An Enriched Calculus for Kernels and Linear Operators

PEDRO H. AZEVEDO DE AMORIM, University of Oxford, UK

1 INTRODUCTION

The last decade has seen rapid development in probabilistic semantics, a good portion of which fall into two categories: those based on monads and those based on linear operators. While there has been plenty of work on understanding what are the natural syntactic and semantic abstractions for programming with monads, its linear operator counterpart is still not fully developed, due to a mismatch between the natural affordances of linear logic and linear operators.

One of these mismatches was first observed during the full-abstraction study of the probabilistic coherence spaces model (**PCoh**) of linear logic[5], where it was noted that the familiar coKleisli semantics of the λ-calculus is unsuitable for practical probabilistic programming, since its call-by-name nature makes it impossible to reuse sampled values.

This was elegantly addressed in subsequent work on probabilistic variants of call-by-push-value (CBPV), where by making use of coalgebras for the exponential modality, one can recover reusability of samples at certain types [6]. This construction, however, could not be easily generalized to continuous datatypes, which has led to a careful semantic study of measurability paths in cones [4] which resulted in soundly adding a sample reuse operation, as long as the output type is $\mathbb{R}$. Recent work has finally found a model where spaces of measures can be equipped with a !-coalgebra structure, making it possible to define a syntax and semantics for a calculus with continuous datatypes and reusability of samples [3].

However, there are still a couple of shortcomings in this approach:

- The construction of such model is extremely involved and there does not seem to have underlying semantic structures that can be repurposed to other models. This can be problematic when studying variants of models, where such a construction would have to be done from scratch. Since there are many different aspects to probabilistic programming, such as its interactions with differentiable algorithms and cryptography, we should strive for developing semantic techniques that generalize beyond a single model.
- It departs from the deep connections between vector spaces and probability theory. Since one of the main use cases of denotational semantics is reasoning about programs, it is important to keep it as close as possible to well-established areas of mathematics, so that theorems from a large body of work can be readily used when reasoning about programs. Making the syntax rely so much on the exponential widens the gap between the semantics and the mathematical literature. The exponential should be mainly used as a syntactic artifact for unlocking the full expressive power of the λ-calculus when programming with linear operators.

As a way of mitigating these issues, another approach was recently suggested [1], where it is proposed a two-level calculus for programming with kernels and linear operators. Unfortunately, that calculus suffers from expressivity issues.

Contributions. In this abstract we extend the calculus of [1] using ideas from enriched category theory and provide a uniform way of lifting any model of the original calculus to this enriched setting. By doing this, we also provide a systematic semantic justification to the sampling construct of [4].

Author's address: Pedro H. Azevedo de Amorim, pedro.azevedo.de.amorim@cs.ox.ac.uk, University of Oxford, Oxford, UK.

$$\tau := \mathbb{N} \mid \tau \times \tau$$
$$\underline{\tau} := \mathcal{M}\tau \mid \underline{\tau} \multimap \underline{\tau} \mid \underline{\tau} \otimes \underline{\tau}$$

$$M, N := x \mid n \mid \text{let } x = M \text{ in } N \mid f(M) \mid (M, N) \mid \pi_1 M \mid \pi_2 N$$
$$t, u := x \mid n \mid \lambda x \,.\, t \mid t\ u \mid t \otimes u \mid \text{let } x \otimes y = t \text{ in } u \mid \text{sample } t_1, \dots, t_n \text{ as } x_1, \dots, x_n \text{ in } M$$

Fig. 1. Term and Type Grammars of λ_{MK}^{LL}

2 λ_{MK}^{LL}: A TWO-LEVEL SYSTEM FOR KERNELS AND LINEAR OPERATORS

We begin by presenting the two-level calculus λ_{MK}^{LL} [1]. It is based on categorical models of linear logic and on recent developments on synthetic probability theory [7]. The syntax is presented in Figure 1, where the first term grammar is a language for defining Markov kernels or, more abstractly, Kleili arrows, while the second grammar is a linear λ-calculus for defining linear operators. Its novelty is the introduction of the modality $\mathcal{M}$ that transports "Kleisli types" into linear types. The two-level structure gives rise to two typing judgement relations $\Gamma \vdash_{LL} t : \underline{\tau}$ and $\Delta \vdash_{MK} M : \tau$. Most of the typing rules are standard, the novelty is the rule for manipulating the $\mathcal{M}$ modality:

$$\frac{\{\Gamma_i \vdash_{LL} t_i : \mathcal{M}\tau_i\}_{i \in \{1,\dots,n\}} \qquad x_1 : \tau_1, \dots, x_n : \tau_n \vdash_{MK} M : \tau}{\Gamma_1, \dots, \Gamma_n \vdash_{LL} \text{sample } \{t_i\} \text{ as } \{x_i\} \text{ in } M : \mathcal{M}\tau}$$

