Semantics of Higher-Order Probabilistic Programs with Continuous Distributions

Hongseok Yang
University of Oxford

Based on work with or by Chris Heunen, Ohad Kammar, Sam Staton, and Frank Wood

Learning outcome

- Can explain what one can do with higher-order probabilistic programming language.
- Can use measure theory to interpret prob. prog. with continuous distributions.
- Can use quasi-Borel space to interpret higher-order prob. prog. with conditioning.

References

1. A convenient category for higher-order probability theory. Heunen et a. LICS'17.
2. Commutative semantics for probabilistic programs. Staton. ESOP'17.

What is probabilistic programming?

(Bayesian) probabilistic modelling of data

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

(Bayesian) probabilistic modelling of data in a prob. prog. language

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

(Bayesian) probabilistic modelling of data

in a prob. prog. language

as a program

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

(Bayesian) probabilistic modelling of data

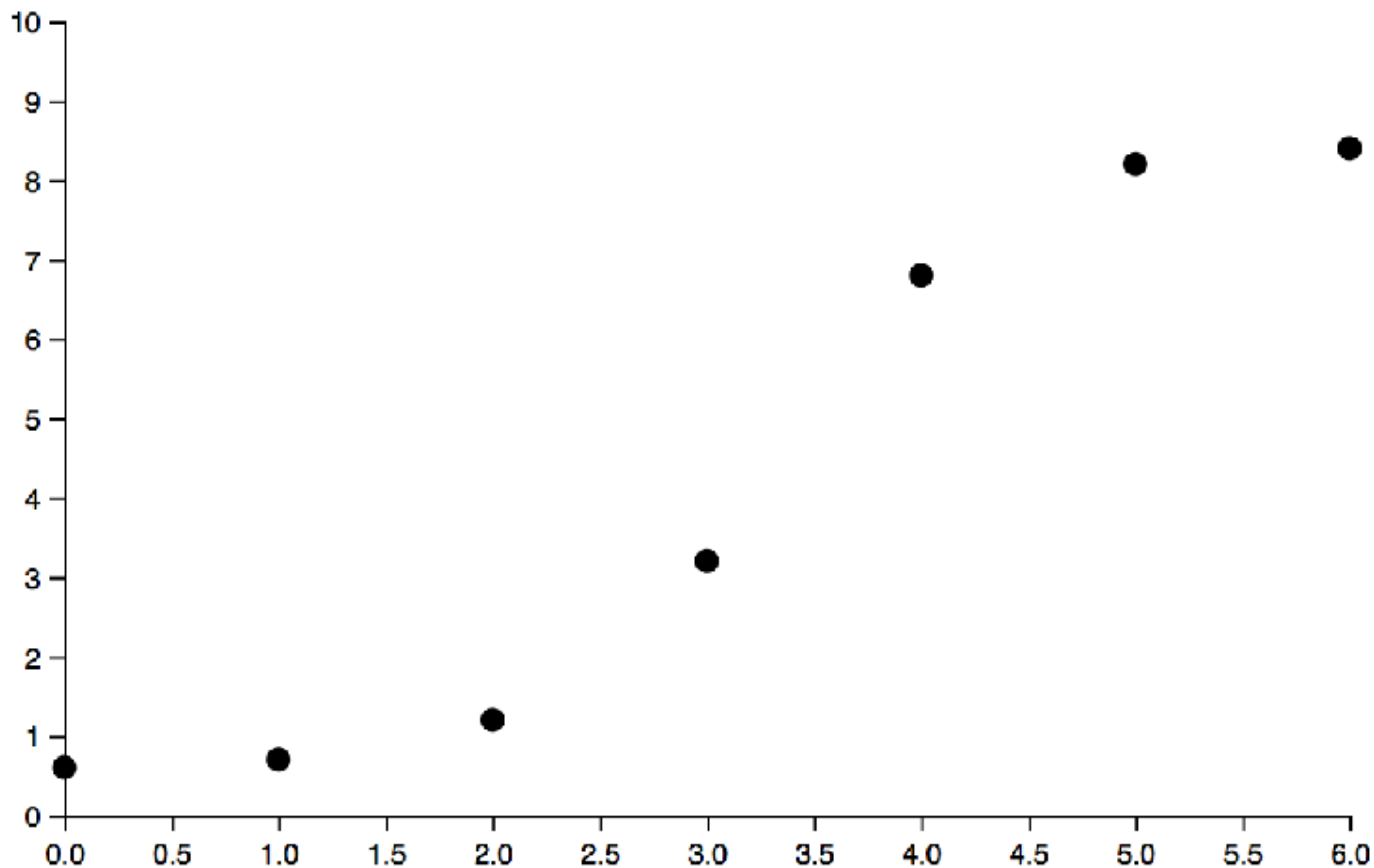
in a prob. prog. language

as a program

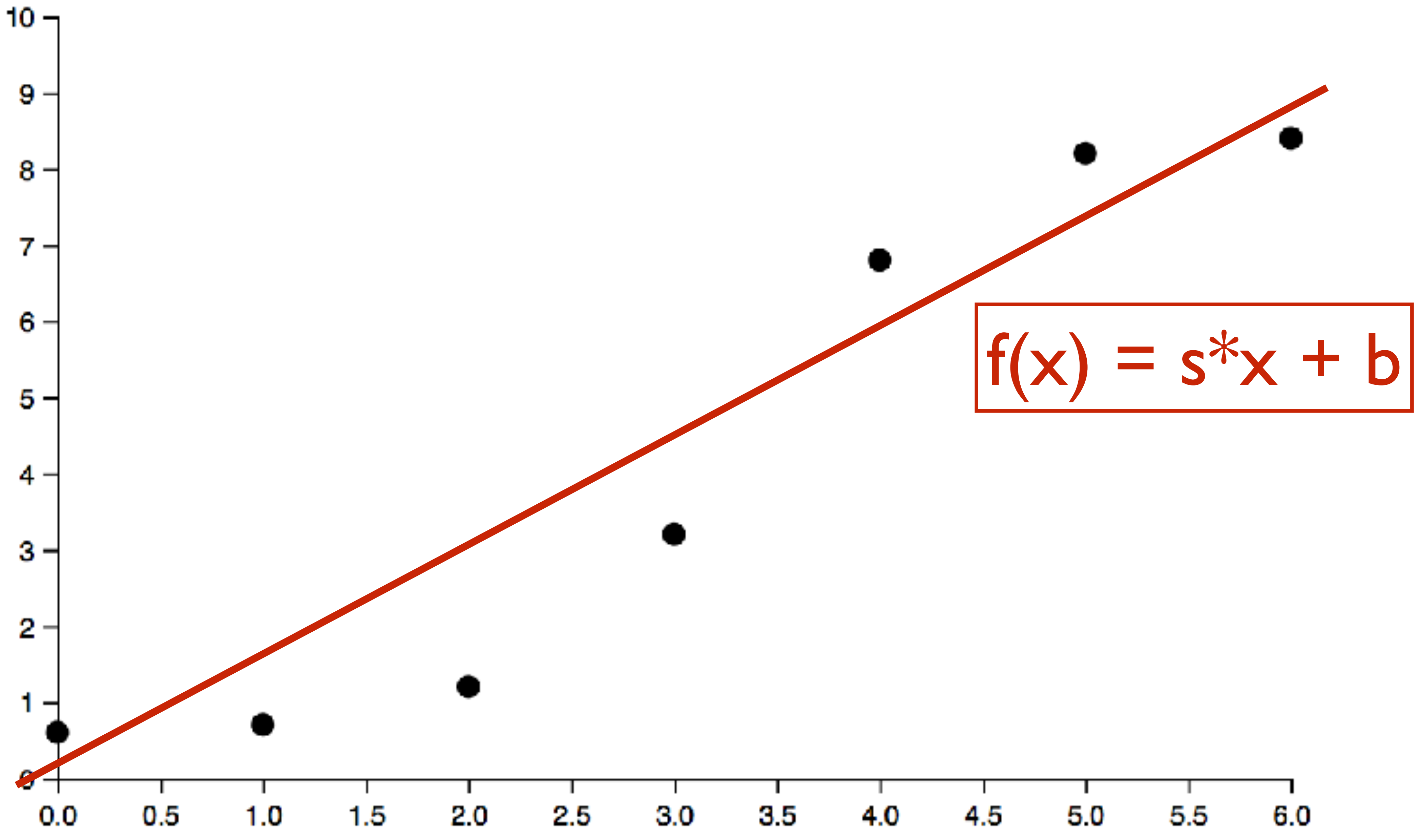
1. Develop a new probabilistic (generative) model.
2. ~~Design an inference algorithm for the model.~~
3. Using ~~the algo.~~, fit the model to the data.

a generic inference algo. of the language

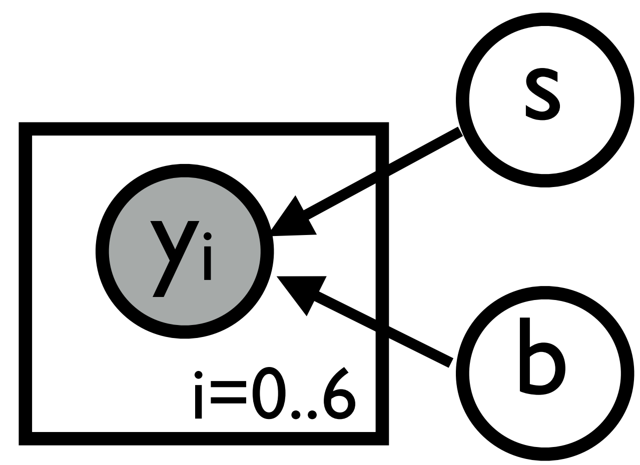
Line fitting



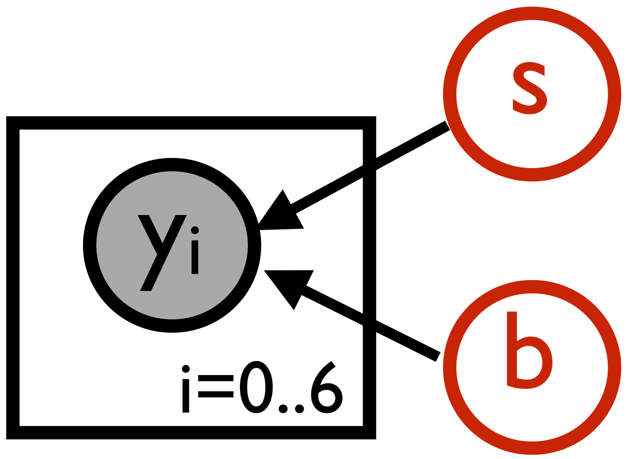
Line fitting



Bayesian generative model

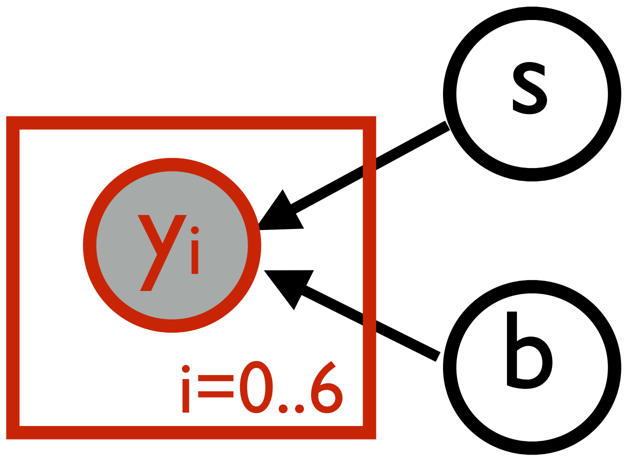


Bayesian generative model



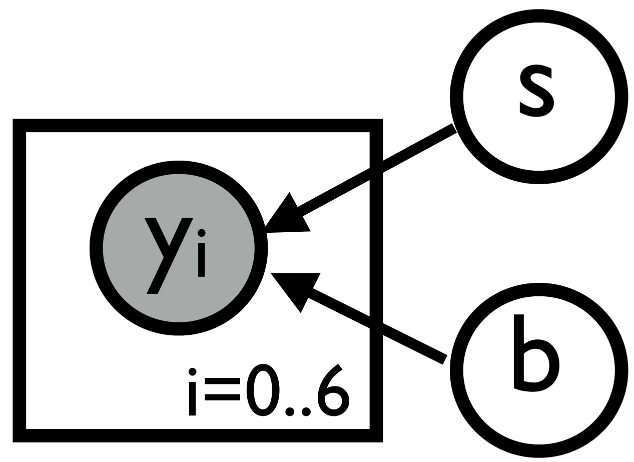

```
s      ~ normal(0, 2)
b      ~ normal(0, 6)
```

Bayesian generative model




```
s     ~ normal(0, 2)
b     ~ normal(0, 6)
f(x)  = s*x + b
yi     ~ normal(f(i), 0.5)
              where i = 0 .. 6
```

Bayesian generative model




```
s     ~ normal(0, 2)
b     ~ normal(0, 6)
f(x)  = s*x + b
yi    ~ normal(f(i), 0.5)
          where i = 0 .. 6
```

Q: posterior of (s,b) given y_0=0.6, …, y_6=8.4?

