A Type Theory for Probabilistic and Bayesian Reasoning

Citation for the original published paper (version of record):

Jacobs, B., Adams, R. (2018)
A Type Theory for Probabilistic and Bayesian Reasoning
Leibniz International Proceedings in Informatics (LIPIcs), 69(1): 1-34
http://dx.doi.org/10.4230/lipics.types.2015.1

A Type Theory for Probabilistic and Bayesian Reasoning*

Robin Adams[1] **and Bart Jacobs**[2]

1 **Institutt for Informatikk,**
Universitetet i Bergen, Norway
`robin.adams@uib.no`

2 **Institute for Computing and Information Sciences,**
Radboud University, the Netherlands
`bart@cs.ru.nl`

Abstract

This paper introduces a novel type theory and logic for probabilistic reasoning. Its logic is quantitative, with fuzzy predicates. It includes normalisation and conditioning of states. This conditioning uses a key aspect that distinguishes our probabilistic type theory from quantum type theory, namely the bijective correspondence between predicates and side-effect free actions (called instrument, or assert, maps). The paper shows how suitable computation rules can be derived from this predicate-action correspondence, and uses these rules for calculating conditional probabilities in two well-known examples of Bayesian reasoning in (graphical) models. Our type theory may thus form the basis for a mechanisation of Bayesian inference.

1998 ACM Subject Classification F.4.1 [Mathematical Logic and Formal Languages]: Mathematical Logic — Lambda calculus and related systems; G.3 [Probability and Statistics]: Probabilistic algorithms; F.3.1 [Logics and Meanings of Programs]: Specifying and Verifying and Reasoning about Programs

Keywords and phrases Probability theory, type theory, effect module, Bayesian reasoning

Digital Object Identifier 10.4230/LIPIcs.TYPES.2015.<article-no>

1 Introduction

A probabilistic program is understood (semantically) as a stochastic process. A key feature of probabilistic programs as studied in the 1980s and 1990s is the presence of probabilistic choice, for instance in the form of a weighted sum $x +_r y$, where the number $r \in [0, 1]$ determines the ratio of the contributions of x and y to the result. This can be expressed explicitly as a convex sum $r \cdot x + (1 - r) \cdot y$. Some of the relevant sources are [12, 13], and [11], and [15], and also [17] for the combination of probability and non-determinism. In the language of category theory, a probabilistic program is a map in the Kleisli category of the distribution monad $\mathcal{D}$ (in the discrete case) or of the Giry monad $\mathcal{G}$ (in the continuous case).

In recent years, with the establishement of Bayesian machine learning as an important area of computer science, the meaning of probabilistic programming shifted towards conditional inference. The key feature is no longer probabilistic choice, but normalisation of distributions (states), see *e.g.* [3]. Interestingly, this can be done in basically the same underlying models, where a program still produces a distribution — discrete or continuous — over its output.

* This work was supported by ERC Advanced Grant QCLS: Quantum Computation, Logic and Security.

This paper contributes to this latest line of work by formulating a novel type theory for probabilistic and Bayesian reasoning. We list the key features of our type theory.

- It includes a logic, which is quantitative in nature. This means that its predicates are best understood as 'fuzzy' predicates, taking values in the unit interval $[0, 1]$ of probabilities, instead of in the two-element set $\{0, 1\}$ of Booleans.
- As a result, the predicates of this logic do not form Boolean algebras, but effect modules (see *e.g.* [8]). The double negation rule does hold, but the sum $\ovee$ is a partial operation. Moreover, there is a scalar multiplication $s \cdot p$, for a scalar s and a predicate p, which produces a scaled version of the predicate p.
- This logic is a special case of a more general quantum type theory [1]. What we describe here is the probabilistic subcase of this quantum type theory, which is characterised by a bijective correspondence between predicates and side-effect free assert maps (see below for details).
- The type theory includes normalisation (and also probabilistic choice). Abstractly, normalisation means that each non-zero 'substate' in the type theory can be turned into a proper state (like in [9]). This involves, for instance, turning a *sub*distribution $\sum_i r_i x_i$, where the probabilities $r_i \in [0, 1]$ satisfy $0 < r \leq 1$ for $r \stackrel{\text{def}}{=} \sum_i r_i$, into a proper distribution $\sum_i \frac{r_i}{r} x_i$ — where, by construction, $\sum_i \frac{r_i}{r} = 1$.
- The type theory also includes conditioning, via the combination of assert maps and normalisation (from the previous two points). Hence, we can calculate conditional probabilities inside the type theory, via appropriate (derived) computation rules. In contrast, in the language of [3], probabilistic (graphical) models can be formulated, but actual computations are done in the underlying mathematical models. Since these computation are done inside our calculus, our type theory can form the basis for mechanisation.

The type theory that we present is based on a new categorical foundation for quantum logic, called effectus theory, see [8, 9, 4, 5][1]. This theory involves a basic duality between states and effects (predicates), which is implicitly also present in our type theory. A subclass of 'commutative' effectuses can be defined, forming models for probabilistic computation and logic. Our type theory corresponds to these commutative effectuses, and will thus be called **COMET**, as abbreviation of COMmutative Effectus Theory. This **COMET** can be seen as an internal language for commutative effectuses.

A key feature of quantum theory is that observations have a side-effect: measuring a system disturbs it at the quantum level. In order to perform such measurements, each quantum predicate comes with an associated 'measurement' instrument operation which acts on the underlying space. Probabilistic theories also have such instruments . . . but they are side-effect free!

The idea that predicates come with an associated action is familiar in mathematics. For instance, in a Hilbert space $\mathcal{H}$, a closed subspace $P \subseteq \mathcal{H}$ (a predicate) can equivalently be described as a linear idempotent operator $p\colon \mathcal{H} \to \mathcal{H}$ (an action) that has P has image. We sketch how these predicate-action correspondences also exist in the models that underly our type theory.

First, in the category **Sets** of sets and functions, a predicate p on a set X can be identified with a subset of X, but also with a 'characteristic' map $p\colon X \to 1 + 1$, where $1 + 1 = 2$ is the two-element set. We prefer the latter view. Such a predicate corresponds bijectively to a

[1] A general introduction to effectus theory [6] will soon be available.

'side-effect free' instrument $\mathsf{instr}_p \colon X \to X + X$, namely to:

$$\mathsf{instr}_p(x) \;=\; \begin{cases} \mathsf{inl}\,(x) & \text{if } p(x) = 1 \\ \mathsf{inr}\,(x) & \text{if } p(x) = 0 \end{cases}$$

Here we write $X + X$ for the sum (coproduct), with left and right coprojections (also called injections) $\mathsf{inl}\,(\_)\,, \mathsf{inr}\,(\_) : X \to X+X$. Notice that this instrument merely makes a left-right distinction, as described by the predicate, but does not change the state x. It is called side-effect free because it satisfies $\nabla \circ \mathsf{instr}_p = \mathrm{id}$, where $\nabla = [\mathrm{id}, \mathrm{id}] \colon X + X \to X$ is the codiagonal. It easy to see that each map $f \colon X \to X + X$ with $\nabla \circ f = \mathrm{id}$ corresponds to a predicate $p \colon X \to 1 + 1$, namely to $p = (! + !) \circ f$, where $! \colon X \to 1$ is the unique map to the final (singleton, unit) set 1.

Our next example describes the same predicate-action correspondence in a probabilistic setting. It assumes familiarity with the discrete distribution monad $\mathcal{D}$ — see [8] for details, and also Subsection 4.1 — and with its Kleisli category $\mathcal{K}\ell(\mathcal{D})$. A predicate map $p \colon X \to 1+1$ in $\mathcal{K}\ell(\mathcal{D})$ is (essentially) a fuzzy predicate $p \colon X \to [0, 1]$, since $\mathcal{D}(1 + 1) = \mathcal{D}(2) \cong [0, 1]$. There is also an associated instrument map $\mathsf{instr}_p \colon X \to X + X$ in $\mathcal{K}\ell(\mathcal{D})$, given by the function $\mathsf{instr}_p \colon X \to \mathcal{D}(X + X)$ that sends an element $x \in X$ to the distribution (formal convex combination):

$$\mathsf{instr}_p(x) \;=\; p(x) \cdot \mathsf{inl}\,(x) + (1 - p(x)) \cdot \mathsf{inr}\,(x)\,.$$

This instrument makes a left-right distinction, with the weight of the distinction given by the fuzzy predicate p. Again we have $\nabla \circ \mathsf{instr}_p = \mathrm{id}$, in the Kleisli category, since the instrument map does not change the state. It is easy to see that we get a bijective correspondence.

These instrument maps $\mathsf{instr}_p \colon X \to X + X$ can in fact be simplified further into what we call assert maps. The (partial) map $\mathsf{assert}_p \colon X \to X + 1$ can be defined as $\mathsf{assert}_p = (\mathrm{id} + !) \circ \mathsf{instr}_p$. We say that such a map is side-effect free if there is an inequality $\mathsf{assert}_p \leq \mathsf{inl}\,(\_)$, for a suitable order on the homset of partial maps $X \to X + 1$. Given assert maps for p, and for its orthosupplement (negation) $p^{\perp}$, we can define the associated instrument via a partial pairing operation as $\mathsf{instr}_p = \langle\!\langle \mathsf{assert}_p, \mathsf{assert}_{p^{\perp}} \rangle\!\rangle$, see below for details.

The key aspect of a probabilistic model, in contrast to a quantum model, is that there is a bijective correspondence between:

- predicates $X \to 1 + 1$
- side-effect free instruments $X \to X + X$ — or equivalently, side-effect free assert maps $X \to X + 1$.

We shall define conditioning via normalisation after assert. More specifically, for a state $\omega \colon X$ and a predicate p on X we define the conditional state $\omega|_p = \mathsf{cond}\,(\omega, p)$ as:

$$\mathsf{cond}\,(\omega, p) \;=\; \mathsf{nrm}\,(\mathsf{assert}_p(\omega))\,,$$

where $\mathsf{nrm}\,(-)$ describes normalisation (of substates to states). This description occurs, in semantical form in [9]. Here we formalise it at a type-theoretic level and derive suitable computation rules from it that allow us to do (exact) conditional inference.

The paper is organised as follows. Section 2 provides an overview of the type theory, with some key results, without giving all the details and proofs. Section 3 takes two familiar examples of Bayesian reasoning and formalises them in our type theory **COMET**. Next, Section 4 sketches how our type theory can be interpreted in set-theoretic and probabilistic models. Subsequently, Section 5 explores the type theory in greater depth, and provides justification for the computation rules in the examples. Appendix A contains a formal presentation of the type theory **COMET**.

$$\text{(var)}\ \frac{x : A \in \Gamma}{\Gamma \vdash x : A} \quad \text{(unit)}\ \frac{}{\Gamma \vdash * : 1} \quad (\otimes)\ \frac{\Gamma \vdash s : A \qquad \Delta \vdash t : B}{\Gamma, \Delta \vdash s \otimes t : A \otimes B}$$

$$\text{(lett)}\ \frac{\Gamma \vdash s : A \otimes B \qquad \Delta, x : A, y : B \vdash t : C}{\Gamma, \Delta \vdash \mathsf{let}\ x \otimes y = s\ \mathsf{in}\ t : C}$$

$$\text{(magic)}\ \frac{\Gamma \vdash t : 0}{\Gamma \vdash\ ¡t : A} \quad \text{(inl)}\ \frac{\Gamma \vdash t : A}{\Gamma \vdash \mathsf{inl}\,(t) : A + B} \quad \text{(inr)}\ \frac{\Gamma \vdash t : B}{\Gamma \vdash \mathsf{inr}\,(t) : A + B}$$

$$\text{(case)}\ \frac{\Gamma \vdash r : A + B \qquad \Delta, x : A \vdash s : C \qquad \Delta, y : B \vdash t : C}{\Gamma, \Delta \vdash \mathsf{case}\ r\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto s \mid \mathsf{inr}\,(y) \mapsto t : C}$$

$$\text{(inlr)}\ \frac{\Gamma \vdash s : A + 1 \qquad \Gamma \vdash t : B + 1 \qquad \Gamma \vdash s \downarrow = t \uparrow : \mathbf{2}}{\Gamma \vdash «s, t» : A + B}$$

$$\text{(left)}\ \frac{\Gamma \vdash t : A + B \qquad \Gamma \vdash \mathsf{inl?}\,(t) = \top : \mathbf{2}}{\Gamma \vdash \mathsf{left}\,(t) : A} \quad \text{(instr)}\ \frac{x : A \vdash t : \mathbf{n} \qquad \Gamma \vdash s : A}{\Gamma \vdash \mathsf{instr}_{\lambda x t}(s) : n \cdot A}$$

$$(1/n)\ \frac{}{\Gamma \vdash 1/n : \mathbf{2}} \quad \text{(nrm)}\ \frac{\vdash t : A + 1 \qquad \vdash 1/n \leq t \downarrow : \mathbf{2}}{\Gamma \vdash \mathsf{nrm}\,(t) : A}$$

$$(\ovee)\ \frac{\begin{array}{cc} \Gamma \vdash s : A + 1 & \Gamma \vdash t : A + 1 \\ \Gamma \vdash b : (A + A) + 1 & \Gamma \vdash \mathsf{do}\ x \leftarrow b; \rhd_1(x) = s : A + 1 \\ \multicolumn{2}{c}{\Gamma \vdash \mathsf{do}\ x \leftarrow b; \rhd_2(x) = t : A + 1} \end{array}}{\Gamma \vdash s \ovee t : A + 1}$$

Figure 1 Typing rules for **COMET**

2 Syntax and Rules of Deduction

We present here the terms and types of **COMET**. We shall describe the system at a high level here, giving the intuition behind each construction. The complete list of the rules of deduction of **COMET** is given in Appendix A, and the properties that we use are all proved in Section 5.

2.1 Syntax

Assume we are given a set of *type constants* **C**, representing the base data types needed for each example. (These may typically include for instance **bool**, **nat** and **real**.) Then the types of **COMET** are the following.

$$\begin{aligned} \text{Type}\ \ A &::= \mathbf{C} \mid 0 \mid 1 \mid A + B \mid A \otimes B \\ \text{Term}\ \ t &::= x \mid * \mid t \otimes t \mid \mathsf{let}\ x \otimes y = t\ \mathsf{in}\ t \mid\ ¡t \mid \mathsf{inl}\,(t) \mid \mathsf{inr}\,(t) \mid \\ &\quad\ (\mathsf{case}\ t\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto t \mid \mathsf{inr}\,(x) \mapsto t) \mid «s, t» \mid \mathsf{left}\,(t) \mid \mathsf{instr}_{\lambda x t} t \mid 1/n \mid \\ &\quad\ \mathsf{nrm}\,(t) \mid s \ovee t \end{aligned}$$

We explain the intended meaning of these terms in the remaining parts of Section 2.

