Semantic Foundations for Probabilistic Programming

Chris Heunen

Ohad Kammar, Sam Staton, Frank Wood, Hongseok Yang
Semantic foundations

Motivation:
- Ground programmer's unspoken intuitions
- Justify/refute/suggest program transformations
- Understand

Operational: remember implementation details (efficiency)
Denotational: see what program does conceptually (correctness)
Semantic foundations

programs $\rightarrow$ mathematical objects

\[ s_1 \rightarrow s_2 \rightarrow s_1; s_2 \rightarrow \cdots \]

Operational: remember implementation details (efficiency)

Denotational: see what program does conceptually (correctness)

Motivation:

- Ground programmer's unspoken intuitions
- Justify/refute/suggest program transformations
- Understand
Semantic foundations

Operational: remember implementation details (efficiency)
Denotational: see what program does conceptually (correctness)
Semantic foundations

Motivation:

- Ground programmer’s unspoken intuitions
- Justify/refute/suggest program transformations
- Understand programming through mathematics

Operational: remember implementation details (efficiency)

Denotational: see what program does conceptually (correctness)
Semantic foundations

Programs $\rightarrow$ Mathematical objects

- Operational: remember implementation details (efficiency)
- Denotational: see what program does conceptually (correctness)

Motivation:
- Ground programmer’s unspoken intuitions
- Justify/refute/suggest program transformations
- Understand probability through program equations
Probabilistic programming

\[ P(A \mid B) = \frac{P(B \mid A) \times P(A)}{P(B)} \]
Probabilistic programming

\[ P(A | B) \propto P(B | A) \times P(A) \]
Probabilistic programming

\[ P(A \mid B) \propto P(B \mid A) \times P(A) \]

posterior $\propto$ likelihood $\times$ prior

http://www.robots.ox.ac.uk/~fwood/anglican
Probabilistic programming

\[ P(A \mid B) \propto P(B \mid A) \times P(A) \]

posterior \( \propto \) likelihood \( \times \) prior

idealized Anglican = functional programming +

normalize    observe    sample

http://www.robots.ox.ac.uk/~fwood/anglican
Overview

- Interpret types as measurable spaces  
  e.g. \([\text{real}] = \mathbb{R}\)
- Interpret (open) terms as kernels
- Interpret closed terms as measures
- Inference \textit{normalizes} measures  
  posterior $\propto$ likelihood $\times$ prior

Overview

- Interpret types as measurable spaces  
  e.g. \([\text{real}] = \mathbb{R}\)
- Interpret (open) terms as kernels
- Interpret closed terms as measures
- Inference \textbf{normalizes} measures  
  \(\text{posterior} \propto \text{likelihood} \times \text{prior}\)

But:

- Commutativity?
- Higher order functions?
- Extensionality?
- Recursion?

Example

1. Toss a fair coin to get outcome $x$
2. Set up exponential decay with rate $r$ depending on $x$
3. Observe immediate decay
4. What is the outcome $x$?
Example

1. **Toss** a fair coin to get outcome $x$
2. Set up exponential decay with rate $r$ depending on $x$
3. **Observe** immediate decay
4. **What** is the outcome $x$?

```plaintext
let x = sample(bern(0.5)) in
let r = if x then 2.0 else 1.0
observe(0.0 from exp(r));
return x
```
Example

1. **Toss** a fair coin to get outcome $x$
2. Set up exponential decay with rate $r$ depending on $x$
3. **Observe** immediate decay
4. **What** is the outcome $x$?

  two traces:

<table>
<thead>
<tr>
<th></th>
<th>0.5</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=true</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x=false</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

let $x = \text{sample}(\text{bern}(0.5))$ in
let $r = \text{if } x \text{ then } 2.0 \text{ else } 1.0$
observe(0.0 from exp(r));
return $x$
Example

1. **Toss** a fair coin to get outcome $x$
2. Set up exponential decay with rate $r$ depending on $x$
3. **Observe** immediate decay
4. What is the outcome $x$?

<table>
<thead>
<tr>
<th>Trace 1</th>
<th>Trace 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x=\text{true}$</td>
<td>$x=\text{false}$</td>
</tr>
<tr>
<td>$r=2.0$</td>
<td>$r=1.0$</td>
</tr>
</tbody>
</table>

two traces:

```
let x = sample(bern(0.5)) in
let r = if x then 2.0 else 1.0
observe(0.0 from exp(r));
return x
```

```
let x = sample(bern(0.5)) in
let r = if x then 2.0 else 1.0
observe(0.0 from exp(r));
return x
```

score 2

return true
Example

1. Toss a fair coin to get outcome $x$
2. Set up exponential decay with rate $r$ depending on $x$
3. Observe immediate decay
4. What is the outcome $x$?

