as of $\sigma_1 \otimes \sigma_2$. Hence, $\sigma_1 \otimes \sigma_2 = \sigma_3$. □

Proof of Lemma 4.12. For (1), we first consider the $\Rightarrow$ direction where $\sigma \sqsubseteq \sigma'$. For every D-map m in u, there exists some D'-map m' such that $m \sqsubseteq_M m'$. By the definition of $\sqsubseteq_M$, it must be the case that $D \subseteq D'$. Thus $m = \pi(m', D)$, the D-restriction of m'. Now assume that $\sigma = \oplus_{i \in I}\, \delta(m_i, \rho_i)$. As $\sigma \sqsubseteq \sigma'$, there exists $(m'_i)_{i \in I}$ such that $\sigma' = \oplus_{i \in I}\, \delta(m'_i, \rho_i)$ and $\pi(m'_i, D) = m_i$ for every $i \in I$. Thus $\sigma = \pi(\sigma', D)$. The $\Leftarrow$ direction is similar.

For (2), let $D' = \mathrm{fv}(\phi)$ and $\sigma' = \pi(\sigma, D') \sqsubseteq \sigma$. We prove $\sigma \models \phi$ iff $D' \subseteq D \wedge \sigma' \models \phi$ by applying structural induction on the formula ϕ as follows:

(1) $\phi = [W]$. Note that for every D-map m, we have $m \models_M [W]$ iff $D' \subseteq D \wedge \pi(m, D') \models_M [W]$ (i.e. by structural induction on W). Thus $\sigma \models [W]$ iff $D' \subseteq D \wedge \sigma' \models [W]$.

(2) $\phi = \rho$. Direct from the fact that $\|\sigma\| = \|\sigma'\|$.

(3) $\phi = \phi_1 \boxplus \phi_2$. The $\Leftarrow$ direction is similar to its counterpart in the proof of the Monotonicity Lemma 4.7. For the $\Rightarrow$ direction, assume that $\sigma = \sigma_1 \oplus \sigma_2$ and $\sigma_1 \models \phi_1$, $\sigma_2 \models \phi_2$. We have $D' \subseteq D$ from the induction hypothesis. Also by the induction hypothesis, $\sigma' = \pi(\sigma, D') = \pi(\sigma_1, D') \oplus \pi(\sigma_2, D') \models \phi_1 \boxplus \phi_2$.

(4) $\phi = \boxplus_{i\in I}\phi_i$. Similar to the previous case.
(5) $\phi = \phi_1 \boxtimes \phi_2$. The $\Leftarrow$ direction is similar to its counterpart in the proof of the Monotonicity Lemma 4.7. For the $\Rightarrow$ direction, let $\sigma \models \phi_1 \boxtimes \phi_2$, i.e. there exist σ_1, σ_2 such that $\sigma_1 \otimes \sigma_2 \sqsubseteq \sigma$ and $\sigma_1 \models \phi_1, \sigma_2 \models \phi_2$. Now, let $D_1 = \mathsf{fv}(\phi_1)$ and $D_2 = \mathsf{fv}(\phi_2)$. Then $D_1 \cup D_2 = D'$ and $D_1 \cap D_2 = \emptyset$ (since separability implies disjointness, i.e. Lemma 4.10). By the induction hypothesis, we have $D_1 \subseteq \mathsf{vdom}(\sigma_1)$ and $D_2 \subseteq \mathsf{vdom}(\sigma_2)$. Thus $D' = D_1 \cup D_2 \subseteq \mathsf{vdom}(\sigma_1 \otimes \sigma_2) \subseteq D$. Now, let $\sigma_1' = \pi(\sigma_1, D_1)$, $\sigma_2' = \pi(\sigma_2, D_2)$ then $\sigma_1' \models \phi_1$, $\sigma_2' \models \phi_2$ by the induction hypothesis. As $D_1 \cap D_2 = \emptyset$, it follows that $(\sigma_1' \otimes \sigma_2')\downarrow$ and so $\sigma_1' \otimes \sigma_2' \models \phi_1 \boxtimes \phi_2$. Note that $\sigma_1' \otimes \sigma_2' \sqsubseteq \sigma_1 \otimes \sigma_2 \sqsubseteq \sigma$. Thus $\sigma' \models \phi_1 \boxtimes \phi_2$.
(6) $\phi = \psi/[W]$. For the $\Rightarrow$ direction, assume $\sigma \models \psi/[W]$. Thus there exists some σ_1 such that $\sigma_1 \models \psi$ and $\sigma = \sigma_1/[W]$. The former implies that $\mathsf{fv}(\psi) \subseteq \mathsf{vdom}(\sigma_1)$ and the latter implies that $\mathsf{fv}(W) \subseteq \mathsf{vdom}(\sigma_1) = \mathsf{vdom}(\sigma)$. Thus $\mathsf{fv}(\phi) \subseteq \mathsf{vdom}(\sigma)$. By the induction hypothesis, we have $\pi(\sigma_1, \mathsf{fv}(\phi)) \models \psi$. Note that $\pi(\sigma, \mathsf{fv}(\phi)) = \pi(\sigma_1, \mathsf{fv}(\phi))/[W]$. Thus $\pi(\sigma, \mathsf{fv}(\phi)) \models \psi/[W]$.
For the $\Leftarrow$ direction, assume $R = \mathsf{fv}(\phi) \subseteq \mathsf{vdom}(\sigma)$ and $\sigma' \models \psi/[W]$ where $\sigma' \triangleq \pi(\sigma, R) \sqsubseteq \sigma$. Thus there exists some σ_1' such that $\sigma_1' \models \psi$ and $\sigma' = \sigma_1'/[W]$. Let $\sigma_2' \triangleq \sigma_1'/[\neg W]$. Then $\sigma_1' \oplus \sigma_2' = \sigma'$. Let σ_2 be some D-state such that $\pi(\sigma_2, R) = \sigma_2'$. By the induction hypothesis, we have $\sigma_2 \models [\neg W]$ and $\sigma \models [W]$. As $(\sigma' \oplus \sigma_2')\downarrow$, we have $(\sigma \oplus \sigma_2)\downarrow$ and thus $\sigma \oplus \sigma_2 \models \psi$ by the induction hypothesis. Hence, $\sigma \models \psi/[W]$.
(7) $\phi = \psi_{\downarrow\rho}$. Direct from the definition.
(8) The other cases are direct from the induction hypothesis.

Hence, we are done. □

Proof of Lemma 4.13. The base cases are easy. Now let ϕ, ψ be convex predicates. Then:

(1) $\rho \cdot \phi$: Let $\sigma \models \boxplus_{i=0}^{\infty} \rho_i \cdot (\rho \cdot \phi)$, i.e. there exist $(\sigma_i)_{i=0}^{\infty}$ such that $\sigma = \oplus_{i=0}^{\infty} (\rho_i \rho \bullet \sigma_i)$ and $\sigma_i \models \phi$ for every i. Let $\sigma' \triangleq \oplus_{i=0}^{\infty} (\rho_i \bullet \sigma_i)$ then $\sigma' \models \phi$ and $\sigma = \rho \bullet \sigma'$. Thus $\sigma \models \rho \cdot \phi$.
(2) $(\hat{x} = \vec{v}) \boxtimes \phi$: Let $\sigma \models \boxplus_{i=0}^{\infty} \rho_i \cdot ((\hat{x} = \vec{v}) \boxtimes \phi)$, i.e. there exist $(\sigma_i)_{i=0}^{\infty}$ such that $\sigma = \oplus_{i=0}^{\infty} (\rho_i \bullet \sigma_i)$ and $\sigma_i \models (\hat{x} = \vec{v}) \boxtimes \phi$ for every i. We can deduce that $\sigma \models (\hat{x} = \vec{v}) \boxtimes \left(\oplus_{i=0}^{\infty} \rho_i \cdot \phi\right)$ and so $\sigma \models (\hat{x} = \vec{v}) \boxtimes \phi$.
(3) $\phi \wedge \psi$: Direct from the induction hypothesis.

Hence, we are done. □

C Proofs of the entailment rules in Fig. 3

A comprehensive rule set for entailment is provided in Fig. 12. Here the rules are meant for the program state model although some are also true with respect to the multiverse model. Most rules can be derived either directly from the forcing semantics or from the properties of the multiverse, i.e. Lemma 4.3. Thus it is sufficient to cover the following non-trivial rules:

(1)
$$\frac{\phi \vdash 1 \quad \mathsf{fv}(\psi_1) = \mathsf{fv}(\psi_2)}{(\phi \boxtimes \psi_1) \wedge (\phi \boxtimes \psi_2) \vdash \phi \boxtimes (\psi_1 \wedge \psi_2)}\ \boxtimes^{\wedge 2}$$
Let $\sigma \models (\phi \boxtimes \psi_1) \wedge (\phi \boxtimes \psi_2)$. Thus there exist $\sigma_1, \sigma_1', \sigma_2, \sigma_2'$ such that $\sigma_1 \otimes \sigma_2 \sqsubseteq \sigma$, $\sigma_1' \otimes \sigma_2' \sqsubseteq \sigma$, and $\sigma_1 \models \phi$, $\sigma_1' \models \phi$, $\sigma_2 \models \psi_1$, $\sigma_2' \models \psi_2$. As $\phi \vdash 1$, it follows that $\|\sigma_1\| = \|\sigma_1'\| = 1$. By Lemma 4.12, we deduce that $\sigma_2 \sqsubseteq \sigma$ and $\sigma_2' \sqsubseteq \sigma$ and therefore, also by Lemma 4.12, we have $\sigma_3 \triangleq \pi(\sigma, R) \models \psi_1 \wedge \psi_2$ where $R \triangleq \mathsf{fv}(\psi_1) = \mathsf{fv}(\psi_2)$. Consequently, $(\sigma_1 \otimes \sigma_3)\downarrow$ and $\sigma_1 \otimes \sigma_3 \models \phi \boxtimes (\psi_1 \wedge \psi_2)$. Hence, $\sigma \models \phi \boxtimes (\psi_1 \wedge \psi_2)$ by Lemma 4.12.
(2)
$$\frac{}{\phi \boxtimes 1 \vdash \phi}\ \boxtimes^{1}$$

$$\frac{}{\phi \boxtimes \psi \vdash \psi \boxtimes \phi}\,\boxtimes^{\mathsf{C}} \qquad \frac{}{(\phi \boxtimes \psi) \boxtimes \theta \dashv\vdash \phi \boxtimes (\psi \boxtimes \theta)}\,\boxtimes^{\mathsf{A}} \qquad \frac{}{\bot \boxtimes \phi \vdash \bot}\,\boxtimes^{\bot} \qquad \frac{}{\phi \boxtimes 1 \vdash \phi}\,\boxtimes^{1}$$

$$\frac{\phi \vdash \psi}{\phi \boxtimes \theta \vdash \psi \boxtimes \theta}\,\boxtimes^{\vdash} \qquad \frac{}{(\phi \vee \psi) \boxtimes \theta \dashv\vdash (\phi \boxtimes \theta) \vee (\psi \boxtimes \theta)}\,\boxtimes^{\vee} \qquad \frac{}{\phi \boxtimes (\boxplus_{i\in I} \psi_i) \vdash \boxplus_{i\in I} (\phi \boxtimes \psi_i)}\,\boxtimes^{\boxplus}$$

$$\frac{}{\phi \boxtimes (\psi_1 \wedge \psi_2) \vdash (\phi \boxtimes \psi_1) \wedge (\phi \boxtimes \psi_2)}\,\boxtimes^{\wedge 1} \qquad \frac{\phi \vdash 1 \quad \mathsf{fv}(\psi_1) = \mathsf{fv}(\psi_2)}{(\phi \boxtimes \psi_1) \wedge (\phi \boxtimes \psi_2) \vdash \phi \boxtimes (\psi_1 \wedge \psi_2)}\,\boxtimes^{\wedge 2}$$

