PROBABILISTIC SEPARATION LOGICS FOR RANDOMIZED ALGORITHMS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Jialu Bao

August 2025

PROBABILISTIC SEPARATION LOGICS FOR RANDOMIZED ALGORITHMS

Jialu Bao, Ph.D.

Cornell University 2025

Randomized algorithms are hard to test, thus accentuating the need for formal methods to ensure their correctness. When probabilistic separation logic was first developed as a formal method for proving probabilistic independence between program variables, it was unclear whether this approach generalizes to weaker forms of probabilistic separation used in program analysis.

We first overview existing work in Bunched logic — the assertion logic underlying separation logic — and probabilistic separation logic for independence in chapter 2.

In chapter 3, we extend probabilistic separation logic to reason about *negative dependence*, a relation in which an increase in one variable makes others less likely to increase. We demonstrate the utility of this program logic by analyzing hash-based data structures, such as Bloom filters.

In chapter 4, we introduce a variation of probabilistic separation logic for reasoning about dependence and independence. Specifically, we use it to establish conditional independence between programs variables in simple programs.

Last, in chapter 5, we present the unary fragment of BLUEBELL to provide a more ergonomic way to reason about conditional independence and independence. We illustrate its application through more intricate examples drawn from cryptography, security, and probabilistic graphical models.

All the program logics developed in this thesis target imperative programs that can sample from probability distributions.

BIOGRAPHICAL SKETCH

Jialu spent the first eleventh years of her life in Ningbo, a coastal city with a long history of fishing and trades. She went on to attend middle school and high school in Hangzhou. After high school, she briefly enrolled at the University of Virginia for one semester before beginning her undergraduate studies at Cornell University as a spring admit. At Cornell, she earned Bachelor of Arts in Math and Computer Science. She then moved to Wisconsin to start her Ph.D. at University of Wisconsin–Madison. After two years, she transferred back to Cornell following her advisor's move. Her doctoral research lies in the field of programming languages, formal verification, and probabilistic programs. After completing her Ph.D. in Computer Science at Cornell, she will start as a postdoctoral researcher at Northeastern University with Prof. Steven Holtzen.

To my family.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my advisor Justin Hsu, without whom this thesis would not have existed. Justin has been such an exceptional teacher, an insightful mentor, and an inspiring role model! Through thoughtful explanations and detailed feedback, he taught me how to think, write, and present more clearly and break down complex problems into manageable parts.

I am also grateful to have Joseph Halpern, Dexter Kozen, and Alexandra Silva on my committee. I was blessed to meet many wonderful teachers, and Joe was one of them. His course "Reasoning about Knowledge" gave a fascinating introduction to epistemic logic and sparked my interest in modal logic. I also feel fortunate to have the opportunity to learn from Dexter Kozen about Kleene Algebra in a class. Outside of the classroom, Dexter's wisdom has also led to many inspiring discussions in PLDG and in the hallway. I am immensely grateful to Alexandra Silva for collaborating with me and hosting me on various occasions, including my visit to her UCL group this year, during which part of this thesis was written. The lab's friendly and intellectually engaging atmosphere made it an ideal place for me to reflect and write.

I am indebted to all my collaborators, Jessica Cho, Simon Dorcherty, Emanuele D'Osualdo, Azadeh Farzan, Marco Gaboardi, Tao Gu, Kun He, John Hopcroft. Justin Hsu, Drashti Pathak, Oliver Richardson, Subhajit Roy, Alexandra Silva, Joseph Tassarotti, Nitesh Trivedi, Xiaodong Xin, and Fabio Zanasi. In particular, I would like to thank Emanuele and Azadeh for their close mentorship during our collaboration. They gave me new perspectives on program logics and showed me fresh ways to approach research problems. I would also like to thank Shuchi Chawla for kindly mentoring me on a project, though it did

not result in a publication.

I am also grateful to have Eli Bingham and Zenna Tavares as my mentors when I interned at Basis, and Ellie Cheng, Ayush Chopra, Poorva Garg, Palka Puri, Raffi Sanna, and Andy Zane as my intern cohorts.

During my undergraduate studies, the courses taught by Paul Ginsparg, Michael Clarkson, and Jon Kleinberg deeply influenced my career choice. I would like to thank them for giving intellectually stimulating lectures and thoughtful assignments. I am also grateful to Michael Macy, Chris Cameron, John Hopcroft, and Nate Foster for mentoring me and introducing me to the world of research.

Although my years in Wisconsin were cast in the shadow of the Covid lock-downs, the wonderful people I met and the beautiful outdoor scenery colored my memories. It has been my pleasure to meet and have great conversations with John Cyphert, Samuel Drews, Patrick Nicodemus, Calvin Smith, Yuhao Zhang, and others in the programming languages group. I also would like to thank Patrick for giving feedback on this thesis. I would like to thank Evangelia Gergatsouli, Yang Guo, Xiating Ouyang, Rojin Rezvan and Laura Stegner for many fun gatherings when we could meet in person. I also greatly appreciate Kyrylo Chernyshov, Yuchen Han, Lu Yang, Yujia Zhang, and other friends for sharing their daily moments remotely and kept me in a good spirit during that period.

After returning to Cornell, I also had great pleasure to be surrounded by fantastic friends and colleagues. Although we did not have a lab officially, other students of Justin (Noah Bertram, Max Fan, Karuna Grewal, Vaibhav Mehta, Kei Imada, Zachary Susag, and Laura Zielinski) and my officemates (Keri D'Angelo, Kangbo Li, Khonzoda Umarova, and Noam Zilberstein) filled that role, offering

knowledge and support whenever it was needed. Their kindness, curiosity, and good humor made both the research and the everyday moments delightful. I am also especially thankful to Mark Moeller and Yulun Yao for being amazing PLDG co-czars – I hope this PL group tradition continues for many years to come! In my last year, the Sunday casual tennis organized by Ayaka Yorihiro and Nitika Saran became a cherished social routine. I had great fun hitting with Ayaka, Ethan Yang, Max, Nitika, Rebecca Liu, Yunxi Shen and others. I also feel fortunate to share my Ph.D. journey with Pedro de Amorim, Ryan Doenges, Ali Farahbakhsh, Wen-Ding Li, Yueying Li, Rishabh Madan, Anshuman Mohan, Rolph Recto, Oliver Richardson, Goktug Saatcioglu, Albert Tseng, Nathan Yan, and Alicia Yang. I am deeply thankful for the support they offered and the insights they generously shared. I hope our friendship will continue for many years.

Outside of the CS department, I am fortunate to have Ning Duan, Lijun Zhang, and Yujia Zhang as my close friends locally in Ithaca. I am also grateful to have had Shi Tang and Yu Pan as best friends since high school — our enduring friendship offers a sanctuary for reflecting on my personal journey and sharing my feelings.

Last, I am infinitely grateful to my parents Fang and Guanzhen, 外婆 Yazhen, 外公 Meiding, and the rest of my family for their unconditional love, unwavering support, and the countless cherished moments we have shared.

Hangzhou, China

July, 2025

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Probabilistic Programs

Whether one believes the world we live in is fundamentally deterministic or the result of some dice rolling, probability offers a useful lens to model and analyze various phenomena. For example: "Would it rain tomorrow?" "Who would win US Open this year?" "How unlikely are these constituencies to be divided in a such biased way? " These are all scenarios where we can use probability to distill our uncertainties into some quantities.

For computer programs, probabilities again play an important role. For instance, when an algorithm's efficiency can vary largely depending on the input, it makes sense to consider the *average cost*, which makes an assumption about the program's distribution and then computes the expected value of the algorithm's cost, when executed on inputs drawn from that distribution. Here, the probability is not used by the program — we just use it to model our uncertainties; moreover, we can also harness probabilistic mechanisms in the design of the algorithms. For example, while the deterministic version of Quicksort Hoare [1961] needs $O(n^2)$ comparisons to sort n elements at the worst case, the randomized Quicksort partitions the array based on a randomized pivot and makes the average cost for every scenario (including the worst case) $O(n \log n)$. Similarly, using random bits allows algorithms to guarantee better average performance against adversaries in distributed systems [Fischer et al., 1985, Lynch, 1996] and cache management [Psounis and Prabhakar, 2001, Suri, 2020].

Randomness also has many other usages in algorithms. Randomized assignments ensures fairness when we want different outcomes all have possibilities to occur, each with the desired probability. In cryptography, randomness makes the secrets hard to guess and thus leads to security guarantees. Randomness also allows us to trade accuracy for efficiency. For instance, although finding solutions for integer linear programming is NP-hard, randomized rounding [Raghavan and Tompson, 1987] finds solutions with a good probability and runs in polynomial time. In primality testing, where the goal is to determine whether a given number n is prime, Miller–Rabin primality test [Rabin, 1980] probabilistically samples integers that may witness n to be composite and in polynomial time determines n's primality, with an exponentially small probability of mistaking a composite number as a prime.

Following early breakthroughs in randomized algorithms, the seminal work Kozen [1981] gives formal semantics for a programming language that allows the usage of randomness. Roughly, probabilistic programs in Kozen [1981] extend standard imperative programming language with a command for sampling from distributions. Kozen presents two natural and equivalent semantics of probabilistic programs: the first reflects the view of probabilistic programs as standard programs reading a tape of random bits, and the second directly interprets probabilistic programs as maps from distributions to distributions.

Expressing randomized algorithms as probabilistic programs pins down their behaviors precisely through the formal semantics and then facilitates rigorous analysis of these algorithms. In the study of programming languages, researchers developed various formal methods for systematically checking the correctness of programs. One kind of formal method is *deductive verification*,

where we use an expressive logic to specify the desired behavior of a program and apply logical rules, i.e., deduction, to prove the validity of such specifications. This thesis will focus on the deductive verification of probabilistic programs through program logic.

It is worth noting that, besides randomized algorithm, probabilistic programming languages are also developed and implemented to describe complicated probabilistic processes succinctly [Gordon et al., 2014]. There, an important addition to the language is the conditioning operator, sometimes also called the *observe statement*, which transforms a distribution to a conditional distribution. Notably, the effect of the conditioning operator can be simulated using a while loop, but adding the conditioning operator in the language facilitates potentially different implementations of this command, whose effect is computationally expensive to implement exactly and often only approximated. Historically, the design of randomized algorithms rarely use the conditioning operator, Since we focus on verifying probabilistic programs for randomized algorithms in this thesis, and we leave out the conditioning operator in our probabilistic programming language.

1.2 Independence and Dependencies in Programs

In our discourse, we consider probabilistic program variables as a superset of deterministic program variables: each probabilistic program variable's value is sampled from a distribution, and a deterministic program variable can be considered as sampling its value from a point-mass distribution. We also abbreviate "probabilistic program variables" as "variables" sometimes.

An important and ubiquitous relation between two probabilistic programs variables is probabilistic independence. Independence between two variables means that their values are unrelated, i.e., knowing the outcome of one of the variables does not change one's knowledge of the distribution of the other, vice versa. Intuitively (and somewhat tautologically), two probabilistic variables are independent if they are derived from fresh and distinct sources of randomness, like two coin flips. In contrast, a coin flip x and the derived variable $x + 1$ are clearly not independent because the value of x dictates the value $x + 1$ gets. However, two variables that use a shared source of randomness and even have logical dependency can also be probabilistically independent. For instance, in one-time-pad encryption, we assume a l-bit message m is drawn from some distribution over binary strings, and we draw a key k from the uniform distribution over l-bit binary strings and encrypt the message m into c, defined to be $m \mathsf{xor} k$. This ciphered message c is probabilistically independent of the original message m, though it is also clearly derived from the original message. Also, in the degenerated case, when a variable is deterministic, then knowing its value does not give any information about how the outcome of other variables are sampled. Because of that, deterministic variables are independent from any other variables.

Sometimes, two variables A, B are not exactly probabilistic independent but, when we fix the value of a third variable C, then the values of A and B become irrelevant. This is a case where A and B are *conditionally independent* given C. Or, two variables A, B may be *negatively associated* in that when A attains a higher value, then B tends to attain a lower value. We will refer to such relations between program variables concerning their probabilistic dependencies and independence as (in)dependencies.

Knowing the (in)dependencies between program variables can be extremely helpful in program analysis for multiple reasons. First, sometimes ensuring the desired (in)dependencies is straightforwardly the goal. For example, in cryptography, perfect security means that the public information is independent from the secrets. In multi-party secured computation, multiple parties want to compute a value that depends on each party's secrets without divulging their own secrets. Perfect security is not an appropriate goal here because the different parties want the computed result to be made public and often that value is not independent from their secrets. A more appropriate goal is the conditional independence of each party's view and the other parties' secrets *given* the outcome of the computed result, so establishing that conditional independence proves the protocol correct.

Second, (in)dependencies facilitates further analysis. For example, the law of large numbers says that, if we draw a large number of independent samples from a distribution, then the sample average of the results converges to the expected value. Various inequalities upper-bound the probability that the sample average deviates from the expected value for more than certain amount — these inequalities are called the "concentration bounds." Concentration bounds can be applied, for instance, to upper-bound the probability that randomized Quicksort terminates within some desired time bound on an arbitrary instance, because the choice of each randomized pivot is independent and it is unlikely to always choose the "bad" pivots. Some concentration bounds also hold for negatively associated variables. Intuitively, if one variable getting a bigger outcome means the others get smaller outcomes, then their deviation from the expected value would likely cancel out. As an application, concentration bounds also help analyzing the collision probability or overflowing probability of hash algo-

rithms: when we hash a fixed number of items into a set of buckets, one bucket getting more items means less items can go to the other buckets, so the number of items hashed to different buckets are negatively associated, and thus we can apply concentration bounds to deduce that it is unlikely for many buckets to get a lot of items.

Other than program analysis, we can also leverage (in)dependencies to represent a probabilistic model more concisely [Koller and Friedman, 2009], identify parallelizable computations, and perform more efficient probabilistic inference [Holtzen, 2021].

Analyzing (in)dependencies between probabilistic program variables, however, is intricate. First of all, testing probabilistic properties is hard. For deterministic programs, we can run an implementation and test whether a property is violated by the implementation; for probabilistic behaviors, however, testing can only exhibit a finite number of execution traces, from which we cannot conclude (in)dependencies between program variables in the distribution of execution traces with certainty. Second, our mental model of probabilistic (in)dependencies can be unreliable. As we illustrated above through the example of one-time-pad encryption, somewhat counter-intuitively, logically dependent variables can also be probabilistically independent. As another example, consider the Bloom filter [Bloom, 1970], a widely-used randomized data structure for membership queries, which is highly space-efficient, at the price of returning false positives sometimes. A Bloom filter stores a relatively small array of 0-1 bits, and an item is mapped to a set of indices on the array using distinct hash functions and the corresponding bits are flipped to 1 when an item is added. When analyzing the Bloom filter's false positive rate, many sources

(e.g., Mullin [1983], Blustein and El-Maazawi [2002]) have mistaken the value on different indices of the Bloom filter as independent,[1] while *they are not* because one index flipped to 1 means other indices are more likely to be 0. A possible explanation of such confusion is that people may intuitively think that independence is preserved through arbitrary composition, while they are not.

These difficulties all speak to the need for deductive verification of (in)dependencies in probabilistic programs, whose rigor allows us to confidently use (in)dependencies in analysis.

1.3 Separation Logic for Independence and Dependencies

Separation logic extends Hoare logic to reason about programs. Originally, it was developed to verify programs that manipulate pointer data structures, i.e., heaps. The core innovation is the introduction of "separating conjunction" (symbolized by $*$), a logical connective that allows assertions about distinct, non-overlapping regions of memory to be combined. Unlike traditional conjunction $P \wedge Q$ that only requires the validity of two assertions P and Q, separating conjunction $P * Q$ also asserts the disjointness of the subheaps validating P and Q. At the program logic level, the signature *frame rule* allows local reasoning about heap manipulations while preserving propositions on disjoint pieces of memory. Using these new assertions and rules, Separation Logic addresses a critical limitation of classical Hoare logic, that reasoning about pointer-manipulating programs was hindered by complex aliasing and interference be-

[1] This issue is first pointed out by Bose et al. [2008], which also attempted to fix it. Christensen et al. [2010] later identified an issue in the definition of Stirling numbers of the second kind in Bose et al. [2008]. Gopinathan and Sergey [2020] formally certifies the analysis using the theorem prover Rocq.

tween memory regions.

The ideas that "we can reason about separate components separately" makes no special assumptions about heaps, so Separation Logic can be a general tool for reasoning about resources that can be separated or shared among different entities. A influential extension of heap-based Separation Logic is Concurrent Separation Logic (CSL) [Brookes, 2007a, Vafeiadis and Parkinson, 2007, Brookes, 2007b], which leverages the separating conjunction to ensure that concurrent modifications to the heap are localized and do not interfere with each other. It has led to practical and scalable verification tools like Infer [Facebook] for automatically verifying properties important to security, concurrency and in other domains.

More recently, probabilistic separation logic (PSL) by Barthe et al. [2019] reappropriates Separation Logic for reasoning about probabilistic programs, with the insight that independence is a separation between different components of a distribution. They do not make the distinction between the store and the heap — both are considered as memories — and build their program logic for probabilistic programs interpreted as maps between distributions over memories. In PSL, the separation conjunction $P * Q$ asserts the independence of the formula P and formula Q by requiring P and Q to use disjoint sets of variables and the two sets of variables to be independent. PSL enjoys a proof system analogous to Separation Logic, also with a frame rule, but instead for establishing probabilistic independence of probabilistic program variables. While Barthe et al. [2019] demonstrates that their program logic, with the help of domain-specific axioms, can establish probabilistic independence in several cryptography-based examples, we want to know how much further we

can push this idea.

Concretely, we ask the following questions:

1. Can we also adapt separation logic for reasoning about "probabilistic separation" notions that are weaker than independence, such as conditional independence or negative association?
2. Can we make the assertion logic more expressive? For instance, existing PSL conflated probabilistic independence and variable disjointness; can we precisely assert probabilistic independence without assuming variable disjointness?
3. Can this style of "probabilistic separation logic" scale to bigger, more complicated programs?

1.4 Outline of the Thesis

In this thesis, we first overview the assertion logic underpinning separation logic, *Bunched Logic* (abbreviated as BI for "the logic of Bunched Implications"), in chapter 2. The original BI is an important stepping stone before we introduce its variations and other practical models of probabilistic separation logic. In chapter 3, we extend probabilistic separation logic to also support compositional reasoning of negative association and call the new logic LINA. In chapter 4, we introduce a new assertion logic DIBI, which extends BI with a *non-commutative* conjunction for modeling dependent resources, and design a program logic CPSL on top of DIBI for proving conditional independence in probabilistic programs. Chapter 3 and Chapter 4 together give a positive answer to

Question 1.

Last, in chapter 5, we focus on the unary fragment of BLUEBELL, a program logic designed for integrating unary and relational reasoning of probabilistic program. The unary fragment of BLUEBELL gives an alternative program logic for proving conditional independence and independence. While CPSL expresses conditional independence using two different conjunctions, BLUEBELL, inspired by Li et al. [2023a], introduces a modality to the logic for conditioning on distributions and expresses conditional independence using the modality and the usual separation conjunction for independence. This new modality also allows us to express probabilistic dependence such as, depending on the outcome v of the variable x, the variable y is distributed as some $\kappa(v)$. Meanwhile, similar to LINA and CPSL, BLUEBELL is a program logic developed for *imperative* probabilistic programs. In BLUEBELL, we are able to decouple the assumption of variable disjointness from assertion of probabilistic independence, using the probabilistic independence BI model proposed by Li et al. [2023a] and *permissions*, a concept developed in the concurrent separation logic for tracking who can read from and write into a resource. This feature also answers Question 2 positively.

In addition, we apply LINA and BLUEBELL on some non-trivial probabilistic programs, demonstrating their potential to scale. CPSL is only applied to smaller examples. One difficulty in applying CPSL to more complicated programs is that, as a result of our design choices, the program logic rules only apply to assertions following certain syntactic restrictions. In designing BLUEBELL, we prioritize the ergonomics and no longer impose syntactic restrictions to assertion logic; instead, all assertions can be used in the program logic rules.

We also see this as a step towards a more scalable probabilistic separation logic, thus making progress in answering Question 3.

We also want to note that chapter 2 is mainly based on prior work Docherty [2019] and Barthe et al. [2019]. Chapter 4 is based on Bao et al. [2021]; Chapter 3 is based on Bao et al. [2022]; Chapter 5 is based on Bao et al. [2025].

CHAPTER 2

BUNCHED LOGIC AND PROBABILISTIC SEPARATION LOGIC

2.1 Background

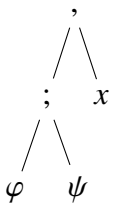
A key feature of Separation Logic is using bunched logic instead of the usual propositional logic or first-order logic for asserting program states. Bunched logic is a *substructural logic* formulated by O'Hearn and Pym [1999]. The usual propositional logic satisfies three structural rules — WEAKENING, CONTRACTION and EXCHANGE. Intuitively, WEAKENING allows one to add unused things to the context; CONTRACTION allows one to contract duplicated things in the context, and EXCHANGE allows one to exchange things in the context. Bunched logic does not require WEAKENING and CONTRACTION. The lack of contraction makes its contexts behave like non-duplicable resources; in addition, the lack of weakening makes its contexts behave like resources that have to be used. While this choice of structural rules is exactly the same as in linear logic, bunched logic also allows contexts joined by another connective ';' that satisfies all three structural rules. That is, in sequent calculus style presentation,

$$\frac{\Gamma \vdash \psi}{\Gamma; \phi \vdash \psi}\ \textsc{Weakening} \qquad \frac{\Gamma; \phi; \phi \vdash \psi}{\Gamma; \phi \vdash \psi}\ \textsc{Contraction}$$

$$\frac{\Gamma_1; \phi; \Gamma_2; \psi; \Gamma_3 \vdash \theta}{\Gamma_1; \psi; \Gamma_2; \phi; \Gamma_3 \vdash \theta}\ \textsc{Exchange-1}$$

$$\frac{\Gamma_1, \phi, \Gamma_2, \psi, \Gamma_3 \vdash \theta}{\Gamma_1, \psi, \Gamma_2, \phi, \Gamma_3 \vdash \theta}\ \textsc{Exchange-2}$$

(a) Substructural rules for bunched logic



(b) An example context

bunched logic has the structural rules in fig. 2.1a. Contexts that interleave these two connectives are tree-structured instead of list-structured, for example, as the context given in fig. 2.1b, thus giving the logic the name *bunched* logic.

Since it allows different ways to combine the contexts, bunched logic provides a flexible foundation for reasoning about resources whose sharing and separation need careful accounting. We can already see it from the connectives in bunched logic. First, the two ways to combine the contexts induce two conjunctions, the multiplicative conjunction $*$ and the additive conjunction $\wedge$,

$$\frac{\Gamma \vdash \phi \qquad \Delta \vdash \psi}{\Gamma, \Delta \vdash \phi * \psi} *\text{-I} \qquad\qquad \frac{\Gamma \vdash \phi \qquad \Delta \vdash \psi}{\Gamma; \Delta \vdash \phi \wedge \psi} \wedge\text{-I}$$

Informally, the assertion $\phi * \psi$ can be used to ensure properties ϕ, ψ hold on separate resources, while $\phi \wedge \psi$ allows us to assert the validity of facts ϕ, ψ without extra requirements. Analogously, bunched logic has a multiplicative implication as well as a standard implication $\rightarrow$,

$$\frac{\Gamma, \phi \vdash \psi}{\Gamma \vdash \phi \mathrel{-\!\!*} \psi} \mathrel{-\!\!*}\text{-I} \qquad\qquad \frac{\Gamma; \varphi \vdash \psi}{\Gamma \vdash \phi \rightarrow \psi} \rightarrow\text{-I}$$

The multiplicative version $\phi \mathrel{-\!\!*} \psi$ asserts that combining current state with a separate resource satisfying ϕ would validate ψ, while $\phi \rightarrow \psi$ simply asserts that fact ϕ implies ψ.

In this chapter, we first give a formal overview of bunched logic, introducing its syntax, semantics, and proof system; we then show that the proof system is sound and complete. All the methodology and proofs of this part are taken from Docherty [2019]. What we aim for is to list the precise definitions and results needed for the rest of the chapters; we also detail some cases of induction proofs omitted in Docherty [2019] to illustrate how they are proved.

It is worth noting that there are varied presentations of bunched logic's semantics in the literature: the original paper by [O'Hearn and Pym, 1999] interprets BI formula over doubly closed categories; early works in separation logic often interpret BI over partial commutative monoids that satisfies extra conditions [Calcagno et al., 2007]; more recent works in higher-order concurrent separation logic use a customized resource algebra whose binary operation is total and may not have a single unit, with extra functions on elements [Jung et al., 2018], etc. We adopt the system from Simon Docherty's thesis [Docherty, 2019], because it provides a uniform account of various bunched logics, accompanied with a completeness proof — we do not know if the proof system is complete with other variations of semantics.

After introducing the metatheory of bunched logic, we introduce a probabilistic separation logic based on bunched logic. First, we describe a concrete bunched logic model $\mathcal{X}_{\mathbb{D}}$ based on probabilistic memories, i.e., distributions over program memories. In this model, separating conjunction can be used to assert probabilistic independence. Then, we define an imperative probabilistic language pWhile that operates on probabilistic memories. Last, in this chapter, we describe a program logic that reasons about pWhile programs with specifications asserted using the bunched logic formulas of the concrete probabilistic model $\mathcal{X}_{\mathbb{D}}$. This program logic is a simplified but also generalized version of probabilistic separation logic in prior work [Barthe et al., 2019] for proving probabilistic independence.

2.2 Bunched Logic (BI)

2.2.1 Syntax and Semantics

The set of BI formulas, Form_{BI}, extends propositional formula with the multiplicative conjunction $P * Q$, and the implication $P \mathrel{-\!\!*} Q$ and the unit I associated with it.

$$P, Q ::= p \in \mathcal{AP} \mid \top \mid \bot \mid I \mid P \wedge Q \mid P \vee Q \mid P \to Q \mid P * Q \mid P \mathrel{-\!\!*} Q$$

BI formulas are interpreted in a kind of mathematical structure named *BI frames*.

Definition 2.2.1 (Downwards-Closed BI Frame)**.** A *Downwards-Closed BI frame* is a structure $\mathcal{X} = (X, \sqsubseteq, \circ, E)$ such that $\sqsubseteq$ is a preorder (i.e., a transitive and reflexive relation), $E \subseteq X$, and $\circ \colon X \times X \to \mathcal{P}(X)$ is a non-deterministic binary operation, satisfying the rules in Figure 2.2.

$z \in x \circ y$	$\to$	$z \in y \circ x;$	(Commutativity)
$w \in t \circ z \ \wedge \ t \in x \circ y$	$\to$	$\exists s \ (s \in y \circ z \ \wedge \ w \in x \circ s);$	(Associativity)
$\exists e \in E \ (x \in e \circ x);$			(Unit Existence)
$e \in E \ \wedge \ e \sqsubseteq e'$	$\to$	$e' \in E;$	(Unit Closure)
$e \in E \ \wedge \ y \in x \circ e$	$\to$	$x \sqsubseteq y;$	(Unit Coherence)
$z \in x \circ y \wedge x' \sqsubseteq x \wedge y' \sqsubseteq y$	$\to$	$\exists z' \ (z' \sqsubseteq z \wedge z' \in x' \circ y').$	(Down-Closed)

Figure 2.2: BI frame requirements (with outermost universal quantification omitted).

Intuitively, X is a set of states, the preorder $\sqsubseteq$ relates two states, the binary operator $\circ$ offers a way of combining states, and E is a set of states that act like units with respect to $\circ$. The binary operator returns a *set* of states instead of a single state, and thus it can be deterministic (at most one state returned) or non-deterministic, partial (empty set returned) or total. In alternative presentations

of BI frames as partial commutative monoids, the binary operator is defined to be a partial map $X \times X \rightharpoonup X$. But the proof of Bunched logic's completeness relies on the frame's admission of non-deterministic models Docherty [2019]. Furthermore, the non-deterministic combination is useful for reasoning about probabilistic states, as we showcase in Chapter 3 for negative dependence. For the preorder, there are two opposite but equally sensible readings of $x \sqsubseteq y$ where x and y are interpreted as resources:

1. y as a resource is an extension of resource x and we can convert y to x by using up some part of y;
2. Or, resource x converts to resource y.

To avoid confusion, in this thesis, we consistently use the first reading. Also, we sometimes write $x \sqsupseteq y$ as an interchangeable notation for $y \sqsubseteq x$.

The frame conditions define properties that must hold for all models of BI. The first three properties (Commutativity), (Associativity), and (Unit Existence) can be viewed as generalizations of familiar algebraic properties of monoids to non-deterministic operations. (Associativity) is only in one direction because, together with (Commutativity), it also implies the other direction: if $s \in y \circ z \wedge w \in x \circ s$, then $\exists w(w \in t \circ z \wedge t \in (x \circ y))$. (Unit Existence) also relaxes the usual unit existence axiom for monoids, which states that there is one element e that is the unit for all other elements with respect to the binary operation, to allow different units $e \in E$ chosen for different x. (Unit Closure) states that the set E is closed under the preorder $\sqsubseteq$. (Unit Coherence) say that if y can be obtained composing x with a unit $e \in E$, then y is an extension of x; roughly, this ensures that E only has elements that behave like units. Last,

(Down-Closed) is another coherence condition for the order $\sqsubseteq$ and the composition $\circ$, which says that for $z \in x \circ y$, then the composition of any x' smaller than x and any y' smaller than y contains an element z' smaller than z. Informally, it says that the resource conversion of the components x, y translates into the resource conversion of the composition z.[1]

We then use a Kripke-style semantics for BI. Given a BI frame, the semantics defines which states in the frame satisfy each formula. Since the semantics is defined inductively on formulas, we first need a specification of which states satisfy the atomic propositions.

Definition 2.2.2 (Valuation and model)**.** A *persistent valuation* is an assignment $\mathcal{V} : \mathcal{AP} \to \mathcal{P}(X)$ of atomic propositions to subsets of states of a BI frame satisfying: if $x \in \mathcal{V}(p)$ and $y \sqsupseteq x$ then $y \in \mathcal{V}(p)$. A BI model $(\mathcal{X}, \mathcal{V})$ is a BI frame $\mathcal{X}$ together with a persistent valuation $\mathcal{V}$.

We now give a semantics to BI formulas in a BI model.

x	$\models_{\mathcal{V}}$	$\top$		always
x	$\models_{\mathcal{V}}$	$\bot$		never
x	$\models_{\mathcal{V}}$	I	iff	$x \in E$
x	$\models_{\mathcal{V}}$	p	iff	$x \in \mathcal{V}(\mathrm{p})$
x	$\models_{\mathcal{V}}$	$P \wedge Q$	iff	$x \models_{\mathcal{V}} P$ and $x \models_{\mathcal{V}} Q$
x	$\models_{\mathcal{V}}$	$P \vee Q$	iff	$x \models_{\mathcal{V}} P$ or $x \models_{\mathcal{V}} Q$
x	$\models_{\mathcal{V}}$	$P \to Q$	iff	for all $y \sqsupseteq x$, $y \models_{\mathcal{V}} P$ implies $y \models_{\mathcal{V}} Q$
x	$\models_{\mathcal{V}}$	$P * Q$	iff	there exist x', y, z s.t. $x \sqsupseteq x' \in y \circ z$, $y \models_{\mathcal{V}} P$ and $z \models_{\mathcal{V}} Q$
x	$\models_{\mathcal{V}}$	$P \mathrel{-\!\!*} Q$	iff	for all y, z s.t. $z \in x \circ y$, $y \models_{\mathcal{V}} P$ implies $z \models_{\mathcal{V}} Q$

Figure 2.3: Satisfaction for BI

Definition 2.2.3 (BI Satisfaction and Validity)**.** Satisfaction at a state x of a model $(\mathcal{X}, \mathcal{V})$ is inductively defined by the clauses in Figure 2.3. P is *valid in a model*,

[1] It is also possible to interpret BI formulas on structures without (Down-Closed) while still ensuring soundness and completeness with respect to usual BI proof system and the persistence of formulas, but other axioms ((Associativity)) needs to be more delicate. The assumption of (Down-Closed) is common in the presentation of BI models in the literature.

$\mathcal{X} \models_{\mathcal{V}} P$, iff $x \models_{\mathcal{V}} P$ for all $x \in \mathcal{X}$. P is *valid*, $\models P$, iff P is valid in all models. $P \models Q$ iff, for all models $(\mathcal{X}, \mathcal{V})$, for any state $x \in \mathcal{X}$, $x \models_{\mathcal{V}} P$ implies $x \models_{\mathcal{V}} Q$.

Where the context is clear, we omit the subscript $\mathcal{V}$ on the satisfaction relation. With the semantics in Figure 2.3, persistence on propositional atoms extends to all formulas:

Lemma 2.2.1 (Persistence Lemma)**.** *For all BI formula P, if $x \models P$ and $y \sqsupseteq x$ then $y \models P$.*

Remark The emphasis on properties being persistent roots back to the history of intuitionistic logic. Classical logic has *the law of excluded middle*, $\vdash p \vee \neg p$, which says that for any property p, either p holds or p does not hold. However, with some readings of formula satisfaction, the law seems to be on precarious ground. For instance, if we interpret $x \models p$ as saying that at state x, the fact p has been verified to be true, and then, $x \models \neg p$ would be saying that at state x, the fact $\neg p$ has been verified to be true, then we should not expect the law of excluded middle to be valid — it is possible that neither p nor $\neg p$ has been verified. This motivates non-classical logic without the law of excluded middle, and furthermore, many properties that motivate such readings of formulas are naturally persistent. For instance, suppose states are ordered by temporal order, if p has been verified to be true at state x, then for every state x' following x, p has been verified to be true at x' too.

$$\frac{}{P \vdash P}\ \textsc{Ax} \qquad \frac{}{P \vdash \top}\ \textsc{Top} \qquad \frac{}{\bot \vdash P}\ \textsc{Bot}$$

$$\frac{P \vdash R \qquad Q \vdash R}{P \vee Q \vdash R}\ \vee\text{-E} \qquad \frac{P \vdash Q_i}{P \vdash Q_1 \vee Q_2}\ \vee\text{-I}$$

$$\frac{P \vdash Q \qquad P \vdash R}{P \vdash Q \wedge R}\ \wedge\text{-I-R} \qquad \frac{Q \vdash R}{P \wedge Q \vdash R}\ \wedge\text{-I-L} \qquad \frac{P \vdash Q_1 \wedge Q_2}{P \vdash Q_i}\ \wedge\text{-E}$$

$$\frac{P \wedge Q \vdash R}{P \vdash Q \to R}\ \to\text{-I} \qquad \frac{P \vdash Q \to R \qquad P \vdash Q}{P \vdash R}\ \to\text{-E}$$

$$\frac{P \vdash R \qquad Q \vdash S}{P * Q \vdash R * S}\ *\text{-Conj} \qquad \frac{P * Q \vdash R}{P \vdash Q \mathrel{-\!\!*} R}\ \mathrel{-\!\!*}\text{-I} \qquad \frac{P \vdash Q \mathrel{-\!\!*} R \qquad S \vdash Q}{P * S \vdash R}\ \mathrel{-\!\!*}\text{-E}$$

$$\frac{}{P \dashv\vdash P * I}\ *\text{-Unit} \qquad \frac{}{P * Q \vdash Q * P}\ *\text{-Comm}$$

$$\frac{}{(P * Q) * R \dashv\vdash P * (Q * R)}\ *\text{-Assoc}$$

Figure 2.4: Hilbert system for BI

2.2.2 Proof System

In the study of logic, we are not only interested in when a formula holds, which is captured by the semantics, but also interested in *how to prove* a formula holds – a useful approach is to derive new formulas using formulas known to hold following syntactic rules in a *proof system*. We present a Hilbert-style proof system for BI in fig. 2.4. This calculus extends a system for propositional logic with additional rules governing the multiplicative connectives $*$ and $\mathrel{-\!\!*}$ and the multiplicative unit I. These rules say that the multiplicative conjunction $*$ is commutative, associative, the multiplicative unit I interacts with $*$ as expected,

and $-\!\!*$ is adjoint to $*$ just as the regular $\rightarrow$ is adjoint to $\wedge$.

A useful proof system for a logic should be *sound* with respect to its semantics. That is, if a formula ϕ is derivable from another formula φ using the rules in the proof system, then ψ should always hold when ϕ holds. On top of that, it is nicer if the proof system is also *complete* with respect to its semantics. That requires, if ψ always holds when ϕ holds, then ψ is derivable from ϕ as well.

2.2.3 Soundness and Completeness of BI

A methodology for proving the soundness and completeness of bunched logic is given by Docherty [2019], inspired by the duality-theoretic approach to modal logic Goldblatt [1989]. This proof introduces an algebraic semantics of BI by interpreting the rules of BI proof system as algebraic axioms. First, BI is proved sound and complete with respect to the algebraic semantics. Next, the algebraic soundness is used to establish soundness of the proof system with respect to the Kripke semantics, and similarly, the algebraic completeness is used to establish overall completeness.

Notably, a more straightforward proof for the soundness of the proof system is by induction on the proof rules; here, we instead present the duality-theoretic approach for proving soundness to illustrate the technique.

Algebraic Soundness and Completeness of BI Proof System

The algebraic semantics interpret BI formulas into elements in a structure that we call *BI algebra*.

Definition 2.2.4 (BI Algebra)**.** A *BI algebra* is an algebra $\mathbb{A} = (A, \wedge_{\mathbb{A}}, \vee_{\mathbb{A}}, \to_{\mathbb{A}}, \top_{\mathbb{A}}, \bot_{\mathbb{A}}, *_{\mathbb{A}}, -\!\!*_{\mathbb{A}}, I_{\mathbb{A}})$ such that, for all $a, b, c, d \in A$:

- $(A, \wedge_{\mathbb{A}}, \vee_{\mathbb{A}}, \to_{\mathbb{A}}, \top_{\mathbb{A}}, \bot_{\mathbb{A}})$ is a Heyting algebra, i.e., $(A, \wedge_{\mathbb{A}}, \vee_{\mathbb{A}}, \top_{\mathbb{A}}, \bot_{\mathbb{A}})$ forms a bounded lattice (with join and meet operations written $\vee_{\mathbb{A}}$ and $\wedge_{\mathbb{A}}$ and with least element $\bot_{\mathbb{A}}$ and greatest element $\top_{\mathbb{A}}$) and $\to_{\mathbb{A}}$ is a binary operation such that $a \wedge_{\mathbb{A}} b \leq_{\mathbb{A}} c$ is equivalent to $a \leq b \to_{\mathbb{A}} c$.
- $(A, *_{\mathbb{A}}, I_{\mathbb{A}})$ is a commutative monoid;
- $a *_{\mathbb{A}} b \leq c$ iff $a \leq b -\!\!*_{\mathbb{A}} c$, where $\leq$ is the ordering associated with the Heyting algebra.

In the following, we drop the subscripts $_{\mathbb{A}}$ when it is clear that we are referring to elements and operations in the BI algebra and overload the notations $\top, \bot, *, -\!\!*, I$, which are also used as connectives in BI formulas. By Goldblatt [1989], the residuation property $a *_{\mathbb{A}} b \leq c$ iff $a \leq b -\!\!*_{\mathbb{A}} c$ implies the following useful properties.

Lemma 2.2.2. *Given any BI algebra $\mathbb{A}$, for any $a, b, c \in A$, the following properties hold:*

$$(a \vee b) * c = (a * c) \vee (b * c) \qquad \text{(BI-Alg:Dist-1)}$$

$$a * (b \vee c) = (a * b) \vee (a * c) \qquad \text{(BI-Alg:Dist-2)}$$

$$a \leq a' \text{ and } b \leq b' \text{ implies } a * b \leq a' * b' \qquad \text{(BI-Alg:Coh)}$$

$$\bot * a = \bot = a * \bot \qquad \text{(BI-Alg:Bot)}$$

We can interpret bunched logic formulas in a BI algebra $\mathbb{A}$. Given an assignment $\mathcal{V}$ from atomic propositions to the carrier set of $\mathbb{A}$, we can extend it to an

algebraic interpretation of bunched logic formulas $[\![-]\!]_{\mathbb{A}} : \mathrm{Form}_{\mathrm{BI}} \to A$ by taking the unique homomorphic extension of this assignment:

$$[\![p]\!]_{\mathbb{A}} = \mathcal{V}(p)$$

$$[\![\top]\!]_{\mathbb{A}} = \top$$

$$[\![I]\!]_{\mathbb{A}} = I^*$$

$$[\![\bot]\!]_{\mathbb{A}} = \bot$$

$$[\![P \wedge Q]\!]_{\mathbb{A}} = [\![P]\!]_{\mathbb{A}} \wedge [\![Q]\!]_{\mathbb{A}}$$

$$[\![P \vee Q]\!]_{\mathbb{A}} = [\![P]\!]_{\mathbb{A}} \vee [\![Q]\!]_{\mathbb{A}}$$

$$[\![P \to Q]\!]_{\mathbb{A}} = [\![P]\!]_{\mathbb{A}} \to [\![Q]\!]_{\mathbb{A}}$$

$$[\![P * Q]\!]_{\mathbb{A}} = [\![P]\!]_{\mathbb{A}} * [\![Q]\!]_{\mathbb{A}}$$

$$[\![P \mathrel{-\!\!*} Q]\!]_{\mathbb{A}} = [\![P]\!]_{\mathbb{A}} \mathrel{-\!\!*} [\![Q]\!]_{\mathbb{A}}$$

Theorem 2.2.3 (Algebraic Soundness)**.** *If $P \vdash Q$ is derivable, then $[\![P]\!]_{\mathbb{A}} \leq [\![Q]\!]_{\mathbb{A}}$ for all algebraic interpretations $[\![-]\!]_{\mathbb{A}}$.*

Proof. By induction on the derivation of $P \vdash Q$. For instance, for the case of $*$-Conj: if $P \vdash R$ and $Q \vdash S$, then by inductive hypothesis, we have $[\![P]\!]_{\mathbb{A}} \leq [\![R]\!]_{\mathbb{A}}$ and $[\![Q]\!]_{\mathbb{A}} \leq [\![S]\!]_{\mathbb{A}}$ for all algebraic interpretations $[\![-]\!]_{\mathbb{A}}$. By BI-Alg:Coh, that means $[\![P]\!]_{\mathbb{A}} * [\![Q]\!]_{\mathbb{A}} \leq [\![R]\!]_{\mathbb{A}} * [\![S]\!]_{\mathbb{A}}$; therefore, for any algebraic interpretation $[\![-]\!]_{\mathbb{A}}$,

$$[\![P * Q]\!]_{\mathbb{A}} = [\![P]\!]_{\mathbb{A}} * [\![Q]\!]_{\mathbb{A}} \leq [\![R]\!]_{\mathbb{A}} * [\![S]\!]_{\mathbb{A}} = [\![R * S]\!]_{\mathbb{A}}$$

□

To prove algebraic completeness, we construct a term BI algebra by quotienting formulas by equiderivability.

Definition 2.2.5 (Lindenbaum-Tarski Algebra)**.** The Lindenbaum-Tarski algebra corresponding to the bunched logic is the set of all equivalence classes of interprovable propositions. That is, define the equivalence relation $P \sim Q$ as $P \vdash Q$ and $Q \vdash P$. Take $I_{\mathbb{L}}$, $\top_{\mathbb{L}}$, and $\bot_{\mathbb{L}}$ to be $[I]_{\sim}$, $[\top]_{\sim}$, and $[\bot]_{\sim}$, respectively. Then we define:

$$[P]_{\sim} \wedge_{\mathbb{L}} [Q]_{\sim} = [P \wedge Q]_{\sim} \qquad \text{(Lindenbaum–Tarski–And)}$$

$$[P]_{\sim} \vee_{\mathbb{L}} [Q]_{\sim} = [P \vee Q]_{\sim} \qquad \text{(Lindenbaum–Tarski-Or)}$$

$$[P]_{\sim} \to_{\mathbb{L}} [Q]_{\sim} = [P \to Q]_{\sim} \qquad \text{(Lindenbaum–Tarski–Imp)}$$

$$[P]_{\sim} *_{\mathbb{L}} [Q]_{\sim} = [P * Q]_{\sim} \qquad \text{(Lindenbaum–Tarski–SepAnd)}$$

$$[P]_{\sim} \mathrel{-\!\!*}_{\mathbb{L}} [Q]_{\sim} = [P \mathrel{-\!\!*} Q]_{\sim} \qquad \text{(Lindenbaum–Tarski–SepImp)}$$

Lemma 2.2.4. *The operations $\wedge_{\mathbb{L}}, \vee_{\mathbb{L}}, \to_{\mathbb{L}}, *_{\mathbb{L}}, \mathrel{-\!\!*}_{\mathbb{L}}$ are well-defined. Also, the structure $(\{[P]_{\sim}\}_{P \in \mathrm{Form}_{\mathrm{BI}}}, \wedge_{\mathbb{L}}, \vee_{\mathbb{L}}, \to_{\mathbb{L}}, \top_{\mathbb{L}}, \bot_{\mathbb{L}}, *_{\mathbb{L}}, \mathrel{-\!\!*}_{\mathbb{L}}, I_{\mathbb{L}})$ in Lindenbaum-Tarski algebra forms a BI algebra.*

Furthermore, let $[\![-]\!]_{\mathbb{L}}$ be the algebraic interpretation obtained by extending the assignment $p \mapsto [p]_{\sim}$ for each atomic proposition p.

Lemma 2.2.5. *For any formula $P \in \mathrm{Form}_{\mathrm{BI}}$, $[\![P]\!]_{\mathbb{L}} = [P]_{\sim}$.*

The proof for lemma 2.2.4 and lemma 2.2.5 are straightforward, and we omit them here. The Lindenbaum-Tarski algebra is crucially used in the proof of algebraic completeness.

Theorem 2.2.6 (Algebraic Completeness)**.** *If $[\![P]\!]_{\mathbb{A}} \leq [\![Q]\!]_{\mathbb{A}}$ for all algebraic interpretations $[\![-]\!]_{\mathbb{A}}$, then $P \vdash Q$ is derivable.*

Proof. For any $P, Q \in \mathrm{Form}_{\mathrm{BI}}$, if $[\![P]\!]_{\mathbb{A}} \leq [\![Q]\!]_{\mathbb{A}}$ for all algebraic interpretations, then $[\![P]\!]_{\mathbb{L}} \leq [\![Q]\!]_{\mathbb{L}}$ in the Lindenbaum-Tarski algebra. By lemma 2.2.5, that

means $[P]_\sim \leq [Q]_\sim$. In addition,

$$
\begin{aligned}
&[P]_\sim \leq [Q]_\sim \\
\Leftrightarrow &[\top]_\sim \wedge_{\mathbb{A}} [P]_\sim \leq [Q]_\sim \\
&(P \dashv\vdash \top \wedge P \text{ by } \textsc{Top}, \wedge\text{-E}, \wedge\text{-I-R}, \text{and also by definition Lindenbaum–Tarski–And}) \\
\Leftrightarrow &[\top]_\sim =_{\mathbb{A}} [P]_\sim \to_{\mathbb{A}} [Q]_\sim \quad \text{(By the residuation property of the bounded lattice)} \\
\Leftrightarrow &[\top]_\sim =_{\mathbb{A}} [P \to Q]_\sim \quad \text{(By definition Lindenbaum–Tarski–Imp)} \\
\Leftrightarrow &\top \vdash P \to Q \quad (\textsc{Top} \text{ gives the other direction } P \to Q \vdash \top \text{ of equiderivability}) \\
\Leftrightarrow &P \vdash Q \quad \text{(By } \wedge\text{-I-R}, \wedge\text{-I-L}, \wedge\text{-I}, \to\text{-E)}
\end{aligned}
$$

Thus, if $[\![P]\!]_{\mathbb{A}} \leq_{\mathbb{A}} [\![Q]\!]_{\mathbb{A}}$ for all BI algebras, then $P \vdash Q$. □

Soundness of BI Proof Systems

Next, we establish the soundness and completeness of BI algebra with respect to BI Kripke semantics. To show soundness, we first give a recipe to construct a BI algebra given a BI frame; in particular, the BI algebra's carrier set consists of upwards-closed subsets of states in the BI frame — we can think of these subsets as states satisfied by specific formulas. This construction will help to prove that: if there exists a BI model $(\mathcal{X}, \mathcal{V})$ in which $P \not\models_{(\mathcal{X},\mathcal{V})} Q$, then there exists a BI algebra and an algebraic interpretation $[\![-]\!]_{\mathbb{A}}$, such that $[\![P]\!]_{\mathbb{A}} \not\leq [\![Q]\!]_{\mathbb{A}}$. The construction is called the *complex algebra* of a BI frame.

Definition 2.2.6 (Complex Algebra)**.** If $\mathcal{X}$ is a BI frame, then the *complex algebra*

of $\mathcal{X}$, written $\mathsf{Com}(\mathcal{X})$ is the structure $(\mathcal{P}_{\sqsubseteq}(X), \cap, \cup, \to_{\mathcal{X}}, X, \emptyset, *, \mathrel{-\!\!*}, E)$ where

$$\mathcal{P}_{\sqsubseteq}(X) = \{A \subseteq X \mid a \in A \wedge a \sqsubseteq b \to b \in A\}$$

$$A \to_{\mathcal{X}} B = \{a \mid \forall b.\, a \sqsubseteq b \wedge b \in A \to b \in B\}$$

$$A * B = \{x \mid \exists w, y, z.\, w \sqsubseteq x \wedge w \in y \circ z \wedge y \in A \wedge z \in B\}$$

$$A \mathrel{-\!\!*} B = \{x \mid \forall w, y, z.\, (x \sqsubseteq w \wedge z \in w \circ y \wedge y \in A) \to z \in B\}$$

The complex algebra of any BI frame forms a BI algebra.

Lemma 2.2.7. *If $\mathcal{X} = (X, \sqsubseteq, \circ, E)$ is a BI frame, then $\mathsf{Com}(\mathcal{X})$ is a BI algebra.*

Proof. Given $\mathcal{X} = (X, \sqsubseteq, \circ, E)$. Let us show that for any $A \in \mathcal{P}_{\sqsubseteq}(X)$, $A * E = E * A = A$ and omit the rest of the conditions. For the first part,

$$\begin{aligned} A * E &= \{x \mid \exists w, y, z.\, w \sqsubseteq x \wedge w \in y \circ z \wedge y \in A \wedge z \in E\} \\ &= \{x \mid \exists w, y, z.\, w \sqsubseteq x \wedge w \in z \circ y \wedge z \in E \wedge y \in A\} \quad \text{(By Commutativity)} \\ &= E * A \end{aligned} \tag{2.1}$$

For the second part,

$$\begin{aligned} E * A &= \{x \mid \exists w, y, z.\, w \sqsubseteq x \wedge w \in y \circ z \wedge y \in E \wedge z \in A\} \\ &\supseteq \{x \mid \exists z, e_z.\, z \sqsubseteq x \wedge z \in e_z \circ z \wedge e_z \in E \wedge z \in A\} \end{aligned}$$

By Unit Existence, for any $z \in X$, there exists $e_z \in E$ such that $z \in e_y \circ z$. Thus,

$$E * A \supseteq \{x \mid \exists z.z \sqsubseteq x \wedge z \in A\} = A$$

On the other hand, by Unit Coherence and Commutativity, $w \in y \circ z \wedge y \in E$ implies that $z \sqsubseteq w$, and thus,

$$\begin{aligned} E * A &\subseteq \{x \mid \exists w, z.\, w \sqsubseteq x \wedge z \sqsubseteq w \wedge z \in A\} \\ &= \{x \mid \exists z.\, z \sqsubseteq x \wedge z \in A\} \\ &= A \end{aligned}$$

Therefore, $A * E = E * A = A$. □

The complex algebra is constructed in a way that allows us to regard any persistent valuation on the BI frame as an algebraic interpretation of the complex algebra.

Theorem 2.2.8. *Let $\mathcal{X} = (X, \sqsubseteq, \circ, E)$ be a BI frame and let $\mathcal{V}_f : \mathcal{AP} \to \mathcal{P}(X)$ be a persistent valuation on $\mathcal{X}$. Define the algebraic assignment $\mathcal{V}_a : \mathcal{AP} \to \mathsf{Com}(X)$ by letting $\mathcal{V}_a(p) = \mathcal{V}_f(p)$ for all atomic proposition p. Define the algebraic interpretation $[\![-]\!]_a$ by taking the homomorphic extension of $\mathcal{V}_a$ Then we have: $x \models_{\mathcal{V}_f} P$ if and only if $x \in [\![P]\!]_a$.*

Proof. We proceed by induction on P. We show the base case and one inductive case, and omit the rest of the inductive cases.

- Case $P = p$: We have:

$$x \models_{\mathcal{V}_\mathsf{f}} p \quad \text{iff} \quad x \in \mathcal{V}_\mathsf{f}(p) \quad \text{iff} \quad x \in \mathcal{V}_\mathsf{a}(p) \quad \text{iff} \quad x \in [\![p]\!]_\mathsf{a}$$

- Case $P = Q_1 \wedge Q_2$:

$$\begin{aligned} x \models_{\mathcal{V}_\mathsf{f}} Q_1 \wedge Q_2 \quad &\text{iff} \quad x \models_{\mathcal{V}_\mathsf{f}} Q_1 \text{ and } x \models_{\mathcal{V}_\mathsf{f}} Q_2 && \text{(By satisfication rule)} \\ &\text{iff} \quad x \in [\![Q_1]\!]_\mathsf{a} \text{ and } x \in [\![Q_2]\!]_\mathsf{a} && \text{(Inductive Hypothesis)} \\ &\text{iff} \quad x \in [\![Q_1]\!]_\mathsf{a} \cap [\![Q_2]\!]_\mathsf{a} \\ &\text{iff} \quad x \in [\![Q_1 \wedge Q_2]\!]_\mathsf{a} \end{aligned}$$

(By the $\wedge$ operation in Complex algebra and the recursive definition of $\mathcal{V}$)

□

This equivalence between persistent valuations and algebraic interpretations to complex algebra bridges the remaining gap between algebraic soundness we proved in theorem 2.2.3 and overall soundness of the proof system with respect to BI models.

Theorem 2.2.9 (Soundness of BI)**.** *If $P \vdash Q$ is derivable, then $P \models Q$.*

Proof. We prove the contra-positive. If $P \not\models Q$, then there exists a BI model $(\mathcal{X}, \mathcal{V})$ and a state $x \in X$ such that $x \models P$ but $x \not\models Q$. By theorem 2.2.8, if we define $\mathcal{V}_{\mathsf{a}} : \mathcal{AP} \to \mathsf{Com}(\mathcal{X})$ by $\mathcal{V}_{\mathsf{a}}(p) = \mathcal{V}(p)$, then we can extend it into an algebraic interpretation $\mathcal{V}_{\mathsf{a}}$ such that $x \models_{\mathcal{V}} P$ if and only if $x \in [\![P]\!]_{\mathsf{a}}$. Thus, there exists algebraic interpretation $[\![-]\!]_a$ of $\mathsf{Com}(\mathcal{X})$ such that $x \in [\![P]\!]_{\mathsf{a}}$ and $x \notin [\![Q]\!]_{\mathsf{a}}$. So $[\![P]\!]_{\mathsf{a}} \not\subseteq [\![Q]\!]_{\mathsf{a}}$; since the order $\leq_{\mathcal{V}_a}$ in the algebra is exactly the set inclusion $\subseteq$, we have $[\![P]\!]_{\mathsf{a}} \not\leq [\![Q]\!]_{\mathsf{a}}$. By algebraic soundness, that implies $P \vdash Q$ is not derivable. □

Completeness of BI Proof Systems

In the following, we show the completeness of the BI proof system. Dual to the approach for proving soundness, we show that if there exists an algebraic interpretation $[\![-]\!]_{\mathbb{A}}$ and some formulas P, Q such that $[\![P]\!]_{\mathbb{A}} \not\leq [\![Q]\!]_{\mathbb{A}}$, then there exists a BI model $(\mathcal{X}, \mathcal{V})$ such that $P \not\models_{(\mathcal{X},\mathcal{V})} Q$. To show that, we utilize a map dual to the complex algebra construction in the soundness proof: here, given an algebraic interpretation of BI formulas to that a BI algebra, we construct a BI frame corresponding to the BI algebra and a valuation to that BI frame corresponding to the algebraic interpretation.

We first recall a structure on a bounded distributive lattice, called *prime filter*.

Definition 2.2.7 (Prime Filter)**.** If $(L, \wedge, \vee)$ is a bounded distributive lattice, a *filter* F on L is a non-empty subset of A such that:

- If $x \in F$ and $x \leq y$ then $y \in F$.
- If $x \in F$ and $y \in F$ then $x \wedge y \in F$.

A filter is *proper* if it is a proper subset of A, i.e.,, it does not contain $\bot$. A *prime filter* is a proper filter that in addition satisfies: if $x \vee y \in F$ then $x \in F$ or $y \in F$.

Given a BI algebra, we can construct a BI frame whose states are prime filters. We write $\mathsf{Prf}(L)$ for the set of prime filters on L.

Definition 2.2.8 (Prime Filter Frame)**.** If $\mathbb{A} = (A, \wedge, \vee, \rightarrow, \top, \bot, *, \mathrel{-\!\!*}, I)$ is a BI algebra, then the prime filter frame of $\mathcal{A}$ is defined as $\mathsf{Prf}(\mathcal{A}) = (\mathsf{Prf}(A), \subseteq, \circ, E)$ where

$$F_1 \circ F_2 = \{F \in \mathsf{Prf}(A) \mid \forall a_1 \in F_1. \forall a_2 \in F_2. a_1 * a_2 \in F\}$$

$$E = \{F \in \mathsf{Prf}(A) \mid I \in F\}$$

We need to check that the constructed structure is a BI frame.

Lemma 2.2.10. *If $\mathbb{A} = (A, \wedge, \vee, \rightarrow, \top, \bot, *, \mathrel{-\!\!*}, I)$ is a BI algebra, then $\mathsf{Prf}(\mathcal{A})$ is a BI frame.*

Proof. Let us show Unit Coherence and Down-Closed.

For Unit Coherence, if $e \in E$, then for any $x \in \mathsf{Prf}(A)$,

$$\begin{aligned} x \circ e &= \{F \in \mathsf{Prf}(A) \mid \forall a_1 \in x. \forall a_2 \in e. a_1 * a_2 \in F\} \\ &\supseteq \{F \in \mathsf{Prf}(A) \mid \forall a_1 \in x. a_1 * I \in F\} \\ &= \{F \in \mathsf{Prf}(A) \mid \forall a_1 \in x. a_1 \in F\} \\ &= \{F \in \mathsf{Prf}(A) \mid x \subseteq F\} \end{aligned}$$

Thus, $y \in x \circ e$ implies $x \sqsubseteq y$.

For Down-Closed, for any $x, x', y, y', z \in \mathsf{Prf}(A)$, if $z \in x \circ y$ and $x' \subseteq x$ and $y' \subseteq y$ then for any $a_1 \in x', a_2 \in y'$, it must $a_1 \in x, a_2 \in y$ as well, and we have $a_1 * a_2 \in z$. Thus, $z \in x' \circ y'$. □

Below, we show that any algebraic interpretation to $\mathbb{A}$ corresponds to a "morally equivalent" persistent valuation on the prime filter frame $\mathsf{Prf}(\mathbb{A})$. This result is in dual to theorem 2.2.8 used in the soundness proof. In the theorem and its proof, we use the following notation: for any element a in a lattice, we write

$$[a) := \{x \mid a \leq x\}.$$

By construction, any such $[a)$ is upwards-closed and closed under meet, thus a filter.

Theorem 2.2.11. *Let $\mathcal{A} = (A, \dots)$ be a BI algebra and let $[\![-]\!] : \mathrm{Form}_{\mathrm{BI}} \to A$ be an algebraic interpretation that homomorphically extends the assignment $\mathcal{V}_a : \mathcal{AP} \to A$. Define the persistent valuation $\mathcal{V}_f : \mathcal{AP} \to \mathcal{P}(\mathsf{Prf}(A))$ on the prime filter frame $\mathsf{Prf}(\mathcal{A})$ by:*

$$\mathcal{V}_f(p) = \{F \in \mathsf{Prf}(A) \mid \mathcal{V}_a(p) \in F\}$$

Then for $F \in \mathsf{Prf}(A)$, we have $F \models_{\mathcal{V}_f} P$ if and only if $[\![P]\!] \in F$.

Proof. We proceed by induction on the formula P.

- Case $P = p$: For any $F \in \mathsf{Prf}(A)$ and atomic proposition p, we have

$$\begin{aligned} F \models_{\mathcal{V}_\mathsf{f}} p \quad &\text{iff} \quad F \in \mathcal{V}_\mathsf{f}(p) && \text{(By Kripke semantics fig. 2.3)} \\ &\text{iff} \quad F \in \mathsf{Prf}(\mathcal{A}) \text{ and } \mathcal{V}_\mathsf{a}(p) \in F && \text{(By definition of } \mathcal{V}_\mathsf{f}) \\ &\text{iff} \quad \mathcal{V}_\mathsf{a}(p) \in F && \text{(By assumption } F \in \mathsf{Prf}(A)) \\ &\text{iff} \quad [\![p]\!] \in F. && (\,[\![-]\!] \text{ extends } \mathcal{V}_\mathsf{a}(-)) \end{aligned}$$

- Case $P = Q_1 * Q_2$. For any $F \in \mathsf{Prf}(A)$ and formula Q_1, Q_2,

$$\begin{aligned} & F \models_{\mathcal{V}_\mathsf{f}} Q_1 * Q_2 \\ \text{iff} \quad & \text{there exists } F', F_y, F_z \text{ s.t. } F \sqsupseteq F' \in F_y \circ F_z, F_y \models_{\mathcal{V}_\mathsf{f}} Q_1 \text{ and } F_z \models_{\mathcal{V}_\mathsf{f}} Q_2 \\ & \qquad\qquad \text{(By Kripke semantics fig. 2.3)} \\ \text{iff} \quad & \text{there exists } F', F_y, F_z \text{ s.t. } F \supseteq F' \in F_y \circ F_z, [\![Q_1]\!] \in F_y \text{ and } [\![Q_2]\!] \in F_z \\ & \qquad\qquad \text{(By inductive hypothesis and definition of the preorder in } \mathsf{Prf}(\mathbb{A})) \end{aligned}$$

  For the forward direction, by definition of prime filter frames, $[\![Q_1]\!] \in F_y$ and $[\![Q_2]\!] \in F_z$ imply that for any $F' \in F_y \circ F_z$, it must $[\![Q_1]\!] * [\![Q_2]\!] \in F'$. Thus, it implies that $[\![Q_1 * Q_2]\!] \in F$.

  For the other direction, if $[\![Q_1 * Q_2]\!] \in F$, then $[\![Q_1]\!] * [\![Q_2]\!] \in F$. We do a case analysis:

  - Suppose Q_i is $\bot$, then $[\![Q_i]\!] = \bot_\mathbb{A}$. By BI-Alg:Bot, $[\![Q_1]\!] * [\![Q_2]\!] = \bot_\mathbb{A}$. And then $F \ni \bot_\mathbb{A}$, contradicting with F being a proper set. Thus, this case is impossible.
  - Q_1 and Q_2 are both not $\bot$. This means that $[[\![Q_1]\!]_\mathbb{A})$, $[[\![Q_2]\!]_\mathbb{A})$ are both proper filters. On the high level, we first show that $F \in [[\![Q_1]\!]_\mathbb{A}) \circ [[\![Q_2]\!]_\mathbb{A})$, and then use that to show that there exist prime filters F_y, F_z

such that $F \in F_y \circ F_z$ and $[\![Q_1]\!]_{\mathbb{A}} \in F_1$ and $[\![Q_2]\!]_{\mathbb{A}} \in F_2$, which would then be used to show $F \models_{\mathcal{V}_f} Q_1 * Q_2$. First,

$$
\begin{aligned}
&[[\![Q_1]\!]_{\mathbb{A}}) \circ [[\![Q_2]\!]_{\mathbb{A}}) \\
&= \{F \in \mathsf{Prf}(A) \mid \forall a \in [[\![Q_1]\!]_{\mathbb{A}}). \forall b \in [[\![Q_2]\!]_{\mathbb{A}}).\ a * b \in F\} \\
&= \{F \in \mathsf{Prf}(A) \mid \forall a, b.\ [\![Q_1]\!]_{\mathbb{A}} \leq a \wedge [\![Q_2]\!]_{\mathbb{A}} \leq b \Rightarrow a * b \in F\}
\end{aligned}
$$

By BI-Alg:Coh, $[\![Q_1]\!]_{\mathbb{A}} \leq a$ and $[\![Q_2]\!]_{\mathbb{A}} \leq b$ implies

$$[\![Q_1]\!] * [\![Q_2]\!] \leq a * b.$$

Thus, our given F being a filter and $[\![Q_1]\!] * [\![Q_2]\!] \in F$ imply that $a * b \in F$ for any such a, b. Therefore, $F \in [[\![Q_1]\!]_{\mathbb{A}}) \circ [[\![Q_2]\!]_{\mathbb{A}})$.

Next, define a predicate P such that $P(F_1, F_2) = 1$ if and only if $F \in F_1 \circ F_2$ and $[\![Q_1]\!]_{\mathbb{A}} \in F_1$ and $[\![Q_2]\!]_{\mathbb{A}} \in F_2$. Because $F \in [[\![Q_1]\!]_{\mathbb{A}}) \circ [[\![Q_2]\!]_{\mathbb{A}})$, we have $P([[\![Q_1]\!]_{\mathbb{A}}), [[\![Q_2]\!]_{\mathbb{A}})) = 1$. This predicate P is a prime predicate according to Docherty [2019] (cf. Definition 5.5) — the proof follows from unfolding definitions and we omit it. Then, applying the Prime Extension Lemma (cf. Lemma 5.7 of Docherty [2019]), the existence of proper filters F_1, F_2 such that $P(F_1, F_2) = 1$ implies that there exists prime filters F_y, F_z such that $P(F_y, F_z) = 1$. Therefore. there exists F_y, F_z such that $F \in F_y \circ F_z$ and $[\![Q_1]\!]_{\mathbb{A}} \in F_y$ and $[\![Q_2]\!]_{\mathbb{A}} \in F_z$ — and by inductive hypothesis this means $F_y \models_{\mathcal{V}_f} Q_1$ and $F_z \models_{\mathcal{V}_f} Q_2$. The existence of such F_y and F_z validates that $F \models_{\mathcal{V}_f} Q_1 * Q_2$.

□

Now we are ready to prove completeness.

Theorem 2.2.12 (BI Completeness). *If $P \models_{\mathcal{V}} Q$ for all BI models $(\mathcal{X}, \mathcal{V})$, then $P \vdash Q$.*

Proof. We prove the contra-positive. Assume $P \vdash Q$ is not derivable. By algebraic completeness, there exists algebra $\mathbb{A}$ and interpretation $[\![-]\!]$ such that $[\![P]\!] \not\leq [\![Q]\!]$. Then, the element $[\![Q]\!]$ is not in $[[\![P]\!])$, the least filter containing P. Let $F = [[\![P]\!])$.

- P is $\bot$. Then $\bot \vdash Q$ by the proof rule BOT, contradicting with $P \nvdash Q$.
- P is not $\bot$. Then $F = [[\![P]\!])$ is a proper filter. Define a predicate P such that $P(F') = 1$ iff $[\![P]\!] \in F'$ and $[\![Q]\!] \notin F'$. Because $[\![Q]\!] \notin [[\![P]\!])$ and $[\![P]\!] \in [[\![P]\!])$, we have $P(F) = 1$. This predicate is a prime predicate, and from prime extension lemma (cf. Lemma 5.7 of Docherty [2019]) it can be established that there is a prime filter F' on $\mathbb{A}$ such that $[\![P]\!] \in F'$ and $[\![Q]\!] \notin F'$.

  Define a persistent valuation $\mathcal{V}_{\mathsf{f}}$ on $\mathsf{Prf}(\mathbb{A})$ by

  $$\mathcal{V}_{\mathsf{f}}(p) = \{F \in \mathsf{Prf}(A) \mid \mathcal{V}_{\mathsf{a}}(p) \in F\}.$$

  By theorem 2.2.11, we have $F_{\mathcal{V}_{\mathsf{f}}} \models P$ and $F_{\mathcal{V}_{\mathsf{f}}} \not\models Q$. Thus, $P \not\models Q$.

□

2.2.4 A Discrete Probabilistic Frame of BI

After displaying the metatheory of bunched logic, next we show a concrete example of BI model. We present a model based on *probabilistic distributions* over *program memories*, which will be useful later in reasoning about probabilistic programs.

Definition 2.2.9 (Discrete Distribution)**.** Given a set S, a discrete subdistribution μ is a countable support function $\mu\colon X \to [0,1]$ satisfying $\sum_{x\in X} \mu(x) \leq 1$. A (full) distribution is a subdistribution that in addition $\sum_{x\in X} \mu(x) = 1$. We use $\mathbb{D}(X)$ to denote the set of discrete (full) distributions μ over X.

Now we can define program memories. Throughout this thesis, we fix a set of variables **Var** and a set of values that the variables can take **Val**.

Definition 2.2.10 (Program Memories)**.** Let $S \subseteq \mathbf{Var}$ be a set of variable names. We call any function $m\colon S \to \mathbf{Val}$ a *program memory* because such a map m assigns a value to each variable in S. Let $\mathbf{Mem}[S]$ denote the set of program memories from S to **Val**; and for each $m \in \mathbf{Mem}[S]$, define the domain of m to be S and denote it as $\mathbf{dom}(S)$

As an example, the empty program memory $\mathbf{Mem}[\emptyset] = \emptyset \to \mathbf{Val}$ contains exactly one element, which is a trivial map with an empty domain; we denote the trivial map by $\langle\rangle$.

We need two operations on memories. First, a memory m with domain S can be projected to a memory $\pi_T m$ with domain T if $T \subseteq S$, defined as $\pi_T m(x) = m(x)$ for any variable $x \in T$. Second, two memories can be combined if they agree on the intersection of their domains.

Definition 2.2.11. Given memories $m_1 \in \mathbf{Mem}[S]$, $m_2 \in \mathbf{Mem}[T]$ such that $\pi_{S\cap T} m_1 = \pi_{S\cap T} m_2$, we define $m_1 \bowtie m_2\colon S \cup T \to \mathbf{Val}$ by

$$m_1 \bowtie m_2(x) \;:=\; \begin{cases} m_1(x) & \text{if } x \in S \setminus T \\ m_2(x) & \text{if } x \in T \setminus S \\ m_1(x) = m_2(x) & \text{if } x \in S \cap T \end{cases}$$

This operation is not defined when m_1, m_2 disagree on $S \cap T$. This operation is well-defined exactly because m_1, m_2 agrees on $S \cap T$.

We also lift the projection map to distributions. We define the *projection* π_S to marginalize a distribution μ on $\mathcal{D}(\mathbf{Mem}[S'])$ to a distribution on $\mathcal{D}(\mathbf{Mem}[S \cap S'])$: for any $x \in \mathbf{Mem}[S \cap S']$,

$$\pi_S \mu(x) := \sum_{x' \in \mathbf{Mem}[S' \setminus S]} \mu(x' \bowtie x).$$

This gives us enough ingredients to define a probabilistic BI frame that will later be useful for reasoning about probabilistic independence.

Definition 2.2.12 (A Discrete Probabilistic BI Frame)**.** Define a *discrete probabilistic BI frame* to be a structure $\mathcal{X}_{\mathbb{D}} = (X_{\mathbb{D}}, \sqsubseteq_{\mathbb{D}}, \otimes_{\mathbb{D}}, E_{\mathbb{D}})$ where

- $\mathcal{X}_{\mathbb{D}} := \cup_{S \subseteq \mathbf{Var}} \mathcal{D}(\mathbf{Mem}[S])$;
- Distributions $\mu_1 \sqsubseteq_{\mathbb{D}} \mu_2$ iff $\mu_1 = \pi_{\mathbf{dom}(\mu_2), \mathbf{dom}(\mu_1)} \mu_2$.
- For distributions $\mu_1 \in \mathcal{D}(\mathbf{Mem}[S])$, $\mu_2 \in \mathcal{D}(\mathbf{Mem}[T])$, the binary operation $\otimes_{\mathbb{D}}$ takes the *independent product* of them iff S and T are disjoint:

$$\mu_1 \otimes_{\mathbb{D}} \mu_2 = \begin{cases} \{\mu \mid \forall x \in \mathbf{Mem}[S \cup T], \mu(x) = \mu_1(\pi_S x) \cdot \mu_2(\pi_T x)\} & \text{if } S, T \text{ disjoint} \\ \emptyset & \text{otherwise} \end{cases}$$

- $E_{\mathbb{D}} := \cup_{S \subseteq \mathbf{Var}} \mathcal{D}(\mathbf{Mem}[S])$;

We check that the structure $\mathcal{X}_{\mathbb{D}}$ is a BI frame. In Barthe et al. [2019], they check a very similar structure is a partial commutative monoid. The structure's carrier set consists of pairs of deterministic memories and randomized memories; our states can be viewed as a degenerated case of their states with trivial

deterministic memories. Meanwhile, BI frames can be viewed as a generalization of partial commutative monoids. So it intuitively follows from their result that $\mathcal{X}_{\mathbb{D}}$ is a BI frame.

Theorem 2.2.13. *$\mathcal{X}_{\mathbb{D}} = (X_{\mathbb{D}}, \sqsubseteq_{\mathbb{D}}, \otimes_{\mathbb{D}}, E_{\mathbb{D}})$ is a BI frame.*

Proof. We show that it satisfies all the frame conditions. For instance,

Down-Closed If $\mu_z \in \mu_x \otimes_{\mathbb{D}} \mu_y$, and $\mu'_x \sqsubseteq_{\mathbb{D}} \mu_x$, $\mu'_y \sqsubseteq_{\mathbb{D}} \mu_y$, then define $X = \mathbf{dom}(\mu_x), Y = \mathbf{dom}(\mu_y), X' = \mathbf{dom}(\mu'_x), Y' = \mathbf{dom}(\mu'_y)$, and define $\mu = \pi_{X' \cup Y'} \mu_z$. The fact that $\mu_z \in \mu_x \otimes_{\mathbb{D}} \mu_y$ implies that for any $m \in \mathbf{Mem}[X \cup Y]$,

$$\mu_z(m) = \mu_x(\pi_X m) \cdot \mu_y(\pi_Y m);$$

Thus,

$$\begin{aligned}
&\mu(m) = (\pi_{X' \cup Y'} \mu_z)(m) \\
&= \sum_{m' \in \mathbf{Mem}[X \cup Y \setminus (X' \cup Y')]} \mu_z(m' \bowtie m) \\
&= \sum_{m' \in \mathbf{Mem}[X \cup Y \setminus (X' \cup Y')]} \mu_x(\pi_X m' \bowtie m) \cdot \mu_y(\pi_Y m' \bowtie m) \\
&= \sum_{m_1 \in \mathbf{Mem}[X \setminus X']} \sum_{m_2 \in \mathbf{Mem}[Y \setminus Y']} \mu_x(\pi_X m_1 \bowtie m_2 \bowtie m) \cdot \mu_y(\pi_Y m_1 \bowtie m_2 \bowtie m) \\
&= \left(\sum_{m_1 \in \mathbf{Mem}[X \setminus X']} \mu_x(\pi_X m_1 \bowtie m) \right) \cdot \left(\sum_{m_2 \in \mathbf{Mem}[Y \setminus Y']} \mu_y(\pi_Y m_2 \bowtie m) \right) \\
&= \pi_{X'} \mu_x(m) \cdot \pi_{Y'} \mu_y(m) \\
&= \mu'_x(m) \cdot \mu'_y(m)
\end{aligned}$$

Hence, $\mu \in \mu'_x \otimes_{\mathbb{D}} \mu'_y$, and by definition, $\mu \sqsubseteq_{\mathbb{D}} \mu_z$.

We delay the full proof to appendix A. □

This theorem indicates that, when given a set of atomic propositions and a persistent valuation, we can interpret BI formulas on distributions over memories. In the next section, we will use BI formulas to specify probabilistic programs and also give proof rules for reasoning about probabilistic programs.

2.3 Probabilistic Separation Logic

In this section, we will overview *probabilistic separation logic*, which consists of a set of rules for analyzing probabilistic programs. Each rule describes how a probabilistic program transforms its input distribution into its output distribution; and the input and output distributions are specified using BI formulas interpreted on the concrete probabilistic BI frame $\mathcal{X}_{\mathbb{D}}$ — we will introduce a set of atomic propositions and a valuation so that the BI formulas can effectively specify probabilistic programs.

At the high level, probabilistic separation logic will utilize a useful and common property in probabilistic distributions — independence — to reason about irrelevant parts of a probabilistic programs modularly. Independence is often defined for *events*. For a discrete distribution $\mu : X \to [0, 1]$, an *event* $\mathcal{EV}$ is a map $X \to \{0, 1\}$, and the probability of event $\mathcal{EV}$ is $\sum_{\omega \in X} \mu(\omega) \cdot \mathcal{EV}(\omega)$, which we will overload the notation and write as $\mu(\mathcal{EV})$. The independence of two events says that the occurrence of one event does not tell anything about the occurrence of the other event. Formally,

Definition 2.3.1 (Probabilistic Independence of Events)**.** Given any distribution $\mu : X \to [0, 1]$, two events $\mathcal{EV}_1, \mathcal{EV}_2$ are independent if and only if

$$\mu(\mathcal{EV}_1 \cap \mathcal{EV}_2) = \mu(\mathcal{EV}_1) \cdot \mu(\mathcal{EV}_2).$$

A set of events $\mathcal{EV}_1, \ldots, \mathcal{EV}_n$ are *mutually independent* if for any subset $S \subseteq [n]$,

$$\mu(\bigcap_{j \in S} \mathcal{EV}_j) = \bigcap_{j \in S} \mu(\mathcal{EV}_j).$$

For program analysis, another useful notion is the probabilistic independence between two program variables, which says that knowing the value of one program variable does not tell anything about the value of the other. To formally define it, we need "a variable x takes a value v" to be an event, which is the case for *distributions over program memories*. Given a distribution μ over $\mathcal{D}(\mathbf{Mem}[S])$, then for any $x \in S$, we write the event $\{\omega \in \mathbf{Mem}[S] \mid \omega(x) = v\}$ as $x = v$.

We then define the probabilistic independence of variables as followings.

Definition 2.3.2 (Probabilistic Independence of Program Variables)**.** Given any distribution $\mu : X \to [0, 1]$ and two variables $x, y \in \mathbf{Var}$, if $x = v_1, y = v_2$ are events for any two values $v_1, v_2 \in \mathbf{Val}$, then we define the variables x and y to be independent if and only if: for any $v_1, v_2 \in \mathbf{Val}$,

$$\mu(x = v_1 \wedge y = v_2) = \mu(x = v_1) \cdot \mu(y = v_2),$$

i.e., the events $x = v_1$ and $y = v_2$ are independent. Similarly, a set of program variables $y_1, \ldots, y_n$ are *mutual independent* iff for any subset $S \subseteq [n]$, for any set of $\{v_i \in \mathbf{Val} \mid i \in S\}$

$$\mu(\bigcap_{i \in S} x = v_i) = \prod_{i \in S} \mu(x = v_i),$$

In the following, when talking about a set of variables, we will abbreviate mutual independence as independence.

Another commonly used notion is the independence between two sets of program variables. We can talk about value assignments on a set of variables

in a similar way: given a distribution μ over $\mathbf{Mem}[S]$, for any $X \subseteq S$ and $m \in \mathbf{Mem}[X]$, we write $X = m$ for $\{\omega \in \mathbf{Mem}[S] \mid \forall x \in X.\omega(x) = m(x)\}$.

Definition 2.3.3 (Probabilistic Independence of Two Sets of Program Variables)**.** Given two sets of variables $X, Y \subseteq \mathbf{Var}$, program memories on X and program memories on Y, X, Y are independent if for any $m_X \in \mathbf{Mem}[X]$, $m_Y \in \mathbf{Mem}[Y]$,

$$\mu(X = m_X \cap Y = m_Y) = \mu(X = m_X) \cdot \mu(Y = m_X).$$

An equivalent condition is as follows: for any $m_X \in \mathbf{Mem}[X]$, $m_Y \in \mathbf{Mem}[Y]$ such that $m_X \bowtie m_Y$ is defined,

$$\pi_{X \cup Y}\mu(m_X \bowtie m_Y) = \pi_X\mu(m_X) \cdot \pi_Y\mu(m_Y).$$

The probabilistic separation logic presented in this section will facilitate its users to prove, track, and utilize the independence of (sets of) program variables.

2.3.1 A Simple Probabilistic Programming Language

We work with an imperative language pWhile that allows sampling from a set of built-in primitive distributions. We first define the set of valid expressions $\mathcal{E}$ and the set of allowed distributions $\mathcal{D}$, and then define the formal grammar of commands C. We assume a fixed set of typed program variables; x stands for a numeric variable, while b stands for a boolean variable. The expression language is standard. Distribution terms $d \in \mathcal{D}$ can be $\mathbf{Bern}_v$ for a Bernoulli (coin-flip) distribution with bias v, $\mathbf{Unif}_S$ for a uniform distribution over elements in S, or some other symbols interpreted into distributions — we will introduce them

$$\mathcal{E} \ni e ::= v \in \mathbf{Val} \mid x, b \in \mathbf{Var} \mid e_1 = e_2 \mid e_1 + e_2 \mid e_1 \times e_2 \mid \ldots$$
$$\mathcal{D} \ni d ::= \mathbf{Bern}_v \mid \mathbf{Unif}_S \mid \ldots$$
$$C \ni c ::= \mathbf{skip} \mid x \leftarrow e \mid x \xleftarrow{\$} d \mid \mathbf{if}\ b\ \mathbf{then}\ c\ \mathbf{else}\ c' \mid c\ ;\ c' \mid \mathbf{while}\ b\ \mathbf{do}\ c$$

Figure 2.5: pWhile command syntax

later when needed. We assume throughout that all expressions and distribution terms are well-typed; in particular, the value v in $\mathbf{Bern}_v$ is a number in the interval $[0, 1]$, and S in $\mathbf{Unif}_S$ is a finite set whose size is $|S|$.

For commands, pWhile has six kinds of programs: the no-op **skip**; assignments $x \leftarrow e$, which assign the evaluated value of the expression e to the program variable x; sampling $x \xleftarrow{\$} d$ for drawing a value from a distribution d and assigning it to x; conditionals **if** b **then** c **else** c' for branching on a (possibly randomized) condition b; sequencing $c\ ;\ c'$; and loops **while** b **do** c for iterating a command c until the condition b is not true. We also write **if** b **then** c as abbreviation of **if** b **then** c **else skip**.

Probabilistic Monad To concisely describe the denotational semantics of these commands, we introduce operations on distributions and *probabilistic monads*.

Since $\mathbb{D}(X)$ is the set of distributions over X, we can view $\mathbb{D}$ as an operation that maps a set into distributions over that set. This operation on sets can be lifted to functions $f\colon X \to Y$, resulting in a map of distributions $\mathbb{D}(f)\colon \mathbb{D}(X) \to \mathbb{D}(Y)$ given by $\mathbb{D}(f)(\mu)(y) := \sum_{f(x)=y} \mu(x)$. Intuitively, $\mathbb{D}(f)$ takes the sum of the probabilities of all elements in the pre-image of y. These operations turn $\mathbb{D}$ into a functor on sets and, further, $\mathbb{D}$ is also a *monad* [Giry, 1982, Moggi, 1991].

Definition 2.3.4 (Distribution Monad)**.** Define $\mathsf{unit}\colon X \to \mathbb{D}(X)$ as $\mathsf{unit}(x) := \delta_x$

where δ_x denotes the *Dirac distribution* on x: for any $y \in X$, we have $\delta_x(y) = 1$ if $y = x$, otherwise $\delta_x(y) = 0$. Further, define $\mathsf{bind} \colon \mathbb{D}(X) \times (X \to \mathbb{D}(Y)) \to \mathbb{D}(Y)$ by $\mathsf{bind}(\mu, f)(y) := \sum_{p \in \mathbb{D}(Y)} \mathbb{D}(f)(\mu)(p) \cdot p(y)$.

Intuitively, unit embeds a set into distributions over the set, and bind enables the sequential combination of probabilistic computations. Both maps are natural transformations and satisfy the following interaction laws, establishing that $(\mathbb{D}, \mathsf{unit}, \mathsf{bind})$ is a monad:

$$\mathsf{bind}(\mathsf{unit}(x), f) = f(x)$$

$$\mathsf{bind}(\mu, x \mapsto \mathsf{unit}(x)) = \mu,$$

$$\mathsf{bind}(\mathsf{bind}(\mu, f), g) = \mathsf{bind}(\mu, \lambda x.\mathsf{bind}(f(x), g)).$$

The distribution monad has an equivalent presentation in which bind is replaced with a multiplication operation $\mathsf{join} : \mathbb{D}(\mathbb{D}(X)) \to \mathbb{D}(X)$, which flattens distributions by averaging:

$$\mathsf{join}(\mu)(x) := \sum_{\rho \sim \mu} \mu(\rho) \cdot \rho(x).$$

Program Semantics Given a program memory containing all variables appearing in an expression, we interpret $\mathcal{E}$ terms as values in **Val** and interpret $\mathcal{D}$ terms as distributions in $\mathbb{D}(\mathbf{Val})$ as in fig. 2.6. We overload the notation and write $[\![e]\!]$ for interpretation of expression e and $[\![d]\!]$ for interpretation of distribution d.

We can also interpret expressions on probabilistic memories through a lifting. For any $\mu \in \mathbb{D}(\mathbf{Mem}[S])$,

$$[\![e]\!](\mu) = \mathsf{bind}(\mu, m \mapsto [\![e]\!](m))$$

$$
\begin{aligned}
[\![v]\!](m) &:= v \\
[\![x]\!](m) &:= m(x) \\
[\![x = y]\!](m) &:= 1 \text{ if } m(x) = m(y) \text{ else } 0 \\
[\![x + y]\!](m) &:= m(x) + m(y) \\
[\![x \times y]\!](m) &:= m(x) \times m(y)
\end{aligned}
$$

$$
[\![\mathbf{Bern}_v]\!] = \begin{cases} 1 \mapsto v \\ 0 \mapsto 1 - v \\ \omega \mapsto 0 \qquad \text{if } \omega \neq 0 \text{ and } \omega \neq 1 \end{cases}
$$

$$
[\![\mathbf{Unif}_S]\!] = \begin{cases} \omega \mapsto \frac{1}{|S|} \quad \text{if } \omega \in S \\ \omega \mapsto 0 \quad \text{otherwise} \end{cases}
$$

Figure 2.6: Semantics of Expressions and Distributions

Then we can interpret programs in pWhile as distribution transformers $\mathcal{D}(\mathbf{Mem}[\mathbf{Var}]) \to \mathcal{D}(\mathbf{Mem}[\mathbf{Var}])$, as in fig. 2.7. The interpretation is standard. The command **skip** simply outputs the input distribution; $x \leftarrow e$ and $x \xleftarrow{\$} d$ use the monadic operation bind to compose the input distribution μ with the updating map describing the output distribution corresponding to each deterministic input memory m; last, $c \,;\, c'$ composes the interpretation of c and c' using usual function composition.

Because the conditional **if** b **then** c **else** c' allows a randomized guard b, interpreting it requires two more operations on distributions: a conditioning operation $\mu \mid S$ to split control flow, and convex combination $\oplus_p$ to merge control flow. Given any distribution $\mu \in \mathbb{D}(A)$ and event $S \subseteq A$, if $\mu(S) > 0$, the conditional distribution of μ given S is:

$$
(\mu \mid S)(a) := \begin{cases} \frac{\mu(a)}{\mu(S)} & : a \in S \\ 0 & : a \notin S. \end{cases} \tag{2.2}
$$

$$
\begin{aligned}
[\![\textbf{skip}]\!](\mu) &:= \mu \\
[\![x \leftarrow e]\!](\mu) &:= \mathsf{bind}(\mu, m \mapsto \mathsf{unit}(m[x \mapsto [\![e]\!](m)])) \\
[\![x \xleftarrow{\$} d]\!](\mu) &:= \mathsf{bind}(\mu, m \mapsto \mathsf{bind}(d, v \mapsto \mathsf{unit}(m[x \mapsto v]))) \\
[\![c\ ;\ c']\!](\mu) &:= [\![c']\!]([\![c]\!](\mu)) \\
[\![\textbf{if } b \textbf{ then } c \textbf{ else } c']\!](\mu) &:= [\![c]\!](\mu \mid b = tt) \oplus_p [\![c']\!](\mu \mid b = ff) \qquad \text{where } p := \mu(b = tt) \\
[\![\textbf{abort}]\!](\mu) &:= \lambda\omega.0 \\
[\![\textbf{while } b \textbf{ do } c]\!](\mu) &:= \lim_{n\to\infty} [\![(\textbf{if } b \textbf{ then } c)^n; \textbf{if } b \textbf{ then abort}]\!](\mu)
\end{aligned}
$$

Figure 2.7: Program semantics

When $\mu(S) = 0$, we leave $\mu \mid S$ undefined. For convex combination, for any $\mu_1, \mu_2 \in \mathbb{D}(A)$, we define $\mu_1 \oplus_0 \mu_2 := \mu_2$ and $\mu_1 \oplus_1 \mu_2 := \mu_1$. When $p \in (0, 1)$, $(\mu_1 \oplus_p \mu_2)(a) := p \cdot \mu_1(a) + (1 - p) \cdot \mu_2(a)$. Conditioning and convex combination are inverses in the sense that $\mu = (\mu \mid S) \oplus_{\mu(S)} (\mu \mid (A \setminus S))$.

The command **abort** is only used in the definition of the while loops and is not accessible by the users. It disregards the input distribution μ and returns a subdistribution that assigns 0 to all possible outcomes ω. The semantics of the while loop **while** b **do** c is the limit of $[\![(\textbf{if } b \textbf{ then } c)^n; \textbf{if } b \textbf{ then abort}]\!]$ as n approaches to infinity. The limit, taken with the point-wise order, exists according to the monotone convergence theorem [Abbott, 2015, Strichartz, 2000] because the subdistribution's mass is non-decreasing as n increases and is upper bound by 1. In practice, we assumed that all loops terminate in finite steps, the limit is always a full distribution, so all commands in pWhile can still be interpreted as distribution transformers.

2.3.2 A Concrete BI Model for Asserting Independence

We define some atomic propositions to describe distributions over program memories $\cup_{S \subseteq \mathbf{Var}} \mathcal{D}(\mathbf{Mem}[S])$. The BI frame $\mathcal{X}_{\mathbb{D}}$ defined in section 2.2.4 together with the valuation $\mathcal{V}^*$ for atomic propositions defined below provide a BI model, on which we can assert properties such as distributions of variables, and independence between variables.

Let atomic propositions

$$\mathcal{AP}_{\mathbb{D}} \ni p ::= \mathbf{Own}(\mathcal{E}) \mid \mathcal{E} \leftsquigarrow \mu \mid \mathbf{Detm}\langle \mathcal{E} \rangle \mid [\mathcal{E} = \mathcal{E}] \mid \mathbb{E}[\mathcal{E}] \bowtie c \tag{2.3}$$

where $\bowtie \in \{=, \leq, \geq\}$, $b \in \{0, 1\}$, and $c \in \mathbb{R}$ is a constant. Roughly, $\mathsf{Own}(\mathcal{E})$ asserts that the distribution of the expression $\mathcal{E}$ is fully determined; $\mathcal{E} \leftsquigarrow \mu$ asserts that the expression $\mathcal{E}$ has distribution μ; $\mathbf{Detm}\langle \mathcal{E} \rangle$ asserts that the expression $\mathcal{E}$ is deterministic; $[\mathcal{E}_1 = \mathcal{E}_2]$ asserts that the expression $\mathcal{E}_1$ and $\mathcal{E}_2$ are always equal; last, $\mathbb{E}[e] \bowtie c$ bound the expected value of an expression e with respect to a constant c. In particular, since events are maps from memories to $\{0, 1\}$, which are the same type as the interpretation of boolean expressions in the language, we assume the set of expressions contains events as well.

We define the satisfaction of atomic proposition on program configurations as follows. Let $\mathrm{FV}(e)$ be the set of free variables in expression e.

Definition 2.3.5 (Valuation)**.** For $\mu \in \mathcal{X}_{\mathbb{D}}$, define $\mathcal{V}^*$ such that

- $\mu \in \mathcal{V}^*(\mathsf{Own}(e))$ holds if $\mathrm{FV}(e) \subseteq \mathbf{dom}(\mu)$;
- $\mu \in \mathcal{V}^*(e \leftsquigarrow \mu')$ iff $\mathrm{FV}(e) \subseteq \mathbf{dom}(\mu)$ and $[\![e]\!](\mu) = \mu'$;
- $\mu \in \mathcal{V}^*(\mathbf{Detm}\langle e \rangle)$ iff $\mathrm{FV}(e) \subseteq \mathbf{dom}(\mu)$ and $[\![e]\!](\mu)$ is a Dirac distribution;

- $\mu \in \mathcal{V}^*([e \bowtie e'])$ iff $\mathrm{FV}(e) \cup \mathrm{FV}(e') \subseteq \mathbf{dom}(\mu)$ and $[\![e]\!](m) \bowtie [\![e']\!](m)$ for any m in the support of μ;
- $\mu \in \mathcal{V}(\mathbb{E}[e] \bowtie c)$ iff the expected value of expression e in μ, i.e., $\mathbb{E}[e] := \sum_{m \in \mathbf{Mem}[\mathbf{dom}(\mu)]} \mu(m) \cdot [\![e]\!](m)$, satisfies $\mathbb{E}[e] \bowtie c$.

For an event ev, we also write $\Pr[ev] \bowtie c$ for $\mathbb{E}[ev] \bowtie c$.

It is straightforward to show that $\mathcal{V}^*$ defined for these atomic propositions is a persistent valuation.

Proposition 2.3.1. *$(\mathcal{X}_{\mathbb{D}}, \mathcal{V}^*)$ forms a BI model.*

With these atomic propositions, we can now use bunched logic formulas to assert interesting probabilistic properties. For instance, we can assert the independence between two variables using the following assertion.

Lemma 2.3.2. *For any distribution $\mu \in \mathcal{X}_{\mathbb{D}}$, for a set of variables $\{X_i\}_{i \in S}$, $\mu \models *_{i \in S} \mathsf{Own}(X_i)$ iff variables $\{X_i\}_{i \in S}$ are distinct and mutually independent.*

We present the proof in appendix A.

The assertion logic has all the axioms for atomic formulas stated in Barthe et al. [2019, Lemma 3, 4]. While this set of axioms is not complete, they are useful for reasoning about a rich family of probabilistic properties.

Lemma 2.3.3. *The following axiom schemas are valid:*

$$\models [e_1 = e_2] \to [e_2 = e_1] \qquad \text{(Eq-Sym)}$$

$$\models [e_1 = e_2] \wedge [e_2 = e_3] \to [e_1 = e_2] \qquad \text{(Eq-Tran)}$$

$$\models \mathsf{Own}(e_1) \to \mathsf{Own}(e_2) \text{ whenever } FV(e_2) \subseteq FV(e_1) \qquad \text{(Own-Incl)}$$

Note that $\models [e_1 = e_1]$ is *not* an axiom — it is not sound, since it may not hold in a randomized memory $\mathcal{D}(\mathbf{Mem}[\emptyset])$ with empty domain. We also have axioms for uniformity propositions.

Lemma 2.3.4. *The following axiom schemas are valid:*

$$\models [e_1 = e_2] \wedge \mathbf{Unif}_S\langle e_1\rangle \to \mathbf{Unif}_S\langle e_2\rangle \tag{Unif-Tran}$$

$$\models \mathbf{Unif}_S\langle e_1\rangle \to [e_1 = e_1] \tag{Unif-Weak}$$

$$\models \mathbf{Unif}_S\langle e_1\rangle \to \mathbf{Unif}_S\langle f(e_1)\rangle \text{ for any bijection } [\![f]\!] : S \to S \text{ and } FV(f) \subseteq FV(e_1) \tag{Unif-Bij}$$

2.3.3 A Program Logic for Reasoning about Independence

We now introduce the program logic layer of probabilistic separation logic. In the spirit of other separation logic (e.g., [Reynolds, 2002, Brookes, 2007a, Jung et al., 2018]), the logic is designed to prove separations and harness separations to prove other properties more easily; in this case, the separation is probabilistic independence. Similar to standard Hoare logic, it has judgments of the form $\{P\}$ prog $\{Q\}$, where prog is a probabilistic program command in C, and P, Q are BI formulas with atomic propositions in $\mathcal{AP}_{\mathbb{D}}$.

Definition 2.3.6 (Validity)**.** A probabilistic separation logic judgment is *valid*, written $\models \{P\}$ prog $\{Q\}$, if for all $\mu \in \mathcal{D}(\mathbf{Mem}[\mathbf{Var}])$ such that $\mu \models P$, we have $[\![\mathsf{prog}]\!](\mu) \models Q$.

Next, we proceed to the proof system, which consists of program rules for each command and structural rules that match any program command. As before, we use $\mathrm{FV}(e)$ to denote the set of free variables in an expression e.

Program Rules The program rules are presented in fig. 2.8a. The rules RASSN and SAMP are for randomized assignment and random sampling. Both rules are presented with the trivial pre-condition $\top$; in practice, one would want to reason about assignments and sampling starting from general pre-conditions, and we will derive variants of these rules with other pre-conditions using the structural rules.

There are two rules governing conditionals. In COND, the precondition implies that the randomized guard b behaves deterministically, and thus either the guard b is true and c executes or the guard b is false and c' executes. If both branches guarantee ψ as the post-condition, then ψ is also the post-condition for the conditional. The rule RCOND, on the other hand, applies when the randomized guard b is separate from the rest of the pre-condition — that is, it must be probabilistically independent of the portion of the randomized memory captured by φ. This independence is crucial for ensuring φ remains valid as the pre-condition of both branches: each branch's input distribution is obtained by conditioning on the guard's value in the original distribution; notably, that conditioning operation can invalidate φ if the guard b and variables in φ are correlated, even if they share no variables. To illustrate this, recall [Barthe et al., 2019, Example 1],

Example 2.3.1. Suppose that x, y, z are boolean program variables, and let μ be the output of:

$$x \xleftarrow{\$} \mathbf{Unif}_{\mathbb{B}}; y \xleftarrow{\$} \mathbf{Unif}_{\mathbb{B}}; z \leftarrow x \vee y$$

In other words, x and y store the results of two fair coin flips, and z stores the value of $x \vee y$. Then x and y are independent in μ, i.e., $\mathsf{Own}(x) * \mathsf{Own}(y)$ holds in μ. However, if $M \subseteq \mathbf{Mem}[\mathbf{Var}]$ is the set of all randomized memories where $z = tt$, representing the event that z is true, then $\mathsf{Own}([)x] * \mathsf{Own}([)y]$ does not

hold in $\mu \mid M$. Intuitively, if we know $z = tt$, then x and y are correlated: if one is false, then the other must be true.

We also need to be more careful when formulating the post-condition for conditionals. Even when both the true branch and the false branch guarantee ψ as the post-condition, it is in general unsound to conclude the post-condition ψ for **if** b **then** c **else** c'. We also illustrate this through an example.

Example 2.3.2. Suppose that x, y, z are boolean program variables, and let μ be the output of:

$$z \xleftarrow{\$} \mathbf{Bern}_{1/2};$$
$$\mathbf{if}\ z\ \mathbf{then}\ x \xleftarrow{\$} \mathbf{Bern}_{0.9}; y \xleftarrow{\$} \mathbf{Bern}_{0.9}$$
$$\mathbf{else}\ x \xleftarrow{\$} \mathbf{Bern}_{0.1}; y \xleftarrow{\$} \mathbf{Bern}_{0.1}$$

In both the true branch's output and the false branch's output, x and y are probabilistically independent, and thus validating $\mathsf{Own}(x) * \mathsf{Own}(y)$ as a post-condition. However, in μ, x and y are not independent: when x is true, then more likely is z true and the true branch to be executed, and thus y is also more likely to be true; the case is similar when x is false.

To make sure that the post-conditions from the branches can be combined into the post-condition of the conditional, the side condition of RCOND checks that the part of post-condition ψ determines *unique* portion of the distribution over randomized memories. Formally, we adapt the following class of assertions from separation logic [Reynolds, 2002].

Definition 2.3.7. A formula φ is *supported* (SP) if there exists a randomized memory μ such that if $\mu' \models \varphi$, then $\mu \sqsubseteq \mu'$.

We can prove by induction that the following syntactic conditions ensure SP.

Lemma 2.3.5. *The following assertions are SP:*

$$\eta ::= p_d \mid [x = v] \mid x \hookrightarrow \mu \mid \eta * \eta$$

Last, the loop rule LOOP is in the same style of COND, which also requires the guard to be deterministic as a consequence of the precondition φ. This side condition essentially restricts the loop to run a deterministic number of iterations. In that case, if we have precondition φ and the program c preserves φ as an invariant, then when the loop **while** b **do** c terminates, we have $\varphi \wedge [b = ff]$ as the post-condition.

Structural Rules The structural rules are in fig. 2.8b and they apply to Hoare triples with any command c as long as the pre- and post-conditions match. The rules WEAK, TRUE, CONJ, and CASE are standard.

CONST is the rule of constancy from Hoare logic, which states that, if a formula η does not mention any of c's modified variables $\mathrm{MV}(c)$, then it can be conjoined to the pre- and post-condition. This rule is *not* sound in standard separation logic — motivating the separating conjunction and the frame rule — but it *is* sound in Probabilistic Separation Logic because writes in pWhile cannot invalidate assertions about other variables.

But, the post-condition in CONST does not ensure that ψ and η use probabilistically independent variables. For this stronger guarantee, we need FRAME, whose side conditions mention several classes of variables. Roughly speaking, $\mathrm{RV}(c)$ is the set of variables that c may read from, while $\mathrm{WV}(c)$ is the set of variables that c *must* write to (before possibly reading from). $\mathrm{MV}(c)$ is the set of variables that c *may* write to, so $\mathrm{WV}(c)$ is a subset of $\mathrm{MV}(c)$. Formally,

Definition 2.3.8. RV, WV, MV are defined as follows:

$$\mathrm{RV}(x_r \leftarrow e_r) \triangleq \mathrm{FV}(e_r) \qquad \mathrm{RV}(x_r \xleftarrow{\$} \mu) \triangleq \emptyset$$

$$\mathrm{RV}(c\ ;\ c') \triangleq \mathrm{RV}(c) \cup (\mathrm{RV}(c') \setminus \mathrm{WV}(c))$$

$$\mathrm{RV}(\textbf{if } b \textbf{ then } c \textbf{ else } c') \triangleq \mathrm{FV}(b) \cup \mathrm{RV}(c) \cup \mathrm{RV}(c')$$

$$\mathrm{RV}(\textbf{while } b \textbf{ do } c) \triangleq \mathrm{FV}(b) \cup \mathrm{RV}(c)$$

$$\mathrm{WV}(x_r \leftarrow e_r) \triangleq \{x_r\} \setminus \mathrm{FV}(e_r) \qquad \mathrm{WV}(x_r \xleftarrow{\$} \mu) \triangleq \{x_r\}$$

$$\mathrm{WV}(c\ ;\ c') \triangleq \mathrm{WV}(c) \cup (\mathrm{WV}(c') \setminus \mathrm{RV}(c))$$

$$\mathrm{WV}(\textbf{if } b \textbf{ then } c \textbf{ else } c') \triangleq (\mathrm{WV}(c) \cap \mathrm{WV}(c')) \setminus \mathrm{FV}(b)$$

$$\mathrm{WV}(\textbf{while } b \textbf{ do } c) \triangleq \mathrm{WV}(c)$$

$$\mathrm{MV}(x_r \leftarrow e) \triangleq \{x_r\} \qquad \mathrm{MV}(x_r \xleftarrow{\$} \mu) \triangleq \{x_r\} \qquad \mathrm{MV}(c\ ;\ c') \triangleq \mathrm{MV}(c) \cup \mathrm{MV}(c')$$

$$\mathrm{MV}(\textbf{if } b \textbf{ then } c \textbf{ else } c') \triangleq \mathrm{MV}(c) \cup \mathrm{MV}(c') \qquad \mathrm{MV}(\textbf{while } b \textbf{ do } c) \triangleq \mathrm{MV}(c)$$

Last, FRAME says that we can conjoin a formula η to both the pre- and post-conditions if

1. η does not use any variables modified by the program c;
2. the program c only reads from the part of memories that the precondition φ describes;
3. the post-condition ψ only talks about variables that the precondition φ already describes or variables the program c writes to.

The first condition is standard in separation logic — separation logics for reasoning about heaps or concurrency also need an analogous condition. The second and the third condition are needed because our star $*$ asserts probabilistic

independence: if c reads from variables in η, then the post-condition ψ may not be independent from η; if ψ talks about variables that are neither written by the command c nor described by ψ, those variables may be already correlated with variables in η. Together, this set of conditions guarantees that η refers to variables that are probabilistically independent of ψ, thus validating $\psi * \eta$ as the post-condition.[2]

All these proof rules are sound.

Theorem 2.3.6 (Soundness). *If* $\vdash \{\varphi\}\ c\ \{\psi\}$ *is derivable, then* $\models \{\varphi\}\ c\ \{\psi\}$.

When proving the soundness of the program rules, we sometimes want to focus on a smaller distribution μ' inside a given distribution $\mu \models \varphi$, such that μ' satisfies some sub-formula of φ. Specifically, such reasoning is used in the proof for CASE, CONST, and FRAME. To ensure there exists such a smaller distribution, we require the assertion logic to satisfy a key condition called *restriction*, which says that to check whether a distribution satisfies φ, it suffices to check whether its marginalization on $\mathrm{FV}(\varphi)$ satisfies φ. BI formulas in $(\mathcal{X}_{\mathbb{D}}, \mathcal{V}^*)$ satisfies restriction:

Lemma 2.3.7 (Restriction). *Let* $\mu \in \mathcal{D}(\mathbf{Mem}[S])$ *and let* φ *be a BI formula. Then:*

$$\mu \models \varphi \Leftrightarrow (\sigma, \pi_{FV(\varphi)}(\mu)) \models \varphi.$$

We leave the proof for theorem 2.3.6 and lemma 2.3.7 to appendix A.

Barthe et al. [2019] demonstrates that probabilistic separation logic can be used to prove the correctness of various cryptographic schemes, where security relies on the independence of secrets and public information.

[2] There also exist other choices for the side conditions of FRAME — we stick with the choice by Barthe et al. [2019].

$$\textsc{Skip}\ \frac{}{\vdash \{\varphi\}\ \mathbf{skip}\ \{\varphi\}} \qquad \textsc{Seqn}\ \frac{\vdash \{\varphi\}\ c\ \{\psi\} \qquad \vdash \{\psi\}\ c'\ \{\eta\}}{\vdash \{\varphi\}\ c\ ;\ c'\ \{\eta\}}$$

$$\textsc{DAssn}\ \frac{}{\vdash \{\mathbf{Detm}\langle e\rangle \wedge \varphi[e/x]\}\ x \leftarrow e\ \{\mathbf{Detm}\langle x\rangle \wedge \varphi\}}$$

$$\textsc{RAssn}\ \frac{x_r \notin \mathrm{FV}(e_r)}{\vdash \{\top\}\ x_r \leftarrow e_r\ \{[x_r = e_r]\}} \qquad \textsc{Samp}\ \frac{}{\vdash \{\top\}\ x_r \xleftarrow{\$} \mu\ \{x_r \overset{\$}{\sim} \mu\}}$$

$$\textsc{Cond}\ \frac{\vdash \{\varphi \wedge [b = tt]\}\ c\ \{\psi\} \qquad \vdash \{\varphi \wedge [b = ff]\}\ c'\ \{\psi\} \qquad \models \varphi \rightarrow \mathbf{Detm}\langle b\rangle}{\vdash \{\varphi\}\ \mathbf{if}\ b\ \mathbf{then}\ c\ \mathbf{else}\ c'\ \{\psi\}}$$

$$\textsc{RCond}\ \frac{\vdash \{\varphi * [b = tt]\}\ c\ \{\psi * [b = tt]\} \qquad \vdash \{\varphi * [b = ff]\}\ c'\ \{\psi * [b = ff]\} \qquad \psi \in \mathrm{SP}}{\vdash \{\varphi * \mathsf{Own}(b)\}\ \mathbf{if}\ b\ \mathbf{then}\ c\ \mathbf{else}\ c'\ \{\psi * \mathsf{Own}(b)\}}$$

$$\textsc{Loop}\ \frac{\vdash \{\varphi \wedge [b = tt]\}\ c\ \{\varphi\} \qquad \models \varphi \rightarrow \mathbf{Detm}\langle b\rangle}{\vdash \{\varphi\}\ \mathbf{while}\ b\ \mathbf{do}\ c\ \{\varphi \wedge [b = ff]\}}$$

(a) Program Rules of Probabilistic Separation Logic

$$\textsc{Weak}\ \frac{\vdash \{\varphi\}\ c\ \{\psi\} \qquad \models \varphi' \rightarrow \varphi \wedge \psi \rightarrow \psi'}{\vdash \{\varphi'\}\ c\ \{\psi'\}} \qquad \textsc{True}\ \frac{}{\vdash \{\top\}\ c\ \{\top\}}$$

$$\textsc{Conj}\ \frac{\vdash \{\varphi_1\}\ c\ \{\psi_1\} \qquad \vdash \{\varphi_2\}\ c\ \{\psi_2\}}{\vdash \{\varphi_1 \wedge \varphi_2\}\ c\ \{\psi_1 \wedge \psi_2\}} \quad \textsc{Case}\ \frac{\vdash \{\varphi_1\}\ c\ \{\psi_1\} \qquad \vdash \{\varphi_2\}\ c\ \{\psi_2\}}{\vdash \{\varphi_1 \vee \varphi_2\}\ c\ \{\psi_1 \vee \psi_2\}}$$

$$\textsc{Const}\ \frac{\vdash \{\varphi\}\ c\ \{\psi\} \qquad \mathrm{FV}(\eta) \cap \mathrm{MV}(c) = \emptyset}{\vdash \{\varphi \wedge \eta\}\ c\ \{\psi \wedge \eta\}}$$

$$\textsc{Frame}\ \frac{\begin{array}{c}\vdash \{\varphi\}\ c\ \{\psi\} \qquad \mathrm{FV}(\eta) \cap \mathrm{MV}(c) = \emptyset \\ \models \varphi \rightarrow \mathsf{Own}(T \cup \mathrm{RV}(c)) \qquad \mathrm{FV}(\psi) \subseteq T \cup \mathrm{RV}(c) \cup \mathrm{WV}(c)\end{array}}{\vdash \{\varphi * \eta\}\ c\ \{\psi * \eta\}}$$

(b) Structural Rules of Probabilistic Separation Logic

Figure 2.8: Rules of Probabilistic Separation Logic

CHAPTER 3

A PROGRAM LOGIC FOR NEGATIVE DEPENDENCE

3.1 Overview

In the last chapter, we have seen a program logic for reasoning about probabilistic independence. While independence is useful for many applications, it is a strict requirement. A natural question is, what if we do not have perfect independence? Can we use other kind of probabilistic dependencies in program analysis?