Two traces:

<table>
<thead>
<tr>
<th></th>
<th>0.5</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>r</td>
<td>2.0</td>
<td>1.0</td>
</tr>
<tr>
<td>score</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>return</td>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>

```python
let x = sample(bern(0.5)) in
let r = if x then 2.0 else 1.0
observe(0.0 from exp(r));
return x
```
Example

1. Toss a fair coin to get outcome \( x \)
2. Set up exponential decay with rate \( r \) depending on \( x \)
3. Observe immediate decay
4. What is the outcome \( x \)?

\[
\text{two traces:} \quad \begin{array}{cc}
0.5 & 0.5 \\
\text{x=true} & \text{x=false} \\
2.0 & 1.0 \\
\text{score 2} & \text{score 1} \\
\text{return true} & \text{return false}
\end{array}
\]

\[
\text{posterior} \propto \text{likelihood} \times \text{prior}
\]

\[
2 \times 0.5: \text{true} \\
1 \times 0.5: \text{false}
\]
Example

1. **Toss** a fair coin to get outcome $x$
2. Set up exponential decay with rate $r$ depending on $x$
3. **Observe** immediate decay
4. **What is the outcome $x$?**

$P(\text{true}) = 1, P(\text{false}) = 0.5$

**two traces:**

\[
\begin{array}{ll}
0.5 & 0.5 \\
\hline
x=\text{true} & x=\text{false} \\
r=2.0 & r=1.0 \\
\text{score 2} & \text{score 1} \\
\text{return true} & \text{return false}
\end{array}
\]

posterior $\propto$ likelihood $\times$ prior

$2 \times 0.5$: true
$1 \times 0.5$: false
Example

1. Toss a fair coin to get outcome $x$
2. Set up exponential decay with rate $r$ depending on $x$
3. Observe immediate decay
   model evidence (score): 1.5
4. What is the outcome $x$?
   $P(\text{true}) = 66\%, P(\text{false}) = 33\%$

   two traces:
   
   let $x = \text{sample(bern(0.5))}$ in
   x=true x=false
   let $r = \text{if}~x~\text{then}~2.0~\text{else}~1.0$
   r=2.0 r=1.0
   observe(0.0 \text{ from exp}(r));
   score 2 score 1
   return $x$
   return true return false

posterior $\propto$ likelihood $\times$ prior

2 $\times$ 0.5: true
1 $\times$ 0.5: false
Example

1. **Toss** a fair coin to get outcome $x$
2. Set up exponential decay with rate $r$ depending on $x$
3. **Observe** immediate decay
   
   model evidence (score): 1.5
4. **What** is the outcome $x$? $P(\text{true}) = 66\%, P(\text{false}) = 33\%$

Programs may also sample continuous distributions so have to deal with uncountable number of traces:

```plaintext
let y = sample(gauss(7,2))
```
Impossible to sample 0.5 from standard normal distribution
But sample in interval (0, 1) with probability around 0.34
Impossible to sample 0.5 from standard normal distribution
But sample in interval (0, 1) with probability around 0.34

A measurable space is a set \(X\) with a family \(\Sigma_X\) of subsets
that is closed under countable unions and complements

A (probability) measure on \(X\) is a function \(p: \Sigma_X \rightarrow [0, \infty]\)
that satisfies \(p(\sum U_n) = \sum p(U_n)\) (and has \(p(X) = 1\))
First order language

- Types: \( A, B ::= \mathbb{R} | P(A) | 1 | A \times B | \sum_{i \in I} A_i \)

  - real numbers
  - finite products
  - distributions over \( A \)
  - countable sums

bool := 1 + 1

nat := \( \sum_{i \in \mathbb{N}} 1 \)
First order language

- **Types:** \( \mathbb{A}, \mathbb{B} ::= \mathbb{R} \mid \text{P}(\mathbb{A}) \mid 1 \mid \mathbb{A} \times \mathbb{B} \mid \sum_{i \in I} \mathbb{A}_i \)

- **Deterministic terms** may not sample:
  - variables \( x, y, z \)
  - constructors for sums and products \( \text{case}, \text{in}_i, \text{if}, \text{false}, \text{true} \)
  - measurable functions \( \text{bern}, \text{exp}, \text{gauss}, \text{dirac} \)

\[\Gamma \vdash_d 42.0 : \mathbb{R}\]
\[\Gamma \vdash_d \text{gauss}(2.0, 7.0) : \text{P}(\mathbb{R})\]
\[x : \mathbb{R}, y : \mathbb{R} \vdash_d x + y : \mathbb{R}\]
\[x : \mathbb{R}, y : \mathbb{R} \vdash_d x < y : \text{bool}\]
First order language

- **Types:** \( \mathbb{A}, \mathbb{B} ::= \mathbb{R} \mid \mathbb{P}(\mathbb{A}) \mid 1 \mid \mathbb{A} \times \mathbb{B} \mid \sum_{i \in I} \mathbb{A}_i \)

- **Deterministic terms may not sample:**
  - variables \( x, y, z \)
  - constructors for sums and products \( \text{case, in}_i, \text{if, false, true} \)
  - measurable functions \( \text{bern, exp, gauss, dirac} \)