The variables x and y are bound within s in $\mathsf{let}\ x \otimes y = s\ \mathsf{in}\ t$. The variable x is bound within s and y within t in $\mathsf{case}\ r\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto s \mid \mathsf{inr}\,(y) \mapsto t$, and x is bound within t in

$$\mathsf{let}\ x \otimes y = r \otimes s\ \mathsf{in}\ t = t[x := r, y := s] \qquad (\beta\otimes)$$

$$\mathsf{case}\ \mathsf{inl}\,(r)\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto s \mid \mathsf{inr}\,(y) \mapsto t = s[x := r] \qquad (\beta{+}_1)$$

$$\mathsf{case}\ \mathsf{inr}\,(r)\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto s \mid \mathsf{inr}\,(y) \mapsto t = t[y := r] \qquad (\beta{+}_2)$$

$$\rhd_1(«s, t») = s \qquad (\beta\mathrm{inlr}_1)$$

$$\rhd_2(«s, t») = t \qquad (\beta\mathrm{inlr}_1)$$

$$\mathsf{inl}\,(\mathsf{left}\,(t)) = t \qquad (\beta\mathrm{left})$$

$$\mathsf{left}\,(\mathsf{inl}\,(t)) = t \qquad (\eta\mathrm{left})$$

$$\mathsf{index}\,(\mathsf{instr}_{\lambda xp}(t)) = p[x := t] \qquad (\mathrm{instr\text{-}test})$$

$$\nabla(\mathsf{instr}_{\lambda xp}(t)) = t \qquad (\nabla\text{-}\mathrm{instr})$$

$$\text{if } \nabla(t) = x \text{ then } \mathsf{instr}_{\lambda x \mathsf{index}(t)}(s) = t[x := s] \qquad (\eta\mathrm{instr})$$

$$\text{if } t : 1 \text{ then } * = t \qquad (\eta 1)$$

$$\text{if } t : A \otimes B \text{ then } \mathsf{let}\ x \otimes y = t\ \mathsf{in}\ x \otimes y = t \qquad (\eta\otimes)$$

$$\text{if } t : A + B \text{ then } \mathsf{case}\ t\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto \mathsf{inl}\,(x) \mid \mathsf{inr}\,(y) \mapsto \mathsf{inr}\,(y) = t \qquad (\eta+)$$

$$\text{if } t : A + B \text{ then } «\rhd_1(t), \rhd_2(t)» = t \qquad (\eta\mathrm{inlr})$$

$$\text{if } t \text{ is well-typed then } \mathsf{do}\ \_ \leftarrow t; \mathsf{return}\ \mathsf{nrm}\,(t) = t \qquad (\beta\mathrm{nrm})$$

$$\text{if } t = \mathsf{do}\ \_ \leftarrow t; \mathsf{return}\ \rho \text{ and } 1/n \le t, \text{ then } \rho = \mathsf{nrm}\,(t) \qquad (\eta\mathrm{nrm})$$

$$n \cdot 1/n = \top \qquad (n \cdot 1/n)$$

$$\text{if } n \cdot t = \top \text{ then } t = 1/n \qquad (\mathrm{divide})$$

$$\text{if } \mathsf{do}\ x \leftarrow b; \rhd_1(x) = s \text{ and } \mathsf{do}\ x \leftarrow b; \rhd_2(x) = t$$
$$\text{then } s \ovee t = \mathsf{do}\ x \leftarrow b; \mathsf{return}\ \nabla(x) \qquad (\ovee\text{-}\mathrm{def})$$

Figure 2 Computation rules for **COMET**

$\mathsf{instr}_{\lambda xt}(s)$. We identify terms up to α-conversion (change of bound variable). We write $t[x := s]$ for the result of substituting s for x within t, renaming bound variables to avoid variable capture. We shall write _ for a vacuous bound variable; for example, we write $\mathsf{case}\ r\ \mathsf{of}\ \mathsf{inl}\,(\_) \mapsto s \mid \mathsf{inr}\,(y) \mapsto t$ for $\mathsf{case}\ r\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto s \mid \mathsf{inr}\,(y) \mapsto t$ when y does not occur free in s.

We shall also sometimes abbreviate our terms, for example writing $\mathsf{instr}_{\mathsf{inl}}(t)$ when we should strictly write $\mathsf{instr}_{\lambda x \mathsf{inl}(x)}(t)$. Each time, the meaning should be clear from context.

The typing rules for these terms are given in Figure 1. (Note that some of these rules make use of defined expressions, which will be introduced in the sections below.)

The computation rules that these terms obey are given in Figure 2.

Figures 1 and 2 should be understood simultaneously. So the term $«s, t»$ is well-typed if and only if we can type $s : A + 1$ and $t : B + 1$ (using the rules in Figure 1), *and* derive the equation $s \downarrow = t \uparrow$ using the rules in Figure 2.

The full set of rules of deduction for the system is given in Appendix A.

2.2 Linear Type Theory

Note the form of several of the typing rules in Figure 1, including $(\otimes)$ and (lett) . These rules do not allow a variable to be duplicated; in particular, we cannot derive the judgement

$x : A \vdash x \otimes x : A \otimes A$. The *contraction* rule does not hold in our type theory — it is not the case in general that, if $\Gamma, x : A, y : B \vdash \mathcal{J}$, then $\Gamma, z : A \vdash \mathcal{J}[x := z, y := z]$. Our theory is thus similar to a *linear* type theory (see for example [2]).

The reason is that these judgements do not behave well with respect to substitution. For example, take the computation $x : \mathbf{2} \vdash x \otimes x : \mathbf{2} \otimes \mathbf{2}$. If we apply this computation to the scalar 1/2, we presumably wish the result to be $\top \otimes \top$ with probability 1/2, and $\bot \otimes \bot$ with probability 1/2. But this is not the semantics for the term $\vdash 1/2 \otimes 1/2 : \mathbf{2} \otimes \mathbf{2}$. This term assigns probability 1/4 to all four possibilities $\top \otimes \top$, $\top \otimes \bot$, $\bot \otimes \top$, $\top \otimes \top$.

2.3 States, Predicates and Scalars

A closed term $\vdash t : A$ will be called a *state* of type A, and intuitively it represents a probability distribution over the elements of A.

A *predicate* on type A is a proposition of the form $x : A \vdash p : \mathbf{2}$. These shall be the formulas of the logic of **COMET** (see Section 2.7).

A *scalar* is a term s such that $\vdash s : \mathbf{2}$. The closed terms t such that $\vdash t : \mathbf{2}$ are called *scalars*, and represent the *probabilities* or *truth values* of our system. In our intended semantics for discrete and continuous probabilities, these denote elements of the real interval $[0, 1]$.

Given a state $\vdash t : A$ and a predicate $x : A \vdash p : \mathbf{2}$, we can find the probability that p is true when measured on t; this probability is simply the scalar $p[x := t]$.

2.4 Empty Type

The typing rule for the term ¡t says that from an inhabitant $t : 0$ we can produce an inhabitant ¡t in any type A. Intuitively, this says 'If the empty type is inhabited, then every type is inhabited', which is vacuously true.

2.5 Coproducts and Copowers

Since we have the coproduct $A + B$ of two types, we can construct the disjoint union of n types $A_1 + \cdots + A_n$ in the obvious way. We write $\mathsf{in}_1^n\,()$, ..., $\mathsf{in}_n^n\,()$ for its constructors; thus, if $a : A_i$ then $\mathsf{in}_i^n\,(a) : A_1 + \cdots + A_n$. And given $t : A_1 + \cdots + A_n$, we can eliminate it as:

$$\mathsf{case}\ t\ \mathsf{of}\ \mathsf{in}_1^n\,(x_1) \mapsto t_1 \mid \cdots \mid \mathsf{in}_n^n\,(x_n) \mapsto t_n \quad .$$

We abbreviate this expression as $\mathsf{case}\ {}_{i=1}^{n}\ t\ \mathsf{of}\ \mathsf{in}_i^n\,(x_i) \mapsto t_i$.

The term $\mathsf{left}\,(t)$ is understood as follows. If we have a term $t : A + B$ and we have derived the judgement $\mathsf{inl?}\,(t) = \top$, then we know that we know that t has the form $\mathsf{inl}\,(a)$ for some term $a : A$. We denote this unique term a by $\mathsf{left}\,(t)$.

We have a similar $\mathsf{right}\,()$ consturction, but there is no need to give primitive rules for this one, as it can be defined in terms of $\mathsf{left}\,()$: $\mathsf{right}\,(t) \stackrel{\text{def}}{=} \mathsf{left}\,(\mathsf{swap}\,(t))$, where $\mathsf{swap}\,(t) \stackrel{\text{def}}{=} \mathsf{case}\ t\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto \mathsf{inr}\,(x) \mid \mathsf{inr}\,(y) \mapsto \mathsf{inl}\,(y)$.

For the special case where all the types are equal, we write $n \cdot A$ for the type $A + \cdots + A$, where there are n copies of A. In category theory, this is known as the nth *copower* of A. (We include the special cases $0 \cdot A \stackrel{\text{def}}{=} 0$ and $1 \cdot A \stackrel{\text{def}}{=} A$.)

The *codiagonal* $\nabla(t) : A$ for $t : n \cdot A$ is defined by $\nabla(t) \stackrel{\text{def}}{=} \mathsf{case}\ {}_{i=1}^{n}\ t\ \mathsf{of}\ \mathsf{in}_i^n\,(x) \mapsto x$. (In particular, whene $n = 2$ and $t : A + A$, then $\nabla(t) \stackrel{\text{def}}{=} \mathsf{case}\ t\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto x \mid \mathsf{inr}\,(x) \mapsto x$.)

We write $\mathbf{n}$ for $n \cdot 1$. We denote the canonical elements by 1, 2, ..., n: $i \stackrel{\text{def}}{=} \mathsf{in}_i^n\,(*) : \mathbf{n}$ for $1 \le i \le n$. For $t : n \cdot A$, we define $\mathsf{index}\,(t) = \mathsf{case}\ {}_{i=1}^{n} t\ \mathsf{of}\ \mathsf{in}_i^n\,(\_) \mapsto i : \mathbf{n}$.

$$\text{(order)}\ \frac{\Gamma \vdash s : A+1 \qquad \Gamma \vdash t : A+1 \qquad \Gamma \vdash b : (A+A)+1 \quad \Gamma \vdash \mathsf{do}\ x \leftarrow b; \rhd_1(x) = s : A+1 \qquad \Gamma \vdash \mathsf{do}\ x \leftarrow b; \mathsf{return}\ \nabla(x) = t : A+1}{\Gamma \vdash s \leq t : A+1}$$

Figure 3 Rule for Ordering in **COMET**

2.6 Partial Functions

A term of type A is intended to represent a *total* computation, that always terminates and returns a value of type A. We can think of a term of type $A+1$ as a *partial* computation that may return a value a of type A (by outputting $\mathsf{inl}\,(a)$) or diverge (by outputting $\mathsf{inr}\,(*)$). The judgement $s \leq t$ should be understood as: the probability that s returns $\mathsf{inl}\,(a)$ is $\leq$ the probability that t returns $\mathsf{inl}\,(a)$, for all a. The rule for this ordering relation is given in Figure 3.

We define:

- If $\Gamma \vdash t : A$ then $\Gamma \vdash \mathsf{return}\ t \stackrel{\text{def}}{=} \mathsf{inl}\,(t) : A+1$. This program converges with probability 1.
- $\Gamma \vdash \mathsf{fail} \stackrel{\text{def}}{=} \mathsf{inr}\,(*) : A+1$. This program diverges with probability 1.
- If $\Gamma \vdash s : A+1$ and $\Delta, x : A \vdash t : B+1$ then $\Gamma, \Delta \vdash \mathsf{do}\ x \leftarrow s; t \stackrel{\text{def}}{=} \mathsf{case}\ s\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto t \mid \mathsf{inr}\,(\_) \mapsto \mathsf{fail}$.
- We introduce the following abbreviation. If f is an expression (such as inl, inr) such that $f(x)$ is a term, then we write $t \gg= f$ for $\mathsf{do}\ x \leftarrow t; f(x)$.

The term $\mathsf{do}\ x \leftarrow s; t$ should be read as the following computation: Run s. If s returns a value, pass this as input x to the computation t; otherwise, diverge.

These constructions satisfy these computation rules:

$$\mathsf{do}\ x \leftarrow \mathsf{return}\ s; t = t[x := s] \qquad \mathsf{do}\ x \leftarrow \mathsf{fail}; t = \mathsf{fail}$$

$$\mathsf{do}\ x \leftarrow r; \mathsf{return}\ x = r \qquad \mathsf{do}\ \_ \leftarrow r; \mathsf{fail} = \mathsf{fail}$$

$$\mathsf{do}\ x \leftarrow r; (\mathsf{do}\ y \leftarrow s; t) = \mathsf{do}\ y \leftarrow (\mathsf{do}\ x \leftarrow r; s); t$$

This construction also allows us to define *scalar multiplication*. If we are given a scalar $\vdash s : \mathbf{2}$ and a substate $\vdash t : A+1$, the result of multiplying or scaling t by s is $\vdash \mathsf{do}\ \_ \leftarrow s; t : A+1$.

2.6.1 Partial Projections

Given $t : n \cdot A$, the *partial projection* $\rhd_i(t) : A+1$ is defined to be

$$\rhd_i(t) \stackrel{\text{def}}{=} \mathsf{case}\ {}^n_{j=1} t\ \mathsf{of}\ \mathsf{in}^n_j\,(x) \mapsto \begin{cases} \mathsf{return}\ x & \text{if } i = j \\ \mathsf{fail} & \text{otherwise} \end{cases}$$

2.6.2 Partial Pairing

The term $«s, t»$ is understood intuitively as follows. We are given two partial computations s and t, and we have derived the judgement $s{\downarrow} = t{\uparrow}$, which tells us that exactly one of s and t converges on any given input. We may then form the computation $«s, t»$ which, given an input x, returns either $s(x)$ or $t(x)$, whichever of the two converges.

2.6.3 Partial Sum

Let $\Gamma \vdash s, t : A + 1$. If these have disjoint domains (i.e. given any input x, the sum of the probability that s and t return a is never greater than 1), then we may form the computation $\Gamma \vdash s \ovee t$, the *partial sum* of s and t. The probability that this program converges with output a is the sum of the probability that s returns a, and the probability that t returns a. The definition is given by the rule ($\ovee$-def) ; see Section 5.3.

We write $n \cdot t$ for the sum $t \ovee \cdots \ovee t$ with n summands. We include the special cases $0 \cdot t \stackrel{\text{def}}{=} \mathsf{fail}$ and $1 \cdot t \stackrel{\text{def}}{=} t$.

With this operation, the partial functions in $A + 1$ form a *partial commutative monoid* (PCM) (see Lemma 10).

2.7 Logic

The type $\mathbf{2} = 1 + 1$ shall play a special role in this type theory. It is the type of *propositions* or *predicates*, and its objects shall be used as the formulas of our logic.