(a) $\boxtimes$-predicate rules

$$\frac{}{\phi \boxplus \psi \vdash \psi \boxplus \phi}\,\boxplus^{\mathsf{C}} \qquad \frac{}{(\phi \boxplus \psi) \boxplus \theta \vdash \phi \boxplus (\psi \boxplus \theta)}\,\boxplus^{\mathsf{A}} \qquad \frac{}{\bot \boxplus \phi \vdash \bot}\,\boxplus^{\bot}$$

$$\frac{}{\phi \boxplus (\psi_1 \wedge \psi_2) \vdash (\phi \boxplus \psi_1) \wedge (\phi \boxplus \psi_2)}\,\boxplus^{\wedge} \qquad \frac{}{(\phi \vee \psi) \boxplus \theta \dashv\vdash (\phi \boxplus \theta) \vee (\psi \boxplus \theta)}\,\boxplus^{\vee}$$

$$\frac{}{0 \boxplus \phi \dashv\vdash \phi}\,\boxplus^{0} \qquad \frac{\phi \vdash \psi}{\phi \boxplus \theta \vdash \psi \boxplus \theta}\,\boxplus^{\vdash} \qquad \frac{}{\boxplus_{i\in I} (\alpha_i \cdot ((\hat{x} = \vec{v}) \boxtimes \phi_i)) \vdash (\hat{x} = \vec{v}) \boxtimes (\boxplus_{i\in I} (\alpha_i \cdot \phi_i))}\,\boxplus^{\boxtimes}$$

(b) $\boxplus$-predicate rules

$$\frac{}{1 \cdot \phi \vdash \phi}\,\mathsf{D}^{1} \qquad \frac{}{0 \cdot \phi \vdash 0}\,\mathsf{D}^{0} \qquad \frac{\beta > 0}{\alpha \cdot \beta \dashv\vdash \alpha \times \beta}\,\mathsf{D}^{\times} \qquad \frac{}{\boxplus_{i\in I} \rho_i \dashv\vdash \sum_{i\in I} \rho_i}\,\mathsf{D}^{+}$$

(c) Dot-predicate rules

$$\frac{P \vdash_M Q}{[P] \vdash [Q]}\,\mathsf{W}^{\vdash} \qquad \frac{}{[P] \boxtimes Q \vdash [P]}\,\mathsf{W}^{\boxtimes} \qquad \frac{}{0 \vdash [P]}\,\mathsf{W}^{0} \qquad \frac{}{[P] \wedge [Q] \dashv\vdash [P \wedge Q]}\,\mathsf{W}^{\wedge}$$

$$\frac{a \in \mathbb{Q} \quad x \notin \mathsf{fv}(\phi)}{\langle x = a\rangle \wedge \phi \vdash \langle x = a\rangle \boxtimes \phi}\,\mathsf{W}^{\wedge}_{\boxtimes} \qquad \frac{}{\rho \cdot \langle P\rangle \dashv\vdash \rho \wedge [P]}\,\mathsf{W}^{\mathsf{D}} \qquad \frac{a \in \mathbb{Q}}{\phi(x) \wedge [x = a] \vdash \phi(a)}\,\mathsf{W}^{=}$$

$$\frac{}{\alpha \cdot \beta \wedge ([P] \boxtimes [Q]) \vdash (\alpha \wedge [P]) \boxtimes (\beta \wedge [Q])}\,\mathsf{W}^{\mathsf{N}}_{\boxtimes}$$

$$\frac{}{\boxplus_{i\in I} [P] \vdash [P]}\,\mathsf{W}^{\boxplus 1} \qquad \frac{}{[P \vee Q] \vdash [P] \boxplus [\neg P \wedge Q]}\,\mathsf{W}^{\boxplus} \qquad \frac{}{[P] \wedge (\boxplus_{i\in I} \phi_i) \vdash \boxplus_{i\in I} ([P] \wedge \phi_i)}\,\mathsf{W}^{\wedge}_{\boxplus}$$

(d) Base-predicate rules

$$\frac{}{(\alpha \cdot \phi \boxplus \overline{\alpha} \cdot \psi)/[P] \dashv\vdash (\alpha \cdot \phi)/[P] \boxplus (\overline{\alpha} \cdot \psi)/[P]}\,\mathsf{F}^{\boxplus} \qquad \frac{\phi \vdash [P]}{\phi/[P] \dashv\vdash \phi}\,\mathsf{F}^{\mathsf{E}}_{1} \qquad \frac{\phi \vdash [\neg P]}{\phi/[P] \vdash 0}\,\mathsf{F}^{\mathsf{E}}_{2}$$

$$\frac{}{\phi/[P] \vdash [P]}\,\mathsf{F}^{\mathsf{W}} \qquad \frac{}{(\phi/[P])/[Q] \dashv\vdash \phi/[P \wedge W]}\,\mathsf{F}^{\wedge} \qquad \frac{\phi \vdash \rho}{(\alpha \cdot \phi)_{\downarrow\rho} \vdash \phi}\,\mathsf{N}^{\mathsf{E}} \qquad \frac{}{\phi_{\downarrow\rho} \vdash \rho}\,\mathsf{N}^{\|\cdot\|}$$

(e) Filter-predicate and normalization-predicate rules

Fig. 12. Useful entailment rules

This is the weakening rule for $\boxplus$ to reason about marginal distributions. Let $\sigma \models \phi \boxtimes 1$. Thus there exist σ_1, σ_2 such that $\sigma_1 \otimes \sigma_2 \sqsubseteq \sigma$ and $\sigma_1 \models \phi$, $\sigma_2 \models 1$. Note that $\|\sigma\| = \|\sigma_1\| \times \|\sigma_2\|$. Accordingly, $\|\sigma\| = \|\sigma_1\|$ and so $\sigma_1 \sqsubseteq \sigma$. By Lemma 4.12, we have $\sigma \models \phi$.

(3)

$$\frac{}{\boxplus_{i\in I}\ (\alpha_i \cdot ((\hat{x} = \vec{v}) \boxtimes \phi_i)) \vdash (\hat{x} = \vec{v}) \boxtimes (\boxplus_{i\in I}\ (\alpha_i \cdot \phi_i))}\ \boxplus^{\boxtimes}$$

W.l.o.g., assume $I = \mathbb{N}$ and $(\alpha_i)_{i=0}^{\infty}$ satisfies $\alpha_i > 0$ for every i. Let $\sigma \models \oplus_{i=0}^{\infty} \alpha_i \cdot ((\hat{x} = \vec{v}) \boxtimes \phi_i)$. Thus there exist $(\sigma_i)_{i=0}^{\infty}$ such that $\sigma = \oplus_{i=0}^{\infty} \sigma_i$ and $\sigma_i \models \alpha_i \cdot ((\hat{x} = \vec{v}) \boxtimes \phi_i)$ for every i. By Lemma 4.12, we have $\pi(\sigma_i, R) \models \alpha_i \cdot (\hat{x} = \vec{v})$ where R is the set of free variables in $\hat{x} = \vec{v}$ (i.e. $R = \hat{x}$). Furthermore, $\alpha_i^{-1} \bullet \pi(\sigma_i, R) = \alpha_j^{-1} \bullet \pi(\sigma_j, R)$ for every i, j (as they contain only variables in $\hat{x}$ and satisfy $\hat{x} = \vec{v}$). Consequently, there exists some $\sigma' \models \hat{x} = \vec{v}$ such that for every i, there exists some σ_i' such that $\sigma_i = \alpha_i \bullet \sigma' \otimes \sigma_i'$ and $\sigma_i' \models \phi_i$. From this, we can deduce that $\sigma = \sigma' \otimes \left(\oplus_{i=0}^{\infty} \alpha_i \bullet \sigma_i'\right) \models (\hat{x} = \vec{v}) \boxtimes (\boxplus_{i\in I}\ (\alpha_i \cdot \phi_i))$.

(4)

$$\frac{}{\alpha \cdot \langle P \rangle \dashv\vdash \alpha \wedge [P]}\ \mathrm{W}^{\mathrm{D}}$$

Note that $\langle P \rangle$ is simply $[P] \wedge 1$. The case $\alpha = 0$ is trivial. Assume $\alpha > 0$. Note that $\alpha \cdot \phi$ is equivalent to $\alpha \boxtimes \phi$ by Lemma 4.8. Thus $\alpha \cdot ([P] \wedge 1)$ is equivalent to $(\alpha \cdot [P]) \wedge \alpha$ by $\boxtimes^{\wedge}$ Furthermore, it is easy to check that $\alpha \cdot [P]$ is equivalent to $[P]$. Hence, we are done.

(5)

$$\frac{a \in \mathbb{Q}}{\phi(x) \wedge [x = a] \vdash \phi(a)}\ \mathrm{W}^{=}$$

We assume $x \in \mathrm{fv}(\phi)$, otherwise the entailment is trivial. We prove by induction on ϕ. For the base cases, note that if $\phi = [W]$ then $[W] \wedge [x = a]$ is equivalent to $[W(x) \wedge x = a]$ which implies $[W(a/x)]$ by the principle of substitution. If $\phi = \phi_1 \boxplus \phi_2$ then either ϕ_1 or ϕ_2 contains x but not both by the entailment follows from the induction hypothesis. The other cases follow from the induction hypothesis.

(6)

$$\frac{}{[P \vee Q] \vdash [P] \boxplus [\neg P \wedge Q]}\ \mathrm{W}^{\boxplus}$$

Let $\sigma \models [P \vee Q]$. Note that $\sigma = \sigma/[P] \oplus \sigma/[\neg P]$ and $\sigma/[P] \models [P]$, $\sigma/[\neg P] \models [\neg P]$. Since every world of σ satisfies $P \vee Q$, we can infer that $\sigma/[\neg P] \models [\neg P \wedge Q]$. Hence, $\sigma \models [P] \boxplus [\neg P \wedge Q]$.

(7)

$$\frac{P \vdash_M Q}{[P] \vdash [Q]}\ \mathrm{W}^{\vdash}$$

Here we write $P \vdash_M Q$ to mean that P proves Q in the world model. This rule basically says that we can lift the entailment in the world model onto the multiverse model. Let $\sigma \models [P]$. If σ is empty then we are done, since $[Q]$ is satisfied by the empty multiverse. Otherwise, let $w \in \mathrm{dom}(\sigma)$. Then $w \models_M P$ and so $w \models_M Q$. Hence, $\sigma \models [Q]$.

(8)

$$\frac{}{[P] \boxtimes \phi \vdash [P]}\ \mathrm{W}^{\boxtimes}$$

Let $\sigma \models [P] \boxtimes Q$. Then there exist σ_1, σ_2 such that $\sigma_1 \otimes \sigma_2 \sqsubseteq \sigma$ and $\sigma_1 \models [P]$, $\sigma_2 \models Q$. Let $w \in \mathrm{dom}(\sigma)$. It follows that $\pi(w, R) \in \pi(\sigma, R)$ where $R \triangleq \mathrm{vdom}(\sigma_1)$. Furthermore, $\mathrm{dom}(\sigma_1) = \mathrm{dom}(\pi(\sigma, R))$ as they only differ by a scalar factor. Thus $\pi(w, R) \in \mathrm{dom}(\sigma_1)$ and so $\pi(w, R) \models_M P$. As the evaluation of P only depends on the evaluation of its free variables, we conclude that $w \models P$ as well. As the choice of $w \in \mathrm{dom}(\sigma)$ is arbitrary, we have $\sigma \models [P]$.

