Denotational validation of higher-order Bayesian inference

Yufei Cai, Zoubin Ghahramani, Chris Heunen, Ohad Kammar, Sean K. Moss, Klaus Ostermann, Adam Ścibior, Sam Staton, Matthijs Vákár, and Hongseok Yang

Séminaire Plume
Preuves et Langages Un Manège Enchanté
École normale supérieure de Lyon
12 July 2017

What is probabilistic programming?

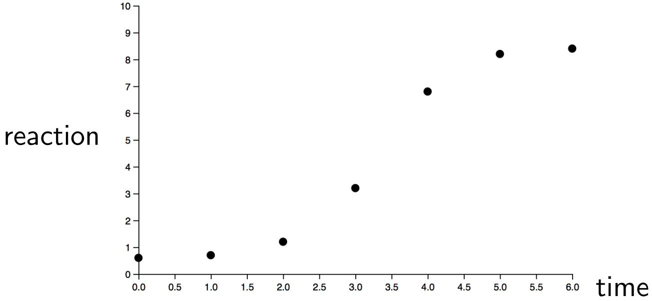
Bayesian data modelling

1. Develop a probabilistic (generative) model.
2. Design an inference algorithm for the model.
3. Using the algorithm, fit the model to the data.

What is probabilistic programming?

Example

Effect of a drug on a patient, given data:



What is probabilistic programming?

Generative model

$$
\begin{aligned}
s &\sim \mathsf{normal}(0, 2) \\
b &\sim \mathsf{normal}(0, 6) \\
f(x) &= s \cdot x + b \\
y_i &= \mathsf{normal}(f(i), 0.5) \\
&\qquad \text{for } i = 0 \ldots 6
\end{aligned}
$$

What is probabilistic programming?

Generative model

$$\begin{aligned} s &\sim \text{normal}(0, 2) \\ b &\sim \text{normal}(0, 6) \\ f(x) &= s \cdot x + b \\ y_i &= \text{normal}(f(i), 0.5) \\ &\qquad \text{for } i = 0 \ldots 6 \end{aligned}$$

Conditioning

$y_0 = 0.6, y_1 = 0.7, y_2 = 1.2, y_3 = 3.2, y_4 = 6.8, y_5 = 8.2, y_6 = 8.4$

Predict f?

What is probabilistic programming?

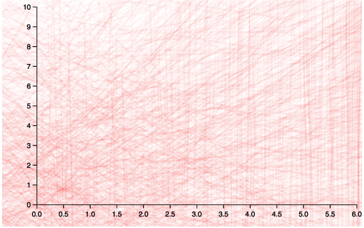
Bayesian inference

$$P(s, b | y_0, \ldots, y_6) = \frac{P(y_0, \ldots, y_6 | s, b) \cdot P(s, b)}{P(y_0, \ldots, y_6)}$$

What is probabilistic programming?

Bayesian inference

$$P(s, b | y_0, \ldots, y_6) = \frac{P(y_0, \ldots, y_6 | s, b) \cdot P(s, b)}{P(y_0, \ldots, y_6)}$$

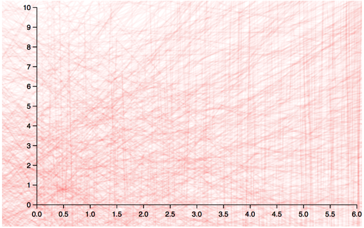


Prior

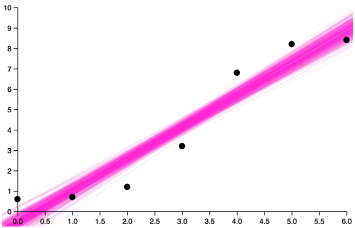
What is probabilistic programming?

Bayesian inference

$$P(s, b | y_0, \ldots, y_6) = \frac{P(y_0, \ldots, y_6 | s, b) \cdot P(s, b)}{P(y_0, \ldots, y_6)}$$



Prior



Posterior

What is probabilistic programming?

Probabilistic programming models

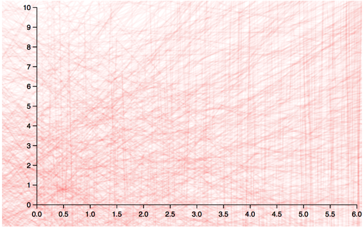
1. Develop a probabilistic (generative) model.
   Write a program.
2. ~~Design an inference algorithm for the model.~~
3. Using the built-in algorithm, fit the model to the data.

What is probabilistic programming?

In Anglican [Wood et al.'14]

```
(let [s (sample (normal 0.0 2.0))
      b (sample (normal 0.0 6.0))
      f (fn [x] (+ (* s x) b)))]
```

```
(predict :f f))
```



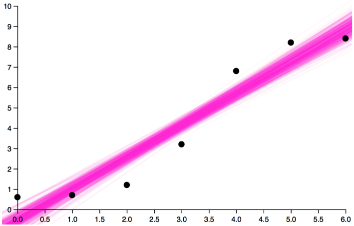
What is probabilistic programming?

In Anglican [Wood et al.'14]

```
(let [s (sample (normal 0.0 2.0))
      b (sample (normal 0.0 6.0))
      f (fn [x] (+ (* s x) b))]

  (observe (normal (f 1.0) 0.5) 2.5)
  (observe (normal (f 2.0) 0.5) 3.8)
  (observe (normal (f 3.0) 0.5) 4.5)
  (observe (normal (f 4.0) 0.5) 6.2)
  (observe (normal (f 5.0) 0.5) 8.0)

  (predict :f f))
```

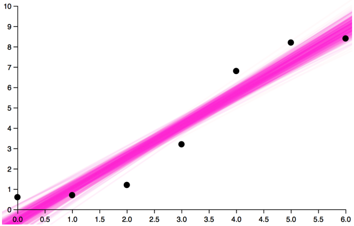


What is probabilistic programming?

In Anglican [Wood et al.'14]

```
(let [F (fn [] (let [s (sample (normal 0.0 2.0))
                     b (sample (normal 0.0 6.0))]
          (fn [x] (+ (* s x) b))))
      f (F)]
  (observe (normal (f 1.0) 0.5) 2.5)
  (observe (normal (f 2.0) 0.5) 3.8)
  (observe (normal (f 3.0) 0.5) 4.5)
  (observe (normal (f 4.0) 0.5) 6.2)
  (observe (normal (f 5.0) 0.5) 8.0)

  (predict :f f))
```

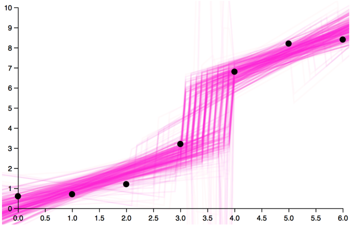


What is probabilistic programming?