We define $\top \stackrel{\text{def}}{=} \mathsf{inl}\,(*)$ and $\bot \stackrel{\text{def}}{=} \mathsf{inr}\,(*)$. We also define the *orthosupplement* of a predicate p, which roughly corresponds to negation:

$$p^{\perp} \stackrel{\text{def}}{=} \mathsf{case}\ p\ \mathsf{of}\ \mathsf{inl}\,(\_) \mapsto \bot \mid \mathsf{inr}\,(\_) \mapsto \top$$

We immediately have that $p^{\perp\perp} = p$, $\top^{\perp} = \bot$ and $\bot^{\perp} = \top$.

The ordering on $\mathbf{2}$ shall play the role of the *derivability* relation in our logic: $p \leq q$ will indicate that q is derivable from p, or that p implies q. The rules for this logic are not the familiar rules of classical or intuitionistic logic. Rather, the predicates over any context form an *effect algebra* (Proposition 13).

2.7.1 n-tests

An *n-test* in a context Γ is an n-tuple of predicates $(p_1, \ldots, p_n)$ on A such that $\Gamma \vdash p_1 \ovee \cdots \ovee p_n = \top : \mathbf{2}$.

Intutively, this can be thought of as a set of n fuzzy predicates whose probabilities always sum to 1. We can think of this as a test that can be performed on the types of Γ with n possible outcomes; and, indeed, there is a one-to-one correspondence between the n-tests of Γ and the terms of type $\mathbf{n}$ (Lemma 18).

2.7.2 Instrument Maps

Let $x : A \vdash t : \mathbf{n}$ and $\Gamma \vdash s : A$. The term $\mathsf{instr}_{\lambda x t}(s) : n \cdot A$ is interpreted as follows: we read the computation $x : A \vdash t : \mathbf{n}$ as a test on the type A, with n possible outcomes. The computation $\mathsf{instr}_{\lambda x t}(s)$ runs t on (the output of) s, and returns $\mathsf{in}_i^n\,(s)$, where i is the outcome of the test.

Given an n-test $(p_1, \ldots, p_n)$ on A, we can write a program that tests which of $p_1, \ldots, p_n$ is true of its input, and performs one of n different calculations as a result. We write this program as $\Gamma \vdash \mathsf{measure}\ p_1 \mapsto t_1 \mid \cdots \mid p_n \mapsto t_n$. It will be defined in Definition 21.

If $x : A \vdash p : \mathbf{2}$ and $\Gamma, x : A \vdash s, t : A$, we define $\Gamma \vdash (\mathsf{if}\ p\ \mathsf{then}\ s\ \mathsf{else}\ t) \stackrel{\text{def}}{=} \mathsf{measure}\ p \mapsto s \mid p^{\perp} \mapsto t$. In the case where s and t do not depend on x, we have the following fact (Lemma 23.2): $\mathsf{if}\ p\ \mathsf{then}\ s\ \mathsf{else}\ t = \mathsf{case}\ p\ \mathsf{of}\ \mathsf{inl}\,(\_) \mapsto s \mid \mathsf{inr}\,(\_) \mapsto t$.

2.7.3 Assert Maps

If $x : A \vdash p : \mathbf{2}$ is a predicate, we define

$$\Gamma \vdash \mathsf{assert}_{\lambda x p}(t) \stackrel{\text{def}}{=} \mathsf{case}\ \mathsf{instr}_{\lambda x p}(t)\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto \mathsf{return}\ x \mid \mathsf{inr}\,(\_) \mapsto \mathsf{fail} : A + 1$$

The computation $\mathsf{assert}_p(t)$ is a partial computation with output type A. It tests whether p is true of t; if so, it leaves t unchanged; if not, it diverges. That is, if $p[x := t]$ returns $\top$, the computation converges and returns t; if not, it diverges.

These constructions satisfy the following computation rules (see Section 5.3.1 below for the proofs).

(assert↓) $(\mathsf{assert}_{\lambda x p}(t)) \downarrow = p[x := t]$
(assert-scalar) For a scalar $\vdash s : \mathbf{2}$: $\mathsf{assert}_{\lambda \_ s}(*) = \mathsf{instr}_{\lambda \_ s}(*) = s : \mathbf{2}$.
(instr+) For $x : A + B \vdash t : \mathbf{n}$:

$$\begin{aligned} \mathsf{instr}_{\lambda x t}(s) = \mathsf{case}\ s\ \mathsf{of}\ & \mathsf{inl}\,(y) \mapsto \mathsf{case}\ {}_{i=1}^{n} \mathsf{instr}_{\lambda a. t[x := \mathsf{inl}(a)]}(y)\ \mathsf{of}\ \mathsf{in}_i^n\,(z) \mapsto \mathsf{in}_i^n\,(\mathsf{inl}\,(z)) \\ & \mathsf{inr}\,(y) \mapsto \mathsf{case}\ {}_{i=1}^{n} \mathsf{instr}_{\lambda b. t[x := \mathsf{inl}(b)]}(y)\ \mathsf{of}\ \mathsf{in}_i^n\,(z) \mapsto \mathsf{in}_i^n\,(\mathsf{inr}\,(z)) \end{aligned}$$

(assert+) For $x : A + B \vdash p : \mathbf{2}$:

$$\begin{aligned} \mathsf{assert}_{\lambda x p}(t) = \mathsf{case}\ t\ \mathsf{of}\ & \mathsf{inl}\,(x) \mapsto \mathsf{do}\ z \leftarrow \mathsf{assert}_{\lambda a. p[x := \mathsf{inl}(a)]}(x); \mathsf{return}\ \mathsf{inl}\,(z) \mid \\ & \mathsf{inr}\,(y) \mapsto \mathsf{do}\ z \leftarrow \mathsf{assert}_{\lambda b. p[x := \mathsf{inr}(b)]}(y); \mathsf{return}\ \mathsf{inr}\,(z) \end{aligned}$$

(instr $\boldsymbol{m}$) For $x : \mathbf{m} \vdash t : \mathbf{n}$: $\mathsf{instr}_{\lambda x t}(s) = \mathsf{case}\ {}_{i=1}^{m} s\ \mathsf{of}\ i \mapsto \mathsf{case}\ {}_{j=1}^{n} t[x := i]\ \mathsf{of}\ j \mapsto \mathsf{in}_j^n\,(i)$
(assert $\boldsymbol{m}$) For $x : \mathbf{m} \vdash p : \mathbf{2}$: $\mathsf{assert}_{\lambda x p}(t) = \mathsf{case}\ {}_{i=1}^{m} t\ \mathsf{of}\ i \mapsto \mathsf{if}\ p[x := i]\ \mathsf{then}\ \mathsf{return}\ i\ \mathsf{else}\ \mathsf{fail}$

In particular, we have $\mathsf{assert}_{\mathsf{inl?}}(t) = \rhd_1(t)$ and $\mathsf{assert}_{\mathsf{inr?}}(t) = \rhd_2(t)$.

2.7.4 Sequential Product

Given two predicates $x : A \vdash p, q : \mathbf{2}$, we can define their *sequential product*

$$x : A \vdash p \mathbin{\&} q \stackrel{\text{def}}{=} \mathsf{do}\ x \leftarrow \mathsf{assert}_p(x); q : \mathbf{2} \quad .$$

The probability of this predicate being true at x is the product of the probabilities of p and q. This operation has many of the familiar properties of conjunction — including commutativity — but not all: in particular, we do not have $p \mathbin{\&} p^\perp = \perp$ in all cases. (For example, $1/2 \mathbin{\&} (1/2)^\perp = 1/4$.)

2.7.5 Coproducts

We can define predicates which, given a term $t : A + B$, test which of A and B the term came from. We write these as $\mathsf{inl?}\,(t)$ and $\mathsf{inr?}\,(t)$. (Compare these with the operators $FstAnd$ and $SndAnd$ defined in [10].) They are defined by

$$\begin{aligned} \mathsf{inl?}\,(t) &\stackrel{\text{def}}{=} \mathsf{case}\ t\ \mathsf{of}\ \mathsf{inl}\,(\_) \mapsto \top \mid \mathsf{inr}\,(\_) \mapsto \perp \\ \mathsf{inr?}\,(t) &\stackrel{\text{def}}{=} \mathsf{case}\ t\ \mathsf{of}\ \mathsf{inl}\,(\_) \mapsto \perp \mid \mathsf{inr}\,(\_) \mapsto \top \end{aligned}$$

2.7.6 Kernels

The predicate $\mathsf{inr?}\,()$ is particularly important for partial maps.

Let $\Gamma \vdash t : A + 1$. The *kernel* of the map denoted by t is

$$t\!\uparrow \overset{\text{def}}{=} \mathsf{inr?}\,(t) \overset{\text{def}}{=} \mathsf{case}\ t\ \mathsf{of}\ \mathsf{inl}\,(\_) \mapsto \bot \mid \mathsf{inr}\,(\_) \mapsto \top$$

Intuitively, if we think of t as a partial computation, then $t\!\uparrow$ is the proposition 't does not terminate', or the function that gives the probability that t will diverge on a given input.

Its orthosupplement, $(t\!\uparrow)^\perp = \mathsf{inl?}\,(t)$, which we shall also write as $t\!\downarrow$, is also called the *domain predicate* of t, and represents the proposition that t terminates. We note that it is equal to $\mathsf{do}\ \_ \leftarrow t; \top$.

2.7.7 Scalar Constants

The term $1/n$ represents the probability distribution on $\mathbf{2} = \{\top, \bot\}$ which returns $\top$ with probability $1/n$ and $\bot$ with probability $(n-1)/n$. It can be thought of as a coin toss, with a weighted coin that returns heads with probability $1/n$.

From this, we have a representation of the rational numbers between 0 and 1. Let m/n denote the term $1/n \ovee \cdots \ovee 1/n$, where there are m summands. The usual arithmetic of rational numbers can be carried out in our system (see Section 5.6).

2.8 Normalisation

Let $\vdash t : A + 1$. Then t represents a *substate* of A. As long as the probability $t\!\downarrow$ is non-zero, we can *normalise* this program over the probability of non-termination. The result is the state denoted by $\mathsf{nrm}\,(t)$. Intuitively, the probability that $\mathsf{nrm}\,(t)$ will output a is the probability that t will output $\mathsf{inl}\,(a)$, conditioned on the event that t terminates.

In order to type $\mathsf{nrm}\,(t)$, we must first prove that t has a non-zero probability of terminating by deriving an inequality of the form $1/n \leq t\!\downarrow$ for some positive integer $n \geq 2$.

If $\vdash t : A$ and $x : A \vdash p(x) : \mathbf{2}$, we write $\mathsf{cond}\,(t, p)$ for

$$\mathsf{cond}\,(t, p) \overset{\text{def}}{=} \mathsf{nrm}\,(\mathsf{assert}_p(t)) \quad .$$

The term t denotes a computation whose output is given by a probability distribution over A. Then $\mathsf{cond}\,(t, p)$ gives the result of normalising that conditional probability distribution with respect to p.

2.9 Marginalisation

The tensor product of type $A \otimes B$ comes with two *projections*. Given $\Gamma \vdash t : A \otimes B$, define

$$\Gamma \vdash \pi_1(t) \overset{\text{def}}{=} \mathsf{let}\ x \otimes \_ = t\ \mathsf{in}\ x : A \qquad \Gamma \vdash \pi_2(t) \overset{\text{def}}{=} \mathsf{let}\ \_ \otimes y = t\ \mathsf{in}\ y : B$$

If t is a state (i..e Γ is the empty context), then $\pi_1(t)$ denotes the result of *marginalising* t, as a probability distribution over $A \otimes B$, to a probability distribution over A.

2.10 Local Definition

In our examples, we shall make free use of *local definition*. This is not a part of the syntax of **COMET** itself, but part of our metalanguage. We write $\mathsf{let}\ x = s\ \mathsf{in}\ t$ for $t[x := s]$. We shall also locally define functions: we write $\mathsf{let}\ f(x) = s\ \mathsf{in}\ t$ for the result of replacing every subterm of the form $f(r)$ with $s[x := r]$ in t.

3 Examples

This section describes two examples of (Bayesian) reasoning in our type theory **COMET**. The first example is a typical exercise in Bayesian probability theory. Since such kind of reasoning is not very intuitive, a formal calculus is very useful. The second example involves a simple graphical model.

▶ **Example 1.** (See also [18, 3]) Consider the following situation.

> 1% of a population have a disease. 80% of subjects with the disease test positive, and 9.6% without the disease also test positive. If a subject is positive, what are the odds they have the disease?

This situation can be described as a very simple graphical model, with associated (conditional) probabilities.

$$\begin{tikzcd}
\boxed{\text{HasDisease}} \arrow[d] \\
\boxed{\text{PositiveResult}}
\end{tikzcd}$$

$\Pr(HD)$
0.01

HD	$\Pr(PR)$
t	0.8
f	0.096

In our type theory **COMET**, we use the following description.

$$\begin{array}{l}
\mathsf{let}\ \mathsf{subject} = 0.01\ \mathsf{in} \\
\qquad \mathsf{let}\ \mathsf{positive\_result}(x) = (\mathsf{if}\ x\ \mathsf{then}\ 0.8\ \mathsf{else}\ 0.096)\ \mathsf{in} \\
\mathsf{cond}\,(\mathsf{subject}, \mathsf{positive\_result})
\end{array}$$

We thus obtain a state $\mathsf{subject} : \mathbf{2}$, conditioned on the predicate $\mathsf{positive\_result}$ on $\mathbf{2}$. We calculate the outcome in semi-formal style. The conditional state $\mathsf{cond}\,(\mathsf{subject}, \mathsf{positive\_result})$ is defined via normalisation of assert, see Section 2.8. We first show what this assert term is, using the rule (assert m)and (assert-scalar):

$$\begin{array}{rcl}
\mathsf{assert}_{\mathsf{positive\_result}}(x) & = & \mathsf{if}\ \ x\ \ \mathsf{then}\ \ \mathsf{do}\ \_ \leftarrow \mathsf{assert}_{\mathsf{positive\_result}(\top)}(x); \mathsf{return}\ \top \\
& & \qquad\qquad \mathsf{else}\ \ \mathsf{do}\ \_ \leftarrow \mathsf{assert}_{\mathsf{positive\_result}(\bot)}(x); \mathsf{return}\ \bot \\
& = & \mathsf{if}\ \ x\ \ \mathsf{then}\ \ \mathsf{do}\ \_ \leftarrow \mathsf{assert}_{0.8}(x); \mathsf{return}\ \top \\
& & \qquad\qquad \mathsf{else}\ \ \mathsf{do}\ \_ \leftarrow \mathsf{assert}_{0.096}(x); \mathsf{return}\ \bot \\
& = & \mathsf{if}\ x\ \ \mathsf{then}\ \ \mathsf{if}\ 0.8\ \mathsf{then}\ \mathsf{return}\ \top\ \mathsf{else}\ \mathsf{fail} \\
& & \qquad\qquad \mathsf{else}\ \ \mathsf{if}\ 0.096\ \mathsf{then}\ \mathsf{return}\ \bot\ \mathsf{else}\ \mathsf{fail}
\end{array}$$

Conditioning requires that the domain of the substate $\mathsf{assert}_{\mathsf{positive\_result}}(\mathsf{subject})$ is non-zero. We compute this domain as:

$$\begin{array}{rcll}
\mathsf{assert}_{\mathsf{positive\_result}}(\mathsf{subject})\downarrow & = & \mathsf{positive\_result}(\mathsf{subject}) & \text{(Rule (assert}\downarrow\text{))} \\
& = & \mathsf{if}\ 0.01\ \mathsf{then}\ 0.8\ \mathsf{else}\ 0.096 & \\
& = & 0.01\ \&\ 0.8 \ovee 0.99\ \&\ 0.096 & \text{(Lemma 23.2)} \\
& = & 0.10304 & \text{(Lemma 25)}
\end{array}$$

Hence we can choose (for example) $n = 10$, to get $\frac{1}{n} \leq 0.10304 = \mathsf{assert}_{\mathsf{positive\_result}}(\mathsf{subject})\downarrow$.