- **Probabilistic terms may sample:**
  - sequencing \( \text{return, let} \)
  - constraints \( \text{score} \)
  - priors \( \text{sample} \)

\[
\begin{align*}
\Gamma \vdash_d t : \mathbb{A} & \quad \Rightarrow \quad \Gamma \vdash_p \text{return}(t) : \mathbb{A} \\
\Gamma \vdash_d t : \mathbb{R} & \quad \Rightarrow \quad \Gamma \vdash_p \text{score}(t) : 1 \\
\Gamma \vdash_p t : \mathbb{A} \quad x : \mathbb{A} \vdash_p u : \mathbb{B} & \quad \Rightarrow \quad \Gamma \vdash_p \text{let} \ x = t \ \text{in} \ u : \mathbb{B} \\
\Gamma \vdash_d t : \mathbb{P}(\mathbb{A}) & \quad \Rightarrow \quad \Gamma \vdash_p \text{sample}(t) : \mathbb{A}
\end{align*}
\]
First order language

- **Types:** \( A, B ::= \mathbb{R} | P(A) | 1 | A \times B | \sum_{i \in I} A_i \)

- **Deterministic terms** may not sample:
  - variables \( x, y, z \)
  - constructors for sums and products \( \text{case, in}_i, \text{if, false, true} \)
  - measurable functions \( \text{bern, exp, gauss, dirac} \)
  - inference \( \text{norm} \)

- **Probabilistic terms** may sample:
  - sequencing \( \text{return, let} \)
  - constraints \( \text{score} \)
  - priors \( \text{sample} \)

\[
\begin{align*}
\Gamma \vdash_d t : A & \quad \Gamma \vdash_p \text{return}(t) : A \\
\Gamma \vdash_p t : A & \quad \Gamma \vdash_p \text{let } x = t \text{ in } u : B \\
\Gamma \vdash_d t : \mathbb{R} & \quad \Gamma \vdash_p \text{score}(t) : 1 \\
\Gamma \vdash_p t : P(A) & \quad \Gamma \vdash_p \text{sample}(t) : A
\end{align*}
\]
First order semantics

Interpret

► type $A$ as measurable space $[A]$  
► deterministic term $\Gamma \vdash_d t : A$ as measurable function $[\Gamma] \rightarrow [A]$  
► probabilistic term $\Gamma \vdash_p t : A$ as kernel $[t] : [\Gamma] \times \Sigma[A] \rightarrow [0, \infty]$  
   fixing first argument: measure, fixing second argument: measurable
First order semantics

Interpret

- **type** \( \mathcal{A} \)
  - as measurable space \( [\mathcal{A}] \)

- **deterministic term** \( \Gamma \vdash_{d} t : \mathcal{A} \)
  - as measurable function \( [\Gamma] \to [\mathcal{A}] \)

- **probabilistic term** \( \Gamma \vdash_{p} t : \mathcal{A} \)
  - as kernel \( [t] : [\Gamma] \times \Sigma_{[\mathcal{A}]} \to [0, \infty] \)
  - fixing first argument: measure,
  - fixing second argument: measurable

\[
\Gamma \vdash_{d} t : \mathbb{R} \quad \frac{}{\Gamma \vdash_{p} \text{score}(t) : 1}
\]

\[
[\text{score}(t)](\gamma, \ast) = [t](\gamma)
\]
First order semantics

Interpret

- **type** $\mathbb{A}$
  - as measurable space $[\mathbb{A}]$
- **deterministic term** $\Gamma \vdash_d t: \mathbb{A}$
  - as measurable function $[\Gamma] \rightarrow [\mathbb{A}]$
- **probabilistic term** $\Gamma \vdash_p t: \mathbb{A}$
  - as kernel $[t]: [\Gamma] \times \Sigma[\mathbb{A}] \rightarrow [0, \infty]$
  - fixing first argument: measure,
  - fixing second argument: measurable

\[
\frac{\Gamma \vdash_d t: \mathbb{R}}{\Gamma \vdash_p \text{score}(t): 1}
\]

\[
\frac{\Gamma \vdash_d t: P(\mathbb{A})}{\Gamma \vdash_p \text{sample}(t): \mathbb{A}}
\]

$\text{[score}(t)](\gamma, \star) = [t](\gamma)$

$\text{[sample}(t)](\gamma, U) = ([t](\gamma))(U)$
First order semantics

Interpret

- **type** \( \mathbb{A} \) as measurable space \([\mathbb{A}]\)
- **deterministic term** \( \Gamma \vdash_d t : \mathbb{A} \) as measurable function \([\Gamma] \to [\mathbb{A}]\)
- **probabilistic term** \( \Gamma \vdash_p t : \mathbb{A} \) as kernel \([t] : [\Gamma] \times \Sigma_{[\mathbb{A}]} \to [0, \infty]\)

