Submitted to MFPS 2024

Early announcement: Compositional imprecise probability

Jack Liell-Cock Sam Staton

University of Oxford, Department of Computer Science

Abstract

Imprecise probability is concerned with uncertainty about which probability distributions to use. It has applications in robust statistics and elsewhere. Imprecise probability can be modelled in various ways, such as by convex sets of probability distributions.

We look at programming language models for imprecise probability. Our desiderata are that we would like our model to support all kinds of composition, categorical and monoidal; in other words, guided by dataflow diagrams. Another equivalent perspective is that we would like a model of synthetic probability in the sense of Markov categories.

The leading monad-based approach to imprecise probability is not fully compositional because the monad involved is not commutative, which means that we do not have a proper monoidal structure. In this work, we provide a new fully compositional account. The key idea is to name the non-deterministic choices. To manage the renamings and disjointness of names, we use graded monads. We show that the resulting compositional model is maximal and relate it with the earlier monad approach, proving that we obtain tighter bounds on the uncertainty.

For further details, please see our preprint [48].

1 Overview

This paper is about using programming language notations to give compositional descriptions of imprecise probability. For illustration, consider a situation with three outcomes: red (r), green (g) and blue (b). A precise probability distribution can be understood as a point in the triangle: the corner (r) represents 100% certainty of red; the points on the edge between g and b represent the probability distributions where r is impossible (Figure 1a).

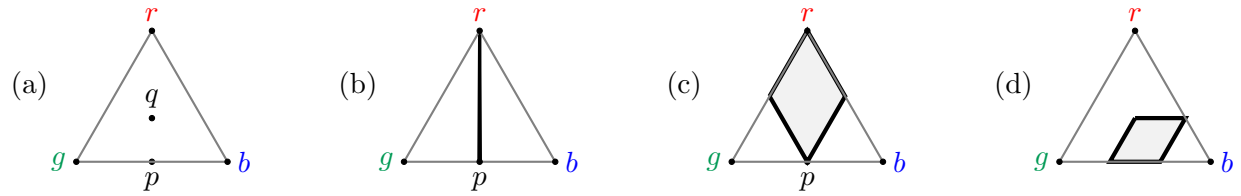


Fig. 1. (a) Five probabilities over the three-point set $\{r, g, b\}$ illustrated as points in the triangle: the three extreme points are the corners; p is the equal odds chance between b and g; q is the equal odds chance between all three points. (b) A line indicating a convex region between r and p, which includes q. (c) A convex region which is the convex hull of four points, including r, p and also the equal odds chance between r and b and between r and g. (d) A different convex region, considered in [70, Ex. 7.3].

An *imprecise probability* on three outcomes is a convex region of the triangle (Figure 1b–1d). One interpretation is that if a probability distribution describes a bet, as in the foundations of Bayesianism,

then a convex region is a collection of bets that would be reasonable given the current imprecise knowledge. Imprecise probability has a long history in terms of statistical robustness (e.g. [27,70]), recently considered as part of infrabayesianism (e.g. [2,3,40]) and the foundations of safe AI [17].

There is already a body of work on semantics models of programming languages with imprecise probability [1,22,23,24,28,36,37,50,51,52,53,54,57,68,69]. Our contribution is to investigate new models that support our compositional desiderata (§2) by naming the non-deterministic choices (§3). We show that this compares favourably with earlier work (Thm. 1, §4) and that it is a maximal approach (Thm. 2).

2 Desiderata: a language for imprecise probability with compositional reasoning

A first language for describing imprecise probability is a first-order functional language without recursion. Rather, we have if/then/else statements, sequencing with immutable variable assignment (like [45, 55]), and the following two commands, which both return a boolean value:

- **bernoulli**: a fair Bernoulli choice [5] which draws a ball from some urn containing two balls labelled 'true' and 'false', and replaces it;
- **knight**: a Knightian choice [38] which draws a ball from a fresh urn containing balls labelled 'true' and 'false', where the number and ratio of balls are unknown and we have no priors on their distribution, except to know that the urn is not empty. (These 'Knightian urns' are fresh each time – they can each be used only once. That is, we are not interested in using multiple draws and frequencies to predict their contents.)

For example, consider the following two programs.

Example 2.1 The following program, we argue, describes the convex region in Figure 1b:

$x \leftarrow$ **knight** ; $z \leftarrow$ **bernoulli** ;
if z then (if x then return r else return g)
else (if x then return r else return b)

We draw two boolean values, x and z, respectively with Knightian uncertainty and from a fair Bernoulli trial. We then combine these two boolean values using the logic on the second and third lines of the program.

Example 2.2 The following program describes the convex region in Figure 1c:

$x \leftarrow$ **knight** ; $y \leftarrow$ **knight** ; $z \leftarrow$ **bernoulli** ;
if z then (if x then return r else return g)
else (if y then return r else return b)

This time, we draw three boolean values, x, y and z, where y is with Knightian uncertainty too. We then combine these three boolean values using the logic on the second and third lines of the program, which is almost the same except for the use of y when z is false. Decoupling the Knightian uncertainties increases the region of imprecise probability because it allows new outcomes (such as an equal chance between r and b when x is true and y is false) that were impossible in Example 2.1.

