CS423: Probabilistic Programming

Denotational Semantics of Probabilistic Programs

Hongseok Yang
KAIST

Denotational semantics describes a mapping from prob. programs to probabilistic models.

Denotational semantics describes a mapping from prob. programs to probabilistic models.

```
(let [x (sample (normal 0 1))
      y (observe (normal x 1) 2)]
  x)
```

Denotational semantics describes a mapping from prob. programs to probabilistic models.

```
(let [x (sample (normal 0 1))
      y (observe (normal x 1) 2)]
  x)
```

Denotational semantics describes a mapping from prob. programs to probabilistic models.

```
(let [x (sample (normal 0 1))
      y (observe (normal x 1) 2)]
  x)
```

Denotational semantics describes a mapping from prob. programs to probabilistic models.

```
(let [x (sample (normal 0 1))
      y (observe (normal x 1) 2)]
  x)
```

Denotational semantics describes a mapping from prob. programs to probabilistic models.

$$\left[\!\left[\ \boxed{\texttt{(let [x (sample (normal 0 1))} \\ \texttt{y (observe (normal x 1) 2)]} \\ \texttt{x)}}\ \right]\!\right] = p(x \mid y{=}2)$$

Denotational semantics describes a mapping from prob. programs to probabilistic models.

$\llbracket$
```
(let [x (sample (normal 0 1))
      y (observe (normal x 1) 2)]
  x)
```
$\rrbracket = p(x \mid y{=}2)$

$\llbracket$
```
(let [x (sample (normal 0 1))
      y (observe (normal x 1) 2)]
  x)
```
$\rrbracket' = p(x, y{=}2)$

Why should one care about deno. semantics?

Why should one care about deno. semantics?

(or formal)

Why should one care about deno. semantics?

(or formal)

Why should one care about deno. semantics?

1. Specification of inference algorithms.
2. Compiler optimisation.
3. Detection of ill-defined models.
4. Clear meaning of complex models.

(or formal)

Why should one care about deno. semantics?

1. Specification of inference algorithms.
2. Compiler optimisation.
3. Detection of ill-defined models.
4. Clear meaning of complex models.

```
(let [x        (sample (normal 0 1))
      x-pdf    (normal-pdf x 0 1)
      y        (observe (exponential (/ 1 x-pdf)) 0)]
  x)
```

```
(let [x        (sample (normal 0 1))
      x-pdf    (normal-pdf x 0 1)
      y        (observe (exponential (/ 1 x-pdf)) 0)]
  x)
```

```
(let [x       (sample (normal 0 1))
      x-pdf   (normal-pdf x 0 1)
      y       (observe (exponential (/ 1 x-pdf)) 0)]
  x)
```

```
(let [x      (sample (normal 0 1))
      x-pdf  (normal-pdf x 0 1)
      y      (observe (exponential (/ 1 x-pdf)) 0)]
  x)
```

```
(let [x       (sample (normal 0 1))
      x-pdf   (normal-pdf x 0 1)
      y       (observe (exponential (/ 1 x-pdf)) 0)]
  x)
```

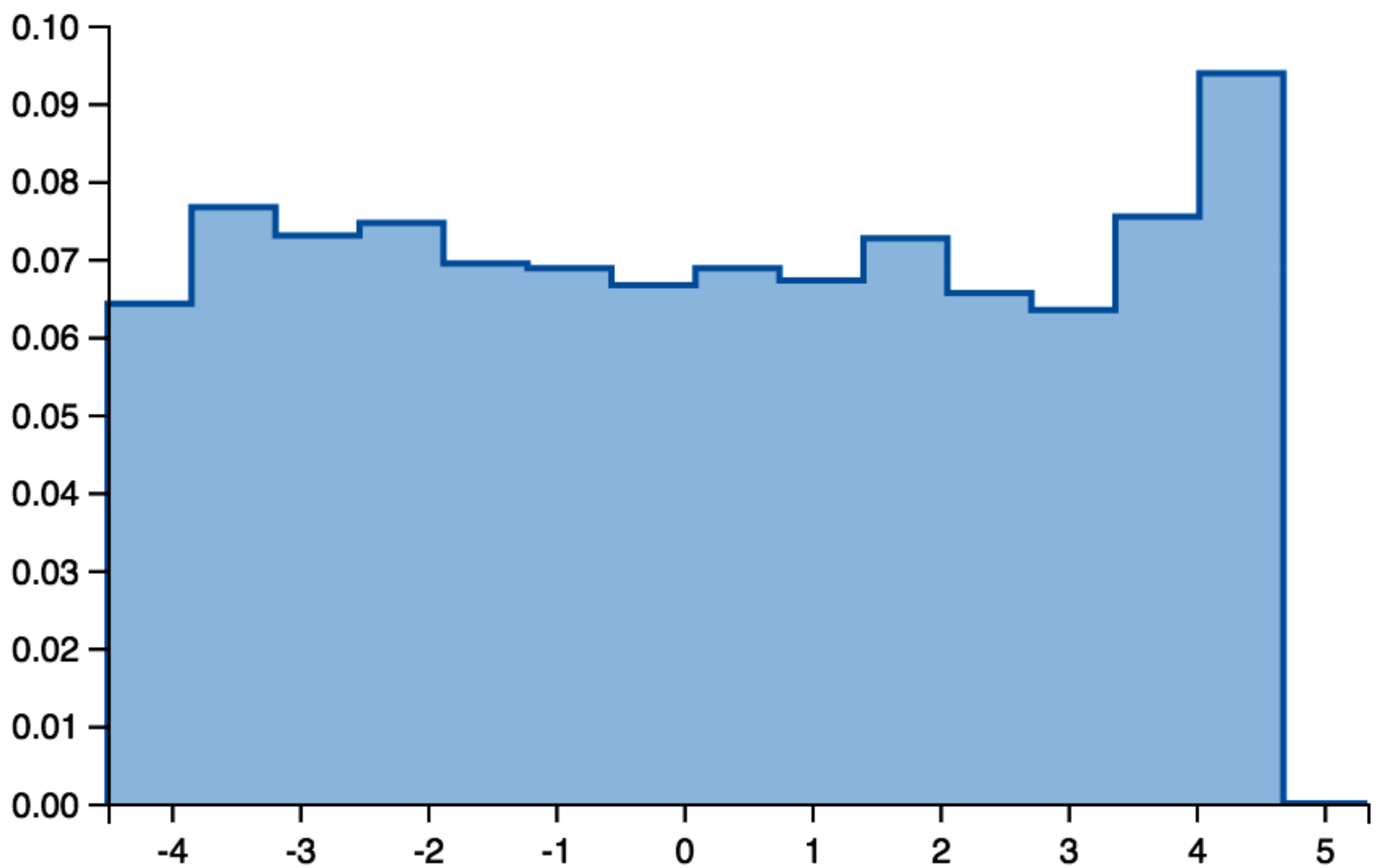
```
(let [x       (sample (normal 0 1))
      x-pdf   (normal-pdf x 0 1)
      y       (observe (exponential (/ 1 x-pdf)) 0)]
  x)
```

```
(def lazy-samples (doquery :lmh example1 []))
(def samples (map :result (take-nth 100 (take 500000 (drop 100000 lazy-samples)))))
(plot/histogram samples :normalize :probability)
```

```
#'uppsala-pp17/lazy-samples

#'uppsala-pp17/samples
```

5K samples



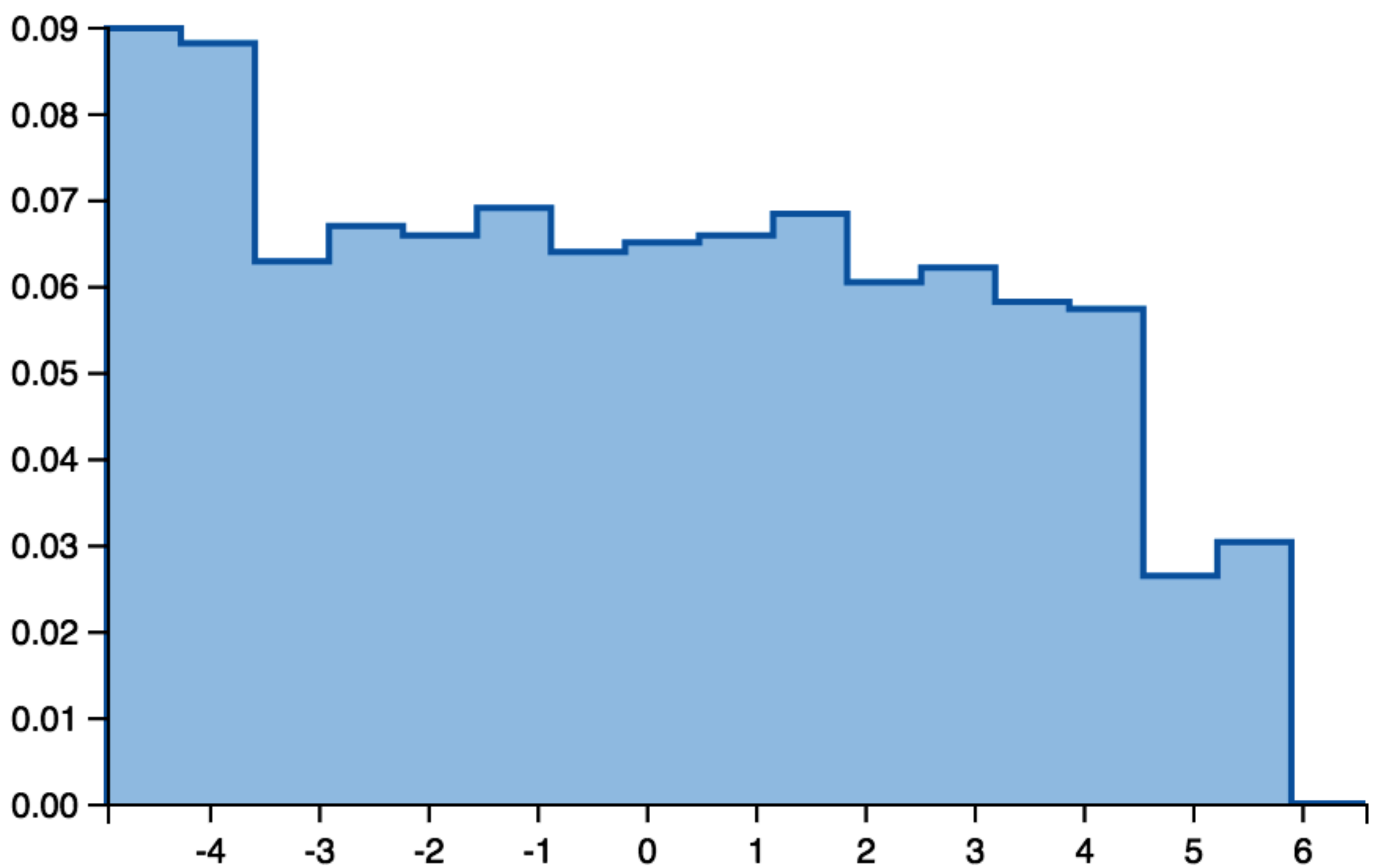
```
(let [x      (sample (normal 0 1))
      x-pdf  (normal-pdf x 0 1)
      y      (observe (exponential (/ 1 x-pdf)) 0)]
  x)
```

```
(def lazy-samples (doquery :lmh example1 []))
(def samples (map :result (take-nth 100 (take 1000000 (drop 100000 lazy-samples)))))
(plot/histogram samples :normalize :probability)
```