fixing first argument: measure, fixing second argument: measurable

\[
\frac{\Gamma \vdash_d t : \mathbb{R}}{\Gamma \vdash_p \text{score}(t) : 1}
\]
\[
\frac{\Gamma \vdash_d t : \mathbb{P}(\mathbb{A})}{\Gamma \vdash_p \text{sample}(t) : \mathbb{A}}
\]
\[
\frac{\Gamma \vdash_p t : \mathbb{A} \quad x : \mathbb{A} \vdash_p u : \mathbb{B}}{\Gamma \vdash_p \text{let } x = t \text{ in } u : \mathbb{B}}
\]

\[
[\text{score}(t)](\gamma, \ast) = [t](\gamma)
\]
\[
[\text{sample}(t)](\gamma, U) = ([t](\gamma))(U)
\]
\[
[\text{let } x = t \text{ in } u](\gamma, U) = \int_{[\mathbb{A}]} [u](\gamma, x, U) [t](\gamma, dx)
\]
First order semantics

Interpret

- **type** $A$ as measurable space $[A]$

- **deterministic term** $\Gamma \vdash_d t : A$ as measurable function $[\Gamma] \rightarrow [A]$

- **probabilistic term** $\Gamma \vdash_p t : A$

  fixing first argument: measure, fixing second argument: measurable

  as kernel $[t] : [\Gamma] \times \Sigma_[A] \rightarrow [0, \infty]$

\[
\Gamma \vdash_d t : \mathbb{R} \\
\Gamma \vdash_p \text{score}(t) : 1
\]

\[
\Gamma \vdash_d t : P(A) \\
\Gamma \vdash_p \text{sample}(t) : A
\]

\[
\Gamma \vdash_p t : A \quad x : A \vdash_p u : B \\
\Gamma \vdash_p \text{let } x = t \text{ in } u : B
\]

$\Gamma \vdash_p \text{let } x = t \text{ in } u : B = \int_{[A]} [u]d[t]$

[8 / 21]
Example

```
let x = sample(bern(0.5)) in
let r = if x then 2.0 else 1.0
observe(0.0 from exp(r));
return x
```

The meaning of a program returning values in $X$ is a measure on $X$

- $\emptyset$ has measure $0.0$
- $\{\text{true}\}$ has measure $1.0 = 0.5 \times 2.0$
- $\{\text{false}\}$ has measure $0.5 = 0.5 \times 1.0$
- $\{\text{true, false}\}$ has measure $1.5$
Normalization: \textbf{posterior} \propto \textbf{likelihood} \times \textbf{prior}

\[ \Gamma \vdash \text{posterior}(t) : R \times P(\mathbb{A}) + 1 + 1 \]

- Model evidence
- Errors
- Normalized posterior
Normalization: \textbf{posterior} $\propto$ \textbf{likelihood} $\times$ \textbf{prior}

\[
\Gamma \vdash_p t : \mathbb{A}
\]

\[
\Gamma \vdash_d \text{norm}(t) : \mathbb{R} \times \mathbb{P}(\mathbb{A}) + 1 + 1
\]

model evidence

errors

normalized posterior

Interpretation of probabilistic term is kernel $[\Gamma] \times \Sigma_{[\mathbb{A}]} \rightarrow [0, \infty]$ so fixing first argument gives measure

\[
[t](\gamma, -)
\]

is normalized probability measure
Normalization: \( \text{posterior} \propto \text{likelihood} \times \text{prior} \)

\[
\Gamma \vdash_p t : A \\
\Gamma \vdash_d \text{norm}(t) : \mathbb{R} \times \mathbb{P}(A) + 1 + 1
\]

- Model evidence
- Errors (constant is 0 or \( \infty \))
- Normalized posterior

Interpretation of probabilistic term is kernel \([\Gamma] \times \Sigma_{[A]} \to [0, \infty] \) so fixing first argument gives measure

\[
[t](\gamma, -) \\
[t](\gamma, [A])
\]

is normalized probability measure

normalizing constant is model evidence
Normalization: \( \text{posterior} \propto \text{likelihood} \times \text{prior} \)

\[
\Gamma \vdash_{p} t : \mathbb{A} \\
\Gamma \vdash_{d} \text{norm}(t) : \mathbb{R} \times \mathbb{P}(\mathbb{A}) + 1 + 1
\]

- **Model evidence**
- **Errors** (constant is 0 or \( \infty \))
- **Normalized posterior**

\[
\begin{array}{l}
\text{let } x = \text{sample}(\text{bern}(0.5)) \text{ in} \\
\text{let } r = \text{if } x \text{ then } 2.0 \text{ else } 1.0 \\
\text{observe}(0.0 \text{ from exp(r)}); \\
\text{return } x
\end{array}
\]

\[
= \left(\begin{array}{c}
\text{true: } 2.0 \times 0.5 \\
\text{false: } 1.0 \times 0.5
\end{array}\right)
\]
Normalization: \textbf{posterior} \propto \textbf{likelihood} \times \textbf{prior}

\[
\begin{align*}
\Gamma \vdash t : A \\
\Gamma \vdash \text{norm}(t) : \mathbb{R} \times \mathbb{P}(A) + 1 + 1
\end{align*}
\]

