A Domain Theory for Statistical Probabilistic Programming

MATTHIJS VÁKÁR, Columbia University, USA
OHAD KAMMAR, University of Oxford, UK
SAM STATON, University of Oxford, UK

We give an adequate denotational semantics for languages with recursive higher-order types, continuous probability distributions, and soft constraints. These are expressive languages for building Bayesian models of the kinds used in computational statistics and machine learning. Among them are untyped languages, similar to Church and WebPPL, because our semantics allows recursive mixed-variance datatypes. Our semantics justifies important program equivalences including commutativity.

Our new semantic model is based on 'quasi-Borel predomains'. These are a mixture of chain-complete partial orders (cpos) and quasi-Borel spaces. Quasi-Borel spaces are a recent model of probability theory that focuses on sets of admissible random elements. Probability is traditionally treated in cpo models using probabilistic powerdomains, but these are not known to be commutative on any class of cpos with higher order functions. By contrast, quasi-Borel predomains do support both a commutative probabilistic powerdomain and higher-order functions. As we show, quasi-Borel predomains form both a model of Fiore's axiomatic domain theory and a model of Kock's synthetic measure theory.

CCS Concepts: • **Theory of computation → Probabilistic computation**; **Bayesian analysis**; **Denotational semantics**; • **Software and its engineering → Language types**; *Functional languages*; *Interpreters*; Domain specific languages; • **Computing methodologies** → *Machine learning*;

Additional Key Words and Phrases: denotational semantics, domain theory, probability, recursion, adequacy

ACM Reference Format:
Matthijs Vákár, Ohad Kammar, and Sam Staton. 2019. A Domain Theory for Statistical Probabilistic Programming. *Proc. ACM Program. Lang.* 3, POPL, Article 36 (January 2019), 35 pages. https://doi.org/10.1145/3290349

1 INTRODUCTION

The idea of statistical probabilistic programming is to use a programming language to specify statistical models and inference problems. It enables rapidly prototyping different models, because: (1) the model specification is separated from the technicalities of the inference/simulation algorithms; and (2) software engineering/programming techniques can be used to manage the complexity of statistical models. Here, we focus on the fundamental programming technique of *recursion*. We consider both recursive terms — looping — and recursive types, e.g., streams and untyped languages.

In a traditional programming language, recursion has long been analyzed using semantic domains based on ω-complete partial orders. The suitability of this approach is given by the adequacy theorem, which connects the compositional interpretation in domains with an operational interpretation. However, there are long standing open problems regarding using these domains with probability and measure. Here we sidestep these problems by introducing a new notion: *ω-quasi Borel spaces*. We look at a language with both statistical constructs and higher-order recursive types and terms, which is close to languages used in practice for statistical probabilistic programming, and we show the following adequacy theorem:

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
© 2019 Copyright held by the owner/author(s).
2475-1421/2019/1-ART36
https://doi.org/10.1145/3290349

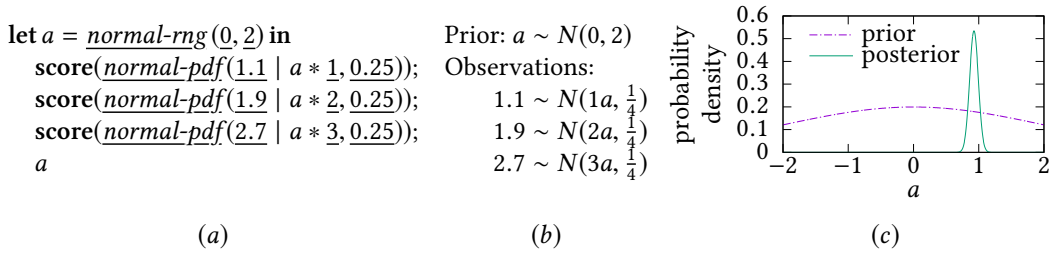


Fig. 1. Bayesian linear regression as (a) a first-order probabilistic program, (b) an informal specification, and (c) a plot of the prior and posterior distributions. Here $N(\mu, \sigma)$ is the normal (Gaussian) distribution with density $\mathit{normal\text{-}pdf}(x \mid \mu, \sigma) = (2\pi\sigma^2)^{-\frac{1}{2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$ and random number generator $\mathit{normal\text{-}rng}(\mu, \sigma)$.

Theorem (6.9: Adequacy). *If two programs are equal in the ωqbs model then they are contextually equivalent with regard to a Monte-Carlo operational semantics.*

Domain theoretic semantics can verify compositional compiler optimizations. As an example, the following reordering transformation is valid, since it is readily verifiable in the ωqbses.

Corollary (6.11: Commutativity). *The following two programs are contextually equivalent:*

$$\begin{array}{l}\textbf{let}\ x = t\ \textbf{in} \\ \textbf{let}\ y = s\ \textbf{in}\ r\end{array} \quad \approx \quad \begin{array}{l}\textbf{let}\ y = s\ \textbf{in} \\ \textbf{let}\ x = t\ \textbf{in}\ r\end{array}$$

This property says there is no implicit sequential state in the language. It is essential for Shan and Ramsey's disintegration-based exact Bayesian inference technique [2017], implemented in the Hakaru system [Narayanan et al. 2016]. The corollary is related to Fubini's theorem for reordering integrals: informally, $\int \mathrm{d}x \int \mathrm{d}y\, r(x,y) = \int \mathrm{d}y \int \mathrm{d}x\, r(x,y)$. The important novelty here is that our semantic model extends this commutativity theorem to higher-order and recursive types, even if they do not fit easily into traditional measure theory.

1.1 Introduction to Statistical Probabilistic Programming

We introduce statistical probabilistic programming through a simple example of a regression problem, in Fig. 1. The problem is: supposing that there is a linear function $x \mapsto ax$ and three noisy measurements $(1, 1.1)$, $(2, 1.9)$ and $(3, 2.7)$ of it with postulated noise scale 0.25, find a posterior distribution on the slope a. As indicated, first-order probabilistic programs can be thought of as a direct translation of a Bayesian statistical problem. The probabilistic program has an operational reading in terms of Monte-Carlo simulation: first use a Gaussian sampler/random number generator, using *normal-rng*, to draw from the prior with mean 0 and standard deviation 2; then weight, using **score**, the resulting samples with respect to the three data points according to the Gaussian likelihood, using *normal-pdf*, assuming noisy measurements with standard deviation $\frac{1}{4}$. The resulting program represents the *unnormalised* posterior distribution, which can then be passed to an inference algorithm to approximate its normalisation, e.g. through Monte-Carlo sampling.

A case for recursion: higher-order functions over infinite data structures. Many probabilistic programming languages also allow other programming language features, including recursion. When looking at a whole closed program of ground type, these extra features pose little conceptual problem because the entire program will reduce to a first-order program, albeit a very large or even

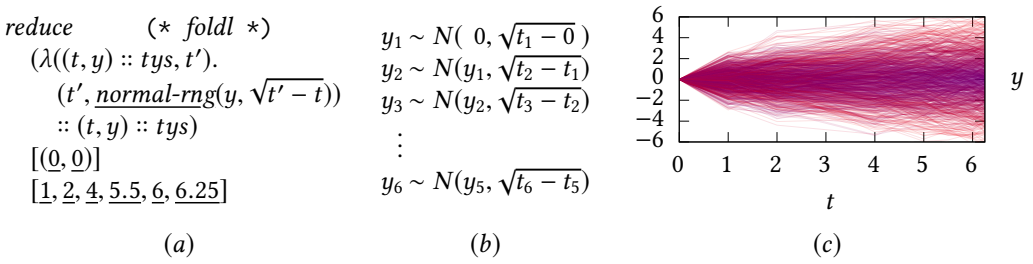


Fig. 2. A Gaussian random walk as (a) a higher-order recursive program and (b) an informal specification, with (c) some samples. The function *reduce* is sometimes called *fold left*.

infinite one. However, this paper is not concerned with the problem of interpreting whole closed programs, but rather interpreting individual aspects of a program in a compositional way.

For example, consider the program in Fig. 2, which takes snapshots of a Gaussian random walk at a stream of times (t) to produce a stream of (t, y) co-ordinate pairs. The meaning of the whole program is clear, and can be reduced to the first-order statistical model (b). But supposing this appears as part of a bigger model, we would like to understand each part separately. What is the mathematical meaning of *reduce* here? It takes a parameterized random operation, an initial value, and a stream, and produces a random stream. By providing mathematical objects that represent these recursive concepts, we can understand *reduce* as a first-class construct and reason about it.

A case for recursion: untyped programs. Rather than distinguish between ground types and higher-order recursive types, an alternative approach is to combine the full *untyped* lambda calculus with the constructions of the simple statistical programming language (Fig. 7). Many probabilistic programming languages take this approach (Church [Goodman et al. 2008], Anglican [Wood et al. 2014], WebPPL [Goodman and Stuhlmüller 2014], Venture [Mansinghka et al. 2014]). Recall that we can express untyped calculi using recursive types, using a single recursive type like $\Lambda = (\mathbb{1} \mid \Lambda * \Lambda \mid \cdots \mid \Lambda \to \Lambda)$. Thus a language with recursive types can be thought of as generalising this untyped situation.

1.2 Summary of Semantics in ω-Quasi Borel Spaces

The usual method for interpreting a programming language is as follows:

- types denote spaces (in an untyped language, there is just one universal space);
- closed programs denote points in a space; and
- program phrases denote functions assigning a point to every valuation of their free variables.

Probabilistic programs, on the other hand, vary this by saying

- closed programs denote *measures* (or distributions) on a space; and
- program phrases denote kernels between spaces.

Recursive probabilistic programming has a tension between what a space is and what it is used for:

- In traditional domain theory, a type denotes a topological space or cpo in which continuity and convergence model recursion using fixed points.
- For first-order probabilistic programs, a type denotes a topological or measurable space whose purpose is to support measures and expectations.

It is tempting to use a single topological (or cpo) structure to interpret both the probability and recursion. By contrast with existing mainstream approaches [Jones and Plotkin 1989], however,

we choose to keep both structures separate but compatible, through the following observation. In probability theory it is widely acknowledged that topological and measurable structure are only a precursor to the notion of random element. Recall that a random element in a set P is a function $\alpha : \Omega \to P$ where Ω is some space of random seeds, e.g. $\Omega = \mathbb{R}$ with a Lebesgue measure. When $P = \mathbb{R}$, one would typically ask that α be measurable so that there is an expected (average, mean) value of α. But random elements are relevant beyond their expectation. A measure on P can be understood as a random element modulo being 'equal in distribution', but it is helpful to also keep the distinction between random element and measure. This leads us to the following definition.

Definition (3.5). *An* ωqbs *comprises a set* P *together with the following structure:*

- *a partial order* $\leq$ *on* P *such that limits of* ω*-chains exist: to model recursion;*
- *a set* M *of functions* $\alpha : \mathbb{R} \to P$*: these are thought of as* random elements*, to interpret probability;*

all subject to some compatibility conditions.

For example:

(1) Let P be the set of subsets of $\mathbb{N}$. This is a space to interpret deterministic programs with a natural number argument: a subset $X \subseteq \mathbb{N}$ represents the program that returns for each member in X. The order $\leq$ is the inclusion order, so program X is below Y if X diverges whenever Y diverges. The random elements M are the functions $\alpha : \mathbb{R} \to P$ such that $\alpha^{-1}[\{S \mid n \in S\}]$ is (Borel) measurable for all $n \in \mathbb{N}$.
(2) Let $\mathbb{R}$ be the set of real *values*. When thought of as values, we use the discrete order. The random elements M are the measurable functions $\mathbb{R} \to \mathbb{R}$.
(3) Let $\mathbb{W} = [0, \infty]$ be the non-negative extended reals. Its elements stand for computation *weights* with the linear order. The random elements M are the measurable functions $\mathbb{R} \to [0, \infty]$.

The semantics in ωqbses supports higher-order functions. Moreover, we can interpret recursive types by using the recipe of Fiore and Plotkin's axiomatic domain theory [1994]. That is to say:

Theorem (Cor. 3.10, §5.3). *The category* ω**Qbs** *has products, sums, function spaces, and a bilimit compact expansion (sufficient structure to interpret recursive types).*

1.3 A Probabilistic Powerdomain

With higher-order functions and recursive types dealt with, the remaining ingredient is measures (and probabilistic programs that generate measures) as first-class constructions. To this end, for every ωqbs P we will associate an ωqbs $T(P)$ of measures on P. Following Moggi [1989], we turn T into a monad encapsulating the probabilistic aspects of the programming language.

Recall that an ωqbs P comes with a set M of random elements, viz. functions $\alpha : \mathbb{R} \to P$. Because a statistical probabilistic program naturally describes an unnormalized posterior measure, we consider the basic space $\mathbb{R}$ with the full Lebesgue measure, which has infinite total measure. To consider finite measures we consider partial random elements, which are given by pairs (α, D) where $\alpha \in M$ and $D \subseteq \mathbb{R}$ is Borel. Given any partial random element (α, D) and any morphism $f : P \to \mathbb{W}$ ($\mathbb{W} = [0, \infty]$), the composite $f \circ \alpha$ is measurable, so we can find an expectation for f:

$$\mathbb{E}_{(\alpha,D)}[f] := \int_D f(\alpha(x))\ \mathrm{d}x. \tag{1}$$

We say that (α, D) and (α', D') are *equivalent* when they give the same expectation operator:

$$\text{for all } f : P \to \mathbb{W}, \quad \int_D f(\alpha(x))\ \mathrm{d}x = \int_{D'} f(\alpha'(x))\ \mathrm{d}x. \tag{2}$$

Definition. *A measure on an ωqbs P is an equivalence class of a partial random element (α, D), modulo the equivalence relation (2). Equivalently, a measure is a morphism $\mathbb{W}^P \to \mathbb{W}$ of the form $\mathbb{E}_{(\alpha,D)}$ for some partial random element (α, D).*

Here we run into a technical problem: the set of all measures has a natural (pointwise) partial order structure but this set might not be closed under suprema of ω-chains. On the other hand, we know that $J(P) := \mathbb{W}^{(\mathbb{W}^P)}$ is always closed, because J is the continuation monad which makes sense in any category with function spaces. Thus we take the closure $T(P)$ of the set of measures in $J(P)$ as our space of measures. In other words, $T(P) \subseteq J(P)$ contains those expectation operators that arise as iterated suprema of ω-chains on P.

Our approach follows existing continuation-passing-style techniques [e.g. Keimel and Plotkin 2009; Kiselyov and Shan 2009; Olmedo et al. 2016]. CPS semantics is analogous to working with the full continuation monad J or a fragment of it. This fragment must be chosen carefully, or else the commutativity property fails in the model. Indeed J also has constants such as $exit_r = \lambda k.r \in J(P)$ violating the commutativity equation: put $t_1 = exit_1$, $t_2 = exit_2$. As we have the commutativity property, these constants lie outside our monad $T(P)$, hence are not definable in the language.

Aside on the Jung-Tix problem. A long standing problem in traditional domain theory is to find a category of Scott domains that is closed under function spaces and a commutative probabilistic powerdomain [Jung and Tix 1998]. This remains an open problem. We side-step this problem by using ωqbses instead of Scott domains. They inherit many of the properties and intuitions of ω-cpos, are closed under function spaces, and support a commutative probabilistic powerdomain. We summarize further work in this direction in Sec. 8.

1.4 Summary

We have provided a domain theory for recursion in statistical probabilistic programming. The main contributions of this work are the following novel constructions:

(1) a Cartesian closed category of (pre)-domains (Sec. 3), that admits the solution of recursive domain equations (Sec. 5),
(2) a commutative probabilistic power-domain (Sec. 4);
(3) an adequate denotational model (Sec. 6) for probabilistic programming with recursive types (Sec. 2), and in consequence
   - an adequate denotational model for a higher-order language with sampling from continuous distributions, term recursion and soft constraints (§2.4);
   - an adequate denotational model for untyped probabilistic programming (§2.3).

2 CALCULI FOR STATISTICAL PROBABILISTIC PROGRAMMING

We consider three call-by-value calculi for statistical probabilistic programming. The main calculus, Statistical FPC (SFPC) is a statistical variant of Fiore and Plotkin's Fixed-Point Calculus (FPC) [1994]. SFPC has sum, product, function, and recursive types, as well as a ground type $\underline{\mathbb{R}}$ of real numbers, constants $\underline{f}$ for all measurable functions $f : \mathbb{R}^n \to \mathbb{R}$, a construct $\mathbf{match}\ -\ \mathbf{with}\ \{\underline{0} \to - \mid \_ \to -\}$ for testing real numbers for zero, a construct **sample** for drawing a random number using the uniform (Lebesgue) measure $\mathbb{U}_{[0,1]}$ on $[0, 1]$ and a construct **score** for reweighting program traces (to implement soft constraints). We express the other two calculi as fragments of SFPC. The first, Idealised Church, is an untyped λ-calculus for statistical probabilistic programming based on the Church modelling language [Wingate et al. 2011]. The second, the Call-by-Value Statistical PCF (CBV SPCF), is a call-by-value variant of Plotkin's [1977] and Scott's [1993] simply typed λ-calculus with higher-order term recursion, extended with statistical primitives.

2.1 Preliminaries: Borel Measurability

We need the following fundamentals of measure theory. The *Borel subsets* of the real line $\mathbb{R}$ are given inductively by taking every interval $[a,b]$ to be a Borel subset, and closing under complements and countable unions. More generally, for every natural number n, the Borel subsets of $\mathbb{R}^n$ are given inductively by taking every n-dimensional box $[a_1,b_1]\times\cdots\times[a_n,b_n]$ to be a Borel subset and closing under complements and countable unions. A (partial) function $f:\mathbb{R}^n \rightharpoonup \mathbb{R}$ is *Borel-measurable* when its inverse image maps every Borel subset $B\subseteq\mathbb{R}$ to a Borel subset $f^{-1}[B]\subseteq\mathbb{R}^n$. The set of Borel-measurable functions contains, for example, all the elementary functions.

A *measure* μ on $\mathbb{R}^n$ is an assignment of possibly-infinite, non-negative real values $\mu(B)$ to every Borel subset $B\subseteq\mathbb{R}^n$, assigning 0 to the empty set $\mu(\emptyset)=0$, linear on disjoint unions $\mu(B_1\uplus B_2)=\mu(B_1)+\mu(B_2)$, and continuous with respect to countably increasing sequences: if for every n, $B_n\subseteq B_{n+1}$, then $\mu(\bigcup_{n=0}^{\infty}B_n)=\lim_{n\to\infty}\mu(B_n)$. The *Lebesgue measure* λ is the unique measure on $\mathbb{R}$ assigning to each interval its length $\lambda([a,b])=b-a$. A *probability measure* on $\mathbb{R}^n$ is a measure μ whose *total measure* $\mu(\mathbb{R}^n)$ is 1. The *uniform probability measure* $\mathbb{U}_{[a,b]}$ on an interval $[a,b]$ assigns to each Borel set B the relative Lebesgue measure it occupies in the interval $[a,b]$: $\mathbb{U}_{[a,b]}(B):=\frac{1}{b-a}\lambda(B\cap[a,b])$. Measurable functions $f:\mathbb{R}^n\to\mathbb{R}^m$ let us transport measures μ on $\mathbb{R}^n$ to their *push-forward* measure $f_*\mu$ on $\mathbb{R}^m$, by setting $f_*\mu(B):=\mu(f^{-1}[B])$.

We think of whole statistical probabilistic programs of real type as a formalism for describing measures. To work compositionally, we need the following analogous concept for program fragments, i.e., terms with unbound variables. A *probability kernel* k from $\mathbb{R}^n$ to $\mathbb{R}$, written as $k:\mathbb{R}^n\rightsquigarrow\mathbb{R}$ is a function assigning to every $\vec{x}\in\mathbb{R}^n$ a probability measure $k(\vec{x},-)$ on $\mathbb{R}$, such that, for every Borel set B, the function $k(-,B):\mathbb{R}^n\to\mathbb{R}$ is measurable. We will use the following key result about probability kernels, the *randomisation lemma*, which says that a straightforward random number generator $\mathbb{U}_{[0,1]}$ suffices to implement any of them.

Lemma 2.1 ([Kallenberg 2006, Lemma 3.22]). *For every probability kernel $k:\mathbb{R}^n\rightsquigarrow\mathbb{R}$, there is a measurable function $rand_k:\mathbb{R}^{n+1}\to\mathbb{R}$, such that $rand_k(x_1,\ldots,x_n,-)_*\mathbb{U}_{[0,1]}=k(x_1,\ldots,x_n)$.*

2.2 SFPC: Bayesian Statistical Modelling with Recursive Types

Syntax. As recursive types contain type-variables, we use a kind system to ensure types are well-formed. Fig. 3 (top left) presents the kinds of our calculus, and Fig. 3 (bottom left) presents the types of SFPC. We include type variables, taken from a countable set ranged over by α,β,γ. We include simple types: unit, product, function, and variant types. Variant types use constructor labels taken from a countable set ranged over by $\ell,\ell_1,\ell_2,\ldots$. In our abstract syntax, variant types are *dictionaries*, partial functions with a finite domain, from the set of constructor labels to the set of types. The recursive type former $\mu\alpha.\tau$ binds α in τ. In our abstract syntax, term variable contexts Γ are dictionaries from the countable set of variables, ranged over by $x,y,z,\ldots$, to the set of types.

We desugar stand-alone labels in a variant type $\{\cdots \mid \ell \mid \cdots\}$ to the unit type $\{\cdots \mid \ell\,\mathbf{1} \mid \cdots\}$. We also desugar top-level-like recursive type declarations $\tau:=\sigma[\alpha\mapsto\tau]$ to $\tau:=\mu\alpha.\sigma$.