Utilizing probabilistic dependencies is, for example, important when we analyze hashing-based probabilistic data structures such as hash tables and Bloom filters. In these applications, a hash function h maps a universe of possible values, typically large, to a set of buckets, typically small, and items are looked up through their hashes. The performance of hash-based data structures is captured by a variety of probabilistic guarantees, e.g., the space usage, the amortized cost of insertion, the amortized cost of look-up, etc. One useful probabilistic guarantee is the false positive rate: the probability that a data structure mistakenly identifies an element as being stored in the data structure, when it was not inserted. We may also be interested in load measures, such as the probability that a bucket in the data structure overflows. A typical way to analyze these quantities is to treat random hash functions as balls-into-bins processes. For example, hashing unique elements into bins can be modeled as throwing balls into bins, where each bin is drawn uniformly at random.

While this modeling is convenient, one complication is that the counts of

the elements in the different buckets are *not* probabilistically independent: one bin containing many elements makes it more likely that other bins contain few elements. The lack of independence makes it difficult to reason about multiple bins, for instance, bounding the number of occupied bins. Moreover, many common tools for analyzing probabilistic processes, like concentration bounds, usually require independence. This subtlety has also been a source of problems in pen-and-paper analyses of probabilistic data structures (e.g., Mullin [1983], Blustein and El-Maazawi [2002]). After many attempts to correct the bounds for Bloom filter's false positive rate [Bose et al., 2008, Christensen et al., 2010] using pen-and-paper proofs, recently, Gopinathan and Sergey [2020] certified its analysis using a complex proof in Rocq. We aim to develop a simpler method to formally reason about hash-based data structures and balls-into-bins processes, drawing on a key concept in probability theory: negative dependence.

While there are multiple incomparable definitions of negative dependence, Joag-Dev and Proschan [1983] proposed a notion called *negative association* (NA) that shares many good probabilistic properties of probabilistic independence. First, some standard theorems about sums of independent random variables apply more generally to sums of NA random variables. In particular, the widely-used *Chernoff bound*, which intuitively says that the sum of independent random variables is close to the expected value of the sum with high probability, holds also for NA variables. Intuitively, it is unlikely for all variables to attain high values compared to their expected value, and equally unlikely for all variables to attain low values; thus, their sum most likely stays close with the expected value of the sum. Second, negative association is preserved by some common operations on random variables. For instance, variables that use an

independent source of randomness are independent — a crucial property that validates FRAME in probabilistic separation logic; while this does not hold for negative associated variables, the following variation holds: if X, Y are negatively associated, and Z is obtained by applying monotone map on X, then Z, Y are also negatively associated. Such closure properties allow one to prove negative dependence in a compositional way.

In this chapter, we introduce a program logic for proving and utilizing negative association and independence. While probabilistic separation logic introduced in the last chapter can assert independence using the multiplicative conjunction, its assertion logic cannot express negative association. Inspired by this approach, we think of asserting negative association using another multiplicative conjunction, but that means we need to support multiple multiplicative conjunctions in the assertion logic. For that purpose, we propose M-BI, an extension to bunched logic where each element in M is associated with its own multiplicative conjunction and implication. In the following, we first present a BI model for asserting negative association, then combine it with the BI model for asserting probabilistic independence into an M-BI model, and last, we design a program logic that incorporates compositional proof principles of NA.

3.2 Negative Association

We now define negative association precisely and state its properties. Negative association is a property of *a set of* random variables, formalized as follows:

Definition 3.2.1 (Negative Association (NA))**.** Let $X_1, \ldots X_n$ be random variables. The set $\{X_i\}_i$ is *negatively associated (NA)* if for every pair of subsets

$I, J \subseteq \{1, \ldots, n\}$ such that $I \cap J = \emptyset$, and every pair of both monotone or both antitone functions[1] $f : \mathbb{R}^{|I|} \to \mathbb{R}$ and $g : \mathbb{R}^{|J|} \to \mathbb{R}$, where f, g is either bounded below or bounded above,[2] we have:

$$\mathbb{E}\big[f(X_i, i \in I) \cdot g(X_j, j \in J)\big] \leq \mathbb{E}[f(X_i, i \in I)] \cdot \mathbb{E}\big[g(X_j, j \in J)\big].$$

We can view NA as generalizing *independence*: a set of independent random variables is NA because equality holds. NA also strengthens *negative covariance*, a simpler notion of negative dependence that occurs frequently in statistics literature. Negative correlation [Rice, 2007, Chapter 4.3] of $X_1, \ldots, X_n$ says that

$$\mathbb{E}\left[\prod_{i \in [n]} X_i\right] \leq \prod_{i \in [n]} \mathbb{E}[X_i],$$

which automatically holds if $\{X_1, \ldots, X_n\}$ are negatively associated. To see that, we show the following:

Lemma 3.2.1. *Let $X_1, \ldots, X_n$ be a sequence of NA random variables, then for any family of non-negative all monotone or all antitone functions $f_i : \mathbb{R} \to \mathbb{R}$,*

$$\mathbb{E}\left[\prod_{i \in [n]} f_i(X_i)\right] \leq \prod_{i \in [n]} \mathbb{E}[f_i(X_i)].$$

Proof. We prove it by induction. The base case is when $n = 1$, then trivially,

$$\mathbb{E}\left[\prod_{i \in [n]} f_i(X_i)\right] = \mathbb{E}[f_1(X_1)] \leq \prod_{i \in [n]} \mathbb{E}[f_1(X_1)].$$

[1] In the following, we will consistently use monotone to mean monotonically non-decreasing and antitone to mean monotonically non-increasing.

[2] Technically, we slightly modify Dubhashi and Ranjan [1998]'s NA by in addition assuming that f, g are bounded from one side. We add the condition to have a cleaner version of theorem 3.3.1 and Theorem 3.3.5. All our other results and properties we state about NA in Section 3.2 hold with or without this condition.

When $n > 1$, note that the map $(X_1, \ldots, X_{n-1}) \to \prod_{i\in[n-1]} f_i(X_i))$ is also monotone if all f_i are monotone, and antitone if all f_i are antitone,

$$\begin{aligned}
\mathbb{E}\left[\prod_{i\in[n]} f_i(X_i)\right] &= \mathbb{E}\left[\left(\prod_{i\in[n-1]} f_i(X_i)\right)\cdot f_n(x_n)\right] \\
&= \mathbb{E}\left[\left(\prod_{i\in[n-1]} f_i(X_i)\right)\right]\cdot \mathbb{E}[f_n(x_n)] \quad \text{(Because the variables are NA)} \\
&= \prod_{i\in[n]} \mathbb{E}[f_i(X_i)] \quad \text{(By inductive hypothesis)}
\end{aligned}$$

□

In particular, when we take all f_i to be identity functions, we derive $\mathbb{E}\left[\prod_{i\in[n]} X_i\right] \leq \prod_{i\in[n]} \mathbb{E}[X_i]$ from variables being NA.

NA variables can arise from various mechanisms.

Theorem 3.2.2 (See Dubhashi and Ranjan [1998])**.** *We enumerate three scenarios:*

1. *The set of independent random variables $\{X_1, \ldots, X_n\}$ is negatively associated.*
2. *If $\{X_1, \ldots, X_n\}$ are Bernoulli random variables such that $\sum_{i\in[n]} X_i = 1$, then the set of variables is negatively associated.*
3. *Let X be a uniformly random permutation of a finite, nonempty multi-set A, and for each i, let X_i be the i-th entry in the vector X. Then $\{X_1, \ldots, X_n\}$ is negatively associated.*

As an example of the second case, consider a deck of cards perfectly shuffled — so that the cards' order is uniformly sampled from all possible permutations. If, for each i, X_i gets the value on the i-th card, then the variables $\{X_i\}_i$ are negatively associated. Also, the third case of this theorem implies that if we draw

a length-n *one-hot vector*, i.e., a vector that has one entry being one and all remaining entries being zero, uniformly at random, then the entries of the vector satisfies negative association.

The following theorem states three key closure properties of NA random variables.

Theorem 3.2.3 (See Dubhashi and Ranjan [1998]). *We enumerate three scenarios:*

1. *For any negatively associated set of variables T, and for any S that is a non-empty subset of T, the set S of random variables is negatively associated;*
2. *For any two sets of negatively associated random variables T, U such that every $X \in T$ and $Y \in U$ is independent of each other, the union set $T \cup U$ of random variables is negatively associated.*
3. *Let $\{X_1, \ldots X_n\}$ be negatively-associated, and $I_1, \ldots, I_m$ be a partition of the set $\{1, \ldots, n\}$. For each $1 \leq j \leq m$, let $f_j : \mathbb{R}^{|I_j|} \to \mathbb{R}$ be monotone. Let $S = \{f_1(X_k, k \in I_1), \ldots, f_m(X_k, k \in I_m)\}$. Then S is negatively associated.*

The first case shows that NA is preserved if we discard random variables, while the second case allows us to join two independent sets of negatively associated random variables to form a larger negatively associated set. Finally, the third case guarantees that negative association is preserved under applying monotone maps on disjoint subsets of variables.

Chernoff's Bound and Negative Association Another nice property of NA is that negatively associated random variables satisfies some frequently used tail bounds, including Chernoff's bound.

> Chernoff's bound is one of the most basic and versatile tools in the life of a theoretical computer scientist, with a seemingly endless amount of applications. — Mulzer [2019]

Qualitatively, Chernoff's bound says the sum $X_1 + \cdots + X_n$ is usually close to its expected value, and upper bounds the probability that the sum deviates from the mean for more than a tolerated amount. This kind of analysis is useful for establishing *high-probability guarantees* of randomized algorithms, e.g., showing that the error of a random estimate is at most 0.01 with probability at least 99%. There are various formulations of Chernoff's bound, with different assumptions of the random variables (e.g., $\{X_i\}_i$ being independent Bernoulli random variables, or $\{X_i\}_i$ simply being independent bounded random variables) and different ways to measure the error (e.g., the additive form uses the absolute difference between the realized value and the expected value, and the multiplicative form uses error ratio). While the mainstream formulation of Chernoff's bound all require the variables $\{X_i\}_i$ to be independent, Dubhashi and Ranjan [1998] observes that Chernoff's bound also holds on negatively associated random variables.

We state the result using a formulation in the additive form for $[0, 1]$ bounded random variables. This version is also known as the *Hoeffding's inequality*.

Theorem 3.2.4 (Chernoff-Hoeffiding Bound for NA variables [Dubhashi and Ranjan, 1998])**.** *Let $X_1, \ldots, X_n$ be a sequence of NA random variables, each bounded in $[0, 1]$, and let $Y = \sum_{i=1}^{n} X_i$. Then for any failure probability $\beta \in (0, 1]$, we have:*

$$\Pr[|Y - \mathbb{E}[Y]| \geq \beta] \leq F(\beta, n) \quad \textit{where } F(\beta, n) = e^{-2\beta^2/n}.$$

Or, an equivalent way to express it is,

$$\Pr[|Y - \mathbb{E}[Y]| \geq T(\beta, n)] \leq \beta \quad \textit{where } T(\beta, n) = \sqrt{(n/2)\ln(1/\beta)}.$$

Proof. The proof uses Hoeffding's lemma: for any real-valued random variable X such that $X \in [a, b]$ almost surely. Then for any $\lambda \in \mathbb{R}$, $\mathbb{E}\left[e^{\lambda(X-\mathbb{E}[X])}\right] \leq e^{\lambda^2(b-a)^2/8}$. (See, e.g., Romaní [2021], for the proof of Hoeffding's lemma.)

For any $\lambda > 0$, the event $Y - \mathbb{E}[Y] \geq \beta$ is the same as the event $e^{\lambda(Y-\mathbb{E}[Y])} \geq e^{\lambda\beta}$,

$$\Pr(Y - \mathbb{E}[Y] \geq \beta) = \Pr(e^{\lambda(Y-\mathbb{E}[Y])} \geq e^{\lambda\beta}).$$

By Markov inequality, for any positive random variable X, $\Pr(X \geq a) \leq \frac{\mathbb{E}[X]}{a}$. By regarding $e^{\lambda(Y-\mathbb{E}[Y])}$ as the random variable, the Markov inequality gives us

$$\Pr(e^{\lambda(Y-\mathbb{E}[Y])} \geq e^{\lambda\cdot\beta}) \leq \frac{\mathbb{E}\left[e^{\lambda(Y-\mathbb{E}[Y])}\right]}{e^{\lambda\cdot\beta}}$$

Now we analyze the nominator on the right,

$$\begin{aligned}
\mathbb{E}\left[e^{\lambda(Y-\mathbb{E}[Y])}\right] &= \mathbb{E}\left[e^{\lambda(\sum_{i\in[n]} X_i - \mathbb{E}\left[\sum_{i\in[n]} X_i\right])}\right] && \text{(By definition of } Y\text{)} \\
&= \mathbb{E}\left[e^{\lambda\cdot\sum_{i\in[n]}(X_i-\mathbb{E}[X_i])}\right] && \text{(By linearity of expectation)} \\
&= \mathbb{E}\left[\prod_{i\in[n]} e^{\lambda\cdot(X_i-\mathbb{E}[X_i])}\right] \\
&\leq \prod_{i\in[n]} \mathbb{E}\left[e^{\lambda\cdot(X_i-\mathbb{E}[X_i])}\right] && \text{(Because } \{X_i\}_i \text{ are NA and by lemma 3.2.1)} \\
&\leq \prod_{i\in[n]} e^{\lambda^2/8}. && \text{(Hoeffding's lemma)}
\end{aligned}$$

Therefore,

$$\Pr(e^{\lambda(Y-\mathbb{E}[Y])} \geq e^{\lambda\cdot\beta}) \leq \frac{\prod_{i\in[n]} e^{\lambda^2/8}}{e^{\lambda\cdot\beta}} = e^{(n\cdot\lambda^2/8)-\lambda\cdot\beta}$$

Similarly,

$$
\begin{aligned}
\Pr(\mathbb{E}[Y] - Y \geq \beta) &= \Pr(e^{\lambda(\mathbb{E}[Y]-Y)} \geq e^{\lambda\cdot\beta}) \\
&= \frac{\mathbb{E}\left[e^{\lambda\cdot\sum_{i\in[n]}(\mathbb{E}[X_i]-X_i)}\right]}{e^{\lambda\cdot\beta}} && \text{(By linearity of expectation)} \\
&\leq \frac{\prod_{i\in[n]}\mathbb{E}\left[e^{\lambda\cdot(\mathbb{E}[X_i]-X_i)}\right]}{e^{\lambda\cdot\beta}} \\
& && \text{(Because } \{X_i\}_i \text{ are NA and by lemma 3.2.1)} \\
&\leq \frac{\prod_{i\in[n]} e^{\lambda^2/8}}{e^{\lambda\cdot\beta}} && \text{(Hoeffding's lemma)} \\
&= e^{(n\cdot\lambda^2/8)-\lambda\cdot\beta}
\end{aligned}
$$

The λ that minimizes $e^{(n\cdot\lambda^2/8)-\lambda\cdot\beta}$ is the λ that minimizes $(n\cdot\lambda^2/8)-\lambda\cdot\beta$, which is $\frac{4\beta}{n}$. Substitute $\frac{4\beta}{n}$ for λ in $e^{(n\cdot\lambda^2/8)-\lambda\cdot\beta}$, we can reduce the bound $e^{(n\cdot\lambda^2/8)-\lambda\cdot\beta}$ into $e^{-2\beta^2/n}$.

□

Crucially, the step that previously relied on independence of $\{X_i\}_i$ now follows from $\{X_i\}_i$ being NA and lemma 3.2.1.

3.3 A BI Frame for Negative Dependence

Now that we have seen some nice properties of negative association, we start the quest of building a bunched logic that can assert both negative association and independence. Concretely, we construct a BI frame $\mathcal{X}_{\text{PNA}}$ that can capture negative association and then combine it with our BI model for probabilistic independence $(\mathcal{X}_{\mathbb{D}}, \mathcal{V}^*)$. To be compatible with $(\mathcal{X}_{\mathbb{D}}, \mathcal{V}^*)$, we let $\mathcal{X}_{\text{PNA}}$ have the same set of states and the same pre-order as $\mathcal{X}_{\mathbb{D}}$. The important remaining piece

of the puzzle is the binary operation $\oplus$, which must satisfy the frame conditions while capturing negative association.

The meaning of "capturing negative association" has so far been left ambiguous. Previously, in the design process of $(\mathcal{X}_{\mathbb{D}}, \mathcal{V}^*)$, we want the satisfaction of $P * Q$ ensures that P and Q hold on independent components of distributions, and more precisely, we choose to require all variables involved in P are independent of all variables involved in Q. Because $P * Q$ is interpreted through the binary operation $\otimes_{\mathbb{D}}$, we define $\otimes_{\mathbb{D}}$ to take the independent product of two distributions when possible. Now, analogously, we want to interpret the formula $P \circledast Q$ through a binary operation $\oplus$:

$$x \models_{\mathcal{V}^*} P \circledast Q \quad \text{iff} \quad \text{there exist } x', y, z \text{ s.t. } x \sqsupseteq x' \in y \oplus z,\ y \models_{\mathcal{V}^*} P \text{ and } z \models_{\mathcal{V}^*} Q$$

such that $P \circledast Q$ ensures that P and Q hold on negatively associated components of distributions — we use $\circledast$ for the separating conjunction interpreted on $\mathcal{X}_{\text{PNA}}$ to distinguish it from the separating conjunction for asserting independence. But negative association is defined for a set of variables instead of two (groups) of variables, and it is unclear what "P, Q holds on negatively associated components" should mean. In the following, we explore several plausible definitions of $\oplus$, with the goal that we can express a set of variables $x_1, \ldots, x_n$ is negatively associated using formulas involving $\circledast$.

3.3.1 Initial Attempts at a BI Frame for Negative Association

One first attempt is to let $\mu_1 \oplus \mu_2$ be the set of distributions that agree with μ_1, μ_2, and satisfy *strong NA* — we say μ satisfies strong NA if $\mathbf{dom}(\mu)$ satisfies NA.

Definition 3.3.1. (Attempt 1: Strong NA model) Recall that $X_{\mathbb{D}} = E_{\mathbb{D}} = \cup_{S \subseteq \mathbf{Var}} \mathcal{D}(\mathbf{Mem}[S])$, and for $\mu, \mu' \in X_{\mathbb{D}}$, we have $\mu \sqsubseteq_{\mathbb{D}} \mu'$ iff $\mathbf{dom}(\mu) \subseteq \mathbf{dom}(\mu')$ and $\pi_{\mathbf{dom}(\mu)}\mu' = \mu$. Define $\oplus_s : X_{\mathbb{D}} \times X_{\mathbb{D}} \to \mathcal{P}(X_{\mathbb{D}})$:

$$\mu_1 \oplus_s \mu_2 = \{\mu \in \mathcal{D}(\mathbf{Mem}[S \cup T]) \mid \mu \text{ satisfies strong NA}, \pi_S \mu = \mu_1, \pi_T \mu = \mu_2, S \cap T = \emptyset\}.$$

We call $\mathcal{X}_s = (X, \sqsubseteq, \oplus_s, E_s)$ the *strong NA structure*.

Unfortunately, the strong NA structure fails to satisfy the Unit Existence condition: if μ does not satisfy strong NA, then there exists no μ' that marginalizes to μ and satisfies strong NA; and because our definition of $e \oplus_s \mu$ only includes distributions μ' that marginalize to μ, then there is no e such that $e \oplus_s \mu$ is not empty. The failure of this property implies that whether or not two states can be combined depends not just on how the two states relate to each other, but also critically on properties of the single states in isolation (e.g., whether a distribution satisfies strong NA); this is hard to justify if we are to read $\oplus$ as describing which pairs of states can be safely combined.

Looking for a different way of capturing NA, we try working with a weaker notion of NA. We try letting $\mu_1 \oplus \mu_2$ return distributions that agree with μ_1, μ_2 where any variable x in $\mathbf{dom}(\mu_1)$ must be negatively associated with any variable y in $\mathbf{dom}(\mu_2)$, but variables within $\mathbf{dom}(\mu_1)$ and variables within $\mathbf{dom}(\mu_2)$ need not be negatively associated. We call this notion *weak NA*.

Definition 3.3.2 (Weak NA)**.** Let $S \subseteq \mathbf{Var}$ be a set of variables, and let A, B be two disjoint subsets of S. A distribution $\mu \in \mathcal{D}(\mathbf{Mem}[S])$ satisfies (A, B)-*NA* if for every pair of both monotone or both antitone functions $f : \mathbf{Mem}[A] \to \mathbb{R}$, $g : \mathbf{Mem}[B] \to \mathbb{R}$, where we take the point-wise orders on $\mathbf{Mem}[A]$ and $\mathbf{Mem}[B]$, such that f, g is either lower bounded or upper bounded, we have

$$\mathbb{E}_{m \sim \mu}[f(\pi_A m) \cdot g(\pi_B m)] \leq \mathbb{E}_{m \sim \mu}[f(\pi_A m)] \cdot \mathbb{E}_{m \sim \mu}[g(\pi_B m)].$$

By definition, being (A, B)-NA for all disjoint $A, B \subseteq S$ is equivalent to strong NA on S. Also, (A, B)-NA is closed under projection in the sense that if μ satisfies (A, B)-NA and $A' \subseteq A$, $B' \subseteq B$¡ then μ satisfies (A', B')-NA as well. Now, we try defining a model based on weak NA.

Definition 3.3.3. (Attempt 2: Weak NA model) Define $\oplus_w : X_{\mathbb{D}} \times X_{\mathbb{D}} \to \mathcal{P}(X_{\mathbb{D}})$:

$$\mu_1 \oplus_w \mu_2 = \{\mu \in \mathcal{D}(\mathbf{Mem}[S \cup T]) \mid \mu \text{ satisfies } (S, T)\text{-NA}, \pi_S \mu = \mu_1, \pi_T \mu = \mu_2, S \cap T = \emptyset\}.$$

We call $\mathcal{X}_w = (X_{\mathbb{D}}, \sqsubseteq_{\mathbb{D}}, \oplus_w, E_{\mathbb{D}})$ the *weak NA structure*.

This weak NA structure satisfies most BI frame conditions, except that Associativity is unclear. In short, the definition of $\oplus_w$ and Associativity requires that: if w satisfies $(R \cup S, T)$-NA and (R, S)-NA, then w also satisfies (S, T)-NA and $(R, S \cup T)$-NA. Now w satisfies (S, T)-NA by projection closure, but it is unclear whether $(R, S \cup T)$-NA follows from these conditions. Failing to satisfy Associativity would lead to a logic where separating conjunction is not associative, and significantly more difficult to use. Since we could not find a counter-example nor prove that $\mathcal{X}_w$ satisfies Associativity and thus forms a BI frame, we will leave this question as an open problem and define another structure to capture negative association.

3.3.2 Our BI Frame for Negative Association

Facing the problems with the strong NA structure and the weak NA structures, we define a BI model for negative association based on a new notion of negative association called $\mathcal{S}$*-partition negative association* ($\mathcal{S}$-PNA), where $\mathcal{S}$ is a partition of a set of random variables. This notion *interpolates* weak NA and strong NA in

the following sense: when A, B are both sets of variables, $\{A, B\}$-PNA is equivalent to (A, B)-NA for disjoint A, B, and $\{\{x\} \mid x \in S\}$-PNA is equivalent to strong NA for distributions in $\mathcal{D}(\mathbf{Mem}[S])$.

We say a partition $\mathcal{S}'$ *coarsens* a partition $\mathcal{S}$ if $\cup\mathcal{S} = \cup\mathcal{S}'$ and for any $s' \in \mathcal{S}'$, $s' = \cup\mathcal{R}$ for some $\mathcal{R} \subseteq \mathcal{S}$. In particular, any partition $\mathcal{S}$ coarsens itself.

Definition 3.3.4 (Partition Negative Association)**.** A distribution μ is $\mathcal{S}$-PNA if and only if for any $\mathcal{T}$ that coarsens $\mathcal{S}$, for any family of *non-negative* monotone functions (or family of *non-negative* antitone functions) $\{f_A : \mathbf{Mem}[A] \to \mathbb{R}^+\}_{A\in\mathcal{T}}$,[3] where for each $A \in \mathcal{T}$ the order on $\mathbf{Mem}[A]$ is taken to be the pointwise order, we have

$$\mathbb{E}_{m\sim\mu}\left[\prod_{A\in\mathcal{T}} f_A(\pi_A m)\right] \leq \prod_{A\in\mathcal{T}} \mathbb{E}_{m\sim\mu}[f_A(\pi A m)].$$

We can use PNA to encode NA:

Theorem 3.3.1. *Given a set of variables S, S satisfies NA in μ iff μ satisfies $\mathcal{S}$-PNA for any $\mathcal{S}$ partitioning S iff μ satisfies $\{\{x\} \mid x \in S\}$-PNA.*

See appendix B.2.1 for the proof. We require PNA to be closed under coarsening, which helps us to prove the structure defined next is a BI frame.

Definition 3.3.5. Define the operation $\oplus : X_{\mathbb{D}} \times X_{\mathbb{D}} \to \mathcal{P}(X_{\mathbb{D}})$:

$$\begin{aligned}\mu_1 \oplus \mu_2 = \{\mu \in \mathcal{D}(\mathbf{Mem}[S \cup T]) \mid \pi_S\mu = \mu_1, \pi_T\mu = \mu_2,\\ \mu \text{ is } (\mathcal{S} \cup \mathcal{T})\text{-PNA for any partition } \mathcal{S}, \mathcal{T} \text{ such that}\\ \mu_1 \text{ is } \mathcal{S}\text{-PNA}, \mu_2 \text{ is } \mathcal{T}\text{-PNA, and } (\cup\mathcal{S}) \cap (\cup\mathcal{T}) = \emptyset.\}\end{aligned}$$

[3] We restrict the family of functions to be non-negative: prior work like Joag-Dev and Proschan [1983] has assumed non-negativity when working with notions of NA on partitions; furthermore, without that requirement, for partitions with an odd number of components, PNA would be equivalent to independence, a strange property.

This definition of $\oplus$ interpolates $\oplus_w$ and $\oplus_s$, in the following sense.

Theorem 3.3.2. *For any two states $\mu_1, \mu_2 \in X$, $\mu_1 \oplus_s \mu_2 \subseteq \mu_1 \oplus \mu_2 \subseteq \mu_1 \oplus_w \mu_2$.*

The first inclusion is because μ satisfying strong NA implies μ is $\mathcal{R}$-PNA for any partition $\mathcal{R}$ on $\mathbf{dom}(\mu)$. The second inclusion is because $\mu_1 \in \mathcal{D}(\mathbf{Mem}[S])$ satisfies $\{S\}$-PNA and $\mu_2 \in \mathcal{D}(\mathbf{Mem}[T])$ satisfies $\{T\}$-PNA trivially, which implies any $\mu \in \mu_1 \oplus \mu_2$ would satisfy (S, T)-NA.

Note that $\oplus$ is non-deterministic, and not just partial.

Theorem 3.3.3. *There are distributions μ_1, μ_2 such that $|\mu_1 \oplus \mu_2| \geq 2$.*

Proof. Let $\mu_1 \in \mathcal{D}(\mathbf{Mem}[\{x\}])$ and $\mu_2 \in \mathcal{D}(\mathbf{Mem}[\{y\}])$ be uniform distribution over memories over boolean variables x, y. Then the independent product $\mu_* \in \mu_1 \otimes_{\mathbb{D}} \mu_2$ is in $\mu_1 \oplus \mu_2$, because the projections to x and to y are μ_1 and μ_2 respectively, and μ_* satisfies PNA since independence implies PNA (we will see this shortly in Theorem 3.4.5). But the one-hot uniform distribution μ_{oh} over variables x and y, i.e., $\mu_{oh}([x \mapsto 1, y \mapsto 0]) = \mu_{oh}([x \mapsto 0, y \mapsto 1]) = 1/2$, is also in $\mu_1 \oplus \mu_2$, since again the projections match μ_1 and μ_2 and the one-hot distribution satisfies NA, and hence PNA. Since $\mu_{oh} \neq \mu_*$, we are done. □

Thus, we build the following BI frame that crucially uses a non-deterministic binary operation on states to capture negative association.

Theorem 3.3.4. *The structure $\mathcal{X}_{PNA} = (X_{\mathbb{D}}, \sqsubseteq_{\mathbb{D}}, \oplus, E_{\mathbb{D}})$ is a Down-Closed BI frame.*

For the frame conditions where the previous attempts failed, (Unit Existence) holds by letting the unit e to always be the trivial distribution on the empty set, and (Associativity) can be proved using the facts that PNA is closed

under coarsening and coarsening commute with projections. See the full proof in Appendix B.2.2.

Furthermore, the binary combination $\oplus$ in $\mathcal{X}_{\mathrm{PNA}}$ captures negative association: consider the atomic propositions introduced in eq. (2.3) and the valuation $\mathcal{V}^*$ for them, clearly $(\mathcal{X}_{\mathrm{PNA}}, \mathcal{V}^*)$ forms a BI model; when interpreting BI formulas on the model $(\mathcal{X}_{\mathrm{PNA}}, \mathcal{V}^*)$, we can express negative association of a set of variables using the separating conjunction $\circledast$. We use the iterative version of the connective $*$, which is well-defined because it is associative.

Definition 3.3.6. For any connective $\odot \in \{\wedge, \vee, \circledast, *\}$, we use the corresponding big-connective $\bigodot \in \left\{\bigwedge, \bigvee, \circledast, *\right\}$.

- For any constant or logical variable $N \geq 1$, let $\bigodot_{i=0}^{N} P_i = P_0$ abbreviate $((P_0 \odot P_1) \odot \cdots) \odot P_{N-1}$. Formally, let $\bigodot_{i=0}^{N} P_i = \top$ if $N = 0$, and let $\bigodot_{i=0}^{N} P_i \triangleq \left(\bigodot_{i=0}^{N-1} P_i\right) \odot P_N$ for $N > 0$.
- For a finite multiset of formula $\{P_i\}_{i \in S}$, let $\bigodot_{s \in S} P_s$ abbreviate $((P_{s_0} \odot P_{s_1}) \odot \cdots) \odot P_{s_k}$, where $s_0, \ldots, s_k$ is an arbitrary ordering of S. The satisfaction is not ambiguous since $\odot$ is associative and commutative.
- For any program variable $v \in \mathbf{Var}$, for any state $\mu \models [v = N]$, we want $\bigodot_{i=0}^{v} P_i$ to be equivalent to $\bigodot_{i=0}^{N} P_i$. Formally, $\bigodot_{i=0}^{v} P_i$ abbreviates $\bigvee_{N \in \mathbf{Val}}([v = N] \wedge \bigodot_{i=0}^{N} P_i)$.

Theorem 3.3.5. *Let S be any subset of* **Var**. *A set of randomized program variables $Y = \{y_i \mid 0 \leq i < K\}$ satisfies NA in the distribution $\mu \in \mathcal{D}(\mathbf{Mem}[S])$ if and only if we have $\mu \models \circledast_{i=0}^{K} \mathsf{Own}(y_i)$.*

Proof. Forward direction: We denote $\{y_i\}$ as $Y[i]$, and denote $\{y_i \mid 0 \leq i < j\}$ as $Y[:j]$. We prove by induction on j that $\pi_{Y[:j]}\mu \models \circledast_{i=0}^{j} \mathsf{Own}(y_i)$.

Base case $j = 1$: Trivially, $\pi_{Y[:1]}\mu \models \mathsf{Own}(y_0)$, and then by persistence.

Inductive case $j \geq 1$: Assuming $\pi_{Y[:j]}\mu \models \circledast_{i=0}^{j} \mathsf{Own}(y_i)$. Since Y satisfies NA in μ, by Theorem 3.3.1, μ is $\mathcal{T}$-PNA for any partition $\mathcal{T}$ of Y. In particular, for any partition $\mathcal{T}_1$ on $Y[:j]$ and any (trivial) partition $\mathcal{T}_2$ on $Y[j]$, μ must be $\mathcal{T}_1 \cup \mathcal{T}_2$-PNA. Thus, $\pi_{Y[:j+1]}\mu \in \pi_{Y[:j]}\mu \oplus \pi_{y[j+1]}\mu$. Since $\pi_{Y[j]}\mu \models \circledast_{i=0}^{j} \mathsf{Own}(y_i)$ and $\pi_{Y[j]}\mu \models \mathsf{Own}(y_j)$, that implies $\pi_{Y[:j+1]} \models \circledast_{i=0}^{j+1} \mathsf{Own}(y_i)$.

Thus, we have $\pi_Y\mu \models \circledast_{i=0}^{K} \mathsf{Own}(y_i)$. By persistence, $\mu \models \circledast_{i=0}^{K} \mathsf{Own}(y_i)$.

Backward direction: for any A, B being disjoint subsets of $[n]$, by commutativity and associativity of $\circledast$, we can reorder formula and get

$$\mu \models \left(\circledast_{i \in A} \mathsf{Own}(y_i) \circledast \circledast_{i \in B} \mathsf{Own}(y_i) \right) \circledast \circledast_{y_i \in [n] \setminus (A \cup B)} \mathsf{Own}(y_i)$$

By satisfaction rules, there exists μ_1, μ_2, μ' such that $\mu \sqsupseteq \mu' \in \mu_1 \oplus \mu_2$, and $\mu_1 \models \circledast_{i \in A} \mathsf{Own}(y_i)$, and $\mu_2 \models \circledast_{i \in B} \mathsf{Own}(y_i)$. Note that μ_1 is trivially $\{A\}$-PNA, and μ_2 is trivially $\{B\}$-PNA. Thus, μ' satisfies $\{A, B\}$-PNA.

Therefore, μ satisfies (A, B)-NA for any A, B being disjoint subsets of Y, i.e., μ satisfies NA on Y. □

3.4 M-BI: Combining BI Models

Now that we have a BI model for capturing independence and a BI model for capturing negative association, we want to combine them and design an assertion logic that can express both independence and negative association; furthermore, it would be helpful to internalize the fact that independence implies

negative association in the assertion logic. To achieve that goal, we now extend bunched logic to support multiple separating conjunctions related by a pre-order. While our motivation is to use one separating conjunction to assert independence, and use another to assert negative association, the logic potentially also has other interesting models.

3.4.1 The Syntax and Proof Rules

Let $\mathcal{AP}$ be a set of atomic propositions, and $(M, \leq)$ be a finite pre-order. The formula in the logic of M-bunched implications (M-BI) has the following grammar:

$$P, Q ::= p \in \mathcal{AP} \mid \top \mid I_{m \in M} \mid \bot \mid P \wedge Q \mid P \vee Q \mid P \to Q \mid P *_{m \in M} Q \mid P \mathrel{-\!\!*}_{m \in M} Q.$$

We associate each element of $m \in M$ with a separating conjunction $*_m$, a corresponding multiplicative identity I_m and a separating implication $\mathrel{-\!\!*}_m$. The proof system for M-BI is based on the proof system for BI, with an indexed copy of rules for each separation, and additionally has the $*$-WEAKENING rules. We present the full Hilbert-style proof system in fig. 3.1. The new rule $*$-WEAKENING simply says that the separation conjunction associated with a bigger element in M is weaker: if $m_1 \leq m_2$, then the assertion $P *_{m_1} Q$ implies $P *_{m_2} Q$.

We can derive analogous weakening rules for separating implications and multiplicative identities, in the reverse direction.

$$\frac{}{P \vdash P}\ \textsc{Ax} \qquad \frac{}{P \vdash \top}\ \textsc{Top} \qquad \frac{}{\bot \vdash P}\ \textsc{Bot} \qquad \frac{P \vdash R \qquad Q \vdash R}{P \vee Q \vdash R}\ \vee\text{-E}$$

$$\frac{P \vdash Q_i}{P \vdash Q_1 \vee Q_2}\ \vee\text{-I} \qquad \frac{P \vdash Q \qquad P \vdash R}{P \vdash Q \wedge R}\ \wedge\text{-I-R} \qquad \frac{Q \vdash R}{P \wedge Q \vdash R}\ \wedge\text{-I-L}$$

$$\frac{P \vdash Q_1 \wedge Q_2}{P \vdash Q_i}\ \wedge\text{-E} \qquad \frac{P \wedge Q \vdash R}{P \vdash Q \to R}\ \to\text{-I} \qquad \frac{P \vdash Q \to R \qquad P \vdash Q}{P \vdash R}\ \to\text{-E}$$

$$\frac{P \vdash R \qquad Q \vdash S}{P *_m Q \vdash R *_m S}\ *\text{-Conj} \qquad \frac{P *_m Q \vdash R}{P \vdash Q \mathrel{-\!\!*}_m R}\ \mathrel{-\!\!*}\text{-I} \qquad \frac{P \vdash Q \mathrel{-\!\!*}_m R \qquad S \vdash Q}{P *_m S \vdash R}\ \mathrel{-\!\!*}\text{-E}$$

$$\frac{}{P \dashv\vdash P *_m I_m}\ *\text{-Unit} \qquad \frac{}{P *_m Q \vdash Q *_m P}\ *\text{-Comm}$$

$$\frac{}{(P *_m Q) *_m R \dashv\vdash P *_m (Q *_m R)}\ *\text{-Assoc} \qquad \frac{m_1 \leq m_2}{P *_{m_1} Q \vdash P *_{m_2} Q}\ *\text{-Weakening}$$

Figure 3.1: Hilbert system for M-BI

Lemma 3.4.1. *The following rules are derivable in M-BI:*

$$\frac{m_1 \leq m_2}{P \mathrel{-\!\!*}_{m_2} Q \vdash P \mathrel{-\!\!*}_{m_1} Q}\ \mathrel{-\!\!*}\text{-Weakening} \qquad \frac{m_1 \leq m_2}{I_{m_2} \vdash I_{m_1}}\ \textsc{UnitWeakening}$$

3.4.2 Semantics

As is standard with bunched logics, we give a Kripke style semantics to M-BI. We will define a structure called M-BI *frame*, and then define M-BI *models* and the satisfaction rules on M-BI models.

An M-BI frame is a collection of BI frames sharing the same set of states and pre-order, with ordered binary operations.

Definition 3.4.1 (M-BI Frame)**.** An M-BI frame is a structure $\mathcal{X} = (X, \sqsubseteq, \oplus_{m\in M}, E_m)$ such that for each m, $(X, \sqsubseteq, \oplus_m, E_m)$ is a BI frame (see definition 2.2.1), and there is a preorder $\leq$ on M satisfying:

$$m_1 \leq m_2 \to x \oplus_{m_1} y \subseteq x \oplus_{m_2} y \qquad \text{(Operation Inclusion)}$$

The Operation Inclusion condition together with the frame conditions of BI imply an inclusion on unit sets:

Lemma 3.4.2. *Let $\mathcal{X}$ be an M-BI frame. If $m_1 \leq m_2$ then $E_{m_2} \subseteq E_{m_1}$.*

Proof. Let $e_2 \in E_{m_2}$. By Unit Existence, there exists $e_1 \in E_{m_1}$ such that $e_2 \in e_1 \oplus_{m_1} e_2$. By Operation Inclusion, $e_2 \in e_1 \oplus_{m_2} e_2$, so Unit Coherence implies that $e_1 \sqsubseteq e_2$, and then Unit Closure implies $e_2 \in E_{m_1}$. So $E_{m_2} \subseteq E_{m_1}$. □

To obtain a M-BI model over a given M-BI frame, we need a valuation that defines which states in the M-BI frame satisfy each atomic proposition. Again, for the soundness of the proof system, the valuation must be persistent: any formula true at a state remains true at any larger state.

Definition 3.4.2 (Valuation and model)**.** An M-BI model $(\mathcal{X}, \mathcal{V})$ is an M-BI frame $\mathcal{X} = (X, \sqsubseteq, \oplus_m, E_m)$ associated with a persistent valuation $\mathcal{V}$ on it.

Next, we define the satisfaction of M-BI formula in a M-BI model. The definition is almost the same as fig. 2.3, except that it supports the M-indexed separation conjunctions, implications, and units.

Definition 3.4.3. On an M-BI model $(\mathcal{X}, \mathcal{V})$, we define the satisfaction relation $\models_{\mathcal{V}}$ between states in $\mathcal{X}$ and M-BI formula: for any $x \in \mathcal{X}$,

x	$\models_{\mathcal{V}}$	$\top$		always
x	$\models_{\mathcal{V}}$	$\bot$		never
x	$\models_{\mathcal{V}}$	I_m	iff	$x \in E_m$
x	$\models_{\mathcal{V}}$	p	iff	$x \in \mathcal{V}(\mathrm{p})$
x	$\models_{\mathcal{V}}$	$P \wedge Q$	iff	$x \models_{\mathcal{V}} P$ and $x \models_{\mathcal{V}} Q$
x	$\models_{\mathcal{V}}$	$P \vee Q$	iff	$x \models_{\mathcal{V}} P$ or $x \models_{\mathcal{V}} Q$
x	$\models_{\mathcal{V}}$	$P \to Q$	iff	for all $y \sqsupseteq x$, $y \models_{\mathcal{V}} P$ implies $y \models_{\mathcal{V}} Q$
x	$\models_{\mathcal{V}}$	$P *_m Q$	iff	there exist x', y, z s.t. $x \sqsupseteq x' \in y \oplus_m z$, $y \models_{\mathcal{V}} P$ and $z \models_{\mathcal{V}} Q$
x	$\models_{\mathcal{V}}$	$P \mathrel{-\!\!*}_m Q$	iff	for all y, z s.t. $z \in x \oplus_m y$, $y \models_{\mathcal{V}} P$ implies $z \models_{\mathcal{V}} Q$

Analogous to the case in standard BI, we say $P \models Q$ iff, for all models $(\mathcal{X}, \mathcal{V})$, for any state $x \in \mathcal{X}$, $x \models_{\mathcal{V}} P$ implies $x \models_{\mathcal{V}} Q$. We prove that the proof system for M-BI is sound and complete with respect to its semantics using the duality-theoretic framework proposed by Docherty [2019].

Theorem 3.4.3. *Let P and Q be any two M-BI formulas. Then $P \models Q$ iff $P \vdash Q$.*

We show the proof in appendix B.3

3.4.3 A M-BI Model for Independence and NA

We now combine $\mathcal{X}_{\mathrm{PNA}}$ with the BI frame $\mathcal{X}_{\mathbb{D}}$ to construct a M-BI frame. Since the separating conjunction in $\mathcal{X}_{\mathbb{D}}$ captures independence, and separating conjunction in $\mathcal{X}_{\mathrm{PNA}}$ captures negative association, we can expect to use M-BI formulas interpreted on the combined model to express both probabilistic independence and negative association.

We combine $\mathcal{X}_{\mathbb{D}}$ and $\mathcal{X}_{\mathrm{PNA}}$ into a $\widehat{\mathbf{2}}$-BI model where $\widehat{\mathbf{2}}$ denotes the set $\{0, 1\}$ ordered as $0 \leq 1$, the index 0 is associated with the independent combination $\otimes_{\mathbb{D}}$ and the index 1 is associated with the NA combination $\oplus$.

Theorem 3.4.4. *The structure* $\mathcal{X}_{NA} = (X_{\mathbb{D}}, \sqsubseteq_{\mathbb{D}}, (\otimes_{\mathbb{D}}, \oplus), (E_{\mathbb{D}}, E_{\mathbb{D}}))$ *forms a* $\widehat{\mathbf{2}}$*-BI frame.*

Proving $\mathcal{X}_{NA}$ forms a $\widehat{\mathbf{2}}$-BI boils down to showing that for any $\mu_1, \mu_2 \in X$,

$$\mu_1 \otimes_{\mathbb{D}} \mu_2 \subseteq \mu_1 \oplus \mu_2.$$

The inclusion is implied by the following theorem generalizing the independence closure for NA (theorem 3.2.2). Its proof, however, is more involved because PNA is more expressive and is closed under coarsening.

Theorem 3.4.5 (Independence implies PNA)**.** *Let* $S, T \subseteq \mathbf{Var}$ *be two disjoint sets of variables. Suppose* $\mu_1 \in \mathcal{D}(\mathbf{Mem}[S])$*,* $\mu_2 \in \mathcal{D}(\mathbf{Mem}[T])$*. If* μ_1 *satisfies* $\mathcal{S}$*-PNA and* μ_2 *satisfies* $\mathcal{T}$*-PNA, then any* $\mu \in \mu_S \otimes_{\mathbb{D}} \mu_T$ *satisfies* $(\mathcal{S} \cup \mathcal{T})$*-PNA.*

The proof is based on the observation that: for any coarsening $\mathcal{R}$ of $\mathcal{S} \cup \mathcal{T}$, any block p in $\mathcal{R}$ *is the union of* some blocks from $\mathcal{S}$ and some blocks from $\mathcal{T}$. Intuitively, by the independence closure for NA, for any block p in $\mathcal{R}$, the blocks from $\mathcal{S}$ and $\mathcal{T}$ that are in p are negatively associated with the rest of the blocks in $\mathcal{S}$ and $\mathcal{T}$. Because any other block in $\mathcal{R}$ is formed by merging some remaining blocks in $\mathcal{S}$ and some remaining blocks in $\mathcal{T}$, the block p is also negatively associated with any other block in $\mathcal{R}$. Formally, we establish that proof by induction on the number of blocks in the coarsening $\mathcal{R}$ (see appendix B.4.1).

Because $\mathcal{X}_{NA}$ has the same carrier set as $\mathcal{X}_{\mathbb{D}}$, we can combine $\mathcal{X}_{NA}$ with the persistent valuation $\mathcal{V}^* : \mathcal{AP} \to X_{\mathbb{D}}$ (Definition 2.3.5) to form a $\widehat{\mathbf{2}}$-BI model $(\mathcal{X}_{NA}, \mathcal{V}^*)$. In the remaining of this chapter, we take $\widehat{\mathbf{2}}$-BI formulas interpreted in this model as our assertion logic.

3.5 Logic of Independence and Negative Association

3.5.1 Assertion Logic

When designing a separation logic for reasoning about independence and negative association, we also want the assertion logic to satisfy *restriction* (lemma 2.3.7) so that, to check whether a distribution satisfies φ, it suffices to check whether the distribution's projection on $\mathrm{FV}(\varphi)$ satisfies φ. Previously, to prove the soundness of the program logic in section 2.3.3, we show that all BI formulas satisfy the restriction property when interpreted on $\mathcal{X}_{\mathbb{D}}$. Here, not all $\widehat{\mathbf{2}}$-BI formulas satisfy the restriction property when interpreted on $\mathcal{X}_{NA}$; we identify a subset MBI_+ that satisfies restriction.

Definition 3.5.1. We define MBI_+ as

$$\mathrm{MBI}_+ \ni P, Q ::= p \in \mathcal{AP} \mid \top \mid \bot \mid P \wedge Q \mid P \vee Q \mid P \to Q \mid P * Q \mid P \mathrel{-\!\!*} Q \mid P \circledast Q$$

where $\mathcal{AP}$ is defined as in 2.3.

MBI_+ omits multiplicative identities I_m because on $(\mathcal{X}_{NA}, \mathcal{V}^*)$ they are all equivalent to $\top$. The only limitation is that MBI_+ excludes the use of $\mathrel{-\!\!\circledast}$.

Theorem 3.5.1 (Restriction)**.** *For any distribution $\mu \in X_{\mathbb{D}}$, for any φ be an MBI_+ formula interpreted on $(\mathcal{X}_{NA}, \mathcal{V}^*)$, and any valuation $\mathcal{V}$,*

$$\mu \models_{\mathcal{V}} \varphi \Leftrightarrow \pi_{FV(\varphi)}\mu \models_{\mathcal{V}} \varphi.$$

We defer its proof to appendix B.4.3. Indeed, we can exhibit a counterexample showing that $\mathrel{-\!\!\circledast}$ does not satisfy restriction.

Theorem 3.5.2. *There exists* $\mu \in \mathcal{D}(\mathbf{Mem}[S])$ *and formula* φ *such that* $\mu \models \varphi$ *but* $\pi_{FV(\varphi)} \not\models \varphi$.

We also defer its proof to appendix B.4.3. Below, we consider MBI_+ formula on the $(\mathcal{X}_{NA}, \mathcal{V}^*)$ model as the assertion logic. In this assertion logic, all the axioms for the Independence BI model section 2.3.2 and the NA BI model section 3.3.2 still hold. Also, because $(\mathcal{X}_{NA}, \mathcal{V}^*)$ is a conservative extension of $(\mathcal{X}_{\mathrm{PNA}}, \mathcal{V}^*)$, the theorem theorem 3.3.5 that says NA is captured by separating conjunction $\circledast$ also still holds. We also have some new axioms for the negative association conjunction, which involves two new distributions introduced below.

Definition 3.5.2 (One-Hot Vectors)**.** Let $\mathsf{oh}(n)$ denote the set of one-hot vectors of length n, where a one-hot vector $[\ldots, 1, \ldots]$ has exactly one entry set to 1 and all other entries set to 0. We abbreviate $\mathbf{Unif}_{\mathsf{oh}(n)}$ as $\mathbf{OH}_n$.

To describe the next distribution, we generalize the function $\mathbf{Unif}_{(-)}$ so that it can also be used to describe uniform distributions over *multi-sets*. A multi-set is an unordered collection of items that allow an item to occur more than once. When A is an multi-set, the distribution $\mathbf{Unif}_A$ assigns the outcome x with weight

$$\mathbf{Unif}_A(a) = \frac{\text{Multiplicity of } x}{\sum_{y \in A} \text{Multiplicity of } y}.$$

It is clear that, when A is simply a set, this definition agrees with our definition of uniform distribution over a set in fig. 2.6.

Definition 3.5.3 (Permutations)**.** Given a finite multi-set of A, a *permutation of* A is a bijective function $\alpha : A \to A$. We let $\mathsf{permutation}(A)$ be the multi-set of A's permutations. When A has duplicates, we distinguish them using addi-

tional labels; so there are always $|A|!$ elements in $\mathsf{permutation}(A)$. We abbreviate $\mathbf{Unif}_{\mathsf{permutation}(A)}$ as Permu_A.

Then, we have the following axioms that introduce formulas that assert negative association among variables.

Lemma 3.5.3. *Let x_γ be variables. The following axioms are valid in $(\mathcal{X}_{NA}, \mathcal{V}^*)$.*

$$\models \mathsf{OH}_N\langle [x_0, \ldots, x_{N-1}]\rangle \to \bigoplus_{\gamma=0}^{N} \mathsf{Own}(x_\gamma) \qquad \text{(OH-PNA)}$$

$$\models \mathsf{Permu}_A\langle [x_0, \ldots, x_{N-1}]\rangle \to \bigoplus_{\gamma=0}^{N} \mathsf{Own}(x_\gamma) \qquad \text{(Perm-PNA)}$$

The two axioms follow from Theorem 3.2.2, which shows that random variables in one-hot distributions and permutation distributions are NA, and Theorem 3.3.5, which shows that $\circledast$ captures the NA of random variables. We can also encode the monotone map closure in Theorem 3.2.3 as an axiom in the logic.

Lemma 3.5.4 (Binary Monotone Map)**.** *The following is valid in $(\mathcal{X}_{NA}, \mathcal{V}^*)$.*

$$\models (\varphi \circledast \eta \wedge [y = f(FV(\varphi))]) \to \mathsf{Own}(y) \circledast \eta \qquad \textit{where } f \textit{ is monotone}$$

(Binary-Mono-Map)

Proof. For any $\mu \models \varphi \circledast \eta \wedge [y = f(\mathrm{FV}(\varphi))]$, there exists μ_1, μ_2, μ' such that $\mu \sqsupseteq \mu' \in \mu_1 \oplus \mu_2$, $\mu_1 \models \varphi$ and $\mu_2 \models \eta$; furthermore, for any m such that $\mu(m) > 0$, $[\![y]\!](m) = [\![f(X)]\!](m)$.

Let $S = \mathbf{dom}\mu_2$. and let μ'' denote $\pi_{S\cup\{y\}}\mu$. We want to show that $\mu'' \in (\pi_y\mu) \oplus \mu_2$. For any partition $\{S_1, \ldots, S_k\}$ of S, for any family of non-negative all

monotone or all antitone functions $f_0, f_1, \ldots, f_k$,

$$
\begin{aligned}
&\mathbb{E}_{m\in\mu''}\left[f_0(\pi_y m)\cdot\prod_{i\in[k]} f_i(\pi_{S_i} m)\right] \\
&= \mathbb{E}_{m\in\mu}\left[f_0(\pi_y m)\cdot\prod_{i\in[k]} f_i(\pi_{S_i} m)\right] && (\mu'' \text{ is a marginalization of } \mu) \\
&= \mathbb{E}_{m\in\mu}\left[f_0(f(\pi_X m))\cdot\prod_{i\in[k]} f_i(\pi_{S_i} m)\right] && (\text{Because } [\![y]\!](m) = [\![f(X)]\!](m)) \\
&\le \mathbb{E}_{m\in\mu}[f_0(f(\pi_X m))]\cdot\prod_{i\in[k]} \mathbb{E}_{m\in\mu''}\left[f_i(\pi_{S_i} m)\right] && (\text{Because } \mu \sqsupseteq \mu' \in \mu_1 \oplus \mu_2) \\
&\le \mathbb{E}_{m\in\mu}[f_0(y)]\cdot\prod_{i\in[k]} \mathbb{E}_{m\in\mu''}\left[f_i(\pi_{S_i} m)\right] && (\text{Because } [\![y]\!](m) = [\![f(X)]\!](m)) \\
&\le \mathbb{E}_{m\in\mu''}[f_0(y)]\cdot\prod_{i\in[k]} \mathbb{E}_{m\in\mu''}[f_i(\pi_m S_i)] && (\mu'' \text{ is a marginalization of } \mu)
\end{aligned}
$$

Also, because $\mu'' = \pi_{S\cup\{y\}}\mu$, we have $\pi_y\mu'' = \pi_y\mu$ and $\pi_S\mu'' = \pi_S\mu = \mu_2$. Thus, $\mu'' \in \pi_y\mu \oplus \mu_2$. Because $\pi_y\mu \models \mathsf{Own}(y)$, we have $\mu'' \models \mathsf{Own}(y) \circledast \eta$. By persistence, $\mu' \models \mathsf{Own}(y) \circledast \eta$. □

(Mono-Map) will play an important role in reasoning about negative association arising in probabilistic programs. Furthermore, we can prove an N-nary version of the monotone map axiom.

Lemma 3.5.5 (N-NARY MONOTONE MAP)**.** *Let x, $x_{\gamma,\alpha}$ and y_γ be program variables. Let K_γ be natural numbers. The following is valid in $(\mathcal{X}_{NA}, \mathcal{V}^*)$.*

$$
\models \bigcircledast_{\gamma=0}^{N}\left(\bigwedge_{\alpha=0}^{K_\gamma}\mathsf{Own}(x_{\gamma,\alpha})\right) \wedge \bigwedge_{\gamma=0}^{N}\left[y_\gamma = f_\gamma\left(x_{\gamma,0},\ldots,x_{\gamma,K_\gamma}\right)\right] \to \bigcircledast_{\gamma=0}^{N}\mathsf{Own}(y_\gamma)
$$

when $f_1, \ldots, f_N$ all monotone or all antitone (Mono-Map)

We defer the proof to appendix B.4.2.

We also have an axiom particular to permutation distributions. When we establish NA from permutation distributions, it is preserved under not only monotone/antitone maps but also any element-wise homogeneous maps. The reason is that fixing a multi-set and a permutation, permuting first, and then applying the same map on each element is equivalent to applying the map on each element and then permuting. So applying homogeneous maps on a permutation distribution gives another permutation distribution. We capture this property in the following axiom.

Lemma 3.5.6 (Permutation Map)**.** *Let x_γ be variables, and $f(A)$ be $\{f(a) \mid a \in A\}$. The following axiom is valid in $(\mathcal{X}_{NA}, \mathcal{V}^*)$.*

$$\models \mathbf{Permu}_A\langle[x_1, \ldots, x_N]\rangle \wedge [y = [f(x_1), \ldots, f(x_N)]] \to \mathbf{Permu}_{f(A)}\langle y\rangle$$

(Perm-Map)

The proof is straightforward by unfolding the definitions, so we omit it here.

3.5.2 Program Logic

We now build upon the assertion logic and develop a program logic LINA for reasoning about independence and negative association in probabilistic programs. Judgements in LINA have the form $\{P\}\ c\ \{Q\}$, where $c \in C$ is a probabilistic program introduced in fig. 2.8, and bunched formulas P, Q are restricted assertions in MBI_+.

Definition 3.5.4 (Validity)**.** A LINA judgment is *valid*, written $\models \{P\}\ c\ \{Q\}$, if for all $\mu \in \mathbf{Mem}[\mathbf{Var}]$ such that $\mu \models P$, we have $[\![c]\!](\mu) \models Q$.

Next, we present the proof system of LINA. Since our assertions are a conservative extension of assertions from probabilistic separation logic, all the rules from Figure 2.8 carry over unchanged. We have one new program rule NEGFRAME, which acts as the frame rule for the negative association separating conjunction $\circledast$, and one new structural rule RCASE, which does case analysis where each case only has some probability of occurring.

$$\textsc{NegFrame}\ \frac{\models \varphi \to \mathsf{Own}(X) \qquad \mathrm{FV}(\eta) \cap \mathrm{MV}(c) = \emptyset \qquad X \cap \mathrm{MV}(c) = \emptyset \qquad \vdash \{\varphi\}\ c\ \{[y = f(X)]\} \qquad f \text{ is a monotone function}}{\vdash \{\varphi \circledast \eta\}\ c\ \{\mathsf{Own}(y) \circledast \eta\}}$$

$$\textsc{RCase}\ \frac{\eta \in \mathrm{CC} \qquad \forall \alpha \in S.\ \vdash \{\varphi * \eta(\alpha)\}\ c\ \{\psi\} \qquad \models_{\mathbf{Mem}} \eta \to \bigvee_{\alpha \in S} \eta(\alpha) \qquad \psi \in CM}{\vdash \{\varphi * \eta\}\ c\ \{\psi\}}$$

$$\textsc{ProbBound}\ \frac{\vdash \{ev_1 = 1\}\ c\ \{\Pr[ev_2] \le \delta\}}{\vdash \{\Pr[ev_1] \ge 1 - \epsilon\}\ c\ \{\Pr[ev_2] \le \delta + \epsilon\}}$$

Figure 3.2: New LINA rules.

Informally, the NEGFRAME rule says that if a set of variables X is negatively associated with another set of variables Y that satisfy η in a program state, and the program c performs a monotone operation f on X and stores the result in a variable y, then in the resulting program state, y and the untouched variables Y will also be negatively associated, and Y will still satisfy η. Like the FRAME rule for independence $*$, the NEGFRAME rule uses syntactic restrictions to control which variables the program may read and write. The three sets of variables $\mathrm{RV}(c), \mathrm{WV}(c), \mathrm{MV}(c)$ are the ones defined in definition 2.3.8. Roughly, the side conditions guarantee the program c does not read from or modify Y, the set of variables satisfying η; they in addition guarantee that X, the domain of the monotone map will not be modified by c, and y, the codomain of the monotone

map does not belong to Y.

For RCASE, we write $\models_{\mathbf{Mem}} P$ iff $\forall m \in \cup_{S \subseteq \mathbf{Var}} \mathbf{Mem}[S], \delta(m) \models P$. We say a formula η is *closed under conditioning* (CC) iff for any μ, $\mu \models \eta$ implies that for any event S, we have $\mu \mid S \models \eta$. And as in RCOND, a formula η in CM means that η is closed under the mixture. At a high-level, RCASE allows us to first condition the input distribution on one specific case, reason about the post-condition with the conditioned input distribution, and then use the post-condition – we implicitly combined post-conditions from different cases by requiring the post-condition to be closed under the mixture.

Last, we present the rule PROBBOUND to facilitate bounding probabilities. It says that if the pre-condition $ev_1 = 1$ guarantees that event ev_2 happens with at most δ probability after command c, then in general, event ev_2 happens with at most probability $\delta + \epsilon$ after c, where ϵ upper bounds the probability that ev_1 is not true in the pre-condition. The validity of this rule uses the law of total probability, which says for any two events ev_1 and ev_2,

$$\begin{aligned}\Pr(ev_1) &= \Pr(ev_1 \mid ev_2) \cdot \Pr(ev_2) + \Pr(ev_1 \mid \neg ev_2) \cdot \Pr(\neg ev_2) \\ &\leq \Pr(ev_1 \mid ev_2) + \Pr(\neg ev_2).\end{aligned}$$

As expected, the LINA proof system is sound.

Theorem 3.5.7. *(Soundness of LINA) If* $\vdash \{\varphi\}\ c\ \{\psi\}$ *is derivable, then it is valid:* $\models \{\varphi\}\ c\ \{\psi\}$.

Proof. We prove the soundness of each new rule in LINA.

NEGFRAME We show that NEGFRAME follows from Binary-Mono-Map and existing program rules. By CONST, the side conditions $\mathrm{FV}(\eta) \cap \mathrm{MV}(c) = \emptyset$

and $X \cap \mathrm{MV}(c)$ imply that $\{\mathsf{Own}(X) \circledast \eta\}\ c\ \{\mathsf{Own}(X) \circledast \eta\}$. Because $\models \varphi \to \mathsf{Own}(X)$, by $*$-CONJ, it must $\models \varphi \circledast \eta \to \mathsf{Own}(X) \circledast \eta$. Thus, by WEAK,

$$\{\varphi \circledast \eta\}\ c\ \{\mathsf{Own}(X) \circledast \eta\}.$$

Also by WEAK, the premise $\{\varphi\}\ c\ \{[y = f(X)]\}$ implies $\{\varphi \circledast \eta\}\ c\ \{[y = f(X)]\}$. Thus, by CONJ,

$$\{\varphi \circledast \eta\}\ c\ \{\mathsf{Own}(X) \circledast \eta \wedge [y = f(X)]\}.$$

By (Binary-Mono-Map), $\mathsf{Own}(X) \circledast \eta \wedge [y = f(X)]$ implies $\mathsf{Own}(y) \circledast \eta$. Thus, by WEAK again,

$$\{\varphi \circledast \eta\}\ c\ \{\mathsf{Own}(y) \circledast \eta\}.$$

RCASE For any $\mu \models \varphi * \eta$, there exists μ_1, μ_2, μ' such that $\mu \supseteq \mu' \in \mu_1 \circ \mu_2$, $\mu_1 \models \varphi$, and $\mu_2 \models \eta$. The formula η being CC means for any m in the support of μ_2, $\delta_m \models \eta$ as well. Then, with the side condition $\models_{\mathbf{Mem}} \eta \to \bigvee_{\alpha \in S} \eta_\alpha$, we have

$$\delta_m \models \bigvee_{\alpha \in S} \eta(\alpha).$$

Combining the side-condition that $\{\eta(\alpha)\}\ c\ \{\psi\}$ for all α with CASE, we get $\{\bigvee_{\alpha \in S} \eta(\alpha)\}\ c\ \{\psi\}$. Thus, for any $m \in \mathsf{supp}(\mu)$ we have $[\![c]\!](\delta_m) \models \psi$. According to the semantics, $[\![c]\!](\mu)$ is a convex combination of $[\![c]\!](\delta_m)$ for different m, and thus $[\![c]\!]\mu \models \psi$.

PROBBOUND Denote the function $\lambda x.1 - ev_1(x)$ as $\neg ev_1$.

For any program state $\mu \models \Pr[ev_1] \geq 1 - \epsilon$, let $\rho = \mu(ev_1)$, it must $\rho \leq 1 - \epsilon$. Let $\mu_{ev_1} = [\![c]\!](\mu \mid ev_1)$ and let $\mu_{\neg ev_1} = [\![c]\!](\mu \mid \neg ev_1)$. By induction on the denotational semantics of the commands, we can prove that $[\![c]\!](\mu) = \mu_{ev_1} \circ_\rho \mu_{\neg ev_1}$.

Also, by construction, $[\![ev_1]\!](\mu \mid ev_1) = 1$, so $\mu \mid ev_1 \models ev_1$. By the rule's assumption and inductive hypothesis, we have $\models \{ev_1\}\ c\ \{\Pr[ev_2] \leq \delta\}$, which implies

$$\mu_{ev_1} \models \Pr[ev_2] \leq \delta.$$

Thus, we have $\mu_{ev_1}(ev_2) \leq \delta$.

Then, by definition and the law of total probability,

$$\begin{aligned}
[\![c]\!](\mu)(ev_2(x)) &= (\mu_{ev_1} \circ_\rho \mu_{\neg ev_1})(ev_2) \\
&\leq \rho \cdot \mu_{ev_1}(ev_2) + (1 - \rho) \\
&\leq \rho \cdot \delta + (1 - \rho) \\
&\leq \rho \cdot \delta + \epsilon \\
&\leq \delta + \epsilon
\end{aligned}$$

That ensures $[\![c]\!](\mu) \models \Pr[ev_2] \leq \delta + \epsilon$. □

3.6 Examples

Now that we have introduced LINA, we present a series of formalized case studies. Our examples are extracted from various algorithms using hashing and balls-into-bins processes.

3.6.1 Probability-related Axioms for Examples

Our examples will use a handful of standard facts about probability distributions, encoded as axioms in the assertion logic. For completeness, we list the

axioms used below. We also observe the following conventions throughout the examples: logical variables are denoted by Greek ($\alpha, \beta, \gamma, \dots$) and capital Roman letters ($M, N, K, \dots$). Program variables start with lower-case Roman letters ($x, y, z, \dots$).

The most important axiom is the one encoding Chernoff Bound: in each of our examples, we establish negative dependence of a sequence of random variables $\{X_i\}_i$ and apply the Chernoff bound to derive a tail bound. In our assertion logic, the Chernoff bound can be encoded as the following axiom schema:

Theorem 3.6.1 (Chernoff bound, axiom)**.** *Let $\{x_\alpha\}$ be a family of variables indexed by α, where each variable is bounded in $[0,1]$ and is a monotone function of its program variables. Then for any $\beta \in (0,1]$, the following axiom schema is sound in our model:*

$$\models \circledast_{\alpha=0}^{N} \mathsf{Own}(x_\alpha) \to \Pr\left[\left|\sum_{\alpha=0}^{N} x_\alpha - \mathbb{E}\left[\sum_{\alpha=0}^{N} x_\alpha\right]\right| \geq \beta\right] \leq F(\beta, n) \qquad \text{(NA-Chernoff-1)}$$

$$\models \circledast_{\alpha=0}^{N} \mathsf{Own}(x_\alpha) \to \Pr\left[\left|\sum_{\alpha=0}^{N} x_\alpha - \mathbb{E}\left[\sum_{\alpha=0}^{N} x_\alpha\right]\right| \geq T(\beta, n)\right] \leq \beta \qquad \text{(NA-Chernoff-2)}$$

For the other axioms, we present the axioms in binary form for simplicity, though most extend directly to big operations.

- Linearity of expectation. Let e, f be *bounded* expressions.

$$\models [\mathbb{E}[\alpha \cdot e + \beta \cdot f] = \alpha \cdot \mathbb{E}[e] + \beta \cdot \mathbb{E}[f]] \qquad \text{(LinExp)}$$

- Union bound. Let $ev_1, ev_2 \in \mathcal{EV}$,

$$\models \Pr[ev_1 \vee ev_2] \leq \Pr[ev_1] + \Pr[ev_2] \qquad \text{(UnionBd)}$$

- Permutation marginal. Let x be an array variable, and let S be a finite set.

$$\models \mathbf{Permu}_S\langle x \rangle \to \mathbf{Unif}_S\langle x[\alpha] \rangle \qquad \text{(PermMarg)}$$

- Expectation Indicator. Let e be a $0/1$ valued expression,

$$\models [\mathbb{E}[e] = \Pr[e = 1]] \tag{ExpectInd}$$

- Bernoulli variables probabilities. Let e be an expression,

$$\models \mathbf{Bern}_p\langle e\rangle \to \Pr[e = 1] = p \tag{BernProb}$$

- Probability of uniform. Let S be a finite set.

$$\models [\Pr[\mathbf{Unif}_S\langle x\rangle = \alpha] = 1/|S|] \tag{ProbUnif}$$

- Bijection uniform. Let S be a finite set, and let $f : S \to S$ be a bijection.

$$\models \mathbf{Unif}_S\langle x\rangle \to \mathbf{Unif}_S\langle f(x)\rangle \tag{BijectUnif}$$

- One-hot marginal. Let x be an array variable.

$$\models \mathbf{OH}_S\langle x\rangle \to \mathbf{Unif}_S\langle x[\alpha]\rangle \tag{OHMarg}$$

- Independent product one-hot.

$$\models \mathbf{OH}_{[M]}\langle x\rangle * \mathbf{OH}_{[N]}\langle y\rangle \to \mathbf{OH}_{[M]\times[N]}\left\langle x^\top \cdot y\right\rangle \tag{IndProdOH}$$

- Independent map. Let x be an array variable of length N.

$$\models \overset{N}{\underset{\alpha=0}{\circledast}} x[\alpha] \multimap \overset{N}{\underset{\alpha=0}{\circledast}} f(x[\alpha]) \tag{IndMap}$$

- Deterministic independent. Let x be a variable.

$$\models \mathbf{Detm}\langle x\rangle \to x \multimap * e \multimap \tag{DetInd}$$

- Events happen only if they have probability one. Let $ev \in \mathcal{EV}$,

$$\models ev = 1 \to \Pr(ev) = 1 \tag{ProbOne}$$

- Uniform sampling from a population. We represent a population as a bit-vector, where each entry is an individual and 1 indicates they have some feature and 0 indicates not. Then, if we uniformly sample from the population, the probability of getting one is equal to the population-level ratio of ones, regardless of how they are distributed in the population. Let $N \geq J$ be constants or logical variables, b be an array variable of length N, and x, *hit* be variables:

$$\models \left(\left(\mathbf{bv}(b, J, N) * \mathbf{Unif}_{[N]}\langle x\rangle\right) \wedge [hit = b[x]]\right) \rightarrow \mathbf{Bern}_{hit}\left\langle \frac{J}{N} \right\rangle * \left(\sum_{\beta=0}^{N} b[\beta] = J\right). \tag{UniformSamp}$$

- Independent product probabilities. Let $ev_1, ev_2 \in \mathcal{EV}$, J, K be two real numbers,

$$\models \Pr[ev_1] \leq J * \Pr[ev_2] \leq K \rightarrow \Pr[ev_1 \wedge ev_2] \leq J \cdot K. \tag{IndepProb}$$

- Equal probabilities. Let b_1, b_2 be two boolean expressions. Recall that $b_1, b_2 \in \mathcal{EV}$ too.

$$\models [b_1 = b_2] \rightarrow \Pr[b_1] = \Pr[b_2] \tag{EqualProb}$$

3.6.2 Bloom filter, High-level

We demonstrate how NA and its closure properties can be used to analyze Bloom filters. A Bloom filter is a space-efficient probabilistic data structure for storing a set of items from a universe U. An N-bit Bloom filter consists of a length-N array *bloom* holding zero-one entries. We assume there is a family S of hash functions mapping U to $\{0, \ldots, N-1\}$ and a distribution $\mathcal{H}$ over S such that for any $x \in U$ and any bucket k, $\Pr_{f \sim \mathcal{H}}(f(x) = k) = 1/N$. Let $l_1, \ldots, l_H$ be a

```
BLOOM :
  bloom ← zero(N);
  m ← 0;
  while m < M do
    h ← 0
    while h < H do
      bin ←$ OH[N];
      upd ← bloom || bin;
      bloom ← upd;

    h ← h + 1;
  m ← m + 1;
```

(a) Higher-level version

```
BLOOMARRAY :
  bloom ← zero(N);
  m ← 0;
  while m < M do
    h ← 0
    while h < H do
      bin ←$ OH[N];
      n ← 0;
      while n < N do
        upd ← bloom[n] || bin[n];
        bloom[n] ← upd;
        n ← n + 1
      h ← h + 1;
    m ← m + 1
```

(b) Array version

Figure 3.3: Bloom filter examples

collection of hash functions drawn from $\mathcal{H}$. We assume the hash functions are independent, meaning the collection of variables $\{l_i(x) \mid x \in U, i \in \{1, \ldots, H\}\}$ are independent. To add an item $x \in U$ to the filter, we compute $l_1(x), \ldots, l_H(x)$ to get H positions in the bit array $bloom$ and then set the bits at each of these positions to 1. To check if an item y is in the filter, we check whether the bits at positions $l_1(y), \ldots, l_H(y)$ in $bloom$ are all 1. If they are, the item is said to be in the filter, but if any is 0, then the item is not in the filter. This membership test may suffer from *false positives*, i.e., it may show that an item y is in the filter even when y was never added to the filter. This can happen because, with hash collisions, other items added to the Bloom filter could set all the bits at locations $l_1(y), \ldots, l_H(y)$ to 1. A basic quantity of interest is the *false positive rate*: the probability that a Bloom filter reports a false positive.

We model the process of adding M distinct items into a Bloom filter as the program BLOOM in fig. 3.3a. Because the M items are distinct, we model the

hash functions as if they independently, randomly sample hash values for each item as they are added, a standard model used in the analysis of hashing data structures [Mitzenmacher and Upfal, 2005]. That is, we encode the hashing step as sampling a one-hot vector from the distribution $\mathbf{OH}_{[N]}$ and storing it in the variable *bin*, where the hot bit of the vector *bin* represents the selected position. To set the corresponding position in the filter to 1, we update $bloom$, which is set to be an all-zero vector at the beginning of the program, to be $bloom \,||\, bin$, the bitwise-or of the current array and the sampled one-hot array.

Our goal is to bound an N-bit Bloom filter's false positive rate after M distinct items are added. We split the analysis of the false positive rate into two steps. First, we will analyze BLOOM and prove that the entries in $bloom$ are negatively associated at the end of the process. By (NA-Chernoff-2), NA between the entries of $bloom$ gives a tail bound of the fraction of bits in $bloom$ that are set to 1. Second, we analyze a program that checks the membership of a new item in a given Bloom filter, presented as CHECKMEM in fig. 3.4, and bound the probability that the H hashed values of the new item are all already in the Bloom filter. Last, we combine them into one proof that bounds the false positive rate of a Bloom filter with M elements.

Proving NA of BLOOM Recall that the code models inserting M distinct elements into a Bloom filter backed by an array $bloom$ of length N, where each element is hashed by H functions, each producing an element of $[N]$ uniformly at random.

We refer to the outer loop as *outer*, and the inner loop as *inner*. For both the outer and the inner loop, we apply the rule LOOP with the loop invari-

ant: $\circledast_{\beta=0}^{N} \mathsf{Own}(bloom[\beta])$. We consider the inner loop first. We show that the invariant is preserved by the body of *inner*. After the sampling command $bin \xleftarrow{\$} \mathbf{OH}_{[N]}$, SAMP gives:

$$\left(\circledast_{\beta=0}^{N} \mathsf{Own}(bloom[\beta])\right) * \mathbf{OH}_{[N]}\langle bin\rangle$$

By negative association of the one-hot distribution (OH-PNA), we get

$$\left(\circledast_{\beta=0}^{N} \mathsf{Own}(bloom[\beta])\right) * \left(\circledast_{\gamma=0}^{N} bin[\gamma]\right)$$

which implies

$$\left(\circledast_{\beta=0}^{N} \mathsf{Own}(bloom[\beta])\right) \circledast \left(\circledast_{\gamma=0}^{N} bin[\gamma]\right)$$

using WEAK. By rearranging terms, this is equivalent to

$$\circledast_{\beta=0}^{N} \mathsf{Own}(bloom[\beta]) \circledast \mathsf{Own}(bin[\beta]).$$

After the assignment to upd, we have:

$$\left(\circledast_{\beta=0}^{N} \mathsf{Own}(bloom[\beta]) \circledast \mathsf{Own}(bin[\beta])\right) \wedge [upd = bloom \,||\, bin].$$

Because $||$ is monotone, applying the monotone mapping axiom (Mono-Map) gives us:

$$\circledast_{\beta=0}^{N} \mathsf{Own}(upd[\beta]).$$

Using the assignment rule (RASSN) on the assignment to *bloom* shows that the loop invariant is preserved by the inner loop. Thus, LOOP gives:

$$\left\{\circledast_{\beta=0}^{N} \mathsf{Own}(bloom[\beta])\right\}\ inner\ \left\{\circledast_{\beta=0}^{N} \mathsf{Own}(bloom[\beta])\right\}$$

Next, we turn to the outer loop. The argument showing that the invariant is preserved by the outer loop follows from a straightforward argument, since the

outer loop only modifies *bloom* through the inner loop, so LOOP gives:

$$\left\{\circledast_{\beta=0}^{N} \mathsf{Own}(bloom[\beta])\right\}\ outer\ \left\{\circledast_{\beta=0}^{N} \mathsf{Own}(bloom[\beta])\right\}$$

Then, we have:

$$\left\{\top\right\}\ \textsc{Bloom}\ \left\{\circledast_{\beta=0}^{N} \mathsf{Own}(bloom[\beta])\right\}$$

because initializing *bloom* to the all-zeros vector, a deterministic value, establishes the loop invariant. This judgment shows that the *bloom* vector satisfies NA at the end of the program.

We now apply the Chernoff bound to the NA variables (NA-Chernoff-2) to prove that, with high probability, the number of occupied bins in BLOOM is near its mean with high probability:

$$\left\{\top\right\}\textsc{Bloom}\left\{\Pr\left[\left|\sum_{\beta=0}^{N} bloom[\beta] - \mathbb{E}\left[\sum_{\beta=0}^{N} bloom[\beta]\right]\right| \geq T(\delta, N)\right] \leq \delta\right\}.$$

This concentration bound implies that a tail bound, which says with high probability $\sum_{\beta=0}^{N} bloom[\beta]$ is upper bounded by its expected value plus $T(\delta, N)$,

$$\left\{\top\right\}\textsc{Bloom}\left\{\Pr\left[\sum_{\beta=0}^{N} bloom[\beta] < \mathbb{E}\left[\sum_{\beta=0}^{N} bloom[\beta]\right] + T(\delta, N)\right] \geq 1 - \delta\right\}. \quad (3.1)$$

Furthermore, since the first line of the program, *bloom* has been kept as a *bit-array*, i.e., all its entries are either 0 or 1. So it is easy to prove that

$$\left\{\top\right\}\ \textsc{Bloom}\ \left\{\bigwedge_{\beta=0}^{N} (bloom[\beta] = 0 \vee bloom[\beta] = 1)\right\}.$$

In the following, we will abbreviate formulas that assert b is a bit-array where exactly J of its first N entries are one, i.e.,

$$\left(\sum_{\beta=0}^{N} b[\beta] = J\right) \wedge \bigwedge_{\beta=0}^{N} (b[\beta] = 0 \vee b[\beta] = 1),$$

as $\mathbf{bv}(b, J, N)$. Similarly, we will use $\mathbf{bv}(b, < J, N)$ to abbreviate

$$\left(\sum_{\beta=0}^{N} b[\beta] < J\right) \wedge \bigwedge_{\beta=0}^{N} (b[\beta] = 0 \vee b[\beta] = 1).$$

Now we restate our goal as

$$\{\mathbf{bv}(bloom, < K, N)\}\ \textsc{CheckMem}\ \{\Pr[allhit] \leq (K/N)^H\}.$$

```
CheckMem(H, bloom) :
  h ← 0;
  allhit ← 1
  while h < H do
    bin ←$ Unif_[N];
    hit ← bloom[bin];
    allhit ← hit && allhit;
    h ← h + 1;
```

Figure 3.4: Check the membership of a new item

Bounding the false positive rate Now, we turn to verifying a bound on the false positive rate of the Bloom filter. Recall that a false positive occurs if the filter returns true when querying with an element that was not inserted. We can encode the membership check of a new element as a program CheckMem $(H, bloom)$, listed in Figure 3.4. The program hashes the new element into H uniformly random positions and checks if these positions are all set to one in the filter. If so, the Bloom filter will report that the new element is in set, even though it was never inserted — a false positive.

To verify the false positive rate, we place the program CheckMem$(H, bloom)$ immediately after Bloom, and then verify a bound on the probability that $allhit$ is 1 at the end of the combined program.

CHECKMEM first initializes h and $allhit$ deterministically to 1. Then, using RASSN and FRAME, we can show that

$$\vdash \{\top\}\ h \leftarrow 0; allhit \leftarrow 1\ \{[h = 0] * [allhit = 1]\}.$$

Using the (ProbOne) axiom and the fact that $1 \leq (K/N)^0$ for any K and N, we can show $\models ([h = 0]) * ([allhit = 1]) \rightarrow \Pr[allhit] \leq (K/N)^h$. Thus,

$$\vdash \{\top\}\ h \leftarrow 0; allhit \leftarrow 1\ \{\Pr[allhit] \leq (K/N)^h\}.$$

Because the assignments $h \leftarrow 0; allhit \leftarrow 1$ do not modify the Bloom filter array $bloom$, we can then apply FRAME to derive

$$\vdash \{\mathbf{bv}(bloom, < K, N)\}\ h \leftarrow 0; allhit \leftarrow 1\ \{\mathbf{bv}(bloom, < K, N) * \Pr[allhit] \leq (K/N)^h\}. \tag{3.2}$$

We will abbreviate $\mathbf{bv}(bloom, < K, N) * \Pr[allhit] \leq (K/N)^h \wedge h < H$ as η. Because $\sum_{\beta=0}^{N} b[\beta]$ is an integer upper bounded by N,

$$\models_{\mathbf{Mem}} \eta \rightarrow \bigvee_{0 \leq J < K} \eta_J,$$

where η_J abbreviates $J < K \wedge \left(\mathbf{bv}(bloom, J, N) * \Pr[allhit] \leq (K/N)^h\right) \wedge h \leq H$.

We will then prove that for each J, the formula η_J is a loop invariant of CHECKMEM's loop body. The loop body first uniformly samples an element from $[N]$, so by SAMP and FRAME, we have the following as the post-condition:

$$\eta_J * \mathbf{Unif}_{[N]}\langle bin \rangle. \tag{3.3}$$

Together with the axiom $\models ((P \wedge Q) * R) \rightarrow (P \wedge (Q * R))$, the post-condition 3.3 implies

$$J < K \wedge h \leq H \wedge \left(\mathbf{bv}(bloom, J, N) * \left(\Pr[allhit] \leq (K/N)^h\right) * \mathbf{Unif}_{[N]}\langle bin \rangle\right).$$

Then, hit gets assigned to $bloom[bin]$, so by RASSN and CONST, we have

$$\begin{array}{l}\{\mathbf{bv}(bloom, J, N) * \mathbf{Unif}_{[N]}\langle bin\rangle\} \\ \quad hit \leftarrow bloom[\beta] \\ \{(\mathbf{bv}(bloom, J, N) * \mathbf{Unif}_{[N]}\langle bin\rangle) \wedge [hit = bloom[bin]]\}\end{array}.$$

Since the array $bloom$ only contains zero-one entries, when the sum of its entries is J, an entry $bloom[bin]$ drawn uniformly at random has probability $\frac{J}{N}$ to be 1. If the entry is in addition chosen independently from values in $bloom$, then the bit $bloom[bin]$ is distributed independent from the distribution of $bloom$. The (UniformSamp) axiom encodes this fact:

$$\models ((\mathbf{bv}(b, J, N) * \mathbf{Unif}_{[N]}\langle x\rangle) \wedge [hit = b[x]]) \rightarrow \mathbf{Bern}_{\frac{J}{N}}\langle hit\rangle * \mathbf{bv}(b, J, N).$$

Thus, we have

$$\{\mathbf{bv}(bloom, J, N) * \mathbf{Unif}_{[N]}\langle bin\rangle\}\ hit \leftarrow bloom[\beta]\ \{\mathbf{Bern}_{\frac{J}{N}}\langle hit\rangle * \mathbf{bv}(bloom, J, N)\}.$$

Because $hit \leftarrow bloom[bin]$ does not modify $allhit$, we can apply FRAME and get

$$\begin{array}{l}\left\{\mathbf{bv}(bloom, J, N) * \mathbf{Unif}_{[N]}\langle bin\rangle * \Pr[allhit] \leq \left(\frac{K}{N}\right)^h\right\} \\ \quad hit \leftarrow bloom[\beta] \\ \left\{\mathbf{Bern}_{\frac{J}{N}}\langle hit\rangle * \mathbf{bv}(bloom, J, N) * \Pr[allhit] \leq \left(\frac{K}{N}\right)^h\right\}.\end{array}$$

Next, with the assignment $allhit \leftarrow hit \,\&\&\, allhit$, by applying the RASSN rule and the axioms (IndepProb), (EqualProb), we get:

$$\begin{array}{l}\left\{\mathbf{Bern}_{\frac{J}{N}}\langle hit\rangle * \mathbf{bv}(bloom, J, N) * \Pr[allhit] \leq \left(\frac{K}{N}\right)^h\right\} \\ \quad allhit \leftarrow hit \,\&\&\, allhit \\ \left\{\left(\Pr[allhit] \leq \frac{J}{N} \cdot \left(\frac{K}{N}\right)^h\right) * \mathbf{bv}(bloom, J, N)\right\}\end{array}$$

We can then apply the rule of constancy CONST and get

$$\left\{J < K \wedge h \leq H \wedge \left(\mathbf{bv}(bloom, J, N) * \mathbf{Unif}_{[N]}\langle bin\rangle * \Pr[allhit] \leq \left(\frac{K}{N}\right)^h\right)\right\}$$

$$hit \leftarrow bloom[\beta]; allhit \leftarrow hit \,\&\&\, allhit$$

$$\left\{J < K \wedge h \leq H \wedge \left(\left(\Pr[allhit] \leq \frac{J}{N} \cdot \left(\frac{K}{N}\right)^h\right) * \mathbf{bv}(bloom, J, N)\right)\right\}$$

When we have $J < K$, then $(K/N)^h \cdot \frac{J}{N} \leq (K/N)^{h+1}$, so the postcondition implies

$$J < K \wedge h \leq H \wedge \left(\left(\Pr[allhit] \leq \left(\frac{K}{N}\right)^{h+1}\right) * \mathbf{bv}(bloom, J, N)\right)$$

The last step in the loop body is the assignment $h \leftarrow h + 1$. By the deterministic assignment rule DASSN, we can establish the postcondition η_J afterwards:

$$J < K \wedge h \leq H\left(\mathbf{bv}(bloom, J, N) * \Pr[allhit] \leq \left(\frac{K}{N}\right)^h\right).$$

Thus, we have $\{\eta_J\}$ *loop body* $\{\eta_J\}$

By LOOP rule, we can establish $\{\eta_J\}$ *loop* $\{\eta_J \wedge h \geq H\}$. Recall η abbreviates $\mathbf{bv}(bloom, < K, N) * \Pr[allhit] \leq (K/N)^h \wedge h < H$, so the post-condition $\eta_J \wedge h \geq H$ implies $h = H$, which further implies $\Pr[allhit] \leq (K/N)^H$. We then have

$$\left\{\eta_J\right\}\ loop\ \left\{\Pr[allhit] \leq \left(\frac{K}{N}\right)^H\right\}.$$

Because $\Pr[allhit] \leq (K/N)^H$ is closed under mixtures, and η is closed under conditioning, we can then apply RCASE to prove that

$$\left\{\eta\right\}\ loop\ \left\{\Pr[allhit] \leq \left(\frac{K}{N}\right)^H\right\}. \tag{3.4}$$

Using the SEQN rule to combine the proved judgments for CHECKMEM's initialization (3.2) and loop (3.4), we derive

$$\left\{\mathbf{bv}(bloom, < K, N)\right\}\ \textsc{CheckMem}\ \left\{\Pr[allhit] \leq \left(\frac{K}{N}\right)^H\right\}$$

Then, by the PROBBOUND rule and basic axioms about probabilities, we have

$$\left\{\Pr\left[\sum_{\beta=0}^{N} bloom[\beta] < K\right] \geq 1-\delta \wedge \bigwedge_{\beta=0}^{N}(b[\beta]=0 \vee b[\beta]=1)\right\}$$

CHECKMEM

$$\{\Pr[allhit] \leq (K/N)^H + \delta\} \quad . \tag{3.5}$$

We then use SEQN to combine the proved judgements for BLOOM (3.1) and CHECKMEM (3.5) to derive that, for any δ,

$$\left\{\top\right\}\ \text{BLOOM; CHECKMEM}\ \left\{\Pr[allhit] \leq \left(\frac{\mathbb{E}\left[\sum_{\beta=0}^{N} bloom[\beta]\right] + T(\delta, N)}{N}\right)^H + \delta\right\}.$$

Since *allhit* is 1 exactly when there is a false positive, this judgment proves an upper bound on the false positive rate of the Bloom filter.[4]

3.6.3 Bloom filter, Low-level

The previous Bloom filter uses a vector operation $bloom \,||\, bin$ to transform an array of negatively associated values. We next consider a lower-level version of the previous example, BLOOMARRAY, in Figure 3.3b, where the vector operation is replaced by a loop that applies the Boolean-or.

Let *outer* and *mid* be the outer-most and second outer-most loops, and let *inner* be the inner-most loop. Again, our goal is to show that the vector *bloom* is negatively associated at the end of the program. We first prove the following

[4]The precise expected value is $N \cdot (1 - (1 - 1/N)^{M \cdot H})$, a fact that can also be shown in our logic. Roughly speaking, this fact follows because each element of *bloom* is the logical-or of $M \cdot H$ probabilistically independent bits, each 1 with probability $1/N$ and 0 otherwise. This argument does not rely on negative association.

judgment for *inner*:

$$\left\{\circledast_{\beta=0}^{N} \mathsf{Own}(bloom[\beta]) * \circledast_{\gamma=0}^{N} \mathsf{Own}(bin[\gamma])\right\}\ \mathit{inner}\ \left\{\circledast_{\beta=0}^{N} \mathsf{Own}(bloom[\beta]) \circledast \circledast_{\gamma=n}^{N} \mathsf{Own}(bin[\gamma])\right\}$$

We will apply the rule LOOP on *inner* with the following loop invariant:

$$\varphi = \circledast_{\beta=0}^{N} \mathsf{Own}(bloom[\beta]) \circledast \circledast_{\gamma=n}^{N} \mathsf{Own}(bin[\gamma])$$

To show that the loop invariant is preserved by the body, we can first show:

$$\{\mathsf{Own}(bloom[n], bin[n])\}\ upd \leftarrow bloom[n]\ ||\ bin[n]\ \{[upd = bloom[n]\ ||\ bin[n]]\}$$

using RASSN. Noting that the boolean-or operator is a monotone operation, we may apply NEGFRAME to obtain:

$$\{\mathsf{Own}(bloom[n], bin[n]) \circledast \eta\}\ upd \leftarrow bloom[n]\ ||\ bin[n]\ \{\mathsf{Own}(upd) \circledast \eta\}$$

with the framing condition

$$\eta = \left(\circledast_{\beta=0}^{n-1} \mathsf{Own}(bloom[\beta])\right) \circledast \left(\circledast_{\beta=n+1}^{N} \mathsf{Own}(bloom[\beta])\right) \circledast \left(\circledast_{\gamma=n+1}^{N} \mathsf{Own}(bin[\gamma])\right).$$

Thus, by re-associating the separating conjunction and applying DASSN for the remaining two assignments in the inner-most loop, we have:

$$\{\varphi\}\ upd \leftarrow bloom[n]\ ||\ bin[n]; bloom[n] \leftarrow upd; n \leftarrow n+1\ \{\varphi\}$$

and thus by LOOP, we have:

$$\left\{\circledast_{\beta=0}^{N} \mathsf{Own}(bloom[\beta]) \circledast \circledast_{\gamma=n}^{N} \mathsf{Own}(bin[\gamma])\right\}\ \mathit{inner}\ \left\{\circledast_{\beta=0}^{N} \mathsf{Own}(bloom[\beta]) \circledast \circledast_{\gamma=n}^{N} \mathsf{Own}(bin[\gamma])\right\}.$$

Now for loop *mid*, we establish the same loop invariant as we took before:

$$\psi = \circledast_{\beta=0}^{N} \mathsf{Own}(bloom[\beta])$$

If ψ holds at the beginning of *mid*, then invariant for the inner-most loop φ holds after assigning 0 to n and sampling *bin*, since *bin* is independent of ψ and *bin* is distributed as $\mathbf{OH}_n$, which implies entries in *bin* are negatively associated (OH-PNA). Furthermore, φ implies ψ at the exit of *inner*, by dropping the conjunct describing *bin*. Thus, ψ is a valid invariant for *mid*, and the rest of the proof proceeds unchanged.

3.6.4 Permutation Hashing

PERMHASH :
$g \xleftarrow{\$} \mathsf{Permu}_{[B \cdot K]}$;
$n \leftarrow 0$;
$ct \leftarrow 0$;
while $n < N$ **do**
$bin[n] \leftarrow mod(g[n], B)$;
$hitZ[n] \leftarrow [bin[n] = Z]$;
$ct \leftarrow ct + hitZ[n]$;
$n \leftarrow n + 1$

Figure 3.5: Permutation hashing

Our second example considers a scheme for hashing using a random permutation. Consider the program in Figure 3.5, from an algorithm for fast set intersection [Ding and König, 2011]. Letting B be the number of bins, and the data universe be $[B \cdot K] = \{1, \ldots, B \cdot K\}$ where $B \cdot K \geq N$, we first draw a uniformly random permutation g of the data universe. Then, we hash the numbers $n \in [N]$ into $bin[n]$ by applying the hash function g and then taking the result modulo B. Then, we record whether the item landed in a specific bucket Z by computing the indicator $hitZ[n] = [bin[n] = Z]$, which is 1 if $bin[n] = Z$ and 0 otherwise, and accumulate the result into the count ct.

Our goal is to show that ct is usually not far from its expected value, which is N/B. If the quantities $\{[bin[n] = Z]\}_n$ were independent, we would be able to apply a standard concentration bound to the sum ct. However, $\{[bin[n] = Z]\}_n$ are *not* independent: for instance, since exactly K elements from $[B \cdot K]$ map to Z, if $bin[n] = Z$ for $n \in \{0, 1, \ldots, K-1\}$, then $bin[K] = Z$ must be false.

Nevertheless, we can show that $\{[bin[n] = Z]\}_n$ are negatively associated random variables. Intuitively, $\{g[n]\}_n$ are NA random variables because the result of a uniformly random permutation is NA. Then, $\{bin[n]\}_n$ is computed by mapping the function $mod(-, B)$ over the array g; since this produces another uniform permutation distribution, the vector $\{bin[n]\}_n$ is also NA. By similar reasoning $\{[bin[n] = Z]\}_n$ is also NA, as it is obtained by mapping the function $[- = Z]$ over $\{bin[n]\}_n$.

We formalize the reasoning using the program logic LINA. For the main loop, we apply the rule LOOP with the following loop invariant:

$$\bigwedge_{\alpha=0}^{n} [hitZ[\alpha] = [mod(g[\alpha], B) = Z]] \wedge \mathbf{Permu}_{[B \cdot K]}\langle g \rangle$$
$$\wedge\, ct = \sum_{\alpha=0}^{n} hitZ[\alpha] \wedge ((n \geq N) \rightarrow [n = N])$$

The loop invariant is preserved by the body of the loop, using RASSN and CONST.

Thus we can show the following judgment:

$$\{[ct = 0] \wedge [n = 0]\}$$
$$loop$$
$$\left\{\bigwedge_{\alpha=0}^{N} [hitZ[\alpha] = [mod(g[\alpha], B) = Z]] \wedge \mathbf{Permu}_{[B \cdot K]}\langle g \rangle \wedge \left[ct = \sum_{\alpha=0}^{N} hitZ[\alpha]\right]\right\}$$

Applying (Perm-Map), the post-condition implies:

$$\bigwedge_{\alpha=0}^{N} [hitZ[\alpha] = [mod(g[\alpha], B) = Z]] \wedge \mathop{\Large *}_{\alpha=0}^{N} \mathsf{Own}(hitZ[\alpha])$$

$$\wedge \mathbf{Permu}_{[B \cdot K]}\langle g \rangle \wedge \left[ct = \sum_{\alpha=0}^{N} hitZ[\alpha] \right]$$

Applying basic axioms about expected value and the permutation distribution ((PermMarg) (ProbUnif) (BijectUnif)), we have:

$$\mathop{\Large *}_{\alpha=0}^{N} \mathsf{Own}(hitZ[\alpha]) \wedge \left[ct = \sum_{\alpha=0}^{N} hitZ[\alpha] \right] \wedge [\mathbb{E}[ct] = N/B]$$

And we can apply the negative-association Chernoff bound (NA-Chernoff-2) to conclude:

$$\{\top\}\ \textsc{PermHash}\ \{\Pr[|ct - N/B| > T(\beta, N)] < \beta\}$$

This conclusion corresponds to Proposition A.2 in Ding and König [2011] algorithm for fast set intersection.[5]

3.6.5 Fully-dynamic Dictionary

For our next example, we consider a hashing scheme for a fully-dynamic dictionary, a space-efficient data structure that supports insertions, deletions, and membership queries. The top level of the data structure by Bercea and Even [2022] uses a two-level hashing scheme: elements are first hashed into a *crate*, and then hashed into a *pocket dictionary* within each crate. As part of the space analysis of their scheme, Bercea and Even [2022] proves a high-probability

[5] Ding and König [2011] apply a variant of the Chernoff bound to obtain a multiplicative, rather than an additive, error guarantee. We present the additive version since the bound is a bit simpler, but there is no difficulty in handling the multiplicative version in our framework.

```
FDDICT :
  binCt ← zero(C, P);
  overCt ← zero(C);
  n ← 0;
  while n < N do
    crate[n] ←$ OH_[C];
    pocket[n] ←$ OH_[P];
    bin[n] ← crate[n]^⊤ · pocket[n];
    c ← 0;
    while c < C do
      p ← 0;
      while p < P do
        upd ← binCt[c][p] + bin[n][c][p];
        binCt[c][p] ← upd;
        p ← p + 1;
      c ← c + 1;
    n ← n + 1;
  c ← 0;
  while c < C do
    p ← 0;
    while p < P do
      over[c][p] ← [binCt[c][p] > T_bin];
      upd ← overCt[c] + over[c][p];
      overCt[c] ← upd;
      p ← p + 1;
    c ← c + 1
```

Figure 3.6: Fully-dynamic dictionary [Ding and König, 2011]

bound on the number of pocket dictionaries that overflow after a given number of elements are inserted.

We extract the program FDDICT in Figure 3.6 from the scheme in Bercea and Even [2022]. The program models the insertion of N elements. Each element is first hashed into one of C possible crates uniformly at random and then hashed into one of P possible pocket dictionaries uniformly at random. The variable $bin[n]$ is a C by P matrix, with all entries zero except for the entry at $(crate[n], pocket[n])$, which is set to 1. Next, the program totals up the number

of elements hashing to each (crate, pocket) pair, storing the result in the C by P matrix $binCt$. Finally, the program checks which $(crate, pocket)$ pairs have count larger than some concrete threshold T_{bin} and records that in $over$, and totals up the number of full pocket dictionaries in each crate ($overCt$).

Our logic can prove a judgment of the following form for $T_{bin} \geq N/(P \cdot C)$:

$$\{\top\}\ \text{FDDICT}\ \{\bigwedge_{\gamma=0}^{C} \Pr[overCt[\gamma] \geq P \cdot F(T_{bin} - N/(P \cdot C), N) + T(\rho_{over}, P)] \leq \rho_{over}\},$$

where the logical variables ρ_{bin} and ρ_{over} represents the parametric overflow properties. This formalizes a result similar to Bercea and Even [2022, Claim 21], which states that except with probability β, all crates have at most T_{over} overfull pocket dictionaries. The core of the proof shows that for every crate index γ, the counts $binCt[\gamma][\beta]$ are negatively associated, using the NEGFRAME rule as in the array version of the Bloom filter example. Then, we show that vector $over[\gamma][\beta]$, which indicates whether each pocket dictionary β in crate γ is overfull or not, is also negatively associated. This holds because $over[\gamma][\beta]$ is obtained from $binCt[\gamma][\beta]$ by applying a monotone function. Furthermore, the count of overflows $overCt[\gamma]$ is obtained by another monotone function on $over[\gamma][\beta]$ and thus its entries are also negatively associated.

Now we prove each step using the program logic. We will refer to the two outer-most loops as (1) and (2), the next two outer-most loops as (1.1) and (2.1), and the inner-most loop as (1.1.1).

Computing $\mathbb{E}[binCt[c][p]]$. For loop (1), we apply LOOP with the following loop invariant φ:

$$\bigwedge_{\gamma=0}^{C} \bigwedge_{\beta=0}^{P} [\mathbb{E}[binCt[\gamma][\beta]] = n/(P \cdot C)] \wedge ([n \geq N] \rightarrow [n = N]) \wedge \mathbf{Detm}\langle n \rangle.$$

To show that this invariant is preserved by the loop, by applications of SAMP and RASSN and CONST and FRAME, the following holds after the sampling and the assignment:

$$\mathbf{OH}_{[P]}\langle pocket[n]\rangle * \mathbf{OH}_{[C]}\langle crate[n]\rangle \wedge \left[bin[n] = crate[n]^\top \cdot pocket[n]\right]. \quad (3.6)$$

Using an axiom about independence and products of one-hot vectors (IndProdOH), this implies:

$$\mathbf{OH}_{[C]\times[P]}\langle bin[n]\rangle.$$

Using an axiom about the one-hot encoding (OHMarg):

$$\mathbb{E}[bin[\alpha][\gamma][\beta]] = 1/(P \cdot C)$$

for every α, γ, and β. Standard loop invariants for loop (1.1) and (1.1.1) show that:

$$\left[binCt[c][p] = \sum_{\alpha=0}^{n} bin[\alpha][c][p]\right],$$

and linearity of expectation establishes the invariant condition 3.6 for loop (1). The invariant holds at the start of the loop (1) since *binCt* is zero-initialized, and it also holds at the end of the loop (1). Since *binCt* is not modified further, the expectation equality remains valid at the end of the program due to CONST.

Bounding $\Pr[binCt[c][p] > T_{bin}]$**.** For loop (1), we also apply LOOP with the following loop invariant:

$$\left(\mathop{\Large *}_{\alpha=0}^{n} \mathsf{Own}(bin[\alpha])\right) \wedge \bigwedge_{\gamma=0}^{C}\bigwedge_{\beta=0}^{P}\left[binCt[\gamma][\beta] = \sum_{\alpha=0}^{N} bin[\alpha][\gamma][\beta]\right] \wedge [n = N] \wedge \mathbf{Detm}\langle n\rangle.$$

The first conjunction is an invariant, by applying SAMP and FRAME. The rest of the invariant is preserved, following standard invariants for loops (1.1) and

(1.1.1). By projection (IndMap), at the end of the loop (1) we can conclude:

$$\bigwedge_{\gamma=0}^{C}\bigwedge_{\beta=0}^{P}\left(\mathop{\Large *}_{\alpha=0}^{N}\mathsf{Own}(bin[\alpha][\gamma][\beta])\right)\wedge\left[binCt[\gamma][\beta]=\sum_{\alpha=0}^{N}bin[\alpha][\gamma][\beta]\right].$$

Thus, a standard Chernoff bound gives (here, we apply NA-Chernoff-1 to independent variables by first changing the independence star into NA star using WEAK) :

$$\bigwedge_{\gamma=0}^{C}\bigwedge_{\beta=0}^{P}\Pr[binCt[\gamma][\beta]\geq\mathbb{E}[binCt[\gamma][\beta]]+\rho_{bin}]\leq F(\rho_{bin},N).$$

where $\mathbb{E}[binCt[\gamma][\beta]]$ is $N/(P{\cdot}C)$ by the previous step. Thus, by applying CONJ, we can combine the post-conditions and derive

$$\bigwedge_{\gamma=0}^{C}\bigwedge_{\beta=0}^{P}\Pr[binCt[\gamma][\beta]\geq N/(P\cdot C)+\rho_{bin}]\leq F(\rho_{bin},N). \tag{3.7}$$

Again, the property holds until the end of the program since $binCt$ is not modified further (CONST).

Bounding $\mathbb{E}[overCt[c]]$. Using standard loop invariants, at the end of the loop (2) we have:

$$\bigwedge_{\gamma=0}^{C}\left[overCt[\gamma]=\sum_{\beta=0}^{P}over[\gamma][\beta]\right]\wedge\bigwedge_{\beta=0}^{P}[over[\gamma][\beta]=[binCt[\gamma][\beta]>T_{bin}]].$$

Using linearity of expectation and the fact that $over[\gamma][\beta]$ is either zero or one, we have:

$$\begin{aligned}\mathbb{E}[overCt[\gamma]]&=\sum_{\beta=0}^{P}\mathbb{E}[over[\gamma][\beta]]\\&=\sum_{\beta=0}^{P}\mathbb{E}[binCt[\gamma][\beta]>T_{bin}]\\&=\sum_{\beta=0}^{P}\Pr[binCt[\gamma][\beta]>T_{bin}]\end{aligned}$$

Because the bound we obtained in eq. (3.7),

$$\sum_{\beta=0}^{P} \Pr[binCt[\gamma][\beta] > T_{bin}] \leq \sum_{\beta=0}^{P} F(T_{bin} - N/(P \cdot C), N)$$
$$= P \cdot F(T_{bin} - N/(P \cdot C), N)$$

for $T_{bin} > N/(P \cdot C)$. Thus, $\mathbb{E}[overCt[\gamma]] = P \cdot F(T_{bin} - N/(P \cdot C), N)$.

Bounding $\Pr[overCt[c]] > T_{over}]$. At the high level, we want the following loop invariant for Loop (1):

$$\bigwedge_{\gamma=0}^{C} \circledast_{\beta=0}^{P} \mathsf{Own}(binCt[\gamma][\beta])$$

We want the following loop invariant for (1.1):

$$\bigwedge_{\gamma=c}^{C} \circledast_{\beta=0}^{P} \mathsf{Own}(binCt[\gamma][\beta]) \circledast \circledast_{\beta=0}^{P} \mathsf{Own}(bin[n][\gamma][\beta])$$

$$\wedge \bigwedge_{\gamma=0}^{c} \circledast_{\beta=0}^{P} \mathsf{Own}(binCt[\gamma][\beta]) \wedge [c \leq C] \wedge \mathbf{Detm}\langle c \rangle$$

And the following loop invariant for (1.1.1):

$$\bigwedge_{\gamma=c+1}^{C} \circledast_{\beta=0}^{P} \mathsf{Own}(binCt[\gamma][\beta]) \circledast \circledast_{\beta=0}^{P} \mathsf{Own}(bin[n][\gamma][\beta])$$

$$\wedge \circledast_{\beta=0}^{P} \mathsf{Own}(binCt[c][\beta]) \circledast \circledast_{\beta=p}^{P} \mathsf{Own}(bin[n][c][\beta])$$

$$\wedge \bigwedge_{\gamma=0}^{c} \circledast_{\beta=0}^{P} \mathsf{Own}(binCt[\gamma][\beta]) \wedge [c \leq C] \wedge \mathbf{Detm}\langle c \rangle \wedge [p \leq P] \wedge \mathbf{Detm}\langle p \rangle$$

We show the loops preserve the respective invariant for a fixed γ; the big conjunction then follows by applying Conj. Working from inside to outside, we

start with loop (1.1.1). To establish the invariant condition, the critical case is $\gamma = c$. We can pull out:

$$\varphi := \circledast_{\beta=0}^{p} \mathsf{Own}(binCt[c][\beta]) \circledast \circledast_{\beta=p+1}^{P} \mathsf{Own}(binCt[c][\beta])$$

$$\circledast \circledast_{\beta=p+1}^{P} \mathsf{Own}(bin[n][c][\beta]) \circledast \underbrace{\mathsf{Own}(binCt[c][p]) \circledast \mathsf{Own}(bin[n][c][p])}_{\Phi}$$

Now, we can use the assignment rule to show:

$$\{\Phi\}\ upd \leftarrow binCt[c][p] + bin[n][c][p]\ \{\varphi \wedge [upd = binCt[c][p] + bin[n]cp]\}$$

Since addition is a monotone function, the NA frame rule NEGFRAME gives:

$$\circledast_{\beta=0}^{p} \mathsf{Own}(binCt[c][\beta]) \circledast \circledast_{\beta=p+1}^{P} \mathsf{Own}(binCt[c][\beta]) \circledast \circledast_{\beta=p+1}^{P} \mathsf{Own}(bin[n][c][\beta]) \circledast \mathsf{Own}(upd)$$

after the assignment to upd. After the assignment to $bin[c][p]$, we can fold it into

$$\circledast_{\beta=0}^{P} \mathsf{Own}(binCt[c][\beta]) \circledast \circledast_{\beta=p+1}^{P} \mathsf{Own}(bin[n][c][\beta]).$$

Then, by applying DASSN, we get

$$\circledast_{\beta=0}^{P} \mathsf{Own}(binCt[c][\beta]) \circledast \circledast_{\beta=p}^{P} \mathsf{Own}(bin[n][c][\beta]).$$

The assertion for all the other γ remains unchanged, so we establish the invariant for loop (1.1.1).

