A Bunched Logic for Conditional Independence

Jialu Bao
University of Wisconsin–Madison

Simon Docherty
University College London

Justin Hsu
University of Wisconsin–Madison

Alexandra Silva
University College London

***Abstract*—*Independence* and *conditional independence* are fundamental concepts for reasoning about groups of random variables in probabilistic programs. Verification methods for independence are still nascent, and existing methods cannot handle conditional independence. We extend the logic of bunched implications (BI) with a non-commutative conjunction and provide a model based on Markov kernels; conditional independence can be directly captured as a logical formula in this model. Noting that Markov kernels are Kleisli arrows for the distribution monad, we then introduce a second model based on the powerset monad and show how it can capture *join dependency*, a non-probabilistic analogue of conditional independence from database theory. Finally, we develop a program logic for verifying conditional independence in probabilistic programs.**

I. Introduction

The study of probabilistic programming languages and their semantics dates back to the 1980s, starting from the seminal work of Kozen [1]. The last decade has seen a surge of richer probabilistic languages [2, 3, 4], motivated by applications in machine learning, and accompanying research into their semantics [5, 6, 7]. This burst of activity has also created new opportunities and challenges for formal verification.

Independence and conditional independence are two fundamental properties that are poorly handled by existing verification methods. Intuitively, two random variables are *probabilistically independent* if information about one gives no information about the other (for example, the results of two coin flips). *Conditional independence* is more subtle: two random variables X and Y are independent conditioned on a third variable Z if for every fixed value of Z, information about one of X and Y gives no information about the other.

Both forms of independence are useful for modelling and verification. *Probabilistic independence* enables compositional reasoning about groups of random variables: if a group of random variables are independent, then their joint distribution is precisely described by the distribution of each variable in isolation. It also captures the semantics of random sampling constructs in probabilistic languages, which generate a fresh random quantity that is independent of the program state. *Conditional independence* often arises in programs with probabilistic control flow, as conditioning models probabilistic branching. Bayesian networks encode conditional independence statements in complex distributions, and conditional independence captures useful properties in many applications. For instance, criteria ensuring that algorithms do not discriminate based on sensitive characteristics (e.g., gender or race) can be formulated using conditional independence [8].

978-1-6654-4895-6/21/$31.00

Aiming to prove independence in probabilistic programs, Barthe et al. [9] recently introduced Probabilistic Separation Logic (PSL) and applied it to formalize security for several well-known constructions from cryptography. The key ingredient of PSL is a new model of the logic of bunched implications (BI), in which separation is interpreted as probabilistic independence. While PSL enables formal reasoning about independence, it does not support conditional independence. The core issue is that the model of BI underlying PSL provides no means to describe the distribution of one set of variables obtained by fixing (*conditioning*) another set of variables to take specific values. Accordingly, one cannot capture the basic statement of conditional independence—X and Y are independent random variables conditioned on any value of Z.

In this paper, we develop a logical framework for *formal reasoning about notions of dependence and independence*. Our approach is inspired by PSL but the framework is more sophisticated: to express conditional independence, we develop a novel assertion logic extending BI with new connectives—$\multimap$ and its adjoints. The key intuition is that conditional independence can be expressed as independence plus composition of *Markov kernels*; as our leading example, we give a kernels model of our logic.

Then, we show how to adapt the probabilistic model to other settings. As is well-known in category theory, Markov kernels are the arrows in the Kleisli category of the distribution monad. By varying the monad, our logic smoothly extends to analogues of conditional independence in other domains. To demonstrate, we show how replacing the distribution monad by the *powerset* monad gives a model where we can capture *join/multivalued dependencies* in relational algebra and database theory. We also show that the *semi-graphoid laws*, introduced by Pearl and Paz [10] in their work axiomatizing conditional independence, can be translated into formulas that are valid in both of our models.

The rest of the paper is organized as follows. We give a bird's-eye view in Section II, providing intuitions on our design choices and highlighting differences with existing work. Section III presents the main contribution: the design of DIBI, a new bunched logic to reason about dependence and independence. We show that the proof system of DIBI is

sound and complete with respect to its Kripke semantics. Then, we present two concrete models in Section IV, based on probability distributions and relations. In Section V, we consider how to express dependencies in DIBI: we show that the same logical formula captures conditional independence and join dependency in our two models, and our models validate the semi-graphoid laws. Finally, in Section VI, we design a program logic with DIBI assertions, and use it to verify conditional independence in two probabilistic programs.

II. Overview of the contributions

The logic DIBI: The starting point of our work is the logic of bunched implications (BI) [11]. BI extends intuitionistic propositional logic with substructural connectives to facilitate reasoning about sharing and separation of resources, an idea most prominently realized in Separation Logic's handling of heap-manipulating programs [12]. The novel connectives are a *separating conjunction* $P * Q$, intuitively stating that P and Q hold in separate resources, and its adjoint $\mathrel{-\!\!*}$, called *magic wand*. We will extend BI with a non-commutative conjunction, written $P \fatsemi Q$. Intuitively, $\fatsemi$ expresses a *possible dependency* of Q on P. The end result is a logic with two conjunctive connectives—$*$ and $\fatsemi$—capturing notions of *independence* and *dependence*. We call the logic *Dependence and Independence Bunched Implications (DIBI)*.

To give a semantics to our logic, we start from the semantics of BI. The simplest BI models are *partial resource monoids*: Kripke structures $(M, \sqsubseteq, \circ, e)$ in which $\circ$ is an order-preserving, partial, commutative monoid operation with unit e. The operation $\circ$ allows interpreting the separating conjunction $P * Q$ and magic wand $P \mathrel{-\!\!*} Q$. For example, the probabilistic model of BI underlying PSL [9] is a partial resource monoid: by taking M to be the set of distributions over program memories and $\circ$ to be the independent product of distributions over memories with disjoint variables, the interpretation of $P * Q$ gives the desired notion of *probabilistic independence*.

This is the first point where we fundamentally differ from PSL. To capture both dependence and independence, we change the structure in which formulas are interpreted. In Section III, we will introduce a structure $\mathcal{X} = (X, \sqsubseteq, \oplus, \odot, E)$, a *DIBI frame*, with two operations $\oplus\colon X^2 \to \mathcal{P}(X)$ and $\odot\colon X^2 \to \mathcal{P}(X)$, and a set of units $E \subseteq X$. Three remarks are in order. First, the preorder $\sqsubseteq$ makes DIBI an *intuitionistic* logic. There are many design trade-offs between intuitionistic and classical, but the most important consideration is that intuitionistic formulas can describe proper subsets of states (e.g., random variables), leaving the rest of the state implicit. Second, DIBI frames contain an additional monoidal operation $\odot$ for interpreting $\fatsemi$ ($\oplus$ will be used in interpreting $*$). Third, as the completeness of BI for its simple PCM models is an open problem [13], our models are examples of a broader notion of BI model with non-deterministic operations (following [14, 15]). These models subsume partial resource monoids, and enable our completeness proof of DIBI. While the conditions that DIBI frames must satisfy are somewhat cryptic at first sight, they can be naturally understood as axioms defining monoidal operations in a partial, non-deterministic setting. E.g., we will require:

($\oplus$ Comm.)	$z \in x \oplus y \to z \in y \oplus x$;
($\oplus$ Assoc.)	$w \in t \oplus z \wedge t \in x \oplus y \to \exists s (s \in y \oplus z \wedge w \in x \oplus s)$;
($\odot$ Unit Exist.$_{\mathrm{L}}$)	$\exists e \in E.\ (x \in e \odot x)$

where unbound variables are universally quantified. Crucially, the operation $\odot$ need *not* be commutative: this operation interprets the dependence conjunction $\fatsemi$, where commutativity is undesirable. In a DIBI frame, $*$ and $\fatsemi$ are interpreted as:

$x \models P * Q$	iff exists x', y, z s.t. $x \sqsupseteq x' \in y \oplus z$, $y \models P$, and $z \models Q$
$x \models P \fatsemi Q$	iff exists y, z s.t. $x \in y \odot z$, $y \models P$, and $z \models Q$

In DIBI, $*$ has a similar reading as in PSL: it states that two parts of a distribution can be combined because they are *independent*. In contrast, the new conjunction $P \fatsemi Q$ asserts that the Q part of a distribution *may depend* on the P part. Combined with the separating conjunction $*$, the new conjunction $\fatsemi$ can express more complex dependencies: e.g. $P \fatsemi (Q * R)$ asserts that Q and R both may depend on P, and are independent conditioned on P.

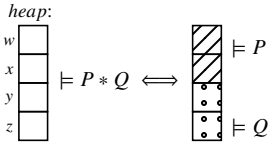
A sound and complete proof system for DIBI: To reason about DIBI validity, in Section III we also provide a Hilbert-style proof system for DIBI, and prove soundness and completeness. The proof system extends BI with rules for the new connective $\fatsemi$, e.g. $\fatsemi$ Conj, and for the interaction between $\fatsemi$ and $*$, e.g., RevEx:

$$\frac{P \vdash R \qquad Q \vdash S}{P \fatsemi Q \vdash R \fatsemi S}\ \fatsemi\,\textsc{Conj} \qquad\qquad \frac{}{(P \fatsemi Q) * (R \fatsemi S) \vdash (P * R) \fatsemi (Q * S)}\ \textsc{RevEx}$$

RevEx—*reverse-exchange*—captures the fundamental interaction between the two conjunctions. Computations $T = P \fatsemi Q$ and $U = R \fatsemi S$ are built from dependent components, yet T and U are independent and hence can be combined with $*$. We can then infer that the building blocks of T and U must also be pair-wise independent and can be combined, yielding formulas $P * R$ and $Q * S$. These can then be combined with $\fatsemi$ as they retain the dependency of the original building blocks.

Models and applications of DIBI: Separation logics are based on a concrete BI model over program states, together with a choice of atomic assertions. Before explaining the models of DIBI, we recall two prior models of BI.

In the heap model, states are heaps: partial maps from memory addresses to values. Atomic assertions of the form $x \mapsto v$ indicate that the location to which x points has value v. Then, $x \mapsto v * y \mapsto u$ states that x points to v and y points to u, and x and y do not alias—they must point to different locations. In general, $P * Q$ holds when a heap can be split into two subheaps with disjoint domains, satisfying P and Q respectively.

