A convenient category of tracing measure kernels

ELI SENNESH, Northeastern University, USA
JAN-WILLEM VAN DE MEENT, University of Amsterdam, the Netherlands

Many probabilistic programming languages (PPLs) allow users to define custom proposals as programs. This requires conditionally evaluating a sampled proposal trace in the target program. Unfortunately, formulations of probability theory based on Markov categories as yet have no model of traces. This abstract introduces tracing and traced importance sampling in the higher-order Markov category **QBS**, building a Kleisli subcategory whose morphisms model traced sampling with unnormalized densities.

1 INTRODUCTION

Probabilistic programming languages (PPLs) provide language-level support for probabilistic modeling and inference [18]. Unfortunately, since no inference algorithm works well for all models, PPLs increasingly also provide metaprogramming constructs for inference [11]; these can include writing one probabilistic program as a generative model and another (often with parameters tuned by optimization) as an inference proposal [1, 2, 8, 12, 15]. Operationally, probabilistic programmers know what it means to use one program as a proposal for another, but denotational semantics for this operation has typically considered explicitly typed traces [10].

This abstract builds on the category **QBS** for higher-order probability theory [6], borrowing a useful sample space from Dash et al. [3]. Section 2 builds a model of *tracing* in **QBS**. Section 3 exploits tracing to convert expectations taken with respect to a proposal probability kernel into expectations with respect to a target kernel. Section 4 summarizes and discusses future work. Appendix A reviews preliminaries on **QBS**, and its resulting category of probability kernels $\mathcal{Q}$ provides our general setting. Appendix B develops the abstract's model of tracing measure kernels. Appendix C develops a change-of-measure theorem for tracing measure kernels.

2 EAGER AND LAZY TRACING IN PROBABILISTIC PROGRAMMING

A program in a PPL commonly writes out, as a side-effect, a *trace* containing the names and values of random variables sampled during execution. This requires a countably infinite set $\mathbf{A}$ of *addresses*, such as strings in Python-based PPLs [1, 12] or symbols in Lisp-based PPLs [5, 17, 19]. Internally, probabilistic program evaluation also reads from a stream of random numbers, converting them into meaningful samples from a distribution. This section formalizes randomness sinks and sources as mappings from hierarchical addresses $\mathbf{A}^*$ to random variates, within the ambient category $\mathcal{Q}$ of probability kernels. We refer to the typical traces found in PPLs as "eager" traces.

Eager traces map addresses to a pair of a random element of the unit interval, and an element of a *support* $\mathbf{s} \in \mathbf{S}$ with corresponding QBS $[\![\mathbf{s}]\!] \in Ob(\mathbf{QBS})$. Work such as Lew et al. [10] has specified such support types, and we give one such definition below.

Definition 2.1 (Support types). Support types $\mathbf{s} \in \mathbf{S}$ are drawn from the following grammar

$$\mathbf{S} := I \mid \mathbb{N} \mid \mathbb{Z} \mid \mathbb{R} \mid \mathbb{R}_0^+ \mid [0,1] \mid [1..n] : n \in \mathbb{N}.$$

Definition 2.2 (Eager traces). An *eager trace* is a deterministic partial function $\tau : \mathbf{A}^* \rightharpoonup_n [0,1] \times \coprod_{\mathbf{s}\in\mathbf{S}} [\![\mathbf{s}]\!]$ mapping $n \in \mathbb{N}$ addresses into a product of unit interval with the pair of a support type and an element of its corresponding space. Eager traces have type $\mathbb{T} \in Ob(\mathcal{Q})$.

Authors' addresses: Eli Sennesh, sennesh.e@northeastern.edu, Northeastern University, Huntington Ave, Boston, Massachusetts, USA, 02115; Jan-Willem van de Meent, j.w.vandemeent@uva.nl, University of Amsterdam, Amsterdam, the Netherlands.

Eager traces are deterministic partial mappings, because once a random variable has been sampled, its value cannot change. In contrast, lazy traces below provide a formalization of the randomness source from which a probabilistic program reads (as a side-effect); these must be stochastic in order to provide new random numbers for new random variables.

Definition 2.3 (Lazy traces). A *lazy trace* is a random function $p : \mathbf{A}^* \rightsquigarrow [0,1]$ mapping each address-list to a standard uniform random variable. Lazy traces have type $\mathbb{P}(\Omega) \in Ob(\mathcal{Q})$.

Lazy traces are lazily evaluated (stochastically memoized [5]) random functions. Evaluating a lazy trace at a set of specific hierarchical addresses yields an eager trace.

The next definition gives a semantic model in the category $\mathcal{Q}$ for a probabilistic program that internally draws from a lazy trace, and then combines it with the argument to write out a likelihood weight and an eager trace as side effects alongside the random output.

Definition 2.4 (Categories of eagerly traced measure kernels). The category $\mathbf{MeasKer}(\mathbf{Traced}(\mathcal{Q}))$ of eagerly traced measure kernels has the same objects as $\mathcal{Q}$. For domain and codomain $Z, X \in Ob(\mathcal{Q})$ the category $\mathbf{MeasKer}(\mathbf{Traced}(\mathcal{Q}))$ has morphisms $(f, \omega) : (\Omega \times Z) \to (\mathbb{R}_0^+ \times \mathbb{T} \times X)$. See Corollary B.12 in Appendix B for a diagram of these morphisms' structure.

Since programs take lazy traces as input but log eager traces as output, reevaluating a program or using one as a proposal for another requires embedding an eager trace into a lazy one. The following proposition demonstrates how to do so.

Proposition 2.5 (Eager traces embed into lazy traces). *Given an eager trace $\tau : \mathbb{T}$ and a lazy trace $p : \mathbb{P}(\Omega)$, the operation $p[\tau]$ embeds the former into the latter*

$$p[\tau] = \alpha \mapsto \begin{cases} \tau(\alpha)_1 & \alpha \in \mathrm{dom}(\tau) \\ p(\alpha) & \text{otherwise} \end{cases} \qquad \cdot[\cdot] : \mathbb{P}(\Omega) \times \mathbb{T} \to \mathbb{P}(\Omega).$$

3 IMPORTANCE WEIGHTING WITH TRACES

Inference based on importance sampling does not only entail sampling a trace from the proposal and reevaluating it under the target program, but also evaluating the ratio of the trace's densities under the target and proposal. Modeling this semantically requires a notion of densities for measure kernels. Theorem B.17, Corollary B.18, and Corollary B.19 show how to model densities in **QBS**, subject to certain requirements met by most programs in common PPLs. We then write that eagerly traced measure kernels with densities $\mathbf{MeasKer}(\mathbf{Traced}(\mathcal{Q}))$ provide unnormalized densities $\gamma_{(f,\omega)}(x, \tau; z) := \omega(z, \tau, x) p_f(x, \tau \mid z)$ for inference with traces $\tau : \mathbb{T}$.

The following theorem models nested importance sampling [13] compositionally in **QBS**.

Theorem 3.1 (Changes of measure give coslice categories of measure kernels). *Consider the image $\mathcal{Q}_{W \circ T} \subseteq \mathcal{Q}$ of the category $\mathbf{MeasKer}(\mathbf{Traced}(\mathcal{Q}))$ in $\mathcal{Q}$. The coslice category $I/\mathcal{Q}_{W \circ T}$ over this image describes how to sample from one eagerly-traced measure via another. It has eagerly traced measures (e.g. with no parameters left to provide) as objects, and a generalization of Equation 3 from Stites et al. [16] for importance weighting in dynamic trace-spaces as morphisms.*

Proof. See Theorem C.1 in Appendix C. □

4 DISCUSSION AND FUTURE WORK

This abstract has provided a semantic model for the reading and writing of randomness traces in probabilistic programming. Section 2 modeled for writing out random choices as an eager trace (or simply a trace) and for reading in random choices as a lazy trace. Section 3 then showed how

to embed an eager trace into a lazy one to reuse its randomness, and gave novel compositional semantics for sampling from a proposal program and weighing under a target program.