Now we reason about the loop (1.1). Since $binCt$ is zero-initialized (DetInd), the invariant for loop (1.1.1) holds on loop entry. Then, apply LOOP with the

loop invariant established above for loop (1.1.1) gives us

$$\bigwedge_{\gamma=c+1}^{C} \circledast_{\beta=0}^{P} \mathsf{Own}(binCt[\gamma][\beta]) \circledast \circledast_{\beta=0}^{P} \mathsf{Own}(bin[n][\gamma][\beta])$$

$$\wedge \bigwedge_{\gamma=0}^{c+1} \circledast_{\beta=0}^{P} \mathsf{Own}(binCt[\gamma][\beta]) \wedge [c < C] \wedge \mathbf{Detm}\langle c \rangle$$

after the termination of loop (1.1.1). The program then deterministically increases c by 1, and by DAssn, we can establish the loop invariant for loop 1.1.

Similarly, loop invariant for loop (1) is established when loop (1.1) exits and we increase n by 1. Thus, after loop (1) terminates, we have the postcondition:

$$\bigwedge_{\gamma=0}^{C} \circledast_{\beta=0}^{P} \mathsf{Own}(binCt[\gamma][\beta])$$

Next, we tackle loop (2). We take the invariant:

$$\bigwedge_{\gamma=0}^{c} \circledast_{\beta=0}^{P} \mathsf{Own}(over[\gamma][\beta]) \wedge \left[overCt[\gamma] = \sum_{\beta=0}^{P} over[\gamma][\beta] \right]$$

$$\wedge \bigwedge_{\gamma=0}^{C} \circledast_{\beta=0}^{P} \mathsf{Own}(binCt[\gamma][\beta]) \wedge [c \leq C] \wedge \mathbf{Detm}\langle c \rangle$$

For the inner loop (2.1), we take the invariant:

$$\bigwedge_{\gamma=0}^{c} \circledast_{\beta=0}^{P} \mathsf{Own}(over[\gamma][\beta]) \wedge \left[overCt[\gamma] = \sum_{\beta=0}^{P} over[\gamma][\beta] \right]$$

$$\wedge \circledast_{\beta=0}^{p} \mathsf{Own}(over[c][\beta]) \circledast \circledast_{\beta=p}^{P} \mathsf{Own}(binCt[c][\beta]) \wedge \left[overCt[\gamma] = \sum_{\beta=0}^{p} over[\gamma][\beta] \right]$$

$$\wedge \bigwedge_{\gamma=0}^{C} \circledast_{\beta=0}^{P} \mathsf{Own}(binCt[\gamma][\beta]) \wedge [c \leq C] \wedge \mathbf{Detm}\langle c \rangle$$

Again, we show the invariant post-conditions for a fixed γ. For the critical iteration $\gamma = c$, we again isolate $binCt[c][p]$, observe that addition is monotone and the function $[binCt[c][p] > T_{bin}]$ is monotone in $binCt[\gamma][p]$, and apply the NA frame rule NEGFRAME.

Finally, at the end of the program, we can show:

$$\circledast_{\beta=0}^{P} \mathsf{Own}(over[\gamma][\beta])$$

along with the regular invariant

$$\left[overCt[\gamma] = \sum_{\beta=0}^{P} over[\gamma][\beta] \right].$$

We can then apply the negative-dependence Chernoff bound (NA-Chernoff-2):

$$\Pr[overCt[\gamma] \geq \mathbb{E}[overCt[\gamma]] + T(\rho_{over}, P)] \leq \rho_{over}.$$

Using the expectation bound from the previous step and putting everything together, we conclude:

$$\left\{\top\right\} \text{FDDICT} \left\{ \bigwedge_{\gamma=0}^{C} \Pr[overCt[\gamma] \geq P \cdot F(T_{bin} - N/(P \cdot C), N) + T(\rho_{over}, P)] \leq \rho_{over} \right\},$$

thus showing a high-probability upper-bound on the number of overfull pockets dictionaries within each crate.

3.6.6 Repeated Balls-into-bins Process

Our final example considers a probabilistic protocol proposed by Becchetti et al. [2019], implemented as REPEATBIB in Figure 3.7. Intuitively, the program implements a repeated balls-into-bins process. Initially, N balls are distributed

```
REPEATBIB :
  r ← 0;
  while r < R do
    n ← 0
    rem ← 0;
    while n < N do
      ct[n] ← ct[n] − [ct[n] > 0];
      rem ← rem + [n > 0];
      n ← n + 1;
    j ← 0;
    while j < rem do
      bin[j] ←$ OH_[N];
      k ← 0;
      while k < N do
        upd ← ct[k] + bin[j][k];
        ct[k] ← upd;
        k ← k + 1;
      j ← j + 1;
    n ← 0;
    emptyCt[r] ← 0;
    empty ← isZero(ct);
    while n < N do
      upd ← emptyCt[r] + empty[n];
      emptyCt[r] ← upd;
      n ← n + 1;
    r ← r + 1;
```

Figure 3.7: Repeated balls-into-bins [Becchetti et al., 2019]

among N bins ($ct[n]$). For R rounds, in each round, a ball is first removed from every non-empty bin. Then, the rem removed balls are randomly reassigned to bins. This process is useful for distributed protocols and scheduling algorithms, where the balls represent tasks and the bins represent computation nodes. Becchetti et al. [2019] proposed and analyzed this algorithm (e.g., bounding the maximum load, proving how long it takes for all balls to visit all bins). We can verify the following lower-bound on the number of empty bins, analogous to

Becchetti et al. [2019, Lemma 1 and Lemma 2]:

$$\left\{N \geq 2 \wedge \left[\sum_{\alpha=0}^{N} ct[\alpha] = N\right]\right\}$$

RepeatBIB

$$\left\{\Pr\left[\bigvee_{\beta=0}^{R} (emptyCt[\beta] < \frac{N}{15} - T(\rho_{empty}, N))\right] \leq R \cdot \rho_{empty}\right\}$$

Two aspects of this program make it more difficult to verify. First, there is a loop with a randomized guard: the number of removed balls rem is a randomized quantity. Reasoning about such loops is challenging because our Loop rule is not directly applicable and only far weaker rules are available for loops with general randomized guards. Becchetti et al. [2019] sidestep this problem by *conditioning* on the number of balls in each bin, which also fixes rem to be some value, proving the target property for every fixed setting, and then combining the proofs together. LINA can formalize this style of reasoning using the randomized case analysis rule RCase to condition on rem's value, and then apply the section 2.3.3 rule; however, the post-condition of section 3.5.2 must be closed under mixtures, while independence and negative association are known *not* to satisfy this side-condition. Thus, it is not possible to prove negative association by first conditioning and then combining. To work around this second problem, we use a technique from Becchetti et al. [2019] and prove, on each conditional distribution, a high-probability bound using the Chernoff bound. The benefit of this approach is that high-probability bounds *are* closed under mixture, so we can apply RCase to combine the results.

In the formal proof, we will refer to the loops in Figure 3.7 using the same scheme we used before: the outer-most loop is loop (1), the three next-outer-most loops are loops (1.1), (1.2), and (1.3), and the inner-most loop is loop (1.2.1).

Starting from the outside, we take the following invariant for loop (1):

$$\Pr\left[\bigvee_{\beta=0}^{r}(emptyCt[\beta] > T_{empty})\right] \leq r \cdot \rho_{empty} \wedge \left[\sum_{\alpha=0}^{N} ct[\alpha] = N\right]$$

Showing the invariant condition requires some work. First, note that:

$$\models_{\mathbf{Mem}} \left[\sum_{\alpha=0}^{N} ct[\alpha] = N\right] \rightarrow \bigvee_{\sigma:[N]\rightarrow[N]} \bigwedge_{\alpha=0}^{N} \left[ct[\alpha] = |\sigma^{-1}(\alpha)|\right]$$

where $\sigma : [N] \rightarrow [N]$ ranges over all assignments of N balls to N bins. We write $\tau(\alpha) = |\sigma^{-1}(\alpha)|$ for the number of balls in bin α. We will show:

$$\left\{\bigwedge_{\alpha=0}^{N}[ct[\alpha] = \tau(\alpha)]\right\}\ body\ \left\{\Pr\left[emptyCt[r] < T_{empty}\right] \leq \rho_{empty}\right\}$$

where *body* is the body of loop (1). For loop (1.1), it is straightforward to show the invariant using RASSN:

$$\bigwedge_{\alpha=n}^{N}([ct[\alpha] = \tau(\alpha)]) \wedge \bigwedge_{\alpha=0}^{n}([ct[\alpha] = \tau(\alpha) - [\tau(\alpha) > 0]])$$
$$\wedge \left[rem = \sum_{\alpha=0}^{n}[\sigma(\alpha) > 0]\right] \wedge [n \leq N]$$

Using the loop rule LOOP, we derive the following at the exit of loop (1.1),

$$\bigwedge_{\alpha=0}^{N}([ct[\alpha] = \tau(\alpha) - [\tau(\alpha) > 0]]) \wedge \left[rem = \sum_{\alpha=0}^{N}[\sigma(\alpha) > 0]\right]$$

Since counts are all equal to expressions of logical variables, conditioning on the logical variable σ, they are all deterministic; Thus, we have

$$\bigwedge_{\alpha=0}^{N} \mathbf{Detm}\langle ct[\alpha]\rangle \wedge \mathbf{Detm}\langle rem\rangle$$

which implies $\circledast_{\alpha=0}^{N} \mathsf{Own}(ct[\alpha]) \wedge \mathbf{Detm}\langle rem\rangle$.

We take $\circledast_{\alpha=0}^{N} \mathsf{Own}(ct[\alpha]) \wedge \mathbf{Detm}\langle rem\rangle$ to be the invariant for loop (1.2). To establish this, we reason much as in the previous examples. The sampling rule SAMP gives:

$$\circledast_{\alpha=0}^{N} \mathsf{Own}(ct[\alpha]) * \mathsf{Own}(bin[j])$$

By negative association for one-hot encoding (OH-PNA):

$$\circledast_{\alpha=0}^{N} \mathsf{Own}(ct[\alpha]) * \circledast_{\alpha=0}^{N} \mathsf{Own}(bin[j][\alpha]).$$

For the inner-most loop (1.2.1), we apply the same technique as for loop (1.2). Since loop (1.2) has a randomized guard, k is a random variable, and loop (1.2.1) also has a randomized guard. However, under the conditioning, we may assume that k is deterministic and apply LOOP on loop (1.2.1) with the following invariant:

$$\circledast_{\alpha=0}^{k} \mathsf{Own}(ct[\alpha]) \circledast \circledast_{\alpha=k}^{N} (\mathsf{Own}(ct[k]) \circledast \mathsf{Own}(bin[j][\alpha])) \wedge [k \leq N]$$

Like in earlier examples, we can establish this invariant using NEGFRAME since $ct[n] + bin[j][n]$ is monotone. Thus at the exit of loop (1.2.1), we have:

$$\circledast_{\alpha=0}^{N} \mathsf{Own}(ct[\alpha])$$

And that is preserved to the end of loop (1.2). Next, three applications of the assignment rule RASSN give:

$$\circledast_{\alpha=0}^{N} \mathsf{Own}(ct[\alpha]) \wedge [n = 0] \wedge [emptyCt[r] = 0] \wedge [empty = isZero(ct)]$$

The function $isZero(v)$ takes a numerical vector v and returns a vector where each index i is 1 if $v[i]$ is zero, else it holds 0. This is an antitone function: it is non-increasing in its argument. Thus, the monotone mapping axiom (Mono-Map) gives:

$$\circledast_{\alpha=0}^{N} \mathsf{Own}(empty[\alpha])$$

Then, a standard loop invariant for loop (1.3) gives:

$$\circledast_{\alpha=0}^{N} \mathsf{Own}(empty[\alpha]) \wedge \left[emptyCt[r] = \sum_{\alpha=0}^{N} empty[\alpha] \right]$$

at the end of loop (1.3). Thus,

$$\Big\{\bigwedge_{\alpha=0}^{N}[ct[\alpha]=\tau(\alpha)]\Big\}\ body\ \Big\{\circledast_{\alpha=0}^{N}\mathsf{Own}(empty[\alpha])\wedge\Big[emptyCt[r]=\sum_{\alpha=0}^{N}empty[\alpha]\Big]\Big\}.$$

Now, we are in a position to apply the negative association Chernoff bound (NA-Chernoff-2), giving the judgment:

$$\Big\{\bigwedge_{\alpha=0}^{N}[ct[\alpha]=\tau(\alpha)]\Big\}\ body\ \Big\{\Pr[emptyCt[r]\le\mathbb{E}[emptyCt[r]]-T(\rho_{empty},N)]\le\rho_{empty}\Big\}$$

where *body* is the body of loop (1).

Next we bound $\mathbb{E}[emptyCt[r]]$ by translating an argument by Becchetti et al. [2019, Lemma 2] into our logic gives:

$$\Big\{\bigwedge_{\alpha=0}^{N}[ct[\alpha]=\tau(\alpha)]\wedge[N\ge 2]\Big\}\ body\ \Big\{\mathbb{E}[emptyCt[r]]\ge N/15\Big\}$$

The argument makes use of basic properties of expected values and the exponential function; we omit the details. Thus, we can conclude that

$$\Big\{\bigwedge_{\alpha=0}^{N}[ct[\alpha]=\tau(\alpha)]\Big\}\ body\ \Big\{\Pr[emptyCt[r]\le N/15-T(\rho_{empty},N)]\le\rho_{empty}\Big\}.$$

Note that this post-condition is closed under the mixture. So now we can apply the randomized case analysis rule RCase to combine the proof for different assignments σ. We can take the trivial pre-condition $\varphi=\top$, and the case condition:

$$\eta:=\bigvee_{\sigma:\{N\}\to\{N\}}\bigwedge_{\alpha=0}^{N}[ct[\alpha]=\tau(\alpha)].$$

Since η asserts that one of the equality holds with probability 1, it is closed under conditioning. Applying RCase, we have:

$$\Big\{\bigvee_{\sigma:\{N\}\to\{N\}}\bigwedge_{\alpha=0}^{N}[ct[\alpha]=\tau(\alpha)]\Big\}\ body\ \Big\{\Pr[emptyCt[r]<N/15-T(\rho_{empty},N)]\le\rho_{empty}\Big\}$$

Also, since $\models \sum_{\alpha=0}^{N}[ct[\alpha] = N] \rightarrow \bigvee_{\sigma:\{N\}\to\{N\}} \bigwedge_{\alpha=0}^{N}[ct[\alpha] = \tau(\alpha)]$, we can weaken the precondition into $\sum_{\alpha=0}^{N}[ct[\alpha] = N] \rightarrow \bigvee_{\sigma:\{N\}\to\{N\}}$.

Recalling that we wanted the following invariant to get preserved by loop (1):

$$\Pr\left[\bigvee_{\beta=0}^{r}(emptyCt[\beta] < T_{empty})\right] \leq r \cdot \rho_{empty} \wedge \sum_{\alpha=0}^{N}[ct[\alpha] = N] \wedge \mathbf{Detm}\langle r\rangle \wedge [N \geq 2]$$

We can use the rule of constancy CONST and the assignment rule DASSN to preserve the first conjunct to show:

$$\Pr\left[\bigvee_{\beta=0}^{r-1}(emptyCt[\beta] < T_{empty})\right] \leq (r-1) \cdot \rho_{empty}$$

at the end of the body of loop (1). Combined with the probability bound for $emptyCt[r]$, an application of the union bound (UnionBd) establishes the invariant for loop (1). Putting everything together, we have:

$$\left\{N \geq 2 \wedge \sum_{\alpha=0}^{N}[ct[\alpha] = N]\right\}$$

REPEATBIB

$$\left\{\Pr\left[\bigvee_{\beta=0}^{R}(emptyCt[\beta] < N/15 - T(\rho_{empty}, N))\right] \leq R \cdot \rho_{empty}\right\}$$

analogous to Becchetti et al. [2019, Lemma 1 and 2].

3.7 Related Work

Verifying approximate data structures and applying concentration bounds. Bloom filters are a data structure supporting *approximate membership queries* (AMQs). Ceramist [Gopinathan and Sergey, 2020] is a recent framework for

verifying hash-based AMQ structures in the Coq theorem prover. Besides handling Bloom filters, Ceramist supports subtle proofs of correctness for many other AMQs. Compared with our approach, Ceramist proofs are more precise but also more intricate, applying theorems about Stirling numbers to achieve a precise bound on the false positive probability. In contrast, our approach reasons about negative dependence to achieve a substantially simpler proof, albeit with less precise bounds.

Prior works in verification have also applied the Chernoff bound to bound sums of independent random quantities (e.g., [Wang et al., 2021, Chakarov and Sankaranarayanan, 2013]). While independence is easier to establish, the negative association property that we need is more subtle.

Negative dependence. There are multiple definitions of negative dependence in the literature, each with their own strengths and weaknesses. We work with negative association (NA) [Joag-Dev and Proschan, 1983, Dubhashi and Ranjan, 1998], because it holds in many situations where negative dependence should hold and it is closed under various notions of composition. Recently, the notion of Strong Rayleigh (SR) [Borcea et al., 2009] distribution has been proposed as an ideal definition of negative dependence. The SR condition satisfies more closure properties than NA does; in particular, it is preserved under various forms of conditioning. However, SR distributions have mostly been studied for Boolean variables only, and we do not know if an analogue of the monotone maps property of NA holds for SR.

Beyond theoretical investigations, negative dependence plays a useful role in many practical applications. In machine learning, negative dependence can

help ensure diversity in predictions by a model [Kulesza and Taskar, 2012], and fast algorithms are known to learn and sample from negatively-dependent distributions [Anari et al., 2016]. In algorithm design, negative dependence is a useful tool to randomly round solutions of linear programs to integral solutions [Srinivasan, 2001]. Negative dependence can ensure that certain constraints are satisfied exactly after rounding, while still allowing concentration bounds to be applied to analyze the quality of the rounded solution.

CHAPTER 4

A BUNCHED LOGIC FOR DEPENDENCE AND INDEPENDENCE

Conditional independence (CI) is a well-studied notion in probability theory and statistics [Dawid, 1979, Pearl et al., 1989, Dawid, 2001, Simpson, 2018]. While there are many interpretations of CI, a natural reading is in terms of *irrelevance*: X and Y are independent conditioned on Z if knowing the value of Z renders X and Y unrelated; in other words, observing one gives no further information about the other.

Conditional independence has a wide range of applications. For example, it enables distinguishing superfluous correlation from causation. For instance, suppose researchers found a strong positive correlation between a nation's per capita Nobel laureates number and chocolate consumption. A convenient (mis)interpretation would be that chocolate consumption makes people smarter and leads to more Nobel Laureates. But the correlation is likely due to other factors, e.g., a nation's economic status, and the two are conditionally independent fixing the third factor.

Conditional independence can also succinctly encode interesting properties. As more and more life-changing decisions, e.g., job hiring, judicial decisions, and loan approvals, are automated using prediction algorithms, algorithmic fairness has gained more attention. To prevent algorithms from discriminating based on sensitive features (e.g., race and gender), researchers formalized notions of fairness originated from different philosophies using conditional independence. For instance, one school of thought is that an algorithm is fair if it satisfies *equalized odds*, i.e., the algorithm's predictions and the sensitive features are conditionally independent, fixing the innate quality (i.e., the target label)

that the algorithm is aiming to predict; another proposal for fairness is *calibration*, which says that fixing on the algorithm's prediction, the sensitive features and the target label are conditionally independent. (More details are presented in Barocas et al. [2023].)

Since we are studying probabilistic programs, we want to reason about conditional independence of (sets of) program variables, which is defined as follows:

Definition 4.0.1 (Conditional independence)**.** Let $X, Y, Z \subseteq \mathbf{Var}$. For any $m \in \mathbf{Mem}[\mathbf{Var}]$, we write the event $\{\omega \in \mathbf{Mem}[\mathbf{Var}] \mid \forall x \in X.\omega(x) = m(x)\}$ as $X = m$. The set of variables X and Y are *independent conditioned on* Z, written $X \perp\!\!\!\perp Y \mid Z$, if for all $x \in \mathbf{Mem}[X]$, $y \in \mathbf{Mem}[Y]$, and $z \in \mathbf{Mem}[Z]$:

$$\mu(X = x \mid Z = z) \cdot \mu(Y = y \mid Z = z) = \mu(X = x, Y = y \mid Z = z).$$

When $Z = \emptyset$, we say X and Y are *independent*, written $X \perp\!\!\!\perp Y$.

Conditional independence of program variables allows for more efficient representation of distributions over program memories. For instance, if $X \perp\!\!\!\perp Y \mid Z$, then instead of storing the joint distribution of X, Y, Z, one can store the distribution of Z, the marginal distribution of X given Z, and the marginal distribution of Y given Z; when there are n possible outcomes for each of X, Y, Z, storing the former takes $O(n^3)$ space, while storing the latter only takes $O(n^2)$ space. The factored representation also enables more efficient inference algorithms (e.g., Holtzen [2021]), which are developed to compute or approximate the distribution after conditioning on an observation.

Thus, we want to extend probabilistic separation logic to prove conditional independence of program variables. To achieve that, we need an assertion logic

that can express conditional independence. The existing probabilistic BI model (Section 2.3.2) provides no means to describe the distribution over program memories conditioned on the values a set of variables takes. Accordingly, one cannot capture the basic statement of conditional independence, i.e., X and Y are independent conditioned on any value of Z. To address that problem, we develop a novel assertion logic DIBI, short for *Dependence and Independence BI*. DIBI extends BI with new connectives: the conjunction $P \fatsemi Q$ for modeling dependence between states and its adjoints $P \mathrel{-\!\!\circ} Q$ and $P \mathrel{\circ\!\!-} Q$. We then develop a probabilistic model of DIBI so that $P * Q$ can assert probabilistic independence and $P \fatsemi Q$ can assert dependence. Then, we express conditional independence of X and Y given Z roughly as $Z \fatsemi (X * Y)$, which asserts the independence of X and Y while they both depend on Z¿

Intuitively, to assert dependence with the conjunction $P \fatsemi Q$, we want to interpret $\fatsemi$ through a binary operator $\odot$, where the operator $\odot$ is defined so that in the composed distribution $f \odot g$, the variables described by g depends on the variables described f; however, it is unclear how to define such an operator $\odot$ for distributions. To address this problem, we design a DIBI model whose states are not distributions but *Markov kernels* [Panangaden, 2009], which are essentially maps from a set A to a distribution over a set B and get the name because of their role in the theory of general Markov processes [Dynkin, 2012]. We will sometimes abbreviate them as *kernels* for convenience.

Crucially, Markov kernels can be composed sequentially using the bind operation in the distribution monad: given $f \colon X \to \mathbb{D}(Y)$, $g \colon Y \to \mathbb{D}(Z)$, the *Kleisli composition* $f ; g \colon X \to \mathbb{D}(Z)$ is:

$$(f ; g)(x) := \mathsf{bind}(f(x), g) \tag{4.1}$$

```
z ←$ Bern_{1/2};
if z then
    x ←$ Bern_{3/4};
    y ←$ Bern_{3/4};
else
    x ←$ Bern_{1/2};
    y ←$ Bern_{1/2}
```

(a) Probabilistic program p

x	y	z	μ
0	0	0	1/8
0	0	1	1/32
1	0	0	1/8
1	0	1	3/32
0	1	0	1/8
0	1	1	3/32
1	1	0	1/8
1	1	1	9/32

(b) Distribution μ generated by p

x	y	μ_0
0	0	1/4
1	0	1/4
0	1	1/4
1	1	1/4

(c) μ conditioned on $z = 0$

x	y	μ_1
0	0	1/16
1	0	3/16
0	1	3/16
1	1	9/16

(d) μ conditioned on $z = 1$

Figure 4.1: From probabilistic programs to kernels

Markov kernels generalize distributions because we can lift any distribution $\mu : \mathbb{D}(X)$ to a kernel $f_\mu : 1 \to \mathbb{D}(X)$ by assigning μ to the single element of 1. Kernels can also encode conditional distributions, which play a key role in conditional independence. We show an example of how to encode conditional distributions using kernels below.

Example 4.0.1 (Kernels and Conditional Probabilities)**.** Consider the program p in Figure 4.1a, where x, y, and z are Boolean variables. First, flip a fair coin and store the result in z. If $z = 0$, flip a fair coin twice, and store the results in x and y, respectively. If $z = 1$, flip a coin with bias $1/4$ twice, and store the results in x and y. This program produces a distribution μ, shown in Figure 4.1b.

If we condition μ on $z = 0$, then the resulting distribution μ_0 models two independent fair coin flips: $1/4$ probability for each possible pair of outcomes (Figure 4.1c). If we condition on $z = 1$, however, then the distribution μ_1 will be skewed — there will be a much higher probability that we observe $(1, 1)$ than $(0, 0)$, but x and y are still independent given z (Figure 4.1d).

To connect μ_0 and μ_1 to the original distribution μ, we package μ_0 and μ_1

into a Markov kernel $k : \mathbf{Mem}[z] \to \mathcal{D}(\mathbf{Mem}[\{x, y, z\}])$ given by

$$k(z = i)(x = v_x, y = v_y) = \mu_i(x = v_x, y = v_y)$$

for any $v_x, v_y \in \mathbf{Val}$. Then, the relation between the conditional and original distributions is $f_\mu = f_{\mu_z}; k$, where μ_z is the projection of μ on $\{z\}$.

In the following, we first introduce the metatheory of the DIBI logic in section 4.1, then define the probabilistic model of DIBI on Markov kernels in section 4.2. After showing that conditional independence can be asserted in the probabilistic model of DIBI in subsection 4.2.2, we lay out a program logic using DIBI as the assertion logic in section 4.3.

4.1 DIBI Logic

Analogous to the sections about bunched logic and LINA, we first introduce the syntax and semantics for DIBI formulas, then provide a proof system, and last show that the proof system is sound and complete.

4.1.1 Syntax and semantics

The syntax of DIBI extends BI with a non-commutative conjunctive connective $\fatsemi$ and its associated implications $\multimap$ and $\circ\!\!-$. Let $\mathcal{AP}$ be a set of propositional atoms. The set of DIBI formulas, $\mathrm{Form}_{\mathrm{DIBI}}$, is generated by the following gram-

mar:

$$P, Q ::= p \in \mathcal{AP} \mid \top \mid I \mid \bot \mid P \wedge Q \mid P \vee Q \mid P \to Q$$
$$\mid P * Q \mid P \mathrel{-\!\!*} Q \mid P \fatsemi Q \mid P \multimap Q \mid P \mathrel{\circ\!\!-} Q.$$

DIBI formulas are interpreted on DIBI frames, which extend BI frames. As in BI frames, we want to define one binary operator, denoted $\oplus$ here, to interpret $P * Q$, which asserts the separation of resources validating P and Q. The main extension is a new binary operator $\odot$ for interpreting the formulas $P \fatsemi Q$, $P \multimap Q$ and $P \mathrel{\circ\!\!-} Q$. Informally, we want $P \fatsemi Q$ to assert that the resource validating Q depends on the resource validating P. Because dependence in general is not commutative, we define $\odot$ as a non-commutative operator.

Definition 4.1.1 (DIBI Frame)**.** A *DIBI frame* is a structure $\mathcal{X} = (X, \sqsubseteq, \oplus, \odot, E)$ such that $\sqsubseteq$ is a preorder, $E \subseteq X$, and $\oplus \colon X^2 \to \mathcal{P}(X)$ and $\odot \colon X^2 \to \mathcal{P}(X)$ are binary operations, satisfying the rules in Figure 4.2.

Similar to the case in BI frames, X is a set of states, the preorder $\sqsubseteq$ describes when a smaller state can be extended to a larger state, the binary operators $\odot$, $\oplus$ offer two ways of combining states, and E is the set of states that act like units with respect to these operations. For instance, in our intended model for probabilistic programs, the states would be Markov kernels that *preserve their input through to their output*, which present conditional distributions. We would define $f \oplus g$ to return the set of independent products of two kernels — there is no standard definition for this but roughly it should generalize independent product of distributions, and define $f \odot g$ to return the set of kernels obtained by the sequential composition of two kernels, which is based on the monadic bind. The definition of pre-order would generalize the pre-order in PSL's assertion logic, which says μ_1 is smaller than μ_2 if μ_1 is a marginal distribution of μ_2.

$$z \in x \oplus y \wedge x \sqsupseteq x' \wedge y \sqsupseteq y' \rightarrow \exists z'(z \sqsupseteq z' \wedge z' \in x' \oplus y'); \quad (\oplus \text{ Down-Closed})$$

$$z \in x \odot y \wedge z' \sqsupseteq z \rightarrow \exists x', y'(x' \sqsupseteq x \wedge y' \sqsupseteq y \wedge z' \in x' \odot y') \quad (\odot \text{ Up-Closed})$$

$$z \in x \oplus y \rightarrow z \in y \oplus x; \quad (\oplus \text{ Commutativity})$$

$$w \in t \oplus z \wedge t \in x \oplus y \rightarrow \exists s(s \in y \oplus z \wedge w \in x \oplus s); \quad (\oplus \text{ Associativity})$$

$$\exists e \in E(x \in e \oplus x); \quad (\oplus \text{ Unit Existence})$$

$$e \in E \wedge x \in y \oplus e \rightarrow x \sqsupseteq y; \quad (\oplus \text{ Unit Coherence})$$

$$\exists t(w \in t \odot z \wedge t \in x \odot y) \leftrightarrow \exists s(s \in y \odot z \wedge w \in x \odot s); \quad (\odot \text{ Associativity})$$

$$\exists e \in E(x \in e \odot x); \quad (\odot \text{ Unit Existence}_{\mathrm{L}})$$

$$\exists e \in E(x \in x \odot e); \quad (\odot \text{ Unit Existence}_{\mathrm{R}})$$

$$e \in E \wedge x \in y \odot e \rightarrow x \sqsupseteq y; \quad (\odot \text{ Coherence}_{\mathrm{R}})$$

$$e \in E \wedge e' \sqsupseteq e \rightarrow e' \in E; \quad (\text{Unit Closure})$$

$$x \in y \oplus z \wedge y \in y_1 \odot y_2 \wedge z \in z_1 \odot z_2 \rightarrow \exists u, v(u \in y_1 \oplus z_1 \wedge v \in y_2 \oplus z_2 \wedge x \in u \odot v). \quad (\text{Reverse Exchange})$$

Figure 4.2: DIBI frame requirements (with outermost universal quantification omitted for readability).

The frame conditions define properties that must hold for all models of DIBI. The frame conditions required for $\oplus$ are exactly the frame conditions satisfied by the binary combination in a BI frame; that is, $(X, \sqsubseteq, \oplus, E)$ forms a BI frame. The binary combination $\odot$, in contrast, is not commutative, but it is still associative and has units. Having $\odot$ being non-commutative splits the $\odot$ analogues of $\oplus$ axioms into pairs of axioms, although we exclude the left version of ($\odot$ Coherence) for reasons we explain in section 4.1.2. Also, while $\oplus$ is downwards-closed as in the binary operation in BI frames, the new binary combination $\odot$ is upwards-closed. These choices of closedness conditions match the desired interpretations of $\oplus$ as independence and $\odot$ as dependence: independence should drop down to substates (which must necessarily be independent if the superstates were so), while independence should be inherited by superstates (the source of dependence will still be present in any extensions). Finally, the (Reverse Exchange) condition defines the interaction between $\oplus$ and $\odot$: intuitively, if y_2 depends on

y_1 and z_2 depends on z_1, and y_1, y_2 are independent from z_1, z_2, then the combination of y_2 and z_2 depends on y_1 and z_1.

We give a Kripke-style semantics for DIBI.

Definition 4.1.2 (Valuation and model)**.** A *persistent valuation* of DIBI is an assignment $\mathcal{V}\colon \mathcal{AP} \to \mathcal{P}(X)$ of atomic propositions to subsets of states of a DIBI frame satisfying persistence: if $x \in \mathcal{V}(p)$ and $y \sqsupseteq x$ then $y \in \mathcal{V}(p)$. A *DIBI model* $(\mathcal{X}, \mathcal{V})$ is a DIBI frame $\mathcal{X}$ together with a persistent valuation $\mathcal{V}$.

We now inductively define satisfaction of DIBI formulas in a DIBI model.

x	$\models_{\mathcal{V}}$	$\top$		always
x	$\models_{\mathcal{V}}$	$\bot$		never
x	$\models_{\mathcal{V}}$	I	iff	$x \in E$
x	$\models_{\mathcal{V}}$	p	iff	$x \in \mathcal{V}(p)$
x	$\models_{\mathcal{V}}$	$P \wedge Q$	iff	$x \models_{\mathcal{V}} P$ and $x \models_{\mathcal{V}} Q$
x	$\models_{\mathcal{V}}$	$P \vee Q$	iff	$x \models_{\mathcal{V}} P$ or $x \models_{\mathcal{V}} Q$
x	$\models_{\mathcal{V}}$	$P \to Q$	iff	for all $y \sqsupseteq x$, $y \models_{\mathcal{V}} P$ implies $y \models_{\mathcal{V}} Q$
x	$\models_{\mathcal{V}}$	$P * Q$	iff	there exist x', y, z s.t. $x \sqsupseteq x' \in y \oplus z$, $y \models_{\mathcal{V}} P$ and $z \models_{\mathcal{V}} Q$
x	$\models_{\mathcal{V}}$	$P \fatsemi Q$	iff	there exist y, z s.t. $x \in y \odot z$, $y \models_{\mathcal{V}} P$ and $z \models_{\mathcal{V}} Q$
x	$\models_{\mathcal{V}}$	$P \mathrel{-\!\!*} Q$	iff	for all y, z s.t. $z \in x \oplus y$: $y \models_{\mathcal{V}} P$ implies $z \models_{\mathcal{V}} Q$
x	$\models_{\mathcal{V}}$	$P \multimap Q$	iff	for all x', y, z s.t. $x' \sqsupseteq x$ and $z \in x' \odot y$: $y \models_{\mathcal{V}} P$ implies $z \models_{\mathcal{V}} Q$
x	$\models_{\mathcal{V}}$	$P \mathrel{-\!\!\circ} Q$	iff	for all x', y, z s.t. $x' \sqsupseteq x$ and $z \in y \odot x'$: $y \models_{\mathcal{V}} P$ implies $z \models_{\mathcal{V}} Q$

Figure 4.3: Satisfaction for DIBI

Definition 4.1.3 (DIBI Satisfaction and Validity)**.** Satisfaction at a state x in a model is inductively defined by the clauses in Figure 4.3. As before, we say P is *valid in a model*, $\mathcal{X} \models_{\mathcal{V}} P$, iff $x \models_{\mathcal{V}} P$ for all $x \in \mathcal{X}$. P is *valid*, $\models P$, iff P is valid in all models. $P \models Q$ iff, for all models, $x \models_{\mathcal{V}} P$ implies $x \models_{\mathcal{V}} Q$.

Where the context is clear, we omit the subscript $\mathcal{V}$ on the satisfaction relation. With the semantics in Figure 4.3, persistence on propositional atoms indeed extends to all formulas:

Lemma 4.1.1 (Persistence Lemma). *For all* $P \in \mathrm{Form}_{\mathrm{DIBI}}$*, if* $x \models P$ *and* $x \sqsubseteq y$*, then* $y \models P$*.*

Proof. We prove that induction on the syntax of the formulas. Specifically, the persistence of $\top$ and $\bot$ is trivial, and the persistent of I follows from Unit Closure. $P \wedge Q$ and $P \vee Q$ are persistent because of their inductive hypothesis. For $P \to Q$, $P * Q$, $P \circ\!\!- Q$, and $P \mathrel{-\!\!\circ} Q$, persistence is evident because their semantic clauses account for the order. □

Notably, in fig. 4.3, the semantic clauses for $\fatsemi$ and $*$ are different even besides that they use different binary operations — the semantic clause for $*$ has an additional variable x' under the existential quantifier and only requires $x \sqsupseteq x' \in y \oplus z$ instead of $x \in y \oplus z$; the semantic clauses for $\mathrel{-\!\!*}$ and $\mathrel{-\!\!\circ}$ are also different — the semantic clause for $\mathrel{-\!\!\circ}$ also uses an additional variable x' under the existential quantifier. This difference is due to the different frame axioms satisfied by $\odot$ and $\oplus$ and our goal to ensure lemma 4.1.1 holds. The satisfaction of $\odot$ Up-Closed frame axiom ensures the persistence of the simpler clause for $\fatsemi$, and similarly $\oplus$ Down-Closed ensures the persistence of $\mathrel{-\!\!*}$ [Cao et al., 2017].

4.1.2 Proof system

Now we describe how DIBI formulas can be derived. We give a Hilbert-style proof system for DIBI in Figure 4.4. This calculus extends the proof system for BI with additional rules governing the new connectives $\fatsemi$, $\mathrel{-\!\!\circ}$, and $\circ\!\!-$. In section 4.1.3, we will prove this calculus is sound and complete. Here we comment on two important details in this proof system.

$$\frac{}{P \vdash P}\ \textsc{Ax} \qquad \frac{}{P \vdash \top}\ \textsc{Top} \qquad \frac{}{\bot \vdash P}\ \textsc{Bot}$$

$$\frac{P \vdash R \qquad Q \vdash R}{P \vee Q \vdash R}\ \vee\text{-E} \qquad \frac{P \vdash Q_i}{P \vdash Q_1 \vee Q_2}\ \vee\text{-I}$$

$$\frac{P \vdash Q \qquad P \vdash R}{P \vdash Q \wedge R}\ \wedge\text{-I-R} \qquad \frac{Q \vdash R}{P \wedge Q \vdash R}\ \wedge\text{-I-L} \qquad \frac{P \vdash Q_1 \wedge Q_2}{P \vdash Q_i}\ \wedge\text{-E}$$

$$\frac{P \wedge Q \vdash R}{P \vdash Q \to R}\ \to\text{-I} \qquad \frac{P \vdash Q \to R \qquad P \vdash Q}{P \vdash R}\ \to\text{-E}$$

$$\frac{P \vdash R \qquad Q \vdash S}{P * Q \vdash R * S}\ *\text{-Conj} \qquad \frac{P * Q \vdash R}{P \vdash Q \mathrel{-\!\!*} R}\ \mathrel{-\!\!*}\text{-I} \qquad \frac{P \vdash Q \mathrel{-\!\!*} R \qquad S \vdash Q}{P * S \vdash R}\ \mathrel{-\!\!*}\text{-E}$$

$$\frac{}{P \dashv\vdash P * I}\ *\text{-Unit} \qquad \frac{}{P * Q \vdash Q * P}\ *\text{-Comm}$$

$$\frac{}{(P * Q) * R \dashv\vdash P * (Q * R)}\ *\text{-Assoc}$$

$$\frac{P \mathbin{⨟} Q \vdash R}{P \vdash Q \mathrel{-\!\!\circ} R}\ \mathrel{-\!\!\circ}\text{-I} \qquad \frac{P \vdash Q \mathrel{-\!\!\circ} R \qquad S \vdash Q}{P \mathbin{⨟} S \vdash R}\ \mathrel{-\!\!\circ}\ \text{MP}$$

$$\frac{P \mathbin{⨟} Q \vdash R}{Q \vdash P \mathrel{\circ\!\!-} R}\ \mathrel{\circ\!\!-}\text{-I} \qquad \frac{P \vdash Q \mathrel{\circ\!\!-} R \qquad S \vdash Q}{S \mathbin{⨟} P \vdash R}\ \mathrel{\circ\!\!-}\ \text{MP} \qquad \frac{}{P \vdash I \mathbin{⨟} P}\ ⨟\text{-Left Unit}$$

$$\frac{}{P \dashv\vdash P \mathbin{⨟} I}\ ⨟\text{-Right Unit} \qquad \frac{P \vdash R \qquad Q \vdash S}{P \mathbin{⨟} Q \vdash R \mathbin{⨟} S}\ ⨟\text{-Conj}$$

$$\frac{}{(P \mathbin{⨟} Q) \mathbin{⨟} R \dashv\vdash P \mathbin{⨟} (Q \mathbin{⨟} R)}\ ⨟\text{-Assoc}$$

$$\frac{}{(P \mathbin{⨟} Q) * (R \mathbin{⨟} S) \vdash (P * R) \mathbin{⨟} (Q * S)}\ \textsc{RevEx}$$

Figure 4.4: Hilbert system for DIBI

Reverse exchange The proof system of DIBI shares many similarities with Concurrent Kleene Bunched Logic (CKBI) [Docherty, 2019], which also extends BI with a non-commutative conjunction. Inspired by concurrent Kleene algebra (CKA) Hoare et al. [2011], CKBI supports the following exchange axiom, derived from CKA's exchange law:

$$\frac{}{(P * R) \fatsemi (Q * S) \vdash_{\text{CKBI}} (P \fatsemi Q) * (R \fatsemi S)}\ \textsc{Exch}$$

In models of CKBI, $*$ describes interleaving concurrent composition, while $\fatsemi$ describes sequential composition. The exchange rule states that the process on the left has *fewer* behaviors than the process on the right — e.g., $P \fatsemi Q$ allows fewer behaviors than $P * Q$, so $P \fatsemi Q \vdash_{\text{CKBI}} P * Q$ is derivable.

In our models, $*$ has a different reading: it states that two computations can be combined because they are *independent* (i.e., non-interfering). Accordingly, DIBI replaces Exch by the *reversed* version RevEx — the fact that the process on the left is *safe* to combine implies that the process on the right is also safe. $P * Q$ is now *stronger* than $P \fatsemi Q$, and instead $P * Q \vdash P \fatsemi Q$ is derivable (Lemma 4.1.2).

Lemma 4.1.2. *In the proof system given by fig. 4.4, $P * Q \vdash P \fatsemi Q$.*

Proof. For better readability, we break the proof tree down into two components.

$$\dfrac{\dfrac{\dfrac{}{P \vdash P \fatsemi I}\ \fatsemi\text{-Right Unit} \quad \dfrac{}{Q \vdash I \fatsemi Q}\ \fatsemi\text{-Left Unit}}{P * Q \vdash (P \fatsemi I) * (I \fatsemi Q)}\ *\text{-Conj} \qquad \dfrac{}{(P \fatsemi I) * (I \fatsemi Q) \vdash (P * I) \fatsemi (I * Q)}\ \textsc{RevEx}}{P * Q \vdash (P * I) \fatsemi (I * Q)}\ \textsc{Cut}$$

With $P * Q \vdash (P * I) \fatsemi (I * Q)$, we construct the following

$$\dfrac{P * Q \vdash (P * I) \fatsemi (I * Q) \qquad \dfrac{\dfrac{}{P * I \vdash P}\ *\text{-Unit} \qquad \dfrac{\dfrac{}{I * Q \vdash Q * I}\ *\text{-Comm} \qquad \dfrac{}{Q * I \vdash Q}\ *\text{-Unit}}{I * Q \vdash Q}\ \text{Cut}}{(P * I) \fatsemi (I * Q) \vdash P \fatsemi Q}\ \fatsemi\text{-Conj}}{P * Q \vdash P \fatsemi Q}\ \text{Cut}$$

This proof uses the admissible rule Cut, which can be derived as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{Q \vdash R}{P \wedge Q \vdash R}\ \wedge 2}{P \vdash Q \to R}\ \to \qquad P \vdash Q}{P \vdash R}\ \text{MP}$$

□

Left unit While $\fatsemi$ has a *right* unit in our logic, it does not have a proper *left* unit. Semantically, this corresponds to the lack of a frame condition

$$e \in E \wedge x \in e \odot y \quad \to \quad x \sqsupseteq y; \qquad (\odot \text{ Coherence}_{\text{L}})$$

in our definition of DIBI frames. This difference can also be seen in our proof rules: while $\fatsemi$-Unit-R gives entailment in both directions, $\fatsemi$-Unit-R only shows entailment in one direction — there is no axiom stating $I \fatsemi P \vdash P$.

We make this relaxation to support our intended model, which we will see in Section 4.2. In a nutshell, states in our models are Markov kernels that *preserve their input through to their output*. Our models take $\odot$ to be Kleisli composition, which exhibits an important asymmetry for such arrows: f can always be recovered from $f \odot e$, but not from arbitrary $e \odot f$. As a result, the set of all kernels naturally serves as the set of right units, but these kernels cannot all serve as left units.[1]

[1] In the special case that e maps the input of f to the Dirac distribution on it, then $e \odot f = f$. But because we also want Unit Closure, which says the set of units is closed under the pre-order $\sqsubseteq$, our unit set E contains other elements g such that f cannot be recovered from $g \odot f$.

4.1.3 Soundness and Completeness of DIBI

The soundness and completeness of DIBI follow the same recipe as before, using the methodology given by Docherty [2019]. First, DIBI is proved sound and complete with respect to an algebraic semantics obtained by interpreting the rules of the proof system as algebraic axioms. We then establish a representation theorem: every DIBI algebra $\mathbb{A}$ embeds into a DIBI algebra generated by a DIBI frame, that is in turn generated by $\mathbb{A}$. Soundness and completeness of the algebraic semantics can then be transferred to the Kripke semantics.

We prove algebraic soundness and completeness of DIBI proof systems with respect to a new structure called DIBI algebra.

Definition 4.1.4 (DIBI Algebra)**.** A *DIBI algebra* is an algebra $\mathbb{A} = (A, \wedge, \vee, \rightarrow, \top, \bot, *, \mathrel{-\!\!*}, \fatsemi, \mathrel{\multimap}, \mathrel{\circ\!\!-}, I)$ such that, for all $a, b, c, d \in A$:

- $(A, \wedge, \vee, \rightarrow, \top, \bot)$ is a Heyting algebra;
- $(A, *, I)$ is a commutative monoid;
- $(A, \fatsemi, I)$ is a *weak monoid*: $\fatsemi$ is an associative operation with right unit I and $a \leq I \fatsemi a$;
- $a * b \leq c$ iff $a \leq b \mathrel{-\!\!*} c$;
- $a \fatsemi b \leq c$ iff $a \leq b \mathrel{\multimap} c$ iff $b \leq a \mathrel{\circ\!\!-} c$;
- $(a \fatsemi b) * (c \fatsemi d) \leq (a * c) \fatsemi (b * d)$.

An *algebraic interpretation* of DIBI is specified by an assignment on the atomic propositions $[\![-]\!] : \mathcal{AP} \rightarrow A$. The interpretation is obtained as the unique homomorphic extension of this assignment, and so we use the notation $[\![-]\!]$

interchangeably for both assignment and interpretation. Soundness and completeness can be established by constructing a term DIBI algebra by quotienting formulas by equiderivability.

Theorem 4.1.3. *$P \vdash Q$ is derivable iff $[\![P]\!] \leq [\![Q]\!]$ for all algebraic interpretations $[\![-]\!]$.*

We now connect these algebras to DIBI frames so we can transfer the soundness and completeness of DIBI proof systems with respect to these algebras to the DIBI frames. Again, we use the notion of complex algebras and prime filters. We denote the set of prime filters of a DIBI algebra $\mathbb{A}$ by $\mathsf{Prf}(\mathbb{A})$.

Definition 4.1.5 (Prime Filter Frame)**.** Given a DIBI algebra $\mathbb{A}$, the *prime filter frame* of $\mathbb{A}$ is defined as $Pr(\mathbb{A}) = (\mathsf{Prf}(\mathbb{A}), \subseteq, \oplus_{\mathbb{A}}, \odot_{\mathbb{A}}, E_{\mathbb{A}})$, where

$$\begin{aligned} F \oplus_{\mathbb{A}} G &= \{H \in \mathsf{Prf}(\mathbb{A}) \mid \forall a \in F, b \in G (a * b \in H)\} \\ F \odot_{\mathbb{A}} G &= \{H \in \mathsf{Prf}(\mathbb{A}) \mid \forall a \in F, b \in G (a \fatsemi b \in H)\} \\ E_{\mathbb{A}} &= \{F \in \mathsf{Prf}(\mathbb{A}) \mid I \in F\}. \end{aligned}$$

Lemma 4.1.4. *For any DIBI algebra $\mathbb{A}$, the prime filter frame $Pr(\mathbb{A})$ is a DIBI frame.*

In the other direction, DIBI frames generate DIBI algebras.

Definition 4.1.6 (Complex Algebra)**.** Given a DIBI frame $\mathcal{X} = (X, \sqsubseteq, \oplus, \odot, E)$, the *complex algebra* of $\mathcal{X}$ is $\mathsf{Com}(\mathcal{X}) = (\mathcal{P}_{\sqsubseteq}(X), \cap, \cup, \Rightarrow_{\mathcal{X}}, X, \emptyset, \bullet_{\mathcal{X}}, \mathrel{-\!\!\bullet}_{\mathcal{X}}, \mathrel{\triangleright}_{\mathcal{X}}, \mathrel{\blacktriangleright}_{\mathcal{X}}, \mathrel{\blacktriangleleft\!\!-}_{\mathcal{X}}, E)$:

$$\begin{aligned} \mathcal{P}_{\sqsubseteq}(X) &= \{A \subseteq X \mid \text{ if } a \in A \text{ and } a \sqsubseteq b \text{ then } b \in A\} \\ A \Rightarrow_{\mathcal{X}} B &= \{a \mid \text{for all } b, \text{ if } b \sqsupseteq a \text{ and } b \in A \text{ then } b \in B\} \\ A \bullet_{\mathcal{X}} B &= \{x \mid \text{there exist } x', a, b \text{ s.t } x \sqsupseteq x' \in a \oplus b, a \in A \text{ and } b \in B\} \\ A \mathrel{-\!\!\bullet}_{\mathcal{X}} B &= \{x \mid \text{for all } a, b, \text{ if } b \in x \oplus a \text{ and } a \in A \text{ then } b \in B\} \\ A \mathrel{\triangleright}_{\mathcal{X}} B &= \{x \mid \text{there exist } a, b \text{ s.t } x \in a \odot b, a \in A \text{ and } b \in B\} \\ A \mathrel{\blacktriangleright}_{\mathcal{X}} B &= \{x \mid \text{for all } x', a, b, \text{ if } x \sqsubseteq x', b \in x' \odot a \text{ and } a \in A \text{ then } b \in B\} \\ A \mathrel{\blacktriangleleft\!\!-}_{\mathcal{X}} B &= \{x \mid \text{for all } x', a, b, \text{ if } x \sqsubseteq x', b \in a \odot x' \text{ and } a \in A \text{ then } b \in B\}. \end{aligned}$$

Lemma 4.1.5. *For any DIBI frame $\mathcal{X}$, the complex algebra $\mathsf{Com}(\mathcal{X})$ is a DIBI algebra.*

The following main result facilitates the transference of soundness and completeness.

Theorem 4.1.6 (Representation of DIBI algebras)**.** *Every DIBI algebra is isomorphic to a subalgebra of a complex algebra: given a DIBI algebra $\mathbb{A}$, the map $\theta_{\mathbb{A}} : \mathbb{A} \to \mathsf{Com}(\mathsf{Prf}(\mathbb{A}))$ defined by $\theta_{\mathbb{A}}(a) = \{F \in \mathsf{Prf}(\mathbb{A}) \mid a \in F\}$ is an embedding.*

Given the previous correspondence between DIBI algebras and frames, we only need to show that θ is a monomorphism: the necessary argument is identical to that for similar bunched logics [Docherty, 2019, Theorems 6.11, 6.25]. Given $[\![-]\!]$ on $\mathbb{A}$, the representation theorem establishes that $\mathcal{V}_{[\![-]\!]}(p) := \theta_{\mathbb{A}}([\![p]\!])$ is a persistent valuation on $Pr(\mathbb{A})$ such that $F \models_{\mathcal{V}_{[\![-]\!]}} P$ iff $[\![P]\!] \in F$, from which our main theorem can be proved.

Theorem 4.1.7 (Soundness and Completeness)**.** *$P \vdash Q$ is derivable iff $P \models Q$.*

4.2 A Probabilistic Model of DIBI

Now we develop a probabilistic model of DIBI where $P * Q$ can assert probabilistic independence and $P \fatsemi Q$ can assert dependence.

Because our DIBI model is designed to describe probabilistic programs' program states, in the remainder of this chapter, we use the term (Markov) kernels to specifically refer to maps $f : \mathbf{Mem}[S] \to \mathcal{D}(\mathbf{Mem}[U])$ with $S, U \subseteq \mathbf{Var}$. For a kernel f, we define its domain $\mathbf{dom}(f) = S$ and its range $\mathbf{range}(f) = U$. We can also project kernels to a smaller range.

Definition 4.2.1 (Marginalizing kernels)**.** For a Markov kernel $f : \mathbf{Mem}[S] \to \mathcal{D}(\mathbf{Mem}[U])$ and $V \subseteq U$, the *marginalization of f to V* is the map $\pi_V f : \mathbf{Mem}[S] \to \mathcal{D}(\mathbf{Mem}[V])$: $(\pi_V f)(d)(r) := \sum_{m \in \mathbf{Mem}[U \setminus V]} f(d)(r \bowtie m)$ for $d \in \mathbf{Mem}[S], r \in \mathbf{Mem}[V]$.

Now we define an important requirement for our DIBI model's states.

Definition 4.2.2. We use unit_S to denote the kernel $g : \mathbf{Mem}[S] \to \mathcal{D}(\mathbf{Mem}[S])$ defined by $g(m) = \mathsf{unit}(m)$ for all $m \in \mathbf{Mem}[S]$. We say a kernel $f : \mathbf{Mem}[S] \to \mathcal{D}(\mathbf{Mem}[U])$ *preserves its input to its output* if $S \subseteq U$ and $\pi_S f = \mathsf{unit}_S$.

Intuitively, kernels that preserve their input to their output are suitable for encoding conditional distributions: once a variable has been conditioned, its value should not change. We define two ways to compose these kernels.

Definition 4.2.3 (Composing Markov kernels on memories)**.** Given $f : \mathbf{Mem}[S] \to \mathcal{D}(\mathbf{Mem}[T])$ and $g : \mathbf{Mem}[U] \to \mathcal{D}(\mathbf{Mem}[V])$ that preserve their inputs, we define their *parallel composition*, whenever $S \cap U = T \cap V$, as the map $f \oplus g : \mathbf{Mem}[S \cup U] \to \mathcal{D}(\mathbf{Mem}[T \cup V])$ given by

$$(f \oplus g)(d)(m) := f(d^S)(m^T) \cdot g(d^U)(m^V).$$

If $T = U$, the *sequential composition* $f \odot g : \mathbf{Mem}[S] \to \mathcal{D}(\mathbf{Mem}[V])$ is just Kleisli composition (eq. (4.1)).

Example 4.2.1 (Kernel decomposition)**.** Recall the distribution μ on $\mathbf{Mem}[\{x, y, z\}]$ from Example 4.0.1. Let $k_x : \mathbf{Mem}[z] \to \mathcal{D}(\mathbf{Mem}[\{x, z\}])$ encode the conditional distribution of x given z, and let $k_y : \mathbf{Mem}[z] \to \mathcal{D}(\mathbf{Mem}[\{y, z\}])$ encode the conditional distribution of y given z. Explicitly, for $\alpha = x$ or y,

$$k_\alpha(z = 0)(\alpha = 1, z = 0) = 1/2 \qquad k_\alpha(z = 0)(\alpha = 0, z = 0) = 1/2$$
$$k_\alpha(z = 1)(\alpha = 1, z = 1) = 1/4 \qquad k_\alpha(z = 1)(\alpha = 0, z = 1) = 3/4.$$

Since k_x, k_y exactly include z in their range, $k_x \oplus k_y$ is defined. A small calculation shows that $k_x \oplus k_y = k$, where $k : \mathbf{Mem}[z] \to \mathcal{D}(\mathbf{Mem}[\{x, y, z\}])$ is the conditional distribution of (x, y, z) given z. This decomposition shows that x and y are independent conditioned on z. The correspondence between the decomposition of kernels and conditional independence is proved in

4.2.1 A Concrete Probabilistic Frame of DIBI

We now have all the ingredients to define a first concrete model: states are Markov kernels that preserve their input; the binary operation $\oplus$ behaves as a parallel composition, and the binary operation $\odot$ serves as the sequential composition. While there is a canonical choice for the sequential composition of Markov kernels, i.e., Kleisli composition, there are many choices for the parallel composition. For instance, it is unclear whether we should only allow parallel composition of kernels with the same domain, or work with a more relaxed condition. Another difficulty is in the definition of the pre-order. We are going to define two very different binary operations, and not only do we need both of their unit sets be closed under the pre-order, we also need the coherence conditions for the pre-order and both binary operations ($\oplus$ Down-Closed, $\odot$ Up-Closed, $\oplus$ Unit Coherence, $\odot$ Coherence$_{\mathrm{R}}$) to hold.

Definition 4.2.4 (Probabilistic frame)**.** We define the frame $(\mathcal{X}_{CI}, \sqsubseteq, \widehat{\oplus}, \widehat{\odot}, \mathcal{X}_{CI})$ as follows:

- $\mathcal{X}_{CI}$ are Markov kernels that preserve their input to their output;
- $\widehat{\oplus}$ and $\widehat{\odot}$ are defined through the parallel and sequential composition of

kernels:

$$f \mathbin{\widehat{\oplus}} g = \begin{cases} \{f \oplus g\} & \text{if } \mathbf{range}(f) \cap \mathbf{range}(g) = \mathbf{dom}(f) \cap \mathbf{dom}(g) \\ \emptyset & \text{otherwise} \end{cases}$$

$$f \mathbin{\widehat{\odot}} g = \begin{cases} \{f \odot g\} & \text{if } \mathbf{range}(f) = \mathbf{dom}(g) \\ \emptyset & \text{otherwise} \end{cases}$$

- Given $f, g \in \mathcal{X}_{CI}$, $f \sqsubseteq g$ if there exist a set of variables $R \subseteq \mathbf{Val}$ and another kernel $h \in \mathcal{X}_{CI}$ such that $g = (f \oplus \mathsf{unit}_R) \odot h$.

We make three remarks. First, the binary combinations $\widehat{\oplus}$ and $\widehat{\odot}$ return sets with at most one element. So they are essentially a wrapper over their underlying operations $\oplus$ and $\odot$, which are partial and deterministic; in the following, including when proving the structure $(\mathcal{X}_{CI}, \sqsubseteq, \widehat{\oplus}, \widehat{\odot}, \mathcal{X}_{CI})$ is a DIBI frame, we will work directly with the underlying operations $\oplus$ and $\odot$.

Second, the definition of $f \odot g$ on $\mathcal{X}_{CI}$ can be simplified. Given $f : \mathbf{Mem}[S] \to \mathcal{D}(\mathbf{Mem}[T])$ and $g : \mathbf{Mem}[T] \to \mathcal{D}(\mathbf{Mem}[V])$, eq. (4.1) yields the formula:

$$(f \odot g)(d)(m) := \sum_{m' \in \mathbf{Mem}[T]} f(d)(m') \cdot g(m')(m).$$

Since $f, g \in \mathcal{X}_{CI}$ preserve input to output, this reduces to

$$(f \odot g)(d)(m) = f(d)(m^T) \cdot g(m^T)(m^V). \tag{4.2}$$

Third, the preorder is defined so that $f \sqsubseteq g$ holds when g can be obtained from extending f. If g is obtained by composing f in parallel with unit_R, and then extending the range via composition with h, then we can recover f from g by marginalizing g to $\mathbf{range}(f) \cup R$, and then ignoring the R portion.

We show that our probabilistic frame is indeed a DIBI frame.

Theorem 4.2.1. *$(\mathcal{X}_{CI}, \sqsubseteq, \widehat{\oplus}, \widehat{\odot}, \mathcal{X}_{CI})$ is a DIBI frame.*

Proof sketch. Since $\widehat{\oplus}$ and $\widehat{\odot}$ returns either a singleton set or an empty set, for any axioms that mention $x \in y \,\widehat{\oplus}\, z$ (resp. $x \in y \,\widehat{\odot}\, z$), we can always use the x such that $x = y \oplus z$ (resp. $x = y \odot z$).

We first show that $\mathcal{X}_{CI}$ is closed under $\oplus$ and $\odot$, and $\sqsubseteq$ is transitive and reflexive. Then we show the frame axioms, which are mostly straightforward. Several conditions rely on a property of our model that we call *Exchange Equality*: if both $(f_1 \oplus f_2) \odot (f_3 \oplus f_4)$ and $(f_1 \odot f_3) \oplus (f_2 \odot f_4)$ are defined, then they are equal, and if the second is defined, then so is the first. While its connection with RevEx is the most obvious, the Exchange Equality is also useful for proving other conditions since the preorder in $\mathcal{X}_{CI}$ is defined through the binary combinations $\oplus$ and $\odot$. For example:

($\oplus$ Unit Coherence): Since the unit set in this frame is the entire state space $\mathcal{X}_{CI}$, we must show that for any $f_1, f_2 \in \mathcal{X}_{CI}$, if $f_1 \oplus f_2$ is defined, then $f_1 \sqsubseteq f_1 \oplus f_2$:

$$\begin{aligned} f_1 \oplus f_2 &= (f_1 \odot \mathsf{unit}_{\mathbf{range}(f_1)}) \oplus (\mathsf{unit}_{\mathbf{dom}(f_2)} \odot f_2) \\ &= (f_1 \oplus \mathsf{unit}_{\mathbf{dom}(f_2)}) \odot (\mathsf{unit}_{\mathbf{range}(f_1)} \oplus f_2) && \text{(By Exchange Equality)} \\ &= (f_1 \oplus \mathsf{unit}_{\mathbf{dom}(f_2)}) \odot (f_2 \oplus \mathsf{unit}_{\mathbf{range}(f_1)}) && \text{(By } \oplus \text{ Commutativity)} \end{aligned}$$

Also, for the commutativity and associativity of the binary combinations, the main difficulty lies in showing that both terms are defined at the same time. In particular, the associativity of $\oplus$ requires $(f \oplus g) \oplus h$ being defined iff $f \oplus (g \oplus h)$ being defined, which takes some non-trivial set manipulations to prove

We present the complete proof in appendix C.1.5 □

4.2.2 Capturing Conditional Independence

Now we return to our original goal: express conditional independence of program variables. For that, we introduce some basic atomic propositions and interpret DIBI formulas on the probabilistic DIBI frame $(\mathcal{X}_{CI}, \sqsubseteq, \widehat{\oplus}, \widehat{\odot}, \mathcal{X}_{CI})$. If the only property we need to express is conditional independence of program variables, we only need atomic propositions in the form of $(A \triangleright B)$, which intends to describe the domain and range of the current kernel.

Definition 4.2.5 (Basic atomic proposition)**.** For sets of variables $A, B \subseteq \mathbf{Var}$, a basic atomic proposition has the form $(A \triangleright B)$ and the semantics:

$$f \models (A \triangleright B) \text{ iff there exists } f' \sqsubseteq f \text{ such that } \mathbf{dom}(f') = A \text{ and } \mathbf{range}(f') \supseteq B.$$

For example, $f : \mathbf{Mem}[y] \to \mathcal{D}(\mathbf{Mem}[y, z])$ defined by $f(y \mapsto v) := \mathsf{unit}(y \mapsto v, z \mapsto v)$ satisfies $(y \triangleright y)$, $(y \triangleright z)$, $(y \triangleright \emptyset)$, $(y \triangleright y, z)$, $(\emptyset \triangleright \emptyset)$, and no other atomic propositions.

With these atomic propositions, we can assert conditional independence of program variables using a simple formula:

Theorem 4.2.2. *Given distribution $\mu \in \mathcal{D}(\mathbf{Mem}[\mathbf{Var}])$, then for any $X, Y, Z \subseteq \mathbf{Var}$,*

$$f_\mu \models (\emptyset \triangleright Z) \fatsemi (Z \triangleright X) * (Z \triangleright Y) \tag{4.3}$$

if and only if $X \perp\!\!\!\perp Y \mid Z$ and $X \cap Y \subseteq Z$ are both satisfied.

We prove it in the restriction $X \cap Y \subseteq Z$ is harmless: when $X \perp\!\!\!\perp Y \mid Z$ but $X \cap Y \not\subseteq Z$, then the variables in $X \cap Y$ must be determined by variables in Z

(see lemma C.2.7), and it suffices to check $X \perp\!\!\!\perp Y \mid Z \cup (X \cap Y)$. For simplicity, we abbreviate the formula $(\emptyset \triangleright Z) \fatsemi ((Z \triangleright X) * (Z \triangleright Y))$ as $[Z] \fatsemi ([X] * [Y])$.

Proof sketch. For the forward direction, suppose f_μ satisfies 4.3. We first show in lemma C.2.6 that this intuitionistic logic has some classical flavor: whenever f_μ satisfies 4.3 there exist f, g, and h in $\mathcal{X}_{CI}$ with $f \odot (g \oplus h) \sqsubseteq f_\mu$, where $f : \mathbf{Mem}[\emptyset] \to \mathcal{D}(\mathbf{Mem}[Z])$, $g : \mathbf{Mem}[Z] \to \mathcal{D}(\mathbf{Mem}[Z \cup X])$, and $h : \mathbf{Mem}[Z] \to \mathcal{D}(\mathbf{Mem}[Z \cup Y])$; we also have $X \cap Y \subseteq Z$ as $f \odot (g \oplus h)$ is defined. Since $\mathbf{dom}(f_\mu) = \mathbf{Mem}[\emptyset]$, $f \odot (g \oplus h) \sqsubseteq f_\mu$ implies:

$$f \odot (g \oplus h) = \pi_{Z \cup X \cup Y} f_\mu \quad \text{and} \qquad f = \pi_Z f_\mu.$$

Further, we can show that $f \odot (g \oplus h) = f \odot g \odot (\mathsf{unit}_X \oplus h) = f \odot h \odot (\mathsf{unit}_Y \oplus g)$, and thus:

$$f \odot g = \pi_{Z \cup X} f_\mu \quad \text{and} \quad f \odot h = \pi_{Z \cup Y} f_\mu.$$

These imply that g (resp. h) encodes the conditional distributions of X (resp. Y) given Z, and $g \oplus h$ encodes the conditional distribution of (X, Y) given Z. Hence, $f \odot (g \oplus h) \sqsubseteq f_\mu$ implies that the conditional distribution of (X, Y) given Z is equal to the product distribution of X given Z and Y given Z, and so $X \perp\!\!\!\perp Y \mid Z$ holds in μ.

For the reverse direction, suppose that $X \perp\!\!\!\perp Y \mid Z$ holds in μ and $X \cap Y \subseteq Z$. Now, consider $\pi_{X \cup Y \cup Z} f_\mu$, the marginal distribution on (X, Y, Z) encoded as a kernel, and observe that $\pi_{X,Y,Z} f_\mu = f \odot f'$, where f encodes the marginal distribution of Z, and f' is the conditional distribution of (X, Y) given values of Z. From (a), the conditional distribution of (X, Y) given Z is the product of the conditional distributions of X given Z, and Y given Z, that is $f' = g \oplus h$, where g (resp. h) encode the conditional distribution of X (resp. Y) given Z. Then by (b),

$f \odot (g \oplus h)$ is defined and $f \odot (g \oplus h) = \pi_{X \cup Y \cup Z} f_\mu \sqsubseteq f_\mu$. It is straightforward to see that $f \odot (g \oplus h)$ satisfies $[Z] \fatsemi ([X] * [Y])$. Hence, persistence shows that f_μ also satisfies $[Z] \fatsemi ([X] * [Y])$.

See lemma C.2.8 for details. □

4.2.3 Validating the Semi-graphoid Axioms

Notions analogous to conditional independence are useful in different domains. For instance, in database theory [Abiteboul et al., 1995], join dependency, which can be seen as conditional independence for *powersets* instead of distributions, allows more efficient storage and querying of relational databases [Fagin and Vardi, 1984]. There is a long line of research on logical characterizations of conditional independence and join dependency. *Graphoids* is perhaps the most well-known approach [Pearl and Paz, 1985]; later, Dawid [2001] has a similar notion called *separoids*. Here, we focus on graphoids.

Definition 4.2.6 (Graphoids and semi-graphoids)**.** Suppose that $I(X, Z, Y)$ is a ternary relation on subsets of **Var** (i.e., $X, Z, Y \subseteq$ **Var**). Then the relation I is a *graphoid* if it satisfies:

$$I(X, Z, Y) \Leftrightarrow I(Y, Z, X) \qquad \text{(Symmetry)}$$

$$I(X, Z, Y \cup W) \Rightarrow I(X, Z, Y) \wedge I(X, Z, W) \qquad \text{(Decomposition)}$$

$$I(X, Z, Y \cup W) \Rightarrow I(X, Z \cup W, Y) \qquad \text{(Weak Union)}$$

$$I(X, Z, Y) \wedge I(X, Z \cup Y, W) \Leftrightarrow I(X, Z, Y \cup W) \qquad \text{(Contraction)}$$

$$I(X, Z \cup W, Y) \wedge I(X, Z \cup Y, W) \Rightarrow I(X, Z, Y \cup W) \qquad \text{(Intersection)}$$

If I satisfies the first four properties, then it is a *semi-graphoid*.

Because $I(X, Z, Y)$ intends to capture CI-like notions, these conditions aims at axiomatizing the relation "knowing Z renders X irrelevant to Y." As an example, it is known that conditional independence relation forms a semi-graphoid: if we fix a distribution over $\mu \in \mathcal{D}(\mathbf{Mem}[\mathbf{Var}])$, then taking $I(X, Z, Y)$ to be the set of triples such that $X \perp\!\!\!\perp Y \mid Z$ holds in μ defines a semi-graphoid.

Below, we show that the semi-graphoid axioms can be naturally translated into valid formulas in our probabilistic model.

Theorem 4.2.3. *We abbreviate our probabilistic model as M. Define $I(X, Z, Y)$ iff $M \models [Z] \fatsemi ([X] * [Y])$. Then,* SYMMETRY, DECOMPOSITION, WEAK UNION, *and* CONTRACTION *are valid. Furthermore,* SYMMETRY *is derivable in the proof system, and* DECOMPOSITION *is derivable given the following axiom, valid in M:*

$$(Z \triangleright Y \cup W) \leftrightarrow (Z \triangleright Y) \wedge (Z \triangleright W) \tag{SPLIT}$$

Proof sketch. We show the proof for the derivable axioms. To derive SYMMETRY, we use the $*$-COMM rule to commute the separating conjunction.

$$\dfrac{\dfrac{\dfrac{}{P \vdash P}\,\text{AX} \quad \dfrac{}{Q * R \vdash R * Q}\,*\text{-COMM}}{P \fatsemi (Q * R) \vdash P \fatsemi (R * Q)}\,\fatsemi\text{-Conj}}{\vdash P \fatsemi (Q * R) \to P \fatsemi (R * Q)}\,\to$$

The proof of DECOMPOSITION uses the axiom SPLIT to split up $Y \cup W$, and then uses proof rules to derive the following.

$$\dfrac{\dfrac{\dfrac{\text{AX}\,\dfrac{}{P \vdash P} \qquad \dfrac{\text{AX}\,\dfrac{}{Q \vdash Q} \qquad \dfrac{\text{AX}\,\dfrac{}{R \wedge S \vdash R \wedge S}}{R \wedge S \vdash R}\,\wedge 3}{Q * (R \wedge S) \vdash Q * R}\,*\text{-CONJ}}{P \fatsemi (Q * (R \wedge S)) \vdash P \fatsemi (Q * R)}\,\fatsemi\text{-CONJ}}{P \fatsemi (Q * (R \wedge S)) \vdash P \fatsemi (Q * R) \wedge P \fatsemi (Q * S)}\,\wedge 1 \qquad \dfrac{\text{Similar to left}}{P \fatsemi (Q * (R \wedge S)) \vdash P \fatsemi (Q * S)}}{\vdash P \fatsemi (Q * (R \wedge S)) \to P \fatsemi (Q * R) \wedge P \fatsemi (Q * S)}\,\to$$

Thus, as an instance

$$\vdash (\emptyset \triangleright Z) \fatsemi ((Z \triangleright X) * ((Z \triangleright Y) \wedge (Z \triangleright W)))$$
$$\rightarrow (\emptyset \triangleright Z) \fatsemi ((Z \triangleright X) * (Z \triangleright Y)) \wedge (\emptyset \triangleright Z) \fatsemi ((Z \triangleright X) * (Z \triangleright W))$$

Combine that with eq. (SPLIT), we have

$$\vdash (\emptyset \triangleright Z) \fatsemi ((Z \triangleright X) * ((Z \triangleright Y \cup W)))$$
$$\rightarrow (\emptyset \triangleright Z) \fatsemi ((Z \triangleright X) * (Z \triangleright Y)) \wedge (\emptyset \triangleright Z) \fatsemi ((Z \triangleright X) * (Z \triangleright W))$$

We prove validity of WEAK UNION and CONTRACTION in appendix C.2.3.

□

Our conference paper Bao et al. [2021] in addition introduces a relational model of DIBI, where $[Z] \fatsemi ([X] * [Y])$ asserts join dependency, and the semi-graphoid axioms can be translated into valid formulas in the relational model as well.

4.3 Conditional Probabilistic Separation Logic

Conditional independence of program variables can be subtle to reason about, motivating formal methods for proving it. We design a program logic CPSL for formally proving conditional independence in a simplified probabilistic imperative language. The language has assignments, sampling, sequencing, and conditionals, but no loops, which would make the reasoning even trickier. Here, our goal is to simply show how a DIBI-based program logic could work in a basic setting.

4.3.1 CPSL: Assertion Logic

Like PSL and LINA, CPSL is constructed in two layers: the *assertion logic* describes program states — probability distributions here — while the *program logic* describes probabilistic programs, using the assertion logic to specify pre- and post-conditions. Our starting point for the assertion logic is the probabilistic model of DIBI introduced in section 4.2, with atomic assertions we introduced to assert conditional independence in section 4.2.2. We encode distributions as Markov kernels with domain $\mathbf{Mem}[\emptyset]$ in order to interpret DIBI on program states. However, it turns out that the full logic DIBI is not suitable for a program logic. The main problem is that not all formulas in DIBI satisfy a key technical condition, the *restriction* property.

Definition 4.3.1 (Restriction)**.** A formula P satisfies *restriction* if: a Markov kernel f satisfies P iff there exists $f' \sqsubseteq f$ such that $\mathbf{range}(f') \subseteq \mathrm{FV}(P)$ and $f' \models P$.

A similar restriction property plays an important role in the soundness of Frame-like rules in PSL and LINA because formulas satisfying restriction are preserved if the program does not modify variables appearing in the formula. Here, we also need it, not only to prove FRAME but also to reason about how the preconditions are preserved in ASSN, SAMP and COND.

Thus, we want to show that the restriction property holds for DIBI formulas. The reverse direction is immediate by persistence, but the forward direction is more delicate – there are simple formulas where restriction fails.

Example 4.3.1 (Failure of restriction)**.** Consider the kernel $f : \mathbf{Mem}[z] \to \mathcal{D}(\mathbf{Mem}[x, z])$ with $f(z \mapsto c) := \mathsf{unit}(x \mapsto c, z \mapsto c)$. We can show that f satisfies the formula $\varphi := \top \fatsemi (x \triangleright \mathsf{Own}(x))$: letting $f_1 : \mathbf{Mem}[z] \to \mathcal{D}(\mathbf{Mem}[x, z])$ and

$f_2 \colon \mathbf{Mem}[x, z] \to \mathcal{D}(\mathbf{Mem}[x, z])$ with

$$f_1(z \mapsto c) := \mathsf{unit}(x \mapsto c, z \mapsto c)$$

$$f_2 := \mathsf{unit}_{\mathbf{Mem}[x]} \oplus \mathsf{unit}_{\mathbf{Mem}[z]}$$

then we have $f_1 \models \top$ and $f_2 \models (x \triangleright [x])$. Also, $f = f_1 \odot f_2$, so

$$f \models \top \fatsemi (x \triangleright [x]).$$

Since $\mathrm{FV}(\varphi) = \{x\}$, any subkernel $f' \sqsubseteq f$ simultaneously satisfying φ and witnessing restriction must be of type $f' : \mathbf{Mem}[z] \to \mathcal{D}(\mathbf{Mem}[x])$, but there are no input-preserving kernels of that type.

To address this problem, we will identify a fragment of DIBI that satisfies restriction and is sufficiently rich to support an interesting program logic. Intuitively, restriction may fail for φ when the satisfaction of φ implicitly requires unexpected variables in the domain of the kernel, or φ does not describe needed variables in its range. Thus, we employ syntactic conditions to over-approximate variables that can appear in the domain of a kernel satisfying φ as $\mathrm{FV}_{\mathrm{D}}(\varphi)$ and under-approximate variables that can appear in the range as $\mathrm{FV}_{\mathrm{R}}(\varphi)$.

Definition 4.3.2 (FV_{D} and FV_{R})**.** For DIBI formulas generated by probabilistic atomic propositions, conjunctions ($\wedge$, $*$, $\fatsemi$) and disjunction ($\vee$), we define two

sets of variables:

$$\begin{aligned}
\mathrm{FV}_{\mathrm{D}}(\top) = \mathrm{FV}_{\mathrm{D}}(\bot) &:= \emptyset & \mathrm{FV}_{\mathrm{R}}(\top) = \mathrm{FV}_{\mathrm{R}}(\bot) &:= \emptyset \\
\mathrm{FV}_{\mathrm{D}}(A \triangleright B) &:= \mathrm{FV}(A) & \mathrm{FV}_{\mathrm{R}}(A \triangleright B) &:= \mathrm{FV}(A) \cup \mathrm{FV}(B) \\
\mathrm{FV}_{\mathrm{D}}(P \wedge Q) &:= \mathrm{FV}_{\mathrm{D}}(P) \cup \mathrm{FV}_{\mathrm{D}}(Q) & \mathrm{FV}_{\mathrm{R}}(P \wedge Q) &:= \mathrm{FV}_{\mathrm{R}}(P) \cup \mathrm{FV}_{\mathrm{R}}(Q) \\
\mathrm{FV}_{\mathrm{D}}(P * Q) &:= \mathrm{FV}_{\mathrm{D}}(P) \cup \mathrm{FV}_{\mathrm{D}}(Q) & \mathrm{FV}_{\mathrm{R}}(P * Q) &:= \mathrm{FV}_{\mathrm{R}}(P) \cup \mathrm{FV}_{\mathrm{R}}(Q) \\
\mathrm{FV}_{\mathrm{D}}(P \fatsemi Q) &:= \mathrm{FV}_{\mathrm{D}}(P) \cup \mathrm{FV}_{\mathrm{D}}(Q) & \mathrm{FV}_{\mathrm{R}}(P \fatsemi Q) &:= \mathrm{FV}_{\mathrm{R}}(P) \cup \mathrm{FV}_{\mathrm{R}}(Q) \\
\mathrm{FV}_{\mathrm{D}}(P \vee Q) &:= \mathrm{FV}_{\mathrm{D}}(P) \cup \mathrm{FV}_{\mathrm{D}}(Q) & \mathrm{FV}_{\mathrm{R}}(P \vee Q) &:= \mathrm{FV}_{\mathrm{R}}(P) \cap \mathrm{FV}_{\mathrm{R}}(Q)
\end{aligned}$$

Now, we have all the ingredients to introduce our assertions. The logic DIBI_+ is a fragment of DIBI with atomic propositions $\mathcal{AP}$, with formulas DIBI_+ defined by the following grammar:

$$\begin{aligned}
P, Q ::= {} & \mathcal{AP} \mid \top \mid \bot \mid P \vee Q \mid P * Q \\
& \mid P \fatsemi Q \quad (\mathrm{FV}_{\mathrm{D}}(Q) \subseteq \mathrm{FV}_{\mathrm{R}}(P)) \\
& \mid P \wedge Q \quad (\mathrm{FV}_{\mathrm{R}}(P) = \mathrm{FV}_{\mathrm{R}}(Q) = \mathrm{FV}(P) = \mathrm{FV}(Q)).
\end{aligned}$$

The side-condition for $P \fatsemi Q$ ensures that variables used by Q are described by P. The side-condition for $P \wedge Q$ is the most restrictive — to understand why we need it, consider the following example.

Example 4.3.2 (Failure of restriction for And)**.** Consider the formula $P := (\emptyset \triangleright \{x\}) \wedge (\emptyset \triangleright \{y\})$, and kernel $f : \mathbf{Mem}[z] \to \mathcal{D}(\mathbf{Mem}[x, y, z])$ with $f(z \mapsto tt)$ being the distribution with x a fair coin flip, $y = x$, and $z = tt$, and $f(z \mapsto ff)$ being the distribution with x a fair coin flip, $y = \neg x$, and $z = ff$.

Then, there exist $f_1 : \mathbf{Mem}[\emptyset] \to \mathcal{D}(\mathbf{Mem}[x])$ and $f_2 : \mathbf{Mem}[\emptyset] \to \mathcal{D}(\mathbf{Mem}[y])$ such that $f_1 \sqsubseteq f$ and $f_2 \sqsubseteq f$. Since $f_1 \models (\emptyset \triangleright \{x\})$ and $f_2 \models (\emptyset \triangleright \{y\})$, it follows $f \models P$. But, because z is correlated with (x, y), there is no kernel $f' : \mathbf{Mem}[\emptyset] \to$

$\mathcal{D}(\mathbf{Mem}[x, y])$ satisfying P such that $f' \sqsubseteq f$ because that means f can be obtained by parallel combination of f' with another kernel with domain $\{z\}$, which requires them to be independent.

With atomic propositions introduced to express conditional independence, i.e., $(A \triangleright B)$ where $A, B \subseteq \mathbf{Var}$, all formulas in DIBI_+ satisfy the restriction property. But before proving the restriction property for DIBI_+, we enrich the atomic propositions to describe more fine-grained information about the domain and range of kernels, and then show that DIBI_+ with the enriched set of atomic propositions still satisfies the restriction property. In particular, we want to enrich the atomic propositions in the following ways.

Domain. Given a kernel f, the existing atomic propositions $(A \triangleright B)$ can only describe properties that hold for all (well-typed) inputs m to f. We would like to be able to describe properties that hold for only certain inputs, e.g., for memories m where a variable z is true.

Range. Given any input m to a kernel f, the existing atomic propositions can only guarantee the presence of variables in the output distribution $f(m)$. We would like to describe more precise information about $f(m)$, e.g., that certain variables are independent conditioned on a *particular* value of m, rather than on all values of m.

Thus, we extend atomic propositions to all pairs of logical formula $(\phi \triangleright \psi)$, where ϕ is a logical formula over the kernel domain (i.e., memories), while ψ is a logical formula over the kernel range (i.e., distributions over memories).

To describe memories, we take a simple propositional logic.

Definition 4.3.3 (Domain logic)**.** The *domain logic* has formulas ϕ of the form $S : p_d$, where $S \subseteq \mathbf{Var}$ is a subset of variables and

$$p_d ::= [e_1 = e_2] \mid \top \mid \bot \mid p_d \wedge p'_d \mid p_d \vee p'_d.$$

A formula $S : p_d$ is *satisfied* in by a memory m, written $m \models_d S : p_d$, if $\mathbf{dom}(m) = S$ and p_d holds in m. In particular, $[e_1 = e_2]$ holds in m iff $[\![e_1]\!](m) = [\![e_2]\!](m)$.

We read $S : p_d$ as "memories over S such that p_d" and abbreviate $S : \top$ as S.

To describe distributions over memories, we adapt formulas in probabilistic BI for the range logic.

Definition 4.3.4 (Range logic)**.** The *range logic* has the following formulas from probabilistic BI:

$$p_r ::= [S] \mid x \,\$\, d \mid [x = e] \mid \top \mid \bot \mid p_r \wedge p'_r \mid p_r * p'_r.$$

We give a semantics where states are distributions over memories:

$$M_r = \{\mu : \mathcal{D}(\mathbf{Mem}[S]) \mid S \subseteq \mathbf{Var}\}.$$

We define a preorder on states via $\mu_1 \sqsubseteq_r \mu_2$ if and only if $\mathbf{dom}(\mu_1) \subseteq \mathbf{dom}(\mu_2)$ and $\pi_{\mathbf{dom}(\mu_1)}\mu_2 = \mu_1$, and we define a partial binary operation on states: for any $\mu_1 \in \mathcal{D}(\mathbf{Mem}[S])$ and $\mu_2 \in \mathcal{D}(\mathbf{Mem}[T])$,

$$\mu_1 \oplus_r \mu_2 := \begin{cases} \{\pi_{S\setminus T}\mu_1 \otimes \delta_m \otimes \pi_{T\setminus S}\mu_2\} & \text{if } \exists m \in \mathbf{Mem}[S \cap T] \text{ s.t. } \pi_{S\cap T}\mu_1 = \pi_{S\cap T}\mu_2 = \delta_m \\ \{\} & \text{otherwise} \end{cases}$$

where $\otimes$ takes the independent product of two distributions over disjoint domains. That is, for any $x \in \mathcal{D}(\mathbf{Mem}[S \cup T])$,

$$(\pi_{S\setminus T}\mu_1 \otimes \delta_m \otimes \pi_{T\setminus S}\mu_2)(x) := \pi_{S\setminus T}\mu_1(\pi_{S\setminus T}x) \cdot 1 \cdot \pi_{T\setminus S}\mu_2(\pi_{T\setminus S}x)$$

This operation generalizes the monoid from the probabilistic BI frame to allow for combining distributions with overlapping domains if the distributions over the overlap is deterministic and equal; this mild generalization is useful for our setting, where distributions often have deterministic variables (e.g., variables corresponding to the input of kernels).

Then, we define the semantics of the range logic as:

$\mu \models_r \top$	always
$\mu \models_r \bot$	never
$\mu \models_r [s]$	iff $s \subseteq \mathbf{dom}(\mu)$ or $s \in \mathbf{dom}(\mu)$
$\mu \models_r e \mathbin{\$} d$	iff $\mathrm{FV}(e) \subseteq \mathbf{dom}(\mu)$ and $[\![e]\!](\mu) = d$
$\mu \models_r [e_1 = e_2]$	iff $\mathrm{FV}(e_1) \cup \mathrm{FV}(e_2) \subseteq \mathbf{dom}(\mu)$ and $[\![e]\!](m) = [\![e']\!](m)$ for any m in the support of μ
$\mu \models_r p_r \wedge p'_r$	iff $\mu \models_r p_r$ and $\mu \models_r p'_r$
$\mu \models_r p_r * p'_r$	iff there exists $\mu_1 \oplus_r \mu_2 \sqsubseteq \mu$ with $\mu_1 \models_r p_r$ and $\mu_2 \models_r p'_r$.

We only use domain formulas ϕ and range formulas ψ in the enriched atomic propositions of the form $(\phi \triangleright \psi)$, so we do not need to show formulas in the domain logic are persistent and similarly formulas in the range logic are persistent. Now, we can give a semantics to our enriched atomic propositions.

Definition 4.3.5. Given a kernel f and atomic proposition $(\phi \triangleright \psi)$, we define $f \models (\phi \triangleright \psi)$ iff there exists $f' \sqsubseteq f$ such that $m \models_d \phi$ implies $m \in \mathbf{dom}(f')$ and $f(m) \models_r \psi$.

This valuation is persistent by construction. Furthermore, formulas in DIBI_+ with these atomic propositions satisfy restriction.

Theorem 4.3.1 (Restriction in DIBI_+)**.** *Let $P \in DIBI_+$ with atomic propositions $(\phi \triangleright$*

ψ), *as described above. Then* $f \models P$ *if and only if there exists* $f' \sqsubseteq f$ *such that* $\boldsymbol{range}(f') \subseteq FV(P)$ *and* $f' \models P$.

Proof sketch. We prove a stronger statement by induction on P: $f \models P$ if and only if there exists $f' \sqsubseteq f$ such that $\mathbf{dom}(f') \subseteq \mathrm{FV}_{\mathrm{D}}(P)$, and $\mathrm{FV}_{\mathrm{R}}(P) \subseteq \mathbf{range}(f') \subseteq \mathrm{FV}(P)$. □

Last, atomic propositions satisfy some axiom schemas, inspired by proof rules of BI.

Proposition 4.3.2. *The following axiom schemas for atomic propositions are valid.*

$$(S : p_d \triangleright p_r) \wedge (S : p'_d \triangleright p'_r) \to (S : p_d \wedge p'_d \triangleright p_r \wedge p'_r) \qquad \text{if } FV(p_r) = FV(p'_r) \tag{AP-And}$$

$$(S : p_d \triangleright p_r) \wedge (S : p'_d \triangleright p'_r) \to (S : p_d \vee p'_d \triangleright p_r \vee p'_r) \tag{AP-Or}$$

$$(S : p_d \triangleright p_r) * (S' : p'_d \triangleright p'_r) \to (S \cup S' : p_d \wedge p'_d \triangleright p_r * p'_r) \tag{AP-Par}$$

$$p'_d \to p_d \text{ and } \models_r p_r \to p'_r \text{ implies } \models (S : p_d \triangleright p_r) \to (S : p'_d \triangleright p'_r) \tag{AP-Imp}$$

We omitted the proofs to appendix C.3.2.

4.3.2 Conditional Probabilistic Separation Logic (CPSL)

With the assertion logic set, we are now ready to introduce our program logic. We call it Conditional Probabilistic Separation Logic, abbreviated as CPSL. Judgments in CPSL have the form $\{P\}\ c\ \{Q\}$, where c is a loopless probabilistic program in pWhile and $P, Q \in \mathrm{DIBI}_+$ are restricted assertions serving as the pre- and post-conditions. As usual, a judgment holds if the program in the judgment

$$\textsc{Assn}\ \frac{x \notin \mathrm{FV}(e) \cup \mathrm{FV}(P)}{\vdash \{P\}\ x \leftarrow e\ \{P \fatsemi (\mathrm{FV}(e) \triangleright [x = e])\}} \qquad \textsc{Samp}\ \frac{x \notin \mathrm{FV}(P)}{\vdash \{P\}\ x \xleftarrow{\$} d\ \{P \fatsemi (\emptyset \triangleright x \$ d)\}}$$

$$\textsc{Skip}\ \frac{}{\vdash \{P\}\ \mathbf{skip}\ \{P\}} \qquad \textsc{Seqn}\ \frac{\vdash \{P\}\ c\ \{Q\} \qquad \vdash \{Q\}\ c'\ \{R\}}{\vdash \{P\}\ c\ ;\ c'\ \{R\}}$$

$$\textsc{Cond}\ \frac{\begin{array}{c}\vdash \{(\emptyset \triangleright [b = tt]) \fatsemi P\}\ c\ \{(\emptyset \triangleright [b = tt]) \fatsemi (b : [b = tt] \triangleright Q_1)\} \\ \vdash \{(\emptyset \triangleright [b = ff]) \fatsemi P\}\ c'\ \{(\emptyset \triangleright [b = ff]) \fatsemi (b : [b = ff] \triangleright Q_2)\}\end{array}}{\vdash \{(\emptyset \triangleright [b]) \fatsemi P\}\ \mathbf{if}\ b\ \mathbf{then}\ c\ \mathbf{else}\ c'\ \{(\emptyset \triangleright [b]) \fatsemi ((b : [b = tt] \triangleright Q_1) \wedge (b : [b = ff] \triangleright Q_2))\}}$$

$$\textsc{Weak}\ \frac{\begin{array}{c}\vdash \{P\}\ c\ \{Q\} \\ \models P' \to P \wedge Q \to Q'\end{array}}{\vdash \{P'\}\ c\ \{Q'\}} \qquad \textsc{Frame}\ \frac{\begin{array}{cc}\vdash \{P\}\ c\ \{Q\} & \mathrm{FV}(R) \cap \mathrm{MV}(c) = \emptyset \\ \mathrm{FV}(Q) \subseteq \mathrm{FV}_{\mathrm{R}}(P) \cup \mathrm{WV}(c) & \mathrm{RV}(c) \subseteq \mathrm{FV}_{\mathrm{R}}(P)\end{array}}{\vdash \{P * R\}\ c\ \{Q * R\}}$$

Figure 4.5: Proof rules: CPSL

maps states satisfying the pre-condition to states satisfying the post-condition. One small difference is that DIBI_+ formulas are interpreted on kernels while the program states are distributions — the mismatch is handled by the natural lifting of the distributions to kernels.

Definition 4.3.6 (CPSL Validity)**.** A CPSL judgment $\{P\}\ c\ \{Q\}$ is *valid*, written $\models \{P\}\ c\ \{Q\}$, if for every input distribution $\mu \in \mathcal{D}(\mathbf{Mem}[\mathbf{Var}])$ such that the lifted input $f_\mu \triangleq \langle\rangle \mapsto \mu$ satisfies $f_\mu \models P$, the lifted output satisfies $f_{[\![c]\!]\mu} \models Q$.

The proof rules of CPSL are presented in Figure 4.5. Note that the requirement that the assertions in the judgments are in DIBI_+ poses implicit side conditions. For example, the rule Assn requires that the post-condition $P \fatsemi (\mathrm{FV}(e) \triangleright x = e)$ is a formula in DIBI_+, which in turn requires that $\mathrm{FV}(e) \subseteq \mathrm{FV}_{\mathrm{R}}(P)$.

The rules Skip, Seq, Weak are standard, we comment on the other, more interesting rules. Assn and Samp allow forward reasoning across assignments and random sampling commands. In both cases, a pre-condition that does not mention the assigned variable x is augmented with new information tracking

the value or distribution of x, and variables x may depend on.

COND allows reasoning about probabilistic control flow, and the ensuing conditional dependence that may result. The main pre-condition P is allowed to depend on the guard variable b but nothing else — because we need $\mathrm{FV}_{\mathrm{D}}(P) \subseteq \mathrm{FV}_{\mathrm{R}}(\emptyset \triangleright [b])$ for the formula to be in DIBI_{+} — and P is preserved as a pre-condition for both branches. The post-conditions allows introducing new facts $(b : b = tt \triangleright Q_1)$ and $(b : b = ff \triangleright Q_2)$, which are then combined in the post-condition of the entire conditional command. As in PSL, the rule for conditionals does not allow the branches to modify the guard b — this restriction is needed to accurately associate each post-condition to each branch.

Finally, FRAME is the frame rule for CPSL. Much like in PSL, the rule involves three classes of variables: $\mathrm{MV}(c)$ is the set of variables that c may write to, $\mathrm{RV}(c)$ is the set of variables that c may read from the input, and $\mathrm{WV}(c)$ is the set of variables that c must write to; these variable sets are defined as in definition 2.3.8. Then, the first side-condition $\mathrm{FV}(R) \cap \mathrm{MV}(c) = \emptyset$ of FRAME ensures that the framing condition is not modified, which is a fairly standard condition in frame-like rules. The second and third side-conditions are more specialized. Observe that the variables described by Q in the post-condition are either already described by P in the pre-condition, or are written by c. These two side conditions ensure that variables mentioned by Q that were not already independent of R are freshly written, and freshly written variables are computed using variables that were already independent of R in the precondition, which can be guaranteed if the variables c reads from are all in $\mathrm{FV}(P)$.

Theorem 4.3.3 (CPSL Soundness)**.** *CPSL is sound: derivable judgments are valid.*

Proof sketch. By induction on the proof derivation. The restriction property is

$z \xleftarrow{\$} \textbf{Bern}_{1/2}$;
$x \xleftarrow{\$} \textbf{Bern}_{1/2}$;
$y \xleftarrow{\$} \textbf{Bern}_{1/2}$;
$a \leftarrow x \vee z$;
$b \leftarrow y \vee z$

(a) CommonCause

$z \xleftarrow{\$} \textbf{Bern}_{1/2}$;
if z **then**
    $x \xleftarrow{\$} \textbf{Bern}_p$; $y \xleftarrow{\$} \textbf{Bern}_p$
else
    $x \xleftarrow{\$} \textbf{Bern}_q$; $y \xleftarrow{\$} \textbf{Bern}_q$

(b) CondSamples

Figure 4.6: Example programs

used repeatedly to constrain the domains and ranges of kernels witnessing different sub-assertions, ensuring that pre-conditions about unmodified variables continue to hold in the post-condition. □

We include the full proof in appendix C.4.

4.3.3 Example: CPSL in Action

Now, we demonstrate CPSL on two example programs.

Example 4.3.3. Figure 4.6 introduces two more example programs. The program CommonCause (Figure 4.6a) generates a distribution where two random observations share a common cause. Specifically, z, x, and y are independent random samples, and a and b are values computed from (x, z) and (y, z), respectively. Intuitively, z, x, and y could represent independent noisy measurements, while a and b could represent quantities derived from these measurements. Since a and b share a common source of randomness z, they are not independent. However, a and b are independent conditioned on the value of z; this is a textbook example of conditional independence.

The program CondSamples (Figure 4.6b) is a bit more complex: it branches

on a random value z, and then assigns x and y with two independent samples from $\mathbf{Bern}_p$ in the true branch, and $\mathbf{Bern}_q$ in the false branch (p, q are constant value in $[0, 1]$). While we might think that x and y are independent at the end of the program since they are independent at the end of each branch, this is not true because their distributions are different in the two branches. For example, suppose that $p = 1$ and $q = 0$. Then at the end of the first branch $(x, y) = (tt, tt)$ with probability 1, while at the end of the second branch $(x, y) = (ff, ff)$ with probability 1. Thus, observing whether $x = tt$ or $x = ff$ determines the value of y — clearly, x and y can't be independent. However, x and y *are* independent conditioned on z.

In both cases, we will prove a conditional independence assertion as the post-condition. We will need some axioms for implications between formulas in DIBI_+. The following axioms are valid in our probabilistic model $\mathcal{X}_{CI}$.

Proposition 4.3.4. *(Axioms for DIBI_+) The following axioms are sound, assuming both precedent and antecedent are in DIBI_+.*

$$(P \fatsemi Q) \fatsemi R \to P \fatsemi (Q * R) \qquad \text{(Indep-1)}$$

$$P \fatsemi Q \to P * Q \qquad \text{if } FV_D(Q) = \emptyset \qquad \text{(Indep-2)}$$

$$P \fatsemi Q \to P \fatsemi (Q * (S \triangleright [S])) \qquad \text{(Pad)}$$

$$(P * Q) \fatsemi (R * S) \to (P \fatsemi R) * (Q \fatsemi S) \qquad \text{(RestExch)}$$

We briefly explain the axioms. Indep-1 may look surprising, and it does not hold if we do not require the formulas to be in DIBI_+. Under this assumption, it holds because $P \fatsemi (Q * R) \in \mathrm{DIBI}_+$ implies that R only mentions variables that are guaranteed to be in P, and then with some maneuver, we can change one sequential composition into a parallel composition. Indep-2 holds because any

kernel witnessing Q depends on no variables and thus is independent of any kernel witnessing P. PAD allows conjoining $(S \triangleright [S])$ to the second conjunct: since $P \fatsemi (Q * (S \triangleright [S]))$ is in DIBI$_+$, S can only mention variables that are already in P. Finally, RESTEXCH shows that the standard exchange law also holds for restricted assertions. We defer the proof to Appendix C.3.2.

We also need the following axioms for a particular form of atomic propositions, in addition to the axioms for general atomic propositions in Proposition 4.3.2.

Proposition 4.3.5. *(AXIOMS FOR ATOMIC PROPOSITIONS) The following axioms are sound. For any $S, A, B, C \subseteq \mathbf{Var}$,*

$$(S \triangleright [A] * [B]) \to (S \triangleright [A]) * (S \triangleright [B]) \qquad \text{if } A \cap B \subseteq S \qquad \text{(REVPAR)}$$

$$(S \triangleright [A] * [B]) \to (S \triangleright [A \cup B]) \qquad \text{(UNIONRAN)}$$

$$(A \triangleright B) \fatsemi (B \triangleright C) \to (A \triangleright C) \qquad \text{(ATOMSEQ)}$$

$$(A \triangleright B) \to (A \triangleright A) \fatsemi (A \triangleright B) \qquad \text{(UNITL)}$$

$$(A \triangleright B) \to (A \triangleright B) \fatsemi (B \triangleright B) \qquad \text{(UNITR)}$$

We defer the proof to Appendix C.3.2.

Now, we have all the ingredients for verifying our example programs, COMMONCAUSE and CONDSAMPLES. Throughout, we must ensure that all formulas used in CPSL rules and DIBI$_+$ axioms are in DIBI$_+$. The conjunction $\fatsemi$ raises a tricky point: DIBI$_+$ is not closed under reassociating $\fatsemi$, so we add parentheses for formulas that must be in DIBI$_+$. However, we may soundly use the full proof system of DIBI when proving implications between DIBI$_+$ assertions, since DIBI$_+$ is a fragment of DIBI.

Verification of CommonCause We aim to prove the following judgment:

$$\vdash \{\top\}\ \textsc{CommonCause}\ \{(\emptyset \triangleright [z]) \fatsemi ((z \triangleright [a]) * (z \triangleright [b]))\}$$

By Theorem 4.2.2, this shows that a, b are conditionally independent given z at the end of the program. First, using Samp to handle the sampling for z, x, y, we can prove the assertion: $(\emptyset \triangleright [z]) \fatsemi (\emptyset \triangleright [x]) \fatsemi (\emptyset \triangleright [y])$. Using Axioms Pad, AP-Par, UnionRan, and $\fatsemi$ Assoc, this assertion implies $(\emptyset \triangleright [z]) \fatsemi (z \triangleright [z, x]) \fatsemi (z \triangleright [z, y])$.

$$\dfrac{\dfrac{\dfrac{\big((\emptyset \triangleright [z]) \fatsemi (\emptyset \triangleright [x])\big) \fatsemi (\emptyset \triangleright [y])}{\big((\emptyset \triangleright [z]) \fatsemi ((\emptyset \triangleright [x]) * (z \triangleright [z]))\big) \fatsemi ((\emptyset \triangleright [y]) * (z \triangleright [z])) * (z \triangleright [z])}\ \textsc{Pad}}{(\emptyset \triangleright [z]) \fatsemi (z \triangleright [z] * [x]) \fatsemi (z \triangleright [z] * [y])}\ \textsc{AP-Par}}{(\emptyset \triangleright [z]) \fatsemi (z \triangleright [z, x]) \fatsemi (z \triangleright [z, y])}\ \textsc{UnionRan}$$

We take the proved formula as the pre-condition before assigning to a and assigning to b. After the assignments, Assn proves:

$$\Big(\big((\emptyset \triangleright [z]) \fatsemi (z \triangleright [z, x]) \fatsemi (z \triangleright [z, y])\big) \fatsemi (z, x \triangleright [a])\Big) \fatsemi (z, y \triangleright [b]).$$

Then, we can reassociate and apply Indep-1 to derive:

$$(\emptyset \triangleright [z]) \fatsemi \big((z \triangleright [z, x]) \fatsemi (z, x \triangleright [a])\big) * \big((z \triangleright [z, y]) \fatsemi (z, y \triangleright [b])\big).$$

By Axiom AtomSeq, we obtain the desired post-condition:

$$(\emptyset \triangleright [z]) \fatsemi ((z \triangleright [a]) * (z \triangleright [b])).$$

□

Verification of CondSamples We aim to show the following judgment:

$$\vdash \{\top\}\ \textsc{CondSamples}\ \{(\emptyset \triangleright [z]) \fatsemi ((z \triangleright [x]) * (z \triangleright [y]))\}$$

Again, by Theorem 4.2.2, this shows that x, y are conditionally independent given z at the end of the program. Starting with the sampling statement for z, applying SAMP, the Axiom INDEP-2, $*$-UNIT and $\fatsemi$-UNIT-R gives:

$$\vdash \{\top\}\ z \xleftarrow{\$} \mathbf{Bern}_{1/2}\ \{(\emptyset \triangleright [z]) \fatsemi \top\}.$$

To reason about the branching, we use COND. We start with the first branch. By SAMP, ASSN, SKIP, WEAK and SEQ, we have $\vdash \{(\emptyset \triangleright z = tt) \fatsemi \top\}\ x \xleftarrow{\$} \mathbf{Bern}_p \fatsemi y \xleftarrow{\$} \mathbf{Bern}_p\ \{(\emptyset \triangleright z = tt) \fatsemi (\emptyset \triangleright [x]) \fatsemi (\emptyset \triangleright [y])\}$. As before, Axioms PAD, AP-PAR, UNIONRAN, together with $\fatsemi$ ASSOC give the post-condition

$$(\emptyset \triangleright z = tt) \fatsemi (z \triangleright [z, x]) \fatsemi (z \triangleright [z, y]).$$

Applying Axiom INDEP-1, we can show $(\emptyset \triangleright z = tt) \fatsemi ((z \triangleright [z, x]) * (z \triangleright [z, y]))$ at the end of the branch. Thus:

$$\vdash \{(\emptyset \triangleright z = tt) \fatsemi \top\}\ x \xleftarrow{\$} \mathbf{Bern}_p \fatsemi y \xleftarrow{\$} \mathbf{Bern}_p\ \{(\emptyset \triangleright z = tt) \fatsemi (z : z = tt \triangleright [z, x] * [z, y])\}.$$

The second branch is similar:

$$\vdash \{(\emptyset \triangleright z = ff) \fatsemi \top\}\ x \xleftarrow{\$} \mathbf{Bern}_q \fatsemi y \xleftarrow{\$} \mathbf{Bern}_q\ \{(\emptyset \triangleright z = ff) \fatsemi (z : z = ff \triangleright [z, x] * [z, y])\}.$$

Applying COND, we have:

$$\{(\emptyset \triangleright [z])\}$$
$$\vdash \quad \text{CONDSAMPLES}$$
$$\{(\emptyset \triangleright [z]) \fatsemi ((z : z = tt \triangleright [z, x] * [z, y]) \wedge (z = ff \triangleright [z, x] * [z, y]))\}$$

By AP-OR, the postcondition implies

$$(\emptyset \triangleright [z]) \fatsemi ((z : z = tt \vee z = ff) \triangleright [z, x] * [z, y] \vee [z, x] * [z, y]).$$

In the domain and range logic, we have: $\models_d z : \top \to z : (z = tt \vee z = ff)$ and

$$\models_r [z, x] * [z, y] \vee [z, x] * [z, y] \to [z, x] * [z, y].$$

So AP-IMP implies $(\emptyset \triangleright [z]) \fatsemi (z \triangleright [z,x] * [z,y])$. We can then apply REVPAR because $\{z,x\} \cap \{z,y\} = z$, deriving the postcondition $(\emptyset \triangleright [z]) \fatsemi ((z \triangleright [z,x]) * (z \triangleright [z,y]))$. By Axiom SPLIT, we obtain the desired post-condition: $(\emptyset \triangleright [z]) \fatsemi ((z \triangleright [x]) * (z \triangleright [y]))$. □

4.4 Related Work

While our program logic is the first separation logic for proving conditional independence, related work has explored other approaches to capture dependencies and independence and has potential to lead to alternative formal methods for reasoning about conditional independence.

Other non-classical logics for modeling dependencies There are other non-classical logics that aim to model dependencies. *Independence-friendly (IF) logic* [Hintikka and Sandu, 1989] and *dependence logic* [Väänänen, 2007] introduce new quantifiers and propositional atoms to state that a variable depends, or does not depend, on another variable logically; these logics are each equivalent in expressivity to existential second-order logic. More recently, Durand et al. [2018] proposed a probabilistic team semantics for dependence logic, and Hannula et al. [2020] gave a descriptive complexity result connecting this logic to real-valued Turing machines. Under probabilistic team semantics, the universal and existential quantifiers bear a resemblance to our separating and dependent conjunctions, respectively. It would be interesting to understand the relation between these two logics, akin to how the semantics of propositional IF forms a model of BI [Abramsky and Väänänen, 2009].

Conditional independence, join dependency, and logic There is a long line of research on logical characterizations of conditional independence and join dependency. The literature is too vast to survey here. On the conditional independence side, we can point to work by Geiger and Pearl [1993] on graphical models; on the join dependency side, the survey by Fagin and Vardi [1984] describes the history of the area in database theory. There are several broadly similar approaches to axiomatizing the general properties of conditional dependence, including graphoids [Pearl and Paz, 1985] and separoids [Dawid, 2001].

Graphical Approach to Conditional Independence Probabilistic graphical models offer a powerful framework for representing probabilistic relationships [Koller and Friedman, 2009, Pearl, 2014]. In particular, Bayesian networks model joint distributions of program variables using directed acyclic graphs (DAGs), where edges represent conditional dependencies: each child node is associated with a conditional distribution given its parent nodes. This structure enables a more compact representation of the overall distribution by leveraging conditional independence between variables.

Bayesian networks are widely used in machine learning as a flexible and interpretable class of models for fitting data [Friedman et al., 1997, Murphy, 2012]. In many cases, the structure of the network is fixed, and the parameters—defining the conditional distributions along the edges—are learned from data via probabilistic inference. In such settings, the conditional independence encoded in the network can significantly improve the efficiency of inference [Obermeyer et al., 2019]. Moreover, the graphical structure makes it easier to identify and exploit these independence relations.

However, when the structure of a suitable Bayesian network is not known in advance, structure learning techniques are applied to discover it from data [Chickering, 2002, 1996, Kitson et al., 2023]. These methods often rely on identifying conditional independence relationships among variables—either through statistical tests or scoring criteria—to constrain the space of possible graph structures. Consequently, the accuracy and reliability of structure learning depend heavily on the ability to verify conditional independence in observed data.

Categorical probability The view of conditional independence as a factorization of Markov kernels has previously been explored [Jacobs and Zanasi, 2017, Cho and Jacobs, 2019, Fritz, 2020]. Taking a different approach, Simpson [2018] has recently introduced category-theoretic structures for modeling generalized conditional independence, capturing conditional independence and join dependency as well as analogues in heaps and nominal sets [Pitts, 2013]. Roughly speaking, conditional independence in heaps requires two disjoint portions except for a common overlap contained in the part that is conditioned; this notion can be smoothly accommodated in our framework as a DIBI model where kernels are Kleisli arrows for the identity monad (Brotherston and Calcagno [2009]) also consider a similar notion of separation). Simpson [2018]'s notion of conditional independence in nominal sets suggests that there might be a DIBI model where kernels are Kleisli arrows for some monad in nominal sets, although the appropriate monad is unclear.