Example 2.2. The type of booleans is $\mathsf{bool}:=\{\mathsf{True}\mid\mathsf{False}\}$. The type of natural numbers is $\mathbb{N}:=\{\mathsf{Zero}\mid\mathsf{Succ}\,\mathbb{N}\}$ desugaring to $\mathbb{N}:=\mu\alpha.\{\mathsf{Zero}\mid\mathsf{Succ}\,\alpha\}$. The type of τ-lists is $\mathsf{List}\,\tau:=\{\mathsf{Nil}\mid\mathsf{Cons}\,\tau*\mathsf{List}\,\tau\}$, desugaring to $\mathsf{List}\,\tau:=\mu\alpha.\{\mathsf{Nil}\,\mathbf{1}\mid\mathsf{Cons}\,\tau*\alpha\}$. The type of (infinite) stochastic processes in τ is $\mathsf{Stoch}\,\tau:=\mathbf{1}\to(\tau*\mathsf{Stoch}\,\tau)$, desugaring to $\mathsf{Stoch}\,\tau:=\mu\alpha.\mathbf{1}\to(\tau*\alpha)$. The type of untyped λ-terms with values in τ is $\Lambda\text{-}\mathsf{Term}\,\tau:=\{\mathsf{Val}\,\tau\mid\mathsf{Fun}\,(\Lambda\text{-}\mathsf{Term}\,\tau\to\Lambda\text{-}\mathsf{Term}\,\tau)\}$, desugaring to $\Lambda\text{-}\mathsf{Term}\,\tau:=\mu\alpha.\{\mathsf{Val}\,\tau\mid\mathsf{Fun}\,(\alpha\to\alpha)\}$.

Fig. 3 (right) presents the terms of SFPC. Value variables are taken from a countable set ranged over by $x,y,z,\ldots$. We include primitive function constants $\underline{f}(t_1,\ldots,t_n)$ for every measurable

$k ::=$	kinds	$t, s, r ::=$	terms		function:
type	type	x	term variable	$\mid \lambda x : \tau.t$	abstraction
$\mid$ context	context	$\mid \underline{f}(t_1, \ldots, t_n)$	primitive	$\mid t\ s$	application
$\tau, \sigma, \rho ::=$	types	$\mid$ **match** t **with**	conditional		pattern matching:
α	variable	$\{\underline{0} \to s \mid \_ \to r\}$		$\mid$ **match** t **with**	unit
$\mid \underline{\mathbb{R}}$	reals	$\mid$ **sample**	sampling	$() \to s$	
$\mid \mathbf{1}$	unit	$\mid$ **score** t	conditioning	$\mid$ **match** t **with**	product
$\mid \tau * \sigma$	product		constructors:	$(x, y) \to s$	
$\mid \tau \to \sigma$	function	$\mid ()$	unit	$\mid$ **match** t **with**	variant
$\mid \{\ell_1\ \tau_1$	variant	$\mid (t, s)$	pairing	$\{\ell_1\ x_1 \to s_1 \mid \cdots$	
$\mid \ldots$		$\mid \tau.\ell\ t$	variant	$\mid \ell_n\ x_n \to s_n\}$	
$\mid \ell_n\ \tau_n\}$		$\mid \tau.\mathbf{roll}\,(t)$	iso-recursive	$\mid$ **match** t **with**	recursive
$\mid \mu\alpha.\tau$	iso-recursive			**roll** $x \to s$	

$\Gamma ::= x_1 : \tau_1, \ldots, x_n : \tau_n$ variable contexts

Fig. 3. SFPC kinds and types (left) and terms (right)

Sugar	Elaboration
• _	fresh variable
• **let** $(x : \tau) = t$ **in** s	$(\lambda x : \tau.s)\,t$
• **letrec** $(x : \rho) = t$ **in** s	$(\lambda x : \rho.s)(\mu x : \rho.t)$
• $t; s$	**match** t **with** $() \to s$
• **unroll** t	**match** t **with roll** $x \to x$

• $\mu x : \tau.t$	**let** $(body : \sigma \to \tau)) = \lambda y : \sigma.$ **let** $(x : \tau) = \lambda z : \tau_1.$**unroll** $y\ y$ **in** t **in** $body\,(\sigma.$**roll** $body)$
	where $\tau := \tau_1 \to \tau_2$ and $\sigma := \mu\alpha.\alpha \to \tau$
• $\bot_\rho$	$\mu x : \rho.x$

Fig. 4. Term-level syntactic sugar

(partial) function $f : \mathbb{R}^n \rightharpoonup \mathbb{R}$. The conditional construct tests whether its argument of type $\underline{\mathbb{R}}$ evaluates to 0. We include an effect **sample** for sampling a real number uniformly from the interval $[0, 1]$, for defining the prior distribution of the program. We also include an effect **score** r for reweighting the posterior distribution by the non-negative factor $|r| \in [0, \infty)$. We include standard constructors and pattern-matching constructs for the simple types, and standard function abstraction and application constructs. Finally, we include the standard *iso-recursive* constructors and pattern matching, which require an explicit rolling and unrolling of the recursive definition such as $\mathbb{N}.$**roll** (Zero()). Variant and iso-recursive constructor terms as well as function abstraction annotate their binding occurrences with the appropriate closed type τ to ensure unique typing.

To aid readability, we use the standard syntactic sugar of Fig. 4, e.g. **let** $x = t$ **in** s for $(\lambda x.\, t)s$, $\mu x : \tau \to \sigma.t$ for the usual encoding of term level recursion using type level recursion [Abadi and Fiore 1996; Fiore 1996]. When dealing with a recursive variant type $\sigma = \mu\alpha.\{\ldots \mid \ell\ \tau \mid \ldots\}$, we write $\ell\ t$ for the more cumbersome constructor $\sigma.$**roll** $(\{\ldots \mid \ell\ \tau \mid \ldots\}.\ell\ t)$ as long as σ is clear from the context. We can encode constructs $\underline{k}$ in our calculus for drawing from arbitrary probability kernels $k : \mathbb{R}^n \rightsquigarrow \mathbb{R}$. Using the Randomisation Lemma 2.1, first find the appropriate measurable function $rand_k : \mathbb{R}^{n+1} \to \mathbb{R}$ and express $\underline{k}(t_1, \ldots, t_n)$ by $\underline{rand_k}(t_1, \ldots, t_n,$ **sample**$)$.

Example 2.3. For Stoch $\underline{\mathbb{R}} = \mu\alpha.\mathbf{1} \to (\underline{\mathbb{R}} * \alpha)$, define **draw** $:= \lambda x :$ Stoch $\underline{\mathbb{R}}.$**unroll** $(x)\,()$, which draws a value from the process and moves the process to the corresponding new state. As an example stochastic process, writing $normal\_rng : \mathbb{R} \times \mathbb{R} \rightsquigarrow \mathbb{R}$ for a Gaussian probability kernel taking the mean and standard deviation as arguments, we define an example Gaussian random

$$\frac{}{\Delta \vdash_{\mathrm{k}} \alpha : \mathrm{type}}(\alpha \in \Delta) \qquad \frac{}{\Delta \vdash_{\mathrm{k}} \underline{\mathbb{R}} : \mathrm{type}} \qquad \frac{}{\Delta \vdash_{\mathrm{k}} \mathbf{1} : \mathrm{type}} \qquad \frac{\Delta \vdash_{\mathrm{k}} \tau : \mathrm{type} \qquad \Delta \vdash_{\mathrm{k}} \sigma : \mathrm{type}}{\Delta \vdash_{\mathrm{k}} \tau * \sigma : \mathrm{type}}$$

$$\frac{\Delta \vdash_{\mathrm{k}} \tau : \mathrm{type} \quad \Delta \vdash_{\mathrm{k}} \sigma : \mathrm{type}}{\Delta \vdash_{\mathrm{k}} \tau \to \sigma : \mathrm{type}} \qquad \frac{\text{for all } 1 \le i \le n\text{:}\ \Delta \vdash_{\mathrm{k}} \tau_i : \mathrm{type}}{\Delta \vdash_{\mathrm{k}} \{\ell_1\, \tau_1 \mid \ldots \mid \ell_n\, \tau_n\} : \mathrm{type}} \qquad \frac{\Delta, \alpha \vdash_{\mathrm{k}} \tau : \mathrm{type}}{\Delta \vdash_{\mathrm{k}} \mu\alpha.\tau : \mathrm{type}} \qquad \frac{\text{for all } (x : \tau) \in \Gamma\text{:}\ \vdash_{\mathrm{k}} \tau : \mathrm{type}}{\vdash_{\mathrm{k}} \Gamma : \mathrm{context}}$$

Fig. 5. SFPC kind system

$$\frac{(x : \tau) \in \Gamma}{\Gamma \vdash x : \tau} \qquad \frac{f : \mathbb{R}^n \to \mathbb{R} \quad \text{for all } 1 \le i \le n\text{: } \Gamma \vdash t_i : \underline{\mathbb{R}}}{\Gamma \vdash \underline{f}(t_1, \ldots, t_n) : \underline{\mathbb{R}}} \qquad \frac{\Gamma \vdash t : \underline{\mathbb{R}} \qquad \Gamma \vdash s : \tau \qquad \Gamma \vdash r : \tau}{\Gamma \vdash \mathbf{match}\ t\ \mathbf{with}\ \{\underline{0} \to s \mid \_ \to r\} : \tau}$$

$$\frac{}{\Gamma \vdash \mathbf{sample} : \underline{\mathbb{R}}} \qquad \frac{\Gamma \vdash t : \underline{\mathbb{R}}}{\Gamma \vdash \mathbf{score}\, t : \mathbf{1}} \qquad \frac{}{\Gamma \vdash () : \mathbf{1}} \qquad \frac{\Gamma \vdash t : \tau \quad \Gamma \vdash s : \sigma}{\Gamma \vdash (t, s) : \tau * \sigma}$$

$$\frac{\Gamma \vdash t : \tau_i}{\Gamma \vdash \tau.\ell_i\, t : \tau}(\tau = \{\ell_1\, \tau_1 \mid \ldots \mid \ell_n\, \tau_n\}) \qquad \frac{\Gamma \vdash t : \sigma[\alpha \mapsto \tau]}{\Gamma \vdash \tau.\mathbf{roll}\,(t) : \tau}(\tau = \mu\alpha.\sigma) \qquad \frac{\Gamma, x : \tau \vdash t : \sigma}{\Gamma \vdash \lambda x : \tau.t : \tau \to \sigma}$$

$$\frac{\Gamma \vdash t : \sigma \to \tau \qquad \Gamma \vdash s : \sigma}{\Gamma \vdash t\, s : \tau} \qquad \frac{\Gamma \vdash t : \sigma * \rho \qquad \Gamma, x : \sigma, y : \rho \vdash s : \tau}{\Gamma \vdash \mathbf{match}\ t\ \mathbf{with}\ (x, y) \to s : \tau} \qquad \frac{\Gamma \vdash t : \mathbf{1} \qquad \Gamma \vdash s : \tau}{\Gamma \vdash \mathbf{match}\ t\ \mathbf{with}\ () \to s : \tau}$$

$$\frac{\Gamma \vdash t : \{\ell_1\, \tau_1 \mid \ldots \mid \ell_n\, \tau_n\} \qquad \text{for each } 1 \le i \le n\text{: } \Gamma, x_i : \tau_i \vdash s_i : \tau}{\Gamma \vdash \mathbf{match}\ t\ \mathbf{with}\ \{\ell_1\, x_1 \to s_1 \mid \cdots \mid \ell_n\, x_n \to s_n\} : \tau}$$

$$\frac{\Gamma \vdash t : \mu\alpha.\sigma \qquad \Gamma, x : \sigma[\alpha \mapsto \mu\alpha.\sigma] \vdash s : \tau}{\Gamma \vdash \mathbf{match}\ t\ \mathbf{with}\ \mathbf{roll}\, x \to s : \tau} \qquad \frac{\Gamma, x : \tau \to \sigma \vdash t : \tau \to \sigma}{\Gamma \vdash \mu x : \tau \to \sigma.t : \tau \to \sigma}$$

Fig. 6. SFPC type system, including (in gray) the derivable typing rule for term recursion

walk $\mathbf{RW}\ \underline{\sigma}\ \underline{\mu}$ with initial position μ and standard deviation σ by setting:

$$\mathbf{RW} = \lambda x : \underline{\mathbb{R}}.\mu y : \underline{\mathbb{R}} \to \mathrm{Stoch}\,\underline{\mathbb{R}}.\lambda z : \underline{\mathbb{R}}.\mathrm{Stoch}\,\underline{\mathbb{R}}.\mathbf{roll}\,(\lambda\_ : \mathbf{1}.(z, y(\underline{normal\_rng}(z, x))))$$

Kind and Type Systems. To ensure the well-formedness of types, which involve type variables, we use the kind system presented in Fig. 5. Each kinding judgement $\Delta \vdash_{\mathrm{k}} \tau : \mathrm{type}$ asserts that a given type τ is well-formed in the *type variable context* Δ, which is a finite set of type variables. The kinding judgements are standard. All type variables must be bound by the enclosing context, or by a recursive type binder. *Variable contexts* Γ must assign closed types. We treat α-conversion and capture avoiding substitution of variables as usual, possessing the standard structural properties.

Fig. 6 presents the resulting type system, including the derivable typing judgement for the sugar $\mu x : \tau \to \sigma.t$ for term recursion. Each typing judgement $\Gamma \vdash t : \tau$ asserts that the term t is well-typed with the well-formed closed type $\vdash_{\mathrm{k}} \tau : \mathrm{type}$ in the variable context $\vdash_{\mathrm{k}} \Gamma : \mathrm{context}$. The rules are standard. By design, every term has at most one type in a given context.

Example 2.4. The types from Ex. 2.2 are well-kinded: $\mathrm{bool}, \mathbb{N}, \mathrm{List}\,\underline{\mathbb{R}}, \mathrm{Stoch}\,\underline{\mathbb{R}}, \Lambda\text{-}\mathrm{Term}\,\underline{\mathbb{R}} : \mathrm{type}$. The terms from Ex. 2.3 are well-typed: $\mathbf{draw} : \mathrm{Stoch}\,\underline{\mathbb{R}} \to (\underline{\mathbb{R}} * \mathrm{Stoch}\,\underline{\mathbb{R}})$, $\mathbf{RW} : \underline{\mathbb{R}} \to \underline{\mathbb{R}} \to \mathrm{Stoch}\,\underline{\mathbb{R}}$.

$t, s, r ::=$	terms			v	values	
x	variable	$\mid \mathbf{factor}\, t$	conditioning	$::= x$	variable	
$\mid \underline{k}(t_1, \ldots, t_n)$	kernel	$\mid \lambda x.t$	abstraction	$\mid \underline{r}$	real number	
$\mid \mathbf{ifz}\, t\, \mathbf{then}\, s\, \mathbf{else}\, r$	conditional	$\mid t\ s$	application	$\mid \lambda x.t$	abstraction	

Fig. 7. Idealised Church, terms and values

Values v	
$x \mapsto x \quad \underline{r} \mapsto \mathsf{Val}\, \underline{r} \quad \lambda x.t \mapsto \mathsf{Fun}\, (\lambda x : \tau.t^\dagger)$	
Terms t, s, r	
v	$\mapsto v^\dagger$
$t\ s$	$\mapsto (u := t^\dagger;) \bullet u\ s^\dagger$
$\underline{k}(t_1, \ldots, t_n)$	$\mapsto (; x_1 := t_1^\dagger, \ldots, x_n := t_n^\dagger) \bullet$ $\mathsf{Val}\, \underline{k}(x_1, \ldots, x_n)$
$\mathbf{ifz}\, t\, \mathbf{then}\, r\, \mathbf{else}\, s$	$\mapsto (; x_1 := t^\dagger) \bullet \mathbf{match}\, x\, \mathbf{with}$ $\{\underline{0} \to r^\dagger \mid \_ \to s^\dagger\}$
$\mathbf{factor}\, (t)$	$\mapsto (; x := t) \bullet \mathbf{score}\, x; \mathsf{Val}\, x$

Auxiliary sequential unpacking	
$(u_1 := t_1, \ldots u_n := t_n; x_1 := s_1, \ldots, x_m := s_m) \bullet r$	
$(;) \bullet r$	$\mapsto r$
$(; x_1 := s_1, \ldots) \bullet r$	$\mapsto \mathbf{match\ unroll}\, s_1\, \mathbf{with}\, \{$ $\mathsf{Val}\, x_1 \to (; \ldots) \bullet r$ $\mid \mathsf{Fun}\, \_ \to \bot_\sigma\}$
$(u_1 := t_1, \ldots; \ldots) \bullet r$	$\mapsto \mathbf{match\ unroll}\, t_1\, \mathbf{with}\, \{$ $\mathsf{Fun}\, u_1 \to (\ldots; \ldots) \bullet r$ $\mid \mathsf{Val}\, \_ \to \bot_\sigma\}$

Fig. 8. A faithful translation $(-)^\dagger$ of Idealised Church into SFPC using the type $\sigma := \Lambda\text{-}\mathsf{Term}\, \underline{\mathbb{R}}$.

2.3 Idealised Church: Untyped Statistical Modelling

In applied probabilistic programming systems, there has been considerable interest in using an *untyped* λ-calculus as the basis for probabilistic programming. For instance, the Church modelling language [Wingate et al. 2011], in idealised form, is an untyped call-by-value λ-calculus over the real numbers with constructs $\underline{k}(x_1, \ldots, x_n)$ for drawing from probability kernels $k : \mathbb{R}^n \rightsquigarrow \mathbb{R}$ (including all real numbers, measurable functions like a Gaussian density $\underline{normal\_pdf}(y \mid \mu, \sigma)$ and proper kernels like a Gaussian random number generator $\underline{normal\_rng}(\mu, \sigma)$), and a construct **factor** for reweighting program traces, to enforce soft constraints [Borgström et al. 2016]. Fig. 7 presents the syntax of Idealised Church, and we desugar $\mathbf{let}\, x = t\, \mathbf{in}\, s$ to mean $(\lambda x.s)\, t$.

Idealised Church arises as a sublanguage of SFPC. We encode Idealised Church terms as SFPC terms of the type $\Lambda\text{-}\mathsf{Term}\, \underline{\mathbb{R}} := \{\mathsf{Val}\, \tau \mid \mathsf{Fun}\, (\Lambda\text{-}\mathsf{Term}\, \underline{\mathbb{R}} \to \Lambda\text{-}\mathsf{Term}\, \underline{\mathbb{R}})\}$ from Ex. 2.2, using the translation $(-)^\dagger$ in Fig. 8. The translation uses an auxiliary SFPC construct $(\ldots;\ldots) \bullet r$ for sequentially evaluating its arguments $x := t$, unpacking each term t, ensuring it is either a function or a real value, and binding its unpacked value to x in r. This translation is faithful (Lemma 6.2).

2.4 CBV SPCF: simply typed recursive modelling

We consider a simply typed sublanguage of SFPC, a call-by-value (CBV) probabilistic variant of Plotkin and Scott's PCF [1977; 1993]. The types and terms are given by the following grammars:

$$\tau, \sigma, \rho \quad ::= \underline{\mathbb{R}} \mid \tau \to \sigma \qquad\qquad t, s, r \quad ::= x \mid \underline{f}(t_1, \ldots, t_n) \mid \mathbf{match}\, t\, \mathbf{with}\, \{\underline{0} \to s \mid \_ \to r\} \mid \mathbf{sample} \mid \mathbf{factor}\, (t) \mid \lambda x : \tau.t \mid t\ s \mid \mu x : \tau \to \sigma.t,$$

SPCF is a fragment of SFPC: term recursion $\mu x : \tau \to \sigma.t$ is interpreted as in Fig. 4 and conditioning $\mathbf{factor}\, (t)$ as $\mathbf{let}\, (x : \underline{\mathbb{R}}) = t\, \mathbf{in}\, \mathbf{score}\, x; x$. SPCF derives its kind and type systems from SFPC.

3 QUASI-BOREL PRE-DOMAINS

Previous works on quasi-Borel spaces (qbses) give a denotational semantics for higher-order probabilistic languages with a range of types, but crucially excludes higher-order term recursion and recursive types. To do that, we further equip a qbs with a compatible ωcpo structure. We call this new semantic structure a *quasi-Borel pre-domain* or an *ωqbs*.

3.1 Preliminaries

Category theory. We assume familiarity with categories $\mathcal{C}$, $\mathcal{D}$, functors $F, G : \mathcal{C} \to \mathcal{D}$, natural transformations $\alpha, \beta : F \to G$, and their theory of (co)limits and adjunctions. We write:

- unary, binary, and I-ary products as $\mathbb{1}$, $X_1 \times X_2$, and $\prod_{i\in I} X_i$, writing π_i for the projections and $\langle\rangle$, $\langle x_1, x_2\rangle$, and $\langle x_i\rangle_{i\in I}$ for the tupling maps;
- unary, binary, and I-ary coproducts as $\mathbb{0}$, $X_1 + X_2$, and $\sum_{i\in I} X_i$, writing ι_i for the injections and $[]$, $[x_1, x_2]$, and $[x_i]_{i\in I}$ for the cotupling maps;
- exponentials as X^Y, writing Λ for the currying maps.

Domain theory. We recall some basic domain theory. Let $\omega = \{0 \leq 1 \leq \ldots\}$ be the ordinary linear order on the naturals. An *ω-chain* in a poset $P = \big(|P|, \leq\big)$ is a monotone function $a_- : \omega \to P$. A poset P is an *ωcpo* when every ω-chain $\langle a_n\rangle_{n\in\mathbb{N}}$ has a least upper bound (lub) $\bigvee_{n\in\mathbb{N}} a_n$ in P.

Example 3.1. Each set X equipped with the discrete partial order forms an ωcpo $(X, =)$. E.g., the discrete ωcpo $\mathbb{R}$ over the real line. The non-negative extended reals equipped with the ordinary order, $\mathbb{W} := \big([0,\infty], \leq\big)$, is an ωcpo. The Borel subsets of $\mathbb{R}^n$ ordered by inclusion form an ωcpo $\mathcal{B}_n$.

For every pair of ωcpos P and Q, a *Scott-continuous* function $f : P \to Q$ is a monotone function $f : |P| \to \big|Q\big|$ such that for every ω-chain a_-, we have: $f(\bigvee_n a_n) = \bigvee_n f(a_n)$. A Scott-continuous function $f : P \to Q$ is a *full mono* when, for every $a, b \in |P|$, we have $f(a) \leq f(b) \implies a \leq b$. Recall that the category ω**Cpo** of ωcpos and Scott-continuous functions is Cartesian closed: products are taken componentwise and the exponential Q^P has carrier $\omega\mathbf{Cpo}(P, Q)$ and order $f \leq_{Q^P} g$ iff $\forall p \in |P|. f(p) \leq_Q g(p)$. A *domain* is an ωcpo with a least element $\bot$. A *strict* function between domains is a Scott-continuous function that preserves their least elements.

