Denotational validation of Bayesian inference

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

Mobility Reading Group Seminar
Imperial College London
11 April 2018

What is statistical probabilistic programming?

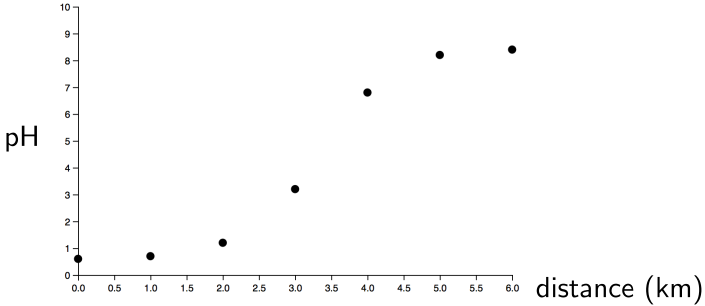
Bayesian data modelling

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 statistical probabilistic programming?

Example

Acidity in soil



What is statistical probabilistic programming?

Generative model

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

What is statistical probabilistic programming?

Generative model

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

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 statistical probabilistic programming?

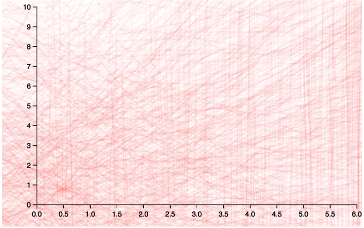
Bayesian inference

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

What is statistical probabilistic programming?

Bayesian inference

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

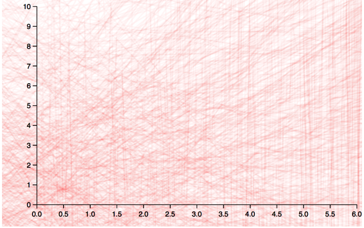


Prior

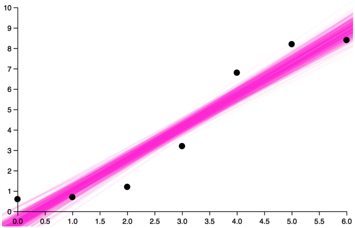
What is statistical probabilistic programming?

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 statistical probabilistic programming?

Probabilistic programming models

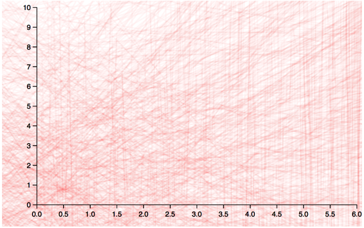
1. Develop a probabilistic (generative) model.
   Write a program.
2. ~~Design an inference algorithm for the model.~~
3. Using the built-in algorithm, fit the model to the data.

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))
```



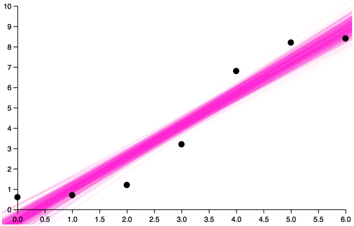
What is probabilistic programming?

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))
```

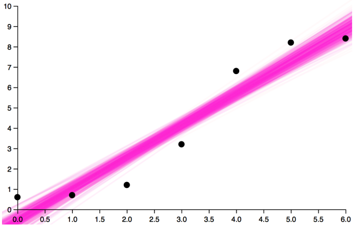


What is probabilistic programming?

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))
```

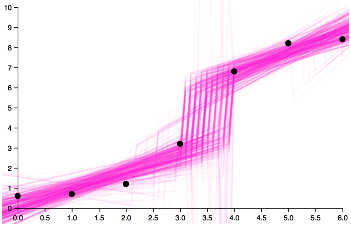


What is probabilistic programming?

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))
```

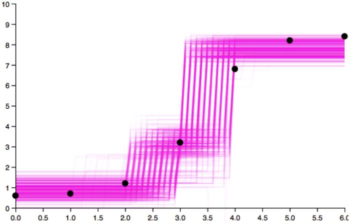


What is probabilistic programming?

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?

High-level analogy

graph algorithms + graph library + graph manipulating program

inference algorithms + inference library + probabilistic program

What is probabilistic programming?

Components

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

What is probabilistic programming?

Components

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

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

What is probabilistic programming?