#'uppsala-pp17/lazy-samples

#'uppsala-pp17/samples

10K samples



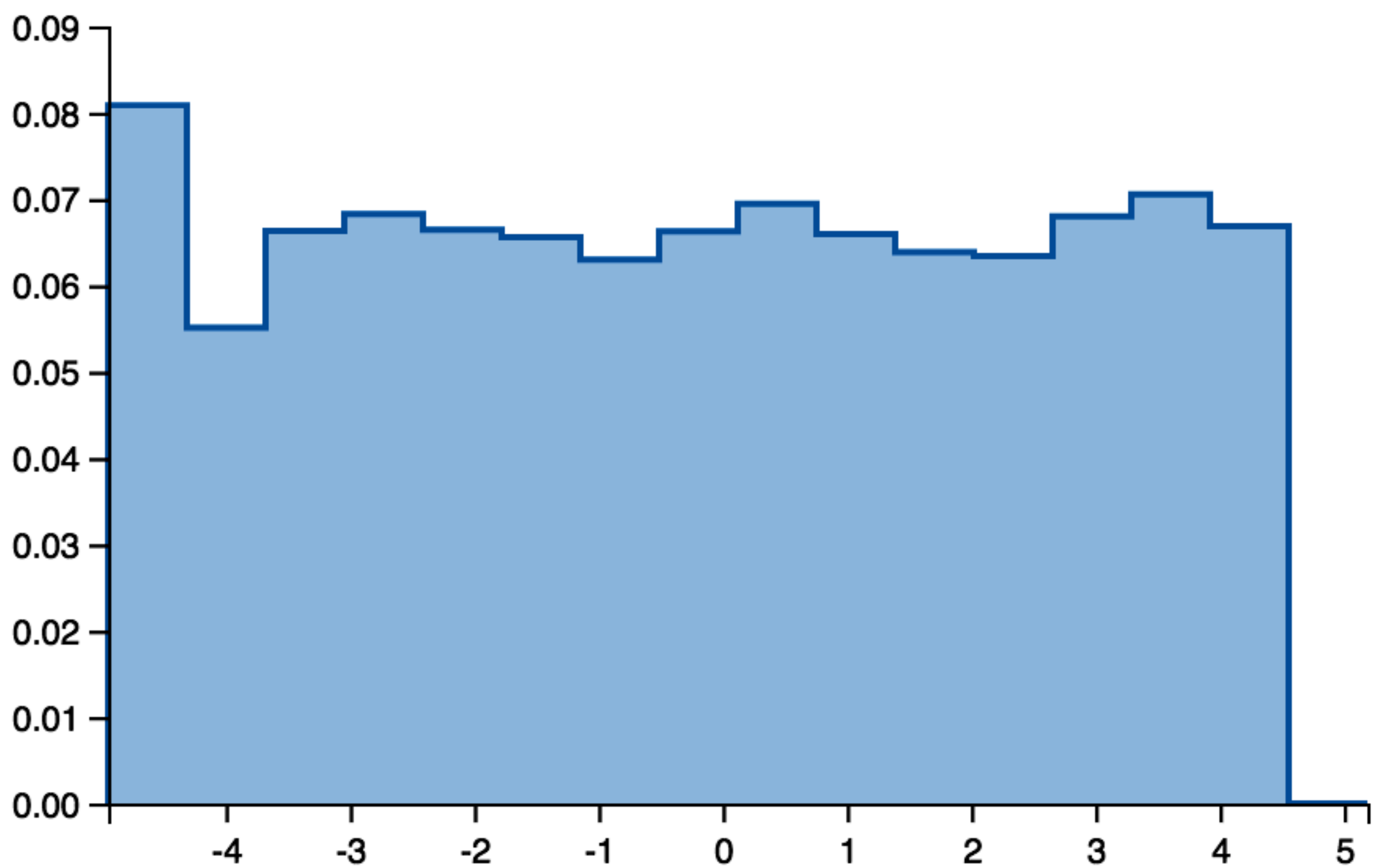
```
(let [x      (sample (normal 0 1))
      x-pdf  (normal-pdf x 0 1)
      y      (observe (exponential (/ 1 x-pdf)) 0)]
  x)
```

```
(def lazy-samples (doquery :lmh example1 []))
(def samples (map :result (take-nth 100 (take 1500000 (drop 100000 lazy-samples)))))
(plot/histogram samples :normalize :probability)
```

#'uppsala-pp17/lazy-samples

#'uppsala-pp17/samples

15K samples



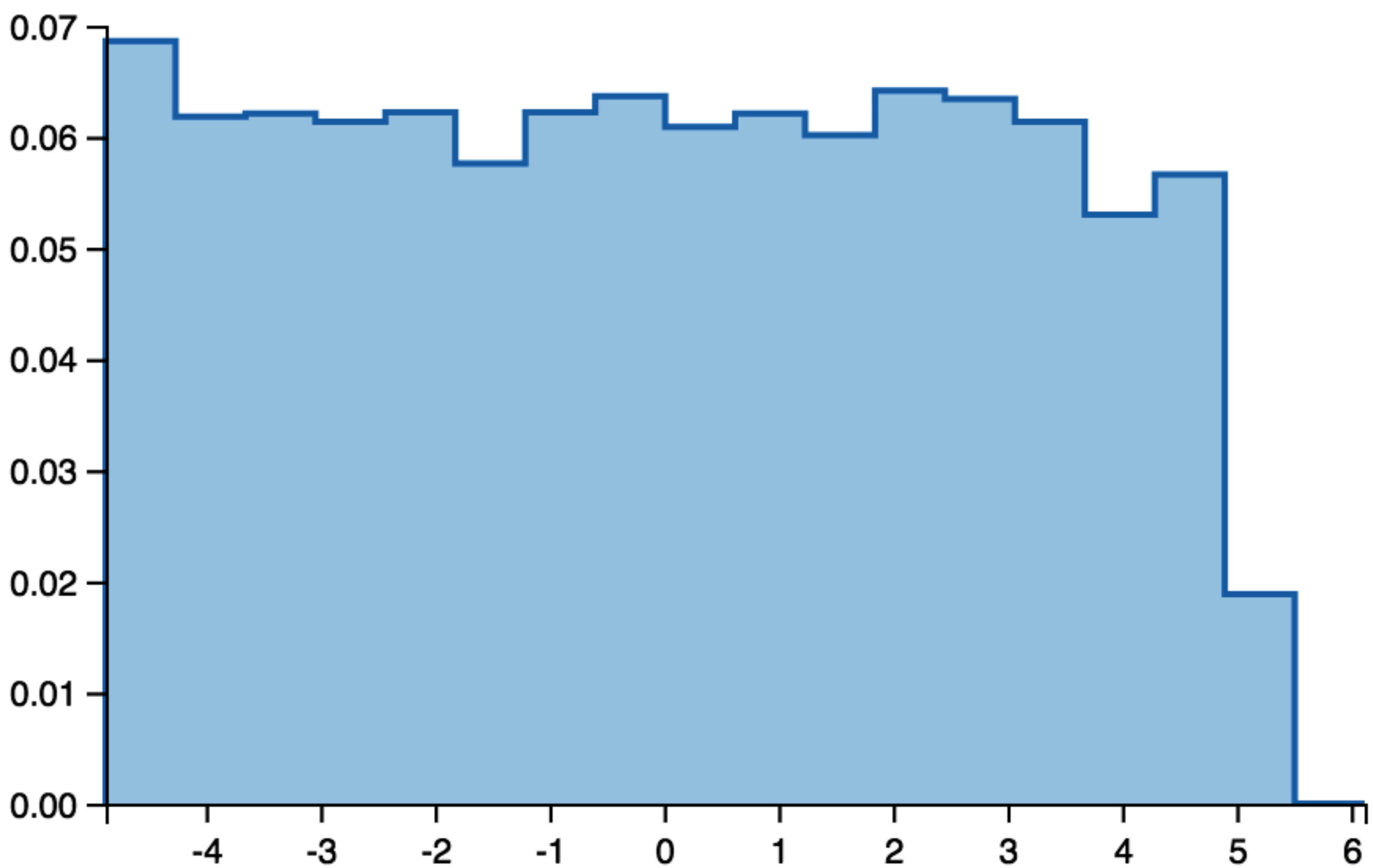
```
(let [x      (sample (normal 0 1))
      x-pdf  (normal-pdf x 0 1)
      y      (observe (exponential (/ 1 x-pdf)) 0)]
  x)
```

```
(def lazy-samples (doquery :lmh example1 []))
(def samples (map :result (take-nth 100 (take 2000000 (drop 100000 lazy-samples)))))
(plot/histogram samples :normalize :probability)
```

#'uppsala-pp17/lazy-samples

#'uppsala-pp17/samples

20K samples



```
(let [x       (sample (normal 0 1))
      x-pdf   (normal-pdf x 0 1)
      y       (observe (exponential (/ 1 x-pdf)) 0)]
  x)
```