(9)

$$\frac{a \in \mathbb{Q} \quad x \notin \mathrm{fv}(\phi)}{\langle x = a \rangle \wedge \phi \vdash \langle x = a \rangle \boxtimes \phi}\ \mathrm{W}^{\wedge}_{\boxtimes}$$

Let $\sigma \models \langle x = a \rangle \wedge \phi$. Then $\sigma_1 \triangleq \pi(\sigma, \{x\}) \models [x = a]$ and $\sigma_2 \triangleq \pi(\sigma, \mathrm{vdom}(\sigma) - \{x\}) \models \phi$. Note that $\|\sigma\| = 1$ and $\sigma = \sigma_1 \otimes \sigma_2$. Hence, $\sigma \models \langle x = a \rangle \boxtimes \phi$.

(10)

$$\frac{}{[P] \wedge (\boxplus_{i \in I}\ \phi_i) \vdash \boxplus_{i \in I}\ ([P] \wedge \phi_i)}\ \mathsf{W}^{\wedge}_{\boxplus}$$

W.l.o.g., assume $I = \mathbb{N}$. Let $\sigma \models [P] \wedge \boxplus_{i=0}^{\infty}\ \phi_i$. Then there exists $(\sigma_i)_{i=0}^{\infty}$ such that $\sigma = \oplus_{i=0}^{\infty}\ \sigma_i$ and $\sigma_i \models \phi_i$ for every i. Furthermore, if $w \in \mathrm{dom}(\sigma_i)$ then $w \in \mathrm{dom}(\sigma)$ and so $w \models_M P$ as $\sigma \models [P]$. Accordingly, we have $\sigma_i \models [P] \wedge \phi_i$ for every i. Hence, $\sigma \models \boxplus_{i \in I}\ ([P] \wedge \phi_i)$.

(11)

$$\frac{}{\alpha \cdot \beta \wedge ([P] \boxtimes [Q]) \vdash (\alpha \wedge [P]) \boxtimes (\beta \wedge [Q])}\ \mathsf{W}^{\mathsf{N}}_{\boxtimes}$$

Let $\sigma \models \alpha \cdot \beta \wedge ([P] \boxtimes [Q])$. Clearly, $\alpha \times \beta > 0$ as $\boxtimes$ requires σ to be non-empty. As $\sigma \models [P] \boxtimes [Q]$, there exist σ_1, σ_2 such that $\sigma_1 \otimes \sigma_2 \sqsubseteq \sigma$ and $\sigma_1 \models [P]$, $\sigma_2 \models [Q]$. Note that if we let $\sigma_1' \triangleq (\alpha \times \|\sigma_1\|^{-1}) \bullet \sigma_1$ and $\sigma_2' \triangleq (\beta \times \|\sigma_2\|^{-1}) \bullet \sigma_2$ then $\sigma_1' \models [P]$ and $\sigma_2' \models [Q]$. Furthermore, $\sigma_1' \otimes \sigma_2' = \sigma_1 \otimes \sigma_2$ and $\|\sigma_1'\| = \alpha$, $\|\sigma_2'\| = \beta$. By Lemma 4.12, we conclude that $\sigma \models (\alpha \wedge [P]) \boxtimes (\beta \wedge [Q])$.

(12)

$$\frac{}{(\alpha \cdot \phi \boxplus \overline{\alpha} \cdot \psi)/[P] \dashv\vdash (\alpha \cdot \phi)/[P] \boxplus (\overline{\alpha} \cdot \psi)/[P]}\ \mathsf{F}^{\boxplus}$$

We mostly use this rule to split the filter predicate into sub-distributions that satisfies/does not satisfy the world predicate. For the $\dashv$ direction, let $\sigma \models (\alpha \cdot \phi)/[P] \boxplus (\overline{\alpha} \cdot \psi)/[P]$. Thus there exist σ_1, σ_2 such that $\sigma = \sigma_1 \oplus \sigma_2$, $\sigma_1 \models (\alpha \cdot \phi)/[P]$ and $\sigma_2 \models (\overline{\alpha} \cdot \psi)/[P]$. Accordingly, there exist σ_1', σ_2' such that $\sigma_1 = \sigma_1'/[P]$, $\sigma_2 = \sigma_2'/[P]$ and $\sigma_1' \models \alpha \cdot \phi$ whereas $\sigma_2' \models \overline{\alpha} \cdot \psi$. Note that $\|\sigma_1'\| + \|\sigma_2'\| \leq \alpha + \overline{\alpha} = 1$. As a result, $(\sigma_1' \oplus \sigma_2')\downarrow$ and $(\sigma_1' \oplus \sigma_2')/[P] = \sigma_1 \oplus \sigma_2 = \sigma$. Hence, $\sigma \models (\alpha \cdot \phi \boxplus \overline{\alpha} \cdot \psi)/[P]$.
The $\vdash$ direction is similar.

(13)

$$\frac{\phi \vdash \rho}{(\alpha \cdot \phi)_{\downarrow \rho} \vdash \phi}\ \mathsf{N}^{\mathsf{E}}$$

This rule allows us to eliminate the re-normalization predicate. The case $\rho = 0$ is trivial. Assume $\rho > 0$. Let $\sigma \models (\alpha \cdot \phi)_{\downarrow \rho}$. Thus $\|\sigma\| = \rho$ and there exists some $\beta \in \mathbb{R}_{>0}$ such that $\beta \bullet \sigma \models \alpha \cdot \phi$. We assume that $\alpha > 0$ as the case $\alpha = 0$ is trivial. Thus $\alpha^{-1} \times \beta \bullet \sigma \models \phi$. It follows that $\alpha^{-1} \times \beta \times \|\sigma\| = \rho$. Note that $\|\sigma\| = \rho$. Thus $\alpha = \beta$ and so $\sigma \models \phi$.

D Proof of lemmas in Section 5

Proof of Lemma 5.1. For (1), the proof is straightforward by applying induction on the number of steps n. For (2), we write $n_2 = n_1 + d$ where $d \geq 0$ and apply induction on d to show that $\lfloor \kappa_1 \rfloor \preceq \lfloor \kappa_2 \rfloor$. Finally, note that $\sup\,(\lfloor \kappa_i \rfloor)_{i=0}^{\infty}$ is well-defined because the sequence is $\preceq$-ordered. □

E Proof of Theorem 5.2

We proceed to prove the rules in Fig. 13. Recall that $[b]$ and $[\neg b]$ are simply shorthands for $[b \Downarrow 1]$ and $[b \Downarrow 0]$, respectively.

E.1 Proofs of IFrame and DFrame

First, we prove the following intermediate results:

Proposition E.1. *Let σ, σ' be D-states such that $\sigma \circledcirc_c \sigma'$. Then:*

(1) $\|\sigma'\| \leq \|\sigma\|$.
(2) *Let σ'' be a D-state such that $\sigma \circledcirc_c \sigma''$. Then $\sigma' = \sigma''$.*

$$\frac{\{P\}\, c\, \{Q\} \qquad \mathsf{fv}(F) \cap \mathsf{vars}(c) = \emptyset}{\{P \boxtimes F\}\, c\, \{Q \boxtimes F\}}\ [\textsc{IFrame}] \qquad \frac{\{P\}\, c\, \{Q\} \qquad Q \vdash 1 \qquad \mathsf{mv}(c) \cap \mathsf{fv}(F) = \emptyset}{\{P \wedge F\}\, c\, \{Q \wedge F\}}\ [\textsc{DFrame}]$$

$$\frac{\{P_1\}\, c\, \{Q_1\} \qquad \{P_2\}\, c\, \{Q_2\}}{\{P_1 \boxplus P_2\}\, c\, \{Q_1 \boxplus Q_2\}}\ [\textsc{Split}] \qquad \frac{\forall i \in I.\ \{P_i\}\, c\, \{Q_i\}}{\{\boxplus_{i\in I} P_i\}\, c\, \{\boxplus_{i\in I} Q_i\}}\ [\textsc{ISplit}]$$

(a) Rules for local reasoning

$$\frac{}{\{P\}\ \mathbf{skip}\ \{P\}}\ [\textsc{Skip}] \qquad \frac{\{P\}\, c_1\, \{R\} \qquad \{R\}\, c_2\, \{Q\}}{\{P\}\, c_1; c_2\, \{Q\}}\ [\textsc{Seq}]$$

$$\frac{}{\{\langle W(e/x)\rangle\}\ x := e\ \{\langle W\rangle\}}\ [\textsc{Assign}] \qquad \frac{\{P\}\, c_1\, \{Q_1\} \qquad \{P\}\, c_2\, \{Q_2\} \qquad P \vdash [e \Downarrow \rho]}{\{P\}\, c_1 [e] c_2\, \{\rho \cdot Q_1 \boxplus \overline{\rho} \cdot Q_2\}}\ [\textsc{Choice}]$$

$$\frac{\begin{array}{c} P(n) \vdash (1 - \sum_{i=0}^{n} \rho_i) \cdot \langle b\rangle \qquad Q(n) \vdash \rho_n \cdot \langle \neg b\rangle \\ \{P(n)\}\, c\, \{P(n+1) \boxplus Q(n+1)\} \qquad \sum_{i=0}^{\infty} \rho_i = \rho \le 1 \end{array}}{\{I(0)\}\ \mathbf{while}\ b\ \mathbf{do}\ c\ \left\{\rho \cdot \langle b = 1\rangle \wedge \left(\boxplus_{i=0}^{\infty} Q(i)\right)\right\}}\ [\textsc{While}]$$

(b) Rules for basic statements

$$\frac{}{\{\bot\}\, c\, \{Q\}}\ [\textsc{Bot}] \qquad \frac{}{\{0\}\, c\, \{0\}}\ [\textsc{Emp}] \qquad \frac{P' \vdash P \qquad \{P\}\, c\, \{Q\} \qquad Q \vdash Q'}{\{P'\}\, c\, \{Q'\}}\ [\textsc{Cons}]$$

$$\frac{\{P\}\, c\, \{Q_1\} \qquad \{P\}\, c\, \{Q_2\}}{\{P\}\, c\, \{Q_1 \wedge Q_2\}}\ [\textsc{Conj}] \qquad \frac{\{P_1\}\, c\, \{Q\} \qquad \{P_2\}\, c\, \{Q\}}{\{P_1 \vee P_2\}\, c\, \{Q\}}\ [\textsc{Disj}]$$

(c) Other rules

$$\frac{\{P\}\, c\, \{Q\}}{\{\rho \cdot P\}\, c\, \{\rho \cdot Q\}}\ [\textsc{Dot}] \qquad \frac{\forall 1 \le i \le n.\ \{\langle P_i\rangle\}\, c\, \{\langle Q_i\rangle\}}{\left\{\boxplus_{i=1}^{n} \rho_i \cdot \langle P_i\rangle\right\}\, c\, \left\{\boxplus_{i=1}^{n} \rho_i \cdot \langle Q_i\rangle\right\}}\ [\textsc{Cases}]$$

$$\frac{\vec{v} = \bigoplus_{i=1}^{n} (\rho_i : v_i)}{\{1\}\ x := \vec{v}\ \{\langle x = \vec{v}\rangle\}}\ [\textsc{DAssign}] \qquad \frac{\{P/[b]\}\, c_1\, \{Q_1\} \qquad \{P/[\neg b]\}\, c_2\, \{Q_2\}}{\{P\}\ \mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2\ \{Q_1 \boxplus Q_2\}}\ [\textsc{If}]$$

$$\frac{\begin{array}{c} I \vdash 1 \quad \mathsf{convex}(I/\langle \neg b\rangle) \quad \rho < 1 \\ \{I/\langle b\rangle\}\, c\, \{I \wedge (\rho \cdot \langle b\rangle \boxplus \overline{\rho} \cdot \langle \neg b\rangle)\} \end{array}}{\{I\}\ \mathbf{while}\ b\ \mathbf{do}\ c\ \{I/\langle \neg b\rangle\}}\ [\textsc{SWhile}] \qquad \frac{\begin{array}{c} n_1 \le n_2 + 1 \quad i \notin \mathsf{fv}(I) \cup \mathsf{vars}(c) \\ \{\langle I(i) \wedge i \le n_2\rangle\}\, c\, \{\langle I(i+1)\rangle\} \end{array}}{\{\langle I(n_1)\rangle\}\ \mathbf{for}\ i := [n_1, n_2]\ \mathbf{do}\ c\ \{\langle I(n_2+1)\rangle\}}\ [\textsc{For}]$$

(d) Derived rules

Fig. 13. Full proof rules.