Components

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

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

Which we refine to:

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

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)

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

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

and

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

A probabilistic program is a measure

For $t : X$

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

where $\mathsf{prior}\,[\![t]\!]$ 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 $t : X$

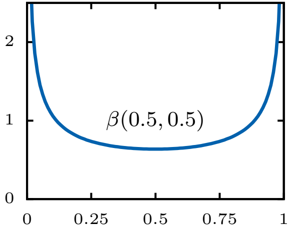
$$[\![t]\!] = w \odot \text{prior}\,[\![t]\!]$$

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

Note

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

- 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?

Inference engine

$$\boxed{\text{Model}} \mapsto \boxed{\text{IR}_1} \xmapsto{t_1} \boxed{\text{IR}_2} \xmapsto{t_2} \cdots \xmapsto{t_{n-1}} \boxed{\text{IR}_n} \mapsto \boxed{\text{Sampler}}$$

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**

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
- Ongoing work
- 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.:

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

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

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

- $0 : \mathbb{1} \to \mathrm{M}\,\mathbb{0}$
- $\sum_{n \in \mathbb{N}} X : \prod_{i \in \mathbb{N}} MX \cong \mathrm{M}(\coprod_{i \in \mathbb{N}} X) \xrightarrow{\mathrm{M}\,\nabla} \mathrm{M}\,X$
- $R := \mathrm{M}\,\mathbb{1}$ 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
$\oint_X f(x)\underline{\mu}(\mathrm{d}x)$	$:= \underline{\mu} \gg= f$	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
$\int\!\!\!\!\square_X f(x)\underline{\mu}(\mathrm{d}x)$	$:= \underline{\mu} \gg\!= f$	Kock integral
$w \odot \underline{\mu}$	$:= \int\!\!\!\!\square_X (w(x) \odot \underline{\delta}_x)\underline{\mu}(\mathrm{d}x)$	Rescaling
$\int\!\!\!\!\square_Y f(x,y)k(x,\mathrm{d}y)$	$:= \int\!\!\!\!\square_Y f(x,y)k(x)(\mathrm{d}y)$	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
$\oint\!\!\oint_{X\times Y} f(x,y)\underline{\mu}(\mathrm{d}x,\mathrm{d}y)$	$:= \oint_{X\times Y} f(z)\underline{\mu}(\mathrm{d}z)$	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

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
$\oint_X f(x)\underline{\mu}(\mathrm{d}x)$	$:= \underline{\mu} \gg= f$	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
$\mathbb{E}^A_{x\sim\underline{\mu}}[f(x)]$	$:= \underline{\mu} \gg= f$	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: $P : X \to \text{bool}$, 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
- Ongoing work
- Conclusion

Brief measure theory

Measurables 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:

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

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** $X = \left(|X|, M_X\right)$ consists of:

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

such that the random elements are closed under:

The category $\mathbf{Qbs}$

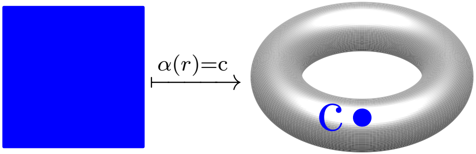
Objects

A **quasi-Borel space** $X = \left(|X|, M_X\right)$ consists of:

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

such that the random elements are closed under:

- constant functions $\underline{c}$;



The category $\mathbf{Qbs}$

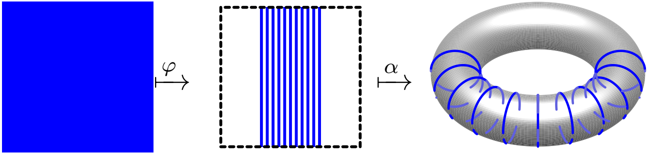
Objects

A **quasi-Borel space** $X = (|X|, M_X)$ consists of:

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

such that the random elements are closed under:

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



The category Qbs

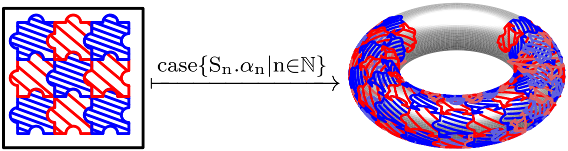
Objects

A **quasi-Borel space** $X = (|X|, M_X)$ consists of:

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

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:

$$\alpha \in M_X \qquad \Longrightarrow \qquad f \circ \alpha \in M_Y$$

Categorical structure

Measurable spaces

Adjunction with measurable spaces ($M \in \mathcal{C}Meas$, $X \in \mathbf{Qbs}$):

$$
\begin{aligned}
M_{\mathbf{Qbs}M} &:= \mathcal{C}Meas(\mathbb{R}, M) \\
\Sigma_{(\mathcal{C}Meas X)} &:= \left\{ B \subseteq X \,\middle|\, \forall \alpha \in M_X, \alpha^{-1}[X] \in \mathcal{B}(\mathbb{R}) \right\}
\end{aligned}
$$

$$
\begin{tikzcd}
\mathbf{Qbs} \arrow[rr, bend left, "\mathcal{C}Meas"] \arrow[rr, phantom, "\perp"] \arrow[dr] & & \mathcal{C}Meas \arrow[ll, bend left, "\mathbf{Qbs}"] \arrow[dl] \\
& \mathbf{Set} \arrow[u, phantom, "="] &
\end{tikzcd}
$$

NB: $\mathcal{C}Meas \circ \mathbf{Qbs}X = X$ for standard Borel spaces X.

Free and cofree spaces

Equip a set $A \in \mathbf{Set}$ with:

$$M_{\mathsf{Free}A} := \left\{ \mathsf{case}\, \{S_n.\underline{a}_n | n \in \mathbb{N}\} \middle| (S_n) \text{ a measurable partition} \right\}$$
$$M_{\mathsf{Cofree}A} := A^{\mathbb{R}}$$

$$\begin{tikzcd} \mathbf{Set} \arrow[rr, bend left, "F"] \arrow[rr, bend right, "C"'] & & \mathbf{Qbs} \arrow[ll] \end{tikzcd}$$

Categorical structure

Products

Correlated random elements:

$$M_{X \times Y} := \left\{ r \mapsto \big(\alpha(r), \beta(r)\big) \,\middle|\, \alpha \in M_X, \beta \in M_Y \right\}$$

Function spaces

$$\left|Y^X\right| := \mathbf{Qbs}(X, Y)$$

$$M_{Y^X} := \left\{ f : \mathbb{R} \to \left|Y^X\right| \,\middle|\, \text{uncurry } f \in \mathbf{Qbs}(\mathbb{R} \times X, Y) \right\}$$

NB: $X^{\mathbb{R}} = M_X$

Categorical structure

Subspaces

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

$$M_S := \{\alpha : \mathbb{R} \to S | \alpha \in M_X\}$$

equiv. a strong sub-object.

Categorical structure

Subspaces

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

$$M_S := \{\alpha : \mathbb{R} \to S | \alpha \in M_X\}$$

equiv. a strong sub-object.

More structure

Coproducts, limits, colimits, Grothendieck quasi-topos, locally presentable, ...

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

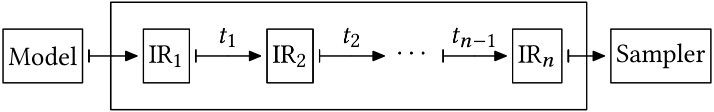
The measure category $(\mathbf{Qbs}, \underline{\mathrm{M}})$

- $\mathbf{Qbs}(\mathbb{1}, R) \cong_\sigma [0, \infty]$;
- $\mathbf{Qbs}(R, \mathbb{1} + \mathbb{1}) \cong \mathcal{B}([0, \infty])$ 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
- Ongoing work
- Conclusion

Representations



Program representation

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

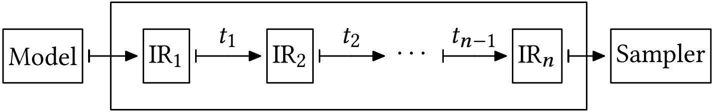
- $(T, \mathrm{return}^{\underline{T}}, \gg\!=^{\underline{T}})$: monadic interface;
- $m^{\underline{T}}_X : T\,X \to \mathrm{M}\,X$: meaning morphism for every space X