model evidence

errors
(constant is 0 or \(\infty\))

normalized posterior

\[
\text{norm}(
\begin{array}{l}
\text{let } x = \text{sample}(\text{bern}(0.5)) \text{ in} \\
\text{let } r = \text{if } x \text{ then } 2.0 \text{ else } 1.0 \\
\text{observe}(0.0 \text{ from exp}(r)); \\
\text{return } x
\end{array}
) \quad = \quad \text{in}_1(1.5, \text{bern}(0.66))
\]
Example: sequential Monte Carlo

\[
\begin{bmatrix}
\text{norm( let } x=t \\
in u) \end{bmatrix} = \begin{bmatrix}
\text{norm( let (e,d) = norm(t) in} \\
\text{score(e); let } x=\text{sample(d)} \\
in u) \end{bmatrix}
\]
Example: importance sampling

\[
\begin{bmatrix}
\text{sample}(\exp(2))
\end{bmatrix}
\]

\[
= \begin{bmatrix}
\text{let } x = \text{sample}(\text{gauss}(0,1)) \\
\text{score}(\exp-\text{pdf}(2,x) / \text{gauss}-\text{pdf}(0,1,x)); \\
\text{return } x
\end{bmatrix}
\]
Example: importance sampling

\[
\begin{bmatrix}
\text{sample}(\exp(2))
\end{bmatrix}
\]

\[
\begin{bmatrix}
\text{let } x = \text{sample}(\text{gauss}(0,1)) \\
\text{score}(\exp-\text{pdf}(2,x) / \text{gauss}-\text{pdf}(0,1,x));
\text{return } x
\end{bmatrix}
\]

\[
\begin{bmatrix}
\text{let } x = \text{sample}(\text{gauss}(0,1)) \\
\text{score}(1 / \text{gauss}-\text{pdf}(0,1,x)); \\
\text{score}(\exp-\text{pdf}(2,x)); \\
\text{return } x
\end{bmatrix}
\]

Don't normalize as you go
Example: importance sampling

\[
\begin{bmatrix}
\text{norm}( \text{sample}(\text{exp}(2)) ) \\
\end{bmatrix}
\]

\[
= \begin{bmatrix}
\text{norm}
\begin{cases}
\text{let } x = \text{sample}(\text{gauss}(0,1)) \\
\text{score}(\text{exp-pdf}(2,x) / \text{gauss-pdf}(0,1,x)) \\
\text{return } x 
\end{cases}
\end{bmatrix}
\]

\[
\neq \begin{bmatrix}
\text{norm}( \text{norm}
\begin{cases}
\text{let } x = \text{sample}(\text{gauss}(0,1)) \\
\text{score}(1 / \text{gauss-pdf}(0,1,x)) \\
\text{score}(\text{exp-pdf}(2,x)) \\
\text{return } x 
\end{cases}
\)
\end{bmatrix}
\]

Don’t normalize as you go
Commutativity

Reordering lines is very useful program transformation
\[
\begin{align*}
\left[ \begin{array}{c}
\text{let } x = t \text{ in} \\
\text{let } y = u \text{ in} \\
v
\end{array} \right] & = \left[ \begin{array}{c}
\text{let } y = u \text{ in} \\
\text{let } x = t \text{ in} \\
v
\end{array} \right]
\end{align*}
\]

Amounts to Fubini's theorem
\[
\int_J A K \int_J B K J v K d J u K d J t K = \int_J B K \int_J A K J v K d J t K d J u K
\]

Not true for arbitrary kernels, only for s-finite kernels

A kernel is s-finite when countable sum of bounded ones
\ks: \Gamma \times \Sigma \Gamma A K \to [0, \infty]

A kernel \( k \) is s-finite iff it can be built from sub-probability distributions, score, and binding
\[
\begin{align*}
\text{let } \gamma, V \mapsto & \int_J A K l (\gamma, x, V) k (\gamma, dx) \\
\text{measurable spaces and s-finite kernels form} & \text{distributive symmetric monoidal category}
\end{align*}
\]
Commutativity

Reordering lines is very useful program transformation

\[
\begin{bmatrix}
\text{let } x=t \text{ in } \\
\text{let } y=u \text{ in }
\end{bmatrix}_v =
\begin{bmatrix}
\text{let } y=u \text{ in } \\
\text{let } x=t \text{ in }
\end{bmatrix}_v
\]

amounts to Fubini’s theorem

\[
\int_{[A]} \int_{[B]} [v] \, d[u] \, d[t] = \int_{[B]} \int_{[A]} [v] \, d[t] \, d[u]
\]
Commutativity

Reordering lines is very useful program transformation

\[
\begin{bmatrix}
\text{let } x=t \text{ in} \\
\text{let } y=u \text{ in}
\end{bmatrix}
\begin{bmatrix}
\text{let } y=u \text{ in} \\
\text{let } x=t \text{ in}
\end{bmatrix}
\begin{bmatrix}
v
\end{bmatrix}
\begin{bmatrix}
v
\end{bmatrix}
\]

amounts to Fubini’s theorem

\[
\int_{[A]} \int_{[B]} [v] \, d[u] \, d[t] = \int_{[B]} \int_{[A]} [v] \, d[t] \, d[u]
\]