Our desiderata for a *compositional account* of a first-order language are the following. We are inspired by recent compositional accounts of probability theory (e.g. [19, 29, 39]), statistics (e.g. [9, 32, 43]), and probabilistic programming (e.g. [14,33,61]), and the connections between them (e.g. [64]). These desiderata are formalized in [48, §2].

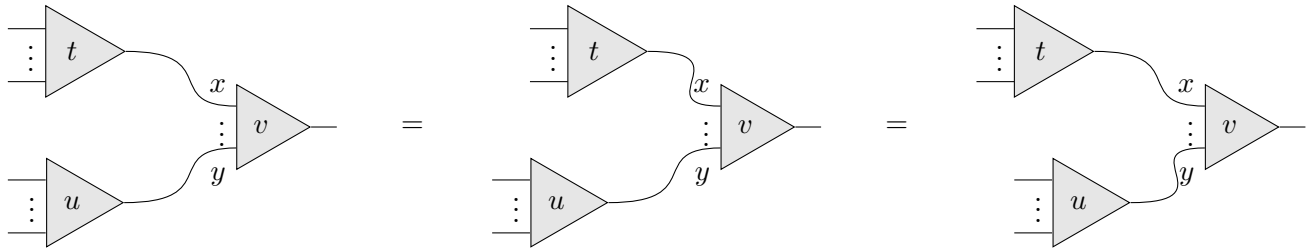
Desideratum 1 *The language should be commutative:*

$$x \leftarrow t\ ;\ y \leftarrow u\ ;\ v \quad = \quad y \leftarrow u\ ;\ x \leftarrow t\ ;\ v \qquad \textit{(if } x \textit{ is not free in } u \textit{ and } y \textit{ not free in } t\textit{)}$$

and affine:

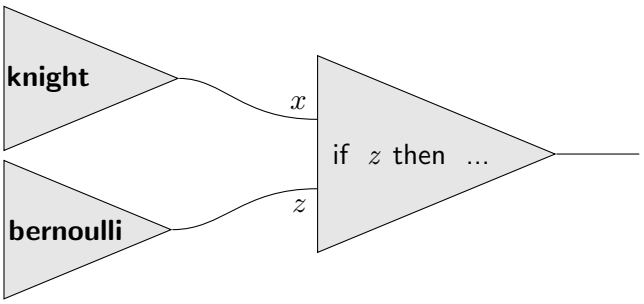
$$(x \leftarrow t\ ;\ u) \quad = \quad u \qquad \textit{(if } x \textit{ is not free in } u\textit{).}$$

This means that we can regard composition graphically, as a data flow graph. For instance, the notation



is not ambiguous.

Although this requirement does not hold generally in the presence of memory side effects and mutable variables, we do not have mutable variables here, and it is desirable in a declarative language. For example, we would like to notate the program from Example 2.1 as



Desideratum 2 *The standard equational reasoning about if/then/else should apply, and in particular the following hoisting equation should be allowed:*

$$\mathsf{if}\ b\ \mathsf{then}\ (x \leftarrow t\ ;\ u)\ \mathsf{else}\ (x \leftarrow t\ ;\ v) \quad = \quad x \leftarrow t\ ;\ \mathsf{if}\ b\ \mathsf{then}\ u\ \mathsf{else}\ v$$

where x is not free in b.

One earlier approach to a semantic study of a language like this is provided by a convex powerset of distributions monad (e.g. [6, 7, 24, 28, 37, 52, 53]). This does not satisfy the desiderata for compositional reasoning. In fact, no semantic model satisfying the desiderata can allow Examples 2.1 and 2.2 to be distinguished, as we show in Figure 2. The key issue is with the third program in Figure 2:

$$\begin{array}{l} z \leftarrow \textbf{bernoulli}\ ;\ \mathsf{if}\ z\ \mathsf{then}\ (x \leftarrow \textbf{knight}\ ;\ \mathsf{if}\ x\ \mathsf{then}\ \mathsf{return}\ \ r\ \ \mathsf{else}\ \ \mathsf{return}\ \ g\,) \\ \qquad\qquad\qquad\quad \mathsf{else}\ (x \leftarrow \textbf{knight}\ ;\ \mathsf{if}\ x\ \mathsf{then}\ \mathsf{return}\ \ r\ \ \mathsf{else}\ \ \mathsf{return}\ \ b\,) \end{array}$$

This program draws a boolean value with Knightian uncertainty on each of the branches of the if statement. The paradox arises in whether each choice comes from different urns or the same urn. Perhaps there is one Knightian draw that is used in both branches. Or perhaps we draw a boolean value from a new Knightian urn on the second branch. Our proposed solution is to make this distinction explicit.