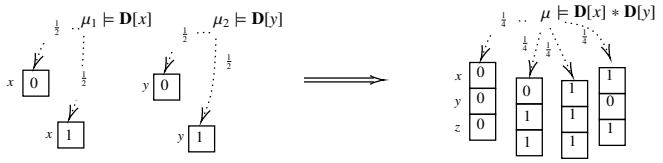


In PSL, states are distributions over program memories, basic assertions $\mathbf{D}[x]$ indicate that x is a random variable,

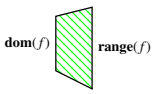
and $P * Q$ states that a distribution μ can be factored into two *independent* distributions μ_1 and μ_2 satisfying P and Q, respectively. Consider the following simple program:

$$x \xleftarrow{\$} \mathbf{B}_{1/2}; y \xleftarrow{\$} \mathbf{B}_{1/2}; z \leftarrow x \vee y \tag{1}$$

Here, x and y are Boolean variables storing the result of two fair coin flips and z stores the result of $x \vee y$. The output distribution μ is a distribution over a memory with variables x, y and z (depicted below on the right). In μ, the variables x and y are independent and $\mathbf{D}[x] * \mathbf{D}[y]$ holds, since the marginal distribution of μ is a product of μ_1 and μ_2, which satisfy $\mathbf{D}[x]$ and $\mathbf{D}[y]$ respectively:

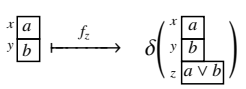


In Section IV, we develop two concrete models for DIBI: one based on probability distributions, and one based on relations. Here we outline the probabilistic model, as it generalizes the model of PSL. Let **Val** be a finite set of values and S a finite set of memory locations. We use $\mathbf{Mem}[S]$ to denote functions $S \to \mathbf{Val}$, representing program memories. The states in the DIBI probabilistic model, over which the formulas will be interpreted, are Markov kernels on program memories. More precisely, given sets of memory locations $S \subseteq U$, these are functions $f\colon \mathbf{Mem}[S] \to \mathcal{D}(\mathbf{Mem}[U])$ that preserve their input. Regular distributions can be lifted to Markov kernels: the distribution $\mu\colon \mathcal{D}(\mathbf{Mem}[U])$ corresponds to the kernel $f_\mu\colon \mathbf{Mem}[\emptyset] \to \mathcal{D}(\mathbf{Mem}[U])$ that assigns μ to the only element in $\mathbf{Mem}[\emptyset]$. We depict input-preserving Markov kernels as trapezoids, where the smaller side represents the domain and the larger side the range; our basic assertions will track $\mathbf{dom}(f)$ and $\mathbf{range}(f)$, justifying this simplistic depiction.



Separating and dependent conjunction will be interpreted via $\oplus$ and $\odot$ on Markov kernels. Intuitively, $\oplus$ is a parallel composition that takes union on both domains and ranges, whereas $\odot$ composes the kernels using Kleisli composition.

To demonstrate, recall the simple program (1). In the output distribution μ, z depends on x and y since z stores $x \vee y$, and x and y are independent. In our setting, this dependency structure can be seen when decomposing $f_\mu = (f_{\mu_1} \oplus f_{\mu_2}) \odot f_z$, where kernel $f_z\colon \mathbf{Mem}[\{x,y\}] \to \mathcal{D}(\mathbf{Mem}[\{x,y,z\}])$ captures how the value of z depends on the values of $\{x,y\}$:



$\delta\colon X \to \mathcal{D}(X)$ is the Dirac distribution $\delta(v)(w) = 1$ if $v = w$, 0 otherwise.

We can then prove:

$$f_{\mu_1} \oplus f_{\mu_2} \models P_{x*y} \quad \text{and} \quad f_z \models Q_z \quad \text{implies} \quad f_\mu \models P_{x*y} \fatsemi Q_z \tag{2}$$

When analyzing composition of Markov kernels, the domains and ranges provide key information: the domain determines which variables a kernel may depend on, and the range determines which variables a kernel describes. Accordingly, we use basic assertions of the form $(A \triangleright [B])$, where A and B are sets of memory locations. A Markov kernel $f\colon \mathbf{Mem}[S] \to \mathcal{D}(\mathbf{Mem}[T])$ satisfies $(A \triangleright [B])$ if there exists a $f' \sqsubseteq f$ with $\mathbf{dom}(f') = A$ and $\mathbf{range}(f') \supseteq B$ (we will define $f' \sqsubseteq f$ formally later and for now read it as f extends f'). For instance, the kernel f_z above satisfies $(\{x,y\} \triangleright [x,y])$, $(\{x,y\} \triangleright [x,y,z])$, and $(\{x,y\} \triangleright [\emptyset])$. One choice for P_{x*y} and Q_z in (2) can be: $P_{x*y} = (\emptyset \triangleright [x]) * (\emptyset \triangleright [y])$ and $Q_z = (\{x,y\} \triangleright [x,y,z])$)

Formalizing conditional independence: The reader might wonder how to use such simple atomic propositions, which only talk about the domain/range of a kernel and do not describe numeric probabilities, to assert conditional independence. The key insight is that conditional independence can be formulated using sequential ($\odot$) and parallel ($\oplus$) composition of kernels. In Section V, we show that given $\mu \in \mathcal{D}(\mathbf{Mem}[\mathsf{Var}])$, for any $X, Y, Z \subseteq \mathsf{Var}$, the satisfaction of

$$f_\mu \models (\emptyset \triangleright [Z]) \fatsemi (Z \triangleright [X]) * (Z \triangleright [Y]) \tag{3}$$

captures conditional independence of X, Y given Z in μ.

Moreover, the formula in (3) smoothly generalizes to other models. In the relational model of DIBI—obtained by switching the distribution monad to the powerset monad—the exact same formula encodes *join dependency*, a notion of conditional independence from the databases and relational algebra literature. More generally, we also show that the semi-graphoid axioms of Pearl and Paz [10] are valid in these two models, and two of the axioms can be derived in the DIBI proof system.

III. The Logic DIBI

A. Syntax and semantics

The syntax of DIBI extends the logic of bunched implications (BI) [11] with a non-commutative conjunctive connective $\fatsemi$ and its associated implications. Let $\mathcal{AP}$ be a set of propositional atoms. The set of DIBI formulas, $\mathrm{Form}_{\mathrm{DIBI}}$, is generated by the following grammar:

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

DIBI is interpreted on DIBI frames, which extend BI frames.

Definition III.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 1.

Intuitively, 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

($\oplus$ Down-Closed)	$z \in x \oplus y \wedge x \sqsupseteq x' \wedge y \sqsupseteq y'$	$\rightarrow$	$\exists z'(z \sqsupseteq z' \wedge z' \in x' \oplus y')$;
($\odot$ Up-Closed)	$z \in x \odot y \wedge z' \sqsupseteq z$	$\rightarrow$	$\exists x', y'(x' \sqsupseteq x \wedge y' \sqsupseteq y \wedge z' \in x' \odot y')$
($\oplus$ Commutativity)	$z \in x \oplus y$	$\rightarrow$	$z \in y \oplus x$;
($\oplus$ Associativity)	$w \in t \oplus z \wedge t \in x \oplus y$	$\rightarrow$	$\exists s(s \in y \oplus z \wedge w \in x \oplus s)$;
($\oplus$ Unit Existence)	$\exists e \in E(x \in e \oplus x)$;		
($\oplus$ Unit Coherence)	$e \in E \wedge x \in y \oplus e$	$\rightarrow$	$x \sqsupseteq y$;
($\odot$ Associativity)	$\exists t(w \in t \odot z \wedge t \in x \odot y)$	$\leftrightarrow$	$\exists s(s \in y \odot z \wedge w \in x \odot s)$;
($\odot$ Unit Existence$_{\mathrm{L}}$)	$\exists e \in E(x \in e \odot x)$;		
($\odot$ Unit Existence$_{\mathrm{R}}$)	$\exists e \in E(x \in x \odot e)$;		
($\odot$ Coherence$_{\mathrm{R}}$)	$e \in E \wedge x \in y \odot e$	$\rightarrow$	$x \sqsupseteq y$;
(Unit Closure)	$e \in E \wedge e' \sqsupseteq e$	$\rightarrow$	$e' \in E$;
(Reverse Exchange)	$x \in y \oplus z \wedge y \in y_1 \odot y_2 \wedge z \in z_1 \odot z_2$	$\rightarrow$	$\exists u, v(u \in y_1 \oplus z_1 \wedge v \in y_2 \oplus z_2 \wedge x \in u \odot v)$.

Fig. 1: DIBI frame requirements (with outermost universal quantification omitted for readability).

operations. The binary operators return a *set* of states instead of a single state, and thus can be either deterministic (at most one state returned) or non-deterministic, either partial (empty set returned) or total. The operators in the concrete models below will be deterministic, but the proof of completeness relies on the frame's admission of non-deterministic models, as is standard for bunched logics [14].

The frame conditions define properties that must hold for all models of DIBI. Most of these properties can be viewed as generalizations of familiar algebraic properties to non-deterministic operations, suitably interacting with the preorder. The "Closed" properties give coherence conditions between the order and the composition operators. It is known that having the Associativity frame condition together with either the Up- or Down-Closed property for an operator is sufficient to obtain the soundness of associativity for the conjunction associated with the operator [16, 14]. The choices of Closed 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), while dependence should be inherited by superstates (the source of dependence will still be present in any extensions). Having $\odot$ non-commutative also splits the $\odot$ analogues of $\oplus$ axioms into pairs of axioms, although we note that we exclude the left version of ($\odot$ Coherence) for reasons we explain in Section III-B. Finally, the (Reverse Exchange) condition defines the interaction between $\oplus$ and $\odot$.

We will give a Kripke-style semantics for DIBI, much like the semantics for BI [17]. Given a DIBI frame, the semantics defines which states in the frame *satisfy* each formula. Since the definition is inductive on formulas, we must specify which states satisfy the atomic propositions.

Definition III.2 (Valuation and model)**.** A *persistent valuation* is an assignment $\mathcal{V}\colon \mathcal{AP} \to \mathcal{P}(X)$ of atomic propositions to subsets of states of a DIBI frame satisfying: 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 $\mathcal{V}$.

Since DIBI is an intuitionistic logic, persistence is necessary for soundness. We can now give a semantics to DIBI formulas in a DIBI model.

Definition III.3 (DIBI Satisfaction and Validity)**.** Satisfaction at a state x in a model is inductively defined by the clauses in Figure 2. 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, $\mathcal{X} \models_{\mathcal{V}} P$ implies $\mathcal{X} \models_{\mathcal{V}} Q$.

Where the context is clear, we omit the subscript $\mathcal{V}$ on the satisfaction relation. With the semantics in Figure 2, persistence on propositional atoms extends to all formulas:

Lemma III.1 (Persistence Lemma)**.** *For all* $P \in \mathrm{Form}_{\mathrm{DIBI}}$*, if* $x \models P$ *and* $y \sqsupseteq x$ *then* $y \models P$.

The reader may note the difference between the semantic clauses for $⨾$ and $*$, and $-\!\!*$ and $-\!\!\circ$: the satisfaction of the Up-Closed (Down-Closed) frame axiom for $\odot$ ($\oplus$) leads to the persistence and thus the soundness of the simpler clause for $⨾$ ($-\!\!*$) [16]. Without the other Closed property, we must use a satisfaction clause which accounts for the order, as in BI.