Future work can use this abstract's semantics to interpret inference metaprogramming and reason about how to compose inference programs. The authors particularly plan to pursue semantics for relaxing from deterministic weightings to random but *proper* weights [13]. Such a class of inference procedures would provide denotational semantics for most inference programs, including Sequential Monte Carlo, Markov chain Monte Carlo, and variational inference [9, 16].

ACKNOWLEDGMENTS

This work was supported by NSF awards 1835309 and 2047253. We would like to thank Adam Ścibior for his introduction to categorical semantics of probabilistic programs, and Mitch Wand for his feedback on an early work-in-progress form of this material. The authors would like to especially thank Alex Lew for his consultations on trace typing in higher-order probabilistic programming systems and the use of dependent sums in **QBS** to encode type heterogeneity in eager traces.

REFERENCES

[1] Eli Bingham, Jonathan P Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D Goodman. 2019. Pyro: Deep universal probabilistic programming. *The Journal of Machine Learning Research* 20, 1 (2019), 973–978.

[2] Marco F Cusumano-Towner, Feras A Saad, Alexander K Lew, and Vikash K Mansinghka. 2019. Gen: a general-purpose probabilistic programming system with programmable inference. In *Proceedings of the 40th acm sigplan conference on programming language design and implementation*. 221–236.

[3] Swaraj Dash, Younesse Kaddar, Hugo Paquet, and Sam Staton. 2022. Affine monads and lazy structures for Bayesian programming. (2022).

[4] Tobias Fritz. 2020. A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics. *Advances in Mathematics* 370 (2020), 107239. https://doi.org/10.1016/j.aim.2020.107239 arXiv: 1908.07021 Citation Key: Fritz2020.

[5] Noah D Goodman, Vikash K Mansinghka, Daniel M Roy, Keith Bonawitz, and Joshua B Tenenbaum. 2008. Church: a language for generative models. In *Uncertainty in Artificial Intelligence (UAI)*. https://doi.org/10.1016/j.foar.2013.08.005 arXiv: 1206.3255 Citation Key: Goodman2008 ISSN: 20952635.

[6] Chris Heunen, Ohad Kammar, Sam Staton, and Hongseok Yang. 2017. A convenient category for higher-order probability theory. In *Proceedings - Symposium on Logic in Computer Science*. https://doi.org/10.1109/LICS.2017.8005137 arXiv: 1701.02547 Citation Key: Heunen2017 ISSN: 10436871.

[7] Olav Kallenberg. 2021. *Foundations of Modern Probability*. Probability Theory and Stochastic Modelling, Vol. 99. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-030-61871-1

[8] Tuan Anh Le, Atilim Gunes Baydin, and Frank Wood. 2017. Inference compilation and universal probabilistic programming. In *Artificial Intelligence and Statistics*. PMLR, 1338–1348.

[9] Alexander K Lew, Marco Cusumano-Towner, and Vikash K Mansinghka. 2022. Recursive Monte Carlo and Variational Inference with Auxiliary Variables. In *Proceedings of the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022)*. 11.

[10] Alexander K Lew, Marco F Cusumano-Towner, Benjamin Sherman, Michael Carbin, and Vikash K Mansinghka. 2019. Trace types and denotational semantics for sound programmable inference in probabilistic languages. *Proceedings of the ACM on Programming Languages* 4, POPL (2019), 1–32.

[11] Vikash K. Mansinghka, Ulrich Schaechtle, Shivam Handa, Alexey Radul, Yutian Chen, and Martin Rinard. 2018. Probabilistic programming with programmable inference. In *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*. ACM, Philadelphia PA USA, 603–616. https://doi.org/10.1145/3192366.3192409

[12] Siddharth N, Brooks Paige, Jan-Willem van de Meent, Alban Desmaison, Noah Goodman, Pushmeet Kohli, Frank Wood, and Philip Torr. 2017. Learning Disentangled Representations with Semi-Supervised Deep Generative Models. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/9cb9ed4f35cf7c2f295cc2bc6f732a84-Paper.pdf

[13] Christian Naesseth, Fredrik Lindsten, and Thomas Schon. 2015. Nested Sequential Monte Carlo Methods. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 37)*, Francis

Bach and David Blei (Eds.). PMLR, Lille, France, 1292–1301. https://proceedings.mlr.press/v37/naesseth15.html

[14] Paolo Perrone. 2019. Notes on Category Theory with examples from basic mathematics. *arXiv preprint arXiv:1912.10642* (2019).

[15] Daniel Ritchie, Paul Horsfall, and Noah D Goodman. 2016. Deep amortized inference for probabilistic programs. *arXiv preprint arXiv:1610.05735* (2016).

[16] Sam Stites, Heiko Zimmermann, Hao Wu, Eli Sennesh, and Jan-Willem van de Meent. 2021. Learning proposals for probabilistic programs with inference combinators. In *Uncertainty in Artificial Intelligence*. PMLR, 1056–1066.

[17] David Tolpin, Jan Willem Van De Meent, Hongseok Yang, and Frank Wood. 2016. Design and implementation of probabilistic programming language Anglican. In *Proceedings of the 28th Symposium on the Implementation and Application of Functional Programming Languages*. https://doi.org/10.1145/3064899.3064910 arXiv: 1608.05263 Citation Key: Tolpin2016 ISBN: 9781450347679.

[18] Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood. 2018. An introduction to probabilistic programming. *arXiv preprint arXiv:1809.10756* (2018).

[19] D Wingate and Andreas Stuhlmüller. 2011. Lightweight implementations of probabilistic programming languages via transformational compilation. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, Vol. 15. 770–778. Citation Key: Wingate2011.

A A MATHEMATICAL SETTING FOR PROBABILISTIC PROGRAM SEMANTICS

A.1 Quasi-Borel spaces

Existing work typically assigns denotational semantics to higher-order probabilistic programs in terms of the category **QBS** of quasi-Borel spaces and stochastic maps between them [6]. Probability measures in this category are defined as push-forwards of probability measures over a standard Borel space. Such a space is defined as follows.

Definition A.1 (Standard Borel space [7]). A measurable space (X, Σ_X) is considered a *standard Borel space* when there exist

$$f : \mathbf{Meas}(X, \mathbb{R}) \qquad g : \mathbf{Meas}(\mathbb{R}, X)$$
$$g \circ f = id_X,$$

which then implies that (X, Σ_X) is either countable with a discrete σ-algebra or measurably isomorphic to the real line $(X, \Sigma_X) \simeq (\mathbb{R}, \Sigma_{\mathbb{R}})$.

The next definition will describe the right sort of sample spaces to model data-types in higher-order probabilistic programming, assuming a fixed standard Borel space $(\Omega, \Sigma_\Omega) \in Ob(\mathbf{Meas})$.

Definition A.2 (Quasi-Borel space [6]). A *quasi-Borel space* consists of a pair (X, M_X) where

$$X \in Ob(\mathbf{Set}) \qquad M_X \subseteq X^\Omega \in Ob(\mathbf{Set}),$$

such that the set M_X of "random variables" satisfies the following conditions:

- It is closed under measurable endomorphisms
$$\frac{f : \mathbf{Meas}(\Omega, \Omega) \qquad \rho \in M_X}{\rho \circ f \in M_X};$$
- It contains all constant random variables
$$\frac{\rho : \mathbf{Set}(\Omega, X) \qquad \exists f : \mathbf{Set}(\Omega, I), \exists g : \mathbf{Set}(I, X).g \circ f = \rho}{\rho \in M_X};$$
- Countable Borel partitions of the sample space Ω give countable coproduct random variables
$$\frac{\Omega = \coprod_{i \in \mathbb{N}} S_i \qquad \forall i \in \mathbb{N}, \rho_i \in M_X}{\beta = \coprod_{i \in \mathbb{N}} \rho_i \in M_X}.$$

In order to use quasi-Borel spaces as sample spaces for probabilistic programming, the next definition gives a notion of a probability measure on such a space.