Not true for arbitrary kernels, only for s-finite kernels

Kernel is s-finite when countable sum of bounded ones

\(k : [\Gamma] \times \Sigma_{[A]} \to [0, \infty]\) bounded if \(\exists n \forall \gamma \forall U : k(\gamma, U) < n\)
Commutativity

Reordering lines is very useful program transformation

\[
\begin{align*}
\left[ \begin{array}{c}
\text{let } x=t \text{ in} \\
\text{let } y=u \text{ in}
\end{array} \right] = \\
\left[ \begin{array}{c}
\text{let } y=u \text{ in} \\
\text{let } x=t \text{ in}
\end{array} \right]
\end{align*}
\]

amounts to Fubini’s theorem

\[
\int_{[A]} \int_{[B]} [v] \, d[u] \, d[t] = \int_{[B]} \int_{[A]} [v] \, d[t] \, d[u]
\]

Not true for arbitrary kernels, only for s-finite kernels

kernel is s-finite when countable sum of bounded ones

\(k: [\Gamma] \times \Sigma_{[A]} \to [0, \infty]\) bounded if \(\exists n \forall \gamma \forall U: k(\gamma, U) < n\)

- kernel \(k\) is s-finite iff it can be built from sub-probability distributions, score, and binding

\[
k \ggg l \text{ is } (\gamma, V) \mapsto \int_{[A]} l(\gamma, x, V)k(\gamma, dx)
\]

- measurable spaces and s-finite kernels form distributive symmetric monoidal category
Commutativity

Reordering lines is very useful program transformation

\[
\begin{bmatrix}
\text{let } x=t \text{ in } \\
\text{let } y=u \text{ in }
\end{bmatrix}_v = \begin{bmatrix}
\text{let } y=u \text{ in } \\
\text{let } x=t \text{ in }
\end{bmatrix}_v
\]

amounts to Fubini’s theorem

\[
\int_{[A]} \int_{[B]} [v] \, d[u] \, d[t] = \int_{[B]} \int_{[A]} [v] \, d[t] \, d[u]
\]

\[\begin{Vcenter}
\text{Not true for arbitrary kernels, only for s-finite kernels}
\end{Vcenter}\]

kernel is s-finite when countable sum of bounded ones

\[k: [\Gamma] \times \Sigma_{[A]} \to [0, \infty] \text{ bounded if } \exists n \forall \gamma \forall U: k(\gamma, U) < n\]

Interpret terms as s-finite kernels
Example: facts about distributions

\[
\begin{align*}
\text{let } x &= \text{sample}(\text{gauss}(0.0, 1.0)) \\
\text{in return } (x < 0)
\end{align*}
\]

\[
\begin{align*}
\text{let } x &= \text{sample}(\text{gauss}(0.0, 1.0)) \\
\text{in return } (x < 0)
\end{align*}
\]
Example: conjugate priors

\[
\begin{align*}
\text{let } x &= \text{sample}(\text{beta}(1,1)) \\
\text{in } \text{observe}(\text{bern}(x), \text{true}); \\
\text{return } x
\end{align*}
\]

\[
\begin{align*}
\text{observe}(\text{bern}(0.5), \text{true}); \\
\text{let } x &= \text{sample}(\text{beta}(2,1)) \\
\text{in } \text{return } x
\end{align*}
\]
Higher order functions

Allow probabilistic terms as input/output for other terms

(define (ibp-stick-breaking-process concentration base-measure)
  (let ((sticks (mem (lambda j (random-beta 1.0 concentration))))
        (atoms (mem (lambda j (base-measure))))
        (lambda ()
          (let loop (((j 1) (dualstick (sticks 1))))
            (append (if (flip dualstick) ;; with prob. dualstick
                        (atoms j) ;; add feature j
                       ')()) ;; otherwise, next stick
              (loop (+ j 1) (* dualstick (sticks (+ j 1))))))))))

[Roy et al, “A stochastic programming perspective on nonparametric Bayes”, ICML 2008]
Higher order functions

Allow probabilistic terms as input/output for other terms

\[
\begin{align*}
\text{(define (ibp-stick-breaking-process concentration base-measure)} \\
\text{(let ((sticks (mem (lambda j (random-beta 1.0 concentration)))))} \\
\text{(atoms (mem (lambda j (base-measure)))))} \\
\text{(lambda ()} \\
\text{(let loop ((j 1) (dualstick (sticks 1)))} \\
\text{(append (if (flip dualstick) \text{;; with prob. dualstick}} \\
\text{(atoms j) \text{;; add feature j}} \\
\text{'}()) \text{;; otherwise, next stick}} \\
\text{(loop (+ j 1) (* dualstick (sticks (+ j 1)))))))}}
\end{align*}
\]

\[
\mathbf{A, B ::= R | P(A) | 1 | A \times B | \sum_{i \in I} A_i | A \rightarrow B}
\]