In Anglican [Wood et al.'14]

```
(let [F (fn [] (let [s (sample (normal 0.0 2.0))
                     b (sample (normal 0.0 6.0))]
           (fn [x] (+ (* s x) b))))
      f (add-change-points F 0 6) ]
  (observe (normal (f 1.0) 0.5) 2.5)
  (observe (normal (f 2.0) 0.5) 3.8)
  (observe (normal (f 3.0) 0.5) 4.5)
  (observe (normal (f 4.0) 0.5) 6.2)
  (observe (normal (f 5.0) 0.5) 8.0)

  (predict :f f))
```

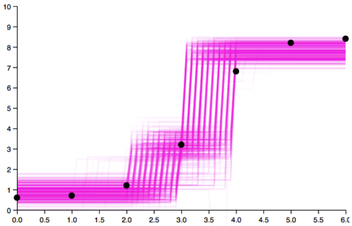


What is probabilistic programming?

In Anglican [Wood et al.'14]

```
(let [F (fn [] (let [
                     b (sample (normal 0.0 6.0))]
          (fn [x]    b )))
      f (add-change-points F 0 6) ]
  (observe (normal (f 1.0) 0.5) 2.5)
  (observe (normal (f 2.0) 0.5) 3.8)
  (observe (normal (f 3.0) 0.5) 4.5)
  (observe (normal (f 4.0) 0.5) 6.2)
  (observe (normal (f 5.0) 0.5) 8.0)

  (predict :f f))
```



What is probabilistic programming?

Components

- Control flow, e.g.: simply typed λ-calculus
- data types, e.g.: lists, functions, thunks
- Probabilistic choice: `(sample (normal 0.0 2.0))`
- Conditioning: `(observe (normal (f 2.0) 0.5) 3.8)`

What is probabilistic programming?

Components

- Control flow, e.g.: simply typed λ-calculus
- data types, e.g.: lists, functions, thunks
- Probabilistic choice: `(sample (normal 0.0 2.0))`
- Conditioning: `(observe (normal (f 2.0) 0.5) 3.8)`

$$\text{posterior} \propto \text{liklihood} \times \text{prior}$$

What is probabilistic programming?

Components

- Control flow, e.g.: simply typed λ-calculus
- data types, e.g.: lists, functions, thunks
- Probabilistic choice: `(sample (normal 0.0 2.0))`
- Conditioning: `(observe (normal (f 2.0) 0.5) 3.8)`

$$\text{posterior} \propto \text{liklihood} \times \text{prior}$$

Which we refine to:

$$\text{posterior} = \text{weight} \odot \text{prior}$$

Some measure theory

Rescaling

$$\nu = w \odot \mu$$

when for all $\chi : X \to [0, \infty]$:

$$\int_X \chi(x)\nu(\mathrm{d}x) = \int_X \chi(x) \cdot w(x)\mu(\mathrm{d}x)$$

(where X measurable space, $\mu \in MX$ measures on X,
$w : X \to [0, \infty]$ measurable function)

Theorem (Radon-Nikodym)

For all finite ν, μ: if such w exists, then it is unique μ-almost everywhere.

Write: $\nu \ll \mu$, $w = \frac{\mathrm{d}\nu}{\mathrm{d}\mu}$

What is probabilistic programming?

A probabilistic program is a measure

For $t : X$

$$[\![t]\!] = w \odot \text{prior}\,[\![t]\!]$$

where prior $[\![t]\!]$ is the **prior** (ignore conditioning),
and $w = \frac{\mathrm{d}[\![t]\!]}{\mathrm{d}(\text{prior}[\![t]\!])}$

Conditioning

$$\frac{t : x \qquad \varphi : X \to [0, +\infty]}{\texttt{observe}(t, \varphi) : 1}$$

and

$$[\![\texttt{observe}]\!]\,(x, \varphi) = \varphi(x) \odot \delta_{()}$$

What is probabilistic programming?

A probabilistic program is a measure

For $t : X$

$$[\![t]\!] = w \odot \mathsf{prior}\,[\![t]\!]$$

where prior $[\![t]\!]$ is the **prior** (ignore conditioning),
and $w = \frac{\mathrm{d}[\![t]\!]}{\mathrm{d}(\mathsf{prior}[\![t]\!])}$

Conditioning

Replace `observe` by score :

$$\frac{r : [0, \infty]}{\mathsf{score}\, r : 1}$$

and

$$[\![\mathsf{score}\,]\!]\,(r) = r \odot \delta_{()}$$

What is probabilistic programming?

A probabilistic program is a measure

For $t : X$

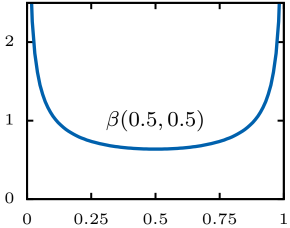
$$[\![t]\!] = w \odot \text{prior}\, [\![t]\!]$$

where prior $[\![t]\!]$ is the **prior** (ignore conditioning),
and $w = \frac{\mathrm{d}[\![t]\!]}{\mathrm{d}(\text{prior}[\![t]\!])}$

Note

For probability measures prior $[\![t]\!]$:

- It's possible that $\max w > 1$, e.g.:



  or even $\max w = \infty$

- If we insist that all measures are sub-probability measures, then w and $[\![t]\!]$ are **not** compositional (i.e., global)

What is probabilistic programming?

A probabilistic program is an s-finite measure [Staton'17]

For $t : X$

$$[\![t]\!] = w \odot \mathsf{prior}\,[\![t]\!]$$

where prior $[\![t]\!]$ is the **prior** (ignore conditioning),
and $w = \frac{\mathrm{d}[\![t]\!]}{\mathrm{d}(\mathsf{prior}[\![t]\!])}$
Sampling manipulates prior.
Conditioning affects w, sequenced multiplicatively.

S-finite measures

$$\sum_{i \in \mathbb{N}} \mu_i$$

μ_i finite: $\mu_i(X) < \infty$

What is inference?

Computing distributions

For $t : X$

$$[\![t]\!] = w \odot \text{prior}\,[\![t]\!]$$

we want to:

- Plot $[\![t]\!]$.
- Sample $[\![t]\!]$ (e.g., to make prediction)

Challenge

Given a fair coin $(\frac{1}{2}\delta_1 + \frac{1}{2}\delta_0)$, how do we sample from a biased coin $(p\delta_1 + (1-p)\delta_0)$?
Generalise:
Given a prior distribution prior $[\![t]\!]$, how do we sample from $[\![t]\!]$?