Definition A.3 (Probability measure on a quasi-Borel space [6]). Given a quasi-Borel space (X, M_X), a *probability measure* on that space consists of a pair of a random variable from M_X and a probability measure on the underlying Borel space Ω

$$\frac{\rho \in M_X \qquad \mu \in \mathbb{P}(\Omega)}{(\rho, \mu) \in \mathbb{P}(X, M_X)}.$$

Finally, in order to give denotational semantics to higher-order probabilistic programs, quasi-Borel spaces and maps between them must form a category. The following definition gives the appropriate category, which Heunen et al. [6] showed to be Cartesian closed with support for arbitrary products and countable products.

Definition A.4 (Category of quasi-Borel spaces [6]). Given a fixed standard Borel space $(\Omega, \Sigma_\Omega) \in Ob(\mathbf{Meas})$, the *category* of quasi-Borel spaces **QBS** has

- Quasi-Borel spaces as objects $(X, M_X) \in Ob(\mathbf{QBS})$, and
- Functions preserving random variables as morphisms

$$\frac{f : \mathbf{Set}(X, Y) \qquad \forall \rho \in M_X, f \circ \rho \in M_Y}{f : \mathbf{QBS}((X, M_X), (Y, M_Y))}.$$

To use the category **QBS** as a model of probability theory, the next proposition will demonstrate that it supports an affine probability monad of pushforward measures.

Proposition A.5 (**QBS** has a monad of pushforward measures). *Given a fixed standard Borel space $(\Omega, \Sigma_\Omega) \in Ob(\mathbf{Meas})$ and a probability measure $p \in \mathbb{P}(\Omega)$ on it, probability measures on quasi-Borel spaces form a strong, affine probability monad $(\mathbb{P}, \eta, \mu)$.*

Proof. See Proposition 5.8 in Dash et al. [3] or Theorem 21 in Heunen et al. [6]. □

Finally, the next proposition shows that the resulting Kleisli category of monadic morphisms forms a Markov category, a synthetic model of probability theory (see Fritz [4] for definition).

Proposition A.6 (Probability kernels in **QBS** form a Markov category). *Probability kernels $\mathbf{QBS}(Z, \mathbb{P}(X))$ in **QBS** form a Markov category $\mathcal{Q}$, a semicartesian symmetric monoidal category.*

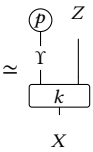
The next subsection will demonstrate that $\mathcal{Q}$ forms the right kind of Markov category to describe in terms of reading randomness from a random-number generator or trace.

A.2 Randomness pushback as semantics for randomness sources

Operationally, probabilistic programs sample from various distributions by drawing variates from a random-number generator, whose state must be passed along as an extra input to each subprogram. This operation receives semantics in a Markov category in terms of randomness pushback. This subsection describes randomness pushback, demonstrates that the Markov category **QBS** has randomness pushback, and then describes a convenient formulation of randomness pushback previously used in the literature to represent the operations of probabilistic programs.

Definition A.7 (Randomness pushback [4]). A Markov category $\mathcal{M}$ has *randomness pushback* if and only if for every morphism $f : \mathcal{M}(Z, X)$ there exists a randomness object $\Upsilon \in Ob(\mathcal{M})$, a distribution over that randomness object $p : \mathcal{M}(I, \Upsilon)$, and a deterministic mapping $k : \mathcal{M}_{det}(\Upsilon \times Z, X)$ which

together yield an isomorphism

$f \simeq$ 

.

In effect, Definition A.7 says that in a Markov category with randomness push*back*, every morphism is equivalent to some deterministic push*forward* of an independent source of randomness. Up until now, we have not described the standard Borel space (Ω, Σ_Ω) we use to construct **QBS**. The next definition will give a concrete space, taken from Dash et al. [3] with slight modification, and show that it is standard Borel.

Definition A.8 (Infinite rose tree [3]). Assume that there exists a countably infinite set, here taken to be the natural numbers $\mathbb{N}$, and also consider the finite lists over that set $\mathbb{N}^*$. Then the set of *infinite rose trees* consists of countably infinitely wide, countably infinitely deep trees, here annotated with an element of the unit interval at each node

$$\begin{aligned}\Omega &= [0,1] \times \prod_{n \in \mathbb{N}} \Omega \\ &= [0,1]^{\mathbb{N}^*}.\end{aligned}$$

The second line interprets each list of natural numbers as addressing a path into the tree from its root node, yielding the unit interval element found at that node. We then make this space an object in the category of measure spaces by equipping it with a product σ-algebra consisting of countably many copies of the Borel σ-algebra for the unit intervals at the nodes

$$\begin{gathered}\Sigma_\Omega = \prod_{n \in \mathbb{N}^*} \Sigma_{[0,1]} \\ (\Omega, \Sigma_\Omega) \in Ob(\mathbf{Meas}).\end{gathered}$$

Dash et al. [3] took infinite rose trees as their base sample space for treating probabilistic programs as quasi-Borel (and therefore quasi-measurable) maps. The next proposition shows that the space of infinite rose trees is standard Borel.

Proposition A.9 (Infinite rose trees are standard Borel spaces). *The space* $\Omega = [0,1]^{\mathbb{N}^*}$ *is a standard Borel space.*

Proof. The countability of both the natural numbers and their finite lists implies that we can biject lists of natural numbers onto natural numbers

$$\mathbb{N}^* \simeq \mathbb{N}$$

and therefore line up countably many copies of the unit next to each other

$$\Omega \simeq \prod_{n \in \mathbb{N}} [0,1]$$

to reassemble the real number line. Definition A.1 then shows that Ω is standard Borel. □

Having that the infinite rose tree provides a standard Borel sample space, the next definition will give a probability measure over infinite rose trees. Intuitively, this distribution models generating an infinite, lazily-evaluated stream of draws from the standard uniform distribution, placing one at each node of an infinite rose tree.

Definition A.10 (Uniform distribution over infinite rose trees). The uniform distribution over infinite rose trees consists of a countably infinite product of standard uniform distributions. It therefore assigns probability to measurable sets $\sigma_\Omega \in \Sigma_\Omega$ by taking the countable product of probability assigned by the standard uniform distribution to each measurable set's countable components

$$\begin{aligned} \mu &: \mathbb{P}(\Omega) \\ \mu &: \mathbf{Meas}(I, \mathbb{P}(\Omega, \Sigma_\Omega)) \\ \mu(I) = \sigma_\Omega &\mapsto \prod_{\alpha \in \mathbf{A}^*} U(0,1)\left((\sigma_\Omega)_\alpha\right). \end{aligned}$$

Having a concrete sample space Ω and a standard probability measure μ over it, the next theorem will demonstrate that the Markov category $\mathcal{Q}$ thereby has randomness pushback.