[Roy et al, “A stochastic programming perspective on nonparametric Bayes”, ICML 2008]
Higher order functions

Allow probabilistic terms as input/output for other terms

\[(\text{define (ibp-stick-breaking-process concentration base-measure)})\]
\[(\text{let ((sticks (mem (lambda j (random-beta 1.0 concentration)))))})\]
\[(\text{atoms (mem (lambda j (base-measure)))))})\]
\[(\text{lambda ()})\]
\[(\text{let loop ((j 1) (dualstick (sticks 1)))})\]
\[(\text{append (if (flip dualstick) \quad \text{; with prob. dualstick}}\]
\[(\text{atoms j) \quad \text{; add feature j}}\]
\[(\text{')}) \quad \text{; otherwise, next stick}}\]
\[(\text{loop (+ j 1) (* dualstick (sticks (+ j 1)))) )))))))\]

\[A, B ::= \mathbb{R} \mid P(A) \mid 1 \mid A \times B \mid \sum_{i \in I} A_i \mid A \to B\]

\[\text{\Logo} \quad \mathbb{R} \to \mathbb{R} \text{ is not a measurable space}\]

[Roy et al, “A stochastic programming perspective on nonparametric Bayes”, ICML 2008]
Higher order functions

Allow probabilistic terms as input/output for other terms

```
(define (ibp-stick-breaking-process concentration base-measure)
  (let ((sticks (mem (lambda j (random-beta 1.0 concentration)))))
    (atoms (mem (lambda j (base-measure)))))))
  (lambda ()
    (let loop ((j 1) (dualstick (sticks 1)))
      (append (if (flip dualstick) ;; with prob. dualstick
                 (atoms j) ;; add feature j
                 '()) ;; otherwise, next stick
        (loop (+ j 1) (* dualstick (sticks (+ j 1))))))))))
```

\[
A, B ::= \mathbb{R} | P(A) | 1 | A \times B | \sum_{i \in I} A_i | A \to B
\]

\[\Rightarrow \mathbb{R} \to \mathbb{R} \text{ is not a measurable space}\]

Easy to handle operationally.

What to do denotationally?

[Roy et al, “A stochastic programming perspective on nonparametric Bayes”, ICML 2008]
[Borgström et al, “Measure transformer semantics for Bayesian machine learning”, ESOP2011]
Higher order semantics

Use category theory to extend measure theory

not enough function spaces

measurable spaces

---

---
Higher order semantics

Use category theory to extend measure theory

not enough function spaces

measurable spaces

preserves all structure

sheaves on measurable spaces

presheaves on measurable spaces that preserve countable products

all function spaces

Giry monad

distribution types

\[ \mathbb{P}(\Omega \times \mathbb{R} \rightarrow \mathbb{R}) \]

left Kan extension

\[ \mathbb{K} \]

consists of random functions measurable

\[ \Omega \times \mathbb{R} \rightarrow \mathbb{R} \]

▶

All definable functions \( \mathbb{R} \rightarrow \mathbb{R} \) are measurable

"Church-Turing"

▶

Denotational and operational semantics match soundness & adequacy

[Power, "Generic models for computational effects", Th Comp Sci 2006]
Higher order semantics

Use category theory to extend measure theory

measurable spaces \xleftarrow{\text{Giry monad}} \xrightarrow{\text{distribution types}} \text{sheaves on measurable spaces}

\[
\begin{array}{c}
\mathbb{A} \\
\downarrow \\
\text{Giry monad} \\
\downarrow \\
\mathcal{P}(\mathbb{A}) \\
\end{array}
\]

measurable spaces \xleftarrow{\text{Giry monad}} \xrightarrow{\text{distribution types}} \text{sheaves on measurable spaces}
Higher order semantics
Use category theory to extend measure theory

measurable spaces $\xleftarrow{\text{Giry monad}}$ sheaves on measurable spaces

$\xrightarrow{\text{left Kan extension}}$

measurable spaces $\xleftarrow{\text{Giry monad}}$ sheaves on measurable spaces

[Power, “Generic models for computational effects”, Th Comp Sci 2006]
Higher order semantics

Use category theory to extend measure theory

measurable spaces \leftarrow \rightarrow \text{sheaves on measurable spaces}

- \([1 \rightarrow (\mathbb{R} \rightarrow \mathbb{R})]\) consists of \emph{random functions}
- All definable functions \(\mathbb{R} \rightarrow \mathbb{R}\) are measurable
- Denotational and operational semantics match soundness & adequacy

measurable \(\Omega \times \mathbb{R} \rightarrow \mathbb{R}\)

“Church-Turing”
Not extensional: \( 1 \rightarrow A \xrightarrow{f} B \) for all \( p \not\Rightarrow f = g \)

Solution: restrict to subcategory that is extensional
Extensionality

Not extensional: \( 1 \xrightarrow{p} A \xrightarrow{f} B \xrightarrow{g} B \) for all \( p \not\Rightarrow f = g \)