Example 3.2. A measure on $\mathbb{R}^n$ is a strict continuous function $\mu : \mathcal{B}_n \to \mathbb{W}$ that is linear on disjoint subsets. The measurable functions $\mathcal{B}(\mathbb{R}, [0,\infty])$ ordered pointwise fully include into the ωcpo $\mathbb{W}^{\mathbb{R}}$. The *integral* is the unique Scott-continuous function $\int \mu : \mathcal{B}(\mathbb{R}, [0,\infty]) \to \mathbb{W}$ satisfying, for all measurable partitions $\mathbb{R} = \sum_{n\in\mathbb{N}} U_n$ and weights $\langle w_n\rangle$ in $\mathbb{W}$: $\int \mu\,[\lambda r : U_n. w_n]_{n\in\mathbb{N}} = \sum_{n\in\mathbb{N}} x_n \cdot \mu U_n$.

An *ω**Cpo**-(enriched) category* $\mathcal{C}$ consists of a locally-small category $\underline{\mathcal{C}}$ together with an assignment of an ωcpo $\mathcal{C}(A, B)$ to every $A, B \in \mathrm{Ob}\,(\underline{\mathcal{C}})$ whose carrier is the set $\underline{\mathcal{C}}(A, B)$ such that composition is Scott-continuous. An *ω**Cpo**-functor*, a.k.a. a *locally-continuous functor*, $F : \mathcal{C} \to \mathcal{D}$ between two ω**Cpo**-categories is an ordinary functor $\underline{F} : \underline{\mathcal{C}} \to \underline{\mathcal{D}}$ between the underlying ordinary categories, such that every morphism map $\underline{F}_{A,B} : \mathcal{C}(A, B) \to \mathcal{D}(FA, FB)$ is Scott-continuous.

Example 3.3. Every locally-small category is an ω**Cpo**-category whose hom-ωcpos are discrete. The category ω**Cpo** itself is an ω**Cpo**-category. If $\mathcal{C}$ is an ω**Cpo**-category, its categorical dual $\mathcal{C}^{\mathrm{op}}$ is an ω**Cpo**-category. The category of locally-continuous functors $\mathcal{C}^{\mathrm{op}} \to \omega\mathbf{Cpo}$, with the order on natural transformations $\alpha : F \to G$ given componentwise, is an ω**Cpo**-category when $\mathcal{C}$ is.

Measure theory. A measurable space $X = \big(|X|, \Sigma_X\big)$ consists of a carrier set $|X|$ and a set of subsets $\Sigma_X \subseteq \mathcal{P}(X)$, called its *$\sigma$-algebra*, containing the empty set, and closed under complements and countable unions, thus axiomatising the measurable subsets of $\mathbb{R}^n$. A measurable function $f : X \to Y$ is a function $f : |X| \to |Y|$ whose inverse image maps measurable subsets to measurable

subsets. Thus every n-dimensional Borel set, together with its Borel subsets, forms a measurable space. The measurable spaces that are measurably isomorphic to a Borel set are called *standard Borel spaces*. A fundamental result in descriptive set theory is that every standard Borel space is measurably isomorphic to $\{i \in \mathbb{N} | i < n\}$ for some $n = 0, 1, \ldots, \omega$, or to $\mathbb{R}$ [Kechris 2012]. We write **Meas** and **Sbs** for the categories of measurable and standard Borel spaces and measurable functions between them.

Quasi-Borel spaces. As a Cartesian closed alternative to measure theory, Heunen et al. [2017] introduced the category **Qbs** of *quasi-Borel spaces* (qbses). Measure theory axiomatises measurable subsets of a space X, and deriving the *random elements*: measurable functions $\alpha : \mathbb{R} \to X$, for pushing measures forward. Qbses axiomatise random elements directly.

A *quasi-Borel space (qbs)* $X = \left(|X|, M_X\right)$ consists of a carrier set $|X|$ and a set of functions $M_X \subseteq |X|^{\mathbb{R}}$, called the *random elements*, such that (i) all the constant functions are in M_X, (ii) M_X is closed under precomposition with measurable functions on $\mathbb{R}$, and (iii) if $\mathbb{R} = \bigcup_{n\in\mathbb{N}} U_n$, where U_n are pairwise-disjoint and Borel measurable, and $\alpha_n \in M_X$ for all n, then the countable case-splitting $\left[\alpha_n|_{U_n}\right]_{n\in\mathbb{N}}$ is in M_X. A *morphism* $f : X \to Y$ is a structure-preserving function $f : |X| \to |Y|$, i.e. if $\alpha \in M_X$ then $(f \circ \alpha) \in M_Y$. Morphisms compose as functions, and we have a category **Qbs**.

Example 3.4. We turn the n-dimensional space $\mathbb{R}^n$ into a qbs by taking the random elements to be the measurable functions $M_{\mathbb{R}^n} := \mathbf{Meas}(\mathbb{R}, \mathbb{R}^n) \cong \prod_{i=1}^{n} \mathbf{Meas}(\mathbb{R}, \mathbb{R})$, i.e., n-tuples of correlated random variables. We also turn every set X into a qbs by taking the random elements to be measurably piece-wise constant functions, i.e., the *step functions*.

Both these examples are special cases of a more abstract situation. Every measurable space X can be turned into a qbs by setting $M_X := \mathbf{Meas}(\mathbb{R}, X)$. This defines a functor $M_- : \mathbf{Meas} \to \mathbf{Qbs}$. It has a left adjoint $\Sigma_- : \mathbf{Qbs} \to \mathbf{Meas}$ which equips a qbs X with the largest σ-algebra such that all random elements $\alpha \in M_X$ are measurable. This adjunction restricts to an adjoint embedding of the category of standard Borel spaces **Sbs** as a full subcategory of **Qbs**. This embedding makes **Qbs** a conservative extension of the well-behaved standard Borel spaces.

The category of qbses possesses substantial pleasant categorical properties: it has all (co)limits and is Cartesian closed, and so can interpret simple types, quotients, and refinements. In fact, **Qbs** is a Grothendieck quasi-topos, and so can interpret an expressive internal logic. The conservativity of the embedding $\mathbf{Sbs} \hookrightarrow \mathbf{Qbs}$ means that interpreting closed programs of ground type and reasoning about them in **Qbs** have standard measure-theoretic counterparts. The benefit comes from doing so *compositionally*: program fragments that are higher-order functions have a compositional interpretation and reasoning principles in **Qbs**, but not in **Sbs** nor in **Meas**.

3.2 Definition and Some Simple Examples

The difficulty inherent in combining domain and measure theory stems from the following considerations [Jung and Tix 1998]. Each (pre-)domain induces a topological space whose open subsets are the *Scott-open* subsets (see Ex. 4.7 and Ex. 5.1), from which one generates a measurable space structure by closing over countable unions and complements. Both the domain-theoretic structure and the induced measure-theoretic structure possess a cartesian product construction. However, without further assumptions, the two product structures may differ. To get a commutative probabilistic powerdomain, one requires conditions that ensure these two structures agree while maintaining, e.g., cartesian closure. The search for such a category is known as the *Jung-Tix problem*. We circumvent the Jung-Tix problem, without solving it, by keeping the two structures, the domain-theoretic and the measurable, separate but compatible. Doing so also allows us to replace the measure-theoretic

	Carrier	Random elements	Partial order
$\mathbb{1}$	$\{r \mapsto \langle\rangle\}$	$\{\langle\rangle\}$	$=_{\mathbb{1}}$
$P \times Q$	$\lvert P\rvert \times \lvert Q\rvert$	$\{\langle \alpha, \beta\rangle \mid \alpha \in M_P, \beta \in M_Q\}$	$\langle p, q\rangle \leq \langle p', q'\rangle \colon p \leq_P p', q \leq_Q q'$
$\sum_{i=1}^n P_i$	$\sum_{i=1}^n \lvert P_i\rvert$	$\left\{ [\iota_i \circ \alpha_i]_{i=1}^n \middle\vert \begin{array}{l} \langle A_i\rangle_{1\leq i\leq n} \in \mathcal{B}^n, \\ \mathbb{R} = \biguplus_{i=1}^n A_i \text{ partition}, \\ \forall 1 \leq i \leq n. \alpha_i \in M_{P_i} \end{array} \right\}$	$\langle j, p\rangle \leq \langle k, q\rangle \colon j = k, p \leq_{P_j} q$
Q^P	$\omega\mathbf{Qbs}(P, Q)$	$\left\{ \Lambda(\alpha) : \mathbb{R} \to \left\lvert Q^P\right\rvert \middle\vert \alpha \in \omega\mathbf{Qbs}(\mathbb{R} \times P, Q)\right\}$	$f \leq_{Q^P} g \colon \forall p \in \lvert P\rvert . f(p) \leq_Q g(p)$
$P_\perp$	$\lvert P\rvert + \{\perp\}$	$\left\{ [B. \uparrow \circ \alpha, B^{\complement}.\perp] \middle\vert B \in \mathcal{B}, \alpha \in M_P \right\}$	$\perp \leq_{P_\perp} x, (\uparrow a \leq_{P_\perp} \uparrow b \iff a \leq_P b)$

Fig. 9. The simply-typed structure of $\omega\mathbf{Qbs}$

structure, which is usually incompatible with higher-order structure, with a quasi-Borel space structure. The result is the following definition:

Definition 3.5. An *ωqbs* P consists of a triple $P = \left\langle |P|, M_P, \leq_P \right\rangle$ where: $\left(|P|, M_P\right)$ is a qbs; $\left(|P|, \leq_P\right)$ is an ωcpo over $|P|$; and M_P is closed under pointwise sups of ω-chains w.r.t. the pointwise order. A *morphism* between ωqbses $f : P \to Q$ is a Scott-continuous function between their underlying ωcpos that is also a **Qbs**-morphism between their underlying qbses. We denote the category of ωqbses and their morphisms by $\omega\mathbf{Qbs}$.

Example 3.6 (Real Values). We have the ωqbs $\mathbb{R} = (\mathbb{R}, \mathbf{Meas}(\mathbb{R}, \mathbb{R}), =_{\mathbb{R}})$ with the discrete order $=_{\mathbb{R}}$. This pre-domain represents a space of *values* returned by probabilistic computations.

Example 3.7 (Real Weights). Contrast this pre-domain with $\mathbb{W} = ([0, \infty], \mathbf{Meas}(\mathbb{R}, [0, \infty]), \leq_{[0,\infty]})$ with the linear order. This pre-domain represents a space of *weights* of computation traces. Compare this space to the Sierpiński space $\{0, 1\}_\leq$, the full subspace $\{0, 1\}$ in $\mathbb{W}$.

The category $\omega\mathbf{Qbs}$ is an $\omega\mathbf{Cpo}$-category, with each homset in $\omega\mathbf{Qbs}$ ordered *pointwise* by setting, for every pair of morphisms $f, g \in \omega\mathbf{Qbs}(X, Y)$: $f \leq g$ when $\forall x \in |X| \,. f(x) \leq_Y g(x)$.

3.3 Interpreting Simple Types and Partiality

We turn every qbs into the *discrete* ωqbs over it by taking the discrete ωcpo structure, i.e., equality as an order. This construction is the left adjoint to the evident forgetful functor $|-| : \omega\mathbf{Qbs} \to \mathbf{Qbs}$. Similarly, we turn every ωcpo into the *free* ωqbs over it whose random elements are lubs of step functions. This construction is the left adjoint to the evident forgetful functor $|-| : \omega\mathbf{Qbs} \to \omega\mathbf{Cpo}$.

Example 3.8. The discrete ωqbs on the qbs structure of the real line is the pre-domain $\mathbb{R}$ of real values. The free ωqbs on the ωcpo of weights (Ex. 3.1) is the pre-domain $\mathbb{W}$ of real weights (Ex. 3.7).

These adjoints equip $\omega\mathbf{Qbs}$ with well-behaved limits and coproducts:

Lemma 3.9. *The forgetful functors $|-| : \omega\mathbf{Qbs} \to \omega\mathbf{Cpo}$, $|-| : \omega\mathbf{Qbs} \to \mathbf{Qbs}$ preserve limits and coproducts. This uniquely determines the limits and coproducts of $\omega\mathbf{Qbs}$, which exist for small diagrams.*

In Sec. 7 we will see that $\omega\mathbf{Qbs}$ also has quotients, but their construction is more subtle.

The category $\omega\mathbf{Qbs}$ is bi-Cartesian closed, with the concrete structure given in Fig. 9. The structural maps, such as tupling, projections, and so forth, are given as for sets. The figure also depicts a locally-continuous *lifting* monad[1] $\langle -_\perp, \text{return}, \gg\!= \rangle$, for interpreting partiality, where:

$$\uparrow x := \iota_1 x \quad \perp := \iota_2 \perp \quad \text{return}\, x := \uparrow x \quad \left((\uparrow x) \gg\!= f\right) := f(x) \quad \left(\perp \gg\!= f\right) := \perp.$$

[1] See §4.1 for the definition of monads.

This monad jointly lifts the partiality monad $P \mapsto P_\perp$ over $\omega\mathbf{Cpo}$ and the exception monad $X \mapsto X + \mathbb{1}$ over $\mathbf{Qbs}$.

Corollary 3.10. *The functor* $|-| : \omega\mathbf{Qbs} \to \mathbf{Set}$ *preserves the bi-Cartesian and partiality structures.*

So simple types, when interpreted in $\omega\mathbf{Qbs}$, retain their natural set-theoretic interpretations.

4 A COMMUTATIVE STATISTICAL POWERDOMAIN

Our powerdomain construction combines two classical ideas in probability theory. The first idea is Schwartz's treatment of distributions as expectation operators. We construct the powerdomain monad T as a submonad of the continuation monad $J := \mathbb{W}^{\mathbb{W}^-}$, where $\mathbb{W}$ is the space of weights over $[0, \infty]$ from Ex. 3.7. The second idea is the Randomisation Lemma 2.1: kernels $X \to TY$ should arise by pushing forward the Lebesgue measure along a partial function $X \times \mathbb{R} \rightharpoonup Y$, i.e., a morphism $X \to (Y_\perp)^{\mathbb{R}}$. Each element of $(Y_\perp)^{\mathbb{R}}$ induces a *randomisable* expectation operator via the Lebesgue integral. Combining the two ideas, we take T to be the smallest full submonad of the Schwartz distribution monad that contains all the randomisable distributions.

Sampling and conditioning have natural interpretations as expectation operators:

$$\mathrm{sample} : \mathbb{1} \to J\mathbb{R}, \mathrm{sample}\langle\rangle[w] := \int_{[0,1]} w(r)\,\mathrm{d}r; \qquad \mathrm{score} : \mathbb{R} \to J\mathbb{1}, \mathrm{score}\, r[w] := |r| \cdot w(\langle\rangle)$$

We will see that T is also the smallest full submonad of J which is closed under sample and score.

4.1 Preliminaries

Monads. A *strong monad structure* $\underline{T}$ over a Cartesian closed category C is a triple $\langle T, \mathrm{return}, \gg\!= \rangle$ consisting of an assignment of an object TX and a morphism $\mathrm{return}_X : X \to TX$ for every object X, and an assignment of a morphism $\gg\!=_{X,Y} : TX \times (TY)^X \to TY$. A *strong monad* is a strong monad structure $\underline{T}$ satisfying the monad laws below, expressed in the Cartesian closed internal language:

$$\big((\mathrm{return}\, x) \gg\!= f\big) = f(x) \qquad (a \gg\!= \mathrm{return}) = a \qquad \big((a \gg\!= f) \gg\!= g\big) = \big(a \gg\!= \lambda x.f(x) \gg\!= g\big)$$

Every monad yields an endofunctor T on C-morphisms: $T(f) := \mathrm{id} \gg\!=^T (\mathrm{return}^T \circ f)$. The *Kleisli* category $C_{\underline{T}}$ consists of the same objects as C, but morphisms are given by $C_{\underline{T}}(X, Y) := C(X, T\,Y)$. A strong monad $\underline{T}$ is *commutative* when, for every pair of objects X, Y:

$$a : T\,X, b : T\,Y \vdash a \gg\!= \lambda x.b \gg\!= \lambda y.\,\mathrm{return}(x, y) = b \gg\!= \lambda y.a \gg\!= \lambda x.\,\mathrm{return}(x, y)$$

Factorisation systems. We use the following concepts to factorise our powerdomain as a submonad of the Schwartz distribution monad. Recall that a *orthogonal factorisation system* on a category C is a pair $(\mathcal{E}, \mathcal{M})$ consisting of two classes of morphisms of C such that:

- Both $\mathcal{E}$ and $\mathcal{M}$ are closed under composition, and contain all isomorphisms.
- Every morphism $f : X \to Y$ in C factors into $f = m \circ e$ for some $m \in \mathcal{M}$ and $e \in \mathcal{E}$.
- Functoriality: for each situation as on the left, there is a unique $h : A \to A'$ as on the right:

$$\begin{array}{ccccc} X & \xrightarrow{e \in \mathcal{E}} & A & \xrightarrow{m \in \mathcal{M}} & Y \\ {\scriptstyle f}\downarrow & & = & & \downarrow{\scriptstyle g} \\ X' & \xrightarrow[e' \in \mathcal{E}]{} & A' & \xrightarrow[m' \in \mathcal{M}]{} & Y' \end{array} \implies \begin{array}{ccccc} X & \xrightarrow{e} & A & \xrightarrow{m} & Y \\ {\scriptstyle f}\downarrow & & \vdots{\scriptstyle h} & & \downarrow{\scriptstyle g} \\ X' & \xrightarrow[e']{} & A' & \xrightarrow[m']{} & Y' \end{array}$$

S-finite measures and kernels. Let X be a measurable space. Define measures and kernels by direct analogy with their definition on $\mathbb{R}^n$. A measure μ is *finite* when $\mu(X) < \infty$, and a kernel $k : X \rightsquigarrow Y$ is *finite* when there is some bound $B \in [0, \infty)$ such that for all $x \in |X|$, $k(x, Y) < B$. We compare measures and kernels pointwise, and both collections are ωcpos. Measures and kernels are closed under countable pointwise sums given as the lubs of the finite partial sums. A measure μ is *s-finite*

when it is a lub of finite measures, equivalently $\mu = \sum_{n\in\mathbb{N}} \mu_n$ for some countable sequence of finite measures, and similarly a kernel k is *s-finite* when it is a lub of finite kernels, equivalently $k = \sum_{n\in\mathbb{N}} k_n$ for some countable sequence of finite kernels [Staton 2017]. The Randomisation Lemma 2.1 generalises to s-finite kernels [Vákár and Ong 2018, Theorem 15]: for every s-finite kernel $k : X \rightsquigarrow Y$, with Y standard Borel, there is a partial measurable function $f : X \times \mathbb{R} \rightharpoonup Y$ such that, for every $x \in |X|$, $k(x,-) = f(x,-)_*\lambda$. Recall the 2-dimensional Lebesgue measure $\lambda \otimes \lambda$, which assigns to each rectangle $[a_1,b_1]\times[a_2,b_2]$ its area $(b_1-a_1)\times(b_2-a_2)$. Applying the Randomisation Lemma to $\lambda\otimes\lambda$ yields the *transfer principle*: there is a measurable $\varphi : \mathbb{R} \rightharpoonup \mathbb{R}\times\mathbb{R}$ such that $\varphi_*\lambda = \lambda\otimes\lambda$.

4.2 Randomisable Expectation Operators

A *randomisation* of an ωqbs X is a partial ω**Qbs**-morphism $\alpha : \mathbb{R} \rightharpoonup X$, equivalently a total ω**Qbs** morphism $\mathbb{R} \to X_\perp$. Thanks to the Cartesian closure, we have an ωqbs of randomisations $RX := (X_\perp)^{\mathbb{R}}$. A randomisation $\alpha \in RX$ represents an intensional description of a measure on X by pushing forward the Lebesgue measure λ. The undefined part of α shaves some of the measure leaving us with the restriction of λ to $\mathrm{Dom}\,(\alpha) := \alpha^{-1}\left[x \in |X_\perp| \middle| x \neq \perp\right]$. By construction, $\mathrm{Dom}\,(\alpha)$ is a Borel set, and for every weighting function $w : X \to \mathbb{W}$, the composition $w\circ\alpha : \mathrm{Dom}\,(\alpha) \to X \to \mathbb{W}$ is an ωqbs morphism. Underlying this composition is a qbs morphism $w\circ\alpha : \mathrm{Dom}\,(\alpha) \to \mathbb{W}$. Because **Qbs** is a conservative extension of **Sbs**, this morphism is a Borel measurable function. Thus, every randomisation induces an expectation operator. Moreover, this assignment is an ωqbs morphism:

$$\mathbb{E} : RX \to JX; \qquad \mathbb{E}_\alpha[w] := \int_{\mathrm{Dom}(\alpha)} \lambda(\mathrm{d}r) w(\alpha(r))$$

A *randomisable* expectation operator $\mu \in JX$ is one where $\mu = \mathbb{E}_\alpha$ for some randomisation $\alpha \in RX$. Let $|SX|$ be the set of randomisable operators $\mathbb{E}_{[X_\perp^{\mathbb{R}}]} \subseteq |JX|$. Similarly, consider the randomisable random operators $M_{SX} := \mathbb{E}\circ[M_{RX}]$. Let $|TX|$ be the ω-chain-lub-closure of $|SX| \subseteq |JX|$, and M_{TX} be the closure of $M_{SX} \subseteq M_{JX}$ under (pointwise) lub of ω-chains. The ωqbs TX is the smallest ωqbs that is a full sub-ωcpo of JX and containing the randomisable random operators.

Example 4.1. When X is a standard Borel space with the discrete order, each randomisable operator defines an s-finite measure, and each randomisable random operator defines an s-finite kernel. By the Randomisation Lemma for s-finite kernels, conversely, every s-finite measure/kernel arises from a randomisable (random) operator. Moreover, each s-finite measure/kernel is a lub of finite measures/kernels. So TX is the smallest sub-ωcpo of JX containing the finite measures as elements and kernels as random elements, and consists of the s-finite measures and kernels.