Posterior of s and b given y_i's

$$P(s, b \mid y_0, .., y_6) = \frac{P(y_0, .., y_6 \mid s,b) \times P(s,b)}{P(y_0, .., y_6)}$$

Posterior of s and b given y_i's

$$P(s, b \mid y_0, .., y_6) = \frac{P(y_0, .., y_6 \mid s,b) \times P(s,b)}{P(y_0, .., y_6)}$$

Posterior of s and b given y_i's

$$P(s, b \mid y_0, .., y_6) = \frac{P(y_0, .., y_6 \mid s,b) \times P(s,b)}{P(y_0, .., y_6)}$$

Posterior of s and b given y_i's

$$P(s, b \mid y_0, .., y_6) = \frac{P(y_0, .., y_6 \mid s,b) \times P(s,b)}{P(y_0, .., y_6)}$$

Posterior of s and b given y_i's

$$P(s, b \mid y_0, .., y_6) = \frac{P(y_0, .., y_6 \mid s,b) \times P(s,b)}{P(y_0, .., y_6)}$$

Posterior of s and b given y_i's

$$P(s, b \mid y_0, .., y_6) = \frac{P(y_0, .., y_6 \mid s,b) \times P(s,b)}{P(y_0, .., y_6)}$$

(almost) Anglican program

```
(let [s (sample (normal 0 2))
      b (sample (normal 0 6))
      f (fn [x] (+ (* s x) b))]
```

(almost) Anglican program

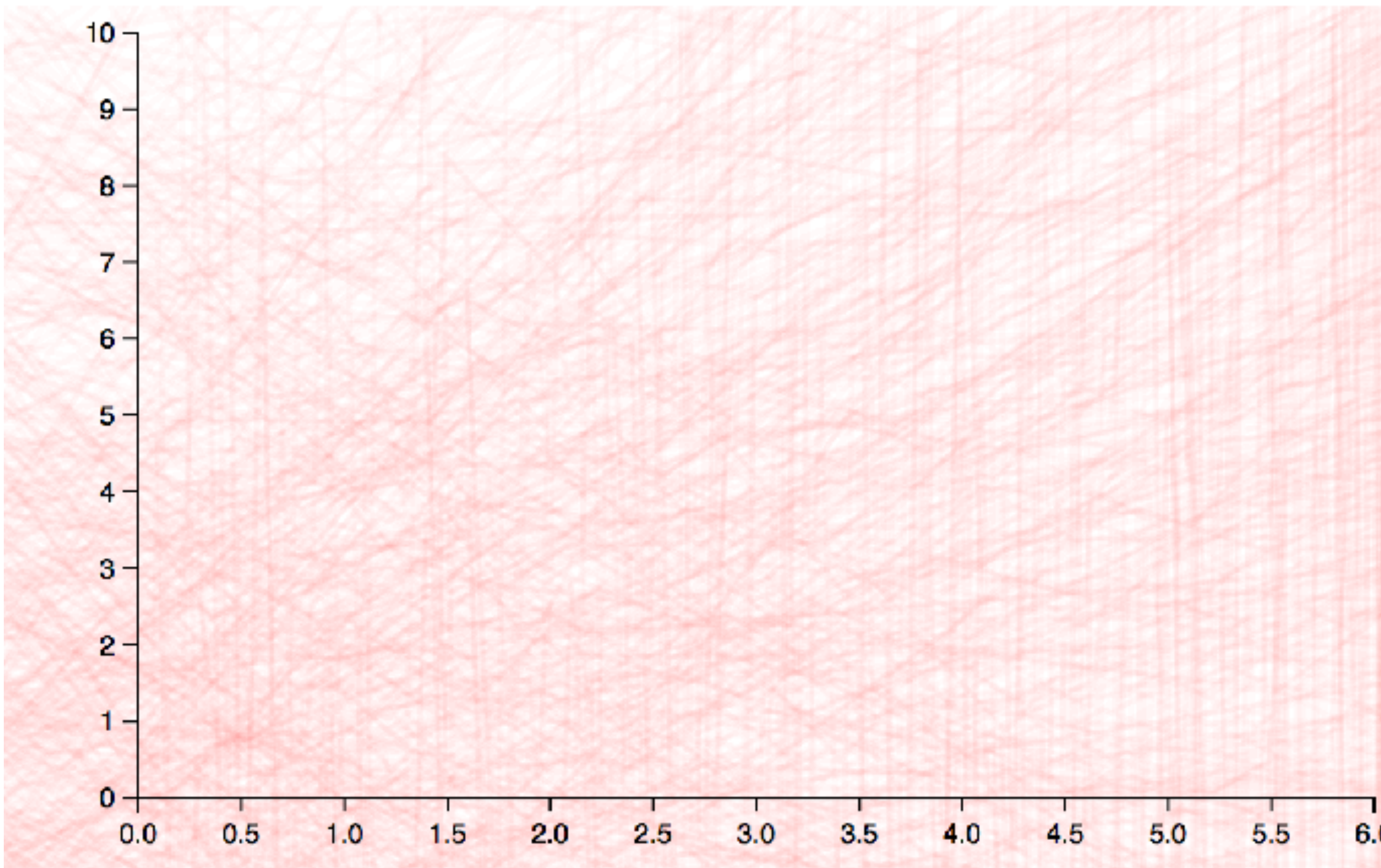
```
(let [s (sample (normal 0 2))
      b (sample (normal 0 6))
      f (fn [x] (+ (* s x) b))]

    (observe (normal (f 0) .5) .6)
    (observe (normal (f 1) .5) .7)
    (observe (normal (f 2) .5) 1.2)
    (observe (normal (f 3) .5) 3.2)
    (observe (normal (f 4) .5) 6.8)
    (observe (normal (f 5) .5) 8.2)
    (observe (normal (f 6) .5) 8.4)
```

(almost) Anglican program

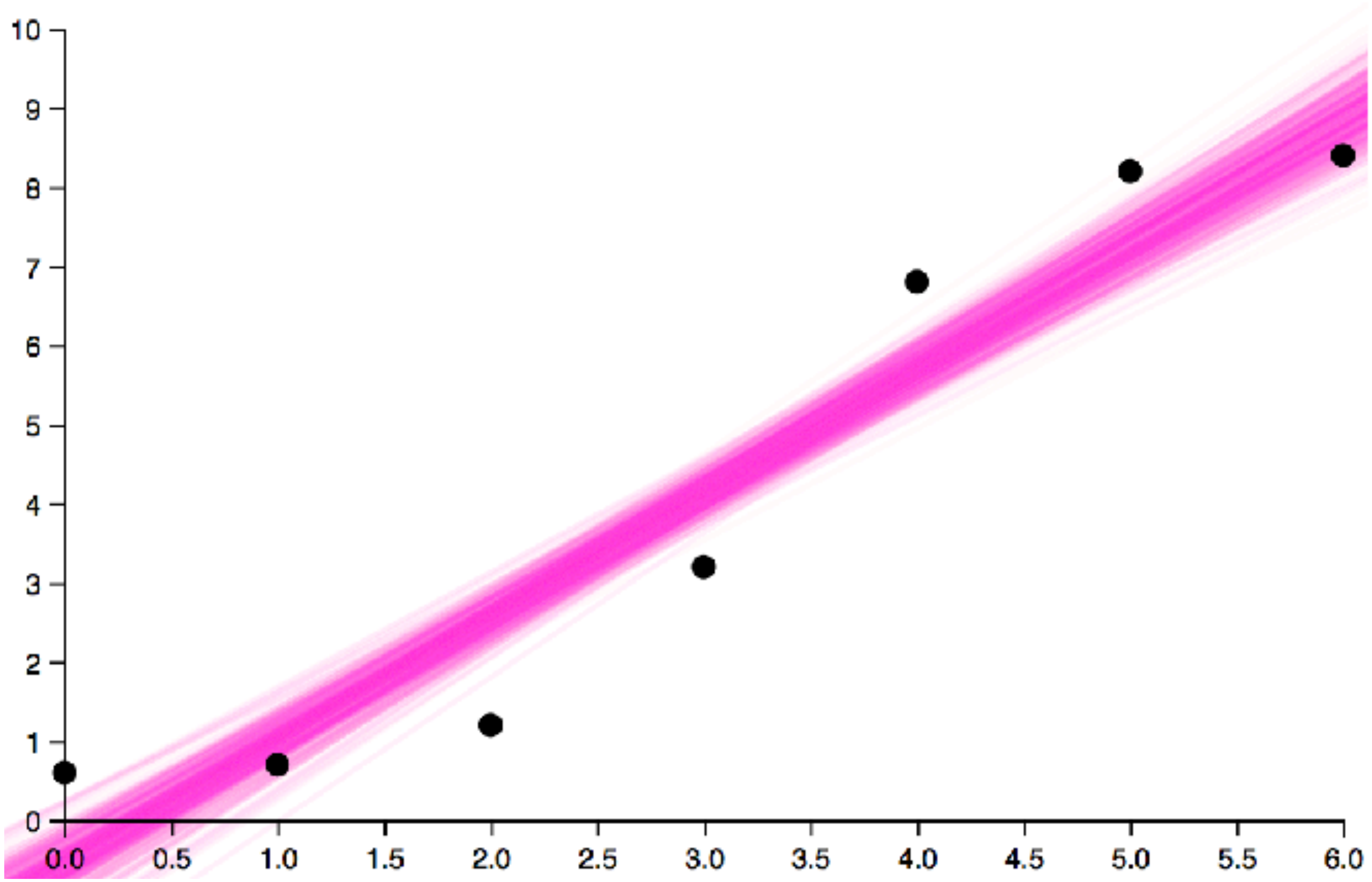
```
(let [s (sample (normal 0 2))
      b (sample (normal 0 6))
      f (fn [x] (+ (* s x) b))]

    (observe (normal (f 0) .5) .6)
    (observe (normal (f 1) .5) .7)
    (observe (normal (f 2) .5) 1.2)
    (observe (normal (f 3) .5) 3.2)
    (observe (normal (f 4) .5) 6.8)
    (observe (normal (f 5) .5) 8.2)
    (observe (normal (f 6) .5) 8.4)

  [s b])
```