Proof. Direct from Lemma 5.1. □

Proposition E.2 (Locality). *Let σ be D-states. Then:*

(1) **Product preserving**: *Let σ_F be a D'-state where $D \cap D' = \emptyset$. Then for every D-state σ':*

$$\sigma \mathbin{⯀_c} \sigma' \quad \textit{implies} \quad (\sigma_F \otimes \sigma) \mathbin{⯀_c} (\sigma_F \otimes \sigma').$$

(2) **Locality**: *Let $S \subseteq D$ and assume $\pi(\sigma, S) \mathbin{⯀_c} \sigma_S$ for some S-state σ_S. Then there exists some D-state σ' such that $\sigma \mathbin{⯀_c} \sigma'$ and $\sigma_S = \pi(\sigma', S)$. Furthermore, let $T \subseteq D$ such that $T \cap \mathsf{mv}(c) = \emptyset$. Then $\pi(\sigma', T) \preceq \pi(\sigma, T)$.*

Proof. (1). First, the following result is straightforward by induction on the number of steps:

$$\forall n.\ \kappa \rightsquigarrow_n \kappa' \longrightarrow \sigma_F \otimes \kappa \rightsquigarrow_n \sigma_F \otimes \kappa'$$

where κ, κ' are D-configs.

Let $(\kappa_i)_{i=0}^{\infty}$ be the induced run of $\langle c, \sigma\rangle$. Consequently, $(\sigma_F \otimes \kappa_i)_{i=0}^{\infty}$ is the induced run of $\langle c, \sigma_F \otimes \sigma\rangle$. Accordingly, $\sup(\lfloor \sigma_F \otimes \kappa_i \rfloor)_{i=0}^{\infty} = \sigma_F \otimes \sup(\lfloor \kappa_i \rfloor)_{i=0}^{\infty} = \sigma_F \otimes \sigma'$. Hence, $(\sigma_F \otimes \sigma) \mathbin{⯀_c} (\sigma_F \otimes \sigma')$.

(2). The following result is straightforward by induction on the number of steps:

$$\forall n.\ \pi(\kappa, S) \rightsquigarrow_n \kappa'_S \longrightarrow \exists \kappa'.\ \kappa \rightsquigarrow_n \kappa' \wedge \pi(\kappa', S) = \kappa'_S$$

where κ, κ' are D-configs and κ'_S is a S-config.

Let $(\kappa_i)_{i=0}^{\infty}$ be the induced run of $\langle c, \pi(\sigma, S)\rangle$. Consequently, there exists a sequence $(\kappa'_i)_{i=0}^{\infty}$ which is the induced run of $\langle c, \sigma\rangle$ and $\pi(\kappa'_i, S) = \kappa_i$ for every i. Let $\sigma' \triangleq \sup(\kappa'_i)_{i=0}^{\infty}$. Then $\sigma \mathbin{⯀_c} \sigma'$ and:

$$\sigma_S \;=\; \sup(\kappa_i)_{i=0}^{\infty} \;=\; \sup(\pi(\kappa'_i, S))_{i=0}^{\infty} \;=\; \pi(\sup(\kappa'_i)_{i=0}^{\infty}, S) \;=\; \pi(\sigma', S)$$

Now let $T \subseteq D$ and $T \cap \mathsf{mv}(c) = \emptyset$. The following result is straightforward by induction on the number of steps:

$$\forall n.\ \kappa \rightsquigarrow_n \kappa' \longrightarrow \pi(\lfloor \kappa \rfloor, T) = \pi(\lfloor \kappa' \rfloor, T)$$

where $\lfloor \kappa \rfloor$ is the state snapshot of κ.

Recall that $\sigma' = \sup(\lfloor \kappa'_i \rfloor)_{i=0}^{\infty}$ where $(\kappa'_i)_{i=0}^{\infty}$ is the induced run of $\langle c, \sigma\rangle$. Accordingly, we deduce that $\pi(\sigma, T) = \pi(\lfloor \kappa'_i \rfloor, T) \succeq \pi(\lfloor \kappa'_i \rfloor, T)$ for every i. Thus:

$$\pi(\sigma, T) \;\succeq\; \sup(\pi(\lfloor \kappa'_i \rfloor, T))_{i=0}^{\infty} \;=\; \pi(\sup(\lfloor \kappa'_i \rfloor)_{i=0}^{\infty}, T) \;=\; \pi(\sigma', T)$$

Hence, we are done. □

Proof of IFrame. We want to prove the following rule:

$$\frac{\{P\}\ c\ \{Q\} \qquad \mathsf{fv}(F) \cap \mathsf{vars}(c) = \emptyset}{\{P \boxtimes F\}\ c\ \{Q \boxtimes F\}}\ [\textsc{IFrame}]$$

Let $\sigma \models P \boxtimes F$ where $R = \mathsf{vars}(c) \subseteq \mathsf{vdom}(\sigma)$. Thus there exist σ_P, σ_F such that $\sigma_P \otimes \sigma_F \sqsubseteq \sigma$ and $\sigma_P \models P$, $\sigma_F \models F$. By Lemmas 4.10 and 4.12, we have $\mathsf{fv}(F) \cup \mathsf{fv}(P) \subseteq \mathsf{vdom}(\sigma)$ and $\mathsf{fv}(F) \cap \mathsf{fv}(P) = \emptyset$.

Let $\sigma'_P \triangleq \|\sigma_F\| \bullet \pi(\sigma, R \cup \mathsf{fv}(P))$ then $\sigma_P \sqsubseteq \sigma'_P$. As $\sigma_P \models P$, Lemma 4.12 implies $\sigma'_P \models P$. Note that $R \subseteq \mathsf{vdom}(\sigma'_P)$. Thus the triple $\{P\}\ c\ \{Q\}$ implies that there is some $(R \cup \mathsf{fv}(P))$-state σ'_Q such that $\sigma'_P \mathbin{⯀_c} \sigma'_Q$ and $\sigma'_Q \models Q$.

Now, let $\sigma'_F \triangleq \pi(\sigma_F, \mathsf{fv}(F))$. As $\sigma_F \models F$, Lemma 4.12 implies $\sigma'_F \models F$. Note that $\mathsf{vdom}(\sigma'_F) \cap \mathsf{vdom}(\sigma'_P) = \mathsf{fv}(F) \cap (\mathsf{fv}(P) \cup R) = \emptyset$. As a result, $(\sigma'_P \otimes \sigma'_F)\downarrow$ and so Prop. E.2 gives us $\sigma'_P \otimes \sigma'_F \mathbin{⯀_c} \sigma'_Q \otimes \sigma'_F$. Note that $\sigma'_P \otimes \sigma'_F = \pi(\sigma, S)$ where $S = \mathsf{fv}(F) \cup \mathsf{fv}(P)$. Accordingly, there exists some state σ' such that $\sigma \mathbin{⯀_c} \sigma'$ and $\sigma'_Q \otimes \sigma'_F = \pi(\sigma', S)$. Hence, $\sigma' \models Q \boxplus F$. □

Proof of DFrame. We want to prove the following rule:

$$\frac{\{P\}\ c\ \{Q\} \qquad Q \vdash 1 \qquad \mathsf{mv}(c) \cap \mathsf{fv}(F) = \emptyset}{\{P \wedge F\}\ c\ \{Q \wedge F\}}\ [\textsc{DFrame}]$$

Let $\sigma \models P \wedge F$ such that $R = \mathsf{vars}(c) \subseteq \mathsf{vdom}(\sigma)$. By Lemma 4.12, let $\sigma_F \triangleq \pi(\sigma, \mathsf{fv}(F))$ then $\sigma_F \models F$.

As $\sigma \models P$, the triple $\{P\}\, c\, \{Q\}$ implies that there exists some state σ' such that $\sigma \mathbin{⚙_c} \sigma'$ and $\sigma' \models Q$. It remains to show that $\sigma' \models F$. Note that $F \cap \mathsf{mv}(c) = \emptyset$. Thus Prop. E.2 implies that $\pi(\sigma', \mathsf{fv}(F)) \preceq \pi(\sigma, \mathsf{fv}(F)) = \sigma_F$. As $Q \vdash 1$, we have $\|\sigma'\| = 1$ and so $\|\pi(\sigma', \mathsf{fv}(F))\| = 1$ as well. Consequently, $\pi(\sigma', \mathsf{fv}(F)) = \sigma_F \models F$. Hence, $\sigma' \models F$ by Lemma 4.12 and so we are done. □

E.2 Proofs of Split and ISplit

First, we prove the following useful results:

Proposition E.3. *If $\kappa \rightsquigarrow_n \kappa'$ then $\|\kappa'\| = \|\kappa\|$.*

Proof. By applying induction on the number of steps n. □

Proposition E.4. *Assume σ_i, σ_i' are D-states and $\sigma_i \mathbin{⚙_c} \sigma_i'$ for every $i \in \{1, 2\}$. If $(\sigma_1 \oplus \sigma_2)\downarrow$ then $(\sigma_1' \oplus \sigma_2')\downarrow$ and $(\sigma_1 \oplus \sigma_2) \mathbin{⚙_c} (\sigma_1' \oplus \sigma_2')$.*

Generally, if $(\sigma_i)_{i\in I}$ and $(\sigma_i')_{i\in I}$ are D-states, $\sigma_i \mathbin{⚙_c} \sigma_i'$ for every $i \in I$, and $(\oplus_{i\in I}\, \sigma_i)\downarrow$ then $(\oplus_{i\in I}\, \sigma_i')\downarrow$ and $(\oplus_{i\in I}\, \sigma_i) \mathbin{⚙_c} (\oplus_{i\in I}\, \sigma_i')$.

Proof. The following result can be proven by applying induction on the number of steps:

$$\forall n.\ \left(\kappa_1 \rightsquigarrow_n \kappa_1' \wedge \kappa_2 \rightsquigarrow_n \kappa_2' \wedge (\kappa_1 \oplus \kappa_2)\downarrow\right) \rightarrow \left((\kappa_1' \oplus \kappa_2')\downarrow \wedge \kappa_1 \oplus \kappa_2 \rightsquigarrow_n \kappa_1' \oplus \kappa_2'\right)$$

where κ_i, κ_i' are D-configs and $(\kappa_1 \oplus \kappa_2)\downarrow$ iff $\|\kappa_1\| + \|\kappa_2\| \leq 1$.