A recent work [Simpson, 2024] studies logical reasoning principles for generalized conditional independence and equality, when equality is a coarser notion than equivalence. They provide a semantic foundation for these reasoning prin-

ciples based on atomic sheaves, and shows a category of probability sheaves as an instantiation. While they do not concern probabilistic programs, and their reasoning principles derive new relations of variables based on known relations in a fixed distribution, it could be interesting to explore alternative assertion logic for capturing probabilistic programs based on their atomic sheaf logic.

A Categorical model to DIBI logic The relational model of DIBI, introduced in the conference version [Bao et al., 2021], and the probabilistic model introduced above are similar but Bao et al. [2021] does not provide a unifying way to construct such similar DIBI models. In our follow-up work Gu et al. [2024], we develop an abstract framework for systematically constructing DIBI models, using category theory as the unifying mathematical language. In particular, we use string diagrams – a graphical presentation of monoidal categories – to give a uniform definition of the parallel composition and preorder in DIBI models. Our approach not only generalises known models, but also yields new models of interest and reduces properties of DIBI models to structures in the underlying categories. Furthermore, our categorical framework enables a logical notion of CI, in terms of the satisfaction of specific DIBI formulas.

CHAPTER 5

BLUEBELL: A UNIFYING FRAMEWORK FOR INDEPENDENCE, CONDITIONAL INDEPENDENCE AND RELATIONAL REASONING

5.1 Overview

In this chapter, we present BLUEBELL, another separation logic for reasoning about probabilistic programs. While BLUEBELL is designed to be a flexible framework for combining unary reasoning and relational reasoning of probabilistic programs, in this thesis, we mostly focus on the unary part of BLUEBELL. The unary part of BLUEBELL shares functionality with CPSL in that they can both be used to prove independence and conditional independence of program variables. However, BLUEBELL allows more expressive assertions and has more ergonomic rules, which enable us to prove conditional independence arising in more complicated programs. Below, we overview our motivation for designing a unifying framework for unary reasoning and relational reasoning, prior work that we get inspiration from, and our key design choices to make it more ergonomic.

Motivation: Independence Helps Relational Reasoning

Unary reasoning means analyzing the behavior of the one target program directly. For example, the program logics introduced in previous chapters, PSL, LINA and CPSL, are all techniques for conducting unary reasoning. This choice aligns with the nature that independence, negative dependence and conditional independence, are all properties of program variables in a single program.

However, sometimes the properties of concern are naturally relational. For instance, we may want to show that two probabilistic programs have the same behaviors, or that their outputs only differ up to a certain margin. To prove such relational goals, it could be easier to compare the two programs step by step, instead of individually capturing their outputs and comparing their outputs. Formalization of relational reasoning of probabilistic programs has been an active research area. One prominent line of work in this area is probabilistic relational Hoare logic (pRHL) [Barthe et al., 2013, Hsu, 2017], which formalizes a technique known as "proof by coupling" by the probability theory community.

Conceptually, we can think of any probabilistic program as a distribution over different execution traces. To compare two probabilistic programs, it can be helpful to pair up the execution traces from the two programs and examine the pair: for example, consider simple programs A and B that both make coin flips, then their coin flips have the same bias iff we can pair up their execution traces such that when A flips head, B flips head as well, and vice versa. This method works not only for proving program equivalence but also for a range of properties important for cryptography and differential privacy [Barthe et al., 2009, 2015, Hsu, 2017, Wang et al., 2019, Zhang and Kifer, 2017]. To describe the pairing, we can use a *coupling*, i.e., the joint distribution of the two distributions, thus making sense of the name "proof by coupling." Note that the pairing and the coupling here are just reasoning tools — the actual execution of the two programs can be either correlated, or completely oblivious of each other.

Our motivation for designing BLUEBELL comes from an observation that independence allows one to decompose relational arguments. As an example, say we want to show two probabilistic programs A and B are equivalent, and A_1, A_2

are two independent components in program A, and similarly, B_1, B_2 are independent two components in program B. Then it is sufficient (and of course not necessary) if we can develop one relational argument showing A_1 equivalent to B_1 and another relational argument showing A_2 equivalent to B_2. Here, the key condition used is independence: when A_1, A_2 are not independent, or B_1, B_2 are not independent, then component-wise equivalence does not guarantee the overall equivalence because the components may be correlated differently; the relations between the two programs do not matter, i.e., we can also replace program equivalence with other relations between A_i and B_i.

Such decomposition can make relational reasoning more scalable, especially when the only other tool for building relational arguments is "proof by coupling," which until recently requires rigid alignments between two programs. Since both reasoning of probabilistic independence and coupling can be subtle, we want to formalize such usage of probabilistic independence in relational proofs. Furthermore, since probabilistic independence is inherently a unary property, it is more natural to prove it in unary style arguments, such as in probabilistic separation logic, prompting us to unify unary reasoning and relational reasoning of probabilistic programs in one framework.

Unary Fragment of BLUEBELL: A More Ergonomic Probabilistic Separation Logic

On the unary reasoning side, we want to present a program separation logic that can cleanly prove independence and conditional independence. Concretely, we want a logic that allows precise description of complicated program states and formalizes subtle probabilistic reasoning as easy-to-apply syntactic rules.

Ideally, the users of the logic can carry out all (or at least most of) the important steps using rules in the logic, instead of resorting to meta-level mathematics. Also, we want to relieve the users from the burden of checking side conditions for applying a rule as much as possible.

In the previous chapter, we show one way to assert conditional independence in bunched logic — using the DIBI logic interpreted on the probabilistic kernels model, and also showed a program logic based on it for reasoning about conditional independence arising in programs. The approach, however, has some limitations. For one thing, DIBI_+ excludes a lot of formulas in DIBI to ensure the restriction property (theorem 4.3.1) which says that a formula P holds in a kernel iff it holds in the subkernel restricted to free variables of P. Although we demonstrated that conditional independence in small programs can be proved using CPSL, whose program rules only involve DIBI_+ formulas, it is cumbersome to always have to check if a formula is in DIBI_+. In addition, in PSL, LINA and DIBI, we cannot assert expressions e_1, e_2 independent if e_1 and e_2 share variables. For example, while x may be independent of $x \mathsf{xor} y$, the formula $\mathsf{Own}(x) * \mathsf{Own}(x \mathsf{xor} y)$ always implies false in their assertion logic.

When designing BLUEBELL, we take inspirations from Li et al. [2023a] (Lilac), which proposes a variant of probabilistic separation logic that addresses these limitations of DIBI for functional programs. We investigate whether we can design a program logic for an *imperative probabilistic programming language* also with these nice features. We work with an imperative probabilistic programming language both because of intellectual curiosity, investigating whether it would allow us to use less technical program semantics than Lilac, and because of our bigger goal to unify unary reasoning and relational reasoning in

one framework — a lot of work in pRHL (e.g., Barthe et al. [2013, 2015, 2016b,a, 2017]) also feature an imperative probabilistic programming language. Allowing the program states to be mutable creates new challenges for validating the frame rule, which requires any separate resource framed to the current one to always be preserved in program execution.

Quick Walkthrough of Lilac [Li et al., 2023a]

Lilac's key innovation is using a new BI model based on measure-theoretic probabilities. In PSL and LINA, no state μ can satisfy assertions such as $\mathsf{Own}(x) * \mathsf{Own}(x \mathsf{\ xor\ } y)$ because evaluating x and $(x \mathsf{\ xor\ } y)$ respectively needs marginal distributions with domain $\{x\}$ and $\{x, y\}$, and the independent product of these two marginal distributions is undefined because their domain overlap on $\{x\}$. However, measure-theoretic probability spaces are specified by a sigma-algebra describing the event space and a measure on the sigma-algebra, and it is possible to separate the event space of x and the event space of $(x \mathsf{\ xor\ } y)$ such that their independent product recovers the original probability space. With measure-theoretic probabilities, they also give a rigorous treatment for continuous probabilities, which enables them to handle examples that sample from a uniform distribution over the interval $[0, 1]$.

To assert conditional independence, Li et al. [2023a] introduce a modality $C_{x \leftarrow X}$ to the assertion logic: their assertion logic model consists of distributions — represented using measure-theoretic probability spaces — instead of kernels as states; in their model, a distribution satisfies $C_{x \leftarrow X} P(x)$ iff, fixing the variable X to any value x it can take, the conditional distribution satisfies $P(X)$. Using that, they show that the conditional independence of variables Y, Z given X can

be asserted using $C_{x \leftarrow X}\ \mathsf{Own}(Y) * \mathsf{Own}(Z)$ and encode several axioms regarding conditioning and independence.

On the program logic level, following the convention adopted by the higher-order concurrent separation logic framework Iris [Jung et al., 2018], Lilac chooses to work with a functional probabilistic language and define the validity of a Hoare triple differently. Their definition of Hoare triples implicitly requires that any frame conjuncted to the current resource must be preserved. This allows the frame rule to be proven easily, without relying on side conditions or verifying that the formulas satisfy the restriction property. In exchange, one needs to inductively prove that each program preserves the frame. As they work with a functional probabilistic programming language, they fix a probability space ("ambient sampling space" in their term) and their program variables are simply random variables on that probability space. They show that everything works out when they fix the ambient sampling space to be the product of countably infinite copies of the $[0, 1]$ interval under a particular set of technical constraints. The choice of the "ambient sampling space" seems highly non-robust, e.g., a product of finite copies of $[0, 1]$ interval would not work in their proofs even if the probabilistic programs only make a finite number of sampling calls.

Bluebell's Design Choices

In BLUEBELL, we combine Lilac's measure-theoretic BI model with a BI model of *permissions*, which are used in Concurrent Separation Logic literature to track who can read from and write to a resource. In our model section 5.3.3, two resources a, b can be composed together only if their permissions can be com-

bined as well, and this extra requirement plays a crucial role in ensuring that the Frame rule holds in our logic.

We draw insights from both DIBI and Lilac to design a modified conditioning modality, introduced in section 5.3.4, that is more expressive and satisfies a richer family of axioms than Lilac's conditioning modality. Our conditioning modality is the key to how we mix unary reasoning and relational reasoning. It supports conditioning on one program state as well as two or more program states, and using that, we can not only capture conditional independence, but also couplings. Furthermore, from the axioms of the conditioning modality, we not only derive reasoning principles important for proving conditional independence, but also relational reasoning principles and their interactions. Even when we only focus on the unary reasoning functionality of BLUEBELL, we can formalize more interesting proof steps in the logic using the richer set of axioms enjoyed by this conditioning modality, showcased by examples in section 5.5.

5.2 Preliminaries: Programs and Probability Spaces

To formally define the model of BLUEBELL and validate its rules, we introduce a number of preliminary notions. Our starting point is the measure-theoretic approach of Li et al. [2023a] in defining probabilistic separation. We recall the main definitions below.

One crucial difference between elementary probability (as how we introduced distributions in definition 2.2.9) and measure-theoretic probability [Rosenthal, 2006, Fristedt and Gray, 2013] is their treatment of event space. In elementary probability, we work with the set of outcomes directly — distri-

butions are defined as maps from outcomes to numbers in $[0, 1]$ and any subset of outcomes is considered as an event. In measure-theoretic probability, events are specified by a structure called σ-algebra.

Definition 5.2.1 (σ-algebra)**.** Given a set of possible *outcomes* Ω, a σ-algebra $\mathcal{F}$ is a set of subsets of Ω that is closed under countable unions and complements, and such that $\Omega \in \mathcal{F}$. We call an element of a σ-algebra *an event*. We denote the set of σ-algebras over a set of outcomes Ω as $\mathbb{A}(\Omega)$.

The *full* σ-algebra over Ω is $\Sigma_\Omega = \mathscr{P}(\Omega)$, the powerset of Ω. For $F \subseteq \mathscr{P}(\Omega)$, we write $\sigma(F) \in \mathbb{A}(\Omega)$ for the smallest σ-algebra containing F.

The measure-theoretic notion of distributions map events in σ-algebras to a number between 0 and 1, which we call the measure of the event, or the probability of the event.

Definition 5.2.2 (Probability Distributions)**.** Given $\mathcal{F} \in \mathbb{A}(\Omega)$, a *probability distribution* $\mu \in \mathcal{D}(\mathcal{F})$, is a function $\mu \colon \mathcal{F} \to [0, 1]$ such that

- For any countable set of disjoint events $\{E_i \mid i \in I\}$, $\mu(\uplus_{i \in I} E_i) = \sum_{i \in I} \mu(E_i)$.
- $\mu(\Omega) = 1$.

The support of a distribution $\mu \in \mathcal{D}(\Sigma_\Omega)$ is the set of outcomes with non-zero probability $\mathrm{supp}(\mu) \triangleq \{a \in \Omega \mid \mu(a) > 0\}$, where $\mu(a)$ abbreviates $\mu(\{a\})$.

Probability spaces are given as a triple of the outcome space, the σ-algebra, and the distribution.

Definition 5.2.3. A *probability space* $\mathcal{P}$ is a pair $\mathcal{P} = (\Omega, \mathcal{F}, \mu)$ of a σ-algebra $\mathcal{F} \in \mathbb{A}(\Omega)$ and a probability distribution $\mu \in \mathcal{D}(\mathcal{F})$. We call the distribution

μ the measure in a probability space $\mathcal{P} = (\mathcal{F}, \mu)$. The *trivial* probability space $\mathbb{1}_\Omega \in \mathbb{P}(\Omega)$ is the trivial σ-algebra $\{\Omega, \emptyset\}$ equipped with the trivial probability distribution that maps Ω to probability 1 and maps $\emptyset$ to probability 0.

When the outcome space is clear, we omit outcome space in the triple of probability spaces.

We define a pre-order on probability spaces to capture the intuition that a probability space A is smaller than a probability space B if A is defined for a subset of events where B is defined on and A agrees with B on the subset. This pre-order will be used in BLUEBELL's BI model over probability spaces.

Definition 5.2.4. Given $\mathcal{F}_1 \subseteq \mathcal{F}_2$ and $\mu \in \mathcal{D}(\mathcal{F}_2)$, the distribution $\mu|_{\mathcal{F}_1} \in \mathcal{D}(\mathcal{F}_1)$ is the restriction of μ to $\mathcal{F}_1$. The *extension pre-order* ($\sqsubseteq$) over probability spaces is defined as $(\mathcal{F}_1, \mu_1) \sqsubseteq (\mathcal{F}_2, \mu_2) \triangleq \mathcal{F}_1 \subseteq \mathcal{F}_2 \wedge \mu_1 = \mu_2|_{\mathcal{F}_1}$.

Given two probability spaces, we identify a set of functions that transfer nicely between them, called *measurable functions*.

Definition 5.2.5. A function $f : \Omega_1 \to \Omega_2$ is *measurable on* $\mathcal{F}_1 \in \mathbb{A}(\Omega_1)$ *and* $\mathcal{F}_2 \in \mathbb{A}(\Omega_2)$ *if for any event* $X \in \mathcal{F}_2$*, we also have* $f^{-1}(X) \in \mathcal{F}_1$*. When* $\mathcal{F}_2 = \Sigma_{\Omega_2}$ *we simply say* f *is measurable on* $\mathcal{F}_1$*.*

Later we will want to decompose one probability space into two, and its definition depends on how we compose two probability spaces into one. Two natural ways to combine two σ-algebras are taking the Cartesian product and taking the union.

Definition 5.2.6 (Product and union spaces)**.** Given $\mathcal{F}_1 \in \mathbb{A}(\Omega_1), \mathcal{F}_2 \in \mathbb{A}(\Omega_2)$,

their product is the σ-algebra $\mathcal{F}_1 \otimes \mathcal{F}_2 \in \mathbb{A}(\Omega_1 \times \Omega_2)$ defined as

$$\mathcal{F}_1 \otimes \mathcal{F}_2 \triangleq \sigma(\{X_1 \times X_2 \mid X_1 \in \mathcal{F}_1, X_2 \in \mathcal{F}_2\}),$$

and their union is the σ-algebra $\mathcal{F}_1 \oplus \mathcal{F}_2 \in \mathbb{A}(\Omega_1 \times \Omega_2)$ defined as $\sigma(\mathcal{F}_1 \cup \mathcal{F}_2)$.

We can take the product of two distributions to obtain a distribution over the product σ-algebra.

Definition 5.2.7. The product of two probability distributions $\mu_1 \in \mathcal{D}(\mathcal{F}_1)$ and $\mu_2 \in \mathcal{D}(\mathcal{F}_2)$ is the distribution $(\mu_1 \otimes \mu_2) \in \mathcal{D}(\mathcal{F}_1 \otimes \mathcal{F}_2)$ defined by

$$(\mu_1 \otimes \mu_2)(X_1 \times X_2) = \mu_1(X_1) \cdot \mu_2(X_2)$$

for all $X_1 \in \mathcal{F}_1$, $X_2 \in \mathcal{F}_2$.

In this chapter, we will frequently use the *independent product* of two distributions, which is over the union of their σ-algebra and is not always defined.

Definition 5.2.8 (Independent product Li et al. [2023a])**.** Given $(\mathcal{F}_1, \mu_1), (\mathcal{F}_2, \mu_2) \in \mathbb{P}(\Omega)$, their *independent product* is the probability space $(\mathcal{F}_1 \oplus \mathcal{F}_2, \mu) \in \mathbb{P}(\Omega)$ where for all $X_1 \in \mathcal{F}_1, X_2 \in \mathcal{F}_2$, $\mu(X_1 \cap X_2) = \mu_1(X_1) \cdot \mu_2(X_2)$. It is unique, if it exists [Li et al., 2023a, Lemma 2.3]. Let $\mathcal{P}_1 \circledast \mathcal{P}_2$ be the unique independent product of $\mathcal{P}_1$ and $\mathcal{P}_2$ when it exists, and be undefined otherwise.

Probabilistic Programming Language

Another important component in BLUEBELL is the probabilistic programming language. We use a simple first-order imperative language very similar to pWhile except that it contains a different construct for loops. As in pWhile, we

$$\mathbb{T} \ni t ::= \texttt{skip} \mid x := e \mid x :\approx d \mid \texttt{if}\ b\ \texttt{then}\ t_1\ \texttt{else}\ t_2 \mid t_1\ ;\ t_2 \mid \texttt{repeat}\ e\ t$$

Figure 5.1: Program Syntax

fix a finite set of *program variables* $x \in \mathbf{Var}$ and countable set of *values* $v \in \mathbf{Val} \triangleq \mathbb{Z}$ and define the program *stores* to be $s \in \mathbf{Mem}[\mathbf{Var}] \triangleq \mathbf{Var} \to \mathbf{Val}$. For simplicity, booleans are encoded by using $0 \in \mathbf{Val}$ as false and any other value as true. Program *terms* $t \in \mathbb{T}$ are formed according to the grammar in fig. 5.1. (We call them terms to follow the terminology in the conference version Bao et al. [2025] and distinguish them from the commands in pWhile.) The expressions e are interpreted into $[\![e]\!] : \mathbf{Mem}[\mathbf{Var}] \to \mathbf{Val}$ following the standard definition (see definition D.1.1). As before, we write $\mathrm{FV}(e)$ for the set of program variables that occur in e. The distributions d are interpreted to be measures over probability space Σ_A for some type A; when $d : \Sigma_A$ is used in the sampling $x :\approx d$, we expect A to be a subset of $\mathbb{Z}$. An example distribution is $\mathbf{Bern}_v$, the Bernoulli distribution with probability v to yield 1 and probability $1 - v$ to yield 0.

Though we do not allow general loops because of difficulties around reasoning about them, we allow iterations implemented through using a simpler construct $\texttt{repeat}\ e\ t$, which evaluates e to a value $n \in \mathbf{Val}$ and, if $n > 0$, executes t in sequence n times. Only allowing this restrictive version of iteration means we only consider a subset of terminating programs.

For the semantics of programs, we interpret each term t to a function $[\![t]\!] : \mathcal{D}(\Sigma_{\mathbf{Mem}[\mathbf{Var}]}) \to \mathcal{D}(\Sigma_{\mathbf{Mem}[\mathbf{Var}]})$, i.e., a map from distributions of input stores to distributions of output stores. The interpretation of the terms is standard, and we defer the mathematical definition to definition D.1.2. Notably, working with a countable set of values **Val** means that the set of program stores are also count-

able, so distributions in $\mathcal{D}(\Sigma_{\mathbf{Mem}[\mathbf{Var}]})$ are also discrete. In BLUEBELL, we work with discrete distributions because it is unclear how continuous distributions interact with some relational constructs in our logic. However, we still use the measure-theoretic definitions for more granular control over the event space.

5.3 The BLUEBELL Logic

We are now ready to define BLUEBELL's semantic model and show its laws.

5.3.1 An Alternative Approach to Bunched Logic

While the assertion logic of BLUEBELL extends from the Bunched logic, we use a different presentation than the one used in PSL, LINA and DIBI. We adapt the approach to BI in Krebbers et al. [2018], which is motivated by efforts in mechanizing various separation logics in a ROCQ framework called Iris. Though BLUEBELL has not been mechanized yet in Bao et al. [2025], we look forward to mechanizing it in the future, and thus, we lay the foundation of the logic in a style that aligns with the Iris framework and its follow-up works. Specifically, instead of interpreting formulas in a structure similar to BI frames and DIBI frames, which combine two states using non-deterministic binary operators, we use a structure called "ordered unital resource algebras " (henceforth RA). RAs allow their states to be combined using either partial or total binary operators: RAs are always equipped with a *total* binary operation and a predicate $\mathbf{V}$ indicating which elements of the carrier are considered *valid* resources; then partiality of the operation manifests as mapping some combinations of arguments to

invalid elements.

Definition 5.3.1 (Ordered Unital Resource Algebra)**.** An *ordered unital resource algebra* (RA) is a tuple $(M, \preceq, \mathbf{V}, \cdot, \varepsilon)$ where $\preceq : M$ is a pre-order called the *resource order*, $\mathbf{V} : M \to \mathsf{Prop}$ is the *validity predicate*, $(\cdot) : M \to M \to M$ is the *resource composition*, a commutative and associative binary operation on M, and $\varepsilon \in M$ is the *unit* of M, satisfying, for all $a, b, c \in M$:

$$\mathbf{V}(\varepsilon); \qquad \text{(Unit Validity)}$$

$$\varepsilon \cdot a = a; \qquad \text{(Unit Existence)}$$

$$\mathbf{V}(a \cdot b) \quad \to \quad \mathbf{V}(a); \qquad \text{(Element Validity)}$$

$$a \preceq b \quad \to \quad a \cdot c \preceq b \cdot c; \qquad \text{(Validity Closure)}$$

$$a \preceq b \quad \to \quad a \cdot c \preceq b \cdot c. \qquad \text{(Order Coherence)}$$

BLUEBELL also differs from PSL, LINA and DIBI in that, in BLUEBELL, we take a semantic approach to assertions: we do not insist on a specific syntax and instead characterize what constitutes an assertion by its type. We embed our definitions in a standard first-order logic, and will refer to it as the meta-level logic. We overload $\wedge$ and $\vee$ as the conjunction and disjunction and write $\Rightarrow$ for the implication in this meta-level logic. Following the convention in ROCQ community, we use Prop denote to the type of propositions.

BLUEBELL uses an alternative definition of BI assertions. To disambiguate from the definition of BI assertions in previous chapters, we denote the BLUEBELL version as BI* assertions.

Definition 5.3.2. We define BI* assertions relative to some RA M as the upward closed functions $M \to \mathsf{Prop}$. A map $P : M \to \mathsf{Prop}$ is upward closed if $\forall a, a' \in M$ such that $a \preceq_M a'$, $P(a) \Rightarrow P(a')$ in the propositional logic.

The requirement that BI* assertions need to be *upward closed* maps is another way to express the persistence condition we impose on assertions in previous chapters. In this chapter, we do not use the symbol $\models$ as the satisfaction relation; instead, we say that a resource a satisfies an assertion P if $P(a)$. Entailment is defined as $(P \vdash Q) \triangleq \forall a \in M.\, \mathbf{V}(a) \Rightarrow (P(a) \Rightarrow Q(a))$. Logical equivalence is defined as entailment in both directions: $P \dashv\vdash Q \triangleq (P \vdash Q) \wedge (Q \vdash P)$.

We introduce two families of assertions useful in separation logic. First, pure assertions $\ulcorner \phi \urcorner$ lift meta-level propositions ϕ to BI* assertions (by ignoring the resource). For example, an formula about specified distributions such as $\mathbf{Bern}_{0.5} = \mathsf{bind}(\mathbf{Bern}_{0.3}, v \mapsto \mathbf{Bern}_{0.5})$ is pure and can be used in separation logic as $\ulcorner \mathbf{Bern}_{0.5} = \mathsf{bind}(\mathbf{Bern}_{0.3}, v \mapsto \mathbf{Bern}_{0.5}) \urcorner$. Second, $\mathsf{Own}(b)$ holds on resources that are greater or equal than b in the RA order; this means b represents a lower bound on the available resources. Mathematically,

$$\ulcorner \phi \urcorner \triangleq \lambda \_.\, \phi$$

$$\mathsf{Own}(b) \triangleq \lambda a.\, b \preceq a$$

We also use standard connectives from BI to produce new assertions given existing ones. We interpret these connectives relative to an RA, and the definition is standard:

$$\begin{array}{rcl}
P \wedge Q & \triangleq & \lambda a.\, P(a) \wedge Q(a) \\
P \vee Q & \triangleq & \lambda a.\, P(a) \vee Q(a) \\
P \to Q & \triangleq & \lambda a.\, \forall b \text{ s.t. } a \preceq b,\, P(b) \Rightarrow Q(b) \\
P * Q & \triangleq & \lambda a.\, \exists b, c \text{ s.t. } a \sqsupseteq b \circ c,\, P(b) \wedge Q(c) \\
P \mathrel{-\!\!*} Q & \triangleq & \lambda a.\, \forall b, c \text{ s.t. } a \circ b \preceq c,\, P(b) \text{ implies } Q(c) \\
\forall x : X.\, P(x) & \triangleq & \lambda a.\, \forall x \in X.\, P(x)(a) \\
\exists x : X.\, P(x) & \triangleq & \lambda a.\, \exists x \in X.\, P(x)(a)
\end{array}$$

Figure 5.2: Satisfaction for BI formulas on RA

5.3.2 A Model of Probabilistic Spaces

BLUEBELL's assertions will be interpreted over a specific RA, which we construct by combining more basic RAs. The main component is the Probability Spaces RA, which uses the independent product as the RA operation.

Definition 5.3.3 (Probability Spaces RA)**.** The *probability spaces RA* PSp_Ω is the resource algebra $(\mathbb{P}(\Omega) \uplus \{\lightning\}, \preceq, \mathbf{V}, \cdot, \mathbb{1}_\Omega)$ where $\preceq$ is the extension pre-order (definition 5.2.4) with $\lightning$ added as the top element, i.e. $\mathcal{P}_1 \preceq \mathcal{P}_2 \triangleq \mathcal{P}_1 \sqsubseteq \mathcal{P}_2$ and $\forall a \in \mathsf{PSp}_\Omega.\, a \preceq \lightning$; $\mathbf{V}(a) \triangleq a \neq \lightning$; composition is the independent product:

$$a \cdot b \triangleq \begin{cases} \mathcal{P}_1 \circledast \mathcal{P}_2 & \text{if } a = \mathcal{P}_1, b = \mathcal{P}_2, \text{ and } \mathcal{P}_1 \circledast \mathcal{P}_2 \text{ is defined} \\ \lightning & \text{otherwise} \end{cases}$$

The fact that PSp_Ω satisfies the axioms of RAs is established in appendix D.3 and builds on the analogous construction in Lilac.

We now introduce assertions that are specific to PSp_Ω. We use the following two abbreviations so we do not need to write out the resource pedantically when using the BI* assertion $\mathsf{Own}(-)$:

$$\mathsf{Own}(\mathcal{F}, \mu, p) \triangleq \mathsf{Own}(((\mathcal{F}, \mu), p)) \qquad \mathsf{Own}(\mathcal{F}, \mu) \triangleq \exists p.\, \mathsf{Own}(\mathcal{F}, \mu, p)$$

We also want to use expressions in assertions. Let *A-typed expressions* be maps E of type $\mathbf{Mem}[\mathbf{Var}] \to A$. We allow PSp_Ω assertions to use *A-typed expressions* for any type A. As an example, the interpretation of any program expression $[\![e]\!] : \mathbf{Mem}[\mathbf{Var}] \to \mathbf{Val}$ is a $\mathbf{Val}$-typed expression. Thus, we seamlessly use program expressions in assertions by implicitly coercing them to their semantics.

The first kind of PSp_Ω assertions we want to introduce are $E \twoheadrightarrow \mu$. Intuitively, we want it to assert that the expression E has the distribution μ in the probability space specified; and to evaluate the expression E, the probability space needs to have enough information — we refer to the condition needed to evaluate an expression E as ownership over E below.

Lilac proposed to use measurability as the notion of ownership. Recall that a function $f : A \to B$ is *measurable* in a sigma-algebra $\mathcal{F} : \mathbb{A}(A)$ if $f^{-1}(b) = \{a \in A \mid f(a) = b\} \in \mathcal{F}$ for all $b \in B$. An A-typed expression E always defines a measurable function (i.e. a *random variable*) on $\Sigma_{\mathbf{Mem[Var]}}$ but might not be measurable on some sub-algebras of $\Sigma_{\mathbf{Mem[Var]}}$. Their definition makes sense because any resource that makes E measurable contains enough information to determine E's distribution, However, we discovered that this choice made axioms used in Lilac's proofs flawed. In short, axioms such as $\mathsf{Own}(x) * \lceil x = y \rceil \models \mathsf{Own}(y)$, which intuitively convey that idea that if x is measurable and x, y are equal in any plausible outcomes, then y is also measurable, played crucial role in Lilac's proofs of example programs but are not sound. [1]

Thus, we propose a slight weakening of the notion of measurability which solves those issues while still retaining the intent behind the notion of ownership. We call this weaker notion "almost measurability".

Definition 5.3.4 (Almost-measurability)**.** Given a probability space $(\mathcal{F}, \mu) \in \mathbb{P}(\Omega)$ and a set $X \subseteq \Omega$, we say X is *almost measurable* in $(\mathcal{F}, \mu)$, written $X \prec (\mathcal{F}, \mu)$, if

$$\exists X_1, X_2 \in \mathcal{F}.\, X_1 \subseteq X \subseteq X_2 \wedge \mu(X_1) = \mu(X_2)$$

We say a function $E : \Omega \to A$, is *almost measurable* in $(\mathcal{F}, \mu)$, written $E \prec (\mathcal{F}, \mu)$, if

[1] A later revision Li et al. [2023b] corrected the issue, although with a different solution from ours.

$E^{-1}(a) \prec (\mathcal{F}, \mu)$ for all $a \in A$.

While almost-measurability does not imply measurability, it constrains the current probability space to uniquely determine the distribution of E in any extension where E becomes measurable.

Example 5.3.1. For example, let $X = \{s \mid s(x) = 42\}$ and $\mathcal{F} = \sigma(\{X\}) = \{\mathbf{Mem}[\mathbf{Var}], \emptyset, X, \mathbf{Mem}[\mathbf{Var}] \setminus X\}$. If $\mu(X) = 1$, then $x \prec (\mathcal{F}, \mu)$ holds but x is not measurable in $\mathcal{F}$, as $\mathcal{F}$ lacks events for $x = v$ for all v except 42. Nevertheless, any extension $(\mathcal{F}', \mu') \sqsupseteq (\mathcal{F}, \mu)$ where x is measurable, would need to assign $\mu'(X) = 1$ and $\mu'(x = v) = 0$ for every $v \neq 42$.

In general, when $X_1 \subseteq X \subseteq X_2$ and $\mu(X_1) = \mu(X_2) = p$, we can unambiguously assign probability p to X, as any extension of μ to Σ_Ω must assign p to X; then we write $\mu(X)$ for p.

When defining $E \mathrel{\text{s}} \mu$, we require E to be almost-measurable and to be distributed as μ in any extension of the local probability space. Formally, given $\mu : \mathcal{D}(\Sigma_A)$ and $E : \mathbf{Mem}[\mathbf{Var}] \to A$, we define:

$$E \mathrel{\text{s}} \mu \triangleq \exists \mathcal{F}, \mu.\, \mathsf{Own}(\mathcal{F}, \mu) * \ulcorner E \prec (\mathcal{F}, \mu) \land \mu = \mu \circ E^{-1} \urcorner$$

Notably, $E \prec (\mathcal{F}, \mu) \land \mu = \mu \circ E^{-1}$ is a pure fact that we can reason without using the local probability space — the probability space $(\mathcal{F}, \mu)$ is fixed by the existential quantifier and not relying on the local probability space.

Using the $E \mathrel{\text{s}} \mu$ assertion, we can define a number of useful derived assertions. In their definition, we use the following events of the outcome space **Val**:

$\mathsf{false} \triangleq \{0\}$ and $\mathsf{true} \triangleq \{n \in \mathbf{Val} \mid n \neq 0\}$.

$$\mathbb{E}[E] = r \triangleq \exists \mu.\, E \twoheadrightarrow \mu * \ulcorner r = \sum *_{a \in \mathrm{supp}(\mu)} \mu(a) \cdot a \urcorner$$

$$\mathsf{Pr}(E) = r \triangleq \exists \mu.\, E \twoheadrightarrow \mu * \ulcorner \mu(\mathsf{true}) = r \urcorner$$

$$\lceil E \rceil \triangleq E \twoheadrightarrow \delta_{\mathsf{true}}$$

$$\mathsf{Own}(E) \triangleq \exists \mu.\, E \twoheadrightarrow \mu$$

Assertions about expectations ($\mathbb{E}[E]$) and probabilities ($\mathsf{Pr}(E)$) both assert E is distributed as μ for some distribution μ and that μ satisfies the desired pure property. To assert $\mathbb{E}[E] = r$, we implicitly assume E is a numerical-typed expression. The assertion holds if E is uniquely determined to distribute as μ and the expected value in μ is r. To assert $\mathsf{Pr}(E) = r$, we implicitly assume E is a **Val**-typed expression. The assertion $\mathsf{Pr}(E) = r$ holds on a probability space if the probability space uniquely determines E to distribute as μ, where μ assign probability r to the event true.

The "almost surely" assertion $\lceil E \rceil$ takes an expression E and asserts that E always "evaluate to true." Because we encode booleans by treating $0 \in \mathbf{Val}$ as false and any other value as true, we define it to evaluate whether E is distributed as the Dirac distribution true — the handling of ownership over E is baked in the definition of distributed as. By this definition, an assertion like $\lceil x = y \rceil$ owns the expression $x = y$ but not necessarily x itself: the only events needed to make the expression $x = y$ almost measurable are $\{s \mid x = y\}$ and $\{s \mid x \neq y\}$, which would be not enough to make x itself almost measurable.

Now we see an example formula that is not satisfiable in PSL's assertion logic, but is satisfiable in the PSp_Ω model.

Example 5.3.2. Assume there are only two variables x and y. Let X_v =

$\{s \mid s(x) = v\}$ and $\mathcal{P}_1 = (\mathcal{F}_1, \mu_1)$ with $\mathcal{F}_1 = \sigma(\{X_v \mid v \in \mathbf{Val}\})$ and let μ_1 give x the distribution of a fair coin, i.e. μ_1 is the extension to $\mathcal{F}_1$ of $\mu_1(X_0) = \mu_1(X_1) = \frac{1}{2}$. Intuitively, the assertion $x \xrightarrow{\$} \mathbf{Bern}_{\frac{1}{2}}$ holds on $\mathcal{P}_1$. Similarly, $\lceil x = y \rceil$ holds on $\mathcal{P}_2 = (\mathcal{F}_2, \mu_2)$ where $\mathcal{F}_2 = \{\emptyset, \mathbf{Mem}[\mathbf{Var}], \{E\}, \mathbf{Mem}[\mathbf{Var}] \setminus E\}$ with $E = \{s \mid s(x) = s(y)\}$ and $\mu_2(E) = 1$. Note that $\mathcal{F}_2$ is very coarse: it does not contain events that can pin the value of x precisely; thanks to this, μ_2 does not need to specify what is the distribution of x, but only that y will coincide on x with probability 1. It is easy to see that the independent product of $\mathcal{P}_1$ and $\mathcal{P}_2$ exists and is $\mathcal{P}_3 = (\mathcal{F}_1 \oplus \mathcal{F}_2, \mu_3)$ where μ_3 is determined by $\mu_3(X_0 \cap E) = \mu_3(X_1 \cap E) = \frac{1}{2}$, i.e. makes x, y the outcomes of the same fair coin. This means $\mathcal{P}_3$ is a model of $x \xrightarrow{\$} \mathbf{Bern}_{\frac{1}{2}} * \lceil x = y \rceil$.

5.3.3 A Model of Mutable Probabilistic Stores

In BLUEBELL, we want to develop a program logic to reason about an imperative probabilistic programming language. Ideally, we want a clean frame rule as in Lilac, which does not need side conditions as in PSL (see section 2.3.3), to make modular reasoning of independent components easy. That means we want to allow assertions on any independent probability spaces to be framed to the pre- and post-conditions of our program judgements simultaneously. Lilac shows that it is sound to do so in their model because their program variables are immutable: their program variables are essentially maps (i.e., random variables) on a fixed probability space over an infinite tape, and they can always perform some manipulations so that the random variables used in the frame assertion depends on a previously unused index of the tape. However, we work with a language with mutation — our program term updates the probability space

over stores as it runs, and it is problematic to allow such a frame rule in our setting.

Example 5.3.3. To illustrate the problem, consider a simple assignment $x := 0$. In the spirit of separation logic's local reasoning, we expect to be able to prove a small footprint triple for the assignment, i.e., one where the precondition only involves ownership of the variable x, such as $\{\mathsf{Own}(x)\}\ x := 0\ \{\lceil x = 0 \rceil\}$. However, we would run into problems when proving the Frame rule, which is the key to enabling modular reasoning in separation logics. As we remarked, an assertion like $\lceil x = y \rceil$ is a valid frame of $\mathsf{Own}(x)$, so the Frame rule would allow us to derive $\vdash \{\mathsf{Own}(x) * \lceil x = y \rceil\}\ x := 0\ \{\lceil x = 0 \rceil * \lceil x = y \rceil\}$. Yet the Hoare triple $\{\mathsf{Own}(x) * \lceil x = y \rceil\}\ x := 0\ \{\lceil x = 0 \rceil * \lceil x = y \rceil\}$ would be invalid because, as long as $y \neq 0$ in input state , the formula $\lceil x = y \rceil$ would not hold after the assignment.

We solve this problem by combining $\mathsf{PSp}_{\mathbf{Mem[Var]}}$, the RA of probability spaces over the outcome space $\mathbf{Mem[Var]}$, with an RA of permissions over variables. The idea is that in addition to information about the distribution, assertions can indicate which "write permissions" we own on variables. An assertion that owns write permissions on x would be incompatible with any frame predicating on x. Then a triple for assignment just needs to require write permission to the assigned variable. We model permissions using a standard fractional permission RA.

Definition 5.3.5. The *permissions RA* is defined as $(\mathsf{Perm}, \preceq, \mathbf{V}, \cdot, \varepsilon)$ where the carrier set Perm is defined to be maps $\mathbf{Var} \to \mathbb{Q}^+$, where $\mathbb{Q}^+$ denotes non-negative rational numbers. The resource pre-order is the point-wise order: for any two $a, b \in \mathsf{Perm}$, we have $a \preceq b$ iff $\forall x \in \mathbf{Var}.\, a(x) \leq b(x)$. A permission is valid if it is upper-bounded by 1: for $a \in \mathsf{Perm}$, $\mathbf{V}(a)$ iff $\forall x \in \mathbf{Var}.\, a(x) \leq 1$. The composition

of two permissions adds the two maps together point-wise: $a_1 \cdot a_2 \triangleq \lambda x.\, a_1(x) + a_2(x)$. The unit with respect to the composition is the constant zero permission: $\varepsilon = \lambda\_.\, 0$.

We now want to associate probability spaces with permissions. The goal is to make sure that, for any resource s with permission 1 on a variable x, any resources that validly compose with s must impose no constraints on the marginal distribution of x. Since resources that validly compose with s must have zero permission on x, we only put restriction on probability spaces' information about variables with zero permission. For variables with strictly positive permission, whether the permission is 0.01 or 1, the probability space can specify full distributions of them, or give no information, or anything in between. This gives rise to the following definition.

Definition 5.3.6 (Compatibility)**.** Given a probability space $\mathcal{P} \in \mathbb{P}(\mathbf{Mem}[\mathbf{Var}])$ and a permission map $p \in \mathsf{Perm}$, let $S = \{x \in \mathbf{Var} \mid p(x) = 0\}$. We say that $\mathcal{P}$ is *compatible* with p, written $\mathcal{P} \# p$, if there exists $\mathcal{P}' \in \mathbb{P}((\mathbf{Var} \setminus S) \to \mathbf{Val})$ such that $\mathcal{P}$ is isomorphic to $\mathcal{P}' \otimes \mathbb{1}_{S \to \mathbf{Val}}$, witnessed by the isomorphism lifted from $\mathbf{Mem}[\mathbf{Var}] \cong ((\mathbf{Var} \setminus S) \to \mathbf{Val}) \times (S \to \mathbf{Val})$ on the outcome space. We extend the definition to $\mathsf{PSp}_{\mathbf{Mem}[\mathbf{Var}]}$ by declaring $\lightning \# p$.

We now construct an RA that associates probability spaces with permissions.

Definition 5.3.7. Let $\mathsf{PSpPm} \triangleq \left\{(\mathcal{P}_{\lightning}, p) \mid \mathcal{P}_{\lightning} \in \mathsf{PSp}_{\mathbf{Mem}[\mathbf{Var}]}, p \in \mathsf{Perm}, \mathcal{P}_{\lightning} \# p\right\}$.

We define *Probability Spaces with Permissions* RA $(\mathsf{PSpPm}, \preceq, \mathbf{V}, \cdot, \varepsilon)$ where

$$\mathbf{V}((\mathcal{P}_{\lightning}, p)) \triangleq \mathcal{P}_{\lightning} \neq \lightning \wedge \forall x. p(x) \leq 1$$

$$(\mathcal{P}_{\lightning}, p) \preceq (\mathcal{P}'_{\lightning}, p') \triangleq \mathcal{P}_{\lightning} \preceq \mathcal{P}'_{\lightning} \text{ and } p \preceq p'$$

$$(\mathcal{P}_{\lightning}, p) \cdot (\mathcal{P}'_{\lightning}, p') \triangleq (\mathcal{P}_{\lightning} \cdot \mathcal{P}'_{\lightning}, p \cdot p')$$

$$\varepsilon \triangleq (\mathbb{1}_{\mathbf{Mem}[\mathbf{Var}]}, \lambda x.\, 0)$$

We define the following assertions specific to $(\mathsf{PSpPm}, \preceq, \mathbf{V}, \cdot, \varepsilon)$.

$$(x{:}q) \triangleq \exists \mathcal{P}, p. \mathsf{Own}(\mathcal{P}, p) * \ulcorner p(i)(x) = q \urcorner \quad P@p \triangleq \exists \mathcal{P}. P(\mathcal{P}) \wedge \mathsf{Own}(\mathcal{P}, p) \quad (5.1)$$

The first assertion $(x{:}q)$ states that the current resource $(\mathcal{P}', p')$ assigns permission at least q to the variable x, i.e., $p'(x) \geq q$. In particular, any resource that can be composed with a resource that satisfies $(x{:}1)$ would have a σ-algebra which is trivial on x. Therefore, having $(x{:}1)$ holds forbids any frame to retain information about x. We can also differentiate between an assertion $\left(x{:}\frac{1}{2}\right)$, which does not allow frames that mutate x but allows frames that predicate on x (e.g. $\lceil x = y \rceil$) , and an assertion $(x{:}1)$ which does not allows frames that predicate on x; consequently, having $\left(x{:}\frac{1}{2}\right)$ hold standalone does not allow mutation of x, but having $(x{:}1)$ enables mutation of x.

The second assertion $P@p$ states P holds in the probability space and p lower bounds the permission. The assertion $(x{:}q)$ is a special case of $P@p$ where P is set to $\top$ and p is defined as: $p(x) = q$ and $p(y) = 0$ for any other variables $y \in \mathbf{Var}$. Also, in practice, preconditions of valid program logic triples are always of the form $P@p$ where p contains full permissions for every variable, the relevant program mutates, and non-zero permissions for the other variables referenced in the assertions or program. For example, we define Hoare triples such that $\{P@p\}\ x := y\ \{Q@q\}$ is valid only if $p(x) = 1$ and $p(y) > 0$.

While permissions allow for a clean semantic treatment of mutation and independence, it does incur some bookkeeping of permissions in practice. The necessary permissions are however easy to infer from the variables used in the assertions, as we will illustrate later in example 5.4.1.

Since we focus on BLUEBELL for unary reasoning in this thesis, our BI model is simply $\mathcal{M} \cong \mathsf{PSpPm}$, and we use assertions in the form of $(x{:}q)$ and $P@p$ to describe resources in this model. We write the type of assertions $\mathcal{M} \to \mathsf{Prop}$ as PA.

5.3.4 Joint Conditioning

To assert conditional independence, we want to assert independence of variables in conditional distributions. We thus introduce the joint conditioning modality $\boldsymbol{C}_\mu\, K$ to assert on conditional distributions. Here we show the definition of $\boldsymbol{C}_\mu$ restricted to the unary setting; a more general version defined for tuples of program states (with permissions) is presented in the conference version [Bao et al., 2025].

Definition 5.3.8 (Joint conditioning modality)**.** Let $\mu' \in \mathcal{D}(\Sigma_A)$ and $K\colon A \to \mathsf{PA}$, then we define the assertion $\boldsymbol{C}_{\mu'}\, K : \mathsf{PA}$ as follows:

$$\begin{aligned}\boldsymbol{C}_{\mu'}\, K \triangleq \lambda a.\, \exists \mathcal{F}, \mu, p, \kappa.\, (\mathcal{F}, \mu, p) \preceq a \land \mu = \mathsf{bind}(\mu', \kappa) \\ \land\, \forall v \in \mathrm{supp}(\mu').\, K(v)(\mathcal{F}, \kappa(v), p)\end{aligned}$$

Intuitively, $\boldsymbol{C}_\mu\, K$ holds on resources whose probability spaces can be seen as the result of binding the given μ and some kernel κ. Then for each outcome v that is in the support of μ, the assertion $K(v)$ is required to hold on the distribution $\kappa(v)$ (packaged with the original σ-algebra and permission to make up a

resource). Note that the definition is upward-closed by construction because of the part $\exists \mathcal{F}, \mu, p.(\mathcal{F}, \mu, p) \preceq a$.

As the name "conditioning modality" suggests, we want to use $\boldsymbol{C}_\mu\, K$ to assert formulas on conditional distributions. To assert $Q(v)$ on the conditional distribution fixing the value of a variable x to v, we assert

$$\exists \mu'.\, \boldsymbol{C}_{\mu'}\, v.\lceil x = v \rceil * Q(v),$$

where we use the notation $v.\lceil x = v \rceil * Q(v)$ to denote the map from any outcome $v \in \mathsf{supp}(\mu')$ to the assertion $\lceil x = v \rceil * Q(v)$. This works because: if a resource $(\mathcal{F}, \mu, p)$ satisfies $\boldsymbol{C}_{\mu'}\, v.\lceil x = v \rceil * Q(v)$, then we can prove from $\boldsymbol{C}_{\mu'}\, v.\lceil x = v \rceil$ that x is distributed as μ' in the μ; furthermore, it says there must exists κ such that μ is the distribution of x extended with κ, i.e., $\mu = \mathsf{bind}(\mu', \kappa)$, and $\lceil x = v \rceil$ in $(\mathcal{F}, \kappa(v), p)$ for every $v \in \mathsf{supp}(\mu')$, these two conditions together constrain $\kappa(v)$ to be the original distribution μ conditioned on $\lceil x = v \rceil$. Because $Q(v)$ is asserted on the distribution $\kappa(v)$ for each $v \in \mathsf{supp}(\mu')$ we have $Q(v)$ holds in the respective conditional distributions. As an example, the conditional independence of variables y and z given x can be asserted as

$$\exists \mu'.\, \boldsymbol{C}_{\mu'}\, v.\lceil x = v \rceil * \mathsf{Own}(y) * \mathsf{Own}(z).$$

In this particular case, $Q(v)$ is invariant with respect to v.

5.3.5 The Rules of Conditioning and Independence

Although we adopt a "shallow embedding" approach to assertions in this chapter, the rules of BLUEBELL provide an axiomatic treatment of these assertions so that the user should *never manipulate raw predicates over the semantic model*. For

— Distribution ownership rules

DIST-INJ
$$E \mathrel{\&} \mu \wedge E \mathrel{\&} \mu' \vdash \ulcorner \mu = \mu' \urcorner$$

SURE-MERGE
$$\lceil E_1 \rceil * \lceil E_2 \rceil \dashv\vdash \lceil (E_1 \wedge E_2) \rceil$$

PROD-SPLIT
$$(E_1, E_2) \mathrel{\&} \mu_1 \otimes \mu_2 \vdash E_1 \mathrel{\&} \mu_1 * E_2 \mathrel{\&} \mu_2$$

— Joint conditioning rules

C-TRUE
$$\vdash \boldsymbol{C}_\mu \_ .\, \mathsf{True}$$

C-FALSE
$$\boldsymbol{C}_\mu\, v.\, \mathsf{False} \vdash \mathsf{False}$$

C-CONS
$$\frac{\forall v.\, K_1(v) \vdash K_2(v)}{\boldsymbol{C}_\mu\, v.\, K_1(v) \vdash \boldsymbol{C}_\mu\, v.\, K_2(v)}$$

C-FRAME
$$P * \boldsymbol{C}_\mu\, v.\, K(v) \vdash \boldsymbol{C}_\mu\, v.\, (P * K(v))$$

C-UNIT-L
$$\boldsymbol{C}_{\delta_{v_0}}\, v.\, K(v) \dashv\vdash K(v_0)$$

C-UNIT-R
$$E \mathrel{\&} \mu \dashv\vdash \boldsymbol{C}_\mu\, v.\, \lceil E = v \rceil$$

C-ASSOC
$$\frac{\mu_0 = \mathsf{bind}(\mu, \lambda v.\, (\mathsf{bind}(\kappa(v), \lambda w.\, \mathbf{return}(v, w))))}{\boldsymbol{C}_\mu\, v.\, \boldsymbol{C}_{\kappa(v)}\, w.\, K(v, w) \vdash \boldsymbol{C}_{\mu_0}(v, w).\, K(v, w)}$$

C-UNASSOC
$$\boldsymbol{C}_{\mathsf{bind}(\mu,\kappa)}\, w.\, K(w) \vdash \boldsymbol{C}_\mu\, v.\, \boldsymbol{C}_{\kappa(v)}\, w.\, K(w)$$

C-SKOLEM
$$\frac{\mu : \mathcal{D}(\Sigma_A)}{\boldsymbol{C}_\mu\, v.\, \exists x : X.\, Q(v, x) \vdash \exists f : A \to X.\, \boldsymbol{C}_\mu\, v.\, Q(v, f(v))}$$

C-TRANSF
$$\frac{f :\ \mathrm{supp}(\mu) \to \mathrm{supp}(\mu') \ \text{ bijective} \quad \forall b \in \mathrm{supp}(\mu').\, \mu'(b) = \mu(f^{-1}(b))}{\boldsymbol{C}_\mu\, a.\, K(a) \vdash \boldsymbol{C}_{\mu'}\, b.\, K(f^{-1}(b))}$$

SURE-STR-CONVEX
$$\boldsymbol{C}_\mu\, v.\, (K(v) * \lceil E \rceil) \vdash \lceil E \rceil * \boldsymbol{C}_\mu\, v.\, K(v)$$

C-FOR-ALL
$$\boldsymbol{C}_\mu\, v.\, \forall x : X.\, Q(v, x) \vdash \forall x : X.\, \boldsymbol{C}_\mu\, v.\, Q(v, x)$$

C-PURE
$$\ulcorner \mu(X) = 1 \urcorner * \boldsymbol{C}_\mu\, v.\, K(v) \dashv\vdash \boldsymbol{C}_\mu\, v.\, (\ulcorner v \in X \urcorner * K(v))$$

Figure 5.3: Primitive rules of BLUEBELL.

brevity, we omit the rules that apply to the basic connectives of separation logic, as they are well-known and have been proven correct for any model that is an RA. For those we refer to Krebbers et al. [2018].

We make a distinction between "primitive" and "derived" rules. The primitive rules require proofs that manipulate the semantic model definitions directly.

— Ownership and distributions

SURE-DIRAC
$E \twoheadrightarrow \delta_v \dashv\vdash \lceil E = v \rceil$

SURE-EQ-INJ
$\lceil E = v \rceil * \lceil E = v' \rceil \vdash \ulcorner v = v' \urcorner$

SURE-SUB
$E_1 \twoheadrightarrow \mu * \lceil (E_2 = f(E_1)) \rceil \vdash E_2 \twoheadrightarrow \mu \circ f^{-1}$

DIST-FUN
$E \twoheadrightarrow \mu \vdash (f \circ E) \twoheadrightarrow \mu \circ f^{-1}$

DIRAC-DUP
$E \twoheadrightarrow \delta_v \vdash E \twoheadrightarrow \delta_v * E \twoheadrightarrow \delta_v$

DIST-SUPP
$E \twoheadrightarrow \mu \vdash E \twoheadrightarrow \mu * \lceil E \in \mathrm{supp}(\mu) \rceil$

PROD-UNSPLIT
$E_1 \twoheadrightarrow \mu_1 * E_2 \twoheadrightarrow \mu_2 \vdash (E_1, E_2) \twoheadrightarrow \mu_1 \otimes \mu_2$

— Joint conditioning

C-FUSE
$\boldsymbol{C}_\mu\, v.\, \boldsymbol{C}_{\kappa(v)}\, w.\, K(v,w) \dashv\vdash \boldsymbol{C}_{\mu \prec \kappa}(v,w).\, K(v,w)$

C-SWAP
$\boldsymbol{C}_{\mu_1}\, v_1.\, \boldsymbol{C}_{\mu_2}\, v_2.\, K(v_1,v_2) \vdash \boldsymbol{C}_{\mu_2}\, v_2.\, \boldsymbol{C}_{\mu_1}\, v_1.\, K(v_1,v_2)$

SURE-CONVEX
$\boldsymbol{C}_\mu\, v.\, \lceil E \rceil \vdash \lceil E \rceil$

DIST-CONVEX
$\boldsymbol{C}_\mu\, v.\, E \twoheadrightarrow \mu' \vdash E \twoheadrightarrow \mu'$

C-SURE-PROJ
$\boldsymbol{C}_\mu(v,w).\, \lceil E(v) \rceil \dashv\vdash \boldsymbol{C}_{\mu \circ \pi_1^{-1}}\, v.\, \lceil E(v) \rceil$

C-EXTRACT
$\boldsymbol{C}_{\mu_1}\, v_1.\, (\lceil E_1 = v_1 \rceil * E_2 \twoheadrightarrow \mu_2) \vdash E_1 \twoheadrightarrow \mu_1 * E_2 \twoheadrightarrow \mu_2$

C-DIST-PROJ
$\boldsymbol{C}_\mu(x,y).\, E(x) \twoheadrightarrow \mu(x) \vdash \boldsymbol{C}_{\mu \circ \pi_1^{-1}}\, x.\, E(x) \twoheadrightarrow \mu(x)$

Figure 5.4: Derived rules.

The derived rules can be proved sound by staying at the level of the logic, i.e. by using the primitive rules of BLUEBELL. Figure 5.3 presents the primitive rules and fig. 5.4 presents the derived rules.[2]

We first present three primitive rules concerning distribution ownership. DIST-INJ allows us to conclude from two assertions on an expression's distribution that the distributions asserted in the two are the same. SURE-MERGE combines two sure assertions into one. PROD-SPLIT rewrites an assertion saying that

[2]We omitted rules for relational reasoning here. They are presented in the appendix of the conference version Bao et al. [2025].

two expressions are distributed as the independent product of μ_1, μ_2 using the independent conjunction $*$ in the logic.

We then present the primitive rules for the conditioning modality. Among the primitive rules, C-TRUE, C-FALSE, C-FRAME, C-SKOLEM and C-FOR-ALL describe how the conditioning modality interacts with other connectives in the logic — respectively True, False, $*$, $\exists$ and $\forall$ here. In particular, C-TRUE allows to introduce a trivial modality; together with C-FRAME this allows for the introduction of the modality around any assertion.

Because distributions form a monad, and the definition of the conditioning modality uses the monadic bind, we also have rules corresponding to the three monad laws: C-UNIT-L (resp. C-UNIT-R) reflects the existence of left units (resp. right unit) for bind; and C-ASSOC and C-UNASSOC) hold because the monadic bind is associative. Among the rest, C-CONS allows us to weaken the assertion under conditioning; C-TRANSF allows for the transformation of the convex combination using μ into using μ' by applying a bijection between their support in a way that does not affect the weight of each outcome; SURE-STR-CONVEX internalizes a stronger version of convexity of $\lceil E \rceil$ assertions and allows us to pull $\lceil E \rceil$ out of conditioning modality — it is the reversal of C-FRAME but only applies for sure assertions; C-PURE allows to translate facts that hold with probability 1 in μ to predicates that hold on every v bound by conditioning on μ.

The derived rules capture other useful reasoning patterns followed from the primitive rules. For instance, C-FUSE is derived from C-ASSOC and C-UNASSOC. It concerns a particular distribution $\mu \prec k$, defined by

$$\mu \prec \kappa := \mathsf{bind}(\mu, v \mapsto \mathsf{unit}(v) \otimes \kappa(v)).$$

We explain the rest of these rules in section 5.5 when they are used.

5.4 Reasoning about Programs in BLUEBELL

To reason about programs, we introduce a *weakest-precondition assertion* (WP) $\mathbf{wp}\, t\, \{Q\}$. Our weakest-precondition assertion $\mathbf{wp}\, t\, \{Q\}$ intuitively states: given the current input distributions, if we run the programs in t, we obtain output distributions that satisfy Q; furthermore, every frame is preserved.

Definition 5.4.1 (Weakest Precondition)**.** For $a \in \mathcal{M}_I$ and $\mu : \mathcal{D}(\Sigma_{\mathbf{Mem}[\mathbf{Var}]^I})$, let $a \preceq \mu$ to abbreviate for $a \preceq (\Sigma_{\mathbf{Mem}[\mathbf{Var}]^I}, \mu, \lambda x.\, 1)$.

$$\mathbf{wp}\, t\, \{Q\} \triangleq \lambda a.\, \forall \mu_0. \forall c.\, (a \cdot c) \preceq \mu_0 \Rightarrow \exists b.\, \big((b \cdot c) \preceq [\![t]\!](\mu_0) \wedge Q(b)\big)$$

The assertion holds on the resources a such that if, together with some frame c, they can be seen as a fragment of the global distribution μ_0, then it is possible to update the resource to some b which still composes with the frame c, and $b \cdot c$ can be seen as a fragment of the output distribution $[\![t]\!](\mu_0)$. Moreover, such b needs to satisfy the postcondition Q.

In the previous chapters, we use Hoare triples for reasoning about programs. We remark two kinds of differences here. The first difference is between the Hoare-style logic and weakest-precondition style specifications. Previously, we treat Hoare triples as judgments in the program logic layer, which uses the assertion logic layer for specifications. But here, we consider WP as a modality of the logic, analogous to the conditioning modality. The WP modality only uses the post-conditions and the programs while Hoare triple in addition takes the preconditions. One can also define Hoare triples on top of the WP by

$$\{P\}\ t\ \{Q\} \triangleq P \vdash \mathbf{wp}\, t\, \{Q\}$$

and computes a sufficient pre-condition. The second difference is our design

choice to require every frame to be preserved in the definition of WP. While this sets us apart from the the original notion of weakest preconditions in Dijkstra's seminal paper Dijkstra [1975], the choice of requiring the frame to be preserved is also prevalent in separation logics literature (e.g., Jung et al. [2015], Li et al. [2023a]). Crucially, this more complicated version of WP frees us from requiring the formulas to satisfy the restriction property, which is needed to isolate a part of the resource sufficient for validating a formula in LINA and DIBI.

We present the full set of WP rules in fig. 5.5. The structural rules has the standard WP-CONS that allows us to weaken the post-condition. WP-CONS, as we desired, does not need side conditions. C-WP-SWAP is a new rule, saying that we can commute the conditioning modality and the WP. This rule facilitates case analysis in program analysis: it implies that, if we can condition the current probability space based on different scenarios $\nu \sim \mu$ and, for each scenario ν, we have $Q(\nu)$ after running t, then we can push the case analysis to the postcondition after running t. There is a side condition, however, that we need to own all the variables in **Var** because subtleties in the interaction of C-WP-swapping frame preservation.

For the program rules, WP-SKIP and WP-SEQ are standard. We discuss the rules for assignments and sampling in more detail below. WP-IF-PRIM is also the standard rule for a conditional whose guard is simply a value; but we can reason about conditionals whose guard is a randomized variable as well by first conditioning on the value of the guard, and then apply WP-IF-PRIM together with WP-BIND and C-WP-SWAP. We encapsulate this reasoning pattern as the derived rule WP-IF-UNARY. The loop rule WP-LOOP-UNF helps unfolding a loop with $(n+1)$ iterations, and WP-LOOP reduces the task of reasoning about n itera-

— Structural WP rules

WP-CONS
$$\frac{Q \vdash Q'}{\mathbf{wp}\, t\, \{Q\} \vdash \mathbf{wp}\, t\, \{Q'\}}$$

WP-FRAME
$$P * \mathbf{wp}\, t\, \{Q\} \vdash \mathbf{wp}\, t\, \{P * Q\}$$

C-WP-SWAP
$$\boldsymbol{C}_\mu\, v.\, \mathbf{wp}\, t\, \{Q(v)\} \wedge \mathsf{own}_{\mathbf{Var}} \vdash \mathbf{wp}\, t\, \left\{\boldsymbol{C}_\mu\, v.\, Q(v)\right\}$$

— Program WP rules

WP-SKIP
$$P \vdash \mathbf{wp}\, [\mathtt{skip}]\, \{P\}$$

WP-SEQ
$$\mathbf{wp}\, [t]\, \left\{\mathbf{wp}\, [t']\, \{Q\}\right\} \vdash \mathbf{wp}\, [t;\ t']\, \{Q\}$$

WP-ASSIGN
$$\frac{x \notin \mathrm{FV}(e) \qquad \forall y \in \mathrm{FV}(e).\, p(y) > 0 \qquad p(x) = 1}{(p) \vdash \mathbf{wp}\, [\mathrm{x} := e]\, \left\{\lceil x = e \rceil @ p\right\}}$$

WP-SAMP
$$(x{:}1) \vdash \mathbf{wp}\, [\mathrm{x} :\approx d\ (\vec{v})\ ]\, \{x \stackrel{\$}{\leftarrow} d(\vec{v})\}$$

WP-IF-PRIM
$$\text{if } v \text{ then } \mathbf{wp}\, [t_1]\, \{Q(1)\} \text{ else } \mathbf{wp}\, [t_2]\, \{Q(0)\} \vdash \mathbf{wp}\, [\mathtt{if}\ v\ \mathtt{then}\ t_1\ \mathtt{else}\ t_2]\, \{Q(v)\}$$

WP-BIND
$$\lceil e = v \rceil * \mathbf{wp}\, \left[\mathcal{E}[v]\right]\, \{Q\} \vdash \mathbf{wp}\, \left[\mathcal{E}[e]\right]\, \{Q\}$$

WP-LOOP-UNF
$$\mathbf{wp}\, [\mathtt{repeat}\ n\ t]\, \{\mathbf{wp}\, [t]\, \{Q\}\} \vdash \mathbf{wp}\, [\mathtt{repeat}\ (n+1)\ t]\, \{Q\}$$

WP-LOOP
$$\frac{\forall i < n.\, P(i) \vdash \mathbf{wp}\, [t]\, \{P(i+1)\}}{P(0) \vdash \mathbf{wp}\, [\mathtt{repeat}\ n\ t]\, \{P(n)\}}\ n \in \mathbb{N}$$

Figure 5.5: The primitive WP rules of BLUEBELL.

tions to reason about each loop iteration. Both loop rules are proved by straightforward inductions on the semantics level, and we can also derive WP-LOOP-0 from these two rules.

We prove the soundness of each rules, using facts in first-order logic, in appendix D.5.

Theorem 5.4.1. *If $P \vdash Q$, then $P \Rightarrow Q$ is derivable in first-order logic.*

WP-SAMP is the expected "small footprint" rule for sampling; the precondi-

WP-LOOP-0
$$P \vdash \mathbf{wp}\,[i\colon \texttt{repeat}\ 0\ t]\,\{P\}$$

WP-IF-UNARY
$$\frac{P * \lceil e = 1 \rceil \Vdash \mathbf{wp}\,[t_1]\,\{Q(1)\} \qquad P * \lceil e = 0 \rceil \Vdash \mathbf{wp}\,[t_2]\,\{Q(0)\}}{P * e \twoheadrightarrow \beta \Vdash \mathbf{wp}\,[\texttt{if}\ e\ \texttt{then}\ t_1\ \texttt{else}\ t_2]\,\{C_\beta\, b.\, Q(b)\}}$$

Figure 5.6: Derived WP rules.

tion only requires full permission on the variable being assigned, to forbid any frame to record information about it. WP-ASSIGN requires full permission on x, and non-zero permission on the variables on the right-hand side of the assignment. This allows the postcondition to assert that x and the expression e assigned to it are equal with probability 1. The condition $x \notin \mathrm{FV}(\vec{e})$ ensures e has the same meaning before and after the assignment, but is not restrictive: if needed the old value of x can be stored in a temporary variable, or the proof can condition on x to work with its pure value.

The assignment and sampling rules are the only ones that impose constraints on the owned permissions. In proofs, this means that most rule applications simply thread through permissions so that the needed permissions can reach the applications of the assignment rules. To avoid cluttering proof derivations with this bookkeeping, we mostly omit permission information from assertions. The appropriate permission annotations can be inferred, as we show in the following example.

Example 5.4.1. Consider the following triple with an unknown permission p:

$$(x \twoheadrightarrow \mu_1 * \lceil x = y \rceil * z \twoheadrightarrow \mu_2)@(p) \vdash \mathbf{wp}\,[\mathrm{x} := \mathrm{z}]\,\{(\lceil x = z \rceil * z \twoheadrightarrow \mu_2)@(p)\}$$

We want to determine a choice for p that makes the proof derivation goes through. Because the assignment only changes the variable x by an assignment,

our proof strategy is to first apply WP-ASSIGN and WP-CONS to prove

$$(x \twoheadrightarrow \mu_1 * \lceil x = y \rceil)@(p') \vdash \mathbf{wp}\,[\mathrm{x} := \mathrm{z}]\,\{\lceil x = z \rceil@(p')\},$$

for some suitable p' and then apply WP-FRAME to frame $z \twoheadrightarrow \mu_2$ and prove the original goal.

To apply WP-ASSIGN, we need to ensure $p'(x) = 1$ and $p'(z) > 0$. Because $\lceil x = y \rceil$ is not trivial on y, so it must $p'(y) > 0$ as well Also, to apply WP-FRAME to frame $z \twoheadrightarrow \mu_2$, we need to ensure p' composes with another permission p'' that is compatible with the probability space where $z \twoheadrightarrow \mu_2$ holds; because $z \twoheadrightarrow \mu_2$ asserts z is non-trivial, it must $p''(z) > 0$, indicating $p'(z) < 1$. Thus, one reasonable way to distribute the permission is

$$p'(x) = 1 \qquad p'(y) = 1/2 p'(z) = 1/3$$
$$p''(x) = 0 \qquad p''(y) = 0 p''(z) = 1/3.$$

We can thus prove that

$$(x \twoheadrightarrow \mu_1 * \lceil x = y \rceil)@\left(x{:}1, y{:}\frac{1}{2}, z{:}\frac{1}{3}\right) * z \twoheadrightarrow \mu_2@\left(z{:}\frac{1}{3}\right)$$
$$\vdash \mathbf{wp}\,[\mathrm{x} := \mathrm{z}]\,\{(\lceil x = z \rceil@\left(x{:}1, y{:}\frac{1}{2}, z{:}\frac{1}{3}\right) * z \twoheadrightarrow \mu_2)@\left(z{:}\frac{1}{3}\right)\}$$

The triple can be further composed with a frame that asserts fractional permission for y and z. because permissions in the range $(0, 1)$ essentially serve the same role, we can also pick different number for $p'(y), p'(z)$ and $p''(z)$ as long as $p'(y), p'(z), p''(z)$ stay in $(0, 1)$ and $p'(z) + p''(z)$ stay in $(0, 1)$.

5.5 Case Studies for BLUEBELL

Our evaluation of BLUEBELL is based on two main lines of enquiry: (1) Are high-level principles about probabilistic reasoning provable from the core constructs of BLUEBELL? (2) Does BLUEBELL, through enabling new reasoning patterns, expand the horizon for verification of probabilistic programs *beyond* what was possible before? We include case studies that try to highlight the contribution of BLUEBELL each question, and sometimes both at the same time. Specifically, our evaluation is guided by the following research questions:

RQ1: Do joint conditioning and independence offer a good abstract interface over the underlying semantic model?

RQ2: Can known unary/relational principles be reconstructed from BLUEBELL's primitives?

RQ3: Can new unary/relational principles be discovered (as new lemmas) and proved from BLUEBELL's primitives?

RQ4: Can BLUEBELL's primitives be successfully incorporated in an effective *program* logic?

Since we only introduced the unary part of BLUEBELL, we only show examples that exercise BLUEBELL's unary reasoning.

5.5.1 One Time Pad Revisited

In fig. 5.7 we show a simple example adapted from Barthe et al. [2019]: the encrypt procedure uses a fair coin flip to generate an encryption key k, generates a plaintext message in boolean variable m (using a coin flip with some bias p) and produces the ciphertext c by XORing the key and the message.

```
def encrypt():
  k :≈ Ber(1/2)
  m :≈ Ber(p)
  c := k xor m
```

Figure 5.7: One time pad.

One way of stating and proving the correctness of encrypt is to establish that in the output distribution c and m are independent, which can be expressed as the *unary* goal:

$$([k\colon 1, m\colon 1, c\colon 1]) \vdash \mathbf{wp}\,[1\colon \texttt{encrypt()}]\,\{c \mathrel{\$} \mathbf{Bern}_{1/2} * m \mathrel{\$} \mathbf{Bern}_p\}$$

The triple states that after running encrypt, the ciphertext c is distributed as a fair coin, and—importantly—is *not* correlated with the plaintext in m. The PSL proof in Barthe et al. [2019] performs some of the steps within the logic, but needs to carry out some crucial entailments at the meta-level, which shows some limitations of its abstractions (**RQ1**). The same applies to the Lilac proof in Li et al. [2023a] which requires ad-hoc lemmas proven on the semantic model. The stumbling block is proving the valid entailment:

$$k \mathrel{\$} \mathbf{Bern}_{\frac{1}{2}} * m \mathrel{\$} \mathbf{Bern}_p * \lceil c = k \;\mathsf{xor}\; m \rceil \vdash m \mathrel{\$} \mathbf{Bern}_p * c \mathrel{\$} \mathbf{Bern}_{\frac{1}{2}}$$

In BLUEBELL we can prove the entailment in two steps: (1) we condition on m and k to compute the result of the *xor* operation and obtain that c is distributed as $\mathbf{Bern}_{\frac{1}{2}}$; (2) we carefully eliminate the conditioning while preserving the independence of m and c.

The first step starts by conditioning the distribution of the message m using C-UNIT-R.

$$
\begin{aligned}
& k \mathrel{\text{≿}} \mathbf{Bern}_{\frac{1}{2}} * m \mathrel{\text{≿}} \mathbf{Bern}_p * \lceil c = k \mathsf{\ xor\ } m \rceil \\
\vdash\ & m \mathrel{\text{≿}} \mathbf{Bern}_p * k \mathrel{\text{≿}} \mathbf{Bern}_{\frac{1}{2}} * \lceil c = k \mathsf{\ xor\ } m \rceil \\
\vdash\ & \left(C_{\mathbf{Bern}_p}\, u.\, \lceil m = u \rceil\right) * k \mathrel{\text{≿}} \mathbf{Bern}_{\frac{1}{2}} * \lceil c = k \mathsf{\ xor\ } m \rceil && \text{(C-UNIT-R)}
\end{aligned}
$$

Because the key k is sampled independently from the message m and the almost sure assertion $\lceil c = k \mathsf{\ xor\ } m \rceil$ also holds on an independent probability space, conditioning on m does not change assertion on them — this idea is formalized by the rule C-FRAME.

$$
\begin{aligned}
& \left(C_{\mathbf{Bern}_p}\, u.\, \lceil m = u \rceil\right) * k \mathrel{\text{≿}} \mathbf{Bern}_{\frac{1}{2}} * \lceil c = k \mathsf{\ xor\ } m \rceil \\
\vdash\ & C_{\mathbf{Bern}_p}\, u.\, \left(\lceil m = u \rceil * k \mathrel{\text{≿}} \mathbf{Bern}_{\frac{1}{2}} * \lceil c = k \mathsf{\ xor\ } m \rceil\right) && \text{(C-FRAME)}
\end{aligned}
$$

Because $\lceil m = u \rceil$ and $\lceil c = k \mathsf{\ xor\ } m \rceil$ under conditioning, so we can merge the two facts into $\lceil c = k \mathsf{\ xor\ } u \rceil$ using SURE-MERGE. After that, we condition on k, again by first using C-UNIT-R and then moving the fact $\lceil c = k \mathsf{\ xor\ } u \rceil$ under the conditioning using C-FRAME.

$$
\begin{aligned}
& C_{\mathbf{Bern}_p}\, u.\, \left(\lceil m = u \rceil * k \mathrel{\text{≿}} \mathbf{Bern}_{\frac{1}{2}} * \lceil c = k \mathsf{\ xor\ } m \rceil\right) \\
\vdash\ & C_{\mathbf{Bern}_p}\, u.\, \left(\lceil m = u \rceil * k \mathrel{\text{≿}} \mathbf{Bern}_{\frac{1}{2}} * \lceil c = k \mathsf{\ xor\ } u \rceil\right) && \text{(SURE-MERGE)} \\
\vdash\ & C_{\mathbf{Bern}_p}\, u.\, \left(\lceil m = u \rceil * \left(C_{\mathbf{Bern}_{\frac{1}{2}}}\, v.\, \lceil k = v \rceil\right) * \lceil c = k \mathsf{\ xor\ } u \rceil\right) && \text{(C-UNIT-R)} \\
\vdash\ & C_{\mathbf{Bern}_p}\, u.\, \left(\lceil m = u \rceil * C_{\mathbf{Bern}_{\frac{1}{2}}}\, v.\, \left(\lceil k = v \rceil * \lceil c = k \mathsf{\ xor\ } u \rceil\right)\right) && \text{(C-FRAME)}
\end{aligned}
$$

Under the conditioning of m and k, we have the fact that $\lceil k = v \rceil * \lceil c = k \mathsf{\ xor\ } u \rceil$. In the final goal, we do not care what the key k is as long as the ciphered message c does not leak any information about the original message m. So next,

we side-step assertions about k to facilitate reasoning about c and m. Formally, we merge $\lceil k = v \rceil * \lceil c = k \text{ xor } u \rceil$ into $\lceil k = v \wedge c = k \text{ xor } u \rceil$ using SURE-MERGE, and then apply propositional reasoning to rewrite it into a case analysis, i.e. $\lceil c = v \rceil$ when $u = 0$ and $\lceil c = \neg v \rceil$ when $u = 1$.

$$
\begin{aligned}
& C_{\mathbf{Bern}_p}\, u.\, \left(\lceil m = u \rceil * C_{\mathbf{Bern}_{\frac{1}{2}}}\, v.\, \left(\lceil k = v \rceil * \lceil c = k \text{ xor } u \rceil \right) \right) \\
\vdash\ & C_{\mathbf{Bern}_p}\, u.\, \left(\lceil m = u \rceil * C_{\mathbf{Bern}_{\frac{1}{2}}}\, v.\, \lceil k = v \wedge c = v \text{ xor } u \rceil \right) && \text{(SURE-MERGE)} \\
\vdash\ & C_{\mathbf{Bern}_p}\, u. \left(\lceil m = u \rceil * \begin{cases} C_{\mathbf{Bern}_{\frac{1}{2}}}\, v.\, \lceil c = v \rceil & \text{if } u = 0 \\ C_{\mathbf{Bern}_{\frac{1}{2}}}\, v.\, \lceil c = \neg v \rceil & \text{if } u = 1 \end{cases} \right) && \text{(C-CONS)}
\end{aligned}
$$

The next is a crucial application of C-TRANSF. Because $\mathbf{Bern}_{\frac{1}{2}}$ is uniform between 0 and 1, if we choose the bijection $f : v \mapsto \neg v$, then the pushforward measure of $\mathbf{Bern}_{\frac{1}{2}}$ by f is also $\mathbf{Bern}_{\frac{1}{2}}$. Thus, applying C-TRANSF, we can rewrite $C_{\mathbf{Bern}_{\frac{1}{2}}}\, v.\, \lceil c = \neg v \rceil$ into $C_{\mathbf{Bern}_{\frac{1}{2}}}\, w.\, \lceil c = \neg\neg w \rceil$. Therefore,

$$
\begin{aligned}
& C_{\mathbf{Bern}_p}\, u. \left(\lceil m = u \rceil * \begin{cases} C_{\mathbf{Bern}_{\frac{1}{2}}}\, v.\, \lceil c = v \rceil & \text{if } u = 0 \\ C_{\mathbf{Bern}_{\frac{1}{2}}}\, v.\, \lceil c = \neg v \rceil & \text{if } u = 1 \end{cases} \right) \\
\vdash\ & C_{\mathbf{Bern}_p}\, u. \left(\lceil m = u \rceil * \begin{cases} C_{\mathbf{Bern}_{\frac{1}{2}}}\, v.\, \lceil c = v \rceil & \text{if } u = 0 \\ C_{\mathbf{Bern}_{\frac{1}{2}}}\, w.\, \lceil c = w \rceil & \text{if } u = 1 \end{cases} \right)
\end{aligned}
$$

Now, the formulas in the two cases of the case analysis are equivalent, so we can combine the two cases.

$$
\begin{aligned}
& C_{\mathbf{Bern}_p}\, u. \left(\lceil m = u \rceil * \begin{cases} C_{\mathbf{Bern}_{\frac{1}{2}}}\, v.\, \lceil c = v \rceil & \text{if } u = 0 \\ C_{\mathbf{Bern}_{\frac{1}{2}}}\, w.\, \lceil c = w \rceil & \text{if } u = 1 \end{cases} \right) && \text{(C-TRANSF)} \\
\vdash\ & C_{\mathbf{Bern}_p}\, u.\, \left(\lceil m = u \rceil * C_{\mathbf{Bern}_{\frac{1}{2}}}\, v.\, \lceil c = v \rceil \right)
\end{aligned}
$$

Now, we can almost read off that no matter what value m takes, the variable c is distributed as a Bernoulli distribution $\mathbf{Bern}_{\frac{1}{2}}$, and that implies that m and c are independent. Formally, we apply a sequence of steps to reach that conclusion. The first two steps are familiar applications of C-FRAME followed by SURE-MERGE. Then, C-ASSOC binds the two distribution together — because the second distribution $\mathbf{Bern}_{\frac{1}{2}}$ does not use any u drawn from the first distribution, we get the independent product of $\mathbf{Bern}_p \otimes \mathbf{Bern}_{\frac{1}{2}}$ as the result. After that, C-UNIT-R eliminates the conditioning. Last, PROD-SPLIT helps us pull the independent product in the distribution $\mathbf{Bern}_p \otimes \mathbf{Bern}_{\frac{1}{2}}$ into the independence conjunction asserted in our logic $m \perp\!\!\!\perp \mathbf{Bern}_p * c \perp\!\!\!\perp \mathbf{Bern}_{\frac{1}{2}}$.

$$
\begin{aligned}
& \boldsymbol{C}_{\mathbf{Bern}_p}\, u.\, \left(\lceil m = u \rceil * \boldsymbol{C}_{\mathbf{Bern}_{\frac{1}{2}}}\, v.\, \lceil c = v \rceil\right) \\
\vdash\ & \boldsymbol{C}_{\mathbf{Bern}_p}\, u.\, \boldsymbol{C}_{\mathbf{Bern}_{\frac{1}{2}}}\, v.\, \left(\lceil m = u \rceil * \lceil c = v \rceil\right) && \text{(C-FRAME)} \\
\vdash\ & \boldsymbol{C}_{\mathbf{Bern}_p}\, u.\, \boldsymbol{C}_{\mathbf{Bern}_{\frac{1}{2}}}\, v.\, \lceil m = u \wedge c = v \rceil && \text{(SURE-MERGE)} \\
\vdash\ & \boldsymbol{C}_{\mathbf{Bern}_p \otimes \mathbf{Bern}_{\frac{1}{2}}}\, (u, v).\, \lceil (m, c) = (u, v) \rceil && \text{(C-ASSOC)} \\
\vdash\ & (m, c) \perp\!\!\!\perp (\mathbf{Bern}_p \otimes \mathbf{Bern}_{\frac{1}{2}}) && \text{(C-UNIT-R)} \\
\vdash\ & m \perp\!\!\!\perp \mathbf{Bern}_p * c \perp\!\!\!\perp \mathbf{Bern}_{\frac{1}{2}} && \text{(PROD-SPLIT)}
\end{aligned}
$$

5.5.2 Markov Blankets

We next study *Markov Blankets* [Pearl, 2014] — a useful concept in Bayesian reasoning — to illustrate BLUEBELL's expressiveness (**RQ1** and **RQ2**). Intuitively, Markov blankets identify a set of variables that contains all useful information of a target set of variables. When knowing the values of variables in a Markov

blanket, we no longer need to worry about how the other variables influence the target set.

For concreteness, consider the program

$$x_1 :\approx d_1 ; x_2 :\approx d_2(x_1) ; x_3 :\approx d_3(x_2).$$

The program describes a Markov chain of three variables, where we first sample x_1, then sample x_2 from a distribution determined by the value of x_1, and last sample x_3 from a distribution determined by the value of x_2. These kinds of dependencies are ubiquitous in, for instance, hidden Markov models and Bayesian network representations of distributions.

Clearly, x_3 depends on x_2 and, indirectly, on x_1. However, Markov chains enjoy the memorylessness property: when fixing a variable in the chain, the variables that follow it are independent of the variables that preceded it. For our example, this means that conditioning on x_2, then x_1 and x_3 are independent (i.e. we can ignore the indirect dependencies). The memorylessness property is used in a lot of analysis of Markov chain based algorithms and also causal inference. In the following, we prove the memorylessness property for this specific program using BLUEBELL.

Using BLUEBELL's program rules, we can prove that after the program execution the output distribution satisfies the assertion

$$\boldsymbol{C}_{d_1}\, v_1.\, \Big(\lceil x_1 = v_1 \rceil * \boldsymbol{C}_{d_2(v_1)}\, v_2.\, \big(\lceil x_2 = v_2 \rceil * x_3 \mathrel{\&} d_3(v_2)\big)\Big)$$

We want to transform the assertion into:

$$\boldsymbol{C}_{\mu_2}\, v_2.\, \big(\lceil x_2 = v_2 \rceil * x_1 \mathrel{\&} \mu_1(v_2) * x_3 \mathrel{\&} d_3(v_2)\big)$$

for appropriate μ_2 and μ_1.

In probability theory, the proof of memorylessness is an application of Bayes' law: we compute the distribution of x_1, x_3 conditioned on x_2, using the distribution of x_2 conditioned on x_1 and the distribution of x_3 conditioned on x_2. In BLUEBELL we can reproduce the transformation using the joint conditioning rules, in particular the right-to-left direction of C-FUSE and the primitive rule C-UNASSOC.

Using these we can prove:

$$
\begin{aligned}
& \boldsymbol{C}_{d_1}\, v_1.\, \Big(\lceil x_1 = v_1 \rceil * \boldsymbol{C}_{d_2(v_1)}\, v_2.\, \big(\lceil x_1 = v_2 \rceil * x_3 \mathrel{\&} d_3(v_2)\big)\Big) \\
\vdash\ & \boldsymbol{C}_{d_1}\, v_1.\, \Big(\boldsymbol{C}_{d_2(v_1)}\, v_2.\, \big(\lceil x_1 = v_1 \rceil * \lceil x_1 = v_2 \rceil * x_3 \mathrel{\&} d_3(v_2)\big)\Big) && \text{(C-FRAME)} \\
\vdash\ & \boldsymbol{C}_{\mu_0}(v_1, v_2).\, \big(\lceil x_1 = v_1 \rceil * \lceil x_2 = v_2 \rceil * x_3 \mathrel{\&} d_3(v_2)\big) && \text{(C-FUSE)} \\
\vdash\ & \boldsymbol{C}_{\mu_2}\, v_2.\, \Big(\boldsymbol{C}_{\mu_1(v_2)}\, v_1.\, \big(\lceil x_1 = v_1 \rceil * \lceil x_2 = v_2 \rceil * x_3 \mathrel{\&} d_3(v_2)\big)\Big) && \text{(C-UNASSOC)} \\
\vdash\ & \boldsymbol{C}_{\mu_2}\, v_2.\, \Big(\lceil x_2 = v_2 \rceil * \boldsymbol{C}_{\mu_1(v_2)}\, v_1.\, \big(\lceil x_1 = v_1 \rceil * x_3 \mathrel{\&} d_3(v_2)\big)\Big) && \text{(SURE-STR-CONVEX)} \\
\vdash\ & \boldsymbol{C}_{\mu_2}\, v_2.\, \big(\lceil x_2 = v_2 \rceil * x_1 \mathrel{\&} \mu_1(v_2) * x_3 \mathrel{\&} d_3(v_2)\big) && \text{(C-EXTRACT)}
\end{aligned}
$$

where $d_1 \prec d_2 = \mu_0 = \mu_2 \prec \mu_1$. The existence of such μ_2 and μ_1 is a simple application of Bayes' law: $\mu_2(v_2) = \sum_{v_1 \in \mathbf{Val}} \mu_0(v_1, v_2)$, and $\mu_1(v_2)(v_1) = \frac{\mu_0(v_1, v_2)}{\mu_2(v_2)}$. The second to last step pull out $\lceil x_2 = v_2 \rceil$ from the second conditioning modality $\boldsymbol{C}_{\mu_2}\, v_2$. It is sound to pull out almost sure assertions like $\lceil x_2 = v_2 \rceil$ but not general assertions. Last, we use the derived rule C-EXTRACT to eliminate the second conditioning and extract the distribution of x_1 given x_2.

We see the ability of BLUEBELL to perform these manipulations as evidence that joint conditioning and independence form a sturdy abstraction over the semantic model (**RQ1**). The meta-reasoning required to manipulate the distributions and the conditioning modality — here it is showing the existence of μ_1, μ_2 such that $d_1 \prec d_2 = \mu_2 \prec \mu_1$ — are minimal and localized. Our abstraction and

rules also offer a good way to inject facts about distributions without interfering with the rest of the proof context.

5.5.3 Multi-party Secure Computation

In *multi-party secure computation* [Goldreich, 1998], the goal is for N parties to compute a function $f(x_1, \ldots, x_N)$ of some private data x_i owned by each party i, without revealing any more information about x_i than the output of f would reveal. For example, if f is addition, a secure computation of f can be used to compute the total number of votes without revealing who voted positively: some information would leak (e.g. if the total is non-zero then *somebody* voted positively) but only what is revealed by knowing the total and nothing more.

To achieve this objective, multi-party secure addition (MPSAdd) works by having the parties break their secret into N *secret shares* which individually look random, but the sum of which amounts to the original secret. These secret shares are then distributed to the other parties so that each party knows an incomplete set of shares of the other parties. Yet, each party can reliably compute the result of the function by computing a function of the received shares.

Barthe et al. [2019] analyze this example, where they prove the independence between each party's view and any other party's secrets *after the first round of communications.* However, there are two rounds of communication in the protocol, and after the second round, each party would get more information about the other party's values. So Barthe et al. [2019]'s proof does not ensure the end-to-end security of the protocol.

We aim to formally establish the end-to-end security of the protocol. As is very often the case, there is no single "canonical" way of specifying this kind of security property. For MPSAdd between three parties, for instance, one way to formalize security is a *unary specification* saying that, conditionally on the secret of party 1 and the sum of the other secrets, all the values received by 1 (we call this the *view* of 1) are independent from the secrets of the other parties. Roughly:

$$(x_1, x_2, x_3) \stackrel{\$}{\leftarrow} \mu_0 \vdash \mathbf{wp}\, [MPSAdd] \left\{ \exists \mu.\, \boldsymbol{C}_\mu\, (v, s).\, \begin{pmatrix} \lceil x_1 = v \wedge (x_2 + x_3) = s \rceil * \\ \mathsf{Own}(view_1) * \mathsf{Own}(x_2, x_3) \end{pmatrix} \right\}$$

where μ_0 is an arbitrary distribution of the three secrets among the three parties. The post-condition asserts that if we condition on the party 1's secret and the sum of party 2 and party 3's secrets, which the party 1 can infer based on the sum of the three secrets, then what party 1 can view, capture by $view_1$ is independent of party 2 and party 3's secrets. Here, the conditioning nicely expresses that the acceptable leakage is just the sum.

There is also a natural relational formulation of the security goal, and in the conference paper Bao et al. [2025] we provide BLUEBELL proofs for:

1. the unary specification;
2. the relational specification (not using unary proof);
3. the equivalence of the two specifications.

The relational specification can also be proved using Probabilistic Relational Hoare Logic Barthe et al. [2009]. In the following, we elaborate on the proof for the unary specification.

First, we can apply the program rules for loops and assignments to obtain the postcondition Q:

$$Q = X * R_{12} * R_3 * S * Sum$$

where $X = (x_1, x_2, x_3) \mathbin{\&} \mu_0$

$$R_{12} = \mathop{\Large *}_{i \in \{1,2,3\}} (r[i][1] \mathbin{\&} \mathsf{U}_p * r[i][2] \mathbin{\&} \mathsf{U}_p)$$

$$R_3 = \mathop{\Large *}_{i \in \{1,2,3\}} \lceil r[i][3] = x_i - r[i][1] - r[i][2] \rceil$$

$$S = \left\lceil \bigwedge_{i \in \{1,2,3\}} s[i] = r[1][i] + r[2][i] + r[3][i] \right\rceil$$

$$Sum = \lceil sum = s[1] + s[2] + s[3] \rceil$$

Now the goal is to show that Q entails the desired postcondition. As a first step we transform X into $(x_1, x_2 + x_2) \mathbin{\&} \mu$ by DIST-FUN. Then we condition on

$(x_1, x_2 + x_3, x_2, x_3)$ and the variables in R_{12}, obtaining:

$$
\boldsymbol{C}_{\mu'}(v_1, v_{23}, v_2, v_3).
\left(
\begin{array}{l}
\lceil x_1 = v_1 \wedge (x_2 + x_3) = v_{23} \wedge (x_2, x_3) = (v_2, v_3) \rceil * \\
\boldsymbol{C}_{\mathsf{U}_p}\, u_{11}.\, \boldsymbol{C}_{\mathsf{U}_p}\, u_{12}.\, \boldsymbol{C}_{\mathsf{U}_p}\, u_{21}.\, \boldsymbol{C}_{\mathsf{U}_p}\, u_{22}.\, \boldsymbol{C}_{\mathsf{U}_p}\, u_{31}.\, \boldsymbol{C}_{\mathsf{U}_p}\, u_{32}. \\
\quad \left\lceil
\begin{array}{l}
r[1][1] = u_{11} \wedge r[1][2] = u_{12} \wedge r[1][3] = v_1 - u_{11} - u_{12} \\
r[2][2] = u_{22} \wedge r[2][3] = v_2 - u_{21} - u_{22} \\
r[3][2] = u_{32} \wedge r[3][3] = (v_{23} - v_2) - u_{31} - u_{32} \\
s[1] = u_{11} + u_{21} + u_{31} \\
s[2] = u_{12} + u_{22} + u_{32} \\
s[3] = v_1 - u_{11} - u_{12} + v_2 - u_{21} - u_{22} + (v_{23} - v_2) - u_{31} - u_{32} \\
sum = s[1] + s[2] + s[3]
\end{array}
\right\rceil
\end{array}
\right)
$$

Here $\mu' = \mathsf{bind}(\mu_0, (x_1, x_2, x_3) \mapsto \mathsf{unit}(x_1, x_2 + x_3, x_2))$. We already weakened the assertion by forgetting the information about $r[2][1]$ and $r[3][1]$, which are not part of $view_1$.

Now we perform a change of variables using C-TRANSF, to express our equalities in terms of $u'_{21} = u_{21} - v_2$ instead of u_{21} and $u'_{31} = u_{31} - (v_{23} - v_2)$ instead of u_{31}. To justify the change we simply observe that, for all $n \in \mathbb{Z}_p$, the function $f_n(u) = (u - n) \mod p$ is a bijection and $\mathsf{U}_p \circ f_n^{-1} = \mathsf{U}_p$. This gives us,

with some simple arithmetic simplifications:

$$\boldsymbol{C}_{\mu'}(v_1, v_{23}, v_2, v_3).\left(\begin{array}{l}\lceil x_1 = v_1 \wedge (x_2 + x_3) = v_{23} \wedge (x_2, x_3) = (v_2, v_3)\rceil * \\ \boldsymbol{C}_{\mathsf{U}_p}\, u_{11}.\, \boldsymbol{C}_{\mathsf{U}_p}\, u_{12}.\, \boldsymbol{C}_{\mathsf{U}_p}\, u'_{21}.\, \boldsymbol{C}_{\mathsf{U}_p}\, u_{22}.\, \boldsymbol{C}_{\mathsf{U}_p}\, u'_{31}.\, \boldsymbol{C}_{\mathsf{U}_p}\, u_{32}. \\ \quad \left\lceil \begin{array}{l} r[1][1] = u_{11} \wedge r[1][2] = u_{12} \wedge r[1][3] = v_1 - u_{11} - u_{12} \\ r[2][2] = u_{22} \wedge r[2][3] = -u'_{21} - u_{22} \\ r[3][2] = u_{32} \wedge r[3][3] = -u'_{31} - u_{32} \\ s[1] = u_{11} + u'_{21} + u'_{31} + v_{23} \\ s[2] = u_{12} + u_{22} + u_{32} \\ s[3] = v_1 - u_{11} - u_{12} - u'_{21} - u_{22} - u'_{31} - u_{32} \\ sum = s[1] + s[2] + s[3] \end{array} \right\rceil \end{array}\right)$$

In particular, we removed all dependencies on v_2 from the inner formula. We can now apply C-ASSOC to collapse all the inner conditioning into a single one:

$$\boldsymbol{C}_{\mu'}(v_1, v_{23}, v_2, v_3).\left(\begin{array}{l}\lceil x_1 = v_1 \wedge (x_2 + x_3) = v_{23} \wedge (x_2, x_3) = (v_2, v_3)\rceil * \\ \boldsymbol{C}_{U(v_1, v_{23})}\, u.\, \lceil * \rceil view_1 = u\end{array}\right)$$

where $U(v_1, v_{23}) = (v \leftarrow \mathbf{U}_p \otimes \ldots \otimes \mathbf{U}_p;\ \mathbf{return}\, g(v))$ takes the six independent samples from U_p and returns the values for each of the components of $view_1$ (which justifies the dependency on v_1 and v_{23}). Finally, we split $\mu' = \mathsf{bind}(\mu, \kappa)$

obtaining:

$$\boldsymbol{C}_{\mu'}(v_1, v_{23}, v_2, v_3).\ \begin{pmatrix} \lceil x_1 = v_1 \wedge (x_2 + x_3) = v_{23} \wedge (x_2, x_3) = (v_2, v_3) \rceil * \\ \boldsymbol{C}_{U(v_1, v_{23})}\, u.\ \lceil view_1 = u \rceil \end{pmatrix}$$

$$\vdash \boldsymbol{C}_{\mu'}(v_1, v_{23}, v_2, v_3).\ \begin{pmatrix} \lceil x_1 = v_1 \wedge (x_2 + x_3) = v_{23} \rceil * \\ \lceil (x_2, x_3) = (v_2, v_3) \rceil * \\ view_1 \mathrel{\&} U(v_1, v_{23}) \end{pmatrix} \quad \text{(SURE-MERGE, C-UNIT-R)}$$

$$\vdash \boldsymbol{C}_{\tilde{\mu}_0}(v_1, v_{23}).\ \begin{pmatrix} \lceil x_1 = v_1 \wedge (x_2 + x_3) = v_{23} \rceil * \\ \boldsymbol{C}_{\kappa(v_1, v_{23})}(v_2, v_3). \\ \quad \left(\lceil (x_2, x_3) = (v_2, v_3) \rceil * view_1 \mathrel{\&} U(v_1, v_{23}) \right) \end{pmatrix}$$

(C-UNASSOC, SURE-STR-CONVEX)

$$\vdash \boldsymbol{C}_{\tilde{\mu}_0}(v_1, v_{23}).\ \begin{pmatrix} \lceil x_1 = v_1 \wedge (x_2 + x_3) = v_{23} \rceil * \\ (x_2, x_3) \mathrel{\&} \kappa(v_1, v_{23}) * view_1 \mathrel{\&} U(v_1, v_{23}) \end{pmatrix} \quad \text{(C-EXTRACT)}$$

$$\vdash \boldsymbol{C}_{\tilde{\mu}_0}(v_1, v_{23}).\ \begin{pmatrix} \lceil x_1 = v_1 \wedge (x_2 + x_3) = v_{23} \rceil * \\ view_1 \mathrel{\&} U(v_1, v_{23}) * \exists \mu_{23}.\ (x_2, x_3) \mathrel{\&} \mu_{23} \end{pmatrix}$$

This gets us the desired postcondition and concludes the proof.

5.5.4 Von Neumann Extractor

A randomness extractor is a mechanism that transforms a stream of "low-quality" randomness sources into a stream of "high-quality" randomness sources. The von Neumann extractor [von Neumann, 1951] is perhaps the earliest instance of such a mechanism, and it converts a stream of independent coins with the same bias p into a stream of independent *fair* coins. Verifying

the correctness of the extractor requires careful reasoning under conditioning, and showcases the use of C-WP-SWAP in a unary setting, which gives positive answer to **RQ2** and **RQ4**.

We can model the extractor, up to $N \in \mathbb{N}$ iterations, in our language[3] as shown in fig. 5.8. The program repeatedly flips two biased coins, and outputs the outcome of the first coin if the outcomes where different, otherwise it retries. As an example, we prove in BLUEBELL that the bits produced in out are independent fair coin flips. Formally, for ℓ produced bits, we want the following to hold:

```
def vn(N):
  len := 0
  repeat N:
    coin₁ :≈ Ber(p)
    coin₂ :≈ Ber(p)
    if coin₁ ≠ coin₂ then:
      out[len] := coin₁
      len := len+1
```

Figure 5.8: Von Neumann extractor.

$$\mathbf{Out}_\ell \triangleq out[0] \mathbin{\text{$\wr$}} \mathbf{Bern}_{\frac{1}{2}} * \cdots * out[\ell-1] \mathbin{\text{$\wr$}} \mathbf{Bern}_{\frac{1}{2}}.$$

To know how many bits were produced, however, we need to condition on len obtaining the specification (recall $P \Vdash Q \triangleq P \wedge \mathsf{own}_{\mathbf{Var}} \vdash Q \wedge \mathsf{own}_{\mathbf{Var}}$):

$$\Vdash wp\,[1\colon vn(N)]\,\{\exists \mu.\, \boldsymbol{C}_\mu\, \ell.\, (\lceil len = \ell \leq N \rceil * Out_\ell)\}$$

The BLUEBELL proof of this specification is shown in the outline in fig. 5.9. The postcondition straightforwardly generalizes to a loop invariant

$$P(i) = \exists \mu.\, \boldsymbol{C}_\mu\, \ell.\, (\lceil len = \ell \leq i \rceil * Out_\ell)$$

At step (5.2) we show, by using C-UNIT-L and the definition of $\lceil \cdot \rceil$, that we can

[3]While technically our language does not support arrays, they can be easily encoded as a collection of N variables.

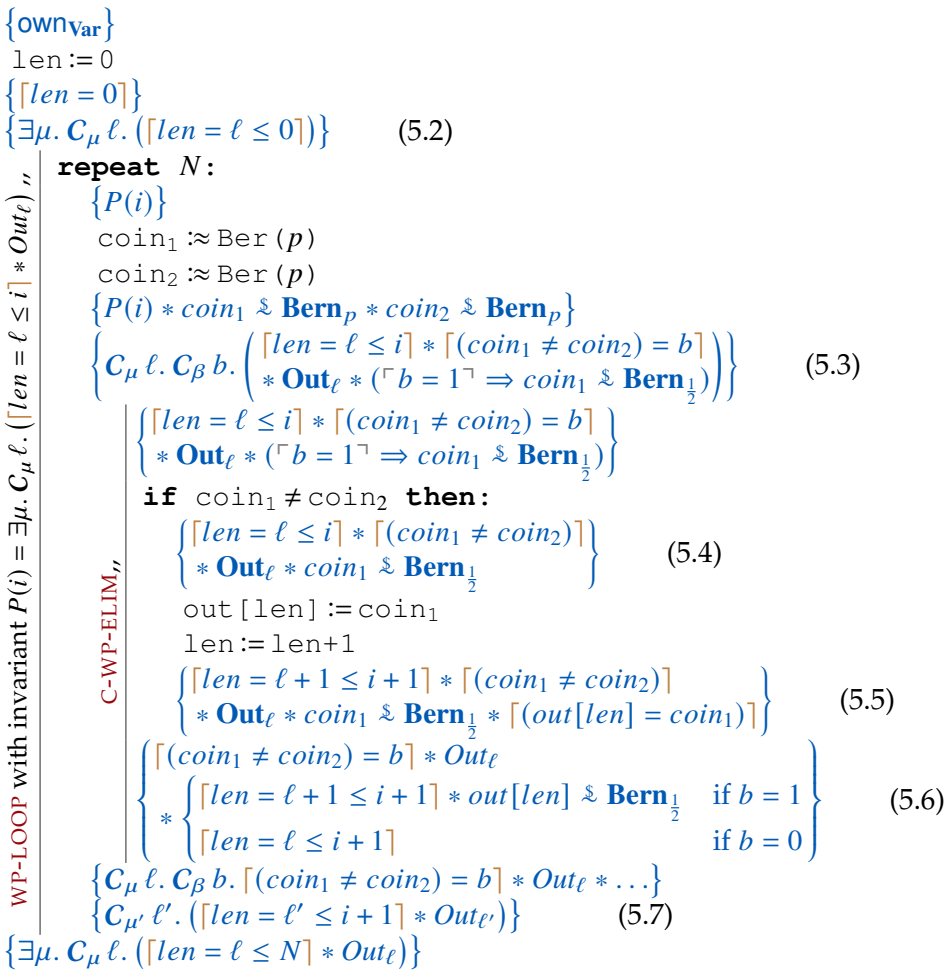


Figure 5.9: Proof outline of the Von Neumann extractor example.

obtain the loop invariant with $i = 0$: $P(0) = \exists\mu.\, \boldsymbol{C}_\mu\, \ell.\, \big(\lceil len = \ell \leq 0\rceil * Out_0\big) = \exists\mu.\, \boldsymbol{C}_\mu\, \ell.\, \big(\lceil len = \ell \leq 0\rceil\big)$.

For the proof of the body of the loop we can assume $P(i)$ and we need to prove the postcondition $P(i+1)$. After sampling the two coins, at step (5.3) we apply the crucial insight behind the extractor. The key idea is that with some probability q the two coins will be different, in which case the outcomes of the two coins can be either $(0,1)$ or $(1,0)$, which both have the same probability $p(1-p)$. Therefore, if the coins are different, $coin_1 = 0$ and $coin_1 = 1$ have the same probability, i.e. $coin_1$ looks like a fair coin.

BLUEBELL is capable of representing this reasoning as follows.

We start with two independent biased coins, which we can combine into a random variable $(coin_1 \neq coin_2, coin_1)$ recording whether the two outcomes were different and the outcome of the first coin. We use PROD-UNSPLIT and DIST-FUN to derive:

$$coin_1 \twoheadrightarrow \mathbf{Bern}_p * coin_2 \twoheadrightarrow \mathbf{Bern}_p \vdash (coin_1 \neq coin_2, coin_1) \twoheadrightarrow \mu_0$$

where $\mu_0 \triangleq \mathsf{bind}(\mathbf{Bern}_p \otimes \mathbf{Bern}_p, (coin_1, coin_2) \mapsto (coin_1 \neq coin_2, coin_1))$ Now we can reformulate μ_0 to reflect our above-mentioned insight into why this extractor works: there exists some probability q and q' such that

$$\mu_0 = \beta \prec \kappa \qquad \beta \triangleq \mathbf{Bern}_q \qquad \kappa(1) \triangleq \mathbf{Bern}_{\frac{1}{2}} \qquad \kappa(0) \triangleq \mathbf{Bern}_{q'}$$

Here one first determines whether the two coins will be different or equal using a Bernoulli distribution that assigns probability q for them to be different; here q can be obtained using a function of p. The process then generates c_1 accordingly using κ: in the "different" branch ($b = 1$) the first coin is distributed as $\mathbf{Bern}_{\frac{1}{2}}$ while in the "equal" branch ($b = 0$) the first coin is distributed with some bias q' (also a function of p).

So using $\mu_0 = \beta \prec \kappa$ we derive:

$$(coin_1 \neq coin_2, coin_1) \twoheadrightarrow (\beta \prec \kappa)$$

$$\vdash \boldsymbol{C}_{\beta \prec \kappa}\,(b, c_1).\, \lceil (coin_1 \neq coin_2) = b \wedge coin_1 = c_1 \rceil \qquad \text{(C-UNIT-R)}$$

$$\vdash \boldsymbol{C}_{\beta}\, b.\, \boldsymbol{C}_{\kappa(b)}\, c_1.\, \lceil (coin_1 \neq coin_2) = b \rceil * \lceil coin_1 = c_1 \rceil \qquad \text{(C-FUSE)}$$

$$\vdash \boldsymbol{C}_{\beta}\, b.\, \left(\lceil (coin_1 \neq coin_2) = b \rceil * \boldsymbol{C}_{\kappa(b)}\, c_1.\, \lceil coin_1 = c_1 \rceil\right) \qquad \text{(SURE-STR-CONVEX)}$$

$$\vdash \boldsymbol{C}_{\beta}\, b.\, \left(\lceil (coin_1 \neq coin_2) = b \rceil * \ulcorner b = 1 \urcorner \Rightarrow \boldsymbol{C}_{\mathbf{Bern}_{\frac{1}{2}}}\, c_1.\, \lceil coin_1 = c_1 \rceil\right) \qquad \text{(C-CONS)}$$

$$\vdash \boldsymbol{C}_{\beta}\, b.\, \left(\lceil (coin_1 \neq coin_2) = b \rceil * \ulcorner b = 1 \urcorner \Rightarrow coin_1 \twoheadrightarrow \mathbf{Bern}_{\frac{1}{2}}\right) \qquad \text{(C-UNIT-R)}$$

The application of C-FUSE allows us to first condition on $coin_1 \neq coin_2$, and then the first coin. We can then weaken the case where $b = 0$ and only record that if $b = 1$ then $coin_1$ is a fair coin.

This takes us through step (5.3) of fig. 5.9. Now the precondition of the if statement is conditional on len and $coin_1 \neq coin_2$. Intuitively, we want to evaluate the effects of the if statement in the two possible outcomes and put together the results. This is precisely the purpose of the C-WP-SWAP rule, which together with C-CONS gives us the derived rule:

$$\text{C-WP-ELIM}\quad \frac{\forall v \in \operatorname{supp}(\mu).\, P(v) \Vdash \mathbf{wp}\, t\, \{Q(v)\}}{\boldsymbol{C}_\mu\, v.\, P(v) \Vdash \mathbf{wp}\, t\, \{\boldsymbol{C}_\mu\, v.\, Q(v)\}}$$

By applying the rule twice (first on the conditioning on len, and then on the conditioning on $coin_1 \neq coin_2$), we can process the if statement case by case, and then combine the postconditions obtained in each case. For the conditional with the guard $coin_1 \neq coin_2$, the false branch is a **`skip`** (omitted), so it preserves the precondition with $b = 0$. In the true branch, starting with the precondition at (5.4), we apply WP-ASSIGN to the assignments to obtain (5.5). Last, we combine the results from both branches by making the overall postcondition at (5.6) to be parametric on the value of b (and ℓ).