Theorem A.11 (Probability kernels in $\mathcal{Q}$ have randomness pushback). *The category $\mathcal{Q}$ (of probability kernels between quasi-Borel spaces) has randomness pushback (Definition A.7)*

$$\frac{f : \mathcal{Q}(Z, X)}{\exists \Upsilon \in Ob(\mathcal{Q}), \exists k : \mathbf{QUS}_{det}(Z, X^{\Omega_{\mathbb{N}}}), \exists p : \mathcal{Q}(I, \mathbb{P}(\Upsilon)), f \simeq z \mapsto \sigma_X \mapsto \int_{v \in \Upsilon} \mathbb{I}[k(z)(v) \in \sigma_X] p(dv)}.$$

Proof. We first observe that

$$\mathcal{Q}(Z, X) = \mathbf{QBS}(Z, \mathbb{P}(X))$$

and therefore, upon expansion to further detail

$$\frac{f : \mathcal{Q}(Z, X)}{f : \mathbf{QBS}(Z, \mathbb{P}(X, M_X))}.$$

Definition A.3 requires that f must map from the parameter Z to a random variable $\rho \in M_X$

$$\frac{f : \mathbf{QBS}(Z, \mathbb{P}(X, M_X))}{f \simeq v : \mathbf{QBS}_{det}(Z, M_X)}$$

which can combine with the uniform distribution $\mu : \mathbb{P}(\Omega)$ to witness randomness pushback

$$\Upsilon := \Omega \qquad k := v \qquad p := \mu$$

$$\begin{aligned} f(z) = \sigma_X &\mapsto \int_{v \in \Upsilon} \mathbb{I}[k(v, z) \in \sigma_X]\, p(dv) \\ f(z) = \sigma_X &\mapsto \int_{\omega \in \Omega} \mathbb{I}[v(z)(\omega) \in \sigma_X]\, \mu(d\omega). \end{aligned}$$

□

Theorem A.11 shows $\mathcal{Q}$ to be a Markov category with randomness pushback, licensing us to treat probability kernels as maps that deterministically push forward the uniform distribution over the sample space Ω. Probability kernels in $\mathcal{Q}$ serve as models for higher-order probabilistic programs that read from a randomness source to perform **sample** operations. The next subsection will further build up a model of probabilistic programs with a **factor** operation for observing feedback.

A.3 Measure kernels model probabilistic programs with unnormalized measures

Probabilistic programs do not only serve as random simulators. They also incorporate data by means of **observe**, **score**, or **factor** primitives. Doing so then denotes a non-probability measure over the sample space, whose integral across the whole sample space will have be arbitrary. Since the Markov category $\mathcal{Q}$ only models probability kernels, an additional *weighting* construction is required to model kernels with unnormalized measures, appropriately called *measure kernels*.

This subsection models weightings as morphisms $\mathcal{Q}(\cdot, \mathbb{R}_0^+)$ into the positive reals, and the next proposition demonstrates that such weightings can form a writer monad. This writer monad can lift probability kernels into measure kernels, and models the operation of executing a probabilistic program that reads randomness and writes out a weight to the inference state.

Proposition A.12 (Weightings form a writer monad [1]). *Weightings (morphisms $\mathcal{Q}(\cdot, \mathbb{R}_0^+)$) induce a commutative writer monad.*

Proof. Let $(\mathbb{R}_0^+, 1, \cdot)$ be the multiplication monoid over the positive reals. A writer monad for such a monoid consists of an endofunctor

$$\begin{aligned} W &: \mathcal{Q} \to \mathcal{Q} \\ W(Z) &= \mathbb{R}_0^+ \times Z & W(Z) &: Ob(\mathcal{Q}) \to Ob(\mathcal{Q}) \\ W(f) &= id_{\mathbb{R}_0^+} \times f & W(f) &: \mathcal{Q}(Z, X) \to \mathcal{Q}(\mathbb{R}_0^+ \times Z, \mathbb{R}_0^+ \times X), \end{aligned}$$

with natural transformations for the unit

$$\begin{aligned} \eta &: \mathrm{Id} \to W \\ \eta_Z(z) &= (1, z) & \eta_Z &: \mathcal{Q}(Z, \mathbb{R}_0^+ \times Z) \end{aligned}$$

and multiplication

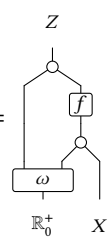
$$\begin{aligned} \mu &: W \circ W \to W \\ \mu_Z(w, w', z) &= (ww', z) & \mu_Z &: \mathcal{Q}(\mathbb{R}_0^+ \times \mathbb{R}_0^+ \times Z, \mathbb{R}_0^+ \times Z). \end{aligned}$$

□

Note that the Kleisli category of the weightings writer monad is *not* affine, as every weighting over an object yields a different morphism from that object to the unit $I \in Ob(Kl(W))$.

Definition A.13 (Categories of measure kernels). The category of *measure kernels* $\mathbf{MeasKer}(\mathcal{Q}) \subseteq \mathrm{Kl}(W)$ is the symmetric monoidal subcategory formed by considering Kleisli morphisms with deterministic weightings

$$f : \mathcal{Q}(Z, X) \qquad\qquad \omega : \mathcal{Q}_{det}(Z \times X, \mathbb{R}_0^+)$$

$(f, \omega) =$ 

$: \mathrm{Kl}(W)(Z, X).$

We call such a morphism a *measure kernel* $(f, \omega) : \mathbf{MeasKer}(\mathcal{Q})(Z, X)$.

B A NOVEL CATEGORY OF TRACING MEASURE KERNELS

Probabilistic programs typically emit traces and weights as they execute. The traces contain densities and random variates from a basic family of random variables, and combining the density over traces from a program with the weight emitted by that program gives an unnormalized density, corresponding to an unnormalized measure. This section will give a model in **QBS** of tracing measure kernels with densities. Section B.1 formalizes tracing by means of addressing random

[1] Example 5.1.7 in [14]

variables and their sample spaces. Section B.3 describes a way to interpret spaces of traces as having a base measure, and therefore as having densities induced by tracing probability kernels. Finally, Section B.2 extends the density construction to tracing measure kernels.

B.1 Formalizing tracing by addressing random variables

Definition A.8 and Theorem A.11 showed that morphisms in Q draw their randomness from an abstraction of the traces used in typical PPLs. This subsection will bring the model closer to actually-existing tracing. It considers a countably infinite set $\mathbf{A}$ of *addresses*, analogous to the set of strings in Python-based PPLs [1, 12] or the set of symbols in Lisp-based PPLs [5, 17, 19]. This set must include an empty address $\varepsilon \in \mathbf{A}$. Bounded sequences of addresses then form a space $\mathbf{A}^*$ of hierarchical addresses, which will also be countably infinite.

Addresses will be coupled to an element of a countable set $\mathbf{S}$ of support types, and so the first definition gives that set explicitly.

Definition B.1 (Sample-space types). Support or sample-space types $\mathbf{s} \in \mathbf{S}$ are drawn from the following grammar

$$\mathbf{S} := I \mid \mathbb{N} \mid \mathbb{Z} \mid \mathbb{R} \mid \mathbb{R}_0^+ \mid [0,1] \mid [1..n] : n \in \mathbb{N},$$

and the relevant quasi-Borel spaces are denoted by $[\![\mathbf{s}]\!] \in Ob(\mathbf{QBS})$. We can then also denote standard base measures over these (counting measure for discrete types, Lebesgue measure for continuous ones) as $\mu_{[\![\mathbf{s}]\!]}$.

The next definition shows how to use addresses to denote paths in infinite rose trees, giving a sample space that straightforwardly generalizes tracing in PPLs.

Definition B.2 (Addressed infinite rose trees). Addressed infinite rose trees consist of mappings from bounded address sequences to elements of the unit interval

$$\begin{aligned}\Omega &= \prod_{\alpha \in \mathbf{A}^*} [0,1] \\ &= [0,1]^{\mathbf{A}^*}.\end{aligned}$$

We can quickly justify the use of addressed infinite rose trees as a sample space, just as Proposition A.9 did for infinite rose trees over natural numbers.