As $\sigma_1 \mathbin{⚙_c} \sigma_1'$, there exists $(\kappa_i)_{i=0}^{\infty}$ which is the induced run of $\langle c, \sigma_1\rangle$ and $\sup\left(\lfloor \kappa_i \rfloor\right)_{i=0}^{\infty} = \sigma_1'$. Similarly, there exists $(\kappa_i')_{i=0}^{\infty}$ which is the induced run of $\langle c, \sigma_2\rangle$ and $\sup\left(\lfloor \kappa_i' \rfloor\right)_{i=0}^{\infty} = \sigma_2'$. By Prop. E.3, $\|\kappa_i\| + \|\kappa_i'\| = \|\sigma_1\| + \|\sigma_2\| \leq 1$ and so $(\kappa_i \oplus \kappa_i')\downarrow$. Consequently, $(\kappa_i \oplus \kappa_i')_{i=0}^{\infty}$ is the induced run of $\langle c, \sigma_1 \oplus \sigma_2\rangle$. Furthermore:

$$\sup\left(\lfloor \kappa_i \oplus \kappa_i' \rfloor\right)_{i=0}^{\infty} = \sup\left(\lfloor \kappa_i \rfloor \oplus \lfloor \kappa_i' \rfloor\right)_{i=0}^{\infty} = \sup\left(\lfloor \kappa_i \rfloor\right)_{i=0}^{\infty} \oplus \sup\left(\lfloor \kappa_i' \rfloor\right)_{i=0}^{\infty} = \sigma_1' \oplus \sigma_2'$$

Accordingly, we deduce that $(\sigma_1 \oplus \sigma_2) \mathbin{⚙_c} (\sigma_1' \oplus \sigma_2')$. The general case is similar. □

Proof of Split. We want to prove the following rule:

$$\frac{\{P_1\}\, c\, \{Q_1\} \qquad \{P_2\}\, c\, \{Q_2\}}{\{P_1 \boxplus P_2\}\, c\, \{Q_1 \boxplus Q_2\}}\ [\textsc{Split}]$$

Let $\sigma \models P_1 \boxplus P_2$ such that $R \subseteq D$ where $R = \mathsf{vars}(c)$ and $D = \mathsf{vdom}(\sigma)$. Thus there exist D-states σ_1, σ_2 such that $\sigma_1 \oplus \sigma_2 = \sigma$ and $\sigma_1 \models P_1$, $\sigma_2 \models P_2$.

As $\sigma_1 \models P_1$, the triple $\{P_1\}\, c\, \{Q_1\}$ implies that there exists some D-state σ_1' such that $\sigma_1 \mathbin{⚙_c} \sigma_1'$ and $\sigma_1' \models Q_1$. Similarly, there exists some D-state σ_2' such that $\sigma_2 \mathbin{⚙_c} \sigma_2'$ and $\sigma_2' \models Q_2$. By Prop. E.4, we deduce that $(\sigma_1 \oplus \sigma_2) \mathbin{⚙_c} (\sigma_1' \oplus \sigma_2')$. Accordingly, let $\sigma' \triangleq \sigma_1' \oplus \sigma_2'$ then $\sigma \mathbin{⚙_c} \sigma'$ and $\sigma' \models Q_1 \boxplus Q_2$. Hence, we are done. □

Proof of ISplit. We want to prove the following rule:

$$\frac{\forall i \in I.\ \{P_i\}\, c\, \{Q_i\}}{\{\boxtimes_{i\in I}\, P_i\}\, c\, \{\boxtimes_{i\in I}\, Q_i\}}\ [\textsc{ISplit}]$$

which is similar to the proof of Split. □

E.3 Proof of Choice

Proof of Choice. We want to prove the following rule:

$$\frac{\{P\}\, c_1\, \{Q_1\} \quad \{P\}\, c_2\, \{Q_2\} \quad P \vdash [e \Downarrow \rho]}{\{P\}\, c_1 [e] c_2\, \{\rho \cdot Q_1 \boxplus \overline{\rho} \cdot Q_2\}}\text{[Choice]}$$

Let $\sigma \models P$ such that $R = \mathrm{vars}(c) \subseteq \mathrm{vdom}(\sigma)$. As $P \vdash [e \Downarrow \rho]$, we deduce that $[\![e]\!](m) = \rho$ for every mapping $m \in \mathrm{dom}(\sigma)$. Thus:

$$\langle c_1 [e] c_2, \sigma\rangle \rightsquigarrow \kappa \quad \text{where} \quad \kappa \triangleq \langle c_1, \sigma\rangle \oplus_\rho \langle c_2, \sigma\rangle$$

As $\sigma \models P$, the triple $\{P\}\, c_1\, \{Q_1\}$ implies that there exists some σ_1' such that $\sigma \,⚙_{c_1}\, \sigma_1'$ and $\sigma_1' \models Q_1$. Similarly, there exists some σ_2' such that $\sigma ⚙_{c_2} \sigma_2'$ and $\sigma_2' \models Q_2$. Let $(\kappa_i)_{i=0}^{\infty}$, $(\kappa_i')_{i=0}^{\infty}$ be the induced runs of $\langle c_1, \sigma\rangle$, $\langle c_2, \sigma\rangle$, respectively. Then $(\kappa_i \oplus_\rho \kappa_i')_{i=0}^{\infty}$ is the induced run of κ and so $\{\kappa\} \cup (\kappa_i \oplus_\rho \kappa_i')_{i=0}^{\infty}$ is the induced run of $\langle c_1 [e] c_2, \sigma\rangle$. Furthermore:

$$\sup\left(\{\kappa\} \cup (\kappa_i \oplus_\rho \kappa_i')_{i=0}^{\infty}\right) = \sup (\kappa_i \oplus_\rho \kappa_i')_{i=0}^{\infty} = \sup (\kappa_i)_{i=0}^{\infty} \oplus_\rho \sup (\kappa_i')_{i=0}^{\infty} = \sigma_1' \oplus_\rho \sigma_2'$$

Consequently, let $\sigma' \triangleq \sup\left(\{\kappa\} \cup (\kappa_i \oplus_\rho \kappa_i')_{i=0}^{\infty}\right)$ then $\sigma \,⚙_{c_1[e]c_2}\, \sigma'$ and $\sigma' \models \rho \cdot Q_1 \boxplus (1-\rho) \cdot Q_2$. Hence, we are done. □

E.4 Proof of Assign

Proof of Assign. We want to prove the following rule:

$$\frac{}{\{\langle W(e/x)\rangle\}\, x := e\, \{\langle W\rangle\}}\text{[Assign]}$$

Let $\sigma \models \langle W(e/x)\rangle$ such that $R = \mathrm{fv}(e) \cup \{x\} \subseteq \mathrm{vdom}(\sigma)$. As a result, $m_i \models_M W(e/x)$ for every $m_i \in \mathrm{dom}(\sigma)$. Let m_i' be the corresponding mapping of m_i after the assignment. It follows that $m_i' \models_M W$ (i.e. by induction on W). Assume $\sigma = \oplus_{i\in I}\, \delta(m_i, \rho_i)$ then $\sigma \rightsquigarrow \sigma'$ where $\sigma' \triangleq \oplus_{i\in I}\, \delta(m_i', \rho_i)$. Thus $\sigma' \models \langle W\rangle$. □

E.5 Proof of Skip

Proof of Skip. We want to prove the following rule:

$$\frac{}{\{P\}\ \mathbf{skip}\ \{P\}}\text{[Skip]}$$

Direct from the definition. □

E.6 Proof of Seq

We need the following result:

Proposition E.5. *If $\sigma \,⚙_{c_1}\, \sigma'$ and $\sigma' \,⚙_{c_2}\, \sigma''$ then $\sigma \,⚙_{c_1;c_2}\, \sigma''$.*

Proof of Seq. We want to prove the following rule:

$$\frac{\{P\}\, c_1\, \{R\} \quad \{R\}\, c_2\, \{Q\}}{\{P\}\, c_1; c_2\, \{Q\}}\text{[Seq]}$$

Let $\sigma \models P$ such that $T = \mathrm{vars}(c_1) \cup \mathrm{vars}(c_2) \subseteq \mathrm{vdom}(\sigma)$. As $\{P\}\, c_1\, \{Q\}$ holds, there exists some σ' such that $\sigma \,⚙_{c_1}\, \sigma'$ and $\sigma' \models R$. As $\{R\}\, c_2\, \{Q\}$ holds, there exists some σ'' such that $\sigma' \,⚙_{c_2}\, \sigma''$ and $\sigma'' \models Q$. Using Prop. E.5, we infer that $\sigma \,⚙_{c_1;c_2}\, \sigma''$. Hence, $\{P\}\, c_1; c_2\, \{Q\}$. □

We now proceed to prove Prop. E.5. First, we need the following definitions and notations:

Definition 16 (Coloring). *Let $\kappa \triangleq \langle c_1; c_2, \sigma\rangle$ and suppose $(\kappa_i)_{i=0}^{n}$ is the induced run of κ. We color the basic configs $\delta = \langle c, m\rangle$ in κ_i using 3 colors red, yellow, and green using the following rules:*

(1) *If δ still hasn't finished c_1 yet, i.e. $c = c_1'; c_2$ then it is colored red, i.e. $\langle c_1'; c_2, m\rangle$.*
(2) *If δ has just finished c_1, i.e. $c = \textbf{\textit{skip}}; c_2$ then it is colored yellow, i.e. $\langle \textbf{\textit{skip}}; c_2, m\rangle$.*
(3) *Otherwise, δ has finished c_1 and is running c_2 in which it is colored green, i.e. $\langle c, m\rangle$*

As a result, all basic configs in the initial config κ are colored red (except the case $c_1 = \textbf{\textit{skip}}$ in which they are colored yellow). During the run, some of them will turn yellow and then green in the next step.

Definition 17 (Snapshot). *Let $\kappa = \langle c_1; c_2, \sigma\rangle$ and assume that κ never gets stuck for the first n steps, i.e. there exists an induced run $(\kappa_i)_{i=0}^n$ of κ. The (c_1, n)-snapshot of κ, denoted as $\overline{\pi}_{c_1}(\kappa, n)$, is the state portion σ' that turns yellow during the first n steps of κ. Formally, for each $\kappa_i = \kappa_i^1 \oplus \kappa_i^2 \oplus \kappa_i^3$, we define $\kappa_i[c_1] \triangleq \lfloor \kappa_i^2 \rfloor$ to be the state snapshot that turns yellow in the i^{th} step. Then $\overline{\pi}_{c_1}(\kappa, n) \triangleq \oplus_{i=0}^n \kappa_i[c_1]$.*

Proposition E.6. *Assume σ ⚙$_{c_1}$ σ' and σ' ⚙$_{c_2}$ σ''. Then there exists an infinite induced run $(\kappa_i)_{i=0}^\infty$ of $\langle c_1; c_2, \sigma\rangle$, i.e. $\langle c_1; c_2, \sigma\rangle$ never gets stuck.*

Proof. Let $\kappa \triangleq \langle c_1; c_2, \sigma\rangle$. For the sake of contradiction, assume that κ gets stuck after n steps, i.e. $\kappa \rightsquigarrow_n \kappa_n$ and κ_n cannot make the next step. This boils down to the fact that there is some basic config δ in κ_n that cannot make a basic step. There are three cases:

(1) Red: $\delta = \langle c_1'; c_2, m\rangle$, i.e. δ still has not finished c_1 yet. By construct, this implies that the basic config $\langle c_1', m\rangle$ must be contained in some config κ_k' in the infinite induced run $(\kappa_n')_{i=0}^\infty$ of $\langle c_1, \sigma\rangle$. However, this means that κ_k' cannot make the next step, which is a contradiction.
(2) Yellow: $\delta = \langle \textbf{skip}; c_2, m\rangle$. This is impossible as δ can always make the next step.
(3) Green: $\delta = \langle c_2', m\rangle$, i.e. δ is running c_2. Let $\sigma_n \triangleq \overline{\pi}_{c_1}(\kappa, n)$ be the (c_1, n)-snap shot of κ. As $\{P\}\ c_1\ \{R\}$ holds, there exists the induced run $(\kappa_i')_{i=0}^\infty$ of $\langle c_1, \sigma\rangle$ and $\sigma' = \sup(\lfloor \kappa_i' \rfloor)_{i=0}^\infty$. Note that $\sigma_n = \lfloor \kappa_n' \rfloor$ and so $\sigma_n \preceq \lfloor \kappa_i' \rfloor$ for every $i \geq n$. Consequently, $\sigma_n \preceq \sigma'$.
Likewise, as $\{R\}\ c_2\ \{Q\}$ holds and $\sigma' \models R$, there exists an induced run $(\kappa_i'')_{i=0}^\infty$ of $\langle c_2, \sigma'\rangle$ and $\sigma'' = \sup(\lfloor \kappa_i'' \rfloor)_{i=0}^\infty$. Since $\sigma_n \preceq \sigma'$ and $\langle c_2, \sigma'\rangle$ never gets stuck, it follows that $\langle c_2, \sigma_n\rangle$ never gets stuck as well. However, this is a contradiction as δ must be in some config of the infinite induced run of $\langle c_2, \sigma_n\rangle$.