NB: (predict :sb [s b]) should be used instead of [s b] in Anglican

Samples from prior



Samples from posterior



Semantic challenges

```
(let [s (sample (normal 0 2))
      b (sample (normal 0 6))
      f (fn [x] (+ (* s x) b))]

     (observe (normal (f 0) .5) .6)
     (observe (normal (f 1) .5) .7)
     (observe (normal (f 2) .5) 1.2)
     (observe (normal (f 3) .5) 3.2)
     (observe (normal (f 4) .5) 6.8)
     (observe (normal (f 5) .5) 8.2)
     (observe (normal (f 6) .5) 8.4)

  [s b])
```

```
(let [s (sample (normal 0 2))
      b (sample (normal 0 6))
      f (fn [x] (+ (* s x) b))]

    (observe (normal (f 0) .5) .6)
    (observe (normal (f 1) .5) .7)
    (observe (normal (f 2) .5) 1.2)
    (observe (normal (f 3) .5) 3.2)
    (observe (normal (f 4) .5) 6.8)
    (observe (normal (f 5) .5) 8.2)
    (observe (normal (f 6) .5) 8.4)

  [s b])
```

1. Continuous distributions.

Challenge 1: Continuous distributions

- Need care for handling continuous distributions on $\mathbb{R}$, to avoid paradoxes.
- Something like measure theory needed.
- Complex math.

```
(let [s (sample (normal 0 2))
      b (sample (normal 0 6))
      f (fn [x] (+ (* s x) b))]

    (observe (normal (f 0) .5) .6)
    (observe (normal (f 1) .5) .7)
    (observe (normal (f 2) .5) 1.2)
    (observe (normal (f 3) .5) 3.2)
    (observe (normal (f 4) .5) 6.8)
    (observe (normal (f 5) .5) 8.2)
    (observe (normal (f 6) .5) 8.4)

  [s b])
```

1. Continuous distributions.
2. Higher-order functions.

```
(let [s (sample (normal 0 2))
      b (sample (normal 0 6))
      f (fn [x] (+ (* s x) b))]

    (observe (normal (f 0) .5) .6)
    (observe (normal (f 1) .5) .7)
    (observe (normal (f 2) .5) 1.2)
    (observe (normal (f 3) .5) 3.2)
    (observe (normal (f 4) .5) 6.8)
    (observe (normal (f 5) .5) 8.2)
    (observe (normal (f 6) .5) 8.4)
```

~~[s b])~~
f)

1. Continuous distributions.
2. Higher-order functions.

```
(let [F (fn []
          (let [s (sample (normal 0 2))
                b (sample (normal 0 6))]
            (fn [x] (+ (* s x) b))))
      f (F)]
    (observe (normal (f 0) .5) .6)
    (observe (normal (f 1) .5) .7)
    (observe (normal (f 2) .5) 1.2)
    (observe (normal (f 3) .5) 3.2)
    (observe (normal (f 4) .5) 6.8)
    (observe (normal (f 5) .5) 8.2)
    (observe (normal (f 6) .5) 8.4)
```

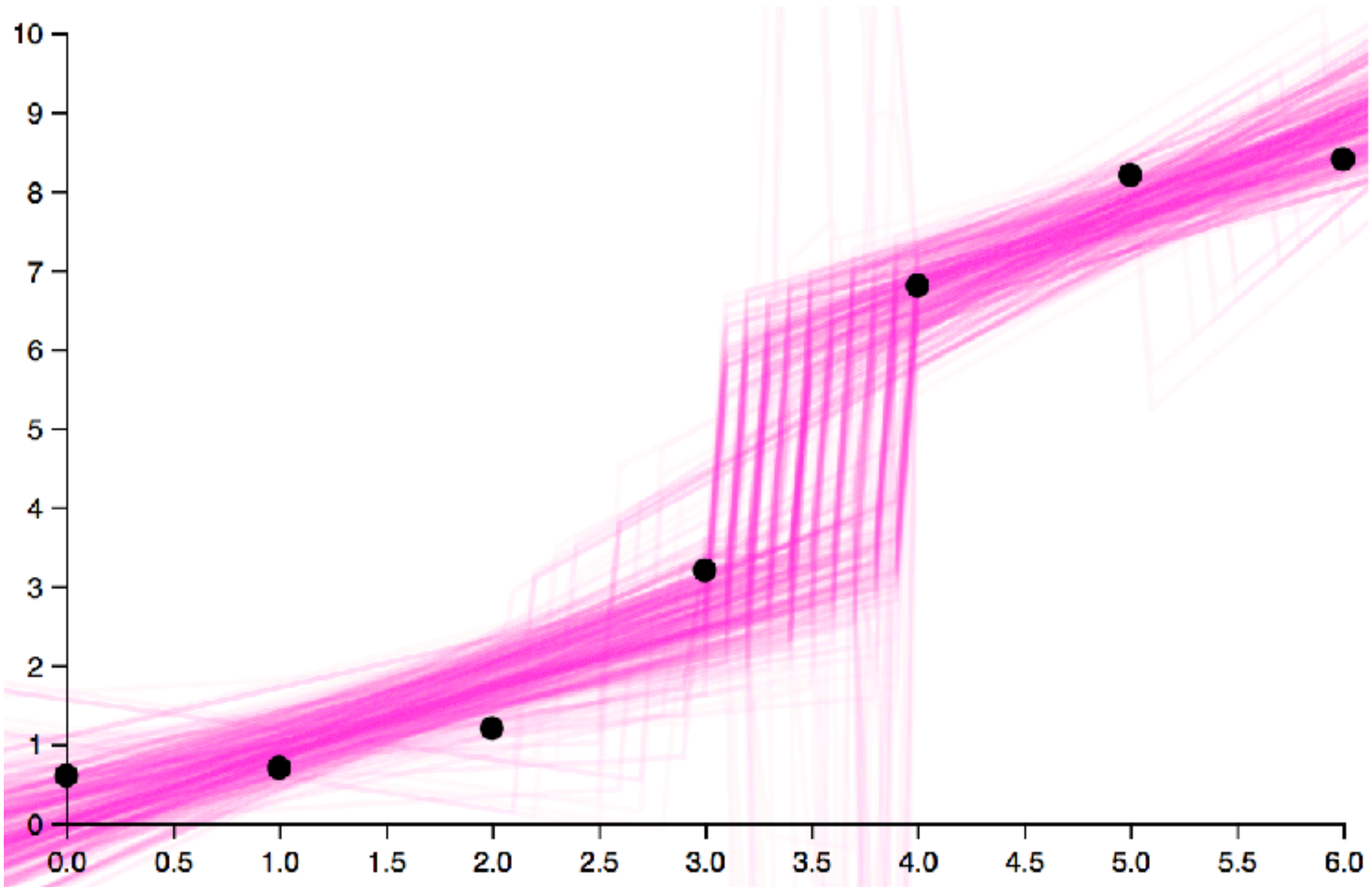
~~[s b])~~
f)

1. Continuous distributions.
2. Higher-order functions.

```
(let [F (fn []
          (let [s (sample (normal 0 2))
                b (sample (normal 0 6))]
            (fn [x] (+ (* s x) b))))
      f (add-change-points F 0 6)]
    (observe (normal (f 0) .5) .6)
    (observe (normal (f 1) .5) .7)
    (observe (normal (f 2) .5) 1.2)
    (observe (normal (f 3) .5) 3.2)
    (observe (normal (f 4) .5) 6.8)
    (observe (normal (f 5) .5) 8.2)
    (observe (normal (f 6) .5) 8.4)

  [s b])
  f)
```

1. Continuous distributions.
2. Higher-order functions.

Samples from posterior



```
(let [F (fn []
          (let [s (sample (normal 0 2))
                b (sample (normal 0 6))]
            (fn [x] (+ (* s x) b))))
      f (add-change-points F 0 6)]
    (observe (normal (f 0) .5) .6)
    (observe (normal (f 1) .5) .7)
    (observe (normal (f 2) .5) 1.2)
    (observe (normal (f 3) .5) 3.2)
    (observe (normal (f 4) .5) 6.8)
    (observe (normal (f 5) .5) 8.2)
    (observe (normal (f 6) .5) 8.4)
```

~~[s b])~~

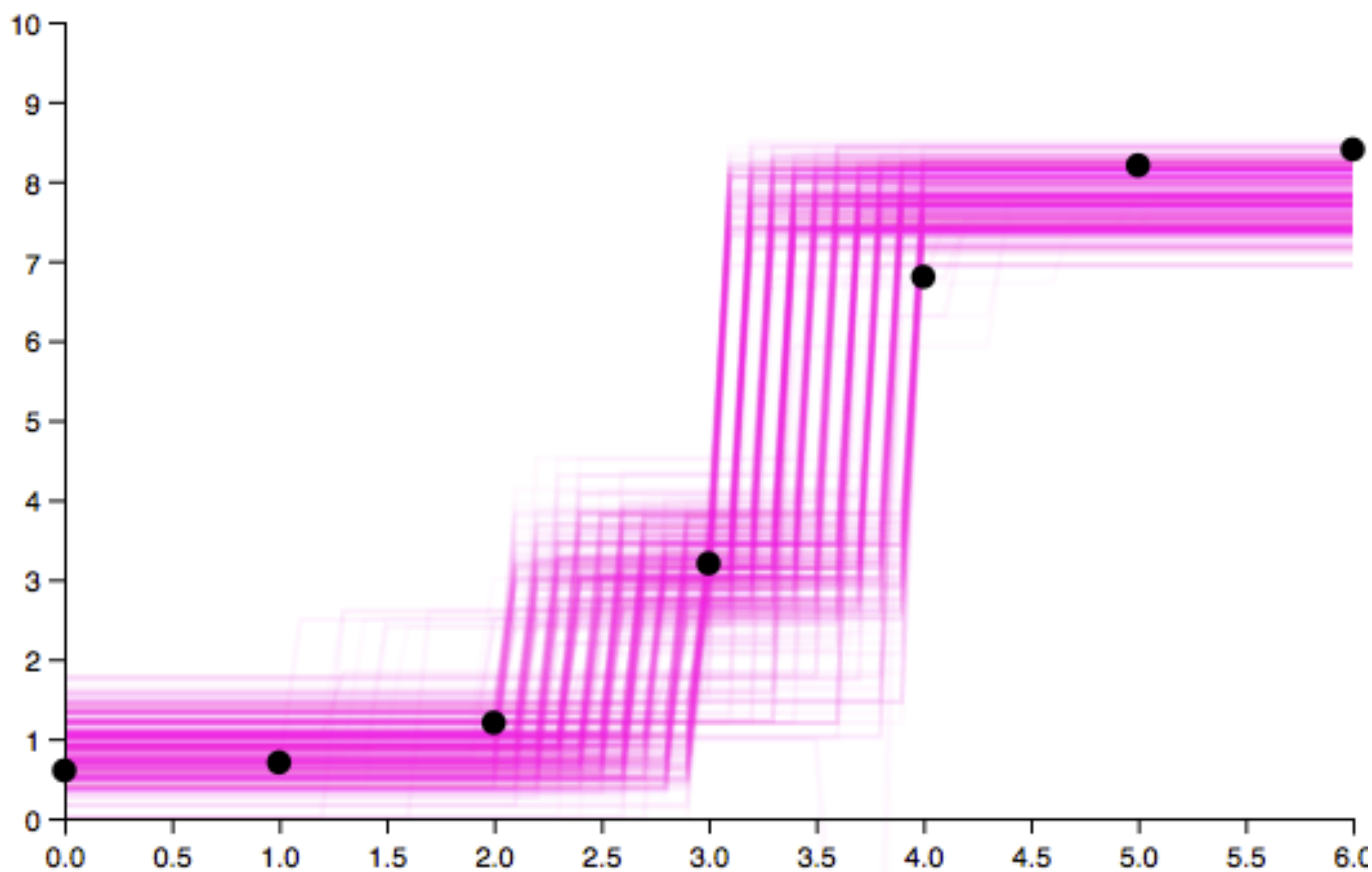
```
f)
```

1. Continuous distributions.
2. Higher-order functions.

```
(let [F (fn []
          (let [s (sample (normal 0 2))
                b (sample (normal 0 6))]
            (fn [x] (+ (* s x) b))))
      f (add-change-points F 0 6)]
    (observe (normal (f 0) .5) .6)
    (observe (normal (f 1) .5) .7)
    (observe (normal (f 2) .5) 1.2)
    (observe (normal (f 3) .5) 3.2)
    (observe (normal (f 4) .5) 6.8)
    (observe (normal (f 5) .5) 8.2)
    (observe (normal (f 6) .5) 8.4)

  [s b])
  f)
```

1. Continuous distributions.
2. Higher-order functions.

Samples from posterior



Challenge 2: Higher-order functions

Measure theory doesn't support HO fns well.

$$\mathrm{ev} : (\mathbb{R} \to_m \mathbb{R}) \times \mathbb{R} \to \mathbb{R}, \qquad \mathrm{ev}(f,x) = f(x).$$

[Aumann 61] ev is not measurable no matter which σ-algebra is used for $\mathbb{R} \to_m \mathbb{R}$.