The randomisation functor $R : \omega\mathbf{Qbs} \to \omega\mathbf{Qbs}$ has the following monad structure. Using the transfer principle, fix any measurable $\varphi : \mathbb{R} \rightharpoonup \mathbb{R}\times\mathbb{R}$ satisfying $\varphi_*\lambda = \lambda\otimes\lambda$, and define:

$$\mathrm{return}\, x := \left[[0,1].\uparrow x, [0,1]^{\complement}.\perp\right] \qquad (\alpha \gg\!= f) : \mathbb{R} \xrightarrow{\varphi} \mathbb{R}\times\mathbb{R} \xrightarrow{\mathbb{R}\times\alpha} \mathbb{R}\times X \xrightarrow{\mathbb{R}\times f} \mathbb{R}\times(Y_\perp)^{\mathbb{R}} \xrightarrow{\mathrm{eval}} Y$$

The unit is a randomisation of the Dirac distribution, shaving from Lebesgue all but a probability distribution concentrated on x. The monadic bind splits the source of randomness, using the transfer principle, into two independent sources of randomness, one for α and one for the kernel f. The expectation morphism $\mathbb{E} : RX \to JX$ preserves this monad structure. The monad structure R does *not* satisfy the monad laws. While RX depends on the choice of φ, TX is independent of the choice.

4.3 Factorising Monad Structure Morphisms

To show that the monad structure of J restricts to T, we rely on a general theory, recently developed by McDermott and Kammar [2018]. The full monos between ωcpos form the $\mathcal{M}$-class of an orthogonal factorisation system on $\omega\mathbf{Cpo}$, where the $\mathcal{E}$-class consists of the *dense* epis: Scott-continuous functions $e : P \to Q$ whose image is dense, i.e., the closure of $e[P]$ is Q. Kammar and Plotkin [2012] and McDermott and Kammar [2018] use this factorisation system to decompose a locally-continuous monad over $\omega\mathbf{Cpo}$ into appropriate sub-monads indexed by the sub-collection of effect operation subsets. We use this construction to carve a sub-monad for sampling and conditioning.

A *full mono* between ωqbses is a full mono between them as ωcpos, i.e., an order reflecting ωqbs morphism. A *densely strong epi* $e : X \twoheadrightarrow Y$ is an ωqbs morphism that maps the random elements M_X into a Scott dense subset of M_Y w.r.t. the pointwise order.

Lemma 4.2. *Densely strong epis and full monos form an orthogonal factorisation system on $\omega\mathbf{Qbs}$. Moreover, the densely strong epis are closed under countable products, exponentiation with standard Borel spaces, and the lifting monad $(-)_\perp$.*

Therefore, the densely strong epis are closed under the randomisation functor R. We can now directly apply McDermott and Kammar's construction [2018] to turn TX into a canonical monad:

Theorem 4.3. *The unit and bind of J restrict to T. The (densely strong epi, full mono)-factorisation of the expectation operator $\mathbb{E} : RX \twoheadrightarrow \overline{T}X \rightarrowtail JX$ preserves these monad structures.*

4.4 Sampling and Conditioning

In the introduction to this section, we defined sampling and conditioning as expectation operators. Both arise as expectation operators for the following randomisations:

$$\begin{array}{ll} \text{sample}: \mathbb{1} \to R\mathbb{R} & \text{score} \quad : \mathbb{R} \to R\mathbb{1} \\ \text{sample} \coloneqq [[0,1].\uparrow, [0,1]^{\complement}.\perp]; & \text{score}(r) \coloneqq [[0,|r|].\uparrow\langle\rangle, [0,|r|]^{\complement}.\perp] \end{array}$$

Post-composing with $\mathbb{E} : R \to T$, we have analogous operations for T. Let FX be the *free monad* over $\omega\mathbf{Qbs}$ with Kleisli arrows $\text{sample} : \mathbb{1} \to F\mathbb{R}$ and $\text{score} : \mathbb{R} \to F\mathbb{1}$. It exists because, for example, $\omega\mathbf{Qbs}$ is locally presentable (see Sec. 7). The monad morphism $m_J : \underline{F} \to \underline{J}$ preserving sample and score given by initiality of F factors through the full inclusion of T in J as $m_J : F \xrightarrow{m_T} T \xrightarrow{\mathbb{E}} J$, where $m_T : F \to T$ is the unique monad morphism given by initiality. The randomisable operators are *fully definable* by sample and score, lubs, and the monad operations in the following sense:

Lemma 4.4. *The unique monad morphism from the free monad preserving sampling and conditioning is a component-wise densely strong epi $m_T : F \twoheadrightarrow T$.*

Proof (sketch). Define the Lebesgue integral $\mathbb{E}_\uparrow(w) \coloneqq \int_{\mathbb{R}} \lambda(\mathrm{d}r) w(r)$ by rescaling the normal distribution. Let $\text{normal-rng} : [0,1] \to \mathbb{R}$ be a randomisation of the gaussian with mean 0 and standard deviation 1, and let $\text{normal-pdf} : \mathbb{R} \to \mathbb{R}$ be the probability density function of this gaussian. Set $\xi \in F\mathbb{R}$ as on the left, and calculate as on the right:

$$\begin{array}{l} \xi \coloneqq \text{sample} \gg\!= \lambda s. \\ \quad \textbf{let } r = \text{normal-rng}(s) \textbf{ in} \\ \quad \text{score } \frac{1}{\text{normal-pdf } r}; \text{return } r \end{array} \qquad (m_T \xi)(w) = \int_{[0,1]} \lambda(\mathrm{d}s) \frac{w(\text{normal-rng}(s))}{\text{normal-pdf} \circ \text{normal-rng } s} = \int_{\mathbb{R}} \lambda(\mathrm{d}r) w(r)$$

For each non-empty ωqbs X, choose $x_0 \in |X|$, and consider a randomisable random operator $\alpha = \mathbb{E} \circ \Lambda f$, for some uncurried $f : \mathbb{R} \times \mathbb{R} \rightharpoonup X$. Set $\zeta(s) \coloneqq \xi \gg\!= \lambda r.\textbf{handle } f(s,r) \textbf{ with } x_0$ where the auxiliary function $\textbf{handle } - \textbf{ with } x_0 : X_\perp \to FX$; is given by: $\textbf{handle} \uparrow x \textbf{ with } x_0 \coloneqq \text{return } x$ and $\textbf{handle} \perp \textbf{ with } x_0 \coloneqq \text{score } 0; \text{return } x_0$. Then: $m_T \circ \zeta(s)(w) = \int_{\mathbb{R}} \lambda(\mathrm{d}r) f(s,r) = \alpha(s,w)$. □

Using the functoriality of the factorisation system, we deduce:

Proposition 4.5. *The monad T is the minimal full submonad of J that contains* sample *and* score.

4.5 Synthetic Measure Theory

Synthetic mathematics identifies structure and axioms from which we can recover the main concepts and results of specific mathematical theories, and transport them to new settings. Kock [2012] demonstrates that some measure theory can be reproduced for any suitable monad on a suitable category. Ścibior et al. [2017] impose this categorical structure on **Qbs** and use it to verify implementations of Bayesian inference algorithms. Our statistical powerdomain T is a very well-behaved monad. On the full subcategory $\mathbf{Qbs} \subseteq \omega\mathbf{Qbs}$, it restricts to the distribution monad given by Ścibior et al. [2017]. Like there, T makes $\omega\mathbf{Qbs}$ into a model of synthetic measure theory, enabling the interpretation of a statistical calculus. We hope to replicate their proofs for the more expressive SFPC in the future, and only state that $\omega\mathbf{Qbs}$ has this structure.

Recall the central definition to synthetic measure theory. A *measure category* $\left(C, \underline{T}\right)$ consists of a Cartesian closed category C with countable limits and coproducts; and a commutative monad $\underline{T}$ over C such that the morphisms $\langle\rangle_{T\mathbb{0}} : T\mathbb{0} \to \mathbb{1}$ and $\gg= \left[\left\langle \delta_{i,j} \right\rangle_j\right]_i : T\sum_{i\in\mathbb{N}} X_i \to \prod_{j\in\mathbb{N}} TX_j$ are invertible, where $\delta_{i,i} = \mathrm{return}^T_{X_i}$ and $\delta_{i\neq j} = x \mapsto (\langle\rangle^{-1}_{T\mathbb{0}}\langle\rangle) \gg= (\mathrm{return}^T \circ [])$. The intuition is that elements in $T\sum_i X_i$, thought of as a measure on a countable coproduct spaces are in one-to-one correspondence with tuples of measures on the component spaces. Surprisingly, this short definition guarantees that $T\mathbb{1}$ is a countably-additive semi-ring whose elements resemble *scalars*, and measures have a countably-additive structure and scalar multiplication operations. We then also have a morphism $\mathrm{total} := T\langle\rangle : TX \to T\mathbb{1}$, which we think of as assigning to each measure the scalar consisting of its *total measure*.

Theorem 4.6. *The statistical powerdomain T equips $\omega\mathbf{Qbs}$ with a measure category structure. The countable semiring of scalars is given by weights $T\mathbb{1} \cong \mathbb{W}$ with addition and multiplication. In particular, elements of TX are linear and T is a commutative monad.*

Recall that we obtain a *probabilistic* powerdomain as a full submonad $\underline{P}$ of $\underline{T}$ as the equalizer of $1, \mathrm{total} : TX \to T\mathbb{1}$, and a *sub*probabilistic powerdomain similarly. We make no use of these additional powerdomains in this work, except to note that there is a continuous normalization function $TX \to (PX)_\perp + \{\top\}$, where $\top$ is maximal, which returns $\perp$ or $\top$ if the overall measure is 0 or ∞ respectively, and a normalized probability measure otherwise. In probabilistic programming, this normalization operation is usually used at the top level to extract a posterior distribution.

4.6 Valuations on Borel-Scott Open Subsets

We compare our approach to traditional notions of probabilistic powerdomains using the following concepts. Let X be a set. A *lub-lattice of X-subsets* is a family of subsets of X that is closed under finite unions and intersections, and countable unions of ω-chains $B_0 \subseteq B_1 \subseteq \ldots$ w.r.t. inclusion.

Example 4.7 (Scott open subsets). Let P be an ωcpo. A subset $U \subseteq |P|$ is *Scott open* when it is

- upwards closed: for all $x \in U$, and $y \in \left|Q\right|$, if $x \leq y$ then $y \in U$;
- ω-inaccessible: for every ω-lub $\bigvee_{n\in\mathbb{N}} y_n \in U$, there is some $n \in \mathbb{N}$ for which $y_n \in U$.

The Scott open subsets form a lub-lattice of subsets.

Example 4.8 (Borel subsets). Let X be a qbs. A subset $B \subseteq |X|$ is *Borel* when, for every random element α, the subset $\alpha^{-1}[B] \subseteq \mathbb{R}$ is Borel. The Borel subsets form a lub-lattice of subsets.

Example 4.9 (Borel-Scott open subsets). Let X be a qbs. A subset $D \subseteq |X|$ is *Borel-Scott open* when it is Borel w.r.t. the underlying qbs and a Scott open w.r.t. the underlying ωcpo. The Borel-Scott open subsets form a lub-lattice of subsets.

The measurable functions into a Borel space are characterised as approximated by step functions (see Ex. 3.4). The same is true for ωqbses. Let X be a ωqbs, and B a Borel-Scott open subset of X. The *characteristic function* of B is given by $[- \in B] := [B.1, B^{\complement}.0] : X \to \{0,1\} \subseteq \mathbb{W}$. A morphism $f : X \to \mathbb{W}$ is a *step function* when it is a finite weighted sum of characteristic functions.

Lemma 4.10 (Approximation by Simple Functions). *Let $f : X \to \mathbb{W}$ be a ωqbs-morphism. Then, there is an ω-chain $f_n : X \to \mathbb{W}$ of step functions such that $\bigvee_{n\in\mathbb{N}} f_n = f$.*

Proof (Sketch). Define $f_n := \sum_{1 \le i \le 4^n} \frac{[-\in f^{-1}[(\frac{i}{2^n}, \infty]]]}{2^n}$. □

Let $\mathcal{O} = \left(|\mathcal{O}|, \subseteq\right)$ be a lub-lattice over a set X ordered w.r.t. inclusion. An *$\mathcal{O}$-valuation* [Lawson 1982] is a strict Scott-continuous function $v : \mathcal{O} \to \mathbb{W}$ satisfying the binary *inclusion-exclusion principle*: $v(A \cup B) + v(A \cap B) = v(A) + v(B)$ for all $A, B \in \left|\mathcal{O}\right|$. Every lub-lattice of subsets on X induces a measurable space structure on X by closing under complements and countable unions. Every valuation induces a unique measure on this measurable space. A valuation v over X is *finite* when $v(X) < \infty$, and *s-finite* when it is the pointwise lub of finite valuations. The s-finite valuations form an ω-chain-closed subset of the valuations.

The linearity and Scott continuity of expectation operators in TX now gives us the following.

Corollary 4.11. *Expectation operators are determined by their value on characteristic functions.*

This corollary tells us that elements of TX can be thought of as certain (s-finite) valuations on the lub-lattice of Borel-Scott open subsets. If P is an ωcpo, the valuations over the free ωqbs generated by P (see §3.3) coincide with traditional valuations on the Scott opens of P. When X is a qbs with the discrete order, restricting the expectation operators in TX to characteristic functions yields s-finite measures on the underlying measurable space Σ_X, and by Cor. 4.11, each such s-finite measure uniquely determines the operator. Similarly, when X and Y are qbses with the discrete order, each Kleisli arrow $X \to TY$ determines at most one s-finite kernel $\Sigma_X \rightsquigarrow \Sigma_Y$ by restricting to characteristic functions. For an sbs X, TX consists of all s-finite measures on X and M_{TX} of all s-finite kernels $\mathbb{R} \rightsquigarrow X$, by the randomisation lemma for s-finite kernels [Vákár and Ong 2018].

5 AXIOMATIC DOMAIN THEORY

Domain theory develops order-theoretic techniques for solving recursive domain equations. The theorems guaranteeing such solutions exist are technically involved. Fiore and Plotkin's *axiomatic domain theory* [1996; 1994] axiomatises categorical structure sufficient for solving such equations. They aggregate axioms of different strengths dealing with the same domain-theoretic aspects of the category at hand; the strength of one axiom can compensate for the weakness of another. This theory allows us to treat recursive domain equations in ω**Qbs** methodically.

This section is technical, and the main result is that ω**Qbs** has an expansion to a category **pBS** of ωqbses and *partial maps* between them, supporting the solution of recursive domain equations. The type-formers of SFPC, that will denote locally continuous mixed-variance functors over ω**Qbs**, then have a locally continuous extension to **pBS**, allowing us to use the solutions of recursive domain equations as denotations of recursive types.

5.1 Axiomatic Structure

We begin by isolating the structure Fiore postulates as it applies to ω**Qbs**. For the full account, see Fiore's thesis. A *(**Pos**)-domain structure* $\mathfrak{D}$ is a pair $\langle C_{\mathfrak{D}}, \mathfrak{M}_{\mathfrak{D}} \rangle$ consisting of a **Pos**-enriched category

$C_{\mathfrak{D}}$, and a locally small full-on-objects subcategory $\mathfrak{M}_{\mathfrak{D}}$ consisting solely of monomorphisms such that for every $m : D \rightarrowtail Y$ in $\mathfrak{M}_{\mathfrak{D}}$ and $f : X \to Y$ in $C_{\mathfrak{D}}$:

- the pullback of m along f exists in $C_{\mathfrak{D}}$; and
- in every pullback diagram, the pulled back morphism $f^*m : f^*D \rightarrowtail X$ is in $\mathfrak{M}_{\mathfrak{D}}$.

We call $C_{\mathfrak{D}}$ the category of *total maps* and the monos in $\mathfrak{M}_{\mathfrak{D}}$ *admissible*.

Example 5.1 (Scott open monos). A *Scott open mono* $m : P \to Q$ between ωcpos is a Scott-continuous function that is a full mono such that $m[P] \subseteq Q$ is Scott open (see Ex. 4.7). Equivalently, m is mono and m-images of Scott open subsets are Scott open. Taking the Scott open monos as admissible monos yields a domain structure **Scott** over ω**Cpo**.

To define partial maps for **Qbs** and ω**Qbs**, we use the following two examples. First, define a *strong mono* $m : X \rightarrowtail Y$ between qbses to be an injective function $m : |X| \rightarrowtail |Y|$ such that $m \circ [M_X] = M_Y \cap |m[X]|^{|\mathbb{R}|}$. The strong monos in **Qbs** coincide with the regular monos, and so are closed under pullbacks. They are also closed under composition and include all isomorphisms.

Example 5.2 (Borel open monos). Let X, Y be qbses. A *Borel open* mono $m : X \rightarrowtail Y$ is a strong mono whose image $m[X] \subseteq Y$ is Borel (see Ex. 4.8). Equivalently, a strong mono such that m-images of Borel subsets are Borel. The Borel open monos are closed under pullbacks, composition, and contain all isos. By the completeness of **Qbs** we have all pullbacks, so taking the Borel open monos as admissible monos yields a domain structure **Borel** over **Qbs**.

Example 5.3 (Borel-Scott open monos). Taking the ω**Qbs**-morphisms that are both Borel open and Scott open yields **BS**, our domain structure of interest over ω**Qbs**. It is a domain structure as by Lemma 3.9 the pullbacks in ω**Qbs** are computed using the pullbacks of ω**Cpo** and **Qbs**.

To incorporate effects, we extend Fiore's [1996] account in the following special case. When the *representability* axiom (⊣), which we define below, holds, there is a strong monad over C called the *lifting* monad $-_\perp$. We define an *effectful* domain structure to be a triple $\langle \mathfrak{D}, T, m\rangle$ consisting of a representable domain structure $\mathfrak{D}$ with finite products; a strong monad T over the category of total maps; and a strong monad morphism $m : -_\perp \to T$, thought of as *encoding* partiality using T's effects.

Our effectful domain structure consists of ω**Qbs**, together with the Borel-Scott open monos; the probabilistic power-domain of Sec. 4; and, as the resulting lifting monad is the partiality monad of §3.3, the monad morphism m is the function mapping $\perp$ to the zero measure and every other element x to the Dirac measure δ_x.

5.2 Axioms and Derived Structure

We develop the domain theory following Fiore's development and describe the axioms it validates, summarised in Fig. 10. While doing so, we recall the structure Fiore derives from these axioms.

Partial maps. Each domain structure $\mathfrak{D} = (C, \mathfrak{M})$ constructs a *category* $\mathbf{p}\mathfrak{D}$ *of partial maps*. Let X, Y be C-objects. A *partial map description* $u : X \rightharpoonup Y$ from X to Y, is a pair $u = \left(\partial_u, \overline{u}\right)$ consisting of an admissible mono $\partial_u : D_u \rightarrowtail X$ and a C-morphism $\overline{u} : D_u \to Y$. Two descriptions u and v are *equivalent*, $u \equiv v$, when there is an isomorphism $i : D_u \xrightarrow{\cong} D_v$ satisfying: $\overline{v} \circ i = \overline{u}$ and $\partial_v \circ i = \partial_u$.

A *partial map* $u : X \rightharpoonup Y$ is the equivalence class of a description, bearing in mind that $\mathfrak{M}$ is locally small. E.g., for **BS**, choose inclusions as the canonical representatives $\partial_u : D_u \subseteq X$, which uniquely determine the description. Partial maps form a category $\mathbf{p}\mathfrak{D}$, with identities given by $[\mathrm{id}, \mathrm{id}]$, and the composition $v \circ u : X \xrightarrow{u} Y \xrightarrow{v} Z$ via pullback (see Fig. 11, left). We have an identity-on-objects, faithful functor $J : C \to \mathbf{p}\mathfrak{D}$ mapping each total map $f : X \to Y$ to $[\mathrm{id}, f] : X \rightharpoonup Y$.

Structure		**Axioms**			
$C_{\mathfrak{D}}$	total map category ω**Qbs**	(⊣)	every object has a partial map classifier $\uparrow_X : X \to X_\perp$	($\to_\leq$)	$C_{\mathfrak{D}}$ has locally monotone exponentials
$f \leq g$	**Pos**-enrichment pointwise order	(fup)	every admissible mono is full and upper-closed	(+)	locally continuous total coproducts
$\mathfrak{M}_{\mathfrak{D}}$	admissible monos Borel-Scott opens	($\dashv_\leq$)	$\llcorner - \lrcorner$ is locally monotone	(?!)	$\mathbb{0} \to \mathbb{1}$ is admissible
T	monad for effects power-domain	($C_\vee$)	$C_{\mathfrak{D}}$ is ω**Cpo**-enriched	($\times_\vee$)	$C_{\mathfrak{D}}$ has a locally continuous products
m	partiality encoding $m : -_\perp \to T$, $\perp \mapsto \underline{0}$	(U)	ω-colimits behave uniformly (Lemma 5.8)	(CL)	$C_{\mathfrak{D}}$ is cocomplete
		($\mathbb{1}$)	$C_{\mathfrak{D}}$ has a terminal object	($T_\vee$)	T is locally continuous

Derived axioms/structure					
$\mathbf{p}\mathfrak{D}$	partial map category	($\otimes$)	$\mathbf{p}\mathfrak{D}$ has partial products	($\mathbf{p}CL$)	$\mathbf{p}\mathfrak{D}$ is cocomplete
$-_\perp$	partiality monad	($\otimes_\vee$)	($\otimes$) is locally continuous	($\mathbf{p}+_\vee$)	$\mathbf{p}\mathfrak{D}$ has locally continuous partial coproducts
($\dashv_\vee$)	the adjunction $J \dashv L$ is locally continuous	($\to_\vee$)	$C_{\mathfrak{D}}$ has locally continuous exponentials	(BC)	$J : C \hookrightarrow \mathbf{p}\mathfrak{D}$ is a bilimit compact expansion
($\mathbf{p}_\vee$)	$\mathbf{p}\mathfrak{D}$ is ω**Cpo**-enriched	($\rightleftharpoons_\vee$)	$\mathbf{p}\mathfrak{D}$ has locally continuous partial exponentials		
($\mathbb{1}_\leq$)	$\mathbf{p}\mathfrak{D}$ has a partial terminal				

Fig. 10. The axiomatic domain theory of ω**Qbs** and its probabilistic power-domain

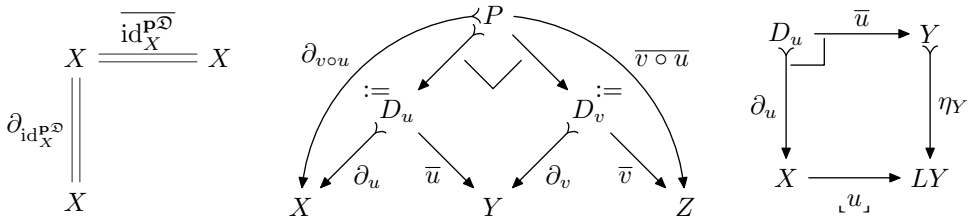


Fig. 11. Partial identities and composition (left) and characteristic maps (right).