Hence, we are done. □

Proposition E.7. *Assume $(\kappa_i)_{i=0}^\infty$ is the infinite induced runs of $\kappa = \langle c_1; c_2, \sigma\rangle$. Then for every n, if $\sigma_n = \overline{\pi}_{c_1}(\kappa, n)$ and $\langle c_2, \sigma_n\rangle \rightsquigarrow_n \kappa_n'$ then $\lfloor \kappa_{n+1} \rfloor \preceq \lfloor \kappa_n' \rfloor \preceq \lfloor \kappa_{2n+1} \rfloor$.*

Proof. By induction on the number of steps n. □

Proof of Prop. E.5. Assume that σ ⚙$_{c_1}$ σ' and σ' ⚙$_{c_2}$ σ''. By Prop. E.6, there exists $(\kappa_i)_{i=0}^\infty$ which is the induced run of $\langle c_1; c_2, \sigma\rangle$. It remains to show that $\sigma'' = \sup(\lfloor \kappa_i \rfloor)_{i=0}^\infty$, or equivalently:

(1) $\lfloor \kappa_n \rfloor \preceq \sigma''$ for every n, and
(2) For every $\epsilon > 0$, there exists some N such that $\|\lfloor \kappa_n \rfloor\| > \|\sigma''\| - \epsilon$ for every $n \geq N$.

For each n, we define $\sigma_n \triangleq \overline{\pi}_{c_1}(\kappa, n)$. Also, note that $\sup(\sigma_i)_{i=0}^\infty = \sup(\kappa_i^{c_1})_{i=0}^\infty = \sigma'$ where $(\kappa_i^{c_1})_{i=0}^\infty$ is the induced run of $\langle c_1, \sigma\rangle$.

For (1), note that $\sigma_n \preceq \sup(\sigma_i)_{i=0}^\infty = \sigma'$. From σ' ⚙$_{c_2}$ σ'', we deduce that the config $\langle c_2, \sigma'\rangle$ never gets stuck. Consequently, the config $\langle c_2, \sigma_n\rangle$ where $\sigma_n \preceq \sigma'$ never gets stuck as well. In particular, there exists some config κ_n' such that $\langle c_2, \sigma_n\rangle \rightsquigarrow_n \kappa_n'$ and $\lfloor \kappa_n' \rfloor \preceq \sigma''$. By Prop. E.7, we have $\lfloor \kappa_{n+1} \rfloor \preceq \lfloor \kappa_n' \rfloor$ and so $\lfloor \kappa_{n+1} \rfloor \preceq \sigma''$. Also, $\lfloor \kappa_0 \rfloor = \lfloor \langle c_1; c_2, \sigma\rangle \rfloor = \kappa_\emptyset \preceq \sigma''$. Hence, we conclude that $\lfloor \kappa_n \rfloor \preceq \sigma''$ for every n.

For (2), as $\sup(\sigma_i)_{i=0}^\infty = \sigma'$, there exists some N_1 such that $\|\sigma_n\| > \|\sigma'\| - \epsilon/2$ for every $n \geq N_1$. As σ' ⚙$_{c_2}$ σ'', there exists some $(\kappa_i^{c_2})_{i=0}^\infty$ which is the induced run of $\langle \sigma', c_2\rangle$ and $\sigma'' = \sup(\lfloor \kappa_i^{c_2} \rfloor)_{i=0}^\infty$. Accordingly, there exists some N_2 such that $\left\|\lfloor \kappa_n^{c_2} \rfloor\right\| > \|\sigma''\| - \epsilon/2$ for every $n \geq N_2$. On the other hand, for each $n \geq N_1$, recall that $\sigma_n \preceq \sigma'$ and $\|\sigma_n\| > \|\sigma'\| - \epsilon/2$. Consequently, there exists some

sequence $(\kappa_i^{\sigma_n})_{i=0}^{\infty}$ which is the induced run of $\langle \sigma_n, c_2 \rangle$ and $\left\| \lfloor \kappa_i^{\sigma_n} \rfloor \right\| > \left\| \lfloor \kappa_i^{c_n} \rfloor \right\| - \epsilon/2$ for every i. Thus for every $n \geq \max\{N_1, N_2\}$, we have $\left\| \lfloor \kappa_i^{\sigma_n} \rfloor \right\| > \|\sigma''\| - \epsilon$. By Prop. E.7, we have $\|\lfloor \kappa_n \rfloor\| > \|\sigma''\| - \epsilon$ for every $n \geq 2 \times \max\{N_1, N_2\} + 1$. Hence, we are done. □

E.7 Proof of While

Proof of While. We want to prove the following rule:

$$\frac{\begin{array}{cc} P(n) \vdash (1 - \sum_{i=0}^{n} \rho_i) \cdot \langle b \rangle & Q(n) \vdash \rho_n \cdot \langle \neg b \rangle \\ \{P(n)\}\, c\, \{P(n+1) \boxplus Q(n+1)\} & \sum_{i=0}^{\infty} \rho_i = \rho \leq 1 \end{array}}{\{I(0)\}\ \textbf{while}\ b\ \textbf{do}\ c\ \left\{\rho \cdot \langle \neg b \rangle \wedge \left(\boxplus_{i=0}^{\infty} Q(i)\right)\right\}} \text{[While]}$$

Let $\sigma \models P(0) \boxplus Q(0)$ such that $R = \mathrm{fv}(b) \cup \mathrm{fv}(c) \subseteq \mathrm{vdom}(\sigma)$. For convenience, we let[8] $C \triangleq$ **while** b **do** c and $C_n \triangleq (\textbf{if}\ b\ \textbf{do}\ c)^n$. We unroll the while loop n times as $C_n; C$ where C_n captures the first n iterations. Note that the following Hoare triple holds:

$$\{P(0) \boxplus Q(0)\}\, C\, \{P(1) \boxplus Q(0) \boxplus Q(1)\}$$

By induction on n, we can show that the following triple also holds for every $n \geq 0$:

$$\{P(0) \boxplus Q(0)\}\, C_n\, \left\{P(n) \boxplus (\boxplus_{i=0}^{n} Q(i))\right\}$$

First, we show that the config $\langle C, \sigma \rangle$ never gets stuck. For the sake of contradiction, assume that it gets stuck after n steps. This boils down to the fact that there exists some basic config $\delta = \langle c', m \rangle$ that cannot make the next step. Assume that δ is executing the k^{th} iteration of the while loop. This is clearly a contradiction because the Hoare triple $\{P(0) \boxplus Q(0)\}\, C_k\, \left\{P(k) \boxplus (\boxplus_{i=0}^{k} Q(i))\right\}$ implies that every basic config cannot get stuck in the first k iterations. Consequently, there exists some infinite induced run $(\kappa_i)_{i=0}^{\infty}$ of $\langle C, \sigma \rangle$. Let $\sigma' \triangleq \sup(\lfloor \kappa_i \rfloor)_{i=0}^{\infty}$. We are left to show that $\sigma' \models \rho \cdot \langle \neg b \rangle \wedge (\boxplus_{i=0}^{\infty} Q(i))$.

Recall that for every n, the triple $\{P(0) \boxplus Q(0)\}\, C_n\, \left\{P(n) \boxplus (\boxplus_{i=0}^{n} Q(i))\right\}$ holds. Also, note that $\sigma \models P(0) \boxplus Q(0)$. Accordingly, for every n, there exists some state σ'_n such that:

$$\sigma \circledcirc_{C_n} \sigma'_n \quad \text{and} \quad \sigma'_n \models P(n) \boxplus (\boxplus_{i=0}^{n} Q(i))$$

Let σ_n^P, σ_n^Q satisfy $\sigma_n^P \oplus \sigma_n^Q = \sigma'_n$ and $\sigma_n^P \models P(n)$, $\sigma_n^Q \models \boxplus_{i=0}^{n} Q(i)$. It follows that:

$$\sigma_n^P \models \left(1 - \sum_{i=0}^{n} \rho_i\right) \cdot \langle b \rangle \quad \text{and} \quad \sigma_n^Q \models \left(\sum_{i=0}^{n} \rho_i\right) \cdot \langle \neg b \rangle$$

Since C_n contains the first n iterations of the while loop C, we have $\sigma_n^Q \preceq \sigma'$ and $\sigma_{n+1}^Q = \sigma_n^Q \oplus \sigma_{n+1}'^Q$ where $\sigma_{n+1}'^Q \models Q(n+1)$ for every n. By letting $\sigma_0'^Q \triangleq \sigma_0^Q$, we deduce that $\sigma_n^Q = \oplus_{i=0}^{n} \sigma_n'^Q$ where $\sigma'^Q(i) \models Q(i)$ for every i. As the sequence $(\sigma_i^Q)_{i=0}^{\infty}$ is $\preceq$-ordered, its supremum $\sigma^Q \triangleq \sup(\sigma_Q^i)_{i=0}^{\infty}$ exists. As $\sigma_n^Q \preceq \sigma'$ for every n, it follows that $\sigma^Q \preceq \sigma'$ as well. Note that $\left\|\sigma^Q\right\| = \lim_{n\to\infty} \sum_{i=0}^{n} \rho_i = \rho$ and every world $m \in \mathrm{dom}(\sigma^Q)$ is contained in some state σ_n^Q. Thus, $\sigma^Q \models \rho \cdot \langle \neg b \rangle$. Also, we have $\sigma^Q = \lim_{n\to\infty} \oplus_{i=0}^{n} \sigma_i'^Q$ and $\sigma_i'^Q \models Q(i)$ for every i. Thus $\sigma^Q \models \oplus_{i=0}^{\infty} Q(i)$ and so:

$$\sigma^Q \models \rho \cdot \langle \neg b \rangle \wedge \left(\oplus_{i=0}^{\infty} Q(i)\right)$$

We are done if we can show that $\sigma' = \sigma^Q$. Recall that $\sigma^Q \preceq \sigma'$ and so it suffices to show that $\|\sigma'\| = \left\|\sigma^Q\right\|$ or $\|\sigma'\| = \rho$. For the sake of contradiction, assume that $\|\sigma'\| < \rho$. Let $\epsilon > 0$ satisfy $\epsilon > \rho - \|\sigma'\|$. As $\sigma \circledcirc_C \sigma'$, there must exists some N such that the total weight of the converging state after unrolling the loop N times and executing C_N exceeds $\|\sigma'\| - \epsilon$. Formally, there exits some

[8] We let $c^0 \triangleq$ **skip**, $c^1 \triangleq c$, and $c^{n+2} \triangleq c; c^{n+1}$

N and $\sigma_b, \sigma_{\neg b}$ such that $\sigma \circledcirc_{C_N} (\sigma_b \oplus \sigma_{\neg b})$ where $\sigma_b \models [b], \sigma_{\neg b} \models [\neg b]$ and $\|\sigma_{\neg b}\| > \|\sigma'\| - \epsilon$. On the other hand, the triple $\{P(0) \boxplus Q(0)\}\, C_N\, \left\{P(n) \boxplus (\boxplus_{i=0}^{N} Q(i))\right\}$ holds and so $\sigma \circledcirc_{C_N} (\sigma_N^P \oplus \sigma_N^Q)$ where $\sigma_N^P \models \left(1 - \sum_{i=0}^{N} \rho_i\right) \cdot \langle b \rangle$ and $\sigma_N^Q \models \left(\sum_{i=0}^{N} \rho_i\right) \cdot \langle \neg b \rangle$. Consequently, $\sigma_b = \sigma_N^P$ and $\sigma_{\neg b} = \sigma_N^Q$. Thus, $\left\|\sigma_N^Q\right\| = \|\sigma_{\neg b}\| > \|\sigma'\| - \epsilon > \rho$. However, $\left\|\sigma_N^Q\right\| = \sum_{i=0}^{N} \rho_i \leq \rho$ which implies a contradiction. Hence, we are done.