[Cor] The category of measurable spaces is not cartesian closed.

```
(let [F (fn []
          (let [s (sample (normal 0 2))
                b (sample (normal 0 6))]
            (fn [x] (+ (* s x) b))))
      f (add-change-points F 0 6)]
    (observe (normal (f 0) .5) .6)
    (observe (normal (f 1) .5) .7)
    (observe (normal (f 2) .5) 1.2)
    (observe (normal (f 3) .5) 3.2)
    (observe (normal (f 4) .5) 6.8)
    (observe (normal (f 5) .5) 8.2)
    (observe (normal (f 6) .5) 8.4)
```

~~`[s b])`~~
`f)`

1. Continuous distributions.
2. Higher-order functions.

```
(let [F (fn []
          (let [s (sample (normal 0 2))
                b (sample (normal 0 6))]
            (fn [x] (+ (* s x) b))))
      f (add-change-points F 0 6)]
    (observe (normal (f 0) .5) .6)
    (observe (normal (f 1) .5) .7)
    (observe (normal (f 2) .5) 1.2)
    (observe (normal (f 3) .5) 3.2)
    (observe (normal (f 4) .5) 6.8)
    (observe (normal (f 5) .5) 8.2)
    (observe (normal (f 6) .5) 8.4)

  [s b])
  f)
```

1. Continuous distributions.
2. Higher-order functions.
3. Conditioning and prog. eqs.

Challenge 3: Conditioning and prog. eqs

$$[\![\texttt{e:real}]\!] \in M(\mathbb{R})$$

- M should model prob. computations.
- M should validate equations from statistics.
- M should be commutative.
- Difficult to find such M due to conditioning.

Challenge 3: Conditioning and prog. eqs

$[\![\texttt{e:real}]\!] \in M(\mathbb{R})$ nonfinite measures

- M should model prob. computations.
- M should validate equations from statistics.
- M should be commutative.
- Difficult to find such M due to conditioning.

Challenge 3: Conditioning and prog. eqs

nearly-finite measures

$[\![\, \texttt{e:real} \,]\!] \in M(\mathbb{R})$ nonfinite measures

- M should model prob. computations.
- M should validate equations from statistics.
- M should be commutative.
- Difficult to find such M due to conditioning.

```
(let [F (fn []
          (let [s (sample (normal 0 2))
                b (sample (normal 0 6))]
            (fn [x] (+ (* s x) b))))
      f (add-change-points F 0 6)]
    (observe (normal (f 0) .5) .6)
    (observe (normal (f 1) .5) .7)
    (observe (normal (f 2) .5) 1.2)
    (observe (normal (f 3) .5) 3.2)
    (observe (normal (f 4) .5) 6.8)
    (observe (normal (f 5) .5) 8.2)
    (observe (normal (f 6) .5) 8.4)
```

~~[s b])~~

```
f)
```

1. Continuous distributions.
2. Higher-order functions.
3. Conditioning and prog. eqs.

```
(let [F (fn []
          (let [s (sample (normal 0 2))
                b (sample (normal 0 6))]
            (fn [x] (+ (* s x) b))))
      f (add-change-points F 0 6)]
    (observe (normal (f 0) .5) .6)
    (observe (normal (f 1) .5) .7)
    (observe (normal (f 2) .5) 1.2)
    (observe (normal (f 3) .5) 3.2)
    (observe (normal (f 4) .5) 6.8)
    (observe (normal (f 5) .5) 8.2)
    (observe (normal (f 6) .5) 8.4)
```

~~[s b])~~

```
f)
```

Quasi-Borel space (QBS)

1. Continuous distributions.
2. Higher-order functions.
3. Conditioning and prog. eqs.

Big picture 1:
Extend measure theory
using category theory.

1. Continuous distr.
2. Higher-order fns.
3. Conditioning, prog. eqs.

~~1. Continuous distr.~~

2. Higher-order fns.

3. Conditioning, prog. eqs.

Meas_B

~~1. Continuous distr.~~

~~2. Higher-order fns.~~

3. Conditioning, prog. eqs.

Meas_B

Yoneda embedding ↘

$[\mathrm{Meas}_B{}^{op}, \mathrm{Set}]_\Pi$

~~1. Continuous distr.~~

~~2. Higher-order fns.~~

3. Conditioning, prog. eqs.

Meas_B

Yoneda embedding

Preserves nearly all the structures

$[\mathrm{Meas}_B{}^{op}, \mathrm{Set}]_\Pi$

~~1. Continuous distr.~~

~~2. Higher-order fns.~~

3. Conditioning, prog. eqs.

Meas_B

Yoneda embedding ↘

$[\mathrm{Meas}_B{}^{op}, \mathrm{Set}]_\Pi$

Enough structure for function types

~~1. Continuous distr.~~

~~2. Higher-order fns.~~

3. Conditioning, prog. eqs.

Meas_B

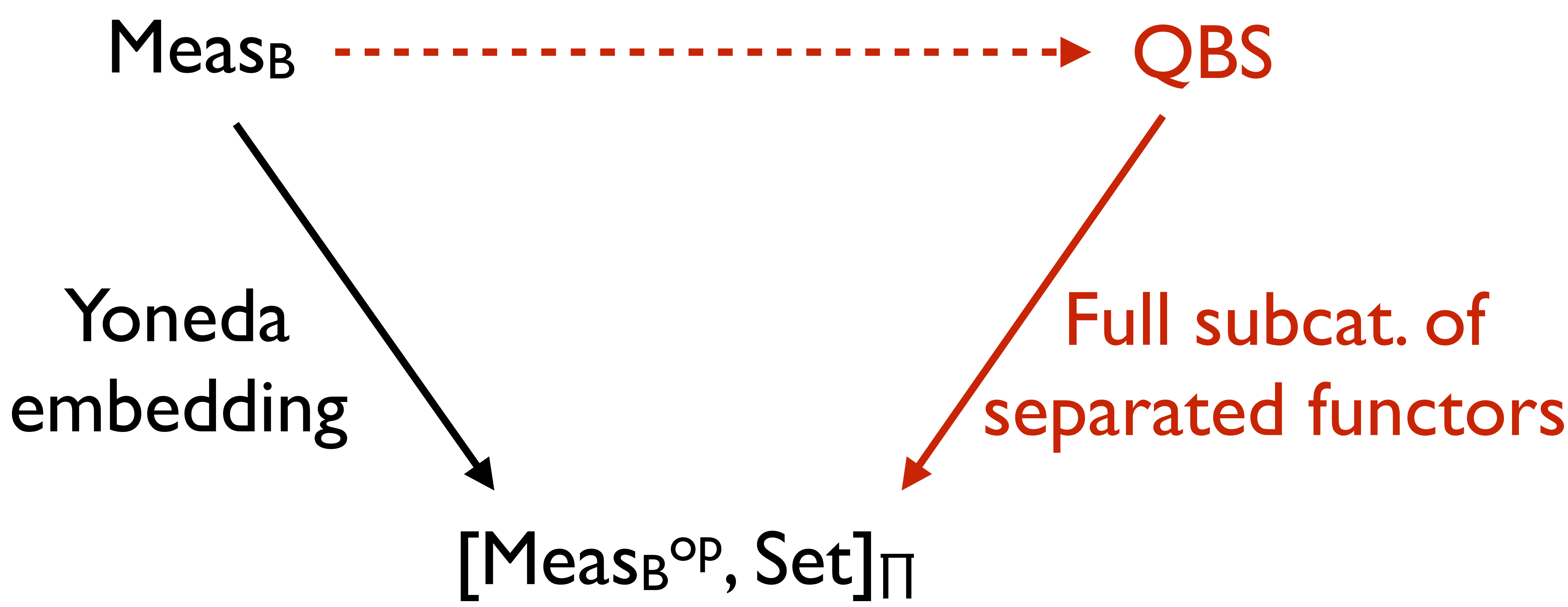
Yoneda embedding ↘

$[\mathrm{Meas}_B{}^{op}, \mathrm{Set}]_\Pi$

~~1. Continuous distr.~~

~~2. Higher-order fns.~~

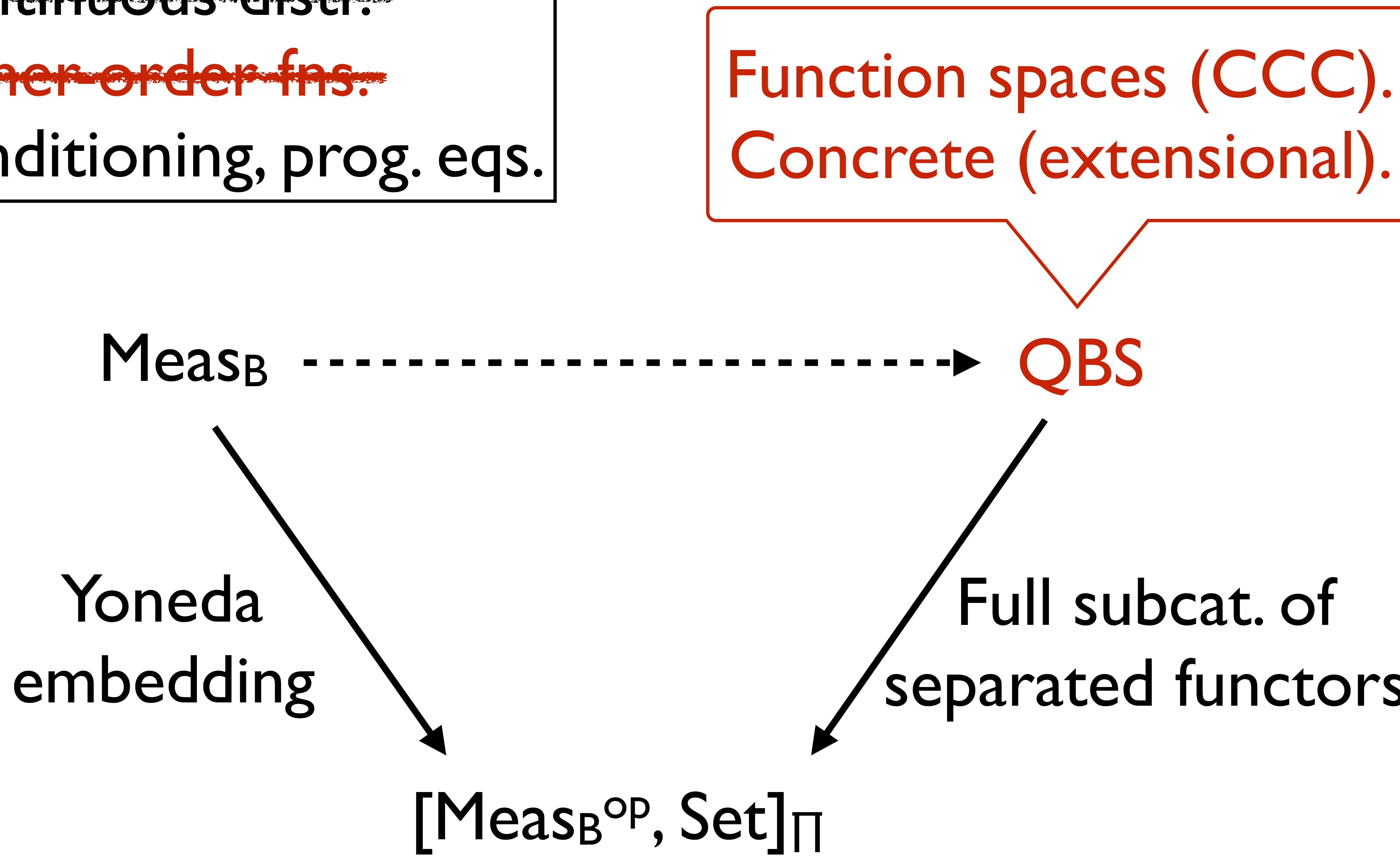
3. Conditioning, prog. eqs.