B. Proof system

A Hilbert-style proof system for DIBI is given in Figure 3. This calculus extends a system for BI with additional rules governing the new connectives $⨾$, $-\!\!\circ$ and $\circ\!\!-$: in Section III-C we will prove this calculus is sound and complete. We briefly comment on two important details in this proof system.

Reverse exchange: The proof system of DIBI shares many similarities with Concurrent Kleene Bunched Logic (CKBI) [14], which also extends BI with a non-commutative conjunction. Inspired by concurrent Kleene algebra (CKA) [18], CKBI supports the following exchange axiom, derived from CKA's exchange law:

$$(P * R) ⨾ (Q * S) \vdash_{\mathrm{CKBI}} (P ⨾ Q) * (R ⨾ S)$$

In models of CKBI, $*$ describes interleaving concurrent composition, while $⨾$ describes sequential composition. The exchange rule states that the process on the left has *fewer* behaviors than the process on the right—e.g., $P ⨾ Q$ allows fewer behaviors than $P * Q$, so $P ⨾ Q \vdash_{\mathrm{CKBI}} P * Q$ 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

$$
\begin{array}{llll}
x & \models_{\mathcal{V}} & \top & \text{always} \\
x & \models_{\mathcal{V}} & I & \text{iff } x \in E \\
x & \models_{\mathcal{V}} & P \wedge Q & \text{iff } x \models_{\mathcal{V}} P \text{ and } x \models_{\mathcal{V}} Q \\
x & \models_{\mathcal{V}} & P \vee Q & \text{iff } x \models_{\mathcal{V}} P \text{ or } x \models_{\mathcal{V}} Q \\
x & \models_{\mathcal{V}} & P \to Q & \text{iff for all } y \sqsupseteq x,\ y \models_{\mathcal{V}} P \text{ implies } y \models_{\mathcal{V}} Q \\
x & \models_{\mathcal{V}} & P * Q & \text{iff there exist } x', y, z \text{ s.t. } x \sqsupseteq x' \in y \oplus z,\ y \models_{\mathcal{V}} P \text{ and } z \models_{\mathcal{V}} Q \\
x & \models_{\mathcal{V}} & P \mathbin{⨟} Q & \text{iff there exist } y, z \text{ s.t. } x \in y \odot z,\ y \models_{\mathcal{V}} P \text{ and } z \models_{\mathcal{V}} Q \\
x & \models_{\mathcal{V}} & P \mathrel{-\!\!*} Q & \text{iff for all } y, z \text{ s.t. } z \in x \oplus y\text{: } y \models_{\mathcal{V}} P \text{ implies } z \models_{\mathcal{V}} Q \\
x & \models_{\mathcal{V}} & P \mathrel{\multimap} Q & \text{iff for all } x', y, z \text{ s.t. } x' \sqsupseteq x \text{ and } z \in x' \odot y\text{: } y \models_{\mathcal{V}} P \text{ implies } z \models_{\mathcal{V}} Q \\
x & \models_{\mathcal{V}} & P \mathrel{\circ\!\!-} Q & \text{iff for all } x', y, z \text{ s.t. } x' \sqsupseteq x \text{ and } z \in y \odot x'\text{: } y \models_{\mathcal{V}} P \text{ implies } z \models_{\mathcal{V}} Q
\end{array}
\qquad
\begin{array}{llll}
x & \models_{\mathcal{V}} & \bot & \text{never} \\
x & \models_{\mathcal{V}} & \mathrm{p} & \text{iff } x \in \mathcal{V}(\mathrm{p})
\end{array}
$$

Fig. 2: Satisfaction for DIBI

$$
\frac{}{P \vdash P}\ \textsc{Ax}
\qquad
\frac{P \vdash Q \qquad P \vdash R}{P \vdash Q \wedge R}\ \wedge 1
\qquad
\frac{P \vdash Q \to R \qquad P \vdash Q}{P \vdash R}\ \textsc{MP}
\qquad
\frac{P \vdash Q \mathrel{\multimap} R \qquad S \vdash Q}{P \mathbin{⨟} S \vdash R}\ \mathrel{\multimap}\textsc{MP}
$$

$$
\frac{P \vdash R \qquad Q \vdash S}{P * Q \vdash R * S}\ *\textsc{-Conj}
\qquad
\frac{P \vdash R \qquad Q \vdash S}{P \mathbin{⨟} Q \vdash R \mathbin{⨟} S}\ ⨟\textsc{-Conj}
$$

$$
\frac{}{P \vdash \top}\ \top
\qquad
\frac{Q \vdash R}{P \wedge Q \vdash R}\ \wedge 2
\qquad
\frac{P * Q \vdash R}{P \vdash Q \mathrel{-\!\!*} R}\ \mathrel{-\!\!*}
\qquad
\frac{P \mathbin{⨟} Q \vdash R}{Q \vdash P \mathrel{\circ\!\!-} R}\ \mathrel{\circ\!\!-}
$$

$$
\frac{}{P * Q \vdash Q * P}\ *\textsc{-Comm}
\qquad
\frac{}{P \dashv\vdash P \mathbin{⨟} I}\ ⨟\textsc{-Right Unit}
$$

$$
\frac{}{\bot \vdash P}\ \bot
\qquad
\frac{P \vdash Q_1 \wedge Q_2}{P \vdash Q_i}\ \wedge 3/\wedge 4
\qquad
\frac{P \vdash Q \mathrel{-\!\!*} R \qquad S \vdash Q}{P * S \vdash R}\ \mathrel{-\!\!*}\textsc{MP}
\qquad
\frac{P \vdash Q \mathrel{\circ\!\!-} R \qquad S \vdash Q}{S \mathbin{⨟} P \vdash R}\ \mathrel{\circ\!\!-}\textsc{MP}
$$

$$
\frac{}{(P * Q) * R \dashv\vdash P * (Q * R)}\ *\textsc{-Assoc}
\qquad
\frac{}{(P \mathbin{⨟} Q) \mathbin{⨟} R \dashv\vdash P \mathbin{⨟} (Q \mathbin{⨟} R)}\ ⨟\textsc{-Assoc}
$$

$$
\frac{P \vdash R \qquad Q \vdash R}{P \vee Q \vdash R}\ \vee 1
\qquad
\frac{P \vdash Q_i}{P \vdash Q_1 \vee Q_2}\ \vee 2
\qquad
\frac{P \wedge Q \vdash R}{P \vdash Q \to R}\ \to
\qquad
\frac{P \mathbin{⨟} Q \vdash R}{P \vdash Q \mathrel{\multimap} R}\ \mathrel{\multimap}
$$

$$
\frac{}{P \dashv\vdash P * I}\ *\textsc{-Unit}
\qquad
\frac{}{P \vdash I \mathbin{⨟} P}\ ⨟\textsc{-Left Unit}
$$

$$
\frac{}{(P \mathbin{⨟} Q) * (R \mathbin{⨟} S) \vdash (P * R) \mathbin{⨟} (Q * S)}\ \textsc{RevEx}
$$

Fig. 3: Hilbert system for DIBI

is *safe* to combine implies that the process on the right is also safe. $P * Q$ is now *stronger* than $P \mathbin{⨟} Q$, and $P * Q \vdash P \mathbin{⨟} Q$ is derivable (see the extended version [19]).

Left unit: While $⨟$ 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 for $\odot$-Coherence$_{\mathrm{L}}$ in our definition of DIBI frames. This difference can also be seen in our proof rules: while $⨟$-Right Unit gives entailment in both directions, $⨟$-Left Unit only shows entailment in one direction—there is no axiom stating $I \mathbin{⨟} P \vdash P$.

We make this relaxation to support our intended models, which we will see in Section IV. In a nutshell, states in our models are Kleisli arrows that *preserve their input through to their output*—intuitively, in conditional distributions, the variables that have we conditioned on will remain fixed. Our models take $\odot$ to be Kleisli composition, which exhibits an important asymmetry for such arrows: f can always be recovered from $f \odot g$, but not from $g \odot f$. As a result, the set of all arrows naturally serves as the set of right units, but these arrows cannot all serve as left units.

C. Soundness and Completeness of DIBI

A methodology for proving the soundness and completeness of bunched logics is given by Docherty [14], inspired by the duality-theoretic approach to modal logic [20]. 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. Omitted details can be found in [19].

Definition III.4 (DIBI Algebra)**.** A *DIBI algebra* is an algebra $\mathbb{A} = (A, \wedge, \vee, \to, \top, \bot, *, \mathrel{-\!\!*}, ⨟, \mathrel{\multimap}, \mathrel{\circ\!\!-}, I)$ such that, for all $a, b, c, d \in A$:

- $(A, \wedge, \vee, \to, \top, \bot)$ is a Heyting algebra;
- $(A, *, I)$ is a commutative monoid;
- $(A, ⨟, I)$ is a *weak monoid*: $⨟$ is an associative operation with right unit I and $a \le I \mathbin{⨟} a$;
- $a * b \le c$ iff $a \le b \mathrel{-\!\!*} c$;
- $a \mathbin{⨟} b \le c$ iff $a \le b \mathrel{\multimap} c$ iff $b \le a \mathrel{\circ\!\!-} c$;
- $(a \mathbin{⨟} b) * (c \mathbin{⨟} d) \le (a * c) \mathbin{⨟} (b * d)$.

An *algebraic interpretation* of DIBI is specified by an assignment $[\![-]\!] : \mathcal{AP} \to A$. 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 III.2. *$P \vdash Q$ is derivable iff $[\![P]\!] \le [\![Q]\!]$ for all algebraic interpretations $[\![-]\!]$.*

We now connect these algebras to DIBI frames. A *filter* on a bounded distributive lattice $\mathbb{A}$ is a non-empty set $F \subseteq A$ such that, for all $x, y \in A$, (1) $x \in F$ and $x \le y$ implies $y \in F$; and (2) $x, y \in F$ implies $x \wedge y \in F$. It is a *proper* filter if it additionally satisfies (3) $\bot \notin F$, and a *prime* filter if it also

satisfies (4) $x \vee y \in F$ implies $x \in F$ or $y \in F$. We denote the set of prime filters of $\mathbb{A}$ by $\mathbb{PF}_{\mathbb{A}}$.

Definition III.5 (Prime Filter Frame)**.** Given a DIBI algebra $\mathbb{A}$, the *prime filter frame* of $\mathbb{A}$ is defined as $Pr(\mathbb{A}) = (\mathbb{PF}_{\mathbb{A}}, \subseteq, \oplus_{\mathbb{A}}, \odot_{\mathbb{A}}, E_{\mathbb{A}})$, where $F \oplus_{\mathbb{A}} G := \{H \in \mathbb{PF}_{\mathbb{A}} \mid \forall a \in F, b \in G(a * b \in H)\}$, $F \odot_{\mathbb{A}} G := \{H \in \mathbb{PF}_{\mathbb{A}} \mid \forall a \in F, b \in G(a \fatsemi b \in H)\}$ and $E_{\mathbb{A}} := \{F \in \mathbb{PF}_{\mathbb{A}} \mid I \in F\}$.

