An Application of Computable Distributions to the Semantics of Probabilistic Programs: Part 2

Daniel Huang
Harvard SEAS
dehuang@fas.harvard.edu

Greg Morrisett
Cornell University
jgm19@cornell.edu

Abstract

We continue our previous work [5] on giving semantics to probabilistic programs using computable distributions. First, we highlight a gap in our previous semantics involving the measurability of an environment lookup function and address it. Second, we suggest *topological domains*, a category of domains proposed in the literature that supports both Type-2 computability and domain-theoretic constructions, as an alternative to DCPOs in giving semantics to probabilistic programs based on computable distributions.

1. Introduction

We continue our previous work [5] on giving semantics to probabilistic programs using (Type-2) computable distributions. In the previous semantics, we gave both sampling and distributional semantics to a high-level, PCF-like language with *continuous distributions* using DCPOs. If we think of a Turing-complete probabilistic programming language (PPL) as expressing computable distributions, then a natural attempt at interpreting a PCF-like language with distributions is to use DCPOs. The sampling semantics interprets a distribution as a (Type-2) computable function on bit-streams. The distributional semantics relies on the observation that a (Type-2) computable distribution can be identified with a valuation, which is an element of the appropriate DCPO. It was later pointed out that it was unclear why an environment lookup function (used in the interpretation of bind for a probability monad) was measurable in the DPCO semantics.[1]

Hence, in this paper, we first sketch out a fix that addresses the gap in the DCPO semantics. The intuition for the fix is that enviornment lookup is (Type-2) computable, and hence, continuous and (Borel) measurable. Notably, the fix highlights the explicit dependence of the DCPO semantics on (Type-2) computability. Second, we suggest *topological domains* [1], a category of domains proposed in the literature that supports Type-2 computability and domain-theoretic constructions, as an alternative to DCPOs for giving semantics to PPLs. In particular, it more cleanly addresses the issue of measurability associated with environment lookup.

2. DCPO Semantics

In this section, we will highlight the gap in our previous DCPO semantics and address it. We recall some background first.

2.1 Background

Computable distributions Intuitively, a distribution is (Type-2) computable if we can generate a sample from the distribution to arbitrary precision. For example, consider the standard Uniform distribution on the interval $(0, 1)$. An *algorithm* with access to a random bit-stream can generate a sample from this distribution to arbitrary precision with the following recursive procedure, starting with the interval $(0, 1)$: output the current interval, and bisect and continue with the left half or right half depending on the value of the next bit in the random bit-stream. The generated sample is the intersection of all output intervals. The notion of *algorithm* described here can be grounded on Type-2 Turing machines [8]. Type-2 computable distributions have been studied in the context of computable metric spaces [4] and represented spaces [6].

Valuations Let $(X, \mathcal{O}(X))$ be a topological space, $\mathcal{O}^{\subseteq}(X)$ be the opens of X ordered by subset inclusion, and $[0, 1]^{\uparrow}$ be the interval $[0, 1]$ with the usual $\leq$ ordering. A *valuation* $\nu : \mathcal{O}^{\subseteq}(X) \Rightarrow_{\mathbf{DCPO}} [0, 1]^{\uparrow}$ is a (DCPO-continuous) function that assigns to each open set of a topological space a probability, such that it is strict ($\nu(\emptyset) = 0$), monotone ($\nu(U) \leq \nu(V)$ for $U \subseteq V$), and modular ($\nu(U) + \nu(V) = \nu(U \cup V) + \nu(U \cap V)$ for every open U and V). Valuations possess many of the same properties as measures. Hence, they can be seen as a topological variation of a distribution. However, valuations are not required to satisfy countable additivity, unlike measures. Recall that a valuation ν is called *ω-continuous* if $\nu(\bigcup_{n\in\mathbb{N}} V_n) = \sup_{n\in\mathbb{N}} \nu(V_n)$ for $(V_n)_{n\in\mathbb{N}}$ an increasing sequence of opens. Hence, an ω-continuous valuation satisfies countable additivity. Below, we summarize some of the relationships between (Borel) measures, valuations, and (Type-2) computable distributions.