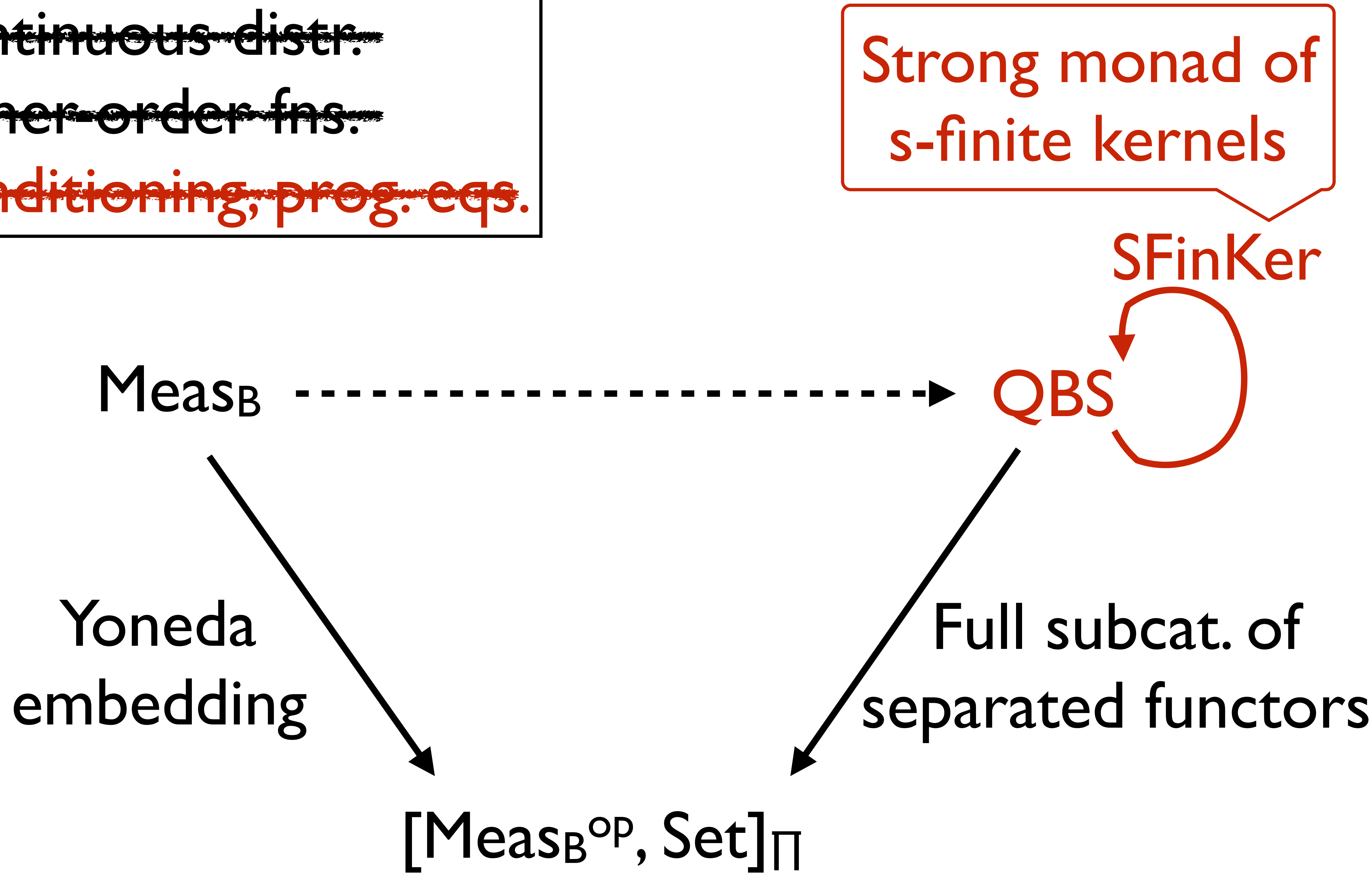
~~1. Continuous distr.~~

~~2. Higher-order fns.~~

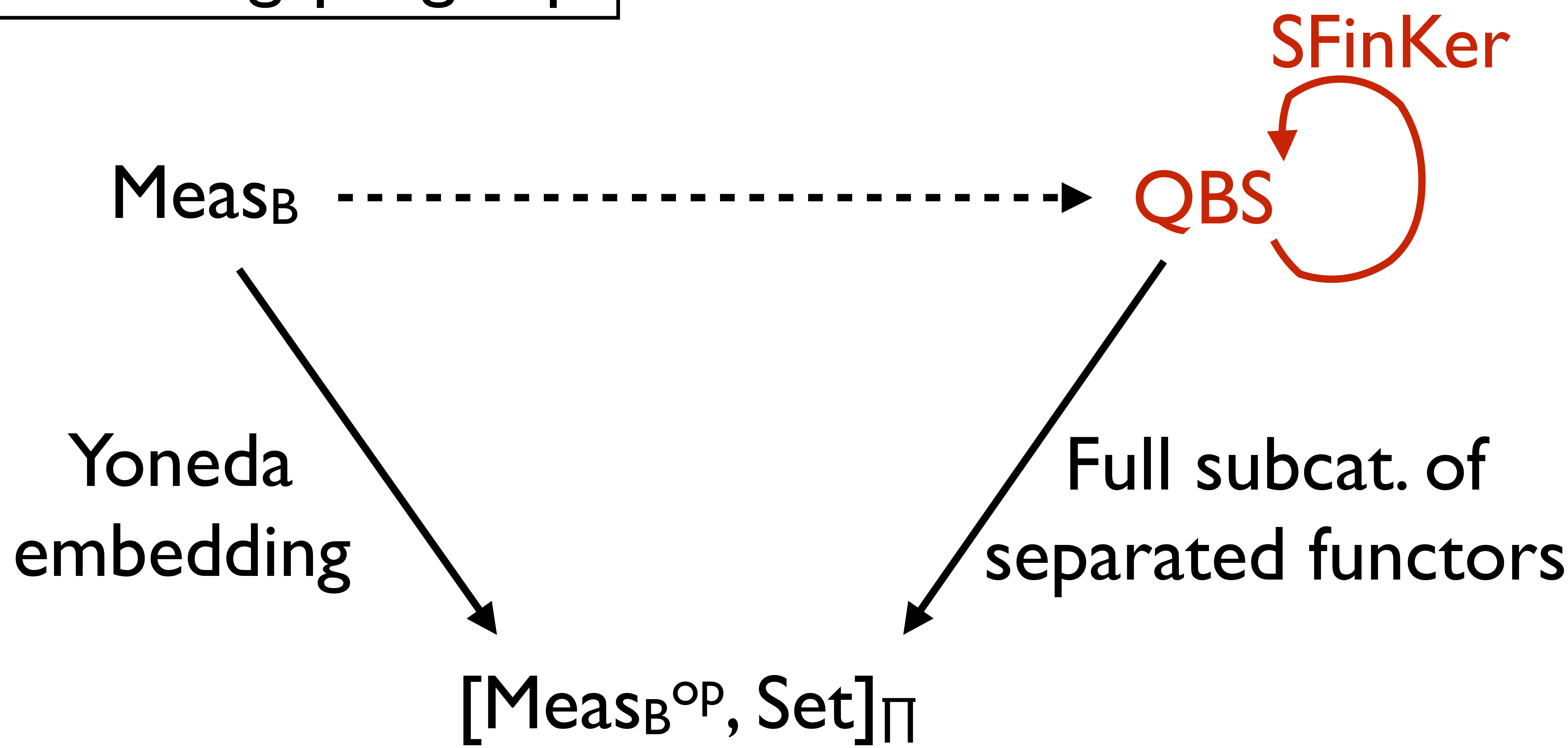
3. Conditioning, prog. eqs.



~~1. Continuous distr.~~
~~2. Higher-order fns.~~
~~3. Conditioning, prog. eqs.~~



~~1. Continuous distr.~~
~~2. Higher-order fns.~~
~~3. Conditioning, prog. eqs.~~



Big picture 2:
Random element first.

Random element α in X

Random element α in X

$$\alpha : \Omega \rightarrow X$$

- X - set of values.
- Ω - set of random seeds.
- Random seed generator.

Random element α in X in measure theory

$$\alpha : \Omega \to X$$

- X - set of values.
- Ω - set of random seeds.
- Random seed generator.

Random element α in X
in measure theory

$$\alpha : \Omega \to X$$

- X - set of values.
- Ω - set of random seeds.
- Random seed generator.

1. $\Sigma \subseteq 2^{\Omega}$, $\Theta \subseteq 2^{X}$

Random element α in X
in measure theory

$$\alpha : \Omega \rightarrow X$$

- X - set of values.
- Ω - set of random seeds.
- Random seed generator.

1. $\Sigma \subseteq 2^{\Omega}$, $\Theta \subseteq 2^{X}$
2. $\mu : \Sigma \rightarrow [0,1]$

Random element α in X
in measure theory

$\alpha : \Omega \to X$ is a random element
if $\alpha^{-1}(A) \in \Sigma$ for all $A \in \Theta$

- X - set of values.
- Ω - set of random seeds.
- Random seed generator.

1. $\Sigma \subseteq 2^{\Omega}$, $\Theta \subseteq 2^{X}$
2. $\mu : \Sigma \to [0,1]$

Random element α in X

$$\alpha : \Omega \to X$$

- X - set of values.
- Ω - set of random seeds.
- Random seed generator.

Random element α in X in quasi-Borel spaces

$$\alpha : \Omega \to X$$

- X - set of values.
- Ω - set of random seeds.
- Random seed generator.

Random element α in X in quasi-Borel spaces

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

- X - set of values.
- $\mathbb{R}$ - set of random seeds.
- Random seed generator.

1. $\mathbb{R}$ as random source
2. Borel subsets $\mathfrak{B} \subseteq 2^{\mathbb{R}}$

Random element α in X
in quasi-Borel spaces

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

- X - set of values.
- $\mathbb{R}$ - set of random seeds.
- Random seed generator.