□

E.8 Proofs of other rules

Proof of Bot.

$$\frac{}{\{\bot\}\, c\, \{Q\}}\,[\textsc{Bot}]$$

Trivial as the precondition is unsatisfiable. □

Proof of Null.

$$\frac{}{\{0\}\, c\, \{0\}}\,[\textsc{Null}]$$

Let $\sigma \models 0$. Then $\|\sigma\| = 0$, i.e. σ is the empty state. Thus $\sigma \circledcirc_c \sigma$ holds trivially. □

Proof of Cons.

$$\frac{P' \vdash P \quad \{P\}\, c\, \{Q\} \quad Q \vdash Q'}{\{P'\}\, c\, \{Q'\}}\,[\textsc{Cons}]$$

Let $\sigma \models P'$. As $P' \vdash P$, we have $\sigma \models P$. From $\{P\}\, c\, \{Q\}$, there exists some σ' such that $\sigma \circledcirc_c \sigma'$ and $\sigma' \models Q$. As $Q \vdash Q'$, we have $\sigma' \models Q'$. This completes the proof. □

Proof of Conj.

$$\frac{\{P\}\, c\, \{Q_1\} \quad \{P\}\, c\, \{Q_2\}}{\{P\}\, c\, \{Q_1 \wedge Q_2\}}\,[\textsc{Conj}]$$

Let $\sigma \models P \wedge P'$. As $\{P\}\, c\, \{Q\}$ holds, there exists some σ' such that $\sigma \circledcirc_c \sigma'$ and $\sigma' \models Q_1$. Similarly, there exists some σ'' such that $\sigma \circledcirc_c \sigma''$ and $\sigma'' \models Q_2$. By Prop. E.3, we have $\sigma' = \sigma''$ and so $\sigma' \models Q_1 \wedge Q_2$. □

Proof of Disj.

$$\frac{\{P_1\}\, c\, \{Q\} \quad \{P_2\}\, c\, \{Q\}}{\{P_1 \vee P_2\}\, c\, \{Q\}}\,[\textsc{Disj}]$$

Let $\sigma \models P_1 \vee P_2$. W.l.o.g. we assume that $\sigma \models P_1$. From $\{P_1\}\, c\, \{Q\}$, there exists some σ' such that $\sigma \circledcirc_c \sigma'$ and $\sigma' \models Q$. Hence, we are done. □

Proof of Dot.

$$\frac{\{P\}\, c\, \{Q\}}{\{\rho \cdot P\}\, c\, \{\rho \cdot Q\}}\,[\textsc{Dot}]$$

Note that $\rho \boxtimes F \dashv\vdash \rho \cdot F$. Thus we have:

$$\cfrac{\cfrac{\{P\}\, c\, \{Q\}}{\{\rho \boxtimes P\}\, c\, \{\rho \boxtimes Q\}}\,\textsc{Frame}}{\{\rho \cdot P\}\, c\, \{\rho \cdot Q\}}\,\textsc{Cons}$$

□

Proof of Cases.

$$\frac{\forall 1 \le i \le n.\ \{P_i\}\, c\, \{Q_i\}}{\left\{\boxplus_{i=1}^{n} \rho_i \cdot P_i\right\} c \left\{\boxplus_{i=1}^{n} \rho_i \cdot Q_i\right\}}\text{[Cases]}$$

We have:

$$\dfrac{\dfrac{\forall i \in [1,n].\ \{P_i\}\, c\, \{Q_i\}}{\forall i \in [1,n].\ \{\rho_i \cdot P_i\}\, c\, \{\rho_i \cdot Q_i\}}\ \textsc{Dot}}{\left\{\boxplus_{i=1}^{n} \rho_i \cdot P_i\right\} c \left\{\boxplus_{i=1}^{n} \rho_i \cdot Q_i\right\}}\ \textsc{Split}$$

□

Proof of If.

$$\frac{\{P/[b]\}\, c_1\, \{Q_1\} \quad \{P/[\neg b]\}\, c_2\, \{Q_2\}}{\{P\}\ \textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2\ \{Q_1 \boxplus Q_2\}}\text{[If]}$$

Rewrite **if** b **then** c_1 **else** c_2 as $c_1[b]c_2$ and note that $P \vdash P/[b] \boxplus P/[\neg b]$. Then:

$$\dfrac{\dfrac{P/[b] \vdash [b]}{\{P/[b]\}\, c_1[b]c_2\, \{Q_1\}}\ \textsc{Choice, Null, Split, Cons} \quad \dfrac{P/[\neg b] \vdash [\neg b]}{\{P/[\neg b]\}\, c_1[b]c_2\, \{Q_2\}}\ \textsc{Choice, }\ldots}{\{P\}\, c_1[b]c_2\, \{Q_1 \boxplus Q_2\}}\ \textsc{Split}$$

□

Proof of DAssign.

$$\frac{\vec{v} = \bigoplus_{i=1}^{n} (\rho_i : v_i)}{\{1\}\, x := \vec{v}\, \{\langle x = \vec{v}\rangle\}}\text{[DistA]}$$

We have:

$$\dfrac{\dfrac{\dfrac{}{\forall i \in [1,n].\ \{1\}\, x := v_i\, \{\langle x = v_i\rangle\}}\ \textsc{Assign}}{\left\{\bigoplus_{i=1}^{n} \rho_i\right\} x := \vec{v} \left\{\bigoplus_{i=1}^{n} (\rho_i \cdot \langle x := v_i\rangle)\right\}}\ \textsc{Cases}}{\{1\}\, x := \vec{v}\, \{\langle x := \vec{v}\rangle\}}\ \textsc{Cons}$$

□

Proof of FWhile.

$$\frac{\begin{array}{c} n \ge 0 \quad I(n) \vdash \langle \neg b\rangle \quad \mathsf{fresh}(k) \\ I(k) \wedge [0 \le k < n] \vdash \langle b\rangle \quad \{I(k) \wedge [k < n]\}\, c\, \{I(k+1)\} \end{array}}{\{I(0)\}\ \textbf{while}\ b\ \textbf{do}\ c\ \{I(n) \wedge \langle \neg b\rangle\}}\text{[FWhile]}$$

We apply While with the following invariant:

$$P(k) \triangleq \begin{cases} I(k) & \text{, if } k < n \\ 0 & \text{, otherwise} \end{cases} \qquad Q(k) \triangleq \begin{cases} I(k) & \text{, if } k = n \\ 0 & \text{, otherwise} \end{cases} \qquad \rho_k \triangleq \begin{cases} 1 & \text{, if } k = n \\ 0 & \text{, otherwise} \end{cases}$$

It can be checked that $\sum_{i=0}^{\infty} \rho_i = 1$. Also, for every $k \ge 0$, we have:

$$P(k) \vdash \left(1 - \sum_{i=0}^{k} \rho_i\right) \cdot \langle b\rangle \quad \text{and} \quad Q(k) \vdash \rho_i \cdot \langle \neg b\rangle$$

Furthermore, $\{P(k)\}\, c\, \{P(k+1) \boxplus Q(k+1)\}$ holds. The case $k < n$ follows from the premise whereas the case $k \ge n$ follows from the Null rule. Thus we obtain the following postcondition for the while loop:

$$\langle \neg b\rangle \wedge \left(\boxplus_{i=0}^{\infty} Q(i)\right)$$

Finally, note that $\boxplus_{i=0}^{\infty} Q(i)$ implies $I(n)$. Hence, we are done.

□

Proof of For.

$$\frac{n_1 \le n_2 + 1 \quad i \notin \mathsf{fv}(I) \cup \mathsf{vars}(c) \quad \{\langle I(i) \wedge i \le n_2 \rangle\}\, c\, \{\langle I(i+1) \rangle\}}{\{\langle I(n_1) \rangle\}\, \textbf{for } i := [n_1, n_2] \textbf{ do } c\, \{\langle I(n_2+1) \rangle\}}\,[\textsc{For}]$$

Note that $\textbf{for } i := [n_1, n_2] \textbf{ do } c \equiv i := n_1; \textbf{while } i \le n_2 \textbf{ do } \{c; i := i+1\}$. Start with the precondition $\langle I(n_1) \rangle$, after the assignment $i := n_1$, we obtain $\langle I(i) \wedge i = n_1 \rangle$(i.e. using DFrame and Assign). We then apply FWhile with the following invariant where k is a fresh variable:

$$\hat{I}(k) \triangleq \langle I(k + n_1) \wedge i = k + n_1 \rangle$$

Clearly, $\hat{I}(k) \vdash \langle i \le n_2 \rangle\ \forall k \in [0, n_2 - n_1]$ and $\hat{I}(n_2 - n_1 + 1) = \langle I(n_1 + 1) \wedge i = n_2 + 1 \rangle$ implies that $\langle \neg(i \le n_2) \rangle$. It remains to check that the triple $\left\{\hat{I}(k) \wedge [k \le n_2 - n_1]\right\}\, c; i := i + 1\, \left\{\hat{I}(k+1)\right\}$ holds. Given the precondition $\langle I(k) \wedge i = k + n_1 \wedge k \le n_2 - n_1 \rangle$, we deduce that $\langle i \le n_2 \rangle$. As the triple $\{\langle I(i) \wedge i \le n_2 \rangle\}\, c\, \{\langle I(i+1) \rangle\}$ holds and $i \notin \mathsf{mv}(c)$, we arrive at the following postcondition of c (i.e. using DFrame):

$$\langle I(k + n_1 + 1) \wedge i + 1 = k + 1 + n \rangle$$

After the assignment $i := i + 1$, we arrive at:

$$\langle I \wedge k + 1 = i - n_1 \rangle \quad \text{which is} \quad \hat{I}(k+1)$$

Hence, we are done. □

Proof of SWhile.

$$\frac{I \vdash 1 \quad \mathsf{convex}(I/\langle \neg b \rangle) \quad \rho < 1 \quad \{I/\langle b \rangle\}\, c\, \{I \wedge (\rho \cdot \langle b \rangle \boxplus \overline{\rho} \cdot \langle \neg b \rangle)\}}{\{I\}\, \textbf{while } b \textbf{ do } c\, \{I/\langle \neg b \rangle\}}\,[\textsc{SWhile}]$$

Since $I \vdash 1$, there exists some probability β such that $I \vdash (\beta \wedge I/[b]) \boxplus (\overline{\beta} \wedge I/[\neg b])$. We can further assume $\beta > 0$ as the cases $\beta = 0$ is easy (i.e. we apply While with the invariant $P(n) \triangleq 0$, $Q(0) = I, Q(n+1) = 0$, and $\rho_0 = 1, \rho_{n+1} = 0$). As $\beta > 0$, we can rewrite I as $\beta \cdot (I/\langle b \rangle) \boxplus (\overline{\beta} \wedge I/[\neg b])$. Thanks to the Cases rule, it is sufficient to prove $\{I/\langle b \rangle\}\, \textbf{while } b \textbf{ do } c\, \{I/\langle \neg b \rangle\}$.