3 Resolution: named Knightian choices

To satisfy both desiderata, **our proposal** is to name each Knightian choice [48, §3]. To do this, we rewrite Example 2.1 by annotating the only Knightian choice with the name a_1:

$$\begin{array}{l} x \leftarrow \textbf{knight}(a_1)\ ;\ z \leftarrow \textbf{bernoulli}\ ; \\ \mathsf{if}\ \ z\ \mathsf{then}\ (\ \mathsf{if}\ \ x\ \mathsf{then}\ \mathsf{return}\ \ r\ \ \mathsf{else}\ \ \mathsf{return}\ \ g\,)\ \ \mathsf{else}\ \ (\ \mathsf{if}\ \ x\ \mathsf{then}\ \mathsf{return}\ \ r\ \ \mathsf{else}\ \ \mathsf{return}\ \ b\,) \end{array}$$

We think of this program as giving rise to the convex set in Figure 1(b). This is then the same as the program where y describes the outcome of the same Knightian choice, i.e. one with the same name:

$x \leftarrow$ **knight** ; $z \leftarrow$ **bernoulli** ;
if z then (if x then return r else return g) else (if x then return r else return b)
= −− *Desideratum 1 (commutativity)*
$z \leftarrow$ **bernoulli** ; $x \leftarrow$ **knight** ;
if z then (if x then return r else return g) else (if x then return r else return b)
= −− *Desideratum 2*
$z \leftarrow$ **bernoulli** ; if z then ($x \leftarrow$ **knight** ; if x then return r else return g)
else ($x \leftarrow$ **knight** ; if x then return r else return b)
= −− *Alpha renaming*
$z \leftarrow$ **bernoulli** ; if z then ($x \leftarrow$ **knight** ; if x then return r else return g)
else ($y \leftarrow$ **knight** ; if y then return r else return b)
= −− *Desideratum 1 (affine)*
$z \leftarrow$ **bernoulli** ; if z then ($x \leftarrow$ **knight** ; $y \leftarrow$ **knight** ; if x then return r else return g)
else ($x \leftarrow$ **knight** ; $y \leftarrow$ **knight** ; if y then return r else return b)
= −− *Desideratum 2*
$z \leftarrow$ **bernoulli** ; $x \leftarrow$ **knight** ; $y \leftarrow$ **knight** ;
if z then (if x then return r else return g) else (if y then return r else return b)
= −− *Desideratum 1 (commutativity)*
$x \leftarrow$ **knight** ; $y \leftarrow$ **knight** ; $z \leftarrow$ **bernoulli** ;
if z then (if x then return r else return g) else (if y then return r else return b)

Fig. 2. An equational derivation that Examples 2.1 and 2.2 must be equal if Desiderata 1 and 2 are satisfied.

$$x \leftarrow \textbf{knight}(a_1) \,;\, y \leftarrow \textbf{knight}(a_1) \,;\, z \leftarrow \textbf{bernoulli} \,;\; \text{if } z \text{ then } (\text{if } x \text{ then return } r \text{ else return } g) \text{ else } (\text{if } y \text{ then return } r \text{ else return } b)$$

but it is different from the program where y describes a different Knightian choice, i.e. one with a different name:

$$x \leftarrow \textbf{knight}(a_1) \,;\, y \leftarrow \textbf{knight}(a_2) \,;\, z \leftarrow \textbf{bernoulli} \,;\; \text{if } z \text{ then } (\text{if } x \text{ then return } r \text{ else return } g) \text{ else } (\text{if } y \text{ then return } r \text{ else return } b)$$

which is intuitively what Example 2.2 describes, and gives rise to the convex set in Figure 1(c). Now when we try to follow the same equational derivation as in Figure 2, the third program becomes:

$$z \leftarrow \textbf{bernoulli} \,;\; \text{if } z \text{ then } (x \leftarrow \textbf{knight}(a_1) \,;\, \text{if } x \text{ then return } r \text{ else return } g) \text{ else } (x \leftarrow \textbf{knight}(a_1) \,;\, \text{if } x \text{ then return } r \text{ else return } b)$$

which conveys the same Knightian value is used on each of the branches of the if statement. This can no longer derive the program:

$$z \leftarrow \textbf{bernoulli} \,;\; \text{if } z \text{ then } (x \leftarrow \textbf{knight}(a_1) \,;\, \text{if } x \text{ then return } r \text{ else return } g) \text{ else } (x \leftarrow \textbf{knight}(a_2) \,;\, \text{if } x \text{ then return } r \text{ else return } b)$$

which explicitly uses a different Knightian choice on the else branch.

The idea of naming non-deterministic choices appears in work outside probability (e.g. proved transitions, [8]) and probabilistic choices are often named in practical probabilistic programming [67, §6.2] which has already been explored using graded monads [46]. More generally, intensionality in non-determinism is known to be a profitable perspective (e.g. [10, 42]).