We now proceed to calculate the result, answering the question in the beginning of this example.

$$\begin{aligned}
&\mathsf{assert}_{\mathsf{positive\_result}}(\mathsf{subject})\\
&= \mathsf{if}\ 0.01\ \ \mathsf{then\ if}\ 0.8\ \mathsf{then\ return}\ \top\ \mathsf{else\ fail}\\
&\qquad\qquad \mathsf{else\ if}\ 0.096\ \mathsf{then\ return}\ \bot\ \mathsf{else\ fail}\\
&= \mathsf{measure}\ \ 0.01\ \&\ 0.8 \quad \mapsto \mathsf{return}\ \top && \text{(Lemma 22.3)}\\
&\qquad\qquad 0.01\ \&\ 0.8^{\perp} \quad \mapsto \mathsf{fail}\\
&\qquad\qquad 0.01^{\perp}\ \&\ 0.096 \quad \mapsto \mathsf{return}\ \bot\\
&\qquad\qquad 0.01^{\perp}\ \&\ 0.096^{\perp} \mapsto \mathsf{fail}\\
&= \mathsf{measure}\ \ 0.008 \quad \mapsto \mathsf{return}\ \top && \text{(Lemma 22.5)}\\
&\qquad\qquad 0.09504 \mapsto \mathsf{return}\ \bot\\
&\qquad\qquad 0.89696 \mapsto \mathsf{fail}\\
&\mathsf{cond}\,(\mathsf{subject}, \mathsf{positive\_result})\\
&\overset{\text{def}}{=} \mathsf{nrm}\,(\mathsf{assert}_{\mathsf{positive\_result}}(\mathsf{subject}))\\
&= \mathsf{measure}\, 0.0776 \mapsto \top && \text{(Corollary 27)}\\
&\qquad\qquad 0.9224 \mapsto \bot\\
&= 0.0776. && \text{(Lemma 23.3)}
\end{aligned}$$

Hence the probability of having the disease after a positive test result is 7.8%.

▶ **Example 2** (Bayesian Network). The following is a standard example of a problem in Bayesian networks, created by [16, Chap. 14].

> I'm at work, neighbor John calls to say my alarm is ringing. Sometimes it's set off by minor earthquakes. Is there a burglar?
>
> We are given that the situation is as described by the following Bayesian network.

$$\begin{tikzcd}
\boxed{\text{Burglary}} \arrow[dr] & & \boxed{\text{Earthquake}} \arrow[dl] \\
& \boxed{\text{Alarm}} \arrow[dl] \arrow[dr] & \\
\boxed{\text{JohnCalls}} & & \boxed{\text{MaryCalls}}
\end{tikzcd}$$

$\Pr(B)$
$\frac{1}{1000}$

A	$\Pr(J)$
t	$\frac{9}{10}$
f	$\frac{1}{20}$

B	E	$\Pr(A)$
t	t	$\frac{95}{100}$
t	f	$\frac{94}{100}$
f	t	$\frac{29}{100}$
f	f	$\frac{1}{1000}$

$\Pr(E)$
$\frac{1}{500}$

A	$\Pr(M)$
t	$\frac{7}{10}$
f	$\frac{1}{100}$

The probability of each event given its preconditions is as given in the tables — for example, the probability that the alarm rings given that there is a burglar but no earthquake is 0.94.

We model the above question in **COMET** as follows.

$$\begin{aligned}
&\mathsf{let}\ b = 0.01\ \mathsf{in}\ \mathsf{let}\ e = 0.002\ \mathsf{in}\\
&\quad \mathsf{let}\ a(x,y) = (\mathsf{if}\ x\ \ \mathsf{then}\ (\mathsf{if}\ y\ \mathsf{then}\ 0.95\ \mathsf{else}\ 0.94)\\
&\qquad\qquad\qquad\qquad \mathsf{else}\ (\mathsf{if}\ y\ \mathsf{then}\ 0.29\ \mathsf{else}\ 0.001))\ \mathsf{in}\\
&\qquad \mathsf{let}\ j(z) = (\mathsf{if}\ z\ \mathsf{then}\ 0.9\ \mathsf{else}\ 0.05)\ \mathsf{in}\\
&\qquad\quad \mathsf{let}\ m(z) = (\mathsf{if}\ z\ \mathsf{then}\ 0.7\ \mathsf{else}\ 0.01)\ \mathsf{in}\\
&\pi_1\big(\mathsf{cond}\,(b \otimes e, j \circ a)\big)
\end{aligned}$$

We first elaborate the predicate $j \circ a$, given in context as $x\colon \mathbf{2}, y\colon \mathbf{2} \vdash j(a(x,y))\colon \mathbf{2}$. It is:

$$\begin{aligned}
j(a(x,y)) &= \mathsf{if}\ a(x,y)\ \mathsf{then}\ 0.90\ \mathsf{else}\ 0.05 \\
&= \mathsf{if}\ x\ \ \mathsf{then}\ (\mathsf{if}\ y\ \mathsf{then}\ (\mathsf{if}\ 0.95\ \mathsf{then}\ 0.90\ \mathsf{else}\ 0.05)\ \mathsf{else}\ (\mathsf{if}\ 0.94\ \mathsf{then}\ 0.90\ \mathsf{else}\ 0.05) \\
&\qquad\ \mathsf{else}\ (\mathsf{if}\ y\ \mathsf{then}\ (\mathsf{if}\ 0.29\ \mathsf{then}\ 0.90\ \mathsf{else}\ 0.05)\ \mathsf{else}\ (\mathsf{if}\ 0.001\ \mathsf{then}\ 0.90\ \mathsf{else}\ 0.05) \\
&= \mathsf{if}\ x\ \ \mathsf{then}\ (\mathsf{if}\ y\ \mathsf{then}\ 0.95 \mathbin{\&} 0.90 \ovee 0.95^{\perp} \mathbin{\&} 0.05\ \mathsf{else}\ 0.94 \mathbin{\&} 0.90 \ovee 0.94^{\perp} \mathbin{\&} 0.05) \\
&\qquad\ \mathsf{else}\ (\mathsf{if}\ y\ \mathsf{then}\ 0.29 \mathbin{\&} 0.90 \ovee 0.29^{\perp} \mathbin{\&} 0.05\ \mathsf{else}\ 0.001 \mathbin{\&} 0.90 \ovee 0.001^{\perp} \mathbin{\&} 0.05 \\
&= \mathsf{if}\ x\ \mathsf{then}\ (\mathsf{if}\ y\ \mathsf{then}\ 0.8575\ \mathsf{else}\ 0.849)\ \mathsf{else}\ (\mathsf{if}\ y\ \mathsf{then}\ 0.2965\ \mathsf{else}\ 0.05085)
\end{aligned}$$

The associated assert map is:

$$\begin{aligned}
\mathsf{assert}_{j\circ a}(b,e) &= \mathsf{measure}\ \begin{array}{ll}
0.001 \mathbin{\&} 0.002 \mathbin{\&} 0.8575 & \mapsto \mathsf{return}\ \top \otimes \top \\
0.001 \mathbin{\&} 0.998 \mathbin{\&} 0.849 & \mapsto \mathsf{return}\ \top \otimes \perp \\
0.999 \mathbin{\&} 0.002 \mathbin{\&} 0.2965 & \mapsto \mathsf{return}\ \perp \otimes \top \\
0.999 \mathbin{\&} 0.998 \mathbin{\&} 0.05085 & \mapsto \mathsf{return}\ \perp \otimes \perp \\
0.052138976^{\perp} & \mapsto \mathsf{fail}
\end{array} \\
&= \mathsf{measure}\ \begin{array}{ll}
0.000001715 & \mapsto \mathsf{return}\ \top \otimes \top \\
0.000847302 & \mapsto \mathsf{return}\ \top \otimes \perp \\
0.000592407 & \mapsto \mathsf{return}\ \perp \otimes \top \\
0.050697552 & \mapsto \mathsf{return}\ \perp \otimes \perp \\
0.052138976^{\perp} & \mapsto \mathsf{fail}
\end{array}
\end{aligned}$$

Hence by Corollary 27 we obtain the marginalised conditional:

$$\begin{aligned}
\pi_1\big(\mathsf{cond}\,(b \otimes e, j \circ a)\big) &= \pi_1\big(\mathsf{nrm}\,(\mathsf{assert}_{j\circ a}(b,e))\big) \\
&= \pi_1\big(\mathsf{measure}\ \begin{array}{ll}
{}^{0.000001715}/_{0.052138976} & \mapsto \top \otimes \top \\
{}^{0.000847302}/_{0.052138976} & \mapsto \top \otimes \perp \\
{}^{0.000592407}/_{0.052138976} & \mapsto \perp \otimes \top \\
{}^{0.050697552}/_{0.052138976} & \mapsto \perp \otimes \perp\big)
\end{array} \\
&= \mathsf{measure}\ \begin{array}{ll}
0.000032893 & \mapsto \pi_1(\top \otimes \top) \\
0.016250837 & \mapsto \pi_1(\top \otimes \perp) \\
0.011362078 & \mapsto \pi_1(\perp \otimes \top) \\
0.972354194 & \mapsto \pi_1(\perp \otimes \perp)
\end{array} \\
&= \mathsf{measure}\ \begin{array}{ll}
0.000032893 & \mapsto \top \\
0.016250837 & \mapsto \top \\
0.011362076 & \mapsto \perp \\
0.972354194 & \mapsto \perp
\end{array} \\
&= \mathsf{measure}\ \begin{array}{ll}
0.01628373 & \mapsto \top \\
0.98371627 & \mapsto \perp
\end{array} \\
&= 0.01628373
\end{aligned}$$

We conclude that there is a 1.6% chance of a burglary when John calls.

$$P(x_i(\vec{a}) = b) = \begin{cases} 1 \text{ if } b = a_i \\ 0 \text{ if } b \neq a_i \end{cases}$$

$$P(1/n(\vec{g}) = \top) = 1/n$$

$$P(1/n(\vec{g}) = \bot) = (n-1)/n$$

$$P(\mathsf{nrm}\,(t)\,(\vec{g}) = a) = P(t(\vec{g}) = a_1)/(1 - P(t(\vec{g}) = *_2))$$

$$P(\langle\!\langle s, t \rangle\!\rangle(\vec{g}) = a_1) = P(s(\vec{g}) = a_1)$$

$$P(\langle\!\langle s, t \rangle\!\rangle(\vec{g}) = b_2) = P(t(\vec{g}) = b_1)$$

$$P(\mathsf{left}\,(t)\,(\vec{g}) = a) = P(t(\vec{g}) = a_1)$$

$$P(\mathsf{instr}_{\lambda x t}(s)(\vec{g}) = a_i) = P(s(\vec{g}) = a)P(t(a) = i)$$

$$P((s \mathbin{\ovee} t)(\vec{g}) = a_1) = P(s(\vec{g}) = a_1) + P(t(\vec{g}) = a_1)$$

$$P((s \mathbin{\ovee} t)(\vec{g}) = *_2) = P(s(\vec{g}) = *_2) + P(t(\vec{g}) = *_2) - 1$$

$$P((\mathsf{let}\ x \otimes y = s\ \mathsf{in}\ t)(\vec{g}, \vec{d}) = c) = \sum_a \sum_b P(s(\vec{g}) = (a, b))P(t(\vec{d}, a, b) = c)$$

$$P(\mathsf{case}\ r\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto s \mid \mathsf{inr}\,(y) \mapsto t(\vec{g}, \vec{d}) = c)$$
$$= \sum_a P(r(\vec{g}) = a_1)P(s(\vec{d}, a) = c) + \sum_b P(r(\vec{g}) = b_2)P(t(\vec{d}, b) = c)$$

Figure 4 Semantics for **COMET** in $\mathcal{K}\ell\mathcal{D}$

4 Semantics

The terms of **COMET** are intended to represent probabilistic programs. We show how to give semantics to our system in three different ways: using discrete and continuous probability distributions, and simple set-theoretic semantics for deterministic computation.

4.1 Discrete Probabilistic Computation

We give an interpretation that assigns, to each term, a discrete probability distribution over its output type.

▶ **Definition 3.** Let A be a set.

- The *support* of a function $\phi : A \to [0, 1]$ is $\operatorname{supp} \phi = \{a \in A : \phi(a) \neq 0\}$.
- A *(discrete) probability distribution* over A is a function $\phi : A \to \phi$ with finite support such that $\sum_{a \in A} \phi(a) = 1$.
- Let $\mathcal{D}A$ be the set of all probability distributions on A.

We shall interpret every type A as a set $[\![A]\!]$. Assume we are given a set $[\![\mathbf{C}]\!]$ for each type constant $\mathbf{C}$. Define a set $[\![A]\!]$ for each type A thus:

$$[\![0]\!] = \emptyset \qquad [\![1]\!] = \{*\} \qquad [\![A + B]\!] = [\![A]\!] \uplus [\![B]\!] \qquad [\![A \otimes B]\!] = [\![A]\!] \times [\![B]\!]$$

where $A \uplus B = \{a_1 : a \in A\} \cup \{b_2 : b \in B\}$. We extend this to contexts by defining $[\![x_1 : A_1, \ldots, x_n : A_n]\!] = [\![A_1]\!] \times \cdots \times [\![A_n]\!]$.

Now, to every term $x_1 : A_1, \ldots, x_n : A_n \vdash t : B$, we assign a function $[\![t]\!] : [\![A_1]\!] \times \cdots \times [\![A_n]\!] \to \mathcal{D}\,[\![B]\!]$. The value $[\![t]\!]\,(a_1, \ldots, a_n)(b) \in [0, 1]$ will be written as $P(t(a_1, \ldots, a_n) = b)$, and should be thought of as the probability that b will be the output if a_1, …, a_n are the inputs. We give a few of the clauses in Figure 4. The others follow the same pattern.

The sums involved here are all well-defined because, for all t and $\vec{g}$, the function $P(t(\vec{g}) = -)$ has finite support.

