Formal Semantics of Probabilistic Programming Languages: Issues, Results and Opportunities

Hongseok Yang
KAIST, Korea

Based on several joint projects with Yufei Cai, Zoubin Ghahramani, Bradley Gram-hansen, Chris Heunen, Ohad Kammar, Sean Moss, Klaus Ostermann, Adam Scibior, Sam Staton, Matthijs Vakar, Frank Wood, and Yuan Zhou
Formal Semantics of Probabilistic Programming Languages: Issues, Results and Opportunities

Hongseok Yang
KAIST, Korea

Based on several joint projects with Yufei Cai, Zoubin Ghahramani, Bradley Gram-hansen, Chris Heunen, Ohad Kammar, Sean Moss, Klaus Ostermann, Adam Scibior, Sam Staton, Matthijs Vakar, Frank Wood, and Yuan Zhou
Formal Semantics of Probabilistic Programming Languages: Issues, Results and Opportunities

Hongseok Yang
KAIST, Korea

Based on several joint projects with Yufei Cai, Zoubin Ghahramani, Bradley Gram-hansen, Chris Heunen, Ohad Kammar, Sean Moss, Klaus Ostermann, Adam Scibior, Sam Staton, Matthijs Vakar, Frank Wood, and Yuan Zhou
Formal semantics describes a mapping from probabilistic programs to probabilistic models.
Formal semantics describes a mapping from probabilistic programs to probabilistic models.

```scheme
(let [x (sample (normal 0 1))
      y (observe (normal x 1) 2)]
x)
```
Formal semantics describes a mapping from probabilistic programs to probabilistic models.

```
(let [x (sample (normal 0 1))
      y (observe (normal x 1) 2)]
  x)
```
Formal semantics describes a mapping from probabilistic programs to probabilistic models.

\[
\text{(let [} \ x \ \text{(sample (normal 0 1))} \\
\quad \ y \ \text{(observe (normal x 1) 2)}]\text{]} \\
\text{x})
\]
Formal semantics describes a mapping from probabilistic programs to probabilistic models.

\[
\begin{aligned}
&\text{(let [x (sample (normal 0 1))}
\hspace{1cm}
&\text{y (observe (normal x 1) 2)]}
\hspace{1cm}
&\text{x)}
\end{aligned}
\]
Formal semantics describes a mapping from probabilistic programs to probabilistic models.

\[
\begin{align*}
\text{(let [x (sample (normal 0 1))}
& \quad y (observe (normal x 1) 2)]
\quad x) = p(x \mid y=2)
\end{align*}
\]
Formal semantics describes a mapping from probabilistic programs to probabilistic models.

\[
\begin{align*}
\text{let } [x \text{ (sample (normal 0 1))} \\
    y \text{ (observe (normal x 1) 2)}] \\
x
\end{align*}
\]  
\[= p(x | y=2)
\]

\[
\begin{align*}
\text{let } [x \text{ (sample (normal 0 1))} \\
    y \text{ (observe (normal x 1) 2)}] \\
x
\end{align*}
\]  
\[= p(x, y=2)
\]
Why should one care about formal semantics?
Why should one care about formal semantics?
Why should one care about formal semantics?
Why should one care about *formal* semantics?

2. Compiler optimisation.
3. Detection of ill-defined models.
Why should one care about formal semantics?

2. Compiler optimisation.
3. Detection of ill-defined 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-ppl7/lazy-samples

#'uppsala-ppl7/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)

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]?
\begin{verbatim}
(let [x (sample (normal 0 1))
     x-pdf (normal-pdf x 0 1)
     y (observe (exponential (/ 1 x-pdf)) 0)]
  x)
\end{verbatim}

\begin{verbatim}
(def lazy-samples (doquery :lmh example1 [[]])
(def samples (map :result (take-nth 100 (take 2000000 (drop 100000 lazy-samples))))
(plot/histogram samples :normalize :probability)
\end{verbatim}

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)

20K samples
Uniform[-6,6]?
No posterior.
Uniform[−∞,∞].
\[ p(x) = \text{normal-pdf}(x;0,1) \]

\[ p(y|x) = \exp(y \cdot \frac{1}{p(x)\cdot p(x)}) \]

\[ p(x,y=0) = p(x) \cdot p(y=0|x) = p(x) \cdot \frac{1}{p(x)} = 1 \]

\[ p(y=0) = \int p(x,y=0)dx = \int dx = \infty \]