3.1 Named Knightian choices via a reader monad

The set-up with named Knightian choices is consistent with Desiderata 1 and 2, which we can show by building a monad (e.g. [55]), namely the reader transformer (e.g. [47]) of the finite distributions monad (e.g. [31, Ch. 2]):

$$T_{2^A}(X) = [2^A \Rightarrow D(X)] \tag{1}$$

where X is the set of outcomes, A is the set of names required, and D is the finite distributions monad. Then the Knightian choices are interpreted by reading, and the Bernoulli choices use the distributions monad. This combined monad is well known to be commutative and affine. Thus both desiderata are satisfied.

We can recover a convex set of probability distributions from any $t \in T_{2^A}(X)$ by pushing forward all

the possible probability distributions on 2^A. Formally, we can express this using the monadic bind ($\gg=_D$, Kleisli composition) of D:

$$[\![t]\!]_{2^A} = \{p \gg=_D t \mid p \in D(2^A)\} \subseteq D(X).$$

3.2 Grading to account for renamings

A remaining concern with named Knightian choices is that we ought to take seriously name-space issues in composition. When composing programs with named Knightian choices, we may wish to avoid name clashes. This is dependent on how we interpret the set A in (1).

We resolve this issue by regarding the monad (1) as a graded monad [34,35,56]. This is closely related to the 'para' construction (e.g. [18,25]). Some of the crucial steps are as follows:

- Any injection $\iota : A \to B$ induces a renaming of programs using names A to programs using names B, and indeed a natural map $T_{2^\iota} : T_{2^A}(X) \to T_{2^B}(X)$;
- We can regard monadic bind (Kleisli composition) in T as operating on distinct sets of names:

$$\gg=_T \; : \; T_{2^A}(X) \times (X \Rightarrow T_{2^B}(Y)) \to T_{2^{A \uplus B}}(Y)$$

  Thus a computation using names A is sequenced with a computation using names B to build a computation that involves names $(A \uplus B)$.
- This monad is graded-monoidal too, via a map

$$T_{2^A}(X) \times T_{2^B}(Y) \to T_{2^{(A \uplus B)}}(X \times Y)$$

  which juxtaposes computations using names A and B to give a computation using $(A \uplus B)$.
- The induced convex set of distributions is invariant under renaming: $[\![t]\!]_{2^A} = [\![T_{2^\iota}(t)]\!]_{2^B}$.

The crucial element is that the injective renaming ι induces a surjection $2^\iota : 2^B \to 2^A$ between the spaces of Knightian choices. We abstract and generalize by allowing arbitrary surjections $2^B \to 2^A$, further by allowing sets other than 2^A, and further still by allowing surjective stochastic maps rather than surjections.

As an aside, we note that probability monads too can often be regarded as sort-of reader monads (e.g. [4, 14, 59, 60, 65, 66]), since probability distributions $D(X)$ can be described by random variables $\Omega \to X$, for some base probability sample space Ω. Thus we could regard our monad $T(X)$ as a quotient of

$$[(\Omega \times \Xi) \Rightarrow X]$$

where Ω is a sample space for Bernoulli probability and $\Xi = 2^A$ is a sample space for Knightian uncertainty. In this work, we will quotient by the 'law' of random variables in Ω, so that the usual equational reasoning about Bernoulli probability is valid.

4 Results about quotienting our theory

The names for the Knightian choices in our language appear to be additional intensional information, and the reader monad does not quotient this away. For this reason, we show two results about the equational theory. First, we connect our approach to the convex powerset of distributions monad, showing that our bounds are tighter. Second, we show it is maximal — no further quotient is possible.

Theorem 1 [48, §4]: Improved bounds on uncertainty

In our resulting language, every closed term describes a convex set of distributions. We thus establish a connection to the non-compositional approach that uses the Kleisli category of the convex powerset of distributions monad (e.g. [6,7,24,28,37,52,53]). We have an 'op-lax' functor

$$R : \mathbf{ImP} \to \mathrm{Kl}(\mathrm{CP}).$$

from our locally graded category **ImP** [48, §3.2] to the Kleisli category of the convex powerset of distributions monad CP. Being an op-lax functor means that

$$R(g \circ f) \subseteq R(g) \circ R(f),$$

i.e. composition in our category gives a tighter bound on the Knightian uncertainty than the composition using the Kleisli category of the convex powerset of distributions monad (see also the Example [48, §4.4]).

Note that this could not be a proper functor because we would then have a quotient theory in violation of the maximality theorem (Theorem 2). But an op-lax functor is beneficial as an interpretation of giving a tighter bound.

Theorem 2 [48, §5]: Maximality.
Our language also gives rise to a compositional theory of equality. We prove our equational theory is maximal in that we can add no further equations on open terms without equating different convex sets of distributions or compromising the compositional structure.