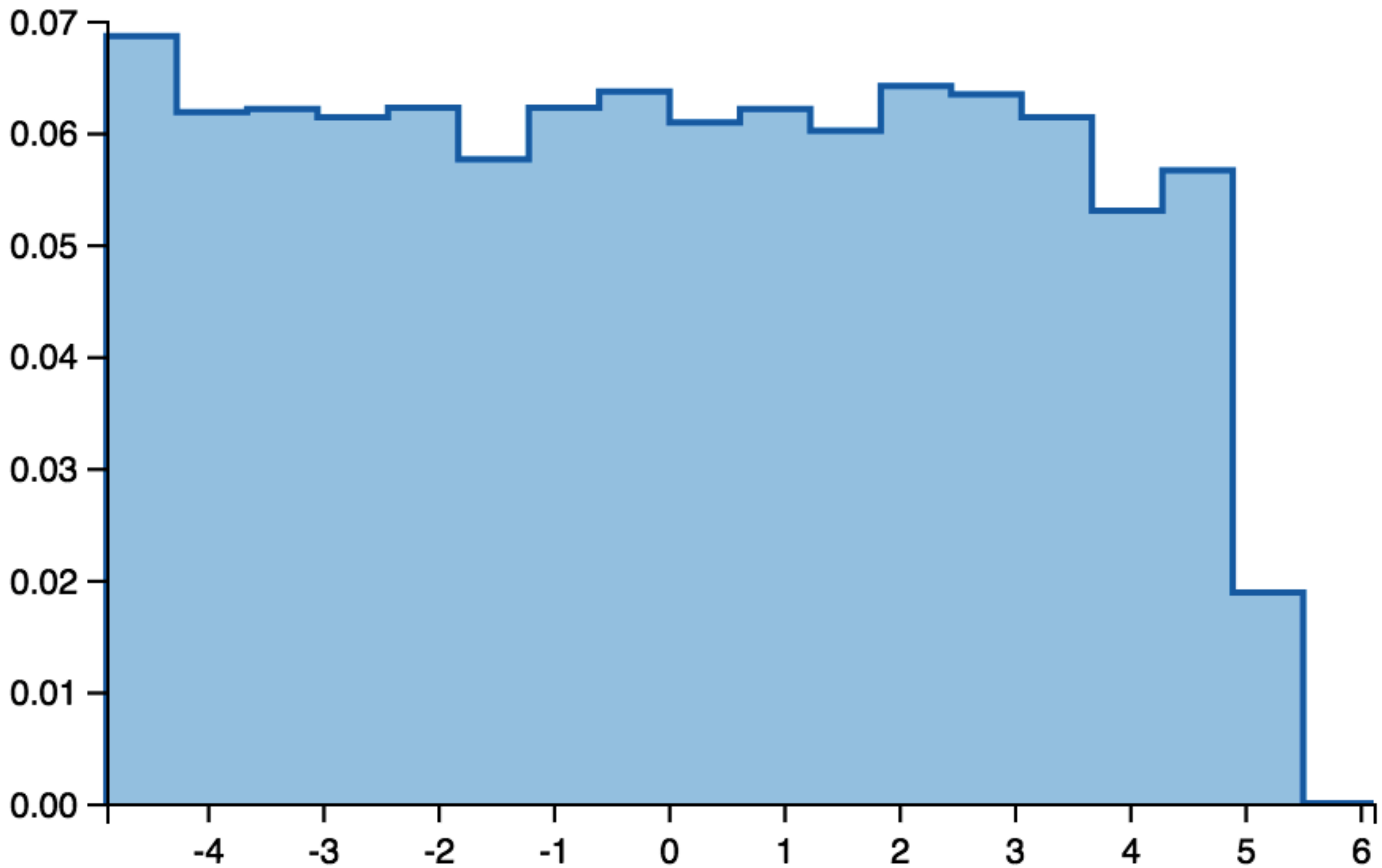
```
(def lazy-samples (doquery :lmh example1 []))
(def samples (map :result (take-nth 100 (take 2000000 (drop 100000 lazy-samples)))))
(plot/histogram samples :normalize :probability)
```

#'uppsala-pp17/lazy-samples

#'uppsala-pp17/samples

20K samples

Uniform[-6,6]?

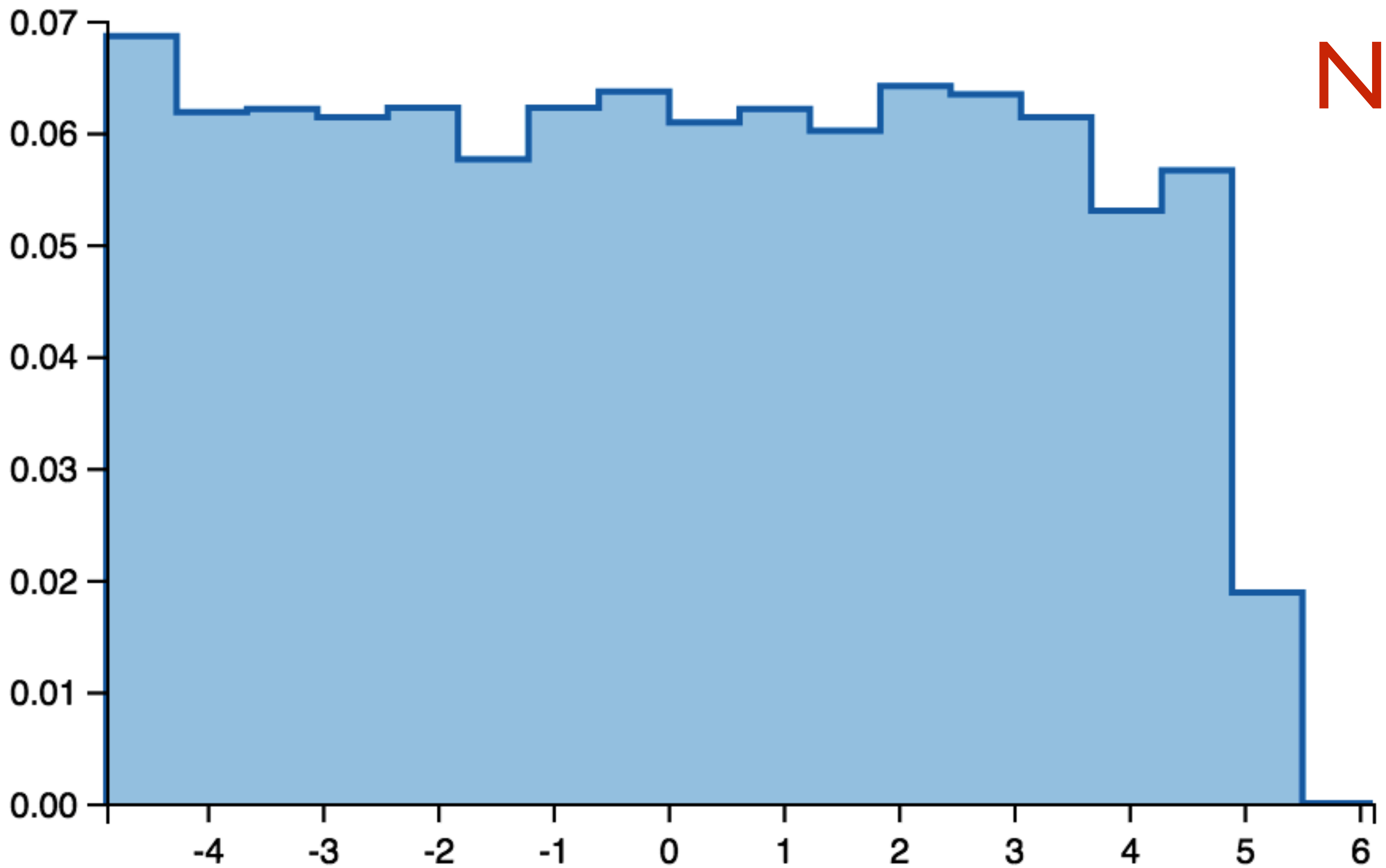


```
(let [x      (sample (normal 0 1))
      x-pdf  (normal-pdf x 0 1)
      y      (observe (exponential (/ 1 x-pdf)) 0)]
  x)
```

```
(def lazy-samples (doquery :lmh example1 []))
(def samples (map :result (take-nth 100 (take 2000000 (drop 100000 lazy-samples)))))
(plot/histogram samples :normalize :probability)
```

#'uppsala-pp17/lazy-samples

#'uppsala-pp17/samples



20K samples

Uniform[-6,6]?

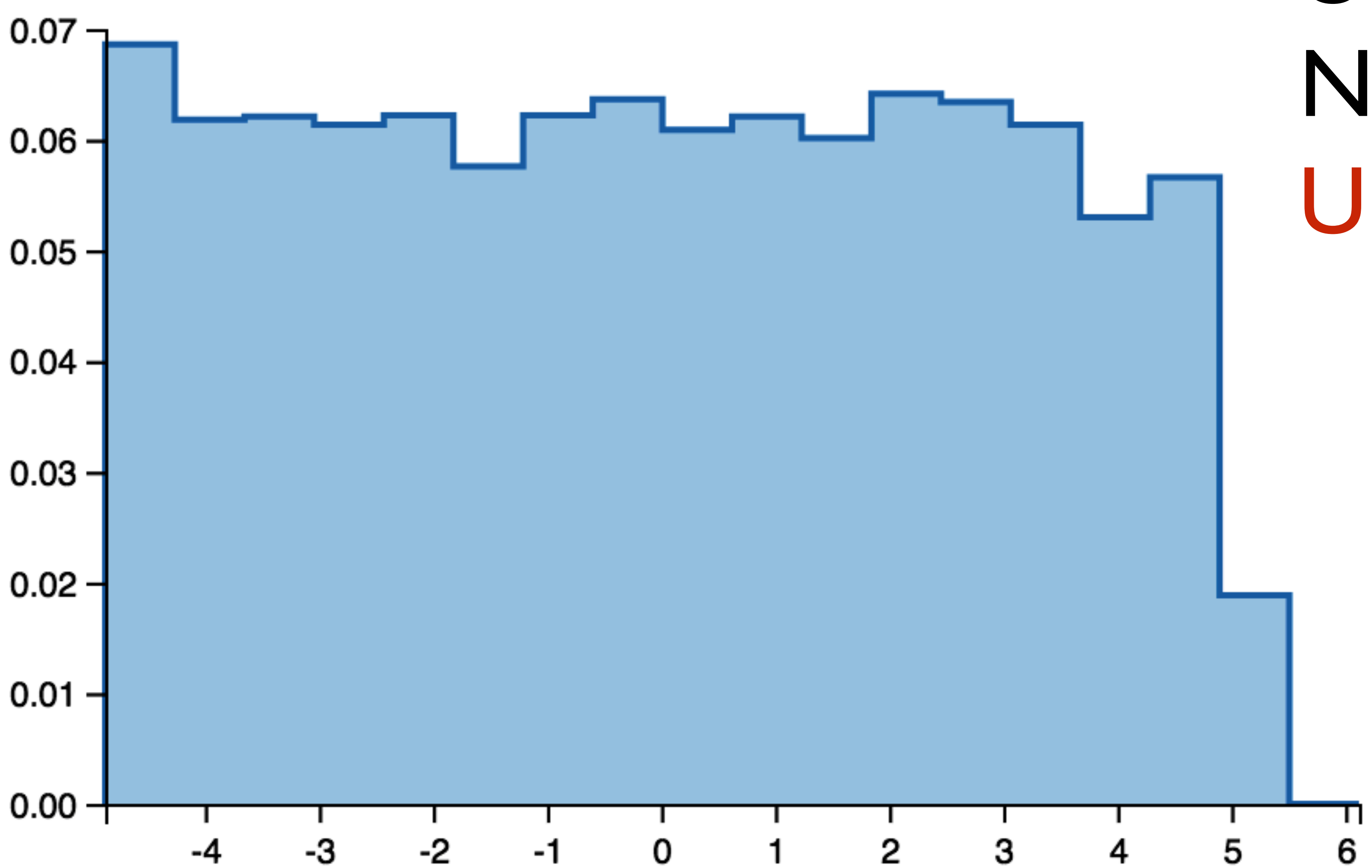
No posterior.

```
(let [x       (sample (normal 0 1))
      x-pdf   (normal-pdf x 0 1)
      y       (observe (exponential (/ 1 x-pdf)) 0)]
  x)
```

```
(def lazy-samples (doquery :lmh example1 []))
(def samples (map :result (take-nth 100 (take 2000000 (drop 100000 lazy-samples)))))
(plot/histogram samples :normalize :probability)
```