Representability. Given a domain structure, a *classifier of partial maps* is a collection of admissible monos $\langle \uparrow_X : X \rightarrowtail LX \rangle_{X \in C}$, indexed by the objects of C, such that for every Y, and every partial map $f : X \rightharpoonup Y$, there is a unique (total) map $\llcorner f \lrcorner : X \to LY$ such that Fig. 11 (right) is a pullback square. We call $\llcorner f \lrcorner$ the *total representation* of f. Each classifier $\uparrow$ of partial maps induces a right adjoint $J \dashv L : \mathbf{p}\mathfrak{D} \to C$ with $\uparrow$ as unit. We give two well-known and two novel examples.

Example 5.4 (partiality in **Set***).* The collection $\langle \iota_1 : X \to X + \{\perp\} \rangle_{X \in \mathbf{Set}}$ classifies the partial maps of **Fn**, the domain structure over **Set** with injections as admissible monos. The representation of a partial function maps the elements outside the domain to $\perp$. The induced adjunction is isomorphic to the Kleisli resolution of the partiality monad $(-) + \{\perp\}$.

Example 5.5 (partiality in ω**Cpo***).* Let $X_\perp$ be the lifting of the ωcpo X by adjoining a new bottom element. The collection $\langle \uparrow : X \to X_\perp \rangle_{X \in \omega\mathbf{Cpo}}$ classifies the **Scott**-partial maps with $\llcorner - \lrcorner$ as in **Fn**. The induced adjunction is isomorphic to the Kleisli resolution of the lifting monad $(-)_\perp$.

Example 5.6 (partiality in **Qbs***).* The collection $\langle \iota_1 : X \to X + \{\perp\} \rangle_{X \in \mathbf{Qbs}}$ classifies the **Borel**-partial maps with $\llcorner - \lrcorner$ as in **Fn**. The induced adjunction is isomorphic to the Kleisli resolution of the partiality monad $(-) + \{\perp\}$.

Example 5.7 (partiality in $\omega\mathbf{Qbs}$). The collection $\left\langle \uparrow : X \to X_\perp \right\rangle_{X \in \mathbf{Qbs}}$ classifies the **BS**-partial maps with $\ulcorner - \urcorner$ as in **Fn**. The induced adjunction is the Kleisli resolution of the monad $(-)_\perp$ of §3.3.

Enrichment. The order enrichment of $C_{\mathrm{BS}} = \omega\mathbf{Qbs}$ is an $\omega\mathbf{Cpo}$-enrichment. Moreover, the partial map category inherits a potential **Pos**-enrichment: for $u, v : X \rightharpoonup Y$, write $u \sqsubseteq v$ to mean that there is some $i : D_u \to D_v$ such that $\partial_u = \partial_v \circ i$ and $\overline{u} \leq \overline{v} \circ i$. The isomorphism $\mathbf{p}\mathfrak{D} \cong \omega\mathbf{Qbs}_{-\perp}$ respects $\sqsubseteq$ and $\leq$, i.e., $u \sqsubseteq v : X \rightharpoonup Y$ iff $\ulcorner u \urcorner \leq \ulcorner v \urcorner$ as morphisms $X \to Y_\perp$ in $\omega\mathbf{Qbs}$, and so $\mathbf{p}\mathfrak{D}$ is $\omega\mathbf{Cpo}$-enriched. We can also deduce this fact from more fundamental axioms. First, every admissible mono is full and its image is upper-closed. As a consequence of Fiore's Prop. 4.2.4, the order $\sqsubseteq$ is a partial order and **Pos**-enriches $\mathbf{p}\mathfrak{D}$. Denote the inverse map to representation by $\ulcorner - \urcorner : C_{\mathrm{BS}}(X, Y_\perp) \to \mathbf{pBS}(X, Y)$. Because $\ulcorner - \urcorner$ is monotone, the adjunction $J \dashv L : \mathbf{p}\mathfrak{D}$ is locally monotone [*ibid.*, Prop. 4.5.4], and, as a consequence $\mathbf{p}\mathfrak{D}$ is $\omega\mathbf{Cpo}$-enriched [*ibid.*, Prop. 4.5.3].

Uniformity. While by Cor. 7.2 $\omega\mathbf{Qbs}$ has local ω-lubs, but these lubs can behave pathologically [*ibid.*, Sec. 4.3.2]. The following *uniformity* axiom avoids such pathologies.

Lemma 5.8 (Uniformity). *Let $\langle i_{n+1} : D_n \rightarrowtail D_{n+1} \rangle_{n \in \mathbb{N}}$ be an ω-chain of Borel-Scott open monos.*

- *Every colimiting cocone $\left(D, \left\langle \mu_n : D_n \to D \right\rangle_{n \in \mathbb{N}}\right)$ consists of Borel-Scott opens.*
- *The mediating morphism into any other cocone of Borel-Scott opens is also Borel-Scott open.*

Unit type. As $\omega\mathbf{Qbs}$ has a terminal object $\mathbb{1}$, the partial maps have $\mathbb{1}$ as a *partial* terminal object: every hom-poset $\mathbf{pBS}(A, \mathbb{1})$ has a unique maximal morphism [*ibid.*, p. 84].

Product types. As $\omega\mathbf{Qbs}$ has binary products $X \times Y$, we can extend the binary product functor $(\times) : C_{\mathfrak{D}} \times C_{\mathfrak{D}} \to C_{\mathfrak{D}}$ to a *partial-product* functor $\otimes : \mathbf{p}\mathfrak{D} \times \mathbf{p}\mathfrak{D} \to \mathbf{p}\mathfrak{D}$ [*ibid.*, Prop. 5.1.1]. Because $\omega\mathbf{Qbs}$ has exponentials, the functor $- \times X$ preserves colimits. As a consequence, as products are locally continuous, and as axioms (U), $(\mathbf{p}_\vee)$ hold, $(\otimes)$ is locally continuous [*ibid.*, Prop. 5.1.3].

Variant types. By Cor. 7.2, $\omega\mathbf{Qbs}$ is cocomplete. Using axioms $(\dashv)$,$(\mathbb{1})$ we deduce that $\mathbf{p}\mathfrak{D}$ is cocomplete [*ibid.*, Theorem 5.3.14], and colimiting cocones of total diagrams comprise of total maps and are colimiting in $C_{\mathfrak{D}}$ [*ibid.*, Prop. 5.2.4]. As the coproducts of $\omega\mathbf{Qbs}$ are locally continuous, so are the coproducts in **pBS** [*ibid.*, Prop. 5.3.13]. As a consequence, the locally continuous finite-coproduct functor $\sum_{i \in I} : \omega\mathbf{Qbs}^I \to \omega\mathbf{Qbs}$ extends to a locally continuous functor $\coprod_{i \in I} : \mathbf{p}\mathfrak{D}^I \to \mathbf{p}\mathfrak{D}$ on partial maps [*ibid.*, remark following Cor. 5.3.10].

Effectful function types. The following development is new, as Fiore only considered the partiality effect. As we now saw, when the representability axioms $(\dashv)$,$(\dashv_\leq)$, and the enrichment axioms $(\mathbf{p}_\vee)$,$(\otimes_\vee)$ hold, we have a locally continuous strong lifting monad $-_\perp$ over C. Recall the additional structure given in an effectful domain structure $\langle \mathfrak{D}, T, m \rangle$, namely the locally monotone strong monad T over $C_{\mathfrak{D}}$, and the strong monad morphism $m : -_\perp \to T$. We further assume that T is locally continuous, which we call axiom $(T_\vee)$. Thm. 4.3 validates it for the statistical powerdomain.

We have an identity-on-objects locally continuous functor $\ulcorner - \urcorner_T : \mathbf{p}\mathfrak{D} \to C_T$, from the category of partial maps into the Kleisli category for T, given for every partial map $u : X \rightharpoonup Y$ by:

$$\ulcorner u \urcorner_T : X \xrightarrow{\ulcorner u \urcorner} Y_\perp \xrightarrow{m} TY$$

When axiom $(\to_\leq)$ holds, the composite functor $(\odot X) : \ulcorner J - \otimes X \urcorner_T : C \to C_T$ has a right adjoint:

$$(X \multimap_T -) : C_T \to C \quad X \multimap_T Y := (TY)^X \quad X \multimap_T v := \lambda X. \left((TY_1)^X \times X \xrightarrow{\mathrm{eval}} TY_1 \xrightarrow{\gg= v} TY_2 \right)$$

for every $v : Y_1 \to TY_2$. Axiom $(\to_\leq)$ implies the exponential adjunction is locally continuous, and as a consequence, $(\multimap_T)$ is locally continuous. We use it to extend Kleisli exponentiation

$(- \Rightarrow T-) : C^{\mathrm{op}} \times C \to C$ to a locally continuous functor $(\rightharpoonup_T) : \mathbf{p}\mathfrak{D}^{\mathrm{op}} \times \mathbf{p}\mathfrak{D} \to \mathbf{p}\mathfrak{D}$ by setting, for every $u : X_2 \rightharpoonup X_1$ and $v : Y_1 \rightharpoonup vY_2$:

$$u \rightharpoonup_T v := J\lambda X. \left((TY_1)^{X_1} \times X_2 \xrightarrow{\imath \mathrm{id} \otimes u \jmath_T} T((TY_1)^{X_1} \times X_1) \xrightarrow{\gg= \mathrm{eval}} TY_1 \xrightarrow{\gg= \imath v \jmath_T} TY_2 \right)$$

Recursive types. To solve recursive domain equations, we synthesise axiomatic domain theory with Levy's more modern account [2004]. Recall that an embedding-projection-pair (ep-pair) $u : A \leftrightharpoons B$ in an $\omega\mathbf{Cpo}$-enriched category $\mathcal{B}$ is a pair consisting of a $\mathcal{B}$-morphism $u^{\mathrm{e}} : A \to B$, the *embedding*, and a $\mathcal{B}$-morphism $u^{\mathrm{p}} : B \to A$, the *projection*, such that $e \circ p \leq \mathrm{id}$ and $p \circ e = \mathrm{id}$. An *embedding* $u : A \hookrightarrow B$ is the embedding part of some ep-pair $A \leftrightharpoons B$. Every embedding $u : A \hookrightarrow B$ in a partial map category with axiom $(\mathbf{p}_\vee)$ is a total map $u : A \to B$ [Fiore 1996, Prop. 5.4.2].

An *ω-chain of ep-pairs* $\left(\langle A_n \rangle_{n\in\mathbb{N}}, \langle a_n \rangle_{n\in\mathbb{N}}\right)$ in $\mathcal{B}$ consists of a countable sequence of objects A_n and a countable sequence of ep-pairs $a_n : A_n \leftrightharpoons A_{n+1}$. A *bilimit* (D, d) of such an ω-chain consists of an object D and a countable sequence of ep-pairs $d_n : A_n \leftrightharpoons D$ such that, for all $n \in \mathbb{N}$, $d_{n+1} \circ a_n = d_n$, and $\bigvee_{n\in\mathbb{N}} d_n^{\mathrm{e}} \circ d_n^{\mathrm{p}} = \mathrm{id}_D$. The celebrated *limit-colimit coincidence* [Smyth and Plotkin 1982] states that the bilimit structure is equivalent to a colimit structure (D, d^{e}) for $(\langle A_n \rangle , \left\langle a_n^{\mathrm{e}} \right\rangle)$, in which case d_n^{p} are uniquely determined, and similarly equivalent to a limit structure (D, d^{p}) for $(\langle A_n \rangle , \left\langle a_n^{\mathrm{p}} \right\rangle)$, in which case d_n^{e} are uniquely determined. As we saw, the partial map category **pBS** has all colimits (derived axiom $(\mathbf{p}CL)$), and so **pBS** has bilimits of ω-chains of ep-pairs.

A *zero object* is an object that is both initial and terminal. An *ep-zero object* in an $\omega\mathbf{Cpo}$-category is a zero object such that every morphism into it is an embedding and every morphism out of it is a projection. We say that axiom (?!) holds in a domain structure $\mathfrak{D}$ in which C has both an initial object $\mathbb{0}$ and terminal object $\mathbb{1}$, when the unique morphism $\mathbb{0} \to \mathbb{1}$ is an admissible mono. The Borel-Scott domain structure satisfies (?!). When axioms $(+)$,$(\mathbb{1})$,$(\times_\vee)$,$(\to_\leq)$,(CL),$(?!)$,$(\dashv)$, and $(\dashv_\leq)$ hold, $\mathbf{p}\mathfrak{D}$ has $\mathbb{0}$, the initial object of C, as an ep-zero object. So in **pBS** the empty ωqbs is an ep-zero.

A *bilimit compact category* is an $\omega\mathbf{Cpo}$-category $\mathcal{B}$ with an ep-zero and ep-pair ω-chain bilimits. So **pBS** is bilimit compact. When $\mathcal{A}, \mathcal{B}$ are bilimit compact, every locally continuous, mixed-variance functor $F : \mathcal{A}^{\mathrm{op}} \times \mathcal{B}^{\mathrm{op}} \times \mathcal{A} \times \mathcal{B} \to \mathcal{B}$ has a parameterised solution to the recursive equation $\mathrm{roll} : F(A, X, A', X) \xrightarrow{\cong} X$, for every A, A' in $\mathcal{A}$, qua the bilimit $\left(\mu B.F(A, B, A', B), d_{F,A,A'}\right)$ of

$$\mathbb{0} \leftrightharpoons F(A, \mathbb{0}, A', \mathbb{0}) \overset{F^{\mathrm{ep}}(A, \leftrightharpoons, A', \leftrightharpoons)}{\leftrightharpoons} F(A, F(A, \mathbb{0}, A', \mathbb{0}), A', F(A, \mathbb{0}, A', \mathbb{0})) \leftrightharpoons \cdots \leftrightharpoons F^n(A, A') \leftrightharpoons \cdots$$

The solution is minimal in the sense of Pitts [1996], and we denote the inverse to roll by unroll. The assignments $\mu B.F(A, B, A', B)$ extend to a mixed-variance functor $\mu B.F(-, B, -, B) : \mathcal{A}^{\mathrm{op}} \times \mathcal{A} \to \mathcal{B}$ by $\mu B.F(f, B, g, B) := \bigvee_n d_n^{\mathrm{e}} \circ F^n(f, g) \circ d_n^{\mathrm{p}}$.

Finally, a *bilimit compact expansion* $J : C \hookrightarrow \mathcal{B}$ is a triple consisting of an $\omega\mathbf{Cpo}$-category C, a bilimit compact category $\mathcal{B}$; and an identity-on-objects, locally continuous, order reflecting functor $J : C \to \mathcal{B}$ such that, for every ep-pair ω-chains (A, a), (B, b) in $\mathcal{B}$, their bilimits (D, d), (E, e), and countable collection of C-morphisms $\langle \alpha_n : A_n \to B_n \rangle_{n\in\mathbb{N}}$ such that for all n:

$$J\alpha_{n+1} \circ a_n^{\mathrm{e}} = b_n^{\mathrm{e}} \circ J\alpha_n, \qquad J\alpha_n \circ a_n^{\mathrm{p}} = b_n^{\mathrm{p}} \circ J\alpha_{n+1}$$

(i.e., $J\alpha : (A, a^{\mathrm{e}}) \to (B, b^{\mathrm{e}})$ and $J\alpha : (A, a^{\mathrm{p}}) \to (B, b^{\mathrm{p}})$ are natural transformations), there is a C-morphism $f : A \to B$ such that $Jf = \bigvee_n e_n^{\mathrm{e}} \circ \alpha_n \circ d_n^{\mathrm{p}}$. The motivation for this definition: given two bilimit compact expansions $I : \mathcal{D} \hookrightarrow \mathcal{A}$, $J : C \hookrightarrow \mathcal{B}$, and two locally continuous functors

$$F : \mathcal{D}^{\mathrm{op}} \times C^{\mathrm{op}} \times \mathcal{D} \times C \to C \quad G : \mathcal{A}^{\mathrm{op}} \times \mathcal{B}^{\mathrm{op}} \times \mathcal{A} \times \mathcal{B} \to \mathcal{B} \quad \text{s.t.} \quad G \circ I^{\mathrm{op}} \times J^{\mathrm{op}} \times I \times J = J \circ F$$

the functor $\mu B.G(-,B,-,B) : \mathcal{A}^{\mathrm{op}} \times \mathcal{A} \to \mathcal{B}$ restricts to $\mu B.G(J^{\mathrm{op}}-,B,J-,B) : \mathcal{D}^{\mathrm{op}} \times \mathcal{D} \to C$. Bilimit compact expansions are closed under products and opposites [Levy 2004].

Returning to axiomatic domain theory, when axioms $(\dashv)$,$(\dashv_{\leq})$,$(\mathbb{1})$,$(\times_{\bigvee})$,$(\to_{\leq})$,(CL), and (?!) hold, the embedding $J : C \hookrightarrow \mathbf{p}\mathfrak{D}$ is a bilimit compact expansion of total maps to partial maps. To show the third condition in the definition of expansion, note that embeddings in $\mathbf{p}\mathfrak{D}$ are total, and we can use the limit-colimit coincidence to reflect the two bilimits along J to the colimits in C, and find a mediating morphism $f : D \to E$. This morphism maps to the mediating morphism in $\mathbf{p}\mathfrak{D}$, which is precisely $\bigvee_n e^{\mathrm{e}}_n \circ \alpha_n \circ d^{\mathrm{p}}_n$.

5.3 Summary: Semantics of Types

Using axiomatic domain theory, we constructed an expansion $J : \omega\mathbf{Qbs} \hookrightarrow \mathbf{pBS}$ to a suitable category of partial maps. We use it to interpret SFPC's type-system. We define two interpretations for type-variable contexts Δ as $\omega\mathbf{Cpo}$-categories, and two locally continuous interpretations of well-kinded types $\Delta \vdash_{\mathrm{k}} \tau : \mathrm{type}$: a total map interpretation $[\![-]\!]$ and its extension $\{\!|-|\!\}$ to partial maps, i.e. $J \circ [\![-]\!] = \{\!|-|\!\} \circ J^{\mathrm{op}} \times J$:

$$[\![\Delta]\!] := \prod_{\alpha\in\Delta} \omega\mathbf{Qbs} \quad \{\!|\Delta|\!\} := \prod_{\alpha\in\Delta} \mathbf{pBS} \qquad [\![\tau]\!] : [\![\Delta]\!]^{\mathrm{op}} \times [\![\Delta]\!] \to \omega\mathbf{Qbs} \quad \{\!|\tau|\!\} : \{\!|\Delta|\!\}^{\mathrm{op}} \times \{\!|\Delta|\!\} \to \mathbf{pBS}$$

$$\begin{array}{llll} [\![\alpha]\!] := \pi_\alpha & \{\!|\alpha|\!\} := \pi_\alpha & & \\ [\![\underline{\mathbb{R}}]\!] := \mathbb{R} & \{\!|\underline{\mathbb{R}}|\!\} := \mathbb{R} & [\![\tau * \sigma]\!] := [\![\tau]\!] \times [\![\sigma]\!] & \{\!|\tau * \sigma|\!\} := \{\!|\tau|\!\} \otimes \{\!|\sigma|\!\} \\ [\![\underline{\mathbf{1}}]\!] := \mathbb{1} & \{\!|\mathbf{1}|\!\} := \mathbb{1} & [\![\tau \to \sigma]\!] := [\![\tau]\!] \Rightarrow T\,[\![\sigma]\!] & \{\!|\tau \to \sigma|\!\} := \{\!|\tau|\!\} \rightharpoonup_T \{\!|\sigma|\!\} \end{array}$$

$$\left[\!\!\left[\begin{Bmatrix} \ell_1\,\tau_1 \\ \vert\ \dots \\ \vert\ \ell_{\mathrm{n}}\,\tau_n \end{Bmatrix} \right]\!\!\right] := \sum_{i=1}^{n} [\![\tau_i]\!] \qquad \left\{\!\!\left| \begin{Bmatrix} \ell_1\,\tau_1 \\ \vert\ \dots \\ \vert\ \ell_{\mathrm{n}}\,\tau_n \end{Bmatrix} \right|\!\!\right\} := \coprod_{i=1}^{n} \{\!|\tau_i|\!\}$$

$$\begin{array}{ll} [\![\mu\alpha.\tau]\!] := \mu\alpha. & \{\!|\mu\alpha.\tau|\!\} := \mu\alpha. \\ \quad \{\!|\tau|\!\}(J-,\alpha,J-,\alpha) & \quad \{\!|\tau|\!\}(-,\alpha,-,\alpha) \end{array}$$

6 SEMANTICS FOR SFPC

Operational semantics for statistical probabilistic calculi such as SFPC require a mathematically technical development. The difficulty comes from dealing with continuous distributions over programs that manipulate continuous data. We follow the recipe previously employed by Staton et al. [2016] (others have used broadly similar methods, e.g. Borgström et al. [2016]). Each program phrase involving the real numbers $r_1,\dots,r_n$ can be represented as an open term t with free variables $x_1,\dots,x_n$ each used linearly and in sequence, together with the tuple $\langle r_1,\dots,r_n\rangle \in \mathbb{R}^n$. This separation lets us equip the abstract syntax with a measurable space structure, and define a big-step operational semantics as kernels between these syntactic spaces [Staton et al. 2016].