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

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

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

Representations



Example representation: lists

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

Representations

Example representation: lists

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

$$m_{\mathsf{List}}[x] = \delta_x$$

Representations

Example representation: lists

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

$$
m_{\mathsf{List}}\left([x_1, \ldots, x_n] \gg\!=^{\mathsf{List}} f\right) = m\left(f(x_1) +\!\!+ \cdots +\!\!+ f(x_n)\right)
$$

$$
= \sum_{i=1}^{n} m\, f(x_i) = \sum_{i=1}^{n} \int m_{\mathsf{List}} \circ f(y)\delta_{x_i}(\mathrm{d}y) = \int m \circ f(y) \sum_{i=1}^{n} \delta_{x_i}(\mathrm{d}y)
$$

$$
= \int m \circ f(y)\, m[x_1, \ldots, x_n](\mathrm{d}y) = m[x_1, \ldots, x_n] \gg\!=^{\mathrm{M}} (m \circ f)
$$

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

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

Example: free sampler

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

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

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:

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

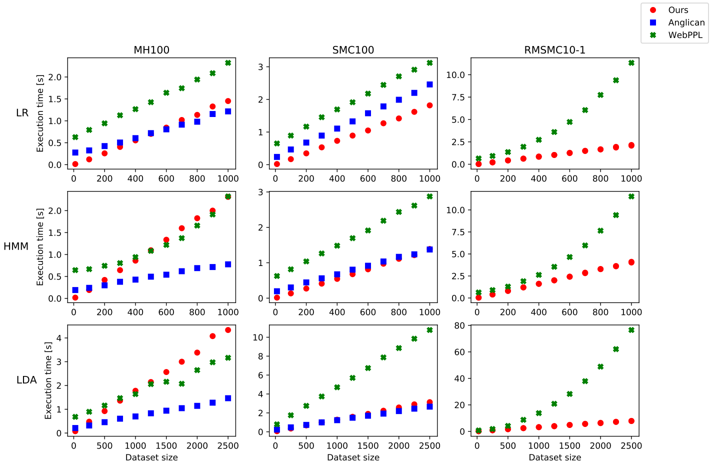
A single compositional step in an inference algorithm

Unnaturality

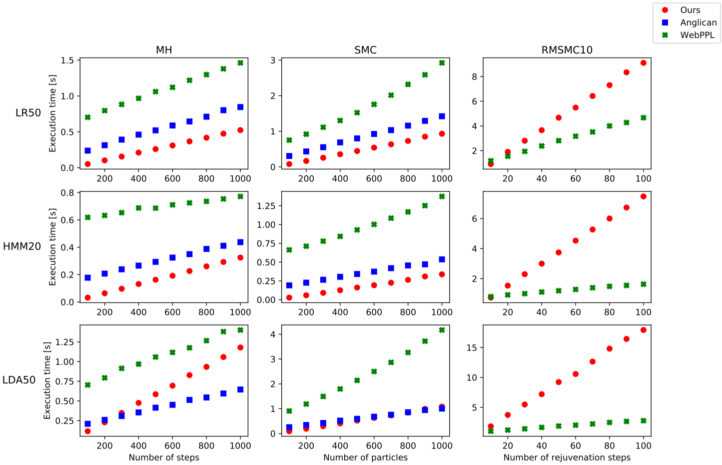
$\mathrm{aggr}_X : \mathsf{List}(\mathbb{R}_+ * X) \to \mathsf{List}(\mathbb{R}_+ * X)$
aggregating (r, x), (s, x) to $(r + s, x)$
Then $\mathrm{aggr} : \underline{\mathsf{List}} \to \underline{\mathsf{List}}$ but not natural:

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

Performance evaluation (1)



Performance evaluation (2)



Talk structure

- Probabilistic programming and Bayesian inference
- Synthetic measure theory
- Quasi-Borel spaces
- Inference representations
- Ongoing work
- Conclusion

Ongoing work: term and type recursion

ω-quasi-Borel spaces

$P = \big(|P|, \leq_P, M_P\big)$:

- $(P, \leq_P)$ is an ω-cpo;
- (P, M_P) is a qbs; and
- M_P is pointwise ω-chain closed.

and Scott-continuous qbs-morphisms

Axiomatic domain theory [Fiore'94]

Model of Fiore's axiomatic domain theory, with admissible maps $f : P \rightarrowtail Q$ are Scott-open and **Borel open**:

$$f[P] \in \Sigma_Q = \left\{ S \subseteq |Q| \middle| \forall \alpha \in M_Q . \alpha^{-1}[S] \in \mathcal{B} \right\}$$

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)
- Markov Chain Monte Carlo (MCMC) and Metropolis-Hastings-Green Theorem for $\mathbf{Qbs}$
- Combining SMC and MCMC into Move-Resample SMC