Proposition B.3 (Addressed infinite rose trees are standard Borel spaces). *The space of addressed infinite rose trees $\Omega = [0,1]^{\mathbf{A}^*}$ is a standard Borel space and a measure space*

$$([0,1]^{\mathbf{A}^*}, \Sigma_{[0,1]^{\mathbf{A}^*}}) \in Ob(\mathbf{Meas}).$$

Proof. The set $\mathbf{A}$ is countably infinite, so its finite list set $\mathbf{A}^*$ is countably infinite. Being countable, we can just assign a natural number $\alpha \simeq n$ to each address $\alpha \in \mathbf{A}^*$ and then apply Proposition A.9. □

From here on, we will assume that we have constructed the category **QBS** with respect to a universal sample space of addressed infinite rose trees

$$\begin{aligned}\Omega &= [0,1]^{\mathbf{A}^*} \\ (\Omega, \Sigma_\Omega) &\in Ob(\mathbf{Meas}).\end{aligned}$$

We then define randomness sources in this universal sample space.

Definition B.4 (Lazy traces). A *lazy trace* is a probability measure $p : \mathbb{P}(\Omega)$ randomly assigning standard uniform variates to each address list and sample-space type.

Intuitively, a lazy trace is a random function under lazy evaluation[2]. When called with a novel argument, it samples from a standard uniform distribution and associates the new random variate with the argument. When called with an argument previously evaluated, it just returns the associated random variate. This interpretation provides operational intuition, but Proposition B.3 demonstrates that this is an appropriate standard randomness source for higher-order probability.

The next definition will use the notation

$$Z \rightharpoonup_n X = (X^Z)_n$$

to refer to partial functions from $Z \in Ob(\mathbf{Set})$ to $X \in Ob(\mathbf{Set})$ defined at only n arguments, assumed to be ordered. Such a partial function returns a privileged value $\perp$ where it is not defined. The following definition then models the common meaning of "trace" in probabilistic programming: a dictionary mapping addresses to random variates. It makes use of the fact that **QBS** has countable indexed coproducts.

Definition B.5 (Eager traces). An *eagerly-evaluated trace* is a partial function, defined at finitely many values, from address lists to unit interval elements and dependent pairs of sample type

$$\mathbb{T} := \coprod_{n \in \mathbb{N}} (\mathbf{A}^*) \rightharpoonup_n [0,1] \times \coprod_{\mathbf{s} \in \mathbf{S}} [\![\mathbf{s}]\!] \in Ob(\mathcal{Q}).$$

Eager traces form an object in $\mathcal{Q}$ because they are finite sets of pairs in which no first element repeats. Working with this set-theoretic notion of a partial function implies trivial definitions of cardinality, domain, and codomain operations for eager traces:

$$|\cdot| : \mathbb{T} \to \mathbb{N}$$

$$\mathrm{dom} : \mathbb{T} \to \coprod_{n \in \mathbb{N}} (\mathbf{A}^* \times \mathbf{S})^n \qquad \mathrm{cod} : \mathbb{T} \to \coprod_{n \in \mathbb{N}} [0,1]^n.$$

Probability kernels in $\mathcal{Q}$ take their randomness pushback from lazy traces. Reusing or reevaluating the random variates from an eager trace therefore requires embedding it into a lazy trace. The next proposition shows how to do so.

Proposition B.6 (Eager traces embed into lazy traces). *Given a lazy trace $p : \mathbb{P}(\Omega)$ and an eager trace $\tau : \mathbb{T}$, the operation $p[\tau]$ embeds the latter into the former.*

Proof. Given an infinite rose tree $\omega \sim \mathbb{P}(\Omega)$ and an eager trace $\tau : \mathbb{T}$, the operation $\omega[\tau]$ embeds the latter into the former

$$\omega[\tau] = \alpha \mapsto \begin{cases} \tau(\alpha)_1 & \alpha \in \mathrm{dom}(\tau) \\ \omega(\alpha) & \text{otherwise} \end{cases} \qquad \cdot[\cdot] : \Omega \times \mathbb{T} \to \Omega.$$

We can therefore see that

$$p[\tau] = (\omega \mapsto \omega[\tau]) \circ p.$$

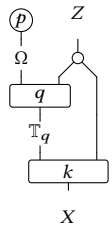
□

The next definition will specialize randomness pushback to permit separating the stochastic and deterministic portions of a probability kernel. Such morphisms will later be seen to admit a notion of a base measure and density on their trace subobjects.

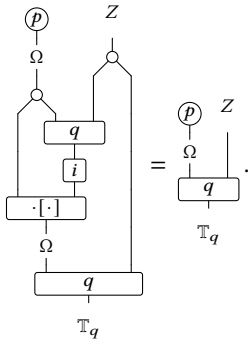
[2]Or in eagerly evaluated languages, a stochastically memoized function [5].

Definition B.7 (Randomness projection). A probability kernel $(p, q, k) : \mathcal{Q}(Z, X)$ admits a *randomness projection* when it "squeezes" all randomness through an internal wire conveying a subobject $i : \mathbb{T}_q \hookrightarrow_{\mathcal{Q}_{det}} \mathbb{T}$ of the eager traces.

$$p : \mathcal{Q}(I, \Omega) \qquad q : \mathcal{Q}_{det}(\Omega \times Z, \mathbb{T}_q) \qquad k : \mathcal{Q}_{det}(\mathbb{T}_q \times Z, X)$$

$(p, q, k) =$  $: \mathcal{Q}(Z, X).$

Since the subobject $\mathbb{T}_q$ may have only a subset of the random variables $M_{\mathbb{T}_q}$ of $\mathbb{T}$ as a whole, it must additionally be invariant to re-embedding into the lazy trace



The following proposition will show the connection between a common class of probability kernels and randomness projection. We leave considerations regarding infinite-dimensional random variables, such as Bayesian nonparametric models, to future work.

Proposition B.8 (Probability kernels over standard sample spaces with quantile functions have randomness projection). *A probability kernel $f : \mathcal{Q}(Z, X)$ admitting a quantile function (e.g. inverse CDF) $F^{-1} : \mathbf{Meas}(Z \times [0, 1], X)$ has randomness projection if $\exists \mathbf{s} \in \mathbf{S}.X = [\![\mathbf{s}]\!]$.*

Proof. We need only write the probability kernel $f : \mathcal{Q}(Z, X)$ in the form

$$p : \mathcal{Q}(I, \Omega) \qquad q : \mathcal{Q}_{det}(\Omega \times Z, \mathbb{T}_q) \qquad k : \mathcal{Q}_{det}(\mathbb{T}_q \times Z, X)$$

required by Definition B.7 to fill in the diagram. By Theorem A.11 we know that we can use the standard lazy trace as p. We then pick the empty address $\varepsilon \in \mathbf{A}$ and fill in the diagram

$$q(\omega, z) = \alpha \mapsto \begin{cases} (\omega(\varepsilon), (\mathbf{s}, F^{-1}(z, \omega(\varepsilon)))) & \alpha = \varepsilon \\ \bot & \text{otherwise} \end{cases}$$
$$k(\tau, z) = \tau(\varepsilon)_2.$$

□

In practice, most PPLs provide only random variables with quantile functions, and their outputs consist of deterministic quasi-Borel maps on those random variables. Finite-dimensional randomness pushback can therefore model most PPLs without loss of generality.

Intuitively, if a probability kernel has randomness projection, then a writer monad can capture the eager traces for later examination.