#'uppsala-pp17/lazy-samples

#'uppsala-pp17/samples



20K samples

Uniform[-6,6]?

No posterior.

Uniform[-∞,∞].

```
(let [x        (sample (normal 0 1))
      x-pdf   (normal-pdf x 0 1)
      y        (observe (exponential (/ 1 x-pdf)) 0)]
  x)
```

p(x) = normal-pdf(x;0,1)

```
(let [x       (sample (normal 0 1))
      x-pdf   (normal-pdf x 0 1)
      y       (observe (exponential (/ 1 x-pdf)) 0)]
  x)
```

p(x) = normal-pdf(x;0,1)

p(y=0|x) = exponential-pdf(y=0;1/p(x))

```
(let [x       (sample (normal 0 1))
      x-pdf   (normal-pdf x 0 1)
      y       (observe (exponential (/ 1 x-pdf)) 0)]
  x)
```

$p(x) = \text{normal-pdf}(x;0,1)$

$p(y=0|x) = \text{exponential-pdf}(y=0;1/p(x)) = 1/p(x)$

```
(let [x       (sample (normal 0 1))
      x-pdf   (normal-pdf x 0 1)
      y       (observe (exponential (/ 1 x-pdf)) 0)]
  x)
```

p(x) = normal-pdf(x;0,1)

p(y=0|x) = exponential-pdf(y=0;1/p(x)) = 1/p(x)

p(x,y=0) = p(x) * p(y=0|x)

```
(let [x      (sample (normal 0 1))
      x-pdf  (normal-pdf x 0 1)
      y      (observe (exponential (/ 1 x-pdf)) 0)]
  x)
```

p(x) = normal-pdf(x;0,1)

p(y=0|x) = exponential-pdf(y=0;1/p(x)) = 1/p(x)

p(x,y=0) = p(x) * p(y=0|x) = p(x) * 1/p(x) = 1

```
(let [x       (sample (normal 0 1))
      x-pdf   (normal-pdf x 0 1)
      y       (observe (exponential (/ 1 x-pdf)) 0)]
  x)
```

p(x) = normal-pdf(x;0,1)

p(y=0|x) = exponential-pdf(y=0;1/p(x)) = 1/p(x)

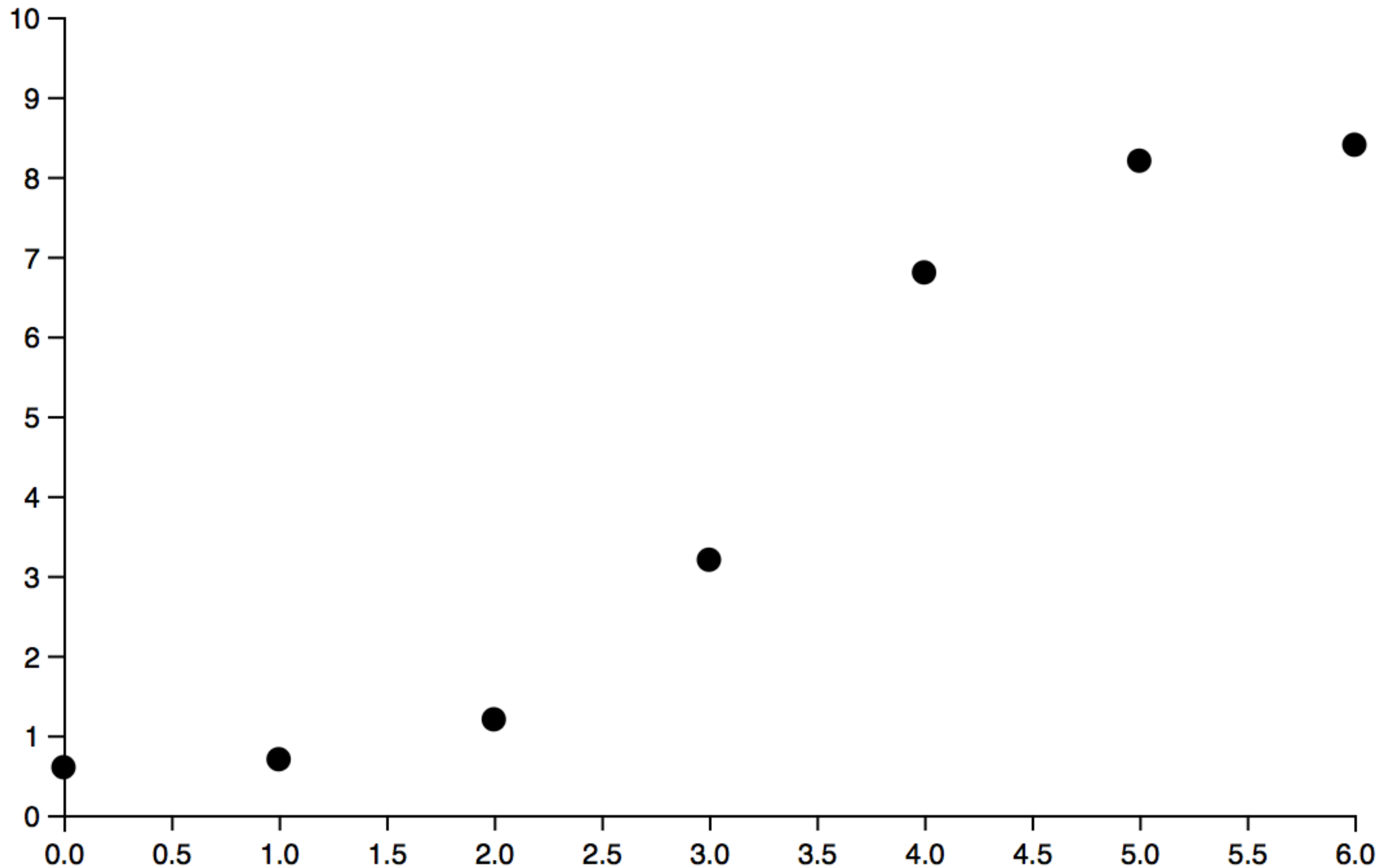
p(x,y=0) = p(x) * p(y=0|x) = p(x) * 1/p(x) = 1

p(y=0) = ∫p(x,y=0)dx = ∫dx = ∞

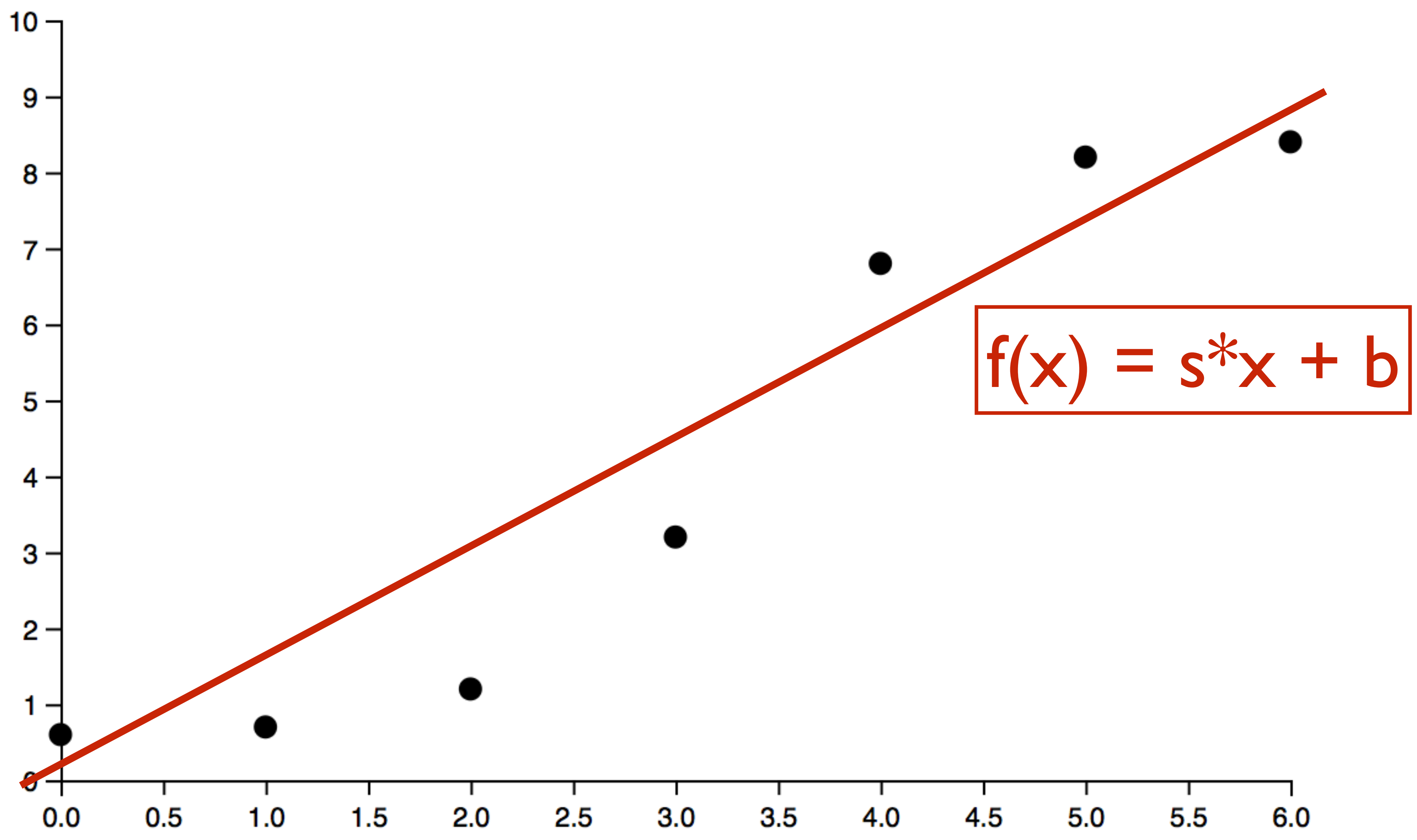
Why should one care about deno. semantics?