▶ **Theorem 4** (Soundness). *1. If $\Gamma \vdash t : A$ is derivable, then for all $\vec{g} \in [\![\Gamma]\!]$, we have $P(t(\vec{g}) = -)$ is a probability distribution on $[\![A]\!]$.*
2. If $\Gamma \vdash s = t : A$, then $P(s(\vec{g}) = a) = P(t(\vec{g}) = a)$.

Proof. The proof is by induction on derivations. First prove $P(t[x := s](\vec{g}, \vec{a}) = b) = \sum_{a \in [\![A]\!]} P(s(\vec{g}) = a) P(t(\vec{d}, a) = b)$ whenever $t[x := s]$ is well-typed. ◀

As a corollary, we know that **COMET** is non-degenerate:

▶ **Corollary 5.** *Not every judgement is derivable; in particular, the judgement $\vdash \top = \bot : \mathbf{2}$ is not derivable.*

4.2 Alternative Semantics

It is also possible to give semantics to **COMET** using continuous probabilities. We assign a measurable space $[\![A]\!]$ to every type A. Each term then gives a measurable function $[\![A_1]\!] \times \cdots \times [\![A_n]\!] \to \mathcal{G}\,[\![B]\!]$, where $\mathcal{G}X$ is the space of all probability distributions over the measurable space X. ($\mathcal{G}$ here is the *Giry monad* [7].)

If we remove the constants $1/n$ from the system, we can give *deterministic* semantics to the subsystem, in which we assign a set to every type, and a function $[\![A_1]\!] \times \cdots \times [\![A_n]\!] \to [\![B]\!]$ to every term.

More generally, we can give an interpretation of **COMET** in any *commutative monoidal effectus with normalisation* in which there exists a scalar s such that $n \cdot s = 1$ for all positive integers n [6]. The discrete and continuous semantics we have described are two instances of this interpretation.

5 Metatheorems

We presented an overview of the system in Section 2, and gave the intuitive meaning of the terms of **COMET**. In this section, we proceed to a more formal development of the theory, and investigate what can be proved within the system.

The type theory we have presented enjoys the following standard properties.

▶ **Lemma 6.**
1. ***Weakening*** *If $\Gamma \vdash \mathcal{J}$ and $\Gamma \subseteq \Delta$ then $\Delta \vdash \mathcal{J}$.*
2. ***Substitution*** *If $\Gamma \vdash t : A$ and $\Delta, x : A \vdash \mathcal{J}$ then $\Gamma, \Delta \vdash \mathcal{J}[x := t]$.*
3. ***Equation Validity*** *If $\Gamma \vdash s = t : A$ then $\Gamma \vdash s : A$ and $\Gamma \vdash t : A$.*
4. ***Inequality Validity*** *If $\Gamma \vdash s \leq t : A + 1$ then $\Gamma \vdash s : A + 1$ and $\Gamma \vdash t : A + 1$.*
5. ***Functionality*** *If $\Gamma \vdash r = s : A$ and $\Delta, x : A \vdash t : B$ then $\Gamma, \Delta \vdash t[x := r] = t[x := s] : B$.*

Proof. The proof in each case is by induction on derivations. Each case is straightforward. ◀

The following lemma shows that substituting within our binding operations works as desired.

▶ **Lemma 7.** *1. If $\Gamma \vdash r : A \otimes B$; $\Delta, x : A, y : B \vdash s : C$; and $\Theta, z : C \vdash t : D$ then $\Gamma, \Delta, \Theta \vdash t[z := \mathsf{let}\ x \otimes y = r\ \mathsf{in}\ s] = \mathsf{let}\ x \otimes y = r\ \mathsf{in}\ t[z := s] : D$.*
2. If $\Gamma \vdash r : A + B$; $\Delta, x : A \vdash s : C$; $\Delta, y : B \vdash s' : C$; and $\Theta, z : C \vdash t : D$ then

$$\begin{aligned} \Gamma, \Delta, \Theta \vdash\ & t[z := \mathsf{case}\ r\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto s \mid \mathsf{inr}\,(y) \mapsto s'] \\ & = \mathsf{case}\ r\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto t[z := s] \mid \mathsf{inr}\,(y) \mapsto t[z := s'] : D \end{aligned} .$$

Proof. For part 1, we us the following 'trick' to simulate local definition (see [1]):

$$\begin{aligned}
& t[z := \mathsf{case}\ r\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto s \mid \mathsf{inr}\,(y) \mapsto s'] \\
&= \mathsf{let}\ z \otimes \_ = (\mathsf{case}\ r\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto s \mid \mathsf{inr}\,(y) \mapsto s') \otimes * \ \mathsf{in}\ t && (\beta\otimes) \\
&= \mathsf{let}\ z \otimes \_ = \mathsf{case}\ r\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto s \otimes * \mid \mathsf{inr}\,(y) \mapsto s' \otimes * \ \mathsf{in}\ t && (\text{case-}\otimes) \\
&= \mathsf{case}\ r\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto \mathsf{let}\ z \otimes \_ = s \otimes * \ \mathsf{in}\ t \mid \mathsf{inr}\,(y) \mapsto \mathsf{let}\ z \otimes \_ = s' \otimes * \ \mathsf{in}\ t && (\text{let-case}) \\
&= \mathsf{case}\ r\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto t[z := s] \mid \mathsf{inr}\,(y) \mapsto t[z := s'] && (\beta\otimes)
\end{aligned}$$

Part 2 is proven similarly using (let-⊗) and (let-let) . ◀

▶ **Corollary 8.** *1.* *If* $\Gamma \vdash s : A \otimes B$ *and* $\Delta \vdash t : C$ *then* $\Gamma, \Delta \vdash \mathsf{let}\ \_ \otimes \_ = s\ \mathsf{in}\ t = t : C$.
2. *If* $\Gamma \vdash s : A + B$ *and* $\Delta \vdash t : C$ *then* $\Gamma, \Delta \vdash \mathsf{case}\ s\ \mathsf{of}\ \mathsf{inl}\,(\_) \mapsto t \mid \mathsf{inr}\,(\_) \mapsto t = t : C$.

Proof. These are both the special case where z does not occur free in t. ◀

5.1 Coproducts

We generalise the $\mathsf{inl}?\,()$ and $\mathsf{inr}?\,()$ constructions as follows. Define the predicate $\mathsf{in}_i?\,()$ on $n \cdot A$, which tests whether a term comes from the ith component, as follows.

$$\mathsf{in}_i?\,(t) \stackrel{\text{def}}{=} \mathsf{case}\ {}_{j=1}^{n} t\ \mathsf{of}\ \mathsf{in}_j^n\,(\_) \mapsto \begin{cases} \top & \text{if } i = j \\ \bot & \text{if } i \neq j \end{cases}$$

5.2 Kernels

▶ **Lemma 9.**
1. *If* $\Gamma \vdash t : A + 1$ *then* $\Gamma \vdash t{\downarrow} = (\mathsf{do}\ \_ \leftarrow t; \top) : \mathbf{2}$
2. *Let* $\Gamma \vdash t : A + 1$*. Then* $\Gamma \vdash t{\downarrow} = \bot : \mathbf{2}$ *if and only if* $\Gamma \vdash t = \mathsf{fail} : A + 1$.
3. *Let* $\Gamma \vdash s : A + 1$ *and* $\Delta, x : A \vdash t : B + 1$*. Then* $\Gamma, \Delta \vdash (\mathsf{do}\ x \leftarrow s; t){\downarrow} = \mathsf{do}\ x \leftarrow s; t{\downarrow} : \mathbf{2}$.

Proof.
1. This holds just by expanding definitions.
2. Obviously, $(\mathsf{fail}{\downarrow}) = \bot$. For the converse, if $t{\downarrow} = \bot$ then $t{\uparrow} = \top$ and so $t = \mathsf{inr}\,(\mathsf{right}\,(t)) = \mathsf{inr}\,(*)$ by $(\eta 1)$.
3. $$\begin{aligned}(\mathsf{case}\ s\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto t \mid \mathsf{inr}\,(\_) \mapsto \mathsf{fail}{\downarrow}) &= \mathsf{case}\ s\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto t{\downarrow} \mid \mathsf{inr}\,(\_) \mapsto \mathsf{fail}{\downarrow} \\ &= \mathsf{case}\ s\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto t{\downarrow} \mid \mathsf{inr}\,(\_) \mapsto \bot\end{aligned}$$

◀

5.3 Ordering on Partial Maps and the Partial Sum

Note that, from the rules (⊚) and (⊚-def) , we have $\Gamma \vdash s \ovee t : A + 1$ if and only if there exists $\Gamma \vdash b : (A + A) + 1$ such that

$$\Gamma \vdash b \gg\!= \rhd_1 = s : A + 1, \qquad \Gamma \vdash b \gg\!= \rhd_2 = t : A + 1 \ ,$$

in which case $\Gamma \vdash s \ovee t = \mathsf{do}\ x \leftarrow b; \mathsf{return}\ \nabla(x) : A + 1$. We say that such a term b is a *bound* for $s \ovee t$. By the rule (JM) , this bound is unique if it exists.

The set of *partial* maps $A \to B + 1$ between any two types A and B form a *partial commutative monoid* (PCM) with least element fail, as shown by the following results.

▶ **Lemma 10.**

1. *If* $\Gamma \vdash t : A + 1$ *then* $\Gamma \vdash t \oslash \mathsf{fail} = t : A + 1$.
2. ***(Commutativity)*** *If* $\Gamma \vdash s \oslash t : A+1$ *then* $\Gamma \vdash t \oslash s : A+1$ *and* $\Gamma \vdash s \oslash t = t \oslash s : A+1$.
3. ***(Associativity)*** $\Gamma \vdash (r \oslash s) \oslash t : A+1$ *if and only if* $\Gamma \vdash r \oslash (s \oslash t) : A+1$*, in which case* $\Gamma \vdash r \oslash (s \oslash t) = (r \oslash s) \oslash t : A+1$.

Proof. We prove part 2 here. Let b be a bound for $s \oslash t$. Then $\mathsf{do}\ x \leftarrow b; \mathsf{return\ swap}\,(x)$ is a bound for $t \oslash s$ and we have $t \oslash s = \mathsf{do}\ y \leftarrow (\mathsf{do}\ x \leftarrow b; \mathsf{return\ swap}\,(x)); \mathsf{return}\ \nabla(y) = s \oslash t$. ◀

▶ **Lemma 11.** *Let* $\Gamma \vdash r : A+1$ *and* $\Gamma \vdash s : A+1$*. Then* $\Gamma \vdash r \leq s : A+1$ *if and only if there exists* t *such that* $\Gamma \vdash r \oslash t = s : A+1$.

Proof. Suppose $r \leq s$. If b is such that $\mathsf{do}\ x \leftarrow b; \rhd_1(x) = r$ and $\mathsf{do}\ x \leftarrow b; \mathsf{return}\ \nabla(x) = s$ then take $t = \mathsf{do}\ x \leftarrow b; \rhd_2(x)$.

Conversely, if $r \oslash t = s$, then inverting the derivation of $\Gamma \vdash r \oslash t : A+1$ we have that there exists b such that $r = \mathsf{do}\ x \leftarrow b; \rhd_1(x)$, $t = \mathsf{do}\ x \leftarrow b; \rhd_2(x)$ and $s = r \oslash t = \mathsf{do}\ x \leftarrow b; \mathsf{return}\ \nabla(x)$. Therefore, $r \leq s$ by (order) . ◀

In this case, the bound for $r \oslash s$ will also be called a bound for $r \leq s$.

▶ **Lemma 12.**
1. *If* $\Gamma \vdash s \oslash t : A+1$ *then* $\Gamma \vdash s \leq s \oslash t : A+1$ *and* $\Gamma \vdash t \leq s \oslash t : A+1$.
2. *If* $\Gamma \vdash t : A+1$ *then* $\Gamma \vdash t \leq t : A+1$.
3. *If* $\Gamma \vdash t : A+1$ *then* $\Gamma \vdash \mathsf{fail} \leq t : A+1$.
4. *If* $\Gamma \vdash r \leq s : A+1$ *and* $\Gamma \vdash s \leq t : A+1$ *then* $\Gamma \vdash r \leq t : A+1$.
5. *If* $\Gamma \vdash r \leq s : A+1$ *and* $\Gamma \vdash s \oslash t : A+1$ *then* $\Gamma \vdash r \oslash t \leq s \oslash t : A+1$.

Proof. Parts 1–4 follow by applying Lemma 11 to the appropriate part of Lemma 10. For part 5, let $r \oslash x = s$. Then $r \oslash x \oslash t = s \oslash t$ and so $r \oslash t \leq s \oslash t$. ◀

On the predicates, we have the following structure, which shows that they form an *effect algebra*. (In fact, they have more structure: they form an *effect module* over the scalars, as we will prove in Proposition 17.)

▶ **Proposition 13.** *Let* $\Gamma \vdash p, q, r : \mathbf{2}$.
1. *If* $\Gamma \vdash p : \mathbf{2}$ *then* $\Gamma \vdash p \oslash p^\perp = \top : \mathbf{2}$.
2. *If* $\Gamma \vdash p \oslash q = \top : \mathbf{2}$ *then* $\Gamma \vdash q = p^\perp : \mathbf{2}$.
3. ***(Zero-One Law)*** *If* $\Gamma \vdash p \oslash \top : \mathbf{2}$ *then* $\Gamma \vdash p = \bot : \mathbf{2}$.
4. $\Gamma \vdash p \oslash q : \mathbf{2}$ *if and only if* $\Gamma \vdash p \leq q^\perp : \mathbf{2}$.
5. *Suppose* $\Gamma \vdash r : A + B$ *and* $\Delta, x : A \vdash s \oslash t : C+1$ *and* $\Delta, y : B \vdash s' \oslash t' : C+1$*. Then*

$$\begin{aligned}\Gamma, \Delta \vdash\ & \mathsf{case}\ r\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto s \oslash t \mid \mathsf{inr}\,(y) \mapsto s' \oslash t' \\ &= (\mathsf{case}\ r\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto s \mid \mathsf{inr}\,(y) \mapsto s') \oslash (\mathsf{case}\ r\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto t \mid \mathsf{inr}\,(y) \mapsto t') : C+1\end{aligned}$$

6. *If* $\Gamma \vdash r : A+1$ *and* $\Delta, x : A \vdash s \oslash t : B+1$ *then* $\Gamma, \Delta \vdash \mathsf{do}\ x \leftarrow r; s \oslash t = (\mathsf{do}\ x \leftarrow r; s) \oslash (\mathsf{do}\ x \leftarrow r; t) : B+1$.