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

\[ p(x) = \text{normal-pdf}(x;0,1) \]
\[ p(x) = \text{normal-pdf}(x;0,1) \]
\[ p(y=0|x) = \text{exponential-pdf}(y=0;1/p(x)) \]
\[ = \frac{1}{p(x)} \times \exp\left(y \times \frac{1}{p(x)}\right) \]
\[ p(x,y=0) = p(x) \times p(y=0|x) = p(x) \times \frac{1}{p(x)} = 1 \]
\[ p(y=0) = \int p(x,y=0)dx = \int dx = \infty \]

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

\[ p(x) = \text{normal-pdf}(x;0,1) \]
\[ p(y=0|x) = \text{exponential-pdf}(y=0;1/p(x)) \]
\[ p(x) = \text{normal-pdf}(x; 0, 1) \]
\[ p(y=0|x) = \text{exponential-pdf}(y=0; 1/p(x)) = \frac{1}{p(x)} \]
\[ p(x,y=0) = p(x) \cdot p(y=0|x) = p(x) \cdot \frac{1}{p(x)} = 1 \]
\[ p(y=0) = \int p(x,y=0) \, dx = \int dx = \infty \]

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

\[ p(x) = \text{normal-pdf}(x; 0, 1) \]
\[ p(y=0|x) = \text{exponential-pdf}(y=0; 1/p(x)) = \frac{1}{p(x)} \]
\( p(x) = \text{normal-pdf}(x;0,1) \)
\( p(y=0|x) = \text{exponential-pdf}(y=0;1/p(x)) = 1/p(x) \)

\[ p(x, y=0) = p(x) \times p(y=0|x) \]
\[
\begin{align*}
p(x) &= \text{normal-pdf}(x; 0, 1) \\
p(y=0|x) &= \text{exponential-pdf}(y=0; 1/p(x)) = 1/p(x) \\
p(x, y=0) &= p(x) \cdot p(y=0|x) = p(x) \cdot 1/p(x) = 1
\end{align*}
\]
\[ p(x) = \text{normal-pdf}(x;0,1) \]
\[ p(y=0|x) = \text{exponential-pdf}(y=0;\frac{1}{p(x)}) = \frac{1}{p(x)} \]
\[ p(x,y=0) = p(x) \times p(y=0|x) = p(x) \times \frac{1}{p(x)} = 1 \]
\[ p(y=0) = \int p(x,y=0) \, dx = \int dx = \infty \]
\[
\begin{align*}
\text{let } & [x \quad \text{(sample (normal 0 1))} \\
& \quad \quad \quad \quad \quad \quad x-pdf \quad \text{(normal-pdf x 0 1)} \\
& \quad \quad \quad \quad \quad \quad y \quad \text{(observe (exponential (/ 1 x-pdf)) 0))} \\
& \quad \quad \quad \quad \quad \quad x) \\
& = p(x, y=0) = \text{Lebesgue}(x)
\end{align*}
\]
Issues and results
Issue 1: Normalised posterior or unnormalized posterior?
(let [x   (sample (normal 0 1))
     x-pdf (normal-pdf x 0 1)
     y    (observe (exponential (/ 1 x-pdf) 0))]
  x)

= p(x, y=0) = Lebesgue(x)
\[
\begin{align*}
\text{(let } [x \quad (\text{sample } (\text{normal } 0 \ 1)) \\
\quad x\text{-pdf } (\text{normal-pdf } x \ 0 \ 1) \\
\quad y \quad (\text{observe } (\text{exponential } (/ \ 1 \ x\text{-pdf}) \ 0)))] \\
x) \\
= p(x, y=0) = \text{Lebesgue}(x)
\end{align*}
\]
(let [x (sample (normal 0 1))
  x-pdf (normal-pdf x 0 1)
  y (observe (exponential (/ 1 x-pdf) 0))]
  x)

= p(x, y=0) = Lebesgue(x)