1. Specification of inference algorithms.
2. Compiler optimisation.
3. Detection of ill-defined models.
4. Clear meaning of complex models.

Line fitting



Line fitting



Anglican program

```
(defquery lin-regression []
```

Anglican program

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

Anglican program

```
(defquery lin-regression []
  (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)
```

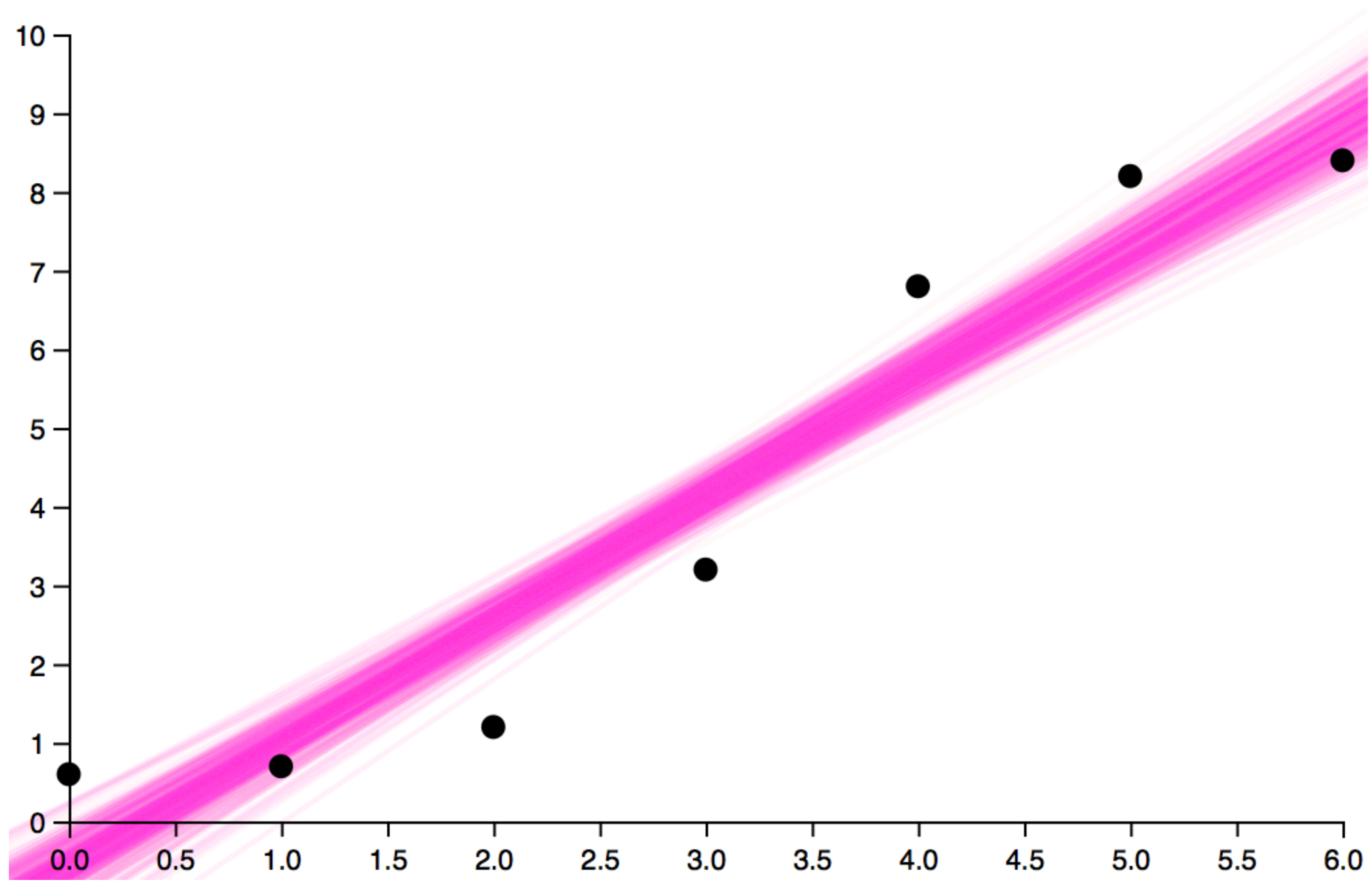
Anglican program

```
(defquery lin-regression []
  (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)

    f))
```

Samples from posterior



```
(defquery lin-regression []
  (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)

    f))
```

[Q] Which posterior?

```
(defquery lin-regression []
  (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)

    f))
```

[Q] Which posterior?

Inference algo. gives only approximation.

```
(defquery lin-regression []
  (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)

    f))
```

[Q] Which posterior?

Inference algo. gives only approximation.

Should define distr. on functions. Not easy.

Foundational question

Measure theory provides a foundation for modern probability theory.

But it doesn't support higher-order 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 $\mathbb{R} \to_m \mathbb{R}$.

```
(defquery lin-regression []
  (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)

    f))
```

[Q] Which posterior?

Inference algo. gives only approximation.

Should define distr. on functions. Not easy.

Denotational semantics: Compositional method. Answers a deep Q.

Learning outcome

- Can define a denotational semantics for a simple programming language.
- Can use measure theory to interpret prob. prog. with continuous distributions.
- Can use quasi-Borel space to interpret higher-order prob. prog.

References

1. A convenient category for higher-order probability theory. Heunen et al. LICS'17.
2. Commutative semantics for probabilistic programs. Staton. ESOP'17.
3. Reynolds's "Theories of Programming Languages".
4. Billingsley's "Probability and Measure".

Plan for the rest

1. Denotational semantics.
   PL with discrete random choices.
2. Baby measure theory.
   PL with cont. distribution.
3. Quasi-Borel space (QBS).
   PL with cont. distr. & higher-order (HO) fns.
4. [To be skipped] SFinKer monad on QBS.
   PL with cont. distr., HO fns & conditioning.

Plan for the rest

1. Denotational semantics.
   PL with discrete random choices.

2. Baby measure theory.
   PL with cont. distribution.

3. Quasi-Borel space (QBS).
   PL with cont. distr. & higher-order (HO) fns.

4. [To be skipped] SFinKer monad on QBS.
   PL with cont. distr., HO fns & conditioning.

ill-defined model

Plan for the rest

1. Denotational semantics.
   PL with discrete random choices.
2. Baby measure theory.
   PL with cont. distribution.
3. Quasi-Borel space (QBS).
   PL with cont. distr. & higher-order (HO) fns.

   linear regression example
4. [To be skipped] SFinKer monad on QBS.
   PL with cont. distr., HO fns & conditioning.

ill-defined model

Plan for the rest

1. Denotational semantics.
   PL with discrete random choices.

2. Baby measure theory.
   PL with cont. distribution.

3. Quasi-Borel space (QBS).
   PL with cont. distr. & higher-order (HO) fns.

4. [To be skipped] SFinKer monad on QBS.
   PL with cont. distr., HO fns & conditioning.

Denotational semantics

- Defines formal meanings of programs.
- Interprets type and expr. mathematically.
  - Type as space (e.g. set, measurable space).
  - Expr. as good function between spaces.
- Used for justifying compiler optimisation, inference algorithms and language constructs.

Denotational semantics

- Defines formal meanings of programs.
- Interprets type and expr. mathematically.
    - Type as space (e.g. set, measurable space).
    - Expr. as good function between spaces.
- Used for justifying compiler optimisation, inference algorithms and language constructs.

First-order PL with discrete random choices

t ::= bool | rational | dist[bool] | dist[rational]

e ::= c | x | (p e ... e) | (let [x e] e) | (if e e e)

c ::= true | false | 0 | 1 | ...

p ::= sample | flip | poisson | and | + | ...

First-order PL with discrete random choices

```
t  ::=  bool | rational | dist[bool] | dist[rational]

e  ::=  c | x | (p e ... e) | (let [x e] e) | (if e e e)

c  ::=  true | false | 0 | 1 | ...

p  ::=  sample | flip | poisson | and | + | ...
```

First-order PL with discrete random choices

```
t  ::=  bool | rational | dist[bool] | dist[rational]

e  ::=  c | x | (p e ... e) | (let [x e] e) | (if e e e)

c  ::=  true | false | 0 | 1 | ...

p  ::=  sample | flip | poisson | and | + | ...
```

No function type.

First-order PL with discrete random choices

t ::= bool | rational | dist[bool] | dist[rational]

e ::= c | x | (p e … e) | (let [x e] e) | (if e e e)

c ::= true | false | 0 | 1 | …

p ::= sample | flip | poisson | and | + | …

No function type.

No (fn [x … x] e) case.

First-order PL with discrete random choices

```
t  ::=  bool | rational | dist[bool] | dist[rational]

e  ::=  c | x | (p e ... e) | (let [x e] e) | (if e e e)

c  ::=  true | false | 0 | 1 | ...

p  ::=  sample | flip | poisson | and | + | ...
```

No function type.