Its operational interpretation is that given n programs $\{t_i\}_{i \in \{1,\dots,n\}}$ of type $\mathcal{M}\tau$, which should be thought of as programs that can be sampled from, we first sample from them, bind the results to variables $\{x_i\}_{i \in \{1,\dots,n\}}$ which are then used in the program M. Each typing judgement is interpreted in its own category. The linear language has its types interpreted in a symmetric monoidal closed category, contexts are interpreted as the tensor over the semantics of its elements and a well-typed program $x_1 : \tau_1, \dots, x_n : \tau_n \vdash_{LL} t : \tau$ is interpreted as a morphism $[\![\tau_1]\!] \otimes \cdots \otimes [\![\tau_n]\!] \to [\![\tau]\!]$, and the kernel language is interpreted in a similar manner, but using a CD category instead, which we now define.

Definition 2.1. A CD category is a symmetric monoidal category C equipped with transformations $A \to I$ and $A \to A \otimes A$ making certain diagrams commute.

The sampling modality will be interpreted as a lax monoidal functor, which we now define.

Definition 2.2. A lax monoidal functor is a functor between monoidal categories C and $\mathcal{D}$ equipped with a pair of natural transformations $\epsilon : I_{\mathcal{D}} \to FI_C$ and $\mu_{A,B} : FA \otimes_{\mathcal{D}} FB \to F(A \otimes_C B)$ satisfying certain coherence conditions.

One of the consequences of the coherence conditions is that the μ natural transformation can be extended uniquely to its n-ary variant $\mu^n_{A_i,\dots,A_n} : F(A_1) \otimes \cdots \otimes F(A_n) \to F(A_1 \otimes \cdots \otimes A_n)$. It is now possible to define what is a λ_{MK}^{LL} model.

Definition 2.3. A λ_{MK}^{LL} model is a triple $(C, \mathcal{L}, \mathcal{M})$ where C is a CD category, $\mathcal{L}$ is symmetric monoidal closed and $\mathcal{M} : C \to \mathcal{L}$ is a lax monoidal functor.

The semantics is mostly standard: the linear λ-calculus is interpreted using the symmetric monoidal closed structure of $\mathcal{L}$ and the CD category interprets the first-order expression language. Sample is the notable construct of the calculus and its semantics is defined using the lax monoidal structure of $\mathcal{M}$[1]:

$$\frac{\{\Gamma_i \xrightarrow{t_i} \mathcal{M}\tau_i\}_{i\in\{1,\dots,n\}} \qquad \tau_1 \times \cdots \times \tau_n \xrightarrow{N} \tau}{\Gamma_1 \otimes \cdots \otimes \Gamma_n \xrightarrow{t_1 \otimes \cdots \otimes t_n} \mathcal{M}\tau_1 \otimes \cdots \otimes \mathcal{M}\tau_n \xrightarrow{\mu^n} \mathcal{M}(\tau_1 \times \cdots \times \tau_n) \xrightarrow{\mathcal{M}N} \mathcal{M}\tau}$$

We now show how the $\mathcal{M}$ modality can used to copy and discard samples.

Example 2.4. The program sample t as x in $(x + x)$ samples from a distribution t, binds the value to x and adds the value to itself.

Example 2.5. The program sample t as x in 5 samples from t, binds the value to x and never uses it, outputting the constant 5.

This calculus admits many familiar models, we highlight two of them.

Example 2.6. Let **CStoch** be the category of countable sets and transitions matrices and $\mathcal{M}$: **CStoch** $\to$ **PCoh** be the functor that maps countable sets to the probabilistic coherence space of sub-probability distributions over them and it is the identity on morphisms. The triple (**CStoch**, **PCoh**, $\mathcal{M}$) is a λ^{LL}_{MK} model.

Example 2.7. Let **sStoch** be the category of measurable spaces and subMarkov kernels, **RoBan** is the category of regular ordered Banach spaces [2] and $\mathcal{M}$ is the functor that maps measurable spaces to the Banach space of signed measures and subMarkov kernels are mapped to their Kleisli extension. The triple (**Stoch**, **RoBan**, $\mathcal{M}$) is a λ^{LL}_{MK} model.

Note, however, that the functor $\mathcal{M}$ above is not full. For example, it is possible to use the decomposition $\mathcal{M}X = \mathcal{M}_d X \oplus \mathcal{M}_c X$, where $\mathcal{M}_d X$ are the discrete measures over X, $\mathcal{M}_c X$ are the continuous measures over X and $\oplus$ is the direct sum operation. Under this decomposition it is possible to define a linear operation that maps discrete measures to 0 and it the identity on continuous measures. This function is not in the image of the functor $\mathcal{M}$.