What is inference?

Programming-language experts needed

In the traditional areas:

- Verification
- Correctness
- Static analysis
- Semantics
- Optimisation
- Programming abstractions
- Type systems

This talk

Correctness of inference

Inference algorithm: distribution/meaning preserving transformation from one inference representation to another

Requirements

- Represented data is continuous
- Compositional inference representations (IRs)
- IRs are **higher-order**

Traditional measure theory. . .

This talk

Correctness of inference

Inference algorithm: distribution/meaning preserving transformation from one inference representation to another

Requirements

- Represented data is continuous
- Compositional inference representations (IRs)
- IRs are **higher-order**

Traditional measure theory... is unsuitable:

Theorem (Aumann'61)

The set $\mathbf{Meas}(\mathbb{R}, \mathbb{R})$ *cannot be made into a measurable space with*

$$eval : \mathbf{Meas}(\mathbb{R}, \mathbb{R}) \times \mathbb{R} \to \mathbb{R}$$

measurable.

Contribution

Correctness of inference

- Modular validation of inference algorithms:
  Sequential Monte Carlo, Trace Markov Chain Monte Carlo
  By combining:
- Synthetic measure theory [Kock'12]: measure theory without measurable spaces
- Quasi-Borel spaces: a convenient category for higher-order measure theory [LICS'17]

Talk structure

- Probabilistic programming and Bayesian inference
- Synthetic measure theory
- Quasi-Borel spaces
- Inference representations
- Trace Markov Chain Monte Carlo (Trace MCMC)
- Conclusion

Synthetic measure theory: axioms

Measure category [Kock'12]

A pair $(\mathcal{C}, \underline{\mathrm{M}})$

- Cartesian-closed category $\mathcal{C}$

Synthetic measure theory: axioms

Measure category [Kock'12]

A pair $(\mathcal{C}, \underline{\mathrm{M}})$

- Cartesian-closed category $\mathcal{C}$
- Countable coproducts and countable limits

Synthetic measure theory: axioms

Measure category [Kock'12]

A pair $(\mathcal{C}, \underline{\mathrm{M}})$

- Cartesian-closed category $\mathcal{C}$
- Countable coproducts and countable limits
- $\underline{\mathrm{M}} = (\mathrm{M}, \mathrm{return}\,, \gg\!=)$ a strong commutative monad, i.e.:

$$\mathrm{M} : |\mathcal{C}| \to |\mathcal{C}| \qquad \mathrm{return}_X : X \to \mathrm{M}\,X$$

$$\gg\!=_{X,Y} : \mathrm{M}\,X \times (\mathrm{M}\,Y)^X \to \mathrm{M}\,Y$$

satisfying the monad laws and

$$\underline{T}.\mathbf{do}\,\{x \leftarrow a; y \leftarrow b; \mathbf{return}(x, y)\}$$
$$=$$
$$\underline{T}.\mathbf{do}\,\{y \leftarrow b; x \leftarrow a; \mathbf{return}(x, y)\}$$

Synthetic measure theory: axioms

Measure category [Kock'12]

A pair $(\mathcal{C}, \underline{\mathrm{M}})$

- Cartesian-closed category $\mathcal{C}$
- Countable coproducts and countable limits
- $\underline{\mathrm{M}} = (\mathrm{M}, \mathrm{return}, \gg\!=)$ a strong commutative monad, i.e.:
- Canonical morphisms are invertible:

$$\mathrm{M}\,\mathbb{0} \cong \mathbb{1} \qquad \mathrm{M}(\coprod_{n\in\mathbb{N}} X) \cong \prod_{n\in\mathbb{N}} \mathrm{M}\,X$$

Synthetic measure theory: consequences

Surprisingly rich structure

- $0 : \mathbb{1} \to \mathrm{M}\,\mathbb{0}$
- $\sum_{n\in\mathbb{N}} X : \prod_{i\in\mathbb{N}} \cong \mathrm{M}(\coprod_{i\in\mathbb{N}} X) \xrightarrow{\mathrm{M}\,\nabla} \mathrm{M}\,X$
- $R := \mathrm{M}\,\mathbb{1}$ a σ-semiring:

$$(\cdot) : R \times R \xrightarrow{\text{double strength}} R \qquad 1 := \mathrm{return}() \in R$$

- Every algebra is an R-module:

$$\odot : R \times \mathrm{M}\,X \xrightarrow{\text{strength}} \mathrm{M}\,X$$

- Associated affine monad:

$$\mathrm{P}\,X \overset{\mathrm{sub}_X}{\dashrightarrow} \mathrm{M}\,X \underset{\underline{1}}{\overset{\mathrm{M}!}{\rightrightarrows}} R$$

Synthetic measure theory: notation

Kock integration

$$\oint_X f(x)\underline{\mu}(\mathrm{d}x) \coloneqq \underline{\mu} \gg= f$$

- Measure-valued, hence analogous to

$$\int_X \chi(x) \cdot f(x)\underline{\mu}(\mathrm{d}x)$$

  for generic $\chi : X \to [0, \infty)$
- η-expanded integrand

Synthetic measure theory: notation

Notation	Meaning	Terminology
R	$:= \mathrm{M}\,\mathbb{1}$	Scalars
$f_*\underline{\mu}$	$:= (\mathrm{M}\,f)(\underline{\mu})$	Push-forward

Synthetic measure theory: notation

Notation	Meaning	Terminology
R	$:= \mathrm{M}\,\mathbb{1}$	Scalars
$f_*\underline{\mu}$	$:= (\mathrm{M}\,f)(\underline{\mu})$	Push-forward
$\underline{\mu}(X)$	$:= !_*\underline{\mu}$	The total measure

Synthetic measure theory: notation

Notation	Meaning	Terminology
R	$:= \mathrm{M}\,\mathbb{1}$	Scalars
$f_*\underline{\mu}$	$:= (\mathrm{M}\,f)(\underline{\mu})$	Push-forward
$\underline{\mu}(X)$	$:= !_*\underline{\mu}$	The total measure
$\underline{\delta}_x$	$:= \mathbf{return}(x)$	Dirac distribution

Synthetic measure theory: notation

Notation	Meaning	Terminology
R	$:= \mathrm{M}\,\mathbb{1}$	Scalars
$f_*\underline{\mu}$	$:= (\mathrm{M}\,f)(\underline{\mu})$	Push-forward
$\underline{\mu}(X)$	$:= !_*\underline{\mu}$	The total measure
$\underline{\delta}_x$	$:= \mathbf{return}(x)$	Dirac distribution
$⨏_X f(x)\underline{\mu}(\mathrm{d}x)$	$:= \underline{\mu} \gg= f$	Kock integral