No (fn [x ... x] e) case.

First-order PL with discrete random choices

t ::= bool | rational | dist[bool] | dist[rational]

e ::= c | x | (p e ... e) | (let [x e] e) | (if e e e)

c ::= true | false | 0 | 1 | ...

p ::= sample | flip | poisson | and | + | ...

No function type.
No (fn [x ... x] e) case.
Only primitive functions can be applied.

First-order PL with discrete random choices

t ::= bool | rational | dist[bool] | dist[rational]

e ::= c | x | (p e ... e) | (let [x e] e) | (if e e e)

c ::= true | false | 0 | 1 | ...

p ::= sample | flip | poisson | and | + | ...

First-order PL with discrete random choices

```
t  ::=  bool | rational | dist[bool] | dist[rational]

e   ::=  c | x | (p e ... e) | (let [x e] e) | (if e e e)

c   ::=  true | false  | 0 | 1 | ...

p   ::=  sample | flip | poisson | and | + | ...
```

First-order PL with discrete random choices

t ::= bool | rational | dist[bool] | dist[rational]

e ::= c | x | (p e ... e) | (let [x e] e) | (if e e e)

c ::= true | false | 0 | 1 | ...

p ::= sample | flip | poisson | and | + | ...

First-order PL with discrete random choices

```
t  ::=  bool | rational | dist[bool] | dist[rational]

e  ::=  c | x | (p e ... e) | (let [x e] e) | (if e e e)

c  ::=  true | false | 0 | 1 | ...

p  ::=  sample | flip | poisson | and | + | ...
```

[Q] Denotational semantics of this PL?

First-order PL with discrete random choices

t ::= bool | rational | dist[bool] | dist[rational]

e ::= c | x | (p e ... e) | (let [x e] e) | (if e e e)

c ::= true | false | 0 | 1 | ...

p ::= sample | flip | poisson | and | + | ...

[Q] Denotational semantics of this PL?

Interpret type as set and expr. as function.

Types mean sets

$[\![\mathtt{t}]\!]$ is the meaning of t.

$$[\![\mathtt{bool}]\!] = \ldots$$

$$[\![\mathtt{rational}]\!] = \ldots$$

$$[\![\mathtt{dist[bool]}]\!] = \ldots$$

$$[\![\mathtt{dist[rational]}]\!] = \ldots$$

Types mean sets

$[\![\texttt{t}]\!]$ is the meaning of t.

$[\![\texttt{bool}]\!] = \mathbb{B} = \{tt, ff\}$

$[\![\texttt{rational}]\!] = \ldots$

$[\![\texttt{dist[bool]}]\!] = \ldots$

$[\![\texttt{dist[rational]}]\!] = \ldots$

Types mean sets

$[\![\mathtt{t}]\!]$ is the meaning of t.

$[\![\mathtt{bool}]\!] = \mathbb{B} = \{tt, ff\}$

$[\![\mathtt{rational}]\!] = \ldots$

$[\![\mathtt{dist[bool]}]\!] = \{p{:}\mathbb{B}\to[0,1] \mid p(tt)+p(ff)=1\}$

$[\![\mathtt{dist[rational]}]\!] = \ldots$

Types mean sets

$[\![t]\!]$ is the meaning of t.

$[\![\mathrm{bool}]\!] = \mathbb{B} = \{\mathrm{tt}, \mathrm{ff}\}$

$[\![\mathrm{rational}]\!] = \ldots$

$[\![\mathrm{dist}[\mathrm{bool}]]\!] = \{p{:}\mathbb{B}\to[0,1] \mid p(\mathrm{tt})+p(\mathrm{ff})=1\}$

$[\![\mathrm{dist}[\mathrm{rational}]]\!] = \ldots$

[Q] Fill in . . .

Types mean sets

$[\![t]\!]$ is the meaning of t.

$[\![\text{bool}]\!] = \mathbb{B} = \{\text{tt}, \text{ff}\}$

$[\![\text{rational}]\!] = \mathbb{Q} = \{0, 1, -1/3, 1/7, \ldots\}$

$[\![\text{dist[bool]}]\!] = \{p{:}\mathbb{B}\rightarrow[0,1] \mid p(\text{tt})+p(\text{ff})=1\}$

$[\![\text{dist[rational]}]\!] = \{p{:}\,\mathbb{Q}\rightarrow[0,1] \mid \sum_r p(r)=1\}$

[Q] Fill in . . .

Types mean sets

$[\![t]\!]$ is the meaning of t.

$[\![\text{bool}]\!] = \mathbb{B} = \{tt, ff\}$

$[\![\text{rational}]\!] = \mathbb{Q} = \{0, 1, -1/3, 1/7, \ldots\}$

$[\![\text{dist[bool]}]\!] = \{p{:}\mathbb{B}\to[0,1] \mid p(tt)+p(ff)=1\}$

$[\![\text{dist[rational]}]\!] = \{p{:}\ \mathbb{Q}\to[0,1] \mid \sum_r p(r)=1\}$

[Q] Fill in . . .

Types mean sets

$[\![\mathtt{t}]\!]$ is the meaning of $\mathtt{t}$.

$[\![\mathtt{bool}]\!] = \mathbb{B} = \{tt, ff\}$

$[\![\mathtt{rational}]\!] = \mathbb{Q} = \{0, 1, -1/3, 1/7, \ldots\}$

$[\![\mathtt{dist[bool]}]\!] = \mathrm{DiscProb}([\![\mathtt{bool}]\!])$

$[\![\mathtt{dist[rational]}]\!] = \mathrm{DiscProb}([\![\mathtt{rational}]\!])$

[Q] Fill in . . .

Typed expressions

$$\mathtt{x_1{:}t_1,\ x_2{:}t_2,\ \ldots,\ x_n{:}t_n} \vdash \mathtt{e : t}$$

Typed expressions

$$\underbrace{\mathtt{x_1{:}t_1,\ x_2{:}t_2,\ \ldots,\ x_n{:}t_n}}^{\text{typing context } \Gamma} \vdash \mathtt{e : t}$$

- Γ is a finite map from variables to types.

Typed expressions

$$\underbrace{\mathtt{x}_1\mathtt{:t}_1,\ \mathtt{x}_2\mathtt{:t}_2,\ \ldots,\ \mathtt{x}_n\mathtt{:t}_n}^{\text{typing context } \Gamma} \vdash \mathtt{e : t}$$

- Γ is a finite map from variables to types.
- Denotes a $\mathtt{t}$-typed expr. $\mathtt{e}$ under Γ.

Typed expressions

$$\underbrace{x_1{:}t_1,\ x_2{:}t_2,\ \ldots,\ x_n{:}t_n}^{\text{typing context }\Gamma} \vdash e : t$$

- Γ is a finite map from variables to types.
- Denotes a t-typed expr. e under Γ.

x:bool, y:bool ⊢ (if x y y): bool

x:bool, y:rational ⊢ (if x y y) : rational

x:rational ⊢ (sample (flip x)) : bool

Typed expressions mean fns to discrete prob. distr.

Typed expressions mean fns to discrete prob. distr.

$[\![\mathtt{x_1{:}t_1, \ldots, x_n{:}t_n \vdash e : t}]\!]$

Typed expressions mean fns to discrete prob. distr.

$$[\![\mathtt{x}_1\mathtt{:t}_1, \ldots, \mathtt{x}_n\mathtt{:t}_n \vdash \mathtt{e} : \mathtt{t}]\!]$$
$$= g : [\![\mathtt{x}_1\mathtt{:t}_1, \ldots, \mathtt{x}_n\mathtt{:t}_n]\!] \rightarrow \mathrm{DiscProb}([\![\mathtt{t}]\!])$$

Typed expressions mean fns to discrete prob. distr.

$$[\![x_1{:}t_1, \ldots, x_n{:}t_n \vdash e : t]\!]$$
$$= g : [\![x_1{:}t_1, \ldots, x_n{:}t_n]\!] \to \mathrm{DiscProb}([\![t]\!])$$

1. $[\![x_1{:}t_1, \ldots, x_n{:}t_n]\!] =$
   $\{\text{map } \eta \text{ from } \{x_1,\ldots,x_n\} \mid \eta(x_i) \in [\![t_i]\!] \text{ for all } i\}$

Typed expressions mean fns to discrete prob. distr.

$[\![x_1{:}t_1, \ldots, x_n{:}t_n \vdash e : t]\!]$

$= \; g : [\![x_1{:}t_1, \ldots, x_n{:}t_n]\!] \to \mathrm{DiscProb}([\![t]\!])$

1. $[\![x_1{:}t_1, \ldots, x_n{:}t_n]\!] =$
   $\{\text{map } \eta \text{ from } \{x_1,\ldots,x_n\} \mid \eta(x_i) \in [\![t_i]\!] \text{ for all } i\}$
2. $\mathrm{DiscProb}(A) = \{p{:}A \to [0,1] \mid \sum_a p(a) = 1\}$

Typed expressions mean fns to discrete prob. distr.

$$[\![x_1{:}t_1, \ldots, x_n{:}t_n \vdash e : t]\!]$$
$$= \; g : [\![x_1{:}t_1, \ldots, x_n{:}t_n]\!] \to \mathrm{DiscProb}([\![t]\!])$$

1. $[\![x_1{:}t_1, \ldots, x_n{:}t_n]\!] =$
   $\{\text{map } \eta \text{ from } \{x_1,\ldots,x_n\} \mid \eta(x_i) \in [\![t_i]\!] \text{ for all } i\}$
2. $\mathrm{DiscProb}(A) = \{p{:}A \to [0,1] \mid \sum_a p(a) = 1\}$

[Q] Any problem with $\mathrm{DiscProb}([\![\mathtt{dist[bool]}]\!])$?

Typed expressions mean fns to discrete prob. distr.

$$[\![x_1{:}t_1, \ldots, x_n{:}t_n \vdash e : t]\!]$$
$$= \; g : [\![x_1{:}t_1, \ldots, x_n{:}t_n]\!] \to \mathrm{DiscProb}([\![t]\!])$$

1. $[\![x_1{:}t_1, \ldots, x_n{:}t_n]\!] =$
   $\{$map η from $\{x_1,\ldots,x_n\} \mid \eta(x_i) \in [\![t_i]\!]$ for all $i\}$

2. $\mathrm{DiscProb}(A) = \{p{:}A \to [0,1] \mid \sum_a p(a) = 1\}$

[Q] Any problem with $\mathrm{DiscProb}([\![\mathtt{dist[bool]}]\!])$?