6.1 Measure Theoretic Preliminaries

Constructing spaces. Every set is the carrier of a *discrete* measurable space, in which all subsets are measurable. We will typically only consider discrete spaces of countable carriers, as these are precisely the discrete standard Borel spaces. Let I be a set indexing some measurable spaces X_i. The measurable subsets of the disjoint union $\sum_{i\in I} X_i$ are the disjoint unions $\sum_{i\in I} A_i$ of measurable subsets A_i in X_i. When I is countable, the measurable subsets of the cartesian product $\prod_{i\in I} X_i$ are given by taking the boxes $\prod_{i\in I} A_i$, with each A_i measurable in X_i, and closing under countable unions and complements. These three constructions are the free measurable space over a set, categorical coproduct, and categorical countable product in the category of measurable spaces.

Constructing kernels. The *Dirac* kernel $\delta_- : X \rightsquigarrow X$ assigns to each $x \in |X|$ its *Dirac distribution* defined by $\delta_x(A) = 1$ for $x \in A$ and $\delta_x(A) = 0$ otherwise. Every measurable function $f : X \to Y$

inudces a kernel $f : X \rightsquigarrow Y$ by setting $f(x, A) := \delta_{f(x)}(A)$. Each kernel $k : X \rightsquigarrow Y$ acts on measures μ over X yielding the unique measure $\mu \gg\!= k$ over Y satisfying, for all measurable $\varphi : Y \to [0, \infty]$:

$$\int_Y (\mu \gg\!= k)(\mathrm{d}y)\varphi(y) = \int_X \mu(\mathrm{d}x) \int_Y k(x, \mathrm{d}y)\varphi(y)$$

Moreover, given $k : X \rightsquigarrow Y$ and $l : Y \rightsquigarrow Z$, the mapping $x \mapsto k(x) \gg\!= l$ is a kernel $k \gg l : X \rightsquigarrow Z$. The s-finite kernels are closed under this composition [Kallenberg 2017; Staton 2017].

6.2 Syntax Spaces

Consider the set of terms of type τ in variable context Γ, $\mathsf{Trm}^{\Gamma \vdash \tau} := \{t | \Gamma \vdash t : \tau\}$. The set of *values* of type τ in context Γ consists of the subset $\mathsf{Val}^{\Gamma \vdash \tau} \subseteq \mathsf{Trm}^{\Gamma \vdash \tau}$ given inductively by:

$$v, w, u ::= x \mid \underline{r} \mid () \mid (v, w) \mid \tau.\ell\, v \mid \tau.\mathbf{roll}\,(v) \mid \lambda x : \tau.t \qquad \text{values.}$$

We want to equip $\mathsf{Trm}^{\Gamma \vdash \tau}$ and $\mathsf{Val}^{\Gamma \vdash \tau}$ with a measurable space structure. Let Δ range over *non-symmetric linear* variable contexts of type $\underline{\mathbb{R}}$, namely finite sequences of variables, and let *mixed* variable contexts $\Gamma; \Delta$ be pairs of variable context Γ and linear contexts Δ with disjoint sets of variables. Each such mixed context can be turned into an ordinary variable context $\Gamma, \Delta : \underline{\mathbb{R}}$ by treating the identifiers in Δ as variables of type $\underline{\mathbb{R}}$. We will use the identifiers $\square, \square_1, \square_2$ for the variables in Δ. A term *template* is a term that does not involve nullary primitives $\underline{r}$. We refine the SFPC type-system to judgements of the form $\Gamma; \Delta \vdash t : \tau$ by treating Δ non-symmetric linearly, e.g.:

$$\frac{}{\Gamma; \square \vdash \square : \underline{\mathbb{R}}} \qquad \frac{\Gamma; \Delta_1 \vdash t : \sigma \to \tau \qquad \Gamma; \Delta_2 \vdash s : \sigma}{\Gamma; \Delta_1, \Delta_2 \vdash t\, s : \tau}$$

Let $\mathsf{TTrm}^{\Gamma;\Delta \vdash \tau} \subseteq \mathsf{Trm}^{\Gamma, \Delta; \underline{\mathbb{R}} \vdash \tau}$ be the set of well-formed templates $\Gamma; \Delta \vdash t : \tau$ and $\mathsf{TVal}^{\Gamma;\Delta \vdash \tau} \subseteq \mathsf{TVal}^{\Gamma;\Delta \vdash \tau}$ be the set of well-formed template values. Given any $\Delta = \square_1, \ldots, \square_k$, and $r_1, \ldots, r_k \in \mathbb{R}$, we define the following function

$$-[r_1, \ldots, r_n] : \mathsf{TTrm}^{\Gamma;\Delta \vdash \tau} \to \mathsf{Trm}^{\Gamma \vdash \tau} \qquad t[r_1, \ldots, r_n] := t[\square_1 \mapsto \underline{r_1}, \ldots, \square_k \mapsto \underline{r_k}]$$

Combining these functions we have a bijection $\sum_{n \in \mathbb{N},\, t \in \mathsf{TTrm}^{\Gamma;\square_1,\ldots,\square_n \vdash \tau}} \mathbb{R}^n \cong \mathsf{Trm}^{\Gamma \vdash \tau}$.

Example 6.1. The Bayesian linear regression program from Fig. 1(a) is represented by:

$$\left(\begin{array}{ll} & \mathbf{let}\, a = \underline{\mathit{normal\text{-}rng}}\,(\square_1, \square_2)\, \mathbf{in} \\ 11, & \quad \mathbf{score}(\underline{\mathit{normal\text{-}pdf}}(\square_3 \mid a * \square_4, \square_5)); \\ & \quad \mathbf{score}(\underline{\mathit{normal\text{-}pdf}}(\square_6 \mid a * \square_7, \square_8)); \\ & \quad \mathbf{score}(\underline{\mathit{normal\text{-}pdf}}(\square_9 \mid a * \square_{10}, \square_{11})); a \end{array}, \begin{array}{l} (0, 2, \\ 1.1, 1, 0.25, \\ 1.9, 2, 0.25, \\ 2.7, 3, 0.25) \end{array}\right)$$

The representation $\sum_{n \in \mathbb{N},\, t \in \mathsf{TTrm}^{\Gamma;\square_1,\ldots,\square_n \vdash \tau}} \mathbb{R}^n$ has a canonical measurable space structure, combining the Borel σ-algebra on $\mathbb{R}^n$ and the coproduct σ-algebra. As a consequence, we equip $\mathsf{Trm}^{\Gamma \vdash \tau}$ with the measurable space structure making the representation a measurable isomorphism. Similarly, we equip $\mathsf{Val}^{\Gamma \vdash \tau}$ with a measurable space structure. Summing over all types, we get the measurable spaces of closed terms, Trm, and closed values, Val.

6.3 Operational Semantics

We define a structural operational semantics as a kernel $\Downarrow : \mathsf{Trm} \rightsquigarrow \mathsf{Val}$. As usual, the operational semantics is *not* compositional. E.g., the semantics of function application is not defined in terms of that of a subterm, but of a substituted subterm. As a consequence, we construct an increasing sequence of kernels $\Downarrow_n : \mathsf{Trm} \rightsquigarrow \mathsf{Val}$ indexed by the natural numbers and set $\Downarrow := \bigvee_n \Downarrow_n$. At level 0, the evaluation kernel is the least kernel, namely $t \Downarrow_0 A := 0$ for every closed term t and measurable

$$t \Downarrow_0 A := 0 \qquad \frac{t_1 \Downarrow_n \underline{r_1} \quad \dots \quad t_k \Downarrow_n \underline{r_k}}{\underline{f}(t_1,\dots,t_k) \Downarrow_n \underline{f}(\underline{r_1},\dots,\underline{r_n})} \qquad (\mathbf{sample} \Downarrow_n) := \mathbb{U}_{[0,1]} \gg\!= \lambda r.\delta_{\underline{r}} \qquad (\mathbf{score}(t) \Downarrow_n) := (t \Downarrow_n) \gg\!= \lambda \underline{r}.|r| \cdot \delta_{()}$$

$$\frac{t \Downarrow_n \underline{0} \quad s_1 \Downarrow_n v}{\mathbf{match}\ t\ \mathbf{with}\ \{\underline{0} \to s_1 \mid \_ \to s_2\}) \Downarrow_n v} \qquad \frac{t \Downarrow_n \underline{r} \quad s_2 \Downarrow_n v}{\mathbf{match}\ t\ \mathbf{with}\ \{\underline{0} \to s_1 \mid \_ \to s_2\}) \Downarrow_n v}(r \neq 0)$$

$$\frac{}{v \Downarrow_n v}(v = (), \lambda x:\tau.t) \qquad \frac{t \Downarrow_n () \quad s \Downarrow_n v}{\mathbf{match}\ t\ \mathbf{with}\ () \to s \Downarrow_n v} \qquad \frac{t \Downarrow v \quad s \Downarrow w}{(t,s) \Downarrow_n (v,w)} \qquad \frac{t \Downarrow_n v}{\tau.\ell\ t \Downarrow_n \tau.\ell\ v}$$

$$\frac{t \Downarrow_n (w,u) \quad s[x \mapsto w, y \mapsto u] \Downarrow_{n-1} v}{\mathbf{match}\ t\ \mathbf{with}\ (x,y) \to s \Downarrow_n v} \qquad \frac{t \Downarrow_n \ell_i\, w \quad s_i[x_i \mapsto w] \Downarrow_{n-1} v}{\mathbf{match}\ t\ \mathbf{with}\ \{\ell_1\ x_1 \to s_1 \mid \cdots \mid \ell_m x_m \to s_m\} \Downarrow_n v} \qquad \frac{t \Downarrow_n v}{\tau.\mathbf{roll}\,(t) \Downarrow_n \tau.\mathbf{roll}\, v}$$

$$\frac{t \Downarrow_n \tau.\mathbf{roll}\, w \quad s[x \mapsto w] \Downarrow_{n-1} v}{\mathbf{match}\ t\ \mathbf{with}\ \mathbf{roll}\ x \to s \Downarrow_n v} \qquad \frac{t \Downarrow_n \lambda x:\tau.r \quad s \Downarrow_n w \quad r[x \mapsto w] \Downarrow_{n-1} v}{t\ s \Downarrow_n v}$$

Fig. 12. SFPC big-step operational semantics ($n > 0$)

$A \subseteq \mathsf{Val}$. For positive levels $n > 0$, we use the following three notational conventions that highlight that the semantics is a standard adaptation of the more familiar operational semantics. First:

$$\frac{k_1(t)\, w_1 \quad k_2(t,w_1)\, w_2 \quad \dots \quad k_n(t,w_1,\dots,w_n)\, v}{l(t)\, f(t,w_1,\dots,w_n,v)} \quad \text{means} \quad \begin{aligned} l(t) := k_1(t) &\gg\!= \lambda w_1. \\ k_2(t,w_1) &\gg\!= \lambda w_2. \ \dots \\ k_n(t,w_1,\dots,w_n) &\gg\!= \lambda v. \\ \delta_{f(t,w_1,\dots,w_n,v)} & \end{aligned}$$

Second: $l(t) := k_1(t)\ \ l(t) := k_2(t)$ means $l(t) := k_1(t) + k_2(t)$

i.e., we sum overlapping definitions. Finally, if one of the intermediate kernels k_i is undefined (on a measurable subset), we define $l(t)$ to have measure 0 on this subset.

Fig. 12 presents the indexed evaluation kernels. Primitives evaluate their arguments left-to-right. Sampling and conditioning evaluate using the interpreter's statistical sampling and conditioning primitives. We could equivalently use a lower-level interpreter supporting only (sub)-probabilistic sampling and maintaining an aggregate weight, i.e., a kernel $\Downarrow'_n \colon \mathsf{Trm} \rightsquigarrow \mathbb{R} \times \mathsf{Val}$ and the rule:

$$\frac{t \Downarrow'_n (r_1, \underline{r_2})}{\mathbf{score}\,(t) \Downarrow'_n (r_1 \cdot |r_2|\,, ())}$$

and the other rules changing analogously. The remaining rules are standard. The index decreases only when we apply the kernel non-compositionally in the elimination forms for binary products, variants, recursive types, and function application. The evaluation kernel $\Downarrow$ is s-finite, as the s-finite kernels are closed under composition and lubs [Staton 2017].

6.4 Contextual Equivalence

The operational semantics lets us compare the meaning of terms using the following standard notions. Given variable context Γ_1, Γ_2, we say that Γ_2 *extends* Γ_1, and write $\Gamma_2 \geq \Gamma_1$ when, for all $(x:\tau) \in \Gamma_1$, we have $(x:\tau) \in \Gamma_2$. *Program contexts of type σ with a hole $-$ of type* $\Gamma \vdash \tau$ are terms $C[\Gamma \vdash - : \tau]$ of type σ with a single variable of type τ, where this variable $-$ always occurs inside the term in contexts $\Gamma' \geq \Gamma$. Write $C[t]$ for the *capturing* substitution $C[\Gamma \vdash - : \tau][- \mapsto t]$.

Two terms $t, s \in \mathsf{Trm}^{\Gamma \vdash \tau}$ are in the *contextual preorder* $t \precsim s$ when for all program contexts $C[\Gamma \vdash - : \tau]$ of type $\underline{\mathbb{R}}$, we have that $C[t] \Downarrow \leq C[s] \Downarrow$ in the usual (pointwise) order of measures on $\mathsf{Val}^{\vdash \underline{\mathbb{R}}}$. We say that t and s are *contextually equivalent*, writing $t \approx s$, when $t \precsim s$ and $s \precsim t$

The observational preorder and equivalence are the same even if we vary the definitions:

(1) using contexts of type $\mathbf{1}$ and observing only the weight of convergence;
(2) using contexts of ground type (i.e. iterated sums and products of primitive types) and observing the distribution over $\mathsf{Val}^{\vdash\tau}$;
(3) using contexts of arbitrary type τ and observing the induced distribution over $\mathsf{Val}^{\vdash\tau}/\sim$, where $\sim$ is the smallest congruence identifying all λ-abstractions (as done in [Pitts 1996]).

Indeed, for the equivalence of 1., 2. and 3., use characteristic functions $\mathbb{R}^n \to \mathbb{R}$. For the equivalence with our notion, observe that $\mathbf{1}$ embeds into $\underline{\mathbb{R}}$ and, conversely, distinguish any distribution on $\mathsf{Val}^{\vdash\underline{\mathbb{R}}}$ with contexts of type $\mathbf{1}$ by using $\mathbf{match}\,[\underline{-} \in A](\underline{-})\,\mathbf{with}\,\{\underline{0} \to \bot \mid \_ \to ()\}$, for Borel $A \subseteq \mathbb{R}$.

6.5 Idealised Church and SPCF

Since both Idealised Church (§2.3) and SPCF (§2.4) are fragments of SFPC, they inherit an operational semantics and a notion of contextual equivalence w.r.t. contexts in the fragment. For Idealised Church, these contexts translate into certain SFPC program contexts of type $\Lambda\text{-}\,\mathsf{Term}\,\underline{\mathbb{R}}$. The induced notion of contextual preorder is now that for two Idealised Church terms we set $t \precsim s$ when $C[t]^\dagger \Downarrow \leq C[s]^\dagger \Downarrow$ as distributions on $\mathsf{Val}^{\vdash\Lambda\text{-}\,\mathsf{Term}\,\underline{\mathbb{R}}}/\sim$ for all Idealised Church program contexts $C[-]$, where $\sim$ identifies all values of the form $\mathsf{Fun}\,(\lambda x.t)$. For CBV SPCF program contexts of type $\underline{\mathbb{R}}$ induce an analogous notion of contextual preorder and equivalence.

Lemma 6.2. *The translations Idealised Church* $\xrightarrow{(-)^\dagger}$ *SFPC* $\xleftarrow{(-)^\ddagger}$ *CBV SPCF are adequate:*

$$t_1^\dagger \precsim_{\mathit{SFPC}} t_2^\dagger \implies t_1 \precsim_{\mathit{Idealised\ Church}} t_2 \qquad s_1^\ddagger \precsim_{\mathit{SFPC}} s_2^\ddagger \implies s_1 \precsim_{\mathit{CBV\ SPCF}} s_2$$

Proof. Both languages are fragments of SFPC with the induced operational semantics. Moreover, the source contexts are a subset of those of SFPC. Therefore, the SFPC contextual preorder is as fine-grained as each fragment's contextual preorder. □

6.6 Denotational Semantics

Recall the type semantics of §5.3 for SFPC: closed types τ denote ωqbses $[\![\tau]\!]$. We extend this assignment to contexts by: $[\![\Gamma]\!] := \prod_{(x:\tau)\in\Gamma} [\![\tau]\!]$. Fig. 13 defines semantics for values and terms:

$$[\![-]\!]^v : \mathsf{Val}^{\Gamma\vdash\tau} \to \omega\mathbf{Qbs}([\![\Gamma]\!], [\![\tau]\!]) \qquad [\![-]\!] : \mathsf{Trm}^{\Gamma\vdash\tau} \to \omega\mathbf{Qbs}([\![\Gamma]\!], T\,[\![\tau]\!])$$

such that $\mathrm{return}^T_{[\![\tau]\!]}\,([\![v]\!]^v\,\gamma) = [\![v]\!]\,\gamma$ for every $v \in \mathsf{Val}^{\Gamma\vdash\tau}$ and $\gamma \in [\![\Gamma]\!]$. This interpretation is the standard semantics of a call-by-value calculus using a monad over a bi-cartesian closed category, where we interpret the operations $[\![\mathbf{sample}]\!]\,\gamma := \mathrm{sample}()$ and $[\![\mathbf{score}\,t]\!]\,\gamma := [\![t]\!]\,\gamma \gg=^T \mathrm{score}$.

Induction on terms and values proves that the semantics has the following standard properties:

Lemma 6.3 (Substitution). *Let* $\vdash v_x : \sigma, (x:\sigma) \in \Gamma$ *be closed SFPC values and* $\Gamma \vdash t : \tau$ *be an SFPC term. Then* $\left[\!\left[t[x \mapsto v_x]_{(x:\sigma)\in\Gamma}\right]\!\right] = [\![t]\!]\left[x \mapsto [\![v_x]\!]^v\,()\right]_{(x:\sigma)\in\Gamma}$.

Theorem 6.4 (Compositionality). *Let* $C[\Gamma \vdash - : \tau]$ *be an SFPC program context and let* $\Gamma \vdash t, s : \tau$ *be SFPC terms. If* $[\![t]\!] \leq [\![s]\!]$ *then* $[\![C[t]]\!] \leq [\![C[s]]\!]$. *As a consequence, the meaning of a term depends only on the meaning of its sub-terms: if* $[\![t]\!] = [\![s]\!]$ *then* $[\![C[t]]\!] = [\![C[s]]\!]$.

6.7 Enriched Semantics

These semantic definitions can be phrased inside the category $\omega\mathbf{Qbs}$. Recall from §3.1 the adjunction $\Sigma_- \dashv M_- : \mathbf{Qbs} \to \mathbf{Meas}$. The coproduct representation of $\mathsf{Trm}^{\Gamma\vdash\tau}$ as $\sum_{n\in\mathbb{N},\, t\in\mathsf{TTrm}^{\Gamma;\square_1,\ldots,\square_n\vdash\tau}} \mathbb{R}^n$ is meaningful in $\omega\mathbf{Qbs}$ too. Because left adjoints preserve co-products, its underlying measurable space is the measurable space structure of $\mathsf{Trm}^{\Gamma\vdash\tau}$ from §6.2. Similarly, $\mathsf{Val}^{\Gamma\vdash\tau}$ is the underlying measurable space of a qbs. Both become ωqbses via the discrete order. The denotational interpretation functions are ωqbs-morphisms $[\![-]\!]^v : \mathsf{Val}^{\Gamma\vdash\tau} \to [\![\tau]\!]^{[\![\Gamma]\!]}$ and $[\![-]\!] : \mathsf{Trm}^{\Gamma\vdash\tau} \to (T\,[\![\tau]\!])^{[\![\Gamma]\!]}$.

$$\textbf{Values:}\quad [\![x]\!]^v\,\gamma := \pi_x\gamma \quad [\![\underline{r}]\!]^v\,\gamma := r \quad [\![()]\!]^v\,\gamma := \langle\rangle \quad [\![(v,w)]\!]^v\,\gamma := \langle [\![v]\!]^v\,\gamma, [\![w]\!]^v\,\gamma\rangle$$
$$[\![\tau.\ell_i\, v]\!]^v\,\gamma := \iota_i([\![v]\!]^v\,\gamma) \qquad [\![\tau.\mathbf{roll}\,(v)]\!]^v := \mathrm{roll}([\![v]\!]^v\,\gamma) \qquad [\![\lambda x:\sigma.t]\!]^v\,\gamma := \lambda a.\ [\![t]\!]\gamma[x\mapsto a]$$

$$\textbf{Terms:}\quad [\![v]\!]\,\gamma := \mathrm{return}([\![v]\!]^v\,\gamma) \quad \text{for } v = x, (), \lambda x:\tau.t$$
$$[\![\mathbf{sample}]\!]\,\gamma := \mathrm{sample}() \qquad [\![\mathbf{score}\,t]\!]\,\gamma := [\![t]\!]\,\gamma \gg\!=^T \mathrm{score}$$
$$[\![\underline{f}(t_1,\dots,t_n)]\!]\,(\gamma) := [\![t_1]\!]\,\gamma \gg\!= \lambda r_1.\ \dots\ [\![t_n]\!]\,\gamma \gg\!= \lambda r_n.\ \mathrm{return}(f(r_1,\dots,r_n))$$
$$\left[\!\!\left[\begin{array}{l}\mathbf{match}\ t\ \mathbf{with}\\ \{\underline{0}\to s\\ \mid\ \_\to r\}\end{array}\right]\!\!\right]\gamma := \begin{array}{l}[\![t]\!]\,\gamma \gg\!=\\ [\{0\}.\,[\![s]\!]\,\gamma,\\ \{0\}^{\complement}.\,[\![r]\!]\,\gamma]\end{array}$$

$$[\![(t,s)]\!]\,\gamma := [\![t]\!]\,\gamma \gg\!= \lambda a.\ [\![s]\!]\,\gamma \gg\!= \lambda b.\ \mathrm{return}(\langle a,b\rangle)$$
$$[\![\tau.\ell_i\, t]\!]\,\gamma := T\iota_i([\![t]\!]\,\gamma) \qquad [\![\tau.\mathbf{roll}\,t]\!]\,\gamma := T\mathrm{roll}([\![t]\!]\,\gamma)$$
$$[\![t\ s]\!]\,\gamma := [\![t]\!]\,\gamma \gg\!= \lambda f.\ [\![s]\!]\,\gamma \gg\!= \lambda a.\ f\ a$$
$$\left[\!\!\left[\begin{array}{l}\mathbf{match}\ t\ \mathbf{with}\\ () \to s\end{array}\right]\!\!\right]\gamma := [\![t]\!]\,\gamma \gg\!= \lambda\_.\ [\![s]\!]\,\gamma$$
$$\left[\!\!\left[\begin{array}{l}\mathbf{match}\ t\ \mathbf{with}\\ (x,y) \to s\end{array}\right]\!\!\right]\gamma := [\![t]\!]\,\gamma \gg\!= \lambda\langle a,b\rangle.\ [\![s]\!]\,\gamma[x\mapsto a, y\mapsto b]$$
$$\left[\!\!\left[\begin{array}{l}\mathbf{match}\ t\ \mathbf{with}\\ \{\ell_1\,x_1 \to s_1 \mid \cdots\\ \mid\ \ell_n\,x_n \to s_n\}\end{array}\right]\!\!\right]\gamma := [\lambda a_1.\ [\![s_1]\!]\,\gamma[x_1\mapsto a_1], \dots, \lambda a_n.\ [\![s_n]\!]\,\gamma[x_n\mapsto a_n]]$$
$$\left[\!\!\left[\begin{array}{l}\mathbf{match}\ t\ \mathbf{with}\\ \mathbf{roll}\,x \to s\end{array}\right]\!\!\right]\gamma := [\![t]\!]\,\gamma \gg\!= \lambda a.\ [\![s]\!]\,\Gamma[x\mapsto \mathrm{unroll}\ a]$$