Proposition 2.1. *Let $(X, \mathcal{O}(X))$ be a topological space.*

1. *Every (Borel) measure μ on X can be restricted to an ω-continuous valuation $\mu|_{\mathcal{O}(X)} : [\mathcal{O}^{\subseteq}(X) \Rightarrow_{\mathbf{DCPO}} [0, 1]^{\uparrow}]$ (see Schröder [7, Sec. 3.1]). Moreover, μ is uniquely determined by its restriction to the opens $\mu|_{\mathcal{O}(X)}$.*
2. *When X is countably based, $[\mathcal{O}^{\subseteq}(X) \Rightarrow_{\mathbf{DCPO}} [0, 1]^{\uparrow}] \cong \mathscr{C}(\mathcal{O}(X), [0, 1]_<)$ (see Schroder [7, Sec. 3.1, Thm 3.5, Cor. 3.5]), where $\mathscr{C}(\mathcal{O}(X), [0, 1]_<)$ is the continuous function space between represented spaces.*

2.2 Fix

The previous semantics interprets an expression from a PCF-like language with a probability monad `Dist` τ as an element of a DCPO. The probability monad is restricted to types whose interpretation have computable metric space structure. We write $\mathcal{V}[\![\cdot]\!] : \mathbf{DCPO}$ for the type denotation and $\mathcal{E}[\![\Gamma \vdash e : \tau]\!] : \mathcal{V}[\![\Gamma]\!] \Rightarrow_{\mathbf{DCPO}} \mathcal{V}[\![\tau]\!]$ for the expression denotation. The semantics of bind for the distributional semantics is given below.

$$\mathcal{E}[\![\Gamma \vdash x \leftarrow e_1 \ ; \ e_2 : \texttt{Dist}\ \tau_2]\!]\rho = U \mapsto \int f_U \, d\mu$$

[1] We thank Mitch Wand for pointing this gap out.

where

$$\mu = \mathcal{E}[\![\Gamma \vdash e_1 : \texttt{Dist}\ \tau_1]\!]\rho$$
$$f_U = \bar{x} \mapsto \mathcal{E}[\![\Gamma, x : \tau_1 \vdash e_2 : \texttt{Dist}\ \tau_2]\!]\rho[x \mapsto \bar{x}](U)\,.$$

The Gap For the integral to be well-defined, the function f_U needs to be measurable. Typically, we can conclude this from the categorical structure a la curry. Taking this approach on the previous DCPO semantics results in a DCPO-continuous function, which is not Borel measurable with respect to the appropriate topologies. As a reminder, the canonical topology associated with a DCPO in domain theory is the Scott topology. However, because the original semantics interprets reals as a lifted, discrete domain, the Scott topology does not result in the usual Euclidean topology. Thus, the measurability of f_U does not follow from the usual argument and so it was unclear why it was measurable.

The Fix The intuition for the fix is that environment lookup is (Type-2) computable, and hence, continuous and (Borel) measurable. In particular, f_U is an environment lookup function, which can be realized by a (Type-2) Turing machine that looks at the environment tape named by x. Crucially, this fix relies on Type-2 computability, something which was not readily apparent in the original DCPO semantics. We sketch out a more complete argument now. First, we can give an operational semantics and show that the operational semantics is realizable by a (Type-2) Turing machine. Notably, this step cannot be accomplished without taking the computability of reals and distributions into account. The constructs of the language from the PCF-like subset can be given their usual encodings on Turing machines. Second, we can show soundness and adequacy of the denotational semantics with respect to the operational semantics. Third and finally, we can explicitly realize environment lookup as a Turing machine tape lookup, which shows that environment lookup is indeed (Type-2) computable.

3. Using Topological Domains

Although the gap in the DCPO semantics can be addressed with an auxiliary environment lookup argument, the solution is somewhat ad-hoc. Hence, we were motivated to find an alternative category of domains that captured both Type-2 computability and the concerns of domain theory (*e.g.*, fix-points). *Topological domains* [1] form one such category. In the typical domain-theoretic setting, the partial order is taken as primary and the topology is derived (*e.g.*, Scott topology). In contrast, topological domains take the topology as primary and derive the order (*e.g.*, specialization preorder). Hence, the main issue is ensuring that directed chains have least upper bounds so that (least) fix-points exist.

Type-2 Computability Topological domain theory begins with the Cartesian closed category of qcb_0 spaces [3]. Roughly speaking, a qcb_0 space is a topological space whose elements can be encoded (*i.e.*, named or represented) by bit-streams, and hence, has an associated theory of computability. More formally, a topological space X is a *qcb_0* space if it is the (T0) quotient of a countably based space.[2]

Domain-theoretic The next few definitions deal with ensuring that least fixed-points exists. A qcb_0 space is a *topological predomain* if every ascending chain $(x_n)_{n\in\mathbb{N}}$ (with respect to the specialization preorder) has an upper bound x such that $(x_i)_{i\in\mathbb{N}} \to x$ (with respect to its topology) [1, Defn. 5.1]. A topological space is a *monotone convergence space* if its specialization preorder results in a DCPO and every open is Scott-open. Hence, the Scott topology is in general finer than the topology associated with a topological predomain. Importantly, a qcb_0 space is a topological predomain iff it is a monotone convergence space [1, Prop. 5.4].