1. $\mathbb{R}$ as random source
2. Borel subsets $\mathfrak{B} \subseteq 2^{\mathbb{R}}$

Random element α in X in quasi-Borel spaces

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

- X - set of values.
- $\mathbb{R}$ - set of random seeds.
- Random seed generator.

1. $\mathbb{R}$ as random source
2. Borel subsets $\mathfrak{B} \subseteq 2^{\mathbb{R}}$
3. $M \subseteq [\mathbb{R} \to X]$

Random element α in X
in quasi-Borel spaces

$\alpha : \mathbb{R} \to X$ is a random variable
if $\alpha \in M$

- X - set of values.
- $\mathbb{R}$ - set of random seeds.
- Random seed generator.

1. $\mathbb{R}$ as random source
2. Borel subsets $\mathfrak{B} \subseteq 2^{\mathbb{R}}$
3. $M \subseteq [\mathbb{R} \to X]$

- Measure theory:
  - Measurable space $(X, \Theta \subseteq 2^X)$.
  - Random element is an induced concept.
- QBS:
  - Quasi-Borel space $(X, M \subseteq [\mathbb{R} \to X])$.
  - M is the set of random elements.

Rest of this tutorial

1. Baby measure theory.
   PL with cont. distribution.
2. Quasi-Borel space (QBS).
   PL with cont. distr. & HO fns.
3. SFinKer monad on QBS.
   PL with cont. distr., HO fns & conditioning.

Rest of this tutorial

1. Baby measure theory.
   PL with cont. distribution.
2. Quasi-Borel space (QBS).
   PL with cont. distr. & HO fns.
3. SFinKer monad on QBS.
   PL with cont. distr., HO fns & conditioning.

Programming language

- Will be sloppy about its syntax.
- Higher-order call-by-value probabilistic PL.

```
t  ::=  bool  |  real  |  t x t | t → t

e  ::=  ...
```

Baby measure theory

How to specify prob. μ?

How to specify prob. μ?

$X = \{0, 1, 2\}$.

Define $\mu : X \to [0,1]$. E.g., $\mu = [0.4, 0.4, 0.2]$.

Lifted $\mu : 2^X \to [0,1]$ by $\mu(A) = \sum_{x \in A} \mu(x)$.

How to specify prob. μ?

$X = \mathbb{R}$.

Define $\mu : X \to [0,1]$.

Lifted $\mu : 2^X \to [0,1]$ by $\mu(A) = \sum_{x \in A} \mu(x)$.

How to specify prob. μ?

$X = \mathbb{R}$.

Define $\mu : X \to [0,1]$.

Lifted $\mu : 2^X \to [0,1]$ by $\mu(A) = \sum_{x \in A} \mu(x)$.

Uncountable sum.
Typically ∞.

How to specify prob. μ?

$X = \mathbb{R}$.

~~Define $\mu : X \to [0,1]$.~~

~~Lifted~~ $\mu : 2^X \to [0,1]$ by ~~$\mu(A) = \sum_{x \in A} \mu(x)$.~~

Define

How to specify prob. μ?

X = $\mathbb{R}$.

~~Define μ : X→[0,1]~~

~~Lifted μ : 2^X→[0,1] by μ(A) = $\sum_{x \in A}$ μ(x).~~

~~Define~~

Pick a good collection $\Sigma \subseteq 2^X$.

Define μ : Σ→[0,1] with some care.

How to specify prob. μ?

$X = \mathbb{R}$.

~~Define $\mu : X \to [0,1]$~~

Lifted ~~$\mu : 2^X \to [0,1]$ by $\mu(A) = \sum_{x \in A} \mu(x)$.~~

~~Define~~

σ-algebra

Pick a good collection $\Sigma \subseteq 2^X$.

Define μ : $\Sigma \to [0,1]$ with some care.

probability measure

Let $\Sigma \subseteq 2^X$.

Σ is a σ-algebra if it contains X, and is closed under countable union and set subtraction.

(X, Σ) is a measurable space if Σ is a σ-algebra.

Let $\Sigma \subseteq 2^X$.

Σ is a <u>σ-algebra</u> if it contains X, and is closed under countable union and set subtraction.

(X, Σ) is a <u>measurable space</u> if Σ is a σ-algebra.

$\mu : \Sigma \to [0,1]$ is a <u>probability measure</u> if $\mu(X)=1$ and $\mu(\biguplus_{n\in\mathbb{N}} A_n) = \sum_{n\in\mathbb{N}} \mu(A_n)$ for all disjoint A_n's.

(X,Σ,μ) is a <u>probability space</u> if …

[Q] What are not measurable spaces?

1. $(\mathbb{B}, 2^{\mathbb{B}})$.
2. $(\mathbb{B}\text{x}\mathbb{B}, \{ A\text{x}B \mid A \in 2^{\mathbb{B}} \text{ and } B \in 2^{\mathbb{B}} \})$.
3. $(\mathbb{R}, \{ A \subseteq \mathbb{R} \mid A \text{ or } (\mathbb{R}\text{-}A) \text{ countable} \})$.
4. $(\mathbb{R}, \{ (r,s] \mid r<s \})$.

[Q] What are not measurable spaces?

1. $(\mathbb{B}, 2^{\mathbb{B}})$.
2. $(\mathbb{B}\text{x}\mathbb{B}, \{ A\text{x}B \mid A\in 2^{\mathbb{B}} \text{ and } B\in 2^{\mathbb{B}} \})$.
3. $(\mathbb{R}, \{ A\subseteq\mathbb{R} \mid A \text{ or } (\mathbb{R}\text{-}A) \text{ countable} \})$.
4. $(\mathbb{R}, \{ (r,s] \mid r<s \})$.

[Q] Convert them to measurable spaces.

1. $(\mathbb{B}, 2^{\mathbb{B}})$.
2. $(\mathbb{B} \times \mathbb{B}, \{ A \times B \mid A \in 2^{\mathbb{B}} \text{ and } B \in 2^{\mathbb{B}} \})$.
3. $(\mathbb{R}, \{ A \subseteq \mathbb{R} \mid A \text{ or } (\mathbb{R} - A) \text{ countable} \})$.
4. $(\mathbb{R}, \{ (r,s] \mid r < s \})$.

[Q] Convert them to measurable spaces.

1. $(\mathbb{B}, 2^{\mathbb{B}})$.
2. $(\mathbb{B}\text{x}\mathbb{B}, \sigma\{\, A\text{x}B \mid A \in 2^{\mathbb{B}} \text{ and } B \in 2^{\mathbb{B}} \,\})$.
3. $(\mathbb{R}, \{\, A \subseteq \mathbb{R} \mid A \text{ or } (\mathbb{R}\text{-}A) \text{ countable} \,\})$.
4. $(\mathbb{R}, \sigma\{\, (r,s] \mid r<s \,\})$.

Closure exists.
$\sigma(\Pi)$ smallest σ-algebra containing Π.

(X, Σ), (Y, Θ) - mBle spaces.

Product σ-algebra: $\Sigma \otimes \Theta = \sigma\{A \times B \mid A \in \Sigma, B \in \Theta\}$.

Product space: $(X,\Sigma) \times_m (Y, \Theta) = (X \times Y, \Sigma \otimes \Theta)$.

Borel σ-algebra on $\mathbb{R}$: $\mathfrak{B} = \sigma\{(r,s] \mid r < s\}$.

Borel space: $(\mathbb{R}, \mathfrak{B})$.

Types mean mBle spaces

Types mean mBle spaces

$$[\![\texttt{bool}]\!] = (\mathbb{B}, 2^{\mathbb{B}})$$

$$[\![\texttt{real}]\!] = (\mathbb{R}, \mathfrak{B})$$

Types mean mBle spaces

$$[\![\texttt{bool}]\!] = (\mathbb{B}, 2^{\mathbb{B}})$$

$$[\![\texttt{real}]\!] = (\mathbb{R}, \mathfrak{B})$$

$$[\![t \times t']\!] = [\![t]\!] \times_m [\![t']\!]$$

$$[\![x_1{:}t_1, \ldots, x_n{:}t_n]\!] = [\![t_1]\!] \times_m \ldots \times_m [\![t_n]\!]$$

(X, Σ), (Y, Θ) - mBle spaces.

$f:X\to Y$ is measurable (denoted $f:X\to_m Y$) if $f^{-1}(A)\in\Sigma$ for all $A\in\Theta$.

$(X, \Sigma), (Y, \Theta)$ - mBle spaces.

$f: X \to Y$ is <u>measurable</u> (denoted $f: X \to_m Y$) if $f^{-1}(A) \in \Sigma$ for all $A \in \Theta$.

$k: X \times \Theta \to [0,1]$ is a <u>prob. kernel</u> if $k(x,-)$ is a prob. measure and $k(-,A)$ is measurable for all x, A.

Terms mean prob. kernels

$[\![\Gamma \vdash \mathtt{e} : \mathtt{t}]\!]$ is a prob. kernel from $[\![\Gamma]\!]$ to $[\![\mathtt{t}]\!]$.

Terms mean prob. kernels

$[\![\Gamma \vdash \mathtt{e} : \mathtt{t}]\!]$ is a prob. kernel from $[\![\Gamma]\!]$ to $[\![\mathtt{t}]\!]$.

$[\![\mathtt{y} : \mathtt{real} \vdash \mathtt{y + sample(norm(0,1))} : \mathtt{real}]\!]$

Terms mean prob. kernels

$[\![\Gamma \vdash \texttt{e} : \texttt{t}]\!]$ is a prob. kernel from $[\![\Gamma]\!]$ to $[\![\texttt{t}]\!]$.

$[\![\texttt{y} : \texttt{real} \vdash \texttt{y + sample(norm(0,1))} : \texttt{real}]\!](r, A)$

Terms mean prob. kernels

$[\![\Gamma \vdash \mathtt{e} : \mathtt{t}]\!]$ is a prob. kernel from $[\![\Gamma]\!]$ to $[\![\mathtt{t}]\!]$.

$[\![\mathtt{y} : \mathtt{real} \vdash \mathtt{y + sample(norm(0,1))} : \mathtt{real}]\!](r, A)$

$= \int_A$ density-norm(s | r, 1) ds.

Rest of this tutorial

1. Baby measure theory.
   PL with cont. distribution.
2. Quasi-Borel space (QBS).
   PL with cont. distr. & HO fns.
3. SFinKer monad on QBS.
   PL with cont. distr., HO fns & conditioning.

Quasi-Borel space

Quasi-Borel space - set with random elements.

Quasi-Borel space - set with random elements.

$$(X, M \subseteq [\mathbb{R} \to X])$$

such that M has enough random elements.

Quasi-Borel space - set with random elements.

$$(X, M \subseteq [\mathbb{R} \to X])$$

such that M has enough random elements.

Quasi-Borel space - set with random elements.

$$(X, M \subseteq [\mathbb{R} \to X])$$

such that M has enough random elements.

Quasi-Borel space - set with random elements.

$$(X, M \subseteq [\mathbb{R} \to X])$$

such that M has enough random elements.

1. M contains all constant functions.

Quasi-Borel space - set with random elements.

$$(X, M \subseteq [\mathbb{R} \to X])$$

such that M has enough random elements.

1. M contains all constant functions.
2. $(\alpha \circ \beta) \in M$ for all $\alpha \in M$ and mBle $\beta : \mathbb{R} \to \mathbb{R}$.

Quasi-Borel space - set with random elements.

$$(X, M \subseteq [\mathbb{R} \to X])$$

such that M has enough random elements.

1. M contains all constant functions.
2. $(\alpha \circ \beta) \in M$ for all $\alpha \in M$ and mBle $\beta : \mathbb{R} \to \mathbb{R}$.
3. If $\mathbb{R} = \uplus_{i \in \mathbb{N}} R_i$ with $R_i \in \mathfrak{B}$ and $\alpha_1, \alpha_2, \ldots \in M$, then $(\alpha_i \text{ when } R_i)_{i \in \mathbb{N}} \in M$.