Proposition III.3. *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 III.6 (Complex Algebra)**.** Given a DIBI frame $\mathcal{X} = (X, \sqsubseteq, \oplus, \odot, E)$, the *complex algebra* of $\mathcal{X}$ is defined to be $Com(\mathcal{X}) = (\mathcal{P}_{\sqsubseteq}(X), \cap, \cup, \Rightarrow_{\mathcal{X}}, X, \emptyset, \bullet_{\mathcal{X}}, \mathrel{-\!\!\bullet}_{\mathcal{X}}, \mathrel{\triangleright}_{\mathcal{X}}, \mathrel{\rightarrowtriangle}_{\mathcal{X}}, \mathrel{\triangleleft\!\!-}_{\mathcal{X}}, E)$:

$$
\begin{array}{rl}
\mathcal{P}_{\sqsubseteq}(X) & = \{A \subseteq X \mid \text{ if } a \in A \text{ and } a \sqsubseteq b \text{ then } b \in A\} \\
A \Rightarrow_{\mathcal{X}} B & = \{a \mid \text{for all } b, \text{ if } b \sqsupseteq a \text{ and } b \in A \text{ then } b \in B\} \\
A \bullet_{\mathcal{X}} B & = \{x \mid \text{there exist } x', a, b \text{ s.t } x \sqsupseteq x' \in a \oplus b, a \in A \text{ and } b \in B\} \\
A \mathrel{-\!\!\bullet}_{\mathcal{X}} B & = \{x \mid \text{for all } a, b, \text{ if } b \in x \oplus a \text{ and } a \in A \text{ then } b \in B\} \\
A \mathrel{\triangleright}_{\mathcal{X}} B & = \{x \mid \text{there exist } a, b \text{ s.t } x \in a \odot b, a \in A \text{ and } b \in B\} \\
A \mathrel{\rightarrowtriangle}_{\mathcal{X}} B & = \{x \mid \text{for all } x', a, b, \text{ if } x \sqsubseteq x', b \in x' \odot a \text{ and } a \in A \text{ then } b \in B\} \\
A \mathrel{\triangleleft\!\!-}_{\mathcal{X}} B & = \{x \mid \text{for all } x', a, b, \text{ if } x \sqsubseteq x', b \in a \odot x' \text{ and } a \in A \text{ then } b \in B\}.
\end{array}
$$

Proposition III.4. *For any DIBI frame $\mathcal{X}$, the complex algebra $Com(\mathcal{X})$ is a DIBI algebra.*

The following main result facilitates transference of soundness and completeness.

Theorem III.5 (Representation of DIBI algebras)**.** *Every DIBI algebra is isomorphic to a subalgebra of a complex algebra: given a DIBI algebra $\mathbb{A}$, the map $\theta_{\mathbb{A}} : \mathbb{A} \to Com(Pr(\mathbb{A}))$ defined by $\theta_{\mathbb{A}}(a) = \{F \in \mathbb{PF}_{\mathbb{A}} \mid a \in F\}$ 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 [14, 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 III.6 (Soundness and Completeness)**.** *$P \vdash Q$ is derivable iff $P \models Q$.*

IV. Models of DIBI

In this section, we introduce two concrete models of DIBI to facilitate logical reasoning about (in)dependence in probability distributions and relational databases. In both models the operations $\odot$ and $\oplus$ will be *deterministic* partial functions; we write $h = f \bullet g$ instead of $\{h\} = f \bullet g$, for $\bullet \in \{\odot, \oplus\}$. We start with some preliminaries on memories and distributions.

A. Memories, distributions, and Markov kernels

Operations on Memories: Let **Val** be a fixed set of values (e.g., the Booleans), S be a set of variable names, and let $\mathbf{Mem}[S]$ denote the set of functions of type $m : S \to \mathbf{Val}$. We call such functions *memories* because we can think of m as assigning a value to each variable in S; we will refer to S as the *domain* of m. The only element in $\mathbf{Mem}[\emptyset]$ is the empty memory, which we write as $\langle\rangle$.

We need two operations on memories. First, a memory m with domain S can be projected to a memory m^T with domain T if $T \subseteq S$, defined as $m^T(x) = m(x)$ for all $x \in T$. Second, two memories can be combined if they agree on the intersection of their domains: given memories $m_1 \in \mathbf{Mem}[S]$, $m_2 \in \mathbf{Mem}[T]$ such that $m_1^{S \cap T} = m_2^{S \cap T}$, we define $m_1 \otimes m_2 : S \cup T \to \mathbf{Val}$ by

$$
m_1 \otimes m_2(x) := \begin{cases} m_1(x) & \text{if } x \in S \\ m_2(x) & \text{if } x \in T \end{cases} \tag{4}
$$

Probability distributions and Markov kernels: We use the distribution monad to model distributions over memories. Given a set X, let $\mathcal{D}(X)$ denote the set of finite distributions over X, i.e., the set containing all finite support functions $\mu : X \to [0, 1]$ satisfying $\sum_{x \in X} \mu(x) = 1$. This operation on sets can be lifted to functions $f : X \to Y$, resulting in a map of distributions $\mathcal{D}(f) : \mathcal{D}(X) \to \mathcal{D}(Y)$ given by $\mathcal{D}(f)(\mu)(y) := \sum_{f(x)=y} \mu(x)$ (intuitively, $\mathcal{D}(f)$ takes the sum of the probabilities of all elements in the pre-image of y). These operations turn $\mathcal{D}$ into a functor on sets and, further, $\mathcal{D}$ is also a *monad* [21, 22].

Definition IV.1 (Distribution Monad)**.** Define $\mathsf{unit} : X \to \mathcal{D}(X)$ as $\mathsf{unit}_X(x) := \delta_x$ 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 $\mathsf{bind} : \mathcal{D}(X) \to (X \to \mathcal{D}(Y)) \to \mathcal{D}(Y)$ by $\mathsf{bind}(\mu)(f)(y) := \sum_{p \in \mathcal{D}(Y)} \mathcal{D}(f)(\mu)(p) \cdot p(y)$.

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 $\langle \mathcal{D}, \mathsf{unit}, \mathsf{bind} \rangle$ is a monad:

$$
\begin{array}{c}
\mathsf{bind}(\mathsf{unit}(x))(f) = f(x), \qquad \mathsf{bind}(\mu)(\mathsf{unit}) = \mu, \\
\mathsf{bind}(\mathsf{bind}(\mu)(f))(g) = \mathsf{bind}(\mu)(\lambda x.\mathsf{bind}(f(x))(g)).
\end{array} \tag{5}
$$

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

The monad $\mathcal{D}$ gives rise to the *Kleisli category* of $\mathcal{D}$, denoted $\mathcal{K}\ell(\mathcal{D})$, with sets as objects and arrows of the form $f : X \to \mathcal{D}(Y)$, also known as *Markov kernels* [23]. Arrow composition in $\mathcal{K}\ell(\mathcal{D})$ is defined using bind: given $f : X \to \mathcal{D}(Y)$, $g : Y \to \mathcal{D}(Z)$, the composition $f \odot g : X \to \mathcal{D}(Z)$ is:

$$
(f \odot g)(x) := \mathsf{bind}(f(x))(g) \tag{6}
$$

Markov kernels generalize distributions: we can lift a distribution $\mu : \mathcal{D}(X)$ to the kernel $f_\mu : 1 \to \mathcal{D}(X)$ assigning μ to the single element of 1. Kernels can also encode conditional distributions, which play a key role in conditional independence.

Example IV.1. Consider the program p in Figure 4a, 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

$z \xleftarrow{\$} \mathbf{B}_{1/2}$;
if z **then**
  $x \xleftarrow{\$} \mathbf{B}_{1/4}$;
  $y \xleftarrow{\$} \mathbf{B}_{1/4}$;
else
  $x \xleftarrow{\$} \mathbf{B}_{1/2}$;
  $y \xleftarrow{\$} \mathbf{B}_{1/2}$

(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 $z = 1$

Fig. 4: From probabilistic programs to kernels

bias $1/4$ twice, and store the results in x and y. This program produces a distribution μ, shown in Figure 4b.

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 4c). 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 (Figure 4d).

To connect μ_0 and μ_1 to the original distribution μ, we package μ_0 and μ_1 into a Markov kernel $k\colon \mathbf{Mem}[z] \to \mathcal{D}(\mathbf{Mem}[\{x, y, z\}])$ given by $k(i)(d) = \mu_i(d^{\{x,y\}})$. Then, the relation between the conditional and original distributions is $f_\mu = f_{\mu_z} \odot k$, where μ_z is the projection of μ on $\{z\}$.

Finite distributions of memories over U, denoted $\mathcal{D}(\mathbf{Mem}[U])$, will play a central role in our models. We will refer to maps $f\colon \mathbf{Mem}[S] \to \mathcal{D}(\mathbf{Mem}[U])$ as *(Markov) kernels*, and define $\mathbf{dom}(f) = S$ and $\mathbf{range}(f) = U$.

We can marginalize/project kernels to a smaller range.

Definition IV.2 (Marginalizing kernels)**.** For a Markov kernel $f\colon \mathbf{Mem}[S] \to \mathcal{D}(\mathbf{Mem}[U])$ and $V \subseteq U$, the *marginalization of f by V* is the map $\pi_V f\colon \mathbf{Mem}[S] \to \mathcal{D}(\mathbf{Mem}[V])$: $(\pi_V f)(d)(r) := \sum_{m \in \mathbf{Mem}[U \setminus V]} f(d)(r \otimes m)$ for $d \in \mathbf{Mem}[S], r \in \mathbf{Mem}[V]$; undefined terms do not contribute to the sum.

We say a kernel $f\colon \mathbf{Mem}[S] \to \mathcal{D}(\mathbf{Mem}[U])$ *preserves its input to its output* if $S \subseteq U$ and $\pi_S f = \mathsf{unit}_{\mathbf{Mem}[S]}$. Intuitively, such kernels are suitable for encoding conditional distributions: once a variable has been conditioned on, its value should not change. We can compose these kernels in two ways.

Definition IV.3 (Composing Markov kernels on memories)**.** Given $f\colon \mathbf{Mem}[S] \to \mathcal{D}(\mathbf{Mem}[T])$ and $g\colon \mathbf{Mem}[U] \to \mathcal{D}(\mathbf{Mem}[V])$ that preserve their inputs, we define their *parallel composition*, whenever $S \cap U = T \cap V$, as the map $f \oplus g\colon \mathbf{Mem}[S \cup U] \to \mathcal{D}(\mathbf{Mem}[T \cup V])$ given by

$$(f \oplus g)(d)(m) := f(d^S)(m^T) \cdot g(d^U)(m^V).$$

If $T = U$, the *sequential composition* $f \odot g\colon \mathbf{Mem}[S] \to \mathcal{D}(\mathbf{Mem}[V])$ is just Kleisli composition (Eq. (6)).