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

= p(x|y=0) = undefined

(let [x' (sample (normal 0 1))
  x-pdf (normal-pdf x' 0 1)
  y (observe (normal x' 1) 0)]
  x')
\[
\begin{align*}
\text{let } & \quad \mathbf{x} \left(\text{sample } (\text{normal } 0 1)\right) \\
& \quad \mathbf{x-pdf} \left(\text{normal-pdf } \mathbf{x} 0 1\right) \\
& \quad \mathbf{y} \left(\text{observe } (\text{exponential } \left(\frac{1}{\mathbf{x-pdf}}\right) 0)\right) \\
\mathbf{x} & = p(\mathbf{x}, \mathbf{y}=0) = \text{Lebesgue}(\mathbf{x}) \\
\end{align*}
\]

\[
\begin{align*}
\text{let } & \quad \mathbf{x} \left(\text{sample } (\text{normal } 0 1)\right) \\
& \quad \mathbf{x-pdf} \left(\text{normal-pdf } \mathbf{x} 0 1\right) \\
& \quad \mathbf{y} \left(\text{observe } (\text{exponential } \left(\frac{1}{\mathbf{x-pdf}}\right) 0)\right) \\
\mathbf{x} & = p(\mathbf{x}|\mathbf{y}=0) = \text{undefined} \\
\end{align*}
\]

\[
\begin{align*}
\text{let } & \quad \mathbf{x}' \left(\text{let } \mathbf{x} \left(\text{sample } (\text{normal } 0 1)\right) \\
& \quad \mathbf{x-pdf} \left(\text{normal-pdf } \mathbf{x} 0 1\right) \\
& \quad \mathbf{y} \left(\text{observe } ...\right)\right) \\
\mathbf{x}' & = \text{...} \\
\end{align*}
\]
\[
\begin{align*}
\text{(let }& [x \quad (\text{sample } (\text{normal } 0 1)) \\
& x-pdf \quad (\text{normal-pdf } x 0 1) \\
& y \quad (\text{observe } (\text{exponential } (/ 1 x-pdf) 0))) \\
& x) \\
= p(x, y=0) = \text{Lebesgue}(x) \\
\end{align*}
\]

\[
\begin{align*}
\text{(let }& [x \quad (\text{sample } (\text{normal } 0 1)) \\
& x-pdf \quad (\text{normal-pdf } x 0 1) \\
& y \quad (\text{observe } (\text{exponential } (/ 1 x-pdf) 0))) \\
& x) \\
= p(x|y=0) = \text{undefined} \\
\end{align*}
\]
\[ \begin{align*}
(\text{let } [x \hspace{1em} (\text{sample } (\text{normal } 0 \hspace{0.5em} 1)) \\
\hspace{1em} x\text{-pdf } (\text{normal-pdf } x \hspace{0.5em} 0 \hspace{0.5em} 1) \\
\hspace{1em} y (\text{observe } (\text{exponential } (/ \hspace{0.5em} 1 \hspace{0.5em} x\text{-pdf}) \hspace{0.5em} 0))) \\
\hspace{1em} x) &= p(x, y=0) = \text{Lebesgue}(x) \\
(\text{let } [x \hspace{1em} (\text{sample } (\text{normal } 0 \hspace{0.5em} 1)) \\
\hspace{1em} x\text{-pdf } (\text{normal-pdf } x \hspace{0.5em} 0 \hspace{0.5em} 1) \\
\hspace{1em} y (\text{observe } (\text{exponential } (/ \hspace{0.5em} 1 \hspace{0.5em} x\text{-pdf}) \hspace{0.5em} 0))) \\
\hspace{1em} x) &= p(x|y=0) = \text{undefined} \\
(\text{let } [x' (\text{let } [x \hspace{1em} (\text{sample } (\text{normal } 0 \hspace{0.5em} 1)) \\
\hspace{1em} x\text{-pdf } (\text{normal-pdf } x \hspace{0.5em} 0 \hspace{0.5em} 1) \\
\hspace{1em} y (\text{observe } \ldots)) \hspace{1em} x) \\
\hspace{1em} y' (\text{observe } (\text{normal } x' \hspace{0.5em} 1 \hspace{0.5em} 0))] \\
\hspace{1em} x') &= \text{undefined}
\end{align*} \]
\[
\begin{align*}
\text{(let } [x \quad \text{(sample (normal 0 1))} \\
x\text{-pdf} \quad \text{(normal-pdf } x \, 0 \, 1) \\
y \quad \text{(observe (exponential (/ 1 x-pdf) 0))}] \\
x \text{)}
\end{align*}
\]