Synthetic measure theory: notation

Notation	Meaning	Terminology
R	$:= \mathrm{M}\,\mathbb{1}$	Scalars
$f_* \underline{\mu}$	$:= (\mathrm{M}\,f)(\underline{\mu})$	Push-forward
$\underline{\mu}(X)$	$:= !_* \underline{\mu}$	The total measure
$\underline{\delta}_x$	$:= \mathbf{return}(x)$	Dirac distribution
$\oint_X f(x) \underline{\mu}(\mathrm{d}x)$	$:= \underline{\mu} \gg\!= f$	Kock integral
$w \odot \underline{\mu}$	$:= \oint_X (w(x) \odot \underline{\delta}_x) \underline{\mu}(\mathrm{d}x)$	Rescaling

Synthetic measure theory: notation

Notation	Meaning	Terminology
R	$:= \mathrm{M}\,\mathbb{1}$	Scalars
$f_*\underline{\mu}$	$:= (\mathrm{M}\,f)(\underline{\mu})$	Push-forward
$\underline{\mu}(X)$	$:= !_*\underline{\mu}$	The total measure
$\underline{\delta}_x$	$:= \mathbf{return}(x)$	Dirac distribution
$\oint_X f(x)\underline{\mu}(\mathrm{d}x)$	$:= \underline{\mu} \gg= f$	Kock integral
$w \odot \underline{\mu}$	$:= \oint_X (w(x) \odot \underline{\delta}_x)\underline{\mu}(\mathrm{d}x)$	Rescaling
$\oint_Y f(x,y)k(x,\mathrm{d}y)$	$:= \oint_Y f(x,y)k(x)(\mathrm{d}y)$	Kernel integration

Synthetic measure theory: notation

Notation	Meaning	Terminology
R	$:= \mathrm{M}\,\mathbb{1}$	Scalars
$f_*\underline{\mu}$	$:= (\mathrm{M}\,f)(\underline{\mu})$	Push-forward
$\underline{\mu}(X)$	$:= !_*\underline{\mu}$	The total measure
$\underline{\delta}_x$	$:= \mathbf{return}(x)$	Dirac distribution
$\oint_X f(x)\underline{\mu}(\mathrm{d}x)$	$:= \underline{\mu} \gg= f$	Kock integral
$w \odot \underline{\mu}$	$:= \oint_X (w(x) \odot \underline{\delta}_x)\underline{\mu}(\mathrm{d}x)$	Rescaling
$\oint_Y f(x,y)k(x,\mathrm{d}y)$	$:= \oint_Y f(x,y)k(x)(\mathrm{d}y)$	Kernel integration
$\oiint_{X\times Y} f(x,y)\underline{\mu}(\mathrm{d}x,\mathrm{d}y)$	$:= \oint_{X\times Y} f(z)\underline{\mu}(\mathrm{d}z)$	Iterated integrals

Synthetic measure theory: notation

Notation	Meaning	Terminology
R	$:= \mathrm{M}\,\mathbb{1}$	Scalars
$f_*\underline{\mu}$	$:= (\mathrm{M}\,f)(\underline{\mu})$	Push-forward
$\underline{\mu}(X)$	$:= !_*\underline{\mu}$	The total measure
$\underline{\delta}_x$	$:= \mathbf{return}(x)$	Dirac distribution
$\oint_X f(x)\underline{\mu}(\mathrm{d}x)$	$:= \underline{\mu} \gg= f$	Kock integral
$w \odot \underline{\mu}$	$:= \oint_X (w(x) \odot \underline{\delta}_x)\underline{\mu}(\mathrm{d}x)$	Rescaling
$\oint_Y f(x,y)k(x,\mathrm{d}y)$	$:= \oint_Y f(x,y)k(x)(\mathrm{d}y)$	Kernel integration
$\oiint_{X\times Y} f(x,y)\underline{\mu}(\mathrm{d}x,\mathrm{d}y)$	$:= \oint_{X\times Y} f(z)\underline{\mu}(\mathrm{d}z)$	Iterated integrals
$\underline{\mu}\otimes\underline{\nu}$	$:= \oint_X \left(\oint_Y \underline{\delta}_{(x,y)}\underline{\nu}(\mathrm{d}y)\right)\underline{\mu}(\mathrm{d}x)$	Product measure

Synthetic measure theory: notation

Notation	Meaning	Terminology
R	$:= \mathrm{M}\,\mathbb{1}$	Scalars
$f_*\underline{\mu}$	$:= (\mathrm{M}\,f)(\underline{\mu})$	Push-forward
$\underline{\mu}(X)$	$:= !_*\underline{\mu}$	The total measure
$\underline{\delta}_x$	$:= \mathbf{return}(x)$	Dirac distribution
$\oint_X f(x)\underline{\mu}(\mathrm{d}x)$	$:= \underline{\mu} \gg\!= f$	Kock integral
$w \odot \underline{\mu}$	$:= \oint_X (w(x) \odot \underline{\delta}_x)\underline{\mu}(\mathrm{d}x)$	Rescaling
$\oint_Y f(x,y)k(x,\mathrm{d}y)$	$:= \oint_Y f(x,y)k(x)(\mathrm{d}y)$	Kernel integration
$\oint\!\!\oint_{X\times Y} f(x,y)\underline{\mu}(\mathrm{d}x,\mathrm{d}y)$	$:= \oint_{X\times Y} f(z)\underline{\mu}(\mathrm{d}z)$	Iterated integrals
$\underline{\mu} \otimes \underline{\nu}$	$:= \oint_X \left(\oint_Y \underline{\delta}_{(x,y)}\underline{\nu}(\mathrm{d}y)\right)\underline{\mu}(\mathrm{d}x)$	Product measure
$\mathbb{E}^A_{x\sim\underline{\mu}}[f(x)]$	$:= \underline{\mu} \gg\!= f$	Expectation