B. A concrete probabilistic model of DIBI

We now have all the ingredients to define a first concrete model: states are Markov kernels that preserve their input; $\oplus$ (resp. $\odot$) will be parallel (resp. sequential) composition. The use of $\oplus$ to model independence generalizes the approach in Barthe et al. [9]. Combining both compositions—sequential and parallel—enables capturing conditional independence.

Definition IV.4 (Probabilistic frame)**.** We define the frame $(M^D, \sqsubseteq, \oplus, \odot, M^D)$ as follows:

- Let M^D consist of Markov kernels that preserve their input to their output;
- $\oplus$, $\odot$ are parallel and sequential composition of kernels;
- Given $f, g \in M^D$, $f \sqsubseteq g$ if there exist $R \subseteq \mathbf{Val}$, $h \in M^D$ such that $g = (f \oplus \mathsf{unit}_{\mathbf{Mem}[R]}) \odot h$.

We make two remarks. First, $f \sqsubseteq g$ holds when g can be obtained from extending f: compose f in parallel with $\mathsf{unit}_{\mathbf{Mem}[R]}$, then extend the range via composition with h. We can recover f from g by marginalizing g to $\mathbf{range}(f) \cup R$, then ignoring the R portion. Second, the definition of $f \odot g$ on M^D can be simplified. Given $f\colon \mathbf{Mem}[S] \to \mathcal{D}(\mathbf{Mem}[T])$ and $g\colon \mathbf{Mem}[T] \to \mathcal{D}(\mathbf{Mem}[V])$, Eq. (6) yields the formula:

$$(f \odot g)(d)(m) := \sum_{m' \in \mathbf{Mem}[T]} f(d)(m') \cdot g(m')(m).$$

Since $f, g \in M^D$ preserve input to output, this reduces to

$$(f \odot g)(d)(m) = f(d)(m^T) \cdot g(m^T)(m^V). \tag{7}$$

We show that our probabilistic frame is indeed a DIBI frame.

Theorem IV.1. *$(M^D, \sqsubseteq, \oplus, \odot, M^D)$ is a DIBI frame.*

Proof sketch. First, we show that M^D is closed under $\oplus$ and $\odot$, and $\sqsubseteq$ is transitive and reflexive. The frame axioms are mostly straightforward, but some conditions rely on a property of our model 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. For example:

($\oplus$ Unit Coherence): The unit set in this frame is the entire state space M^D: we must show that for any $f_1, f_2 \in M^D$, if $f_1 \oplus f_2$ is defined, then $f_1 \sqsubseteq f_1 \oplus f_2$:

$$\begin{aligned} f_1 \oplus f_2 &= (f_1 \odot \mathsf{unit}_{\mathbf{range}(f_1)}) \oplus (\mathsf{unit}_{\mathbf{dom}(f_2)} \odot f_2) \\ &= (f_1 \oplus \mathsf{unit}_{\mathbf{dom}(f_2)}) \odot (\mathsf{unit}_{\mathbf{range}(f_1)} \oplus f_2) \quad (\text{Exch. Eq.}) \\ &= (f_1 \oplus \mathsf{unit}_{\mathbf{dom}(f_2)}) \odot (f_2 \oplus \mathsf{unit}_{\mathbf{range}(f_1)}) \quad (\oplus \text{ Comm.}) \end{aligned}$$

We present the complete proof in [19]. □

Example IV.2 (Kernel decomposition)**.** Recall the distribution μ on $\mathbf{Mem}[\{x, y, z\}]$ from Example IV.1. Let $k_x \colon \mathbf{Mem}[z] \to \mathcal{D}(\mathbf{Mem}[\{x, z\}])$ encode the conditional distribution of x given z, and let $k_y \colon \mathbf{Mem}[z] \to \mathcal{D}(\mathbf{Mem}[\{y, z\}])$ encode the conditional distribution of y given z. Explicitly, for $v = x$ or y,

$$k_v(z = 0)(v = 1, z = 0) = 1/2 \quad k_v(z = 0)(v = 0, z = 0) = 1/2$$
$$k_v(z = 1)(v = 1, z = 1) = 1/4 \quad k_v(z = 1)(v = 0, z = 1) = 3/4.$$

Since k_x, k_y include z in their range, $k_x \oplus k_y$ is defined. A small calculation shows that $k_x \oplus k_y = k$, where $k \colon \mathbf{Mem}[z] \to \mathcal{D}(\mathbf{Mem}[\{x, y, z\}])$ is the conditional distribution of (x, y, z) given z. This decomposition shows that x and y are independent conditioned on z (we shall formally prove this later in Section V-A).

C. Relations, join dependency, and powerset kernels

We developed the probabilistic model in the previous section using operations from the distribution monad $\mathcal{D}$. Instantiating our definitions with operations from other monads gives rise to other interesting models of DIBI. In this section, we develop a *relational* model based on the powerset monad $\mathcal{P}$, and show how our logic can be used to reason about join dependency properties of tables from database theory. Before we present our relational model, we introduce some notations and basic definitions on relations.

Tables are often viewed as *relations*—sets of tuples where each component of the tuple corresponds to an *attribute*. Formally, a relation R over a set of attributes S is a set of *tuples* indexed by S. Each tuple maps an attribute in S to a value in **Val**, and hence can be seen as a memory in $\mathbf{Mem}[S]$, as defined in Section IV-A. The projection and $\otimes$ operations on $\mathbf{Mem}[S]$ from Equation (4) can be lifted to relations.

Definition IV.5 (Projection and Join)**.** The *projection* of a relation R over attributes X to $Y \subseteq X$ is given by $R^Y := \{r^Y \mid r \in R\}$. The *natural join* of relations R_1 and R_2 over attributes X_1 and X_2, respectively, is the relation $R_1 \bowtie R_2 := \{m_1 \otimes m_2 \mid m_1 \in R_1 \text{ and } m_2 \in R_2\}$ over attributes $X_1 \cup X_2$.

Since tables can often be very large, finding compact representations for them is useful. These representations can leverage additional structure common in real-world databases; for instance, the value of one attribute might determine the value of another, a so-called *functional dependency*. Other dependency structures can enable a large relation to be factored as a combination of smaller ones. A classical example is on *join dependency*, a relational analogue of conditional independence.

Definition IV.6 (Join dependency [24, 25])**.** A relation R over attribute set $X_1 \cup X_2$ satisfies the *join dependency* $X_1 \bowtie X_2$ if $R = (R^{X_1}) \bowtie (R^{X_2})$.

Example IV.3 (Decomposition)**.** Consider the relation R in Figure 5, with three attributes: Researcher, Field, and Conference. R contains triple (a, b, c) if and only if researcher a works in field b and attends conference c. If we know that researchers in the same field all have a shared set of conferences they attend, then we can recover R by joining two relations: one associating researchers to their fields, and another associating fields to conferences. As shown below, R satisfies the join dependency $\{\mathsf{Researcher}, \mathsf{Field}\} \bowtie \{\mathsf{Conference}, \mathsf{Field}\}$. While the factored form is only a bit smaller (12 entries instead of 15), savings can be significant for larger relations.

Powerset monad and kernels: Much like how we decomposed distributions as Markov kernels—Kleisli arrows for the distribution monad—we will decompose relations using Kleisli arrows for the powerset monad, $\mathcal{K}\ell(\mathcal{P})$.

Definition IV.7 (Powerset monad)**.** Let $\mathcal{P}$ be the endofunctor $\mathbf{Set} \to \mathbf{Set}$ mapping every set to the set of its subsets $\mathcal{P}(X) = \{U \mid U \subseteq X\}$. We define $\mathsf{unit}_X \colon X \to \mathcal{P}(X)$ mapping each $x \in X$ to the singleton $\{x\}$, and $\mathsf{bind} \colon \mathcal{P}(X) \to (X \to \mathcal{P}(Y)) \to \mathcal{P}(Y)$ by $\mathsf{bind}(U)(f) := \cup\{y \mid \exists x \in U. f(x) = y\}$.

The triple $\langle \mathcal{P}, \mathsf{unit}, \mathsf{bind} \rangle$ forms a monad, and obeys the laws in Equation (5). We overload the use of unit and bind as it will be clear from the context which monad, powerset or distribution, we are considering. The Kleisli category $\mathcal{K}\ell(\mathcal{P})$ is defined analogously as for $\mathcal{D}$, with sets as objects and arrows $X \to \mathcal{P}(Y)$, and composition given as in Equation (6).

Like before, we consider maps $\mathbf{Mem}[S] \to \mathcal{P}(\mathbf{Mem}[T])$, which we call *powerset kernels* in analogy to Markov kernels, or simply kernels when the monad is clear from the context. Powerset kernels can also be projected to a smaller range.

Definition IV.8 (Marginalization)**.** Suppose that $T \subseteq U$. A map f of type $\mathbf{Mem}[S] \to \mathcal{P}(\mathbf{Mem}[U])$ can be marginalized to $\pi_T f \colon \mathbf{Mem}[S] \to \mathcal{P}(\mathbf{Mem}[T])$ by defining: $(\pi_T f)(s) := f(s)^T$

We need two composition operations on powerset kernels. We say that powerset kernel $f \colon \mathbf{Mem}[S] \to \mathcal{P}(\mathbf{Mem}[S \cup T])$ *preserves input to output* if $\pi_S f = \mathsf{unit}_{\mathbf{Mem}[S]}$.

Definition IV.9 (Composition of powerset kernels)**.** Given kernels $f \colon \mathbf{Mem}[S] \to \mathcal{P}(\mathbf{Mem}[T])$ and $g \colon \mathbf{Mem}[U] \to \mathcal{P}(\mathbf{Mem}[V])$ that preserve input to output, we define their *parallel composition* whenever $T \cap V = S \cap U$ as the map $f \oplus g \colon \mathbf{Mem}[S \cup U] \to \mathcal{P}(\mathbf{Mem}[T \cup V])$ given by $(f \oplus g)(d) := f(d^S) \bowtie g(d^U)$. Whenever $T = U$ we define the *sequential composition* $f \odot g \colon \mathbf{Mem}[S] \to \mathcal{P}(\mathbf{Mem}[V])$ using Kleisli composition. Explicitly: $(f \odot g)(s) = \{v \mid u \in f(s) \text{ and } v \in g(u)\}$.

D. A concrete relational model of DIBI

We can now define the second concrete model of DIBI: states will be powerset kernels, and we will use the parallel and sequential composition in a construction similar to M^D.

Definition IV.10 (Relational frame)**.** We define the frame $(M^P, \sqsubseteq, \oplus, \odot, M^P)$ as follows:

- M^P consists of powerset kernels preserving input to output;
- $\oplus$, $\odot$ are parallel and sequential composition of powerset kernels;
- Given $f, g \in M^P$, $f \sqsubseteq g$ if there exist $R \subseteq \mathbf{Val}$, $h \in M^P$ such that $g = (f \oplus \mathsf{unit}_{\mathbf{Mem}[R]}) \odot h$.