Proof. We prove part 2 here. Let b be a bound for $p \oslash q$. We have $\top = \mathsf{do}\ x \leftarrow b; \mathsf{return}\ \nabla(x) = \mathsf{do}\ x \leftarrow b; \top = b\downarrow$. Therefore, $b = \mathsf{inl}\,(\mathsf{left}\,(b))$ by (βleft) , and so $p = \rhd_1(\mathsf{left}\,(b))$ and $q = \rhd_2(\mathsf{left}\,(b)) = \rhd_1(\mathsf{left}\,(b))^\perp = p^\perp$. ◀

▶ **Corollary 14.** *1.* ***(Cancellation)*** *If* $\Gamma \vdash p \oslash q = p \oslash r : \mathbf{2}$ *then* $\Gamma \vdash q = r : \mathbf{2}$.
2. ***(Positivity)*** *If* $\Gamma \vdash p \oslash q = \bot : \mathbf{2}$ *then* $\Gamma \vdash p = \bot : \mathbf{2}$ *and* $\Gamma \vdash q = \bot : \mathbf{2}$.
3. *If* $\Gamma \vdash p : \mathbf{2}$ *then* $\Gamma \vdash p \leq \top : \mathbf{2}$.

4. *If* $\Gamma \vdash p \leq q : \mathbf{2}$ *then* $\Gamma \vdash q^{\perp} \leq p^{\perp} : \mathbf{2}$.
5. *If* $\Gamma \vdash p \leq q : \mathbf{2}$ *and* $\Gamma \vdash q \leq p : \mathbf{2}$ *then* $\Gamma \vdash p = q : \mathbf{2}$.

Proof. We prove part 1 here. We have $p \owedge q \owedge (p \owedge q)^{\perp} = p \owedge r \owedge (p \owedge q)^{\perp} = \top$, and therefore. $q = r = (p \owedge (p \owedge q)^{\perp})^{\perp}$. ◀

5.3.1 Assert Maps

Recall that, for $x : A \vdash p : \mathbf{2}$ and $\Gamma \vdash t : A$, we define $\Gamma \vdash \mathsf{assert}_{\lambda x p}(t) \stackrel{\text{def}}{=} \rhd_1(\mathsf{instr}_{\lambda x p}(t)) : A + 1$.

We now give rules for calculating $\mathsf{instr}_{\lambda x p}$ and $\mathsf{assert}_{\lambda x p}$ directed by the type.

▶ **Lemma 15** ((assert-scalar))**.** *If* $\vdash s : \mathbf{2}$ *then*

$$\vdash \mathsf{assert}_{\lambda\_s}(*) = \mathsf{instr}_{\lambda\_s}(*) = s : \mathbf{2}$$

Proof. We have $\nabla(s) = *$ by $(\eta 1)$ and $s{\downarrow} = s$ by $(\eta+)$. The result follows by (ηinstr) . ◀

▶ **Lemma 16.** *The rules* (instr+) *and* (assert+) *are admissible.*

Proof. For $x : A + B$, let us write $f(x)$ for $\mathsf{case}\ x\ \mathsf{of}\ \mathsf{inl}\,(y) \mapsto (\mathsf{inl} + \mathsf{inl})(\mathsf{instr}_{\lambda a.p[\mathsf{inl}(a)]}(y)) \mid \mathsf{inr}\,(z) \mapsto (\mathsf{inr} + \mathsf{inr})(\mathsf{instr}_{\lambda b.p[\mathsf{inr}(b)]}(z))$. We shall prove $f(x) = \mathsf{instr}_{\lambda x p}(x)$.

We have

$$\begin{aligned}
\nabla(f(x)) &= \mathsf{case}\ x\ \mathsf{of}\ \ \mathsf{inl}\,(y) \mapsto \mathsf{inl}\,\big(\nabla(\mathsf{assert}_{\lambda a.p[x:=\mathsf{inl}(a)]}(y))\big) \mid \\
&\qquad\qquad\quad\ \mathsf{inr}\,(z) \mapsto \mathsf{inr}\,\big(\nabla(\mathsf{assert}_{\lambda b.p[\mathsf{inr}(b)]}(z))\big) \\
&= \mathsf{case}\ x\ \mathsf{of}\ \mathsf{inl}\,(y) \mapsto \mathsf{inl}\,(y) \mid \mathsf{inr}\,(z) \mapsto \mathsf{inr}\,(z) = x && \text{by } (\eta+) \\
f(x){\downarrow} &= \mathsf{case}\ x\ \mathsf{of}\ \mathsf{inl}\,(y) \mapsto \mathsf{instr}_{\lambda a.p[x:=\mathsf{inl}(a)]}(y){\downarrow} \mid \mathsf{inr}\,(z) \mapsto \mathsf{instr}_{\lambda b.p[\mathsf{inr}(b)]}(z){\downarrow} \\
&= \mathsf{case}\ x\ \mathsf{of}\ \mathsf{inl}\,(y) \mapsto p[x := \mathsf{inl}\,(y)] \mid \mathsf{inr}\,(z) \mapsto p[x := \mathsf{inr}\,(z)] = p
\end{aligned}$$

the last step using Corollary 8.2. Hence $f(x) = \mathsf{instr}_p(x)$ by (ηinstr) . ◀

The rules (instr m)and (assert m)follow easily.

5.4 Sequential Product

We do not have conjunction or disjunction in our language for predicates over the same type, as this would involve duplicating variables. However, we do have the following *sequential product.* (This was called the ‘and-then’ test operator in Section 9 in [10].)

Let $x : A \vdash p, q : \mathbf{2}$. We define the *sequential product* $p \mathbin{\&} q$ by

$$x : A \vdash p \mathbin{\&} q \stackrel{\text{def}}{=} \mathsf{do}\ x \leftarrow \mathsf{assert}_{\lambda x p}(x); q : \mathbf{2}\ \ .$$

▶ **Proposition 17.** *Let* $x : A \vdash p, q : \mathbf{2}$.

1. $\mathsf{instr}_{p\&q}(x) = \mathsf{case}\ \mathsf{instr}_p(x)\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto \mathsf{instr}_q(x) \mid \mathsf{inr}\,(y) \mapsto \mathsf{inr}\,(y)$
2. $\mathsf{assert}_{p\&q}(x) = \mathsf{do}\ x \leftarrow \mathsf{assert}_p(x); \mathsf{assert}_q(x) \stackrel{def}{=} \mathsf{assert}_p(x) \gg\!= \mathsf{assert}_q$
3. ***(Commutativity)*** $p \mathbin{\&} q = q \mathbin{\&} p$.
4. $(p \owedge q) \mathbin{\&} r = p \mathbin{\&} r \owedge q \mathbin{\&} r$ *and* $p \mathbin{\&} (q \owedge r) = p \mathbin{\&} q \owedge p \mathbin{\&} r$.
5. $p \mathbin{\&} \perp = \perp \mathbin{\&} q = \perp$
6. $p \mathbin{\&} \top = p$ *and* $\top \mathbin{\&} q = q$
7. $p \mathbin{\&} (q \mathbin{\&} r) = (p \mathbin{\&} q) \mathbin{\&} r$
8. *Let* $x : A \vdash p : \mathbf{2}$. *If* x *does not occur in* q*, then* $p \mathbin{\&} q = \mathsf{case}\ p\ \mathsf{of}\ \mathsf{inl}\,(\_) \mapsto q \mid \mathsf{inr}\,(\_) \mapsto \perp$.

Proof. We shall prove part 1 here.

$$\begin{aligned}&\mathsf{inl?}\,(\mathsf{case}\ \mathsf{instr}_p(x)\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto \mathsf{instr}_q(x) \mid \mathsf{inr}\,(y) \mapsto \mathsf{inr}\,(y))\\ &\ = \mathsf{case}\ \mathsf{instr}_p(x)\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto q \mid \mathsf{inr}\,(y) \mapsto \bot = \mathsf{do}\ x \leftarrow \mathsf{assert}_p(x); q = p \,\&\, q\\ &\nabla(\mathsf{case}\ \mathsf{instr}_p(x)\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto \mathsf{instr}_q(x) \mid \mathsf{inr}\,(y) \mapsto \mathsf{inr}\,(y))\\ &\ = \mathsf{case}\ \mathsf{instr}_p(x)\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto x \mid \mathsf{inr}\,(y) \mapsto y = \nabla(\mathsf{instr}_p(x)) = x\end{aligned}$$

so the result follows by $(\eta\mathrm{instr})$. ◀

These results show that the scalars form an *effect monoid*, and the predicates on any type form an *effect module* over that effect monoid (see [10] Lemma 13 and Proposition 14).

5.5 n-tests

Recall that an *n-test* on a type A is an n-tuple $(p_1, \ldots, p_n)$ such that $x : A \vdash p_1 \otimes \cdots \otimes p_n = \top : \mathbf{2}$.

The following lemma shows that there is a one-to-one correspondance between the n-tests on A, and the maps $A \to \mathbf{n}$.

▶ **Lemma 18.** *For every n-test $(p_1, \ldots, p_n)$ on A, there exists a term $x : A \vdash t(x) : \mathbf{n}$, unique up to equality, such that $x : A \vdash p_i(x) = \rhd_i(t(x)) : \mathbf{2}$.*

Proof. The proof is by induction on n. The case $n = 1$ is trivial.

Suppose the result is true for n. Take an $n+1$-test $(p_1, \ldots, p_{n+1})$. Then $(p_1, p_2, \ldots, p_n \otimes p_{n+1})$ is an n-test. By the induction hypothesis, there exists $t : \mathbf{n}$ such that $\rhd_i(t) = p_i$ for $i < n$ and $\rhd_n(t) = p_n \otimes p_{n+1}$. Let $b : \mathbf{3}$ be the bound for $p_n \otimes p_{n+1}$. Reading t and b as partial functions in $\mathbf{n-1}+1$ and $\mathbf{2}+1$, we have that $t{\uparrow} = b{\downarrow} = p_n \otimes p_{n+1}$. Hence «$b, t$» $: \mathbf{2} + \mathbf{n-1}$ exists. Reading it as a term of type $\mathbf{n+1}$, we have that

$$\rhd_1(«b,t») = p_n, \quad \rhd_2(«b,t») = p_{n+1}, \quad \rhd_{i+2}(«b,t») = p_i\ (i < n)\ .$$

From this it is easy to construct the term of type $\mathbf{n+1}$ required. ◀

We write $\mathsf{instr}_{(p_1,\ldots,p_n)}(s)$ for $\mathsf{instr}_t(s)$, where t is the term such that $\rhd_i(t) = p_i$ for each i.

▶ **Lemma 19.** *$\mathsf{instr}_{(p_1,\ldots,p_n)}(x)$ is the unique term such that $\mathsf{in}_i?\,\big(\mathsf{instr}_{(p_1,\ldots,p_n)}(x)\big) = p_i$ for all i and $\nabla(\mathsf{instr}_{(p_1,\ldots,p_n)}(x)) = x$.*

▶ **Lemma 20.**

$$\mathsf{instr}_{p_i}(x) = \mathsf{case}\ {}^{n}_{j=1}\mathsf{instr}_{(p_1,\ldots,p_n)}(x)\ \mathsf{of}\ \mathsf{in}^n_j\,(x) \mapsto \begin{cases}\mathsf{inl}\,(x) & \textit{if } i = j\\ \mathsf{inr}\,(x) & \textit{if } i \neq j\end{cases}$$

$$\mathsf{assert}_{p_i}(x) = \mathsf{case}\ {}^{n}_{j=1}\mathsf{instr}_{(p_1,\ldots,p_n)}(x)\ \mathsf{of}\ \mathsf{in}^n_j\,(x) \mapsto \begin{cases}\mathsf{return}\ x & \textit{if } i = j\\ \mathsf{fail} & \textit{if } i \neq j\end{cases}$$

Proof. The first formula holds because $\mathsf{inl?}\,()$ maps the right-hand side to $\mathsf{in}_i?\,\big(\mathsf{instr}_{(p_1,\ldots,p_n)}(x)\big) = p_i$, and ∇ maps the right-hand side to x. The second formula follows immediately from the first. ◀

We can now define the program that divides into n branches depending on the outcome of an n-test:

▶ **Definition 21.** Given $x : A \vdash p_1(x) \owedge \cdots \owedge p_n(x) = \top : \mathbf{2}$, define

$$\begin{aligned} x : A \vdash & \ \mathsf{measure}\ p_1(x) \mapsto t_1(x) \mid \cdots \mid p_n(x) \mapsto t_n(x) \\ & \overset{\text{def}}{=} \mathsf{case}\ \mathsf{instr}_{(p_1,\ldots,p_n)}(x)\ \mathsf{of}\ \mathsf{in}_1?\,(x) \mapsto t_1(x) \mid \cdots \mid \mathsf{in}_n?\,(x) \mapsto t_n(x) \end{aligned}$$

▶ **Lemma 22.** *The* measure *construction satisfies the following laws.*

1. $(\mathsf{measure}\ \top \mapsto t) = t$
2. $(\mathsf{measure}\ p_1 \mapsto t_1 \mid \cdots \mid p_n \mapsto t_n \mid \bot \mapsto t_{n+1}) = (\mathsf{measure}\ p_1 \mapsto t_1 \mid \cdots \mid p_n \mapsto t_n)$
3. $(\mathsf{measure}_i\ p_i \mapsto \mathsf{measure}_j\ q_{ij} \mapsto t_{ij}) = (\mathsf{measure}_{i,j}\ p_i \ \&\ q_{ij} \mapsto t_{ij})$
4. *For any permutation* π *of* $\{1,\ldots,n\}$, $\mathsf{measure}_i\ p_i \mapsto t_i = \mathsf{measure}_i\ p_{\pi(i)} \mapsto t_{\pi(i)}$.
5. *If* $t_n = t_{n+1}$ *then* $\mathsf{measure}_{i=1}^{n} p_i \mapsto t_i = \mathsf{measure}\ p_1 \mapsto t_1 \mid \cdots \mid p_{n-1} \mapsto t_{n-1} \mid p_n \owedge p_{n+1} \mapsto t_n$.

Proof. We shall prove part 3. The proof for the other parts follows the same pattern. Let us write $\mathsf{in}_{i,j}\,()$ $(1 \leq i \leq m,\ 1 \leq j \leq n_i)$ for the constructors of $(n_1 + \cdots + n_m) \cdot A$, and $\mathsf{in}_{i,j}?\,()$ for the corresponding predicates.

It suffices to prove that $\mathsf{instr}_{(p_i \& q_{ij})_{i,j}}(x) = \mathsf{case}\ {}_{i=1}^{m}\ \mathsf{instr}_{\vec{p}}(x)\ \mathsf{of}\ \mathsf{in}_i^m\,(x) \mapsto \mathsf{case}\ {}_{j=1}^{n_i}\ \mathsf{instr}_{\vec{q_i}}(x)\ \mathsf{of}\ \mathsf{in}_j^{n_i}\,(x) \mapsto \mathsf{in}_{i,j}\,(x)$.