Synthetic measure theory: notation

Notation	Meaning	Terminology
R	$:= \mathrm{M}\,\mathbb{1}$	Scalars
$f_*\underline{\mu}$	$:= (\mathrm{M}\,f)(\underline{\mu})$	Push-forward
$\underline{\mu}(X)$	$:= !_*\underline{\mu}$	The total measure
$\underline{\delta}_x$	$:= \mathbf{return}(x)$	Dirac distribution
$\fint_X f(x)\underline{\mu}(\mathrm{d}x)$	$:= \underline{\mu} \gg\!= f$	Kock integral
$w \odot \underline{\mu}$	$:= \fint_X (w(x) \odot \underline{\delta}_x)\underline{\mu}(\mathrm{d}x)$	Rescaling
$\fint_Y f(x,y)k(x,\mathrm{d}y)$	$:= \fint_Y f(x,y)k(x)(\mathrm{d}y)$	Kernel integration
$\fint\!\!\fint_{X\times Y} f(x,y)\underline{\mu}(\mathrm{d}x,\mathrm{d}y)$	$:= \fint_{X\times Y} f(z)\underline{\mu}(\mathrm{d}z)$	Iterated integrals
$\underline{\mu} \otimes \underline{\nu}$	$:= \fint_X \left(\fint_Y \underline{\delta}_{(x,y)}\underline{\nu}(\mathrm{d}y)\right)\underline{\mu}(\mathrm{d}x)$	Product measure
$\mathbb{E}^A_{x\sim\underline{\mu}}[f(x)]$	$:= \underline{\mu} \gg\!= f$	Expectation
$\int_X f(x)\underline{\mu}(\mathrm{d}x)$	$:= \mathbb{E}^R_{x\sim\underline{\mu}}[f(x)]$	Lebesgue integral

Synthetic measure theory: Radon-Nikodym

Radon-Nikodym derivatives

- $\underline{\nu} \ll \underline{\mu}$ when $\underline{\nu} = w \odot \underline{\mu}$;
- w and v are **equal** $\underline{\mu}$**-almost everywhere** when $w \odot \underline{\mu} = v \odot \underline{\mu}$.
- Measurable property: $P : X \to$ bool, induces $[P] : X \to [0, \infty]$
- P over X **holds** $\underline{\mu}$**-a.e.** when $[P] = 1$ $\underline{\mu}$-a.e..

Theorem (Radon-Nikodym)

Let $(\mathcal{C}, \mathrm{M})$ *be a well-pointed measure category. For every* $\underline{\nu} \ll \underline{\mu}$ *in* $\mathrm{M}\,X$*, there exists a* $\underline{\mu}$*-a.e. unique morphism* $\frac{\mathrm{d}\underline{\nu}}{\mathrm{d}\underline{\mu}} : X \to R$ *satisfying* $\frac{\mathrm{d}\underline{\nu}}{\mathrm{d}\underline{\mu}} \odot \underline{\mu} = \underline{\nu}$.

Talk structure

- Probabilistic programming and Bayesian inference
- Synthetic measure theory
- Quasi-Borel spaces
- Inference representations
- Trace Markov Chain Monte Carlo (Trace MCMC)
- Conclusion

Brief measure theory

Measures subsets of $\mathbb{R}$

Borel subsets $\mathcal{B}(\mathbb{R})$ as closure under:

- Intervals $[a, b]$.
- Countable unions.
- Complements.

$\varphi : \mathbb{R} \to \mathbb{R}$ is **measurable** when:

$$B \in \mathcal{B}(\mathbb{R}) \qquad \Longrightarrow \qquad \varphi^{-1}[B] \in \mathcal{B}(\mathbb{R})$$

Source of randomness

Key idea

Propagating randomness from discrete and continuous sampling:

$$\alpha : \mathbb{I} \to X$$

along "random elements":

- for measurable spaces: **derived** through measurable functions;
- for quasi-Borel spaces: **axiomised** through structure.

The category $\mathbf{Qbs}$

Objects

A **quasi-Borel space** $X = \left(|X|, X^{\mathbb{I}}\right)$ consists of:

- a **carrier set** X;
- a set of **random elements** $X^{\mathbb{I}} \subseteq |X|^{\mathbb{I}}$

such that the random elements are closed under:

The category $\mathbf{Qbs}$

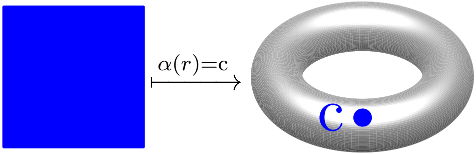
Objects

A **quasi-Borel space** $X = \left(|X|, X^{\mathbb{I}}\right)$ consists of:

- a **carrier set** X;
- a set of **random elements** $X^{\mathbb{I}} \subseteq |X|^{\mathbb{I}}$

such that the random elements are closed under:

- constant functions $\underline{c}$;



The category Qbs

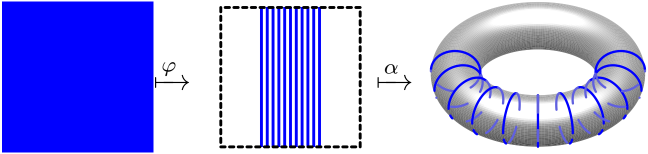
Objects

A **quasi-Borel space** $X = \left(|X|, X^{\mathbb{I}}\right)$ consists of:

- a **carrier set** X;
- a set of **random elements** $X^{\mathbb{I}} \subseteq |X|^{\mathbb{I}}$

such that the random elements are closed under:

- constant functions $\underline{c}$;
- precomposition with a measurable $\varphi : \mathbb{I} \to \mathbb{I}$



The category $\mathbf{Qbs}$

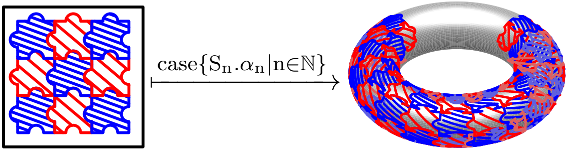
Objects

A **quasi-Borel space** $X = \left(|X|, X^{\mathbb{I}}\right)$ consists of:

- a **carrier set** X;
- a set of **random elements** $X^{\mathbb{I}} \subseteq |X|^{\mathbb{I}}$

such that the random elements are closed under:

- constant functions $\underline{c}$;
- precomposition with a measurable $\varphi : \mathbb{I} \to \mathbb{I}$
- countable measurable case split.



The category $\mathbf{Qbs}$

Morphisms $f : X \to Y$

Functions $f : |X| \to |Y|$ such that:

$$\alpha \in X^{\mathbb{R}} \qquad \Longrightarrow \qquad f \circ \alpha \in Y^{\mathbb{R}}$$

Subspaces

Every subset $S \subseteq |X|$ inherits the subspace structure:

$$S^{\mathbb{R}} := \left\{ \alpha : \mathbb{R} \to S \middle| \alpha \in X^{\mathbb{I}} \right\}$$

The commutative monad

Measures

(Ω, α, μ):

- Ω is a standard Borel space
- $\alpha \in X^{\Omega}$
- and μ is a σ-finite measure on Ω

Induced integration operator

For $f : X \to [0, \infty]$:

$$\int f \, \mathrm{d}(\Omega, \alpha, \mu) \coloneqq \int_{\Omega} f(\alpha(x)) \, \mu(\mathrm{d}x)$$

Monad of measures

$(\Omega, \alpha, \mu) \approx (\Omega', \alpha', \mu')$ when they determine the same integration operator.