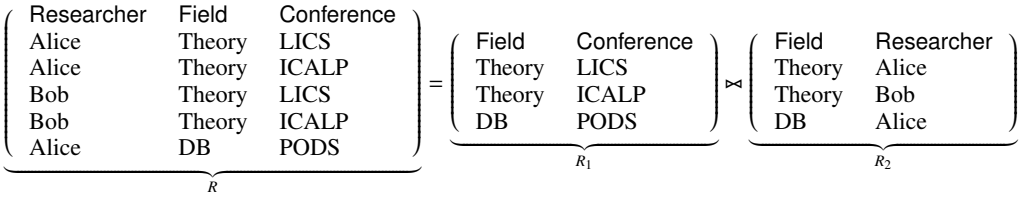


Fig. 5: Factoring a relation

Like in M^D, $f \sqsubseteq g$ iff g can be obtained from f by adding attributes that are preserved from domain to range, and then mapping tuples in the range to relations over a larger set of attributes. We can recover f from g by marginalizing to $\mathbf{range}(f) \cup R$, and then ignoring the attributes in R.

M^P is also a DIBI frame.

Theorem IV.2. *$(M^P, \sqsubseteq, \oplus, \odot, M^P)$ is a DIBI frame.*

Proof sketch.. The proof follows Theorem IV.1 quite closely, since M^P also satisfies Exchange equality. We present the full proof in [19]. □

V. Application: Modeling Conditional and Join Dependencies

In our concrete models, distributions and relations can be factored into simpler parts. Here, we show how DIBI formulas capture conditional independence and join dependency.

A. Conditional independence

Conditional independence (CI) is a well-studied notion in probability theory and statistics [26]. 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 irrelevant to Y—observing one gives no further information about the other.

Before defining CI, we introduce some notations. Let $\mu \in \mathcal{D}(\mathbf{Mem}[\mathsf{Var}])$ be a distribution. For any subset $S \subseteq \mathsf{Var}$ and assignment $s \in \mathbf{Mem}[S]$, we write:

$$\mu(S = s) := \sum_{m \in \mathbf{Mem}[\mathsf{Var}]} \mu(s \otimes m).$$

Terms with undefined $s \otimes m$ contribute zero to the sum. We can now define conditional probabilities:

$$\mu(S = s \mid S' = s') := \frac{\mu(S = s, S' = s')}{\mu(S' = s')},$$

where $\mu(S = s, S' = s') := \mu(S \cup S' = s \otimes s')$. Intuitively, this ratio is the probability of $S = s$ given $S' = s'$, and it is only defined when the denominator is non-zero and s, s' are consistent (i.e., $s \otimes s'$ is defined). CI can be defined as follows.

Definition V.1 (Conditional independence)**.** Let $X, Y, Z \subseteq \mathsf{Var}$. 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 $X \perp\!\!\!\perp Y$.

Example V.1. We give two simple examples of CI.

Chocolate and Nobel laureates: Researchers found a strong positive correlation between a nation's per capita Nobel laureates number and chocolate consumption. But the correlation may be due to other factors, e.g., a nation's economic status. A simple check is to see if the two are conditionally independent fixing the third factor.

Algorithmic fairness: To prevent algorithms from discriminating based on sensitive features (e.g., race and gender), researchers formalized notions of fairness using conditional independence [8]. For instance, let A be the sensitive features, Y be the target label, and $\widehat{Y}$ be the algorithm's prediction for Y. Considering the joint distribution of $(A, Y, \widehat{Y})$, an algorithm satisfies *equalized odds* if $\widehat{Y} \perp\!\!\!\perp A \mid Y$; *calibration* if $Y \perp\!\!\!\perp A \mid \widehat{Y}$.

We will define a DIBI formula P such that a distribution μ satisfies $X \perp\!\!\!\perp Y \mid Z$ if and only if its lifted kernel $f_\mu := \langle\rangle \mapsto f$ satisfies P. For this, we will need a basic atomic proposition which describes the domain and range of kernels.

Definition V.2 (Basic atomic proposition)**.** For sets of variables $A, B \subseteq \mathsf{Var}$, a basic atomic proposition has the form $(A \triangleright [B])$. We give the following semantics to these formulas:

$$f \models (A \triangleright [B]) \text{ iff there exists } f' \sqsubseteq f \text{ such that } \mathbf{dom}(f') = A \text{ and } \mathbf{range}(f') \supseteq B.$$

For example, $f\colon \mathbf{Mem}[y] \to \mathcal{D}(\mathbf{Mem}[y, z])$ 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.

Theorem V.1. *Given distribution $\mu \in \mathcal{D}(\mathbf{Mem}[\mathsf{Var}])$, then for any $X, Y, Z \subseteq \mathsf{Var}$,*

$$f_\mu \models (\emptyset \triangleright [Z]) \fatsemi (Z \triangleright [X]) * (Z \triangleright [Y]) \tag{8}$$

if and only if $X \perp\!\!\!\perp Y \mid Z$ and $X \cap Y \subseteq Z$ are both satisfied.

The restriction $X \cap Y \subseteq Z$ is harmless: when $X \perp\!\!\!\perp Y \mid Z$ but $X \cap Y \not\subseteq Z$, $X \cap Y$ must be deterministic given Z (see [19]), 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 8. Then by [19, Lemma A.38], there exist f, g, and h in M^D with $f \odot (g \oplus h) \sqsubseteq f_\mu$, where $f\colon \mathbf{Mem}[\emptyset] \to \mathcal{D}(\mathbf{Mem}[Z])$, $g\colon \mathbf{Mem}[Z] \to \mathcal{D}(\mathbf{Mem}[Z \cup X])$, and $h\colon \mathbf{Mem}[Z] \to$

$\mathcal{D}(\mathbf{Mem}[Z \cup Y])$; 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 (h resp.) encodes the conditional distributions of X (Y resp.) given Z, and $g \oplus h$ encodes the conditional distribution of (X, Y) given Z. Hence, 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 (a) $X \perp\!\!\!\perp Y \mid Z$ holds in μ and (b) $X \cap Y \subseteq Z$. 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 $[Z]^{\circledS}_{\circledS}([X] * [Y])$. Hence, persistence shows that f_μ also satisfies $[Z]^{\circledS}_{\circledS}([X] * [Y])$.

See [19, Theorem A.11] for details. □

B. Join dependency

Recall that a relation R over attributes $X \cup Y$ satisfies the *Join Dependency* (JD) $X \bowtie Y$ if $R = R^X \bowtie R^Y$. As we illustrated through the Researcher-Field-Conference example in Section IV, join dependencies can enable a relation to be represented more compactly. By interpreting the atomic propositions in the relational model, JD is captured by the same formula we used for CI.

Theorem V.2. *Let $R \in \mathcal{P}(\mathbf{Mem}[\mathsf{Var}])$ and X, Y be sets of attributes such that $X \cup Y = \mathsf{Var}$. The lifted relation $f_R = \langle\rangle \mapsto R$ satisfies $f_R \models [X \cap Y]^{\circledS}_{\circledS}([X] * [Y])$ iff R satisfies the join dependency $X \bowtie Y$.*

JD is a special case of Embedded Multivalued Dependency (EMVD), where the relation R may have more attributes than $X \cup Y$. It is straightforward to encode EMVD in our logic, but for simplicity we stick with JD.

Proof sketch. For the forward direction, by [19, Lemma A.38], there exist f, g, and $h \in M^P$ such that $f \colon \mathbf{Mem}[\emptyset] \to \mathcal{P}(\mathbf{Mem}[X \cap Y])$, $g \colon \mathbf{Mem}[X \cap Y] \to \mathcal{P}(\mathbf{Mem}[X])$, $h \colon \mathbf{Mem}[X \cap Y] \to \mathcal{P}(\mathbf{Mem}[Y])$, and $f \odot (g \oplus h) \sqsubseteq f_R$. Since by assumption $X \cup Y = \mathsf{Var}$, we must have $f \odot (g \oplus h) = f_R$.

Unfolding $\oplus$ and $\odot$ and using the fact that $\mathbf{range}(f) = \mathbf{dom}(g) = \mathbf{dom}(h)$, we can show:

$$f \odot (g \oplus h)(\langle\rangle) = \{u \bowtie (v_1 \bowtie v_2) \mid u \in f(\langle\rangle), v_1 \in g(u), v_2 \in h(u)\}.$$

Since $\bowtie$ is commutative, associative and idempotent, we have:

$$\begin{aligned} f \odot (g \oplus h)(\langle\rangle) &= \{(u \bowtie v_1) \bowtie (u \bowtie v_2) \mid u \in f(\langle\rangle), v_1 \in g(u), v_2 \in h(u)\} \\ &= f \odot g(\langle\rangle) \bowtie f \odot h(\langle\rangle). \end{aligned}$$

We can also convert the parallel composition of g, h into sequential composition by padding to make the respective domain and range match: $f \odot (g \oplus h) = f \odot g \odot (\mathsf{unit}_X \oplus h) = f \odot h \odot (\mathsf{unit}_Y \oplus g)$. Hence $f \odot g = \pi_X f_R$ and $f \odot h = \pi_Y f_R$, which implies $f \odot g(\langle\rangle) = R^X$ and $f \odot h(\langle\rangle) = R^Y$. Thus:

$$R = f \odot (g \oplus h)(\langle\rangle) = f \odot g(\langle\rangle) \bowtie f \odot h(\langle\rangle) = R^X \bowtie R^Y,$$

so R satisfies the join dependency $X \bowtie Y$. The reverse direction is analogous to Theorem V.1. See [19, Theorem A.14] for details. □

C. Proving and validating the semi-graphoid axioms

Conditional independence and join dependency are closely related in our models. Indeed, there is a long line of research on generalizing these properties to other independence-like notions, and identifying suitable axioms. *Graphoids* are perhaps the most well-known approach [10]; Dawid [27] has a similar notion called *separoids*.

Definition V.3 (Graphoids and semi-graphoids)**.** Suppose that $I(X, Z, Y)$ is a ternary relation on subsets of Var (i.e., $X, Z, Y \subseteq \mathsf{Var}$). Then 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*.

Intuitively, $I(X, Z, Y)$ states that knowing Z renders X irrelevant to Y. If we fix a distribution over $\mu \in \mathcal{D}(\mathbf{Mem}[\mathsf{Var}])$, 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. Likewise, if we fix a relation $R \in \mathcal{P}(\mathbf{Mem}[\mathsf{Var}])$, then the triples of sets of attributes such that R satisfies an Embedded Multivalue Dependency (EMVD) forms a semi-graphoid [24, 28].