The last non-obvious step is (5.7) in fig. 5.9, where we show that the conditional postcondition of the if statement implies the loop invariant $P(i+1)$. Let

$$K(\ell, b) = \begin{cases} \lceil len = \ell + 1 \leq i + 1 \rceil * out[len] \Rsh \mathbf{Bern}_{\frac{1}{2}} & \text{if } b = 1 \\ \lceil len = \ell \leq i + 1 \rceil & \text{if } b = 0 \end{cases}$$

then the step is proven as follows:

$$
\begin{aligned}
& \boldsymbol{C}_{\mu}\, \ell.\, \boldsymbol{C}_{\beta}\, b.\, \big(\lceil (coin_1 \neq coin_2) = b \rceil * Out_{\ell} * K(\ell, b)\big) \\
\vdash\ & \boldsymbol{C}_{\mu \otimes \beta}(\ell, b).\, \big(\lceil (coin_1 \neq coin_2) = b \rceil * Out_{\ell} * K(\ell, b)\big) && \text{(C-ASSOC)} \\
\vdash\ & \boldsymbol{C}_{\mu \otimes \beta}(\ell, b).\, \big(Out_{\ell} * K(\ell, b)\big) && \text{(C-CONS)} \\
\vdash\ & \boldsymbol{C}_{\mu''}(\ell', \ell).\, \begin{cases} \lceil len = \ell' \leq i + 1 \rceil * \mathbf{Out}_{\ell'-1} * out[len] \mathrel{\$} \mathbf{Bern}_{\frac{1}{2}} & \text{if } \ell' = \ell + 1 \\ \lceil len = \ell' \leq i + 1 \rceil * Out_{\ell'} & \text{if } \ell' = \ell \end{cases} && \text{(C-TRANSF)} \\
\vdash\ & \boldsymbol{C}_{\mu'' \circ \pi_1^{-1}}\, \ell'.\, \big(\lceil len = \ell' \leq i + 1 \rceil * Out_{\ell'}\big) && \text{(C-DIST-PROJ)} \\
\vdash\ & \exists \mu'.\, \boldsymbol{C}_{\mu'}\, \ell'.\, \big(\lceil len = \ell' \leq i + 1 \rceil * Out_{\ell'}\big)
\end{aligned}
$$

The application of C-TRANSF uses the function $f(\ell, b) = (\ell + b, \ell)$ to introduce the new ℓ' and then we project away the unused ℓ using the derived C-DIST-PROJ (note that the rule applies to $\lceil \cdot \rceil$ assertions and multiple ownership assertions in a separating conjunction thanks to PROD-SPLIT and PROD-UNSPLIT).

5.6 Related Work

Research on deductive verification of probabilistic programs has developed a wide range of techniques that employ *unary* and *relational* styles of reasoning. BLUEBELL advances the state of the art in both styles, by coherently unifying the strengths of both. Since this chapter focuses on the unary fragment of BLUEBELL, here we compare BLUEBELL with unary-style deductive techniques in more details, and overview relational-style deductive techniques only at a high-level. At the end, we briefly survey other relevant techniques of reasoning about probabilistic programs.

Other Unary Reasoning Techniques for Probabilistic Programs Outside of probabilistic separation logic, another line of unary deductive verification techniques is the expectation-based approach. The high-level idea is to reason about expected quantities of probabilistic programs via a weakest-pre-expectation operator that propagates information about expected values backwards through the program. The approach has been classically used to verify randomized algorithms [Kozen, 1983, Morgan et al., 1996, Kaminski et al., 2016, Kaminski, 2019, Aguirre et al., 2021, Bartocci et al., 2020]. These logics offer ergonomic principles for expectations, but do not aim at unifying principles for analyzing more general classes of properties or proof techniques, like we attempt here. Ellora [Barthe et al., 2018] proposes an assertion-based logic to overcome the limitation of working only with expectations. But it does not support reasoning about separation or conditioning.

Detailed Comparison with Lilac [Li et al., 2023a] Now that we have introduced the unary fragment of BLUEBELL thoroughly, we revisit its comparison with Lilac [Li et al., 2023a]. Lilac supports reasoning about independence and conditional independence in continuous distributions, thanks to their measure-theoretic model based on Borel spaces. BLUEBELL also uses a measure-theory based model, similar to Lilac, but is limited to discrete distributions. The limitation is imposed because we are only able to prove some key lemmas [Bao et al., 2025, Lemma C.1 - C.7] for discrete measures, though we speculate that they also hold for continuous measures. These lemmas are used in the proof of key rules such as C-WP-SWAP and C-FRAME, and Lilac's proof system does not include similar rules.

Also, while BLUEBELL uses Lilac's independent product as a model of sep-

arating conjunction, it differs from Lilac in three aspects: (1) the treatment of ownership, (2) support for mutable state, and (3) the model of conditioning.

In Lilac, ownership over program variables and expressions is defined as measurability. In BLUEBELL, however, we define ownership as almost-measurability, which is required to support inferences like $\mathsf{Own}(x) * \lceil x = y \rceil \vdash \mathsf{Own}(y)$ These inferences were implicitly used in the first version of Lilac, but were not valid in its model. Their arxiv version Li et al. [2023b] fixes the issue by changing the meaning of $\lceil x = y \rceil$, while our fix changes the meaning of ownership (and we define $\lceil E \rceil$ assertions based on regular ownership).

Lilac works with immutable state [Staton, 2020], which simplifies reasoning in certain contexts (e.g., the frame rule and the if rule). BLUEBELL's model supports mutable state through a creative use of permissions, obtaining a clean frame rule, at the cost of some predictable bookkeeping.

The more significant difference with Lilac is however in the definition of the conditioning modality. Lilac's modality $\mathbf{C}_{v \leftarrow E}\, P(v)$ roughly corresponds to the BLUEBELL assertion $\exists \mu.\, \boldsymbol{C}_\mu\, v.\, (\lceil E = v \rceil * P(v))$. The difference is not merely syntactic, and requires changing the model of the modality. For example, Lilac's modality satisfies $\mathbf{C}_{v \leftarrow E}\, P_1(v) \wedge \mathbf{C}_{v \leftarrow E}\, P_2(v) \vdash \mathbf{C}_{v \leftarrow E}\, (P_1(v) \wedge P_2(v))$, but the analogous rule $\boldsymbol{C}_\mu\, v.\, K_1(v) \wedge \boldsymbol{C}_\mu\, v.\, K_2(v) \vdash \boldsymbol{C}_\mu\, v.\, (K_1(v) \wedge K_2(v))$ is unsound in BLUEBELL: the meaning of the modalities in the premise ensures the *existence* of two kernels κ_1 and κ_2 supporting K_1 and K_2 respectively, but the conclusion requires the existence of a *single* kernel supporting both K_1 and K_2. Lilac's rule holds because when one conditions on a random variable, the corresponding kernels are unique. We did not find losing this rule limiting.

On the other hand, Lilac's conditioning has two disadvantages: (i) it does not record the distribution of E, losing this information when conditioning, (ii) it does not generalize to the relational setting. Even considering only the unary setting, having access to the distribution μ unlocks a number of new rules (e.g. C-UNIT-R and C-FUSE). In particular, the rules of BLUEBELL provide more ways to convert a conditional assertion back into an unconditional one, which is crucial when the end goal is not under conditioning but conditioning is helpful in intermediate steps.

Relational Reasoning We summarize several extensions of pRHL-style relational reasoning, the vanilla version of which is overviewed in section 5.1

Polaris [Tassarotti and Harper, 2019], is an early instance of a probabilistic relational (concurrent) separation logic. The motivation is to reconcile the relational lifting based reasoning with the semantics of concurrent programs. However, separation in Polaris is again classic disjointness of state and is not related to (conditional) independence in the style of PSL, Lilac, or BLUEBELL.

Gregersen et al. [2024] recently proposed Clutch, a logic to prove contextual refinement in a probabilistic higher-order language, where "out of order" couplings between samplings are achieved by using ghost code that pre-samples some assignments, a technique inspired by *prophecy variables* [Jung et al., 2019]. In the conference version Bao et al. [2025], we showed how BLUEBELL can resolve this issue without ghost code (in the context of first-order imperative programs) by using framing and probabilistic independence creatively. In contrast to BLUEBELL, Clutch can only express relational properties; it also uses separation but with its classical interpretation as disjointness of deterministic state.

CHAPTER 6

DISCUSSION

In this thesis, we show three extensions of probabilistic separation logic for reasoning about probabilistic programs involving not only independence, but also negative dependence or conditional independence. We demonstrated that we can use these program logics to formalize the correctness of a range of randomized algorithms. With rules that utilize probabilistic (in)dependencies for modular reasoning, these program logic allows relatively compact and clean formal proofs. In this final chapter, we discuss more related work and directions for future work.

6.1 Related Work

Concurrent Developments in Probabilistic Separation Logic During the course of my doctoral study, various papers have explored other variations of probabilistic separation logic. Dal Lago et al. [2024] extends PSL to prove computational independence, which relaxes probabilistic independence by only requiring variables' distribution to be computationally indistinguishable from independence. They demonstrate their logic by formalizing several simple cryptographic protocols developed to guard against against adversaries with computational power. Ho et al. [2025] extends Lilac [Li et al., 2023a] to support a probabilistic programming language that can not only sample from distributions but also condition based on observations and soft constraints. They then apply the logic to reason about a range of classic examples in Bayesian reasoning, including Bayesian coin flip, collider Bayesian network, and the burglar alarm model. Yan et al. [2024] mechanizes a variation of PSL in Isabelle/HOL

and uses it to mechanize the security of several probabilistic oblivious algorithms. Similar to the unary fragment of BLUEBELL, Ignacij Jereb and Simpson [2025] also tackles the problem of formulating a clean frame rule for imperative probabilistic programs, using a model more similar to PSL's than Lilac's. However, they do not provide a full program logic with a soundness proof.

On the foundation side, Li et al. [2024] bridges the gap between Lilac's independent product with the standard independent product in probability theory by showing that they are equivalent up to a suitable equivalence of categories.

Quantitative Separation Logic An alternative separation logic for probabilistic programs is Quantitative Separation Logic (QSL) by Batz et al. [2019]. QSL is developed to unify separation logic for heap-manipulating programs and *weakest preexpectation*, which captures program states using *expectations*, i.e., numerical quantities, instead of assertions. QSL uses connectives in the bunched logic to construct new expectations over the existing one; for example, $Q_1 * Q_2$ is the maximum of the numerical product of the quantity $Q_1(h_1)$ and $Q_2(h_2)$, where h_1, h_2 are disjoint subheaps of the current heap. As a result of these design choices, applications of QSL often involve reasoning about some quantities relating to the heap — for example, the expected length of a list, which are quite different from the applications of PSL. QSL is automated in Batz et al. [2022] through using a weakest preexpectation calculus to reason about programs and reducing entailments checking to standard separation logic's entailments checking by Echenim et al. [2020].

Automated Verification of Probabilistic Programs Automated verification of probabilistic programs is an active field of research. For example, there

is a long line of work in automating weakest pre-expectation style calculus for probabilistic programs [Gretz et al., 2013, Chen et al., 2015, Feng et al., 2017, Batz et al., 2023, Bao et al., 2024], where the bottleneck is to compute the weakest pre-expectation of probabilistic loops. Bartocci et al. [2019, 2020] also utilize moment-based analysis to develop automated tools for analyzing probabilistic programs. Probabilistic model checking provides verification tools [Dehnert et al., 2017, Kwiatkowska et al., 2002] of a different flavor. They typically target probabilistic models specified as discrete-time Markov chains (DTMCs), continuous-time Markov chains (CTMCs), or Markov decision processes (MDPs) and ask users to specify desired properties using temporal formulas involving probabilities. While probabilistic programs can be translated into such probabilistic models, it is unclear how to encode probabilistic (in)dependencies as a property in these tools, whose specification languages do not naturally support relating the probabilities of more than one events.

6.2 Directions for Future Work

While recent work shows the potential of probabilistic separation logic, more work has to be done to make this technique appealing to practitioners of probabilistic programs. Here we envision some directions for further development of probabilistic separation logic. First of all, currently it takes expertise to manually construct the formal proof, so automation of the proof construction is important to make it easier to adopt. Second, users may want to use a richer set of language features, e.g., concurrency, the conditioning operator, a more general loop, or higher-order programs, and it would be helpful to enrich probabilistic separation logic to support these language features as well. Third, it would

be great to have an unifying framework that is also extensible, so one does not need to apply different program logic to prove different properties.

In my most recent project, I work on automating PSL for loop-less probabilistic programs, joint with former Cornell undergraduate student Jessica Cho and my advisor Justin Hsu. We draw inspiration from the automation of standard separation logic [Berdine et al., 2004, 2005, 2006, Appel, 2011, Piskac et al., 2014] to develop a syntax-directed version of PSL. However, we observe that probabilistic programs introduce several additional challenges. For example, automating PSL must account for conditionals and loops that branch on randomized guards and apply custom axioms about independence at appropriate places. Moreover, various verification tasks demand the ability to infer the distribution of an expression given an assertion. We view our work as a pilot study that demonstrates one approach to automating probabilistic separation logic, with significant opportunities for further exploration. For instance, how can we effectively leverage the post-condition to prune the proof during construction? Is there a loop rule particularly well-suited to automation? Furthermore, can existing techniques for synthesizing probabilistic loop invariants be adapted to support the automation of probabilistic separation logic in programs with loops?

Advances in automation are also closely tied to the foundational development of probabilistic separation logic itself, which may lead to a cleaner and more automatable set of rules. In addition, a unified and extensible framework would enable automation of new extensions to build upon the automation infrastructure developed for existing ones.

BIBLIOGRAPHY

Stephen Abbott. *Understanding analysis*. Springer, 2015.

Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*, volume 8. Addison-Wesley Reading, 1995.

Samson Abramsky and Jouko A. Väänänen. From IF to BI. *Synthese*, 167(2): 207–230, 2009. URL https://doi.org/10.1007/s11229-008-9415-6.

Alejandro Aguirre, Gilles Barthe, Justin Hsu, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. A pre-expectation calculus for probabilistic sensitivity. *Proceedings of the ACM on Programming Languages*, 5(POPL), January 2021. URL https://doi.org/10.1145/3434333.

Nima Anari, Shayan Oveis Gharan, and Alireza Rezaei. Monte carlo markov chain algorithms for sampling strongly rayleigh distributions and determinantal point processes. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, editors, *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 103–115, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR. URL https://proceedings.mlr.press/v49/anari16.html.

Andrew W Appel. Verismall: Verified smallfoot shape analysis. In *International Conference on Certified Programs and Proofs*, pages 231–246. Springer, 2011.

Jialu Bao, Simon Docherty, Justin Hsu, and Alexandra Silva. A bunched logic for conditional independence. In *Proceedings of the 36th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '21, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781665448956. URL https://doi.org/10.1109/LICS52264.2021.9470712.

Jialu Bao, Marco Gaboardi, Justin Hsu, and Joseph Tassarotti. A separation logic for negative dependence. *Proceedings of the ACM on Programming Languages*, 6 (POPL), January 2022. URL https://doi.org/10.1145/3498719.

Jialu Bao, Nitesh Trivedi, Drashti Pathak, Justin Hsu, and Subhajit Roy. Data-driven invariant learning for probabilistic programs. *Formal Methods in System Design*, pages 1–29, 2024.

Jialu Bao, Emanuele D'Osualdo, and Azadeh Farzan. Bluebell: An alliance of relational lifting and independence for probabilistic reasoning. *Proceedings of the ACM on Programming Languages*, 9(POPL), January 2025. URL https://doi.org/10.1145/3704894.

Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and machine learning: Limitations and opportunities*. MIT press, 2023.

Gilles Barthe, Benjamin Grégoire, and Santiago Zanella Béguelin. Formal certification of code-based cryptographic proofs. In *Proceedings of the 36th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 90–101, 2009. URL https://doi.org/10.1145/1594834.1480894.

Gilles Barthe, Boris Köpf, Federico Olmedo, and Santiago Zanella-Béguelin. Probabilistic relational reasoning for differential privacy. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 35(3):1–49, 2013.

Gilles Barthe, Thomas Espitau, Benjamin Grégoire, Justin Hsu, Léo Stefanesco, and Pierre-Yves Strub. Relational reasoning via probabilistic coupling. In *International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR)*, pages 387–401. Springer, 2015.

Gilles Barthe, Noémie Fong, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. Advanced probabilistic couplings for differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 55–67, 2016a.

Gilles Barthe, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. Proving differential privacy via probabilistic couplings. In *2016 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–10, 2016b. URL https://doi.org/10.1145/2933575.2934554.

Gilles Barthe, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. Coupling proofs are probabilistic product programs. *SIGPLAN Not.*, 52(1):161–174, January 2017. ISSN 0362-1340. URL https://doi.org/10.1145/3093333.3009896.

Gilles Barthe, Thomas Espitau, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. An assertion-based program logic for probabilistic programs. In *Programming Languages and Systems*, pages 117–144, Cham, 2018. Springer International Publishing.

Gilles Barthe, Justin Hsu, and Kevin Liao. A probabilistic separation logic. *Proc. ACM Program. Lang.*, 4(POPL), December 2019. URL https://doi.org/10.1145/3371123.

Ezio Bartocci, Laura Kovács, and Miroslav Stankovič. Automatic generation of moment-based invariants for prob-solvable loops. In *Automated Technology for Verification and Analysis*, Taipei, Taiwan, 2019. URL https://doi.org/10.1007/978-3-030-31784-3_15.

Ezio Bartocci, Laura Kovács, and Miroslav Stankovič. Mora-automatic generation of moment-based invariants. In *International Conference on Tools and Al-*

gorithms for the Construction and Analysis of Systems (TACAS), Dublin, Ireland, 2020. URL https://doi.org/10.1007/978-3-030-45190-5_28.

Kevin Batz, Benjamin Lucien Kaminski, Joost-Pieter Katoen, Christoph Matheja, and Thomas Noll. Quantitative separation logic: a logic for reasoning about probabilistic pointer programs. *Proc. ACM Program. Lang.*, 3(POPL), January 2019. URL https://doi.org/10.1145/3290347.

Kevin Batz, Ira Fesefeldt, Marvin Jansen, Joost-Pieter Katoen, Florian Keßler, Christoph Matheja, and Thomas Noll. Foundations for entailment checking in quantitative separation logic (extended version). *arXiv preprint arXiv:2201.11464*, 2022.

Kevin Batz, Mingshuai Chen, Sebastian Junges, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. Probabilistic program verification via inductive synthesis of inductive invariants. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 410–429. Springer, 2023.

Luca Becchetti, Andrea Clementi, Emanuele Natale, Francesco Pasquale, and Gustavo Posta. Self-stabilizing repeated balls-into-bins. *International Symposium on Distributed Computing (DISC)*, 2019.

Ioana O. Bercea and Guy Even. Dynamic dictionaries for multisets and counting filters with constant time operations. *Algorithmica*, 85(6):1786–1804, December 2022. ISSN 0178-4617. URL https://doi.org/10.1007/s00453-022-01057-0.

Josh Berdine, Cristiano Calcagno, and Peter W. O'Hearn. A decidable fragment of separation logic. In *Proceedings of the 24th International Conference on Foundations of Software Technology and Theoretical Computer Science*, FSTTCS'04, page

97–109, Berlin, Heidelberg, 2004. Springer-Verlag. ISBN 3540240586. URL https://doi.org/10.1007/978-3-540-30538-5_9.

Josh Berdine, Cristiano Calcagno, and Peter W O'hearn. Symbolic execution with separation logic. In *Programming Languages and Systems: Third Asian Symposium, APLAS 2005, Tsukuba, Japan, November 2-5, 2005. Proceedings 3*, pages 52–68. Springer, 2005.

Josh Berdine, Cristiano Calcagno, and Peter W O'hearn. Smallfoot: Modular automatic assertion checking with separation logic. In *Formal Methods for Components and Objects: 4th International Symposium, FMCO 2005, Amsterdam, The Netherlands, November 1-4, 2005, Revised Lectures 4*, pages 115–137. Springer, 2006.

Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970. URL https://doi.org/10.1145/362686.362692.

James Blustein and Amal El-Maazawi. Bloom filters. a tutorial, analysis, and survey. *Halifax, NS: Dalhousie University*, pages 1–31, 2002.

Julius Borcea, Petter Brändén, and Thomas M. Liggett. Negative dependence and the geometry of polynomials. *Journal of the American Mathematical Society*, 22(2):521–567, 2009.

Prosenjit Bose, Hua Guo, Evangelos Kranakis, Anil Maheshwari, Pat Morin, Jason Morrison, Michiel Smid, and Yihui Tang. On the false-positive rate of bloom filters. *Information Processing Letters*, 108(4):210–213, October 2008. ISSN 0020-0190. URL https://doi.org/10.1016/j.ipl.2008.05.018.

Stephen Brookes. A semantics for concurrent separation logic. *tcs*, 375(1–3): 227–270, 2007a. URL https://doi.org/10.1016/j.tcs.2006.12.034.

Stephen Brookes. A semantics for concurrent separation logic. *Theoretical Computer Science*, 375(1):227–270, 2007b. ISSN 0304-3975. URL https://doi.org/10.1016/j.tcs.2006.12.034. Festschrift for John C. Reynolds's 70th birthday.

James Brotherston and Cristiano Calcagno. Classical BI: A Logic for Reasoning about Dualising Resources. In *Proceedings of the 36th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '09, page 328–339, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605583792. URL https://doi.org/10.1145/1480881.1480923.

Cristiano Calcagno, Peter W. O'Hearn, and Hongseok Yang. Local Action and Abstract Separation Logic. In *22nd Annual IEEE Symposium on Logic in Computer Science (LICS 2007)*, pages 366–378, Wroclaw, Poland, 2007. IEEE. ISBN 978-0-7695-2908-0. URL https://doi.org/10.1109/LICS.2007.30.

Qinxiang Cao, Santiago Cuellar, and Andrew W. Appel. Bringing order to the separation logic jungle. In *Asian Symposium on Programming Languages and Systems (APLAS)*, pages 190–211, Suzhou, China, 2017. Springer.

Aleksandar Chakarov and Sriram Sankaranarayanan. Probabilistic program analysis with martingales. In *International Conference on Computer Aided Verification (CAV)*, pages 511–526, Saint Petersburg, Russia, 2013. Springer. URL https://doi.org/10.1007/978-3-642-39799-8_34.

Yu-Fang Chen, Chih-Duo Hong, Bow-Yaw Wang, and Lijun Zhang. Counterexample-guided polynomial loop invariant generation by Lagrange interpolation. In *CAV*, 2015. doi: 10.1007/978-3-319-21690-4_44.

David Maxwell Chickering. Learning bayesian networks is np-complete. In *Learning from data: Artificial intelligence and statistics V*, pages 121–130. Springer, 1996.

David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554, 2002.

Kenta Cho and Bart Jacobs. Disintegration and Bayesian inversion via string diagrams. *Mathematical Structures in Computer Science*, 29(7):938–971, 2019. URL https://doi.org/10.1017/S0960129518000488.

Ken Christensen, Allen Roginsky, and Miguel Jimeno. A new analysis of the false positive rate of a bloom filter. *Information Processing Letters*, 110(21):944–949, 2010. ISSN 0020-0190. URL https://doi.org/10.1016/j.ipl.2010.07.024.

Ugo Dal Lago, Davide Davoli, and Bruce M Kapron. On Separation Logic, Computational Independence, and Pseudorandomness. In *2024 IEEE 37th Computer Security Foundations Symposium (CSF)*, pages 651–666. IEEE Computer Society, 2024.

A Philip Dawid. Conditional independence in statistical theory. *Journal of the Royal Statistical Society: Series B (Methodological)*, 41(1):1–15, 1979.

A. Philip Dawid. Separoids: A mathematical framework for conditional independence and irrelevance. *Annals of Mathematics and Artificial Intelligence*, 32 (1-4):335–372, 2001.

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

Edsger W Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs. *Communications of the ACM*, 18(8):453–457, 1975.

Bolin Ding and Arnd Christian König. Fast set intersection in memory. *Proceedings of the VLDB Endowment*, 4(4):255–266, 2011. URL https://doi.org/10.14778/1938545.1938550.

Simon Docherty. *Bunched Logics: A Uniform Approach*. PhD thesis, UCL (University College London), 2019.

Devdatt P. Dubhashi and Desh Ranjan. Balls and bins: A study in negative dependence. *Random Structures and Algorithms*, 13(2):99–124, 1998.

Arnaud Durand, Miika Hannula, Juha Kontinen, Arne Meier, and Jonni Virtema. Probabilistic Team Semantics. In *International Symposium on Foundations of Information and Knowledge Systems (FoIKS), Budapest, Hungary*, volume 10833 of *Lecture Notes in Computer Science*, pages 186–206. Springer, 2018. URL https://doi.org/10.1007/978-3-319-90050-6_11.

Evgeniĭ Borisovich Dynkin. *Theory of Markov processes*. Courier Corporation, 2012.

Mnacho Echenim, Radu Iosif, and Nicolas Peltier. The bernays-schönfinkel-ramsey class of separation logic with uninterpreted predicates. *ACM Trans. Comput. Logic*, 21(3), March 2020. ISSN 1529-3785. doi: 10.1145/3380809. URL https://doi.org/10.1145/3380809.

Facebook. Infer static analyzer. URL https://fbinfer.com/.

Ronald Fagin and Moshe Y. Vardi. The Theory of Data Dependencies - An Overview. In *International Colloquium on Automata, Languages and*

Programming (ICALP), pages 1–22, 1984. URL https://doi.org/10.1007/3-540-13345-3_1.

Yijun Feng, Lijun Zhang, David N Jansen, Naijun Zhan, and Bican Xia. Finding polynomial loop invariants for probabilistic programs. In *ATVA*, 2017.

Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32 (2):374–382, 1985.

Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine learning*, 29(2):131–163, 1997.

Bert E Fristedt and Lawrence F Gray. *A modern approach to probability theory*. Springer Science & Business Media, 2013.

Tobias Fritz. A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics. *Advances in Mathematics*, 370:107–239, 2020. URL https://doi.org/10.1016/j.aim.2020.107239.

Dan Geiger and Judea Pearl. Logical and Algorithmic Properties of Conditional Independence and Graphical Models. *The Annals of Statistics*, 21(4):2001–2021, 1993.

Michele Giry. A categorical approach to probability theory. *Categorical aspects of topology and analysis*, pages 68–85, 1982.

Robert Goldblatt. Varieties of complex algebras. *Annals of Pure and Applied Logic*, 44(3):173–242, 1989. URL https://doi.org/10.1016/0168-0072(89)90032-8.

Oded Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, 78(110):1–108, 1998.

Kiran Gopinathan and Ilya Sergey. Certifying certainty and uncertainty in approximate membership query structures. In *International Conference on Computer Aided Verification (CAV)*, volume 12225 of *Lecture Notes in Computer Science*, pages 279–303, Los Angeles, California, 2020. springer. URL https://doi.org/10.1007/978-3-030-53291-8_16.

Andrew D Gordon, Thomas A Henzinger, Aditya V Nori, and Sriram K Rajamani. Probabilistic programming. In *Future of Software Engineering Proceedings*, pages 167–181. 2014.

Simon Oddershede Gregersen, Alejandro Aguirre, Philipp G Haselwarter, Joseph Tassarotti, and Lars Birkedal. Asynchronous Probabilistic Couplings in Higher-Order Separation Logic. *Proceedings of the ACM on Programming Languages*, 8(POPL):753–784, 2024.

Friedrich Gretz, Joost-Pieter Katoen, and Annabelle McIver. Prinsys - on a quest for probabilistic loop invariants. In *QEST*, 2013. doi: 10.1007/978-3-642-40196-1_17.

Tao Gu, Jialu Bao, Justin Hsu, Alexandra Silva, and Fabio Zanasi. A categorical approach to dibi models. In *9th International Conference on Formal Structures for Computation and Deduction, FSCD 2024, July 10-13, 2024, Tallinn, Estonia*, volume 299 of *LIPIcs*, pages 17:1–17:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. URL https://doi.org/10.4230/LIPIcs.FSCD.2024.17.

Miika Hannula, Juha Kontinen, Jan Van den Bussche, and Jonni Virtema. Descriptive complexity of real computation and probabilistic independence logic. In *"IEEE Symposium on Logic in Computer Science (LICS)*, pages 550–563, 2020.

Jaakko Hintikka and Gabriel Sandu. Informational Independence as a Semantical Phenomenon. In *Logic, Methodology and Philosophy of Science VIII*, volume 126 of *Studies in Logic and the Foundations of Mathematics*, pages 571–589. Elsevier, 1989. URL https://doi.org/10.1016/S0049-237X(08)70066-1.

Shing Hin Ho, Nicolas Wu, and Azalea Raad. Bayesian separation logic. *arXiv preprint arXiv:2507.15530*, 2025.

Charles Antony Richard Hoare. Algorithm 64: quicksort. *Communications of the ACM*, 4(7):321, 1961.

Tony Hoare, Bernhard Möller, Georg Struth, and Ian Wehrman. Concurrent Kleene algebra and its foundations. *The Journal of Logic and Algebraic Programming*, 80(6):266–296, 2011.

Steven J Holtzen. *Exploiting Program Structure for Scaling Probabilistic Programming*. University of California, Los Angeles, 2021.

Justin Hsu. *Probabilistic Couplings for Probabilistic Reasoning*. PhD thesis, University of Pennsylvania, 2017.

Janez Ignacij Jereb and Alex Simpson. Safety, relative tightness and the probabilistic frame rule. *arXiv e-prints*, pages arXiv–2506, 2025.

Bart Jacobs and Fabio Zanasi. A Formal Semantics of Influence in Bayesian Reasoning. In *International Symposium on Mathematical Foundations of Computer Science (MFCS), Aalborg, Denmark*, volume 83 of *Leibniz International Proceedings in Informatic*, pages 21:1–21:14. dagstuhl, 2017. URL https://doi.org/10.4230/LIPIcs.MFCS.2017.21.

Kumar Joag-Dev and Frank Proschan. Negative association of random variables

with applications. *The Annals of Statistics*, 11(1):286–295, 1983. URL https://doi.org/10.1214/aos/1176346079.

Ralf Jung, David Swasey, Filip Sieczkowski, Kasper Svendsen, Aaron Turon, Lars Birkedal, and Derek Dreyer. Iris: Monoids and invariants as an orthogonal basis for concurrent reasoning. *ACM SIGPLAN Notices*, 50(1):637–650, 2015.

Ralf Jung, Robbert Krebbers, Jacques-Henri Jourdan, Ales Bizjak, Lars Birkedal, and Derek Dreyer. Iris from the ground up: A modular foundation for higher-order concurrent separation logic. *Journal of Functional Programming*, 28:e20, 2018. URL https://doi.org/10.1017/S0956796818000151.

Ralf Jung, Rodolphe Lepigre, Gaurav Parthasarathy, Marianna Rapoport, Amin Timany, Derek Dreyer, and Bart Jacobs. The future is ours: prophecy variables in separation logic. *Proc. ACM Program. Lang.*, 4(POPL), December 2019. URL https://doi.org/10.1145/3371113.

Benjamin Lucien Kaminski. *Advanced Weakest Precondition Calculi for Probabilistic Programs*. PhD thesis, RWTH Aachen University, 2019.

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

Neville Kenneth Kitson, Anthony C Constantinou, Zhigao Guo, Yang Liu, and Kiattikun Chobtham. A survey of bayesian network structure learning. *Artificial Intelligence Review*, 56(8):8721–8814, 2023.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Dexter Kozen. Semantics of probabilistic programs. *Journal of Computer and System Sciences*, 22(3):328–350, 1981. URL https://doi.org/10.1016/0022-0000(81)90036-2.

Dexter Kozen. A probabilistic PDL. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, pages 291–297, 1983.

Robbert Krebbers, Jacques-Henri Jourdan, Ralf Jung, Joseph Tassarotti, Jan-Oliver Kaiser, Amin Timany, Arthur Charguéraud, and Derek Dreyer. MoSeL: A general, extensible modal framework for interactive proofs in separation logic. *Proceedings of the ACM on Programming Languages*, 2(ICFP):77:1–77:30, 2018. URL https://doi.org/10.1145/3236772.

Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *Found. Trends Mach. Learn.*, 5(2-3):123–286, 2012. URL https://doi.org/10.1561/2200000044.

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

John M. Li, Amal Ahmed, and Steven Holtzen. Lilac: A modal separation logic for conditional probability. *Proc. ACM Program. Lang.*, 7(PLDI), June 2023a. URL https://doi.org/10.1145/3591226.

John M. Li, Amal Ahmed, and Steven Holtzen. Lilac: A modal separation logic for conditional probability, 2023b. URL https://arxiv.org/abs/2304.01339.

John M. Li, Jon Aytac, Philip Johnson-Freyd, Amal Ahmed, and Steven Holtzen. A nominal approach to probabilistic separation logic. In *IEEE Symposium on Logic in Computer Science (LICS)*, pages 55:1–55:14. ACM, 2024.

Nancy A Lynch. *Distributed algorithms*. Elsevier, 1996.

Michael Mitzenmacher and Eli Upfal. *Probability and Computing - Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

Eugenio Moggi. Notions of computation and monads. *Information and Computation*, 93(1):55–92, 1991. URL https://doi.org/10.1016/0890-5401(91)90052-4.

Carroll Morgan, Annabelle McIver, and Karen Seidel. Probabilistic predicate transformers. *ACM Transactions on Programming Languages and Systems*, 1996. URL https://doi.org/10.1145/229542.229547.

James K Mullin. A second look at bloom filters. *Communications of the ACM*, 26 (8):570–571, 1983.

Wolfgang Mulzer. Five Proofs of Chernoff's Bound with Applications, May 2019. URL https://doi.org/10.48550/arXiv.1801.03365.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Fritz Obermeyer, Eli Bingham, Martin Jankowiak, Neeraj Pradhan, Justin Chiu, Alexander Rush, and Noah Goodman. Tensor variable elimination for plated factor graphs. In *International Conference on Machine Learning*, pages 4871–4880. PMLR, 2019.

Peter W. O'Hearn and David J. Pym. The logic of bunched implications. *bsl*, pages 215–244, 1999.

Prakash Panangaden. *Labelled Markov Processes*. Imperial College Press, 2009. URL https://doi.org/10.1142/p595.

Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 2014.

Judea Pearl and Azaria Paz. *Graphoids: A Graph-Based Logic for Reasoning about Relevance Relations*. University of California (Los Angeles). Computer Science Department, ˜, 1985.

Judea Pearl, Dan Geiger, and Thomas Verma. Conditional independence and its representations. *Kybernetika*, 25(7):33–44, 1989.

Ruzica Piskac, Thomas Wies, and Damien Zufferey. Automating separation logic with trees and data. In *Computer Aided Verification: 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings 26*, pages 711–728. Springer, 2014.

Andrew M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2013. URL https://doi.org/10.1017/CBO9781139084673.

Konstantinos Psounis and Balaji Prabhakar. A randomized web-cache replacement scheme. In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, volume 3, pages 1407–1415 vol.3, 2001. URL https://doi.org/110.1109/INFCOM.2001.916636.

Michael O Rabin. Probabilistic algorithm for testing primality. *Journal of number theory*, 12(1):128–138, 1980.

Prabhakar Raghavan and Clark D Tompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7 (4):365–374, 1987.

John C Reynolds. Separation logic: A logic for shared mutable data structures. In *Proceedings 17th Annual IEEE Symposium on Logic in Computer Science*, pages 55–74. IEEE, 2002.

John A Rice. *Mathematical statistics and data analysis*, volume 371. Thomson/Brooks/Cole Belmont, CA, 2007.

Marc Romaní. A short proof of Hoeffding's lemma, May 2021.

Jeffrey S. Rosenthal. *A First Look at Rigorous Probability Theory*. World Scientific Publishing Company, 2006.

Alex Simpson. Category-theoretic Structure for Independence and Conditional Independence. In *Conference on the Mathematical Foundations of Programming Semantics (MFPS)*, pages 281–297, 2018. URL https://doi.org/10.1016/j.entcs.2018.03.028.

Alex Simpson. Equivalence and conditional independence in atomic sheaf logic. In *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 1–14, 2024.

Aravind Srinivasan. Distributions on level-sets with applications to approximation algorithms. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 588–597, Las Vegas, Nevada, 2001. IEEE. URL https://doi.org/10.1109/SFCS.2001.959935.

Sam Staton. Probabilistic programs as measures. *Foundations of Probabilistic Programming*, page 43, 2020.

Robert S Strichartz. *The way of analysis*. Jones & Bartlett Learning, 2000.

Subhash Suri. Caching, January 2020. URL https://sites.cs.ucsb.edu/~suri/ccs130a/Caching.pdf.

Joseph Tassarotti and Robert Harper. A separation logic for concurrent randomized programs. *Proceedings of the ACM on Programming Languages*, 3(POPL): 64:1–64:30, 2019. URL https://doi.org/10.1145/3290377.

Jouko Väänänen. *Dependence Logic: A New Approach to Independence Friendly Logic*. London Mathematical Society Student Texts. Cambridge University Press, 2007. URL https://doi.org/10.1017/CBO9780511611193.

Viktor Vafeiadis and Matthew Parkinson. A marriage of rely/guarantee and separation logic. In *CONCUR 2007, Lisbon, Portugal, September 3-8, 2007. Proceedings 18*, pages 256–271. Springer, 2007.

John von Neumann. Various techniques used in connection with random digits. *Journal of Research of the National Bureau of Standards, Applied Math Series*, pages 36–38, 1951.

Jinyi Wang, Yican Sun, Hongfei Fu, Krishnendu Chatterjee, and Amir Kafshdar Goharshady. Quantitative analysis of assertion violations in probabilistic programs. In *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pages 1171–1186, Virtual, 2021. acmpress. URL https://doi.org/10.1145/3453483.3454102.

Yuxin Wang, Zeyu Ding, Guanhong Wang, Daniel Kifer, and Danfeng Zhang. Proving differential privacy with shadow execution. *Proceedings of the ACM on Programming Languages*, (POPL, 19):655–669, 2019.

Pengbo Yan, Toby Murray, Olga Ohrimenko, Van-Thuan Pham, and Robert Sison. Combining classical and probabilistic independence reasoning to verify the security of oblivious algorithms. In *International Symposium on Formal Methods*, pages 188–205. Springer, 2024.

Danfeng Zhang and Daniel Kifer. LightDP: Towards automating differential privacy proofs. In *ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 888–901, 2017.

APPENDIX A

BUNCHED LOGIC AND PROBABILISTIC SEPARATION LOGIC

A.1 Proofs related to Bunched Logic

Theorem 2.2.13. *$\mathcal{X}_{\mathbb{D}} = (X_{\mathbb{D}}, \sqsubseteq_{\mathbb{D}}, \otimes_{\mathbb{D}}, E_{\mathbb{D}})$ is a BI frame.*

Proof. We show that the defined structure satisfies all the frame conditions.

Commutativity For any $\mu_1 \in \mathcal{D}(\mathbf{Mem}[S]), \mu_2 \in \mathcal{D}(\mathbf{Mem}[T])$,

$$\begin{aligned}
&\mu \in \mu_1 \otimes_{\mathbb{D}} \mu_2 \\
\Leftrightarrow &\mu(m) = \mu_1(\pi_S m) \cdot \mu_2(\pi_T m) \text{ for any } m \in \mathbf{Mem}[S \cup T] \\
\Leftrightarrow &\mu \in \mu_2 \otimes_{\mathbb{D}} \mu_1.
\end{aligned}$$

Associativity Given $\mu_1 \in \mathcal{D}(\mathbf{Mem}[S]), \mu_2 \in \mathcal{D}(\mathbf{Mem}[T]), \mu_3 \in \mathcal{D}(\mathbf{Mem}[R])$, for any $\mu_0 \in \mu_1 \otimes_{\mathbb{D}} \mu_2$ and $\mu \in \mu_0 \otimes_{\mathbb{D}} \mu_3$,

$$\begin{aligned}
&\mu \in \mu_0 \otimes_{\mathbb{D}} \mu_3 \\
\Leftrightarrow &\mu(m) = \mu_0(\pi_{S\cup T} m) \cdot \mu_3(\pi_R m) \text{ for any } m \in \mathbf{Mem}[S \cup T \cup R] \\
\Leftrightarrow &\mu(m) = (\mu_1(\pi_S m) \cdot \mu_2(\pi_T m)) \cdot \mu_3(\pi_R m) \text{ for any } m \in \mathbf{Mem}[S \cup T \cup R] \\
\Leftrightarrow &\mu(m) = \mu_1(\pi_S m) \cdot (\mu_2(\pi_T m) \cdot \mu_3(\pi_R m)) \text{ for any } m \in \mathbf{Mem}[S \cup T \cup R]
\end{aligned}$$

Define $\mu' = \pi_{T \cup R}\mu$. Then for any $m \in \mathbf{Mem}[T \cup R]$,

$$\begin{aligned}
\mu'(m) &= \sum_{m' \in \mathbf{Mem}[S]} \mu(m' \bowtie m) \\
&= \sum_{m' \in \mathbf{Mem}[S]} \mu_1(\pi_S m' \bowtie m) \cdot (\mu_2(\pi_T m' \bowtie m) \cdot \mu_3(\pi_R m' \bowtie m)) \\
&= \left(\sum_{m' \in \mathbf{Mem}[S]} \mu_1(m') \right) \cdot (\mu_2(\pi_T m) \cdot \mu_3(\pi_R m)) \\
&= \mu_2(\pi_T m) \cdot \mu_3(\pi_R m)
\end{aligned}$$

Thus, $\mu' \in \mu_2 \otimes_{\mathbb{D}} \mu_3$, and furthermore,

$$\begin{aligned}
&\mu(m) = \mu_1(\pi_S m) \cdot (\mu_2(\pi_T m) \cdot \mu_3(\pi_R m)) \text{ for any } m \in \mathbf{Mem}[S \cup T \cup R] \\
\Rightarrow &\mu(m) = \mu_1(\pi_S m) \cdot \mu'(\pi_{T \cup R} m) \text{ for any } m \in \mathbf{Mem}[S \cup T \cup R] \\
\Rightarrow &\mu \in \mu_1 \otimes_{\mathbb{D}} \mu'
\end{aligned}$$

Unit Existence Given any $\mu \in \mathcal{D}(\mathbf{Mem}[S])$, for any $m \in \mathbf{Mem}[S]$,

$$(\mu \otimes_{\mathbb{D}} \langle\rangle)(m) = \mu(\pi_S m) \cdot \langle\rangle(\pi_\emptyset m) = \mu(m).$$

Thus, $\mu \in \mu \otimes_{\mathbb{D}} \langle\rangle$.

Unit Closure $E_{\mathbb{D}}$ is closed under the pre-order as $E_{\mathbb{D}} = X_{\mathbb{D}}$.

Unit Coherence For any $\mu_x \in \mathcal{D}(\mathbf{Mem}[S]), \mu_e \in \mathcal{D}(\mathbf{Mem}[T])$, if $\mu_y \in \mu_x \otimes_{\mathbb{D}} \mu_e$, then for any $m \in \mathbf{Mem}[S \cup T]$, $\mu_y(m) = \mu_x(\pi_S m) \cdot \mu_e(\pi_T m)$. Thus, for any $m_S \in \mathbf{Mem}[S]$, $m_T \in \mathbf{Mem}[T]$, $\mu_y(m_S \bowtie m_T) = \mu_x(m_S) \cdot \mu_e(m_T)$, which implies that

$$\sum_{m'_T \in \mathbf{Mem}[T]} \mu_y(m_S \bowtie m'_T) = \sum_{m'_T \in \mathbf{Mem}[T]} \mu_x(m_S) \cdot \mu_e(m'_T) = \mu_x(m_S).$$

Therefore, $\mu_x \sqsubseteq_{\mathbb{D}} \mu_y$.

Down-Closed If $\mu_z \in \mu_x \otimes_{\mathbb{D}} \mu_y$, and $\mu'_x \sqsubseteq_{\mathbb{D}} \mu_x$, $\mu'_y \sqsubseteq_{\mathbb{D}} \mu_y$, then define $X = \mathbf{dom}(\mu_x), Y = \mathbf{dom}(\mu_y), X' = \mathbf{dom}(\mu'_x), Y' = \mathbf{dom}(\mu'_y)$, and define $\mu = \pi_{X' \cup Y'} \mu_z$. The fact that $\mu_z \in \mu_x \otimes_{\mathbb{D}} \mu_y$ implies that for any $m \in \mathbf{Mem}[X \cup Y]$,

$$\mu_z(m) = \mu_x(\pi_X m) \cdot \mu_y(\pi_Y m);$$

Thus,

$$\begin{aligned}
\mu(m) = (\pi_{X' \cup Y'} \mu_z)(m) &= \sum_{m' \in \mathbf{Mem}[X \cup Y \setminus (X' \cup Y')]} \mu_z(m' \bowtie m) \\
&= \sum_{m' \in \mathbf{Mem}[X \cup Y \setminus (X' \cup Y')]} \mu_x(\pi_X m' \bowtie m) \cdot \mu_y(\pi_Y m' \bowtie m) \\
&= \sum_{m_1 \in \mathbf{Mem}[X \setminus X']} \sum_{m_2 \in \mathbf{Mem}[Y \setminus Y']} \mu_x(\pi_X m_1 \bowtie m_2 \bowtie m) \cdot \mu_y(\pi_Y m_1 \bowtie m_2 \bowtie m) \\
&= \left(\sum_{m_1 \in \mathbf{Mem}[X \setminus X']} \mu_x(\pi_X m_1 \bowtie m) \right) \cdot \left(\sum_{m_2 \in \mathbf{Mem}[Y \setminus Y']} \mu_y(\pi_Y m_2 \bowtie m) \right) \\
&= \pi_{X'} \mu_x(m) \cdot \pi_{Y'} \mu_y(m) \\
&= \mu'_x(m) \cdot \mu'_y(m)
\end{aligned}$$

Hence, $\mu \in \mu'_x \otimes_{\mathbb{D}} \mu'_y$, and by definition, $\mu \sqsubseteq_{\mathbb{D}} \mu_z$.

□

Lemma 2.3.2. *For any distribution $\mu \in \mathcal{X}_{\mathbb{D}}$, for a set of variables $\{X_i\}_{i \in S}$, $\mu \models *_{i \in S} \mathsf{Own}(X_i)$ iff variables $\{X_i\}_{i \in S}$ are distinct and mutually independent.*

Proof. For the forward definition: $\mu \models *_{i \in S} \mathsf{Own}(X_i)$ implies $\mu \models *_{i \in T} \mathsf{Own}(X_i)$ for any subset $T \subseteq S$. by inductively unfolding the satisfaction definition and applying eq. (Down-Closed), there exists a set of distributions $\{\mu_i\}_{i \in T}$ such that

- $\mu_i \models \mathsf{Own}(X_i)$ for any $i \in T$.

- $\mathbf{dom}(\mu_i)$ are all disjoint
- Let $T' = \cup_{i \in T}\{X_i\}$. For any $m \in \mathbf{Mem}[T]$, $\pi_T \mu(m) = \prod_{i \in T} \mu_i(\pi_{\mathbf{dom}(\mu_i)} m)$.

Thus, the first condition implies $X_i \in \mathbf{dom}(\mu_i)$ for each i, and thus $\mu_i(X_i = v)$ can be evaluated for any $i \in T$ and $v \in \mathbf{Val}$. Combining this with the third condition, we have that: for any set of $\{v_i \in \mathbf{Val}\}_{i \in T}$

$$\begin{aligned}\pi_R \mu(m) &= \prod_{i \in T} \mu_i(X_i = v_i) \\ &= \prod_{i \in T} \mu(X_i = v_i).\end{aligned}$$

Also, the second condition combined with the third one imply that all X_i are distinct.

For the backwards direction. We define $\mu_i = \pi_{X_i} \mu$ for each $i \in T$. Then clearly, each μ_i satisfies $\mathsf{Own}(X_i)$. For convenience, relabel the variables in T as $T_1, \ldots, T_m$, and denote $\cup_{i=1}^{k}\{T_i\}$ as $T[:k]$. Furthermore, we can prove by induction that $\pi_{T[:k]}\mu \models \ast_{i=1}^{k} \mathsf{Own}(X_i)$ for $1 \leq k \leq m$.

Base: $\pi_{X_i}\mu \models \mathsf{Own}(X_i)$.

Inductive: X_i being mutually independent implies that for any set of values $\{v_i \in \mathbf{Val}\}_{1 \leq i \leq m}$,

$$\mu\left(\bigwedge_{i \leq k} T_i = v_i\right) = \prod_{i \leq k} \mu(T_i = v_i) = \mu\left(\bigwedge_{1 \leq i < k} T_i = v_i\right) \cdot \mu(T_k = v_k)$$

Thus, for any $m \in \mathbf{Mem}[\cup_{1 \leq i \leq k}\{T_i\}]$,

$$\mu(m) = \pi_{T[:k-1]}\mu\big(\pi_{T[:k-1]} m\big) \cdot \pi_{T_k}\mu(\pi_{T_k} m)$$

And therefore, $\pi_{T[:k]}\mu \in \big(\pi_{T[:k-1]}\mu\big) \circ \pi_{T_k}\mu$. By inductive hypothesis, $\big(\pi_{T[:k]}\mu\big) \models \ast_{i=1}^{k-1} \mathsf{Own}(X_i)$, and by satisfaction, we have $\pi_{T[:k]}\mu \models \ast_{i=1}^{k} \mathsf{Own}(X_i)$.

□

A.2 Proofs related to Probabilistic Separation Logic

Lemma 2.3.7 (Restriction). *Let $\mu \in \mathcal{D}(\mathbf{Mem}[S])$ and let φ be a BI formula. Then:*

$$\mu \models \varphi \Leftrightarrow (\sigma, \pi_{FV(\varphi)}(\mu)) \models \varphi.$$

Proof. The reverse direction follows by the persistence. The forward direction follows by induction on φ.

- $\varphi \equiv \top, \bot$, and atomic propositions p. Trivial.
- $\varphi \equiv \varphi_1 \wedge \varphi_2$. By induction, we have

  $$\pi_{\mathrm{FV}(\varphi_1)}\mu \models \varphi_1 \quad \text{and} \quad \pi_{\mathrm{FV}(\varphi_2)}\mu \models \varphi_2.$$

  By persistence, we have

  $$\pi_{\mathrm{FV}(\varphi_1 \wedge \varphi_2)}\mu \models \varphi_1 \quad \text{and} \quad \pi_{\mathrm{FV}(\varphi_1 \wedge \varphi_2)}\mu \models \varphi_2$$

  so $\pi_{\mathrm{FV}(\varphi_1 \wedge \varphi_2)}\mu \models \varphi_1 \wedge \varphi_2$.
- $\varphi \equiv \varphi_1 \vee \varphi_2$. By induction, we have $\pi_{\mathrm{FV}(\varphi_i)}\mu \models \varphi_i$ for $i = 1$ or $i = 2$. By Kripke monotonicity, we have $\pi_{\mathrm{FV}(\varphi_1,\varphi_2)}\mu \models \varphi_i$ so $\pi_{\mathrm{FV}(\varphi_1 \wedge \varphi_2)}\mu \models \varphi_1 \vee \varphi_2$.
- $\varphi \equiv \varphi_1 \to \varphi_2$. Take any $\mu' \sqsupseteq \pi_{\mathrm{FV}(\varphi_1 \to \varphi_2)}\mu$ such that $\mu' \models \varphi_1$. By inductive hypothesis, $\pi_{\mathrm{FV}(\varphi_1)}\mu' \models \varphi_1$ Because $\mu' \sqsupseteq \pi_{\mathrm{FV}(\varphi_1 \to \varphi_2)}\mu$, we have $\pi_{\mathrm{FV}(\varphi_1 \to \varphi_2)}\mu' = \pi_{\mathrm{FV}(\varphi_1 \to \varphi_2)}\mu$, and thus $\pi_{\mathrm{FV}(\varphi_1)}\mu' = \pi_{\mathrm{FV}(\varphi_1)}\mu$. There exists a distribution μ'' such that $\mathbf{dom}(\mu'') = \mathbf{dom}(\mu) \cup \mathbf{dom}(\pi_{\mathrm{FV}(\varphi_1)}\mu')$, and $\pi_{\mathbf{dom}(\mu)}(\mu'') = \mu$ and $\pi_{\mathbf{dom}(\pi_{\varphi_1}\mu')}(\mu'') = \pi_{\varphi_1}\mu'$. In particular, $\mu'' \sqsupseteq \mu$. By

persistence, we have $\mu'' \models \varphi_1$ and by validity, we have $\mu'' \models \varphi_2$. By induction, $\pi_{\mathrm{FV}(\varphi_2)}(\mu'') \models \varphi_2$. Since $\pi_{\mathrm{FV}(\varphi_2)}(\mu'') \sqsubseteq \mu'$, persistence gives $\mu' \models \varphi_2$. So, $\pi_{\mathrm{FV}(\varphi_1 \to \varphi_2)}\mu \models \varphi_1 \to \varphi_2$ as desired.

- $\varphi \equiv \varphi_1 * \varphi_2$. There exists μ_1 and μ_2 with $\mu_1 \circ \mu_2 \sqsubseteq \mu$ and $\mu_1 \models \varphi_1$ and $\mu_2 \models \varphi_2$. By induction, we have $\pi_{\mathrm{FV}(\varphi_1)}\mu_1 \models \varphi_1$ and $\pi_{\mathrm{FV}(\varphi_2)}\mu_2 \models \varphi_2$. By persistence, we have $\pi_{\mathrm{FV}(\varphi_1 * \varphi_2)}\mu_1 \models \varphi_1$ and $\pi_{\mathrm{FV}(\varphi_1 * \varphi_2)}\mu_2 \models \varphi_2$. Now, since $\mu_1 \circ \mu_2$ is defined, $\pi_{\mathrm{FV}(\varphi_1 * \varphi_2)}\mu_1 \circ \pi_{\mathrm{FV}(\varphi_1 * \varphi_2)}\mu_2$ is defined as well and $\pi_{\mathrm{FV}(\varphi_1 * \varphi_2)}\mu_1 \circ \pi_{\mathrm{FV}(\varphi_1 * \varphi_2)}\mu_2 \sqsubseteq \pi_{\mathrm{FV}(\varphi_1 * \varphi_2)}\mu$. So, $\pi_{\mathrm{FV}(\varphi_1 * \varphi_2)}\mu \models \varphi_1 * \varphi_2$ as desired.

- $\varphi \equiv \varphi_1 \mathrel{-\!\!*} \varphi_2$. Take any μ' such that $\mu' \circ \pi_{\mathrm{FV}(\varphi_1 \mathrel{-\!\!*} \varphi_2)}\mu \downarrow$ and $\mu' \models \varphi_1$. If $\mu' \circ \mu \downarrow$, then $\mu' \circ \mu \models \varphi_2$ and by induction, $(\pi_{\mathrm{FV}(\varphi_1 \mathrel{-\!\!*} \varphi_2)}\mu') \circ (\pi_{\mathrm{FV}(\varphi_1 \mathrel{-\!\!*} \varphi_2)}\mu) \models \varphi_2$. persistence gives $\mu' \circ \pi_{\mathrm{FV}(\varphi_1 \mathrel{-\!\!*} \varphi_2)}\mu \models \varphi_2$.

  Otherwise, suppose that $\mu' \circ \mu$ is not defined. Since $\mu' \circ \pi_{\mathrm{FV}(\varphi_1 \mathrel{-\!\!*} \varphi_2)}\mu \downarrow$, it must be the case that $\emptyset \neq \mathbf{dom}(\mu') \cap \mathbf{dom}(\mu) \subseteq \mathbf{Var} \setminus \mathrm{FV}(\varphi_1 \mathrel{-\!\!*} \varphi_2)$; thus, $(\pi_{\mathrm{FV}(\varphi_1)}(\mu') \circ \mu) \downarrow$. By induction, $\pi_{\mathrm{FV}(\varphi_1)}(\mu') \models \varphi_1$ and so $\pi_{\mathrm{FV}(\varphi_1)}(\mu') \circ \mu \models \varphi_2$. By induction again, $(\pi_{\mathrm{FV}(\varphi_1) \cap \mathrm{FV}(\varphi_2)}\mu') \circ (\pi_{\mathrm{FV}(\varphi_2)}\mu) \models \varphi_2$. By persistence and the fact that the extension is defined, we have $\mu' \circ (\pi_{\mathrm{FV}(\varphi_1 * \varphi_2)}\mu) \models \varphi_2$. So, $(\pi_{\mathrm{FV}(\varphi_1 \mathrel{-\!\!*} \varphi_2)}\mu) \models \varphi_1 \mathrel{-\!\!*} \varphi_2$ as desired.

□

Lemma A.2.1 (Soundness for RV, WV, MV [Barthe et al., 2019])**.** *Let* $\mu' = [\![c]\!]\mu$, *and let* $R = RV(c), W = WV(c), S = \mathbf{Var} \setminus MV(c)$. *Then:*

1. *Variables outside of* $MV(c)$ *are not modified:* $\pi_C(\mu') = \pi_C(\mu)$.
2. *The sets* R *and* W *are disjoint.*

3. *There exists* $f : \mathbf{Mem}[R] \to \mathcal{D}(\mathbf{Mem}[MV(c)])$ *with* $\mu' = \mathsf{bind}(\mu, m \mapsto f(\pi_R(m)) \otimes \mathsf{unit}(\pi_S(m)))$.

Theorem 2.3.6 (Soundness)**.** *If* $\vdash \{\varphi\}\ c\ \{\psi\}$ *is derivable, then* $\models \{\varphi\}\ c\ \{\psi\}$.

Proof. By induction on the derivation. Let μ satisfy the pre-condition of the conclusion.

SKIP Trivial.

SEQ By induction hypothesis.

DASSN By induction on the syntax of φ.

RASSN The output distribution $[\![x \leftarrow e]\!]\mu = \mathsf{bind}(\mu, m \mapsto \mathsf{unit}(m[x \mapsto [\![e]\!](m)]))$. Because $x \notin \mathrm{FV}(e)$, for any m,

$$[\![x]\!](m[x \mapsto [\![e]\!](m)]) = [\![e]\!](m[x \mapsto [\![e]\!](m)]).$$

Thus, $[\![x \leftarrow e]\!]\mu \models [x = e]$.

SAMP The output distribution $[\![x \xleftarrow{\$} d]\!](\mu) = \mathsf{bind}(\mu, m \mapsto \mathsf{bind}(d, v \mapsto \mathsf{unit}(m[x \mapsto v])))$. Thus,

$$\begin{aligned}
[\![x]\!]([\![x \xleftarrow{\$} d]\!](\mu)) &= \mathsf{bind}([\![x \xleftarrow{\$} d]\!](\mu), m \mapsto [\![x]\!](m)) \\
&= \mathsf{bind}(\mathsf{bind}(\mu, m \mapsto \mathsf{bind}(d, v \mapsto \mathsf{unit}(m[x \mapsto v]))), m \mapsto [\![x]\!](m)) \\
&= \mathsf{bind}(\mu, m \mapsto \mathsf{bind}(d, v \mapsto \mathsf{unit}(v))) \\
&= \mathsf{bind}(\mu, m \mapsto d) \\
&= d.
\end{aligned}$$

Therefore, $[\![x \xleftarrow{\$} d]\!](\mu) \models x \overset{\$}{\sim} d$.

COND Since $\mu \models \varphi$ and $\models \varphi \to \mathbf{Detm}\langle b \rangle$, we have $\mu \models \mathbf{Detm}\langle b \rangle$. Thus, either $\mu \models [b = tt]$ or $\mu \models [b = ff]$. Note that exactly one case holds. If $\mu \models [b = tt]$ holds, then $\mu \models \varphi \wedge [b = tt]$ and thus $[\![\mathbf{if}\ b\ \mathbf{then}\ c\ \mathbf{else}\ c']\!](\mu) = [\![c]\!](\mu)$, we can conclude by induction. The case $\mu \models [b = ff]$ is similar.

RCOND Because $\mu \models \varphi * \mathsf{Own}(b)$, there exist μ_1, μ_2 such that $\mu_1 \circ \mu_2 \sqsubseteq \mu$, and $\mu_1 \models \varphi$ and $\mu_2 \models \mathsf{Own}(b)$. Let ρ be the probability $[\![b]\!](\mu_2)(tt)$. We may assume that $\rho \in (0, 1)$ — if ρ is equal to zero or one then we can conclude by induction.

By the semantics of commands, we have

$$[\![\mathbf{if}\ b\ \mathbf{then}\ c\ \mathbf{else}\ c']\!](\mu) = \rho \cdot [\![c]\!](\mu_t) + (1 - \rho) \cdot [\![c']\!](\mu_f)$$

where μ_t is the distribution μ conditioned on $b = tt$, and μ_f is the distribution μ conditioned on $b = ff$. Recall that by the induction hypothesis, we have:

$$[\![c]\!](\mu_t) \models \psi * [b = tt] \text{ and } [\![c']\!](\mu_f) \models \psi * [b = ff].$$

Thus, we can decompose the output states into

$$\nu \circ \nu_t \sqsubseteq [\![c]\!](\mu_t) \text{ and } \nu \circ \nu_f \sqsubseteq [\![c]\!](\mu_f)$$

such that

$$\nu \models \psi \text{ and } \nu_t \models [b = tt] \text{ and } \nu_f \models [b = ff]$$

noting that ν can be taken to be the same in both branches since $\psi \in \mathrm{SP}$; by lemma 2.3.7, we may also assume that $\mathbf{dom}(\nu_t) = \mathbf{dom}(\nu_f)$. Thus, we

have:

$$\begin{aligned}\rho \cdot \nu \circ \nu_t + (1-\rho) \cdot \nu \circ \nu_f &= \rho \cdot (\nu \otimes \nu_t) + (1-\rho) \cdot (\nu \otimes \nu_f) \\ &= \nu \otimes (\nu_t \oplus_\rho \nu_f) \\ &= \nu \circ (\nu_t \oplus_\rho \nu_f) \\ &\sqsubseteq [\![\textbf{if } b \textbf{ then } c \textbf{ else } c']\!](\mu),\end{aligned}$$

and we can conclude since $\nu \models \psi$ and $\nu_t \oplus_\rho \nu_f \models \mathsf{Own}(b)$.

Loop For any $\mu \models \varphi$, the side condition implies $\mu \models \textbf{Detm}\langle b \rangle$. We show by induction that $\mu \models \varphi$ implies that for any $n > 0$, $[\![(\textbf{if } b \textbf{ then } c)^n]\!](\mu) \models \varphi \wedge \textbf{Detm}\langle b \rangle$.

Base case: $[\![(\textbf{if } b \textbf{ then } c)^0]\!](\mu) = \mu$, so it satisfies φ. By side condition that $\models \varphi \to \textbf{Detm}\langle b \rangle$, we have $[\![(\textbf{if } b \textbf{ then } c)^0]\!](\mu) \models \varphi \wedge \textbf{Detm}\langle b \rangle$.

Inductive case: Say $\mu' = [\![(\textbf{if } b \textbf{ then } c)^n]\!](\mu)$. By inductive hypothesis, $\mu' \models \varphi \wedge \textbf{Detm}\langle b \rangle$, there are two possibilities:

- $\mu' \models \varphi \wedge [b = \mathit{ff}]$, then

  $$[\![(\textbf{if } b \textbf{ then } c)^{n+1}]\!](\mu) = [\![\textbf{if } b \textbf{ then } c]\!](\mu') = \mu',$$

  which implies that $[\![(\textbf{if } b \textbf{ then } c)^{n+1}]\!](\mu) \models \varphi \wedge [b = \mathit{ff}]$.

- $\mu' \models \varphi \wedge [b = \mathit{tt}]$, then

  $$[\![(\textbf{if } b \textbf{ then } c)^{n+1}]\!](\mu) = [\![\textbf{if } b \textbf{ then } c]\!](\mu') = [\![c]\!](\mu'),$$

  which implies that $[\![(\textbf{if } b \textbf{ then } c)^{n+1}]\!](\mu) \models \varphi$ because $\vdash \{\varphi \wedge [b = \mathit{tt}]\}\ c\ \{\varphi\}$. Since $\models \varphi \to \textbf{Detm}\langle b \rangle$, so $[\![\textbf{if } b \textbf{ then } c^{n+1}]\!](\sigma', \mu') \models \varphi \wedge \textbf{Detm}\langle b \rangle$.

In both cases, $[\![\textbf{if } b \textbf{ then } c^{n+1}]\!](\sigma', \mu') \models \varphi \wedge \textbf{Detm}\langle b \rangle$.

Since we assumed that the loop ends in finite step, there exists a finite number N such that $[\![(\textbf{if } b \textbf{ then } c)^N]\!](\mu) \models \varphi \wedge [b = ff]$ (and also $[\![(\textbf{if } b \textbf{ then } c)^{N-1}]\!](\mu) \models \varphi \wedge [b = tt]$ if $N > 1$, but this fact is not used in this proof).

Then $[\![\textbf{while } b \textbf{ do } c]\!](\mu) = [\![(\textbf{if } b \textbf{ then } c)^N]\!](\mu) \models \varphi \wedge [b = ff]$.

WEAK By induction hypothesis and semantics of implication.

TRUE Trivial.

CONJ By induction hypothesis and semantics of conjunction.

CASE By case analysis.

CONST The fact that $[\![c]\!](\mu) \models \psi$ follows by induction. To show $[\![c]\!](\mu) \models \eta$, by the restriction property we have $\pi_{\mathrm{FV}(\eta)}(\mu) \models \eta$ initially, and since the free variables of η are disjoint from the modified variables of c, we have $\pi_{\mathrm{FV}(\eta)}([\![c]\!]\mu) \models \eta$ as well. Thus, by monotonicity, $[\![c]\!](\mu) \models \eta$ as desired.

FRAME There exist μ_1, μ_2 such that $\mu_1 \circ \mu_2 \sqsubseteq \mu$, and $\mu_1 \models \varphi$ and $\mu_2 \models \eta$; let $S_1 \triangleq \textbf{dom}(\mu_1)$, and note that $T \cup \mathrm{RV}(c) \subseteq S_1$ by the last side-condition.

By the restriction property we have $\pi_{\mathrm{FV}(\eta)}(\mu_2) \models \eta$; let $S_2 \triangleq \textbf{dom}(\mu_2) \cap \mathrm{FV}(\eta)$ and note that S_1 and S_2 are disjoint. Let S_3 be the set of all variables not contained in S_1 or S_2. Since $\mathrm{WV}(c)$ is disjoint from S_2 by the first side-condition, we must have $\mathrm{WV}(c) \subseteq S_1 \cup S_3$.

By induction, we have $[\![c]\!](\mu) \models \psi$. The restriction property gives $\pi_{\mathrm{FV}(\psi)}([\![c]\!](\mu)) \models \psi$.

By the third side-condition, $\mathrm{RV}(c) \subseteq S_1$. By soundness of RV and WV, all variables in $\mathrm{WV}(c)$ must be written to before they are read and there is a function $F : \textbf{Mem}[S_1] \to \mathcal{D}(\textbf{Mem}[\mathrm{WV}(c) \cup S_1])$ such that:

$$\pi_{\mathrm{WV}(c) \cup S_1}([\![c]\!]\mu) = \mathsf{bind}(\mu, m \mapsto F(\pi_{S_1}(m))).$$

Since $S_2 \subseteq \mathrm{FV}(\eta)$, variables in S_2 are not in $\mathrm{MV}(c)$ by the first side-condition, and S_2 is disjoint from $\mathrm{WV}(c) \cup S_1$. By soundness of MV, we have:

$$\pi_{(\mathrm{WV}(c)\cup S_1)\cup S_2}([\![c]\!]\mu) = \mathsf{bind}(\pi_{(\mathrm{WV}(c)\cup S_1)\cup S_2}(\mu), (m_1, m_2) \mapsto F(m_1) \otimes \mathsf{unit}(m_2)).$$

Since S_1 and S_2 are independent in μ, we know that $S_1 \cup \mathrm{WV}(c)$ and S_2 are independent in $[\![c]\!](\mu)$ as well. Hence:

$$[\![c]\!]\mu \sqsupseteq (\pi_{S_1 \cup \mathrm{WV}(c)}([\![c]\!]\mu)) \circ (\pi_{S_2}([\![c]\!]\mu)).$$

We know that $FV(\psi) \subseteq T \cup \mathrm{WV}(c) \subseteq S_1 \cup \mathrm{WV}(c)$ so since ψ is valid in $[\![c]\!](\mu)$, it is valid in the first conjunct by the restriction property and the second side-condition. Since $\pi_{S_2}([\![c]\!]\mu) = \pi_{S_2}(\mu)$, and η does not depend on modified deterministic variables, η is valid in the second conjunct. Thus, we can conclude:

$$[\![c]\!](\mu) \models \psi * \eta. \qquad \square$$

APPENDIX B

LINA: A SEPARATION LOGIC FOR NEGATIVE DEPENDENCE

B.1 Preliminaries

Lemma B.1.1. *Say* $\mathcal{S} = \{S_i \mid 1 \le i \le N\}$ *where* S_i *are disjoint,* $S = \cup\mathcal{S}$ *and* $\mu \in \mathbf{Mem}[S]$*,*

Then, S_i *are independent in* μ *if and only if for any family of all monotone or all antitone functions* $f_i : \mathbf{Mem}[S_i] \to \mathbb{R}^+$*,*

$$\mathbb{E}_{x\sim\mu}\left[\prod_{S_i\in\mathcal{S}} f_i(\pi_{S_i}x)\right] = \prod_{S_i\in\mathcal{S}} \mathbb{E}_{x\sim\mu}[f_i(\pi_{S_i}x)]. \tag{B.1}$$

Proof. The forward direction is straightforward. The backward direction needs more careful analysis. In general, zero correlation does not imply independence, but here, we have the equality for all family of monotone or antitone functions, so that suffices for independence.

We prove by induction on $\mathcal{T} = \{S_i \mid 1 \le i \le K\}$ that for any family of $v_i \in \mathbf{Mem}[S_i]$,

$$\mathbb{E}_{x\sim\mu}\left[\left(\bigwedge_{S_i\in\mathcal{T}} \pi_{S_i}x = v_i\right) \wedge \left(\bigwedge_{S_i\in\mathcal{S}\setminus\mathcal{T}} \pi_{S_i}x < v_i\right)\right] = \prod_{S_i\in\mathcal{T}} \mathbb{E}_{x\sim\mu}\left[\pi_{S_i}x = v_i\right] \cdot \prod_{S_i\in\mathcal{S}\setminus\mathcal{T}} \mathbb{E}_{x\sim\mu}\left[\pi_{S_i}x < v_i\right]. \tag{B.2}$$

Case $|\mathcal{T}| = 1$**:** Say $\mathcal{T} = \{S_j\}$. Since indicator functions $S_i < v_i$ and $S_i \le v_i$ are

both monotonically decreasing,

$$\mathbb{E}_{x\sim\mu}\left[\pi_{S_j}x = v_j \wedge \left(\bigwedge_{S_i\in\mathcal{S}\setminus\mathcal{T}} \pi_{S_i}x < v_i\right)\right]$$
$$= \mathbb{E}_{x\sim\mu}\left[\pi_{S_j}x \le v_j \wedge \left(\bigwedge_{S_i\in\mathcal{S}\setminus\mathcal{T}} \pi_{S_i}x < v_i\right)\right] - \mathbb{E}_{x\sim\mu}\left[\pi_{S_j}x < v_j \wedge \left(\bigwedge_{S_i\in\mathcal{S}\setminus\mathcal{T}} \pi_{S_i}x < v_i\right)\right]$$
$$= \mathbb{E}_{x\sim\mu}\left[\pi_{S_j}x \le v_j\right] \cdot \prod_{S_i\in\mathcal{S}\setminus\mathcal{T}} \mathbb{E}_{x\sim\mu}\left[\pi_{S_i}x < v_i\right] - \mathbb{E}_{x\sim\mu}\left[\pi_{S_j}x < v_j\right] \cdot \prod_{S_i\in\mathcal{S}\setminus\mathcal{T}} \mathbb{E}_{x\sim\mu}\left[\pi_{S_i}x < v_i\right]$$

(By Equation (B.1))

$$= \left(\mathbb{E}_{x\sim\mu}\left[\pi_{S_j}x \le v_j\right] - \mathbb{E}_{x\sim\mu}\left[\pi_{S_j}x < v_j\right]\right) \cdot \prod_{S_i\in\mathcal{S}\setminus\mathcal{T}} \mathbb{E}_{x\sim\mu}\left[\pi_{S_i}x < v_i\right]$$
$$= \mathbb{E}_{x\sim\mu}\left[\pi_{S_j}x = v_j\right] \cdot \prod_{S_i\in\mathcal{S}\setminus\mathcal{T}} \mathbb{E}_{x\sim\mu}\left[\pi_{S_i}x < v_i\right]$$

Case $|\mathcal{T}| > 1$ Let S_j be an element in $\mathcal{T}$.

$$\mathbb{E}_{x\sim\mu}\left[\left(\bigwedge_{S_i\in\mathcal{T}} \pi_{S_i}x = v_i\right) \wedge \left(\bigwedge_{S_i\in\mathcal{S}\setminus\mathcal{T}} \pi_{S_i}x < v_i\right)\right]$$
$$= \mathbb{E}_{x\sim\mu}\left[\pi_{S_j}x \le v_j \wedge \left(\bigwedge_{S_i\in\mathcal{T}\setminus\{S_j\}} \pi_{S_i}x = v_i\right) \wedge \left(\bigwedge_{S_i\in\mathcal{S}\setminus\mathcal{T}} \pi_{S_i}x < v_i\right)\right]$$
$$- \mathbb{E}_{x\sim\mu}\left[\pi_{S_j}x < v_j \wedge \left(\bigwedge_{S_i\in\mathcal{T}\setminus\{S_j\}} \pi_{S_i}x = v_i\right) \wedge \left(\bigwedge_{S_i\in\mathcal{S}\setminus\mathcal{T}} \pi_{S_i}x < v_i\right)\right]$$
$$= \mathbb{E}_{x\sim\mu}\left[\pi_{S_j}x \le v_j\right] \cdot \prod_{S_i\in\mathcal{T}\setminus\{S_j\}} \mathbb{E}_{x\sim\mu}\left[\pi_{S_i}x = v_i\right] \cdot \prod_{S_i\in\mathcal{S}\setminus\mathcal{T}} \mathbb{E}_{x\sim\mu}\left[\pi_{S_i}x < v_i\right]$$
$$- \mathbb{E}_{x\sim\mu}\left[\pi_{S_j}x < v_j\right] \cdot \prod_{S_i\in\mathcal{T}\setminus\{S_j\}} \mathbb{E}_{x\sim\mu}\left[\pi_{S_i}x = v_i\right] \cdot \prod_{S_i\in\mathcal{S}\setminus\mathcal{T}} \mathbb{E}_{x\sim\mu}\left[\pi_{S_i}x < v_i\right]$$
$$= \mathbb{E}_{x\sim\mu}\left[\pi_{S_j}x = v_j\right] \cdot \prod_{S_i\in\mathcal{T}\setminus\{S_j\}} \mathbb{E}_{x\sim\mu}\left[\pi_{S_i}x = v_i\right] \cdot \prod_{S_i\in\mathcal{S}\setminus\mathcal{T}} \mathbb{E}_{x\sim\mu}\left[\pi_{S_i}x < v_i\right]$$

When $\mathcal{T} = \mathcal{S}$, Equation (B.2) implies

$$\mathbb{E}_{x\sim\mu}\left[\bigwedge_{S_i\in\mathcal{S}} \pi_{S_i}x = v_i\right] = \prod_{S_i\in\mathcal{S}} \mathbb{E}_{x\sim\mu}\left[\pi_{S_i}x = v_i\right]$$

for any v_i's. Thus, components in $\mathcal{S}$ are independent. □

We prove some properties of coarsening. In the following we will use an alternative definition of coarsening, which will be shown to be equivalent to what we define in the main text.

Definition B.1.1 (Alternative definition of coarsening)**.** We first index any partition $\mathcal{S}$ as $\mathcal{S}_1, \ldots, \mathcal{S}_{|\mathcal{S}|}$. Say $|\mathcal{S}'| = m$, $|\mathcal{S}| = n$. We say $\mathcal{S}'$ coarsens a partition $\mathcal{S}$ there exists a function a $f : [m] \to \mathcal{P}([n])$ such that 1) $\cup_{i\in[m]} f(i) = [n]$; 2) for any $i, j \in [m]$, either $i = j$ or $f(i), f(j)$ are disjoint; 3) $\mathcal{S}' = \{\cup\{\mathcal{S}_j \mid j \in f(i)\} \mid i \in [m]\}$.

Lemma B.1.2. *Let $\mathcal{S}$, $\mathcal{S}'$ be two partitions. Then $\mathcal{S}'$ coarsens $\mathcal{S}$ according to Definition B.1.1 if and only if $\mathcal{S}'$ coarsens $\mathcal{S}$ according to Definition 3.3.4 .*

Proof. We index $\mathcal{S}$ as $\mathcal{S}_1, \ldots, \mathcal{S}_n$ and $\mathcal{S}'$ as $\mathcal{S}'_1, \ldots, \mathcal{S}'_{|m|}$.

Backward direction: By that definition, we know a) for any $\mathcal{S}'_i \in \mathcal{S}'$, $\mathcal{S}'_i = \cup\mathcal{R}$ for some $\mathcal{R} \subseteq \mathcal{S}$; b) $\cup\mathcal{S} = \cup\mathcal{S}'$.

We define the function $g : [m] \to \mathcal{P}([n])$ as $g(i) = \{j \mid \mathcal{S}_j \subseteq \mathcal{S}'_i\}$. This g would satisfies all the conditions required:

1. By substitution, $\cup_{i\in[m]} g(i) = \cup_{i\in[m]}\{j \mid \mathcal{S}_j \subseteq \mathcal{S}'_i\} = \cup_{s'\in\mathcal{S}'}\{j \mid \mathcal{S}_j \subseteq s'\}$. By b), for any $j \in [n]$, $\mathcal{S}_j \subseteq \cup\mathcal{S}'$. Then by a) and that $\mathcal{S}$ is a partition, if s' covers any of $\mathcal{S}_j$, it must covers all of $\mathcal{S}_j$, then $\mathcal{S}_j \subseteq \cup\mathcal{S}'$ implies there exists $s' \in \mathcal{S}'$ such that $\mathcal{S}_j \subseteq s'$. Thus, $j \in \{j \mid \mathcal{S}_j \subseteq s'\} \subseteq \cup_{s'\in\mathcal{S}'}\{j \mid \mathcal{S}_j \subseteq s'\}$. For any $j \notin [n]$, $\mathcal{S}_j$ is undefined, so it is impossible that $\mathcal{S}_j \subseteq s'$ for some $s' \subseteq \mathcal{S}'$. Therefore, $\cup_{s'\in\mathcal{S}'}\{j \mid \mathcal{S}_j \subseteq s'\} = [n]$.
2. For any $k \in g(i)$, $\mathcal{S}_k \subseteq \mathcal{S}'_i$. If $i \neq j$, then $\mathcal{S}'_i$ and $\mathcal{S}'_j$ are disjoint since $\mathcal{S}'$ is a partition. Thus, $\mathcal{S}_k \not\subseteq \mathcal{S}'_j$, and $k \notin g(j)$. So for any $i \neq j$, $g(i), g(j)$ are disjoint.

3. By substitution,

$$\{\cup\{\mathcal{S}_j \mid j \in g(i)\} \mid i \in [m]\} = \{\cup\{\mathcal{S}_j \mid \mathcal{S}_j \subseteq \mathcal{S}'_i\} \mid i \in [m]\} \quad = \{\cup\{\mathcal{S}_j \mid \mathcal{S}_j \subseteq s'\} \mid s' \in \mathcal{S}'\}.$$

Again, by a) and that $\mathcal{S}$ is a partition, if $s' \in \mathcal{S}$ covers any part of of $\mathcal{S}_j$, it must covers all of $\mathcal{S}_j$, so $\cup\{\mathcal{S}_j \mid \mathcal{S}_j \subseteq s'\} = s'$. Thus, $\{\cup\{\mathcal{S}_j \mid \mathcal{S}_j \subseteq s'\} \mid s' \in \mathcal{S}'\} = \mathcal{S}'$.

Forward direction: By 3), we know that $\mathcal{S}' = \{\cup\{\mathcal{S}_j \mid j \in f(i)\} \mid i \in [m]\}$. So for any $\mathcal{S}'_i \in \mathcal{S}'$, we have $s' = \cup\{\mathcal{S}_j \mid j \in f(i)\}$, which is a subset of $\mathcal{S}'$ by construction. So we proved a). Also, $\cup\mathcal{S}' = \cup\{\cup\{\mathcal{S}_j \mid j \in f(i)\} \mid i \in [m]\} = \cup\{\mathcal{S}_j \mid j \in f(i) \mid i \in [m]\}$, and by 1), that is equivalent to $\cup\{\mathcal{S}_j \mid j \in [n]\}$, which is equivalent to $\cup\mathcal{S}$. □

We can prove that coarsening commute with projections.

Lemma B.1.3. *Given a partition $\mathcal{S} = \{\mathcal{S}_i\}_i$ and a set X, let $\mathcal{S}_X = \{\mathcal{S}_i \cap X \mid \mathcal{S}_i \in \mathcal{S}\}$. For any $\mathcal{T}$ coarsening $\mathcal{S}_X$, there exists a coarsening $\mathcal{S}'$ of $\mathcal{S}$ such that $\mathcal{T} = \{\mathcal{S}_i \cap X \mid \mathcal{S}_i \in \mathcal{S}'\}$; conversely, for any $\mathcal{S}'$ coarsening $\mathcal{S}$, and $\mathcal{S}'_X = \{\mathcal{S}_i \cap X \mid \mathcal{S}_i \in \mathcal{S}'\}$, we have $\mathcal{S}'_X$ coarsens $\mathcal{S}_X$.*

Proof. Forward direction: By Definition B.1.1, there exists a coarsening function f such that

$$\begin{aligned}
\mathcal{T} &= \{\cup\{(\mathcal{S}_X)_j \mid j \in f(i)\} \mid i \in [|\mathcal{T}|]\} \\
&= \{\cup\{\mathcal{S}_j \cap X \mid j \in f(i)\} \mid i \in [|\mathcal{T}|]\} \\
&= \{(\cup\{\mathcal{S}_j \mid j \in f(i)\}) \cap X \mid i \in [|\mathcal{T}|]\} \\
&= \{\mathcal{S}' \cap X \mid \mathcal{S}' \in \mathcal{S}'\} \qquad \text{(where } \mathcal{S}' = \{\cup\{\mathcal{S}_j \mid j \in f(i)\} \mid i \in [|\mathcal{T}|]\}\text{)}
\end{aligned}$$

$\mathcal{S}'$ has the same size as $\mathcal{T}$, so $\mathcal{S}' = \{\cup\{\mathcal{S}_j \mid j \in f(i)\} \mid i \in [|\mathcal{S}'|]\}$, and thus $\mathcal{S}'$ coarsens $\mathcal{S}$.

Backward direction: $\mathcal{S}'$ coarsens $\mathcal{S}$, so there exists a coarsening function f such that

$$\mathcal{S}' = \{\cup\{S_j \mid j \in f(i)\} \mid i \in [|\mathcal{S}'|]\}.$$

Thus,

$$\begin{aligned}\mathcal{S}'_X &= \{(\cup\{S_j \mid j \in f(i)\}) \cap X \mid i \in [|\mathcal{S}'|]\} \\ &= \{\cup\{S_j \cap X \mid j \in f(i)\} \mid i \in [|\mathcal{S}'|]\} \\ &= \{\cup\{S_{Xj} \mid j \in f(i)\} \mid i \in [|\mathcal{S}'|]\}.\end{aligned}$$

Therefore, $\mathcal{S}'_X$ coarsens $\mathcal{S}_X$.

□

B.2 A BI Frame for Negative Association

B.2.1 Capturing Negative Association

Theorem 3.3.2. *For any two states $\mu_1, \mu_2 \in X$, $\mu_1 \oplus_s \mu_2 \subseteq \mu_1 \oplus \mu_2 \subseteq \mu_1 \oplus_w \mu_2$.*

Proof. Let S denote $\mathbf{dom}(\mu_1)$ and T denote $\mathbf{dom}(\mu_2)$.

For any $\mu \in \mu_1 \oplus_s \mu_2$, we have $\pi_S\mu = \mu_1$, $\pi_T\mu = \mu_2$, and μ satisfies NA. μ being NA implies μ is $\mathcal{R}$-PNA for any partition $\mathcal{R}$ on $\mathbf{dom}(\mu)$ So for any partition $\mathcal{S}$ on S, partition $\mathcal{T}$ on T, μ is $\mathcal{S} \cup \mathcal{T}$-PNA. Therefore, $\mu \in \mu_1 \oplus \mu_2$.

For any $\mu \in \mu_1 \oplus \mu_2$, $\pi_S \mu = \mu_1$, $\pi_T \mu = \mu_2$, and μ is $\{S, T\}$-PNA since μ_1 is $\{S\}$-PNA, μ_2 is $\{T\}$-PNA. Thus, $\mu \in \mu_1 \oplus_w \mu_2$. □

Theorem 3.3.1. *Given a set of variables S, S satisfies NA in μ iff μ satisfies $\mathcal{S}$-PNA for any $\mathcal{S}$ partitioning S iff μ satisfies $\{\{x\} \mid x \in S\}$-PNA.*

Proof. The second equivalence is straightforward:

- $\{\{s\} \mid s \in \mathcal{S}\}$ is a partition of S, so we have the backward direction.
- Any $\mathcal{S}$ partitioning S coarsens $\{\{s\} \mid s \in S\}$, so we have the first direction.

For the forward direction of the first equivalence, it suffices to prove that for any partition $\mathcal{S}$ of S, any family of all monotone or all antitone functions $f_i : \mathbf{Mem}[S_i] \to \mathbb{R}^+$,

$$\mathbb{E}_{m \sim \mu}\left[\prod_{S_i \in \mathcal{S}} f_i(\pi_{S_i} m)\right] \leq \prod_{S_i \in \mathcal{S}} \mathbb{E}_{m \sim \mu}\left[f_i(\pi_{S_i} m)\right]. \tag{B.3}$$

We prove that by induction on the size of $\mathcal{S}$.

Base case $|\mathcal{S}| = 1$: $\mathcal{S}$-PNA is trivial.

Base case $|\mathcal{S}| = 2$: $\mathcal{S}$-PNA is straightforward from NA.

Inductive case: Assuming μ satisfies $\mathcal{S}$-PNA for any partition with size less than K, we want to show that μ satisfies $\mathcal{S}$-PNA for any partition with size equals to K.

Say $\mathcal{S} = \{S_1, \ldots, S_K\}$. For any family of all monotone or all antitone functions $f_i : \mathbf{Mem}[S_i] \to \mathbb{R}^+$, either both $m \mapsto \prod_{i=1}^{K-1} f_i(\pi_{S_i} m)$ and f_K are monotone, or both $m \mapsto \prod_{i=1}^{K-1} f_i(\pi_{S_i} m)$ and f_K are antitone. Thus, by the induc-

tive hypothesis

$$\mathbb{E}_{x\sim\mu}\left[\prod_{i=1}^{K} f_i(\pi_{S_i}m)\right] \le \mathbb{E}_{x\sim\mu}\left[\prod_{i=1}^{K-1} f_i(\pi_{S_i}m)\right]\cdot\mathbb{E}_{x\sim\mu}\left[f_K(\pi_{S_K}m)\right]$$

(We can partition $\cup_{i=1}^{K} S_i$ into $\{\cup_{i=1}^{K-1} S_i, S_K\}$)

$$\le \prod_{i=1}^{K-1}\mathbb{E}_{x\sim\mu}\left[f_i(\pi_{S_i}m)\right]\cdot\mathbb{E}_{x\sim\mu}\left[f_K(\pi_{S_K}m)\right] \tag{B.4}$$

(We can partition $\cup_{i=1}^{K} S_i$ into $\{S_1,\ldots,S_K\}$)

$$= \prod_{i=1}^{K}\mathbb{E}_{x\sim\mu}\left[f_i(\pi_{S_i}m)\right]. \tag{B.5}$$

The backward direction of the first equivalence is more involved. For any two disjoint $A, B \subseteq S$, we know μ satisfies $\{A, B\}$-PNA, so for every pair of both monotone or both antitone functions $f : \mathbf{Mem}[A] \to \mathbb{R}^+$, $g : \mathbf{Mem}[B] \to \mathbb{R}^+$, we have

$$\mathbb{E}_{m\sim\mu}[f(\pi_A m)\cdot g(\pi_B m)] \le \mathbb{E}_{m\sim\mu}[f(\pi_A m)]\cdot\mathbb{E}_{m\sim\mu}[g(\pi_B m)].$$

But the problem is to show this inequality when f, g are not both non-negative. We prove that in three steps:

1. If f, g are lower-bounded by $-L$, i.e., $f(x) \ge -L$ and $g(x) \ge -L$ for any x. Then $x \to f(x)+L$ and $x \to g(x)+L$ are both non-negative functions. Thus,

$$\mathbb{E}_{m\sim\mu}[(f(\pi_A m)+L)\cdot(g(\pi_B m)+L)] \le \mathbb{E}_{m\sim\mu}[f(\pi_A m)+L]\cdot\mathbb{E}_{m\sim\mu}[g(\pi_B m)+L]. \tag{B.6}$$

Meanwhile,

$$\begin{aligned}
&\mathbb{E}[(f(\pi_A m) + L) \cdot (g(\pi_B m) + L)] \\
&= \mathbb{E}[f(\pi_A m) \cdot g(\pi_B m)] + L \cdot \mathbb{E}[f(\pi_A m)] + L \cdot \mathbb{E}[g(\pi_B m)] + L^2 \\
&\mathbb{E}[f(\pi_A m) + L] \cdot \mathbb{E}[g(\pi_B m) + L] \\
&= (\mathbb{E}[f(\pi_A m)] + L) \cdot (\mathbb{E}[g(\pi_B m)] + L) \\
&= \mathbb{E}[f(\pi_A m)] \cdot \mathbb{E}[g(\pi_B m)] + L \cdot \mathbb{E}[f(\pi_A m)] + L \cdot \mathbb{E}[g(\pi_B m)] + L^2.
\end{aligned}$$

So Equation (B.6) implies that

$$\mathbb{E}_{m\sim\mu}[f(\pi_A m) \cdot g(\pi_B m)] \leq \mathbb{E}_{m\sim\mu}[f(\pi_A m)] \cdot \mathbb{E}_{m\sim\mu}[g(\pi_B m)].$$

2. If the codomain of f or g does not range across both negative and positive numbers, then we can also prove the desired inequality by applying the monotone convergence theorem on the result for lower-bounded functions.

   - Say f is non-negative and g is non-positive. For any natural number n, $m \in \mathbf{Mem}[A \cup B]$, we define $g_n(\pi_B m) = \max(g(\pi_B m), -n)$, $h_n(m) = f(\pi_A m) \cdot g_n(\pi_B m)$. Then for any n, g_n and h_n are lower-bounded non-positive functions; and for any m, $\{g_n(m)\}_{n\in\mathbb{N}}$ is a monotonically decreasing sequence converging to $g(m)$, $\{h_n(m)\}_{n\in\mathbb{N}}$ is a monotonically decreasing sequence converging to $f(\pi_A m)\cdot g(\pi_B m)$. By the monotone convergence theorem,

$$\begin{aligned}
\mathbb{E}_{m\sim\mu} f(\pi_A m) \cdot g(\pi_B m) &= \lim_{n\to\infty} \mathbb{E}_{m\sim\mu} h_n(m) \\
\mathbb{E}_{m\sim\mu} g(\pi_B m) &= \lim_{n\to\infty} \mathbb{E}_{m\sim\mu} g_n(\\
pi_B m).
\end{aligned}$$

By what we proved above, for any n, we have

$$\mathbb{E}_{m\sim\mu}[h_n(m)] \le \mathbb{E}_{m\sim\mu}[f(\pi_A m)] \cdot \mathbb{E}_{m\sim\mu}[g_n(\pi_B m)]$$

Taking that to the limit $n \to \infty$,

$$\begin{aligned}\lim_{n\to\infty} \mathbb{E}_{m\sim\mu}[h_n(m)] &\le \lim_{n\to\infty} \left(\mathbb{E}_{m\sim\mu}[f(\pi_A m)] \cdot \mathbb{E}_{m\sim\mu}[g(\pi_B m)]\right) \\ &= \lim_{n\to\infty} \mathbb{E}_{m\sim\mu}[f(\pi_A m)] \cdot \lim_{n\to\infty} \mathbb{E}_{m\sim\mu}[g(\pi_B m)]\end{aligned}$$

Therefore, for any distribution $\mu \in \mathcal{D}(\mathbf{Mem}[A \cup B])$,

$$\mathbb{E}_{m\sim\mu}[f(\pi_A m) \cdot g(\pi_B m)] \le \mathbb{E}_{m\sim\mu}[f(\pi_A m)] \cdot \mathbb{E}_{m\sim\mu}[g(\pi_B m)].$$

- The case where f is non-positive and g is non-negative is symmetric.
- The case where f and g are both non-positive is also similar. We will define $f_n(\pi_A m) = \max(f(\pi_A m), -n)$, $g_n(\pi_B m) = \max(g(\pi_B m), -n)$, $h_n(m) = f_n(\pi_A m) \cdot g_n(\pi_B m)$. Then we have

$$\begin{aligned}\mathbb{E}_{m\sim\mu} f(\pi_A m) \cdot g(\pi_B m) &= \lim_{n\to\infty} \mathbb{E}_{m\sim\mu} h_n(m) \\ \mathbb{E}_{m\sim\mu} g(\pi_B m) &= \lim_{n\to\infty} \mathbb{E}_{m\sim\mu} g_n(\\ & pi_B m) \\ \mathbb{E}_{m\sim\mu} f(\pi_B m) &= \lim_{n\to\infty} \mathbb{E}_{m\sim\mu} f_n(\\ & pi_A m).\end{aligned}$$

  And the rest follows.

3. Now we consider the general case where we only know both f and g are either lower-bounded or upper bounded.
   - If both f and g are lower-bounded, reduce to the first case.

- If f is lower-bounded by L, g is upper-bounded by U, then we can consider function $f' = f + L$ and $g' = g - U$. Then f' is non-negative and g' is non-positive, so by step 2, we have

  $$\mathbb{E}_{m\sim\mu}[f'(\pi_A m) \cdot g'(\pi_B m)] \leq \mathbb{E}_{m\sim\mu}[f'(\pi_A m)] \cdot \mathbb{E}_{m\sim\mu}[g'(\pi_B m)].$$

  By calculations analogous to what we did in step 1, that implies

  $$\mathbb{E}_{m\sim\mu}[f(\pi_A m) \cdot g(\pi_B m)] \leq \mathbb{E}_{m\sim\mu}[f(\pi_A m)] \cdot \mathbb{E}_{m\sim\mu}[g(\pi_B m)].$$

- If f is upper-bounded and g is lower-bounded: analogous to above.
- If both f and g are upper-bounded: also, analogous to above.

Thus, μ satisfies $\{A, B\}$-PNA implies μ satisfies (A, B)-NA. And therefore, μ satisfies $\{A, B\}$-PNA for any $A, B \subseteq S$ implies S satisfies strong NA in μ.

□

B.2.2 Omitted Proofs of Frame Conditions

Theorem 3.3.4. *The structure $\mathcal{X}_{PNA} = (X_{\mathbb{D}}, \sqsubseteq_{\mathbb{D}}, \oplus, E_{\mathbb{D}})$ is a Down-Closed BI frame.*

Proof. We sketch the conditions, using the notation from the definition:

Down-Closed. Let $\mathbf{dom}(x) = S, \mathbf{dom}(x') = S', \mathbf{dom}(y) = T, \mathbf{dom}(y') = T'$. We claim that we can take $z' = \pi_{S'\cup T'} z$. We evidently have $z \sqsupseteq z'$, and $\pi_{S'} z' = \pi_{S'}\pi_S z = x'$ and $\pi_{T'} z' = \pi_{T'}\pi_T z = y'$.

What remains to show is that z' is $\mathcal{S} \cup \mathcal{T}$-PNA for any $\mathcal{S}, \mathcal{T}$ such that x' is $\mathcal{S}$-PNA, y' is $\mathcal{T}$-PNA, and $(\cup\mathcal{S}) \cap (\cup\mathcal{T}) = \emptyset$.

If x' is $\mathcal{S}$-PNA, then x is $\mathcal{S}$-PNA; if y' is $\mathcal{T}$-PNA, then y is $\mathcal{T}$-PNA; then $z \in x \oplus y$ must be $\mathcal{S} \cup \mathcal{T}$-PNA. Since $z' := \pi_{S' \cup T'} z$, and $(\cup \mathcal{S}) \cup (\cup \mathcal{T}) \subseteq S' \cup T'$, we have z' is $\mathcal{S} \cup \mathcal{T}$-PNA too. And evidently, $\mathbf{dom}(z') = S' \cup T' = \mathbf{dom}(x') \cup \mathbf{dom}(y')$. So $z' \in x' \oplus y'$.

Commutativity Immediate.

Associativity Let $\mathbf{dom}(x) = R, \mathbf{dom}(y) = S, \mathbf{dom}(z) = T$. We can assume that these sets are all disjoint, otherwise there is nothing to prove. We claim that we can take $s = \pi_{S \cup T} w$. For any w in $t \oplus z$, $t \in x \oplus y$, we want to show that $w \in x \oplus s$ and $s \in y \oplus z$.

- For any partition $\mathcal{R}, \mathcal{S}$ such that $(\cup \mathcal{R}) \cap (\cup \mathcal{S}) = \emptyset$ and x is $\mathcal{R}$-PNA, s is $\mathcal{S}$-PNA. For set $X \subseteq \mathbf{Var}$, write $\{Y \cap X \mid Y \in \mathcal{S}\}$ as $\mathcal{S}_X$. Then, by Lemma B.1.3, s is $\mathcal{S}$-PNA implies y must be $\mathcal{S}_S$-PNA. Similarly, s is $\mathcal{S}$-PNA implies z must be $\mathcal{S}_T$-PNA.

  Then, $t \in x \oplus y$ must be $\mathcal{R} \cup (\mathcal{S}_S)$-PNA, and $w \in t \oplus z$ must be $\mathcal{R} \cup \mathcal{S}_S \cup \mathcal{S}_T$-PNA. Note that $\mathcal{S}$ coarsens $\mathcal{S}_S \cup \mathcal{S}_T$ so w is $\mathcal{R} \cup \mathcal{S}_S \cup \mathcal{S}_T$-PNA implies that w is $\mathcal{R} \cup \mathcal{S}$-PNA.

  Also, $\pi_R w = \pi_R \pi_{R \cup S} w = \pi_R t = x$, and $\mathbf{dom}(w) = R \cup S \cup T = \mathbf{dom}(x) \cup \mathbf{dom}(s)$.

  Hence, $w \in x \oplus s$.

- Note that x is trivially $\{R\}$-PNA. Then, for any partition $\mathcal{S}, \mathcal{T}$ such that $R \cap (\cup \mathcal{S}) \cap (\cup \mathcal{T}) = \emptyset$ and y is $\mathcal{S}$-PNA and z is $\mathcal{T}$-PNA, first t must be $(\{R\} \cup \mathcal{S})$-PNA, and then w must be $(\{R\} \cup \mathcal{S} \cup \mathcal{T})$-PNA. By projection, $s = \pi_{S \cup T}$ must be $\mathcal{S} \cup \mathcal{T} z$-PNA.

  Also, $\pi_S s = \pi_S \pi_{S \cup T} w = \pi_S w = \pi_S \pi_{R \cup S} w = \pi_S t = y$, and similarly, $\pi_T s = z$. Also, $\mathbf{dom}(s) = S \cup T = \mathbf{dom}(y) \cup \mathbf{dom}(z)$.

Hence, $s \in y \oplus z$.

Unit Existence Take e to be μ where μ is the (unique) distribution in $\mathcal{D}(\mathbf{Mem}[\emptyset])$.

Unit Closure Immediate as we take $E = M$.

Unit Coherence $x \in y \oplus e$ entails $y = \pi_{\mathbf{dom}(y)} x$, which implies $y \sqsubseteq x$. □

B.3 Soundness and Completeness of M-BI algebras

B.3.1 Algebraic Soundness and Completeness

The proof is very similar to the proof of BI soundness and completeness in section 2.2.3: we first construct a new algebra – "M-BI algebra," prove the algebraic soundness and completeness and then establish the overall theorem.

Definition B.3.1 (M-BI algebra)**.** An M-BI algebra is an algebra $\mathbb{A}_M = (A, \wedge, \vee, \rightarrow, \top, \bot, *_{m \in M}, -\!\!*_{m \in M}, \top^*_{m \in M})$ such that

- For each $m \in M$, the structure $(a, \wedge, \vee, \rightarrow, \top, \bot, *_m, -\!\!*_m, \top^*_m)$ is a BI algebra;
- If $m_1 \leq m_2$ then $a *_{m_1} b \leq a *_{m_2} b$.

We can interpret M-BI in an M-BI algebra $\mathbb{A}_M$. Let $\mathcal{V} : \mathcal{AP} \rightarrow \mathbb{A}_M$ be a map assigning atomic propositions to elements of $\mathbb{A}_M$. We extend $\mathcal{V}$ to an interpretation $[\![-]\!]_{\mathbb{A}}$ mapping M-BI propositions to elements of $\mathbb{A}_M$, defined by:

$$[\![P]\!]_{\mathbb{A}} = \mathcal{V}(P)$$

$$[\![\top]\!]_{\mathbb{A}} = \top$$

$$[\![I_m]\!]_{\mathbb{A}} = \top^*_m$$

$$[\![\bot]\!]_{\mathbb{A}} = \bot$$

$$[\![P \wedge Q]\!]_{\mathbb{A}} = [\![P]\!]_{\mathbb{A}} \wedge [\![Q]\!]_{\mathbb{A}}$$

$$[\![P \vee Q]\!]_{\mathbb{A}} = [\![P]\!]_{\mathbb{A}} \vee [\![Q]\!]_{\mathbb{A}}$$

$$[\![P \to Q]\!]_{\mathbb{A}} = [\![P]\!]_{\mathbb{A}} \to [\![Q]\!]_{\mathbb{A}}$$

$$[\![P *_m Q]\!]_{\mathbb{A}} = [\![P]\!]_{\mathbb{A}} *_m [\![Q]\!]_{\mathbb{A}}$$

$$[\![P \mathrel{-\!\!*}_m Q]\!]_{\mathbb{A}} = [\![P]\!]_{\mathbb{A}} \mathrel{-\!\!*}_m [\![Q]\!]_{\mathbb{A}}$$

Theorem B.3.1 (Algebraic Soundness)**.** *If $P \vdash Q$ is provable, then $[\![P]\!]_{\mathbb{A}} \leq [\![Q]\!]_{\mathbb{A}}$ for any algebraic interpretation $[\![-]\!]_{\mathbb{A}}$.*

Proof. By induction on the derivation of $P \vdash Q$. The cases for everything except $*$-WEAKENING follow from the exact same argument as for standard BI and BI-algebra, as in theorem 2.2.3. For the remaining case of $*$-WEAKENING, which derives $P *_{m_2} Q$ from $P *_{m_1} Q$ if $m_1 \leq m_2$. We have

$$\begin{aligned}
[\![P *_{m_1} Q]\!]_{\mathbb{A}} &= [\![P]\!]_{\mathbb{A}} *_{m_1} [\![Q]\!]_{\mathbb{A}} && \text{(By definition of } [\![-]\!]_{\mathbb{A}}\text{)} \\
&\leq [\![P]\!]_{\mathbb{A}} *_{m_2} [\![Q]\!]_{\mathbb{A}} && \text{(By definition of } M\text{-BI algebra)} \\
&= [\![P *_{m_2} Q]\!]_{\mathbb{A}}. && \text{(By definition of } [\![-]\!]_{\mathbb{A}}\text{)}
\end{aligned}$$

□

Next, we want to show the algebraic completeness. Analogous to before theorem 2.2.6, we construct the Lindenbaum-Tarski algebra corresponding to M-BI.

Definition B.3.2 (Lindenbaum-Tarski Algebra)**.** Define the equivalence relation $P \sim Q$ as $P \vdash Q$ and $Q \vdash P$. Let $[P]_\sim$ be the equivalence class of P under $\sim$. Take I_m, $\top$, and $\bot$ to be $[I_m]_\sim$, $[\top]_\sim$, and $[\bot]_\sim$, respectively. Then we define:

$$\vdots$$

$$[P]_\sim *_m [Q]_\sim = [P *_m Q]_\sim$$

$$[P]_\sim \mathrel{-\!\!*}_m [Q]_\sim = [P \mathrel{-\!\!*}_m Q]_\sim$$

The fact that these operations are well-defined and form a M-BI algebra follows almost entirely from lemma 2.2.4. The only remaining case is to check that if $m_1 \leq m_2$ then $[P]_\sim *_{m_1} [Q]_\sim \leq [P]_\sim *_{m_2} [Q]_\sim$. We have

$$\begin{aligned} [P]_\sim *_{m_1} [Q]_\sim &= [P *_{m_1} Q]_\sim \\ &\leq [P *_{m_2} Q]_\sim && \text{(Since } P *_{m_1} Q \vdash P *_{m_2} Q\text{)} \\ &= [P]_\sim *_{m_2} [Q]_\sim \end{aligned}$$

Then, we can construct an algebraic interpretation into Lindenbaum-Tarski algebra, $[\![-]\!]_{\mathbb{L}}$, and use it to prove algebraic completeness.

Theorem B.3.2 (Algebraic Completeness)**.** *If* $[\![P]\!]_{\mathbb{A}} \leq [\![Q]\!]_{\mathbb{A}}$ *for all algebraic interpretations* $[\![-]\!]_{\mathbb{A}}$*, then* $P \vdash Q$*.*

The proof is identical to the proof for theorem 2.2.6.

B.3.2 Soundness of M-BI formulas

M-BI formulas are interpreted on M-BI frames. We define a structure called *complex algebra* on M-BI frames and show that the complex algebra of every M-

BI frame is an M-BI algebra.

Definition B.3.3 (Complex Algebra)**.** If $\mathcal{X}$ is an M-BI frame, then the *complex algebra* of $\mathcal{X}$, written $\mathsf{Com}(\mathcal{X})$ is the structure $(\mathcal{P}_{\sqsubseteq}(X), \cap, \cup, \rightarrow_{\mathcal{X}}, X, \emptyset, *_{m\in M}, -\!\!*_{m\in M}, E_{m\in M})$ where

$$\mathcal{P}_{\sqsubseteq}(X) = \{A \subseteq X \mid a \in A \wedge a \sqsubseteq b \rightarrow b \in A\}$$

$$A \rightarrow_{\mathcal{X}} B = \{a \mid \forall b.\, a \sqsubseteq b \wedge b \in A \rightarrow b \in B\}$$

$$A *_m B = \{x \mid \exists w, y, z.\, w \sqsubseteq x \wedge w \in y \oplus_m z \wedge y \in A \wedge z \in B\}$$

$$A -\!\!*_m B = \{x \mid \forall w, y, z.\, (x \sqsubseteq w \wedge z \in w \oplus_m y \wedge y \in A) \rightarrow z \in B\}$$

Lemma B.3.3. *If $\mathcal{X}$ is an M-BI frame, then $\mathsf{Com}(\mathcal{X})$ is an M-BI algebra.*

Proof. Each $(X, \sqsubseteq, \oplus_m, E_m)$ is a BI frame. Lemma 2.2.7 shows that the complex of a BI frame is a BI algebra. Thus the only thing to check is that the ordering on $*$ respects the ordering on M. Let $m_1 \leq m_2$. We must show that $A *_{m_1} B \subseteq A *_{m_2} B$. Let $x \in A *_{m_1} B$. Then there exists w, y, z such that $w \sqsubseteq x$ and $w \in y \oplus_{m_1} z$, with $y \in A$ and $z \in B$. by Operation Inclusion property, we have that $w \in y \oplus_{m_2} z$, hence $x \in A *_{m_2} B$. □

Theorem B.3.4. *Let $\mathcal{X} = (X, \sqsubseteq, \circ_m, E_m)$ be a M-BI frame and let $\mathcal{V}_f : \mathcal{AP} \rightarrow \mathcal{P}(X)$ be a persistent valuation on $\mathcal{X}$. Define the algebraic assignment $\mathcal{V}_a : \mathcal{AP} \rightarrow \mathsf{Com}(\mathcal{X})$ by letting $\mathcal{V}_a(p) = \mathcal{V}_f(p)$ for all atomic proposition p. Define the algebraic interpretation $[\![-]\!]_a$ by taking the homomorphic extension of $\mathcal{V}_a$ Then we have: $x \models_{\mathcal{V}_f} P$ if and only if $x \in [\![P]\!]_a$.*

Proof. The proof is almost identical to the proof for theorem 2.2.8. We show that by induction on the syntax of the formula P. M-BI formula only differs from BI formula by having indexed version of the $*, I, -\!\!*$, so the only difference in the

proof is that: in the induction proof for formula $*_m, I_m, -\!\!*_m$, we use the indexed version of the operations in the complex algebra. □

Theorem B.3.5 (Soundness of M-BI)**.** *In M-BI logic, if $P \vdash Q$ is derivable, then $P \models Q$.*

The proof is identical to the proof of theorem 2.2.9.

B.3.3 Completeness of M-BI formulas

We reverse the direction now; we define a *prime filter frame* for every M-BI algebra and show that a prime filter frame of any M-BI algebra is an M-BI frame.

Definition B.3.4 (Prime Filter Frame)**.** If $\mathcal{A}$ is an M-BI algebra, then the prime filter M-frame of $\mathcal{A}$ is defined as $\mathsf{Prf}(\mathcal{A}) = (\mathsf{Prf}(A), \subseteq, \oplus_{m\in M}, E_{m\in M})$ where

$$F_1 \oplus_m F_2 = \{F \in \mathsf{Prf}(A) \mid \forall a_1 \in F_1.\, \forall a_2 \in F_2.\, a_1 *_m a_2 \in F\}$$
$$E_m = \{F \in \mathsf{Prf}(A) \mid \top^*_m \in F\}$$

Lemma B.3.6. *If $\mathcal{A}$ is an M-BI algebra, then $\mathsf{Prf}(\mathcal{A})$ is an M-BI frame.*

Proof. Lemma 2.2.10 shows that for each $m \in M$, $(\mathsf{Prf}(A), \subseteq, \oplus_m, E_m)$ is a BI frame. Therefore, we only need to check the Operation Inclusion property. Let $m_1 \leq m_2$ and let $F, G, H \in \mathsf{Prf}(A)$ with $F \in G \oplus_{m_1} H$. Let $a \in G$ and $b \in H$. Then $a *_{m_1} b \in F$. Since $a *_{m_1} b \leq a *_{m_2} b$, and filters are upward-closed, $a *_{m_2} b \in F$, hence $F \in G \oplus_{m_2} H$. □

Theorem B.3.7. *Let $\mathcal{A} = (A, \dots)$ be a M-BI algebra and let $[\![-]\!] : \mathrm{Form}_{\mathrm{BI}} \to A$ be an algebraic interpretation that homomorphically extends the assignment $\mathcal{V}_a : \mathcal{AP} \to A$.*

Define the persistent valuation $\mathcal{V}_f : \mathcal{AP} \to \mathcal{P}(\mathsf{Prf}(A))$ *on the prime filter frame* $\mathsf{Prf}(\mathcal{A})$ *by:*

$$\mathcal{V}_f(p) = \{F \in \mathsf{Prf}(A) \mid \mathcal{V}_a(p) \in F\}$$

Then for $F \in \mathsf{Prf}(A)$*, we have* $F \models_{\mathcal{V}_f} P$ *if and only if* $[\![P]\!] \in F$ *.*

The proof is almost identical to the proof of theorem 2.2.11. Then, we can prove the completeness of M-BI using the same argument as for theorem 2.2.12.