$\mathrm{M}\,X$ consists of equivalence classes of $\approx$.

A synthetic model

The measure category $(\mathbf{Qbs}, \underline{\mathrm{M}})$

- $\mathbf{Qbs}(\mathbb{1}, R) \cong_\sigma [0, \infty]$;
- $\mathbf{Qbs}(R, \mathbb{1} + \mathbb{1}) \cong \mathcal{B}([0, \infty])$ as characteristic functions
- $\mathbf{Qbs}(R, R) \cong \mathbf{Meas}([0, \infty], [0, \infty])$
- Giry $[0, \infty] \rightarrowtail \mathbf{Qbs}(\mathbb{1}, \mathrm{M}(R)) \rightarrowtail$ Measures $[0, \infty]$
- $R^R \times \mathrm{M}(R) \to R$, $(f, \underline{\mu}) \mapsto \int f(x)\, \underline{\mu}(\mathrm{d}x)$ is the Lebesgue integral

Talk structure

- Probabilistic programming and Bayesian inference
- Synthetic measure theory
- Quasi-Borel spaces
- Inference representations
- Trace Markov Chain Monte Carlo (Trace MCMC)
- Conclusion

Representations

Program representation

A **representation** $\underline{T}$ $(T, \mathrm{return}^{\underline{T}}, \gg\!=^{\underline{T}}, m^{\underline{T}})$ consists of:

- $(T, \mathrm{return}^{\underline{T}}, \gg\!=^{\underline{T}})$: monadic interface;
- $m_X^{\underline{T}} : T\,X \to \mathrm{M}\,X$: meaning morphism for every space X

and $m^{\underline{T}}$ preserves $\mathrm{return}^{\underline{T}}$ and $\gg\!=^{\underline{T}}$:

$$\mathrm{return}^{\underline{\mathrm{M}}}\,x = m(\mathrm{return}^{\underline{T}}\,x)$$

$$m(a \gg\!=^{\underline{T}} f) = (m\,a) \gg\!=^{\underline{\mathrm{M}}} \lambda x.\ m(f\ x)$$

Representations

Example representation: lists

$$
\begin{array}{l}
\textbf{instance } \mathit{Rep}\,(\mathsf{List})\ \textbf{where} \\
\quad \textbf{return}\, x \qquad\qquad = [x] \\
\quad x_s \gg\!= f \qquad\qquad = \mathsf{foldr}\,[\ ] \\
\qquad\qquad\qquad\qquad\qquad (\lambda(x, y_s). \\
\qquad\qquad\qquad\qquad\qquad f(x) +\!\!+\, y_s)\ x_s \\
\quad m_{\mathsf{List}}[x_1, \ldots, x_n] = \sum_{i=1}^{n} \underline{\delta}_{x_i}
\end{array}
$$

Representations

Sampling representation

$(T, \mathrm{return}^{\underline{T}}, \gg\!=^{\underline{T}}, m^{\underline{T}}, \mathbf{sample}^{\underline{T}})$

- $(T, \mathrm{return}^{\underline{T}}, \gg\!=^{\underline{T}}, m^{\underline{T}})$: program representation
- $\mathbf{sample}^{\underline{T}} : \mathbb{1} \to T\,\mathbb{I}$

and $m^{\underline{T}} \circ \mathbf{sample}^{\underline{T}} = \mathbf{U}_{\mathbb{I}}$

Representations

Example: free sampler

$\mathsf{Sam}\,\alpha \coloneqq \{\mathsf{Return}\,\alpha \mid \mathsf{Sample}\,(\mathbb{I} \to \mathsf{Sam}\,\alpha)\}$:

$$
\begin{aligned}
&\textbf{instance}\ \textit{Sampling Rep}\,(\mathsf{Sam})\ \textbf{where}\\
&\quad \textbf{return}\,x = \mathsf{Return}\,x\\
&\quad a \gg\!= f \quad = \textbf{match}\,a\,\textbf{with}\,\{\\
&\qquad\qquad\qquad\qquad \mathsf{Return}\,x \to f(x)\\
&\qquad\qquad\qquad\qquad \mathsf{Sample}\,k \to\\
&\qquad\qquad\qquad\qquad\quad \mathsf{Sample}\,(\lambda r.\,k(r) \gg\!= f)\}\\
&\quad \mathrm{sample} \quad = \mathsf{Sample}\,\lambda r.\,(\mathsf{Return}\,r)\\
&\quad m\,a \quad = \textbf{match}\,a\,\textbf{with}\,\{\\
&\qquad\qquad\qquad\qquad \mathsf{Return}\,x \to \underline{\delta}_x\\
&\qquad\qquad\qquad\qquad \mathsf{Sample}\,k \to \oint_{\mathbb{I}} m(k(x))\mathbf{U}(\mathrm{d}x)\}
\end{aligned}
$$

Representations

Conditioning representation

$(T, \mathrm{return}^{\underline{T}}, \gg\!=^{\underline{T}}, m^{\underline{T}}, \mathrm{score}^{\underline{T}})$

- $(T, \mathrm{return}^{\underline{T}}, \gg\!=^{\underline{T}}, m^{\underline{T}})$: program representation
- $\mathrm{score}^{\underline{T}} : [0, \infty) \to T\,\mathbb{1}$

and $m^{\underline{T}} \circ \mathrm{score}^{\underline{T}}\, r = r \odot \underline{\delta}_{()}$

Representations

Weighted values

For every representation $\underline{T}$, $\mathsf{W}\,\underline{T}\,X := T(\mathbb{R}_+ * X)$

$$
\begin{aligned}
&\textbf{instance } \mathit{Conditioning\ Rep}\,(\mathsf{W}\,\underline{T})\ \textbf{where}\\
&\quad \textbf{return}_{\mathsf{W}\,\underline{T}}\, x = \mathrm{return}^{\underline{T}}(1, x)\\
&\quad a \gg\!=_{\mathsf{W}\,\underline{T}} f \quad = \underline{T}.\textbf{do}\,\{(r, x) \leftarrow a;\\
&\qquad\qquad\qquad\qquad (s, y) \leftarrow f(x);\\
&\qquad\qquad\qquad\qquad \textbf{return}(r \cdot s, y)\}\\
&\quad m_{\mathsf{W}\,\underline{T}}\, a \quad = \lambda x.\ \oint_{\mathbb{R}_+ \times X} r \odot \underline{\delta}_x m^{\underline{T}}(a)(\mathrm{d}r, \mathrm{d}x)\\
&\quad \textbf{score}_{\mathsf{W}\,\underline{T}}\, r \quad = \mathrm{return}^{\underline{T}}(r, ())
\end{aligned}
$$