3 AN ENRICHED TWO-LEVEL CALCULUS

Though λ^{LL}_{MK} enables reusability and discardability of samples without requiring the exponential modality, it still suffers from some limitations as it does not handle parametric distributions well. For example, suppose that the linear language has been used for defining closed parametric distributions t_1, t_2 of type $\mathcal{M}\mathbb{R} \multimap \mathcal{M}\mathbb{R}$. Naturally, it should be possible to call this function from within the kernel language with the same parameter, e.g. let x = uniform in $t_1(x) + t_2(x)$. However, in its original form, the only interaction between languages occurs at their boundaries through the Sample construct, making the above program impossible to write in λ^{LL}_{MK}.

Upon closer inspection, this limitation is a consequence of, in general, there being more linear operators than kernels, meaning that in theory it should not possible to call a linear function from the kernel language because it could not be in the image of $\mathcal{M}$.

At a syntactic level, there should be a way of extending the calculus in a way that makes it possible to "reify" programs of type $\mathcal{M}\tau_1 \multimap \mathcal{M}\tau_2$ back into open MK programs of type $x : \tau_1 \vdash_{MK} t : \tau_2$. We propose an extension of λ^{LL}_{MK} using ideas from enriched category theory in order to accommodate such reification primitives. We start with a basic lemma from enriched category theory.

Lemma 3.1. *If $F : \mathcal{C} \to \mathcal{L}$ is a full and faithful functor and $\mathcal{L}$ a symmetric monoidal closed category then $\mathcal{C}$ is an $\mathcal{L}$-category.*

[1]Without the appropriate color coding

$$\begin{tikzcd} & \widetilde{\mathcal{L}} \arrow[d, "U"] \\ C \arrow[ur, "\widetilde{\mathcal{M}}"] \arrow[r, "\mathcal{M}"'] & \mathcal{L} \end{tikzcd}$$

Fig. 2. Lifted model

$$\begin{tikzcd} (\mathcal{N} \downarrow id) \arrow[r] \arrow[d] & \mathbf{Set}^{C^{op}} \arrow[d, "id"] \\ \mathcal{L} \arrow[r, "\mathcal{N}"'] & \mathbf{Set}^{C^{op}} \end{tikzcd}$$

Fig. 3. Glued model along the nerve functor

In our case, since $\mathcal{L}$ is assumed to be monoidal closed, it is enriched over itself and C is also enriched over $\mathcal{L}$ by using the hom-object $C(A,B) = \mathcal{M}A \multimap \mathcal{M}B$. This motivates changing a bit the interpretation of programs. A program $\Delta \vdash_{MK} M : \tau$ should be interpreted as an $\mathcal{L}$ morphism $I \to (\mathcal{M} [\![\Delta]\!] \multimap \mathcal{M}\tau)$, i.e. a global element of $\mathcal{M} [\![\Delta]\!] \multimap \mathcal{M} [\![\tau]\!]$. Since now the entire semantics of λ_{MK}^{LL} lives inside of $\mathcal{L}$, it is possible to make kernel programs depend on linear contexts: $\Gamma; \Delta \vdash_{MK} M : \tau$ and its semantics is an $\mathcal{L}$-morphism $[\![\Gamma]\!] \to (\mathcal{M} [\![\Delta]\!] \multimap \mathcal{M} [\![\tau]\!])$. Furthermore, we can add the following rule:

$$\frac{\Gamma_1; \Delta \vdash_{MK} M : \tau \qquad \Gamma_2 \vdash_{LL} t : \mathcal{M}\tau \multimap \mathcal{M}\tau'}{\Gamma_1, \Gamma_2; \Delta \vdash_{MK} t(M) : \tau'}$$

Note that this rule is very similar to the one defined in [4] where the authors introduce an eager sampling rule that has as restriction that the output must be of type $\mathbb{R}$.

Definition 3.2. An enriched λ_{MK}^{LL} model is an λ_{MK}^{LL} model $(C, \mathcal{L}, \mathcal{M})$ such that $\mathcal{M}$ is full and faithful.