Typed expressions mean fns to discrete prob. distr.

$$[\![x_1:t_1, \ldots, x_n:t_n \vdash e : t]\!]$$
$$= g : [\![x_1:t_1, \ldots, x_n:t_n]\!] \to \mathrm{DiscProb}([\![t]\!])$$

1. $[\![x_1:t_1, \ldots, x_n:t_n]\!] =$
   $\{\text{map } \eta \text{ from } \{x_1,\ldots,x_n\} \mid \eta(x_i) \in [\![t_i]\!] \text{ for all } i\}$

2. $\mathrm{DiscProb}(A) = \{p{:}A \to [0,1] \mid \sum_a p(a)=1\}$

   p(a)=0 except for countably many a's and

[Q] Any problem with $\mathrm{DiscProb}([\![\mathtt{dist[bool]}]\!])$?

Typed expressions mean fns to discrete prob. distr.

$[\![x_1{:}t_1, \ldots, x_n{:}t_n \vdash e : t]\!]$

$= g : [\![x_1{:}t_1, \ldots, x_n{:}t_n]\!] \to \mathrm{DiscProb}([\![t]\!])$

1. $[\![x_1{:}t_1, \ldots, x_n{:}t_n]\!] =$
   $\{\text{map } \eta \text{ from } \{x_1, \ldots, x_n\} \mid \eta(x_i) \in [\![t_i]\!] \text{ for all } i\}$

2. $\mathrm{DiscProb}(A) = \{p{:}A \to [0,1] \mid \sum_a p(a) = 1\}$

p(a)=0 except for countably many a's and

[Q] Define the interpretation recursively.

Compiler optimisation

Show the following equations:

$[\![\Gamma \vdash (\texttt{if true } e_1 \ e_2) : t]\!] = [\![\Gamma \vdash e_1 : t]\!]$

$[\![\Gamma \vdash (\texttt{sample (flip (+ 0.1 0.2))}) : \texttt{bool}]\!]$
$= [\![\Gamma \vdash (\texttt{sample (flip 0.3)}) : \texttt{bool}]\!]$

Plan for the rest

1. Denotational semantics.
   PL with discrete random choices.

2. Baby measure theory.
   PL with cont. distribution.

3. Quasi-Borel space (QBS).
   PL with cont. distr. & higher-order (HO) fns.

4. [To be skipped] SFinKer monad on QBS.
   PL with cont. distr., HO fns & conditioning.

First-order PL with discrete random choices

```
t  ::=  bool | rational | dist[bool] | dist[rational]

e  ::=  c | x | (p e ... e) | (let [x e] e) | (if e e e)

c  ::=  true | false | 0 | 1 | ...

p  ::=  sample | flip | poisson | and | + | ...
```

First-order PL with discrete random choices

and continuous

t ::= bool | ~~rational~~ real | dist[bool] | dist[~~rational~~ real]

e ::= c | x | (p e … e) | (let [x e] e) | (if e e e)

c ::= true | false | 0 | 1 | …

p ::= sample | flip | poisson | and | + | …
| normal | uniform-continuous | …

How to define denotational semantics?

Types as spaces and expressions as functions.

How to define denotational semantics?

Types as spaces and expressions as functions.

$$\mathtt{t} \ ::= \ \mathtt{bool} \mid \mathtt{real} \mid \mathtt{dist[bool]} \mid \mathtt{dist[real]}$$

[Try] Interpret $[\![\mathtt{t}]\!]$ as a set.

How to define denotational semantics?

Types as spaces and expressions as functions.

$$\mathtt{t} \ ::= \ \mathtt{bool} \mid \mathtt{real} \mid \mathtt{dist[bool]} \mid \mathtt{dist[real]}$$

[Try] Interpret $[\![\mathtt{t}]\!]$ as a set. Then, we get stuck since $[\![\mathtt{dist[real]}]\!]$ can't be DiscProb($[\![\mathtt{real}]\!]$).

How to define denotational semantics?

Types as spaces and expressions as functions.

$$\texttt{t} \ ::= \ \texttt{bool} \mid \texttt{real} \mid \texttt{dist[bool]} \mid \texttt{dist[real]}$$

[Try] Interpret $[\![\texttt{t}]\!]$ as a set. Then, we get stuck since $[\![\texttt{dist[real]}]\!]$ can't be DiscProb($[\![\texttt{real}]\!]$).

[Sol] Use measure theory. $[\![\texttt{t}]\!]$ as a measurable space, and $[\![\Gamma \vdash \texttt{e} : \texttt{t}]\!]$ as a measurable function.

How to specify prob. p?

How to specify prob. p?

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

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

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

How to specify prob. p?

$X = \mathbb{R}$.

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

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

How to specify prob. p?

$X = \mathbb{R}$.

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

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

Uncountable sum.
Typically ∞.

How to specify prob. p?

$X = \mathbb{R}$.

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

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

How to specify prob. p?

X = $\mathbb{R}$.

~~Define p : X→[0,1]~~

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

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

Define p : Σ→[0,1] with some care.

How to specify prob. p?

$X = \mathbb{R}$.

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

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

~~Define~~

σ-algebra

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

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

probability measure

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.

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.

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

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

[Q] What are not 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. $(\mathbb{R}, \{ A \subseteq \mathbb{R} \mid A \text{ or } (\mathbb{R}-A) \text{ finite or countable} \})$.
4. $(\mathbb{R}, \{ (r,s] \mid r<s \})$.

[Q] What are not measurable spaces?

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

[Q] Convert them to measurable spaces.

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

Closure exists.

$\sigma(\mathscr{A})$ smallest σ-algebra containing $\mathscr{A}$.

(X, Σ), (Y, Θ) - mBle spaces.

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

Product space: $(X,\Sigma) x_m (Y, \Theta) = (X x Y, \Sigma \otimes \Theta)$.

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

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

(X, Σ), (Y, Θ) - mBle spaces.

$\Pr(\Sigma) = \ \ldots$

Probability space: $\Pr(X,\Sigma) = (\Pr(X), \Pr(\Sigma))$

[Q] What is $\Pr(\Sigma)$?

(X, Σ), (Y, Θ) - mBle spaces.

$Pr(\Sigma) = \sigma\{ \{p \mid p(A)<r\} \mid A\in\Sigma, r\in\mathbb{R} \}$.

Probability space: $Pr(X,\Sigma) = (Pr(X), Pr(\Sigma))$

[Q] What is $Pr(\Sigma)$?

Types mean mBle spaces

Types mean mBle spaces

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

$$[\![\texttt{dist[bool]}]\!] = \mathrm{Pr}([\![\texttt{bool}]\!])$$

Types mean mBle spaces

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

$$[\![\texttt{real}]\!] = \ \ldots$$

$$[\![\texttt{dist[bool]}]\!] = \mathrm{Pr}([\![\texttt{bool}]\!])$$

$$[\![\texttt{dist[real]}]\!] = \ \ldots$$

[Q] Fill in ...

Types mean mBle spaces

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

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

$$[\![\mathtt{dist}[\mathtt{bool}]]\!] = \mathrm{Pr}([\![\mathtt{bool}]\!])$$

$$[\![\mathtt{dist}[\mathtt{real}]]\!] = \mathrm{Pr}([\![\mathtt{real}]\!])$$

[Q] Fill in …

Types mean mBle spaces

$$[\![\mathtt{x}_1\mathtt{:t}_1, \ldots, \mathtt{x}_n\mathtt{:t}_n]\!] = (X, \Sigma)$$

Types mean mBle spaces

$$[\![x_1 : t_1, \ldots, x_n : t_n]\!] = (X, \Sigma)$$

where

$(X_i, \Sigma_i) = [\![t_i]\!]$

$X = \;\ldots$

$\Sigma = \;\ldots$

Types mean mBle spaces

$$[\![\mathtt{x}_1{:}\mathtt{t}_1, \ldots, \mathtt{x}_n{:}\mathtt{t}_n]\!] = (X, \Sigma)$$

where

$(X_i, \Sigma_i) = [\![\mathtt{t}_i]\!]$

$X = \{\text{map } \eta \text{ from } \{\mathtt{x}_1,\ldots,\mathtt{x}_n\} \mid \eta(\mathtt{x}_i) \in X_i \text{ for all } i\}$

$\Sigma = \ \ldots$

Types mean mBle spaces

$$[\![\mathtt{x}_1{:}\mathtt{t}_1, \ldots, \mathtt{x}_n{:}\mathtt{t}_n]\!] = (X, \Sigma)$$

where

$(X_i, \Sigma_i) = [\![\mathtt{t}_i]\!]$

$X = \{\text{map } \eta \text{ from } \{\mathtt{x}_1,\ldots,\mathtt{x}_n\} \mid \eta(\mathtt{x}_i) \in X_i \text{ for all } i\}$

$\Sigma = \;\ldots$

[Q] Fill in ...

Types mean mBle spaces

$$[\![\mathtt{x}_1{:}\mathtt{t}_1, \ldots, \mathtt{x}_n{:}\mathtt{t}_n]\!] = (X, \Sigma)$$

where

$(X_i, \Sigma_i) = [\![\mathtt{t}_i]\!]$

$X = \{\text{map } \eta \text{ from } \{\mathtt{x}_1,\ldots,\mathtt{x}_n\} \mid \eta(\mathtt{x}_i) \in X_i \text{ for all } i\}$

$\Sigma = \sigma\{ \{\eta \mid \eta(\mathtt{x}_i) \in A_i \text{ for all } i\} \mid A_i \in \Sigma_i \text{ for all } i\} \}$

[Q] Fill in …

(X, Σ), (Y, Θ) - mBle spaces.

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

Exprs mean mBle fns

$[\![\Gamma \vdash \mathtt{e} : \mathtt{t}]\!]$ is a mBle fn from $[\![\Gamma]\!]$ to $\mathrm{Pr}[\![\mathtt{t}]\!]$.

Exprs mean mBle fns

$[\![\Gamma \vdash \mathtt{e} : \mathtt{t}]\!]$ is a mBle fn from $[\![\Gamma]\!]$ to Pr$[\![\mathtt{t}]\!]$.

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

Exprs mean mBle fns

$[\![\Gamma \vdash \texttt{e} : \texttt{t}]\!]$ is a mBle fn from $[\![\Gamma]\!]$ to $Pr[\![\texttt{t}]\!]$.