Proposition B.9 (Eager traces form a writer monad). *Eager traces $\mathbb{T}$ induce a writer monad for which probability kernels with randomness projection can serve as Kleisli morphisms.*

Proof. Let $(\mathbb{T}, \emptyset, \odot)$ be a monoid over eager traces defined by constructing binary trees over hierarchical addresses. A writer monad for such a monoid consists of an endofunctor

$$\begin{aligned} T &: \mathcal{Q} \to \mathcal{Q} \\ T(Z) &= \mathbb{T} \times Z & T(Z) &: Ob(\mathcal{Q}) \to Ob(\mathcal{Q}) \\ T(f) &= id_{\mathbb{T}} \times f & T(f) &: \mathcal{Q}(Z, X) \to \mathcal{Q}(\mathbb{T} \times Z, \mathbb{T} \times X), \end{aligned}$$

with natural transformations for the unit

$$\begin{aligned} \eta &: \mathrm{Id} \to T \\ \eta_Z(z) &= (\emptyset, z) & \eta_Z &: \mathcal{Q}(Z, \mathbb{T} \times Z), \end{aligned}$$

and multiplication

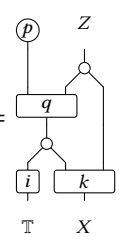
$$\begin{aligned} \mu &: T \circ T \to T \\ \mu_Z(\tau, \tau', z) &= (\tau \odot \tau', z) & \mu_Z &: \mathcal{Q}(\mathbb{T} \times \mathbb{T} \times Z, \mathbb{T} \times Z). \end{aligned}$$

□

This writer monad will have its own Kleisli category, which admits a trivial embedding of any morphism with randomness projection.

Definition B.10 (Eagerly traced probability kernels). The category of eagerly traced probability kernels $\mathbf{Traced}(\mathcal{Q}) \subseteq Kl(T)(\mathcal{Q})$ is the image of the following functor from the subcategory $\mathcal{Q}_{\mathbb{T}} \subseteq \mathcal{Q}$ with randomness projection into the Kleisli category of the writer monad:

$$\begin{aligned} F &: \mathcal{Q}_{\mathbb{T}} \to Kl(T)(\mathcal{Q}) \\ F(Z) &= \mathbb{T} \times Z & F(Z) &: Ob(\mathcal{Q}_{\mathbb{T}}) \to Ob(Kl(T)(\mathcal{Q})) \\ F((p, q, k)) &: \mathcal{Q}_{\mathbb{T}}(Z, X) \to Kl(T)(\mathcal{Q})(Z, X) \end{aligned}$$

$F((p, q, k)) =$ 

$: \mathbf{Traced}(\mathcal{Q})(Z, X).$

B.2 Eagerly traced measure kernels

Proposition B.11 (The weighting monad composes with the tracing monad). *The weighting monad above (Proposition A.12) composes with the tracing monad above (Proposition B.9) to form a monad for weighted, eagerly traced probability kernels.*

Proof. While monads in general do not compose, writer monads do (non-commutatively) compose. We obtain an endofunctor

$$
\begin{aligned}
W \circ T &: \mathcal{Q} \to \mathcal{Q} \\
(W \circ T)(Z) &= \mathbb{R}_0^+ \times \mathbb{T} \times Z \qquad & (W \circ T)(Z) &: Ob(\mathcal{Q}) \to Ob(\mathcal{Q}) \\
(W \circ T)(f) &= id_{\mathbb{R}_0^+} \times id_{\mathbb{T}} \times f \qquad & (W \circ T)(f) &: \mathcal{Q}(Z, X) \to \mathcal{Q}(\mathbb{R}_0^+ \times \mathbb{T} \times Z, \mathbb{R}_0^+ \times \mathbb{T} \times X),
\end{aligned}
$$

whose Kleisli category will have Kleisli morphisms

$$
\begin{aligned}
f &: Kl(W \circ T)(\mathcal{Q})(Z, X) \\
f &: Kl(W \circ T)(\mathcal{Q})(Z, \mathbb{R}_0^+ \times \mathbb{T} \times X).
\end{aligned}
$$

□

Note that the above monads are *not* affine: in programming terms, a Kleisli morphism of either monad still has a stateful effect (logs an output trace and/or weight) even if it produces no output.

Corollary B.12 (Categories of measure kernels on eagerly traced probability kernels). *The category of measure kernels on eagerly traced probability kernels*

$$\mathbf{MeasKer}(\mathbf{Traced}(\mathcal{Q})) \subseteq Kl(W \circ T)(\mathcal{Q}).$$

consists of trace-recording morphisms that also record a deterministic weight based on their domain, trace, and codomain.

Proof. We consider a measure kernel on a traced Markov kernel $(f, \omega) : \mathbf{MeasKer}(\mathbf{Traced}(\mathcal{Q}))$. This consists of a trace-recording Markov kernel

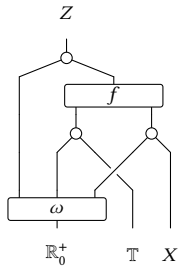
$$
\begin{aligned}
f &: \mathbf{Traced}(\mathcal{Q})(Z, X) \\
f &: \mathcal{Q}(Z, \mathbb{T} \times X),
\end{aligned}
$$

and a deterministic weighting over the Markov kernel's domain and codomain in $\mathcal{Q}$

$$\omega : \mathcal{Q}_{det}(Z \times \mathbb{T} \times X, \mathbb{R}_0^+).$$

We interpret these according to the diagram

$$
\begin{aligned}
(f, \omega) &: \mathbf{MeasKer}(\mathbf{Traced}(\mathcal{Q}))(Z, X) \\
(f, \omega) &: \mathcal{Q}(Z, \mathbb{R}_0^+ \times \mathbb{T} \times X)
\end{aligned}
$$

$(f, \omega) =$ 

.

□

B.3 Base measures and densities for eagerly traced kernels

Monte Carlo inference often requires evaluating the target measure's density at the points obtained by random sampling. While probabilistic programs operationally encode probability densities over their traces, densities have not been defined or shown to exist for probability kernels in **QBS** (and many other Markov categories). This section will define a natural notion of base measure on probability and measure kernels supporting eager tracing in the form of randomness pushback (Definition A.7) and randomness projection (Definition B.7).

The first definition gives the Lebesgue measure over infinite rose trees.

Definition B.13 (Lebesgue measure on an infinite rose tree). The space of infinite rose trees $\Omega = [0,1]^{\mathbf{A}^*}$ admits a *Lebesgue measure* defined via the Lebesgue measure $\lambda : \Sigma_{[0,1]} \to \mathbb{R}_0^+$

$$\lambda_\Omega : \Sigma_\Omega \to \mathbb{R}_0^+$$
$$\lambda_\Omega(\sigma) = \prod_{\alpha \in \mathbf{A}^*} \lambda(\sigma_\alpha).$$

The following proposition demonstrates that this is a σ-finite measure.

Proposition B.14 (The Lebesgue measure on infinite rose trees is σ-finite). *The Lebesgue measure λ_Ω on the space of infinite rose trees $(\Omega, \Sigma_\Omega) \in$ **Meas** is σ-finite.*

Proof. Each address list in the countable space $\mathbf{A}^*$ indexes a copy of the σ-algebra $\Sigma_{[0,1]}$ over the unit interval, and thus indexes a family of measurable subsets of the space of infinite rose trees. Picking the full unit interval from each "copy", we then have a countable family of disjoint sets each of which has Lebesgue measure 1.

Or more intuitively, since we can assemble the real line out of countably many unit intervals, the Lebesgue measure on that countable product of unit intervals is σ-finite. □

The next definition restates Proposition 14 of Heunen et al. [6].

Definition B.15 (σ-algebra on a quasi-Borel space). Given a base sample space Ω for the category **QBS**, the collection of all measurable subsets of a quasi-Borel space $(X, M_X) \in Ob(\mathbf{QBS})$, and the σ-algebra for that space, is defined as

$$\Sigma_{M_X} := \{S \subseteq X : \forall \rho \in M_X, \rho^{-1}(S) \in \Sigma_\Omega\}$$
$$(X, \Sigma_{M_X}) \in Ob(\mathbf{Meas}),$$

and this implies that quasi-Borel maps also induce measurable maps

$$\frac{f : \mathbf{QBS}((X, M_X), (Y, M_Y))}{f : \mathbf{Meas}((X, \Sigma_{M_X}), (Y, \Sigma_{M_Y}))}.$$

The following proposition gives a base measure on eager traces.