Let R denote the right-hand side. We have

$$\begin{aligned} \mathsf{in}_{i,j}?\,(R) &= \mathsf{case}\ {}_{i'=1}^{m}\ \mathsf{instr}_{\vec{p}}(x)\ \mathsf{of}\ \mathsf{in}_{i'}^m\,(x) \mapsto \begin{cases} q_{ij} & \text{if } i = i' \\ \bot & \text{if } i \neq i' \end{cases} \\ &= \mathsf{do}\ x \leftarrow \mathsf{assert}_{p_i}(x); q_{ij} = p_i \ \&\ q_{ij} && \text{(by Lemma 20)} \\ \nabla(R) &= \mathsf{case}\ {}_{i=1}^{m}\ \mathsf{instr}_{\vec{p}}(x)\ \mathsf{of}\ \mathsf{in}_i^m\,(x) \mapsto x = \nabla(\mathsf{instr}_{\vec{p}}(x)) = x \end{aligned}$$

The result follows by $(\eta\mathrm{instr})$. ◀

Let $x : A \vdash p : \mathbf{2}$ and $\Gamma, x : A \vdash s, t : B$. We define $\mathsf{if}\ p\ \mathsf{then}\ s\ \mathsf{else}\ t \overset{\text{def}}{=} \mathsf{measure}\ p \mapsto s \mid p^\perp \mapsto t : B$. Note that, in the case where s and t do not depend on x, this is equal to $\mathsf{case}\ p\ \mathsf{of}\ \mathsf{inl}\,(\_) \mapsto s \mid \mathsf{inr}\,(\_) \mapsto t$.

▶ **Lemma 23.** *1.* *If* $x : A \vdash p_1 \owedge \cdots \owedge p_n = \top : \mathbf{2}$ *and* $x : A \vdash q_1, \ldots, q_n : \mathbf{2}$, *then*

$$(\mathsf{measure}\ p_1 \mapsto q_1 \mid \cdots \mid p_n \mapsto q_n) = p_1 \ \&\ q_1 \owedge \cdots \owedge p_n \ \&\ q_n \ .$$

2. *Let* $x : A \vdash p : \mathbf{2}$ *and* $\Gamma \vdash q, r : B$ *where* $x \notin \Gamma$. *Then* $\mathsf{if}\ p\ \mathsf{then}\ q\ \mathsf{else}\ r = \mathsf{case}\ p\ \mathsf{of}\ \mathsf{inl}\,(\_) \mapsto q \mid \mathsf{inr}\,(\_) \mapsto r : B$.
3. *Let* $x : A \vdash p : \mathbf{2}$. *Then* $x : A \vdash \mathsf{if}\ p\ \mathsf{then}\ \top\ \mathsf{else}\ \bot = p : \mathbf{2}$.

Proof. **1.** Immediate from Lemma 19.

2. We have

$$\begin{aligned} \mathsf{measure}\ p \mapsto q \mid p^\perp \mapsto r &\overset{\text{def}}{=} \mathsf{case}\ \mathsf{instr}_{\lambda x p}(x)\ \mathsf{of}\ \mathsf{inl}\,(\_) \mapsto q \mid \mathsf{inr}\,(\_) \mapsto r \\ &= \mathsf{case}\ \mathsf{inl}?\,(\mathsf{instr}_{\lambda x p}(x))\ \ \mathsf{of}\ \mathsf{inl}\,(\_) \mapsto q \mid \mathsf{inr}\,(\_) \mapsto r \\ &= \mathsf{case}\ p\ \mathsf{of}\ \mathsf{inl}\,(\_) \mapsto q \mid \mathsf{inr}\,(\_) \mapsto r \end{aligned}$$

3. $\mathsf{if}\ p\ \mathsf{then}\ \top\ \mathsf{else}\ \bot = \mathsf{case}\ p\ \mathsf{of}\ \mathsf{inl}\,(\_) \mapsto \top \mid \mathsf{inr}\,(\_) \mapsto \bot = p$ by $(\eta+)$.

◀

5.6 Scalars

From the rules given in Figure 2, the usual algebra of the rational interval from 0 to 1 follows.

▶ **Lemma 24.** *If $p/q = m/n$ as rational numbers, then* $\vdash p \cdot (1/q) = m \cdot (1/n) : \mathbf{2}$.

Proof. We first prove that $\vdash a \cdot (1/ab) = 1/b : \mathbf{2}$ for all a, b. This holds because $ab \cdot (1/ab) = \top$ by $(n \cdot 1/n)$, hence $a \cdot (1/ab) = 1/b$ by (divide) .

Hence we have $p \cdot (1/q) = pn \cdot (1/nq) = qm \cdot (1/nq) = m \cdot (1/n)$. ◀

Recall that within **COMET**, we are writing m/n for the term $m \cdot (1/n)$. Similar reasoning leads us to

▶ **Lemma 25.** *Let q and r be rational numbers in* $[0, 1]$.

1. *If $q \leq r$ in the usual ordering, then* $\vdash q \leq r : \mathbf{2}$.
2. $\vdash q \varovee r : \mathbf{2}$ *iff* $q + r \leq 1$, *in which case* $\Gamma \vdash q \varovee r = q + r : \mathbf{2}$.
3. $\vdash q \mathbin{\&} r = qr : \mathbf{2}$.

5.7 Normalisation

The following lemma gives us a rule that allows us to calculate the normalised form of a substate in many cases, including the examples in Section 3.

▶ **Lemma 26.** *Let* $\vdash t : A + 1$, $\vdash p_1 \varovee \cdots \varovee p_n = \top : \mathbf{2}$, *and* $\vdash q : \mathbf{2}$. *Let* $\vdash s_1, \ldots, s_n : A$. *Suppose* $\vdash 1/m \leq q : \mathbf{2}$. *If*

$$\vdash t = \mathsf{measure}\ p_1 \mathbin{\&} q \mapsto \mathsf{return}\ s_1 \mid \cdots \mid p_n \mathbin{\&} q \mapsto \mathsf{return}\ s_n \mid q^\perp \mapsto \mathsf{fail} : A + 1 \ \ , \ \ \textit{then}$$
$$\vdash \mathsf{nrm}\,(t) = \mathsf{measure}\ p_1 \mapsto s_1 \mid \cdots \mid p_n \mapsto s_n : A$$

Proof. Let $\rho \overset{\text{def}}{=} \mathsf{measure}_{i=1}^n p_i \mapsto s_i$. By the rule $(\eta\mathrm{nrm})$, it is sufficient to prove that $t = \mathsf{do}\ \_ \leftarrow t; \mathsf{return}\ \rho$. We have

$$\begin{aligned}\mathsf{do}\ \_ \leftarrow t; \mathsf{return}\ \rho &= \mathsf{measure}\ p_1 \mathbin{\&} q \mapsto \mathsf{return}\ \rho \mid \cdots \mid p_n \mathbin{\&} q \mapsto \mathsf{return}\ \rho \mid q^\perp \mapsto \mathsf{fail} \\ &= \mathsf{measure}\ q \mapsto \mathsf{return}\ \rho \mid q^\perp \mapsto \mathsf{fail} \\ &= \mathsf{measure}_{i=1}^n\ q \mathbin{\&} p_i \mapsto \mathsf{return}\ s_i \mid q^\perp \mapsto \mathsf{fail} = t\end{aligned}$$

(We used the commutativity of & in the last step.) ◀

▶ **Corollary 27.** *Let α_1, …, α_n, β be rational numbers that sum to 1, with $\beta \neq 1$. If*

$$\vdash t = \mathsf{measure}\ \alpha_1 \mapsto \mathsf{return}\ s_1 \mid \cdots \mid \alpha_n \mapsto \mathsf{return}\ s_n \mid \beta \mapsto \mathsf{fail} : A + 1 \ \ , \ \ \textit{then}$$
$$\vdash \mathsf{nrm}\,(t) = \mathsf{measure}\ \alpha_1/(\alpha_1 + \cdots + \alpha_n) \mapsto s_1 \mid \cdots \mid \alpha_n/(\alpha_1 + \cdots + \alpha_n) \mapsto s_n : A \ .$$

6 Conclusion

The system **COMET** allows for the specification of probabilistic programs and reasoning about their properties, both within the same syntax.

There are several avenues for further work and research.

- The type theory that we describe can be interpreted both in discrete and in continuous probabilistic models, that is, both in the Kleisli category $\mathcal{K}\ell(\mathcal{D})$ of the distribution monad $\mathcal{D}$ and in the Kleisli category $\mathcal{K}\ell(\mathcal{G})$ of the Giry monad $\mathcal{G}$. On a finite type each distribution is discrete. The discrete semantics were exploited in the current paper in the examples in Section 3. In a follow-up version we intend to elaborate also continuous examples.

- The normalisation and conditioning that we use in this paper can in principle also be used in a quantum context, using the appropriate (non-side-effect free) assert maps that one has there. This will give a form of Bayesian quantum theory, as also explored in [14].
- A further ambitious follow-up project is to develop tool support for **COMET**, so that the computations that we carry out here by hand can be automated. This will provide a formal language for Bayesian inference.

Acknowledgements Thanks to Kenta Cho for discussion and suggestions during the writing of this paper, and very detailed proofreading. Thanks to Bas Westerbaan for discussions about effectus theory.

References

1 Robin Adams. QPEL: Quantum programming and effect language. In *11th Workshop on Quantum Physics and Logic*, pages 133–153, 2014.

2 Nick Benton, Gavin Bierman, Valeria De Paiva, and Martin Hyland. A term calculus for intuitionistic linear logic. In *TLCA*, volume 664 of *Lecture Annotes in Computer Science*, pages 75–90. Springer-Verlag, 1993.

3 Johannes Borgström, Andrew D. Gordon, Michael Greenberg, James Margetson, and Jurgen van Gael. Measure transformer semantics for Bayesian machine learning. In *ESOP'11/ETAPS'11 Proceedings of the 20th European conference on Programming languages and systems*, pages 77–96, 2011.

4 K. Cho. Total and partial computation in categorical quantum foundations. In C. Heunen, P. Selinger, and J. Vicary, editors, *Quantum Physics and Logic (QPL) 2015*, number 195 in Elect. Proc. in Theor. Comp. Sci., pages 116–135, 2015.

5 K. Cho, B. Jacobs, A. Westerbaan, and B. Westerbaan. Quotient comprehension chains. In C. Heunen, P. Selinger, and J. Vicary, editors, *Quantum Physics and Logic (QPL) 2015*, number 195 in Elect. Proc. in Theor. Comp. Sci., pages 136–147, 2015.

6 Kenta Cho, Bart Jacobs, Bas Westerbaan, and Bram Westerbaan. An introduction to effectus theory. Unpublished.

7 B. Jacobs. Measurable spaces and their effect logic. In *Logic in Computer Science.* IEEE, Computer Science Press, 2013.

8 B. Jacobs. New directions in categorical logic, for classical, probabilistic and quantum logic. *Logical Methods in Comp. Sci.*, 11(3):1–76, 2015.

9 B. Jacobs, B. Westerbaan, and A. Westerbaan. States of convex sets. In A. Pitts, editor, *Foundations of Software Science and Computation Structures*, number 9034 in Lect. Notes Comp. Sci., pages 87–101. Springer, Berlin, 2015.

10 Bart Jacobs. New directions in categorical logic, for classical, probabilistic and quantum logic. arXiv:1205.3940, 2014. URL: `http://arxiv.org/abs/1205.3940`.

11 C. Jones and G. Plotkin. A probabilistic powerdomain of evaluations. In *Logic in Computer Science*, pages 186–195. IEEE, Computer Science Press, 1989.

12 D. Kozen. Semantics of probabilistic programs. *Journ. Comp. Syst. Sci*, 22(3):328–350, 1981.

13 D. Kozen. A probabilistic PDL. *Journ. Comp. Syst. Sci*, 30(2):162–178, 1985.

14 Matthew S Leifer and Robert W Spekkens. Towards a formulation of quantum theory as a causally neutral theory of bayesian inference. *Physical Review A*, 88(5):052130, 2013.

15 C. Morgan, A. McIver, and K. Seidel. Probabilistic predicate transformers. *ACM Trans. on Progr. Lang. and Systems*, 18(3):325–353, 1996.

16 S. Russel and P. Norvig. *Artificial Intelligence. A Modern Approach.* Prentice Hall, 2003.

17 R. Tix, K. Keimel, and G. Plotkin. *Semantic Domains for Combining Probability and Non-Determinism.* Number 129 in Elect. Notes in Theor. Comp. Sci. Elsevier, Amsterdam, 2005.

18 E. S. Yudkowsky. An intuitive explanation of Bayesian reasoning. Available at `http://yudkowsky.net/rational/bayes`, 2003.

A Formal Presentation of COMET

The full set of rules of deduction for **COMET** are given below.

A.1 Structural Rules

$$(\text{exch})\ \frac{\Gamma, x : A, y : B, \Delta \vdash \mathcal{J}}{\Gamma, y : B, x : A, \Delta \vdash \mathcal{J}} \qquad (\text{var})\ \frac{x : A \in \Gamma}{\Gamma \vdash x : A}$$

$$(\text{ref})\ \frac{\Gamma \vdash t : A}{\Gamma \vdash t = t : A} \quad (\text{sym})\ \frac{\Gamma \vdash s = t : A}{\Gamma \vdash t = s : A} \quad (\text{trans})\ \frac{\Gamma \vdash r = s : A \qquad \Gamma \vdash s = t : A}{\Gamma \vdash r = t : A}$$

A.2 The Unit Type

$$(\text{unit})\ \frac{}{\Gamma \vdash * : 1} \qquad (\eta 1)\ \frac{\Gamma \vdash t : 1}{\Gamma \vdash t = * : 1}$$

A.3 Tensor Product

$$(\otimes)\ \frac{\Gamma \vdash s : A \qquad \Delta \vdash t : B}{\Gamma, \Delta \vdash s \otimes t : A \otimes B} \quad (\text{lett})\ \frac{\Gamma \vdash s : A \otimes B \qquad \Delta, x : A, y : B \vdash t : C}{\Gamma, \Delta \vdash \mathsf{let}\ x \otimes y = s\ \mathsf{in}\ t : C}$$

$$(\text{paireq})\ \frac{\Gamma \vdash s = s' : A \qquad \Delta \vdash t = t' : B}{\Gamma, \Delta \vdash s \otimes t = s' \otimes t' : A \otimes B}$$

$$(\text{leteq})\ \frac{\Gamma \vdash s = s' : A \otimes B \qquad \Delta, x : A, y : B \vdash t = t' : C}{\Gamma, \Delta \vdash (\mathsf{let}\ x \otimes y = s\ \mathsf{in}\ t) = (\mathsf{let}\ x \otimes y = s'\ \mathsf{in}\ t') : C}$$

$$(\beta\otimes)\ \frac{\Gamma \vdash r : A \qquad \Delta \vdash s : B \qquad \Theta, x : A, y : B \vdash t : C}{\Gamma, \Delta, \Theta \vdash (\mathsf{let}\ x \otimes y = r \otimes s\ \mathsf{in}\ t) = t[x := r, y := s] : C}$$

$$(\eta\otimes)\ \frac{\Gamma \vdash t : A \otimes B}{\Gamma \vdash t = (\mathsf{let}\ x \otimes y = t\ \mathsf{in}\ x \otimes y) : A \otimes B}$$

$$(\text{let-let})\ \frac{\Gamma \vdash r : A \otimes B \qquad \Delta, x : A, y : B \vdash s : C \otimes D \qquad \Theta, z : C, w : D \vdash t : E}{\begin{array}{l}\Gamma, \Delta, \Theta \vdash\ \mathsf{let}\ x \otimes y = r\ \mathsf{in}\ (\mathsf{let}\ z \otimes w = s\ \mathsf{in}\ t) \\ \qquad\qquad = \mathsf{let}\ z \otimes w = (\mathsf{let}\ x \otimes y = r\ \mathsf{in}\ s)\ \mathsf{in}\ t : E\end{array}}$$

$$(\text{let-}\otimes)\ \frac{\Gamma \vdash r : A \otimes B \qquad \Delta, x : A, y : B \vdash s : C \qquad \Theta \vdash t : D}{\Gamma, \Delta, \Theta \vdash \mathsf{let}\ x \otimes y = r\ \mathsf{in}\ (s \otimes t) = (\mathsf{let}\ x \otimes y = r\ \mathsf{in}\ s) \otimes t : D}$$