Fig. 13. The denotational interpretation of values (top), and terms (bottom).

$$r \trianglelefteq_v^{\vdash\mathbb{R}} \underline{r} \Leftrightarrow \top \qquad \langle\rangle \trianglelefteq_v^{\vdash\mathbf{1}} () \Leftrightarrow \top \qquad \langle a_1,a_2\rangle \trianglelefteq_v^{\vdash\tau_1*\tau_2} (v_1,v_2) \Leftrightarrow a_1 \trianglelefteq_v^{\vdash\tau_1} v_1 \wedge a_2 \trianglelefteq_v^{\vdash\tau_2} v_2$$
$$\iota_i(a) \trianglelefteq_v^{\vdash\tau} \tau.\ell_j v \Leftrightarrow i=j \wedge a \trianglelefteq_v^{\vdash\tau_i} v \qquad \mathrm{roll}\ a \trianglelefteq_v^{\vdash\tau} \tau.\mathbf{roll}\,v \Leftrightarrow (\tau = \{\ell_1\,\tau_1 \mid \dots \mid \ell_n\,\tau_n\}) \qquad \Lambda(k) \trianglelefteq_v^{\vdash\tau\to\sigma} \lambda x:\tau.t \Leftrightarrow a \trianglelefteq_v^{\vdash\sigma[\alpha\mapsto\tau]} v \quad (\tau = \mu\alpha.\sigma) \qquad k \le [\![t]\!] \wedge k \trianglelefteq_c^{x:\tau\vdash\sigma} t$$
$$\mu \trianglelefteq_c^{\vdash\tau} t \Leftrightarrow \mu \le [\![t \Downarrow]\!]_T^v \qquad b \trianglelefteq_v^{\Gamma\vdash\tau} w \Leftrightarrow \forall \left\langle a_x \trianglelefteq_v^{\vdash\sigma} v_x\right\rangle_{(x:\sigma)\in\Gamma}. b\,(a_x)_{(x:\sigma)\in\Gamma} \trianglelefteq_v^{\vdash\tau} w[x\mapsto v_x]_{(x:\sigma)\in\Gamma}$$
$$k \trianglelefteq_c^{\Gamma\vdash\tau} t \Leftrightarrow \forall \left\langle a_x \trianglelefteq_v^{\vdash\sigma} v_x\right\rangle_{(x:\sigma)\in\Gamma}. k\,(a_x)_{(x:\sigma)\in\Gamma} \trianglelefteq_c^{\vdash\tau} t\ [x\mapsto v_x]_{(x:\sigma)\in\Gamma}$$

Fig. 14. The properties of the relational interpretation (logical relation) $\trianglelefteq_v$ and $\trianglelefteq_c$.

Lemma 6.5. *The denotational semantics* $[\![-]\!]^v : \mathsf{Val}^{\vdash\mathbb{R}} \to \mathbb{R}$ *is a an isomorphism. Therefore,* $[\![-]\!]_T^v := T\,[\![-]\!]^v : T\mathsf{Val}^{\vdash\mathbb{R}} \to T\mathbb{R}$ *is an isomorphism.*

The notational conventions defining the big-step operational semantics kernel internalise in $\omega\mathbf{Qbs}$, if we read return^T for δ and sample for $\mathbb{U}_{[0,1]}$. We can therefore define an abstract interpreter as an ω-qbs map $\Downarrow : \mathsf{Trm}^{\vdash\tau} \to T\mathsf{Val}^{\vdash\tau}$. As in §4.6, because we equipped $\mathsf{Val}^{\vdash\tau}$ with the discrete order, the Kleisli arrow $\Downarrow : \mathsf{Trm}^{\vdash\tau} \to T\mathsf{Val}^{\vdash\tau}$ induces at most one s-finite kernel $\Downarrow : \mathsf{Trm}^{\vdash\tau} \rightsquigarrow \mathsf{Val}^{\vdash\tau}$ by restriction to characteristic functions. This kernel is in fact the big-step semantics from §6.3.

6.8 Adequacy

The key to relating the denotational and operational semantics is to establish that $[\![t]\!] = [\![t \Downarrow]\!]_T^v$ as expectation operators in $T\,[\![\tau]\!]$. As usual, we prove one inequality by induction on the syntax.

Lemma 6.6 (Computational Soundness). *For every closed term* $\vdash t : \tau$*, we have* $[\![t]\!] \ge [\![t \Downarrow]\!]_T^v$.

For the other direction, we construct a logical relation, using a variation of Pitts' minimal invariant relations method in bilimit compact categories [Levy 2004; Pitts 1996].

Lemma 6.7 (Relational Interpretation). *For every context* Γ *and type* τ *of SFPC, there are relations* $\trianglelefteq_v^{\Gamma\vdash\tau} \subseteq \omega\mathbf{Qbs}([\![\Gamma]\!], [\![\tau]\!]) \times \mathsf{Val}^{\Gamma\vdash\tau}$ *and* $\trianglelefteq_c^{\Gamma\vdash\tau} \subseteq \omega\mathbf{Qbs}([\![\Gamma]\!], T\,[\![\tau]\!]) \times \mathsf{Trm}^{\Gamma\vdash\tau}$ *satisfying Fig. 14.*

We have that $a \trianglelefteq_v^{\vdash\tau} v$ implies that $a \le [\![v]\!]^v$. Therefore, using the Substitution Lemma 6.3 and the definitions of $\Downarrow$, $[\![-]\!]^v$ and $[\![-]\!]$, we establish the following fundamental lemma, by a lengthy mutual induction on the structure of w and s.

Lemma 6.8 (Fundamental). *For every* $w \in \mathsf{Val}^{\Gamma\vdash\tau}$ *and* $s \in \mathsf{Trm}^{\Gamma\vdash\tau}$ $[\![w]\!]^{\upsilon} \trianglelefteq^{\Gamma\vdash\tau}_{\upsilon} w$ *and* $[\![s]\!] \trianglelefteq^{\Gamma\vdash\tau}_{c} s$.

In particular, for closed terms $\vdash t : \tau$ we have $[\![t]\!] \leq [\![t \Downarrow]\!]^{\upsilon}_{T}$. Adequacy now follows:

Theorem 6.9 (Adequacy). *For all types* τ *of SFPC and all closed terms* $t, s \in \mathsf{Trm}^{\vdash\tau}$*, we have that* $[\![t]\!] \leq [\![s]\!]$ *implies that* $t \precsim s$*. In particular,* $[\![t]\!] = [\![s]\!]$ *implies that* $t \approx s$.

Proof. Assume $[\![t]\!] \leq [\![s]\!]$ and consider any any context $C[\Gamma \vdash - : \tau]$ of type $\underline{\mathbb{R}}$. By the Compositionality Thm. 6.4, we have: $[\![C[t]]\!] \leq [\![C[s]]\!]$. Therefore: $[\![C[t] \Downarrow]\!]^{\upsilon}_{T} = [\![C[t]]\!] \leq [\![C[s]]\!] = [\![C[s] \Downarrow]\!]^{\upsilon}_{T}$. By Lemma 6.5, we deduce $C[t]\Downarrow \leq C[s]\Downarrow$. So $t \precsim s$. □

By Lemma 6.2, it now follows that the induced denotational semantics of Idealised Church at the ωqbs $[\![\Lambda\text{-}\mathsf{Term}\,\underline{\mathbb{R}}]\!]$ and the induced denotational semantics of CBV PPCF are adequate.

Corollary 6.10. *For Idealised Church or CBV PPCF terms* t, s*,* $[\![t]\!] \leq [\![s]\!]$ *implies that* $t \precsim s$.

We conclude with two applications of the Adequacy Thm. 6.9. First, evaluation order in SFPC and its sub-fragments does not matter, as a consequence of the monad commutativity (Thm. 4.6):

Corollary 6.11. *For every* $\Gamma \vdash t : \tau$*,* $\Gamma \vdash s : \sigma$*, and* $\Gamma, x : \tau, y : \sigma \vdash r : \rho$ *we have:*

$$\begin{array}{l}\mathbf{let}\,(x:\tau) = t\,\mathbf{in}\\ \mathbf{let}\,(y:\sigma) = s\,\mathbf{in}\,r\end{array} \quad \approx \quad \begin{array}{l}\mathbf{let}\,(y:\sigma) = s\,\mathbf{in}\\ \mathbf{let}\,(x:\tau) = t\,\mathbf{in}\,r\end{array}$$

Second, we show that our definable term-level recursion operator is indeed a fixed-point operator:

Corollary 6.12. *Let* $\tau = \tau_1 \to \tau_2$ *be a function type. For every* $\Gamma, x : \tau \vdash t : \tau$ *we have* $\mu x : \tau.t \approx t[x \mapsto \mu x : \tau.t]$*. Therefore, the following derivation rule is admissible:*

$$\frac{t[x \mapsto \mu x : \tau.t] \Downarrow \upsilon}{\mu x : \tau.t \Downarrow \upsilon}$$

7 CHARACTERISING QUASI-BOREL PRE-DOMAINS

We describe three additional categories equivalent to $\omega\mathbf{Qbs}$. Each approaches the question of how to combine qbs and ωcpo structures in a different way. We summarise the characterisations:

Theorem 7.1. *We have equivalences of categories:* $\omega\mathbf{Qbs} \simeq \mathcal{E}_{\omega q} \simeq \omega\mathbf{Cpo}(\mathbf{Qbs}) \simeq \mathrm{Mod}(\boldsymbol{\omega}\mathbf{qbs}, \mathbf{Set})$.

We describe these equivalent categories. For our semantics, we only need these consequences:

Corollary 7.2. *The category of* ω*qbses,* $\omega\mathbf{Qbs}$*, is locally* $\mathfrak{c}^+$*-presentable, where* $\mathfrak{c}^+$ *is the successor cardinal of the continuum. In particular, it has all small limits and colimits.*

7.1 $\mathcal{E}_{\omega q}$: a Domain-Theoretic Completion of Standard Borel Spaces

Quasi-Borel spaces are a quasi-topos completion of the category **Sbs** of standard Borel spaces. Heunen et al. [2017] establish an equivalence $-^{\sharp} : \mathcal{E}_q \simeq \mathbf{Qbs}$, where $\mathcal{E}_q$ is the full sub-category of presheaves $[\mathbf{Sbs}^{\mathrm{op}}, \mathbf{Set}]$ consisting of those functors $F : \mathbf{Sbs}^{\mathrm{op}} \to \mathbf{Set}$ that are:

- countable coproduct preserving, a sheaf condition: for every countable I and coproduct cocone $\left(\sum_{i\in I} S_i, \langle \iota_i \rangle_{i\in I}\right)$, the pair $\left(F\sum_{i\in I} S_i, \langle F\iota_i \rangle_{i\in I}\right)$ forms a product cone of $\prod_{i\in I} FS_i$; and
- separated: the function $m_F := \left\langle F\underline{r} \right\rangle_{r\in\mathbb{R}} : F\mathbb{R} \to \prod_{r\in\mathbb{R}} F\mathbb{1}$ is injective.

This equivalence is given on objects by mapping F to the qbs $F^{\sharp}$ whose carrier is $F\mathbb{1}$ and whose random elements are the image $m_F[F\mathbb{R}]$ under the injection from the separatedness condition.

Therefore, $m_F : F\mathbb{R} \to \prod_{r\in\mathbb{R}} F\mathbb{1}$ restricts to a bijection $\xi_F : F\mathbb{R} \cong M_{F^\sharp}$. The equivalence is given on morphisms $\alpha : F \to G$ by $\alpha^\sharp := \alpha_{\mathbb{1}}$. Conversely, given any **Qbs**-morphism $f : F^\sharp \to G^\sharp$, by setting:

$$\alpha_{[n]} : F[n] \xrightarrow{\cong} \prod_{i\in[n]} F\mathbb{1} \xrightarrow{f^n} \prod_{i\in[n]} G\mathbb{1} \xrightarrow{\cong} G[n] \quad \alpha_{\mathbb{R}} : F\mathbb{R} \xrightarrow{\xi_F} M_{F^\sharp} \xrightarrow{f\circ(-)} M_{G^\sharp} \xrightarrow{\xi_G} G\mathbb{R}$$

we obtain the natural transformation $\alpha : F \to G$ for which $\alpha^\sharp = f$. Here, we write $[n]$ for a standard Borel space with n elements. As an adjoint equivalent to $-^\sharp$, we can choose, for every qbs X, the separated sheaf $X^\flat a := X^a$ where a is either an object or a morphism, and the (co)unit of this adjoint equivalence is given by $\eta_X : X^{\flat\sharp} = X^\flat\mathbb{1} = X^{\mathbb{1}} \xrightarrow{\cong} X$.

We define a similar category $\mathcal{E}_{\omega q}$ to be the full subcategory of $[\mathbf{Sbs}^{\mathrm{op}}, \omega\mathbf{Cpo}]$ consisting of those functors $F : \mathbf{Sbs}^{\mathrm{op}} \to \omega\mathbf{Cpo}$ that are:

- countable coproduct preserving, i.e., the same sheaf condition; and
- $\omega\mathbf{Cpo}$-separated: the function $m_F := \langle F\underline{r}\rangle_{r\in\mathbb{R}} : F\mathbb{R} \to \prod_{r\in\mathbb{R}} F\mathbb{1}$ is a full mono.

Post-composing with the forgetful functor $|-| : \omega\mathbf{Cpo} \to \mathbf{Set}$ yields a forgetful functor $|-| : \mathcal{E}_{\omega q} \to \mathcal{E}_q$, as the underlying function of a full mono is injective.

We equip $\mathcal{E}_{\omega q}$ with an $\omega\mathbf{Cpo}$-category structure by setting the order componentwise, defining $\alpha \le \beta$ when $\forall S \in \mathbf{Sbs}.\alpha_S \le \beta_S$. Recall that an $\omega\mathbf{Cpo}$-*equivalence* is an adjoint pair of locally-continuous functors whose unit and counit are isomorphisms. To specify an $\omega\mathbf{Cpo}$-equivalence, it suffices to give a fully-faithful locally-continuous essentially surjective functor in either way, together with a choice of sources and isomorphisms for the essential surjectivity.

Proposition 7.3. *The equivalence $-^\sharp : \mathcal{E}_{\omega q} \simeq \omega\mathbf{Qbs}$ of Thm. 7.1 is $\omega\mathbf{Cpo}$-enriched, and given by:*

$$F^\sharp := \left\langle |F\mathbb{1}|, m_F[F\mathbb{R}], \le_{F\mathbb{1}} \right\rangle \qquad (\alpha : F \to G)^\sharp := \alpha_{\mathbb{1}}$$

Moreover, the two forgetful functors $|-| : \mathcal{E}_{\omega q} \to \mathcal{E}_q$ and $|-| : \omega\mathbf{Qbs} \to \mathbf{Qbs}$ form a map of adjoints.

7.2 ωCpo(Qbs): ωcpos Internal to Qbs

The category **Qbs** is a Grothendieck quasi-topos, which means it has a canonical notion of a subspace: *strong monos* (see after Ex. 5.1). Sub-spaces let us define relations/predicates, and interpret a fragment of higher-order logic formulae as subspaces. In particular, we can interpret ωcpo's definition *internally* to **Qbs** and Scott-continuous morphisms between such ωcpos as a subspace of the **Qbs**-function space, to form the category $\omega\mathbf{Cpo}(\mathbf{Qbs})$. We interpret functional operations, like the sup-operation of a ωcpo and the homorphisms of ωcpos, as internal functions, rather than as internal functional relations, as the two notions differ in a non-topos quasi-topos such as **Qbs**.

7.3 Mod($\boldsymbol{\omega}$**qbs**, **Set**): Models of an Essentially Algebraic Theory

Both $\omega\mathbf{Cpo}$ and **Qbs** are locally presentable categories. Therefore, there are essentially algebraic theories $\boldsymbol{\omega}$**cpo** and **qbs** and equivalences $\mathrm{Mod}(\boldsymbol{\omega}\mathbf{cpo}, \mathbf{Set}) \simeq \omega\mathbf{Cpo}$ and $\mathrm{Mod}(\mathbf{qbs}, \mathbf{Set}) \simeq \mathbf{Qbs}$ of their categories of set-theoretic algebras. We combine these two presentations into a presentation $\boldsymbol{\omega}$**qbs** for ωqbs, given in Appx. A, by taking their union, identifying the element sorts and adding a sup operation for ω-chains of the random elements, and a single axiom stating this sup is computed pointwise. Local presentability, for example, implies the existence of all small limits and colimits.

8 RELATED WORK AND CONCLUDING REMARKS

There is an extensive body of literature on probabilistic power-domain constructions. We highlight the first denotational models of higher-order probabilistic languages with recursion [Saheb-Djahromi 1980] and the work of Jones and Plotkin [1989] who give an adequate model of FPC with *discrete* probabilistic choice based on a category of pre-domains and a probabilistic power-domain

construction using valuations. Crucially, in this approach, commutativity of the probabilistic power-domain may fail unless one restricts to certain subcategories of continuous domains which are known to not be Cartesian closed. Jung and Tix [1998] survey the challenges in the search for such a sub-category of continuous domains that admits function spaces and probabilistic power-domains. One ingredient of this problem is that the measurable space structure used is generated from the Scott topology of the domain at hand. Subsequently Barker [2016]; Goubault-Larrecq and Varacca [2011]; Mislove [2016]; and Bacci et al. [2018] proposed variations on this in which random variables play a crucial role, as they do in qbses. We extend their reach by establishing commutativity.

Ehrhard et al. [2017] gave an adequate semantics for a *call-by-name* PCF with continuous probabilistic choice. The semantics is based on a variation of *stable* domain theory, using cones and stable functions, equipped with a notion of random elements analogous to a qbs-structure which they call *measurability tests*. We extend their reach by interpreting soft constraints, recursive types, and commutativity. A large technical difference with our work arises from their choice to work with cones rather than ωcpos, and with stable functions, rather than mere continuous functions, and from our choice to use monadic semantics.

Borgström et al. [2016] conducted a thorough operational treatment of an untyped probabilistic calculus, à la Idealised Church. Our work complements this analysis with a denotational counterpart.

Topological domain theory (TDT) [Battenfeld et al. 2007] accommodates most of the features that we have considered in this paper. In particular, it provides a cartesian closed category with an algebraically compact expansion and a commutative probabilistic powerdomain. Indeed, Huang et al. [2018] have already proposed it as a basis for statistical probabilistic programming. We leave the formal connections between topological domains and ωqbses for future investigation, but we compare briefly, following §1.2. In traditional domain theory, the Borel structure is derived from the order; in TDT, both order and Borel structure are derived from the topology; and in ωqbses, both order and Borel structure are independent (see §3.3). Concretely, TDT disallows topological discontinuities such as our zero-testing conditionals; moreover it makes a direct connection to computability [Ackerman et al. 2011].

To conclude, we developed pre-domains (§3) for statistical probabilistic recursive programs via:

- a *convenient* category $\omega\mathbf{Qbs}$, being Cartesian closed and (co)complete (Cor. 3.10);
- a commutative probabilistic power-domain modelling synthetic measure theory (§4);
- an interpretation of recursive types, through axiomatic domain theory (§5);
- adequate models of recursive languages with continuous probabilistic choice and soft constraints, of recursively typed, untyped and simply types varieties (§6);
- canonicity through four independent characterisations of $\omega\mathbf{Qbs}$ (§7).

This semantics gives sound reasoning principles about recursive probabilistic programs.

ACKNOWLEDGMENTS

Research supported by Balliol College Oxford (Career Development Fellowship), EPSRC (grants EP/N007387/1, EP/M023974/1), and the Royal Society. We are grateful to the reviewers for their suggestions. It has been helpful to discuss this work with the Oxford PL and Foundations groups, at the Domains 2018 and HOPE 2018 workshops, and with Marcelo Fiore, Chris Heunen, Paul Levy, Carol Mak, Gordon Plotkin, Alex Simpson, Hongseok Yang, amongst others.

REFERENCES

Martin Abadi and Marcelo P Fiore. 1996. Syntactic considerations on recursive types. In *Proc. LICS 1996*. IEEE, 242–252.