Further detail: two quotients

In slightly more detail, we consider two candidates from the literature for quotients of a graded monad. They are general methods, but appear in our language as follows. Notice that an open term in our language contains both names and variables: names for Knightian choices, and free variables standing for ordinary values that might be substituted later. There are two ways to quotient the names away:

Quotient A Following [20,21] we could equate two open terms if, for every valuation of the free variables, there is a renaming that equates them.

Quotient B Following [25], we could equate two open terms if there is a renaming such that for every valuation of the free variables they are made equal.

For closed terms with no free variables, the two approaches are the same and give rise to a convex set of distributions [48, Prop. 4.3].

Quotient A does not directly give a compositional theory in our setting: the criteria of [20] are not satisfied. Nonetheless, the construction of [20] can be adjusted, giving the op-lax functor of Theorem 1 rather than a monad morphism.

Quotient B does not satisfy Desideratum 2 (commuting if-then-else). Informally, it would allow us to rename on the 'then' branch but not the 'else' branch, which is inconsistent with Desideratum 2. Nonetheless, it could be a useful approach in a metalanguage for combining models that do not need a general if-then-else construction. For this reason, Quotient B is not a counterexample to Theorem 2.

5 Notes about context

Other situations combining probability and non-determinism.

Our focus here is on imprecise probability. This is one form of non-determinism, but there is a broader body of general interest in non-determinism and its combination with probability. Of course, non-determinism can arise in many different semantic situations, beyond the motivation from imprecise probability. One motivation is in program abstraction and refinement: there, one describes a problem by writing a non-deterministic program that solves it; one then solves the problem by refining that non-deterministic program. The mathematical analysis is similar, and for instance illustrations essentially the same as Figure 1 appear in work on refinement of probabilistic programs [49, Fig. 6.4.2, Fig. 6.5.1]. But the motivation is different, and the desiderata (§2) may be less relevant in program refinement.

An arguably different kind of non-determinism appears where there are many appropriate results that we want to collect. In this sense, for instance, database queries are non-deterministic if they return multiple results, and Prolog is non-deterministic. When combined with probability, this leads more naturally to random sets or random bags, which are in contrast to the sets of distributions shown in Figure 1. Random bags do arise in probabilistic databases and point process theory. We looked at these applications from a monad perspective here [15,16], and the monads have long been discussed (e.g. [12,13,30,37,41,69]).

Work on distributive laws.
There is a large literature on finding elegant explanations for combining existing monads for probability and non-determinism, exploring distributive laws (e.g. [7,11,12,24,41,57,69]). Although the reader monad transformer is a distributive law, our emphasis here is on the commutativity desiderata (§2) rather than distributivity issues. Even when there is a distributive law between commutative monads, the resulting composite monad need not be commutative. Indeed both the random bags monad (e.g. [13]) and the powerdomain of indexed valuations monad [69] arise from distributive laws between commutative monads, but neither composite monads are commutative.

Algebraic perspective.
Some of this work on distributive laws takes the perspective of algebraic theories. Our desiderata can be viewed from the point of view of algebraic theories, via algebraic effects (e.g. [58]), which we now briefly explore. We define two binary operations:

$$(t +_{0.5} u) \stackrel{\text{def}}{=} \mathsf{if}\ \ \textbf{bernoulli}\ \ \mathsf{then}\ \ t\ \ \mathsf{else}\ \ u \qquad (t \vee u) \stackrel{\text{def}}{=} \mathsf{if}\ \ \textbf{knight}\ \ \mathsf{then}\ \ t\ \ \mathsf{else}\ \ u$$

Regarding Desideratum 1, Commutativity means that each operation is a homomorphism for the other:

$$\begin{aligned}(s \vee t) +_{0.5} (u \vee v) &= (s +_{0.5} u) \vee (t +_{0.5} v)\\ (s +_{0.5} t) +_{0.5} (u +_{0.5} v) &= (s +_{0.5} u) +_{0.5} (t +_{0.5} v)\\ (s \vee t) \vee (u \vee v) &= (s \vee u) \vee (t \vee v)\end{aligned}$$

and affinity says that $t \vee t = t$ and $t +_{0.5} t = t$. Desideratum 2 is always assumed in algebraic effects. From these five axioms, the derivation of Figure 2 can be made algebraically:

$$r \vee (g +_{0.5} b) = (r +_{0.5} r) \vee (g +_{0.5} b) = (r \vee g) +_{0.5} (r \vee b).$$

If we regard $\vee$ as a Minkowski sum (see [48, §4.3]) then the left hand side appears to be Fig. 1(b) and the right hand side is Fig. 1(c), especially since the latter can be further rearranged using the commutativity and affinity laws to

$$(r \vee (g +_{0.5} r)) \vee ((r +_{0.5} b) \vee (g +_{0.5} b))$$

which enumerates the four extreme points of Fig. 1(c). From this algebraic perspective, we propose to label the Knightian branching, with a family of binary operators $\vee_{a_1}, \vee_{a_2}, \ldots$, and then we still have that

$$r \vee_{a_1} (g +_{0.5} b) = (r +_{0.5} r) \vee_{a_1} (g +_{0.5} b) = (r \vee_{a_1} g) +_{0.5} (r \vee_{a_1} b)$$

yet it is consistent to assume

$$(r \vee_{a_1} g) +_{0.5} (r \vee_{a_1} b) \neq (r \vee_{a_1} g) +_{0.5} (r \vee_{a_2} b).$$

with different names on the right-hand side, corresponding to the difference between Figures 1(b) and 1(c).