= \( p(x, y=0) = \text{Lebesgue}(x) \)

\[
\begin{align*}
\text{(let } [x \quad \text{(sample (normal 0 1))} \\
x\text{-pdf} \quad \text{(normal-pdf } x \, 0 \, 1) \\
y \quad \text{(observe (exponential (/ 1 x-pdf) 0))}] \\
x \text{)}
\end{align*}
\]

= \( p(x|y=0) = \text{undefined} \)

\[
\begin{align*}
\text{(let } [x' \quad \text{(let } [x \quad \text{(sample (normal 0 1))} \\
x\text{-pdf} \quad \text{(normal-pdf } x \, 0 \, 1) \\
y \quad \text{(observe ...)}] \\
x \text{)} \\
y' \quad \text{(observe (normal } x' \, 1 \text{) 0}] \\
x' \text{)}
\end{align*}
\]

= \( \text{undefined} \neq p(x'|y=0,y'=0) = \text{normal-pdf}(x';0,1) \)
(let [x (sample (normal 0 1))
x-pdf (normal-pdf x 0 1)
y (observe (exponential (/ 1 x-pdf) 0))]
x)

= p(x, y=0) = Lebesgue(x)

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

= p(x|y=0) = undefined

(let [x' (let [x (sample (normal 0 1))
x-pdf (normal-pdf x 0 1)
y (observe ...)] x)
y' (observe (normal x' 1) 0)]
x')

= p(x, y=0, y'=0) = normal-pdf(x;0,1)
In general, semantics interprets programs as unnormalized conditional distributions [Staton17].
In general, semantics interprets programs as unnormalized conditional distributions [Staton17].

\[
\text{(let [x (sample (normal z 1))}
\quad \text{x-pdf (normal-pdf x 0 1)}
\quad \text{y (observe (exponential (/ 1 x-pdf) 0))}] 
\quad x) \]

= \mathbb{P}(x, y=0 \mid z)
In general, semantics interprets programs as unnormalized conditional distributions [Staton17].

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

= $p(x, y=0 \mid z)$

Always **s-finite** unnormalised cond. distributions (also called **s-finite** kernels) [Staton17].
A kernel $k$ from $X$ to $Y$ is \textit{finite} if $\sup_x k(x,Y) < \infty$. A kernel $k$ is \textit{s-finite} if it is a \textit{countable} sum of finite kernels.
S-finite kernel

A kernel $k$ from $X$ to $Y$ is finite if $\sup_x k(x, Y) < \infty$.

A kernel $k$ is s-finite if it is a countable sum of finite kernels.

The fact that programs mean s-finite kernels justifies program reordering for optimisation:

\[
\left[\left[\text{let } [x \ e_1 \\
y \ e_2] e\right]\right] = \left[\left[\text{let } [y \ e_2 \\
x \ e_1] e\right]\right] \quad \text{if } x \notin \text{Free}(e_1) \wedge y \notin \text{Free}(e_2)
\]
Issue 2: Should marginalise or not?
(let [x1 (sample (normal 0 1))
    x2 (if (> x1 0) 2 (sample (normal -2 1)))
    y (observe (normal x2 1) -1)]
x1)
(let [x1 (sample (normal 0 1))
    x2 (if (> x1 0) 2 (sample (normal -2 1)))
    y  (observe (normal x2 1) -1)]
  x1)
(let [x1 (sample (normal 0 1))
   x2 (if (> x1 0) 2 (sample (normal -2 1)))
   y  (observe (normal x2 1) -1)]
 x1)
(let [x1 (sample (normal 0 1))
     x2 (if (> x1 0) 2 (sample (normal -2 1)))
     y (observe (normal x2 1) -1)]
  x1)
(let [x1 (sample (normal 0 1))
x2 (if (> x1 0) 2 (sample (normal -2 1)))
y (observe (normal x2 1) -1)]
x1)
(let [x1 (sample (normal 0 1))
  x2 (if (> x1 0) 2 (sample (normal -2 1)))
  y (observe (normal x2 1) -1))
  x1)
(let [x1 (sample (normal 0 1))
    x2 (if (> x1 0) 2 (sample (normal -2 1)))
    y (observe (normal x2 1) -1)]
  x1)