Nathanael L Ackerman, Cameron E. Freer, and Daniel M. Roy. 2011. Noncomputable conditional distributions. In *Proc. LICS 2011*. 107–116.

J. Adamek and J. Rosicky. 1994. *Locally Presentable and Accessible Categories*. Cambridge University Press.

Giorgio Bacci, Robert Furber, Dexter Kozen, Radu Mardare, Prakash Panangaden, and Dana Scott. 2018. Boolean-valued semantics for the stochastic λ-calculus. In *Proc. LICS 2018*.

Tyler Barker. 2016. A monad for randomized algorithms, In Proc. MFPS 2016. *Electronic Notes in Theoretical Computer Science* 325, 47–62.

Ingo Battenfeld, Matthias Schröder, and Alex Simpson. 2007. A convenient category of domains. *Electronic Notes in Theoretical Computer Science* 172 (2007), 69–99.

Johannes Borgström, Ugo Dal Lago, Andrew D. Gordon, and Marcin Szymczak. 2016. A lambda-calculus foundation for universal probabilistic programming. In *Proc. ICFP 2016*. 33–46. Full version at arxiv:abs/1512.08990.

Thomas Ehrhard, Michele Pagani, and Christine Tasson. 2017. Measurable cones and stable, measurable functions: a model for probabilistic higher-order programming. *Proceedings of the ACM on Programming Languages* 2, POPL (2017), 59.

Marcelo P. Fiore. 1996. *Axiomatic Domain Theory in Categories of Partial Maps*. Cambridge University Press.

Marcelo P Fiore and Gordon D Plotkin. 1994. An axiomatisation of computationally adequate domain theoretic models of FPC. In *Proc. LICS 1994*. 92–102.

Noah Goodman, Vikash Mansinghka, Daniel M Roy, Keith Bonawitz, and Joshua B Tenenbaum. 2008. Church: a language for generative models. In *Proc. UAI 2008*.

Noah Goodman and Andreas Stuhlmüller. 2014. Design and Implementation of Probabilistic Programming Languages. http://dippl.org. Online book.

Jean Goubault-Larrecq and Daniele Varacca. 2011. Continuous random variables. In *Logic in Computer Science (LICS), 2011 26th Annual IEEE Symposium on*. IEEE, 97–106.

Chris Heunen, Ohad Kammar, Sam Staton, and Hongseok Yang. 2017. A convenient category for higher-order probability theory. In *Proc. LICS 2017*.

Daniel Huang, Greg Morrisett, and Bas Spitters. 2018. An application of computable distributions to the semantics of probabilistic programs. (2018). arxiv:1806.07966.

Claire Jones and Gordon D Plotkin. 1989. A probabilistic powerdomain of evaluations. In *Proc. LICS 1989*. 186–195.

Achim Jung and Regina Tix. 1998. The troublesome probabilistic powerdomain, In Proc. Comprox 1998. *Electronic Notes in Theoretical Computer Science* 13, 70–91.

Olav Kallenberg. 2006. *Foundations of modern probability*. Springer.

Olav Kallenberg. 2017. *Random Measures, Theory and Applications*. Probability Theory and Stochastic Modelling, Vol. 77. Springer.

Ohad Kammar and Gordon D. Plotkin. 2012. Algebraic foundations for effect-dependent optimisations. In *Proc. POPL 2012*. 349–360.

Alexander Kechris. 2012. *Classical descriptive set theory*. Vol. 156. Springer Science & Business Media.

Klaus Keimel and Gordon D Plotkin. 2009. Predicate transformers for extended probability and non-determinism. *Mathematical Structures in Computer Science* 19, 3 (2009), 501–539.

Oleg Kiselyov and Chung-Chieh Shan. 2009. Embedded probabilistic programming. In *Domain-Specific Languages*. Springer, 360–384.

Anders Kock. 2012. Commutative monads as a theory of distributions. *Theory and Applications of Categories* 26, 4 (2012), 97–131.

Jimmy D. Lawson. 1982. Valuations on continuous lattices. In *Continuous Lattices and Related Topics*, Rudolf-Eberhard Höffman (Ed.). Mathematik Arbeitspapiere, Vol. 27. Universitat Bremen, 204–225.

Paul Blain Levy. 2004. *Call-By-Push-Value: A Functional/Imperative Synthesis*. Kluwer.

Vikash K. Mansinghka, Daniel Selsam, and Yura N. Perov. 2014. Venture: a higher-order probabilistic programming platform with programmable inference. *arXiv:1404.0099* (2014). http://arxiv.org/abs/1404.0099

Dylan McDermott and Ohad Kammar. 2018. Factorisation systems for logical relations and monadic lifting in type-and-effect system semantics. *Proc. MFPS 2018* (2018). To appear.

Michael W Mislove. 2016. Domains and random variables. *arXiv preprint arXiv:1607.07698* (2016).

Eugenio Moggi. 1989. Computational lambda-calculus and monads. In *Proc. LICS 1989*. 14–23.

Praveen Narayanan, Jacques Carette, Wren Romano, Chung-chieh Shan, and Robert Zinkov. 2016. Probabilistic inference by program transformation in Hakaru (system description). In *Proc. FLOPS 2016*. Springer, 62–79.

Federico Olmedo, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. 2016. Reasoning about recursive probabilistic programs. In *Proc. LICS 2016*. 672–681.

Andrew M Pitts. 1996. Relational properties of domains. *Inform. Comput.* 127, 2 (1996), 66–90.

Gordon D. Plotkin. 1977. LCF Considered as a Programming Language. *Theor. Comput. Sci.* 5, 3 (1977), 223–255.

Nasser Saheb-Djahromi. 1980. CPO's of measures for nondeterminism. *Theoret. Comput. Sci.* 12, 1 (1980), 19–37.

Adam Ścibior, Ohad Kammar, Matthijs Vákár, Sam Staton, Hongseok Yang, Yufei Cai, Klaus Ostermann, Sean K. Moss, Chris Heunen, and Zoubin Ghahramani. 2017. Denotational validation of higher-order Bayesian inference. *Proc. ACM Program. Lang.* 2, POPL, Article 60 (Dec. 2017), 29 pages.

Dana S. Scott. 1993. A type-theoretical alternative to ISWIM, CUCH, OWHY. *Theoret. Comput. Sci.* 121, 1 (1993), 411–440.

Chung-chieh Shan and Norman Ramsey. 2017. Exact Bayesian inference by symbolic disintegration. In *Proc. POPL 2017*. 130–144.

Michael B. Smyth and Gordon D. Plotkin. 1982. The category-theoretic solution of recursive domain equations. *SIAM J. Comput.* 11, 4 (1982), 761–783.

Sam Staton. 2017. Commutative semantics for probabilistic programming. In *Proc. ESOP 2017*.

Sam Staton, Hongseok Yang, Frank Wood, Chris Heunen, and Ohad Kammar. 2016. Semantics for probabilistic programming: higher-order functions, continuous distributions, and soft constraints. In *Proc. LICS 2016*. 525–534.

Matthijs Vákár and Luke Ong. 2018. On S-Finite Measures and Kernels. *arXiv preprint arXiv:1810.01837* (2018).

David Wingate, Andreas Stuhlmüller, and Noah Goodman. 2011. Lightweight implementations of probabilistic programming languages via transformational compilation. In *Proc. AISTATS 2011*.

Frank Wood, Jan Willem van de Meent, and Vikash Mansinghka. 2014. A new approach to probabilistic programming inference. In *Proc. AISTATS 2014*. 1024–1032.

A AN ESSENTIALLY ALGEBRAIC THEORY FOR ωQbs

An ωcpo or a qbs are essentially algebraic, in a precise sense [Adamek and Rosicky 1994, Chapter 3.D]. We will use this algebraic nature to analyse the quasi-Borel pre-domains and see how the quasi-Borel structure interacts with the ωcpo structure.

A.1 Presentations

For the following, fix a regular cardinal κ. Given a set $\mathcal{S}$ with cardinality $|\mathcal{S}| < \kappa$, whose elements we call *sorts*, an *$\mathcal{S}$-sorted (κ-ary) signature* Σ is a pair $\Sigma = (O, \text{arity})$ consisting of a set of *operations* and $\text{arity} : O \to \mathcal{S}^{<\kappa} \times \mathcal{S}$ assigns to each operation $f \in O$ a sequence $\langle s_i \rangle_{i\in I}$, indexed by some set I of cardinality $|I| < \kappa$, assigning to each index $i \in I$ its *argument sort*, together with another *result sort* s. We write $(f : \prod_{i\in I} s_i \to s) \in \Sigma$ for $\text{arity}(f) = (\langle s_i \rangle_{i\in I}, s)$.

Given an $\mathcal{S}$-sorted signature Σ, and a $\mathcal{S}$-indexed sequence of sets $\mathbb{V} = \langle \mathbb{V}_s \rangle_{s\in\mathcal{S}}$ of *variables* we define the collections of $\mathcal{S}$-sorted terms $\text{Term}^\Sigma\, \mathbb{V} = \left\langle \text{Term}^\Sigma_s\, \mathbb{V} \right\rangle_{s\in\mathcal{S}}$ over $\mathbb{V}$ inductively as follows:

$$\frac{}{x \in \text{Term}_s\, \mathbb{V}}(x \in \mathbb{V}_s) \qquad \frac{\text{for all } i \in I,\ t_i \in \text{Term}_{s_i}\, \mathbb{V}}{f\, \langle t_i \rangle_{n\in A} \in \text{Term}_s\, \mathbb{V}}((f : \prod_{i\in I} s_i \to s) \in \Sigma)$$

Given a signature Σ and a sort s, an *equation of sort s* is a pair of terms $(t_1, t_2) \in \text{Term}_s\, \mathbb{V}$ over some set of variables. As each term must involve less than κ-many variables, due to κ's regularity, and so we may fix the indexed set of variables $\mathbb{V}$ to be any specified collection of sets of cardinality κ.

Definition A.1. An *essentially algebraic presentation* $\mathcal{P}$ is a tuple $\langle \mathcal{S}, \Sigma_\text{t}, \Sigma_\text{p}, \text{Def}, \text{Eq} \rangle$ containing:

- a set $\mathcal{S}$ of *sorts*;
- two $\mathcal{S}$-sorted signatures with disjoint sets of operations:
  - a signature Σ_t of *total* operations;
  - a signature Σ_p of *partial* operations;

  we denote their combined signature by $\Sigma := \left(O_{\Sigma_\text{t}} \cup O_{\Sigma_\text{p}}, [\text{arity}_{\Sigma_\text{t}}, \text{arity}_{\Sigma_\text{p}}]\right)$;
- for each $(f : \prod_{i\in I} s_i \to s) \in \Sigma_\text{p}$, a set $\text{Def}(f)$ of Σ_t-equations over the variables $\{x_i : s_i | i \in I\}$ which we call the *assumptions of* $f\, \langle x_i \rangle_{i\in I}$; and
- a set Eq of Σ-equations which we call the *axioms*.

The point of this definition is just to introduce the relevant vocabulary. We will only be considering the following presentation for posets, then ωcpos, then qbses, then ωqbses:

Example A.2 (poset presentation cf. [Adamek and Rosicky 1994, Examples 3.35(1),(4)]). The presentation of *posets*, **pos**, has two sorts:

- element, which will be the carrier of the poset; and
- inequation, which will describe the poset structure.

The total operations are:

- lower : inequation $\to$ element, assigning to each inequation its lower element;
- upper : inequation $\to$ element, assigning to each inequation its upper element;
- refl : element $\to$ inequation, used to impose reflexivity;

The partial operations are:

- irrel : inequation $\times$ inequation $\to$ inequation, used to impose proof-irrelevance on inequations, with $\text{Def}(\text{irrel}(e_1, e_2))$:

$$\text{lower}(e_1) = \text{lower}(e_2) \qquad \text{upper}(e_1) = \text{upper}(e_2)$$

- antisym : inequation×inequation → element, used to impose anti-symmetry on inequations, with Def(antisym(e, e^{op})):

$$\mathrm{lower}(e) = \mathrm{upper}(e^{\mathrm{op}}) \qquad \mathrm{upper}(e) = \mathrm{lower}(e^{\mathrm{op}})$$

- trans : inequation × inequation → inequation, used to impose transitivity on inequations, with Def(trans(e_1, e_2)):

$$\mathrm{upper}(e_1) = \mathrm{lower}(e_2)$$

The axioms are:

$$e_1 = \mathrm{irrel}(e_1, e_2) = e_2 \qquad \text{(proof irrelevance)}$$

$$\mathrm{lower}(\mathrm{refl}(x)) = x = \mathrm{upper}(\mathrm{refl}(x)) \qquad \text{(reflexivity)}$$

$$\mathrm{lower}(e_1) = \mathrm{antisym}(e_1, e_2) = \mathrm{lower}(e_2) \qquad \text{(anti-symmetry)}$$

$$\mathrm{lower}(\mathrm{trans}(e_1, e_2)) = \mathrm{lower}(e_1) \qquad \mathrm{upper}(\mathrm{trans}(e_1, e_2)) = \mathrm{upper}(e_2) \qquad \text{(transitivity)}$$

Example A.3 (ωcpo presentation). In addition to the operations and axioms for posets, the presentation **ωcpo** of ωcpos includes the following partial operations:

- $\bigvee : \prod_{n\in\mathbb{N}} \text{inequation} \to \text{element}$, used to express lubs of ω-chains, with Def($\bigvee_{n\in\mathbb{N}} e_n$):

$$\mathrm{upper}(e_n) = \mathrm{lower}(e_{n+1}), \text{for each } n \in \mathbb{N}$$

- for each $k \in \mathbb{N}$, $\mathrm{ub}_k : \prod_{n\in\mathbb{N}} \text{inequation} \to \text{inequation}$, collectively used to impose the lub being an upper-bound, with Def($\mathrm{ub}_k \langle e_n \rangle_{n\in\mathbb{N}}$):

$$\mathrm{upper}(e_n) = \mathrm{lower}(e_{n+1}), \text{for each } n \in \mathbb{N}$$

- least : element × $\prod_{n\in\mathbb{N}}$ inequation × $\prod_{n\in\mathbb{N}}$ inequation → inequation, used to express that the lub is the least bound, with Def(least($x, \langle e_n \rangle_{n\in\mathbb{N}}, \langle b_n \rangle_{n\in\mathbb{N}}$)):

$$\mathrm{upper}(e_n) = \mathrm{lower}(e_{n+1}) \qquad \mathrm{upper}(b_n) = x \qquad \mathrm{lower}(e_n) = \mathrm{lower}(b_n), \text{for each } n \in \mathbb{N}$$

The axioms are:

$$\mathrm{lower}(\mathrm{ub}_k \langle e_n \rangle_n) = \mathrm{lower}(e_k) \qquad \mathrm{upper}(\mathrm{ub}_k \langle e_n \rangle_n) = \bigvee \langle e_n \rangle_n \qquad \text{(upper bound)}$$

$$\mathrm{lower}(\mathrm{least}(x, \langle e_n \rangle_n, \langle b_n \rangle_n)) = \bigvee \langle e_n \rangle_n \qquad \mathrm{upper}(\mathrm{least}(x, \langle e_n \rangle_n, \langle b_n \rangle_n)) = x \qquad \text{(least upper bound)}$$

Example A.4 (qbs presentation). The presentation of *qbses*, **qbs**, has two sorts:

- element, which will be the carrier of the qbs; and
- rand, which will be the random elements.

The total operations are:

- ev_r : rand → element, for each $r \in \mathbb{R}$, evaluating a random element at $r \in \mathbb{R}$;
- const : element → rand assigning to each element x the constantly-x random element;
- $\mathrm{rearrange}_\varphi$: rand → rand, for each $\varphi : \mathbb{R} \to \mathbb{R}$ Borel-measurable, precomposing random elements with φ; and
- $\mathrm{match}_{\langle S_i \rangle_{i\in I}} : \prod_{i\in I} \text{rand} \to \text{rand}$, for each $I \subseteq \mathbb{N}$ and measurable partition $\mathbb{R} = \sum_{i\in I} S_i$, pasting together a countable collection of random elements into a countable-case split.

There is one partial operation:

- ext : rand×rand → rand, used for establishing that a random element is uniquely determined extensionally, with Def(ext(α, β)) given by

$$\left\{ \mathrm{ev}_r(\alpha) = \mathrm{ev}_r(\beta) : \text{element} \middle| r \in \mathbb{R} \right\}$$

The axioms are:

$$\alpha = \mathrm{ext}(\alpha, \beta) = \beta \qquad \text{(extensionality)}$$

$$\left\{\mathrm{ev}_r(\mathrm{const}\,(x)) = x \middle| r \in \mathbb{R}\right\} \qquad \text{(constantly)}$$

$$\left\{\mathrm{ev}_r(\mathrm{rearrange}_\varphi\,\alpha) = \mathrm{ev}_{\varphi(r)}\,\alpha \middle| \varphi : \mathbb{R} \to \mathbb{R}\ \text{Borel-measurable}, r \in \mathbb{R}\right\} \qquad \text{(rearrange)}$$

$$\left\{\mathrm{ev}_r\left(\mathrm{match}_{\langle S_i\rangle_{i\in I}}\,\langle\alpha_i\rangle_{i\in I}\right) = \mathrm{ev}_r(\alpha_i) \middle| I \subseteq \mathbb{N}, \mathbb{R} = \sum_{i\in I} S_i\ \text{Borel-measurable}, i \in I, r \in S_i\right\} \qquad \text{(match)}$$

We can now present the ωqbses:

Example A.5 (ωqbs presentation). The presentation $\boldsymbol{\omega}$**qbs** of ωqbses extends the presentations $\boldsymbol{\omega}$**cpo** and **qbs**, identifying the element sorts, with the following additional partial operation:

- $\bigsqcup : \prod_{n\in\mathbb{N}} \mathrm{rand} \times \prod_{n\in\mathbb{N}, r\in\mathbb{R}} \mathrm{inequation} \to \mathrm{rand}$, used for establishing that the random-elements are closed under lubs w.r.t. the pointwise order, with Def $\bigsqcup(\langle\alpha_n\rangle_{n\in\mathbb{N}}, \langle e_n^r\rangle_{n\in\mathbb{N}, r\in\mathbb{R}})$ given by:

$$\left\{\mathrm{lower}(e_n^r) = \mathrm{ev}_r(\alpha_n), \mathrm{upper}(e_n^r) = \mathrm{ev}_r(\alpha_{n+1}), \middle| n \in \mathbb{N}, r \in \mathbb{R}\right\}$$

The additional axioms are:

$$\left\{\mathrm{ev}_r\left(\bigsqcup\left(\langle\alpha_n\rangle_{n\in\mathbb{N}}, \langle e_n^r\rangle_{n\in\mathbb{N}, r\in\mathbb{R}}\right)\right) = \bigvee\langle e_n^r\rangle_{n\in\mathbb{N}} \middle| r \in \mathbb{R}\right\} \qquad \text{(pointwise lubs)}$$

A.2 Algebras

Every essentially algebraic presentation induces a category of set-theoretic models, and this category for the ωcpo presentation is equivalent to ω**Cpo**. Moreover, we can interpret such presentations in any category with sufficient structure, namely countable products and equalisers (i.e., countable limits). We briefly recount how to do this.

Let $\mathcal{C}$ be a category with λ-small limits, with λ regular. As usual, if Σ is any $\mathcal{S}$-sorted λ-ary signature, we define a (multi-sorted) Σ-*algebra* $A = \left(\langle[\![s]\!]\rangle_{s\in\mathcal{S}}, [\![-]\!]\right)$ to consist of an $\mathcal{S}$-indexed family of objects $\langle[\![s]\!]\rangle_{s\in\mathcal{S}}$, the *carrier* of the algebra, and an assignment, to each $f : \prod_{i\in I} s_i \to s$ in Σ, of a morphism:

$$[\![f]\!] : \prod_{i\in I} [\![s_i]\!] \to [\![s]\!]$$

Given such an algebra A, and an $\mathcal{S}$-indexed set $\mathbb{V}$ of variables with $|\mathbb{V}_s| < \lambda$ for each $s \in \mathcal{S}$, each term t in $\mathrm{Term}_s\,\mathbb{V}$ denotes a morphism:

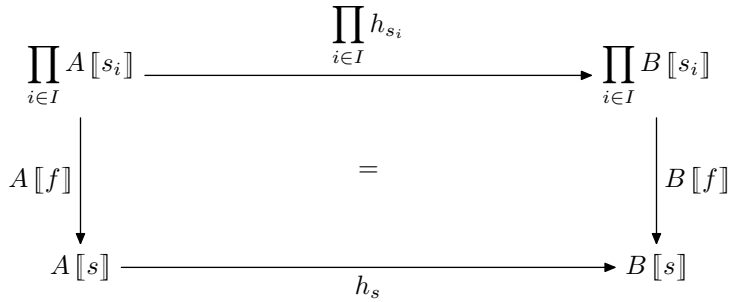
$$[\![t]\!]_s : \prod_{s\in\mathcal{S}} [\![s]\!]^{\mathbb{V}_s} \to [\![s]\!]$$

as follows:

$$[\![x]\!]_s : \prod_{s\in\mathcal{S}} [\![s]\!]^{\mathbb{V}_s} \xrightarrow{\pi_s} [\![s]\!]^{\mathbb{V}_s} \xrightarrow{\pi_x} [\![s]\!] \qquad [\![f\,\langle t_i\rangle_{i\in I}]\!] : \prod_{s\in\mathcal{S}} [\![s]\!]^{\mathbb{V}_s} \xrightarrow{\langle[\![t_i]\!]\rangle_{i\in I}} \prod_{i\in I} [\![s_i]\!] \xrightarrow{[\![f]\!]} [\![s]\!]$$

(When $|\mathbb{V}_s| \geq \lambda$, there are less than λ different variables that actually appear in t, and so we can find a smaller set of sorts and variables for which to define as above.)

A Σ-*homomorphism* $h : A \to B$ between Σ-algebras A, B is an $\mathcal{S}$-indexed family of functions $h_s : A\,[\![s]\!] \to B\,[\![s]\!]$ such that, for every operation symbol $f : \prod_{i\in I} s_i \to s$ in Σ:



We denote the category of Σ-algebras in C and their homomorphisms by $\mathrm{Mod}(\Sigma, C)$.