Under this categorical structure the new rule is interpreted as

$$\frac{\Gamma_1 \xrightarrow{M} (\mathcal{M}\Delta \multimap \mathcal{M}\tau) \qquad \Gamma_2 \xrightarrow{t} (\mathcal{M}\tau \multimap \mathcal{M}\tau')}{\Gamma_1 \otimes \Gamma_2 \xrightarrow{t \circ M} (\mathcal{M}\Delta \multimap \mathcal{M}\tau')}$$

Example 3.3 (Parametric Box-Muller Transform). Let $\cdot \vdash_{LL} \mathsf{uniform} : \mathcal{M}\mathbb{R} \multimap \mathcal{M}\mathbb{R}$ be the program that, given a real number r, samples from the uniform distribution over the interval $[0, r]$ and define

$$\begin{array}{l} \lambda\, \mu.\, \mathsf{sample}\ \mu\ \mathsf{as}\ r\ \mathsf{in} \\ \mathsf{let}\ x{=}\mathsf{uniform}(r)\ \mathsf{in} \\ \mathsf{let}\ y{=}\mathsf{uniform}(r)\ \mathsf{in} \\ \left(\sqrt{-2\ln\frac{y}{r}}\sin(2\pi x), \sqrt{-2\ln\frac{y}{r}}\cos(2\pi x)\right) \end{array}$$

4 CANONICAL LIFTING

Though the enriched version of λ_{MK}^{LL} mitigates the expressivity issues of its non-enriched variant, at first it suffers from the same limitations as the cones models [4], where it would be necessary a possibly complicated semantic argument to justify the soundness of the extension. Fortunately, there is a straightforward way of lifting λ_{MK}^{LL} models to an enriched model by using a categorical gluing argument.

Theorem 4.1. *Every λ_{MK}^{LL} model $(C, \mathcal{L}, \mathcal{M})$ such that $\mathcal{L}$ is locally small lifts to a model $(C, \widetilde{\mathcal{L}}, \widetilde{\mathcal{M}})$ such that $\widetilde{\mathcal{M}}$ is full, there is a fibred functor $U : \widetilde{\mathcal{L}} \to \mathcal{L}$ and $\mathcal{M} = U \circ \widetilde{\mathcal{M}}$, see Figure 2.*

Proof. The proof follows by gluing along the nerve functor, see Figure 3

$$\mathcal{N} : \mathcal{L} \to \mathbf{Set}^{C^{op}}$$
$$\mathcal{N}(V) = \mathcal{L}(\mathcal{M}-, V)$$

It is possible to prove that $\mathcal{N}$ is lax monoidal with respect to the the Day monoidal closed structure of presheaf categories. By using the Hyland and Schalk gluing construction [8], it is possible to show that the comma category $(\mathcal{N} \downarrow id)$ is symmetric monoidal closed with the forgetful functor being a fibration and preserving the symmetric monoidal close on the nose. □

In particular, there is a lifting of the **RoBan** model such that the functor is full. More concretely, the glued model looks quite similar to the measurability requirements of [4], where the objects of $\widetilde{\mathcal{L}}$ are pairs (A, X), where X is an object of $\mathcal{L}$ and X is a plot set of maps $\mathcal{M}B \to A$, for every $B \in C$.

5 FUTURE WORK

Though we have shown that λ_{MK}^{LL} models lift to enriched ones, it is still important to understand what are other structures that will lift to the glued model. In particular, if $\mathcal{L}$ has an exponential, will it also exist in the glued model? By using a double-gluing orthogonality construction, does $*$-autonomy also lift? We are also interested in relating the expressivity of the calculus proposed in [4] with the enriched version of λ_{MK}^{LL}. While at the semantic level they both can potentially represent every Markov kernel, it is unclear if there is a syntatic translation of the former into the latter.

REFERENCES

[1] Pedro H Azevedo de Amorim. 2023. A Higher-Order Language for Markov Kernels and Linear Operators.. In *Foundations of Software Science and Computation Structures (FoSSaCS).*

[2] Fredrik Dahlqvist and Dexter Kozen. 2019. Semantics of higher-order probabilistic programs with conditioning. In *Principles of Programming Languages (POPL).*

[3] Thomas Ehrhard and Guillaume Geoffroy. 2022. Integration in Cones. *arXiv preprint arXiv:2212.02371* (2022).

[4] Thomas Ehrhard, Michele Pagani, and Christine Tasson. 2017. Measurable cones and stable, measurable functions: a model for probabilistic higher-order programming. In *Principles of Programming Languages (POPL).*

[5] Thomas Ehrhard, Michele Pagani, and Christine Tasson. 2018. Full abstraction for probabilistic PCF. *Journal of the ACM (JACM)* (2018).

[6] Thomas Ehrhard and Christine Tasson. 2019. Probabilistic call by push value. *Logical Methods in Computer Science* (2019).

[7] Tobias Fritz. 2020. A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics. *Advances in Mathematics* 370 (2020), 107239.

[8] Martin Hyland and Andrea Schalk. 2003. Glueing and orthogonality for models of linear logic. *Theoretical computer science* (2003).