Theorem B.3.8 (Completeness of M-BI)**.** *In M-BI logic, if $P \models Q$, then $P \vdash Q$ is derivable.*

B.4 A M-BI Model for Independence and Negative Association

B.4.1 Independence Implies PNA

The proof that independence implies PNA will use the following lemma.

Lemma B.4.1. *In a distribution μ, if μ satisfies $\{S_1, S_2\}$-PNA, μ satisfies $\{T_1, T_2\}$-PNA, and $S_1 \cup S_2$ is independent from $T_1 \cup T_2$ in μ then μ is $\{S_1 \cup T_1, S_2 \cup T_2\}$-PNA.*

Proof. By the definition of PNA and independence, S_1, S_2 are disjoint, T_1, T_2 are disjoint, and $S_1 \cup T_1, S_2 \cup T_2$ are disjoint. For any monotonically decreasing/in-

creasing functions $f : \mathbf{Mem}[S_1 \cup T_1] \to \mathbb{R}^+, g : \mathbf{Mem}[S_2 \cup T_2] \to \mathbb{R}^+$,

$$\begin{aligned}
&\mathbb{E}_{m\sim\mu}[f(\pi_{S_1\cup T_1} m) \cdot g(\pi_{S_2\cup T_2} m)] \\
&= \mathbb{E}_{s\sim\pi_{S_1\cup S_2}\mu}\mathbb{E}_{t\sim\pi_{T_1\cup T_2}\mu}[f(\pi_{S_1} s \bowtie \pi_{T_1} t) \cdot g(\pi_{S_2} s \bowtie \pi_{T_2} t)] \\
&\qquad\qquad\qquad\qquad \text{(By independence of } S_1 \cup S_2 \text{ and } T_1 \cup T_2\text{)} \\
&\leq \mathbb{E}_{s\sim\pi_{S_1\cup S_2}\mu}\Big(\mathbb{E}_{t_1\sim\pi_{T_1}\mu}[f(\pi_{S_1} s, t_1)] \cdot \mathbb{E}_{t_2\sim\pi_{T_2}\mu}[g(\pi_{S_2} s, t_2)]\Big) && (\diamondsuit) \\
&\leq \mathbb{E}_{s_1\sim\pi_{S_1}\mu}\mathbb{E}_{t_1\sim\pi_{T_1}\mu}[f(s_1, t_1)] \cdot \mathbb{E}_{s_2\sim\pi_{S_2}\mu}\mathbb{E}_{t_2\sim\pi_{T_2}\mu}[g(s_2, t_2)] && (\heartsuit) \\
&\leq \mathbb{E}_{m\sim\mu}[f(\pi_{S_1\cup T_1} m)] \cdot \mathbb{E}_{m\sim\mu}[g(\pi_{S_2\cup T_2} m)] && (\clubsuit)
\end{aligned}$$

where the step $\diamondsuit$ is because $\pi_{T_1\cup T_2}\mu$ is T_1, T_2-PNA and $f(\pi_{S_1} s, t_1), g(\pi_{S_2} s, t_2)$ are both monotonically decreasing/increasing in T_1, T_2; the step $\heartsuit$ is because $\pi_{S_1\cup S_2}\mu$ is S_1, S_2-PNA and that $\mathbb{E}_{t_1\sim\pi_{T_1}\mu}[f(\pi_{S_1} s, t_1)]$, and $\mathbb{E}_{t_2\sim\pi_{T_2}\mu}[g(\pi_{S_2} s, t_2)]$ are both monotonically decreasing/increasing in S_1, S_2; and the step $\clubsuit$ is by independence of S_1 and T_1 and the independence of S_2 and T_2 in μ. □

Theorem 3.4.5 (Independence implies PNA)**.** *Let $S, T \subseteq$ **Var** be two disjoint sets of variables. Suppose $\mu_1 \in \mathcal{D}(\mathbf{Mem}[S])$, $\mu_2 \in \mathcal{D}(\mathbf{Mem}[T])$. If μ_1 satisfies $\mathcal{S}$-PNA and μ_2 satisfies $\mathcal{T}$-PNA, then any $\mu \in \mu_S \otimes_{\mathbb{D}} \mu_T$ satisfies $(\mathcal{S} \cup \mathcal{T})$-PNA.*

Proof. Fix $\mathcal{S}$ and $\mathcal{T}$. Say $\mathcal{S} = \{S_1, \ldots, S_p\}$ and $\mathcal{T} = \{T_1, \ldots, T_q\}$. For any $\mathcal{R}$ coarsening $\mathcal{S} \cup \mathcal{T}$, indexing $\mathcal{S} \cup \mathcal{T}$ as $\{U_1, \ldots, U_{p+q}\}$, indexing $\mathcal{R}$ as $\{R_1, \ldots, R_n\}$, we have:

$$\mathcal{R} = \{\cup\{U_j \mid j \in f(i)\} \mid i \in [n]\}.$$

Then, given a family of monotonically increasing/decreasing functions g_i : $R_i \to \mathbb{R}^+$

$$\mathbb{E}_{m\sim\mu}\left[\prod_{R_i\in\mathcal{R}} g_i(\pi_{R_i} m)\right] = \mathbb{E}_{m\sim\mu}\left[\prod_{i\in[n]} g_i(\pi_{\cup\{U_j | j\in f(i)\}} m)\right].$$

For each i, $\cup\{U_j \mid j \in f(i)\}$ can be divided into the part in S and the part in T. We refer to them as S'_i and T'_i. (Some of S'_i and T'_i may be empty). Thus, for each i,

$$g_i(\pi_{\cup\{U_j|j\in f(i)\}}m) = g_i(\pi_{S'_i\cup T'_i}m).$$

By Lemma B.1.3, $\mathcal{S}' = \{S'_1, \ldots, S'_n\}$ coarsens $\mathcal{S}$, and $\mathcal{T}' = \{T'_1, \ldots, T'_n\}$ coarsens $\mathcal{T}$. So μ is $\mathcal{S}'$-PNA and $\mathcal{T}'$-PNA.

We prove by induction on $k \in [n]$ that

$$\mathbb{E}_{m\sim\mu}\left[\prod_{i\in[k]} g_i(\pi_{S'_i\cup T'_i}m)\right] \le \prod_{i\in[k]} \mathbb{E}_{m\sim\mu}\left[g_i(\pi_{S'_i\cup T'_i}m)\right].$$

Base case When $k = 1$, trivial.

Inductive case For $k < n$, assume

$$\mathbb{E}_{m\sim\mu}\left[\prod_{i\in[k]} g_i(\pi_{S'_i\cup T'_i}m)\right] \le \prod_{i\in[k]} \mathbb{E}_{m\sim\mu}\left[g_i(\pi_{S'_i\cup T'_i}m)\right].$$

Note that μ is $\mathcal{S}'$-PNA implies that μ is $\{\cup_{i\in[k]}(S'_i), S'_{k+1}\}$-PNA, and μ is $\mathcal{T}'$-PNA implies that $\{\cup_{i\in[k]}(T'_i), T'_{k+1}\}$-NA. Thus, by Lemma B.4.1, μ is also $\{\{\cup_{i\in[k]}(S'_i) \cup \{\cup_{i\in[k]}(T'_i), S'_{k+1} \cup T'_{k+1}\}$-NA. Also, since all g_i is monotonically increasing (decreasing) and non-negative, $m \mapsto \prod_{i\in[k]} g_i(m)$ is also a monotonically increasing (decreasing) function from $\cup_{i\in[k]}S'_i \cup \cup_{i\in[k]}T'_i$ to $\mathbb{R}^+$. Therefore,

$$\begin{aligned}\mathbb{E}_{m\sim\mu}\left[\prod_{i\in[k+1]} g_i(\pi_{S'_i\cup T'_i}m)\right] &\le \mathbb{E}_{m\sim\mu}\left[\prod_{i\in[k]} g_i(\pi_{S'_i\cup T'_i}m)\right] \cdot \mathbb{E}_{m\sim\mu}\left[g_{k+1}(\pi_{S'_{k+1}\cup T'_{k+1}}m)\right] \\ &\le \prod_{i\in[k+1]} \mathbb{E}_{m\sim\mu}\left[g_i(\pi_{S'_i\cup T'_i}m)\right],\end{aligned}$$

where the second inequality follows from the inductive hypothesis.

Thus, the desired inequality holds for any $\mathcal{R}$ coarsening $\mathcal{S} \cup \mathcal{T}$ and any family of monotonically increasing (decreasing) functions on $\mathcal{R}$. Thus, μ is $\mathcal{S} \cup \mathcal{T}$-PNA.

□

B.4.2 Axioms of Negative Association

Lemma 3.5.5 (N-NARY MONOTONE MAP)**.** *Let x, $x_{\gamma,\alpha}$ and y_γ be program variables. Let K_γ be natural numbers. The following is valid in $(\mathcal{X}_{NA}, \mathcal{V}^*)$.*

$$\models \circledast_{\gamma=0}^{N} \left(\bigwedge_{\alpha=0}^{K_\gamma} \mathsf{Own}(x_{\gamma,\alpha}) \right) \wedge \bigwedge_{\gamma=0}^{N} \left[y_\gamma = f_\gamma\left(x_{\gamma,0}, \ldots, x_{\gamma,K_\gamma}\right) \right] \rightarrow \circledast_{\gamma=0}^{N} \mathsf{Own}(y_\gamma)$$

when $f_1, \ldots, f_N$ all monotone or all antitone (Mono-Map)

Proof. Abbreviate the partition of variables $\left\{ \bigcup_{\alpha=0}^{K_\gamma} \{x_{\gamma,\alpha}\} \mid L \leq \gamma \leq M \right\}$ as $X[L : M]$. Intuitively, we group all the $x_{\gamma,\alpha}$ with the same γ as a block in the partition, and different blocks in the partition are separated by the separating conjunction $\circledast$.

For any $\mu \models \circledast_{\gamma=0}^{N} \left(\bigwedge_{\alpha=0}^{K_\gamma} \mathsf{Own}(x_{\gamma,\alpha}) \right)$, we use induction and definition unfolding to show that μ satisfies X_N-PNA. We choose the inductive hypothesis $P(M)$ to be: $\mu \models \circledast_{\gamma=0}^{M} \left(\bigwedge_{\alpha=0}^{K_\gamma} \mathsf{Own}(x_{\gamma,\alpha}) \right)$ implies that μ is $X[0 : M]$-PNA.

Base case: $X[: 0] = \left\{ \bigcup_{\alpha=0}^{K_\gamma} \{x_{\gamma,\alpha}\} \right\}$ is partition that contains a single block. Thus, μ is trivially $X[0 : 0]$-PNA.

Inductive case: For any $0 < M \leq M$, by satisfaction rules, $\mu \models \circledast_{\gamma=0}^{M} \left(\bigwedge_{\alpha=0}^{K_\gamma} \mathsf{Own}(x_{\gamma,\alpha}) \right)$ implies there exists μ', μ_1, μ_2 such that $\mu \sqsupseteq \mu' \in$

$\mu_1 \oplus \mu_2$,

$$\mu_1 \models \circledast_{\gamma=0}^{M-1} \left(\bigwedge_{\alpha=0}^{K_\gamma} \mathsf{Own}(x_{\gamma,\alpha})\right) \qquad \text{and} \qquad \mu_2 \models \bigwedge_{\alpha=0}^{K_M} \mathsf{Own}(x_{M,\alpha})$$

By inductive hypothesis, μ_1 satisfies $X[0:M-1]$-PNA. And $X[M:M]$ is a partition that contains a single block, so trivially, μ_2 satisfies $X[M:M]$-PNA. Therefore, $\mu' \in \mu_1 \oplus \mu_2$ implies that μ' is $X[0:M]$-PNA

Therefore, we can conclude μ is $X[0:N]$-PNA from $\mu \models \circledast_{\gamma=0}^{N}\left(\bigwedge_{\alpha=0}^{K_\gamma} \mathsf{Own}(x_{\gamma,\alpha})\right)$.

If additionally $\mu \models \bigwedge_{\gamma=0}^{N} y_\gamma = f_\gamma(x_{\gamma,1},\ldots,x_{\gamma,K_\gamma})$ and f_γ are all monotone or antitone, then we can show that μ is $\{\{y_\gamma\} \mid 1 \le \gamma \le N\}$-PNA. For any family of non-negative monotone functions g_γ, note that the composed function $g_\gamma \circ f_\gamma$ are either all monotone or all antitone. Thus,

$$\begin{aligned}
&\mathbb{E}_{m\sim\mu}\left[\prod_{1\le\gamma\le N} g_\gamma(y_\gamma)\right] \\
&= \mathbb{E}_{m\sim\mu}\left[\prod_{1\le\gamma\le N} g_\gamma(f_\gamma(x_{\gamma,1},\ldots,x_{\gamma,K_\gamma}))\right] \\
&= \mathbb{E}_{m\sim\mu}\left[\prod_{1\le\gamma\le N} (g_\gamma \circ f_\gamma)(x_{\gamma,1},\ldots,x_{\gamma,K_\gamma})\right] \\
&\le \prod_{1\le\gamma\le N} \mathbb{E}_{m\sim\mu}\left[(g_\gamma \circ f_\gamma)(x_{\gamma,1},\ldots,x_{\gamma,K_\gamma})\right] \qquad \text{(Because } \mu \text{ is } X[0,N]\text{-PNA)} \\
&= \prod_{1\le\gamma\le N} \mathbb{E}_{m\sim\mu}\left[g_\gamma(y_\gamma)\right]. \qquad \text{(B.7)}
\end{aligned}$$

That is, μ is $\{\{y_\gamma\} \mid 1 \le \gamma \le N\}$-PNA. And by Theorem 3.3.1 this implies that $\{\{y_\gamma\} \mid 1 \le \gamma \le N\}$ satisfies NA in μ. Then, by Theorem 3.3.5, $(\sigma,\mu) \models \circledast_{\gamma=1}^{N} \mathsf{Own}(y_\gamma)$. □

B.4.3 The Restriction Property of M-BI Formulas

For the counterexample of the restriction property, we prove a lemma.

Lemma B.4.2. *Let μ be the uniform distribution over one hot vectors on A, B. Then, $\mu \models (\mathbf{Unif}_{\{0,1\}}\langle C\rangle) \circledast (\mathsf{Own}(B) * \mathsf{Own}(C))$.*

Proof. Fix any μ_C such that $\mu_C \models \mathbf{Unif}_{\{0,1\}}\langle C\rangle$, which implies that $\pi_C\mu_C(0) = 0.5$ and $\pi_C\mu_C(1) = 0.5$. Fix $\mu_e \in \mu \oplus \mu_C$.

Since $B \in \mathbf{dom}(\mu)$, μ is trivially $\{\{B\}\}$-PNA. Similarly, μ_C is trivially $\{\{C\}\}$-PNA. Thus, $\mu_e \in \mu \oplus \mu_C$ must be $\{\{B\},\{C\}\}$-PNA. Then for any two both monotone or antitone functions $f : \mathbf{Mem}[B] \to \mathbb{R}^+, g : \mathbf{Mem}[C] \to \mathbb{R}^+$,

$$\mathbb{E}_{m\sim\mu_e}[f(\pi_B m)\cdot g(\pi_C m)] \le \mathbb{E}_{m\sim\mu_e}[f(\pi_B m)]\cdot\mathbb{E}_{m\sim\mu_e}[g(\pi_C m)].$$

Similarly, $\mu_e \in \mu \oplus \mu_C$ must be $\{\{A\},\{C\}\}$-PNA, and thus, for any two both monotone or antitone functions $f : \mathbf{Mem}[A] \to \mathbb{R}^+, g : \mathbf{Mem}[C] \to \mathbb{R}^+$,

$$\mathbb{E}_{m\sim\mu_e}[f(\pi_A m)\cdot g(\pi_C m)] \le \mathbb{E}_{m\sim\mu_e}[f(\pi_A m)]\cdot\mathbb{E}_{m\sim\mu_e}[g(pi_C m)]. \tag{B.8}$$

Next, we want to prove that $\mu_e \models \mathsf{Own}(B) * \mathsf{Own}(C)$. We prove by contradiction. Suppose variables B and C are not independent in μ_e, then by Lemma B.1.1 that says NA definition with equality instead of inequality asserts independence, there must exists some both monotone or both antitone functions $f : \mathbf{Mem}[B] \to \mathbb{R}^+, g : \mathbf{Mem}[C] \to \mathbb{R}^+$ such that

$$\mathbb{E}_{m\sim\mu_e}[f(\pi_B m)\cdot g(\pi_C m)] < \mathbb{E}_{m\sim\mu_e}[f(\pi_B m)]\cdot\mathbb{E}_{m\sim\mu_e}[g(\pi_C m)]$$

Since $\mu_e \in \mu \oplus \mu_C$, we have $\mu_e \sqsupseteq \mu$, and μ being a uniform distribution over one-hot vectors on A, B indicates that for any m in the support of μ_e, $A = 1$ iff

$B = 0$, and $A = 0$ iff $B = 1$. Therefore,

$$\begin{aligned}\mathbb{E}_{m\sim\mu_e}[f(-\pi_A m)\cdot(-g(\pi_C m))] &= \mathbb{E}_{m\sim\mu_e}[f(\pi_B m)\cdot(-g(\pi_C m))]\\ &= -\mathbb{E}_{m\sim\mu_e}[f(\pi_B m)\cdot g(\pi_C m)]\\ &> -\mathbb{E}_{m\sim\mu_e}[f(-\pi_A m)]\cdot\mathbb{E}_{m\sim\mu_e}[g(\pi_C m)]\\ &= \mathbb{E}_{m\sim\mu_e}[f(-\pi_A m)]\cdot\mathbb{E}_{m\sim\mu_e}[-g(\pi_C m)]\end{aligned}$$

where $x \mapsto f(-x)$ and $x \to -g(x)$ are two both monotone or both antitone functions because f, g are so. Thus, this inequality contradicts Equation (B.8).

Therefore, B and C must be independent in μ_e. Hence, $\mu_e \models \mathsf{Own}(B) * \mathsf{Own}(C)$, and $\mu \models (\mathbf{Unif}_{\{0,1\}}\langle C\rangle) \mathrel{-\!\!\circledast} (\mathsf{Own}(B) * \mathsf{Own}(C))$. □

Theorem 3.5.2. *There exists $\mu \in \mathcal{D}(\mathbf{Mem}[S])$ and formula φ such that $\mu \models \varphi$ but $\pi_{FV(\varphi)} \not\models \varphi$.*

Proof. Let A, B, C be three variables in **Var**. Let $\varphi = (\mathbf{Unif}_{\{0,1\}}\langle C\rangle) \mathrel{-\!\!\circledast} (\mathsf{Own}(B) * \mathsf{Own}(C))$. Let μ be the uniform distribution over one hot vectors on A, B. Then, we claim $\mu \models \varphi$ but $\pi_{\{B,C\}}\mu \not\models \varphi$. For $\mu \models \varphi$, it suffices to show that for any μ_C where C's value is the uniform distribution on $\{0, 1\}$, for any $\mu' \sqsupseteq \mu$, and $\mu_e \in \mu' \oplus \mu_C$, B and C are independent in μ_e according to Lemma B.4.2.

To show $\pi_{\{B,C\}}\mu \not\models \varphi$, we first note that $\pi_{\{B,C\}}\mu = \pi_{\{B\}}\mu$ is a uniform distribution of 0 and 1 on B. Let $\mu'_C \in \mathcal{D}(\mathbf{Mem}[\{C\}])$ be the uniform distribution on $\{0, 1\}$, $\mu' \in \mathcal{D}(\mathbf{Mem}[\{B, C\}])$ be the uniform distribution over one-hot vectors on B, C. Clearly, B, C are not independent in μ', so $\mu' \not\models \mathsf{Own}(B) * \mathsf{Own}(C)$. Also, μ' is in $\pi_{\{B,C\}}\mu \oplus \mu'_C$. So $\pi_{B,C}\mu \not\models \mathbf{Unif}_{\{0,1\}}\langle C\rangle \mathrel{-\!\!\circledast} (\mathsf{Own}(B) * \mathsf{Own}(C))$. □

Theorem 3.5.1 (Restriction)**.** *For any distribution $\mu \in X_{\mathbb{D}}$, for any φ be an MBI_+*

formula interpreted on $(\mathcal{X}_{NA}, \mathcal{V}^*)$*, and any valuation* $\mathcal{V}$*,*

$$\mu \models_{\mathcal{V}} \varphi \Leftrightarrow \pi_{FV(\varphi)}\mu \models_{\mathcal{V}} \varphi.$$

Proof. We prove it by induction on the syntax of formula. Most cases are the same as in lemma 2.3.7. So we only show the case for the additional case $P \circledast Q$.

$\varphi = P \circledast Q$**:** Assuming $\mu \models P \circledast Q$, then there exists μ', μ_1, σ_2 such that $\mu \sqsupseteq \mu' \in \mu_1 \oplus \mu_2$, $\mu_1 \models P$, and $\mu_2 \models Q$. By the definition of the pre-order and $\oplus$, it must $\mu \sqsupseteq \mu' \in \mu_1 \oplus \mu_2$.

By inductive hypothesis, $\pi_{\mathrm{FV}(\varphi)}\mu_1 \models P$ and $\pi_{\mathrm{FV}(\varphi)}\mu_2 \models Q$. Also, by eq. (Down-Closed), there exists $\mu'' \sqsubseteq \mu'$ and $\mu'' \pi_{\mathrm{FV}(\varphi)}\mu_1 \oplus \pi_{\mathrm{FV}(\varphi)}\mu_2$. So $\mu'' \models P \circledast Q$ and by persistence, $\mu \models P \circledast Q$.

□

APPENDIX C

DIBI: A BUNCHED LOGIC FOR CONDITIONAL INDEPENDENCE

C.1 A Probabilistic Model of DIBI

Remark. In the following, we sometimes abbreviate $\mathbf{dom}(f_i)$ as D_i and $\mathbf{range}(f_i)$ as R_i.

C.1.1 Well-definedness of the Structure

To facilitate proving that $(\mathcal{X}_{CI}, \sqsubseteq, \widehat{\oplus}, \widehat{\odot}, E)$ is a DIBI frame, we first show some properties of the binary operations and the order. First, we prove that $\mathcal{X}_{CI}$ is closed under $\oplus$ and $\odot$.

Lemma C.1.1. *$\mathcal{X}_{CI}$ is closed under $\oplus$ and $\odot$.*

Proof. For any $f_1, f_2 \in \mathcal{X}_{CI}$, we need to show that

- If $f_1 \oplus f_2$ is defined, then $f_1 \oplus f_2 \in \mathcal{X}_{CI}$.

  Recall that $f_1 \oplus f_2$ is defined if and only if $R_1 \cap R_2 = D_1 \cap D_2$, which implies that $(R_1 \cup R_2) \setminus (D_1 \cup D_1) = (R_1 \setminus D_1) \uplus (R_2 \setminus D_2)$.

  State $f_1 \oplus f_2$ preserves the input because for any $d \in \mathbf{Mem}[D_1 \cup D_2]$, we

can obtain the following (we will refer to this as $\star$):

$$
\begin{aligned}
&(\pi_{D_1 \cup D_2}(f_1 \oplus f_2))(d)(d) \\
&= \sum_{x \in \mathbf{Mem}[(R_1 \cup R_2) \setminus (D_1 \cup D_2)]} (f_1 \oplus f_2)(d)(d \bowtie x) \\
&= \sum_{x_1 \in \mathbf{Mem}[R_1 \setminus D_1], x_2 \in \mathbf{Mem}[R_2 \setminus D_2]} f_1(d^{D_1})(d^{D_1} \bowtie x_1) \cdot f_2(d^{D_2})(d^{D_2} \bowtie x_2) \\
&= \left(\sum_{x_1 \in \mathbf{Mem}[R_1 \setminus D_1]} f_1(d^{D_1})(d^{D_1} \bowtie x_1) \right) \cdot \left(\sum_{x_2 \in \mathbf{Mem}[R_2 \setminus D_2]} f_2(d^{D_2})(d^{D_2} \bowtie x_2) \right) \\
&= 1 \cdot 1 = 1 \qquad \text{(Using } f_1, f_2 \in \mathcal{X}_{CI}\text{)}
\end{aligned}
$$

Then, for any input $d \in \mathbf{Mem}[D_1 \cup D_2]$, $(f_1 \oplus f_2)(d)$ is a distribution since:

$$
\begin{aligned}
&\sum_{m \in \mathbf{Mem}[R_1 \cup R_2]} (f_1 \oplus f_2)(d)(m) \\
&= \sum_{m \in \mathbf{Mem}[R_1 \cup R_2]} f_1(d^{D_1})(m^{R_1}) \cdot f_2(d^{D_2})(m^{R_2}) \\
&\overset{\ddagger}{=} \sum_{x_1 \in \mathbf{Mem}[R_1 \setminus D_1], x_2 \in \mathbf{Mem}[R_2 \setminus D_2]} f_1(d^{D_1})(d^{D_1} \bowtie x_1) \cdot f_2(d^{D_2})(d^{D_2} \bowtie x_2) \\
&= 1 \qquad \text{(Using the last two steps of } (\star)\text{)}
\end{aligned}
$$

Step $\ddagger$ follows f_1 and f_2 being input-preserving, which means that the term $f_i(\pi_d D_i)(\pi_m R_i)$ is 0 when $d^{D_i} \neq m^{D_i}$.

Thus, $f_1 \oplus f_2$ is a kernel in $\mathcal{X}_{CI}$.

- If $f_1 \odot f_2$ is defined, then $f_1 \odot f_2 \in \mathcal{X}_{CI}$.

  Recall that $f_1 \odot f_2 : \mathbf{Mem}[D_1] \to \mathcal{D}(\mathbf{Mem}[R_2])$ is defined iff $R_1 = D_2$. The composition $f_1 \odot f_2$ preserves the input because for any $d \in \mathbf{Mem}[D_1]$, we

can obtain (♠):

$$
\begin{aligned}
&(\pi_{D_1} f_1 \odot f_2)(d)(d) \\
&= \sum_{x \in \mathbf{Mem}[R_2 \setminus D_1]} (f_1 \odot f_2)(d)(d \bowtie x) \\
&= \sum_{x \in \mathbf{Mem}[R_2 \setminus D_1]} f_1(d)(d \bowtie x^{R_1 \setminus D_1}) \cdot f_2(d \bowtie x^{R_1 \setminus D_1})(d \bowtie x) \\
&= \sum_{x_1 \in \mathbf{Mem}[R_1 \setminus D_1]} f_1(d)(d \bowtie x_1) \cdot \left(\sum_{x_2 \in \mathbf{Mem}[R_2 \setminus R_1]} f_2(d \bowtie x_1)(d \bowtie x_1 \bowtie x_2) \right) \\
&= \sum_{x_1 \in \mathbf{Mem}[R_1 \setminus D_1]} (f_1(d)(d \bowtie x_1) \cdot 1) && \text{(Using } f_2 \in \mathcal{X}_{CI}\text{)} \\
&= 1
\end{aligned}
$$

Then, for any $d \in D_1$, $(f_1 \odot f_2)(d)$ is a distribution as

$$
\begin{aligned}
&\sum_{m \in \mathbf{Mem}[R_2]} (f_1 \odot f_2)(d)(m) \\
&= \sum_{m \in \mathbf{Mem}[R_2]} f_1(d)(m^{R_1}) \cdot f_2(m^{R_1})(m) && \text{(Equation (4.2))} \\
&\overset{\heartsuit}{=} \sum_{x \in \mathbf{Mem}[R_2 \setminus D_1]} f_1(d)(d \bowtie x^{R_1 \setminus D_1}) \cdot f_2(d \bowtie x^{R_1 \setminus D_1})(d \bowtie x) \\
&= 1 && \text{(Using the last three steps of (♠))}
\end{aligned}
$$

Step $\heartsuit$ follows from f_1, f_2 being input-preserving, so the f_i terms are 0 when $d^{D_i} \neq m^{D_i}$.

Thus $f_1 \odot f_2$ is a kernel in $\mathcal{X}_{CI}$. □

Lemma C.1.2 (Reflexivity and transitivity of order)**.** *The order $\sqsubseteq$ defined in $\mathcal{X}_{CI}$ is transitive and reflexive.*

Proof. Let $x \colon \mathbf{Mem}[A] \to \mathcal{D}(\mathbf{Mem}[X]) \in M$, $S = \emptyset$, $v = \mathsf{unit}_X$. Then

$$\begin{aligned}(x \oplus \mathsf{unit}_S) \odot v &= (x \oplus \mathsf{unit}_\emptyset) \odot \mathsf{unit}_X \\ &= x \odot \mathsf{unit}_X && \text{(By lemma C.1.7)} \\ &= x\end{aligned}$$

Thus, we have $x \sqsubseteq x$, and the order is reflexive.

For any $x, y, z \in M$, if $x \sqsubseteq y$ and $y \sqsubseteq z$, then by definition of $\sqsubseteq$, there exist S_1 and v_1 such that $y = (x \oplus \mathsf{unit}_{S_1}) \odot v_1$, and there exist S_2 and v_2 such that $z = (y \oplus \mathsf{unit}_{S_2}) \odot v_2$.

We can now calculate:

$$\begin{aligned}z &= (y \oplus \mathsf{unit}_{S_2}) \odot v_2 \\ &= (((x \oplus \mathsf{unit}_{S_1}) \odot v_1) \oplus \mathsf{unit}_{S_2}) \odot v_2 \\ &= (((x \oplus \mathsf{unit}_{S_1}) \odot v_1) \oplus (\mathsf{unit}_{S_2} \odot \mathsf{unit}_{S_2})) \odot v_2 \\ &= (x \oplus \mathsf{unit}_{S_1} \oplus \mathsf{unit}_{S_2}) \odot (v_1 \oplus \mathsf{unit}_{S_2}) \odot v_2 && \text{(By C.1.8 and lemma C.1.9)} \\ &= (x \oplus \mathsf{unit}_{S_1 \cup S_2}) \odot ((v_1 \oplus \mathsf{unit}_{S_2}) \odot v_2)\end{aligned}$$

$\mathcal{X}_{CI}$ is closed under $\oplus, \odot$, so $(v_1 \oplus \mathsf{unit}_{S_2}) \odot v_2 \in \mathcal{X}_{CI}$. Thus,

$$z = (x \oplus \mathsf{unit}_{S_1 \cup S_2}) \odot (v_1 \oplus \mathsf{unit}_{S_2}) \odot v_2$$

showing that $x \sqsubseteq z$. So the order is transitive. □

Next we prove that the parallel composition $\oplus$ is associative, commutative and identify its identity.

C.1.2 Associativity of Parallel Composition

Lemma C.1.3 ($\oplus$ - Associativity)**.** *We show that when* $(f \oplus g) \oplus h$ *and* $f \oplus (g \oplus h)$ *are defined,* $(f \oplus g) \oplus h = f \oplus (g \oplus h)$.

Proof. Consider $f : \mathbf{Mem}[S] \to \mathcal{D}(\mathbf{Mem}[S \cup T])$, $g : \mathbf{Mem}[U] \to \mathcal{D}(\mathbf{Mem}[U \cup V])$, and $h : \mathbf{Mem}[W] \to \mathcal{D}(\mathbf{Mem}[W \cup X])$. For any $d \in \mathbf{Mem}[S \cup U \cup W]$, and $m \in \mathbf{Mem}[S \cup T \cup U \cup V \cup W \cup X]$,

$$\begin{aligned}((f \oplus g) \oplus h)(d)(m) &= \left(f(d^S)(m^{S\cup T}) \cdot g(d^U)(m^{U\cup V})\right) \cdot h(d^W)(m^{W\cup X}) \qquad \text{(def. } \oplus\text{)} \\ &= f(d^S)(m^{S\cup T}) \cdot \left(g(d^U)(m^{U\cup V}) \cdot h(d^W)(m^{W\cup X})\right) \\ &= (f \oplus (g \oplus h))(d)(m)\end{aligned}$$

□

Lemma C.1.4 (Standard associativity of $\oplus$)**.** *For any* $f_1, f_2, f_3 \in M$, $(f_1 \oplus f_2) \oplus f_3$ *is defined if and only if* $f_1 \oplus (f_2 \oplus f_3)$ *is defined and they are equal.*

Proof. To show that $(f_1 \oplus f_2) \oplus f_3$ is defined if and only if $f_1 \oplus (f_2 \oplus f_3)$ is defined, it suffices to show that

$$R_1 \cap R_2 = D_1 \cap D_2 \tag{C.1}$$

$$(R_1 \cup R_2) \cap R_3 = (D_1 \cup D_2) \cap D_3 \tag{C.2}$$

if and only if

$$R_2 \cap R_3 = D_2 \cap D_3 \tag{C.3}$$

$$R_1 \cap (R_2 \cup R_3) = D_1 \cap (D_2 \cup D_3) \tag{C.4}$$

We show that eq. (C.3) and eq. (C.4) follows from eq. (C.1) and eq. (C.2):

Recall that $D_1 \subseteq R_1$, $D_2 \subseteq R_2$, $D_3 \subseteq R_3$, so

- Equation (C.3) follows from $D_2 \cap D_3 \subseteq R_2 \cap R_3$ and $D_2 \cap D_3 \supseteq R_2 \cap R_3$, which holds because

$$\begin{aligned}
R_2 \cap R_3 &= R_2 \cap ((R_1 \cup R_2) \cap R_3) \\
&= R_2 \cap ((D_1 \cup D_2) \cap D_3) && \text{(By eq. (C.2))} \\
&= R_2 \cap ((D_1 \cap D_3) \cup (D_2 \cap D_3)) \\
&\subseteq ((R_2 \cap R_1) \cap D_3) \cup (D_2 \cap D_3) && \text{(By } D_1 \subseteq R_1\text{)} \\
&= ((D_2 \cap D_1) \cap D_3) \cup (D_2 \cap D_3) && \text{(By eq. (C.1))} \\
&\subseteq D_2 \cap D_3
\end{aligned}$$

- Equation (C.4) follows from $(D_1 \cup D_2) \cap D_3 \subseteq (R_1 \cup R_2) \cap R_3$ and $(D_1 \cup D_2) \cap D_3 \supseteq (R_1 \cup R_2) \cap R_3$, which holds because

$$\begin{aligned}
&R_1 \cap (R_2 \cup R_3) \\
&= (R_1 \cap R_2) \cup (R_1 \cap R_3) \\
&\subseteq (R_1 \cap R_2) \cup (R_1 \cap (R_1 \cup R_2) \cap R_3) \\
&= (D_1 \cap D_2) \cup (R_1 \cap (D_1 \cup D_2) \cap D_3) && \text{(By eq. (C.1) and eq. (C.2))} \\
&= (D_1 \cap D_2) \cup ((R_1 \cap D_1 \cap D_3) \cup (R_1 \cap D_2 \cap D_3)) \\
&\subseteq (D_1 \cap D_2) \cup ((D_1 \cap D_3) \cup (R_1 \cap R_2 \cap D_3)) && \text{(By } D_2 \subseteq R_2\text{)} \\
&\subseteq (D_1 \cap D_2) \cup ((D_1 \cap D_3) \cup (D_1 \cap D_2 \cap D_3)) && \text{(By eq. (C.1))} \\
&\subseteq (D_1 \cap D_2) \cup (D_1 \cap D_3) \\
&= D_1 \cap (D_2 \cup D_3)
\end{aligned}$$

We show that eq. (C.1) and eq. (C.2) follows from eq. (C.3) and eq. (C.4):

- Equation (C.1) follows from $D_1 \cap D_2 \subseteq R_1 \cap R_2$ and $D_1 \cap D_2 \supseteq R_1 \cap R_2$,

which holds because

$$\begin{aligned}
R_1 \cap R_2 &= (R_1 \cap (R_2 \cup R_3)) \cap R_2 \\
&= (D_1 \cap (D_2 \cup D_3)) \cap R_2 && \text{(By eq. (C.4))} \\
&= D_1 \cap ((D_2 \cap R_2) \cup (D_3 \cap R_2)) \\
&= D_1 \cap (D_2 \cup (D_3 \cap R_2)) \\
&\subseteq D_1 \cap (D_2 \cup (R_3 \cap R_2)) && \text{(By } D_3 \subseteq R_3\text{)} \\
&= D_1 \cap (D_2 \cup (D_3 \cap D_2)) && \text{(By eq. (C.3))} \\
&= D_1 \cap D_2
\end{aligned}$$

- Equation (C.2) follows from $(D_1 \cup D_2) \cap D_3 \subseteq (R_1 \cup R_2) \cap R_3$ and $(D_1 \cup D_2) \cap D_3 \supseteq (R_1 \cup R_2) \cap R_3$, which holds because

$$\begin{aligned}
&(R_1 \cup R_2) \cap R_3 \\
&= (R_1 \cap R_3) \cup (R_2 \cap R_3) \\
&= (R_1 \cap (R_2 \cup R_3) \cap R_3) \cup (R_2 \cap R_3) \\
&= (D_1 \cap (D_2 \cup D_3) \cap R_3) \cup (D_2 \cap D_3) && \text{(By eq. (C.4))} \\
&= (D_1 \cap ((D_2 \cap R_3) \cup (D_3 \cap R_3))) \cup (D_2 \cap D_3) \\
&\subseteq (D_1 \cap ((R_2 \cap R_3) \cup D_3)) \cup (D_2 \cap D_3) && \text{(By } D_2 \subseteq R_2, D_3 \subseteq R_3\text{)} \\
&= (D_1 \cap ((D_2 \cap D_3) \cup D_3)) \cup (D_2 \cap D_3) && \text{(By eq. (C.3))} \\
&= (D_1 \cap D_3) \cup (D_2 \cap D_3) \\
&= (D_1 \cup D_2) \cap D_3
\end{aligned}$$

Thus, eq. (C.1) and eq. (C.2) hold if and only if eq. (C.3) and eq. (C.4) hold. Therefore, $(f_1 \oplus f_2) \oplus f_3$ is defined if and only if $f_1 \oplus (f_2 \oplus f_3)$ is defined. By lemma C.1.3, they are equal when both defined. □

C.1.3 Commutativity of Parallel Composition

Lemma C.1.5 ($\oplus$ - Commutativity)**.** *When $f_1 \oplus f_2$ and $f_2 \oplus f_1$ are both defined, $f_1 \oplus f_2 = f_2 \oplus f_1$.*

Proof. For any $d \in \mathbf{Mem}[D_1 \cup D_2]$, $m \in \mathcal{D}(\mathbf{Mem}[R_1 \cup R_2])$ such that $d \bowtie m$ is defined,

$$\begin{aligned}(f_1 \oplus f_2)(d)(m) &= f_1(d^{D_1})(m^{R_1}) \cdot f_2(d^{D_2})(m^{R_2}) \\ &= f_2(d^{D_2})(m^{R_2}) \cdot f_1(d^{D_1})(m^{R_1}) \\ &= (f_2 \oplus f_1)(d)(m)\end{aligned}$$

Thus, $f_1 \oplus f_2 = f_2 \oplus f_1$. □

Lemma C.1.6 ($\oplus$ - Identity)**.** *For any $f : \mathbf{Mem}[A] \to \mathcal{D}(\mathbf{Mem}[A \cup X]) \in M$, and any $S \subseteq A$, we must show*

$$f \oplus \mathit{unit}_S = f$$

Proof. Since $S \subseteq A$, we have $\mathbf{dom}(f \oplus \mathsf{unit}_S) = A \cup S = A = \mathbf{dom}(f)$ and $\mathbf{range}(f \oplus \mathsf{unit}_S) = A \cup X \cup S = A \cup X = \mathbf{range}(f)$. For any $d \in \mathbf{Mem}[A]$, and any $r \in \mathbf{Mem}[A \cup X]$ such that $d \bowtie r$ is defined, we have

$$\begin{aligned}(f \oplus \mathsf{unit}_S)(d)(r) &= f(d)(r) \cdot \mathsf{unit}(d^S)(r^S) \\ &= f(d)(r) \cdot 1 \\ &= f(d)(r)\end{aligned}$$

If $d \bowtie r$ is not defined, then $(f \oplus \mathsf{unit}_S)(d)(r) = f(d)(r)$. Hence, $f \oplus \mathsf{unit}_S = f$. □

C.1.4 Other Properties Used in Proving Frame Conditions

Lemma C.1.7. *For any* $f : \mathbf{Mem}[A] \to \mathcal{D}(\mathbf{Mem}[A \cup X]) \in M$*, and any* $S \subseteq A$*, we have*

$$f \oplus \mathit{unit}_S = f$$

Proof. Since $S \subseteq A$, we have $\mathbf{dom}(f \oplus \mathsf{unit}_S) = A \cup S = A = \mathbf{dom}(f)$ and $\mathbf{range}(f \oplus \mathsf{unit}_S) = A \cup X \cup S = A \cup X = \mathbf{range}(f)$. For any $d \in \mathbf{Mem}[A]$, and any $r \in \mathbf{Mem}[A \cup X]$ such that $d \otimes r$ is defined, we have

$$\begin{aligned}(f \oplus \mathsf{unit}_S)(d)(r) &= f(d)(r) \cdot \mathsf{unit}(d^S)(r^S) \\ &= f(d)(r) \cdot 1 = f(d)(r)\end{aligned}$$

Hence, $f \oplus \mathsf{unit}_S = f$. □

Lemma C.1.8 (Reverse Exchange Equality)**.** *We show that when both* $(f_1 \oplus f_2) \odot (f_3 \oplus f_4)$ *and* $(f_1 \odot f_3) \oplus (f_2 \odot f_4)$ *are defined, it holds that*

$$(f_1 \oplus f_2) \odot (f_3 \oplus f_4) = (f_1 \odot f_3) \oplus (f_2 \odot f_4). \tag{C.5}$$

Proof. First, the well-definedness of $f_1 \circ f_3$ implies that $D_1 \subseteq R_1 = D_3 \subseteq R_3$, and the well-definedness of $f_2 \circ f_4$ implies that $D_2 \subseteq R_2 = D_4 \subseteq R_4$. Moreover, both terms are of type $\mathbf{Mem}[D_1 \cup D_2] \to \mathcal{D}(\mathbf{Mem}[R_3 \cup R_4])$, and, for any $d \in$

$\mathbf{Mem}[D_1 \cup D_2]$ and $m \in \mathbf{Mem}[R_3 \cup R_4]$, we have:

$$
\begin{aligned}
&\big((f_1 \oplus f_2) \odot (f_3 \oplus f_4)\big)(d)(m) \\
&= (f_1 \oplus f_2)(d)(m^{R_1 \cup R_2}) \cdot (f_3 \oplus f_4)(m^{D_3 \cup D_4})(m) && \text{(Equation (4.2))} \\
&= \big(f_1(d^{D_1})(m^{R_1}) \cdot f_2(d^{D_2})(m^{R_2})\big) \cdot \big(f_3(m^{D_3})(m^{R_3}) \cdot f_4(m^{D_4})(m^{R_4})\big) \\
&\big((f_1 \odot f_3) \oplus (f_2 \odot f_4)\big)(d)(m) \\
&= (f_1 \odot f_3)(d^{D_1})(m^{R_3}) \cdot (f_2 \odot f_4)(d^{D_2})(m^{R_3}) \\
&= \big(f_1(d^{D_1})(m^{R_1}) \cdot f_3(d^{D_3})(m^{R_3})\big) \cdot \big(f_2(d^{D_2})(m^{R_2}) \cdot f_4(d^{D_4})(m^{R_4})\big) \\
&= \big(f_1(d^{D_1})(m^{R_1}) \cdot f_2(d^{D_2})(m^{R_2})\big) \cdot \big(f_3(m^{D_3})(m^{R_3}) \cdot f_4(m^{D_4})(m^{R_4})\big)
\end{aligned}
$$

Thus, $(f_1 \odot f_3) \oplus (f_2 \odot f_4) = (f_1 \oplus f_2) \odot (f_3 \oplus f_4)$. □

Lemma C.1.9. *For any f_1, f_2, f_3, f_4 in $\mathcal{X}_{CI}$, $(f_1 \odot f_3) \oplus (f_2 \odot f_4)$ is defined implies $(f_1 \oplus f_2) \odot (f_3 \oplus f_4)$ is also defined. The converse does not always hold, but if in addition, $f_1 \odot f_3$ and $f_2 \odot f_4$ are defined, then $(f_1 \oplus f_2) \odot (f_3 \oplus f_4)$ is defined implies $(f_1 \odot f_3) \oplus (f_2 \odot f_4)$ is defined too.*

Proof. We prove each direction individually:

- Given $(f_1 \odot f_3) \oplus (f_2 \odot f_4)$ is defined, it must that $R_1 = D_3$, $R_2 = D_4$, and $R_3 \cap R_4 = D_1 \cap D_2$. Thus, $R_1 \cap R_2 = D_3 \cap D_4 \subseteq R_3 \cap R_4 = D_1 \cap D_2$, ensuring that $f_1 \oplus f_2$ is defined;
  $R_3 \cap R_4 = D_1 \cap D_2 \subseteq R_1 \cap R_2 = D_3 \cap D_4$, ensuring that $f_3 \oplus f_4$ is defined;
  $\mathbf{range}(f_1 \oplus f_2) = R_1 \cup R_2 = D_3 \cup D_4 = \mathbf{dom}(f_3 \oplus f_4)$, ensuring $(f_1 \oplus f_2) \odot (f_3 \oplus f_4)$ is defined.
- Given $f_1 \odot f_3$ and $f_2 \odot f_4$ are defined, $(f_1 \odot f_3) \oplus (f_2 \odot f_4)$ is defined if

$R_3 \cap R_4 = D_1 \cap D_2$. When $(f_1 \oplus f_2) \odot (f_3 \oplus f_4)$ is defined,

$$\begin{aligned} R_3 \cap R_4 &= D_3 \cap D_4 && \text{(Because } f_3 \oplus f_4 \text{ is defined)} \\ &= R_1 \cap R_2 && \text{(Because } f_1 \odot f_3 \text{ and } f_2 \odot f_4 \text{ are defined)} \\ &= D_1 \cap D_2 && \text{(Because } f_1 \oplus f_2 \text{ is defined)} \end{aligned}$$

So $(f_1 \odot f_3) \oplus (f_2 \odot f_4)$ is also defined. □

C.1.5 Main Theorem: Proving Frame Conditions

Theorem 4.2.1. *$(\mathcal{X}_{CI}, \sqsubseteq, \widehat{\oplus}, \widehat{\odot}, \mathcal{X}_{CI})$ is a DIBI frame.*

Proof. We restate the frame conditions using concrete definitions of $\oplus$ and $\odot$ and then check that they hold.

$\oplus$ Down-Closed We want to show that for any $x', x, y', y \in M$, if $x' \sqsubseteq x$ and $y' \sqsubseteq y$ and $x \oplus y = z$, then $x' \oplus y'$ is defined, and $x' \oplus y' = z' \sqsubseteq z$.

Since $x' \sqsubseteq x$ and $y' \sqsubseteq y$, there exist sets S_1, S_2, and $v_1, v_2 \in M$ such that $x = (x' \oplus \mathsf{unit}_{S_1}) \odot v_1$, and $y = (y' \oplus \mathsf{unit}_{S_2}) \odot v_2$. Thus,

$$\begin{aligned} x \oplus y &= ((x' \oplus \mathsf{unit}_{S_1}) \odot v_1) \oplus ((y' \oplus \mathsf{unit}_{S_2}) \odot v_2) \\ &= \big((x' \oplus \mathsf{unit}_{S_1}) \oplus (y' \oplus \mathsf{unit}_{S_2})\big) \odot (v_1 \oplus v_2) \\ &\qquad \text{(By lemma C.1.9 and C.1.8)} \\ &= \big((x' \oplus y') \oplus (\mathsf{unit}_{S_1} \oplus \mathsf{unit}_{S_2})\big) \odot (v_1 \oplus v_2) \\ &\qquad \text{(By commutativity and associativity)} \\ &= \big((x' \oplus y') \oplus (\mathsf{unit}_{S_1 \cup S_2})\big) \odot (v_1 \oplus v_2) \end{aligned}$$

This derivation proved that $x' \oplus y'$ is defined, and $x' \oplus y' \sqsubseteq x \oplus y = z$.

⊙ Up-Closed We want to show that for any $z', z, x, y \in M$, if $z = x \odot y$ and $z' \sqsupseteq z$, then there exists x', y' such that $x' \sqsupseteq x$, $y' \sqsupseteq y$, and $z' = x' \odot y'$.

Since $z' \sqsupseteq z$, there exist set S, and $v \in M$ such that $z' = (z \oplus \mathsf{unit}_S) \odot v$. Thus,

$$
\begin{aligned}
z' &= (z \oplus \mathsf{unit}_S) \odot v \\
&= ((x \odot y) \oplus \mathsf{unit}_S) \odot v \\
&= ((x \odot y) \oplus (\mathsf{unit}_S \odot \mathsf{unit}_S)) \odot v \\
&= ((x \oplus \mathsf{unit}_S) \odot (y \oplus \mathsf{unit}_S)) \odot v && \text{(By lemma C.1.9 and C.1.8)} \\
&= (x \oplus \mathsf{unit}_S) \odot ((y \oplus \mathsf{unit}_S) \odot v) && \text{(By standard associativity of } \odot\text{)}
\end{aligned}
$$

Thus, for $x' = x \oplus \mathsf{unit}_S$ and $y' = (y \oplus \mathsf{unit}_S) \odot v$, $z' = x' \odot y'$.

⊕ Commutativity We want to show that $z = x \oplus y$ implies that $z = y \oplus x$. First, $x \oplus y$ is defined iff $\mathbf{range}(x) \cap \mathbf{range}(y) = \mathbf{dom}(x) \cap \mathbf{dom}(y)$ iff $y \oplus x$ is defined; second, when $x \oplus y$ and $y \oplus x$ are both defined, they are equal due to lemma C.1.5. Thus, the $\oplus$ commutativity frame condition is satisfied.

⊕ Associativity We want to show that $z = (x \oplus y) \oplus z$ implies that $z = x \oplus (y \oplus z)$.

We show that in lemma C.1.4.

⊕ Unit Existence We want to show that for any $x \in M$, there exists $e \in E$ such that $x = e \oplus x$. We show that $e = \mathsf{unit}_\emptyset$ serves as the unit under $\oplus$ for any x. For any $x : \mathbf{Mem}[A] \to \mathcal{D}(\mathbf{Mem}[B])$, $x \oplus \mathsf{unit}_\emptyset$ is defined because $B \cap \emptyset = \emptyset = A \cap \emptyset$, and by lemma C.1.7, $(x \oplus \mathsf{unit}_\emptyset) = x$.

⊕ Unit Coherence We want to show that for any $y \in M$, $e \in E = M$, if $x = y \oplus e$,

then $x \sqsupseteq y$.

$$
\begin{aligned}
x &= y \oplus e \\
&= (y \odot \mathsf{unit}_{\mathbf{range}(y)}) \oplus (\mathsf{unit}_{\mathbf{dom}(e)} \odot e) \\
&= (y \oplus \mathsf{unit}_{\mathbf{dom}(e)}) \odot (\mathsf{unit}_{\mathbf{range}(y)} \oplus e) \\
&\qquad\qquad \text{(By lemma C.1.8 and lemma C.1.9)} \\
&= (y \oplus \mathsf{unit}_{\mathbf{dom}(e)}) \odot (e \oplus \mathsf{unit}_{\mathbf{range}(y)}) \qquad (\oplus \text{ Commutativity})
\end{aligned}
$$

Thus, $x \sqsupseteq y$.

$\odot$ Associativity The frame axiom reduces to the standard associativity of $\odot$. Kleisli composition satisfies standard associativity, so $\odot$ also satisfies standard associativity.

$\odot$ Unit Existence$_{\mathrm{L}}$ and $\odot$ Unit Existence$_{\mathrm{R}}$ We need to show that, for any $x \in M$, there exists $e \in E$ such that $e \odot x = x$, and there exists $e' \in E$ such that $x \odot e' = x$. Since $\odot$ is the Kleisli composition, for any morphism $x : \mathbf{Mem}[A] \to \mathcal{D}(\mathbf{Mem}[B])$, unit_A is the left unit, and unit_B is the right unit. In addition, for all S, $\mathsf{unit}_S \in M = E$.

$\odot$ Coherence$_{\mathrm{R}}$ For any $y \in M, e \in E$ such that $x = y \odot e$, we want to show that $x \sqsupseteq y$. We proved in lemma C.1.7 that $(y \oplus \mathsf{unit}_\emptyset) = y$ for any y, so $x = y \odot e = (y \oplus \mathsf{unit}_\emptyset) \odot e$, and $x \sqsubseteq y$ as desired.

Unit Closure We want to show that for any $e \in E$ and $e' \sqsupseteq e$, $e' \in E$. This is evident because $E = M$ and M is closed under $\oplus$ and $\odot$.

Reverse Exchange Given $x = y \oplus z$ and $y = y_1 \odot y_2$, $z = z_1 \odot z_2$, we want to show that there exists $u = y_1 \oplus z_1$, $v = y_2 \oplus z_2$, and $x = u \odot v$.

After substitution, we get $(y_1 \odot y_2) \oplus (z_1 \odot z_2) = y \oplus z = x$. By C.1.8 and lemma C.1.9, when $(y_1 \odot y_2) \oplus (z_1 \odot z_2)$ is defined, $(y_1 \oplus z_1) \odot (y_2 \odot z_2)$

is also defined, and $(y_1 \odot y_2) \oplus (z_1 \odot z_2) = (y_1 \oplus z_1) \odot (y_2 \oplus z_2)$. Thus $(y_1 \oplus z_1) \odot (y_2 \oplus z_2) = y \oplus z = x$, and thus $u = y_1 \oplus z_1$, $v = y_2 \oplus z_2$ completes the proof. □

C.2 Capturing Conditional Independence

C.2.1 Properties of the Probabilistic Frame

We prove some properties of the model that are useful for proving lemma C.2.8.

Lemma C.2.1 (Disintegration)**.** *If $f = f_1 \odot f_2$, then $\pi_{R_1} f = f_1$. Conversely, if $\pi_{R_1} f = f_1$, then there exists g such that $f = f_1 \odot g$.*

Proof. In short, it follows from properties of Kleisli category of discrete probability monad: Kleisli category of discrete probability monad is a Markov category that has conditionals [Fritz, 2020, Example 11.2]; since the kernels are morphisms in this category, and the operator $\odot$ is the morphism composition in the category, we have this lemma. We spell out the detailed proof in the following.

For the forwards direction, suppose that $f = f_1 \odot f_2$. Then,

$$\pi_{R_1} f = \pi_{R_1}(f_1 \odot f_2) = f_1 \odot (\pi_{R_1} f_2) = f_1 \odot \mathsf{unit}_{R_1} = f_1.$$

Thus, $\pi_{R_1} f = f_1$.

For the converse, assume $\pi_{R_1} f = f_1$. Denote $\mathbf{range}(f)$ as R. Define $g : \mathbf{Mem}[R_1] \to \mathcal{D}(\mathbf{Mem}[R])$ such that for any $r \in \mathbf{Mem}[R_1]$, $m \in \mathbf{Mem}[R]$ such

that $f_1(r^{D_1})(r) \neq 0$, let

$$g(r)(m) := \begin{cases} \frac{f(r^{D_1})(m)}{f_1(r^{D_1})(r)} & r \bowtie m \text{ is defined} \\ 0 & r \bowtie m \text{ not defined} \end{cases}$$

We need to check that $g \in \mathcal{X}_{CI}$. Fixing any $r \in \mathbf{Mem}[R_1]$,

$$\begin{aligned} \sum_{m \in \mathbf{Mem}[R]} g(r)(m) &= \sum_{m \in \mathbf{Mem}[R] \text{ and } m \bowtie r \text{ is defined}} \frac{f(r^{D_1})(m)}{f_1(r^{D_1})(r)} && \text{(By definition of } g) \\ &= \sum_{y \in \mathbf{Mem}[R \setminus R']} \frac{f(r^{D_1})(m)}{f_1(r^{D_1})(r)} \\ &= \sum_{y \in \mathbf{Mem}[R \setminus R']} \frac{f(r^{D_1})(y)}{\sum_{x \in \mathbf{Mem}[R \setminus R_1]} f(r^{D_1})(r \bowtie x)} && \text{(Because } \pi_{R_1} f = f_1) \\ &= 1 \end{aligned}$$

so g does map any input to a distribution, and g preserves the input.

By their types, $f_1 \odot g$ is defined. For any $d \in \mathbf{Mem}[D_1]$, $m \in \mathbf{Mem}[R]$, if $f_1(d)(m^{R_1}) \neq 0$, then

$$\begin{aligned} (f_1 \odot g)(d)(m) &= f_1(d)(m^{R_1}) \cdot g(m^{R_1})(m) \\ &= f_1(d)(m^{R_1}) \cdot \frac{f(m^{D_1})(m)}{f_1(m^{D_1})(m^{R_1})} \\ &= f(d)(m) && (d \bowtie m \text{ is defined iff } d = m^{D_1}) \end{aligned}$$

If $(\pi_{R_1} f)(d)(m^{R_1}) = 0$, then $f(d)(m) = 0$, and $(f_1 \odot g)(d)(m) = f_1(d)(m^{R_1}) \cdot g(m^{R_1})(m) = 0 = f(d)(m)$. Thus, $f_1 \odot g = f$. □

Lemma C.2.2 (Uniqueness)**.** *For any $f, g \colon \mathbf{Mem}[X] \to \mathcal{D}(\mathbf{Mem}[X \cup Y])$ in M, and arbitrary $h \in M$, if $f \sqsubseteq h$ and $g \sqsubseteq h$, then $f = g$.*

Proof. $f \sqsubseteq h$ implies that there exists v_1, S_1 such that $(f \oplus \mathsf{unit}_{S_1}) \odot v_1 = h$; $g \sqsubseteq h$ implies that there exists v_2, S_2 such that $(g \oplus \mathsf{unit}_{S_2}) \odot v_2 = h$. Take $h \colon \mathbf{Mem}[W] \to$

$\mathcal{D}(\mathbf{Mem}[Z \cup W])$, and then

$$f \oplus \mathsf{unit}_{S_1} = \pi_{\mathbf{range}(f \oplus \mathsf{unit}_{S_1})} h = \pi_{X \cup Y \cup \mathbf{dom}(h)} h$$
$$g \oplus \mathsf{unit}_{S_2} = \pi_{\mathbf{range}(g \oplus \mathsf{unit}_{S_2})} h = \pi_{X \cup Y \cup \mathbf{dom}(h)} h$$

Thus, $f \oplus \mathsf{unit}_{S_1} = g \oplus \mathsf{unit}_{S_2}$. Now, suppose $f \neq g$. This would imply $f \oplus \mathsf{unit}_{S_1} \neq g \oplus \mathsf{unit}_{S_2}$ which is a contradiction. Thus, $f = g$. □

Lemma C.2.3 ($\odot$ elimination)**.** *For any $f, g \in \mathcal{X}_{CI}$, if $f \odot (g \oplus \mathsf{unit}_X)$ is defined and $\mathbf{dom}(g) \subseteq \mathbf{dom}(f)$, then $f \odot (g \oplus \mathsf{unit}_X) = g \oplus f$.*

Proof. Let $f : \mathbf{Mem}[S] \to \mathcal{D}(\mathbf{Mem}[S \cup T])$ and $g : \mathbf{Mem}[U] \to \mathcal{D}(\mathbf{Mem}[U \cup V])$ be in M. When $U \subseteq S$,

$$\begin{aligned}
& f \odot (g \oplus \mathsf{unit}_X) \\
&= (f \oplus \mathsf{unit}_U) \odot (g \oplus \mathsf{unit}_X \oplus \mathsf{unit}_{S \cup T}) && \text{(By C.1.7)} \\
&= (\mathsf{unit}_U \oplus f) \odot (g \oplus \mathsf{unit}_X \oplus \mathsf{unit}_{S \cup T}) && \text{(By commutativity)} \\
&= (\mathsf{unit}_U \oplus f) \odot (g \oplus \mathsf{unit}_{S \cup T}) && (\dagger) \\
&= (\mathsf{unit}_U \odot g) \oplus (f \odot \mathsf{unit}_{S \cup T}) && \text{(By lemma C.1.9 and C.1.8)} \\
&= g \oplus f && \square
\end{aligned}$$

where † follows from $X \subseteq S \cup T$, which holds as $f \odot (g \oplus \mathsf{unit}_X)$ defined implies $S \cup T = X \cup U$.

Lemma C.2.4 (Converting $\oplus$ to $\odot$)**.** *For any kernel $f : \mathbf{Mem}[S] \to \mathcal{D}(\mathbf{Mem}[S \cup T])$ and $g : \mathbf{Mem}[U] \to \mathcal{D}(\mathbf{Mem}[U \cup V])$ in $\mathcal{X}_{CI}$. If $f \oplus g$ is defined, then $f \oplus g = (f \oplus \mathsf{unit}_U) \odot (\mathsf{unit}_{S \cup T} \oplus g)$.*

Proof.

$$\begin{aligned} f \oplus g &= (f \odot \mathsf{unit}_{S \cup T}) \oplus (\mathsf{unit}_U \odot g) \\ &= (f \oplus \mathsf{unit}_U) \odot (\mathsf{unit}_{S \cup T} \oplus g) && \text{(By lemma C.1.9 and C.1.8)} \end{aligned}$$

□

Lemma C.2.5 (Quasi-Downwards-closure of $\odot$)**.** *For any $f, g, h, i \in \mathcal{X}_{CI}$, if $f \sqsubseteq h$, $g \sqsubseteq i$, and $f \odot g$, $h \odot i$ are all defined, then $f \odot g \sqsubseteq h \odot i$.*

Proof. Since $f \sqsubseteq h$, $g \sqsubseteq i$, there must exist sets S_1, S_2 and $v_1, v_2 \in M$ such that $h = (f \oplus \mathsf{unit}_{S_1}) \odot v_1$, $i = (g \oplus \mathsf{unit}_{S_2}) \odot v_2$. $f \odot g$ is defined, so $\mathbf{dom}(g) = \mathbf{range}(f) \subseteq \mathbf{range}(f \oplus \mathsf{unit}_{S_1}) = \mathbf{dom}(v_1)$. Thus,

$$\begin{aligned} h \odot i &= (f \oplus \mathsf{unit}_{S_1}) \odot v_1 \odot (g \oplus \mathsf{unit}_{S_2}) \odot v_2 \\ &= (f \oplus \mathsf{unit}_{S_1}) \odot (g \oplus v_1) \odot v_2 && \text{(By lemma C.2.3 and } \mathbf{dom}(g) \subseteq \mathbf{dom}(v_1)) \\ &= (f \oplus \mathsf{unit}_{S_1}) \odot (g \oplus \mathsf{unit}_{\mathbf{dom}(v_1)}) \odot (\mathsf{unit}_{\mathbf{range}(g)} \oplus v_1) \odot v_2 \\ & && \text{(By lemma C.2.4)} \\ &= (f \oplus \mathsf{unit}_{S_1}) \odot (g \oplus \mathsf{unit}_{S_1}) \odot (\mathsf{unit}_{\mathbf{range}(g)} \oplus v_1) \odot v_2 && (\dagger) \\ &= ((f \odot g) \oplus (\mathsf{unit}_{S_1} \odot \mathsf{unit}_{S_1})) \odot (\mathsf{unit}_{\mathbf{range}(g)} \oplus v_1) \odot v_2 && (\heartsuit) \\ &= ((f \odot g) \oplus \mathsf{unit}_{S_1}) \odot (\mathsf{unit}_{\mathbf{range}(g)} \oplus v_1) \odot v_2 \end{aligned}$$

where † follows from $\mathbf{dom}(g) = \mathbf{range}(f)$ and lemma C.1.7, and ♡ follows from lemma C.1.9 and C.1.8.

Therefore, $f \odot g \sqsubseteq h \odot i$. □

C.2.2 Key Lemmas: Conditional Independence is Expressed

Lemma C.2.6 (Classical flavor in intuitionistic model)**.** *For any* $f \in M$*,*

$$f \models (\emptyset \triangleright Z) \fatsemi ((Z \triangleright X) * (Z \triangleright Y))$$

if and only if there exist $g, h, i \in M$*, such that* $g\colon \mathbf{Mem}[\emptyset] \to \mathcal{D}(\mathbf{Mem}[Z])$*,* $h\colon \mathbf{Mem}[Z] \to \mathcal{D}(\mathbf{Mem}[Z \cup X])$*,* $i\colon \mathbf{Mem}[Z] \to \mathcal{D}(\mathbf{Mem}[Z \cup Y])$*, and* $g \odot (h \oplus i) \sqsubseteq f$*.*

Proof. The backwards direction trivially follows from persistence. We detail the proof for the forward direction here. Suppose $f \models (\emptyset \triangleright Z) \fatsemi ((Z \triangleright X) * (Z \triangleright Y))$. Then, there exist f_1, f_2, f_3, f_4 such that $f_1 \odot f_2 = f$, $f_3 \oplus f_4 \sqsubseteq f_2$, $f_1 \models (\emptyset \triangleright Z)$, $f_3 \models (Z \triangleright X)$ and $f_4 \models (Z \triangleright Y)$.

- $f_1 \models (\emptyset \triangleright Z)$ implies that there exists $f_1'' \sqsubseteq f_1$ such that $\mathbf{dom}(f_1'') = \emptyset$, and $\mathbf{range}(f_1'') \supseteq Z$. Let $f_1' = \pi_Z f_1''$. Note that $f_1'\colon \mathbf{Mem}[\emptyset] \to \mathcal{D}(\mathbf{Mem}[Z])$ and $f_1' \sqsubseteq f_1'' \sqsubseteq f_1$. Hence, there exists some set S_1 and $v_1 \in M$ such that $f_1 = (f_1' \oplus \mathsf{unit}_{S_1}) \odot v_1$.
- $f_3 \models (Z \triangleright X)$ implies that there exists $f_3'' \sqsubseteq f_3$ such that $\mathbf{dom}(f_3'') = Z$, and $\mathbf{range}(f_3'') \supseteq X$. Define $f_3' = \pi_{Z \cup X} f_3''$. Then $f_3' \sqsubseteq f_3'' \sqsubseteq f_3$, and $f_3'\colon \mathbf{Mem}[Z] \to \mathcal{D}(\mathbf{Mem}[X \cup Z])$.
- $f_4 \models (Z \triangleright Y)$ implies that there exists $f_4'' \sqsubseteq f_4$ such that $\mathbf{dom}(f_4'') = Z$, and $\mathbf{range}(f_4'') \supseteq Y$. Define $f_4' = \pi_{Z \cup Y} f_4''$ and note that $f_4'\colon \mathbf{Mem}[Z] \to \mathcal{D}(\mathbf{Mem}[Y \cup Z])$.
- By $\oplus$ Down-Closed, having $f_3 \oplus f_4$ defined implies that $f_3' \oplus f_4'$ is also defined and $f_3' \oplus f_4' \sqsubseteq f_3 \oplus f_4 \sqsubseteq f_2$. Thus, there exists some $v_2 \in M$ and finite set S_2 such that $f_2 = (f_3' \oplus f_4' \oplus \mathsf{unit}_{S_2}) \odot v_2$.

Using these observations, we can now calculate and show that $f_1' \odot (f_3' \oplus f_4' \oplus \mathsf{unit}_Z) \sqsubseteq f_1 \oplus f_2$:

$$
\begin{aligned}
& f_1 \odot f_2 \\
&= (f_1' \oplus \mathsf{unit}_{S_1}) \odot v_1 \odot (f_3' \oplus f_4' \oplus \mathsf{unit}_{S_2}) \odot v_2 \\
&= (f_1' \oplus \mathsf{unit}_{S_1}) \odot \left(f_3' \oplus f_4' \oplus v_1\right) \odot v_2 \\
& \qquad\qquad \text{(By lemma C.2.3 and } \mathbf{dom}(f_3' \oplus f_4') = Z \subseteq \mathbf{range}(f_1' \oplus \mathsf{unit}_{S_1})) \\
&= (f_1' \oplus \mathsf{unit}_{S_1}) \odot \left((f_3' \oplus f_4' \oplus \mathsf{unit}_{\mathbf{dom}(v_1)}) \odot (\mathsf{unit}_{X \cup Y \cup Z} \oplus v_1)\right) \odot v_2 \quad \text{(By lemma C.2.4)} \\
&= (f_1' \oplus \mathsf{unit}_{S_1}) \odot (f_3' \oplus f_4' \oplus \mathsf{unit}_Z \oplus \mathsf{unit}_{S_1}) \odot (\mathsf{unit}_{X \cup Y \cup Z} \oplus v_1) \odot v_2 \\
& \qquad\qquad \text{(By } \mathbf{dom}(v_1) = Z \cup S_1) \\
&= \left((f_1' \odot (f_3' \oplus f_4' \oplus \mathsf{unit}_Z)) \oplus (\mathsf{unit}_{S_1} \odot \mathsf{unit}_{S_1})\right) \odot (\mathsf{unit}_{X \cup Y \cup Z} \oplus v_1) \odot v_2 \\
& \qquad\qquad \text{(By lemma C.1.8 and lemma C.1.9)} \\
&= \left((f_1' \odot (f_3' \oplus f_4' \oplus \mathsf{unit}_Z)) \oplus \mathsf{unit}_{S_1}\right) \odot (\mathsf{unit}_{X \cup Y \cup Z} \oplus v_1) \odot v_2 \\
&= \left((f_1' \odot (f_3' \oplus f_4')) \oplus \mathsf{unit}_{S_1}\right) \odot (\mathsf{unit}_{X \cup Y \cup Z} \oplus v_1) \odot v_2 \quad \text{(By lemma C.1.7)}
\end{aligned}
$$

To finish, take $g = f_1' \colon \mathbf{Mem}[\emptyset] \to \mathcal{D}(\mathbf{Mem}[Z])$, $h = f_3' \colon \mathbf{Mem}[Z] \to \mathcal{D}(\mathbf{Mem}[Z \cup X])$, $i = f_4' \colon \mathbf{Mem}[Z] \to \mathcal{D}(\mathbf{Mem}[Z \cup Y])$, and note that $g \odot (h \oplus i) = f_1' \odot (f_3' \oplus f_4') \sqsubseteq f_1 \oplus f_2 \sqsubseteq f$. □

Lemma C.2.7. *If X, Y are conditionally independent given S, then values on $X \cap Y$ is determined given values on S.*

Proof. In short, conditional independence is closed under projection (of the conditionally independent components). Thus, $X \cap Y$ is conditionally independent to itself given S. Any random variable independent to itself must be deterministic. Thus, $X \cap Y$ is deterministic given S. We spell out the detailed proof below.

Let $X' = X \setminus Y$, $Y' = Y \setminus X$. By assumption, X, Y are conditionally independent

given S , so $x \in \mathbf{Mem}[X]$, $y \in \mathbf{Mem}[Y]$, $s \in \mathbf{Mem}[S]$,

$$\mu(X = x \mid S = s) \cdot \mu(Y = y \mid S = s) = \mu(X = x \cap Y = y \mid S = s).$$

Thus, if we denote $x' = \pi_{X'} x$, $y' = \pi_{Y'} y$ and let $M = X \cap Y$, then for any $m \in \mathbf{Mem}[M]$,

$$\begin{aligned} &\mu(X' = x' \cap Y' = y' \cap M = m \mid S = s) \\ &= \mu(X' = x' \cap M = m \mid S = s) \cdot \mu(Y' = y' \cap M = m \mid S = s) \end{aligned} \tag{C.6}$$

For any probabilistic events E_1, E_2, E_3, $\mu(E_1 \cap E_2 \mid E_3) = \mu(E_1 \mid E_2, E_3) \cdot \mu(E_2 \mid E_3)$. Thus, eq. (C.6) implies that

$$\begin{aligned} &\mu(X' = x' \mid M = m \cap S = s) \cdot \mu(Y' = y' \mid M = m \cap S = s) \cdot \mu(M = m \mid S = s) \\ &= \mu(X' = x' \cap Y' = y' \mid M = m \cap S = s) \end{aligned} \tag{C.7}$$

Then, for any $s \in \mathbf{Mem}[S], m \in \mathbf{Mem}[M]$ such that $\mu(M = m \cap S = s) \neq 0$,

$$\begin{aligned} &\sum_{x' \in \mathbf{Mem}[X'] \cap y' \in \mathbf{Mem}[Y']} \mu(X' = x' \mid M = m \cap S = s) \cdot \mu(Y' = y' \mid M = m, S = s) \cdot \mu(M = m \mid S = s) \\ &= \sum_{x' \in \mathbf{Mem}[X'], y' \in \mathbf{Mem}[Y']} \mu(X' = x', Y' = y' \mid M = m, S = s) \qquad \text{(Because of eq. (C.7))} \\ &= 1 \end{aligned} \tag{C.8}$$

Meanwhile, for any $s \in \mathbf{Mem}[S], m \in \mathbf{Mem}[M]$ such that $m \bowtie s$ is defined and $\mu(M = m, S = s) \neq 0$,

$$\begin{aligned} &\sum_{x' \in \mathbf{Mem}[X'], y' \in \mathbf{Mem}[Y']} \mu(X' = x' \mid M = m \cap S = s) \cdot \mu(Y' = y' \mid M = m \cap S = s) \cdot \mu(M = m \mid S = s) \\ &= \left(\sum_{x' \in \mathbf{Mem}[X'], y' \in \mathbf{Mem}[Y']} \mu(X' = x' \mid M = m, S = s) \cdot \mu(Y' = y' \mid M = m \cap S = s) \right) \cdot \mu(M = m \mid S = s) \\ &= \left(\sum_{x' \in \mathbf{Mem}[X']} \mu(X' = x' \mid M = m \cap S = s) \right) \cdot \left(\sum_{y' \in \mathbf{Mem}[Y']} \mu(Y' = y' \mid M = m \cap S = s) \right) \cdot \mu(M = m \mid S = s) \\ &= 1 \cdot \mu(M = m \mid S = s) \end{aligned} \tag{C.9}$$

Combining eq. (C.9) and eq. (C.8), we derive $\mu(M = m \mid S = s) = 1$. That is, when $X \perp\!\!\!\perp Y \mid S$, whether $M \supseteq S$ or not, $\mu(M = m, S = s) \neq 0$ implies $\mu(M = m \mid S = s) = 1$. Thus, $X \perp\!\!\!\perp Y \mid S$ renders values on $X \cap Y$ deterministic given values on S. □

Lemma C.2.8. *For a distribution μ on* **Var**, *let f_μ denote the kernel $\langle\rangle \mapsto \mu$. Then, there exist $S, X, Y \subseteq$* **Var**, $f_1 : \mathbf{Mem}[\emptyset] \to \mathcal{D}(\mathbf{Mem}[S])$, $f_2 : \mathbf{Mem}[S] \to \mathcal{D}(\mathbf{Mem}[S \cup X])$, $f_3 : \mathbf{Mem}[S] \to \mathcal{D}(\mathbf{Mem}[S \cup Y])$ *such that $f_1 \odot (f_2 \oplus f_3) \sqsubseteq f_\mu$, if and only if $X \perp\!\!\!\perp Y \mid S$ and also $X \cap Y \subseteq S$.*

Proof. **Forward direction**: Assume the existence of f_1, f_2, f_3 satisfying $f_1 \odot (f_2 \oplus f_3) \sqsubseteq f_\mu$. We must prove $X \perp\!\!\!\perp Y \mid S$ and $X \cap Y \subseteq S$.

1. $X \cap Y \subseteq S$: $f_2 \oplus f_3$ defined implies $(X \cup S) \cap (Y \cup S) \subseteq S \cap S$. Thus, $X \cap Y \subseteq S$.
2. $X \perp\!\!\!\perp Y \mid S$: By assumption, $f_1 \odot (f_2 \oplus f_3) \sqsubseteq f_\mu$. lemma C.2.1 gives us $f_1 \odot (f_2 \oplus f_3) = \pi_{S \cup X \cup Y}(f_\mu)$, and $f_1 = \pi_S(f_\mu)$. Thus, for any $m \in \mathbf{Mem}[X \cup Y \cup S]$,

$$\begin{aligned}\mu(X = m^X, Y = m^Y, S = m^S) &= (\pi_{X \cup Y \cup S}\mu)(m^X \bowtie m^Y \bowtie m^S) \\ &= \pi_{X \cup Y \cup S}(f_\mu)(\langle\rangle)(m^X \bowtie m^Y \bowtie m^S) \\ &= f_1 \odot (f_2 \oplus f_3)(\langle\rangle)(m^X \bowtie m^Y \bowtie m^S)\end{aligned}$$

Similarly, since $f_1 = \pi_S(f_\mu)$, we have

$$\mu(S = m^S) = (\pi_S\mu)(m^S) = \big(\pi_S(f_\mu)\big)(\langle\rangle)(m^S) = f_1(\langle\rangle)(m^S) \tag{C.10}$$

By definition of conditional probability, when $\mu(S = m^S) \neq 0$,

$$\begin{aligned}
&\mu(X = m^X, Y = m^Y \mid S = m^S) \\
&= \frac{\mu(X = m^X, Y = m^Y, S = m^S)}{\mu(S = m^S)} \\
&= \frac{f_1 \odot (f_2 \oplus f_3)(\langle\rangle)(m^S \bowtie m^X \bowtie m^Y)}{f_1(\langle\rangle)(m^S)} \\
&= \frac{f_1(\langle\rangle)(m^S) \cdot (f_2 \oplus f_3)(m^S)(m^S \bowtie m^X \bowtie m^Y)}{f_1(\langle\rangle)(m^S)} && \text{(By eq. (4.2))} \\
&= (f_2 \oplus f_3)(m^S)(m^S \bowtie m^X \bowtie m^Y) \\
&= f_2(m^S)(m^{X \cup S}) \cdot f_3(m^S)(m^{Y \cup S}) && \text{(C.11)}
\end{aligned}$$

Let $f_2' = f_2 \oplus \mathsf{unit}_{\mathbf{Mem}[Y]}$, $f_3' = f_3 \oplus \mathsf{unit}_{\mathbf{Mem}[X]}$. By lemma C.2.4,

$$f_1 \odot (f_2 \oplus f_3) = f_1 \odot f_2 \odot (f_3 \oplus \mathsf{unit}_{\mathbf{Mem}[X]}) = f_1 \odot f_2 \odot f_3'$$

$$f_1 \odot (f_2 \oplus f_3) = f_1 \odot (f_3 \oplus f_2) = f_1 \odot f_3 \odot (f_2 \oplus \mathsf{unit}_{\mathbf{Mem}[Y]}) = f_1 \odot f_3 \odot f_2'$$

Lemma C.2.1 gives us $\pi_{X \cup S}(f_\mu) = f_1 \odot f_2$, and $\pi_{Y \cup S}(f_\mu) = f_1 \odot f_3$, Therefore,

$$\begin{aligned}
\mu(X = m^X, S = m^S) &= (\pi_{X \cup S}(f_\mu))(\langle\rangle)(m^S \bowtie m^X) \\
&= (f_1 \odot f_2)(\langle\rangle)(m^S \bowtie m^X) \\
&= f_1(\langle\rangle)(m^S) \cdot f_2(m^S)(m^S \bowtie m^X) \\
\mu(Y = m^Y, S = m^S) &= (\pi_{Y \cup S}(f_\mu)(\langle\rangle)(m^S \bowtie m^Y) \\
&= (f_1 \odot f_3)(\langle\rangle)(m^S \bowtie m^Y) \\
&= f_1(\langle\rangle)(m^S) \cdot f_3(m^S)(m^S \bowtie m^Y)
\end{aligned}$$

Thus, by definition of conditional probability.

$$\begin{aligned}\mu(X = m^X \mid S = m^S) &= \frac{\mu(X = m^X, S = m^S)}{\mu(S = m^S)} \\ &= \frac{f_1(\langle\rangle)(m^S) \cdot f_2(m^S)(m^{S\cup X})}{f_1(\langle\rangle)(m^S)} \\ &= f_2(m^S)(m^{S\cup X}) \qquad \text{(C.12)} \\ \mu(X = m^Y \mid S = m^S) &= \frac{\mu(X = m^X, S = m^S)}{\mu(S = m^S)} \\ &= \frac{f_1(\langle\rangle)(m^S) \cdot f_3(m^S)(m^{S\cup Y})}{f_1(\langle\rangle)(m^S)} \\ &= f_3(m^S)(m^{S\cup Y}) \qquad \text{(C.13)}\end{aligned}$$

Substituting eq. (C.12) and eq. (C.13) into the equation eq. (C.11), we have

$$\mu(X = m^X, Y = m^Y \mid S = m^S) = \mu(X = m^X \mid S = m^S) \cdot \mu(X = m^Y \mid S = m^S))$$

Thus, X, Y are conditionally independent given S. This completes the proof for the first direction.

Backward direction: We want to show that if $X \perp\!\!\!\perp Y \mid S$ and $X \cap Y \subseteq S$ then there exists such f_1, f_2, f_3 that $f_1 \odot (f_2 \oplus f_3) \sqsubseteq f_\mu$. Given μ, we define $f_1 = \pi_S(f_\mu)$ and construct f_2, f_3 as follows:

Let $f_2 : \mathbf{Mem}[S] \to \mathcal{D}(\mathbf{Mem}[S \cup X])$. For any $s \in \mathbf{Mem}[S]$, $x \in \mathbf{Mem}[X]$, when $f_1(\langle\rangle)(s) \neq 0$, let

$$f_2(s)(s \bowtie x) := \begin{cases} \frac{(\pi_{S\cup X} f_\mu)(\langle\rangle)(s \bowtie x)}{f_1(\langle\rangle)(s)} & \text{if } s \bowtie x \text{ is defined} \\ 0 & \text{otherwise} \end{cases}$$

When $f_1(\langle\rangle)(s) = 0$, we can define $f_2(s)(s \bowtie x)$ arbitrarily as long as $f_2(s)$ is a distribution, because that distribution will be zeroed out in $f_1 \odot (f_2 \oplus f_3)$ anyway.

Similarly, let $f_3 : \mathbf{Mem}[S] \to \mathcal{D}(\mathbf{Mem}[S \cup Y])$. For any $s \in \mathbf{Mem}[S]$, $x \in \mathbf{Mem}[Y]$ such that $s \bowtie y$ is defined, when $f_1(\langle\rangle)(s) \neq 0$, let

$$f_3(s)(s \bowtie y) := \begin{cases} \frac{(\pi_{S \cup Y} f_\mu)(s \bowtie y)}{f_1(\langle\rangle)(s)} & \text{if } s \bowtie y \text{ is defined} \\ 0 & \text{otherwise} \end{cases}$$

By construction, f_1, f_2, f_3 each has the type needed for the lemma. We are left to prove that given any $s \in \mathbf{Mem}[S]$, f_2 and f_3 are kernels in $\mathcal{X}_{CI}$, $f_1 \odot (f_2 \oplus f_3)$ is defined, and $f_1 \odot (f_2 \oplus f_3) \sqsubseteq f_\mu$.

- State f_2 is in $\mathcal{X}_{CI}$, which boils down to show that: for any $s \in \mathbf{Mem}[S]$, $f_2(s)$ forms a distribution, and also f_2 preserves the input. It can be shown through by mechanical calculation and we omit it here.
- State f_3 is in $\mathcal{X}_{CI}$. Similar as above.
- State $f_1 \odot (f_2 \oplus f_3)$ is defined.

  $f_2 \oplus f_3$ is defined because $R_2 \cap R_3 = (S \cup X) \cap (S \cup Y) = S \cup (X \cap Y)$, and by assumption, $X \cap Y \subseteq S$, so $S \cup (X \cap Y) = S = D_2 \cap D_3$. Then $f_1 \odot (f_2 \oplus f_3)$ is defined because $\mathbf{dom}(f_2 \oplus f_3) = D_2 \cup D_3 = S \cup S = S = \mathbf{range}(f_1)$.
- State $f_1 \odot (f_2 \oplus f_3) \sqsubseteq f_\mu$.

  It suffices to show that there exists g such that $(f_1 \odot (f_2 \oplus f_3)) \odot g = f_\mu$.

  For any $s \in \mathbf{Mem}[S]$, $x \in \mathbf{Mem}[X]$, $y \in \mathbf{Mem}[Y]$ such that $s \bowtie x \bowtie y$ is defined,

$$\begin{aligned} & f_1 \odot (f_2 \oplus f_3)(\langle\rangle)(s \bowtie x \bowtie y) && \text{(C.14)} \\ &= f_1(\langle\rangle)(s) \cdot f_2 \oplus f_3(s)(s \bowtie x \bowtie y) \\ &= f_1(\langle\rangle)(s) \cdot (f_2(s)(s \bowtie x) \cdot f_3(s)(s \bowtie y)) \\ &= \mu(S = s) \cdot (\mu(X = x \mid S = s) \cdot \mu(Y = y \mid S = s)) && \text{(C.15)} \end{aligned}$$

Because X, Y are conditionally independent given S in the distribution q,

$$\mu(X = x \mid S = s) \cdot \mu(Y = y \mid S = s) = \mu(X = x, Y = y \mid S = s) \tag{C.16}$$

Substituting eq. (C.16) into eq. (C.15), we have

$$\begin{aligned} f_1 \odot (f_2 \oplus f_3)(\langle\rangle)(s \bowtie x \bowtie y) &= \mu(S = s) \cdot \mu(X = x, Y = y \mid S = s) \\ &= \mu(X = x, Y = y, S = s) \end{aligned}$$

Let $g : \mathbf{Mem}[X \cup Y \cup S] \to \mathcal{D}(\mathbf{Mem}[\mathbf{Val}])$ such that for any $d \in \mathbf{Mem}[X \cup Y \cup S]$, $m \in \mathbf{Mem}[\mathbf{Val}]$ such that $d \bowtie m$ is defined, let

$$g(d)(m) = \mu(\mathbf{Val} = m \mid X \cup Y \cup S = d)$$

Then, $(f_1 \odot (f_2 \oplus f_3)) \odot g$ is defined, and

$$\begin{aligned} (f_1 \odot (f_2 \oplus f_3) \odot g)(\langle\rangle)(m) &= (f_1 \odot (f_2 \oplus f_3))(\langle\rangle)(m^{X \cup Y \cup S}) \cdot g(m^{X \cup Y \cup S})(m) \\ &= \mu(\mathbf{Val} = m) \end{aligned}$$

Thus, $(f_1 \odot (f_2 \oplus f_3)) \odot g = f_\mu$, and therefore $f_1 \odot (f_2 \oplus f_3) \sqsubseteq f_\mu$.

This completes the proof for the backwards direction. □

C.2.3 Validating Graphoid Axioms, Section 4.2.3

Lemma C.2.9 (Weak Union)**.** *The following judgment is valid in $\mathcal{X}_{CI}$:*

$$\models [Z] \fatsemi ([X] * [Y \cup W]) \to [Z \cup W] \fatsemi ([X] * [Y])$$

Proof. For any $f \in \mathcal{X}_{CI}$, if $f \models [Z] \fatsemi ([X] * [Y \cup W])$, by lemma C.2.6, there exist $f_1, f_2, f_3 \in M$ such that $f_1 \odot (f_2 \oplus f_3) \sqsubseteq f$, $f_1 : \mathbf{Mem}[\emptyset] \to \mathcal{D}(\mathbf{Mem}[Z])$, $f_2 : \mathbf{Mem}[Z] \to \mathcal{D}(\mathbf{Mem}[Z \cup X])$, $f_3 : \mathbf{Mem}[Z] \to \mathcal{D}(\mathbf{Mem}[Z \cup Y \cup W])$.

Let $f_3^1 = \pi_{Z \cup W} f_3$, then by Disintegration there exists $f_3^2 \in M$ such that $f_3 = f_3^1 \odot f_3^2$. We note that

$$\begin{aligned}
f_1 \odot (f_2 \oplus f_3) &= f_1 \odot f_3 \odot (\mathsf{unit}_{Z \cup Y \cup W} \oplus f_2) && \text{(By lemma C.2.4)} \\
&= f_1 \odot f_3 \odot (\mathsf{unit}_{Y \cup W} \oplus f_2) && \text{(By } \mathbf{dom}(f_2) = Z\text{)} \\
&= f_1 \odot (f_3^1 \odot f_3^2) \odot (\mathsf{unit}_{Y \cup W} \oplus f_2) \\
&= f_1 \odot f_3^1 \odot (f_3^2 \odot (\mathsf{unit}_{Y \cup W} \oplus f_2)) \\
&= f_1 \odot f_3^1 \odot ((f_2 \oplus \mathsf{unit}_W) \oplus f_3^2) && (\dagger)
\end{aligned}$$

where † follows from lemma C.2.3 and $\mathbf{dom}(f_2 \oplus \mathsf{unit}_W) = Z \cup W \subseteq \mathbf{range}(f_3^1)$. Thus, $f_1 \odot f_3^1 \odot ((f_2 \oplus \mathsf{unit}_W) \oplus f_3^2) \sqsubseteq f$.

Note that $f_1 \odot f_3^1$ has type $\mathbf{Mem}[\emptyset] \to \mathcal{D}(\mathbf{Mem}[Z \cup W])$, so $f_1 \odot f_3^1 \models (\emptyset \triangleright Z \cup W)$. State $f_2 \oplus \mathsf{unit}_W$ has type $\mathbf{Mem}[Z \cup W] \to \mathcal{D}(\mathbf{Mem}[Z \cup W \cup X])$, so $f_2 \oplus \mathsf{unit}_W \models (Z \cup W \triangleright X)$. State f_3^2 has type $\mathbf{Mem}[Z \cup W] \to \mathcal{D}(\mathbf{Mem}[Z \cup W \cup Y])$, so $f_3^2 \models (Z \cup W \triangleright Y)$. Therefore,

$$f_1 \odot f_3^1 \odot ((f_2 \oplus \mathsf{unit}_W) \oplus f_3^2) \models (\emptyset \triangleright Z \cup W) \fatsemi (Z \cup W \triangleright X) * (Z \cup W \triangleright Y).$$

By persistence, $f \models [Z \cup W] \fatsemi ([X] * [Y])$, and Weak Union is valid. □

Lemma C.2.10 (Contraction). *The following judgment is valid in $\mathcal{X}_{CI}$:*

$$\models ([Z] \fatsemi ([X] * [Y])) \wedge ([Z \cup Y] \fatsemi ([X] * [W])) \to [Z] \fatsemi ([X] * [Y \cup W])$$

Proof. If $h \models ([Z] \fatsemi ([X] * [Y])) \wedge ([Z \cup Y] \fatsemi ([X] * [W]))$, then

- $h \models [Z] \fatsemi ([X] * [Y])$. By the Classical flavor in intuitionistic model lemma, there exists f_1, f_2, f_3 such that $f_1 : \mathbf{Mem}[\emptyset] \to \mathcal{D}(\mathbf{Mem}[Z])$, $f_2 : \mathbf{Mem}[Z] \to \mathcal{D}(\mathbf{Mem}[Z \cup X])$, $f_3 : \mathbf{Mem}[Z] \to \mathcal{D}(\mathbf{Mem}[Z \cup Y])$, and $f_1 \odot (f_2 \oplus f_3) \sqsubseteq h$.

  Note $f_1 \odot (f_2 \oplus f_3)$ has type $\mathbf{Mem}[\emptyset] \to \mathcal{D}(\mathbf{Mem}[Z \cup Y \cup Z])$.

- $h \models [Z \cup Y] \fatsemi ([X] * [W])$. By lemma C.2.6, there exists g_1, g_2, g_3 such that $g_1 : \mathbf{Mem}[\emptyset] \to \mathcal{D}(\mathbf{Mem}[Z \cup Y])$, $g_2 : \mathbf{Mem}[Z \cup Y] \to \mathcal{D}(\mathbf{Mem}[Z \cup Y \cup X])$, $g_3 : \mathbf{Mem}[Z \cup Y] \to \mathcal{D}(\mathbf{Mem}[Z \cup Y \cup W])$, and $g_1 \odot (g_2 \oplus g_3) \sqsubseteq h$.

  Note $g_1 \odot g_2$ has type $\mathbf{Mem}[\emptyset] \to \mathcal{D}(\mathbf{Mem}[Z \cup Y \cup X])$.

By lemma C.2.2, $f_1 \odot (f_2 \oplus f_3) = g_1 \odot g_2$.

$$
\begin{aligned}
g_1 \odot (g_2 \oplus g_3) &= g_1 \odot (g_2 \oplus \mathsf{unit}_{Z \cup Y}) \odot (\mathsf{unit}_{Z \cup Y \cup X} \oplus g_3) && \text{(By lemma C.2.4)} \\
&= g_1 \odot g_2 \odot (\mathsf{unit}_{Z \cup X} \oplus g_3) \\
&\qquad\qquad \text{(Because } Z \cup Y \subseteq \mathbf{dom}(g_2),\ Y \subseteq \mathbf{dom}(g_3)\text{)} \\
&= f_1 \odot (f_2 \oplus f_3) \odot (\mathsf{unit}_{Z \cup X} \oplus g_3) && (f_1 \odot (f_2 \oplus f_3) = g_1 \odot g_2) \\
&= f_1 \odot \big((f_2 \odot \mathsf{unit}_{Z \cup X}) \oplus (f_3 \odot g_3)\big) && \text{(By C.1.8)} \\
&= f_1 \odot \big(f_2 \oplus (f_3 \odot g_3)\big)
\end{aligned}
$$

By their types, it is easy to see that $f_1 \models (\emptyset \triangleright Z)$, $f_2 \models (Z \triangleright X)$, $f_3 \odot g_3 \models (Z \triangleright Y \cup W)$. So,

$$f_1 \odot (f_2 \oplus (f_3 \odot g_3)) \models [Z] \fatsemi ([X] * [Y \cup W]).$$

Also, note that $h \sqsupseteq g_1 \odot (g_2 \oplus g_3) = f_1 \odot (f_2 \oplus (f_3 \odot g_3))$, so by persistence,

$$h \models (\emptyset \triangleright Z) \fatsemi ((Z \triangleright X) * (Z \triangleright Y \cup W)).$$ □

C.3 CPSL Assertion Logic

For the proof of Theorem 4.3.1, we need the following characterization of $g \sqsubseteq f$.

Proposition C.3.1. *Let f be a Markov kernel, and let $D \subseteq \mathbf{dom}(f) \subseteq R \subseteq \mathbf{range}(f)$. Then we have $\pi_R(f(m)) = g(m')$ for all $m' \in \mathbf{Mem}[D], m \in \mathbf{Mem}[\mathbf{dom}(f)]$ such that $m^D = m'$ if and only if $g \sqsubseteq f$ and $\mathbf{dom}(g) = D, \mathbf{range}(g) = R$.*

Proof. For the reverse direction, suppose that $f = (g \oplus \mathsf{unit}_S) \odot v$, with S disjoint from $\mathbf{dom}(g)$. Since $\mathbf{range}(g) \subseteq \mathbf{dom}(v)$, we have:

$$\begin{aligned}
\pi_R(f(m)) &= \pi_R((g \oplus \mathsf{unit}_S)(m)) \\
&= \pi_R(g(m^D) \oplus \mathsf{unit}_S(m^S)) \\
&= \pi_R(g(m^D)) \otimes \pi_R(\mathsf{unit}_S(m^S)) \\
&= g(m^D) \\
&= g(m').
\end{aligned}$$

For the forward direction, evidently $\mathbf{dom}(g) = D$ and $\mathbf{range}(g) = R$. Since f preserves input to output, we have $\pi_{\mathbf{dom}(f)}(g(m')) = \pi_{\mathbf{dom}(f)}(f(m)) = \mathsf{unit}(m')$ so g preserves input to output and g is a Markov kernel. We claim that $g \sqsubseteq f$. First, consider $g \oplus \mathsf{unit}_{\mathbf{dom}(f)\setminus D}$; write $D' = \mathbf{dom}(f) \setminus D$. For any $m \in \mathbf{Mem}[\mathbf{dom}(f)]$, we have:

$$\begin{aligned}
\pi_{D' \cup R}(f(m)) &= \pi_R(f(m)) \otimes \pi_{D'}(f(m)) \\
&= g(m^D) \otimes \mathsf{unit}_{D'}(m^{D'}) \\
&= (g \oplus \mathsf{unit}_{D'})(m).
\end{aligned}$$

So by lemma C.2.1, for every $m \in \mathbf{Mem}[\mathbf{dom}(f)]$ there exists a family of kernels $g'_m : \mathbf{Mem}[D' \cup R] \to \mathcal{D}(\mathbf{Mem}[\mathbf{range}(f)])$ such that

$$f(m) = \mathsf{bind}((g \oplus \mathsf{unit}_{D'})(m), g'_m)$$

Defining $g'(m) \triangleq g'_{m^{\mathbf{dom}(f)}}(m)$, we have:

$$f(m) = ((g \oplus \mathsf{unit}_{D'}) \odot g')(m)$$

and so $g \sqsubseteq f$. □

C.3.1 Restriction

Theorem 4.3.1 (Restriction in DIBI_+)**.** *Let $P \in DIBI_+$ with atomic propositions $(\phi \triangleright \psi)$, as described above. Then $f \models P$ if and only if there exists $f' \sqsubseteq f$ such that $\textbf{\textit{range}}(f') \subseteq FV(P)$ and $f' \models P$.*

Proof. The reverse direction is immediate from persistence. For the forward direction, we argue by induction with a stronger hypothesis. If $f \models P$, we call a state f' a *witness* of $f \models P$ if $f' \sqsubseteq f$, $\mathrm{FV_R}(P) \subseteq \mathbf{range}(f') \subseteq \mathrm{FV}(P)$, $\mathbf{dom}(f') \subseteq \mathrm{FV_D}(P)$, and $f' \models P$. We show that $f \models P$ implies that there is a witness $f' \models P$, by induction on P.

Case $(D \triangleright R)$**:** We will use two basic facts, both following from the form of the domain and range assertions:

1. If $m \models_d D$, then $dom(m) = \mathrm{FV}(D)$.
2. If $\mu \models_r R$, then $dom(\mu) \supseteq \mathrm{FV}(D)$.

$f \models (D \triangleright R)$ implies that there exists $f' \sqsubseteq f$ such that for any $m \in M_d$ such that $m \models_d D$, $f'(m)$ is defined and $f'(m) \models_r R$.

Let $T = \mathbf{range}(f') \cap (\mathrm{FV}(D) \cup \mathrm{FV}(R))$. We claim that $\pi_T f'$ is the desired witness for $f \models P$.

- $\pi_T f'$ is defined and $\pi_T f' \sqsubseteq f$ because:

$$\begin{aligned} \mathbf{dom}(f') &= dom(m) && \text{(for any } m \in M_d \text{ such that } m \models_d D) \\ &= \mathrm{FV}(D) \\ &\subseteq T. \end{aligned}$$

Thus $\pi_T f'$ is defined, and $\pi_T f' \sqsubseteq f' \sqsubseteq f$.

- $\mathbf{range}(\pi_T f') = T \subseteq \mathrm{FV}(D) \cup \mathrm{FV}(R) = \mathrm{FV}(P)$.
- $\pi_T f' \models (D \triangleright R)$: For any $m \in M_d$ such that $m \models_d D$, $f'(m)$ is a distribution. Based on the restriction theorem for probabilistic BI, $\pi_{\mathrm{FV}(R) \cap \mathbf{range}(f')}(f'(m)) \models R$ too. Since $T \supseteq \mathrm{FV}(R) \cap \mathbf{range}(f')$, persistence in M_r, implies $\pi_T(f'(m)) \models R$. By definition of marginalization on kernels, $(\pi_T f')(m) = \pi_T(f'(m))$. Since $(\pi_T f')(m) \models R$, we have $\pi_T f' \models (D \triangleright R)$ as well.
- $\mathrm{FV}_{\mathrm{D}}(P) = \mathrm{FV}(D)$, so $\mathbf{dom}(\pi_T f') = \mathbf{dom}(m) = \mathrm{FV}(D) = \mathrm{FV}_{\mathrm{D}}(P)$.
- $\mathrm{FV}_{\mathrm{R}}(P) = \mathrm{FV}(D \triangleright R) = \mathrm{FV}(D) \cup \mathrm{FV}(R)$, so

$$\begin{aligned}
\mathbf{range}(\pi_T f') &\supseteq \mathbf{dom}((\pi_T f')(m)) && \text{(for any } m \in M_d \text{ such that } m \models_d D\text{)} \\
&\supseteq \mathrm{FV}(D) \cup \mathrm{FV}(R) && \text{(By } (\pi_T f')(m) \models R\text{)} \\
&= \mathrm{FV}_{\mathrm{R}}(P).
\end{aligned}$$

so $\pi_T f'$ is a desired witness for $f \models P$.

Case $Q \wedge R$: Assuming $\mathrm{FV}_{\mathrm{R}}(Q) = \mathrm{FV}(Q) = \mathrm{FV}_{\mathrm{R}}(R) = \mathrm{FV}(R)$. By definition, $f \models Q \wedge R$ implies that $f \models Q$ and $f \models R$. By induction, there exists $f' \sqsubseteq f$ such that $\mathrm{FV}_{\mathrm{R}}(Q) = \mathbf{range}(f') = \mathrm{FV}(Q)$, $\mathbf{dom}(f') \subseteq \mathrm{FV}_{\mathrm{D}}(Q)$, and $f' \models Q$, and there exists $f'' \sqsubseteq f$ such that $\mathrm{FV}_{\mathrm{R}}(R) = \mathbf{range}(f'') = \mathrm{FV}(R)$, $\mathbf{dom}(f'') \subseteq \mathrm{FV}_{\mathrm{D}}(R)$ and $f'' \models R$. Thus, $\mathbf{range}(f') = \mathbf{range}(f'')$.

Note that $\mathbf{dom}(f') = \mathbf{dom}(f) \cap \mathbf{range}(f')$ because in our models, $f' \sqsubseteq f$ implies that there exists S and some v such that $f = (f' \oplus \eta_S) \odot v$, and we can make S disjoint of $\mathbf{dom}(f')$ and $\mathbf{range}(f')$ wolog. Then, $\mathbf{dom}(f) = \mathbf{dom}(f' \oplus S) = \mathbf{dom}(f') \cup S$, and $\mathbf{range}(f') = \mathbf{range}(f' \oplus S) \setminus S$, so $\mathbf{dom}(f) \cup \mathbf{range}(f') \subseteq \mathbf{dom}(f')$. Meanwhile, since $\mathbf{dom}(f') \subseteq \mathbf{dom}(f)$ and $\mathbf{dom}(f') \subseteq \mathbf{range}(f')$, $\mathbf{dom}(f') \subseteq \mathbf{dom}(f) \cap \mathbf{range}(f')$. So $\mathbf{dom}(f') =$

$\mathbf{dom}(f) \cap \mathbf{range}(f')$. Similarly, $\mathbf{dom}(f'') \subseteq \mathbf{dom}(f) \cap \mathbf{range}(f'')$, so $\mathbf{range}(f') = \mathbf{range}(f'')$ implies that $\mathbf{dom}(f') = \mathbf{dom}(f')$.

Since $\mathbf{dom}(f') = \mathbf{dom}(f'')$ and $\mathbf{range}(f') = \mathbf{range}(f'')$, proposition C.3.1 implies that $f' = f''$. This is the desired witness: $f' = f'' \models Q$ and $f' = f'' \models R$.

Case $Q \vee R$**:** $f \models Q \vee R$ implies that $f \models Q$ or $f \models R$.

Without loss of generality, suppose $f \models Q$. By induction, there exists $f' \sqsubseteq f$ such that $\mathrm{FV_R}(Q) \subseteq \mathbf{range}(f') \subseteq \mathrm{FV}(Q)$, $\mathbf{dom}(f') \subseteq \mathrm{FV_D}(Q)$. Then:

$$\mathbf{range}(f') \subseteq \mathrm{FV}(Q) \cup \mathrm{FV}(R) = \mathrm{FV}(P)$$

$$\mathbf{range}(f') \supseteq \mathrm{FV_R}(Q) \cap \mathrm{FV_R}(R) = \mathrm{FV_R}(P)$$

$$\mathbf{dom}(f') \subseteq \mathrm{FV}(Q) \cup \mathrm{FV}(R) = \mathrm{FV_D}(P).$$

Thus, f' is a desired witness.

Case $Q \fatsemi R$**:** Assuming $\mathrm{FV_D}(R) \subseteq \mathrm{FV_R}(Q)$.

$f \models Q \fatsemi R$ implies that there exists f_1, f_2 such that $f_1 \odot f_2 = f$, $f_1 \models Q$, and $f_2 \models R$. $f_1 \odot f_2$ is defined so $\mathbf{range}(f_1) = \mathbf{dom}(f_2)$. By induction, there exists $f_1' \sqsubseteq f_1$ such that $f_1' \models Q$, $\mathrm{FV_R}(Q) \subseteq \mathbf{range}(f_1') \subseteq \mathrm{FV}(Q)$ and $\mathbf{dom}(f_1') \subseteq \mathrm{FV_D}(Q)$, and there exists $f_2' \sqsubseteq f_2$ such that $f_2' \models Q$, $\mathrm{FV_R}(R) \subseteq \mathbf{range}(f_2') \subseteq \mathrm{FV}(R)$, and $\mathbf{dom}(f_2') \subseteq \mathrm{FV_D}(R)$.

Now, $\widehat{f} = f_1' \odot (f_2' \oplus \mathsf{unit}_{\mathbf{range}(f_1') \setminus \mathbf{dom}(f_2')})$ is defined because $\mathbf{dom}(f_2') \subseteq \mathrm{FV_D}(R) \subseteq \mathrm{FV_R}(Q) \subseteq \mathbf{range}(f_1')$. Then, we have

$$\widehat{f} \models Q \fatsemi R$$

$$\mathbf{range}(\widehat{f}) = \mathbf{range}(f_1') \cup \mathbf{range}(f_2') \subseteq \mathrm{FV}(Q) \cup \mathrm{FV}(R) = \mathrm{FV}(P)$$

$$\mathbf{range}(\widehat{f}) = \mathbf{range}(f_1') \cup \mathbf{range}(f_2') \supseteq \mathrm{FV_R}(Q) \cup \mathrm{FV_R}(R) = \mathrm{FV_R}(P)$$

$$\mathbf{dom}(\widehat{f}) = \mathbf{dom}(f_1') \subseteq \mathrm{FV_D}(Q) = \mathrm{FV_D}(P).$$

$f_1' \sqsubseteq f$, $f_2' \oplus \mathsf{unit}_{\mathbf{range}(f_1')\setminus\mathbf{dom}(f_2')} \oplus \sqsubseteq f_2$, so by lemma C.2.5, $\widehat{f} = f_1' \odot (f_2' \oplus \mathsf{unit}_{\mathbf{range}(f_1')\setminus\mathbf{dom}(f_2')}) \sqsubseteq f_1 \odot f_2 = f$.

Thus, $\widehat{f}$ is a desired witness.

Case $Q * R$**:** $f \models Q * R$ implies that there exists f_1, f_2 such that $f_1 \oplus f_2 \sqsubseteq f$, $f_1 \models Q$, and $f_2 \models R$.

By induction, there exists $f_1' \sqsubseteq f_1$ such that $f_1' \models Q$, $\mathrm{FV_R}(Q) \subseteq \mathbf{range}(f_1') \subseteq \mathrm{FV}(Q)$ and $\mathbf{dom}(f_1') \subseteq \mathrm{FV_D}(Q)$, and there exists $f_2' \sqsubseteq f_2$ such that $f_2' \models Q$, $\mathrm{FV_R}(R) \subseteq \mathbf{range}(f_2') \subseteq \mathrm{FV}(R)$, and $\mathbf{dom}(f_2') \subseteq \mathrm{FV_D}(R)$. By downwards closure of $\oplus$, $f_1' \oplus f_2'$ is defined and $f_1' \oplus f_2' \sqsubseteq f_1 \oplus f_2 \sqsubseteq f$. We have $f_1' \oplus f_2' \models Q * R$, and

$$\mathbf{range}(f_1' \oplus f_2') = \mathbf{range}(f_1') \cup \mathbf{range}(f_2') \subseteq \mathrm{FV}(Q) \cup \mathrm{FV}(R) = \mathrm{FV}(P)$$

$$\mathbf{range}(f_1' \oplus f_2') = \mathbf{range}(f_1') \cup \mathbf{range}(f_2') \supseteq \mathrm{FV_R}(Q) \cup \mathrm{FV_R}(R) = \mathrm{FV_R}(P)$$

$$\mathbf{dom}(f_1' \oplus f_2') = \mathbf{dom}(f_1') \cup \mathbf{dom}(f_2') \subseteq \mathrm{FV_D}(Q) \cup \mathrm{FV_D}(R) = \mathrm{FV_D}(P).$$

Thus, $f_1' \oplus f_2'$ is a desired witness.

□

C.3.2 Extra Axioms

Proposition 4.3.2. *The following axiom schemas for atomic propositions are valid.*

$$(S : p_d \triangleright p_r) \wedge (S : p'_d \triangleright p'_r) \to (S : p_d \wedge p'_d \triangleright p_r \wedge p'_r) \qquad \text{if } FV(p_r) = FV(p'_r)$$

(AP-AND)

$$(S : p_d \triangleright p_r) \wedge (S : p'_d \triangleright p'_r) \to (S : p_d \vee p'_d \triangleright p_r \vee p'_r) \qquad \text{(AP-OR)}$$

$$(S : p_d \triangleright p_r) * (S' : p'_d \triangleright p'_r) \to (S \cup S' : p_d \wedge p'_d \triangleright p_r * p'_r) \qquad \text{(AP-PAR)}$$

$$p'_d \to p_d \text{ and } \models_r p_r \to p'_r \text{ implies } \models (S : p_d \triangleright p_r) \to (S : p'_d \triangleright p'_r) \qquad \text{(AP-IMP)}$$

Proof. We check each of the axioms.