Solution: restrict to subcategory that is extensional

A quasi-measurable space is a set \( X \) with \( M_X \subseteq [\mathbb{R} \to X] \) satisfying

- if \( f : \mathbb{R} \to \mathbb{R} \) is measurable and \( g \in M \), then \( gf \in M \)
- if \( f : \mathbb{R} \to X \) is constant, then \( f \in M \)
- if \( f : \mathbb{R} \to \mathbb{N} \) is measurable and \( g_n \in M \), then \( [g_n]f \in M \)

morphisms are functions \( f : X \to Y \) with \( g \in M_X \Rightarrow fg \in M_Y \)


Extensionality

Not extensional: \( \begin{array}{ccc} p & \xrightarrow{f} & g \\ A & \xrightarrow{\forall} & B \end{array} \) for all \( p \not\Rightarrow f = g \)

Solution: restrict to subcategory that is extensional

A quasi-measurable space is a set \( X \) with \( M_X \subseteq [\mathbb{R} \to X] \) satisfying

- if \( f : \mathbb{R} \to \mathbb{R} \) is measurable and \( g \in M \), then \( gf \in M \)
- if \( f : \mathbb{R} \to X \) is constant, then \( f \in M \)
- if \( f : \mathbb{R} \to \mathbb{N} \) is measurable and \( g_n \in M \), then \( [g_n]f \in M \)

morphisms are functions \( f : X \to Y \) with \( g \in M_X \Rightarrow fg \in M_Y \)

Example: \( X \) measurable space, \( M_X \) measurable functions \( \mathbb{R} \to X \)

morphism \( X \to Y \) is measurable function
Extensionality

Not extensional: \[ 1 \rightarrow A \xrightarrow{p} B \xrightarrow{g} B \text{ for all } p \not\Rightarrow f = g \]
Solution: restrict to subcategory that is extensional

A quasi-measurable space is a set \( X \) with \( M_X \subseteq [\mathbb{R} \to X] \) satisfying
- if \( f : \mathbb{R} \to \mathbb{R} \) is measurable and \( g \in M \), then \( gf \in M \)
- if \( f : \mathbb{R} \to X \) is constant, then \( f \in M \)
- if \( f : \mathbb{R} \to \mathbb{N} \) is measurable and \( g_n \in M \), then \([g_n]f \in M \)

morphisms are functions \( f : X \to Y \) with \( g \in M_X \Rightarrow fg \in M_Y \)

Example: \( X \) measurable space, \( M_X \) measurable functions \( \mathbb{R} \to X \)
morphism \( X \to Y \) is measurable function

Theorem: this gives cartesian closed category with countable sums
Corollary: if term \( t \) has first order type, then \([t]\) is measurable
even if \( t \) involves higher order functions
Extensionality

Not extensional: \[ 1 \rightarrow A \xrightarrow{f} B \] for all \( p \not\Rightarrow f = g \)

Solution: restrict to subcategory that is extensional

A quasi-measurable space is a set \( X \) with \( M_X \subseteq [\mathbb{R} \rightarrow X] \) satisfying

- if \( f: \mathbb{R} \rightarrow \mathbb{R} \) is measurable and \( g \in M \), then \( gf \in M \)
- if \( f: \mathbb{R} \rightarrow X \) is constant, then \( f \in M \)
- if \( f: \mathbb{R} \rightarrow \mathbb{N} \) is measurable and \( g_n \in M \), then \( [g_n]f \in M \)
  \[ t \mapsto g_{f(t)}(t) \]

A measure on \( (X, M_X) \) is a measure \( \mu \) on \( \mathbb{R} \) with a function \( f \in M \)

Proposition: measures on \([X \rightarrow Y]\) are random functions

measureable map \( \mathbb{R} \times X \rightarrow Y \) modulo measure on \( \mathbb{R} \)
Recursion

No recursion / least fixed points
Idea: restrict to presheaves over domains

An $\omega$-complete partial order has suprema of increasing sequences
morphisms preserve suprema of increasing sequences and infima

A quasi-measurable space is ordered when $X$ is an $\omega$cpo and $M$ is
closed under pointwise increasing suprema

Example: Any $\omega$cpo, e.g. $[0,1]$ take $M$ all measurable functions $\mathbb{R} \to X$
where $X$ has the Borel $\sigma$-algebra on the Lawson topology

Theorem: this gives a cartesian closed category with countable sums
Example: von Neumann’s trick

\[
\begin{aligned}
\text{let } g = \text{bern}(0.66) \text{ in } \\
\text{letrec } f() = (\text{let } x = \text{sample}(g) \\
\quad \text{let } y = \text{sample}(g) \\
\quad \text{if } x = y \text{ then } f() \\
\quad \quad \text{else return } x) \\
\text{in } f()
\end{aligned}
\]
Foundational semantics for probabilistic programming:

- continuous distributions
- soft constraints
- commutativity
- higher order functions
- recursion

Can verify/suggest program transformations. Approximations?