= p(x₁, z₂, y=-1)
\[
\begin{align*}
\text{let } & [x_1 \text{ (sample } \text{normal } 0 \ 1)] \\
x_2 \text{ (if }> x_1 \text{ 0) } 2 \text{ (sample } \text{normal } -2 \ 1)) \\
y \text{ (observe } \text{normal } x_2 \ 1) \ -1)] \\
x_1 \\
\end{align*}
\]

\[= p(x_1, z_2, y=-1) \]
\[= f(x_1;0,1) \ast f(z_2;-2,1) \ast f(y=-1; \text{if } (x_1>0) \text{ then 2 else } z_2,1) \]

Here \( f(x;\mu,\sigma) = \text{normal-pdf}(x;\mu,\sigma). \)
\[
\begin{align*}
\text{let } & \begin{align*}
{x_1} & \sim \text{normal}(0, 1) \\
{x_2} & \sim \text{normal}(-2, 1) \\
y & \sim \text{normal}(x_2, 1) - 1
\end{align*} \\
\text{observe } & (x_1, y = -1) \\
\text{p}(x_1, z_2, y = -1) & = \text{f}(x_1; 0, 1) \times \text{f}(z_2; -2, 1) \times \text{f}(y = -1; \text{if } (x_1 > 0) \text{ then } 2 \text{ else } z_2, 1) \\
\text{Here } & \text{f}(x; \mu, \sigma) = \text{normal-pdf}(x; \mu, \sigma).
\end{align*}
\]
\[
\begin{align*}
\text{let } & \begin{cases} 
  x_1 \text{ (sample (normal 0 1))} \\
  x_2 \text{ (if (> x_1 0) 2 (sample (normal -2 1)))} \\
  y \text{ (observe (normal x_2 1) -1)}
\end{cases} \\
\text{x_1}
\end{align*}
\]

\[= p(x_1, y=-1) = \int p(x_1, z_2, y=-1) \, dz_2 \]

\[= f(x_1;0,1) \times [x_1>0] \times \bullet \bullet \bullet \\
+ f(x_1;0,1) \times [x_1\leq0] \times \bullet \bullet \bullet \]

Here \(f(x;\mu,\sigma) = \text{normal-pdf}(x;\mu,\sigma)\).
(let [x1 (sample (normal 0 1))
     x2 (if (> x1 0) 2 (sample (normal -2 1)))
     y (observe (normal x2 1) -1)]
  x1)

= p(x₁, z₂, y=-1) = ∫ p(x₁, z₂, y=-1) dz₂
= f(x₁;0,1) * [x₁>0] * f(y=-1; 2, 1)
  + f(x₁;0,1) * [x₁≤0] * · · · ·

Here f(x;μ,σ) = normal-pdf(x;μ,σ).
(let [x1 (sample (normal 0 1))
   x2 (if (> x1 0) 2 (sample (normal -2 1)))
   y (observe (normal x2 1) -1)]
    x1)

= \(p(x_1, z_2, y=-1)\)
= \(f(x_1;0,1) \times f(z_2;-2,1) \times f(y=-1; \text{if } (x_1>0) \text{ then } 2 \text{ else } z_2,1)\)
Here \(f(x;\mu,\sigma) = \text{normal-pdf}(x;\mu,\sigma)\).

(let [x1 (sample (normal 0 1))
   x2 (if (> x1 0) 2 (sample (normal -2 1)))
   y (observe (normal x2 1) -1)]
    x1)

= \(p(x_1, y=-1) = \int p(x_1, z_2, y=-1) \, dz_2\)
= \(f(x_1;0,1) \times [x_1>0] \times f(y=-1; 2, 1)\)
+ \(f(x_1;0,1) \times [x_1\leq 0] \times \int f(z_2;-2,1) \times f(y=-1; z_2, 1) \, dz_2\)
(let [x1 (sample (normal 0 1))
    x2 (if (> x1 0) 2 (sample (normal -2 1)))
    y (observe (normal x2 1) -1)]
  x1)

= \int p(x_1, z_2, y=-1) \, dz_2
= f(x_1;0,1) \cdot [x_1>0] \cdot f(y=-1; 2, 1)
  + f(x_1;0,1) \cdot [x_1\leq 0] \cdot \int f(z_2;-2,1) \cdot f(y=-1; z_2, 1) \, dz_2

Here \( f(x;\mu,\sigma) = \text{normal-pdf}(x;\mu,\sigma) \).
(let [x1 (sample (normal 0 1))
    x2 (if (> x1 0) 2 (sample (normal -2 1)))
    y (observe (normal x2 1) -1)]
  x1)