Case: AP-AND. Suppose that $w \models (S : p_d \triangleright p_r) \wedge (S : p'_d \triangleright p'_r)$. By semantics of atomic propositions, there exists $w_1 \sqsubseteq_k w$ and $w_2 \sqsubseteq_k w$ such that for all $m \in \mathbf{Mem}[S]$ such that $m \models_d p_d \wedge p'_d$, we have $w_1(m) \models_r p_r$ and $w_2(m) \models_r p'_r$. By restriction (theorem 4.3.1), we may assume that $\mathbf{range}(w_1) = \mathrm{FV}(p_r) = \mathrm{FV}(p'_r) = \mathbf{range}(w_2)$. Thus, proposition C.3.1 implies that $w_1 = w_2$, and so $w \models (S : p_d \wedge p'_d \triangleright p_r \wedge p'_r)$.

Case: AP-OR. Immediate, by semantics of $\vee$.

Case: AP-PAR. Suppose that $w \models (S : p_d \triangleright p_r) * (S' : p'_d \triangleright p'_r)$. We will show that $w \models (S \cup S' : p_d * p'_d \triangleright p_r * p'_r)$.

By semantics of atomic propositions, there exists $w_1 \sqsubseteq_k w$ and $w_2 \sqsubseteq_k w$ such that $w_1 \oplus w_2 \sqsubseteq w$, and for all $m_1 \in \mathbf{Mem}[S]$ such that $m_1 \models_d p_d$, we have $w_1(m_1) \models_r p_r$, and for all $m_2 \in \mathbf{Mem}[S']$ such that $m_2 \models_d p'_d$, we have $w_2(m_2) \models_r p'_r$.

Now for any $m \in \mathbf{Mem}[S \cup S']$ such that $m \models_d p_d \wedge p'_d$, we have $m^S \models_d p_d$ and $m^{S'} \models_d p'_d$. Thus $w_1(m^S) \models_r p_r$ and $w_2(m^{S'}) \models_r p'_r$. Letting $T = S \cap S'$

and $T_1 = S \setminus T$; $T_2 = S' \setminus T$ be disjoint sets, and noting that w_1, w_2 both preserve inputs on T, we have:

$$\begin{aligned} w_1 \oplus w_2(m) &= \pi_{T_1} w_1(m^S) \otimes \mathsf{unit}(m^T) \otimes \pi_{T_2} w_2(m^{S'}) \\ &= (\pi_{T_1} w_1(m^S) \otimes \mathsf{unit}(m^T)) \oplus_r (\mathsf{unit}(m^T) \otimes \pi_{T_2} w_2(m^{S'})) \\ &= w_1(m^S) \oplus_r w_2(m^{S'}) \\ &\models_r p_r * p'_r \end{aligned}$$

Thus, $w \models (S \cup S' : p_d * p'_d \triangleright p_r * p'_r)$.

Case: AP-IMP. Immediate, by semantics of $\rightarrow$.

□

Proposition 4.3.4. *(AXIOMS FOR DIBI$_+$) The following axioms are sound, assuming both precedent and antecedent are in DIBI$_+$.*

$$(P \fatsemi Q) \fatsemi R \rightarrow P \fatsemi (Q * R) \qquad \text{(INDEP-1)}$$

$$P \fatsemi Q \rightarrow P * Q \qquad \text{if } FV_D(Q) = \emptyset \qquad \text{(INDEP-2)}$$

$$P \fatsemi Q \rightarrow P \fatsemi (Q * (S \triangleright [S])) \qquad \text{(PAD)}$$

$$(P * Q) \fatsemi (R * S) \rightarrow (P \fatsemi R) * (Q \fatsemi S) \qquad \text{(RESTEXCH)}$$

Proof. We prove them one by one.

INDEP-1 We want to show that when $(P \fatsemi Q) \fatsemi R$, $P \fatsemi (Q * R)$ are both formula in DIBI$_+$, $f \models (P \fatsemi Q) \fatsemi R$ implies $f \models P \fatsemi (Q * R)$.

By proof system of DIBI, $f \models (P \fatsemi Q) \fatsemi R$ implies that $f \models P \fatsemi (Q \fatsemi R)$. While $P \fatsemi (Q \fatsemi R)$ may not satisfy the restriction property, that is okay because we will only used conditions guaranteed by the fact that $(P \fatsemi Q) \fatsemi R$, $P \fatsemi (Q *$

$R) \in \mathrm{DIBI}_+$. In particular, we rely on P, Q, R each satisfies restriction, and $\mathrm{FV}_\mathrm{D}(Q * R) \subseteq \mathrm{FV}_\mathrm{R}(P)$, which implies that

$$\mathrm{FV}_\mathrm{D}(R) \subseteq \mathrm{FV}_\mathrm{D}(Q * R) \subseteq \mathrm{FV}_\mathrm{R}(P) \tag{C.17}$$

$f \models P \fatsemi (Q \fatsemi R)$ implies there exists f_p, f_q, f_r such that $f_p \models P$, $f_q \models Q$, and $f_r \models R$, and $f_p \odot (f_q \odot f_r) = f$.

By restriction property theorem 4.3.1, $f_q \models Q$ implies that there exists $f'_q \sqsubseteq f_q$ such that $\mathrm{FV}_\mathrm{R}(Q) \subseteq \mathbf{range}(f'_q) \subseteq \mathrm{FV}(Q)$ and $\mathbf{dom}(f'_q) \subseteq \mathrm{FV}_\mathrm{D}(Q)$. $f'_q \sqsubseteq f_q$ so there exists v, T such that $f_q = (f'_q \oplus_k \mathsf{unit}_T) \odot v$.

Similarly, $f_r \models R$, by theorem 4.3.1, there exists $f'_r \sqsubseteq f_r$ such that $\mathrm{FV}_\mathrm{R}(R) \subseteq \mathbf{range}(f'_r) \subseteq \mathrm{FV}(R)$ and $\mathbf{dom}(f'_r) \subseteq \mathrm{FV}_\mathrm{D}(R)$. $f'_r \sqsubseteq f_r$ so there exists u, S such that $f_r = (f'_r \oplus_k \mathsf{unit}_S) \odot u$.

Now, we claim that $\mathrm{FV}_\mathrm{D}(R) \subseteq \mathbf{dom}(f'_q \oplus \mathsf{unit}_T)$:

By theorem 4.3.1 $f_p \models P$ implies that there exists $f'_p \sqsubseteq f_p$ such that $\mathrm{FV}_\mathrm{R}(P) \subseteq \mathbf{range}(f'_p) \subseteq \mathrm{FV}(P)$, $\mathbf{dom}(f'_p) \subseteq F\mathrm{FV}(P)$, and $f'_p \models P$. Thus, $\mathrm{FV}_\mathrm{R}(P) \subseteq \mathbf{range}(f_p) = \mathbf{dom}(f_q)$.

Recall that $\mathrm{FV}_\mathrm{D}(R) \subseteq \mathrm{FV}_\mathrm{R}(P)$, so $\mathrm{FV}_\mathrm{D}(R) \subseteq \mathbf{dom} f_q = \mathbf{dom} f'_q \oplus \mathsf{unit}_T$.

As a corollary, we have $\mathbf{dom}(f'_r) \subseteq \mathrm{FV}_\mathrm{D}(R) \subseteq \mathbf{dom}(f'_q \oplus \mathsf{unit}_T) \subseteq \mathbf{dom}(v)$,

and $\mathbf{dom}(f'_r) \subseteq \mathrm{FV}_{\mathrm{D}}(R) \subseteq \mathbf{dom}(f'_q \oplus \mathsf{unit}_T)$. Then,

$$
\begin{aligned}
f_q \odot f_r &= \big((f'_q \oplus \mathsf{unit}_T) \odot v\big) \odot \big((f'_r \oplus \mathsf{unit}_S) \odot u\big) \\
&= (f'_q \oplus \mathsf{unit}_T) \odot \big(v \odot (f'_r \oplus \mathsf{unit}_S)\big) \odot u \\
&\qquad\qquad \text{(By standard associativity of } \odot) \\
&= (f'_q \oplus \mathsf{unit}_T) \odot (f'_r \oplus v) \odot u \\
&\qquad\qquad \text{(By lemma C.2.3 and } \mathbf{dom}(f'_r) \subseteq \mathbf{dom}(v)) \\
&= (f'_q \oplus \mathsf{unit}_T) \odot ((f'_r \odot \mathsf{unit}_{\mathbf{range}(f'_r)}) \oplus (\mathsf{unit}_{\mathbf{dom}(v)} \odot v) \odot u \\
&= (f'_q \oplus \mathsf{unit}_T) \odot (f'_r \oplus \mathsf{unit}_{\mathbf{dom}(v)}) \odot (\mathsf{unit}_{\mathbf{range}(f'_r)} \oplus v) \odot u \qquad (\heartsuit) \\
&= ((f'_q \oplus \mathsf{unit}_T) \oplus f'_r) \odot (v \oplus \mathsf{unit}_{\mathbf{range}(f'_r)}) \odot u \qquad (\dagger) \\
&= ((f'_q \oplus \mathsf{unit}_T) \odot v) \oplus (f'_r \odot \mathsf{unit}_{\mathbf{range}(f'_r)}) \odot u \qquad (\heartsuit) \\
&= f_q \oplus f_r
\end{aligned}
$$

where † follows from lemma C.2.3, $\mathbf{dom}(f'_r) \subseteq \mathbf{dom}(f'_q \oplus \mathsf{unit}_T)$ and exact commutativity, ♡ follows from lemma C.1.8 and lemma C.1.9.

Thus, $f_q \odot f_r \models Q * R$. And by satisfaction rules,

$$f \models P \fatsemi (Q * R)$$

INDEP-2 We want to show that under the special condition $\mathrm{FV}_{\mathrm{D}}(Q) = \emptyset$, $f \models P \fatsemi Q$ implies that $f \models P * Q$.

If $f \models P \fatsemi Q$, then there exists f_p, f_q such that $f_p \odot f_q = f$ and $f_p \models P$, $f_q \models Q$.

By restriction property theorem 4.3.1, $f_q \models Q$ implies that there exists $f'_q \sqsubseteq f_q$ such that $\mathrm{FV}_{\mathrm{R}}(Q) \subseteq \mathbf{range}(f'_q) \subseteq \mathrm{FV}(Q)$ and $\mathbf{dom}(f'_q) \subseteq \mathrm{FV}_{\mathrm{D}}(Q)$. $f'_q \sqsubseteq f_q$ so there exists v, T such that $f_q = (f'_q \oplus_k \mathsf{unit}_T) \odot v$.

Since $\mathbf{dom}(f'_q) \subseteq \mathrm{FV}_{\mathrm{D}}(Q)$ and $\mathrm{FV}_{\mathrm{D}}(Q) = \emptyset$, it must $\mathbf{dom}(f'_q) = \emptyset$, and thus

no matter what the domain of f_p is, $\mathbf{dom}(f'_q) \subseteq \mathbf{dom}(f_p)$. Thus,

$$\begin{aligned} f_p \odot f_q &= f_p \odot (f'_q \oplus \mathsf{unit}_T) \odot v \\ &= (f_p \oplus f'_q) \oplus v \qquad \text{(By lemma C.2.3 and } \mathbf{dom}(f'_q) \subseteq \mathbf{dom}(f_p)) \end{aligned}$$

Thus, $f_p \oplus f'_q \sqsubseteq f_p \odot f_q = f$. By satisfaction rules, $f_p \models P$ and $f'_q \models Q$ implies that $f_p \oplus f'_q \models P * Q$. Thus, by persistence, $f \models P * Q$

PAD We want to show that when $P \fatsemi Q$, $P \fatsemi (Q * (S \triangleright S))$ are both in DIBI$_+$, $f \models P \fatsemi Q$ implies $f \models P \fatsemi (Q * (S \triangleright S))$.

One key guarantee we rely on from the grammar of DIBI$_+$ is that

$$\mathrm{FV}_{\mathrm{D}}(Q) \cup S = \mathrm{FV}_{\mathrm{D}}(Q * (S \triangleright S)) \subseteq \mathrm{FV}_{\mathrm{R}}(P).$$

When $f \models P \fatsemi Q$, there exists f_p, f_q such that $f_p \odot f_q = f$ and $f_p \models P$, $f_q \models Q$, By theorem 4.3.1, $f_p \models P$ implies that there exists $f'_p \sqsubseteq f_p$ such that $\mathrm{FV}_{\mathrm{R}}(P) \subseteq \mathbf{range}(f'_p) \subseteq \mathrm{FV}(P)$, $\mathbf{dom}(f'_p) \subseteq F\mathrm{FV}(P)$, and $f'_p \models P$. By the fact that $f_p \odot f_q$ is defined, and that the definition of preorder in our concrete models, $f'_p \sqsubseteq f_p$ implies

$$\mathbf{dom}(f_q) = \mathbf{range}(f_p) \supseteq \mathbf{range}(f'_p) \supseteq \mathrm{FV}_{\mathrm{R}}(P) \supseteq S$$

Since f_q preserves input, $S \subseteq \mathbf{dom}(f_q)$ implies that $f_q = f_q \oplus \mathsf{unit}_S$, and thus $f_p \odot f_q = f_p \odot (f_q \oplus \mathsf{unit}_S)$.

Note that $\mathsf{unit}_S \models (S \triangleright [S])$, and $f_q \models Q$. Thus, $f_q \oplus \mathsf{unit}_S \models Q * (S \triangleright [S])$. Since $f_p \models P$, it follows that

$$f_p \odot (f_q \oplus \mathsf{unit}_S) \models P \fatsemi (Q * (S \triangleright [S]))$$

Since $f = f_p \odot f_q = f_p \odot (f_q \oplus \mathsf{unit}_S)$,

$$f \models P \fatsemi (Q * (S \triangleright [S]))$$

RESTEXCH We want to show that when $(P * Q) \fatsemi (R * S)$ and $(P \fatsemi R) * (Q \fatsemi S)$ are both formula in DIBI$_+$, $f \models (P * Q) \fatsemi (R * S)$ implies $f \models (P * R) * (Q * S)$.

The key properties that being in DIBI$_+$ guarantees us are that

$$\mathrm{FV}_\mathrm{D}(R) \subseteq \mathrm{FV}_\mathrm{R}(P) \qquad \mathrm{FV}_\mathrm{D}(S) \subseteq \mathrm{FV}_\mathrm{R}(Q)$$

$$\mathrm{FV}_\mathrm{D}(R * S) = \mathrm{FV}_\mathrm{D}(R) \cup \mathrm{FV}_\mathrm{D}(S) \subseteq \mathrm{FV}_\mathrm{R}(P * Q) = \mathrm{FV}_\mathrm{R}(P) \cup \mathrm{FV}_\mathrm{R}(Q)$$

If $f \models (P * Q) \fatsemi (R * S)$, then there exists f_1, f_2 such that $f_1 \odot f_2 = f$, $f_1 \models P * Q$, $f_2 \models R * S$. That is, there exist u_1, v_1 such that $u_1 \oplus v_1 \sqsubseteq f_1$, $u_1 \models P$, and $v_1 \models Q$; there exist u_2, v_2 such that $u_2 \oplus v_2 \sqsubseteq f_2$, $u_2 \models R$, $v_2 \models S$.

By theorem 4.3.1,

- $u_1 \models P$ implies there exists $u_1' \sqsubseteq u_1$ such that $\mathrm{FV}_\mathrm{R}(P) \subseteq \mathbf{range}(u_1') \subseteq \mathrm{FV}(P)$, $\mathbf{dom}(u_1') \subseteq \mathrm{FV}_\mathrm{D}(P)$, and $u_1' \models P$.
- $v_1 \models Q$ implies there exists $v_1' \sqsubseteq v_1$ such that $\mathrm{FV}_\mathrm{R}(Q) \subseteq \mathbf{range}(v_1') \subseteq \mathrm{FV}(Q)$, $\mathbf{dom}(v_1') \subseteq \mathrm{FV}_\mathrm{D}(Q)$, and $v_1' \models Q$.
- $u_2 \models R$ implies there exists $u_2' \sqsubseteq u_2$ such that $\mathrm{FV}_\mathrm{R}(R) \subseteq \mathbf{range}(u_2') \subseteq \mathrm{FV}(R)$, $\mathbf{dom}(u_2') \subseteq \mathrm{FV}_\mathrm{D}(R)$, and $u_2' \models R$.
- $v_2 \models S$ implies there exists $v_2' \sqsubseteq v_2$ such that $\mathrm{FV}_\mathrm{R}(S) \subseteq \mathbf{range}(v_2') \subseteq \mathrm{FV}(S)$, $\mathbf{dom}(v_2') \subseteq \mathrm{FV}_\mathrm{D}(S)$, and $v_2' \models S$.

By Downwards closure property of $\oplus$, $u_2' \oplus v_2'$ is defined and $u_2' \oplus v_2' \sqsubseteq u_2 \oplus v_2 \sqsubseteq f_2$. Say that $f_1 = (u_1 \oplus v_1 \oplus \mathsf{unit}_{S_1}) \odot h_1$, $f_2 = (u_2' \oplus v_2' \oplus \mathsf{unit}_{S_2}) \odot h_2$. Also,

$$\begin{aligned}\mathbf{dom}(u_2' \oplus v_2') = \mathbf{dom}(u_2') \cup \mathbf{dom}(v_2') &\subseteq \mathrm{FV}_\mathrm{D}(R) \cup \mathrm{FV}_\mathrm{D}(S) \subseteq \mathrm{FV}_\mathrm{R}(P) \cup \mathrm{FV}_\mathrm{D}(Q) \\ &\subseteq \mathbf{range}(u_1') \cup \mathbf{range}(v_1') \subseteq \mathbf{range}(u_1) \cup \mathbf{range}(v_1) = \mathbf{range}(u_1 \oplus v_1)\end{aligned}$$

Then

$$
\begin{aligned}
f_1 \odot f_2 &= (u_1 \oplus v_1 \oplus \mathsf{unit}_{S_1}) \odot h_1 \odot (u_2' \oplus v_2' \oplus \mathsf{unit}_{S_2}) \odot h_2 \\
&= (u_1 \oplus v_1 \oplus \mathsf{unit}_{S_1}) \odot ((u_2' \oplus v_2') \oplus h_1) \odot h_2 && (\heartsuit) \\
&= (u_1 \oplus v_1 \oplus \mathsf{unit}_{S_1}) \odot ((u_2' \oplus v_2') \odot \mathsf{unit}_{\mathbf{range}(u_2' \oplus v_2')}) \oplus (\mathsf{unit}_{\mathbf{dom}(h_1)} \odot h_1) \odot h_2 \\
&= (u_1 \oplus v_1 \oplus \mathsf{unit}_{S_1}) \odot (u_2' \oplus v_2' \oplus \mathsf{unit}_{\mathbf{dom}(h_1)}) \odot (\mathsf{unit}_{\mathbf{range}(u_2' \oplus v_2')} \oplus h_1) \odot h_2 \\
& && (\dagger) \\
&= (u_1 \oplus v_1 \oplus \mathsf{unit}_{S_1}) \odot (u_2' \oplus v_2' \oplus \mathsf{unit}_{\mathbf{range}(u_1 \oplus v_1)} \oplus \mathsf{unit}_{S_1}) \odot (\mathsf{unit}_{\mathbf{range}(u_2' \oplus v_2')} \oplus h_1) \odot h_2 \\
&= \big(((u_1 \oplus v_1) \odot (u_2' \oplus v_2' \oplus \mathsf{unit}_{\mathbf{range}(u_1 \oplus v_1)})) \oplus \mathsf{unit}_{S_1}\big) \odot (\mathsf{unit}_{\mathbf{range}(u_2' \oplus v_2')} \oplus h_1) \odot h_2 \\
& && (\dagger) \\
&= \big((u_1 \odot (u_2' \oplus \mathsf{unit}_{\mathbf{range}(u_1)})) \oplus (v_1 \odot (v_2' \oplus \mathsf{unit}_{\mathbf{range}(v_1)})) \oplus \mathsf{unit}_{S_1}\big) \\
&\odot (\mathsf{unit}_{\mathbf{range}(u_2' \oplus v_2')} \oplus h_1) \odot h_2 \\
& && (\dagger \text{ and exact commutativity, associativity})
\end{aligned}
$$

where $\heartsuit$ follows from lemma C.2.3, $\mathbf{dom}(u_2' \oplus v_2') \subseteq \mathbf{range}(u_1 \oplus v_1) \subseteq \mathbf{dom}(h_1)$, and $\dagger$ follows from lemma C.1.8 and lemma C.1.9.

Thus, $(u_1 \odot (u_2' \oplus \mathsf{unit}_{\mathbf{range}(u_1)})) \oplus (v_1 \odot (v_2' \oplus \mathsf{unit}_{\mathbf{range}(v_1)})) \sqsubseteq f_1 \odot f_2$. Recall that $u_2' \models R$. By persistence, $u_2' \oplus \mathsf{unit}_{\mathbf{range}(u_1)} \models R$. Similarly, $v_2' \models S$, so by persistence, $v_2' \oplus \mathsf{unit}_{\mathbf{range}(v_1)} \models S$. Therefore,

$$(u_1 \odot (u_2' \oplus \mathsf{unit}_{\mathbf{range}(u_1)})) \oplus (v_1 \odot (v_2' \oplus \mathsf{unit}_{\mathbf{range}(v_1)})) \models (P \fatsemi R) * (Q \fatsemi S)$$

Then, by persistence, $f \models (P \fatsemi R) * (Q \fatsemi S)$.

□

Proposition 4.3.5. *(AXIOMS FOR ATOMIC PROPOSITIONS) The following axioms*

are sound. For any $S, A, B, C \subseteq \mathbf{Var}$,

$$(S \triangleright [A] * [B]) \rightarrow (S \triangleright [A]) * (S \triangleright [B]) \qquad \text{if } A \cap B \subseteq S \qquad \text{(RevPar)}$$

$$(S \triangleright [A] * [B]) \rightarrow (S \triangleright [A \cup B]) \qquad \text{(UnionRan)}$$

$$(A \triangleright B) \fatsemi (B \triangleright C) \rightarrow (A \triangleright C) \qquad \text{(AtomSeq)}$$

$$(A \triangleright B) \rightarrow (A \triangleright A) \fatsemi (A \triangleright B) \qquad \text{(UnitL)}$$

$$(A \triangleright B) \rightarrow (A \triangleright B) \fatsemi (B \triangleright B) \qquad \text{(UnitR)}$$

Proof. We prove it one by one.

RevPar Given any $f \models (S \triangleright [A] * [B])$, by satisfaction rules and semantic of atomic propositions, there exists $f' \sqsubseteq f$ such that for all $m \in M_d$ such that $m \models_d S$, $f'(m) \models_r [A] * [B]$.

Since $f'(m)$ is defined and $f'(m) \models_r [A] * [B]$, it follows that $\mathbf{dom}(f') = S$ and $\mathbf{range}(f') \supseteq S \cup A \cup B$. Thus, we can define $f_1 = \pi_{S \cup A} f'$, $f_2 = \pi_{S \cup B} f'$. Note that $f_1 \models (S \triangleright A)$, $f_2 \models (S \triangleright B)$. Also, because $A \cap B \subseteq S$,

$$\mathbf{range}(f_1) \cap \mathbf{range}(f_2) = (S \cup A) \cap (S \cup B) = S,$$

and thus $f_1 \oplus f_2$ is defined. We now want to show that $f_1 \oplus f_2 \sqsubseteq f$.

Note $f'(m) \models_r [A] * [B]$ implies that there exists μ_1, μ_2 such that $\mu_1 \oplus_r \mu_2 \sqsubseteq f'(m)$, and $dom(\mu_1) \supseteq A$, $dom(\mu_2) \supseteq B$. Since f' preserves input on its domain S, $\pi_S f'(m) = \mathsf{unit}(m)$, so $(\mu_1 \oplus_r \mathsf{unit}(m)) \oplus_r (\mu_2 \oplus_r \mathsf{unit}(m)) \sqsubseteq f'(m) \oplus_r \mathsf{unit}(m) \oplus_r \mathsf{unit}(m) = f'(m)$ too. Let $\mu_1' = \pi_{A \cup S}(\mu_1 \oplus_r \mathsf{unit}(m))$ and $\mu_2' = \pi_{B \cup S}(\mu_2 \oplus_r \mathsf{unit}(m))$. Then due to Downwards closure in M_d, $\mu_1' \oplus_r \mu_2'$ will also be defined, and

$$\mu_1' \oplus_r \mu_2' \sqsubseteq (\mu_1 \oplus_r \mathsf{unit}(m)) \oplus_r (\mu_2 \oplus_r \mathsf{unit}(m)) \sqsubseteq f'(m),$$

which implies that $\mu'_1 \oplus_r \mu'_2 = \pi_{S\cup A\cup B} f'(m)$. In the range model, this means that $\mu'_1 = \pi_{S\cup A} f'(m)$, $\mu'_2 = \pi_{S\cup B} f'(m)$.

Then for any $m' \in \mathbf{Mem}[S]$, any $r \in \mathbf{Mem}[A \cup B \cup S]$,

$$
\begin{aligned}
(\pi_{S\cup A\cup B} f')(m')(r) = (\pi_{S\cup A\cup B} f'(m'))(r) = \mu'_1 \oplus_r \mu'_2(r) = \mu'_1(r^{S\cup A}) \cdot \mu'_2(r^{S\cup B}) \\
(f_1 \oplus f_2)(m')(r) = f_1(m')(r^{S\cup A}) \cdot f_2(m')(r^{S\cup B}) \\
= (\pi_{S\cup A} f')(m')(r^{S\cup A}) \cdot (\pi_{S\cup B} f'(m')(r^{S\cup B}) \\
= \mu'_1(r^{S\cup A}) \cdot \mu'_2(r^{S\cup B})
\end{aligned}
$$

Thus, $f_1 \oplus f_2 = \pi_{S\cup A\cup B} f'$, which implies that $f_1 \oplus f_2 \sqsubseteq f$. By their types, $f_1 \oplus f_2 \models (S \triangleright A) * (S \triangleright B)$.

By persistence, $f \models (S \triangleright A) * (S \triangleright B)$.

UnionRan Obvious from the semantics of atomic proposition and the range logic.

AtomSeq Given any $f \models (A \triangleright B) \fatsemi (B \triangleright C)$, by satisfaction rules and semantic of atomic propositions, there exists

- f_1, f_2 such that $f_1 \odot f_2 = f$;
- $f'_1 \sqsubseteq f_1$ such that for any $m \in M_d$ such that $m \models_d A$, $f'_1(m) \models_r [B]$.
- $f'_2 \sqsubseteq f_2$ such that for any $m \in M_d$ such that $m \models_d B$, $f'_2(m) \models_r \mathcal{D}[C]$.

Note that $f'_1(m) \models_r [B]$ implies that $B \subseteq \mathbf{range}(f'_1)$, so $\pi_B f'_1$ is defined. Let $f''_1 = \pi_B f'_1$.

Note that for any $m \in M_d$ such that $m \models_d A$, $f''_1(m) \models_r [B]$ too, so $f'' \models (A \triangleright B)$ too. Also, by transitivity, $f''_1 \sqsubseteq f'_1 \sqsubseteq f_1$.

Say $f_1 = (f_1'' \oplus \eta_{S_1}) \odot v_1$, $f_2 = (f_2' \oplus \eta_{S_2}) \odot v_2$, then since $\mathbf{range}(f_1'') = B = \mathbf{dom}(f_2')$,

$$
\begin{aligned}
f_1 \odot f_2 &= (f_1'' \oplus \eta_{S_1}) \odot v_1 \odot (f_2' \oplus \eta_{S_2}) \odot v_2 \\
&= (f_1'' \oplus \eta_{S_1}) \odot (f_2' \oplus v_1) \odot v_2 \\
&\qquad \text{(By lemma C.2.3 and } \mathbf{dom}(f_2') = B = \mathbf{range}(f_1'') \subseteq \mathbf{dom}(v_1)\text{)} \\
&= (f_1'' \oplus \eta_{S_1}) \odot (f_2' \oplus \eta_{\mathbf{dom}(v_1)}) \odot (v_1 \oplus \eta_{\mathbf{range}(f_1)}) \odot v_2 \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(By lemma C.2.4)} \\
&= (f_1'' \oplus \eta_{S_1}) \odot (f_2' \oplus \eta_S) \odot (v_1 \oplus \eta_{\mathbf{range}(f_1)}) \odot v_2 \\
&= ((f_1'' \odot f_2') \oplus \eta_{S_1}) \odot (v_1 \oplus \eta_{\mathbf{range}(f_1)}) \odot v_2
\end{aligned}
$$

So $f_1'' \odot f_2' \sqsubseteq f_1 \odot f_2 = f$.

$f_1'' \colon \mathbf{Mem}[A] \to \mathcal{D}(\mathbf{Mem}[B])$, $f_2' \colon \mathbf{Mem}[B] \to \mathcal{D}(\mathbf{Mem}[\mathbf{range}(f_2')])A$, so $f_1'' \odot f_2' \colon \mathbf{Mem}[A] \to \mathcal{D}(\mathbf{Mem}[\mathbf{range}(f_2')])$. Since $\mathbf{range}(f_2') \supseteq C$, it follows that $f_1'' \odot f_2' \models (A \triangleright C)$, and thus $f \models (A \triangleright C)$ by persistence.

UNITL If $f \models (A \triangleright B)$, then there must exists $f' \sqsubseteq f$ such that for all $m \in M_d$ such that $m \models A$, $f'(m) \models_r [B]$.

Given any witness f', $f' = \mathsf{unit}_{\mathbf{Mem}[A]} \odot f'$, and also $f' \models_r (A \triangleright B)$.

Note that $\mathsf{unit}_{\mathbf{Mem}[A]} \models_r (A \triangleright A)$, so $f' = \mathsf{unit}_{\mathbf{Mem}[A]} \odot f' \models (A \triangleright A) \fatsemi (A \triangleright B)$.

UNITR Analogous as the UNITL case, except that now using the fact $f' = f' \odot \mathsf{unit}_{\mathbf{Mem}[B]}$ for any $f' \colon \mathbf{Mem}[A] \to \mathcal{D}(\mathbf{Mem}[B])$.

□

C.4 CPSL Soundness

Definition 4.3.6 (CPSL Validity)**.** A CPSL judgment $\{P\}\ c\ \{Q\}$ is *valid*, written $\models \{P\}\ c\ \{Q\}$, if for every input distribution $\mu \in \mathcal{D}(\mathbf{Mem}[\mathbf{Var}])$ such that the lifted input $f_\mu \triangleq \langle\rangle \mapsto \mu$ satisfies $f_\mu \models P$, the lifted output satisfies $f_{[\![c]\!]\mu} \models Q$.

Now, we are ready to prove soundness of CPSL.

Theorem 4.3.3 (CPSL Soundness)**.** *CPSL is sound: derivable judgments are valid.*

Proof. By induction on the derivation. Throughout, we write $\mu : \mathbf{Mem}[\mathbf{Var}]$ for the input distribution and $f : \mathbf{Mem}[\emptyset] \to \mathcal{D}(\mathbf{Mem}[\mathbf{Var}])$ for the kernel obtained by lifting the input distribution, and we assume that f satisfies the pre-condition of the conclusion.

Assn By restriction (theorem 4.3.1), there exists $k_1 \sqsubseteq f$ such that

$$\mathrm{FV}(e) \subseteq \mathrm{FV}_{\mathrm{R}}(P) \subseteq \mathbf{range}(k_1) \subseteq \mathrm{FV}(P).$$

Since f has empty domain, we have $f = k_1 \odot k_2$ for some $k_2 : \mathbf{Mem}[\mathbf{range}(k_1)] \to \mathcal{D}(\mathbf{Mem}[\mathbf{Var}])$. Let $f' = \langle\rangle \mapsto [\![x \leftarrow e]\!](\mu)$ be the lifted output. By the semantics of the program and associativity, we have:

$$\begin{aligned} f' &= \langle\rangle \mapsto \mathsf{bind}(\mu, m \mapsto \mathsf{unit}(m[x \mapsto [\![e]\!](m)])) \\ &= \pi_{\mathbf{Var}\setminus\{x\}} f \odot (\underbrace{m_1 \mapsto \mathsf{unit}(m_1 \cup (x \mapsto [\![e]\!](m_1)))}_{g_1} \oplus \underbrace{m_2 \mapsto \mathsf{unit}(m_2)}_{g_2}) \end{aligned}$$

where $m : \mathbf{Mem}[\mathbf{Var}]$, $m_1 : \mathbf{Mem}[\mathrm{FV}(e)]$, and $m_2 : \mathbf{Mem}[(\mathbf{Var} \setminus \{x\}) \setminus \mathrm{FV}(e)]$. The maps g_1 and g_2 evidently preserves input to output and are thus kernels. Also, because $\mathbf{range}(k_1) \subseteq \mathrm{FV}(P) \subseteq (\mathbf{Var} \setminus \{x\})$ and $k_1 \sqsubseteq f$,

we have $k_1 \sqsubseteq \pi_{\mathbf{Var}\setminus\{x\}} f$; in addition, since $k_1 \models P$, we have $g \models P$ by persistence. Since $g_1 \sqsubseteq g_1 \oplus g_2$ and $g_1 \models (\mathrm{FV}(e) \triangleright x = e)$, we have $g_1 \oplus g_2 \models (\mathrm{FV}(e) \triangleright x = e)$ as well. Thus, we conclude $f' \models P \fatsemi (\mathrm{FV}(e) \triangleright x = e)$.

SAMP By restriction (theorem 4.3.1), there exists $k_1 \sqsubseteq f$ such that $\mathrm{FV}_{\mathrm{R}}(P) \subseteq \mathbf{range}(k_1) \subseteq \mathrm{FV}(P)$; let $K = \mathbf{range}(k_1)$. Since f has empty domain, we have $f = k_1 \odot k_2$ for some $k_2 : \mathbf{Mem}[K] \to \mathcal{D}(\mathbf{Mem}[\mathbf{Var}])$. Let $f' = \emptyset \mapsto [\![x \xleftarrow{\$} d]\!](\mu)$ be the lifted output. We have:

$$\begin{aligned} f' &= \langle\rangle \mapsto \mathsf{bind}(\mu, m \mapsto \mathsf{bind}([\![d]\!], v \mapsto \mathsf{unit}(m[x \mapsto v]))) \\ &= f \odot (m \mapsto \mathsf{bind}([\![d]\!], v \mapsto \mathsf{unit}(m[x \mapsto v]))) \\ &= \pi_{\mathbf{Var}\setminus\{x\}} f \odot ((m_1 \mapsto \mathsf{unit}(m_1)) \oplus \mathsf{bind}([\![d]\!], v \mapsto \mathsf{unit}(x \mapsto v))) \end{aligned}$$

where $m : \mathbf{Mem}[\mathbf{Var}]$, $m_1 : \mathbf{Mem}[\mathrm{FV}(d)]$, and $m_2 : \mathbf{Mem}[(\mathbf{Var} \setminus \mathrm{FV}(d)) \setminus \{x\}]$. Again, g_1 and g_2 evidently preserves input to the output and thus are kernels . Because $\mathbf{range}(k_1) \subseteq \mathbf{Var} \setminus \{x\}$ and $k_1 \sqsubseteq f$, we have $k_1 \sqsubseteq \pi_{\mathbf{Var}\setminus\{x\}} f$. Because $k_1 \sqsubseteq \pi_{\mathbf{Var}\setminus\{x\}} f$ and $k_1 \models P$, we have $\pi_{\mathbf{Var}\setminus\{x\}} f \models P$ by persistence. Since $g_1 \sqsubseteq g_1 \oplus g_2$ and $g_1 \models (\emptyset \triangleright x \overset{\$}{\sim} d)$, we have $g_1 \oplus g_2 \models (\emptyset \triangleright x \overset{\$}{\sim} d)$ as well. Thus, we conclude $f' \models P \fatsemi (\emptyset \triangleright x \overset{\$}{\sim} d)$.

SKIP Trivial.

SEQ Trivial.

COND At the high level, we proceed the proof in three steps: first, we show that for any f satisfying $(\emptyset \triangleright [b]); P$, there exists j_1, j_2 such that $f = j_1 \odot j_2$ and $\mathbf{range}(j_1) = \{b\}$; second, we describe exactly two kernels l_{tt} and l_{ff} such that $f_{\mu|[\![b=tt]\!]} = l_{tt} \odot j_2$ and $f_{\mu|[\![b=ff]\!]} = l_{ff} \odot j_2$; last, we compute $f_{[\![\textbf{if } b \textbf{ then } c \textbf{ else } c']\!]\mu}$ and show that it satisfies the post-condition.

Since all assertions are in DIBL$_+$, we have $\mathrm{FV}_{\mathrm{D}}(P) \subseteq \mathrm{FV}_{\mathrm{R}}(\emptyset \triangleright [b]) = \{b\}$. Since $f \models (\emptyset \triangleright [b]) \fatsemi P$, there exists k_1, k_2 such that $k_1 \odot k_2 = f$, with

$k_1 \models (\emptyset \triangleright [b])$ and $k_2 \models P$.

By restriction (theorem 4.3.1), there exists j_1 such that $j_1 \sqsubseteq k_1$ and

$$\mathbf{dom}(j_1) \subseteq \mathrm{FV}_{\mathrm{D}}(\emptyset \triangleright [b]) = \emptyset$$
$$\{b\} = \mathrm{FV}_{\mathrm{R}}(\emptyset \triangleright [b]) \subseteq \mathbf{range}(j_1) \subseteq \mathrm{FV}(\emptyset \triangleright [b]) = \{b\}.$$

By restriction (theorem 4.3.1), there exists j_2 such that $j_2 \sqsubseteq k_2$ and $j_2 \models P$, and $\mathbf{dom}(j_2) \subseteq \mathrm{FV}_{\mathrm{D}}(P) \subseteq \mathrm{FV}_{\mathrm{R}}(\emptyset \triangleright [b]) = \{b\}$. Since $\mathbf{dom}(k_2) = \mathbf{range}(k_1) \supseteq \{b\}$, we may assume without loss of generality that $j_2 \models P$, $j_2 \sqsubseteq k_2$, and $\mathbf{dom}(j_2) = \{b\}$. Thus $j_1 \odot j_2$ is defined, and so $j_1 \odot j_2 \sqsubseteq k_1 \odot k_2 \sqsubseteq f$ by lemma C.2.5.

By lemma C.2.1, there exists $j : \mathbf{Mem}[\mathbf{range}(j_2)] \to \mathcal{D}(\mathbf{Mem}[\mathbf{Var}])$ such that $j_1 \odot (j_2 \odot j) = (j_1 \odot j_2) \odot j = f$. Since $j_2 \sqsubseteq j_2 \odot j$, we have $j_2 \odot j \models P$. Thus, we may assume without loss of generality that $\mathbf{range}(j_2) = \mathbf{Var}$ and $j_1 \odot j_2 = f = \langle\rangle \mapsto \mu$.

Let $l_{tt}, l_{ff} : \mathbf{Mem}[\emptyset] \to \mathcal{D}(\mathbf{Mem}[b])$ be defined by $l_{tt}(\langle\rangle) = \mathsf{unit}(b \mapsto tt)$ and $l_{ff}(\langle\rangle) = \mathsf{unit}(b = ff)$; evidently, $l_{tt} \models (\emptyset \triangleright b = tt)$ and $l_{ff} \models (\emptyset \triangleright b = ff)$. Now, we have:

$$f_{\mu|[\![b=tt]\!]} = l_{tt} \odot j_2$$
$$f_{\mu|[\![b=ff]\!]} = l_{ff} \odot j_2$$

where each equality holds if the left side is defined. Regardless of whether the conditional distributions are defined, we always have:

$$l_{tt} \odot j_2 \models (\emptyset \triangleright b = tt) \fatsemi P$$
$$l_{ff} \odot j_2 \models (\emptyset \triangleright b = ff) \fatsemi P.$$

Since both of these kernels have empty domain, we have $l_{tt} \odot j_2 = \overline{\nu_{tt}}$ and $l_{ff} \odot j_2 = \overline{\nu_{ff}}$ for two distributions $\nu_{tt}, \nu_{ff} \in \mathcal{D}(\mathbf{Mem}[\mathbf{Var}])$. By induction, we

have:

$$f_{[\![c]\!]\nu_{tt}} \models (\emptyset \triangleright b = tt) \fatsemi (b : b = tt \triangleright Q_1)$$
$$f_{[\![c]\!]\nu_{ff}} \models (\emptyset \triangleright b = ff) \fatsemi (b : b = ff \triangleright Q_2).$$

By similar reasoning as for the pre-conditions, there exists $k_1', k_2' : \mathbf{Mem}[b] \to \mathcal{D}(\mathbf{Mem}[\mathbf{Var}])$ such that $k_1' \models (b : b = tt \triangleright Q_1)$ and $k_2' \models (b : b = ff \triangleright Q_2)$, and:

$$f_{[\![c]\!]\nu_{tt}} = l_{tt} \odot k_1' \qquad\qquad f_{[\![c]\!]\nu_{ff}} = l_{ff} \odot k_2'.$$

Let $k' : \mathbf{Mem}[b] \to \mathcal{D}(\mathbf{Mem}[\mathbf{Var}])$ be the composite kernel defined as follows:

$$k'([b \mapsto v]) \triangleq \begin{cases} k_1'([b \mapsto tt]) & : v = tt \\ k_2'([b \mapsto ff]) & : v = ff \end{cases}.$$

By assumption, $k' \models ((b : b = tt \triangleright Q_1) \wedge (b : b = ff \triangleright Q_2))$. Now, let $p \triangleq \mu([\![b = tt]\!])$ be the probability of taking the first branch. Then we can conclude:

$$\begin{aligned} f_{[\![\textbf{if } b \textbf{ then } c \textbf{ else } c']\!]\mu} &= f_{[\![c]\!](\mu|[\![b=tt]\!])\oplus_p [\![c']\!](\mu|[\![b=tt]\!])} \\ &= f_{[\![c]\!]\nu_{tt} \oplus_p [\![c]\!]\nu_{ff}} \\ &= f_{[\![c]\!]\nu_{tt}} \mathbin{\overline{\oplus}_p} f_{[\![c]\!]\nu_{ff}} \\ &= (l_{tt} \odot k_1') \mathbin{\overline{\oplus}_p} (l_{ff} \odot k_2') \\ &= (l_{tt} \odot k') \mathbin{\overline{\oplus}_p} (l_{ff} \odot k') \\ &= (l_{tt} \mathbin{\overline{\oplus}_p} l_{ff}) \odot k' \\ &\models (\emptyset \triangleright [b]) \fatsemi ((b : b = tt \triangleright Q_1) \wedge (b : b = ff \triangleright Q_2)). \end{aligned}$$

Above, $k_1 \overline{\oplus}_p k_2$ lifts the convex combination operator $\oplus_p$ from distributions to kernels from $\mathbf{Mem}[\emptyset]$. We show the last equality in more detail. For any

$r \in \mathbf{Mem}[\mathbf{Var}]$:

$$
\begin{aligned}
&(l_{tt} \odot k') \mathbin{\overline{\oplus}_p} (l_{ff} \odot k')(\langle\rangle)(r) \\
&= p \cdot (l_{tt} \odot k')(\langle\rangle)(r) + (1-p) \cdot (l_{ff} \odot k')(\langle\rangle)(r) \\
&= p \cdot (l_{tt} \odot k')(\langle\rangle)(r) + (1-p) \cdot (l_{ff} \odot k')(\langle\rangle)(r) \\
&= p \cdot l_{tt}(\langle\rangle)(b \mapsto tt) \cdot k'(b \mapsto tt)(r) + (1-p) \cdot l_{ff}(\langle\rangle)(b \mapsto ff) \cdot k'(b \mapsto ff)(r) \\
&= ((l_{tt} \mathbin{\overline{\oplus}_p} l_{ff}) \odot k')(\langle\rangle)(r).
\end{aligned}
$$

where the penultimate equality holds because l_{tt} and l_{ff} are deterministic.

Weak Trivial.

Frame The proof for this case follows the argument for the frame rule in PSL, with a few minor changes.

There exists k_1, k_2 such that $k_1 \oplus k_2 \sqsubseteq f$, and $k_1 \models P$ and $k_2 \models R$; let $S_1 \triangleq \mathbf{range}(k_1)$. Also, by restriction, there exists $k_2' \sqsubseteq k_2$ such that $k_2' \models R$ and $\mathbf{range}(k_2') \subseteq \mathrm{FV}(R)$; let $S_2 \triangleq \mathbf{range}(k_2')$. Since k_1 and k_2 have empty domains, S_1 and S_2 must be disjoint. Let $S_3 = \mathbf{Var} \setminus (S_2 \cup S_1)$. Since $\mathrm{MV}(c)$ is disjoint from S_2 by the first side-condition, we have $\mathrm{WV}(c) \subseteq \mathrm{MV}(c) \subseteq S_1 \cup S_3$.

Let $f' = f_{[\![c]\!]\mu}$ be the lifted output. By induction, we have $f' \models Q$; by restriction (theorem 4.3.1), there exists $k_1' \sqsubseteq f'$ such that $\mathbf{range}(k_1') \subseteq \mathrm{FV}(Q)$ and $k_1' \models Q$. By the last side condition, $\mathrm{RV}(c) \subseteq \mathrm{FV}_{\mathrm{R}}(P) \subseteq S_1$.

By soundness of RV and WV (lemma A.2.1), all variables in $\mathrm{WV}(c)$ must be written before they are read and there is a function $F : \mathbf{Mem}[S_1] \to \mathcal{D}(\mathbf{Mem}[\mathrm{WV}(c) \cup S_1])$ such that:

$$\pi_{\mathrm{WV}(c) \cup S_1} [\![c]\!] \mu = \mathsf{bind}(\mu, m \mapsto F(m^{S_1})).$$

Since $S_2 \subseteq \mathrm{FV}(R)$, variables in S_2 are not in $\mathrm{MV}(c)$ by the first side-condition, and S_2 is disjoint from $\mathrm{WV}(c) \cup S_1$. By soundness of MV, we have:

$$\pi_{\mathrm{WV}(c) \cup S_1 \cup S_2} [\![c]\!] \mu = \mathsf{bind}(\pi_{\mathrm{WV}(c) \cup S_1 \cup S_2} \mu, F \oplus \mathsf{unit}_{\mathbf{Mem}[\mathrm{WV}(c) \cup S_2]}).$$

Since S_1 and S_2 are independent in μ, we know that $S_1 \cup \mathrm{WV}(c)$ and S_2 are independent in $[\![c]\!] \mu$. Hence:

$$f_{\pi_{S_1 \cup \mathrm{WV}(c)} [\![c]\!] \mu} \oplus f_{\pi_{S_2} [\![c]\!] \mu} \sqsubseteq f'.$$

By induction, $f' \models Q$. Furthermore, $\mathrm{FV}(Q) \subseteq \mathrm{FV}_{\mathrm{R}}(P) \cup \mathrm{WV}(c) \subseteq S_1 \cup \mathrm{WV}(c)$ by the second side-condition. By restriction (theorem 4.3.1), $f_{\pi_{S_1 \cup \mathrm{WV}(c)} [\![c]\!] \mu} \models Q$. Furthermore, $\pi_{S_2} [\![c]\!] \mu = \pi_{S_2} \mu$, so $\pi_{S_2} [\![c]\!] \mu \models R$ as well. Thus, $f' \models Q * R$ as desired.

□

APPENDIX D

THE UNARY FRAGMENT BLUEBELL FOR REASONING ABOUT INDEPENDENCE AND CONDITIONAL INDEPENDENCE

D.1 The Rules of BLUEBELL

In fig. D.1 we summarize the notation we use for assertions over BLUEBELL's model. Recall that BLUEBELL's assertions $P \in \mathsf{PA}_I \triangleq \mathcal{M}_I \xrightarrow{\mathsf{u}} \mathsf{Prop}$ are the upward-closed predicates over elements of the RA $\mathcal{M}_I$.

$$
\begin{aligned}
\ulcorner \phi \urcorner &\triangleq \lambda\_.\, \phi \\
\mathsf{Own}(b) &\triangleq \lambda a.\, b \preceq a \\
P \wedge Q &\triangleq \lambda a.\, P(a) \wedge Q(a) \\
P * Q &\triangleq \lambda a.\, \exists b_1, b_2.\, (b_1 \cdot b_2) \preceq a \wedge P(b_1) \wedge Q(b_2) \\
\exists x : X.\, K &\triangleq \lambda a.\, \exists x : X.\, K(x)(a) && (K \colon X \to \mathsf{PA}_I) \\
\forall x : X.\, K &\triangleq \lambda a.\, \forall x : X.\, K(x)(a) && (K \colon X \to \mathsf{PA}_I) \\
\mathsf{Own}(\mathcal{F}, \mu) &\triangleq \exists p.\, \mathsf{Own}(\mathcal{F}, \mu, p) \\
E \Subset \mu &\triangleq \exists \mathcal{F}, \mu.\, \mathsf{Own}(\mathcal{F}, \mu) * \ulcorner E \prec (\mathcal{F}(i), \mu(i)) \wedge \mu = \mu(i) \circ E^{-1} \urcorner \\
\boldsymbol{C}_\mu\, K &\triangleq \lambda a.\, \exists \mathcal{F}, \mu, p, \kappa.\, (\mathcal{F}, \mu, p) \preceq a \wedge \forall i \in I.\, \mu(i) = \mathsf{bind}(\mu, \kappa(i)) \\
&\qquad\qquad \wedge \forall v \in \mathrm{supp}(\mu).\, K(v)(\mathcal{F}, \kappa(I)(v), p) && (\mu : \mathcal{D}(A), K \colon A \to \mathsf{PA}_I) \\
\mathbf{wp}\, t\, \{Q\} &\triangleq \lambda a.\, \forall \mu_0.\forall c.\, (a \cdot c) \preceq \mu_0 \Rrightarrow \exists b.\, \big((b \cdot c) \preceq [\![t]\!](\mu_0) \wedge Q(b)\big) \\
\lceil E \rceil &\triangleq (E \in \mathsf{true}) \Subset \delta_{\mathsf{True}} \\
\mathsf{Own}(E) &\triangleq \exists \mu.\, E \Subset \mu \\
(x{:}q) &\triangleq \exists \mathcal{P}, p.\, \mathsf{Own}(\mathcal{P}, p) * \ulcorner p(i)(x) = q \urcorner \\
P @ p &\triangleq P \wedge \exists \mathcal{P}.\, \mathsf{Own}(\mathcal{P}, p) \\
\lfloor R \rfloor &\triangleq \exists \mu : \mathcal{D}(\mathbf{Val}^X).\, \ulcorner \mu(R) = 1 \urcorner * \boldsymbol{C}_\mu\, v.\, \lceil x = v(x) \rceil_{x \in X} && (R \subseteq \mathbf{Val}^X, X \subseteq I \times \mathbf{Var})
\end{aligned}
$$

Figure D.1: The assertions used in BLUEBELL.

Proposition D.1.1 (Upward-closure)**.** *All the assertions in fig. D.1 are upward-closed.*

Proof. Easy by inspection of the definitions. The definitions where upward-closedness is less obvious (e.g. joint conditioning) are made upward-closed by

construction by explicit use of the order $\preceq$ in the definition. □

Lemma D.1.2. *For all $\mu : \mathcal{D}(A \times B)$, there exists a $\kappa : A \to \mathcal{D}(B)$ such that $\mu = (\mu \circ \pi_1^{-1}) \prec \kappa$.*

Proof. Let $\mu_1 = \mu \circ \pi_1^{-1}$. Then the result is immediate by letting

$$\kappa(a)(b) = \begin{cases} \frac{\mu_0(a,b)}{\mu_1(a)} & \text{if} \mu_1(a) > 0 \\ 0 & \text{otherwise} \end{cases}$$

□

D.1.1 Program Semantics

We assume each primitive operator $\varphi \in \{+, -, <, \ldots\}$ has an associated arity $\mathrm{ar}(\varphi) \in \mathbb{N}$, and is given semantics as some function $[\![\varphi]\!] : \mathbf{Val}^{\mathrm{ar}(\varphi)} \to \mathbf{Val}$.

Definition D.1.1. Expressions $e \in \mathcal{E}$ are given semantics as a function $[\![e]\!] : \mathbf{Mem}[\mathbf{Var}] \to \mathbf{Val}$ as standard:

$$[\![v]\!](s) \triangleq v \quad [\![x]\!](s) \triangleq s(x) \quad [\![\varphi(e_1, \ldots, e_{\mathrm{ar}(\varphi)})]\!](s) \triangleq [\![\varphi]\!]([\![e_1]\!], \ldots, [\![e_{\mathrm{ar}(\varphi)}]\!])$$

Definition D.1.2 (Term semantics)**.** Given $t \in \mathbb{T}$ we define its *kernel semantics* $\mathcal{K}[\![t]\!] : \mathbf{Mem}[\mathbf{Var}] \to \mathcal{D}(\Sigma_{\mathbf{Mem}[\mathbf{Var}]})$ as follows:

$$\mathcal{K}[\![\texttt{skip}]\!](s) \triangleq \mathsf{unit}(s)$$

$$\mathcal{K}[\![x := e]\!](s) \triangleq \mathsf{unit}(s[x \mapsto [\![e]\!](s)])$$

$$\mathcal{K}[\![x :\approx d]\!](s) \triangleq \mathsf{bind}([\![d]\!]([\![e_1]\!](s), \ldots, [\![e_n]\!](s)), v \mapsto \mathbf{return}(s[x \mapsto v]))$$

$$\mathcal{K}[\![t_1; t_2]\!](s) \triangleq \mathsf{bind}(\mathcal{K}[\![t_1]\!](s), s' \mapsto \mathcal{K}[\![t_2]\!](s'))$$

$$\mathcal{K}[\![\texttt{if } e \texttt{ then } t_1 \texttt{ else } t_2]\!](s) \triangleq \mathsf{if}\ [\![e]\!](s) \neq 0\ \mathsf{then}\ \mathcal{K}[\![t_1]\!](s)\ \mathsf{else}\ \mathcal{K}[\![t_2]\!](s)$$

$$\mathcal{K}[\![\texttt{repeat } e\ t]\!](s) \triangleq \mathit{loop}_t([\![e]\!](s), s)$$

where $loop_t$ simply iterates t:

$$loop_t(n, s) \triangleq \begin{cases} \mathsf{unit}(s) & n \leq 0 \\ \mathsf{bind}(loop_t(n-1, s), s' \mapsto \mathcal{K}[\![t]\!](s')) & \text{Otherwise} \end{cases}$$

The semantics of a term is then defined as:

$$[\![t]\!] \colon \mathcal{D}(\Sigma_{\mathbf{Mem[Var]}}) \to \mathcal{D}(\Sigma_{\mathbf{Mem[Var]}})$$
$$[\![t]\!](\mu) \triangleq \mathsf{bind}(\mu, s \mapsto \mathcal{K}[\![t]\!](s))$$

Evaluation contexts $\mathcal{E}$ are defined by the following grammar:

$$\mathcal{E} ::= x := E \mid x :\approx d \mid \texttt{if}\ E\ \texttt{then}\ t_1\ \texttt{else}\ t_2 \mid \texttt{repeat}\ E\ t$$
$$E ::= [\,\cdot\,] \mid \varphi(\vec{e}_1, E, \vec{e}_2)$$

A simple property holds for evaluation contexts.

Lemma D.1.3. $\mathcal{K}[\![\mathcal{E}[e]]\!](s) = \mathcal{K}[\![\mathcal{E}[[\![e]\!](s)]]\!](s)$.

Proof. Easy by induction on the structure of evaluation contexts. □

D.2 Measure Theory Lemmas

Notation In what follows, given $n \in \mathbb{N}$ with $n > 1$, we write $[n]$ to denote the set $\{1, \ldots, n\}$. Moreover, for iterated summation we use the notation $\sum_{i \in I \mid \Phi(i)} f(i)$ where $I = \{i_0, i_1, \ldots\}$ is countable and Φ is a predicate on elements of I, to denote the sum $f(j_0) + f(j_1) + \ldots$ where $j_0, j_1, \ldots$ is the sublist of $i_0, i_1, \ldots$ consisting of the elements that satisfy Φ. A similar convention is used for other commutative and associative operators, e.g. $\cup$. A countable partition of Ω is a

partition of Ω, $S \subseteq \mathscr{P}(\Omega)$, with countably many sets. For uniformity, we represent countable partitions as $S = \{A_i\}_{i \in \mathbb{N}}$ with the convention that when the partition has finitely many sets, say n, all the A_i with $i \geq n$ are empty.

As mentioned, BLUEBELL is only concerned with discrete distributions, i.e. distributions over a countable set of outcomes. The following lemma expresses the key property of σ-algebras over countable outcomes that we exploit for proving the other results.

Lemma D.2.1. *Let Ω be an countable set, and $\mathcal{F}$ to be an arbitrary σ-algebra on Ω. Then there exists a countable partition S of Ω such that $\mathcal{F} = \sigma(S)$.*

Proof. For every element $x \in \Omega$, we identify the smallest event $E_x \in \mathcal{F}$ such that $x \in E_x$, and show that for $x, z \in \Omega$, either $E_x = E_z$ or $E_x \cap E_z = \emptyset$. Then the set $S = \{E_x \mid x \in \Omega\}$ is a partition of Ω, and any event $E \in \mathcal{F}$ can be represented as $\bigcup_{x \in E} E_x$, which suffices to show that $\mathcal{F} = \sigma(S)$.

For every x, y, let

$$A_{x,y} = \begin{cases} \Omega & \text{if } \forall E \in \mathcal{F}, \text{either } x, y \text{ both in } E \text{ or } x, y \text{ both not in } E \\ E & \text{otherwise, pick any } E \in \mathcal{F} \text{ such that } x \in E \text{ and } y \notin E \end{cases}$$

Then we show that, for all x, $E_x = \cap_{y \in \Omega} A_{x,y}$ is the smallest event in $\mathcal{F}$ such that $x \in E_x$ as follows. If there exists E'_x such that $x \in E'_x$ and $E'_x \subset E_x$, then $E_x \setminus E'_x$ is not empty. Let y be an element of $E_x \setminus E'_x$, and by the definition of $A_{x,y}$, we have $y \notin A_{x,y}$. Thus, $y \notin \cap_{y \in \Omega} A_{x,y} = E_x$, which contradicts with $y \in E_x \setminus E'_x$.

Next, for any $x, z \in \Omega$, since E_x is the smallest event containing x and E_z is the smallest event containing z, the smaller event $E_z \setminus E_x$ is either equivalent to E_z or not containing z.

If $E_z \setminus E_x = E_z$, then E_x and E_z are disjoint.

If $z \notin E_z \setminus E_x$, then it must $z \in E_x$, which implies that there exists *no* $E \in \mathcal{F}$ such that $x \in E$ and $z \notin E$. Because $\mathcal{F}$ is closed under complement, then there exists *no* $E \in \mathcal{F}$ such that $x \notin E$ and $z \in E$ as well. Therefore, we have $x \in \bigcap_{y \in \Omega} A_{z,y} = E_z$ as well. Furthermore, because E_z is the smallest event in $\mathcal{F}$ that contains z and E_x also contains z, we have $E_z \subseteq E_x$; symmetrically, we have $E_x \subseteq E_z$. Thus, $E_x = E_z$.

Hence, the set $S = \{E_x \mid x \in \Omega\}$ is a countable partition of Ω. □

Lemma D.2.2. *If $S = \{A_i\}_{i \in \mathbb{N}}$ is a partition of Ω, and $\mathcal{F} = \sigma(S)$, then every event $E \in \mathcal{F}$ can be written as $E = \biguplus_{i \in I} A_i$ for some $I \subseteq \mathbb{N}$. In other words, $\sigma(S) = \{[\}\big]\biguplus_{i \in I} A_i | I \subseteq \mathbb{N}$.*

Proof. Because σ-algebras are closed under countable union, for any $I \subseteq \mathbb{N}$, $\biguplus_{i \in I} A_i \in \sigma(S)$. Thus, $\sigma(S) \supseteq \{[\}\big]\biguplus_{i \in I} A_i \mid I \subseteq \mathbb{N}$.

Also, $\{[\}\big]\biguplus_{i \in I} A_i \mid I \subseteq \mathbb{N}$ is a σ-algebra:

- $\Omega = \biguplus_{i \in \mathbb{N}} A_i$.
- Given a countable sequences of events $E_1 = \biguplus_{i \in I_1} A_i$, $E_2 = \biguplus_{i \in I_2} A_i$, …, let $I = \bigcup_{j \in \mathbb{N}} I_j$; then we have $\bigcup_{j \in \mathbb{N}} E_i = \biguplus_{i \in I} A_i$.
- If $E = \biguplus_{i \in I} A_i$, then the complement of E is $(\Omega \setminus E) = \biguplus_{i \in (\mathbb{N} \setminus I)} A_i$.

Then, $\{\biguplus_{i \in I} A_i \mid I \subseteq \mathbb{N}\}$ is a σ-algebra that contains S. Therefore, $\sigma(S) = \{[\}\big]\biguplus_{i \in I} A_i \mid I \subseteq \mathbb{N}$. □

Lemma D.2.3. *Let Ω be a countable set. If $S_1 = \{A_i\}_{i\in\mathbb{N}}$ and $S_2 = \{B_j\}_{j\in\mathbb{N}}$ are both countable partitions of Ω, then $\sigma(S_1) \subseteq \sigma(S_2)$ implies that for any $B_j \in S_2$ with $B_j \neq \emptyset$, we can find a unique $A_i \in S_1$ such that $B_j \subseteq A_i$.*

Proof. For any $B_j \in S_2$ with $B_j \neq \emptyset$, pick an arbitrary element $s \in B_j$ and denote the unique element of S_1 that contains s as A_i. Because $A_i \in S_1$ and $S_1 \subset \sigma(S_1) \subseteq \sigma(S_2)$, we have $A_i \in \sigma(S_2)$. Note that $s \in B_j$ and B_j is an element of the partition S_2 that generates $\sigma(S_2)$, B_j must be the smallest event in $\sigma(S_2)$ that contains s. Because $s \in A_i$ as well, B_j being the smallest event containing s implies that $B_j \subseteq A_i$. □

Lemma D.2.4. *Assume we are given a σ-algebra $\mathcal{F}_1$ over a countable set Ω, measure $\mu_1 \in \mathcal{D}(\mathcal{F}_1)$, a countable set A, a distribution $\mu \in \Sigma_A$, and a function $\kappa_1 : A \to \mathcal{D}(\mathcal{F}_1)$ such that $\mu_1 = \mathsf{bind}(\mu, \kappa_1)$. Then, for any probability space $(\mathcal{F}_2, \mu_2)$ such that $(\mathcal{F}_1, \mu_1) \sqsubseteq (\mathcal{F}_2, \mu_2)$, there exists κ_2 such that $\mu_2 = \mathsf{bind}(\mu, \kappa_2)$, and that for any $a \in \mathrm{supp}(\mu)$, $(\mathcal{F}_1, \kappa_1(a)) \sqsubseteq (\mathcal{F}_2, \kappa_2(a))$.*

Proof. By lemma D.2.1, $\mathcal{F}_i = \sigma(S_i)$ for some countable partition S_i. Also, $(\mathcal{F}_1, \mu_1) \sqsubseteq (\mathcal{F}_2, \mu_2)$ implies that $\mathcal{F}_1 \subseteq \mathcal{F}_2$. So we have $\sigma(S_1) \subseteq \sigma(S_2)$, which by lemma D.2.3 implies that for any $B \in S_2$ with $B \neq \emptyset$, we can find a unique $A \in S_1$ such that $B \subseteq A$. Let f be the mapping associating to any $B \neq \emptyset$ the corresponding $A = f(B)$, and $f(B) = \emptyset$ when $B = \emptyset$.

Then, we define κ_2 as follows: for any $a \in A$, $E \in \mathcal{F}_2$, there exists $S \subseteq S_2$ such that $E = \biguplus_{B\in S} B$, then define

$$\kappa_2(a)(E) = \sum_{B\in S} \kappa_1(a)(f(B)) \cdot h(B),$$

where $h(B) = \mu_2(B)/\mu_2(f(B))$ if $\mu_2(f(B)) \neq 0$ and $h(B) = 0$ otherwise.

Then we calculate:

$$
\begin{aligned}
&\mathsf{bind}(\mu, \kappa_2)(E) \\
&= \sum_{a \in A} \mu(a) \cdot \kappa_2(E) \\
&= \sum_{a \in A} \mu(a) \cdot \sum_{B \in S} \kappa_1(a)(f(B)) \cdot h(B) \\
&= \sum_{B \in S} \sum_{a \in A} \mu(a) \cdot \kappa_1(a)(f(B)) \cdot h(B) \\
&= \sum_{B \in S} \mathsf{bind}(\mu, \kappa_1)(f(B)) \cdot h(B) \\
&= \sum_{B \in S} \mu_1(f(B)) \cdot h(B) \\
&= \sum_{B \in S | \mu_2(f(B)) \neq 0} \mu_1(f(B)) \cdot \frac{\mu_2(B)}{\mu_2(f(B))} \\
&= \sum_{B \in S | \mu_2(f(B)) \neq 0} \mu_2(f(B)) \cdot \frac{\mu_2(B)}{\mu_2(f(B))} && (\mu_1(E') = \mu_2(E') \text{ for any } E' \in \mathcal{F}_1) \\
&= \sum_{B \in S | \mu_2(f(B)) \neq 0} \mu_2(B) \\
&= \sum_{B \in S | \mu_2(f(B)) \neq 0} \mu_2(B) \quad + \quad \sum_{B \in S | \mu_2(f(B)) = 0} \mu_2(B) && (\text{Because } \mu_2(f(B)) = 0 \text{ implies } \mu_2(B) = 0) \\
&= \sum_{B \in S} \mu_2(B) \\
&= \mu_2(\biguplus_{B \in S} B) \\
&= \mu_2(E)
\end{aligned}
$$

Thus, $\mathsf{bind}(\mu, \kappa_2) = \mu_2$.

Also, for any $a \in A_\mu$, for any $E \in \mathcal{F}_1$, there exists $S' \subseteq S_1$ such that $E =$

$\biguplus_{A \in S'} A$.

$$
\begin{aligned}
\kappa_2(a)(E) &= \kappa_2(a)\left(\biguplus_{A \in S'} A\right) \\
&= \sum_{A \in S'} \kappa_2(a)(A) \\
&= \sum_{A \in S'} \sum_{B \subseteq A \mid B \in \mathcal{F}_2} \kappa_2(a)(B) \\
&= \sum_{A \in S'} \sum_{B \subseteq A \mid B \in \mathcal{F}_2, \mu_2(f(B)) \neq 0} \kappa_1(a)(f(B)) \cdot \frac{\mu_2(B)}{\mu_2(f(B))} \\
&= \sum_{A \in S' \mid \mu_2(A) \neq 0} \kappa_1(a)(A) \cdot \frac{\left(\sum_{B \subseteq A \mid B \in \mathcal{F}_2} \mu_2(B)\right)}{\mu_2(A)} \\
&= \sum_{A \in S' \mid \mu_2(A) \neq 0} \kappa_1(a)(A) \cdot \frac{\mu_2(A)}{\mu_2(A)} \\
&= \sum_{A \in S' \mid \mu_2(A) \neq 0} \kappa_1(a)(A) \\
&= \sum_{A \in S'} \kappa_1(a)(A) \\
&= \kappa_1(a)\left(\biguplus_{A \in S'} A\right) \\
&= \kappa_1(a)(E)
\end{aligned}
$$

Thus, for any a, $(\sigma_1, \kappa_1(a)) \sqsubseteq (\sigma_2, \kappa_2(a))$. □

Lemma D.2.5. *Given two σ-algebras $\mathcal{F}_1$ and $\mathcal{F}_2$ over two countable underlying sets Ω_1, Ω_2, then a general element in the product σ-algebra $\mathcal{F}_1 \otimes \mathcal{F}_2$ can be expressed as $\biguplus_{(i,j) \in I}(A_i \times B_j)$ for some $I \subseteq \mathbb{N}^2$ and $A_i \in \mathcal{F}_1, B_j \in \mathcal{F}_2$ for $(i, j) \in I$.*

Proof. By lemma D.2.1, each σ-algebra $\mathcal{F}_i$ is generated by a countable partition over Ω_i. Let $S_1 = \{A_i\}_{i \in \mathbb{N}}$ be the countable partition that generates $\mathcal{F}_1$, $S_2 = \{B_i\}_{i \in \mathbb{N}}$ be the countable partition that generates $\mathcal{F}_2$. By lemma D.2.2, a general element in $\mathcal{F}_1$ can be written as $\biguplus_{j \in J} A_j$ for some $J \subseteq \mathbb{N}$, and similarly, a general element in $\mathcal{F}_2$ can be written as $\biguplus_{k \in K} B_k$ for some $K \subseteq \mathbb{N}$.

Note that $\{A_j \times B_k\}_{j,k\in\mathbb{N}}$ is a partition because: if $(A_j \times B_k) \cap (A_{j'} \times B_{k'}) \neq \emptyset$ for some $j \neq j'$ and $k \neq k'$, then it must $A_j \cap A_{j'} \neq \emptyset$ and $B_k \cap B_{k'} \neq \emptyset$, and that imply that $A_j = A_{j'}$ and $B_j = B_{j'}$; therefore, $A_j \times B_k = A_{j'} \times B_{k'}$.

We next show that $\mathcal{F}_1 \otimes \mathcal{F}_2$ is generated by the partition $\{A_j \times B_k\}_{j,k\in\mathbb{N}}$.

$$\begin{aligned}
\mathcal{F}_1 \otimes \mathcal{F}_2 &= \sigma(\mathcal{F}_1 \times \mathcal{F}_2) \\
&= \sigma\Big(\{*\} \uplus_{j\in J_1} A_j \times \uplus_{j\in J_2} B_j | J_1, J_2 \subseteq \mathbb{N}\Big) \\
&= \sigma\Big(\{*\} \uplus_{j\in J_1, k\in J_2} (A_j \times B_k) | J_1, J_2 \subseteq \mathbb{N}\Big) \\
&= \sigma\big(\{*\} A_j \times B_k | j, k \subseteq \mathbb{N}\big)
\end{aligned}$$

Since each $A_j \in S_1 \subseteq \mathcal{F}_1$ and $B_k \in S_2 \subseteq \mathcal{F}_2$ a general element in $\mathcal{F}_1 \otimes \mathcal{F}_2$ can be expressed as $\{*\} \uplus_{j,k\subseteq I} (A_j \times B_k) \mid A_j \in \mathcal{F}_1, B_k \in \mathcal{F}_2, I \subseteq \mathbb{N}^2$ according to lemma D.2.1. □

Lemma D.2.6. *Given two probability spaces $(\mathcal{F}_a, \mu_a), (\mathcal{F}_b, \mu_b) \in \mathbb{P}(\Omega)$, their independent product $(\mathcal{F}_a, \mu_a) \circledast (\mathcal{F}_b, \mu_b)$ exists if $\mu_a(E_a) \cdot \mu_b(E_b) = 0$ for any $E_a \in \mathcal{F}_a, E_b \in \mathcal{F}_b$ such that $E_a \cap E_b = \emptyset$.*

Proof. We first define $\mu : \{E_a \cap E_b \mid E_a \in \mathcal{F}_a, E_b \in \mathcal{F}_b\} \to [0, 1]$ by $\mu(E_a \cap E_b) = \mu_a(E_a) \cdot \mu_b(E_b)$ for any $E_a \in \mathcal{F}_a, E_b \in \mathcal{F}_b$, and then show that μ could be extended to a probability measure on $\mathcal{F}_a \oplus \mathcal{F}_b$.

- We first need to show that μ is **well-defined**. That is, $E_a \cap E_b = E'_a \cap E'_b$ implies $\mu_a(E_a) \cdot \mu_b(E_b) = \mu_a(E'_a) \cdot \mu_b(E'_b)$.

  When $E_a \cap E_b = E'_a \cap E'_b$, it must $E_a \cap E'_a \supseteq E_a \cap E_b = E'_a \cap E'_b$, Thus, $E_a \setminus E'_a \subseteq E_a \setminus E_b$, and then $E_a \setminus E'_a$ is disjoint from E_b; symmetrically, $E'_a \setminus E_a$ is disjoint from E'_b. Since E_a, E'_a are both in $\mathcal{F}_a$, we have $E_a \setminus E'_a$

and $E'_a \setminus E_a$ both measurable in $\mathcal{F}_a$. Their disjointness and the result above implies that $\mu_a(E_a \setminus E'_a) \cdot \mu_b(E_b) = 0$ and $\mu_a(E'_a \setminus E_a) \cdot \mu_b(E'_b) = 0$. Symmetric reasoning can also show that $E'_b \setminus E_b$ is disjoint from $E'_a \cap E_a$, and $E_b \setminus E'_b$ is disjoint from $E'_a \cap E_a$, which implies $\mu_a(E_b \setminus E'_b) \cdot \mu_b(E'_a \cap E_a) = 0$ and $\mu_a(E'_b \setminus E_b) \cdot \mu_b(E'_a) = 0$.

Then there are four possibilities:

- If $\mu_b(E_b) = 0$ and $\mu_b(E'_b) = 0$, then $\mu_a(E_a) \cdot \mu_b(E_b) = 0 = \mu_a(E'_a) \cdot \mu_b(E'_b)$.
- If $\mu_a(E_a \setminus E'_a) = 0$ and $\mu_b(E'_a \setminus E_a) = 0$. Then

$$\begin{aligned}
\mu_a(E_a) \cdot \mu_b(E_b) &= \mu_a((E'_a \setminus E_a) \uplus (E'_a \cap E_a)) \cdot \mu_b(E_b) \\
&= (\mu_a(E'_a \setminus E_a) + \mu_a(E'_a \cap E_a)) \cdot \mu_b(E_b) \\
&= \mu_a(E'_a \cap E_a) \cdot \mu_b(E_b) \\
&= (\mu_a(E_a \setminus E'_a) + \mu_a(E'_a \cap E_a)) \cdot \mu_b(E_b) \\
&= \mu_a(E'_a) \cdot \mu_b(E_b)
\end{aligned}$$

  Thus, either $\mu_a(E'_a \cap E_a) = 0$, which implies that

$$\mu_a(E_a) \cdot \mu_b(E_b) = (0+0) \cdot \mu_b(E_b) = 0 = (0+0) \cdot \mu_b(E_b) = \mu_a(E'_a) \cdot \mu_b(E'_b),$$

  or we have both $\mu_b(E'_b \setminus E_b) = 0$ and $\mu_b(E_b \setminus E'_b) = 0$, which imply that

$$\begin{aligned}
\mu_a(E_a) \cdot \mu_b(E_b) &= \mu_a(E'_a) \cdot \mu_b(E_b) \\
&= \mu_a(E'_a) \cdot \mu_b((E_b \cap E'_b) \uplus (E_b \setminus E'_b)) \\
&= \mu_a(E'_a) \cdot (\mu_b(E_b \cap E'_b) + 0) \\
&= \mu_a(E'_a) \cdot (\mu_b(E_b \cap E'_b) + \mu_b(E'_b \setminus E_b)) \\
&= \mu_a(E'_a) \cdot \mu_b(E'_b).
\end{aligned}$$

– If $\mu_b(E'_b) = 0$ and $\mu_b(E_a \setminus E'_a) = 0$, then

$$\begin{aligned}\mu_a(E_a) \cdot \mu_b(E_b) &= (\mu_a(E_a \cap E'_a) + \mu_a(E_a \setminus E'_a)) \cdot (\mu_b(E_b \cap E'_b) + \mu_b(E_b \setminus E'_b)) \\ &= \mu_a(E_a \cap E'_a) \cdot \mu_b(E_b \setminus E'_b)\end{aligned}$$

Because $\mu_a(E_b \setminus E'_b) \cdot \mu_b(E'_a \cap E_a) = 0$ and $\mu_a(E'_b \setminus E_b) \cdot \mu_b(E'_a) = 0$. Thus, $\mu_a(E_a) \cdot \mu_b(E_b) = 0 = \mu_a(E'_a) \cdot \mu_b(E'_b)$.

– If $\mu_b(E_b) = 0$ and $\mu_b(E'_a \setminus E_a) = 0$, then symmetric as above.

In all these cases, $\mu_a(E_a) \cdot \mu_b(E_b) = \mu_a(E'_a) \cdot \mu_b(E'_b)$ as desired.

- Show that μ satisfy **countable additivity** in $\{E_a \cap E_b \mid E_a \in \mathcal{F}_a, E_b \in \mathcal{F}_b\}$.

  We start with showing that μ is finite-additive. Suppose $E^n_a \cap E^n_b = \biguplus_{i\in[n]}(A_i \cap B_i)$ where each $A_i \in \mathcal{F}_a$ and $B_i \in \mathcal{F}_b$. Fix any $A_i \cap B_i$, there is unique minimal $A \in \mathcal{F}_a$ containing $A_i \cap B_i$, because if $A \supseteq A_i \cap B_i$ and $A' \supseteq A_i \cap B_i$, then $A \cap A' \supseteq A_i \cap B_i$ and $A \cap A' \in \mathcal{F}_A$ too, and $A \cap A'$ is smaller. Because we have shown that μ is well-defined, in the following proof, we can assume without loss of generality that A_i is the smallest set in $\mathcal{F}_a$ containing $A_i \cap B_i$. Similarly, we let B_i to be the smallest set in $\mathcal{F}_b$ containing $A_i \cap B_i$. Thus, $E^n_a \cap E^n_b = \biguplus_{i\in[n]}(A_i \cap B_i)$ implies every A_i is smaller than E^n_a and every B_i is smaller than E^n_b. Therefore, $E^n_a \supseteq \cup_{i\in[n]} A_i$ and $E^n_b \supseteq \cup_{i\in[n]} B_i$, which implies that

$$E^n_a \cap E^n_b \supseteq (\cup_{i\in[n]} A_i) \cap (\cup_{i\in[n]} B_i) \supseteq \cup_{i\in[n]}(A_i \cap B_i) = E^n_a \cap E^n_b,$$

  which implies that the $\supseteq$ in the inequalities all collapse to $=$.

  For any $I \subseteq [n]$, define $\alpha_I = \cap_{i\in I} A_i \setminus (\cup_{i\in[n]\setminus I} A_i)$, and $\beta_I = \cap_{i\in I} B_i \setminus (\cup_{i\in[n]\setminus I} B_i)$. For any $I \neq I'$, $\alpha_I \cap \alpha_{I'} = \emptyset$. Thus, $\{\alpha_I\}_{I\subseteq[n]}$ is a set of disjoint sets in $\cup_{i\in[n]} A_i$, and similarly, $\{\beta_I\}_{I\subseteq[n]}$ is a set of disjoint sets in $\cup_{i\in[n]} B_i$. Also, for any

$i \in [n]$, we have $A_i = \cup_{I \subseteq [n] | i \in I} \alpha_I$ and $B_i = \cup_{I \subseteq [n] | i \in I} \beta_I$. Furthermore, for any I,

$$\alpha_I \cap \cup_{i \in [n]} B_i \subseteq (\cup_{i \in [n]} A_i) \cap (\cup_{i \in [n]} B_i) = \biguplus_{i \in [n]} A_i \cap B_i,$$

and thus,

$$\begin{aligned}
\alpha_I \cap \cup_{i \in [n]} B_i &= (\biguplus_{i \in [n]} A_i \cap B_i) \cap (\alpha_I \cap \cup_{i \in [n]} B_i) \\
&= \biguplus_{i \in [n]} \left(A_i \cap B_i \cap \alpha_I \cap \cup_{j \in [n]} B_j\right) \\
&= \biguplus_{i \in I} \left(A_i \cap B_i \cap \alpha_I \cap \cup_{j \in [n]} B_j\right) && (A_i \cap \alpha_I = \emptyset \text{ if } i \notin I) \\
&= \biguplus_{i \in I} (A_i \cap B_i \cap \alpha_I) && (B_i \cap \cup_{j \in [n]} B_j = B_i \text{ for any } i) \\
&= \biguplus_{i \in I} (B_i \cap \alpha_I) && (A_i \cap \alpha_I = \alpha_I \text{ for any } i \in I) \\
&= \alpha_I \cap \cup_{i \in I} B_i && \text{(D.1)}
\end{aligned}$$

Now,

$$
\begin{aligned}
&\mu(E_a^n \cap E_b^n) \\
&= \mu((\cup_{i\in[n]} A_i) \cap (\cup_{i\in[n]} B_i)) \\
&= \mu((\uplus_{I\subseteq[n]} \alpha_I) \cap (\cup_{i\in[n]} B_i)) && \text{(By definition of } \alpha_I\text{)} \\
&= \mu_a(\uplus_{I\subseteq[n]} \alpha_I) \cdot \mu_b(\cup_{i\in[n]} B_i) && \text{(By definition of } \mu\text{)} \\
&= \left(\sum_{I\subseteq[n]} \mu_a(\alpha_I)\right) \cdot \mu_b(\cup_{i\in[n]} B_i) && \text{(By finite-additivity of } \mu_a\text{)} \\
&= \sum_{I\subseteq[n]} \mu_a(\alpha_I) \cdot \mu_b(\cup_{i\in[n]} B_i) \\
&= \sum_{I\subseteq[n]} \mu(\alpha_I \cap (\cup_{i\in[n]} B_i)) && \text{(By definition of } \mu\text{)} \\
&= \sum_{I\subseteq[n]} \mu(\alpha_I \cap (\cup_{i\in I} B_i)) && \text{(By eq. (D.1))} \\
&= \sum_{I\subseteq[n]} \mu_a(\alpha_I) \cdot \mu_b(\cup_{i\in I} B_i) && \text{(By definition of } \mu\text{)} \\
&= \sum_{I\subseteq[n]} \mu_a(\alpha_I) \cdot \mu_b(\cup_{i\in I}(\uplus_{I'\subseteq[n]|i\in I'} \beta_{I'})) && \text{(By definition of } \beta_I\text{)} \\
&= \sum_{I\subseteq[n]} \mu_a(\alpha_I) \cdot \mu_b(\uplus_{I'\subseteq[n]|I\cap I'\neq\emptyset} \beta_{I'}) \\
&= \sum_{I\subseteq[n]} \mu_a(\alpha_I) \cdot \sum_{I'\subseteq[n]|I\cap I'\neq\emptyset} \mu_b(\beta_{I'}) \\
&= \sum_{I\subseteq[n]} \sum_{I'\subseteq[n]|I\cap I'\neq\emptyset} \mu_a(\alpha_I) \cdot \mu_b(\beta_{I'})
\end{aligned}
$$

Meanwhile, for any I, I', if $|I \cap I'| \geq 2$, then there exists some j, k such that

$j \in I \cap I'$ and $k \in I \cap I'$, so

$$\begin{aligned}
\mu_a(\alpha_I) \cdot \mu_b(\beta_{I'}) &= \mu_a(\cap_{i\in I} A_i \setminus (\cup_{i\in[n]\setminus I} A_i)) \cdot \mu_b(\cap_{i\in I} B_i \setminus (\cup_{i\in[n]\setminus I} B_i)) \\
&\le \mu_a(A_j \cap A_k) \cdot \mu_b(B_j \cap B_k) \\
&= \mu(A_j \cap A_k \cap B_j \cap B_k) \\
&= \mu((A_j \cap B_j) \cap (A_k \cap B_k)) \\
&= \mu(\emptyset) \\
&= 0.
\end{aligned}$$

Thus, continuing our previous derivation,

$$\begin{aligned}
&\mu(E_a^n \cap E_b^n) \\
&= \sum_{I\subseteq[n]} \sum_{I'\subseteq[n]|I\cap I'\neq\emptyset} \mu_a(\alpha_I) \cdot \mu_b(\beta_{I'}) \\
&= \sum_{I\subseteq[n]} \sum_{I'\subseteq[n]|1=|I\cap I'|} \mu_a(\alpha_I) \cdot \mu_b(\beta_{I'}) \quad \text{(Because } \mu_a(\alpha_I) \cdot \mu_b(\beta_{I'}) = 0 \text{ if } |I \cap I'| \ge 2) \\
&= \sum_{i\in[n]} \sum_{I\subseteq[n]|i\in I} \sum_{I'\subseteq[n]|I\cap I'=\{i\}} \mu_a(\alpha_I) \cdot \mu_b(\beta_{I'}) \\
&= \sum_{i\in[n]} \sum_{I\subseteq[n]|i\in I} \sum_{I'\subseteq[n]|i\in I'} \mu_a(\alpha_I) \cdot \mu_b(\beta_{I'}) \\
&\qquad\qquad \text{(Because } \mu_a(\alpha_I) \cdot \mu_b(\beta_{I'}) = 0 \text{ if } |I \cap I'| \ge 2) \\
&= \sum_{i\in[n]} \left(\sum_{I\subseteq[n]|i\in I} \mu_a(\alpha_I) \cdot \sum_{I'\subseteq[n]|i\in I'} \mu_b(\beta_{I'}) \right) \\
&= \sum_{i\in[n]} \mu_a(A_i) \cdot \mu_b(B_i) \\
&= \sum_{i\in[n]} \mu(A_i \cap B_i)
\end{aligned}$$

Thus, we established the finite additivity. For countable additivity, suppose $E_a \cap E_b = \biguplus_{i\in\mathbb{N}}(A_i \cap B_i)$. By the same reason as above, we also have

$$E_a \cap E_b = (\cup_{i\in\mathbb{N}} A_i) \cap (\cup_{i\in\mathbb{N}} B_i) = \cup_{i\in\mathbb{N}}(A_i \cap B_i) = E_a \cap E_b.$$

Then,

$$
\begin{aligned}
&\mu(E_a \cap E_b) \\
&= \mu((\cup_{i\in\mathbb{N}} A_i) \cap (\cup_{i\in\mathbb{N}} B_i)) \\
&= \mu_a(\cup_{i\in\mathbb{N}} A_i) \cdot \mu_b(\cup_{i\in\mathbb{N}} B_i) \\
&= \mu_a(\lim_{n\to\infty} \cup_{i\in[n]} A_i) \cdot \mu_b(\lim_{n\to\infty} \cup_{i\in[n]} B_i) \\
&= \lim_{n\to\infty} \mu_a(\cup_{i\in[n]} A_i) \cdot \lim_{n\to\infty} \mu_b(\cup_{i\in[n]} B_i) && \text{(By continuity of } \mu_a \text{ and } \mu_b) \\
&= \lim_{n\to\infty} \mu_a(\cup_{i\in[n]} A_i) \cdot \mu_b(\cup_{i\in[n]} B_i) && (\dagger) \\
&= \lim_{n\to\infty} \sum_{i\in[n]} \mu_b(B_i) \cdot \mu_a(A_i) && \text{(By eq. (D.1))} \\
&= \sum_{i\in\mathbb{N}} \mu_b(B_i) \cdot \mu_a(A_i), && \text{(D.2)}
\end{aligned}
$$

where (†) holds because that the product of limits equals to the limit of the product when both $\lim_{n\to\infty} \mu_a(\cup_{i\in[n]} A_i)$ and $\lim_{n\to\infty} \mu_b(\cup_{i\in[n]} B_i)$ are finite. Thus, we proved countable additivity as well.

- Next we show that we can **extend μ to a measure on $\mathcal{F}_a \oplus \mathcal{F}_b$**.

  So far, we proved that μ is a sub-additive measure on the $\{E_a \cap E_b | E_a \in \mathcal{F}_a, E_b \in \mathcal{F}_b\}$, which forms a π-system. By a known theorem in probability theory (e.g. [Rosenthal, 2006, Corollary 2.5.4]), we can extend a sub-additive measure on a π-system to the σ-algebra it generates if the π-system is a semi-algebra. Thus, we can extend μ to a measure on $\sigma(\{E_a \cap E_b \mid E_a \in \mathcal{F}_a,\ E_b \in \mathcal{F}_b\})$ if we can prove $J = \{E_a \cap E_b \mid E_a \in \mathcal{F}_a,\ E_b \in \mathcal{F}_b\}$ is a semi-algebra.

  - J contains $\emptyset$ and Ω: trivial.
  - J is closed under finite intersection: $(E_a \cap E_b) \cap (E'_a \cap E'_b) = (E_a \cap E'_a) \cap (E_b \cap E'_b)$, where $E_a \cap E'_a \in \mathcal{F}_a$, and $E_b \cap E'_b \in \mathcal{F}_b$.

– The complement of any element of J is equal to a finite disjoint union of elements of J:

$$\begin{aligned}(E_a \cap E_b)^C &= E_a^C \cup E_b^C \\ &= (E_a^C \cap \Omega) \uplus (E_a \cap E_b^C)\end{aligned}$$

where $E_a^C, E_a \in \mathcal{F}_a$, and $E_b^C, \Omega \in \mathcal{F}_b$.

As shown in Li et al. [2023a],

$$\sigma(\{E_a \cap E_b \mid E_a \in \mathcal{F}_a, E_b \in \mathcal{F}_b\}) = \mathcal{F}_a \oplus \mathcal{F}_b \tag{D.3}$$

Thus, the extension of μ is a measure on $\mathcal{F}_a \oplus \mathcal{F}_b$.

- Last, we show that μ is a **probability measure** on $\mathcal{F}_a \oplus \mathcal{F}_b$: $\mu(\Omega) = \mu_a(\Omega) \cdot \mu_b(\Omega) = 1$. □

Lemma D.2.7. *Consider two probability spaces $(\mathcal{F}_1, \mu_1), (\mathcal{F}_2, \mu_2) \in \mathbb{P}(\Omega)$, and some other probability space (Σ_A, μ) and kernel κ such that $\mu_1 = \mathsf{bind}(\mu, \kappa)$.*

Then, the independent product $(\mathcal{F}_1, \mu_1) \circledast (\mathcal{F}_2, \mu_2)$ exists if and only if for any $a \in \mathrm{supp}(\mu)$, the independent product $(\mathcal{F}_1, \kappa(a)) \circledast (\mathcal{F}_2, \mu_2)$ exists. When they both exist,

$$(\mathcal{F}_1, \mu_1) \circledast (\mathcal{F}_2, \mu_2) = (\mathcal{F}_1 \oplus \mathcal{F}_2, \mathsf{bind}(\mu, \lambda a.\, \kappa(a) \circledast \mu_2))$$

Proof. We first show the backwards direction. By lemma D.2.6, for any $a \in \mathrm{supp}(\mu)$, to show that the independent product $(\mathcal{F}_1, \kappa(a)) \circledast (\mathcal{F}_1, \mu_1)$ exists, it suffices to show that for any $E_1 \in \mathcal{F}_1, E_2 \in \mathcal{F}_2$ such that $E_1 \cap E_2 = \emptyset$, $\kappa(a)(E_1) \cdot \mu_2(E_2) = 0$.

Fix any such E_1, E_2, because $(\mathcal{F}_1, \mu_1) \circledast (\mathcal{F}_2, \mu_2)$ is defined, we have $\mu_1(E_1) \cdot \mu_2(E_2) = 0$, then either $\mu_1(E_1) = 0$ or $\mu_2(E_2) = 0$.

- If $\mu_1(E_1) = 0$: Recall that

$$\mu_1(E_1) = \mathsf{bind}(\mu, \kappa)(E_1) = \sum_{a \in A} \mu(a) \cdot \kappa(a)(E_1) = \sum_{a \in \mathrm{supp}(\mu)} \mu(a) \cdot \kappa(a)(E_1)$$

  Because all $\mu(a) > 0$ and $\kappa(a)(E_1) \geq 0$ for all $a \in \mathrm{supp}(\mu)$ $\sum_{a \in \mathrm{supp}(\mu)} \mu(a) \cdot \kappa(a)(E_1) = 0$ implies that $\mu(a) \cdot \kappa(a)(E_1) = 0$ for all $a \in \mathrm{supp}(\mu)$. Thus, for all $a \in \mathrm{supp}(\mu)$, it must $\kappa(a)(E_1) = 0$. Therefore, $\kappa(a)(E_1) \cdot \mu_2(E_2) = 0$ for all $a \in \mathrm{supp}(\mu)$ with this E_1, E_2.

- If $\mu_2(E_2) = 0$, then it is also clear that $\kappa(a)(E_1) \cdot \mu_2(E_2) = 0$ for all $a \in \mathrm{supp}(\mu)$.

Thus, we have $\kappa(a)(E_1) \cdot \mu_2(E_2) = 0$ for any $E_1 \cap E_2 = \emptyset$ and $a \in \mathrm{supp}(\mu)$. By lemma D.2.6, the independent product $(\mathcal{F}_1, \kappa(a)) \circledast (\mathcal{F}_1, \mu_1)$ exists.

For the forward direction: for any $E_1 \in \mathcal{F}_1$ and $E_2 \in \mathcal{F}_2$ such that $E_1 \cap E_2 = \emptyset$, the independent product $(\mathcal{F}_1, \kappa(a)) \circledast (\mathcal{F}_2, \mu_2)$ exists implies that

$$\kappa(a)(E_1) \cdot \mu_2(E_2) = 0.$$

Thus,

$$\begin{aligned}
\mu_1(E_1) \cdot \mu_2(E_2) &= \mathsf{bind}(\mu, \kappa)(E_1) \cdot \mu_2(E_2) \\
&= \left(\sum_{a \in A} \mu(a) \cdot \kappa(a)(E_1) \right) \cdot \mu_2(E_2) \\
&= \sum_{a \in A_\mu} \mu(a) \cdot (\kappa(a)(E_1) \cdot \mu_2(E_2)) \\
&= \sum_{a \in A_\mu} \mu(a) \cdot 0 = 0
\end{aligned}$$

Thus, by lemma D.2.6, the independent product $(\mathcal{F}_1, \mu_1) \circledast (\mathcal{F}_2, \mu_2)$ exists. For

any $E_1 \in \mathcal{F}_1$ and $E_2 \in \mathcal{F}_2$,

$$
\begin{aligned}
&\mathsf{bind}(\mu, \lambda a.\, \kappa(a) \circledast \mu_2)(E_1 \cap E_2) \\
&= \sum_{a \in \mathrm{supp}(\mu)} \mu(a) \cdot (\kappa(a) \circledast \mu_2)(E_1 \cap E_2) \\
&= \sum_{a \in \mathrm{supp}(\mu)} \mu(a) \cdot \kappa(a)(E_1) \cdot \mu_2(E_2) \\
&= \left(\sum_{a \in \mathrm{supp}(\mu)} \mu(a) \cdot \kappa(a)(E_1) \right) \cdot \mu_2(E_2) \\
&= \mathsf{bind}(\mu, \kappa)(E_1) \cdot \mu_2(E_2) \\
&= \mu_1(E_1) \cdot \mu_2(E_2) \\
&= (\mu_1 \circledast \mu_2)(E_1 \cap E_2)
\end{aligned}
$$

Thus, $(\mathcal{F}_1, \mu_1) \circledast (\mathcal{F}_2, \mu_2) = (\mathcal{F}_1 \oplus \mathcal{F}_2, \mathsf{bind}(\mu, \lambda a.\, \kappa(a) \circledast \mu_2))$. □

D.3 Construction of the BLUEBELL Model

Lemma D.3.1. *The structure* PSp *is an ordered unital resource algebra (RA) as defined in definition 5.3.1.*

Proof. We defined $\cdot$ and $\preceq$ the same way as in Li et al. [2023a], and they have proved that $\cdot$ is associative and commutative, and $\preceq$ is transitive and reflexive. We check the rest of conditions one by one.

Condition $a \cdot b = b \cdot a$ The independent product is proved to be commutative in Li et al. [2023a].

Condition $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ The independent product is proved to be associative in Li et al. [2023a].

Condition $a \preceq b \Rightarrow b \preceq c \Rightarrow a \preceq c$ The order $\preceq$ is proved to be transitive in Li et al. [2023a].

Condition $a \preceq a$ The order $\preceq$ is proved to be reflexive in Li et al. [2023a].

Condition $\mathbf{V}(a \cdot b) \Rightarrow \mathbf{V}(a)$ Pattern matching on $a \cdot b$, either there exists probability spaces $\mathcal{P}_1, \mathcal{P}_2$ such that $a = \mathcal{P}_1$, $b = \mathcal{P}_2$ and $\mathcal{P}_1 \circledast \mathcal{P}_2$ is defined, or $a \cdot b = \lightning$.

Case: $a \cdot b = \lightning$ Note that $\mathbf{V}(a \cdot b)$ does not hold when $a \cdot b = \lightning$, so we can eliminate this case by ex falso quodlibet.

Case: $a \cdot b = \mathcal{P}_1 \circledast \mathcal{P}_2$ Then $a = \mathcal{P}_1$, and thus $\mathbf{V}(a)$.

Condition $\mathbf{V}(\varepsilon)$ Clear because $\varepsilon \neq \lightning$.

Condition $a \preceq b \Rightarrow \mathbf{V}(b) \Rightarrow \mathbf{V}(a)$ Pattern matching on a and b, either there exists probability spaces $\mathcal{P}_1, \mathcal{P}_2$ such that $a = \mathcal{P}_1$, $b = \mathcal{P}_2$ and $\mathcal{P}_1 \sqsubseteq \mathcal{P}_2$ is defined, or $b = \lightning$.

Case: $b = \lightning$ Then $\mathbf{V}(b)$ does not hold, and we can eliminate this case by ex falso quodlibet.

Case: $a = \mathcal{P}_1$**,** $b = \mathcal{P}_2$ **and** $\mathcal{P}_1 \sqsubseteq \mathcal{P}_2$ We clearly have $\mathbf{V}(a)$.

Condition $\varepsilon \cdot a = a$ Pattern matching on a, either $a = \lightning$ or there exists some probability space $\mathcal{P}$ such that $a = \mathcal{P}$.

Case: $a = \lightning$ Then $\varepsilon \cdot a = \lightning = a$.

Case: $a = \mathcal{P}$ Then $\varepsilon \cdot a = a$.

Condition $a \preceq b \Rightarrow a \cdot c \preceq b \cdot c$ Pattern matching on a and b. If $a \preceq b$, then either $b = \lightning$ or there exists $\mathcal{P}, \mathcal{P}'$ such that $a = \mathcal{P}$ and $b = \mathcal{P}'$.

Case: $b = \lightning$ Then $b \cdot c = \lightning$ is the top element, and then $a \cdot c \preceq b \cdot c$.

Otherwise $a \preceq b$ iff $\mathcal{P} \preceq \mathcal{P}'$, then either $b \cdot c = \lightning$ and $a \cdot c \preceq b \cdot c$ follows, or $b \cdot c = \mathcal{P}' \circledast \mathcal{P}''$ for some probability space $c = \mathcal{P}''$. Then $\mathcal{P} \preceq \mathcal{P}'$ implies that $\mathcal{P} \cdot \mathcal{P}''$ is also defined and $\mathcal{P} \cdot \mathcal{P}' \preceq \mathcal{P} \cdot \mathcal{P}''$. Thus, $a \cdot c \preceq b \cdot c$ too. □

Lemma D.3.2 (RA composition preserves compatibility)**.**

$$\mathcal{F}_1 \# p_1 \Rightarrow \mathcal{F}_2 \# p_2 \Rightarrow (\mathcal{F}_1 \oplus \mathcal{F}_2) \# (p_1 \cdot p_2)$$

Proof. Let $S_1 = \{x \in \mathbf{Var} \mid p_1(x) = 0\}$, $S_2 = \{x \in \mathbf{Var} \mid p_2(x) = 0\}$. If $\mathcal{F}_1 \# p_1$, then there exists $\mathcal{P}_1' \in \mathbb{P}((\mathbf{Var} \setminus S_1) \to \mathbf{Val})$ such that $\mathcal{P}_1 = \mathcal{P}_1' \otimes \mathbb{1}_{S_1 \to \mathbf{Val}}$ In addition, if $\mathcal{F}_2 \# p_2$, then there exists $\mathcal{P}_2' \in \mathbb{P}((\mathbf{Var} \setminus S_2) \to \mathbf{Val})$ such that $\mathcal{P}_2 = \mathcal{P}_2' \otimes \mathbb{1}_{S_2 \to \mathbf{Val}}$. Then,

$$\begin{aligned}\mathcal{P}_1 \cdot \mathcal{P}_2 &= \mathcal{P}_1 \circledast \mathcal{P}_2 \\ &= (\mathcal{P}_1' \otimes \mathbb{1}_{S_1 \to \mathbf{Val}}) \circledast (\mathcal{P}_2' \otimes \mathbb{1}_{S_2 \to \mathbf{Val}})\end{aligned}$$

Say $(\mathcal{F}_1', \mu_1') = \mathcal{P}_1'$, and $(\mathcal{F}_2', \mu_2') = \mathcal{P}_2'$. Then the sigma algebra of $\mathcal{P}_1 \cdot \mathcal{P}_2$ is

$$\begin{aligned}&\sigma(\left\{(E_1 \times S_1 \to \mathbf{Val}) \cap (E_2 \times S_2 \to \mathbf{Val}) \mid E_1 \in \mathcal{F}_1', E_2 \in \mathcal{F}_2'\right\}) \\ =&\sigma(\left\{((E_1 \times (S_1 \setminus S_2) \to \mathbf{Val}) \cap (E_2 \times (S_2 \setminus E_1) \to \mathbf{Val})) \times (S_1 \cap S_2) \mid E_1 \in \mathcal{F}_1', E_2 \in \mathcal{F}_2'\right\})\end{aligned}$$

Then, there exists $\mathcal{P}'' \in \mathbb{P}((\mathbf{Var} \setminus (S_1 \cap S_2)) \to \mathbf{Val})$ such that $\mathcal{P}_1 \cdot \mathcal{P}_2 = \mathcal{P}'' \otimes \mathbb{1}_{(S_1 \cap S_2) \to \mathbf{Val}})$. Also,

$$\begin{aligned}&\{x \in \mathbf{Var} \mid (p_1 \cdot p_2)(x) = 0\} \\ =&\{x \in \mathbf{Var} \mid p_1(x) + p_2(x) = 0\} \\ =&\{x \in \mathbf{Var} \mid p_1(x) = 0 \text{ and } p_2(x) = 0\} \\ =&S_1 \cap S_2\end{aligned}$$

Therefore, $\mathcal{F}_1 \oplus \mathcal{F}_2$ is compatible with $p_1 \cdot p_2$ □

Lemma D.3.3. *The structure* $(\mathsf{Perm}, \preceq, \mathbf{V}, \cdot, \varepsilon)$ *is an ordered unital resource algebra (RA) as defined in definition 5.3.1.*

Proof. We check the conditions one by one.

Condition $a \cdot b = b \cdot a$ Follows from the commutativity of addition.

Condition $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ Follows from the associativity of addition.

Condition $a \preceq b \Rightarrow b \preceq c \Rightarrow a \preceq c$ $\preceq$ is a point-wise lifting of the order $\leq$ on arithmetics, so it follows from the transitivity of $\leq$.

Condition $a \preceq a$ $\preceq$ is a point-wise lifting of the order $\leq$ on arithmetics, so it follows from the reflexivity of $\leq$.

Condition $\mathbf{V}(a \cdot b) \Rightarrow \mathbf{V}(a)$ By definition,

$$\begin{aligned}
\mathbf{V}(a \cdot b) &\Rightarrow \forall x \in \mathbf{Var}, (a \cdot b)(x) \leq 1 \\
&\Rightarrow \forall x \in \mathbf{Var}, a(x) + b(x) \leq 1 \\
&\Rightarrow \forall x \in \mathbf{Var}, a(x) \leq 1 \\
&\Rightarrow \mathbf{V}(a)
\end{aligned}$$

Condition $\mathbf{V}(\varepsilon)$ Note that $\varepsilon = \lambda\_.\, 0$ satisfies that $\forall x \in \mathbf{Var}, \varepsilon(x) \leq 1$, so $\mathbf{V}(\varepsilon)$.

Condition $a \preceq b \Rightarrow \mathbf{V}(b) \Rightarrow \mathbf{V}(a)$ By definition, $a \preceq b$ means $\forall x \in \mathbf{Var}. a(x) \leq b(x)$, and $\mathbf{V}(b)$ means that $\forall x \in \mathbf{Var}. b(x) \leq 1$. Thus, $a \preceq b$ and $\mathbf{V}(b)$ implies that $\forall x \in \mathbf{Var}. a(x) \leq b(x) \leq 1$, which implies $\mathbf{V}(a)$.

Condition $\varepsilon \cdot a = a$ By definition,

$$\begin{aligned}
\varepsilon \cdot a &= \lambda x.\, (\lambda_.\, 0)(x) + a(x) \\
&= \lambda x.\, 0 + a(x) \\
&= a.
\end{aligned}$$

Condition $a \preceq b \Rightarrow a \cdot c \preceq b \cdot c$ By definition,

$$\begin{aligned} a \preceq b &\Leftrightarrow \forall x \in \mathbf{Var}. a(x) \leq b(x) \\ &\Rightarrow \forall x \in \mathbf{Var}. a(x) + c(x) \leq b(x) + c(x) \\ &\Rightarrow a \cdot c \preceq b \cdot c \end{aligned}$$

□

Lemma D.3.4. *The structure* PSpPm *is an ordered unital resource algebra (RA) as defined in definition 5.3.1.*

Proof. We want to check that PSpPm satisfies all the requirements to be an ordered unital resource algebra (RA). Because PSpPm is very close to a product of PSp and Perm, the proof below is very close to the proof that product RAs are RA.

First, lemma D.3.2 implies that $\cdot$ is well-defined.

Then we need to check all the RA axioms are satisfied. For any $a, b \in$ PSpPm and any $\mathcal{P}_1, p_1, \mathcal{P}_2, p_2$ such that $a = (\mathcal{P}_1, p_1), b = (\mathcal{P}_2, p_2)$.

We check the conditions one by one.

Condition $\mathbf{V}(a \cdot b) \Rightarrow \mathbf{V}(a)$ By definition, $a \cdot b = (\mathcal{P}_1, p_1) \cdot (\mathcal{P}_2, p_2) = (\mathcal{P}_1 \cdot \mathcal{P}_2, p_1 \cdot p_2)$. And $\mathbf{V}(\mathcal{P}_1 \cdot \mathcal{P}_2, p_1 \cdot p_2)$ implies that $\mathbf{V}(\mathcal{P}_1 \cdot \mathcal{P}_2)$ and $\mathbf{V}(p_1 \cdot p_2)$. Because PSp and Perm are both RAs, we have $\mathbf{V}(\mathcal{P}_1)$ and $\mathbf{V}(p_1)$. Thus, $\mathbf{V}(\mathcal{P}_1, p_1)$.

Condition $\mathbf{V}(\varepsilon)$ Clear because $\varepsilon = (\mathbb{1}_{\mathbf{Mem[Var]}}, \lambda x. 0)$ and $\mathbb{1}_{\mathbf{Mem[Var]}} \neq \lightning$, and $\forall x. (\lambda x. 0)(x) \leq 1$.

Condition $a \preceq b \Rightarrow \mathbf{V}(b) \Rightarrow \mathbf{V}(a)$ $a \preceq b$ implies that $\mathcal{P}_1 \preceq \mathcal{P}_2$ and $p_1 \preceq p_2$. $\mathbf{V}(b)$ implies that $\mathcal{P}_2 \neq \lightning$, and $\forall x. (p_2)(x) \leq 1$. Thus, $\mathcal{P}_1 \neq \lightning$, and $\forall x. (p_1)(x) \leq 1$. And therefore, $\mathbf{V}(b)$.

Condition $\varepsilon \cdot a = a$ $\varepsilon \cdot a = (\mathbb{1}_{\mathbf{Mem[Var]}}, \lambda x.\, 0) \cdot (\mathcal{P}_1, p_1)$

$= (\mathbb{1}_{\mathbf{Mem[Var]}} \cdot \mathcal{P}_1, \lambda x.\, 0 \cdot p_1)$

$= (\mathcal{P}_1, p_1) = a$.

Condition $a \preceq b \Rightarrow a \cdot c \preceq b \cdot c$ $a \preceq b$ implies that $\mathcal{P}_1 \preceq \mathcal{P}_2$ and $p_1 \preceq p_2$.

Say $c = (\mathcal{P}_3, p_3)$. Then $a \cdot c = (\mathcal{P}_1 \cdot \mathcal{P}_3, p_1 \cdot p_3)$ and $b \cdot c = (\mathcal{P}_2 \cdot \mathcal{P}_3, p_2 \cdot p_3)$. Because $\mathcal{P}_1 \preceq \mathcal{P}_2$, $\mathcal{P}_1 \cdot \mathcal{P}_3 \preceq \mathcal{P}_2 \cdot \mathcal{P}_3$; similarly, $p_1 \cdot p_3 \preceq p_2 \cdot p_3$. Thus, $a \cdot c \preceq b \cdot c$. □

Lemma D.3.5. *If M is an RA, then M^I is also an RA.*

Proof. RA is known to be closed under products, and M^I can be obtained as products of M, so we omit the proof. □

Lemma D.3.6. *$\mathcal{M}_I$ is an RA.*

Proof. By lemma D.3.4, PSpPm is an RA. By lemma D.3.5, $\mathcal{M}_I = \mathsf{PSpPm}^I$ is also an RA. □

D.4 Characterizations of Joint Conditioning

Interestingly, it is possible to characterize the conditioning modality using the other connectives of the logic.

Proposition D.4.1 (Alternative Characterization of Joint conditioning)**.** *The following is a logically equivalent characterization of the joint conditioning modality:*

$$\boldsymbol{C}_\mu\, K \dashv\vdash \exists \mathcal{F}, \mu, p, \kappa.\, \mathsf{Own}(\mathcal{F}, \mu, p) * \ulcorner \forall i \in I.\, \mu(i) = \mathit{bind}(\mu, \kappa(i)) \urcorner \\ * \forall v \in \mathrm{supp}(\mu).\, \mathsf{Own}(\mathcal{F}, \kappa(I)(v), p) \mathrel{-\!\!*} K(v)$$

Proof. In the following, we sometimes abbreviate $\forall i \in I.\, \mu(i) = \mathsf{bind}(\mu, \kappa(i))$ by writing just $\mu = \mathsf{bind}(\mu, \kappa)$.