Our intuition is that $t \vee_a u$ branches left or right depending on the Knightian draw a. Here a is describing the unique draw from that urn. We note that a labelled binary choice already appears in the probabilistic setting in [62], where it has a different meaning: there, a stands for an urn but not a specific draw, and $?_a$ denotes sampling from urn a according to Polya's scheme (replace with two copies of what was drawn).

We note that it is already known that Desiderata 1 is incompatible with symmetry laws ($t \vee u = u \vee t$ and $t +_{0.5} u = u +_{0.5} t$), since we can use those to deduce

$$t \vee u = (t \vee u) +_{0.5} (t \vee u) = (t \vee u) +_{0.5} (u \vee t) = (t +_{0.5} u) \vee (u +_{0.5} t) = (t +_{0.5} u) \vee (t +_{0.5} u) = t +_{0.5} u \quad (2)$$

In our graded semantics (**ImP**), we have $t +_{0.5} u = u +_{0.5} t$. We do *not*, however, have $t \vee u = u \vee t$, although there is a reindexing in the grading that corresponds to this symmetry. One view is that we have side-stepped this Eckmann-Hilton-like obstacle (2) by putting the symmetry $t \vee u \approx u \vee t$ into the grading structure.

6 Summary and outlook

We have shown that by taking a graded perspective and naming Knightian choices we can obtain a compositional account of Bernoulli and Knightian uncertainty together. The account gives a refined bound on the uncertainty (Theorem 1) and is maximal among the compositional accounts (Theorem 2).

There are several future directions. An initial question is how to accommodate iteration. The convex sets considered in this article are all finitely generated, but if we allow iterative programs that have an unbounded number of Knightian choices, this leads to a more general class of convex sets.

The concerns about iteration hold even if we restrict to finite outcome spaces, and thus far we have focused on this for simplicity. Much work on programming semantics for imprecise probability has focused beyond finite outcome spaces, and it will be interesting to revisit this from our perspective: this includes domain theoretic structures (e.g. [23, 36, 37, 69]) and metric structures (e.g. [52, 53]).

It would be interesting to compare to another recent compositional framework combining unknowns with probability by Stein and Samuelson, currently focusing on Gaussians [63].

Our approach is based on random elements, and so is the quasi-Borel-space probability monad (e.g. [26, 66]), so this might be a good approach to accommodating function spaces.

On the more practical side, an open question is how to perform statistical inference in a probabilistic programming language with imprecise probability.

Going beyond statistics, there may be other scenarios where this approach is useful: making a theory compositional by using a graded theory (for a first purely speculative example, the issues with amb outlined in [44]).

Acknowledgements

It has been helpful to talk to many people about this work. The 2023/2024 Aria workshops emphasised aspects of imprecise probability as discussion topics (and so it was helpful to discuss with davidad, Dylan Braithwaite, Elena Di Lavore, Alex Lew, Owen Lynch, Sean Moss, Zenna Tavares, and others). It has been helpful to discuss with colleagues in Oxford (particularly Paolo Perrone, who is running an adjoint school project on uncertainty in Markov categories, and some time ago Kwok Ho Cheung regarding his thesis work [11]), and at the 2024 Bellairs workshop. We also received helpful feedback from Hugo Paquet, Dario Stein, and MFPS 2024 reviewers.

Research supported by Clarendon Scholarship, ERC Consolidator Grant BLAST, and AFOSR Project FA9550-21-1-0038.

References

[1] A. Adjé, O. Bouissou, J. Goubault-Larrecq, E. Goubault, and S. Putot. Static analysis of programs with imprecise probabilistic inputs. In *Proc. VSTTE 2013*, pages 22–47, 2013.

[2] A. Appel and V. Kosoy. Basic inframeasure theory. LessWrong, August 2020. https://www.lesswrong.com/posts/YAa4qcMyoucRS2Ykr/basic-inframeasure-theory.

[3] A. Appel and V. Kosoy. Inframeasures and domain theory. LessWrong, March 2021. https://www.lesswrong.com/posts/vrbidMiczaoHBhZGp/inframeasures-and-domain-theory.

[4] T. Barker. A monad for randomized algorithms. In *Proc. MFPS 2016*, 2016.

[5] J. Bernoulli. *Ars conjectandi, opus posthumum.* Thurneysen, 1713.