= p(x₁, y=-1) = ∫ p(x₁, z₂, y=-1) dz₂
= f(x₁;0,1) * [x₁>0] * f(y=-1; 2, 1)
  + f(x₁;0,1) * [x₁≤0] * ∫ f(z₂;-2,1) * f(y=-1; z₂, 1) dz₂
\[
\begin{align*}
&= p(x_1, z_2, y=-1) \\
&= f(x_1;0,1) \times f(z_2;-2,1) \times f(y=-1; \text{if } (x_1>0) \text{ then } 2 \text{ else } z_2,1)
\end{align*}
\]

Here \( f(x;\mu,\sigma) = \text{normal-pdf}(x;\mu,\sigma) \).

Graphical model

Model on execution traces
\[
\text{(let } [x_1 (\text{sample } (\text{normal } 0 \ 1))]
\text{  } x_2 (\text{if } (> x_1 0) 2 (\text{sample } (\text{normal } -2 \ 1))))
\text{  } y \ (\text{observe } (\text{normal } x_2 1) -1)]
\text{   } x_1)
\]

\[
= p(x_1, z_2, y=-1)
= f(x_1;0,1) \times f(z_2;-2,1) \times f(y=-1; \text{if } (x_1>0) \text{ then } 2 \text{ else } z_2,1)
\]

Here \( f(x;\mu,\sigma) = \text{normal-pdf}(x;\mu,\sigma) \).

Graphical model

\[
\text{(let } [x_1 (\text{sample } (\text{normal } 0 \ 1))]
\text{  } x_2 (\text{if } (> x_1 0) 2 (\text{sample } (\text{normal } -2 \ 1))))
\text{  } y \ (\text{observe } (\text{normal } x_2 1) -1)]
\text{   } x_1)
\]

\[
= p(x_1, y=-1) = \int p(x_1, z_2, y=-1) \ \text{dz}_2
= f(x_1;0,1) \times [x_1>0] \times f(y=-1; 2, 1)
+ f(x_1;0,1) \times [x_1\leq0] \times \int f(z_2;-2,1) \times f(y=-1; z_2, 1) \ \text{dz}_2
\]

Model on execution traces
Semantics without marginalisation.

1. Converts prob. progs to graphical models.
2. Basis for graph-based inference algorithm.

Semantics with marginalisation.

1. Defines models based on prog. execution.
2. Basis for evaluation-based inference algo.
Issue 3:
Higher-order functions
Linear 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)

  [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])
Linear 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)
  [s b])
Linear 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)

[s b])
Linear regression

\[
\text{let } (s \text{ sample (normal 0 2)}), (b \text{ sample (normal 0 6)})\]

\[
f(x) = (+ (* s x) b)
\]

\[
(\text{observe (normal (f 0) .5) .6})
\]

\[
(\text{observe (normal (f 1) .5) .7})
\]

\[
(\text{observe (normal (f 2) .5) 1.2})
\]

\[
(\text{observe (normal (f 3) .5) 3.2})
\]

\[
(\text{observe (normal (f 4) .5) 6.8})
\]

\[
(\text{observe (normal (f 5) .5) 8.2})
\]

\[
(\text{observe (normal (f 6) .5) 8.4})
\]

\[
[s \ b]
\]
Linear 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)

[s b])
Linear 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)
    [s b]) ∈ Measure(ℜ²)
Linear 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)
[\[s b\]] \in Measure(\mathbb{R}^2)
Linear 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)
Linear 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)

Troublemaker in measure theory

Measure(ℝ→ℝ) ∈ Measure(ℝ²)
Measure-theoretic issue

Measure theory provides a foundation for modern probability theory.

But it doesn’t support higher-order fns well.

$$\text{ev} : (\mathbb{R} \rightarrow m\mathbb{R}) \times \mathbb{R} \rightarrow \mathbb{R}, \quad \text{ev}(f,x) = f(x).$$

[Aumann 61] ev is not measurable no matter which $\sigma$-algebra is used for $\mathbb{R} \rightarrow m\mathbb{R}$. 
We formulated a new probability theory that puts random variable as primary concept \cite{LICS'17}.

Used it to define the semantics of expressive prob. programming languages, such as Anglican.

Quasi-Borel space - core notion of this theory.
We formulated a new probability theory that puts random variable as primary concept \cite{LICS'17}.