We start with the embedding:

$$\begin{aligned}
&\exists \mathcal{F}, \mu, p, \kappa.\, \mathsf{Own}(\mathcal{F}, \mu, p) * \ulcorner \forall i \in I.\, \mu(i) = \mathsf{bind}(\mu, \kappa(i)) \urcorner \\
&\qquad * \forall a \in \mathrm{supp}(\mu).\, \mathsf{Own}(\mathcal{F}, \kappa(I)(a), p) \mathrel{-\!\!*} K(a) \\
\dashv\vdash\ & \lambda r.\, \exists \mathcal{F}, \mu, p, \kappa. \big(\mathsf{Own}(\mathcal{F}, \mu', p) * \ulcorner \mu = \mathsf{bind}(\mu, \kappa) \urcorner * \\
&\qquad (\forall a \in \mathrm{supp}(\mu). \mathsf{Own}(\mathcal{F}, \kappa a, p) \mathrel{-\!\!*} K(a))\big)(r) \\
\dashv\vdash\ & \lambda r.\, \exists \mathcal{F}, \mu, p, \kappa, \mathcal{F}_1, \mu_1, p_1, \mathcal{F}_2, \mu_2, p_2, \mathcal{F}_3, \mu_3, p_3, \\
&\qquad r \sqsupseteq (\mathcal{F}_1, \mu_1, p_1) \cdot (\mathcal{F}_2, \mu_2, p_2) \cdot (\mathcal{F}_3, \mu_3, p_3) \wedge \\
&\qquad (\mathcal{F}_1, \mu_1, p_1) \sqsupseteq (\mathcal{F}, \mu, p) \wedge \ulcorner \mu = \mathsf{bind}(\mu, \kappa) \urcorner \wedge \\
&\qquad (\forall a \in \mathrm{supp}(\mu). \forall r_1, r_2.\, r_1 \cdot (\mathcal{F}_3, \mu_3, p_3) = r_2 \wedge r_1 \sqsupseteq (\mathcal{F}, \kappa a, p) \Rightarrow K(a)(r_2)) \\
\dashv\vdash\ & \lambda r.\, \exists \mathcal{F}, \mu, p, \mathcal{F}_3, \mu_3, p_3, \kappa. \\
&\qquad r \sqsupseteq (\mathcal{F}, \mu, p) \cdot (\mathcal{F}_3, \mu_3) \wedge \ulcorner \mu = \mathsf{bind}(\mu, \kappa) \urcorner \wedge \\
&\qquad (\forall a \in \mathrm{supp}(\mu). \forall r_1, r_2.\, r_1 \cdot (\mathcal{F}_3, \mu_3, p_3) = r_2 \wedge r_1 \sqsupseteq (\mathcal{F}, \kappa a, p) \Rightarrow K(a)(r_2))
\end{aligned}$$

For the last equivalence, the forward direction holds because

$$\begin{aligned}
r &\sqsupseteq (\mathcal{F}_1, \mu_1, p_1) \cdot (\mathcal{F}_2, \mu_2, p_2) \cdot (\mathcal{F}_3, \mu_3, p_3) \\
&\sqsupseteq (\mathcal{F}_1, \mu_1, p_1) \cdot (\mathcal{F}_3, \mu_3, p_3) \\
&\sqsupseteq (\mathcal{F}, \mu, p) \cdot (\mathcal{F}_3, \mu_3, p_3).
\end{aligned}$$

The backward direction holds because we can pick $(\mathcal{F}_1, \mu_1, p_1) = (\mathcal{F}, \mu, p)$, $(\mathcal{F}_2, \mu_2)$ be the trivial probability space on s and $p_2 = \lambda\_.\, 0$.

- To show that the embedding implies the original assertion $\boldsymbol{C}_\mu\, K$, we start

with $\mu(i) \circledast \mu_3(i)$. For any i, we have $\mu(i) = \mathsf{bind}(\mu, \kappa(i))$, and thus

$$\mu(i) \circledast \mu_3(i) = \mathsf{bind}(\mu, \kappa(i)) \circledast \mu_3(i).$$

According to lemma D.2.7, $\mu(i) \circledast \mu_3(i)$ is defined implies that $\kappa(i)(a) \circledast \mu_3(i)$ is defined for any $a \in$. Furthermore,

$$\mu(i) \circledast \mu_3(i) = \mathsf{bind}(\mu, \lambda a.\, \kappa(i)(a) \circledast \mu_3(i))$$

We abbreviate the hyperkernel $[i{:}\, \lambda a.\, \kappa(i)(a) \circledast \mu_3(i) \mid i \in I]$ as κ'. For any $a \in \mathrm{supp}(\mu)$, the assertion

$$\forall a \in \mathrm{supp}(\mu). \forall r_1, r_2. r_1 \circledast (\mathcal{F}_3, \mu_3, p_3) = r_2 \wedge r_1 \sqsupseteq (\mathcal{F}, \kappa(I)a, p) \Rightarrow K(a)(r_2)$$

applies with the specific case $r_1 = (\mathcal{F}, \kappa(I)(a), p)$, gives us

$$K(a)((\mathcal{F}, \kappa(I)(a), p) \cdot (\mathcal{F}_3, \mu_3, p_3)])$$

By the definition of composition in our resource algebra, we have that $K(a)$ holds on $(\mathcal{F} \oplus \mathcal{F}_3, \kappa'(I)(a), p + p_3)$.

For any r,

– If $\mathbf{V}(r)$, then there exists $\mathcal{F}', \mu', p'$ such that $r = (\mathcal{F}', \mu', p')$. Note that

$$r = (\mathcal{F}', \mu', p') \sqsupseteq (\mathcal{F}, \mu, p) \cdot (\mathcal{F}_3, \mu_3, p_3) = (\mathcal{F} \oplus \mathcal{F}_3, \mu \circledast \mu_3, p + p_3)$$

By lemma D.2.4, $\mu \circledast \mu_3 = \mathsf{bind}(\mu, \kappa')$ implies that there exists κ'' such that $\mu(i) = \mathsf{bind}(\mu, \kappa''(i))$, and that for any $a \in \mathrm{supp}\,\mu$, $(\mathcal{F} \oplus \mathcal{F}_3, \kappa'(I)(a)) \sqsubseteq (\mathcal{F}', \kappa''(I)(a))$. Thus, by monotonicity with respect to the extension order, that would imply $K(a)$ holds on $(\mathcal{F}', \kappa''(I)(a), p')$. And $K(a)$ holds on $(\mathcal{F}', \kappa''(I)(a), p')$ for any $a \in \mathrm{supp}\,\mu$ together with $\mu(i) = \mathsf{bind}(\mu, \kappa''(i))$ implies that r satisfy the original assertion of conditioning modality.

- If not $\mathbf{V}(r)$, then r satisfies any assertions, so r satisfy the original assertion of conditioning modality.

- To show the other direction that having the original assertion implies the embedded assertion. Assume $\boldsymbol{C}_\mu\, K(r)$, that is,

$$\exists \mathcal{F}, \mu, p, \kappa.\, (\mathcal{F}, \mu, p) \preceq r \wedge \forall i \in I.\, \mu(i) = \mathsf{bind}(\mu, \kappa(i))(r)$$
$$\wedge\, \forall v \in \mathrm{supp}(\mu).K(v)(\mathcal{F}, \kappa(I)(v), p)$$

  To show that r also satisfy the embedding, we pick the witness for the existential quantifier as follows: let $(\mathcal{F}_3, \mu_3)$ be the trivial probability space on $\mathbf{Mem}[\mathbf{Var}]$; let $p_3 = \lambda\_.\, 0$; pick $(\mathcal{F}_{\mathrm{embd}}, \mu_{\mathrm{embd}}, p_{\mathrm{embd}})$ be the $(\mathcal{F}_{\mathrm{orig}}, \mu_{\mathrm{orig}}, p_{\mathrm{orig}})$ that witness $\boldsymbol{C}_\mu\, K(r)$, and $\kappa_{\mathrm{embd}} = \kappa_{\mathrm{orig}}$.

  Then:

  - First we show

$$\begin{aligned} r &\succeq (\mathcal{F}_{\mathrm{orig}}, \mu_{\mathrm{orig}}, p_{\mathrm{orig}}) \\ &= (\mathcal{F}_{\mathrm{orig}}, \mu_{\mathrm{orig}}, p_{\mathrm{orig}}) \cdot (\mathcal{F}_3, \mu_3, p_3) \\ &= (\mathcal{F}_{\mathrm{embd}}, \mu_{\mathrm{embd}}, p_{\mathrm{embd}}) \cdot (\mathcal{F}_3, \mu_3, p_3) \end{aligned}$$

  - $\mu_{\mathrm{orig}} = \mathsf{bind}(\mu, \kappa_{\mathrm{orig}}(I)(a))$ implies $\mu_{\mathrm{embd}} = \mathsf{bind}(\mu, \kappa_{\mathrm{embd}}(I)(a))$.

  - For any r_1, r_2,

$$r_1 \cdot (\mathcal{F}_3, \mu_3, p_3) = r_2 \wedge r_1 \sqsupseteq (\mathcal{F}_{\mathrm{embd}}, \kappa_{\mathrm{embd}}(I)(a), p_{\mathrm{embd}})$$

    implies that $r_2 = r_1 \sqsupseteq (\mathcal{F}_{\mathrm{orig}}, \kappa_{\mathrm{orig}}(I)(a), p_{\mathrm{orig}})$. By the assumption that the orig assertion holds, we have $K(a)(\mathcal{F}_{\mathrm{orig}}, \kappa_{\mathrm{orig}}(I)(a), p_{\mathrm{orig}})$, which implies $K(a)(r_2)$.

  Therefore, r also satisfy the embedding. □

D.5 Soundness

D.5.1 Soundness of Primitive Rules

Soundness of Distribution Ownership Rules

Lemma D.5.1. DIST-INJ *is sound.*

Proof. Assume a valid $a \in \mathcal{M}_I$ is such that both $E \twoheadrightarrow \mu(a)$ and $E \twoheadrightarrow \mu'(a)$ hold. Let $a = (\mathcal{F}, \mu_0, p)$, then we know $\mu = \mu_0 \circ E^{-1} = \mu'$, which proves the claim. □

Lemma D.5.2. SURE-MERGE *is sound.*

Proof. The proof for the forward direction is very similar to the one for section 5.3.5. For $a \in \mathcal{M}_I$, if $(\lceil E_1 \rceil * \lceil E_2 \rceil)(a)$. Then there exists a_1, a_2 such that $a_1 \cdot a_2 \preceq a$ and $\lceil E_1 \rceil(a_1)$, $\lceil E_2 \rceil(a_2)$. Say $a = (\mathcal{F}, \mu, p)$, $a_1 = (\mathcal{F}_1, \mu_1, p_1)$ and $a_2 = (\mathcal{F}_2, \mu_2, p_2)$. Then $\lceil E_1 \rceil(a_1)$ implies that

$$\mu_1(E_1^{-1}(\mathsf{True})) = 1$$

And similarly,

$$\mu_2(E_2^{-1}(\mathsf{True})) = 1$$

Thus,

$$\mu(E_1^{-1}(\mathsf{True}) \cap E_2^{-1}(\mathsf{True})) = \mu_1(E_1^{-1}(\mathsf{True})) \cdot \mu_2(E_2^{-1}(\mathsf{True})) = 1.$$

Hence,

$$\mu(E_1 \wedge E_2^{-1}(\mathsf{True})) = \mu(E_1^{-1}(\mathsf{True}) \cap E_2^{-1}(\mathsf{True})) = 1$$

Thus, $\lceil E_1 \wedge E_2 \rceil(a)$.

Now we prove the backwards direction: Say $a = (\mathcal{F}, \mu, p)$. if $\lceil E_1 \wedge E_2 \rceil(a)$, then $\mu(E_1 \wedge E_2^{-1}(\mathsf{True})) = 1$, and then

$$\mu(E_1^{-1}(\mathsf{True})) \geq \mu(E_1 \wedge E_2^{-1}(\mathsf{True})) = 1$$
$$\mu(E_2^{-1}(\mathsf{True})) \geq \mu(E_1 \wedge E_2^{-1}(\mathsf{True})) = 1$$

Let $\mathcal{F}_1 = \sigma(E_1^{-1}(\mathsf{True}))$ and $\mathcal{F}_2 = \sigma(E_2^{-1}(\mathsf{True}))$. Then,

$$\lceil E_1 \rceil(\mathcal{F}_1, \mu|_{\mathcal{F}_1}, \lambda\_. 0)$$
$$\lceil E_2 \rceil(\mathcal{F}_2, \mu|_{\mathcal{F}_2}, \lambda\_. 0)$$
$$(\mathcal{F}_1, \mu|_{\mathcal{F}_1}, \lambda\_. 0) * (\mathcal{F}_2, \mu|_{\mathcal{F}_2}, \lambda\_. 0) \preceq a$$

Thus, $\lceil E_1 \rceil * \lceil E_2 \rceil$ holds on a. □

Lemma D.5.3. PROD-SPLIT *is sound.*

Proof. For any $(\mathcal{F}, \mu, p)$ such that $((E_1, E_2) \twoheadrightarrow \mu_1 \otimes \mu_2)(\mathcal{F}, \mu, p)$, by definition, it must

$$\exists \mathcal{F}', \mu'. (\mathsf{Own}(\mathcal{F}', \mu'))(\mathcal{F}, \mu, p) * (E_1, E_2) \prec (\mathcal{F}'(i), \mu'(i)) \wedge \mu_1 \otimes \mu_2 = \mu'(i) \circ (E_1, E_2)^{-1}.$$

We can derive from it that

$$\exists \mathcal{F}', \mu', p'. (\mathcal{F}', \mu') \preceq (\mathcal{F}, \mu, p) *$$
$$\Big(\forall a, b \in A. \exists L_{a,b}, U_{a,b} \in \mathcal{F}'(i).\, L_{a,b} \subseteq (E_1, E_2)^{-1}(a, b) \subseteq U_{a,b} \wedge \mu'(L_{a,b}) = \mu'(U_{a,b}) \wedge$$
$$\mu_1 \otimes \mu_2(a, b) = \mu'(i)(L_{a,b}) = \mu'(i)(U_{a,b})\Big)$$

Also, for any $a, b, a', b' \in A$ such that $a \neq a'$ or $b \neq b'$, we have $L_{a,b}$ disjoint from $L_{a',b'}$ because on $L_{a,b} \cap L_{a',b'}$, the random variable (E_1, E_2) maps to both (a, b) and (a', b').

Define

$$\mathcal{F}_1(i) = \sigma(\{(\bigcup_{b\in A} L_{a,b}) \mid a \in A\} \cup \{(\bigcup_{b\in A} U_{a,b}) \mid a \in A\}),$$

and similarly define

$$\mathcal{F}_2(i) = \sigma(\{(\bigcup_{a\in A} L_{a,b}) \mid b \in A\} \cup \{(\bigcup_{a\in A} U_{a,b}) \mid b \in A\}).$$

Denote μ' restricted to $\mathcal{F}_1$ as μ'_1 and μ' restricted to $\mathcal{F}_2$ as μ'_2.

We want to show that $(\mathcal{F}_1(i), \mu'_1(i)) \circledast (\mathcal{F}_2(i), \mu'_2(i)) \sqsubseteq (\mathcal{F}'(i), \mu'(i))$, which boils down to show that for any $X_1 \in \mathcal{F}_1(i)$, any $X_2 \in \mathcal{F}_2(i)$,

$$\mu'(X_1 \cap X_2) = \mu'_1(X_1) \cdot \mu'_2(X_2)$$

For convenience, we will denote $\cup_{b\in A} L_{a,b}$ as L_a, denote $\cup_{a\in A} L_{a,b}$ as L_b, denote $\cup_{b\in A} U_{a,b}$ as U_a, and denote $\cup_{a\in A} U_{a,b}$ as U_b.

First, using a standard construction in measure theory proofs, we rewrite $\mathcal{F}_1$ and $\mathcal{F}_2$ as sigma algebra generated by sets of partitions. Specifically, $\mathcal{F}_1$ is equivalent to

$$\sigma(\{\bigcap_{a\in S_1} L_a \cap \bigcap_{a\in S_2} U_a \setminus (\bigcup_{a\in A\setminus S_1} L_a \cup \bigcup_{a\in A\setminus S_2} U_a) \mid S_1, S_2 \subseteq A\})$$

and similarly, $\mathcal{F}_2$ is equivalent to

$$\sigma(\{\bigcap_{b\in T_1} L_b \cap \bigcap_{b\in T_2} U_b \setminus (\bigcup_{b\in A\setminus T_1} L_b \cup \bigcup_{b\in A\setminus T_2} U_b) \mid T_1, T_2 \subseteq A\}).$$

Thus, by lemma D.2.2, any event X_1 in $\mathcal{F}_1$ can be represented by

$$\biguplus_{S_1\in I_1, S_2\in I_2} \bigcap_{a\in S_1} L_a \cap \bigcap_{a\in S_2} U_a \setminus (\bigcup_{a\in A\setminus S_1} L_a \cup \bigcup_{a\in A\setminus S_2} U_a)$$

for some $I_1, I_2 \subseteq \mathcal{P}(A)$, where $\mathcal{P}$ is the powerset over A. Similarly, any event X_2 in $\mathcal{F}_2$ can be represented by

$$\biguplus_{S_3\in I_3, S_4\in I_4} \bigcap_{b\in S_3} L_b \cap \bigcap_{b\in S_4} U_b \setminus (\bigcup_{b\in A\setminus S_3} L_b \cup \bigcup_{b\in A\setminus S_2} U_b)$$

for some $I_3, I_4 \subseteq \mathcal{P}(A)$. Thus, $X_1 \cap X_2$ can be represented as

$$\begin{aligned} X_1 \cap X_2 = &(\biguplus_{S_1 \in I_1, S_2 \in I_2} \bigcap_{a \in S_1} L_a \cap \bigcap_{a \in S_2} U_a \setminus (\bigcup_{a \in A \setminus S_1} L_a \cup \bigcup_{a \in A \setminus S_2} U_a)) \\ &\bigcap (\biguplus_{S_3 \in I_3, S_4 \in I_4} \bigcap_{b \in S_3} L_b \cap \bigcap_{b \in S_4} U_b \setminus (\bigcup_{b \in A \setminus S_3} L_b \cup \bigcup_{b \in A \setminus S_2} U_b)) \\ = &\biguplus_{S_1 \in I_1, S_2 \in I_2, S_3 \in I_3, S_4 \in I_4} (\bigcap_{a \in S_1} L_a \cap \bigcap_{a \in S_2} U_a \setminus (\bigcup_{a \in A \setminus S_1} L_a \cup \bigcup_{a \in A \setminus S_2} U_a)) \\ &\cap (\bigcap_{b \in S_3} L_b \cap \bigcap_{b \in S_4} U_b \setminus (\bigcup_{b \in A \setminus S_3} L_b \cup \bigcup_{b \in A \setminus S_2} U_b)) \end{aligned}$$

Because $L_{a,b}$ and $L_{a',b'}$ are disjoint as long as not $a = a'$ and $b = b'$, we have L_a disjoint from $L_{a'}$ if $a \neq a'$. Thus, $\bigcap_{a \in S_1} L_a \cap \bigcap_{a \in S_2} U_a \setminus (\bigcup_{a \in A \setminus S_1} L_a \cup \bigcup_{a \in A \setminus S_2} U_a)$ is not empty only when S_1 is singleton and empty.

- If S_1 is empty, then

  $\bigcap_{a \in S_1} L_a \cap \bigcap_{a \in S_2} U_a \setminus (\bigcup_{a \in A \setminus S_1} L_a \cup \bigcup_{a \in A \setminus S_2} U_a) = \bigcap_{a \in S_2} U_a \setminus (\bigcup_{a \in A} L_a \cup \bigcup_{a \in A \setminus S_2} U_a)$

  has measure 0 because $\bigcup_{a \in A} L_a$ has measure 1.

- Otherwise, if S_1 is singleton, say $S_1 = \{a'\}$, then

  $\bigcap_{a \in S_1} L_a \cap \bigcap_{a \in S_2} U_a \setminus (\bigcup_{a \in A \setminus S_1} L_a \cup \bigcup_{a \in A \setminus S_2} U_a) = L_{a'} \cap \bigcap_{a \in S_2} U_a \setminus \bigcup_{a \in A \setminus S_2} U_a)$.

  Furthermore,

  $$\begin{aligned} \mu'(\bigcap_{a \in S_2} U_a) &= \mu'(\bigcap_{a \in S_2} L_a \uplus (U_a \setminus L_a)) \\ &= \mu'(\bigcap_{a \in S_2} L_a) + 0 \end{aligned}$$

  And $\bigcap_{a \in S_2} L_a$ is non-empty only if S_2 is a singleton set or empty set. Thus, $L_{a'} \cap \bigcap_{a \in S_2} U_a \setminus \bigcup_{a \in A \setminus S_2} U_a) \subseteq \bigcap_{a \in S_2} U_a$ has non-zero measure only if S_2 is empty or a singleton set.

  - When S_2 is empty,

    $$L_{a'} \cap \bigcap_{a \in S_2} U_a \setminus \bigcup_{a \in A \setminus S_2} U_a = L_{a'} \setminus \bigcup_{a \in A} U_a \subseteq L_{a'} \setminus U_{a'} = \emptyset$$

– When $S_2 = \{a'\}$,

$$L_{a'} \cap \bigcap_{a\in S_2} U_a \setminus \bigcup_{a\in A\setminus S_2} U_a = L_{a'} \setminus \bigcup_{a\in A, a\neq a'} U_a.$$

– When $S_2 = \{a''\}$ for some $a'' \neq a'$

$$\begin{aligned} L_{a'} \cap \bigcap_{a\in S_2} U_a \setminus \bigcup_{a\in A\setminus S_2} U_a &= L_{a'} \cap U_{a''} \setminus \bigcup_{a\in A, a\neq a''} U_a \\ &= \emptyset \end{aligned}$$

Thus,

$$\begin{aligned} \mu'(X_1) =& \mu'\Big(\bigcup_{S_1\in I_1, S_2\in I_2} \bigcap_{a\in S_1} L_a \cap \bigcap_{a\in S_2} U_a \setminus (\bigcup_{a\in A\setminus S_1} L_a \cup \bigcup_{a\in A\setminus S_2} U_a) \cap) \\ =& \mu'\Big(\bigcup_{\{a'\}\in I_1, S_2\in I_2} (L_{a'} \cap \bigcap_{a\in S_2} U_a \setminus \bigcup_{a\in A\setminus S_2} U_a) \Big) \\ =& \mu'\Big(\bigcup_{\{a'\}\in I_1\cap I_2} L_{a'} \cap U_{a'} \setminus \bigcup_{a\in A, a\neq a'} U_a \Big) \\ =& \mu'\Big(\bigcup_{\{a'\}\in I_1\cap I_2} (L_{a'} \setminus \bigcup_{a\in A, a\neq a'} U_a) \Big) \\ =& \mu'\Big(\bigcup_{\{a'\}\in I_1\cap I_2} (L_{a'} \setminus \bigcup_{a\in A, a\neq a'} (L_a \cup (U_a \setminus L_a))) \Big) \\ =& \mu'\Big(\bigcup_{\{a'\}\in I_1\cap I_2} (L_{a'} \setminus \bigcup_{a\in A, a\neq a'} (L_a)) \Big) \\ =& \mu'\Big(\bigcup_{\{a'\}\in I_1\cap I_2} L_{a'} \Big) \end{aligned}$$

Denote $\bigcup_{\{a'\}\in I_1\cap I_2} L_{a'}$ as X_1'. And $X_1 \setminus X_1'$ and $X_1' \setminus X_1$ both have measure 0.

Similar results hold for X_2 as well, and we can show that

$$\mu'(X_2) = \mu'\Big(\bigcup_{\{b'\}\in I_3\cap I_4} L_{b'} \Big)$$

Denote $\bigcup_{\{b'\}\in I_3\cap I_4} L_{b'}$ as X_2'. And $X_2 \setminus X_2'$ and $X_2' \setminus X_2$ both have measure 0.

Thus,

$$\begin{aligned}
\mu'(X_1 \cap X_2) =& \mu'(X_1 \cap X_2 \cap X_1') + \mu'((X_1 \cap X_2) \setminus X_1') \\
=& \mu'(X_1 \cap X_2 \cap X_1') + 0 \\
=& \mu'(X_1 \cap X_2 \cap X_1' \cap X_2') + \mu'((X_1 \cap X_2 \cap X_1') \setminus X_2') + 0 \\
=& \mu'(X_1 \cap X_2 \cap X_1' \cap X_2') + 0 + 0 \\
=& \mu'(X_1 \cap X_2 \cap X_1' \cap X_2') + \mu'((X_2 \cap X_1' \cap X_2') \setminus X_1) \\
=& \mu'(X_2 \cap X_1' \cap X_2') \\
=& \mu'(X_2 \cap X_1' \cap X_2') + \mu'((X_1' \cap X_2') \setminus X_2) \\
=& \mu'(X_1' \cap X_2') \\
=& \mu'\Big((\bigcup_{\{a'\} \in I_1 \cap I_2} L_{a'}) \cap (\bigcup_{\{b'\} \in I_3 \cap I_4} L_{b'})\Big) \\
=& \mu'\Big(\bigcup_{\{a'\} \in I_1 \cap I_2, \{b'\} \in I_3 \cap I_4} L_{a',b'}\Big) \\
=& \sum_{\substack{\{a'\} \in I_1 \cap I_2 \\ \{b'\} \in I_3 \cap I_4}} \mu'(L_{a',b'})
\end{aligned}$$

Next we show that $\mu'(i)(L_{a,b}) = \mu'(i)(X_1) \cdot \mu'(i)(X_2)$. Note that $\mu'(L_a) = \sum_b \mu'(L_{a,b}) = \mu'(E_1^{-1}(a))$, and $\mu'(L_b) = \sum_a \mu'(L_{a,b}) = \mu'(E_2^{-1}(b))$. And $\mu_1 \otimes \mu_2 = \mu'(i) \circ (E_1, E_2)^{-1}$ implies that

$$\begin{aligned}
\mu'(i)(L_{a,b}) &= \mu_1 \otimes \mu_2(a, b) \\
&= \mu_1(a) \cdot \mu_2(b)
\end{aligned}$$

Then

$$\begin{aligned}\mu_1(a) &= \mu_1(a) \cdot \sum_{b \in A} \mu_2(b) \\ &= \sum_{b \in A} \mu_1(a) \cdot \mu_2(b) \\ &= \sum_{b \in A} \mu'(i)(L_{a,b}) \\ &= \mu'(i)\left(\sum_{b \in A} L_{a,b}\right) \\ &= \mu'(i)(L_a),\end{aligned}$$

and similarly,

$$\begin{aligned}\mu_2(b) &= \left(\sum_{a \in A} \mu_1(a)\right) \cdot \mu_2(b) \\ &= \sum_{a \in A} (\mu_1(a) \cdot \mu_2(b)) \\ &= \sum_{a \in A} \mu'(i)(L_{a,b}) \\ &= \mu'(i)\left(\sum_{a \in A} L_{a,b}\right) \\ &= \mu'(i)(L_b).\end{aligned}$$

Thus,

$$\mu'(i)(L_{a,b}) = \mu_1(a) \cdot \mu_2(b) = \mu'(i)(L_a) \cdot \mu'(i)(L_b)$$

Therefore,

$$\begin{aligned}
\mu'(X_1 \cap X_2) &= \sum_{\substack{\{a'\}\in I_1\cap I_2\\ \{b'\}\in I_3\cap I_4}} \mu'(L_{a',b'}) \\
&= \sum_{\substack{\{a'\}\in I_1\cap I_2\\ \{b'\}\in I_3\cap I_4}} \mu'(L_{a'}) \cdot \mu'(L_{b'}) \\
&= \sum_{\{a'\}\in I_1\cap I_2} \mu'(L_{a'}) \cdot \sum_{\{b'\}\in I_3\cap I_4} \mu'(L_{b'}) \\
&= \mu'(X_1) \cdot \mu'(X_2) \\
&= \mu'_1(X_1) \cdot \mu'_2(X_2)
\end{aligned}$$

Thus we have $(\mathcal{F}_1, {\mu'}_1) \circledast (\mathcal{F}_2, {\mu'}_2) \sqsubseteq (\mathcal{F}', \mu')$. Let $p_1 = p_2 = \lambda x.\, p'(x)/2$.

Next we show that $E_1 \Subset \mu_1(\mathcal{F}_1, {\mu'}_1, p_1)$ and $E_2 \Subset \mu_2(\mathcal{F}_2, {\mu'}_2, p_2)$. By definition, $E_1 \Subset \mu_1(\mathcal{F}_1, {\mu'}_1, p_1)$ is equivalent to

$$\exists \mathcal{F}'', \mu''.\, (\mathsf{Own}(\mathcal{F}'', \mu''))(\mathcal{F}_1, {\mu'}_1, p_1) * E_1 \prec (\mathcal{F}''(i), \mu''(i)) \wedge \mu_1 = \mu''(i) \circ E_1^{-1},$$

which is equivalent to

$$\begin{aligned}
&\exists \mathcal{F}'', \mu''.\, (\mathcal{F}'', \mu'') \preceq (\mathcal{F}_1, {\mu'}_1) * \big(\forall a \in A. \exists S_a, T_a \in \mathcal{F}''(i).\\
&\quad S_a \subseteq E_1^{-1}(a) \subseteq T_a \wedge \mu''(i)(S_a) = \mu''(i)(S_a) \wedge \mu_1(a) = \mu''(i)(S_a) = \mu''(i)(T_a)\big)
\end{aligned}$$

We can pick the existential witness to be $\mathcal{F}_1, {\mu'}_1$. For any $a \in A$, $E_1^{-1}(a) = \bigcup_{b\in A}(E_1, E_2)^{-1}(a, b)$. Because we have $L_{a,b} \subseteq (E_1, E_2)^{-1}(a, b) \subseteq U_{a,b}$, then

$$\textstyle\bigcup_{b\in A} L_{a,b} \subseteq E_1^{-1}(a) = \bigcup_{b\in A}(E_1, E_2)^{-1}(a, b) \subseteq \bigcup_{b\in A} U_{a,b}.$$

By definition, for each a, $\bigcup_{b\in A} L_{a,b} \in \mathcal{F}_1(i)$ and $\bigcup_{b\in A} U_{a,b} \in \mathcal{F}_1(i)$, and we also

have

$$\begin{aligned}
\mu'_1(i)(\textstyle\bigcup_{b\in A} L_{a,b}) &= \sum_{b\in A} \mu'_1(i)(L_{a,b}) \\
&= \sum_{b\in A} \mu'_1(i)(U_{a,b}) \\
&= \mu'_1(i)\big(\textstyle\bigcup_{b\in A} U_{a,b}\big) \\
&= \mu_1(a)
\end{aligned}$$

Thus, $S_a = \bigcup_{b\in A} L_{a,b}$ and $T_a = \bigcup_{b\in A} U_{a,b}$ witnesses the conditions needed for $E_1 \lesssim \mu_1(\mathcal{F}_1, \mu'_1, p_1)$. And similarly, we have $E_2 \lesssim \mu_2(\mathcal{F}_2, \mu'_2, p_2)$. □

Soundness of Conditioning Rules

Lemma D.5.4. C-TRUE *is sound.*

Proof. Let $\varepsilon = (\mathcal{F}_\varepsilon, \mu_\varepsilon, p_\varepsilon) \in \mathcal{M}_I$ be the unit of $\mathcal{M}_I$ and $\kappa = \lambda v.\, \mu_\varepsilon$. Then,

$$\begin{aligned}
&\mathsf{True} \vdash \mathsf{Own}(\mathcal{F}_\varepsilon, \mu_\varepsilon) \\
&\quad\vdash \mathsf{Own}(\mathcal{F}_\varepsilon, \mu_\varepsilon) * \ulcorner \forall i \in I.\, \mu_\varepsilon(i) = \mathsf{bind}(\mu, \kappa(i)) \urcorner \\
&\quad\vdash \mathsf{Own}(\mathcal{F}_\varepsilon, \mu_\varepsilon) * \ulcorner \forall i \in I.\, \mu_\varepsilon(i) = \mathsf{bind}(\mu, \kappa(i)) \urcorner * \mathsf{True} \\
&\quad\vdash \exists \mathcal{F}_\varepsilon, \mu_\varepsilon, \kappa.\, \mathsf{Own}(\mathcal{F}_\varepsilon, \mu_\varepsilon) * \ulcorner \forall i \in I.\, \mu_\varepsilon(i) = \mathsf{bind}(\mu, \kappa(i)) \urcorner \\
&\qquad\qquad * (\forall v \in \mathrm{supp}(\mu).\mathsf{Own}(\mathcal{F}_\varepsilon, \kappa(I)(v), p_\varepsilon) \mathrel{-\!\!*} \mathsf{True}) \\
&\quad\vdash \boldsymbol{C}_\mu _.\mathsf{True}
\end{aligned}$$

□

Lemma D.5.5. C-FALSE *is sound.*

Proof. Assume $a \in \mathcal{M}_I$ is such that $\mathbf{V}(a)$ and that it satisfies $\boldsymbol{C}_\mu\, v.\, \mathsf{False}$. By

definition, this means that, for some $\mathcal{F}_0, \mu_0, p_0$, and κ_0:

$$(\mathcal{F}_0, \mu_0, p_0) \preceq a \tag{D.4}$$

$$\forall i \in I.\, \mu_0(i) = \mathsf{bind}(\mu, \kappa_0(i)) \tag{D.5}$$

$$\forall v \in \mathrm{supp}(\mu).\, \mathsf{False}(\mathcal{F}_0, \kappa_0(I)(v), p_0) \tag{D.6}$$

Let $v_0 \in \mathrm{supp}(\mu)$—we know one exists because μ is a (discrete) probability distribution. Then by (D.6) on v_0 we get $\mathsf{False}(\mathcal{F}_0, \kappa_0(I)(v_0), p_0)$ holds. Since $\mathsf{False}(\_)$ is by definition false, we get $\mathsf{False}(a)$ holds *ex falso*. □

Lemma D.5.6. C-CONS *is sound.*

Proof. Assume $a \in \mathcal{M}_I$ is such that $\mathbf{V}(a)$ and that it satisfies $\boldsymbol{C}_\mu\, v.\, K(v)$. By definition, this means that, for some $\mathcal{F}_0, \mu_0, p_0$, and κ_0:

$$(\mathcal{F}_0, \mu_0, p_0) \preceq a \tag{D.7}$$

$$\forall i \in I.\, \mu_0(i) = \mathsf{bind}(\mu, \kappa_0(i)) \tag{D.8}$$

$$\forall v \in \mathrm{supp}(\mu).\, K(v)(\mathcal{F}_0, \kappa_0(I)(v), p_0) \tag{D.9}$$

Then by the premise $\forall v.\, K(v) \vdash K'(v)$ and (D.9) we obtain

$$\forall v \in \mathrm{supp}(\mu).\, K'(v)(\mathcal{F}_0, \kappa_0(I)(v), p_0) \tag{D.10}$$

By (D.7), (D.8), and (D.10) we get $\boldsymbol{C}_\mu\, v.\, K'(v)$ as desired. □

Lemma D.5.7. C-FRAME *is sound.*

Proof. Assume $a \in \mathcal{M}_I$ is such that $\mathbf{V}(a)$ and that it satisfies $P * \boldsymbol{C}_\mu\, v.\, K(v)$. By definition, this means that there exist some $(\mathcal{F}_1, \mu_1, p_1)$, $(\mathcal{F}_2, \mu_2, p_2)$, and κ such

that

$$(\mathcal{F}_1, \mu_1, p_1) \cdot (\mathcal{F}_2, \mu_2, p_2) \preceq a \tag{D.11}$$

$$P(\mathcal{F}_1, \mu_1, p_1) \tag{D.12}$$

$$\forall i \in I. \mu_2(i) = \mathsf{bind}(\mu, \kappa(i)) \tag{D.13}$$

$$\forall v \in \mathrm{supp}(\mu). K(v)(\mathcal{F}_2, \kappa(I)(v), p_2) \tag{D.14}$$

Now let:

$$(\mathcal{F}', \mu', p') = (\mathcal{F}_1(i), \mu_1(i)) \circledast (\mathcal{F}_2(i), \mu_2(i)) \qquad \kappa'(i) = \lambda v. \mu_1(i) \circledast \kappa(i)(v)$$

By lemma D.2.7, for each $i \in I$:

$$\begin{aligned}(\mathcal{F}', \mu', p') &= (\mathcal{F}_1(i), \mu_1(i)) \circledast (\mathcal{F}_2(i), \mu_2(i)) \\ &= (\mathcal{F}_1(i) \oplus \mathcal{F}_2(i), \mathsf{bind}(\mu, \lambda v. \mu_1(i) \circledast \kappa(i)(v))) \qquad \text{(By lemma D.2.7)} \\ &= (\mathcal{F}_1(i) \oplus \mathcal{F}_2(i), \mathsf{bind}(\mu, \kappa'(i)))\end{aligned}$$

Notice that $\kappa'(I)(v) = \mu_1 \circledast \kappa(I)(v)$. Thus we obtain:

$$(\mathcal{F}', \mu', p') \preceq a \tag{D.15}$$

$$\forall i \in I. \mu'(i) = \mathsf{bind}(\mu, \kappa'(i)) \tag{D.16}$$

and for all $v \in \mathrm{supp}(\mu)$,

$$(\mathcal{F}_1, \mu_1, p_1) \circledast (\mathcal{F}_2, \kappa(I)(v), p_2) = (\mathcal{F}', \mu_1 \circledast \kappa(I)(v), p') \preceq (\mathcal{F}', \kappa'(I)(v), p') \tag{D.17}$$

$$P(\mathcal{F}_1, \mu_1, p_1) \tag{D.18}$$

$$K(v)(\mathcal{F}_2, \kappa(I)(v), p_2) \tag{D.19}$$

which gives us that a satisfies $\boldsymbol{C}_\mu\, v. (P * K(v))$ as desired. □

Lemma D.5.8. C-UNIT-L *is sound.*

Proof. Straightforward. □

Lemma D.5.9. C-UNIT-R *is sound.*

Proof. We prove the two directions separately.

Forward direction $E \twoheadrightarrow \mu \vdash \boldsymbol{C}_\mu\, v.\, \lceil E = v \rceil$ By unfolding the assumption $E \twoheadrightarrow \mu$ we get that there exist $\mathcal{F}, \mu$ such that:

$$\mathsf{Own}(\mathcal{F}, \mu) * \ulcorner E \prec (\mathcal{F}(i), \mu(i)) \urcorner * \ulcorner \mu = \mu(i) \circ E^{-1} \urcorner$$

holds. Let

$$\kappa \triangleq \lambda j. \begin{cases} \lambda v.\, \mu(j) & \text{if } j \neq i \\ \lambda v.\, \gamma_v & \text{if } j = i \end{cases} \qquad \gamma_v \triangleq \lambda X : \mathcal{F}(i).\, \frac{\mu(i)(X \cap (E = v)^{-1})}{\mu(i)((E = v)^{-1})}$$

That is, $\kappa(j)$ maps every v to $\mu(j)$ when $i \neq j$, while when $i = j$ it maps v to the distribution $\mu(i)$ conditioned on $E = v$. Note that κ is well defined because

1. although the events $X \cap (E = v)^{-1}$ and $(E = v)^{-1}$ might not belong to $\mathcal{F}(i)$, their probability is uniquely determined by almost measurability of E;
2. we are only interested in the cases where $v \in \mathrm{supp}(\mu)$, which implies that the denominator is not zero: $\mu(i)((E = v)^{-1}) = \mu(v) > 0$.

By construction we obtain that

$$\forall j \in I.\, \mu(j) = \mathsf{bind}(\mu, \kappa(j)) \tag{D.20}$$

$$\forall v \in \mathrm{supp}(\mu).\, \kappa(i)(v)((E = v)^{-1}) = 1 \tag{D.21}$$

From (D.21) we get that $\lceil E = v \rceil$ holds on $(\mathcal{F}(i), \kappa(i)(v), p(i))$, from which it follows that:

$$\mathsf{Own}(\mathcal{F}, \kappa(I)(v), p) \twoheadrightarrow \lceil E = v \rceil$$

Therefore we obtain

$$\exists \mathcal{F}, \mu, \kappa, p.\, \mathsf{Own}(\mathcal{F}, \mu, p) * \ulcorner \forall j \in I.\mu(j) = \mathsf{bind}(\mu, \kappa(j)) \urcorner$$
$$* (\forall v \in A_\mu.\mathsf{Own}(\mathcal{F}, \kappa(I)(v), p) \mathrel{-\!\!*} \lceil E = v \rceil)$$

which gives us $\boldsymbol{C}_\mu\, v.\, \lceil E = v \rceil$ by proposition D.4.1.

Backward direction $\boldsymbol{C}_\mu\, v.\, \lceil E = v \rceil \vdash E \twoheadrightarrow \mu$ First note that

$$\begin{aligned}
&\lceil E = v \rceil(\mathcal{F}, \kappa(v), p) \\
&\quad \Leftrightarrow \big(((E = v) \in \mathsf{true}) \twoheadrightarrow \delta_{\mathsf{True}}\big)(\mathcal{F}, \kappa(I)(v), p) \\
&\quad \Leftrightarrow ((E = v) \in \mathsf{true}) \prec (\mathcal{F}(i), \kappa(i)(v)) \wedge \delta_{\mathsf{True}} = \kappa(i)(v) \circ ((E = v) \in \mathsf{true})^{-1} \\
&\quad \Leftrightarrow ((E = v) \in \mathsf{true}) \prec (\mathcal{F}(i), \kappa(i)(v)) \wedge \delta_v = \kappa(i)(v) \circ E^{-1}
\end{aligned}$$

for some κ. This implies $\ulcorner E \prec \mathcal{F}(i), \kappa(i)(v) \urcorner$. Then, for any value $v \in \mathrm{supp}(\mu)$,

$$\begin{aligned}
\mu(i) \circ E^{-1}(v) &= (\mathsf{bind}(\mu, \kappa(i)) \circ E^{-1})(v) \\
&= \mathsf{bind}(\mu, \kappa(i))(E^{-1}(v)) \\
&= \sum_{v' \in \mathrm{supp}(\mu)} \mu(v') \cdot \kappa(i)(v')(E^{-1}(v)) \\
&= \sum_{v' \in \mathrm{supp}(\mu)} \mu(v') \cdot (\kappa(i)(v') \circ E^{-1})(v) \\
&= \sum_{v' \in \mathrm{supp}(\mu)} \mu(v') \cdot \delta_{v'}(v) \\
&= \mu(v)
\end{aligned}$$

This implies the pure facts that $E \prec (\mathcal{F}(i), \mu(i))$ and $\mu = \mu(i) \circ E^{-1}$. There-

fore:

$$\begin{aligned}
\boldsymbol{C}_\mu\, v.\, \lceil E = v \rceil \vdash\ & \exists \mathcal{F}, \mu, \kappa, p.\, \mathsf{Own}(\mathcal{F}, \mu, p) * \ulcorner \forall j \in I.\mu(j) = \mathsf{bind}(\mu, \kappa(j)) \urcorner \\
& * (\forall v \in A_\mu.\mathsf{Own}(\mathcal{F}, \kappa(I)(v), p) \mathrel{-\!\!*} \lceil E = v \rceil) \\
\vdash\ & \exists \mathcal{F}, \mu.\, \mathsf{Own}(\mathcal{F}, \mu) * \ulcorner E \prec (\mathcal{F}(i), \mu(i)) \urcorner * \ulcorner \mu = \mu(i) \circ E^{-1} \urcorner \\
\vdash\ & E \Subset \mu
\end{aligned}$$

□

Lemma D.5.10. C-ASSOC *is sound.*

Proof. Define $\kappa' = \lambda v.\, \mathsf{bind}(\kappa(v), \lambda w.\, \mathbf{return}(v, w))$. We start by rewriting the assumption $\boldsymbol{C}_\mu\, v.\, \boldsymbol{C}_{\kappa(v)}\, w.\, K(v, w)$ so that k' is used and K depends only on the binding of the innermost modality:

$$\begin{aligned}
\boldsymbol{C}_\mu\, v.\, \boldsymbol{C}_{\kappa(v)}\, w.\, K(v, w) \vdash\ & \boldsymbol{C}_\mu\, v.\, \boldsymbol{C}_{\kappa'(v)}(v', w).\, K(v, w) && \text{(C-TRANSF, C-CONS)} \\
\vdash\ & \boldsymbol{C}_\mu\, v.\, \boldsymbol{C}_{\kappa'(v)}(v', w).\, K(v', w) && \text{(C-PURE, C-CONS)}
\end{aligned}$$

C-TRANSF is applied to the innermost modality by using the bijection $f_v(w) = (v, w)$. Then, since $(v', w) \in \mathrm{supp}(k'(v)) \Rightarrow v = v'$, we can replace v' for v in K.

Our goal is now to prove:

$$\boldsymbol{C}_\mu\, v.\, \boldsymbol{C}_{\kappa'(v)}(v', w).\, K(v', w) \vdash \boldsymbol{C}_{\mathsf{bind}(\mu, \kappa')}(v', w).\, K(v', w)$$

Let $a \in \mathcal{M}_I$ be such that $\mathbf{V}(a)$ and that it satisfies $\boldsymbol{C}_\mu\, v.\, \boldsymbol{C}_{\kappa'(v)}(v', w).\, K(v', w)$. From this assumption we know that, for some $\mathcal{F}_0, \mu_0, p_0$, and κ_0:

$$(\mathcal{F}_0, \mu_0, p_0) \preceq a \tag{D.22}$$

$$\forall i \in I.\, \mu_0(i) = \mathsf{bind}(\mu, \kappa_0(i)) \tag{D.23}$$

such that $\forall v \in \mathrm{supp}(\mu)$, there are some $\mathcal{F}_1^v, \mu_1^v, p_1^v$, and κ_1^v satisfying:

$$(\mathcal{F}_1^v, \mu_1^v, p_1^v) \preceq (\mathcal{F}_0, \kappa_0(I)(v), p_0) \tag{D.24}$$

$$\forall i \in I.\, \mu_1^v(i) = \mathsf{bind}(\kappa'(v), \kappa_1^v(i)) \tag{D.25}$$

$$\forall (v', w) \in \mathrm{supp}(\kappa'(v)).\, K(v', w)(\mathcal{F}_1^v, \kappa_1^v(I)(v', w), p_1^v) \tag{D.26}$$

Our goal is to prove $\boldsymbol{C}_{\mathsf{bind}(\mu,\kappa')}(v', w).\, K(v', w)$ holds on a. To this end, we want to show that there exists κ_2' such that:

$$\forall i \in I.\, \mu_0(i) = \mathsf{bind}(\mathsf{bind}(\mu, \kappa'), \kappa_2'(i)) \tag{D.27}$$

$$\forall (v', w) \in \mathrm{supp}(\mathsf{bind}(\mu, \kappa')).\, K(v', w)(\mathcal{F}_0, \kappa_2'(I)(v'), p_0) \tag{D.28}$$

Now let

$$\kappa_2(i) = \lambda(v', w).\, \kappa_1^{v'}(i)(v', w).$$

which by construction and eq. (D.25) gives us

$$\mu_1^v(i) = \mathsf{bind}(\kappa'(v), \kappa_1^v(i)) = \mathsf{bind}(\kappa'(v), \kappa_2(i))$$

Therefore, by eq. (D.24), we can apply lemma D.2.4 and obtain that there exists a κ_2' such that

$$\kappa_0(i)(v) = \mathsf{bind}(\kappa'(v), \kappa_2'(i)) \tag{D.29}$$

$$\left(\mathcal{F}_0, \kappa_2'(i)(v', w)\right) \sqsupseteq \left(\mathcal{F}_1^{v'}, \kappa_2(i)(v', w)\right) = \left(\mathcal{F}_1^{v'}, \kappa_1^{v'}(i)(v', w)\right) \tag{D.30}$$

By eqs. (D.23) and (D.29) we have:

$$\begin{aligned}
\mu_0(i) &= \mathsf{bind}(\mu, \kappa_0(i)) \\
&= \mathsf{bind}(\mu, \lambda v.\, \mathsf{bind}(\kappa'(v), \kappa_2'(i))) && \text{By associativity of bind} \\
&= \mathsf{bind}(\mathsf{bind}(\mu, \kappa'), \kappa_2'(i))
\end{aligned}$$

which proves eq. (D.27).

Finally, to prove eq. (D.28), we can observe that $(v', w) \in \mathrm{supp}(\mathsf{bind}(\mu, \kappa'))$ implies $v' \in \mathrm{supp}(\mu)$; therefore, by (D.26), upward closure of $K(v', w)$, and (D.30) and (D.24), we can conclude $K(v', w)$ holds on $(\mathcal{F}_0, \kappa_2'(I)(v'), p_0)$, as desired. □

Lemma D.5.11. C-UNASSOC *is sound.*

Proof. Assume $a \in \mathcal{M}_I$ is such that $\mathbf{V}(a)$ and that it satisfies $\boldsymbol{C}_{\mathsf{bind}(\mu,\kappa)}\, w.\, K(w)$. By definition, this means that, for some $\mathcal{F}_0, \mu_0, p_0$, and κ_0:

$$(\mathcal{F}_0, \mu_0, p_0) \preceq a \tag{D.31}$$

$$\forall i \in I.\, \mu_0(i) = \mathsf{bind}(\mathsf{bind}(\mu, \kappa), \kappa_0(i)) \tag{D.32}$$

$$\forall w \in \mathrm{supp}(\mathsf{bind}(\mu, \kappa)).\, K(w)(\mathcal{F}_0, \kappa_0(I)(w), p_0) \tag{D.33}$$

Our goal is to show that a satisfies $\boldsymbol{C}_{\mu}\, v.\, \boldsymbol{C}_{\kappa(v)}\, w.\, K(w)$, for which it would suffice to show that there is a κ_1 such that:

$$\forall i \in I.\, \mu_0(i) = \mathsf{bind}(\mu, \kappa_1(i)) \tag{D.34}$$

and for all $v \in \mathrm{supp}(\mu)$ there is a κ_2^v with

$$\forall i \in I.\, \kappa_1(i)(v) = \mathsf{bind}(\kappa(v), \kappa_2^v(i)) \tag{D.35}$$

$$\forall w \in \mathrm{supp}(\kappa(v)).\, K(w)(\mathcal{F}_0, \kappa_2^v(I)(w), p_0) \tag{D.36}$$

To prove this we let

$$\kappa_1(i) = \lambda v.\, \mathsf{bind}(\kappa(v), \kappa_0(i)) \qquad\qquad \kappa_2^v(i) = \kappa_0(i)$$

By the associativity of bind we have

$$\mu_0(i) = \mathsf{bind}(\mathsf{bind}(\mu, \kappa), \kappa_0(i)) = \mathsf{bind}(\mu, \lambda v.\, \mathsf{bind}(\kappa(v), \kappa_0(i))) = \mathsf{bind}(\mu, \kappa_1(i))$$

which proves (D.34). By construction,

$$\kappa_1(i)(v) = \mathsf{bind}(\kappa(v), \kappa_0(i)) = \mathsf{bind}(\kappa(v), \kappa_2^v(i))$$

proving (D.35). Finally, $v \in \mathrm{supp}(\mu)$ and $w \in \mathrm{supp}(\kappa(v))$ imply $w \in \mathrm{supp}(\mathsf{bind}(\mu, \kappa))$, so by (D.33) we proved (D.36), concluding the proof. □

Lemma D.5.12. C-SKOLEM *is sound.*

Proof. For any resource $r = (\mathcal{F}, \mu, p)$,

$$\begin{aligned}&\big(\boldsymbol{C}_\mu\, v.\, \exists x : \mathbf{Var}.\, Q(v,x)\big)(\mathcal{F}, \mu, p)\\ \Leftrightarrow\ &\exists \kappa.\, \forall i \in I.\mu(i) = \mathsf{bind}(\mu, \kappa(i)) \wedge \forall v \in \mathrm{supp}(\mu).\, (\exists x : X.\, Q(v,x))(\mathcal{F}, \kappa(I)(v), p)\end{aligned}$$

For all $v \in \mathrm{supp}(\mu)$, $\exists x : X.\, Q(v,x)$ holds on $(\mathcal{F}, \kappa(I)(v), p)$. Thus, $Q(v, x_v)(\mathcal{F}, \kappa(I)(v), p)$ holds for some x_v. Then define $f : A \to \mathbf{Var}$ by letting $f(v) = x_v$ for $v \in \mathrm{supp}(\mu)$. Then,

$$\exists \kappa.\, \forall i \in I.\mu(i) = \mathsf{bind}(\mu, \kappa(i)) \wedge \forall v \in \mathrm{supp}(\mu).\, Q(v, f(v))(\mathcal{F}, \kappa(I)(v), p)$$

And therefore $\mathcal{F}, \mu, p$ satisfies $\exists f : A \to \mathbf{Var}.\, \boldsymbol{C}_\mu\, v.\, Q(v,x)$. □

Lemma D.5.13. C-TRANSF *is sound.*

Proof. For any resource $a = (\mathcal{F}, \mu, p)$, if $\big(\boldsymbol{C}_\mu\, v.\, K(v)\big)((\mathcal{F}, \mu, p))$, then

$$\begin{aligned}&\exists \kappa.\, (\mathcal{F}, \mu, p) \preceq a \wedge \forall i \in I.\, \mu(i) = \mathsf{bind}(\mu, \kappa(i))\\ &\wedge \forall v \in \mathrm{supp}(\mu).(K(v))((\mathcal{F}, \kappa(I)(v), p))\end{aligned}$$

$\mu = \mathsf{bind}(\mu, \kappa)$ says that for any $E \in \mathcal{F}$,

$$\begin{aligned}
\mu(E) &= \sum_{v \in \mathrm{supp}(\mu)} \mu(v) \cdot \kappa(I)(v)(E) \\
&= \sum_{v \mid f(v) \in \mathrm{supp}(\mu)} \mu(f(v)) \cdot \kappa(I)(f(v))(E) && \text{(Because } f \text{ is bijective)} \\
&= \sum_{v \in \mathrm{supp}(\mu')} \mu'(v) \cdot \kappa(I)(f(v))(E) && \text{(Because } \mu'(v) = \mu(f(v))\text{)} \\
&= \mathsf{bind}(\mu', \lambda v.\, \kappa(I)(f(v)))(E)
\end{aligned}$$

Thus, $\mu = \mathsf{bind}(\mu', \lambda v.\, \kappa(I)(f(v)))$. Furthermore, $(K(f(v)))((\mathcal{F}, \kappa(I)(f(v)), p))$.

Thus, if we denote $\lambda v.\, \kappa(I)(f(v))$ as κ', it satisfies

$$\begin{aligned}
&(\mathcal{F}, \mu, p) \preceq a \wedge \forall i \in I.\, \mu(i) = \mathsf{bind}(\mu', \kappa'(i)) \\
&\wedge \forall v \in \mathrm{supp}(\mu).(K(v))((\mathcal{F}, \kappa'(I)(v), p))
\end{aligned}$$

Thus, $\left(\boldsymbol{C}'_\mu\, v.\, K(f(v))\right)((\mathcal{F}, \mu, p))$. □

Lemma D.5.14. SURE-STR-CONVEX *is sound.*

Proof. Assume $a \in \mathcal{M}_I$ is a valid resource that satisfies $\boldsymbol{C}_\mu\, v.(K(v) * \lceil E \rceil)$. Then, by definition, we know that, for some $(\mathcal{F}_0, \mu_0, p_0)$ and κ_0:

$$(\mathcal{F}_0, \mu_0, p_0) \preceq a \tag{D.37}$$

$$\forall i \in I.\, \mu_0(i) = \mathsf{bind}(\mu, \kappa_0(i)) \tag{D.38}$$

and, for all $v \in \mathrm{supp}(\mu)$, there are $(\mathcal{F}_1^v, \mu_1^v, p_1^v)$, $(\mathcal{F}_2^v, \mu_2^v, p_2^v)$ such that

$$(\mathcal{F}_1^v, \mu_1^v, p_1^v) \cdot (\mathcal{F}_2^v, \mu_2^v, p_2^v) \preceq (\mathcal{F}_0, \kappa_0(I)(v), p_0) \tag{D.39}$$

$$K(v)(\mathcal{F}_1^v, \mu_1^v, p_1^v) \tag{D.40}$$

$$\lceil E \rceil(\mathcal{F}_2^v, \mu_2^v, p_2^v) \tag{D.41}$$

From (D.41) we know that for all $v \in \mathrm{supp}(\mu)$ there are $L_1^v, L_0^v, U_1^v, U_0^v \in \mathcal{F}_2^v(i)$ such that:

$$L_0^v \subseteq E^{-1}(\mathsf{False}) \subseteq U_0^v \qquad \mu_2^v(L_0^v) = \mu_2^v(U_0^v) = 0$$
$$L_1^v \subseteq E^{-1}(\mathsf{True}) \subseteq U_1^v \qquad \mu_2^v(L_1^v) = \mu_2^v(U_1^v) = 1$$

Without loss of generality, all $L_0^v, L_1^v, U_0^v, U_1^v$ can be assumed to be only non-trivial on $\mathrm{FV}(E)$. Consequently, we can also assume that $p_2^v(x) < 1$ for every x, and in addition $p_2^v(x) > 0$ if and only if $x \in \mathrm{FV}\, E$ and $j = i$. From these components we can construct a new resource:

$$\mathcal{F}_3(j) \triangleq \begin{cases} \sigma\left(\{\bigcap_{v\in\mathrm{supp}(\mu)} L_1^v, \bigcup_{v\in\mathrm{supp}(\mu)} U_1^v\}\right) & \text{if } j = i \\ \{\mathbf{Mem}[\mathbf{Var}], \emptyset\} & \text{if } j \neq i \end{cases}$$

$$\mu_3 \triangleq \mu_0|_{\mathcal{F}_3}$$

$$p_3 \triangleq \lambda x. \begin{cases} \min\left\{p_2^v(x) | v \in \mathrm{supp}(\mu)\right\} & \text{if } j = i \wedge x \in \mathrm{FV}(E) \\ 0 & \text{otherwise} \end{cases}$$

By construction we obtain that $\forall j \in I. \mathcal{F}_3(j) \subseteq \mathcal{F}_0(j)$, and that $\mathbf{V}(\mathcal{F}_3, \mu_3, p_3)$. Now letting $p_1' = p_0 - p_3$, we obtain a valid resource $(\mathcal{F}_0, \mu_0, p_1')$.

Moreover, we have $\mathcal{F}_0 = \mathcal{F}_0 \oplus \mathcal{F}_3$ and $\forall j \in I. \forall X \in \mathcal{F}_3(j). \mu_3(X) \in \{0, 1\}$, which means that for any $X \in \mathcal{F}_3$ and $Y \in \mathcal{F}_0$, $\mu_3(X) \cdot \mu_0(Y) = \mu_0(X \cap Y)$. Then, by (D.38):

$$(\mathcal{F}_0, \mathsf{bind}(\mu, \kappa_0), p_1') \circledast (\mathcal{F}_3, \mu_3, p_3) \preceq (\mathcal{F}_0, \mu_0, p_0) = a$$

To close the proof it would then suffice to show that $\boldsymbol{C}_\mu\, v.K(v)$ holds on $(\mathcal{F}_0, \mathsf{bind}(\mu, \kappa_0), p_1')$ and that $\lceil E \rceil$ holds on $(\mathcal{F}_3(j), \mu_3, p_3)$. The latter is obvious. The former follows from the fact that $\kappa_0(j)(v)|_{\mathcal{F}_1^v} = \mu_1^v(j)$; by upward-closure and (D.40) this means that, for all $v \in \mathrm{supp}(\mu)$:

$$K(v)(\mathcal{F}_1^v, \mu_1^v, p_1^v) \Rightarrow K(v)(\mathcal{F}_0, \kappa_0(I)(v), p_1')$$

which proves our claim. □

Lemma D.5.15. C-FOR-ALL *is sound.*

Proof. By unfolding the definitions,

$$
\begin{aligned}
& \boldsymbol{C}_\mu\, v.\forall x : X.Q(v) \\
\Leftrightarrow\ & \exists \mathcal{F}, \mu_0, \kappa.\, \mathsf{Own}((\mathcal{F}, \mu_0)) * \ulcorner \forall i \in I.\, \mu_0(i) = \mathsf{bind}(\mu, \kappa(i)) \urcorner \\
& \qquad * (\forall a \in A_\mu.\mathsf{Own}((\mathcal{F}, [i{:}\, \kappa(i)(a) \mid i \in I])) \mathrel{-\!\!*} \forall x : X.Q(v)) \\
\Rightarrow\ & \forall x : X.\exists \mathcal{F}, \mu_0, \kappa.\, \mathsf{Own}((\mathcal{F}, \mu_0)) * \ulcorner \forall i \in I.\, \mu_0(i) = \mathsf{bind}(\mu, \kappa(i)) \urcorner \\
& \qquad * (\forall a \in A_\mu.\mathsf{Own}((\mathcal{F}, [i{:}\, \kappa(i)(a) \mid i \in I])) \mathrel{-\!\!*} Q(v)) \\
\Leftrightarrow & \forall x : X.\, \boldsymbol{C}_\mu\, v.Q(v)
\end{aligned}
$$

□

Lemma D.5.16. C-PURE *is sound.*

Proof. We first prove the forward direction: For any $a \in \mathcal{M}_I$, if $\big(\ulcorner \mu(X) = 1 \urcorner * \boldsymbol{C}_\mu\, .K(v)\big)((a))$, then there exists some $\mathcal{F}_0, \mu_0, p_0$, and κ_0:

$$(\mathcal{F}_0, \mu_0, p_0) \preceq a$$

$$\forall i \in I.\, \mu_0(i) = \mathsf{bind}(\mu, \kappa_0(i))$$

$$\forall v \in \mathrm{supp}(\mu).\, (K(v))((\mathcal{F}_0, \kappa_0(I)(v), p_0))$$

The pure fact $\ulcorner \mu(X) = 1 \urcorner$ implies that $X \supseteq \mathrm{supp}(\mu)$, and thus for every $v \in \mathrm{supp}(\mu)$, $\ulcorner v \in X \urcorner$. Therefore, $(K(v))((\mathcal{F}_0, \kappa_0(I)(v), p_0))$, which witnesses that $\big(\boldsymbol{C}_\mu\, .\ulcorner v \in X \urcorner * K(v)\big)((a))$.

We then prove the backward direction: if $\boldsymbol{C}_\mu\, .\ulcorner v \in X \urcorner * K(v)$, then there exists

$\mathcal{F}_0, \mu_0, p_0$, and κ_0:

$$(\mathcal{F}_0, \mu_0, p_0) \preceq a$$

$$\forall i \in I.\, \mu_0(i) = \mathsf{bind}(\mu, \kappa_0(i))$$

$$\forall v \in \mathrm{supp}(\mu).\, (\ulcorner v \in X \urcorner * K(v))((\mathcal{F}_0, \kappa_0(I)(v), p_0))$$

Then it must $X \supseteq \mathrm{supp}(\mu)$, which implies that $\ulcorner \mu(X) = 1 \urcorner$. Meanwhile, $\ulcorner v \in X \urcorner * K(v)$ holding on $(\mathcal{F}_0, \kappa_0(I)(v), p_0)$ implies that $K(v)$ holds on $(\mathcal{F}_0, \kappa_0(I)(v), p_0)$ Therefore, $\ulcorner \mu(X) = 1 \urcorner * \boldsymbol{C}_\mu .K(v)$ holds on a. □

D.5.2 Soundness of Primitive WP Rules

Structural Rules

Lemma D.5.17. WP-CONS *is sound.*

Proof. For any resource a, if $(\mathbf{wp}\, t\, \{Q\})(a)$, then

$$\forall \mu_0.\, \forall c.\, (a \cdot c) \preceq \mu_0 \Rightarrow \exists b.\, \big((b \cdot c) \preceq [\![t]\!](\mu_0) \wedge (Q)((b))\big)$$

From the premise $Q \vdash Q'$, and the fact that b must be valid for $(b \cdot c) \preceq [\![t]\!](\mu_0)$ to hold, we have that $Q(b)$ implies $Q'(b)$. Thus, it must

$$\forall \mu_0.\, \forall c.\, (a \cdot c) \preceq \mu_0 \Rightarrow \exists b.\, \big((b \cdot c) \preceq [\![t]\!](\mu_0) \wedge Q'(b)\big),$$

which says $(\mathbf{wp}\, t\, \{Q'\})(a)$. □

Lemma D.5.18. WP-FRAME *is sound.*

Proof. Let $a \in \mathcal{M}_I$ be a valid resource such that it satisfies $P * \mathbf{wp}\, t\, \{Q\}$. By definition, this means that, for some a_1, a_2:

$$a_1 \cdot a_2 \preceq a \tag{D.42}$$

$$P(a_1) \tag{D.43}$$

$$\forall \mu_0, c.\, (a_2 \cdot c) \preceq \mu_0 \Rightarrow \exists b.\, \big((b \cdot c) \preceq [\![t]\!](\mu_0) \wedge Q(b)\big) \tag{D.44}$$

Our goal is to prove a satisfies $\mathbf{wp}\, t\, \{P * Q\}$, which, by unfolding the definitions, amounts to:

$$\exists a' \preceq a.\, \forall \mu_0, c'.\, (a' \cdot c') \preceq \mu_0 \Rightarrow \exists b_1, b.\, ((b_1 \cdot b) \cdot c') \preceq [\![t]\!](\mu_0) \wedge P(b_1) \wedge Q(b) \tag{D.45}$$

Our goal can be proven by instantiating $a' = (a_1 \cdot a_2)$ and $b_1 = a_1$, from which we reduce the goal to proving, for all μ_0, c':

$$((a_1 \cdot a_2) \cdot c') \preceq \mu_0 \Rightarrow \exists b.\, ((a_1 \cdot b) \cdot c') \preceq [\![t]\!](\mu_0) \wedge P(a_1) \wedge Q(b) \tag{D.46}$$

We have that $P(a_1)$ holds by (D.43). By associativity and commutativity of the RA operation, we reduce the goal to:

$$(a_2 \cdot (a_1 \cdot c')) \preceq \mu_0 \Rightarrow \exists b.\, (b \cdot (a_1 \cdot c')) \preceq [\![t]\!](\mu_0) \wedge Q(b) \tag{D.47}$$

This follows by applying assumption (D.44) with $c = (a_1 \cdot c')$. □

Lemma D.5.19. C-WP-SWAP *is sound.*

Proof. By the meaning of conditioning modality and weakest precondition transformer,

$$\begin{aligned}
&(\mathsf{own}_{\mathbf{Var}} \wedge \boldsymbol{C}_\mu\, v.\mathbf{wp}\, t\, \{Q(v)\})(a) \\
\Leftrightarrow\ &\mathsf{own}_{\mathbf{Var}}(a) \wedge \exists \mathcal{F}, \mu, p, \kappa.\, (\mathcal{F}, \mu, p) \preceq a \wedge \forall i \in I.\, \mu(i) = \mathsf{bind}(\mu, \kappa(i)) \\
&\qquad \wedge \forall v \in \mathrm{supp}(\mu).(\mathbf{wp}\, t\, \{Q(v)\})(\mathcal{F}, \kappa(I)(v), p)
\end{aligned}$$

Intuitively, for each v, running t on each fibre $(\mathcal{F}, \kappa(I)(v), p)$ gives a output resource that satisfies $Q(v)$.

Assume $\mathbf{V}(a)$ holds and let $a = (\mathcal{F}_a, \mu_a, p_a)$. By lemma D.2.4, when $(\mathcal{F}, \mu, p) \preceq a$, $\mu = \mathsf{bind}(\mu, \kappa)$ iff that there exists κ'' such that $\mu_a = \mathsf{bind}(\mu, \kappa'')$ and $\kappa(I)(v) \sqsubseteq \kappa''(I)(v)$ for every v. Thus,

$$(\boldsymbol{C}_\mu\, v.\mathbf{wp}\, t\, \{Q(v)\})(\mathcal{F}_a, \mu_a, p_a) \Leftrightarrow \exists \kappa.\, \forall i \in I.\, \mu_a(i) = \mathsf{bind}(\mu, \kappa''(i)) \\ \wedge\, \forall v \in \mathrm{supp}(\mu).(\mathbf{wp}\, t\, \{Q(v)\})(\mathcal{F}, \kappa(I)(v), p)$$

We want to show that

$$\mathbf{wp}\, t\, \{\boldsymbol{C}_\mu\, v.Q(v)\}(a)$$

which is equivalent to

$$\forall \mu'.\, \forall c.\, a \cdot c \preceq \mu' \Rightarrow \exists a'.\, a' \cdot c \preceq [\![t]\!](\mu') \wedge (\boldsymbol{C}_\mu\, Q(v))(a).$$

Let's fix an arbitrary μ', c that satisfy $\mathbf{V}(a \cdot c) \wedge a \cdot c \preceq a_{\mu'}$, we try to construct a corresponding a'. The high-level approach that we will take is to show that running t on a takes us to a resource that is equivalent to bind the set of output resource satisfying $Q(v)$ to μ.

Recall that $a = (\mathcal{F}_a, \mu_a, p_a)$ also satisfies $\mathsf{own}_{\mathbf{Var}}$, which says $\mathcal{F}_a = \Sigma_{\mathbf{Var}}$. We claim that $a \cdot c \preceq (\Sigma_{\mathbf{Var}}, \mu', p_1)$ holds implies that the probability space c is trivial. Say $c = (\mathcal{F}_c, \mu_c, p_c)$, then for any $E \in \mathcal{F}_c$, the event E must also in $\mathcal{F}_a$ and $\Sigma_{\mathbf{Var}}$ because they are the full sigma algebra. By definition of $a \cdot c \preceq (\Sigma_{\mathbf{Var}}, \mu', p_1)$, we have

$$\mu_c(E) \cdot \mu_a(E) = \mu'(E \cap E) = \mu'(E). \tag{D.48}$$

Another implication of $a \cdot c \preceq (\Sigma_{\mathbf{Var}}, \mu', p_1)$ is that we have $\mu_c(E) = \mu'(E)$ and

$\mu_a(E) = \mu'(E)$. Combining with eq. (D.48), we can conclude

$$\mu'(E) \cdot \mu'(E) = \mu'(E),$$

which implies that $\mu_c(E) = \mu'(E) \in \{0, 1\}$. Therefore, c is a trivial probability space and

$$(\mathcal{F}_a, \kappa(I)(v), p_a) \cdot c \preceq (\mathcal{F}_a, \kappa(I)(v), p_a)$$

Furthermore, for every $v \in \text{supp}(\mu)$, we have $(\mathbf{wp}\, t\, \{Q(v)\})(\mathcal{F}, \kappa(I)(v), p)$ which implies

$$\forall \kappa'.(\mathcal{F}_a, \kappa(I)(v), p_a) \cdot c \preceq \kappa'(I)(v) \tag{D.49}$$

$$\Rightarrow \exists a_v.\, (a_v \cdot c \preceq [\![t]\!](\kappa'(I)(v))) \wedge Q(v)(a_v). \tag{D.50}$$

Therefore,

$$a \cdot c \preceq a_{\mu'} \Rightarrow \forall v \in \text{supp}(\mu).(\mathbf{V}((\mathcal{F}_a, \kappa(I)(v), p_a) \cdot c) \wedge (\mathcal{F}_a, \kappa(I)(v), p_a) \cdot c \preceq (\Sigma_{\mathbf{Var}}, \kappa(I)(v), \mathbf{1})$$

(By D.2.7 and D.2.4)

$$\Rightarrow \forall v \in \text{supp}(\mu).\exists a_v.\, \mathbf{V}(a_v \cdot c) \wedge (a_v \cdot c \preceq (\Sigma_{\mathbf{Var}}, [\![t]\!](\kappa(I)(v)), \mathbf{1})) \wedge Q(v)(a_v)$$

(By eq. (D.49))

$$\Rightarrow \forall v \in \text{supp}(\mu).p_{a_v} + p_c \preceq \mathbf{1} \wedge Q(v)(\Sigma_{\mathbf{Var}}, [\![t]\!](\kappa'(I)(v)), \mathbf{1}).$$

(By upwards closure)

Let $a'_v = (\Sigma_{\mathbf{Var}}, [\![t]\!](\kappa'(I)(v)), p_a)$. Because $\mu_c(E) \in \{0, 1\}$ for any $E \in \mathcal{F}_c$, for every v, we have $(\Sigma_{\mathbf{Var}}, [\![t]\!](\kappa'(I)(v))) \cdot (\mathcal{F}_c, \mu_c)$ defined and thus $a'_v \cdot c$ valid. Define

$$a' = (\Sigma_{\mathbf{Var}}, \mathsf{bind}(\mu, \lambda v.\, [\![t]\!](\kappa'(I)(v)), p_a)$$

By lemma D.2.7, $\mathbf{V}(a'_v \cdot c)$ for all $v \in \text{supp}_\mu$ implies $\mathbf{V}(a' \cdot c)$. Also, because $Q(v)(a_v)$ for all $v \in A_\mu$, $(\boldsymbol{C}_\mu\, v.Q(v))(a')$. Thus, $(\mathbf{wp}\, t\, \{\boldsymbol{C}_\mu\, v.Q(v)\})(a)$. □

Program Rules

Lemma D.5.20. WP-SKIP *is sound.*

Proof. Assume $a \in \mathcal{M}_I$ is valid and such that $P(a)$ holds. By unfolding the definition of WP, we need to prove

$$\forall \mu_0. \forall c.\, (a \cdot c) \preceq \mu_0 \Rightarrow \exists b.\, \big((b \cdot c) \preceq [\![t]\!](\mu_0) \land P(b)\big)$$

which follows trivially by $[\![[i\colon \texttt{skip}]]\!](\mu_0) = \mu_0$ and picking $b = a$. □

Lemma D.5.21. WP-SEQ *is sound.*

Proof. Assume $a_0 \in \mathcal{M}_I$ is a valid resource such that $(\mathbf{wp}\ [i\colon t]\ \{\mathbf{wp}\ [i\colon t']\ \{Q\}\})(a_0)$ holds. Our goal is to prove $(\mathbf{wp}\ ([i\colon t; t'])\ \{Q\})(a_0)$ holds, which unfolds by definition of WP into:

$$\forall \mu_0.\, \forall c_0.\, (a_0 \cdot c_0) \preceq \mu_0 \Rightarrow \exists a_2.\, \big((a_2 \cdot c_0) \preceq [\![[i\colon t; t']]\!](\mu_0) \land Q(a_2)\big) \tag{D.51}$$

Take an arbitrary μ_0 and c_0 such that $(a_0 \cdot c_0) \preceq \mu_0$. By unfolding the WPs in the assumption, we have that there exists a $a_1 \in \mathcal{M}_I$ such that:

$$(a_1 \cdot c_0) \preceq [\![[i\colon t]]\!](\mu_0) \tag{D.52}$$

$$\forall \mu_1.\, \forall c_1.\, (a_1 \cdot c_1) \preceq \mu_1 \Rightarrow \exists a_2.\, ((a_2 \cdot c_1) \preceq [\![[i\colon t']]\!](\mu_1) \land Q(a_2)) \tag{D.53}$$

We can apply (D.53) to (D.52) by instantiating μ_1 with $[\![[i\colon t]]\!](\mu_0)$, and c_1 with c_0, obtaining:

$$\exists a_2.\, ((a_2 \cdot c_0) \preceq [\![[i\colon t']]\!]([\![[i\colon t]]\!](\mu_0)) \land Q(a_2))$$

Since by definition, $[\![t; t']\!](\mu_0) = [\![t']\!]([\![t]\!](\mu_0))$, we obtain the goal (D.51) as desired. □

Lemma D.5.22. WP-ASSIGN *is sound.*

Proof. Let $a \in \mathcal{M}_I$ be a valid resource, and let $a(i) = (\mathcal{F}, \mu, p)$. By assumption we have $p(x) = 1$ and $p(y) > 0$ for all $y \in \mathrm{FV}(e)$. We want to show that a satisfies $\mathbf{wp}\,[i\colon \mathtt{x} := e]\,\{\lceil x = e \rceil\}$. This is equivalent to

$$\forall \mu_0.\, \forall c.\, (a \cdot c \preceq \mu_0) \Rightarrow \exists b.\, (b \cdot c \preceq [\![\, [i\colon \mathtt{x} := e] \,]\!](\mu_0) \wedge \lceil x = e \rceil(b))$$

We show this holds by picking b as follows:

$$b \triangleq a[i\colon (\mathcal{F}_b, \mu_b, p)] \quad \mathcal{F}_b \triangleq \{\mathbf{Mem}[\mathbf{Var}], \emptyset, A, \mathbf{Mem}[\mathbf{Var}] \setminus A\} \quad A \triangleq \{s[x \mapsto [\![e]\!](s)] | s \in \mathbf{Mem}[\mathbf{Var}]$$

where μ_b is determined by setting $\mu_b(A) = 1$.

By construction we have that $\lceil x = e \rceil(b)$ holds. To close the proof we then need to show that $(b \cdot c) \preceq [\![\, [i\colon \mathtt{x} := e] \,]\!](\mu_0)$.

Let $c(i) = (\mathcal{F}_c, \mu_c, p_c)$. Observe that by the assumptions on p, we have $\mathbf{V}(b)$ since $\mathcal{F}_b$ is only non-trivial on $\mathrm{FV}(e) \cup \{x\}$; moreover, by the assumption $\mathbf{V}(a \cdot c)$ we have that $\mathbf{V}(p + p_c)$ holds, which means that $p_c(x) = 0$, and thus $\mathcal{F}_c$ is trivial on x.

Let us define the function $\mathrm{pre}\colon \mathscr{P}(\mathbf{Mem}[\mathbf{Var}]) \to \mathscr{P}(\mathbf{Mem}[\mathbf{Var}])$ as:

$$\mathrm{pre}(X) \triangleq \{s | s[x \mapsto [\![e]\!](s)] \in X\}.$$

That is, $\mathrm{pre}(X)$ is the weakest precondition (in the standard sense) of the assignment. By construction, we have:

$$\mathrm{pre}(A) = \mathbf{Mem}[\mathbf{Var}] \qquad \mathrm{pre}(X_1 \cap X_2) = \mathrm{pre}(X_1) \cap \mathrm{pre}(X_2)$$
$$\mathrm{pre}(\mathbf{Mem}[\mathbf{Var}] \setminus A) = \emptyset \qquad \mathrm{pre}(X_c) = X_c \text{ for all } X_c \in \mathcal{F}_c$$

In particular, the latter holds because $\mathcal{F}_c$ is trivial in x.

By unfolding the definition of $[\![\cdot]\!]$, it is easy to check that for every $X \in \Sigma_{\mathbf{Mem}[\mathbf{Var}]}$:

$$[\![\mathtt{x} := e]\!](\mu_0)(X) = \mu_0(\mathrm{pre}(X))$$

We are now ready to show $(b \cdot c) \preceq [\![[i\colon \mathtt{x} := e]]\!](\mu_0)$ by showing that $(\mathcal{F}_b, \mu_b) \circledast (\mathcal{F}_c, \mu_c) = (\mathcal{F}_b \oplus \mathcal{F}_c, [\![\mathtt{x} := e]\!](\mu_0)|_{(\mathcal{F}_b \oplus \mathcal{F}_c)})$ where $\mu_0 = \mu_0(i)$. To show this it suffices to prove that for every $X_b \in \mathcal{F}_b$ and every $X_c \in \mathcal{F}_c$, $[\![\mathtt{x} := e]\!](\mu_0)(X_b \cap X_c) = \mu_b(X_b) \cdot \mu_c(X_c)$. We proceed by case analysis on X_b:

Case: $X_b = A$ Then:

$$\begin{aligned}
[\![\mathtt{x} := e]\!](\mu_0)(A \cap X_c) &= \mu_0(\mathrm{pre}(A \cap X_c)) \\
&= \mu_0(\mathrm{pre}(A) \cap \mathrm{pre}(X_c)) \\
&= \mu_0(\mathbf{Mem}[\mathbf{Var}] \cap \mathrm{pre}(X_c)) \\
&= \mu_0(\mathrm{pre}(X_c)) \\
&= \mu_b(A) \cdot \mu_0(X_c) \\
&= \mu_b(A) \cdot \mu_c(X_c)
\end{aligned}$$

Case: $X_b = \mathbf{Mem}[\mathbf{Var}] \setminus A$ Then:

$$\begin{aligned}
[\![\mathtt{x} := e]\!](\mu_0)(\mathbf{Mem}[\mathbf{Var}] \setminus A \cap X_c) &= \mu_0(\mathrm{pre}((\mathbf{Mem}[\mathbf{Var}] \setminus A) \cap X_c)) \\
&= \mu_0(\mathrm{pre}(\mathbf{Mem}[\mathbf{Var}] \setminus A) \cap \mathrm{pre}(X_c)) \\
&= \mu_0(\emptyset \cap \mathrm{pre}(X_c)) \\
&= 0 \\
&= \mu_b(\mathbf{Mem}[\mathbf{Var}] \setminus A) \cdot \mu_c(X_c)
\end{aligned}$$

Case: $X_b = \mathbf{Mem}[\mathbf{Var}]$ **or** $X_b = \emptyset$ Analogous to the previous cases. □

Lemma D.5.23. WP-SAMP *is sound.*

Proof. Assume $a \in \mathcal{M}_I$ is valid and such that $a(i) = (\mathcal{F}, \mu, p)$, with $p(x) = 1$. Our goal is to show that a satisfies $\mathbf{wp}\,[i\colon \mathrm{x} :\approx d\,(\vec{v})\,]\,\{x \not\approx d(\vec{v})\}$ which is equivalent to proving, for all μ_0 and for all c:

$$(a \cdot c \preceq \mu_0) \Rightarrow \exists b.\, \big(b \cdot c \preceq [\![\, [i\colon \mathrm{x} :\approx d\,(\vec{v})\,] \,]\!](\mu_0) \wedge (x \not\approx d(\vec{v}))(b)\big) \tag{D.54}$$

Let $\mu_0 = \mu_0(i)$ and $\mu_1 = [\![\, \mathrm{x} :\approx d\,(\vec{v})\,]\!](\mu_0)$. Moreover, let $c(i) = (\mathcal{F}_c, \mu_c, p_c)$. Observe that by the assumptions on p and validity of $a \cdot c$, we have $p_c(x) = 0$, which means $\mathcal{F}_c$ is trivial on x. We aim to prove (D.54) by letting

$$b \triangleq a[i\colon (\mathcal{F}_b, \mu_b, p_b)] \qquad \mu_b \triangleq \mu_1|_{\mathcal{F}_b}$$
$$\mathcal{F}_b \triangleq \sigma\big(\{[\,]\}\{s \in \mathbf{Mem}[\mathbf{Var}] | s(x) = v\} | v \in \mathbf{Val}\big) \qquad p_b \triangleq (x{:}1)$$

Note that by construction $\mathbf{V}(p_b + p_c)$, and $\mathbf{V}(b)$ since $\mathcal{F}_b$ is only non-trivial in x. Similarly to the proof for section 5.4, we define the function $\mathrm{pre}\colon \mathcal{P}(\mathbf{Mem}[\mathbf{Var}]) \to \mathcal{P}(\mathbf{Mem}[\mathbf{Var}])$ as:

$$\mathrm{pre}(X) \triangleq \{s | \exists v \in \mathbf{Val}.\, s[x \mapsto v] \in X\}.$$

Since $\mathcal{F}_c$ is trivial on x, for all $X_c \in \mathcal{F}_c$, $\mathrm{pre}(X_c) = X_c$. Moreover, for all $X_b \in \mathcal{F}_b \setminus \{\emptyset\}$, $\mathrm{pre}(X_b) = \mathbf{Mem}[\mathbf{Var}]$, since X_b is trivial on every variable except x.

By unfolding the definitions, we have:

$$\begin{aligned} \mu_1(X) &= [\![\, \mathrm{x} :\approx d\,(\vec{v})\,]\!](\mu_0)(X) \\ &= \sum_{s \in X} \mu_0(\mathrm{pre}(s)) \cdot [\![d]\!](\vec{v})(s(x)) \end{aligned}$$

We now show that $(\mathcal{F}_b, \mu_b) \circledast (\mathcal{F}_c, \mu_c) = (\mathcal{F}_b \oplus \mathcal{F}_c, \mu_1|_{(\mathcal{F}_b \oplus \mathcal{F}_c)})$ by showing that for all $X_b \in \mathcal{F}_b$ and $X_c \in \mathcal{F}_c$: $\mu_1(X_b \cap X_c) = \mu_b(X_b) \cdot \mu_c(X_c)$. To prove this we first define $\mathrm{V}\colon \mathcal{P}(\mathbf{Mem}[\mathbf{Var}]) \to \mathcal{P}(\mathbf{Val})$ as $\mathrm{V}(X) \triangleq \{s(x) | s \in X\}$, and $S_w \triangleq \{s | s(x) = w\}$. We

observe that $X_b = \biguplus_{w \in \mathrm{V}(X_b)} S_w$, and thus $X_b \cap X_c = \biguplus_{w \in \mathrm{V}(X_b)} (X_c \cap S_w)$; moreover, $\mathrm{pre}(X_c \cap S_w) = \{s | s[x \mapsto w] \in X_c\} = X_c$. Thus, we can calculate:

$$\begin{aligned}
\mu_1(X_b \cap X_c) &= \sum_{s \in X_b \cap X_c} \mu_0(\mathrm{pre}(s)) \cdot [\![d]\!](\vec{v})(s(x)) \\
&= \sum_{w \in \mathrm{V}(X_b)} \sum_{s \in X_c \cap S_w} \mu_0(\mathrm{pre}(s)) \cdot [\![d]\!](\vec{v})(w) \\
&= \sum_{w \in \mathrm{V}(X_b)} \left([\![d]\!](\vec{v})(w) \cdot \sum_{s \in X_c \cap S_w} \mu_0(\mathrm{pre}(s)) \right) \\
&= \left(\sum_{w \in \mathrm{V}(X_b)} [\![d]\!](\vec{v})(w) \cdot \mu_0(\mathrm{pre}(X_c \cap S_w)) \right) \\
&= \left(\sum_{w \in \mathrm{V}(X_b)} [\![d]\!](\vec{v})(w) \right) \cdot \mu_0(X_c) \\
&= \mu_b(X_b) \cdot \mu_c(X_c)
\end{aligned}$$

The last equation is given by $a \cdot c \preceq \mu_0$ which implies that $\mu_c = \mu_0|_{\mathcal{F}_c}$, and by:

$$\begin{aligned}
\mu_b(X_b) = \mu_1(X_b) &= \sum_{s \in X_b} \mu_0(\mathrm{pre}(s)) \cdot [\![d]\!](\vec{v})(s(x)) \\
&= \sum_{w \in \mathrm{V}(X_b)} \sum_{s \in S_w} \mu_0(\mathrm{pre}(s)) \cdot [\![d]\!](\vec{v})(w) \\
&= \sum_{w \in \mathrm{V}(X_b)} [\![d]\!](\vec{v})(w)
\end{aligned}$$

Finally, we need to show $(x \mathbin{\text{§}} d(\vec{v}))(b)$ which amounts to proving $x \prec (\mathcal{F}_b, \mu_b)$ and $[\![d]\!](\vec{v}) = \mu_b \circ x^{-1}$. The former holds because by construction x is measurable in $\mathcal{F}_b$. For the latter, for all $W \subseteq \mathbf{Val}$:

$$(\mu_b \circ x^{-1})(W) = \mu_b(x^{-1}(W)) = \sum_{w \in \mathrm{V}(x^{-1}(W))} [\![d]\!](\vec{v})(w) = \sum_{w \in W} [\![d]\!](\vec{v})(w) = [\![d]\!](\vec{v})(W). \quad \square$$

Lemma D.5.24. WP-IF-PRIM *is sound.*

Proof. For any valid resource a,

$$
\begin{aligned}
&(\text{if } v \text{ then } \mathbf{wp}\,[i\colon t_1]\,\{Q(1)\} \text{ else } \mathbf{wp}\,[i\colon t_2]\,\{Q(0)\})(a) \\
\Leftrightarrow &\begin{cases} (\mathbf{wp}\,[i\colon t_1]\,\{Q(1)\})(a) & \text{if } v \doteq 1 \\ (\mathbf{wp}\,[i\colon t_2]\,\{Q(0)\})(a) & \text{otherwise} \end{cases} \\
\Leftrightarrow \forall \mu_0. \forall c.\, (a \cdot c) \preceq \mu_0 \Rightarrow &\begin{cases} \exists b.\, (b \cdot c) \preceq [\![i : t_1]\!](\mu_0) \wedge Q(1)(b) & \text{if } v \doteq 1 \\ \exists b.\, (b \cdot c) \preceq [\![i : t_2]\!](\mu_0) \wedge Q(0)(b) & \text{otherwise} \end{cases} \\
\Leftrightarrow \forall \mu_0. \forall c.\, (a \cdot c) \preceq \mu_0 \Rightarrow &\exists b.\, (b \cdot c) \preceq [\![i : \text{if } v \text{ then } t_1 \text{ else } t_2]\!](\mu_0) \wedge Q(v \doteq 1)(b) \\
\Rightarrow &(\mathbf{wp}\,[i\colon \text{if } v \text{ then } t_1 \text{ else } t_2]\,\{Q(v \doteq 1)\})(a)
\end{aligned}
$$

□

Lemma D.5.25. WP-BIND *is sound.*

Proof. For any resource $a = (\mathcal{F}, \mu, p)$, $(\lceil e = v \rceil * \mathbf{wp}\left[i\colon \mathcal{E}[v]\right]\{Q\})(\mathcal{F}, \mu, p)$ iff there exists $(\mathcal{F}_1, \mu_1, p_1), (\mathcal{F}_2, \mu_2, p_2)$ such that

$$
\begin{gathered}
(\lceil e = v \rceil)(\mathcal{F}_1, \mu_1, p_1) \\
(\mathbf{wp}\left[i\colon \mathcal{E}[v]\right]\{Q\})(\mathcal{F}_2, \mu_2, p_2) \\
(\mathcal{F}_1, \mu_1, p_1) \cdot (\mathcal{F}_2, \mu_2, p_2) \preceq (\mathcal{F}, \mu, p)
\end{gathered}
$$

By the upwards closure, we also have

$$
\begin{gathered}
(\lceil e = v \rceil)(\mathcal{F}, \mu, p) \\
(\mathbf{wp}\left[i\colon \mathcal{E}[v]\right]\{Q\})(\mathcal{F}, \mu, p)
\end{gathered}
$$

The fact that $(\lceil e = v \rceil)(\mathcal{F}_1, \mu_1, p_1)$ implies that $\mu_1((e = v)^{-1}(\mathsf{True})) = 1$, which implies that $[\![e]\!](s) = v$ for all $s \in \mathrm{supp}(\mu_1(i))$.

By lemma D.1.3, we have for any $s \in \mathbf{Mem}[\mathbf{Var}]$,

$$\mathcal{K}[\![\mathcal{E}[e]]\!](s) = \mathcal{K}[\![\mathcal{E}[[\![e]\!](s)]]\!](s),$$

which implies that for any μ_0 over $\Sigma_{\mathbf{Mem}[\mathbf{Var}]}$

$$\begin{aligned}
[\![\mathcal{E}[e]]\!](\mu_0) &= s \leftarrow \mu_0; \mathcal{K}[\![\mathcal{E}[e]]\!](s) \\
&= s \leftarrow \mu_0; \mathcal{K}[\![\mathcal{E}[[\![e]\!](s)]]\!](s) \\
&= s \leftarrow \mu_0; \mathcal{K}[\![\mathcal{E}[v]]\!](s) \\
&= [\![\mathcal{E}[v]]\!](\mu_0).
\end{aligned}$$

Define $\mu_0' = [\![[i{:}\,\mathcal{E}[v]]]\!]\mu_0$. Thus, $(\mathbf{wp}\left[i{:}\,\mathcal{E}[v]\right]\{Q\})(a)$ iff

$$\forall \mu_0.\, \forall c.\, (\mathbf{V}(a \cdot c) \wedge a \cdot c \preceq a_{\mu_0}) \Rightarrow \exists a'.\, (\mathbf{V}(a' \cdot c) \wedge a' \cdot c \preceq a_{\mu_0'} \wedge Q(a'))$$

iff

$$\forall \mu_0.\, \forall c.\, (\mathbf{V}(a \cdot c) \wedge a \cdot c \preceq a_{\mu_0}) \Rightarrow \exists a'.\, (\mathbf{V}(a' \cdot c) \wedge a' \cdot c \preceq a_{\mu_0'} \wedge Q(a'))$$

iff $\left(\mathbf{wp}\left[i{:}\,\mathcal{E}[e]\right]\{Q\}\right)((a))$. □

Lemma D.5.26. WP-LOOP-UNF *is sound.*

Proof. By definition,

$$\begin{aligned}
[\![\texttt{repeat}\ (n+1)\ t]\!](\mu) &= \left(s \leftarrow \mu; s' \leftarrow loop_t(n, s); \mathcal{K}[\![t]\!](s')\right) \\
&= [\![(\texttt{repeat}\ n\ t); t]\!](\mu)
\end{aligned}$$

thus the rule follows from the argument of lemma D.5.21. □

Lemma D.5.27. WP-LOOP *is sound.*

Proof. By induction on n.

Base case $n = 0$ Analogously to lemma D.5.20 since, by definition, $[\![\texttt{repeat}\ 0\ t]\!](\mu_0) = \mu_0$.

Induction step $n > 0$ By induction hypothesis $P(0) \vdash \mathbf{wp}\,[j\colon \texttt{repeat}\ (n-1)\ t]\,\{P(n-1)\}$ holds, and we want to show that $P(0) \vdash \mathbf{wp}\,[j\colon \texttt{repeat}\ n\ t]\,\{P(n)\}$. By lemma D.5.26, it suffices to show $P(0) \vdash \mathbf{wp}\,[j\colon \texttt{repeat}\ (n-1)\ t]\,\{\mathbf{wp}\,[j\colon t]\,\{P(n)\}\}$. By applying the induction hypothesis and lemma D.5.17 we are left with proving $P(n-1) \vdash \mathbf{wp}\,[j\colon t]\,\{P(n)\}$ which is implied by the premise of the rule with $i = n - 1 < n$. □

D.5.3 Soundness of Derived Rules

In this section we provide derivations for the rules we claim are derivable in BLUEBELL.

Ownership and Distributions

Lemma D.5.28. SURE-DIRAC *is sound.*

Proof.

$$\begin{aligned} E \twoheadrightarrow \delta_v &\dashv\vdash \exists \mathcal{F}, \mu.\, \mathsf{Own}((\mathcal{F}, \mu)) * \ulcorner \mu \circ E^{-1} = \delta_v \urcorner \\ &\dashv\vdash \exists \mathcal{F}, \mu.\, \mathsf{Own}((\mathcal{F}, \mu)) * \ulcorner \mu \circ (E = v)^{-1} = \delta_{\mathsf{True}} \urcorner \\ &\dashv\vdash \lceil E = v \rceil \end{aligned}$$

□

Lemma D.5.29. SURE-EQ-INJ *is sound.*

Proof.

$$\begin{aligned}
\lceil E = v \rceil * \lceil E = v' \rceil &\vdash E \twoheadrightarrow \delta_v * E \twoheadrightarrow \delta_{v'} && \text{(SURE-DIRAC)} \\
&\vdash E \twoheadrightarrow \delta_v \wedge E \twoheadrightarrow \delta_{v'} \\
&\vdash \ulcorner \delta_v = \delta_{v'} \urcorner && \text{(DIST-INJ)} \\
&\vdash \ulcorner v = v' \urcorner && \square
\end{aligned}$$

Lemma D.5.30. SURE-SUB *is sound.*

Proof.

$$\begin{aligned}
E_1 \twoheadrightarrow \mu * \lceil (E_2 = f(E_1)) \rceil &\vdash \boldsymbol{C}_\mu\, v.\, \lceil E_1 = v \rceil * \lceil (E_2 = f(E_1)) \rceil && \text{(C-UNIT-R, C-FRAME)} \\
&\vdash \boldsymbol{C}_\mu\, v.\, \lceil E_1 = v \wedge E_2 = f(E_1) \rceil && \text{(SURE-MERGE)} \\
&\vdash \boldsymbol{C}_\mu\, v.\, \lceil E_2 = f(v) \rceil && \text{(C-CONS)} \\
&\vdash \boldsymbol{C}_\mu\, v.\, \boldsymbol{C}_{\delta_{f(v)}}\, v'.\, \lceil E_2 = v' \rceil && \text{(C-UNIT-L)} \\
&\vdash \boldsymbol{C}_{\mu'}\, v'.\, \lceil E_2 = v' \rceil && \text{(C-ASSOC, C-SURE-PROJ)}
\end{aligned}$$

where $\mu' = \mathsf{bind}(\mu, \lambda x.\, \delta_{f(x)}) = \mu \circ f^{-1}$. By C-UNIT-R we thus get $E_2 \twoheadrightarrow \mu \circ f^{-1}$. $\square$

Lemma D.5.31. DIST-FUN *is sound.*

Proof. Assume $E : \mathbf{Mem}[\mathbf{Var}] \to A$ and $f : A \to B$, then:

$$\begin{aligned}
E \twoheadrightarrow \mu &\vdash \boldsymbol{C}_\mu\, v.\, \lceil (E = v) \rceil && \text{(C-UNIT-R)} \\
&\vdash \boldsymbol{C}_\mu\, v.\, \lceil (f \circ E) = f(v) \rceil && \text{(C-CONS)} \\
&\vdash \boldsymbol{C}_\mu\, v.\, \boldsymbol{C}_{\delta_{f(v)}}\, v'.\, \lceil (f \circ E) = v' \rceil && \text{(C-UNIT-L)} \\
&\vdash \boldsymbol{C}_{\mu'}\, v'.\, \lceil (f \circ E) = v' \rceil && \text{(C-ASSOC, C-SURE-PROJ)}
\end{aligned}$$

where $\mu' = \mathsf{bind}(\mu, \lambda x.\, \delta_{f(x)}) = \mu \circ f^{-1}$. By C-UNIT-R we thus get $(f \circ E) \twoheadrightarrow \mu \circ f^{-1}$. $\square$

Lemma D.5.32. DIRAC-DUP *is sound.*

Proof.

$$
\begin{aligned}
E \twoheadrightarrow \delta_v &\vdash \lceil E = v \rceil && \text{(SURE-DIRAC)} \\
&\vdash \lceil E = v \rceil * \lceil E = v \rceil && \text{(SURE-MERGE)} \\
&\vdash E \twoheadrightarrow \delta_v * E \twoheadrightarrow \delta_v && \text{(SURE-DIRAC)}
\end{aligned}
$$

□

Lemma D.5.33. DIST-SUPP *is sound.*

Proof.

$$
\begin{aligned}
E \twoheadrightarrow \mu &\vdash \boldsymbol{C}_\mu\, v. \lceil E = v \rceil && \text{(C-UNIT-R)} \\
&\vdash \ulcorner \mu(\mathrm{supp}(\mu)) = 1 \urcorner * \boldsymbol{C}_\mu\, v. \lceil E = v \rceil \\
&\vdash \boldsymbol{C}_\mu\, v. \left(\ulcorner v \in \mathrm{supp}(\mu) \urcorner * \lceil E = v \rceil \right) && \text{(C-PURE)} \\
&\vdash \boldsymbol{C}_\mu\, v. \left(\lceil E = v \rceil * \lceil E \in \mathrm{supp}(\mu) \rceil \right) \\
&\vdash \left(\boldsymbol{C}_\mu\, v. \lceil E = v \rceil \right) * \lceil E \in \mathrm{supp}(\mu) \rceil && \text{(SURE-STR-CONVEX)} \\
&\vdash E \twoheadrightarrow \mu * \lceil E \in \mathrm{supp}(\mu) \rceil && \text{(C-UNIT-R)}
\end{aligned}
$$

□

Lemma D.5.34. PROD-UNSPLIT *is sound.*

Proof.

$$
\begin{aligned}
E_1 \twoheadrightarrow \mu_1 * E_2 \twoheadrightarrow \mu_2 &\vdash \boldsymbol{C}_{\mu_1}\, v_1. \boldsymbol{C}_{\mu_2}\, v_2. \left(\lceil E_1 = v_1 \rceil * \lceil E_2 = v_2 \rceil \right) && \text{(C-UNIT-R, C-FRAME)} \\
&\vdash \boldsymbol{C}_{\mu_1}\, v_1. \boldsymbol{C}_{\mu_2}\, v_2. \lceil (E_1, E_2) = (v_1, v_2) \rceil && \text{(SURE-MERGE)} \\
&\vdash \boldsymbol{C}_{\mu_1 \otimes \mu_2} (v_1, v_2). \lceil (E_1, E_2) = (v_1, v_2) \rceil && \text{(C-ASSOC)} \\
&\vdash (E_1, E_2) \twoheadrightarrow \mu_1 \otimes \mu_2 && \text{(C-UNIT-R)}
\end{aligned}
$$

□

Joint conditioning

Lemma D.5.35. C-FUSE *is sound.*

Proof. Recall that $\mu \prec \kappa \triangleq \lambda(v, w).\, \mu(v)\kappa(v)(w)$. which can be reformulated as $\mu \prec \kappa = \mathsf{bind}(\mu, \lambda v.\, (\mathsf{bind}(\kappa(v), \lambda w.\, \mathbf{return}(v, w))))$.

The ($\vdash$) direction is an instance of C-ASSOC.

The ($\dashv$) direction follows from C-UNASSOC:

$$\begin{aligned}
\boldsymbol{C}_{\mu \prec \kappa}(v', w').\, K(v', w') &\vdash \boldsymbol{C}_{\mu}\, v.\, \boldsymbol{C}_{\mathsf{bind}(\kappa(v), \lambda w.\, \delta_{(v,w)})}(v', w').\, K(v', w') && \text{(C-UNASSOC)} \\
&\vdash \boldsymbol{C}_{\mu}\, v.\, \boldsymbol{C}_{\kappa(v)}\, w.\, \boldsymbol{C}_{\delta_{(v,w)}}(v', w').\, K(v', w') && \text{(C-UNASSOC)} \\
&\vdash \boldsymbol{C}_{\mu}\, v.\, \boldsymbol{C}_{\kappa(v)}\, w.\, K(v, w) && \text{(C-UNIT-L)}
\end{aligned}$$

□

Lemma D.5.36. C-SWAP *is sound.*

Proof.

$$\begin{aligned}
\boldsymbol{C}_{\mu_1}\, v_1.\, \boldsymbol{C}_{\mu_2}\, v_2.\, K(v_1, v_2) &\vdash \boldsymbol{C}_{\mu_1 \otimes \mu_2}(v_1, v_2).\, K(v_1, v_2) && \text{(C-FUSE)} \\
&\vdash \boldsymbol{C}_{\mu_2}\, v_2.\, \boldsymbol{C}_{\mu_1}\, v_1.\, K(v_1, v_2) && \text{(C-FUSE)}
\end{aligned}$$

Where

$$\mu_1 \otimes \mu_2 = \mu_1 \prec (\lambda\_.\, \mu_2) = \mu_2 \prec (\lambda\_.\, \mu_1)$$

justifies the applications of C-FUSE. □

Lemma D.5.37. SURE-CONVEX *is sound.*

Proof. By SURE-STR-CONVEX with $K = \mathsf{True}$. □

Lemma D.5.38. *Section 5.3.5 is sound.*

Proof.

$$\begin{aligned}
\boldsymbol{C}_\mu\, v.E \mathbin{\$} \mu' &\vdash \boldsymbol{C}_\mu\, v.\, \boldsymbol{C}_{\mu'}\, w.\lceil E = w\rceil && \text{(C-UNIT-R)}\\
&\vdash \boldsymbol{C}_{\mu'}\, w.\, \boldsymbol{C}_\mu\, v.\lceil E = w\rceil && \text{(C-SWAP)}\\
&\vdash \boldsymbol{C}_{\mu'}\, w.\lceil E = w\rceil && \text{(SURE-CONVEX)}\\
&\vdash E \mathbin{\$} \mu' && \text{(C-UNIT-R)}
\end{aligned}$$

□

Lemma D.5.39. *The following rule is sound:*

$$\frac{\forall (v, \_) \in \mathrm{supp}(\mu).\forall \mu'.\, \boldsymbol{C}_{\mu'}\, w.\, P(v) \vdash P(v)}{\boldsymbol{C}_\mu(v, w).\, P(v) \dashv\vdash \boldsymbol{C}_{\mu\circ\pi^{-1}}\, v.\, P(v)}$$

Proof. Assume that for all $(v, \_) \in \mathrm{supp}(\mu)$, $\forall \mu'.\, \boldsymbol{C}_{\mu'}\, w.\, P(v) \vdash P(v)$ (i.e. $P(v)$ is convex). By lemma D.1.2 there is some κ such that $\mu = (\mu \circ \pi^{-1}) \prec \kappa$. Then:

$$\begin{aligned}
\boldsymbol{C}_\mu(v, w).\, P(v) &\dashv\vdash \boldsymbol{C}_{\mu\circ\pi^{-1}}\, v.\, \boldsymbol{C}_{\kappa(v)}\, w.\, P(v) && \text{(C-FUSE)}\\
&\dashv\vdash \boldsymbol{C}_{\mu\circ\pi^{-1}}\, v.\, P(v)
\end{aligned}$$

The last step is justified by the convexity assumption in the ($\vdash$) direction, and by C-TRUE and C-FRAME in the ($\dashv$) direction. □

Lemma D.5.40. C-SURE-PROJ *is sound.*

Proof. By lemma D.5.39 and lemma D.5.37. □

Lemma D.5.41. *Section 5.3.5 is sound.*

Proof.

$$\begin{aligned}
\boldsymbol{C}_{\mu_1}\, v_1.\, \left(\lceil E_1 = v_1 \rceil * E_2 \twoheadleftarrow \mu_2\right) &\vdash \boldsymbol{C}_{\mu_1}\, v_1.\, \left(\lceil E_1 = v_1 \rceil * \boldsymbol{C}_{\mu_2}\, v_2.\, \lceil E_2 = v_2 \rceil\right) && \text{(C-UNIT-R)} \\
&\vdash \boldsymbol{C}_{\mu_1}\, v_1.\, \boldsymbol{C}_{\mu_2}\, v_2.\, \left(\lceil E_1 = v_1 \rceil * \lceil E_2 = v_2 \rceil\right) && \text{(C-FRAME)} \\
&\vdash \boldsymbol{C}_{\mu_1}\, v_1.\, \boldsymbol{C}_{\mu_2}\, v_2.\, \lceil E_1 = v_1 \wedge E_2 = v_2 \rceil && \text{(SURE-MERGE)} \\
&\vdash \boldsymbol{C}_{\mu_1 \otimes \mu_2}(v_1, v_2).\, \lceil (E_1, E_2) = (v_1, v_2) \rceil && \text{(C-ASSOC)} \\
&\vdash (E_1, E_2) \twoheadleftarrow (\mu_1 \otimes \mu_2) && \text{(C-UNIT-R)} \\
&\vdash E_1 \twoheadleftarrow \mu_1 * E_2 \twoheadleftarrow \mu_2 && \text{(PROD-SPLIT)}
\end{aligned}$$

□

Lemma D.5.42. *Section 5.3.5 is sound.*

Proof. By lemma D.5.39 and lemma D.5.38. □

Weakest Precondition

Lemma D.5.43. *Section 5.4 is sound.*

Proof. Special case of WP-LOOP with $n = 0$, which makes the premises trivial. □

Lemma D.5.44. *Section 5.4 is sound.*

Proof. From the premises, we derive:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P * \lceil e = 1 \rceil \vdash \mathbf{wp}\,[1{:}\,t_1]\,\{Q(1)\} \qquad P * \lceil e = 0 \rceil \vdash \mathbf{wp}\,[1{:}\,t_2]\,\{Q(0)\}}{\forall b \in \{0,1\}.\, P * \lceil e = 1 \rceil \vdash \text{if } b \text{ then } \mathbf{wp}\,[1{:}\,t_1]\,\{Q(1)\} \text{ else } \mathbf{wp}\,[1{:}\,t_2]\,\{Q(0)\}}\,\text{WP-IF-PRIM}}{\forall b \in \{0,1\}.\, P * \lceil e = b \rceil \vdash \mathbf{wp}\,[1{:}\,(\mathtt{if}\ b\ \mathtt{then}\ t_1\ \mathtt{else}\ t_2)]\,\{Q(b \doteq 1)\}}\,\text{WP-BIND}}{\forall b \in \{0,1\}.\, P * \lceil e = b \rceil \vdash \mathbf{wp}\,[1{:}\,(\mathtt{if}\ \in\ \mathtt{then}\ t_1\ \mathtt{else}\ t_2)]\,\{Q(b \doteq 1)\}}\,\text{C-CONS}}{\boldsymbol{C}_\beta\, b.\, (P * \lceil e = b \rceil) \vdash \boldsymbol{C}_\beta\, b.\, \mathbf{wp}\,[1{:}\,(\mathtt{if}\ \in\ \mathtt{then}\ t_1\ \mathtt{else}\ t_2)]\,\{Q(b \doteq 1)\}}\,\text{C-UNIT-R,C-FRAME}}{P * e \,\&\, \beta \vdash \boldsymbol{C}_\beta\, b.\, \mathbf{wp}\,[1{:}\,(\mathtt{if}\ \in\ \mathtt{then}\ t_1\ \mathtt{else}\ t_2)]\,\{Q(b \doteq 1)\}}\,\text{C-WP-SWAP}}{P * e \,\&\, \beta \vdash \mathbf{wp}\,[1{:}\,(\mathtt{if}\ \in\ \mathtt{then}\ t_1\ \mathtt{else}\ t_2)]\,\{\boldsymbol{C}_\beta\, b.\, Q(b \doteq 1)\}}$$

□