Note that from $I \wedge (\rho \cdot \langle b \rangle \boxplus \overline{\rho} \cdot \langle \neg b \rangle)$, we can deduce that $\rho \cdot (I/\langle b \rangle) \boxplus \overline{\rho} \cdot (I/\langle \neg b \rangle)$. We then apply the below BWhile rule with $P \triangleq I/\langle b \rangle$, $Q \triangleq I/\langle \neg b \rangle$, $\alpha = \rho$. Accordingly, we are able to infer the post-condition Q as desired. □

Proof of BWhile.

$$\frac{P \vdash \langle b \rangle \quad Q \vdash \langle \neg b \rangle \quad \mathsf{convex}(Q) \quad \{P\}\, c\, \{\alpha \cdot P \boxplus (1-\alpha) \cdot Q\} \quad \alpha \in [0,1)}{\{P\}\, \textbf{while } b \textbf{ do } c\, \{Q\}}\,[\textsc{BWhile}]$$

We apply the While rule with the following invariant:

$$P(k) \triangleq \alpha^k \cdot P \qquad Q(k) \triangleq \begin{cases} 0 & \text{, if } k = 0 \\ (1-\alpha)\alpha^k \cdot Q & \text{, otherwise.} \end{cases} \qquad \rho_k \triangleq \begin{cases} 0 & \text{, if } k = 0 \\ (1-\alpha)\alpha^k & \text{, otherwise.} \end{cases}$$

It follows that $P(k) \vdash (1 - \sum_{i=0}^{k} \rho_i) \cdot \langle b \rangle$ and $Q(k) \vdash \rho_k \cdot \langle \neg b \rangle$. Furthermore:

$$\frac{\{P\}\, c\, \{\alpha \cdot P \boxplus (1-\alpha) \cdot Q\}}{\left\{\alpha^{k+1} \cdot P\right\}\, c\, \left\{\alpha^{k+2} \cdot P \boxplus (1-\alpha)\alpha^{k+1} \cdot Q\right\}}\ \textsc{Dot, Cons}$$

And so $\{P(k)\}\, c\, \{P(k+1) \boxplus Q(k+1)\}$ holds for every k. Note that $\sum_{i=0}^{\infty} \rho_i = 1$. Thus we obtain the following postcondition

$$\langle \neg b \rangle \wedge \left(\boxplus_{i=0}^{\infty}\, Q(i)\right)$$

Finally, note that Q is convex and $\sum_{i=0}^{\infty}\ \rho_i = 1$. Thus, $\boxplus_{i=0}^{\infty}\, Q(i)$ implies Q. Hence, we are done. □

F Proof of lemmas in Section 7

Proof of Lemma 7.1. **(PN)⇒(II)**: Assume $\vec{a} = \sum_{i\in I} \rho_i \langle a_i \rangle$. As m is immutable, $m \notin \mathsf{mv}(\Gamma)$. Thus:

$$\dfrac{\dfrac{\dfrac{\forall i \in I.\ \{\langle m = a_i\rangle \boxtimes F\}\, \Gamma\, \{c = \vec{v}\}}{\forall i \in I.\ \{\langle m = a_i\rangle \boxtimes F\}\, \Gamma\, \{\langle m = a_i\rangle \wedge c = \vec{v}\}}\ [\textsc{DFrame}]}{\forall i \in I.\ \{\langle m = a_i\rangle \boxtimes F\}\, \Gamma\, \{\langle m = a_i\rangle \boxtimes (c = \vec{v})\}}\ [\textsc{Cons}]}{\dfrac{\{\boxplus_{i\in I}\, \rho_i \bullet \langle m = a_i\rangle \boxtimes F\}\, \Gamma\, \{\boxplus_{i\in I}\, \rho_i \bullet \langle m = a_i\rangle \boxtimes (c = \vec{v})\}}{\{(m = \vec{a}) \boxtimes F\}\, \Gamma\, \{(m = \vec{a}) \boxtimes (c = \vec{v})\}}\ [\textsc{Cons}]}\ [\textsc{Dot},\textsc{Split}]$$

(II)⇒(PN): We pick $\vec{a} = \langle a \rangle$. Thus:

$$\dfrac{\{\langle m = a\rangle \boxtimes F\}\, \Gamma\, \{\langle m = a\rangle \boxtimes (c = \vec{v})\}}{\{\langle m = a\rangle \boxtimes F\}\, \Gamma\, \{c = \vec{v}\}}\ [\textsc{Cons}]$$

Hence, we are done. □

Proof of Lemma 7.2. Direct from applying the Assign rule together with the DFrame rule and IFrame rule. □

Proof of Lemma 7.3. (1). It suffice to prove the case $n = 2$ as the general case follows by applying induction on n. More specifically, we want to show that $\sigma \models \mathsf{U}(x_1, x_2)$ iff $\sigma \models \mathsf{U}(x_1) \boxtimes \mathsf{U}(x_2)$ where x_1, x_2 are drawn from some non-empty finite sample set $S = \{s_1, \ldots, s_m\}$. By Lemma 4.12, we have:

$$\begin{array}{ll} & \sigma \models \mathsf{U}(x_1, x_2) \\ \Leftrightarrow & \pi(\sigma, \{x_1, x_2\}) \models \mathsf{U}(x_1, x_2) \\ \Leftrightarrow & \pi(\sigma, \{x_1, x_2\}) = \oplus_{i,j=1}^{m} \delta((x_1, x_2 \mapsto_m s_i, s_j), 1/m^2) \\ \Leftrightarrow & \pi(\sigma, \{x_1, x_2\}) = \left[\oplus_{i=1}^{m} \delta((x_1 \mapsto_m s_i), 1/m)\right] \otimes \left[\oplus_{j=1}^{m} \delta((x_2 \mapsto_m s_j), 1/m)\right] \\ \Leftrightarrow & \pi(\sigma, \{x_1, x_2\}) \models \mathsf{U}(x_1) \boxtimes \mathsf{U}(x_2) \\ \Leftrightarrow & \sigma \models \mathsf{U}(x_1) \boxtimes \mathsf{U}(x_2) \end{array}$$

(2). Start from the precondition:

$$\mathsf{U}(x) \boxtimes \langle y = \hat{a} \rangle$$

We unfold $\mathsf{U}(x)$ and distribute x to y:

$$\boxplus_{\hat{s}\in S^n}\ 1/m^n \cdot \langle x, y = \hat{s}, \hat{a} \rangle$$

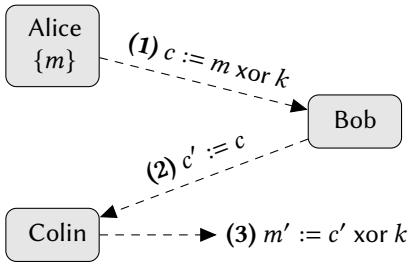
We then apply the Cases rule and the Assign rule to derive the following postcondition of the assignment $z := x \oplus y$:

$$\boxplus_{\hat{s}\in S^n}\ 1/m^n \cdot \langle x, y, z = \hat{s}, \hat{a}, \hat{s} \oplus \hat{a} \rangle$$

Finally, note that the mapping $\lambda \hat{s}.\ \hat{s} \oplus \hat{a}$ is bijective. Thus the distribution of z is uniform. Since x is not modified, we can apply the DFrame to derive that the distribution of x is the same (i.e. uniform). Hence, we are done. □

Proof of Lemma 7.4. Assume $\vec{v} = \sum_i \rho_i \langle \hat{u}_i \rangle$ and let $\alpha = 1/p$. Then:

$$(\hat{a} = \vec{v}) \boxtimes \mathsf{U}(b) \vdash \hat{a}, b = \sum_i \sum_{v\in S} \rho_i \alpha \langle \hat{u}_i, v \rangle$$



$k := (0.5 : 0) \oplus (0.5 : 1);$
$c := m \text{ xor } k;$
$c' := c;$
$m' := c' \text{ xor } k$

(a) One-Time Pad

1 $\{\langle m = \hat{a}\rangle\}$
2 $\searrow$ $\{1\}$
3 $\quad k := (0.5 : 0) \oplus (0.5 : 1);$
4 $\swarrow$ $\{\mathsf{U}(k)\}$
5 $\{\mathsf{U}(k) \boxtimes \langle m = \hat{a}\rangle\}$
6 $c := m \text{ xor } k;$
7 $\{\mathsf{U}(c) \wedge \mathsf{U}(k) \wedge \langle c = m \text{ xor } k\rangle\}$ //privacy correctness
8 $\searrow \{\langle c = m \text{ xor } k\rangle\} \rightsquigarrow F : \mathsf{U}(c)$ //apply DFrame with the frame F
9 $\quad c' := c;$
10 $\quad \{\langle c' = m \text{ xor } k\rangle\}$
11 $\quad m' := c' \text{ xor } k$
12 $\swarrow \{\langle m' = m\rangle\}$//computation correctness
13 $\{\mathsf{U}(c) \wedge \langle m' = m\rangle\}$

(b) Proof sketch

Fig. 14. Verification of One-Time Pad

Thus

$$\begin{aligned}
\text{LHS} \quad &\vdash \quad \hat{a}, b, c = \textstyle\sum_i \sum_{v\in S} \rho_i \alpha \left\langle \hat{u}_i, v, s - \sum_{j=1}^{n} \hat{u}_i(j) - v \right\rangle \\
&\vdash \quad \boxplus_i \left[\rho_i \cdot \langle \hat{a} = \hat{u}_i\rangle \boxtimes \left(\boxplus_{v\in S} \tfrac{1}{p} \cdot \left\langle b, c =_p v, s - \textstyle\sum_{j=1}^{n} \hat{u}_i(j) - v \right\rangle\right)\right] \\
&\vdash \quad \boxplus_i \left[\rho_i \cdot \langle \hat{a} = \hat{u}_i\rangle \boxtimes (\boxplus_{v\in S} \alpha \cdot \langle c = v\rangle)\right] \\
&\vdash \quad (\boxplus_i \rho_i \cdot \langle \hat{a} = \hat{u}_i\rangle) \boxtimes (\boxplus_{v\in S} \alpha \cdot \langle c = v\rangle) \\
&\vdash \quad (\hat{a} = \vec{v}) \boxtimes \mathsf{U}(c)
\end{aligned}$$

Hence, we are done. □

G General case of the One-Time-Pad example

Here we examine a slightly more complicated OTP scenario in comparison to the one discussed in §2.4. Figure 14 presents the One-Time Pad encryption [21] along with its correctness proof. The protocol involves three entities: Alice, Bob, and Colin where Alice wants to send a private message m to Colin through Bob without Bob knowing about its content. It is assumed that Alice and Colin share a randomly generated n-bit key $k := (0.5 : 0) \oplus (0.5 : 1)$ that they use for the XOR-encryption/decryption. First, Alice encodes m as $c := k \text{ xor } m$ and sends it to Bob, who in turn passes a copy $c' := c$ to Colin. Finally, Colin retrieves the content of m by computing $m' := c' \text{ xor } k$. Assuming that $\langle m = \hat{a}\rangle$ holds initially for some arbitrary value $\hat{a}$, we show that the distribution of c

is uniform — meaning that the content of m is not revealed to Bob — and the content of m' is the same as of m, i.e.:

$$\mathsf{U}(c) \wedge \langle m' = m \rangle$$

We proceed through the proof presented in Fig. 14b. First, we apply lemmas 7.2 and 7.3 to deduce that both k and c have uniform distribution (line 7). Since c is not modified after its assignment, we can apply the DFrame rule to assert that the distribution of c remains uniform until the end of the program. For computation correctness, recall that $\langle c = m \text{ xor } k \rangle$ after the computation of c (line 7). As a result, we can deduce that $\langle m = c \text{ xor } k \rangle$ using the identity $(a \text{ xor } b) \text{ xor } b = a$. Since c' is a copy of c, it follows that $\langle m = c' \text{ xor } k \rangle$ (line 12) as desired.