   Here $(\alpha_i \text{ when } R_i)_{i \in \mathbb{N}}(r) = \alpha_i(r)$ for all $r \in R_i$.

[Q] Pick a non-QBS.

1. $(\mathbb{R}, \{\alpha:\mathbb{R}\to\mathbb{R} \mid \alpha \text{ is a constant function}\})$.
2. $(\mathbb{R}, [\mathbb{R}\to\mathbb{R}])$.
3. $(\mathbb{R}, \{\alpha:\mathbb{R}\to\mathbb{R} \mid \alpha \text{ measurable wrt. } \mathfrak{B}\})$.

[Q] Pick a non-QBS.

1. $(\mathbb{R},\ \{\alpha{:}\mathbb{R}\to\mathbb{R} \mid \alpha \text{ is a constant function}\})$.
2. $(\mathbb{R},\ [\mathbb{R}\to\mathbb{R}])$.
3. $(\mathbb{R},\ \{\alpha{:}\mathbb{R}\to\mathbb{R} \mid \alpha \text{ measurable wrt. } \mathfrak{B}\})$.

[Q] Turn it into a QBS.

1. $(\mathbb{R}, \{\alpha{:}\mathbb{R}\to\mathbb{R} \mid \alpha$ is a constant function$\})$.
2. $(\mathbb{R}, [\mathbb{R}\to\mathbb{R}])$.
3. $(\mathbb{R}, \{\alpha{:}\mathbb{R}\to\mathbb{R} \mid \alpha$ measurable wrt. $\mathfrak{B}\})$.

[Q] Turn it to a QBS.

$\{(\alpha_i \text{ when } R_i)_{i\in\mathbb{N}} \mid \alpha_i \text{ constant fn and } R_i\in\mathfrak{B}\}$

1. $(\mathbb{R},$ ~~$\{\alpha:\mathbb{R}\to\mathbb{R} \mid \alpha \text{ is a constant function}\}$~~$)$.
2. $(\mathbb{R}, [\mathbb{R}\to\mathbb{R}])$.
3. $(\mathbb{R}, \{\alpha:\mathbb{R}\to\mathbb{R} \mid \alpha \text{ measurable wrt. } \mathfrak{B}\})$.

[Q] Turn it to a QBS.

Standard way of converting a set to a QBS.

$\{(\alpha_i \text{ when } R_i)_{i\in\mathbb{N}} \mid \alpha_i \text{ constant fn and } R_i \in \mathfrak{B}\}$

1. ($\mathbb{R}$, ~~$\{\alpha:\mathbb{R}\to\mathbb{R} \mid \alpha$ is a constant function$\}$~~).
2. ($\mathbb{R}$, $[\mathbb{R}\to\mathbb{R}]$).
3. ($\mathbb{R}$, $\{\alpha:\mathbb{R}\to\mathbb{R} \mid \alpha$ measurable wrt. $\mathfrak{B}\}$).

[Q] Turn it to a QBS.

Standard way of converting a set to a QBS.

$\{(\alpha_i \text{ when } R_i)_{i\in\mathbb{N}} \mid \alpha_i \text{ constant fn and } R_i \in \mathfrak{B}\}$

1. ($\mathbb{R}$, ~~$\{\alpha:\mathbb{R}\to\mathbb{R} \mid \alpha$ is a constant function$\}$~~).
2. ($\mathbb{R}$, $[\mathbb{R}\to\mathbb{R}]$).
3. ($\mathbb{R}$, $\{\alpha:\mathbb{R}\to\mathbb{R} \mid \alpha$ measurable wrt. $\mathfrak{B}\}$).

Standard way of converting a mBle space to a QBS.

(QBS) morphism

(X,M), (Y,N) - QBSes.

$f : X \to Y$ is a morphism if $(f \circ \alpha) \in N$ for all $\alpha \in M$.

Maps random elements to random elements.

We will write $f : X \to_q Y$.

[Th] QBSes form a cartesian closed category. So, they provide good product and function spaces.

[Q] What are the sets of random elements?

1. Product: $(X,M) \times_q (Y,N) = (Z,O)$.
   - $Z = X \times Y$, $\pi_1(x,y) = x$, $\pi_2(x,y) = y$.
   - $O = ???$
2. Fn space: $[(X,M) \to_q (Y,N)] = (Z,O)$
   - $Z = \{ f \mid f : X \to_q Y \}$, $ev(f,x) = f(x)$
   - $O = ???$

[Q] What are the sets of random elements?

1. Product: $(X,M) \times_q (Y,N) = (Z,O)$.
   - $Z = X \times Y$, $\pi_1(x,y) = x$, $\pi_2(x,y) = y$.
   - $O = ???$
2. Fn space: $[(X,M) \to_q (Y,N)] = (Z,O)$
   - $Z = \{ f \mid f : X \to_q Y \}$, $ev(f,x) = f(x)$.
   - $O = ???$

[Q] What are the sets of random elements?

1. Product: $(X,M) \times_q (Y,N) = (Z,O)$.
   - $Z = X \times Y$, $\pi_1(x,y) = x$, $\pi_2(x,y) = y$.
   - $O = \{ \langle\alpha,\beta\rangle \mid \alpha \in M \text{ and } \beta \in N \}$.
2. Fn space: $[(X,M) \rightarrow_q (Y,N)] = (Z,O)$
   - $Z = \{ f \mid f : X \rightarrow_q Y \}$, $ev(f,x) = f(x)$
   - $O = \{ curry(g) \mid g : \mathbb{R} \times_q X \rightarrow_q Y \}$.

Why works?

[NO] ev : $(\mathbb{R} \to_m \mathbb{R}) \times_m \mathbb{R} \to_m \mathbb{R}$

vs

[YES] ev : $(\mathbb{R} \to_q \mathbb{R}) \times_q \mathbb{R} \to_q \mathbb{R}$

Why works?

[NO] ev : $(\mathbb{R} \to_m \mathbb{R}) \times_m \mathbb{R} \to_m \mathbb{R}$

vs

[YES] ev : $(\mathbb{R} \to_q \mathbb{R}) \times_q \mathbb{R} \to_q \mathbb{R}$

Because the QBS product is more permissive.

Types mean QBSes

$$[\![\texttt{bool}]\!] = \text{MStoQBS}(\mathbb{B}, 2^{\mathbb{B}})$$

$$[\![\texttt{real}]\!] = \text{MStoQBS}(\mathbb{R}, \mathfrak{B})$$

Types mean QBSes

$$[\![\texttt{bool}]\!] = \mathrm{MStoQBS}(\mathbb{B}, 2^{\mathbb{B}})$$

$$[\![\texttt{real}]\!] = \mathrm{MStoQBS}(\mathbb{R}, \mathfrak{B})$$

$$[\![t \times t']\!] = [\![t]\!] \times_q [\![t']\!]$$

$$[\![x_1{:}t_1, \ldots, x_n{:}t_n]\!] = [\![t_1]\!] \times_q \ldots \times_q [\![t_n]\!]$$

Types mean QBSes

$$[\![\texttt{bool}]\!] = \text{MStoQBS}(\mathbb{B}, 2^{\mathbb{B}})$$

$$[\![\texttt{real}]\!] = \text{MStoQBS}(\mathbb{R}, \mathfrak{B})$$

$$[\![t \times t']\!] = [\![t]\!] \times_q [\![t']\!]$$

$$[\![t \to t']\!] = [[\![t]\!] \to_q \text{Monad}([\![t']\!])]$$

$$[\![x_1{:}t_1, \ldots, x_n{:}t_n]\!] = [\![t_1]\!] \times_q \ldots \times_q [\![t_n]\!]$$

Terms mean morphisms almost

$[\![\Gamma \vdash \mathtt{e} : \mathtt{t}]\!]$ is a morphism $[\![\Gamma]\!] \to_{\mathtt{q}}$ Monad$[\![\mathtt{t}]\!]$.

Probability measure on Quasi-Borel space

A probability measure on a QBS (X,M) is a pair (α,μ) of $\alpha \in M$ and a prob. measure μ on $(\mathbb{R},\mathfrak{B})$.

A <u>probability measure</u> on a QBS (X,M) is a pair (α,μ) of $\alpha \in M$ and a prob. measure μ on $(\mathbb{R},\mathfrak{B})$.

random seed generator

A <u>probability measure</u> on a QBS (X,M) is a pair (α,μ) of $\alpha \in M$ and a prob. measure μ on $(\mathbb{R},\mathfrak{B})$.

seed convertor

A <u>probability measure</u> on a QBS (X,M) is a pair (α,μ) of α∈M and a prob. measure μ on ($\mathbb{R}$,$\mathfrak{B}$).

A <u>probability measure</u> on a QBS (X,M) is a pair (α,μ) of $\alpha \in M$ and a prob. measure μ on ($\mathbb{R}$,$\mathfrak{B}$).

E.g.

$$(X,M) = \text{MStoQBS}(\mathbb{B}, 2^{\mathbb{B}})$$

$$\mu = \text{uniform}(0,1], \quad \alpha(r) = \text{if } (r < 0.5) \text{ true false}$$

A <u>probability measure</u> on a QBS (X,M) is a pair (α,μ) of $\alpha \in M$ and a prob. measure μ on $(\mathbb{R}, \mathfrak{B})$.

E.g.

$$(X,M) = \mathrm{MStoQBS}(\mathbb{B}, 2^{\mathbb{B}})$$

μ = uniform(0,1], α(r) = if (r < 0.5) true false

μ' = uniform(0,2]/2, α'(r) = if (r < 1) true false

A probability measure on a QBS (X,M) is a pair (α,μ) of $\alpha \in M$ and a prob. measure μ on $(\mathbb{R},\mathfrak{B})$.

Quotient prob. measures by the smallest ~ s.t.

$$(\alpha,\mu) \sim (\beta,\nu)$$

if $\alpha \circ f = \beta$ and $\nu \circ f^{-1} = \mu$ for some $f:\mathbb{R} \to_m \mathbb{R}$.

$[\alpha,\mu]$ - equivalence class.

QBS of prob. measures

$$\mathrm{Prob}(X,M) = (Y,N)$$

Y = { [α,μ] | (α,μ) is a prob. meas. on (X,M) }.

N = { λr.[α,k(r)] | α∈M and k : $\mathbb{R}$x$\mathfrak{B}$→[0,1] is a prob. kernel }.

[Lem] Prob is a strong monad.

Completing the definitions

$$[\![t \to t']\!] = [[\![t]\!] \to_q \text{Prob}([\![t']\!])]$$

$[\![\Gamma \vdash e : t]\!]$ is a morphism $[\![\Gamma]\!] \to_q \text{Prob}[\![t]\!]$

Rest of this tutorial

1. Baby measure theory.
   PL with cont. distribution.
2. Quasi-Borel space (QBS).
   PL with cont. distr. & HO fns.
3. SFinKer monad on QBS.
   PL with cont. distr., HO fns & conditioning.