The category **TP** of topological predomains and continuous functions is Cartesian closed [1, Thm. 5.5]. As usual, a *topological domain* is a topological predomain with least element. Hence, one can take least fix-points of continuous (endo)functions between topological domains [1, Thm. 5.7]. Again, as usual, one can consider the category $\mathbf{TD}_{\perp!}$ of topological domains and strict continuous functions to obtain better structure for interpreting functional programs [1, Prop. 6.4].

Examples The reals $\mathbb{R}$ with Euclidean topology is a topological predomain—the topology is Hausdorff, and hence, trivially results in a monotone convergence space. The lift of the reals $\mathbb{R}_\perp$ is a topological domain—it adds a single open set $\mathbb{R} \cup \{\perp\}$ to the Euclidean topology. In general, topological domains differ from DCPOs on function spaces. For example, the Scott topology of the DCPO function space $\mathbb{N} \Rightarrow_{\mathbf{DCPO}} 2$ is discrete, where $\mathbb{N}$ and 2 are given the discrete order. In contrast, the topological predomain function space $\mathbb{N} \Rightarrow_{\mathbf{TP}} 2$ coincides with Cantor space, and hence, is a natural setting for (Type-2) computability. As a last example, we identify (Type-2) computable distributions with an appropriate topological domain.

$$\mathscr{C}(\mathcal{O}(X), [0,1]_<) \cong [\mathcal{O}^{\subseteq}(X) \Rightarrow_{\mathbf{TD}} [0,1]^{\uparrow}]$$

When combined with proposition 2.1, we see that the space of valuations as a topological domain coincides with the DCPO interpretation for countably based topological spaces.

Alternative Semantics We can use topological domains to give an alternative semantics to probabilistic programs based on computable distributions. That is, the type denotation $\mathcal{V}'[\![\cdot]\!] : \mathbf{TD}$ denotes into **TD** instead of **DCPO**. The expression denotation $\mathcal{E}'[\![\Gamma \vdash e : \tau]\!] : \mathcal{V}'[\![\Gamma]\!] \Rightarrow_{\mathbf{TD}} \mathcal{V}'[\![\tau]\!]$ is structurally the same as the DCPO denotation $\mathcal{E}[\![\cdot]\!]$. For instance, if the original denotation $\mathcal{E}[\![\cdot]\!]$ uses curry in **DCPO**, then the new denotation $\mathcal{E}'[\![\cdot]\!]$ uses curry in **TD**. Note that the interpretation of distributions from the DCPO semantics coincides with an alternative semantics that denotes into topological domains.[3] However, the interpretations will differ on function spaces differs. Indeed, this difference will enable us to conclude the measurability of f_U in the interpretation of bind from the categorical structure of **TD**.

4. Discussion

Most other work on semantics of probabilistic programs is based on measure theory (*e.g.*, [2]). One of our motivations for using computable distributions, as opposed to more generally measures, is so that we can give both distributional semantics and (algorithmic) sampling semantics to probabilistic programs. As we previously showed, it is possible to faithfully implement the sampling semantics in an ordinary, Turing-complete language without assuming blackbox support for reals or primitive continuous distributions. Indeed, we can think of a Turing-complete probabilistic programming language as expressing computable distributions. It would be interesting to compare and contrast the strengths and weaknesses of approaches based on measure theory versus computable distributions.

Acknowledgments

This work was supported by Oracle labs.

[2] The encoding (partial) function is called a representation. A qcb_0 space with admissible representation is an admissible represented space.

[3] Recall that computable metric spaces are countably based.

References

[1] I. Battenfeld, M. Schröder, and A. Simpson. A Convenient Category of Domains. *Electronic Notes in Theoretical Computer Science*, 172: 69–99, 2007.

[2] J. Borgström, A. D. Gordon, M. Greenberg, J. Margetson, and J. Van Gael. Measure Transformer Semantics for Bayesian Machine Learning. In *Programming Languages and Systems*, pages 77–96. Springer, 2011.

[3] M. Escardó, J. Lawson, and A. Simpson. Comparing cartesian closed categories of (core) compactly generated spaces. *Topology and its Applications*, 143(1):105–145, 2004.

[4] M. Hoyrup and C. Rojas. Computability of Probability Measures and Martin-Löf Randomness over Metric Spaces. *Information and Computation*, 207(7):830–847, 2009.

[5] D. Huang and G. Morrisett. An Application of Computable Distributions to the Semantics of Probabilistics Programming Languages. In *Programming Languages and Systems*, pages 337–363. Springer, 2016.

[6] A. Pauly. On the Topological Aspects of the Theory of Represented Spaces. *Computability*, (Preprint):1–22, 2012.

[7] M. Schröder. Admissible Representations of Probability Measures. *Electronic Notes in Theoretical Computer Science*, 167:61–78, 2007.

[8] K. Weihrauch. *Computable Analysis: An Introduction*. Springer Science & Business Media, 2000.