Previously, we showed that the DIBI formula $[Z]^{\circledS}_{\circledS}([X] * [Y])$ asserts conditional independence of X and Y given Z in M^D, and join dependency $X \bowtie Y$ in $M^P$when $Z = X \cap Y$. Here, we show that the semi-graphoid axioms can be naturally translated into valid formulas in our concrete models.

Theorem V.3. *Given a model M, define $I(X, Z, Y)$ iff $M \models [Z]^{\circledS}_{\circledS}([X] * [Y])$. Then,* Symmetry*,* Decomposition*,* Weak Union*, and* Contraction *are valid when M is the probabilistic or the relational model. Furthermore,* Symmetry *is derivable in the proof system, and* Decomposition *is derivable given the following axiom, valid in both models:*

$$(Z \triangleright [Y \cup W]) \leftrightarrow (Z \triangleright [Y]) \wedge (Z \triangleright [W]) \qquad \text{(Split)}$$

Proof sketch. We comment on the derivable axioms. To derive Symmetry, we use the $*$-Comm proof rule to commute the

separating conjunction. The proof of Decomposition uses the axiom Split to split up $Y \cup W$, and then uses proof rules $\wedge 3$ and $\wedge 4$ to prove the two conjuncts. We show derivations ([19, Theorems A.15 and A.16]) and prove validity ([19, Theorems A.17 and A.18]). □

VI. Application: Conditional Probabilistic Separation Logic

As our final application, we design a separation logic for probabilistic programs. We work with a simplified probabilistic imperative language with assignments, sampling, sequencing, and conditionals; our goal is to show how a DIBI-based program logic could work in the simplest setting. For lack of space, we only show a few proof rules and example programs here; we defer the full presentation of the separation logic, the metatheory, and the examples to [19].

Proof rules: CPSL includes novel proof rules for randomized conditionals and inherits the frame rule from PSL [9]. Here, we show two of the rules and explain how to use them in the simple program from Eq. (1), reproduced here:

$$\textsc{Simple} := x \xleftarrow{\$} \mathbf{B}_{1/2}; y \xleftarrow{\$} \mathbf{B}_{1/2}; z \leftarrow x \vee y$$

CPSL has Hoare-style rules for sampling and assignments:

$$\textsc{Samp}\ \frac{x \notin \mathrm{FV}(d) \cup \mathrm{FV}(P)}{\vdash \{P\}\ x \xleftarrow{\$} d\ \{P \fatsemi (\mathrm{FV}(d) \triangleright [x])\}}$$

$$\textsc{Assn}\ \frac{x \notin \mathrm{FV}(e) \cup \mathrm{FV}(P)}{\vdash \{P\}\ x \leftarrow e\ \{P \fatsemi (\mathrm{FV}(e) \triangleright [x])\}}$$

Using Samp and the fact that the coin-flip distribution $\mathbf{B}_{1/2}$ has no free variables, we can infer:

$$\vdash \{\top\}\ x \xleftarrow{\$} \mathbf{B}_{1/2}\ \{(\emptyset \triangleright [x])\} \qquad \vdash \{\top\}\ y \xleftarrow{\$} \mathbf{B}_{1/2}\ \{(\emptyset \triangleright [y])\}$$

Applying a variant of the frame rule, we are able to derive:

$$\vdash \{\top\}\ x \xleftarrow{\$} \mathbf{B}_{1/2}; y \xleftarrow{\$} \mathbf{B}_{1/2}\ \{(\emptyset \triangleright [x]) * (\emptyset \triangleright [y])\}$$

Using Assn on $P = (\emptyset \triangleright [x]) * (\emptyset \triangleright [y])$ and the fact that z is not a free variable in either P or $x \vee y$:

$$\vdash \{P\}\ z \leftarrow x \vee y\ \{P \fatsemi (\{x, y\} \triangleright [z])\}$$

Putting it all together, we get the validity of triple:

$$\vdash \{\top\}\ \textsc{Simple}\ \{((\emptyset \triangleright [x]) * (\emptyset \triangleright [y])) \fatsemi (\{x, y\} \triangleright [z])\}$$

stating that z depends on x and y, which are independent.

Example programs: Figure 6 introduces two example programs. CommonCause (Figure 6a) models a distribution where two random observations share a common cause. Specifically, we consider z, x, and y to be independent random samples, and a and b to be values computed from (x, z) and (y, z), respectively. Intuitively, z, x, 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. Our program logic can establish the following judgment capturing this fact:

$$\vdash \{\top\}\ \textsc{CommonCause}\ \{(\emptyset \triangleright [z]) \fatsemi ((z \triangleright [a]) * (z \triangleright [b]))\}$$

```
z ←$ B_{1/2};
x ←$ B_{1/2};
y ←$ B_{1/2};
a ← x ∨ z;
b ← y ∨ z
```

(a) CommonCause

```
z ←$ B_{1/2};
if z then
    x ←$ B_p; y ←$ B_p
else
    x ←$ B_q; y ←$ B_q
```

(b) CondSamples

Fig. 6: Example programs

The program CondSamples (Figure 6b) is a bit more complex: it branches on a random value z, and then assigns x and y with two independent samples from $\mathbf{B}_p$ in the true branch, and $\mathbf{B}_q$ in the false branch. 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. Using our program logic's proof rules for conditionals, we are able to prove the following judgment capturing this fact:

$$\vdash \{\top\}\ \textsc{CondSamples}\ \{(\emptyset \triangleright [z]) \fatsemi ((z \triangleright [x]) * (z \triangleright [y]))\}$$

The full development of the separation logic, consisting of a proof system, a soundness theorem, along with the detailed verification of the two examples above, can be found in [19].

VII. Related Work

Bunched implications and other non-classical logics: DIBI extends the logic of bunched implications (BI) [11], and shares many similarities: DIBI can be given a Kripke-style resource semantics, just like BI, and our completeness proof relies on a general framework for proving completeness for bunched logics [14]. The non-commutative conjunction and exchange rules are inspired by the logic CKBI [14]. The main difference is that our exchange rule is reversed, due to our reading of separating conjunction $*$ as "can be combined independently", rather than "interleaved". In terms of models, the probabilistic model of DIBI can be seen as a natural extension of the probabilistic model for BI [9]—by lifting distributions to kernels, DIBI is able to reason about dependencies, while probabilistic BI is not.

There are other non-classical logics that aim to model dependencies. *Independence-friendly (IF) logic* [29] and *dependence logic* [30] introduce new quantifiers and propositional atoms to state that a variable depends, or does not depend, on another variable; these logics are each equivalent in expressivity to existential second-order logic. More recently, Durand et al. [31] proposed a probabilistic team semantics for

dependence logic, and Hannula et al. [32] 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 [33]

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 CI side, we can point to work by Geiger and Pearl [34] on graphical models; on the JD side, the survey by Fagin and Vardi [35] 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 [10] and separoids [27].

Categorical probability: The view of conditional independence as a factorization of Markov kernels has previously been explored [36, 37, 38]. Taking a different approach, Simpson [39] has recently introduced category-theoretic structures for modeling conditional independence, capturing CI and JD as well as analogues in heaps and nominal sets [40]. 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 ([41]) also consider a similar notion of separation). Simpson'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.

Program logics: Bunched logics are well-known for their role in *separation logics*, program logics for reasoning about heap-manipulating [12] and concurrent programs [42, 43]. Recently, separation logics have been developed for probabilistic programs. Our work is most related to PSL [9], where separation models probabilistic independence. Batz et al. [44] gives a different, quantitative interpretation to separation in their logic QSL, and uses it to verify expected-value properties of probabilistic heap-manipulating programs. Finally, there are more traditional program logics for probabilistic program. The Ellora logic by Barthe et al. [45] has assertions for modeling independence, but works with a classical logic. As a result, basic structural properties of independence must be introduced as axioms, rather than being built-in to the logical connectives.

VIII. Discussion and Future Directions

We have presented DIBI, a new bunched logic to reason about dependence and independence, together with its Kripke semantics and a sound and complete proof system. We provided two concrete models, based on Markov and powerset kernels, that can capture conditional independence-like notions. We see several directions for further investigation.

Generalizing the two models: The probabilistic and relational models share many similarities: both M^D and M^P are sets of Kleisli arrows, and use Kleisli composition to interpret $\odot$; both $\oplus$ operators correspond to parallel composition. Since both the distribution and powerset monads are commutative strong monads [46, 47], which come with a *double strength* bi-functor $st_{A,B} : T(A) \times T(B) \to T(A \times B)$ that seems suitable for defining $\oplus$, it is natural to consider more general models based on Kleisli arrows for such monads. Indeed, variants of conditional independence could make sense in other settings; taking the multiset monad instead of the powerset monad would lead to a model where we can assert join dependency in bags, rather than relations, and the free vector space monad could be connected to subspace models of the graphoid axioms [48].

However, it is not easy to define an operation generalizing $\oplus$ from our concrete models. The obvious choice—taking $\oplus$ as $f_1 \oplus f_2 = (f_1 \otimes f_2); st$—gives a *total* operation, but in our concrete models $\oplus$ is partial, since it is not possible to compose two arrows that disagree on their domain overlap. For instance in the probabilistic model, there is no sensible way to use $\oplus$ to combine a kernel encoding the normal distribution $\mathcal{N}(0, 1)$ on x with another encoding the Dirac distribution of $x = 1$. We do not know how to model such coherence requirements between two Kleisli arrows in a general categorical model, and we leave this investigation to future work.

Restriction and intuitionistic DIBI: A challenge in designing the program logic is ensuring that formulas in the assertion logic satisfy *restriction* (see [19]), and one may wonder if a classical version of DIBI would be more suitable for the program logic—if assertions were not required to be preserved under kernel extensions, it might be easier to show that they satisfy restriction. However, a classical logic would require assertions to specify the dependence structure of *all* variables, which can be quite complicated. Moreover, intuitionistic logics like probabilistic BI can also satisfy the restriction property, so the relevant design choice is not classical versus intuitionistic.

Rather, the more important point appears to be whether the preorder can extend a kernel's domain. If this is allowed—as in DIBI—then kernels satisfying an assertion may have extraneous variables in the domain. However, this choice also makes the dependent conjunction $P \fatsemi Q$ more flexible: Q does not need to exactly describe the domain of the second kernel, which is useful since the range of the first kernel cannot be constrained by P. This underlying tension—allowing the range to be extended, while restricting the domain—is an interesting subject for future investigation.

Acknowledgments

We thank the anonymous reviewers for thoughtful comments and feedback. This work was partially supported by the EPSRC grant (EP/S013008/1), the ERC Consolidator Grant AutoProbe (#101002697) and a Royal Society Wolfson Fellowship. This work was also partially supported by the NSF (#2023222 and #1943130) and Facebook.