$[\![\texttt{y:real} \vdash \texttt{(sample (norm y 1))} : \texttt{real}]\!]\eta(A)$

Exprs mean mBle fns

$[\![\Gamma \vdash \mathtt{e} : \mathtt{t}]\!]$ is a mBle fn from $[\![\Gamma]\!]$ to $\Pr[\![\mathtt{t}]\!]$.

$[\![\mathtt{y{:}real} \vdash \mathtt{(sample\ (norm\ y\ 1))} : \mathtt{real}]\!]\eta(A)$

$= \int_A \text{density-norm}(s \mid \eta(y), 1)\, ds.$

Exprs mean mBle fns

$[\![\Gamma \vdash \mathtt{e} : \mathtt{t}]\!]$ is a mBle fn from $[\![\Gamma]\!]$ to Pr$[\![\mathtt{t}]\!]$.

$[\![\mathtt{y{:}real} \vdash \mathtt{(sample\ (norm\ y\ 1))} : \mathtt{real}]\!]\eta(A)$

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

Defined recursively. Complex but doable.

Plan for the rest

1. Denotational semantics.
   PL with discrete random choices.

2. Baby measure theory.
   PL with cont. distribution.

3. Quasi-Borel space (QBS).
   PL with cont. distr. & higher-order (HO) fns.

4. [To be skipped] SFinKer monad on QBS.
   PL with cont. distr., HO fns & conditioning.

Prob. PL with HO fns and continuous random choices

t ::= bool | real | dist[bool] | dist[real] | $(t_1,\ldots,t_n) \rightarrow t$

e ::= c | x | (fn [x ... x] e) | (e e ... e) | (if e e e)

c ::= true | false | 0 | 1 | 2 | and | + | ...
| sample | flip | normal | ...

Prob. PL with HO fns and continuous random choices

t ::= bool | real | dist[bool] | dist[real] | $(t_1,\ldots,t_n) \to t$

e ::= c | x | (fn [x … x] e) | (e e … e) | (if e e e)

c ::= true | false | 0 | 1 | 2 | and | + | …
| sample | flip | normal | …

Function type.

Prob. PL with HO fns and continuous random choices

t ::= bool | real | dist[bool] | dist[real] | $(t_1,\ldots,t_n)\rightarrow t$

e ::= c | x | (fn [x … x] e) | (e e … e) | (if e e e)

c ::= true | false | 0 | 1 | 2 | and | + | …
| sample | flip | normal | …

Function type.
General fn decl. and app.

Prob. PL with HO fns and continuous random choices

t ::= bool | real | dist[bool] | dist[real] | $(t_1,\ldots,t_n) \to t$

e ::= c | x | (fn [x … x] e) | (e e … e) | (if e e e)

c ::= true | false | 0 | 1 | 2 | and | + | …
| sample | flip | normal | …

Function type.
General fn decl. and app.
General constants.

Prob. PL with HO fns and continuous random choices

t ::= bool | real | dist[bool] | dist[real] | $(t_1,\ldots,t_n) \to t$

e ::= c | x | (fn [x … x] e) | (e e … e) | (if e e e)

c ::= true | false | 0 | 1 | 2 | and | + | …
| sample | flip | normal | …

Function type.
General fn decl. and app.
General constants.

Measure theory insufficient due to HO fns.

Use a new foundation of probability theory based on quasi-Borel spaces.

Interpret $[\![\mathtt{t}]\!]$ as a quasi-Borel space (QBS), and $[\![\Gamma \vdash \mathtt{e} : \mathtt{t}]\!]$ as a QBS morphism.

High-level idea:
Random variable first.

Random variable α in X

Random variable α in X

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

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

Random variable α in X in measure theory

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

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

Random variable 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 variable α in X
in measure theory

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

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

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

Random variable α in X in measure theory

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

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

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

Random variable α in X

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

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

Random variable α in X in quasi-Borel spaces

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

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

Random variable 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 variable 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 variable 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 variable 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 variable is an induced concept.
- QBS:
  - Quasi-Borel space $(X, M \subseteq [\mathbb{R} \to X])$.
  - M is the set of random variables.

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

Quasi-Borel space - set with random variables.

Quasi-Borel space - set with random variables.

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

such that M has enough random variables.

Quasi-Borel space - set with random variables.

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

such that M has enough random variables.

Quasi-Borel space - set with random variables.

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

such that M has enough random variables.

Quasi-Borel space - set with random variables.

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

such that M has enough random variables.

1. M contains all constant functions.

Quasi-Borel space - set with random variables.

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

such that M has enough random variables.

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 variables.

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

such that M has enough random variables.

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

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

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 variables to random variables.

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

[Th] QBSes support higher-order functions well.
(The category of QBSes is cartesian closed.)

[Q] What are the sets of random variables?

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 variables?

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 variables?

1. Product: $(X,M) \times_q (Y,N) = (Z,O)$.
   - $Z = X \times Y$, $\pi_1(x,y) = x$, $\pi_2(x,y) = y$.
   - $O = \{ r \mapsto (\alpha(r),\beta(r)) \mid \alpha \in M \text{ and } \beta \in N \}$.
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 variables?

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

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

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

$$[\![\mathtt{dist[bool]}]\!] = \mathrm{Pr}_q([\![\mathtt{bool}]\!])$$

Types mean QBSes

Conversion of mBle space to QBS

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

$$[\![\mathrm{dist[bool]}]\!] = \mathrm{Pr_q}([\![\mathrm{bool}]\!])$$

Types mean QBSes

Conversion of
mBle space to QBS

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

$$[\![\mathrm{dist}[\mathrm{bool}]]\!] = \mathrm{Pr_q}([\![\mathrm{bool}]\!])$$

QBS prob. space

Types mean QBSes

Conversion of mBle space to QBS

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

$[\![\mathtt{real}]\!] = \ldots$

$[\![\mathtt{dist[bool]}]\!] = \mathrm{Pr_q}([\![\mathtt{bool}]\!])$

QBS prob. space

$[\![\mathtt{dist[real]}]\!] = \ldots$

$[\![(\mathtt{t_1,t_2}) \rightarrow \mathtt{t}]\!] = \ldots$

[Q] Fill in …

Types mean QBSes

Conversion of
mBle space to QBS

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

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

$$[\![\mathrm{dist[bool]}]\!] = \mathrm{Pr}_q([\![\mathrm{bool}]\!])$$

QBS prob. space

$$[\![\mathrm{dist[real]}]\!] = \mathrm{Pr}_q([\![\mathrm{real}]\!])$$

$$[\![(t_1,t_2)\rightarrow t]\!] = \ldots$$

[Q] Fill in …

Types mean QBSes

Conversion of mBle space to QBS

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

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

$[\![\mathrm{dist}[\mathrm{bool}]]\!] = \mathrm{Pr}_q([\![\mathrm{bool}]\!])$

QBS prob. space

$[\![\mathrm{dist}[\mathrm{real}]]\!] = \mathrm{Pr}_q([\![\mathrm{real}]\!])$

$[\![(t_1,t_2)\longrightarrow t]\!] = [\![t_1]\!] \times_q [\![t_2]\!] \rightarrow_q \mathrm{Pr}_q([\![t]\!])$

[Q] Fill in …

Types mean QBSes

$$[\![\mathtt{x}_1\mathtt{:t}_1, \ldots, \mathtt{x}_n\mathtt{:t}_n]\!] = (X, M)$$

Types mean QBSes

$$[\![\mathtt{x}_1\mathtt{:t}_1, \ldots, \mathtt{x}_n\mathtt{:t}_n]\!] = (X, M)$$

where

$(X_i, M_i) = [\![\mathtt{t}_i]\!]$

$X = \ldots$

$M = \ldots$

Types mean QBSes

$$[\![\mathtt{x}_1{:}\mathtt{t}_1, \ldots, \mathtt{x}_n{:}\mathtt{t}_n]\!] = (X, M)$$

where

$(X_i, M_i) = [\![\mathtt{t}_i]\!]$

$X = \{\eta \mid \eta(\mathtt{x}_i) \in X_i \text{ for all } i\}$

$M = \ldots$

Types mean QBSes

$$[\![\mathtt{x}_1{:}\mathtt{t}_1, \ldots, \mathtt{x}_n{:}\mathtt{t}_n]\!] = (X, M)$$

where

$(X_i, M_i) = [\![\mathtt{t}_i]\!]$

$X = \{\eta \mid \eta(\mathtt{x}_i) \in X_i \text{ for all } i\}$

$M = \ldots$

[Q] Fill in …

Types mean QBSes

$$[\![x_1{:}t_1, \ldots, x_n{:}t_n]\!] = (X, M)$$

where

$(X_i, M_i) = [\![t_i]\!]$

$X = \{\eta \mid \eta(x_i) \in X_i \text{ for all } i\}$

$M = \{r \longmapsto (x_i \longmapsto \alpha_i(r)) \mid \alpha_i \in M_i \text{ for all } i\}$

[Q] Fill in …

Exprs mean QBS morphisms

$[\![\Gamma \vdash \mathtt{e} : \mathtt{t}]\!]$ is a QBS morphism from $[\![\Gamma]\!]$ to $\mathrm{Pr}_q[\![\mathtt{t}]\!]$.

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 α∈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 $\alpha \in M$ 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 $(\mathbb{R},\mathfrak{B})$.

E.g.

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

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

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], $\alpha(r)$ = if $(r < 0.5)$ true false

μ' = uniform(0,2]/2, $\alpha'(r)$ = 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 $(\mathbb{R},\mathfrak{B})$.

Equate two QBS prob. measures:

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

if $\mu \circ \alpha^{-1} = \nu \circ \beta^{-1}$.

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

QBS of prob. measures

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

$Y = \{\ [\alpha,\mu] \mid (\alpha,\mu)$ is a prob. meas. on $(X,M)\ \}$.

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

k(r,-) is a probability measure
and k(-,A) is measurable for all r,A

Completing the definitions

$$[\![t \to t']\!] = [[\![t]\!] \to_q Pr_q([\![t']\!])]$$

$[\![\Gamma \vdash e : t]\!]$ is a morphism $[\![\Gamma]\!] \to_q Pr_q[\![t]\!]$

We couldn’t cover:

1. SFinKer Monad on QBSes and semantics of conditioning.

If you want to know about them, look at:

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