Representations

Inference representation

$(T, \mathrm{return}^{\underline{T}}, \gg\!=^{\underline{T}}, \mathbf{sample}^{\underline{T}}\, \mathrm{score}^{\underline{T}}, m^{\underline{T}})$: sampling and conditioning

Example: weighted sampler

$\mathsf{WSam}\, X := \mathsf{W}\, \mathsf{Sam}\, X = \mathsf{Sam}([0, \infty) \times X)$

Inference transformations

$\underline{t} : \underline{T} \to \underline{S}$
$\underline{t} : T\,X \to S\,X$ for every space X such that:

$$m_{\underline{S}} \circ \underline{t} = m_{\underline{T}}$$

A single compositional step in an inference algorithm

Inference transformations

$\underline{t} : \underline{T} \to \underline{S}$

$\underline{t} : T\,X \to S\,X$ for every space X such that:

$$m_{\underline{S}} \circ \underline{t} = m_{\underline{T}}$$

A single compositional step in an inference algorithm

Unnaturality

$\mathrm{aggr}_X : \mathsf{List}(\mathbb{R}_+ * X) \to \mathsf{List}(\mathbb{R}_+ * X)$

aggregating (r, x), (s, x) to $(r + s, x)$

Then $\mathrm{aggr} : \underline{\mathsf{List}} \to \underline{\mathsf{List}}$ but not natural:

$$\mathrm{aggr} \circ \mathsf{List}!\ [(\tfrac{1}{2}, \mathsf{False}), (\tfrac{1}{2}, \mathsf{True})][(1, ())] \\ \neq [(\tfrac{1}{2}, ()), (\tfrac{1}{2}, ())]\ \mathsf{Enum}! \circ \mathrm{aggr}\ [(\tfrac{1}{2}, \mathsf{False}), (\tfrac{1}{2}, \mathsf{True})]$$

Talk structure

- Probabilistic programming and Bayesian inference
- Synthetic measure theory
- Quasi-Borel spaces
- Inference representations
- Trace Markov Chain Monte Carlo (Trace MCMC)
- Conclusion

Markov Chain Monte Carlo

Metropolis-Hastings update

$$\begin{aligned}\underline{T}.\mathbf{do}\,\{&x \leftarrow a;\\ &y \leftarrow \psi_a(x);\\ &r \leftarrow \text{sample};\\ &\mathbf{if}\ r < \min(1, \rho_a(x,y))\\ &\mathbf{then\ return}\, y\\ &\mathbf{else\ return}\, x\}\end{aligned}$$

where $\psi_a : X \to T\,X$ and $\rho_a : X \times X \to \overline{\mathbb{R}}_+$

Markov Chain Monte Carlo: abstract foundation

Theorem (Metropolis-Hastings-Green for quasi-Borel spaces)

Given X, $a \in \mathrm{M}\,X$, $\psi_a : X \to \mathrm{M}\,X$, *and* $\rho_a : X \times X \to \overline{\mathbb{R}}_+$, *set* $\underline{\mu}_a \coloneqq [\rho \neq 0] \odot (\iiint \underline{\delta}_{(x,y)} a(\mathrm{d}x)\psi(x, \mathrm{d}y))$.
Assume that:

1. ψ_a *is Markov:* $\psi(x, X) = 1$;
2. $[1 = (\rho \circ \mathsf{swap}) \cdot \rho]$ *holds* $\underline{\mu}_a$*-a.e.;*
3. $\rho = \frac{\mathrm{d}(\mathsf{swap}_* \underline{\mu}_a)}{\mathrm{d}\underline{\mu}_a}$;
4. $\rho(x, y) = 0 \iff \rho(y, x) = 0$ *for all* $x, y \in X$.

Then $(\eta_{\psi_a,\rho_a})(a) = a$.

Proof mimicks measure theoretic proof, e.g. [Geyer'11]

Trace Markov Chain Monte Carlo: Representation

Program traces

- $t \in \mathsf{WSam}\, X$: program structure representation

Trace Markov Chain Monte Carlo: Representation

Program traces

- $t \in \mathsf{WSam}\,X$: program structure representation
- $p : \mathsf{List}\,\mathbb{I}$ a trace in program t

$$
\begin{aligned}
p \in t = \mathbf{match}\,(p, t)\,\mathbf{with}\,\{ \\
&([\,] \quad , \mathsf{Return}\,x \quad) \to \mathsf{True} \\
&(r :: r_s, \mathsf{Sample}\,f \quad) \to [r_s \in f(r)] \\
&-- \quad \text{any other case:} \\
&(\_ \quad , \_ \quad) \to \mathsf{False}\}
\end{aligned}
$$

Trace Markov Chain Monte Carlo: Representation

Program traces

- $t \in \mathsf{WSam}\, X$: program structure representation
- $p : \mathsf{List}\, \mathbb{I}$ a trace in program t
- $$\sum_{t \in \mathsf{WSam}\, X} \mathrm{Paths}\, t := \left\{(t, p) \in \mathsf{WSam}\, X \times \mathsf{List}\, \mathbb{I} \middle| p \in t\right\}$$
  $$\subseteq \mathsf{WSam}\, X \times \mathsf{List}\, \mathbb{I}$$

Trace Markov Chain Monte Carlo: Representation

Program traces

- $t \in \mathsf{WSam}\, X$: program structure representation
- $p : \mathsf{List}\, \mathbb{I}$ a trace in program t
- $$\sum_{t \in \mathsf{WSam}\, X} \mathrm{Paths}\, t := \left\{ (t, p) \in \mathsf{WSam}\, X \times \mathsf{List}\, \mathbb{I} \middle| p \in t \right\}$$
  $$\subseteq \mathsf{WSam}\, X \times \mathsf{List}\, \mathbb{I}$$
- $w_{-} : \sum_{t \in \mathsf{WSam}\, X} \mathrm{Paths}\, t \to \mathbb{R}_{+}$
  $w_{\mathsf{Return}\,(r,x)}([\,]) = r$
  $w_{\mathsf{Sample}\, t_{-}}(s :: r_s) = w_{t_s}(r_s)$