A.4 Empty Type

$$(\text{magic})\ \frac{\Gamma \vdash t : 0}{\Gamma \vdash\ ¡t : A} \qquad (\eta 0)\ \frac{\Gamma \vdash s : 0 \qquad \Gamma \vdash t : A}{\Gamma \vdash\ ¡s = t : A}$$

A.5 Binary Coproducts

$$(\text{inl})\ \frac{\Gamma \vdash t : A}{\Gamma \vdash \mathsf{inl}\,(t) : A+B} \qquad (\text{inr})\ \frac{\Gamma \vdash t : B}{\Gamma \vdash \mathsf{inr}\,(t) : A+B}$$

$$(\text{inl-eq})\ \frac{\Gamma \vdash t = t' : A}{\Gamma \vdash \mathsf{inl}\,(t) = \mathsf{inl}\,(t') : A+B} \qquad (\text{inr-eq})\ \frac{\Gamma \vdash t = t' : B}{\Gamma \vdash \mathsf{inr}\,(t) = \mathsf{inr}\,(t') : A+B}$$

$$(\text{case})\ \frac{\Gamma \vdash r : A+B \qquad \Delta, x : A \vdash s : C \qquad \Delta, y : B \vdash t : C}{\Gamma, \Delta \vdash \mathsf{case}\ r\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto s \mid \mathsf{inr}\,(y) \mapsto t : C}$$

$$(\text{case-eq})\ \frac{\Gamma \vdash r = r' : A+B \qquad \Delta, x : A \vdash s = s' : C \qquad \Delta, y : B \vdash t = t' : C}{\Gamma, \Delta \vdash \mathsf{case}\ r\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto s \mid \mathsf{inr}\,(y) \mapsto t = \mathsf{case}\ r'\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto s' \mid \mathsf{inr}\,(y) \mapsto t' : C}$$

$$(\beta{+}_1)\ \frac{\Gamma \vdash r : A \qquad \Delta, x : A \vdash s : C \qquad \Delta, y : B \vdash t : C}{\Gamma, \Delta \vdash \mathsf{case}\ \mathsf{inl}\,(r)\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto s \mid \mathsf{inr}\,(y) \mapsto t = s[x := r] : C}$$

$$(\beta{+}_2)\ \frac{\Gamma \vdash r : B \qquad \Delta, x : A \vdash s : C \qquad \Delta, y : B \vdash t : C}{\Gamma, \Delta \vdash \mathsf{case}\ \mathsf{inr}\,(r)\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto s \mid \mathsf{inr}\,(y) \mapsto t = t[y := r] : C}$$

$$(\eta{+})\ \frac{\Gamma \vdash t : A+B}{\Gamma \vdash t = \mathsf{case}\ t\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto \mathsf{inl}\,(x) \mid \mathsf{inr}\,(y) \mapsto \mathsf{inr}\,(y) : A+B}$$

$$(\text{case-case})\ \frac{\begin{array}{c}\Gamma \vdash r : A+B\ \ \Delta, x : A \vdash s : C+D\ \ \Delta, y : B \vdash s' : C+D \\ \Theta, z : C \vdash t : E \qquad \Theta, w : D \vdash t' : E\end{array}}{\begin{array}{rl}\Gamma, \Delta, \Theta \vdash & \mathsf{case}\ r\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto \mathsf{case}\ s\ \mathsf{of}\ \mathsf{inl}\,(z) \mapsto t \mid \mathsf{inr}\,(w) \mapsto t' \mid \\ & \quad \mathsf{inr}\,(y) \mapsto \mathsf{case}\ s'\ \mathsf{of}\ \mathsf{inl}\,(z) \mapsto t \mid \mathsf{inr}\,(w) \mapsto t' \\ & = \mathsf{case}\ (\mathsf{case}\ r\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto s \mid \mathsf{inr}\,(y) \mapsto s') \\ & \quad \mathsf{of}\ \mathsf{inl}\,(z) \mapsto t \mid \mathsf{inr}\,(w) \mapsto t' : E\end{array}}$$

$$(\text{case-}\otimes)\ \frac{\Gamma \vdash r : A+B \qquad \Delta, x : A \vdash s : C \qquad \Delta, y : B \vdash s' : C \qquad \Theta \vdash t : D}{\begin{array}{rl}\Gamma, \Delta, \Theta \vdash & (\mathsf{case}\ r\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto s \mid \mathsf{inr}\,(y) \mapsto s') \otimes t = \\ & \mathsf{case}\ r\ \mathsf{of}\ \mathsf{inl}\,(x) \mapsto s \otimes t \mid \mathsf{inr}\,(y) \mapsto s' \otimes t : C \otimes D\end{array}}$$

$$(\text{let-case})\ \frac{\begin{array}{c}\Gamma \vdash r : A+B \qquad \Delta, z : A \vdash s : C \otimes D \\ \Delta, w : B \vdash s' : C \otimes D\ \ \Theta, x : C, y : D \vdash t : E\end{array}}{\begin{array}{rl}\Gamma, \Delta, \Theta \vdash & \mathsf{let}\ x \otimes y = \mathsf{case}\ r\ \mathsf{of}\ \mathsf{inl}\,(z) \mapsto s \mid \mathsf{inr}\,(w) \mapsto s'\ \mathsf{in}\ t = \\ & \mathsf{case}\ r\ \mathsf{of}\ \mathsf{inl}\,(z) \mapsto \mathsf{let}\ x \otimes y = s\ \mathsf{in}\ t \mid \mathsf{inr}\,(w) \mapsto \mathsf{let}\ x \otimes y = s'\ \mathsf{in}\ t : E\end{array}}$$

A.6 Partial Pairing

$$(\text{inlr})\ \frac{\Gamma \vdash s : A+1 \qquad \Gamma \vdash t : B+1 \qquad \Gamma \vdash s \downarrow = t \uparrow : \mathbf{2}}{\Gamma \vdash «s, t» : A+B}$$

$$(\text{inlr-eq})\ \frac{\Gamma \vdash s = s' : A+1 \qquad \Gamma \vdash t = t' : B+1 \qquad \Gamma \vdash s \downarrow = t \uparrow : \mathbf{2}}{\Gamma \vdash «s, t» = «s', t'» : A+B}$$

$$(\beta\text{inlr}_1)\ \frac{\Gamma \vdash s : A+1 \qquad \Gamma \vdash t : B+1 \qquad \Gamma \vdash s \downarrow = t \uparrow : \mathbf{2}}{\Gamma \vdash \rhd_1(«s, t») = s : A+1}$$

$$(\beta\text{inlr}_1)\ \frac{\Gamma \vdash s : A+1 \qquad \Gamma \vdash t : B+1 \qquad \Gamma \vdash s \downarrow = t \uparrow : \mathbf{2}}{\Gamma \vdash \rhd_2(«s, t») = t : B+1}$$

$$(\eta\text{inlr})\ \frac{\Gamma \vdash t : A+B}{\Gamma \vdash t = «\rhd_1(t), \rhd_2(t)» : A+B}$$

A.7 The $\mathsf{left}\,()$ Construction

$$(\mathrm{left})\ \frac{\Gamma \vdash t : A + B \qquad \Gamma \vdash \mathsf{inl?}\,(t) = \top : \mathbf{2}}{\Gamma \vdash \mathsf{left}\,(t) : A}$$

$$(\mathrm{left\text{-}eq})\ \frac{\Gamma \vdash t = t' : A + B \qquad \Gamma \vdash \mathsf{inl?}\,(t) = \top : \mathbf{2}}{\Gamma \vdash \mathsf{left}\,(t) = \mathsf{left}\,(t') : A}$$

$$(\beta\mathrm{left})\ \frac{\Gamma \vdash t : A + B \qquad \Gamma \vdash \mathsf{inl?}\,(t) = \top : \mathbf{2}}{\Gamma \vdash \mathsf{inl}\,(\mathsf{left}\,(t)) = t : A + B} \quad (\eta\mathrm{left})\ \frac{\Gamma \vdash t : A}{\Gamma \vdash \mathsf{left}\,(\mathsf{inl}\,(t)) = t : A}$$

A.8 Instruments

$$(\mathrm{instr})\ \frac{x : A \vdash t : \mathbf{n} \qquad \Gamma \vdash s : A}{\Gamma \vdash \mathsf{instr}_{\lambda x t}(s) : n \cdot A} \quad (\nabla\text{-}\mathrm{instr})\ \frac{x : A \vdash t : \mathbf{n} \qquad \Gamma \vdash s : A}{\Gamma \vdash \nabla(\mathsf{instr}_{\lambda x t}(s)) = s : A}$$

$$(\mathrm{instr\text{-}test})\ \frac{x : A \vdash t : \mathbf{n} \qquad \Gamma \vdash s : A}{\Gamma \vdash \mathsf{case}\ {}_{i=1}^{n}\mathsf{instr}_{\lambda x t}(s)\ \mathsf{of}\ \mathsf{in}_i^n\,(\_) \mapsto i = t[x := s] : \mathbf{n}}$$

$$(\eta\mathrm{instr})\ \frac{x : A \vdash r : n \cdot A \qquad x : A \vdash \nabla(r) = x : A \qquad \Gamma \vdash s : A}{\Gamma \vdash \mathsf{instr}_{\lambda x.\mathsf{case}\ {}_{i=1}^{n} r\ \mathsf{of}\ \mathsf{in}_i^n(\_) \mapsto i}(s) = r[x := s] : n \cdot A}$$

$$(\mathrm{instr\text{-}eq})\ \frac{x : A \vdash t = t' : \mathbf{n} \qquad \Gamma \vdash s = s' : A}{\Gamma \vdash \mathsf{instr}_{\lambda x t}(s) = \mathsf{instr}_{\lambda x t'}(s') : n \cdot A}$$

A.9 Scalar Constants

For any natural number $n \geq 2$, we have the following rules.

$$(1/n)\ \frac{}{\Gamma \vdash 1/n : \mathbf{2}} \quad (n \cdot 1/n)\ \frac{}{\Gamma \vdash n \cdot 1/n = \top : \mathbf{2}}$$

$$(\mathrm{divide})\ \frac{\Gamma \vdash n \cdot t = \top : \mathbf{2}}{\Gamma \vdash t = 1/n : \mathbf{2}} \quad (b_{mn})\ \frac{}{\Gamma \vdash b_{mn} : \mathbf{3}}\ (1 \leq m < n)$$

$$(\rhd_1 - b_{mn})\ \frac{}{\Gamma \vdash \mathsf{do}\ x \leftarrow b_{mn}; \rhd_1(x) = m \cdot 1/n : \mathbf{2}}\ (1 \leq m < n)$$

$$(\rhd_2 - b_{mn})\ \frac{}{\Gamma \vdash \mathsf{do}\ x \leftarrow b_{mn}; \mathsf{return}\ \nabla(x) = 1/n : \mathbf{2}}\ (1 \leq m < n)$$

A.10 Normalisation

$$(\mathrm{nrm})\ \frac{\vdash t : A + 1 \qquad \vdash 1/n \leq t\downarrow : \mathbf{2}}{\Gamma \vdash \mathsf{nrm}\,(t) : A} \quad (\beta\mathrm{nrm})\ \frac{\vdash t : A + 1 \qquad \vdash 1/n \leq t\downarrow : \mathbf{2}}{\Gamma \vdash t = \mathsf{do}\ \_ \leftarrow t; \mathsf{return}\ \mathsf{nrm}\,(t) : A + 1}$$

$$(\eta\mathrm{nrm})\ \frac{\vdash t : A + 1 \qquad \vdash 1/n \leq t\downarrow : \mathbf{2} \qquad \vdash \rho : A \qquad \vdash t = \mathsf{do}\ \_ \leftarrow t; \mathsf{return}\ \rho : A + 1}{\Gamma \vdash \rho = \mathsf{nrm}\,(t) : A}$$

A.11 Partial Sum

$$(\ovee)\ \frac{\begin{array}{cc} \Gamma \vdash s : A + 1 & \Gamma \vdash t : A + 1 \\ \Gamma \vdash b : (A + A) + 1 & \Gamma \vdash \mathsf{do}\ x \leftarrow b; \rhd_1(x) = s : A + 1 \\ \multicolumn{2}{c}{\Gamma \vdash \mathsf{do}\ x \leftarrow b; \rhd_2(x) = t : A + 1} \end{array}}{\Gamma \vdash s \ovee t : A + 1}$$

$$(\ovee\text{-}\mathrm{def})\ \frac{\begin{array}{cc} \Gamma \vdash s : A + 1 & \Gamma \vdash t : A + 1 \\ \Gamma \vdash b : (A + A) + 1 & \Gamma \vdash \mathsf{do}\ x \leftarrow b; \rhd_1(x) = s : A + 1 \\ \multicolumn{2}{c}{\Gamma \vdash \mathsf{do}\ x \leftarrow b; \rhd_2(x) = t : A + 1} \end{array}}{\Gamma \vdash s \ovee t = \mathsf{do}\ x \leftarrow b; \mathsf{return}\ \nabla(x) : A + 1}$$

A.12 Miscellaneous

$$\text{(JM)}\ \frac{\Gamma \vdash s : (A+A)+1 \qquad \Gamma \vdash t : (A+A)+1 \qquad \Gamma \vdash s \gg\!= \rhd_1 = t \gg\!= \rhd_1 : A+1 \quad \Gamma \vdash s \gg\!= \rhd_2 = t \gg\!= \rhd_2 : A+1}{\Gamma \vdash s = t : (A+A)+1}$$

$$\text{(comm)}\ \frac{x : A \vdash p : \mathbf{2} \qquad x : A \vdash q : \mathbf{2} \qquad \Gamma \vdash t : A}{\Gamma \vdash \ \mathsf{assert}_{\lambda x p}(t) \gg\!= \mathsf{assert}_{\lambda x q} = \mathsf{assert}_{\lambda x q}(t) \gg\!= \mathsf{assert}_{\lambda x p} : A+1}$$