References

[1] D. Kozen, "Semantics of probabilistic programs," *Journal of Computer and System Sciences*, vol. 22, no. 3, pp. 328–350, 1981. [Online]. Available: https://doi.org/10.1016/0022-0000(81)90036-2

[2] A. D. Gordon, T. Graepel, N. Rolland, C. V. Russo, J. Borgström, and J. Guiver, "Tabular: a schema-driven probabilistic programming language," in *ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages (POPL), San Diego, California*. ACM, 2014, pp. 321–334. [Online]. Available: https://doi.org/10.1145/2535838.2535850

[3] N. D. Goodman, V. K. Mansinghka, D. M. Roy, K. Bonawitz, and J. B. Tenenbaum, "Church: a language for generative models," *CoRR*, 2012. [Online]. Available: http://arxiv.org/abs/1206.3255

[4] F. D. Wood, J. van de Meent, and V. Mansinghka, "A new approach to probabilistic programming inference," in *International Conference on Artificial Intelligence and Statistics (AISTATS), Reykjavik, Iceland*, 2014, pp. 1024–1032.

[5] T. Ehrhard, M. Pagani, and C. Tasson, "Measurable cones and stable, measurable functions: a model for probabilistic higher-order programming," *Proceedings of the ACM on Programming Languages*, no. POPL, pp. 59:1–59:28, 2018. [Online]. Available: https://doi.org/10.1145/3158147

[6] S. Staton, H. Yang, F. D. Wood, C. Heunen, and O. Kammar, "Semantics for probabilistic programming: higher-order functions, continuous distributions, and soft constraints," in *IEEE Symposium on Logic in Computer Science (LICS), New York, New York*. ACM, 2016, pp. 525–534. [Online]. Available: https://doi.org/10.1145/2933575.2935313

[7] F. Dahlqvist and D. Kozen, "Semantics of higher-order probabilistic programs with conditioning," *Proceedings of the ACM on Programming Languages*, no. POPL, pp. 57:1–57:29, 2020. [Online]. Available: https://doi.org/10.1145/3371125

[8] S. Barocas, M. Hardt, and A. Narayanan, *Fairness and Machine Learning*, 2019, http://www.fairmlbook.org.

[9] G. Barthe, J. Hsu, and K. Liao, "A probabilistic separation logic," *Proceedings of the ACM on Programming Languages*, no. POPL, pp. 55:1–55:30, 2019.

[10] J. Pearl and A. Paz, *Graphoids: A graph-based logic for reasoning about relevance relations*. : University of California (Los Angeles). Computer Science Department, 1985.

[11] P. W. O'Hearn and D. J. Pym, "The logic of bunched implications," *Bulletin of Symbolic Logic*, vol. 5, pp. 215–244, 1999.

[12] P. W. O'Hearn, J. C. Reynolds, and H. Yang, "Local reasoning about programs that alter data structures," in *International Workshop on Computer Science Logic (CSL), Paris, France*, 2001, pp. 1–19. [Online]. Available: https://doi.org/10.1007/3-540-44802-0_1

[13] D. Galmiche, M. Marti, and D. Méry, "Relating labelled and label-free bunched calculi in BI logic," in *Automated Reasoning with Analytic Tableaux and Related Methods*. Springer International Publishing, 2019, pp. 130–146.

[14] S. Docherty, "Bunched logics: a uniform approach," Ph.D. dissertation, UCL (University College London), 2019.

[15] D. Galmiche and D. Larchey-Wendling, "Expressivity properties of Boolean BI through relational models," in *Foundations of Software Technology and Theoretical Computer Science (FSTTCS), Kolkata, India*. Springer, 2006, pp. 357–368.

[16] Q. Cao, S. Cuellar, and A. W. Appel, "Bringing order to the separation logic jungle," in *Asian Symposium on Programming Languages and Systems (APLAS), Suzhou, China*. Springer, 2017, pp. 190–211.

[17] D. J. Pym, P. W. O'Hearn, and H. Yang, "Possible worlds and resources: the semantics of BI," *Theoretical Computer Science*, vol. 315, no. 1, pp. 257–305, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0304397503006248

[18] T. Hoare, B. Möller, G. Struth, and I. Wehrman, "Concurrent Kleene algebra and its foundations," *The Journal of Logic and Algebraic Programming*, vol. 80, no. 6, pp. 266–296, 2011.

[19] J. Bao, S. Docherty, J. Hsu, and A. Silva, "A Bunched Logic for Conditional Independence," in *IEEE Symposium on Logic in Computer Science (LICS), Rome, Italy*, 2021. [Online]. Available: https://arxiv.org/abs/2008.09231

[20] R. Goldblatt, "Varieties of complex algebras," *Annals of Pure and Applied Logic*, vol. 44, no. 3, pp. 173–242, 1989. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0168007289900328

[21] M. Giry, "A categorical approach to probability theory," *Categorical aspects of topology and analysis*, pp. 68–85, 1982.

[22] E. Moggi, "Notions of computation and monads," *Information and Computation*, vol. 93, no. 1, pp. 55–92, 1991, selections from 1989 IEEE Symposium on Logic in Computer Science. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0890540191900524

[23] P. Panangaden, *Labelled Markov Processes*. Imperial College Press, 2009.

[24] R. Fagin, "Multivalued dependencies and a new normal form for relational databases," *ACM Trans. Database Syst.*, vol. 2, no. 3, pp. 262–278, 1977. [Online]. Available: https://doi.org/10.1145/320557.320571

[25] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of databases*. : Addison-Wesley Reading, 1995, vol. 8.

[26] A. P. Dawid, "Conditional independence in statistical theory," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 41, no. 1, pp. 1–15, 1979.

[27] ——, "Separoids: A mathematical framework for conditional independence and irrelevance," *Annals of Mathe-*

matics and Artificial Intelligence, vol. 32, no. 1-4, pp. 335–372, 2001.

[28] J. Pearl and T. Verma, "The logic of representing dependencies by directed graphs," in *AAAI Conference on Artificial Intelligence, Seattle, WA*, 1987, pp. 374–379. [Online]. Available: http://www.aaai.org/Library/AAAI/1987/aaai87-067.php

[29] J. Hintikka and G. Sandu, "Informational independence as a semantical phenomenon," in *Logic, Methodology and Philosophy of Science VIII*, ser. Studies in Logic and the Foundations of Mathematics. Elsevier, 1989, vol. 126, pp. 571–589. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0049237X08700661

[30] J. Väänänen, *Dependence Logic: A New Approach to Independence Friendly Logic*, ser. London Mathematical Society Student Texts. Cambridge University Press, 2007.

[31] A. Durand, M. Hannula, J. Kontinen, A. Meier, and J. Virtema, "Probabilistic team semantics," in *International Symposium on Foundations of Information and Knowledge Systems (FoIKS), Budapest, Hungary*, ser. Lecture Notes in Computer Science, vol. 10833. Springer, 2018, pp. 186–206. [Online]. Available: https://doi.org/10.1007/978-3-319-90050-6_11

[32] M. Hannula, J. Kontinen, J. Van den Bussche, and J. Virtema, "Descriptive complexity of real computation and probabilistic independence logic," in *IEEE Symposium on Logic in Computer Science (LICS), Saarbrücken, Germany*, 2020, pp. 550–563.

[33] S. Abramsky and J. A. Väänänen, "From IF to BI," *Synthese*, vol. 167, no. 2, pp. 207–230, 2009. [Online]. Available: https://doi.org/10.1007/s11229-008-9415-6

[34] D. Geiger and J. Pearl, "Logical and algorithmic properties of conditional independence and graphical models," *The Annals of Statistics*, vol. 21, no. 4, pp. 2001–2021, 1993. [Online]. Available: http://www.jstor.org/stable/2242326

[35] R. Fagin and M. Y. Vardi, "The theory of data dependencies - an overview," in *International Colloquium on Automata, Languages and Programming (ICALP), Antwerp, Belgium*, 1984, pp. 1–22. [Online]. Available: https://doi.org/10.1007/3-540-13345-3_1

[36] B. Jacobs and F. Zanasi, "A formal semantics of influence in bayesian reasoning," in *International Symposium on Mathematical Foundations of Computer Science (MFCS), Aalborg, Denmark*, ser. Leibniz International Proceedings in Informatics, vol. 83. Schloss Dagstuhl–Leibniz Center for Informatics, 2017, pp. 21:1–21:14. [Online]. Available: https://doi.org/10.4230/LIPIcs.MFCS.2017.21

[37] K. Cho and B. Jacobs, "Disintegration and bayesian inversion via string diagrams," *Math. Struct. Comput. Sci.*, vol. 29, no. 7, pp. 938–971, 2019. [Online]. Available: https://doi.org/10.1017/S0960129518000488

[38] T. Fritz, "A synthetic approach to markov kernels, conditional independence and theorems on sufficient statistics," *Advances in Mathematics*, vol. 370, pp. 107–239, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0001870820302656

[39] A. Simpson, "Category-theoretic structure for independence and conditional independence," in *Conference on the Mathematical Foundations of Programming Semantics (MFPS), Halifax, Canada*, 2018, pp. 281–297. [Online]. Available: https://doi.org/10.1016/j.entcs.2018.03.028

[40] A. M. Pitts, *Nominal Sets: Names and Symmetry in Computer Science*, ser. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2013.

[41] J. Brotherston and C. Calcagno, "Classical BI: A logic for reasoning about dualising resources," in *ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages (POPL), Savannah, Georgia*. ACM, 2009, pp. 328—-339. [Online]. Available: https://doi.org/10.1145/1480881.1480923

[42] P. W. O'Hearn, "Resources, concurrency, and local reasoning," *Theoretical Computer Science*, vol. 375, no. 1, pp. 271–307, 2007, festschrift for John C. Reynolds's 70th birthday. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S030439750600925X

[43] S. Brookes, "A semantics for concurrent separation logic," *Theoretical Computer Science*, vol. 375, no. 1–3, pp. 227–270, 2007. [Online]. Available: https://doi.org/10.1016/j.tcs.2006.12.034

[44] K. Batz, B. L. Kaminski, J. Katoen, C. Matheja, and T. Noll, "Quantitative separation logic: a logic for reasoning about probabilistic pointer programs," *Proceedings of the ACM on Programming Languages*, no. POPL, pp. 34:1–34:29, 2019. [Online]. Available: https://doi.org/10.1145/3290347

[45] G. Barthe, T. Espitau, M. Gaboardi, B. Grégoire, J. Hsu, and P. Strub, "An assertion-based program logic for probabilistic programs," in *European Symposium on Programming (ESOP), Thessaloniki, Greece*, 2018, pp. 117–144. [Online]. Available: https://doi.org/10.1007/978-3-319-89884-1_5

[46] B. Jacobs, "Semantics of weakening and contraction," *Annals of pure and applied logic*, vol. 69, no. 1, pp. 73–106, 1994.

[47] A. Kock, "Monads on symmetric monoidal closed categories," *Archiv der Mathematik*, vol. 21, no. 1, pp. 1–10, 1970.

[48] S. Lauritzen, *Graphical Models*. Clarendon Press, 1996.