Used it to define the semantics of expressive prob. programming languages, such as Anglican.

**Quasi-Borel space** - core notion of this theory.
Random variable $\alpha$ in $X$
Random variable $\alpha$ in $X$

$\alpha : \Omega \rightarrow X$

• $X$ - set of values.
• $\Omega$ - set of random seeds.
• Random seed generator.
Random variable $\alpha$ in $X$ in measure theory

$\alpha : \Omega \rightarrow X$

- $X$ - set of values.
- $\Omega$ - set of random seeds.
- Random seed generator.
Random variable $\alpha$ in $X$ in measure theory

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

- $X$ - set of values.
- $\Omega$ - set of random seeds.
- Random seed generator.

1. $\Sigma \subseteq 2^\Omega$, $\Theta \subseteq 2^X$
Random variable $\alpha$ in $X$

in measure theory

$\alpha : \Omega \rightarrow X$

- $X$ - set of values.
- $\Omega$ - set of random seeds.
- Random seed generator.

1. $\Sigma \subseteq 2^{\Omega}$, $\Theta \subseteq 2^X$
2. $\mu : \Sigma \rightarrow [0, 1]$
Random variable $\alpha$ 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.
- $\Omega$ - set of random seeds.
- Random seed generator.

1. $\Sigma \subseteq 2^\Omega$, $\Theta \subseteq 2^X$
2. $\mu : \Sigma \rightarrow [0,1]$
Random variable $\alpha$ in $X$

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

- $X$ - set of values.
- $\Omega$ - set of random seeds.
- Random seed generator.
Random variable $\alpha$ in $X$

in quasi-Borel spaces

$\alpha : \Omega \rightarrow X$

• $X$ - set of values.
• $\Omega$ - set of random seeds.
• Random seed generator.
Random variable $\alpha$ in $X$

in quasi-Borel spaces

$$\alpha : [0, 1] \to X$$

- $X$ - set of values.
- $[0, 1]$ - set of random seeds.
- Random seed generator:
  1. $[0, 1]$ random source
  2. Borel subsets $\mathcal{B} \subseteq 2^{[0, 1]}$
Random variable $\alpha$ in $X$

in quasi-Borel spaces

$\alpha : [0,1] \rightarrow X$

- $X$ - set of values.
- $[0,1]$ - set of random seeds.
- Random seed generator

1. $[0,1]$ random source
2. Borel subsets $\mathcal{B} \subseteq 2^{[0,1]}$
3. Uniform$[0,1]$
Random variable $\alpha$ in $X$

in quasi-Borel spaces

$\alpha : [0,1] \rightarrow X$

- $X$ - set of values.
- $[0,1]$ - set of random seeds.
- Random seed generator

1. $[0,1]$ random source
2. Borel subsets $\mathcal{B} \subseteq 2^{[0,1]}$
3. Uniform $[0,1]$
4. $M \subseteq [([0,1] \rightarrow X]$
Random variable $\alpha$ in $X$

in quasi-Borel spaces

$\alpha : [0,1] \rightarrow X$ is a random variable if $\alpha \in M$

- $X$ - set of values.
- $[0,1]$ - set of random seeds.
- Random seed generator

1. $[0,1]$ random source
2. Borel subsets $\mathcal{B} \subseteq 2^{[0,1]}$
3. Uniform$[0,1]$
4. $M \subseteq ([0,1] \rightarrow X]$
• Measure theory:
  • Measurable space \((X, \Theta \subseteq 2^X)\).
  • Random variable is an induced concept.

• Quasi-Borel space:
  • Quasi-Borel space \((X, M \subseteq \{[0,1] \to X\})\).
  • \(M\) is the set of random variables.
[Theorem] The category of measurable spaces is not cartesian closed.

[Theorem] The category of quasi-Borel spaces is cartesian closed.

Intuitively, cartesian closure means good support for higher-order functions.

Compositional inference algo. by Scibior et al. are justified using quasi-Borel spaces [POPL18].
Choices

1. Unnormalised or normalized posterior?
2. Marginalised or un-marginalised?
3. Focus on measurable sets or random vars?
Choices

1. **Unnormalised** or normalized posterior?

2. **Marginalised** or un-marginalised?

3. Focus on measurable sets or **random vars**?
References


2. A convenient category for higher-order probability theory. Heunen et al. LICS’17.