Definition B.16 (Base measure over eager traces). Eager traces have a sensible *base measure* defined by a sum of products, where the last component in each product is a sum over the indexed coproduct

$$(\mathbb{T}, M_\mathbb{T}) \in Ob(\mathbf{QBS})$$
$$(\mathbb{T}, \Sigma_{M_\mathbb{T}}) \in Ob(\mathbf{Meas})$$
$$\mu_\mathbb{T} : \Sigma_{M_\mathbb{T}} \to \mathbb{R}_0^+$$
$$\mu_\mathbb{T}(\sigma_{M_\mathbb{T}}) = \sum_{\tau \in \sigma_{M_\mathbb{T}}} \prod_{(\alpha, \mathbf{s}) \in \mathrm{dom}(\tau)} \mu_{\mathbf{A}^*}(\alpha) \lambda_{[0,1]}(\tau(\alpha)_1) \sum_{\mathbf{s}' \in \mathbf{S}} \delta_{\mathbf{s}}(\mathbf{s}') \mu_{[\![\mathbf{s}]\!]}(\tau(\alpha)_3).$$

The following theorem demonstrates that eagerly traced probability kernels support density functions over their traces.

Theorem B.17 (Eagerly traced probability kernels support densities). *Given an eagerly traced probability kernel $f \simeq (p, q, k) : \mathbf{Traced}(Q)(Z, X)$, there exists a Radon-Nikodym derivative with respect to the base measure on traces*

$$f(z)(\sigma_{M_{\mathbb{T}}}) = \int_{\sigma_{M_{\mathbb{T}}}} \frac{df}{d\mu_{\mathbb{T}}}\, d\mu_{\mathbb{T}}.$$

Proof. By Definition A.3, the given probability kernel maps takes inputs $z \in Z$ to a random variable that pushes forward a probability measure on Ω

$$\rho \in M_{[\![\mathbf{s}]\!]} \qquad \mu_\Omega \in \mathbb{P}(\Omega)$$
$$p = (\rho, \mu_\Omega).$$

Definition B.16 gives a base measure over eager traces with respect to which we can integrate, which being a sum of products of σ-finite measures (counting measure and Lebesgue measure) is itself σ-finite. The probability measure and this base measure both cover the space, and so the former is absolutely continuous with respect to the latter. The Radon-Nikodym Theorem then implies the existence of the desired derivative by virtue of the numerator being a pushforward of a standard probability measure and the denominator being a restriction a base measure to a subspace

$$\frac{df}{d\mu_{\mathbb{T}}} = \frac{d\left(q(\cdot, z)_*(p)\right)}{d\mu_{\mathbb{T}}}.$$

□

We can extend this result to give a notion of densities on eagerly traced probability kernels.

Corollary B.18 (Eagerly traced probability kernels have densities). *For each morphism $f \simeq (p, q, k) : \mathbf{Traced}(Q)(Z, X)$ consisting of a randomness pushback, randomness projection, and deterministic transformation, there exists a notion of a density over traces in general*

$$p_f(\cdot \mid z) : Q_{det}(Z \times \mathbb{T}, \mathbb{R}_0^+).$$

Proof. Theorem B.17 directly implies that there exists a Radon-Nikodym derivative

$$\frac{df}{d\mu_{\mathbb{T}_q}} : Q_{det}(Z \times \mathbb{T}_q, \mathbb{R}_0^+),$$

and so we merely need extend this to a proper density on the full space of traces. We do so by first giving the notion of embedding an eager trace into the kernel's lazy trace and reevaluating

$$q'(\tau, z) := q(p[\tau], z) \qquad q' : Q(\mathbb{T} \times Z, \mathbb{T}_q).$$

We then utilize the injection $i : \mathbb{T}_q \hookrightarrow \mathbb{T}$ to define the desired density by case analysis

$$p_f(\cdot \mid z) = \begin{cases} \frac{d(q(\cdot,z)_*(p))}{d\mu_{\mathbb{T}_q}}\left(i^{-1}(\tau)\right) & i(q'(\tau, z)) = \tau \\ 0 & \text{otherwise} \end{cases}.$$

□

The first case in the proof above corresponds to the notion of a probabilistic program's support from Stites et al. [16], while the extension to the broader space of traces corresponds to the notion of probabilistic program support in Cusumano-Towner et al. [2].

We can then proceed to the subcategory modeling typical probabilistic programs, one of eagerly traced measure kernels with unnormalized densities.

Corollary B.19 (Eagerly traced measure kernels have unnormalized densities). *An eagerly traced measure kernel We call* $(f, \omega) : \mathbf{MeasKer}(\mathbf{Traced}(\mathcal{Q}))(Z, X)$ *has an unnormalized density* $\gamma_{(f,\omega)}(\tau; z)$ *over traces and* $\gamma_{(f,\omega)}(x, \tau; z)$ *over traces and outputs.*

Proof. Follows directly from Corollary B.18 and Corollary B.12

$$(f, \omega) : \mathbf{MeasKer}(\mathbf{Traced}(\mathcal{Q}))(Z, X) \qquad f \simeq p_f(\tau \mid z)$$
$$f \simeq (p, q, k)$$
$$\gamma_{(f,\omega)}(\tau; z) := \omega(z, \tau, k(\tau, z)) p_f(\tau \mid z) \qquad \gamma_{(f,\omega)}(x; \tau, z) := \mathbb{I}[x = k(\tau, z)].$$

□

C CHANGES OF MEASURE KERNEL VIA TRACING, FROM PROPOSAL TO TARGET

Theorem C.1 generalizes Equation 3 of Stites et al. [16].

Theorem C.1 (Changes of measure give coslice categories of measure kernels). *Consider a category* $\mathbf{MeasKer}(\mathbf{Traced}(\mathcal{Q}))$ *of eagerly traced measure kernels. Its image* $\mathcal{Q}_{W \circ T} \subseteq \mathcal{Q}$ *then admits a coslice category* $I/\mathcal{Q}_{W \circ T}$*, which will have*

- *Objects* $(f, \omega_f) \in Ob(I/\mathcal{Q}_{W \circ T})$ *where* $(f, \omega_f) : \mathbf{MeasKer}(\mathbf{Traced}(\mathcal{Q}))(I, Z)$*;*
- *Morphisms* $(c, \omega_c) : (I/\mathcal{Q}_{W \circ T})((f, \omega_f), (g, \omega_g))$ *where*

$$(f, \omega_f) : \mathbf{MeasKer}(\mathbf{Traced}(\mathcal{Q}))(I, Z) \qquad (g, \omega_g) : \mathbf{MeasKer}(\mathbf{Traced}(\mathcal{Q}))(I, X)$$
$$(c, \omega_c) : \mathcal{Q}(\mathbb{R}_0^+ \times \mathbb{T} \times Z, \mathbb{R}_0^+ \times \mathbb{T} \times X)$$
$$(c, \omega_c) \circ_{\mathcal{Q}} (f, \omega_f) = (g, \omega_g).$$

These coslice morphisms (c, ω_c) *will correspond to change-of-measure ratios incorporated into the weightings.*

Proof. Each coslice morphism (c, ω_c) in the coslice category has as domain and codomain two eagerly traced measure kernels $(f, \omega_f) : \mathbf{MeasKer}(\mathbf{Traced}(\mathcal{Q}))(I, Z)$ and $(g, \omega_g) : \mathbf{MeasKer}(\mathbf{Traced}(\mathcal{Q}))(I, X)$, which will give rise to unnormalized densities we write in terms of the randomness traces, or random variables $z(\tau_f, I), x(\tau_g, I)$,