Conditioning and SFinKer monad on QBS

$[\![\Gamma \vdash \texttt{e} : \texttt{t}]\!]$ is a morphism $[\![\Gamma]\!] \to_{\texttt{q}} \text{Monad}[\![\texttt{t}]\!]$

$[\![\Gamma \vdash \texttt{e} : \texttt{t}]\!]$ is a morphism $[\![\Gamma]\!] \to_q \mathrm{Monad}[\![\texttt{t}]\!]$

Bayes's rule:

$$p(o \mid h) \times p(h) = p(h \mid o) \times p(o)$$

$[\![\Gamma \vdash \texttt{e} : \texttt{t}]\!]$ is a morphism $[\![\Gamma]\!] \to_q \mathrm{Monad}[\![\texttt{t}]\!]$

Bayes's rule:

$$p(o \mid h) \times p(h) = p(h \mid o) \times p(o)$$

1. Monad(_) = Prob(_).

$[\![\Gamma \vdash \mathtt{e} : \mathtt{t}]\!]$ is a morphism $[\![\Gamma]\!] \rightarrow_q \text{Monad}[\![\mathtt{t}]\!]$

Bayes's rule:

p(o | h) x p(h) = p(h | o) x p(o)

1. Monad(_) = Prob(_). Sometimes undefined.

$[\![\Gamma \vdash \mathtt{e} : \mathtt{t}]\!]$ is a morphism $[\![\Gamma]\!] \to_q \text{Monad}[\![\mathtt{t}]\!]$

Bayes's rule:

$$p(o \mid h) \times p(h) = p(h \mid o) \times p(o)$$

1. Monad(_) = Prob(_). Sometimes undefined.

2. Monad(_) = Prob([0,∞) $\times_q$ _).

$[\![\Gamma \vdash \mathtt{e} : \mathtt{t}]\!]$ is a morphism $[\![\Gamma]\!] \to_q \mathrm{Monad}[\![\mathtt{t}]\!]$

Bayes's rule:

$p(o \mid h) \times p(h) = p(h \mid o) \times p(o)$

1. Monad(_) = Prob(_). Sometimes undefined.
2. Monad(_) = Prob([0,∞) $\times_q$ _). Failed eqs.

Failed conjugate-prior equation from statistics

⟦
```
let x=sample(beta(1,1)) in
observe(flip(x),true);
x
```
⟧

$$\neq$$

⟦
```
observe(flip(0.5),true);
sample(beta(2,1))
```
⟧

$[\![\Gamma \vdash \texttt{e} : \texttt{t}]\!]$ is a morphism $[\![\Gamma]\!] \to_q \text{Monad}[\![\texttt{t}]\!]$

Bayes's rule:

$$p(o \mid h) \times p(h) = p(h \mid o) \times p(o)$$

1. Monad(_) = Prob(_). Sometimes undefined.

2. Monad(_) = Prob($[0,\infty) \times_q$ _). Failed eqs.

$[\![\Gamma \vdash \mathtt{e} : \mathtt{t}]\!]$ is a morphism $[\![\Gamma]\!] \to_q \text{Monad}[\![\mathtt{t}]\!]$

Bayes's rule:

$$p(o \mid h) \times p(h) = p(h \mid o) \times p(o) = p(o, h)$$

1. Monad(_) = Prob(_). Sometimes undefined.
2. Monad(_) = Prob($[0,\infty) \times_q$ _). Failed eqs.
3. Monad(_) = Meas(_).

$[\![\Gamma \vdash \mathtt{e} : \mathtt{t}]\!]$ is a morphism $[\![\Gamma]\!] \to_q \mathrm{Monad}[\![\mathtt{t}]\!]$

Bayes's rule:

$$p(o \mid h) \times p(h) = p(h \mid o) \times p(o) = p(o, h)$$

1. Monad(_) = Prob(_). Sometimes undefined.
2. Monad(_) = Prob($[0,\infty) \times_q$ _). Failed eqs.
3. Monad(_) = Meas(_). No commutativity.

Failed commutativity

$$[\![\ \texttt{let x=e in let y=e' in e''}\ ]\!] \neq [\![\ \texttt{let y=e' in let x=e in e''}\ ]\!]$$

if x doesn't occur in e' and y doesn't occur in e

$[\![\Gamma \vdash e : t]\!]$ is a morphism $[\![\Gamma]\!] \to_q \text{Monad}[\![t]\!]$

Bayes's rule:

$$p(o \mid h) \times p(h) = p(h \mid o) \times p(o) = p(o, h)$$

1. Monad(_) = Prob(_). Sometimes undefined.
2. Monad(_) = Prob($[0,\infty) \times_q$ _). Failed eqs.
3. Monad(_) = Meas(_). No commutativity.

$[\![\Gamma \vdash e : t]\!]$ is a morphism $[\![\Gamma]\!] \to_q \text{Monad}[\![t]\!]$

Bayes's rule:

$$p(o \mid h) \times p(h) = p(h \mid o) \times p(o) = p(o, h)$$

1. Monad(_) = Prob(_). Sometimes undefined.
2. Monad(_) = Prob($[0,\infty) \times_q$ _). Failed eqs.
3. Monad(_) = Meas(_). No commutativity.
4. Monad(_) = SFinKer(_).

QBS of prob. measures

$$\mathrm{Prob}(X,M) = (Y,N)$$

$$Y = \{\ [\alpha,\mu]\ \mid\ \alpha \in M,\quad \mu \text{ prob. meas. on } (\mathbb{R},\mathfrak{B})\ \}.$$

$$N = \{\ \lambda r.[\alpha,k(r)]\ \mid$$
$$\alpha \in M,\quad k : \mathbb{R} \times \mathfrak{B} \to [0,1] \text{ prob. kernel }\}.$$

QBS of ~~prob. measures~~ s-finite kernels

SFinKer(X,M)
~~Prob(X,M)~~ = (Y,N)

Y = { [α,μ] | α∈M, μ prob. meas. on ($\mathbb{R}$,$\mathfrak{B}$) }.

N = { λr.[α,k(r)] |
α∈M, k : $\mathbb{R}$x$\mathfrak{B}$→[0,1] prob. kernel }.

QBS of ~~prob. measures~~ s-finite kernels

SFinKer(X,M)
~~Prob(X,M)~~ = (Y,N)

Y = { [α,μ] | α∈M, μ ~~prob. meas.~~ s-finite measure on $(\mathbb{R},\mathfrak{B})$ }.

N = { λr.[α,k(r)] |
α∈M, k : $\mathbb{R}$x$\mathfrak{B}$→[0,1] prob. kernel }.

μ <u>finite</u> if like prob. measure but just $\mu(\mathbb{R})<\infty$.
μ <u>s-finite</u> if countable sum of finite measures.

QBS of prob. measures

k finite if like prob. kernel but $\sup_r k(r,\mathbb{R}) < \infty$.
k s-finite if countable sum of finite kernels.

SFinKer(X,M)
~~Prob(X,M)~~ = (Y,N)

$Y = \{ [\alpha,\mu] \mid \alpha \in M,$ μ ~~prob. meas.~~ on $(\mathbb{R},\mathfrak{B})$ }.
s-finite measure

$N = \{ \lambda r.[\alpha,k(r)] \mid$
$\alpha \in M,$ ~~$k : \mathbb{R} \times \mathfrak{B} \to [0,1]$ prob. kernel~~ }.
$k : \mathbb{R} \times \mathfrak{B} \to [0,\infty]$ s-finite kernel

μ finite if like prob. measure but just $\mu(\mathbb{R}) < \infty$.
μ s-finite if countable sum of finite measures.

[Th] SFinKer can be used to define the semantics of prob. PL with conditioning. (i.e., strong monad.)

[Th] It validates commutativity of programs and prog. eqs from statistics.

$$[\![\texttt{bool}]\!] = \text{MStoQBS}(\mathbb{B}, 2^{\mathbb{B}})$$

$$[\![\texttt{real}]\!] = \text{MStoQBS}(\mathbb{R}, \mathfrak{B})$$

$$[\![t \times t']\!] = [\![t]\!] \times_q [\![t']\!]$$

$$[\![t \to t']\!] = [[\![t]\!] \to_q \text{SFinKer}([\![t']\!])]$$

$$[\![x_1{:}t_1, \ldots, x_n{:}t_n]\!] = [\![t_1]\!] \times_q \ldots \times_q [\![t_n]\!]$$

$[\![\Gamma \vdash \texttt{e} : \texttt{t}]\!]$ is a morphism $[\![\Gamma]\!] \to_q \text{SFinKer}[\![\texttt{t}]\!]$

$$[\![\texttt{bool}]\!] = \mathrm{MStoQBS}(\mathbb{B}, 2^{\mathbb{B}})$$

$$[\![\texttt{real}]\!] = \mathrm{MStoQBS}(\mathbb{R}, \mathfrak{B})$$

$$[\![t \times t']\!] = [\![t]\!] \times_q [\![t']\!]$$

$$[\![t \to t']\!] = [[\![t]\!] \to_q \mathrm{SFinKer}([\![t']\!])]$$

$$[\![x_1{:}t_1, \ldots, x_n{:}t_n]\!] = [\![t_1]\!] \times_q \ldots \times_q [\![t_n]\!]$$

$[\![\Gamma \vdash \texttt{e} : \texttt{t}]\!]$ is a morphism $[\![\Gamma]\!] \to_q \mathrm{SFinKer}[\![\texttt{t}]\!]$

$$[\![\texttt{bool}]\!] = \mathrm{MStoQBS}(\mathbb{B}, 2^{\mathbb{B}})$$

$$[\![\texttt{real}]\!] = \mathrm{MStoQBS}(\mathbb{R}, \mathfrak{B})$$

$$[\![t \times t']\!] = [\![t]\!] \times_q [\![t']\!]$$

$$[\![t \to t']\!] = [[\![t]\!] \to_q \mathrm{SFinKer}([\![t']\!])]$$

$$[\![x_1{:}t_1, \ldots, x_n{:}t_n]\!] = [\![t_1]\!] \times_q \ldots \times_q [\![t_n]\!]$$

$[\![\Gamma \vdash \texttt{e} : \texttt{t}]\!]$ is a morphism $[\![\Gamma]\!] \to_q \mathrm{SFinKer}[\![\texttt{t}]\!]$

$$[\![\texttt{bool}]\!] = \mathrm{MStoQBS}(\mathbb{B}, 2^{\mathbb{B}})$$

$$[\![\texttt{real}]\!] = \mathrm{MStoQBS}(\mathbb{R}, \mathfrak{B})$$

$$[\![t \times t']\!] = [\![t]\!] \times_q [\![t']\!]$$

$$[\![t \to t']\!] = [[\![t]\!] \to_q \mathrm{SFinKer}([\![t']\!])]$$

$$[\![x_1{:}t_1, \ldots, x_n{:}t_n]\!] = [\![t_1]\!] \times_q \ldots \times_q [\![t_n]\!]$$

$[\![\Gamma \vdash \texttt{e} : \texttt{t}]\!]$ is a morphism $[\![\Gamma]\!] \to_q \mathrm{SFinKer}[\![\texttt{t}]\!]$

$$[\![\texttt{bool}]\!] = \mathrm{MStoQBS}(\mathbb{B}, 2^{\mathbb{B}})$$

$$[\![\texttt{real}]\!] = \mathrm{MStoQBS}(\mathbb{R}, \mathfrak{B})$$

$$[\![t \times t']\!] = [\![t]\!] \times_q [\![t']\!]$$

$$[\![t \to t']\!] = [[\![t]\!] \to_q \mathrm{SFinKer}([\![t']\!])]$$

$$[\![x_1{:}t_1, \ldots, x_n{:}t_n]\!] = [\![t_1]\!] \times_q \ldots \times_q [\![t_n]\!]$$

$[\![\Gamma \vdash \texttt{e} : \texttt{t}]\!]$ is a morphism $[\![\Gamma]\!] \to_q \mathrm{SFinKer}[\![\texttt{t}]\!]$

References

1. A convenient category for higher-order probability theory. Heunen et a. LICS'17.
2. Commutative semantics for probabilistic programs. Staton. ESOP'17.

References

1. A convenient category for higher-order probability theory. Heunen et a. LICS'17.
2. Commutative semantics for probabilistic programs. Staton. ESOP'17.