Trace Markov Chain Monte Carlo: Representation

Program traces

- $t \in \mathsf{WSam}\,X$: program structure representation
- $p : \mathsf{List}\,\mathbb{I}$ a trace in program t
- $$\sum_{t \in \mathsf{WSam}\,X} \mathrm{Paths}\,t := \left\{(t, p) \in \mathsf{WSam}\,X \times \mathsf{List}\,\mathbb{I} \middle| p \in t\right\}$$
  $$\subseteq \mathsf{WSam}\,X \times \mathsf{List}\,\mathbb{I}$$
- $w_{-} : \sum_{t \in \mathsf{WSam}\,X} \mathrm{Paths}\,t \to \mathbb{R}_{+}$
- $v_{-} : \sum_{t \in \mathsf{WSam}\,X} \mathrm{Paths}\,t \to X$
  $v_{\mathsf{Return}\,(r,x)}([\,]) = x$
  $v_{\mathsf{Sample}\,t_{-}}(s :: r_s) = v_{t_s}(r_s)$

Trace Markov Chain Monte Carlo: Representation

Program traces

- $t \in \mathsf{WSam}\, X$: program structure representation
- $p : \mathsf{List}\, \mathbb{I}$ a trace in program t
- $$\sum_{t \in \mathsf{WSam}\, X} \mathrm{Paths}\, t := \left\{ (t, p) \in \mathsf{WSam}\, X \times \mathsf{List}\, \mathbb{I} \middle| p \in t \right\}$$
  $$\subseteq \mathsf{WSam}\, X \times \mathsf{List}\, \mathbb{I}$$
- $w_{-} : \sum_{t \in \mathsf{WSam}\, X} \mathrm{Paths}\, t \to \mathbb{R}_{+}$
- $v_{-} : \sum_{t \in \mathsf{WSam}\, X} \mathrm{Paths}\, t \to X$

Tracing representation

$$\mathrm{Tr}\, \underline{T}\ X :=$$

$$\left\{ (t, a) \in \mathsf{WSam}\, X \times T(\mathsf{List}\, \mathbb{I}) \middle| \begin{array}{l} [\ \in t]\ m_{\underline{T}}(a)\text{-a.e., and} \\ m_{\mathsf{WSam}}(t) = \fint_{\mathsf{List}\, \mathbb{I}} \underline{\delta}_{v_t(p)}\, m_{\underline{T}}(a)(\mathrm{d}p) \end{array} \right\}$$

Trace Markov Chain Monte Carlo: Representation

Tracing representation

$$\mathrm{Tr}\,\underline{T}\ X :=$$

$$\left\{ (t,a) \in \mathsf{WSam}\,X \times T(\mathsf{List}\,\mathbb{I}) \;\middle|\; \begin{array}{l} [\,\in t]\ m_{\underline{T}}(a)\text{-a.e., and} \\ m_{\mathsf{WSam}}(t) = \oint_{\mathsf{List}\,\mathbb{I}} \underline{\delta}_{v_t(p)}\, m_{\underline{T}}(a)(\mathrm{d}p) \end{array} \right\}$$

$$\begin{array}{ll}
\multicolumn{2}{l}{\textbf{instance } \mathit{Inf} \implies \mathit{Inf}\ \ \mathit{Monad}\,(\mathrm{Tr}\,\underline{T})\ \textbf{where}} \\
\quad \textbf{return}\,x & = (\mathrm{return}_{\mathsf{WSam}}\,x, \mathrm{return}_{\underline{T}}[\ ]) \\
\quad (t,a) \gg\!= (f,g) & = (t \gg\!=_{\mathsf{WSam}} f, \underline{T}.\textbf{do}\,\{p \leftarrow a; \\
 & \qquad\qquad\qquad\qquad q \leftarrow g \circ v_t(p); \\
 & \qquad\qquad\qquad\qquad \textbf{return}(p +\!\!+ q))\}) \\
\quad m((,t),a) & = m_{\mathsf{WSam}}(t) = \oint_{\mathsf{List}\,\mathbb{I}} \underline{\delta}_{v_t(p)}\, m_{\underline{T}}(a)(\mathrm{d}p) \\
\quad \mathrm{sample} & = (\textbf{sample}_{\mathsf{WSam}}, \\
 & \quad \underline{T}.\textbf{do}\,\{r \leftarrow \mathrm{sample}; \textbf{return}[r]\}) \\
\quad \textbf{score}\,r & = (\textbf{sample}_{\mathsf{WSam}}, \\
 & \quad \underline{T}.\textbf{do}\,\{\textbf{score}\,r; \textbf{return}[\ ]\})
\end{array}$$

Markov Chain Monte Carlo: Transformation

Trace MCMC morphism

$$\eta^{\mathrm{Tr}\,T}_{\psi,\rho} : \mathrm{Tr}\,T\,X \to \mathrm{Tr}\,T\,X$$
$$\eta^{\mathrm{Tr}\,T}_{\psi,\rho}(t,a) := \big(t, \eta_{\psi_t,\rho_t}(a)\big)$$

Concrete proposal kernel and derivative

$$\psi_t : \mathsf{List}(\mathbb{I}) \to T(\mathsf{List}(\mathbb{I}))$$
$$\begin{aligned}\psi_t(p) := \underline{T}.\mathbf{do}\,\{&i \leftarrow \mathrm{U_D}^{\underline{T}}(|p|)\\ &q \leftarrow \mathrm{pri}^{T}(\mathrm{sub}(t, \mathrm{take}(i,p)))\\ &\mathbf{return}(\mathrm{take}(i,p) + q)\}\end{aligned}$$

$$\rho_t : \mathsf{List}(\mathbb{I}) \times \mathsf{List}(\mathbb{I}) \to \overline{\mathbb{R}}_+$$
$$\rho_t(p,q) := \frac{w_t(q)\cdot(|p|+1)}{w_t(p)\cdot(|q|+1)}$$

Contribution

Correctness of inference

- Modular validation of inference algorithms:
  Sequential Monte Carlo, Trace Markov Chain Monte Carlo
  By combining:
- Synthetic measure theory [Kock'12]: measure theory without measurable spaces
- Quasi-Borel spaces: a convenient category for higher-order measure theory [LICS'17]

Conclusion

Summary

- Bayesian inference: (continuous) sampling and conditioning
- Inference representation: monadic interface, sampling, conditioning, and meaning
- Plenty of opportunities for traditional programming language expertise

Further topics

- Sequential Monte Carlo (SMC)
- Combining SMC and MCMC into Move-Resample SMC
- Categorical structure of quasi-borel spaces