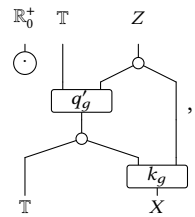
$$(f, \omega_f) \simeq \gamma_{(f,\omega_f)}(\tau_f; I)$$
$$(g, \omega_g) \simeq \gamma_{(g,\omega_g)}(\tau_g; I).$$

We construct a probability kernel between the codomains (including the writer-monad outputs) which takes advantage of randomness projection on g. In order to do so, we will utilize the entrywise cumulative distribution functions P_g and P_f on the elements of traces

$$P_f(\tau_f(\alpha) \mid \tau_f/\alpha) = \int_{s \in [\![\tau_f(\alpha)_2]\!]}^{\tau_f(\alpha)_3} p_f(s \mid \tau_f/\alpha)$$
$$P_g(\tau_g(\alpha) \mid \tau_g/\alpha) = \int_{s \in [\![\tau_g(\alpha)_2]\!]}^{\tau_g(\alpha)_3} p_g(s \mid \tau_g/\alpha).$$

We then define the coslice morphism $(c, \omega_c) : \mathbf{MeasKer}(\mathbf{Traced}(\mathcal{Q}))(\mathbb{R}_0^+ \times \mathbb{T} \times Z, X)$ in terms of its components, first an eagerly traced probability kernel

$$
\begin{aligned}
c &: \mathbf{Traced}(\mathcal{Q})(\mathbb{R}_0^+ \times \mathbb{T} \times Z, X) \\
g \simeq (p_g, q_g, k_g) &: \mathcal{Q}(I, \mathbb{T} \times X) \\
q'_g &: \mathcal{Q}(\mathbb{T} \times Z, \mathbb{T}_{q_g}) \\
q'_g(\tau_f, z) = \alpha^i &\mapsto \begin{cases} P_g\left(P_f^{-1}\left(\tau_f(\alpha^i)_3 \mid \tau_f/\alpha, z\right) \mid q'_g(\tau_f, z)^{(1:(i-1))}\right) & (\alpha^i, \mathbf{s}^i) \in \mathrm{dom}(\tau_f) \\ (q_g(\cdot, z) \circ p_g)(\alpha^i) & \text{otherwise} \end{cases} \quad \text{where} \\
\mathbf{s}^i &= (q_g(\cdot, z) \circ p_g)(\alpha^i)_2
\end{aligned}
$$

$c(w_f, \tau_f, z) =$ 

,

and second a weighting $\omega_c : \mathcal{Q}_{det}(\mathbb{R}_0^+ \times \mathbb{T} \times Z \times \mathbb{T} \times X, \mathbb{R}_0^+)$ that cancels the domain density in favor of the codomain density:

$$
\omega_c(w_f, \tau_f, z, \tau_g, x) = \frac{\gamma_{(g,\omega_g)}(\tau_g; I)}{\omega_f(I, \tau_f, z) \prod_{(\alpha,\mathbf{s}) \in \mathrm{dom}(\tau_f) \cap \mathrm{dom}(\tau_g)} p_f(\tau_f(\alpha) \mid \tau_f/\alpha)}. \tag{1}
$$

In addition to Equation 1 porting the importance weight of Equation 3 from Stites et al. [16] to a categorical setting, it also equals the Lightweight Metropolis-Hastings acceptance ratio [19].

Equality for measure kernels is defined by observational equivalence, so to have a proper importance sampling procedure, we now need to verify that for arbitrary test statistics $h : \mathcal{Q}_{det}(\mathbb{T} \times X, Y)$

$$
(c, \omega_c) \circ (f, \omega_f) = (g, \omega_g)
$$

$$
\mathbb{E}_{p_f(\tau_f, z|I)}\left[\omega_f(I, \tau_f, z) \mathbb{E}_{p_c(\tau_g, x|\tau_f, z)}\left[\omega_c(\tau_f, z, \tau_g, x) h(\tau_g, x)\right]\right] = \mathbb{E}_{p_g(\tau_g, x|I)}\left[\omega_g(I, \tau_g, x) h(\tau_g, x)\right]
$$

$$
\int_{z \in Z} \int_{x \in X} h(\tau_g, x)\, \gamma_{(c,\omega_c)}(\tau_g; \tau_f) dx\, \gamma_{(f,\omega_f)}(\tau_f; I)\, dz = \int_{x \in X} h(\tau_g, x)\, \gamma_{(g,\omega_g)}(\tau_g; I) dx.
$$

We observe that substituting $\omega_c(\tau_f, z, \tau_g, x)$ into the product of densities and weights on the left-hand side allows us to cancel a number of terms

$$
\gamma_{(f,\omega_f)}(\tau_f; I) \gamma_{(c,\omega_c)}(\tau_g; \tau_f) = \gamma_{(f,\omega_f)}(\tau_f; I) p_c(\tau_g, x \mid \tau_f, z) \omega_c(\tau_f, z, \tau_g, x)
$$

$$
\gamma_{(f,\omega_f)}(\tau_f; I) \gamma_{(c,\omega_c)}(\tau_g; \tau_f) =
$$

$$
\gamma_{(f,\omega_f)}(\tau_f; I) \cancel{p_c(\tau_g, x \mid \tau_f, z)} \frac{\gamma_{(g,\omega_g)}(\tau_g; I)}{\omega_f(I, \tau_f, z) \cancel{p_c(\tau_g, x \mid \tau_f, z)} \prod_{(\alpha,\mathbf{s}) \in \mathrm{dom}(\tau_f) \cap \mathrm{dom}(\tau_g)} p_f(\tau_f(\alpha) \mid \tau_f/\alpha)}
$$

$$
\gamma_{(f,\omega_f)}(\tau_f; I) \gamma_{(c,\omega_c)}(\tau_g; \tau_f) = \gamma_{(g,\omega_g)}(\tau_g; I) \frac{\gamma_{(f,\omega_f)}(\tau_f; I)}{\omega_f(I, \tau_f, z) \prod_{(\alpha,\mathbf{s}) \in \mathrm{dom}(\tau_f) \cap \mathrm{dom}(\tau_g)} p_f(\tau_f(\alpha) \mid \tau_f/\alpha)}
$$

$$
\gamma_{(f,\omega_f)}(\tau_f; I) \gamma_{(c,\omega_c)}(\tau_g; \tau_f) = \gamma_{(g,\omega_g)}(\tau_g; I) \prod_{(\alpha,\mathbf{s}) \in \mathrm{dom}(\tau_f)/\mathrm{dom}(\tau_g)} p_f(\tau_f(\alpha) \mid \tau_f/\alpha)
$$

and so upon substituting back into the left-hand side of equation above, we will obtain an identity

$$\int_{z\in Z}\int_{x\in X}\left(\gamma_{(f,\omega_f)}(\tau_f;I)\gamma_{(c,\omega_c)}(\tau_g;\tau_f)\right)h(\tau_g,x)\;dx\;dz=\int_{x\in X}h(\tau_g,x)\;\gamma_{(g,\omega_g)}(\tau_g;I)dx$$

$$\int_{z\in Z}\int_{x\in X}h(\tau_g,x)\;\gamma_{(g,\omega_g)}(\tau_g;I)dx\prod_{(\alpha,\mathbf{s})\in\mathrm{dom}(\tau_f)/\mathrm{dom}(\tau_g)}p_f(\tau_f(\alpha)\mid\tau_f/\alpha)dz=$$

$$\int_{x\in X}h(\tau_g,x)\;\gamma_{(g,\omega_g)}(\tau_g;I)dx.$$

□