[6] F. Bonchi, A. Sokolova, and V. Vignudelli. The theory of traces for systems with nondeterminism and probability. In *Proc. LICS 2019*, 2019.

[7] F. Bonchi, A. Sokolova, and V. Vignudelli. Presenting convex sets of probability distributions by convex semilattices and unique bases. In *Proc. CALCO 2021*, CALCO 2021.

[8] G. Boudol and I. Castellani. Flow models of distributed computations: three equivalent semantics for CCS. *Inform. Comput.*, 114:247–314, 1994.

[9] D. Braithwaite, J. Hedges, and T. St Clere Smithe. The compositional structure of Bayesian inference. In *Proc. MFCS 2023*, 2023.

[10] G. L. Cattani, M. P. Fiore, and G. Winskel. A theory of recursive domains with applications to concurrency. In *Proc. LICS 1998*, 1998.

[11] K.-H. Cheung. *Distributive Interaction of Algebraic Effects*. PhD thesis, University of Oxford Department of Computer Science, 2017.

[12] F. Dahlqvist, L. Parlant, and A. Silva. Layer by layer: composing monads. In *Proc. ICTAC 2018*, 2018.

[13] S. Dash. *A Monadic Theory of Point Processes*. PhD thesis, University of Oxford, 2024.

[14] S. Dash, Y. Kaddar, H. Paquet, and S. Staton. Affine monads and lazy structures for Bayesian programming. In *Proc. POPL 2023*, 2023.

[15] S. Dash and S. Staton. A monad for probabilistic point processes. In *Proc. ACT 2020*, 2020.

[16] S. Dash and S. Staton. Monads for measurable queries in probabilistic databases. In *Proc. MFPS 2021*, 2021.

[17] D. 'davidad' Dalrymple. Safeguarded AI: constructing safety by design. ARIA Programme Thesis, January 2024. https://www.aria.org.uk/wp-content/uploads/2024/01/ARIA-Safeguarded-AI-Programme-Thesis-V1.pdf.

[18] B. Fong, D. Spivak, and R. Tuyéras. Backprop as functor: A compositional perspective on supervised learning. In *Proceedings of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '19. IEEE Press, 2021.

[19] T. Fritz. A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics. *Adv. Math.*, 370, 2020.

[20] T. Fritz and P. Perrone. A criterion for Kan extensions of lax monoidal functors. arxiv:1809.10481, 2018.

[21] T. Fritz and P. Perrone. A probability monad as the colimit of spaces of finite samples. *Theory and Applications of Categories*, 34, 2019.

[22] J. Goubault-Larrecq. Continuous previsions. In *Proc. CSL 2007*, 2007.

[23] J. Goubault-Larrecq. Prevision domains and convex powercones. In *Proc. FOSSACS 2008*, 2008.

[24] A. Goy and D. Petrisan. Combining probabilistic and non-deterministic choice via weak distributive laws. In *LICS 2020*, 2020.

[25] C. Hermida and R. Tennent. Monoidal indeterminates and categories of possible worlds. *Theoretical Computer Science*, 430, 2012.

[26] C. Heunen, O. Kammar, S. Staton, and H. Yang. A convenient category for higher-order probability theory. In *Proc. LICS 2017*, 2017.

[27] P. J. Huber. *Robust statistics*. Wiley, 1981.

[28] B. Jacobs. Coalgebraic trace semantics for combined possibilitistic and probabilistic systems. In *Proc. CMCS 2008*, 2008.

[29] B. Jacobs. From probability monads to commutative effectuses. *J. Log. Algebr. Methods Program.*, 94:200–237, 2018.

[30] B. Jacobs. From multisets over distributions to distributions over multisets. In *Proc. LICS 2021*, 2021.

[31] B. Jacobs. Structured probabilistic reasoning. Available from the author's homepage, July 2023. Draft book.

[32] B. Jacobs, A. Kissinger, and F. Zanasi. Causal inference by string diagram surgery. In *Proc. FOSSACS 2019*, 2019.

[33] X. Jia, B. Lindenhovius, M. W. Mislove, and V. Zamdzhiev. Commutative monads for probabilistic programming languages. In *Proc. LICS 2021*, pages 1–14, 2021.

[34] O. Kammar and G. D. Plotkin. Algebraic foundations for effect-dependent optimisations. In *Proc. POPL 2012*, pages 349–360, 2012.

[35] S. Katsumata. Parametric effect monads and semantics of effect systems. In *Proc. POPL 2014*, 2014.

[36] K. Keimel. Topological cones: Foundations for a domain theoretical semantics combining probability and nondeterminism. In *MFPS 2005*, 2005.

[37] K. Keimel and G. D. Plotkin. Mixed powerdomains for probability and nondeterminism. *Log. Methods Comput. Sci.*, 13, 2017.

[38] F. H. Knight. *Risk, uncertainty and profit*. Houghton Mifflin, 1921.

[39] A. Kock. Commutative monads as a theory of distributions. *Theory Appl. Categ.*, 26(4):97–131, 2012.

[40] V. Kosoy and A. Appel. Infra-Bayesian physicalism: A formal theory of naturalized induction. AI Alignment Forum, November 2021. Available at https://www.alignmentforum.org/posts/gHgs2e2J5azvGFatb/infra-bayesian-physicalism-a-formal-theory-of-naturalized.

[41] D. Kozen and A. Silva. Multisets and distributions. arxiv:2301.10812, 2023.

[42] J. Laird, G. Manzonetto, G. McCusker, and M. Pagani. Weighted relational models of typed lambda-calculi. In *Proc. LICS 2013*, 2013.

[43] E. D. Lavore and M. Román. Evidential decision theory via partial Markov categories. In *Proc. LICS 2023*, 2023.

[44] P. B. Levy. Amb breaks well-pointedness, ground amb doesn't. In *Proc. MFPS 2007*, 2007.

[45] P. B. Levy, J. Power, and H. Thielecke. Modelling environments in call-by-value programming languages. *Inform. Comput.*, 185, 2003.

[46] A. K. Lew, M. F. Cusumano-Towner, B. Sherman, M. Carbin, and V. K. Maninghka. Trace types and denotational semantics for sound programmable inference in probabilistic languages. In *Proc. POPL 2020*, 2020.

[47] S. Liang, P. Hudak, and M. Jones. Monad transformers and modular interpreters. In *Proc. POPL 1995*, 1995.

[48] J. Liell-Cock and S. Staton. Compositional imprecise probability. https://arxiv.org/abs/2405.09391, May 2024.

[49] A. McIver and C. Morgan. *Abstraction, Refinement and Proof for Probabilistic Systems*. Springer, 2005.

[50] A. K. McIver and C. Morgan. Partial correctness for probabilistic demonic programs. *Theoret. Comput. Sci.*, 266:513–541, 2001.

[51] M. Mio. Upper-expectation bisimilarity and Łukasiewicz mu-calculus. In *Proc. FoSSaCS 2014*, 2014.

[52] M. Mio, R. Sarkis, and V. Vignudelli. Combining nondeterminism, probability, and termination: Equational and metric reasoning. In *Proc. LICS 2021*, 2021.

[53] M. Mio and V. Vignudelli. Monads and quantitative equational theories for nondeterminism and probability. In *Proc. CONCUR 2020*, 2020.

[54] M. W. Mislove, J. Ouaknine, and J. Worrell. Axioms for probability and nondeterminism. In *Proc. EXPRESS 2003*, 2003.

[55] E. Moggi. Notions of computation and monads. *Information and Computation*, 1991.

[56] D. Orchard, P. Wadler, and H. E. III. Unifying graded and parameterised monads. In *Proc. MSFP 2020*, 2020.

[57] D. Petrisan and R. Sarkis. Semialgebras and weak distributive laws. In *Proc. MFPS 2021*, 2021.

[58] G. D. Plotkin and J. Power. Notions of computation determine monads. In *Proc. FOSSACS 2002*, 2002.

[59] D. Shiebler. Categorical Stochastic Processes and Likelihood. *Compositionality*, 3, Apr. 2021.

[60] A. Simpson. Probability sheaves and the Giry monad. In *Proc. CALCO 2017*, 2017.

[61] S. Staton. Commutative semantics for probabilistic programming. In *Proc. ESOP 2017*, pages 855–879, 2017.

[62] S. Staton, D. Stein, H. Yang, N. L. Ackerman, C. E. Freer, and D. M. Roy. The Beta-Bernoulli process and algebraic effects. In *Proc. ICALP 2018*, 2018.

[63] D. Stein and R. Samuelson. A categorical treatment of open linear systems, March 2024. arxiv:2403.03934.

[64] D. Stein and S. Staton. Compositional semantics for probabilistic programs with exact conditioning. In *Proc. LICS 2021*, 2021.

[65] Z. Tavares, J. Koppel, X. Zhang, R. Das, and A. Solar-Lezama. A language for counterfactual generative models. In *Proc. ICML 2021*, pages 10173–10182, 2021.

[66] M. Vákár, O. Kammar, and S. Staton. A domain theory for statistical probabilistic programming. *Proc. ACM Program. Lang.*, 3(POPL):36:1–36:29, 2019.

[67] J.-W. van de Meent, B. Paige, H. Yang, and F. Wood. An introduction to probabilistic programming. *arXiv e-print 1809.10756*, 2018.

[68] G. van Heerdt, J. Hsu, J. Ouaknine, and A. Silva. Convex language semantics for nondeterministic probabilistic automata. In *Proc. ICTAC 2018*, 2018.

[69] D. Varacca and G. Winskel. Distributing probability over non-determinism. *Math. Structures Comput. Sci.*, pages 87–113, 2006.

[70] P. Walley. *Statistical Reasoning with Imprecise Probabilities*. Chapman and Hall, 1991.